UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, Decana de América)

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

Curso: Base de Datos II

Sección: 02

Asunto: Informe del Trabajo Individual N°6 de BD2 sobre Transacciones Relacionales en Oracle Database

Alumno:

- Benites Meza, Marco Fabricio.

Docente:

- Mg. Jorge Luis Chávez Soto.

Lima, 2025

TRABAJO N°6 SOBRE TRANSACCIONES RELACIONALES

1. OBJETIVOS

El presente laboratorio tiene por objetivos:

- Aplicar los comandos COMMIT, ROLLBACK y SAVEPOINT para controlar la persistencia de los datos.
- Simular escenarios de bloqueos y concurrencia entre sesiones de Oracle.
- Garantizar la consistencia y atomicidad de las operaciones de actualización.
- Comprender el impacto de las transacciones en la base de datos HR durante procesos de inserción, actualización y eliminación.

2. DESCRIPCIÓN DE LAS TABLAS

- **Countries**

Name	Null?	Type
COUNTRY_ID	NOT NULL	CHAR(2)
COUNTRY_NAME		VARCHAR2(40)
REGION_ID		NUMBER

- **Departments**

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

- **Regions**

Name	Null?	Type
REGION_ID	NOT NULL	NUMBER
REGION_NAME		VARCHAR2(25)

- **Employees**

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

- **Jobs**

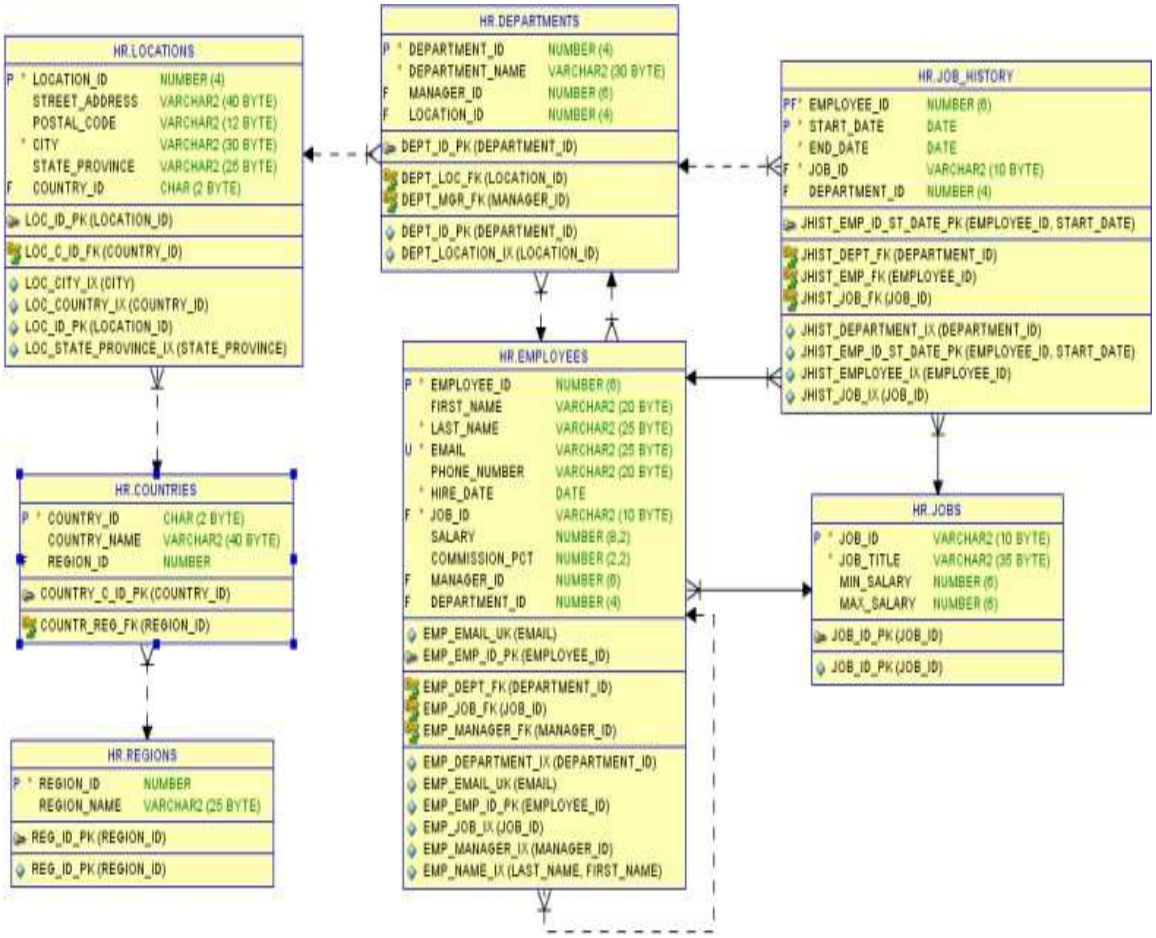
Name	Null?	Type
JOB_ID	NOT NULL	VARCHAR2(10)
JOB_TITLE	NOT NULL	VARCHAR2(35)
MIN_SALARY		NUMBER(6)
MAX_SALARY		NUMBER(6)

- **Job_History**

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

- **Locations**

Name	Null?	Type
LOCATION_ID	NOT NULL	NUMBER(4)
STREET_ADDRESS		VARCHAR2(40)
POSTAL_CODE		VARCHAR2(12)
CITY	NOT NULL	VARCHAR2(30)
STATE_PROVINCE		VARCHAR2(25)
COUNTRY_ID		CHAR(2)



3. EJERCICIOS PROPUESTOS

Ejercicio 01: 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 punto 1.
- Aumente en un 5% el salario de los empleados del departamento 60.
- Realice un ROLLBACK al SAVEPOINT punto 1.
- Ejecute finalmente un COMMIT.

Bloque PL/SQL (ejecutable):

```
SET SERVEROUTPUT ON SIZE 1000000;
BEGIN
  -- Aumentar 10% salario de empleados del departamento 90
  UPDATE hr.employees
  SET salary = salary * 1.10
  WHERE department_id = 90;

  SAVEPOINT punto1;

  -- Aumentar 5% salario de empleados del departamento 60
  UPDATE hr.employees
  SET salary = salary * 1.05
  WHERE department_id = 60;

  -- Revertir solo lo hecho después de punto1
  ROLLBACK TO punto1;

  -- Confirmar la transacción (persistir cambios anteriores a punto1)
  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Bloque finalizado: cambios confirmados.');
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
/
```

Preguntas:

a. ¿Qué departamento mantuvo los cambios?

El departamento 90 mantuvo el incremento del 10% porque ese cambio fue realizado antes del SAVEPOINT punto1. El aumento del 5% para departamento 60 fue deshecho por ROLLBACK TO punto1.

b. ¿Qué efecto tuvo el ROLLBACK parcial?

Efecto del ROLLBACK parcial: ROLLBACK TO punto1 revierte solo las operaciones efectuadas después del savepoint punto1; lo anterior al savepoint permanece.

c. ¿Qué ocurriría si se ejecutara ROLLBACK sin especificar SAVEPOINT?
Si se ejecutara ROLLBACK sin SAVEPOINT: Un ROLLBACK sin especificar savepoint desharía toda la transacción (todos los cambios desde el último COMMIT), dejando la sesión en el estado previo a la transacción.

Ejercicio 02: 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.

Pasos (dos sesiones):

```
-- Sesión A (ejecutar en una conexión)
SET TRANSACTION NAME 'sesion_a';
UPDATE hr.employees
SET salary = salary + 500
WHERE employee_id = 103;
-- NO HACER COMMIT ni ROLLBACK todavía; el registro queda bloqueado
por esta sesión.

-- Sesión B (en otra conexión distinta)
UPDATE hr.employees
SET salary = salary + 200
WHERE employee_id = 103;
-- Esta sentencia quedará bloqueada (esperará) hasta que la sesión A
haga COMMIT o ROLLBACK.

-- Volver a Sesión A para liberar bloqueo
-- En Sesión A (si decides no conservar el cambio) ROLLBACK;
-- O si deseas persistir: COMMIT;

-- Verificación (con privilegios):
SELECT sid, serial#, username, status, event FROM v$session WHERE
username IS NOT NULL;
SELECT l.session_id, s.username, l.locked_mode, o.object_name
  FROM v$locked_object l
  JOIN dba_objects o ON l.object_id = o.object_id
  JOIN v$session s ON l.session_id = s.sid;
SELECT * FROM v$lock WHERE block = 1; -- bloqueadores
```

Preguntas:

a. ¿Por qué la segunda sesión quedó bloqueada?

Porque la primera sesión obtuvo un exclusive lock (bloqueo de fila) sobre employee_id = 103 al ejecutar UPDATE sin COMMIT; Oracle impide cambios conflictivos hasta liberar el lock.

b. ¿Qué comando libera los bloqueos?
COMMIT (para persistir) o ROLLBACK (para deshacer). Ambos liberan los bloqueos.

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

Si se ejecutara ROLLBACK sin SAVEPOINT: Un ROLLBACK sin especificar savepoint desharía toda la transacción (todos los cambios desde el último COMMIT), dejando la sesión en el estado previo a la transacción.

Ejercicio 03: 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.

Bloque PL/SQL (ejecutable):

```
SET SERVEROUTPUT ON SIZE 1000000;
DECLARE
  v_emp_id     NUMBER := 104;
  v_new_dept   NUMBER := 110;
  v_old_dept   NUMBER;
  v_job_id     VARCHAR2(10);
  v_start_date DATE;
  v_end_date   DATE;
BEGIN
  -- Estado actual del empleado (y bloquear la fila)
  SELECT department_id, job_id INTO v_old_dept, v_job_id
  FROM hr.employees WHERE employee_id = v_emp_id FOR UPDATE;

  -- Registrar salida del puesto anterior en JOB_HISTORY (fechas de
ejemplo)
  v_start_date := SYSDATE - 30;
  v_end_date   := SYSDATE - 1;
  INSERT INTO hr.job_history(employee_id, start_date, end_date,
```

```
job_id, department_id)
  VALUES (v_emp_id, v_start_date, v_end_date, v_job_id, v_old_dept);

  -- Actualizar department_id del empleado
  UPDATE hr.employees SET department_id = v_new_dept WHERE
employee_id = v_emp_id;

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Transferencia completada y JOB_HISTORY
actualizado.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error: empleado no encontrado.');
  WHEN OTHERS THEN
    ROLLBACK;
    DBMS_OUTPUT.PUT_LINE('Error inesperado: ' || SQLERRM);
END;
```

Preguntas:

a. ¿Por qué se debe garantizar la atomicidad entre las 2 operaciones?

Porque ambas representan una sola acción lógica (traslado del empleado). Si una ocurre y la otra falla, los datos quedarían inconsistentes.

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

Se ejecuta ROLLBACK y se deshacen tanto el INSERT en JOB_HISTORY como el UPDATE en EMPLOYEES, manteniendo la integridad.

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

Mediante transacciones (COMMIT/ROLLBACK), restricciones de integridad referencial (FK) si existen y validaciones previas (p. ej., verificar existencia del departamento destino).

Ejercicio 04: 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.

Bloque PL/SQL (ejecutable):

```
SET SERVEROUTPUT ON SIZE 1000000;
BEGIN
  -- Aumentar 8% salario para departamento 100
  UPDATE hr.employees SET salary = salary * 1.08 WHERE department_id
= 100;
  SAVEPOINT A;

  -- Aumentar 5% salario para departamento 80
  UPDATE hr.employees SET salary = salary * 1.05 WHERE department_id
= 80;
  SAVEPOINT B;

  -- Eliminar empleados del departamento 50
  DELETE FROM hr.employees WHERE department_id = 50;

  -- Revertir hasta SAVEPOINT B (deshace el DELETE y posteriores a B)
  ROLLBACK TO B;

  -- Confirmar lo anterior a B
  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Operación finalizada: cambios hasta SAVEPOINT
B confirmados.');
EXCEPTION WHEN OTHERS THEN
  ROLLBACK;
  DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

Preguntas:

a. ¿Qué cambios quedan persistentes?

El aumento del 8% (dpto 100) y el aumento del 5% (dpto 80). El DELETE del dpto 50 fue revertido por el ROLLBACK TO B.

b. ¿Qué sucede con las filas eliminadas?

Se restauran al hacer ROLLBACK TO B, asumiendo que no hubo COMMIT intermedio que las haya persistido.

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

En la misma sesión puedes ver los efectos antes de confirmar; en otras sesiones recién después del COMMIT. Ejemplo: SELECT employee_id, first_name, salary, department_id FROM hr.employees WHERE department_id IN (100,80,50).