

TAREA DE LABORATORIO

CURSO : *Base de Datos II*

No. : *Laboratorio 06*

TEMA : *Transacciones Relacionales*

Creación de Tablas

```
-- =====  
-- CREACIÓN DE TABLAS DEL ESQUEMA HR  
-- =====
```

```
CREATE TABLE regions (  
    region_id NUMBER PRIMARY KEY,  
    region_name VARCHAR2(50)  
);
```

```
CREATE TABLE countries (  
    country_id CHAR(2) PRIMARY KEY,  
    country_name VARCHAR2(50),  
    region_id NUMBER REFERENCES regions(region_id)  
);
```

```
CREATE TABLE locations (  
    location_id NUMBER PRIMARY KEY,  
    street_address VARCHAR2(100),  
    postal_code VARCHAR2(12),  
    city VARCHAR2(30),  
    state_province VARCHAR2(25),  
    country_id CHAR(2) REFERENCES countries(country_id)  
);
```

```
CREATE TABLE departments (  
    department_id NUMBER PRIMARY KEY,  
    department_name VARCHAR2(30),  
    manager_id NUMBER,  
    location_id NUMBER REFERENCES locations(location_id)  
);
```

```
CREATE TABLE jobs (  
    job_id VARCHAR2(10) PRIMARY KEY,  
    job_title VARCHAR2(35),  
    min_salary NUMBER(8,2),  
    max_salary NUMBER(8,2)  
);
```

```
CREATE TABLE employees (  
    employee_id NUMBER PRIMARY KEY,  
    first_name VARCHAR2(20),  
    last_name VARCHAR2(25) NOT NULL,  
    email VARCHAR2(25) NOT NULL,  
    phone_number VARCHAR2(20),  
    hire_date DATE NOT NULL,  
    job_id VARCHAR2(10) REFERENCES jobs(job_id),  
    salary NUMBER(8,2),  
    commission_pct NUMBER(2,2),
```

```

manager_id NUMBER,
department_id NUMBER REFERENCES departments(department_id)
);

CREATE TABLE job_history (
employee_id NUMBER REFERENCES employees(employee_id),
start_date DATE NOT NULL,
end_date DATE NOT NULL,
job_id VARCHAR2(10) REFERENCES jobs(job_id),
department_id NUMBER REFERENCES departments(department_id),
CONSTRAINT job_history_pk PRIMARY KEY (employee_id, start_date)
);

```

Inserción de Datos en las Tablas

```

-- =====
-- INSERTS DE DATOS
-- =====

-- REGIONS
INSERT INTO regions VALUES (1, 'Europe');
INSERT INTO regions VALUES (2, 'Americas');
INSERT INTO regions VALUES (3, 'Asia');
INSERT INTO regions VALUES (4, 'Middle East and Africa');

-- COUNTRIES
INSERT INTO countries VALUES ('US', 'United States', 2);
INSERT INTO countries VALUES ('UK', 'United Kingdom', 1);
INSERT INTO countries VALUES ('IN', 'India', 3);
INSERT INTO countries VALUES ('BR', 'Brazil', 2);
INSERT INTO countries VALUES ('EG', 'Egypt', 4);

-- LOCATIONS
INSERT INTO locations VALUES (1000, '2014 NE 45th Ave', '98121', 'Seattle', 'Washington', 'US');
INSERT INTO locations VALUES (1100, '1987 E Maple St', '02210', 'Boston', 'Massachusetts', 'US');
INSERT INTO locations VALUES (1200, '25 Main Street', 'WC2N', 'London', NULL, 'UK');
INSERT INTO locations VALUES (1300, 'Plot 10, IT Park', '560100', 'Bangalore', 'Karnataka', 'IN');
INSERT INTO locations VALUES (1400, 'Rua das Flores', '04567', 'Sao Paulo', 'SP', 'BR');
INSERT INTO locations VALUES (1500, 'Nile Tower', '11511', 'Cairo', NULL, 'EG');

-- DEPARTMENTS
INSERT INTO departments VALUES (50, 'Sales', NULL, 1400);
INSERT INTO departments VALUES (60, 'IT', NULL, 1000);
INSERT INTO departments VALUES (70, 'Marketing', NULL, 1200);
INSERT INTO departments VALUES (80, 'HR', NULL, 1100);
INSERT INTO departments VALUES (90, 'Finance', NULL, 1300);
INSERT INTO departments VALUES (100, 'Operations', NULL, 1500);
INSERT INTO departments VALUES (110, 'Logistics', NULL, 1400);

-- JOBS
INSERT INTO jobs VALUES ('AD_PRES', 'President', 20000, 40000);
INSERT INTO jobs VALUES ('AD_VP', 'Vice President', 15000, 30000);
INSERT INTO jobs VALUES ('IT_PROG', 'Programmer', 4000, 9000);
INSERT INTO jobs VALUES ('MK_MAN', 'Marketing Manager', 7000, 15000);
INSERT INTO jobs VALUES ('HR_REP', 'HR Representative', 4500, 9000);
INSERT INTO jobs VALUES ('FI_MGR', 'Finance Manager', 9000, 15000);

```

```

INSERT INTO jobs VALUES ('SA_REP', 'Sales Representative', 4000, 11000);
INSERT INTO jobs VALUES ('OP_MGR', 'Operations Manager', 8000, 16000);
INSERT INTO jobs VALUES ('LOG_ASST', 'Logistics Assistant', 3000, 7000);

```

-- EMPLOYEES

```

INSERT INTO employees VALUES (101, 'John', 'Smith', 'JSMITH', '515.123.4567', DATE
'2019-01-10', 'IT_PROG', 6000, NULL, NULL, 60);
INSERT INTO employees VALUES (102, 'Mary', 'Johnson', 'MJOHNSON', '515.234.5678', DATE
'2018-04-21', 'HR_REP', 4800, NULL, NULL, 80);
INSERT INTO employees VALUES (103, 'Robert', 'Brown', 'RBROWN', '515.345.6789', DATE
'2020-06-15', 'FI_MGR', 10000, NULL, NULL, 90);
INSERT INTO employees VALUES (104, 'Patricia', 'Taylor', 'PTAYLOR', '515.456.7890', DATE
'2021-08-01', 'SA_REP', 5000, NULL, NULL, 50);
INSERT INTO employees VALUES (105, 'James', 'Davis', 'JDAVIS', '515.567.8901', DATE
'2022-09-20', 'OP_MGR', 9000, NULL, NULL, 100);
INSERT INTO employees VALUES (106, 'Linda', 'Miller', 'LMILLER', '515.678.9012', DATE
'2017-11-05', 'LOG_ASST', 3500, NULL, NULL, 110);
INSERT INTO employees VALUES (107, 'David', 'Wilson', 'DWILSON', '515.789.0123', DATE
'2020-02-12', 'IT_PROG', 6200, NULL, NULL, 60);
INSERT INTO employees VALUES (108, 'Jennifer', 'Moore', 'JMOORE', '515.890.1234', DATE
'2019-07-07', 'SA_REP', 5200, NULL, NULL, 50);
INSERT INTO employees VALUES (109, 'William', 'Anderson', 'WANDER', '515.901.2345', DATE
'2021-10-11', 'HR_REP', 4600, NULL, NULL, 80);
INSERT INTO employees VALUES (110, 'Elizabeth', 'Thomas', 'ETHOMAS', '515.012.3456',
DATE '2022-01-15', 'FI_MGR', 9800, NULL, NULL, 90);

```

-- JOB_HISTORY (historial de cargos anteriores)

```

INSERT INTO job_history VALUES (101, DATE '2018-01-01', DATE '2018-12-31', 'IT_PROG',
60);
INSERT INTO job_history VALUES (102, DATE '2017-03-01', DATE '2018-03-31', 'HR_REP',
80);
INSERT INTO job_history VALUES (103, DATE '2019-01-01', DATE '2019-12-31', 'FI_MGR', 90);
INSERT INTO job_history VALUES (104, DATE '2020-01-01', DATE '2021-07-31', 'SA_REP', 50);
INSERT INTO job_history VALUES (105, DATE '2021-01-01', DATE '2022-08-31', 'OP_MGR',
100);
INSERT INTO job_history VALUES (106, DATE '2016-01-01', DATE '2017-10-31', 'LOG_ASST',
110);

```

COMMIT;

<

1. Ejercicio 1 - Control básico de transacciones

Cree un bloque anónimo PL/SQL que:

- Aumente en un 10% el salario de los empleados del departamento 90.
- Guarde un SAVEPOINT llamado punto1.
- Aumente en un 5% el salario de los empleados del departamento 60.
- Realice un ROLLBACK al SAVEPOINT punto1.
- Ejecute finalmente un COMMIT.

```
-- =====  
-- LABORATORIO DE TRANSACCIONES - EJERCICIO 1  
-- CONTROL BÁSICO DE TRANSACCIONES  
-- =====
```

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
  v_contador90 NUMBER;
```

```
  v_contador60 NUMBER;
```

```
BEGIN
```

```
  -- Aumento del 10% al salario de los empleados del departamento 90
```

```
  UPDATE employees
```

```
  SET salary = salary * 1.10
```

```
  WHERE department_id = 90;
```

```
  SELECT COUNT(*) INTO v_contador90 FROM employees WHERE department_id = 90;
```

```
  DBMS_OUTPUT.PUT_LINE('Aumentado salario en 10% para ' || v_contador90 || ' empleados del  
depto 90.');
```

```
  -- SAVEPOINT PUNTO1
```

```
  SAVEPOINT punto1;
```

```
  DBMS_OUTPUT.PUT_LINE('SAVEPOINT punto1 establecido.');
```

```
  -- Aumento del 5% al salario de los empleados del departamento 60
```

```
  UPDATE employees
```

```
  SET salary = salary * 1.05
```

```
  WHERE department_id = 60;
```

```
  SELECT COUNT(*) INTO v_contador60 FROM employees WHERE department_id = 60;
```

```
  DBMS_OUTPUT.PUT_LINE('Aumentado salario en 5% para ' || v_contador60 || ' empleados del  
depto 60.');
```

```
  -- ROLLBACK AL SAVEPOINT PUNTO1
```

```
  ROLLBACK TO punto1;
```

```
  DBMS_OUTPUT.PUT_LINE('Rollback al SAVEPOINT punto1 ejecutado. Se deshacen cambios  
del depto 60.');
```

```
-- CONFIRMAR TRANSACCIÓN FINAL
COMMIT;
DBMS_OUTPUT.PUT_LINE('Transacción confirmada (COMMIT ejecutado).');
END;
/
```

```
-- CONFIRMAR TRANSACCIÓN FINAL
COMMIT;
DBMS_OUTPUT.PUT_LINE('Transacción confirmada (COMMIT ejecutado).');
END;
/
```

Salida de Script x | Tarea terminada en 0.241 segundos

Aumentado salario en 10% para 2 empleados del depto 90.
 SAVEPOINT puntol establecido.
 Aumentado salario en 5% para 2 empleados del depto 60.
 Rollback al SAVEPOINT puntol ejecutado. Se deshacen cambios del depto 60.
 Transacción confirmada (COMMIT ejecutado).
 Procedimiento PL/SQL terminado correctamente.

Comprobando:

```
SELECT employee_id, first_name, salary, department_id
FROM employees
WHERE department_id IN (60, 90);
```

Salida de Script x | Resultado de la Consulta x

Todas las Filas Recuperadas: 4 en 0.033 segundos

	EMPLOYEE_ID	FIRST_NAME	SALARY	DEPARTMENT_ID
1	101	John	6000	60
2	103	Robert	11000	90
3	107	David	6200	60
4	110	Elizabeth	10780	90

Preguntas:

a. ¿Qué departamento mantuvo los cambios?

El departamento 90, porque los cambios realizados antes del SAVEPOINT fueron confirmados con COMMIT.

b. ¿Qué efecto tuvo el ROLLBACK parcial?

Revirtió únicamente los cambios realizados después del SAVEPOINT punto1, es decir, el aumento del 5% del departamento 60.

c. ¿Qué ocurriría si se ejecutara ROLLBACK sin especificar SAVEPOINT?

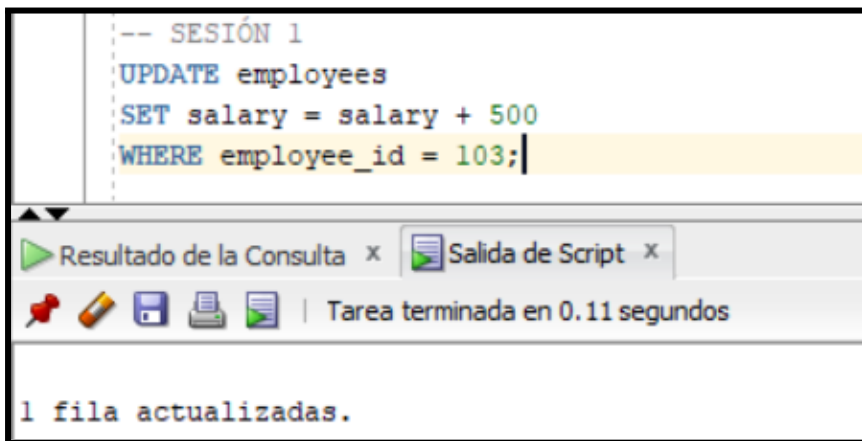
Se deshacerían todos los cambios realizados en la transacción, incluyendo los del departamento 90.

2. Ejercicio 2 - Bloqueos entre sesiones

En dos sesiones diferentes de Oracle:

- En la primera sesión, ejecute:
UPDATE employees
SET salary = salary + 500
WHERE employee_id = 103;
- Sin ejecutar COMMIT, en la segunda sesión, intente modificar el mismo registro.
- Observe el bloqueo y, desde la primera sesión, ejecute:
ROLLBACK;
- Analice el efecto sobre la segunda sesión.

En SESIÓN 1:



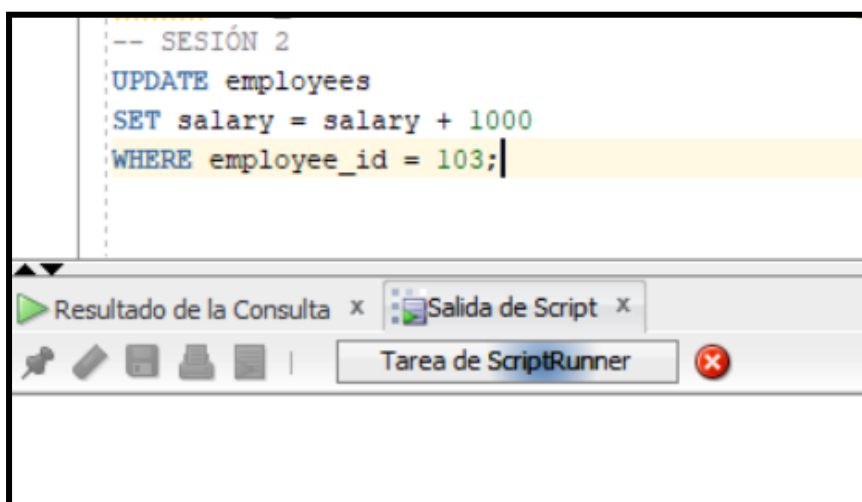
```
-- SESIÓN 1
UPDATE employees
SET salary = salary + 500
WHERE employee_id = 103;
```

Resultado de la Consulta x Salida de Script x

Tarea terminada en 0.11 segundos

1 fila actualizadas.

En SESIÓN 2:

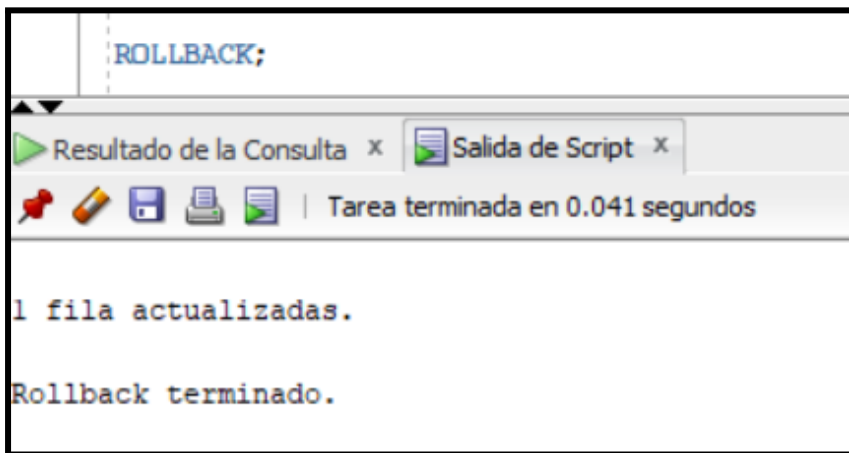


```
-- SESIÓN 2
UPDATE employees
SET salary = salary + 1000
WHERE employee_id = 103;
```

Resultado de la Consulta x Salida de Script x

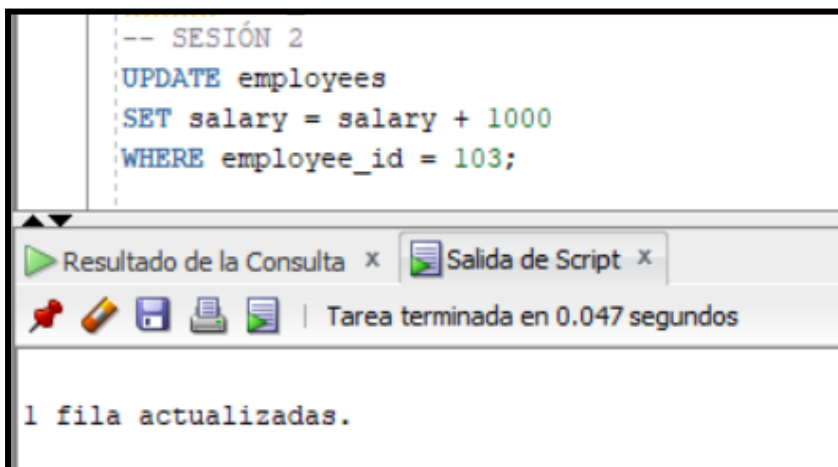
Tarea de ScriptRunner

En SESIÓN 1:



The screenshot shows a SQL Developer window with the command `ROLLBACK;` entered in the script editor. Below the editor, the status bar indicates "Tarea terminada en 0.041 segundos". The output pane shows the results: "1 fila actualizadas." and "Rollback terminado."

En SESIÓN 2:



The screenshot shows a SQL Developer window with the following SQL command: `-- SESIÓN 2
UPDATE employees
SET salary = salary + 1000
WHERE employee_id = 103;`. The status bar indicates "Tarea terminada en 0.047 segundos". The output pane shows the result: "1 fila actualizadas."

Preguntas:

a. ¿Por qué la segunda sesión quedó bloqueada?

Porque Oracle usa **bloqueo de fila (row-level lock)** para mantener la integridad de los datos. Una transacción abierta mantiene el registro ocupado hasta confirmarla o revertirla.

b. ¿Qué comando libera los bloqueos?

COMMIT o ROLLBACK. Ambos terminan la transacción activa y liberan los recursos bloqueados.

c. ¿Qué vistas del diccionario permiten verificar sesiones bloqueadas?

Se puede revisar los bloqueos activos usando:

`SELECT * FROM v$locked_object;`

3. Ejercicio 3 - Transacción controlada con bloque PL / SQL

Cree un bloque anónimo PL/SQL que realice una transferencia de empleado de un departamento a otro, registrando la transacción en `JOB_HISTORY`.

Pasos:

- **Actualice el department_id del empleado 104 al departamento 110.**
- **Inserte simultáneamente el registro correspondiente en JOB_HISTORY.**
- **Si ocurre un error (por ejemplo, departamento inexistente), haga un ROLLBACK y muestre un mensaje con DBMS_OUTPUT.**

```
-- =====
-- LABORATORIO DE TRANSACCIONES - EJERCICIO 3
-- =====

SET SERVEROUTPUT ON;

DECLARE
    v_emp_id    NUMBER := 104;
    v_new_dept_id NUMBER := 110;
    v_old_dept_id NUMBER;
    v_job_id     employees.job_id%TYPE;
    v_start_date DATE;
BEGIN
    -- Obtener datos actuales del empleado
    SELECT department_id, job_id, hire_date
    INTO v_old_dept_id, v_job_id, v_start_date
    FROM employees
    WHERE employee_id = v_emp_id;

    -- Actualizar el departamento del empleado
    UPDATE employees
    SET department_id = v_new_dept_id
    WHERE employee_id = v_emp_id;

    -- Insertar registro en JOB_HISTORY
    INSERT INTO job_history (employee_id, start_date, end_date, job_id, department_id)
    VALUES (v_emp_id, v_start_date, SYSDATE, v_job_id, v_old_dept_id);

    -- Confirmar los cambios
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Transferencia realizada correctamente: Empleado ' || v_emp_id ||
        ' del Dpto ' || v_old_dept_id || ' al Dpto ' || v_new_dept_id);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Error: El empleado no existe. ');
        ROLLBACK;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
        ROLLBACK;
END;
/
```



```
UPDATE employees
SET department_id = v_new_dept_id
WHERE employee_id = v_emp_id;

-- Insertar registro en JOB_HISTORY
INSERT INTO job_history (employee_id, start_date, end_date, job_id, department_id)
VALUES (v_emp_id, v_start_date, SYSDATE, v_job_id, v_old_dept_id);

-- Confirmar los cambios
COMMIT;

DBMS_OUTPUT.PUT_LINE('Transferencia realizada correctamente: Empleado ' || v_emp_id ||
' del Dpto ' || v_old_dept_id || ' al Dpto ' || v_new_dept_id);

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Error: El empleado no existe.');
```

Resultado de la Consulta x Salida de Script x

Tarea terminada en 0.187 segundos

Transferencia realizada correctamente: Empleado 104 del Dpto 50 al Dpto 110

Procedimiento PL/SQL terminado correctamente.

Preguntas:

a. ¿Por qué se debe garantizar la atomicidad entre las dos operaciones?

Porque la actualización e inserción son parte de una misma transacción lógica: si una falla, la otra no debe realizarse. Así se evita que el empleado quede transferido sin registro en el historial, o viceversa.

b. ¿Qué pasaría si se produce un error antes del COMMIT?

Oracle ejecuta el bloque EXCEPTION y hace ROLLBACK, deshaciendo todos los cambios realizados en esa transacción.

c. ¿Cómo se asegura la integridad entre EMPLOYEES y JOB_HISTORY?

Mediante las claves foráneas (FOREIGN KEY) que referencian el employee_id y department_id.

Esto garantiza que no se registren en JOB_HISTORY empleados o departamentos inexistentes.

4. Ejercicio 4 - SAVEPOINT y reversión parcial

Diseñe un bloque anónimo PL/SQL que ejecute las siguientes operaciones en una sola transacción:

- Aumentar el salario en 8% para empleados del departamento 100 → SAVEPOINT A.
- Aumentar el salario en 5% para empleados del departamento 80 → SAVEPOINT B. Eliminar los empleados del departamento 50.
- Revierte los cambios hasta el SAVEPOINT B.
- Finalmente, confirma la transacción con COMMIT.

-- =====

-- LABORATORIO DE TRANSACCIONES - EJERCICIO 4

-- =====

SET SERVEROUTPUT ON;

BEGIN

DBMS_OUTPUT.PUT_LINE('--- Inicio de transacción ---');

-- Aumento del 8% en departamento 100

UPDATE employees

SET salary = salary * 1.08

WHERE department_id = 100;

DBMS_OUTPUT.PUT_LINE('Aumento 8% aplicado al departamento 100.');

SAVEPOINT A;

-- Aumento del 5% en departamento 80

UPDATE employees

SET salary = salary * 1.05

WHERE department_id = 80;

DBMS_OUTPUT.PUT_LINE('Aumento 5% aplicado al departamento 80.');

SAVEPOINT B;

-- Eliminación de empleados del departamento 50

DELETE FROM employees

WHERE department_id = 50;

DBMS_OUTPUT.PUT_LINE('Empleados del departamento 50 eliminados.');

-- Reversión parcial al SAVEPOINT B

ROLLBACK TO SAVEPOINT B;

DBMS_OUTPUT.PUT_LINE('Reversión realizada hasta SAVEPOINT B.');

-- Confirmar la transacción

COMMIT;

DBMS_OUTPUT.PUT_LINE('--- Transacción confirmada con COMMIT ---');

EXCEPTION

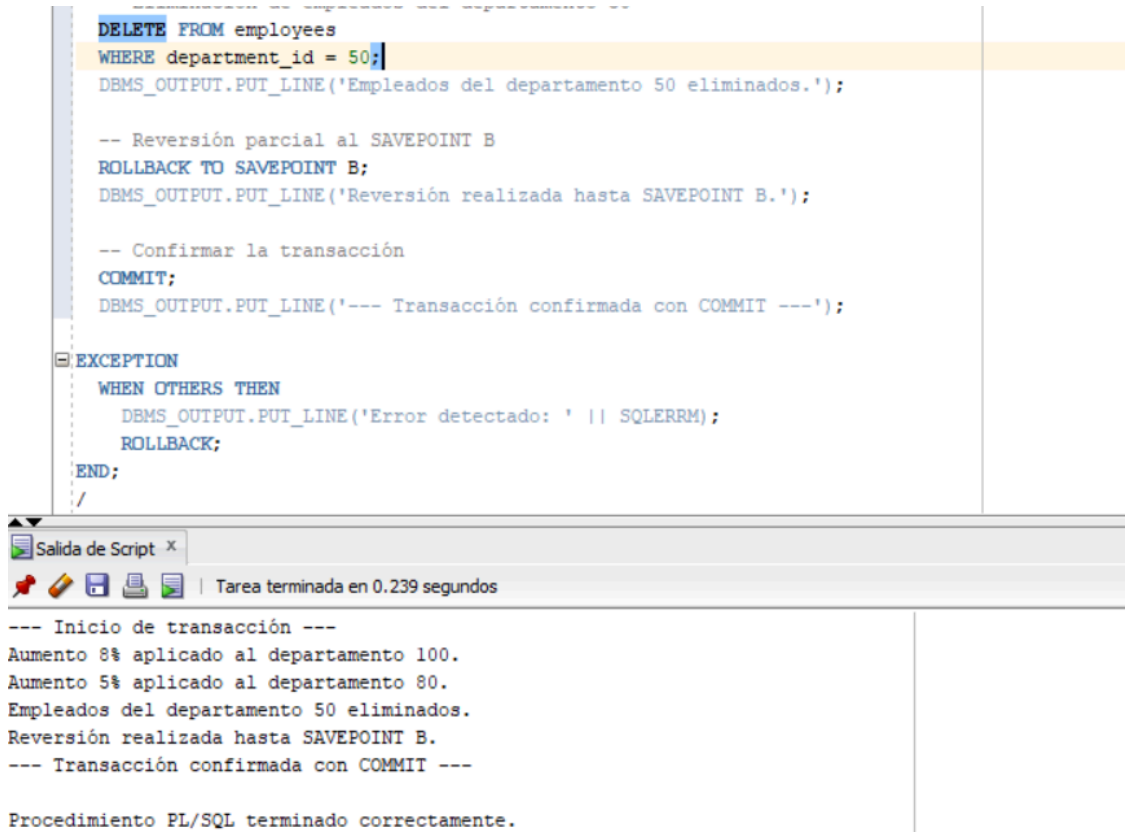
WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('Error detectado: ' || SQLERRM);

ROLLBACK;

END;

/



```
-----  
DELETE FROM employees  
WHERE department_id = 50;  
DBMS_OUTPUT.PUT_LINE('Empleados del departamento 50 eliminados.');
```

-- Reversión parcial al SAVEPOINT B
ROLLBACK TO SAVEPOINT B;
DBMS_OUTPUT.PUT_LINE('Reversión realizada hasta SAVEPOINT B.');

-- Confirmar la transacción
COMMIT;
DBMS_OUTPUT.PUT_LINE('--- Transacción confirmada con COMMIT ---');

EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error detectado: ' || SQLERRM);
ROLLBACK;
END;
/

Salida de Script x

Tarea terminada en 0.239 segundos

--- Inicio de transacción ---
Aumento 8% aplicado al departamento 100.
Aumento 5% aplicado al departamento 80.
Empleados del departamento 50 eliminados.
Reversión realizada hasta SAVEPOINT B.
--- Transacción confirmada con COMMIT ---

Procedimiento PL/SQL terminado correctamente.

Preguntas:

a. ¿Qué cambios quedan persistentes?

Solo los aumentos de salario de los empleados de los departamentos 100 y 80, ya que el ROLLBACK TO B revierte las operaciones posteriores al SAVEPOINT B (la eliminación de dept. 50).

b. ¿Qué sucede con las filas eliminadas?

Las filas de los empleados del departamento 50 no se eliminan definitivamente, porque el ROLLBACK TO SAVEPOINT B las restaura (revierte la eliminación).

c. ¿Cómo puedes verificar los cambios antes y después del COMMIT?

Se puede usar consultas SELECT dentro o fuera del bloque:

SELECT employee_id, salary, department_id FROM employees;

- Antes del COMMIT: los cambios se ven solo dentro de la sesión actual.
- Después del COMMIT: los cambios son visibles para todas las sesiones.