Autor: **Nicola Strappazzon**
Web: http://nicola.strappazzon.me
Revisión: 17/11/11

Lección 8

Manipulando Datos y Transacciones

Este material se encuentra basado en el curso de Fundamentos a SQL de ORACLE, el cual es adaptado para el producto PostgreSQL, todos los ejemplos, códigos fuentes y la Base de Datos HR es propiedad de ORACLE.

Objetivos

Al completar esta lección usted podrá entender los siguientes puntos:

- Describir cada uno de los lenguajes de manipulación de datos (DML, Data Manipulation Lenguage).
- Introducir nuevos registros dentro de una tabla.
- Actualizar registros dentro de una tabla.
- Eliminar registros dentro de una tabla.
- Control de transacciones.

Lenguaje de Manipulación de Datos

- Las sentencias DML son ejecutadas cuando:
  - Introducir nuevos registros dentro de una tabla.
  - Actualizar registros dentro de una tabla.
  - Eliminar registros dentro de una tabla.
- El control de transacciones consiste en una colección de sentencias DML que permiten manejar el flujo de los datos cuando son manipulados.

Agregando un nuevo registro

DEPARTMENTS

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700

. . .

(27 rows)

Nuevo Registro

70	Public Relations	204	2700

DEPARTMENTS

department_id	department_name	manager_id	location_id
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700

. . .

(27 rows)

Usando la sentencia INSERT

- Agrega un nuevo registro dentro de una tabla utilizando la sentencia INSERT.
- Con esta sentencia, solo podrá insertar un registro a la vez.

```
INSERT INTO table [ ( column [, ...] ) ]
VALUES      ( expression [, ...]);
```

Consideraciones al insertar

- Insertar nuevos registros que contengan datos por cada columna.
- La lista de valores debe estar ordenada por el orden de las columnas de la tabla.
- Es opcional listar las columnas en la sentencia **`INSERT`**.
- Encerrar dentro de comillas simples los caracteres y las fechas.
- En caso de que se omite un valor o se agrega la palabra **`null`**, se agregara por defecto un valor que se encuentra especificado en la tabla.

Insertando un nuevo registro

- No se puede definir un Alias a las tablas.
- Puede deshacer los cambios de esta sentencia mediante el uso de transacciones.

```
INSERT INTO departments (department_id, department_name,
                         manager_id,    location_id)
VALUES       (70, 'Public Relations', 100, 1700);
INSERT 0 1
```

Insertando un registro con valores NULL

- Método implícito: Omite el nombre de la columna en la lista.

```
INSERT INTO departments (department_id, department_name)
VALUES      (70, 'Public Relations');
INSERT 0 1
```

- Método explícito: Se indica la palabra NULL en la lista de valores.

```
INSERT INTO departments
VALUES      (70, 'Public Relations', NULL, NULL);
INSERT 0 1
```

Insertando valores especiales

La función `current_date` obtiene la fecha del sistema.

```
INSERT INTO employees (employee_id, first_name, last_name,
                       email, phone_number, hire_date,
                       job_id, salary, commission_pct,
                       manager_id, department_id)
VALUES                (113, 'Louis', 'Popp', 'LPOPP',
                       '515.123.4567', current_date,
                       'AC_ACCOUNT', 6900, NULL, 205, 100);
INSERT 0 1
```

Insertando valores en específicos

Agregando un nuevo empleado.

```
INSERT INTO employees
VALUES      (113,
             'Louis',
             'Popp',
             'LPOPP',
             '515.123.4567',
             TO_DATE('FEB 3, 1999', 'MON, MM YYYY'),
             'AC_ACCOUNT',
             6900,
             NULL,
             205,
             100);
INSERT 0 1
```

Copiando registros desde otra tabla

- Escribir la sentencia **INSERT** con una subconsulta.
- No requiere el uso de la cláusula **VALUES**.
- Debe coincidir el número de columnas de la tabla que se encuentra definida en la sentencia **INSERT** con las que retorna la subconsulta.

```
INSERT INTO sales_reps (id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

INSERT 0 4
```

Entendiendo la salida

Al ejecutar sentencias DML se retorna información sobre los registros.

```
INSERT INTO departments
VALUES      (70, 'Public Relations', NULL, NULL);
INSERT 0 1
```

```
INSERT oid count
```

- **`oid`**: Es el Object ID (oid) que hace referencia al objeto que es afectado, por defecto es 0 y solo se aplica en el sentencia **`INSERT`**.
- **`count`**: Hace referencia a la cantidad de registros que son afectados dependiendo de la sentencia utilizada.

Modificando valores

EMPLOYEES

```
 employee_id |  last_name  | hire_date  |   job_id   | salary | department_id
-------------+-------------+------------+------------+--------+---------------
         145 | Russell     | 1996-10-01 | SA_MAN     |  14000 |            80
         146 | Partners    | 1997-01-05 | SA_MAN     |  13500 |            80
. . .
         103 | Hunold      | 1990-01-03 | IT_PROG    |   9000 |            60
         104 | Ernst       | 1991-05-21 | IT_PROG    |   6000 |            60
         105 | Austin      | 1997-06-25 | IT_PROG    |   4800 |            60
         106 | Pataballa   | 1998-02-05 | IT_PROG    |   4800 |            60
         107 | Lorentz     | 1999-02-07 | IT_PROG    |   4200 |            60

. . .
(107 rows)
```

Se actualizaron los siguientes registros:

```
 employee_id |  last_name  | hire_date  |   job_id   | salary | department_id
-------------+-------------+------------+------------+--------+---------------
         145 | Russell     | 1996-10-01 | SA_MAN     |  14000 |            80
         146 | Partners    | 1997-01-05 | SA_MAN     |  13500 |            80
. . .
         103 | Hunold      | 1990-01-03 | IT_PROG    |   9000 |            30
         104 | Ernst       | 1991-05-21 | IT_PROG    |   6000 |            30
         105 | Austin      | 1997-06-25 | IT_PROG    |   4800 |            30
         106 | Pataballa   | 1998-02-05 | IT_PROG    |   4800 |            30
         107 | Lorentz     | 1999-02-07 | IT_PROG    |   4200 |            30

. . .
(107 rows)
```

Usando la sentencia `UPDATE`

- Modifica un registro existente utilizando la sentencia **`UPDATE`**.
- Puede modificar mas de un registro al mismo tiempo.
- Puede especificar cuales son las columnas que serán modificadas.
- Puede deshacer los cambios de esta sentencia mediante el uso de transacciones. No se puede definir un Alias a las tablas.

```
UPDATE   table
   SET   column = expression [,...]
[WHERE   condition];
```

Modificando un registro

- Modificando uno o varios registros mediante una condición establecida en la cláusula **WHERE**.

```
UPDATE employees
   SET department_id = 70
WHERE  employee_id   = 113;
UPDATE 1
```

- Todos los registros son modificados en una tabla si se omite el uso de la cláusula **WHERE**.

```
UPDATE employees
   SET department_id = 110;
UPDATE 107
```

Modificando con subconsultas

Modificando los datos del `job_id` y el `salary` del empleado cuyo código es 114, obteniendo los datos a modificar del empleado cuyo código es 205.

```
UPDATE employees
   SET job_id = (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 205),
       salary = (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 205)
WHERE  employee_id = 114;
UPDATE 1
```

Modificando con subconsultas

Use una subconsultas para determinar una serie de valores que permitan modificar el registro dentro de una tabla mediante la la sentencia `UPDATE`.

```
UPDATE copy_employees
   SET job_id = (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 205),
       salary = (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 205)
WHERE  employee_id = 114;
UPDATE 1
```

Eliminando un registro

EMPLOYEES

```
 department_id |   department_name    | manager_id | location_id
---------------+----------------------+------------+-------------
            10 | Administration       |        200 |        1700
            20 | Marketing            |        201 |        1800
            30 | Purchasing           |        114 |        1700
            40 | Human Resources      |        203 |        2400
            50 | Shipping             |        121 |        1500
            60 | IT                   |        103 |        1400
            70 | Public Relations     |        204 |        2700
            80 | Sales                |        145 |        2500
            90 | Executive            |        100 |        1700
           100 | Finance              |        108 |        1700
. . .
(107 rows)
```

Se elimino un registro de la tabla:

```
 department_id |   department_name    | manager_id | location_id
---------------+----------------------+------------+-------------
            10 | Administration       |        200 |        1700
            20 | Marketing            |        201 |        1800
            30 | Purchasing           |        114 |        1700
            40 | Human Resources      |        203 |        2400
            50 | Shipping             |        121 |        1500
            60 | IT                   |        103 |        1400
            70 | Public Relations     |        204 |        2700
            80 | Sales                |        145 |        2500
            90 | Executive            |        100 |        1700
           110 | Accounting           |        205 |        1700
. . .
(106 rows)
```

Usando la sentencia `DELETE`

- Eliminar un registro existente utilizando la sentencia **`DELETE`**.
- Puede eliminar mas de un registro al mismo tiempo.
- No se puede definir un Alias a las tablas.
- Puede deshacer los cambios de esta sentencia mediante el uso de transacciones.

```
DELETE FROM table
WHERE  condition;
```

Eliminando un registro

- Especifique el registro a eliminar mediante un criterio establecido en la cláusula **WHERE**.

```
DELETE FROM departments
WHERE       department_name = 'Finance';
DELETE 1
```

- Eliminar todos los registros de una tabla sin usar la cláusula **WHERE**.

```
DELETE FROM copy_employees;
DELETE 68;
```

Eliminando con subconsultas

Use una subconsulta para determinar una serie de valores que permitan eliminar registros dentro de una tabla mediante la sentencia `DELETE`.

```
DELETE FROM employees
WHERE       department_id = (SELECT department_id
                             FROM   departments
                             WHERE  department_name
                                    ILIKE '%Public%');
DELETE 1
```

Sentencia TRUNCATE

- Elimina todos los registros dentro de una tabla.
- Es mas eficiente para las tablas con gran cantidad de registros que la cláusula **DELETE**.
- Solamente puede ser ejecutada por el usuario dueño (owner) de la tabla.
- Puede deshacer los cambios de esta sentencia mediante el uso de transacciones.

Sintaxis:

```
TRUNCATE [TABLE] name [,...];
```

Ejemplo:

```
TRUNCATE copy_employees;
```

Transacciones

Todo sistema gestor de Bases de Datos maneja los siguientes conceptos sobre las transacciones:

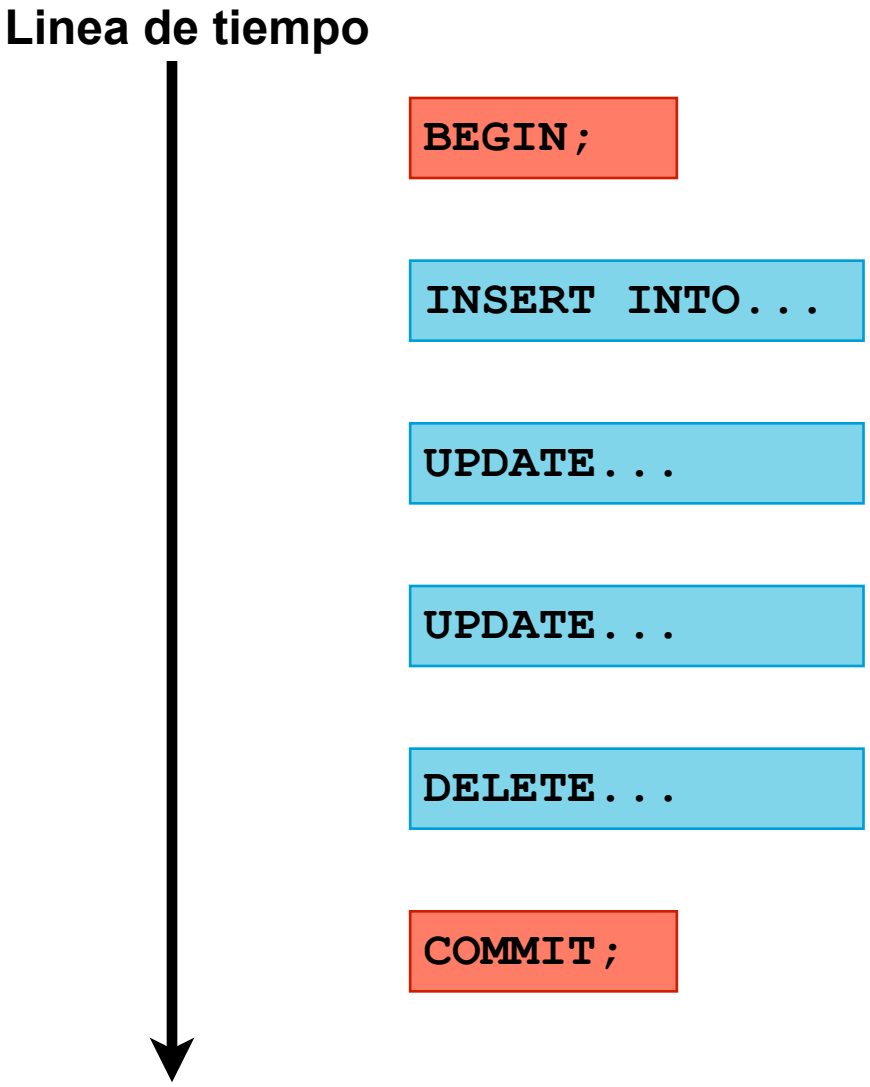
- Son un conjunto de acciones que altera el estado original de los datos y forman una sola unidad.
- Todo lenguaje que manipula los datos (DML) son administrador por las transacciones.
- Las transacciones pueden interrumpir un conjunto de acciones o hacerlas permanentes.
- Mantiene la integridad de los datos cuando alguna acción falla.

Comandos utilizados

Los siguientes comandos son utilizados para tener control de las transacciones:

- BEGIN
- COMMIT
- ROLLBACK
- SAVEPOINT
- ROLLBACK TO
- RELEASE SAVEPOINT

Usando el comando BEGIN y COMMIT



- Se inicia un conjunto de transacciones mediante el comando **BEGIN;**
- Luego se procede a utilizar cualquier comando de tipo DML (**INSERT**, **UPDATE** y/o **DELETE**) para manipular los datos.
- Por ultimo, se finaliza las transacciones haciendo los cambios permanentes con el comando **COMMIT;**.

Ejemplo del comando BEGIN y COMMIT

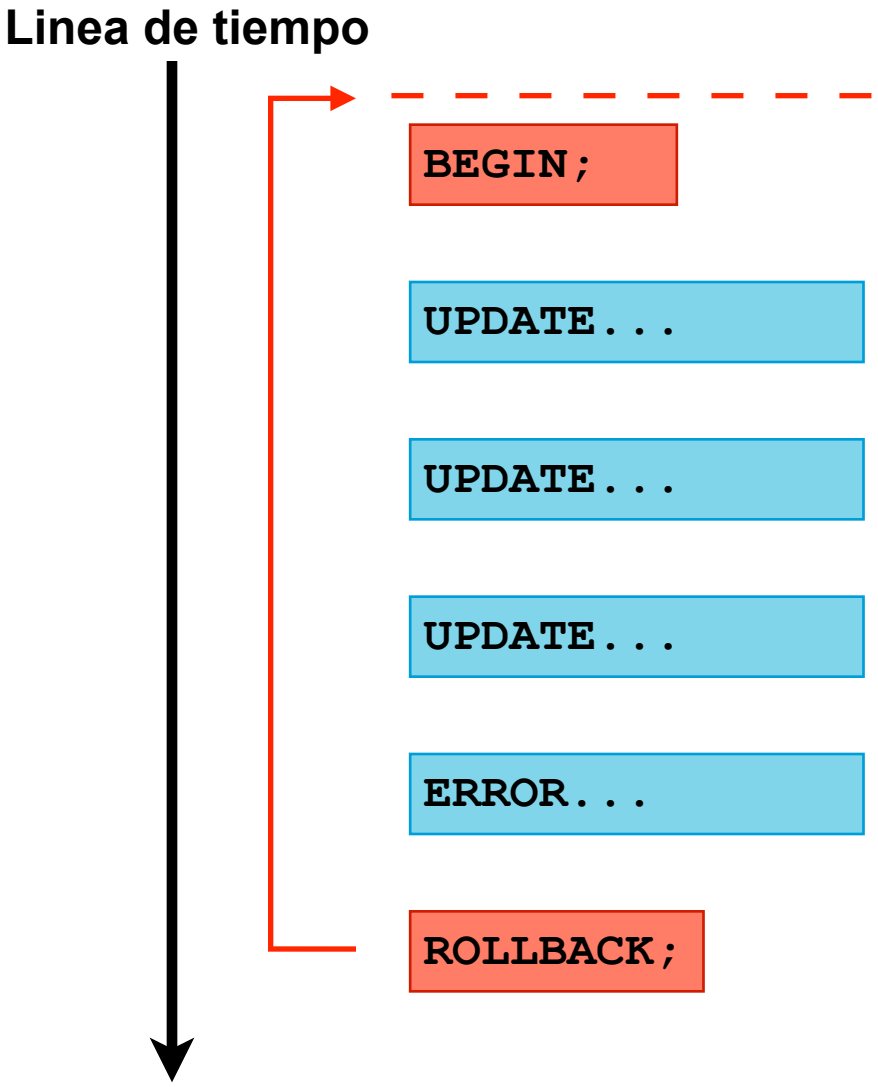
```
HR=# BEGIN;
BEGIN
HR=# SELECT * FROM departments WHERE department_id = 270;
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        1700
(1 row)

HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 270;
UPDATE 1
HR=# SELECT * FROM departments WHERE department_id = 270;
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
(1 row)

HR=# COMMIT;
COMMIT
HR=# SELECT * FROM departments WHERE department_id = 270;
department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
(1 row)

HR=#
```

Usando el comando ROLLBACK



La ultima actualización que se realizo a generado un error inesperado o un resultado no deseado, para deshacer todos los cambios se utiliza el comando **ROLLBACK**.

Ejemplo del comando ROLLBACK

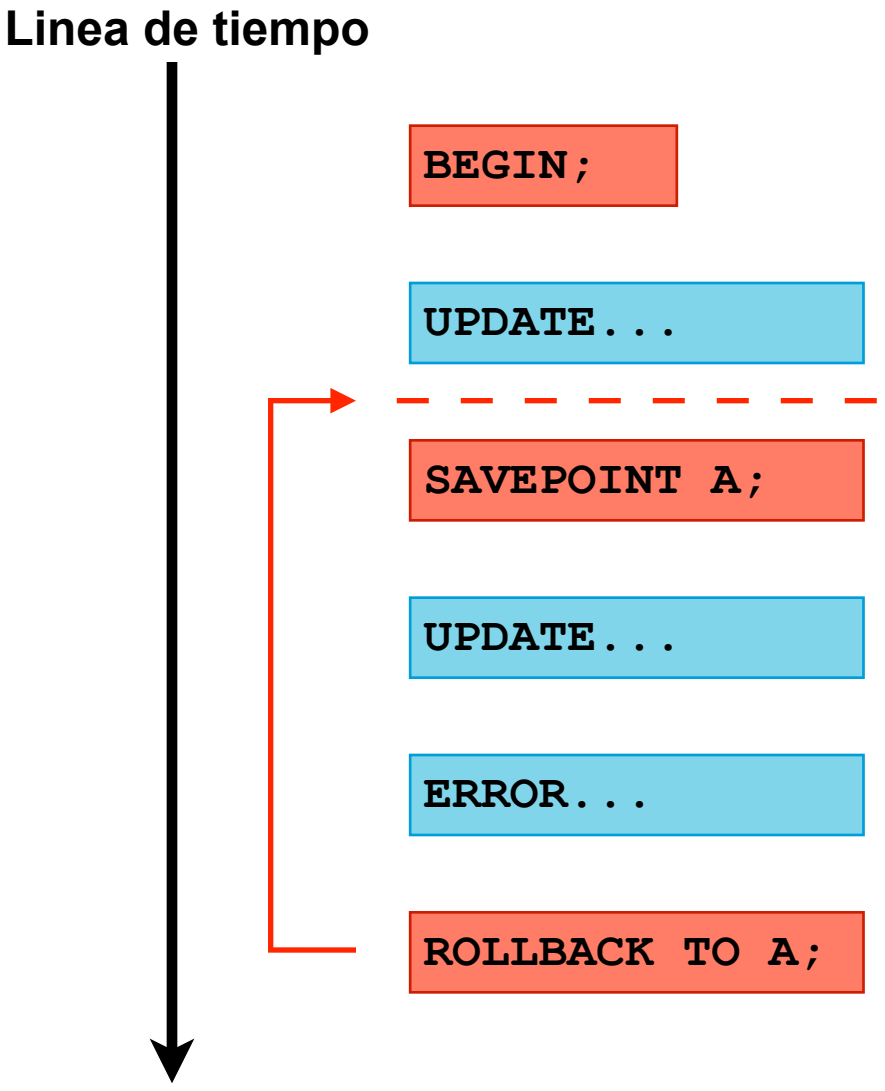
```
HR=# BEGIN;
BEGIN
HR=# SELECT * FROM departments WHERE department_id = 270;
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        1700
(1 row)

HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 270;
UPDATE 1
HR=# SELECT * FROM departments WHERE department_id = 270;
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
(1 row)

HR=# ROLLBACK;
ROLLBACK
HR=# SELECT * FROM departments WHERE department_id = 270;
department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        1700
(1 row)

HR=# COMMIT;
```

Usando el comando SAVEPOINT



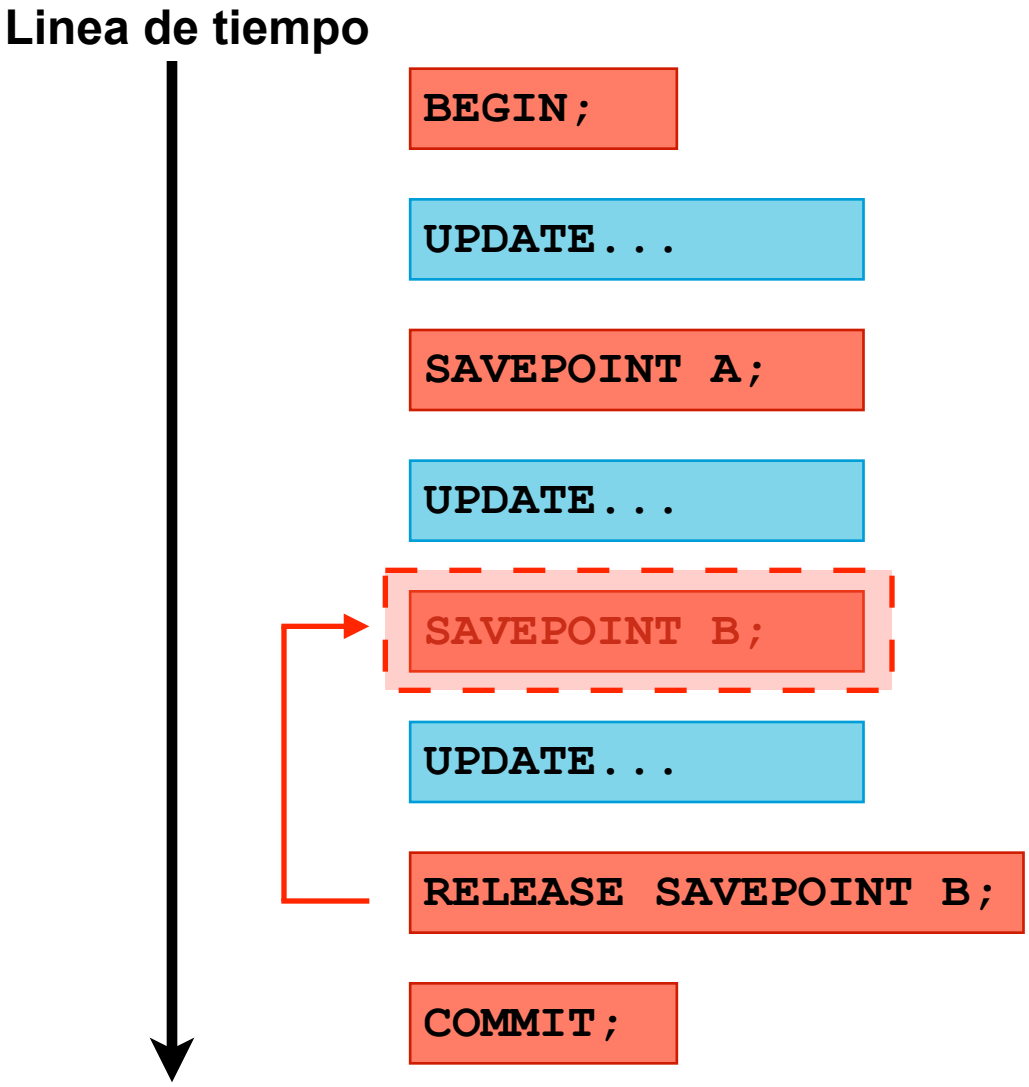
La ultima actualización que se realizo a generado un error inesperado o un resultado no deseado, para deshacer todos los cambios hasta el punto de control A se utiliza el comando **ROLLBACK TO**.

Ejemplo del comando SAVEPOINT

```
HR=# BEGIN;
BEGIN
HR=# SELECT * FROM departments WHERE department_id = 270;
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        1700
(1 row)

HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 270;
UPDATE 1
HR=# SAVEPOINT A;
SAVEPOINT
HR=# SELECT * FROM departments WHERE department_id = 270;
department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
(1 row)
HR=# UPDATE departments SET location_id = 1000 WHERE department_id = 270;
UPDATE 1
HR=# ROLLBACK TO A;
ROLLBACK
HR=# COMMIT;
COMMIT
```

Usando el comando RELEASE SAVEPOINT



Se determino que el punto de control B ya no es utilizado, todos los datos afectados de dicho punto pasaran a ser del punto de control A, para realizar este cambio se utiliza el comando **RELEASE SAVEPOINT**.

Ejemplo del comando SAVEPOINT

```
HR=# BEGIN;
BEGIN
HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 270;
UPDATE 1
HR=# SAVEPOINT A;
SAVEPOINT
HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 260;
UPDATE 1
HR=# SAVEPOINT B;
SAVEPOINT
HR=# UPDATE departments SET location_id = 2500 WHERE department_id = 250;
UPDATE 1
HR=# SELECT * FROM departments WHERE department_id IN (270, 260, 250);
 department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
           260 | Recruiting      |            |        2500
           250 | Retail Sales    |            |        2500
(3 rows)
HR=# RELEASE SAVEPOINT B;
RELEASE
HR=# ROLLBACK TO A;
ROLLBACK
department_id | department_name | manager_id | location_id
---------------+-----------------+------------+-------------
           270 | Payroll         |            |        2500
           260 | Recruiting      |            |        1700
           250 | Retail Sales    |            |        1700
(3 rows)
HR=# COMMIT;
COMMIT
```

Resumen

En esta lección, usted debió entender como:

- Utilizar los comandos DML que permiten insertar o alterar los datos.
- Eliminar el contenido de una tabla.
- Manejo de las transacciones.