ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Laboratorio de:

Materia: Fundamentos de Bases de Datos

Práctica No.: LABORATORIO PRÁCTICO - TÓPICO

Tema: ESTRUCTURA DE UNA BD RELACIONAL - LENGUAJE DML – DATA MANIPULATION LANGAJE
SGBD: oracle Database

TABLA DE CONTENIDOS

1. Objetivos
2. Requisitos Previos
3. Conceptos Fundamentales DML
4. INSERT - Inserción de Datos
5. UPDATE - Actualización de Datos
6. DELETE - Eliminación de Datos
7. Transacciones (COMMIT, ROLLBACK, SAVEPOINT)
8. Ejercicios Prácticos
9. Casos de Prueba

OBJETIVOS

Objetivo General

Dominar los comandos del Lenguaje de Manipulación de Datos (DML) en Oracle para insertar, actualizar y eliminar registros, comprendiendo el manejo de transacciones y la integridad referencial.

Objetivos Específicos

1. **Insertar datos** usando INSERT con todas sus variantes
2. **Actualizar registros** de forma precisa y segura con UPDATE
3. **Eliminar datos** respetando integridad referencial con DELETE
4. **Gestionar transacciones** con COMMIT, ROLLBACK y SAVEPOINT

5. **Manejar errores** comunes en operaciones DML
6. **Insertar datos masivos** usando diferentes técnicas
7. **Aplicar buenas prácticas** en manipulación de datos
8. **Comprender el impacto** de restricciones en operaciones DML

REQUISITOS PREVIOS

Conocimientos Necesarios

- Laboratorio 4 DDL completado
- Conocimiento de tipos de datos Oracle
- Comprensión de restricciones de integridad
- Conocimiento de claves primarias y foráneas

Tablas Requeridas

Debe tener creadas las siguientes tablas del Laboratorio 4:

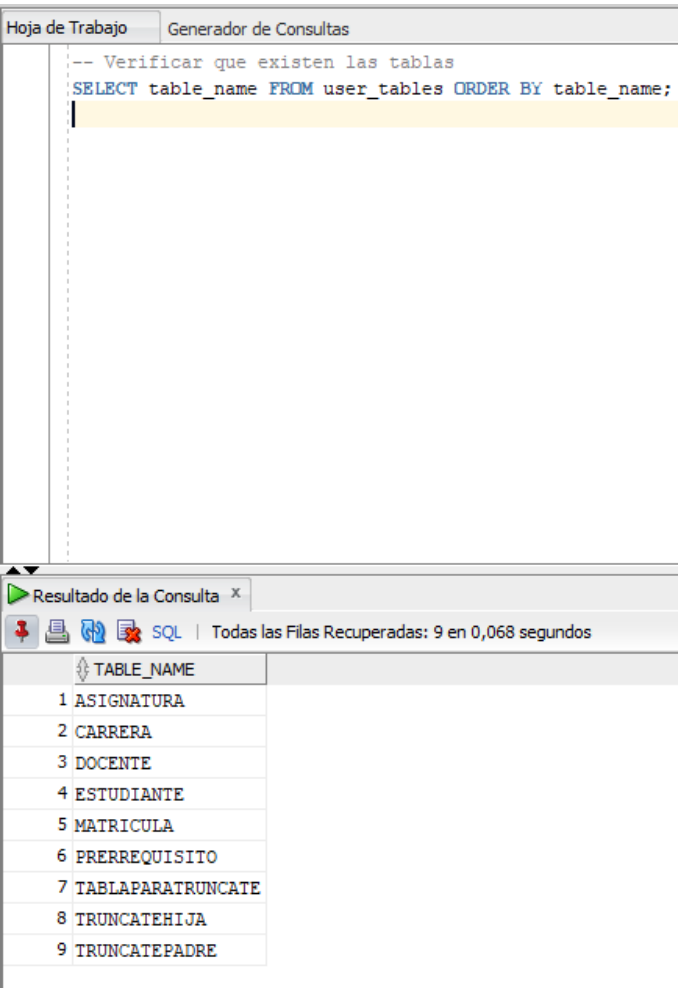
- ✓ CARRERA
- ✓ ESTUDIANTE
- ✓ ASIGNATURA
- ✓ DOCENTE
- ✓ PRERREQUISITO
- ✓ MATRICULA

Verificación del Esquema

```
-- Conectarse como el usuario del laboratorio anterior
CONN gestion_academica/EPN2024Secure;
```

```
SQL> CONNECT gestion_academica/EPN2024Secure
Conectado.
SQL>
```

```
-- Verificar que existen las tablas
SELECT table_name
FROM user_tables
ORDER BY table_name;
```




```
-- Verificar estructura de una tabla
DESC Carrera;
```

```
SQL>
SQL> DESC Carrera;
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CODIGO                                    NOT NULL VARCHAR2(10)
 NOMBRE                                    NOT NULL VARCHAR2(100)
 DURACION_SEMESTRES                        NOT NULL NUMBER(2)
 CREDITOS_TOTALES                          NOT NULL NUMBER(3)
 FACULTAD                                  NOT NULL VARCHAR2(100)
 FECHA_CREACION                            NOT NULL DATE

SQL>
```

```sql
-- Verificar restricciones
SELECT constraint_name, constraint_type, table_name
FROM user_constraints
WHERE table_name IN ('CARRERA', 'ESTUDIANTE', 'ASIGNATURA', 'DOCENTE', 'MATRICULA')
ORDER BY table_name, constraint_type;
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Hoja de Trabajo | Generador de Consultas

```
-- Verificar que existen las tablas
SELECT constraint_name, constraint_type, table_name FROM user_constraints
WHERE table_name IN ('CARRERA', 'ESTUDIANTE', 'ASIGNATURA', 'DOCENTE', 'MATRIC
ORDER BY table_name, constraint_type;
```

Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 72 en 0,341 segundos

	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
1	SYS_C007484	C	ASIGNATURA
2	SYS_C007485	C	ASIGNATURA
3	SYS_C007486	C	ASIGNATURA
4	SYS_C007487	C	ASIGNATURA
5	SYS_C007488	C	ASIGNATURA
6	SYS_C007489	C	ASIGNATURA
7	CHK_ASIGNATURA_CREDITOS	C	ASIGNATURA
8	SYS_C007490	C	ASIGNATURA
9	CHK_ASIGNATURA_NIVEL	C	ASIGNATURA
10	CHK_ASIGNATURA_HORAS	C	ASIGNATURA
11	CHK_ASIGNATURA_CREDITOS_HORAS	C	ASIGNATURA
12	PK_ASIGNATURA	P	ASIGNATURA
13	FK_ASIGNATURA_CARRERA	R	ASIGNATURA
14	UK_ASIGNATURA_NOMBRE_CARRERA	U	ASIGNATURA
15	CHK_CARRERA_CREDITOS	C	CARRERA
16	CHK_CARRERA_DURACION	C	CARRERA
17	SYS_C007462	C	CARRERA
18	SYS_C007457	C	CARRERA
19	SYS_C007461	C	CARRERA
20	SYS_C007460	C	CARRERA
21	SYS_C007459	C	CARRERA
22	SYS_C007458	C	CARRERA
23	PK_CARRERA	P	CARRERA
24	UK_CARRERA_NOMBRE	U	CARRERA
25	SYS_C007503	C	DOCENTE
26	SYS_C007497	C	DOCENTE
27	SYS_C007501	C	DOCENTE
28	SYS_C007500	C	DOCENTE
29	SYS_C007499	C	DOCENTE
30	SYS_C007498	C	DOCENTE
31	CHK_DOCENTE_CEDULA	C	DOCENTE

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

CONCEPTOS FUNDAMENTALES DML

¿Qué es DML?

DML (Data Manipulation Language) es el subconjunto de SQL usado para manipular datos en las tablas. Los comandos principales son:

COMANDO	FUNCIÓN	TRANSACCIONAL
INSERT	Insertar nuevas filas	Requiere COMMIT
UPDATE	Modificar filas existentes	Requiere COMMIT
DELETE	Eliminar filas	Requiere COMMIT
MERGE	Insertar o actualizar	Requiere COMMIT
SELECT	Consultar datos	No transaccional

Características de DML en Oracle

1. **Transaccionales**: Los cambios no son permanentes hasta hacer COMMIT
2. **Reversibles**: Se puede hacer ROLLBACK antes de COMMIT
3. **Generan UNDO**: Oracle guarda información para deshacer cambios
4. **Activan Triggers**: Los triggers DML se disparan automáticamente
5. **Validan Restricciones**: Se verifican todas las restricciones de integridad

DML vs DDL vs DCL

TIPO	COMANDOS	COMMIT	ROLLBACK	EJEMPLOS
DDL	CREATE, ALTER, DROP, TRUNCATE	Automático	No	CREATE TABLE
DML	INSERT, UPDATE, DELETE, MERGE	Manual	Sí	INSERT INTO
DCL	GRANT, REVOKE	Automático	No	GRANT SELECT
TCL	COMMIT, ROLLBACK, SAVEPOINT	N/A	N/A	COMMIT

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

INSERT - INSERCIÓN DE DATOS

Sintaxis Básica

```
-- Formato 1: INSERT especificando todas las columnas
Columnas no nulas
INSERT INTO nombre_tabla (columna1, columna2, columna3, ...)
VALUES (valor1, valor2, valor3, ...);

-- Formato 2: INSERT sin especificar columnas (todas en orden)
INSERT INTO nombre_tabla
VALUES (valor1, valor2, valor3, ...);

-- Formato 3: INSERT con subconsulta
INSERT INTO nombre_tabla (columna1, columna2, ...)
SELECT columna1, columna2, ...
FROM otra_tabla
WHERE condicion;

-- Formato 4: INSERT múltiple (INSERT ALL)
INSERT ALL
   INTO tabla1 VALUES (...)
   INTO tabla2 VALUES (...)
SELECT * FROM DUAL;
```

Ejemplo 1: INSERT Básico en CARRERA

```
-- ============================================================
-- EJEMPLO 1: Inserción básica especificando columnas
```

```
-- ================================================

-- Insertar una carrera especificando todas las columnas
INSERT INTO Carrera (
    codigo,
    nombre,
    duracion_semestres,
    creditos_totales,
    facultad,
    fecha_creacion
) VALUES (
    'ING-SIS',
    'Ingeniería en Sistemas Informáticos',
    10,
    180,
    'Facultad de Sistemas - ESFOT',
    SYSDATE
);
```

```
SQL> INSERT INTO Carrera ( codigo,
  2  nombre, duracion_semestres, creditos_totales, facultad, fecha_creacion
  3  ) VALUES ( 'ING-SIS',
  4  'Ingeniería en Sistemas Informáticos', 10,
  5  180,
  6  'Facultad de Sistemas - ESFOT', SYSDATE
  7  );

1 fila creada.
```

```
-- Verificar inserción
SELECT * FROM Carrera WHERE codigo = 'ING-SIS';
```

```
SELECT * FROM Carrera WHERE codigo = 'ING-SIS';
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 1 en 0,004 segundos

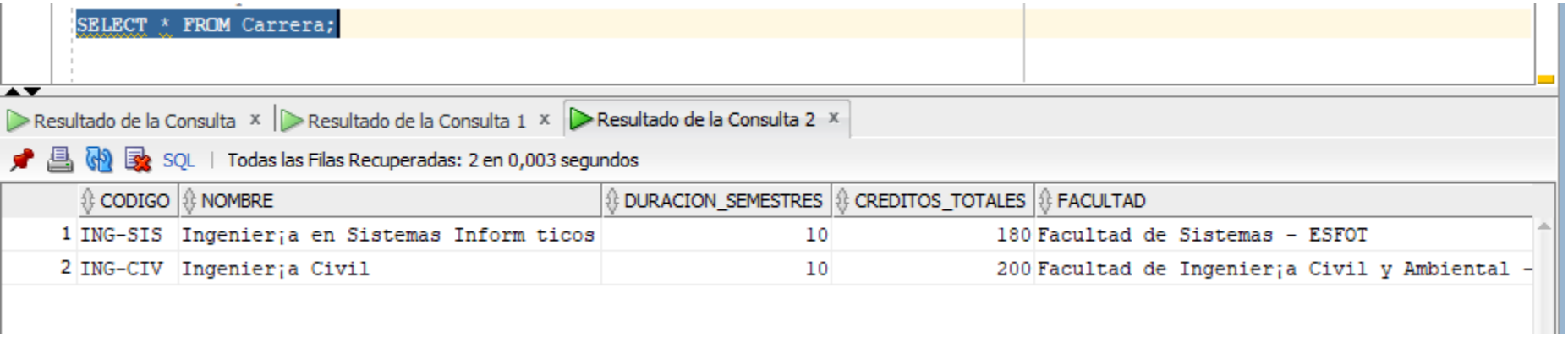
	CODIGO	NOMBRE	DURACION_SEMESTRES	CREDITOS_TOTALES	FACULTAD	FECHA_CREACI
1	ING-SIS	Ingenier¡a en Sistemas Inform ticos	10	180	Facultad de Sistemas - ESFOT	13/11/2025

-- NO hacer COMMIT todavía (para práctica)

Ejemplo 2: INSERT con Valores por Defecto

```
-- ============================================================
-- EJEMPLO 2: INSERT usando valores DEFAULT
-- ============================================================
-- Insertar sin especificar fecha_creacion (usa DEFAULT)
INSERT INTO Carrera (
    codigo,
    nombre,
    duracion_semestres,
    creditos_totales,
    facultad
) VALUES (
    'ING-CIV',
    'Ingeniería Civil',
    10,
    200,
    'Facultad de Ingeniería Civil y Ambiental - FIC'
);
```

```
SQL> INSERT INTO Carrera ( codigo,
  2  nombre, duracion_semestres, creditos_totales, facultad
  3  ) VALUES ( 'ING-CIV',
  4  'Ingeniería Civil', 10,
  5  200,
  6  'Facultad de Ingeniería Civil y Ambiental - FIC'
  7  );

1 fila creada.

SQL> COMMIT;

Confirmaci¾n terminada.
```



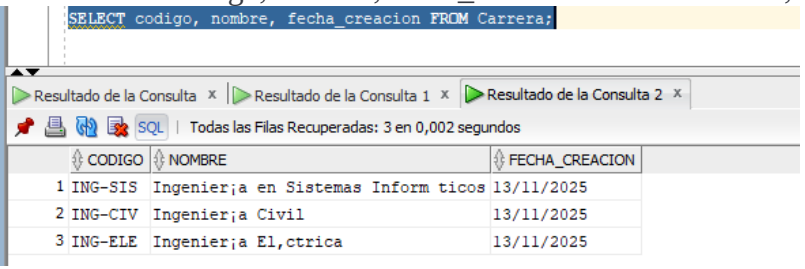
	CODIGO	NOMBRE	DURACION_SEMESTRES	CREDITOS_TOTALES	FACULTAD
1	ING-SIS	Ingenier¡a en Sistemas Inform ticos	10	180	Facultad de Sistemas - ESFOT
2	ING-CIV	Ingenier¡a Civil	10	200	Facultad de Ingenier¡a Civil y Ambiental -

```
-- También se puede usar la palabra DEFAULT explícitamente
INSERT INTO Carrera (
    codigo,
    nombre,
    duracion_semestres,
    creditos_totales,
    facultad,
    fecha_creacion
```

```
) VALUES (
    'ING-ELE',
    'Ingeniería Eléctrica',
    10,
    190,
    'Facultad de Ingeniería Eléctrica y Electrónica - FIE',
    DEFAULT  -- Usa el valor DEFAULT definido en la tabla
);
```

```
SQL> INSERT INTO Carrera ( codigo,
  2  nombre, duracion_semestres, creditos_totales, facultad, fecha_creacion
  3  ) VALUES ( 'ING-ELE',
  4  'Ingeniería Eléctrica', 10,
  5  190,
  6  'Facultad de Ingeniería Eléctrica y Electrónica - FIE', DEFAULT -- Usa el valor DEFAULT definido
en la tabla
  7  );

1 fila creada.

SQL> COMMIT;

Confirmaci¾n terminada.
```

```
-- Verificar
SELECT codigo, nombre, fecha_creacion FROM Carrera;
```



Ejemplo 3: INSERT sin Especificar Columnas

```
-- EJEMPLO 3: INSERT de todas las columnas en orden
-- ================================================

-- IMPORTANTE: Debe proporcionar valores en el ORDEN exacto de la tabla
-- Usar DESC para ver el orden de las columnas
DESC Carrera;
```

```
SQL> DESC Carrera;
 Nombre                                    ┌Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CODIGO                                    NOT NULL VARCHAR2(10)
 NOMBRE                                    NOT NULL VARCHAR2(100)
 DURACION_SEMESTRES                        NOT NULL NUMBER(2)
 CREDITOS_TOTALES                          NOT NULL NUMBER(3)
 FACULTAD                                  NOT NULL VARCHAR2(100)
 FECHA_CREACION                            NOT NULL DATE
```

```
-- Insertar proporcionando valores en orden
INSERT INTO Carrera
VALUES (
    'ING-MEC',                    -- codigo
    'Ingeniería Mecánica',          -- nombre
    10,                           -- duracion_semestres
    195,                          -- creditos_totales
    'Facultad de Ingeniería Mecánica - FIM', -- facultad
    SYSDATE                       -- fecha_creacion
);
```

```
SQL> DESC Carrera;
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CODIGO                                    NOT NULL VARCHAR2(10)
 NOMBRE                                    NOT NULL VARCHAR2(100)
 DURACION_SEMESTRES                        NOT NULL NUMBER(2)
 CREDITOS_TOTALES                          NOT NULL NUMBER(3)
 FACULTAD                                  NOT NULL VARCHAR2(100)
 FECHA_CREACION                            NOT NULL DATE

SQL> INSERT INTO Carrera VALUES (
  2  'ING-MEC', -- codigo
  3  'Ingeniería Mecánica',     -- nombre
  4  10,        -- duracion_semestres
  5  195,       -- creditos_totales
  6  'Facultad de Ingeniería Mecánica - FIM', -- facultad
  7  SYSDATE    -- fecha_creacion
  8  );

1 fila creada.

SQL> COMMIT;

Confirmaci%n terminada.
```

```
SELECT codigo, nombre, fecha_creacion FROM Carrera;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 4 en 0,002 segundos

	CODIGO	NOMBRE	FECHA_CREACION
1	ING-SIS	Ingenier¡a en Sistemas Inform ticos	13/11/2025
2	ING-CIV	Ingenier¡a Civil	13/11/2025
3	ING-ELE	Ingenier¡a El,ctrica	13/11/2025
4	ING-MEC	Ingenier¡a Mec nica	13/11/2025


```
-- ⚠ ADVERTENCIA: Este método es propenso a errores si cambia la estructura
-- RECOMENDACIÓN: Siempre especificar las columnas explícitamente
```

Ejemplo 4: INSERT de Múltiples Filas

```
-- EJEMPLO 4: INSERT múltiple usando INSERT ALL
-- ================================================

-- Oracle no soporta INSERT múltiple con VALUES, VALUES, VALUES
-- En su lugar, usar INSERT ALL con SELECT FROM DUAL

INSERT ALL
    INTO Carrera VALUES ('ING-QUI', 'Ingeniería Química', 10, 185, 'Facultad de Ingeniería Química - FIQ', SYSDATE)
    INTO Carrera VALUES ('ING-IND', 'Ingeniería Industrial', 10, 175, 'Facultad de Ingeniería Industrial - FIIS', SYSDATE)
    INTO Carrera VALUES ('ING-GEO', 'Ingeniería Geológica', 10, 192, 'Facultad de Geología y Petróleos - FGP', SYSDATE)
SELECT * FROM DUAL;

-- Verificar inserciones
SELECT codigo, nombre FROM Carrera ORDER BY codigo;
```

```
SQL> INSERT ALL
  2  INTO Carrera VALUES ('ING-QUI', 'Ingeniería Química', 10, 185, 'Facultad de Ingeniería Química -
FIQ', SYSDATE) INTO Carrera VALUES ('ING-IND', 'Ingeniería Industrial', 10, 175, 'Facultad de Ingenier
ía Industrial - FIIS', SYSDATE) INTO Carrera VALUES ('ING-GEO', 'Ingeniería Geológica', 10, 192, 'Facu
ltad de Geología y Petróleos - FGP', SYSDATE)
  3  SELECT * FROM DUAL;

3 filas creadas.

SQL>
SQL> -- Verificar inserciones
SQL> SELECT codigo, nombre FROM Carrera ORDER BY codigo;

CODIGO     NOMBRE
---------- --------------------------------------------
ING-CIV    Ingeniería Civil
ING-ELE    Ingeniería Eléctrica
ING-GEO    Ingeniería Geológica
ING-IND    Ingeniería Industrial
ING-MEC    Ingeniería Mecánica
ING-QUI    Ingeniería Química
ING-SIS    Ingeniería en Sistemas Informáticos

7 filas seleccionadas.
```

```
SELECT codigo, nombre FROM Carrera ORDER BY codigo;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la

SQL | Todas las Filas Recuperadas: 7 en 0,003 segundos

	CODIGO	NOMBRE
1	ING-CIV	Ingenier¡a Civil
2	ING-ELE	Ingenier¡a El,ctrica
3	ING-GEO	Ingenier¡a Geol◆gica
4	ING-IND	Ingenier¡a Industrial
5	ING-MEC	Ingenier¡a Mec nica
6	ING-QUI	Ingenier¡a Qu¡mica
7	ING-SIS	Ingenier¡a en Sistemas Inform ticos

Ejemplo 5: INSERT con Subconsulta

```
-- EJEMPLO 5: INSERT FROM SELECT
-- ================================================

-- Crear tabla temporal para ejemplo
CREATE TABLE Carrera_Historico (
    codigo VARCHAR2(10),
    nombre VARCHAR2(100),
    fecha_registro DATE
);
```

```
SQL> CREATE TABLE Carrera_Historico ( codigo VARCHAR2(10),
  2  nombre VARCHAR2(100), fecha_registro DATE
  3  );

Tabla creada.

SQL>
```

```sql
-- Insertar usando subconsulta
INSERT INTO Carrera_Historico (codigo, nombre, fecha_registro)
SELECT codigo, nombre, fecha_creacion
FROM Carrera
WHERE duracion_semestres = 10;
```

```
SQL> INSERT INTO Carrera_Historico (codigo, nombre, fecha_registro) SELECT codigo, nombre, fecha_creac
ion
  2  FROM Carrera
  3  WHERE duracion_semestres = 10;

7 filas creadas.
```

```sql
-- Verificar
SELECT * FROM Carrera_Historico;
```

```
SELECT * FROM Carrera_Historico;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 7 en 0,008 segundos

	CODIGO	NOMBRE	FECHA_REGISTRO
1	ING-SIS	Ingenier¡a en Sistemas Inform ticos	13/11/2025
2	ING-CIV	Ingenier¡a Civil	13/11/2025
3	ING-ELE	Ingenier¡a El,ctrica	13/11/2025
4	ING-MEC	Ingenier¡a Mec nica	13/11/2025
5	ING-QUI	Ingenier¡a Qu¡mica	13/11/2025
6	ING-IND	Ingenier¡a Industrial	13/11/2025
7	ING-GEO	Ingenier¡a Geol◦gica	13/11/2025

```
-- Limpiar tabla temporal
DROP TABLE Carrera_Historico PURGE;
```

```
SQL> DROP TABLE Carrera_Historico PURGE;

Tabla borrada.
```

Ejemplo 6: INSERT en DOCENTE

```
-- EJEMPLO 6: Insertar docentes
-- ==========================================================
INSERT INTO Docente (
    cedula,
    nombres,
    apellidos,
    email,
    telefono,
    titulo,
```

```
    especialidad,
    tipo_contrato,
    fecha_ingreso
) VALUES (
    '1712345678',
    'Carlos Alberto',
    'Pérez González',
    'carlos.perez@epn.edu.ec',
    '0998765432',
    'PhD en Ciencias de la Computación',
    'Bases de Datos y Big Data',
    'TIEMPO_COMPLETO',
    SYSDATE
);
```

```
SQL> INSERT INTO Docente ( cedula,
  2  nombres, apellidos, email, telefono, titulo, especialidad, tipo_contrato, fecha_ingreso
  3  ) VALUES ( '1712345678',
  4  'Carlos Alberto', 'Pérez González',
  5  'carlos.perez@epn.edu.ec', '0998765432',
  6  'PhD en Ciencias de la Computación', 'Bases de Datos y Big Data', 'TIEMPO_COMPLETO',
  7  SYSDATE
  8  );

1 fila creada.
```

```
-- Insertar otro docente con valores NULL opcionales
INSERT INTO Docente (
    cedula,
    nombres,
    apellidos,
```

```
    email,
    titulo,
    especialidad,
    tipo_contrato
) VALUES (
    '1723456789',
    'María Fernanda',
    'López Sánchez',
    'maria.lopez@epn.edu.ec',
    'MSc en Redes y Telecomunicaciones',
    'Redes de Computadores',
    'MEDIO_TIEMPO'
    -- telefono y fecha_ingreso se omiten (NULL y DEFAULT respectivamente)
);
```

```
SQL> INSERT INTO Docente ( cedula,
  2  nombres, apellidos,
  3  email, titulo,
  4  especialidad, tipo_contrato
  5  ) VALUES ( '1723456789',
  6  'María Fernanda', 'López Sánchez',
  7  'maria.lopez@epn.edu.ec',
  8  'MSc en Redes y Telecomunicaciones', 'Redes de Computadores', 'MEDIO_TIEMPO'
  9  -- telefono y fecha_ingreso se omiten (NULL y DEFAULT respectivamente)
 10  );

1 fila creada.
```

```
-- Insertar más docentes
INSERT ALL
    INTO Docente VALUES ('1734567890', 'Pedro José', 'Martínez Torres', 'pedro.martinez@epn.edu.ec', '0987654321',
    'PhD en Ingeniería de Software', 'Desarrollo de Software', 'TIEMPO_COMPLETO', SYSDATE)
```

```
    INTO Docente VALUES ('1745678901', 'Ana Lucía', 'García Ramírez', 'ana.garcia@epn.edu.ec', '0976543210',
     'MSc en Inteligencia Artificial', 'Machine Learning', 'TIEMPO_COMPLETO', SYSDATE)
    INTO Docente VALUES ('1756789012', 'Roberto Carlos', 'Sánchez Vera', 'roberto.sanchez@epn.edu.ec', NULL,
     'Ingeniero en Sistemas', 'Programación Web', 'HORA_CLASE', SYSDATE)
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
  2  INTO Docente VALUES ('1734567890', 'Pedro José', 'Martínez Torres', 'pedro.martinez@epn.edu.ec',
'0987654321',
  3  'PhD en Ingeniería de Software', 'Desarrollo de Software', 'TIEMPO_COMPLETO', SYSDATE)
  4  INTO Docente VALUES ('1745678901', 'Ana Lucía', 'García Ramírez', 'ana.garcia@epn.edu.ec', '09765
43210',
  5   'MSc en Inteligencia Artificial', 'Machine Learning', 'TIEMPO_COMPLETO', SYSDATE)
  6  INTO Docente VALUES ('1756789012', 'Roberto Carlos', 'Sánchez Vera', 'roberto.sanchez@epn.edu.ec'
, NULL,
  7   'Ingeniero en Sistemas', 'Programación Web', 'HORA_CLASE', SYSDATE)
  8  SELECT * FROM DUAL;

3 filas creadas.
```

```
-- Verificar
SELECT cedula, nombres, apellidos, especialidad, tipo_contrato
FROM Docente
ORDER BY apellidos;
```

Hoja de Trabajo | Generador de Consultas

```
SELECT cedula, nombres, apellidos, especialidad, tipo_contrato
FROM Docente
ORDER BY apellidos;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 5 en 0,002 segundos

	CEDULA	NOMBRES	APELLIDOS	ESPECIALIDAD	TIPO_CONTRATO
1	1745678901	Ana Luc¡a	Garc¡a Ram¡rez	Machine Learning	TIEMPO_COMPLETO
2	1723456789	Mar¡a Fernanda	L◆pez S nchez	Redes de Computadores	MEDIO_TIEMPO
3	1734567890	Pedro Jos,	Mart¡nez Torres	Desarrollo de Software	TIEMPO_COMPLETO
4	1712345678	Carlos Alberto	P,rez Gonz lez	Bases de Datos y Big Data	TIEMPO_COMPLETO
5	1756789012	Roberto Carlos	S nchez Vera	Programaci◆n Web	HORA_CLASE

Ejemplo 7: INSERT en ESTUDIANTE

```
-- EJEMPLO 7: Insertar estudiantes
-- ================================================
-- Insertar primer estudiante
INSERT INTO Estudiante (
    cedula,
    nombres,
    apellidos,
    email,
    telefono,
    fecha_nacimiento,
    genero,
    codigo_carrera,
    estado,
    creditos_aprobados,
    fecha_ingreso
```

```
) VALUES (
    '1750123456',
    'Juan Pablo',
    'Andrade Morales',
    'juan.andrade@epn.edu.ec',
    '0991234567',
    TO_DATE('2003-05-15', 'YYYY-MM-DD'),
    'M',
    'ING-SIS',
    'ACTIVO',
    0,
    SYSDATE
);
```

```
SQL> INSERT INTO Estudiante ( cedula,
  2  nombres, apellidos, email, telefono,
  3  fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso
  4  ) VALUES ( '1750123456',
  5  'Juan Pablo', 'Andrade Morales',
  6  'juan.andrade@epn.edu.ec', '0991234567',
  7  TO_DATE('2003-05-15', 'YYYY-MM-DD'), 'M',
  8  'ING-SIS',
  9  'ACTIVO', 0, SYSDATE
 10  );

1 fila creada.
```

```
-- Insertar varios estudiantes
INSERT ALL
    INTO Estudiante VALUES ('1750234567', 'María José', 'Benítez Castro', 'maria.benitez@epn.edu.ec', '0992345678',
    TO_DATE('2002-08-20', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'ACTIVO', 0, SYSDATE)
```

```
    INTO Estudiante VALUES ('1750345678', 'Carlos Andrés', 'Castillo Díaz', 'carlos.castillo@epn.edu.ec', '0993456789',
    TO_DATE('2003-11-10', 'YYYY-MM-DD'), 'M', 'ING-CIV', 'ACTIVO', 0, SYSDATE)
    INTO Estudiante VALUES ('1750456789', 'Ana Sofía', 'Domínguez Escobar', 'ana.dominguez@epn.edu.ec', '0994567890',
    TO_DATE('2002-03-25', 'YYYY-MM-DD'), 'F', 'ING-ELE', 'ACTIVO', 0, SYSDATE)
    INTO Estudiante VALUES ('1750567890', 'Pedro Luis', 'Espinoza Flores', 'pedro.espinoza@epn.edu.ec', '0995678901',
    TO_DATE('2003-07-18', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'ACTIVO', 0, SYSDATE)
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
  2  INTO Estudiante VALUES ('1750234567', 'María José', 'Benítez Castro', 'maria.benitez@epn.edu.ec',
 '0992345678',
  3  TO_DATE('2002-08-20', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'ACTIVO', 0, SYSDATE)
  4  INTO Estudiante VALUES ('1750345678', 'Carlos Andrés', 'Castillo Díaz', 'carlos.castillo@epn.edu.
ec', '0993456789',
  5  TO_DATE('2003-11-10', 'YYYY-MM-DD'), 'M', 'ING-CIV', 'ACTIVO', 0, SYSDATE)
  6  INTO Estudiante VALUES ('1750456789', 'Ana Sofía', 'Domínguez Escobar', 'ana.dominguez@epn.edu.ec
', '0994567890',
  7  TO_DATE('2002-03-25', 'YYYY-MM-DD'), 'F', 'ING-ELE', 'ACTIVO', 0, SYSDATE)
  8  INTO Estudiante VALUES ('1750567890', 'Pedro Luis', 'Espinoza Flores', 'pedro.espinoza@epn.edu.ec
', '0995678901',
  9  TO_DATE('2003-07-18', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'ACTIVO', 0, SYSDATE)
 10  SELECT * FROM DUAL;

4 filas creadas.
```

```
-- Verificar inserciones
SELECT cedula, nombres, apellidos, codigo_carrera, estado
FROM Estudiante
ORDER BY apellidos;
```

```
SELECT cedula, nombres, apellidos, codigo_carrera, estado FROM Estudiante
ORDER BY apellidos;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Resultado de la Consulta 3

SQL | Todas las Filas Recuperadas: 5 en 0,008 segundos

	CEDULA	NOMBRES	APELLIDOS	CODIGO_CARRERA	ESTADO
1	1750123456	Juan Pablo	Andrade Morales	ING-SIS	ACTIVO
2	1750234567	Mar¡a Jos,	Ben¡tez Castro	ING-SIS	ACTIVO
3	1750345678	Carlos Andr,s	Castillo D¡az	ING-CIV	ACTIVO
4	1750456789	Ana Sof¡a	Dom¡nguez Escobar	ING-ELE	ACTIVO
5	1750567890	Pedro Luis	Espinoza Flores	ING-SIS	ACTIVO

Ejemplo 8: INSERT en ASIGNATURA

```
-- EJEMPLO 8: Insertar asignaturas
-- ================================================
-- Asignaturas de Ingeniería en Sistemas
INSERT ALL
    -- Primer nivel
    INTO Asignatura VALUES ('MAT-101', 'Cálculo Diferencial', 5, 1, 'ING-SIS', 4, 2, 'Introducción al cálculo diferencial e integral')
    INTO Asignatura VALUES ('FIS-101', 'Física I', 4, 1, 'ING-SIS', 3, 2, 'Fundamentos de mecánica clásica')
    INTO Asignatura VALUES ('PRG-101', 'Fundamentos de Programación', 5, 1, 'ING-SIS', 3, 4, 'Introducción a la programación estructurada')
    -- Segundo nivel
    INTO Asignatura VALUES ('MAT-201', 'Cálculo Integral', 5, 2, 'ING-SIS', 4, 2, 'Cálculo integral y series')
    INTO Asignatura VALUES ('PRG-201', 'Programación Orientada a Objetos', 5, 2, 'ING-SIS', 3, 4, 'POO con Java/C++')
    -- Tercer nivel
    INTO Asignatura VALUES ('EST-301', 'Estructura de Datos', 5, 3, 'ING-SIS', 3, 4, 'Listas, árboles, grafos, algoritmos')
    -- Quinto nivel
    INTO Asignatura VALUES ('BD-501', 'Fundamentos de Bases de Datos', 4, 5, 'ING-SIS', 3, 2, 'Introducción a bases de datos relacionales')
    -- Sexto nivel
    INTO Asignatura VALUES ('BD-601', 'Bases de Datos Avanzadas', 5, 6, 'ING-SIS', 3, 4, 'Administración, optimización y BD distribuidas')
```

```
SELECT * FROM DUAL;
```

```
SQL> -- Asignaturas de Ingeniería en Sistemas
SQL> INSERT ALL
  2  -- Primer nivel
  3  INTO Asignatura VALUES ('MAT-101', 'Cálculo Diferencial', 5, 1, 'ING-SIS', 4, 2, 'Introducción al
 cálculo diferencial e integral') INTO Asignatura VALUES ('FIS-101', 'Física I', 4, 1, 'ING-SIS', 3, 2
, 'Fundamentos de mecánica clásica')
  4  INTO Asignatura VALUES ('PRG-101', 'Fundamentos de Programación', 5, 1, 'ING-SIS', 3, 4, 'Introdu
cción a la programación estructurada')
  5  -- Segundo nivel
  6  INTO Asignatura VALUES ('MAT-201', 'Cálculo Integral', 5, 2, 'ING-SIS', 4, 2, 'Cálculo integral y
 series')
  7  INTO Asignatura VALUES ('PRG-201', 'Programación Orientada a Objetos', 5, 2, 'ING-SIS', 3, 4, 'PO
O con Java/C++')
  8  -- Tercer nivel
  9  INTO Asignatura VALUES ('EST-301', 'Estructura de Datos', 5, 3, 'ING-SIS', 3, 4, 'Listas, árboles
, grafos, algoritmos')
 10  -- Quinto nivel
 11  INTO Asignatura VALUES ('BD-501', 'Fundamentos de Bases de Datos', 4, 5, 'ING-SIS', 3, 2, 'Introd
ucción a bases de datos relacionales')
 12  -- Sexto nivel
 13  INTO Asignatura VALUES ('BD-601', 'Bases de Datos Avanzadas', 5, 6, 'ING-SIS', 3, 4, 'Administrac
ión, optimización y BD distribuidas') SELECT * FROM DUAL;

8 filas creadas.
```

```
-- Asignaturas de otras carreras
INSERT ALL
   INTO Asignatura VALUES ('CIV-101', 'Dibujo Técnico', 4, 1, 'ING-CIV', 2, 4, 'Dibujo técnico para ingeniería civil')
   INTO Asignatura VALUES ('ELE-101', 'Circuitos Eléctricos I', 5, 1, 'ING-ELE', 4, 2, 'Análisis de circuitos eléctricos')
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
  2  INTO Asignatura VALUES ('CIV-101', 'Dibujo Técnico', 4, 1, 'ING-CIV', 2, 4, 'Dibujo técnico para
ingeniería civil') INTO Asignatura VALUES ('ELE-101', 'Circuitos Eléctricos I', 5, 1, 'ING-ELE', 4, 2,
 'Análisis de circuitos eléctricos')
  3  SELECT * FROM DUAL;

2 filas creadas.
```

```
-- Verificar
SELECT codigo, nombre, creditos, nivel, codigo_carrera
from asignatura
ORDER BY codigo_carrera, nivel, codigo;
```

```
SELECT codigo, nombre, creditos, nivel, codigo_carrera
from asignatura
ORDER BY codigo_carrera, nivel, codigo;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Resultado de la Consulta 3

SQL | Todas las Filas Recuperadas: 10 en 0,009 segundos

	CODIGO	NOMBRE	CREDITOS	NIVEL	CODIGO_CARRERA
1	CIV-101	Dibujo T,cnico	4	1	ING-CIV
2	ELE-101	Circuitos El,ctricos I	5	1	ING-ELE
3	FIS-101	F¡sica I	4	1	ING-SIS
4	MAT-101	C lculo Diferencial	5	1	ING-SIS
5	PRG-101	Fundamentos de Programacien	5	1	ING-SIS
6	MAT-201	C lculo Integral	5	2	ING-SIS
7	PRG-201	Programacien Orientada a Objetos	5	2	ING-SIS
8	EST-301	Estructura de Datos	5	3	ING-SIS
9	BD-501	Fundamentos de Bases de Datos	4	5	ING-SIS
10	BD-601	Bases de Datos Avanzadas	5	6	ING-SIS

Ejemplo 9: INSERT en PRERREQUISITO


```
-- EJEMPLO 9: Definir prerrequisitos
-- ======================================================
-- MAT-201 (Cálculo Integral) requiere MAT-101 (Cálculo Diferencial)
INSERT INTO Prerrequisito (codigo_asignatura, codigo_prerrequisito)
VALUES ('MAT-201', 'MAT-101');

-- PRG-201 (POO) requiere PRG-101 (Fundamentos)
INSERT INTO Prerrequisito VALUES ('PRG-201', 'PRG-101');

-- EST-301 (Estructuras) requiere PRG-201 (POO)
INSERT INTO Prerrequisito VALUES ('EST-301', 'PRG-201');

-- BD-601 (BD Avanzadas) requiere BD-501 (Fundamentos BD)
INSERT INTO Prerrequisito VALUES ('BD-601', 'BD-501');

-- BD-601 también requiere EST-301 (Estructuras de Datos)
INSERT INTO Prerrequisito VALUES ('BD-601', 'EST-301');
```

```
SQL> INSERT INTO Prerrequisito (codigo_asignatura, codigo_prerrequisito) VALUES ('MAT-201', 'MAT-101')
;

1 fila creada.

SQL>
SQL> -- PRG-201 (POO) requiere PRG-101 (Fundamentos)
SQL> INSERT INTO Prerrequisito VALUES ('PRG-201', 'PRG-101');

1 fila creada.

SQL>
SQL> -- EST-301 (Estructuras) requiere PRG-201 (POO)
SQL> INSERT INTO Prerrequisito VALUES ('EST-301', 'PRG-201');

1 fila creada.

SQL>
SQL> -- BD-601 (BD Avanzadas) requiere BD-501 (Fundamentos BD)
SQL> INSERT INTO Prerrequisito VALUES ('BD-601', 'BD-501');

1 fila creada.

SQL>
SQL> -- BD-601 también requiere EST-301 (Estructuras de Datos)
SQL> INSERT INTO Prerrequisito VALUES ('BD-601', 'EST-301');

1 fila creada.
```

```
-- Verificar relaciones de prerrequisitos
SELECT
    p.codigo_asignatura,
    a1.nombre AS asignatura,
    p.codigo_prerrequisito,
```

```
    a2.nombre AS prerrequisito
FROM Prerrequisito p
JOIN Asignatura a1 ON p.codigo_asignatura = a1.codigo;
```

```
SELECT
p.codigo_asignatura, a1.nombre AS asignatura, p.codigo_prerrequisito AS prerrequisito
FROM Prerrequisito p
JOIN Asignatura a1 ON p.codigo_asignatura = a1.codigo;
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Resultado de la Consulta 3

SQL | Todas las Filas Recuperadas: 5 en 0,002 segundos

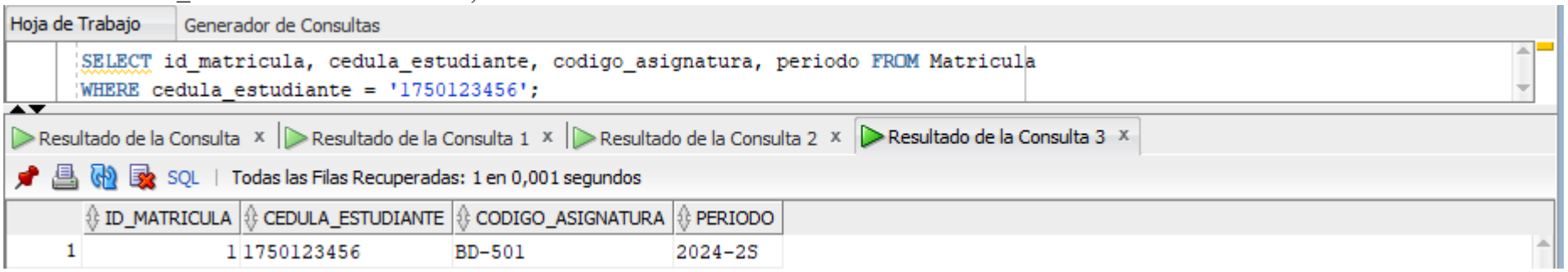
	CODIGO_ASIGNATURA	ASIGNATURA	PRERREQUISITO
1	MAT-201	C lculo Integral	MAT-101
2	PRG-201	Programacion Orientada a Objetos	PRG-101
3	EST-301	Estructura de Datos	PRG-201
4	BD-601	Bases de Datos Avanzadas	BD-501
5	BD-601	Bases de Datos Avanzadas	EST-301

Ejemplo 10: INSERT en MATRICULA (con Secuencia)

```
EJEMPLO 10: Insertar matrículas (AUTO-INCREMENT)
-- ================================================

-- Recordar: id_matricula se genera automáticamente con secuencia + trigger
-- No incluir id_matricula en el INSERT

-- Matricular estudiante en una asignatura
INSERT INTO Matricula (
    cedula_estudiante,
    codigo_asignatura,
    cedula_docente,
    periodo,
    paralelo,
    fecha_matricula,
    estado
) VALUES (
    '1750123456',          -- Juan Pablo Andrade
    'BD-501',              -- Fundamentos de Bases de Datos
    '1712345678',          -- Dr. Carlos Pérez
    '2024-2S',             -- Segundo semestre 2024
    'A',                   -- Paralelo A
    SYSDATE,
    'CURSANDO'
);
```

```
SQL> INSERT INTO Matricula ( cedula_estudiante, codigo_asignatura, cedula_docente, periodo,
  2  paralelo, fecha_matricula, estado
  3  ) VALUES (
  4  '1750123456',
  5  'BD-501',
  6  '1712345678',
  7  '2024-2S',
  8  'A',
  9  SYSDATE,
 10  'CURSANDO'
 11  );

1 fila creada.
```

```
-- El trigger asigna automáticamente el id_matricula
-- Verificar el ID generado
SELECT id_matricula, cedula_estudiante, codigo_asignatura, periodo
FROM Matricula
WHERE cedula_estudiante = '1750123456';
```



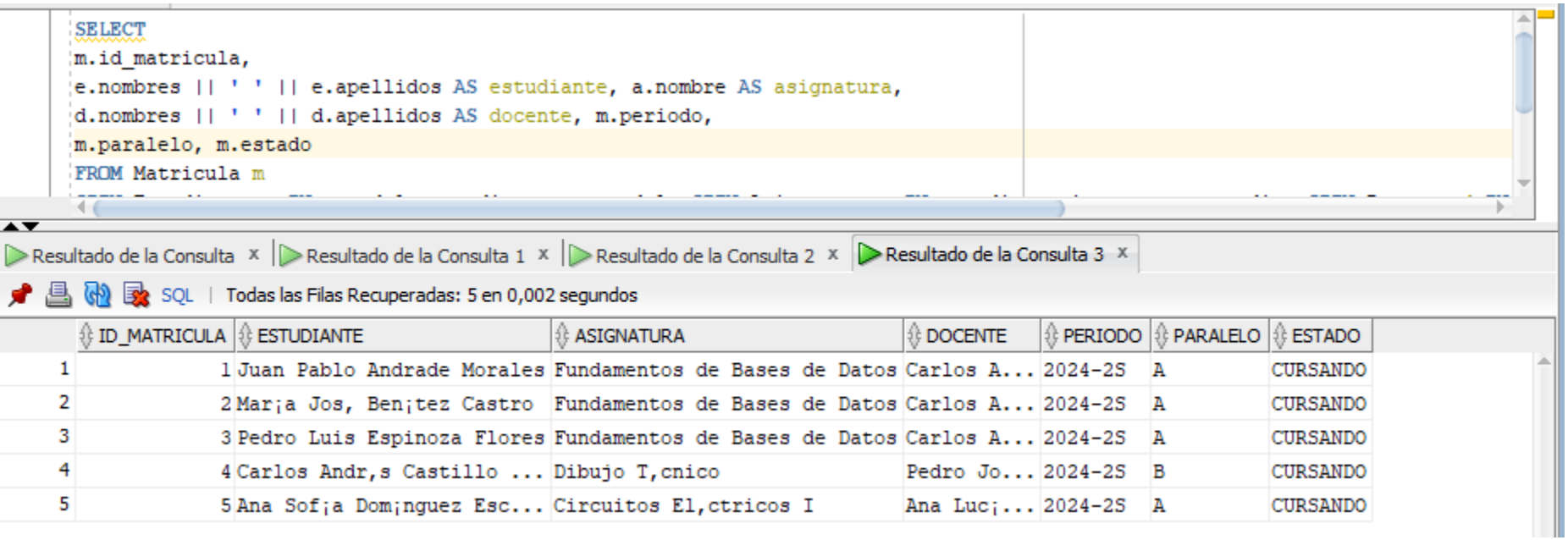
ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

```
-- Insertar más matrículas (sin especificar id_matricula)
INSERT ALL
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
        VALUES ('1750234567', 'BD-501', '1712345678', '2024-2S', 'A')
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
        VALUES ('1750567890', 'BD-501', '1712345678', '2024-2S', 'A')
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
        VALUES ('1750345678', 'CIV-101', '1734567890', '2024-2S', 'B')
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
        VALUES ('1750456789', 'ELE-101', '1745678901', '2024-2S', 'A')
SELECT * FROM DUAL;
```

```
SQL> INSERT ALL
  2  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo) VALUES (
'1750234567', 'BD-501', '1712345678', '2024-2S', 'A')
  3  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo) VALUES (
'1750567890', 'BD-501', '1712345678', '2024-2S', 'A')
  4  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo) VALUES (
'1750345678', 'CIV-101', '1734567890', '2024-2S', 'B')
  5  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo) VALUES (
'1750456789', 'ELE-101', '1745678901', '2024-2S', 'A')
  6  SELECT * FROM DUAL;

4 filas creadas.
```

```
-- Verificar todas las matrículas con información completa
SELECT
    m.id_matricula,
    e.nombres || ' ' || e.apellidos AS estudiante,
    a.nombre AS asignatura,
    d.nombres || ' ' || d.apellidos AS docente,
    m.periodo,
```

```
    m.paralelo,
    m.estado
FROM Matricula m
JOIN Estudiante e ON m.cedula_estudiante = e.cedula
JOIN Asignatura a ON m.codigo_asignatura = a.codigo
JOIN Docente d ON m.cedula_docente = d.cedula
ORDER BY m.id_matricula;
```



```
SELECT
m.id_matricula,
e.nombres || ' ' || e.apellidos AS estudiante, a.nombre AS asignatura,
d.nombres || ' ' || d.apellidos AS docente, m.periodo,
m.paralelo, m.estado
FROM Matricula m
```

Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Resultado de la Consulta 3

SQL | Todas las Filas Recuperadas: 5 en 0,002 segundos

	ID_MATRICULA	ESTUDIANTE	ASIGNATURA	DOCENTE	PERIODO	PARALELO	ESTADO
1	1	Juan Pablo Andrade Morales	Fundamentos de Bases de Datos	Carlos A...	2024-2S	A	CURSANDO
2	2	Mar¡a Jos, Ben¡tez Castro	Fundamentos de Bases de Datos	Carlos A...	2024-2S	A	CURSANDO
3	3	Pedro Luis Espinoza Flores	Fundamentos de Bases de Datos	Carlos A...	2024-2S	A	CURSANDO
4	4	Carlos Andr,s Castillo ...	Dibujo T,cnico	Pedro Jo...	2024-2S	B	CURSANDO
5	5	Ana Sof¡a Dom¡nguez Esc...	Circuitos El,ctricos I	Ana Luc¡...	2024-2S	A	CURSANDO

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Errores Comunes en INSERT

```
-- ERRORES COMUNES Y CÓMO EVITARLOS
-- ================================================

-- ERROR 1: Violación de PRIMARY KEY (duplicado)
INSERT INTO Carrera VALUES ('ING-SIS', 'Otra Carrera', 10, 180, 'Otra Facultad', SYSDATE);
-- ORA-00001: unique constraint (GESTION_ACADEMICA.PK_CARRERA) violated

-- ERROR 2: Violación de FOREIGN KEY (referencia inexistente)
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750999999', 'Test', 'Usuario', 'test@epn.edu.ec', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'XXX-XXX');
-- ORA-02291: integrity constraint (FK_ESTUDIANTE_CARRERA) violated - parent key not found

-- ERROR 3: Violación de CHECK constraint
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750888888', 'Test', 'Usuario', 'test2@epn.edu.ec', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'X', 'ING-SIS');
-- ORA-02290: check constraint (CHK_ESTUDIANTE_GENERO) violated

-- ERROR 4: Violación de UNIQUE constraint
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750777777', 'Otro', 'Usuario', 'juan.andrade@epn.edu.ec', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');
-- ORA-00001: unique constraint (UK_ESTUDIANTE_EMAIL) violated

-- ERROR 5: NOT NULL constraint
INSERT INTO Estudiante (cedula, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750666666', 'Apellido', 'test3@epn.edu.ec', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');
```

```
-- ORA-01400: cannot insert NULL into (GESTION_ACADEMICA.ESTUDIANTE.NOMBRES)

-- ERROR 6: Tipo de dato incorrecto
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera, creditos_aprobados)
VALUES ('1750555555', 'Test', 'Usuario', 'test4@epn.edu.ec', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'CIEN');

-- ORA-01722: invalid number

-- ERROR 7: Formato de fecha incorrecto
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750444444', 'Test', 'Usuario', 'test5@epn.edu.ec', '15/05/2000', 'M', 'ING-SIS');
-- ORA-01861: literal does not match format string
-- Usar TO_DATE('15/05/2000', 'DD/MM/YYYY') o TO_DATE('2000-05-15', 'YYYY-MM-DD')
```

Buenas Prácticas INSERT

```
-- BUENAS PRÁCTICAS
-- ================================================

-- ✓ 1. Siempre especificar las columnas explícitamente
INSERT INTO Carrera (codigo, nombre, duracion_semestres, creditos_totales, facultad)
VALUES ('ING-AMB', 'Ingeniería Ambiental', 10, 188, 'Facultad Ambiental');

-- ✓ 2. Usar TO_DATE para fechas
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750333333', 'Test', 'Usuario', 'test6@epn.edu.ec',
```

```
    TO_DATE('2003-06-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

-- ✓ 3. Manejar valores NULL explícitamente cuando sea necesario
INSERT INTO Docente (cedula, nombres, apellidos, email, telefono, titulo, especialidad, tipo_contrato)
VALUES ('1767890123', 'Luis', 'Vega', 'luis.vega@epn.edu.ec',
        NULL,  -- telefono explícitamente NULL
        'Ingeniero', 'Base de Datos', 'HORA_CLASE');

-- ✓ 4. Usar transacciones para inserciones relacionadas
SAVEPOINT antes_matricula;

INSERT INTO Estudiante (...) VALUES (...);
INSERT INTO Matricula (...) VALUES (...);

-- Si algo falla, volver al savepoint
-- ROLLBACK TO antes_matricula;

-- Si todo OK, confirmar
COMMIT;

-- ✓ 5. Validar datos antes de insertar (en aplicación)
-- Verificar que la carrera existe antes de insertar estudiante
SELECT COUNT(*) FROM Carrera WHERE codigo = 'ING-SIS';
-- Si es > 0, proceder con INSERT

-- ✓ 6. Usar INSERT ALL para múltiples inserciones eficientes
-- Más eficiente que múltiples INSERT individuales
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

UPDATE - ACTUALIZACIÓN DE DATOS

Sintaxis Básica

```
-- Formato básico
UPDATE nombre_tabla
SET columna1 = valor1,
    columna2 = valor2,
    columna3 = valor3
WHERE condicion;

-- UPDATE con subconsulta
UPDATE nombre_tabla
SET columna = (SELECT ... FROM ... WHERE ...)
WHERE condicion;

-- UPDATE de múltiples tablas (limitado en Oracle)
-- Se debe usar MERGE o UPDATE con subconsulta
```

⚠ **ADVERTENCIA CRÍTICA**: Si omite la cláusula WHERE, se actualizarán TODAS las filas de la tabla.

Ejemplo 1: UPDATE Simple

```
-- EJEMPLO 1: Actualización simple de un registro
-- ================================================
```

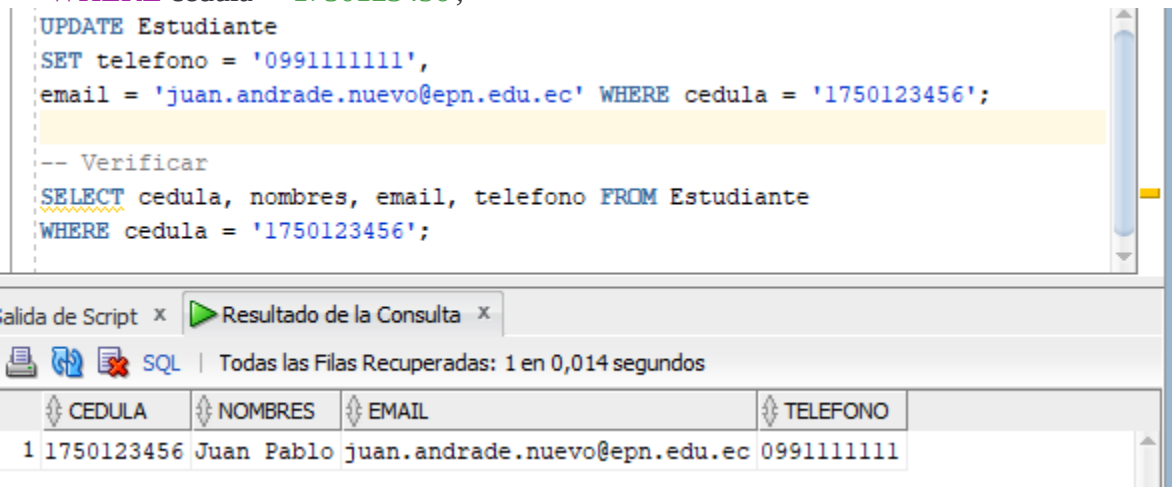
```
-- Actualizar el teléfono de un docente
UPDATE Docente
SET telefono = '0987654321'
WHERE cedula = '1723456789';

-- Verificar cambio
SELECT cedula, nombres, apellidos, telefono
FROM Docente
WHERE cedula = '1723456789';
```

```
SQL> UPDATE Docente
  2  SET telefono = '0987654321' WHERE cedula = '1723456789';

1 fila actualizada.

SQL>
SQL> -- Verificar cambio
SQL> SELECT cedula, nombres, apellidos, telefono FROM Docente
  2  WHERE cedula = '1723456789';

CEDULA     NOMBRES
---------- --------------------------------------------------
APELLIDOS                                          TELEFONO
-------------------------------------------------- ----------
1723456789 María Fernanda
López Sánchez                                      0987654321
```

```
-- Ver cuántas filas se afectaron
-- SQL%ROWCOUNT en PL/SQL o revisar el mensaje del cliente SQL
```

Ejemplo 2: UPDATE Múltiples Columnas

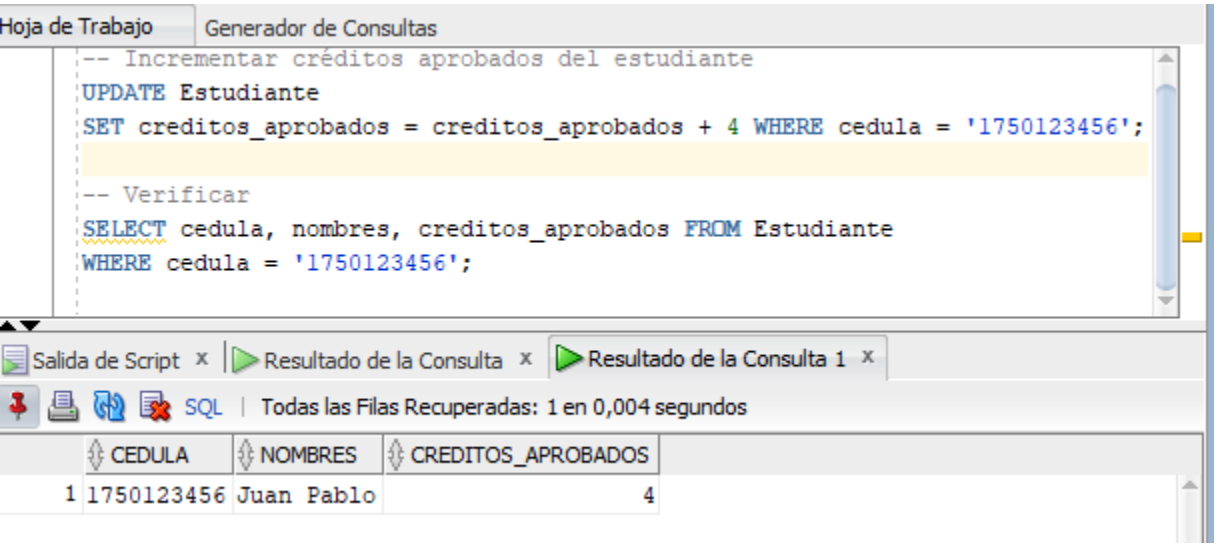
```
-- EJEMPLO 2: Actualizar varias columnas a la vez
-- ================================================

-- Actualizar información de un estudiante
UPDATE Estudiante
SET telefono = '0991111111',
    email = 'juan.andrade.nuevo@epn.edu.ec'
WHERE cedula = '1750123456';

-- Verificar
SELECT cedula, nombres, email, telefono
FROM Estudiante
WHERE cedula = '1750123456';
```



Ejemplo 3: UPDATE con Cálculos

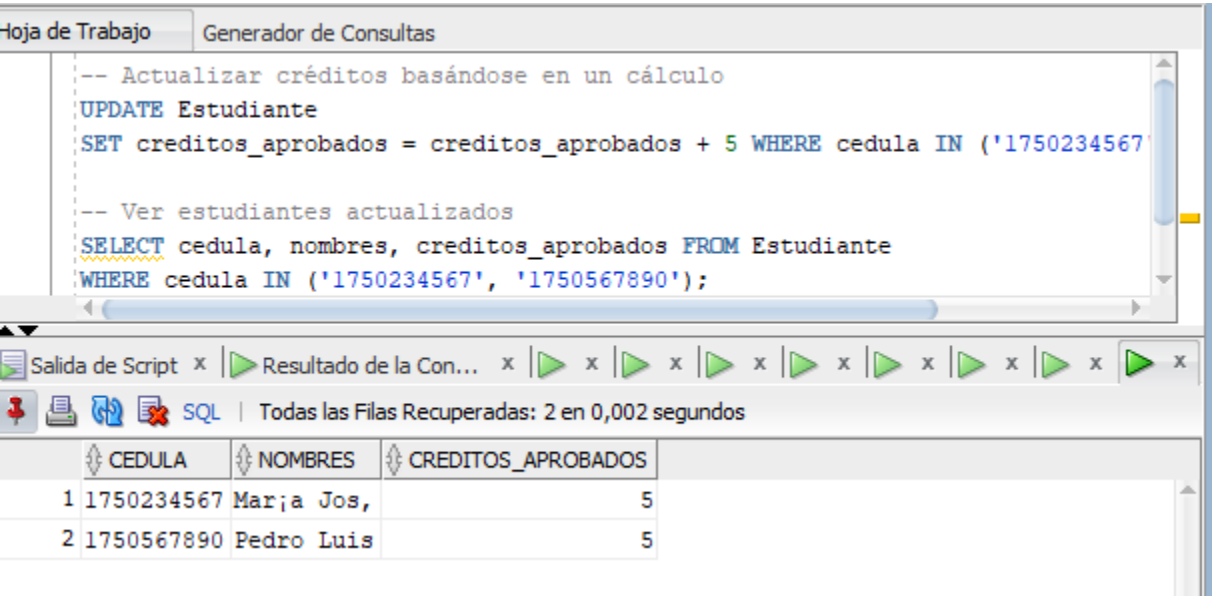
```
-- EJEMPLO 3: UPDATE usando expresiones y cálculos
-- ================================================
```

```
-- Incrementar créditos aprobados del estudiante
UPDATE Estudiante
SET creditos_aprobados = creditos_aprobados + 4
WHERE cedula = '1750123456';

-- Verificar
SELECT cedula, nombres, creditos_aprobados
FROM Estudiante
WHERE cedula = '1750123456';
```




```
-- Actualizar créditos basándose en un cálculo
UPDATE Estudiante
SET creditos_aprobados = creditos_aprobados + 5
WHERE cedula IN ('1750234567', '1750567890');

-- Ver estudiantes actualizados
```

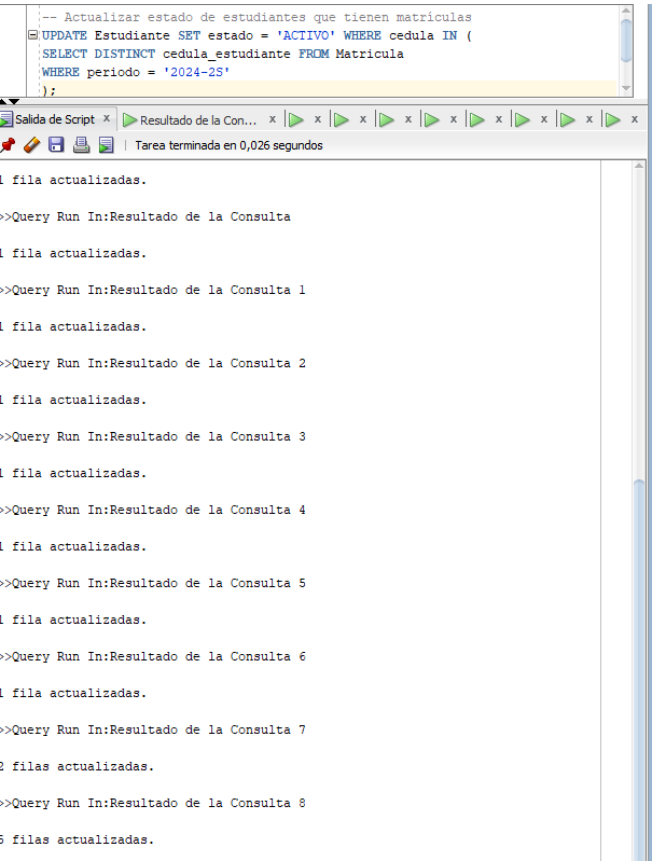
```
SELECT cedula, nombres, creditos_aprobados
FROM Estudiante
WHERE cedula IN ('1750234567', '1750567890');
```



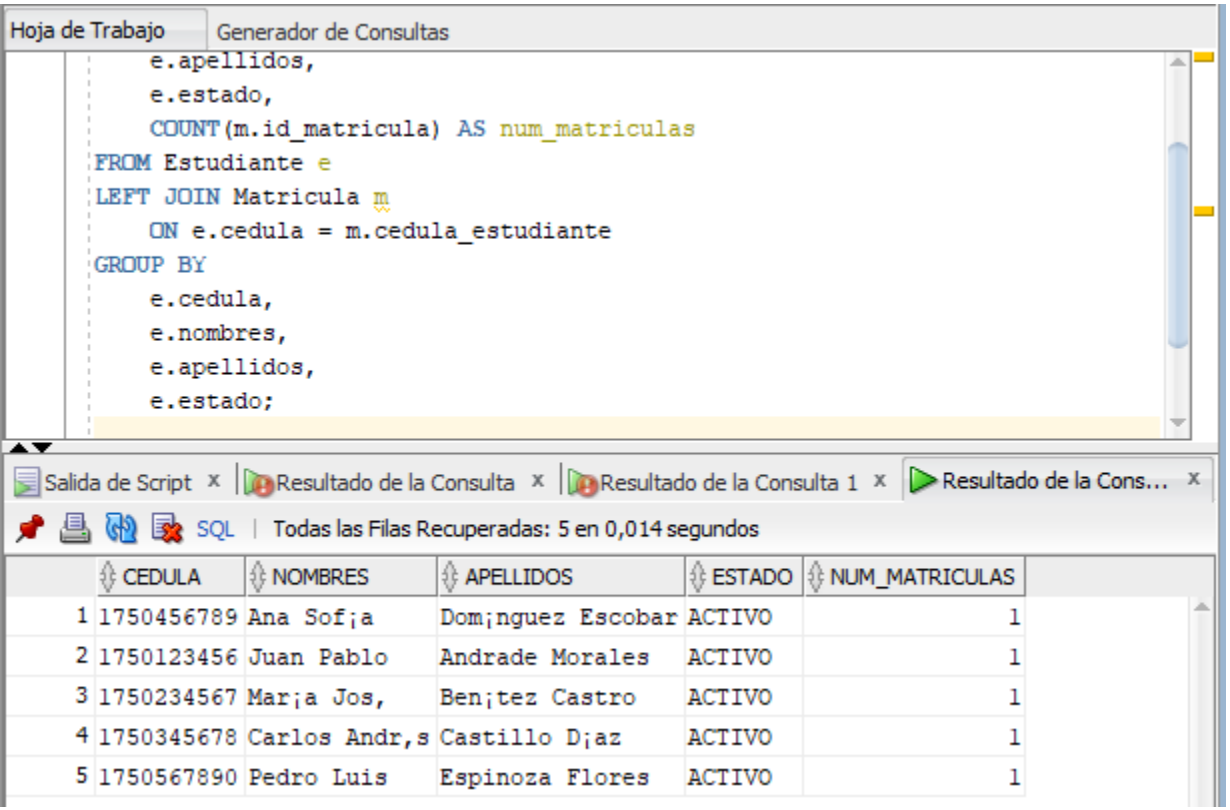
Ejemplo 4: UPDATE con Subconsulta

```
-- EJEMPLO 4: UPDATE usando subconsulta
-- ==================================================

-- Actualizar estado de estudiantes que tienen matrículas
UPDATE Estudiante
SET estado = 'ACTIVO'
WHERE cedula IN (
    SELECT DISTINCT cedula_estudiante
    FROM Matricula
    WHERE periodo = '2024-2S'
```

);



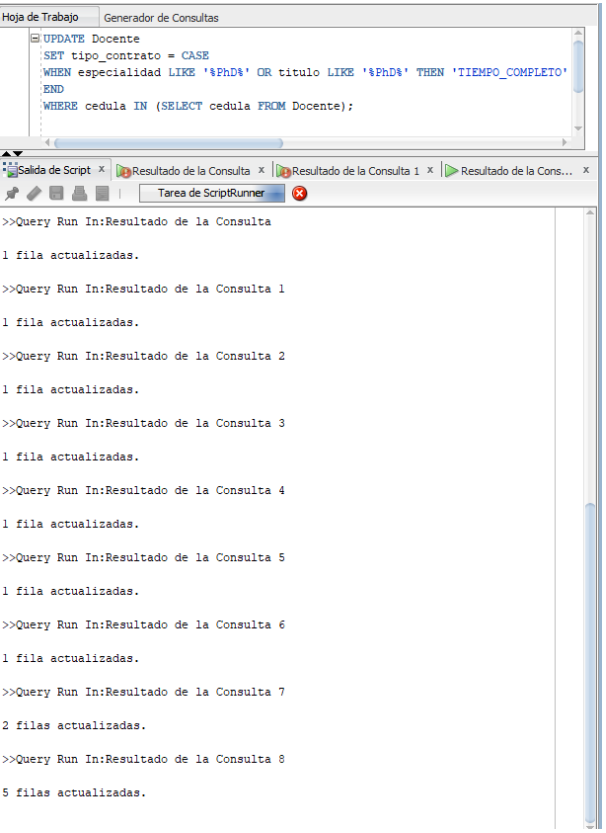

```
-- Verificar
SELECT e.cedula, e.nombres, e.estado, COUNT(m.id_matricula) AS num_matriculas
FROM Estudiante e
LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante
GROUP BY e.cedula, e.nombres, e.estado
ORDER BY e.apellidos;
```



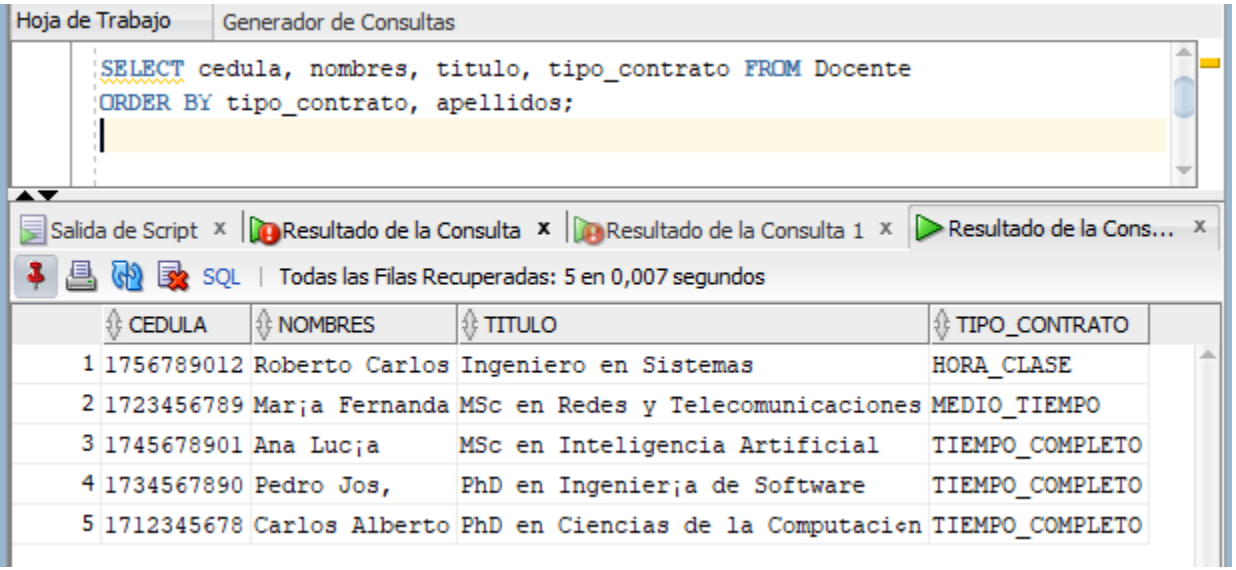
Ejemplo 5: UPDATE con CASE

```
-- EJEMPLO 5: UPDATE condicional con CASE
-- ================================================

-- Actualizar tipo de contrato basándose en especialidad
UPDATE Docente
SET tipo_contrato = CASE
    WHEN especialidad LIKE '%PhD%' OR titulo LIKE '%PhD%' THEN 'TIEMPO_COMPLETO'
    WHEN especialidad LIKE '%MSc%' OR titulo LIKE '%MSc%' THEN 'MEDIO_TIEMPO'
    ELSE 'HORA_CLASE'
```

```
END
WHERE cedula IN (SELECT cedula FROM Docente);
```

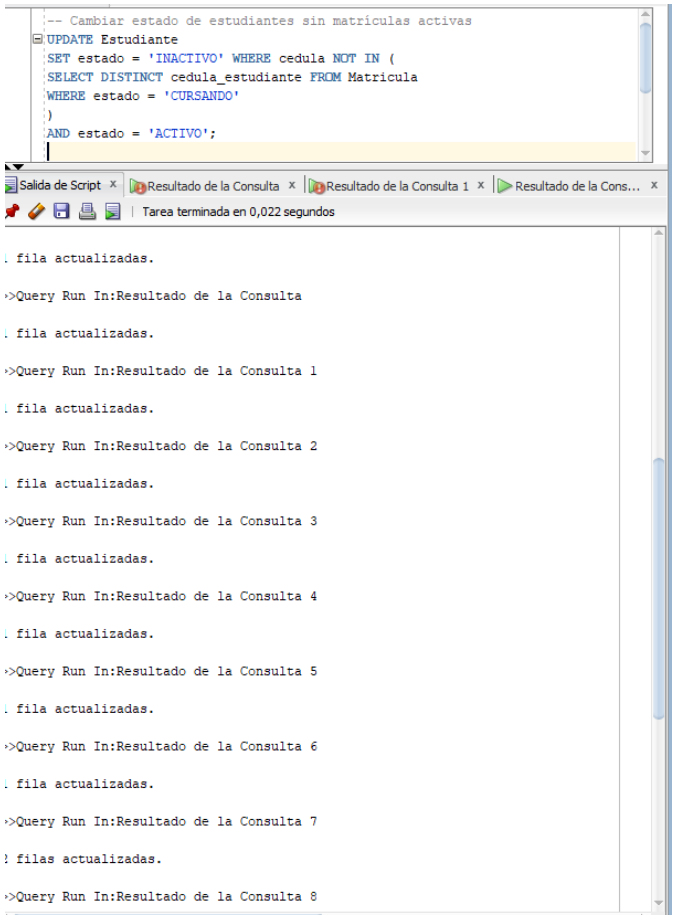



```
-- Verificar
SELECT cedula, nombres, titulo, tipo_contrato
FROM Docente
ORDER BY tipo_contrato, apellidos;
```

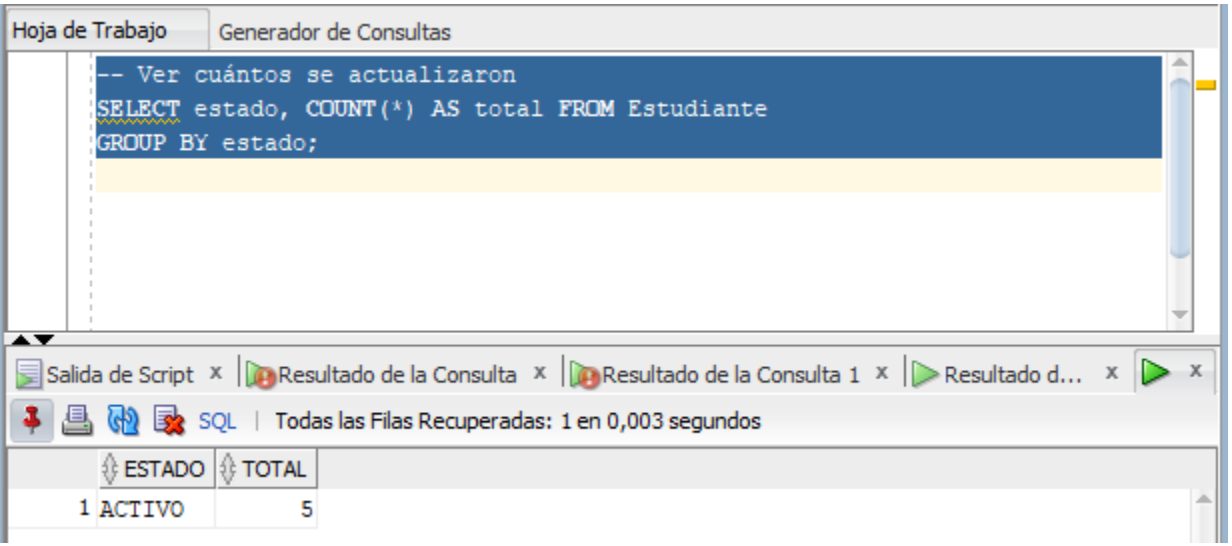


Ejemplo 6: UPDATE Masivo con Condiciones

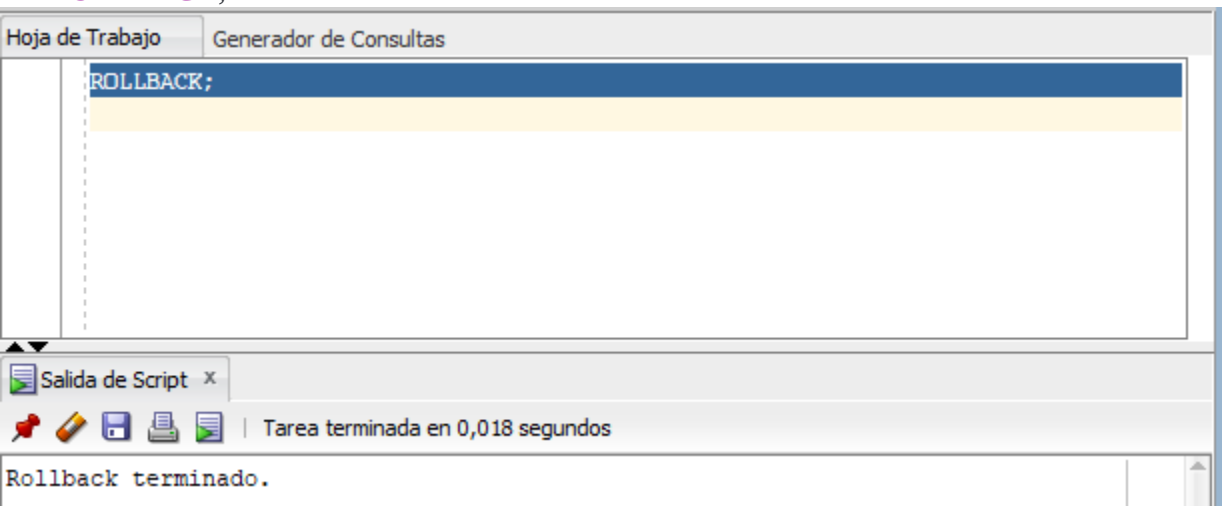
```sql
-- EJEMPLO 6: Actualización masiva con WHERE complejo
-- ==================================================

-- Cambiar estado de estudiantes sin matrículas activas
UPDATE Estudiante
SET estado = 'INACTIVO'
WHERE cedula NOT IN (
    SELECT DISTINCT cedula_estudiante
    FROM Matricula
    WHERE estado = 'CURSANDO'
)
AND estado = 'ACTIVO';
```




```sql
-- Ver cuántos se actualizaron
SELECT estado, COUNT(*) AS total
FROM Estudiante
GROUP BY estado;
```




```
-- Revertir cambio para práctica
ROLLBACK;
```

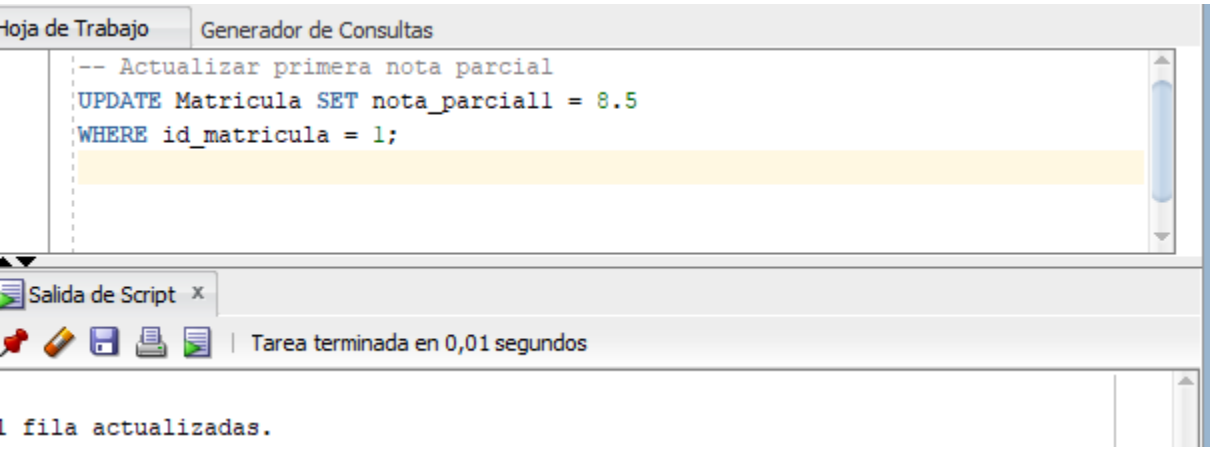


Ejemplo 7: Actualizar Notas en MATRICULA

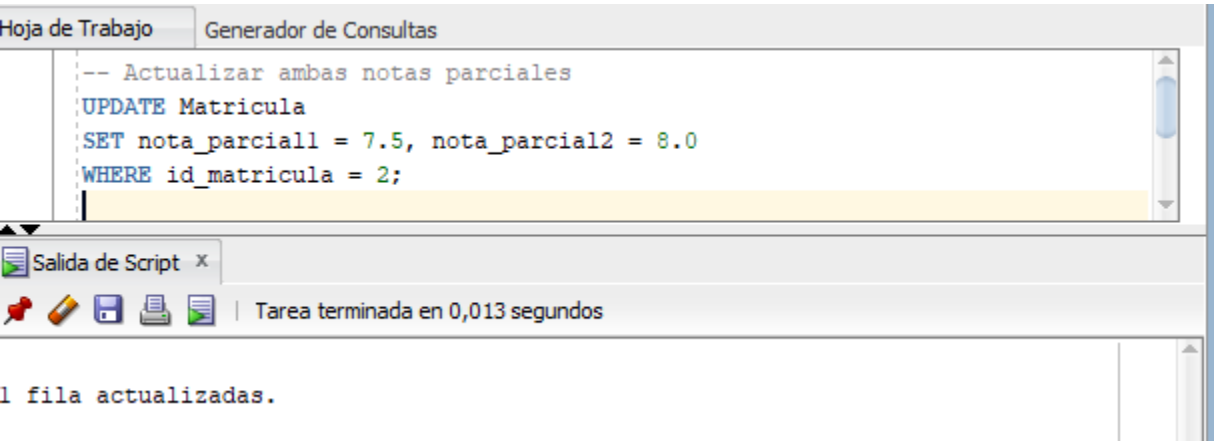
```
-- EJEMPLO 7: Actualizar calificaciones
```

```
-- ============================================================
```

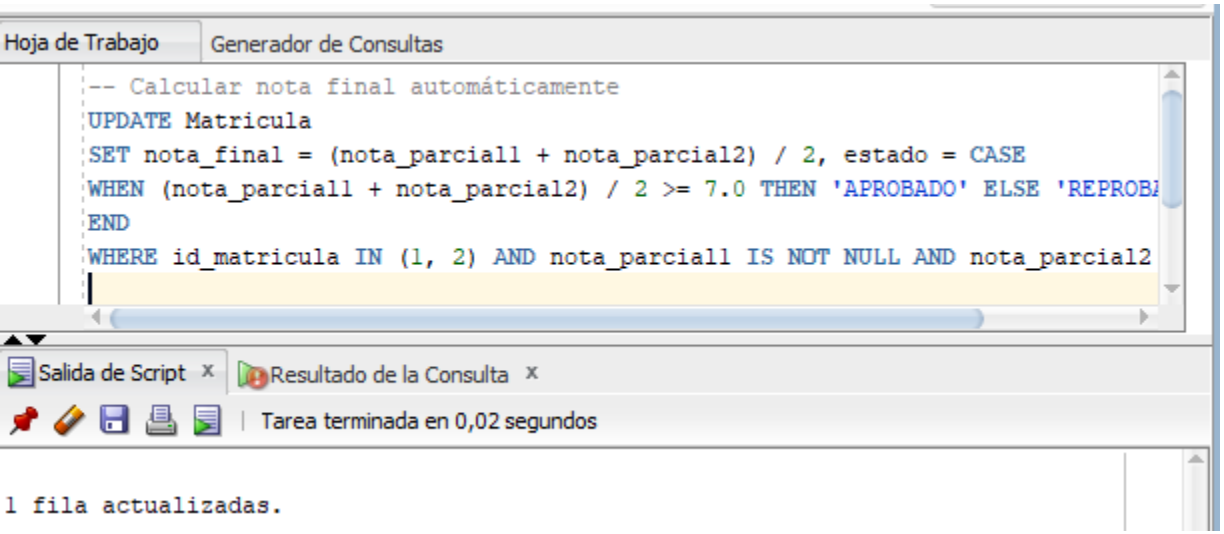
```
-- Actualizar primera nota parcial
UPDATE Matricula
SET nota_parcial1 = 8.5
WHERE id_matricula = 1;
```




```
-- Actualizar ambas notas parciales
UPDATE Matricula
SET nota_parcial1 = 7.5,
    nota_parcial2 = 8.0
WHERE id_matricula = 2;
```

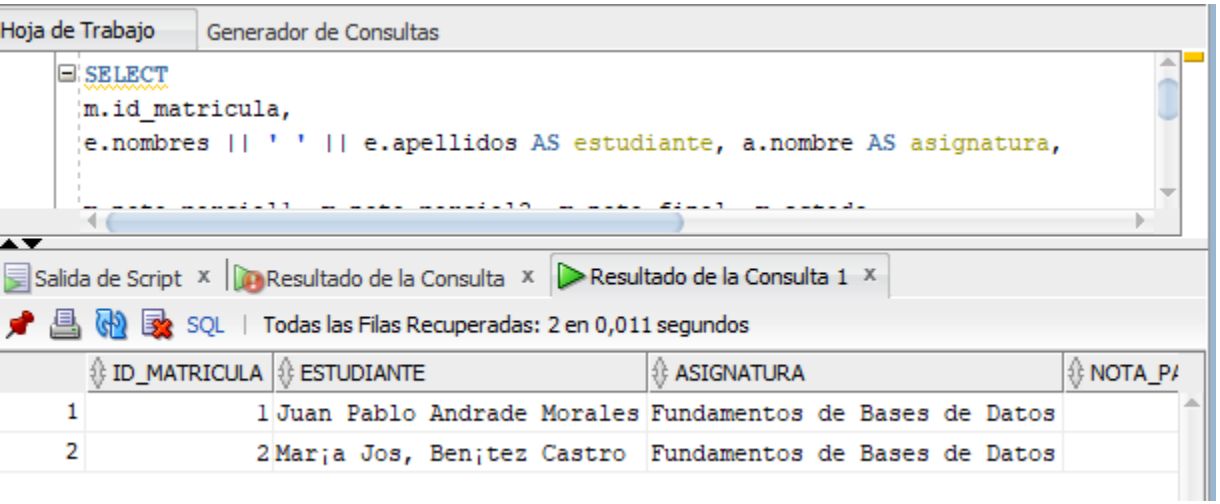



```sql
-- Calcular nota final automáticamente
UPDATE Matricula
SET nota_final = (nota_parcial1 + nota_parcial2) / 2,
    estado = CASE
        WHEN (nota_parcial1 + nota_parcial2) / 2 >= 7.0 THEN 'APROBADO'
        ELSE 'REPROBADO'
    END
WHERE id_matricula IN (1, 2)
AND nota_parcial1 IS NOT NULL
AND nota_parcial2 IS NOT NULL;
```




```sql
-- Verificar calificaciones
SELECT
    m.id_matricula,
    e.nombres || ' ' || e.apellidos AS estudiante,
    a.nombre AS asignatura,
```

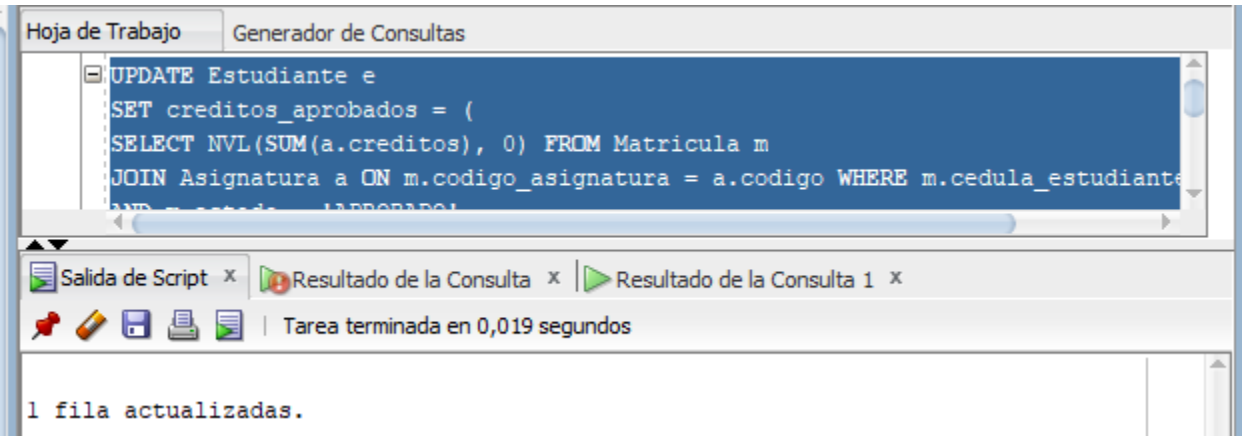
```
    m.nota_parcial1,
    m.nota_parcial2,
    m.nota_final,
    m.estado
FROM Matricula m
JOIN Estudiante e ON m.cedula_estudiante = e.cedula
JOIN Asignatura a ON m.codigo_asignatura = a.codigo
WHERE m.id_matricula IN (1, 2);
```



Ejemplo 8: UPDATE Basado en Otra Tabla

```
-- EJEMPLO 8: Actualizar usando datos de otra tabla
-- ================================================

-- Actualizar créditos aprobados del estudiante sumando
-- créditos de asignaturas aprobadas
UPDATE Estudiante e
SET creditos_aprobados = (
    SELECT NVL(SUM(a.creditos), 0)
```

```
    FROM Matricula m
    JOIN Asignatura a ON m.codigo_asignatura = a.codigo
    WHERE m.cedula_estudiante = e.cedula
    AND m.estado = 'APROBADO'
)
WHERE e.cedula IN (
    SELECT DISTINCT cedula_estudiante
    FROM Matricula
    WHERE estado = 'APROBADO'
);
```




```
-- Verificar actualización
SELECT
    e.cedula,
    e.nombres,
    e.creditos_aprobados,
    COUNT(m.id_matricula) AS materias_aprobadas
FROM Estudiante e
LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante AND m.estado = 'APROBADO'
GROUP BY e.cedula, e.nombres, e.creditos_aprobados
ORDER BY e.apellidos;
```

Hoja de Trabajo | Generador de Consultas

```
    e.nombres,
    e.apellidos,
    e.creditos_aprobados,
    COUNT(m.id_matricula) AS materias_aprobadas
FROM Estudiante e
LEFT JOIN Matricula m
    ON e.cedula = m.cedula_estudiante
    AND m.estado = 'APROBADO'
GROUP BY
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 5 en 0,012 segundos

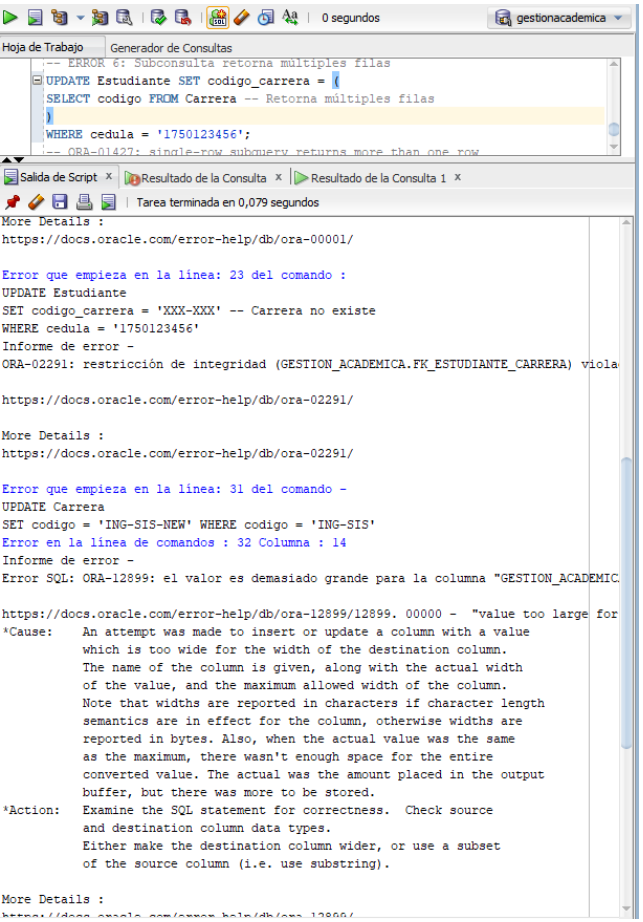
	CEDULA	NOMBRES	APELLIDOS	CREDITOS_APROBADOS	MATERIAS_APROBAI
1	1750123456	Juan Pablo	Andrade Morales	28	
2	1750234567	Mar¡a Jos,	Ben¡tez Castro	4	
3	1750345678	Carlos Andr,s	Castillo D¡az	0	
4	1750456789	Ana Sof¡a	Dom¡nguez Escobar	0	
5	1750567890	Pedro Luis	Espinoza Flores	5	

Errores Comunes en UPDATE

```
-- ERRORES COMUNES EN UPDATE
-- ========================================================


-- ERROR 1: Olvidar WHERE (actualiza TODAS las filas)
-- ✗ PELIGROSO
UPDATE Estudiante
SET estado = 'INACTIVO';
-- Actualiza TODOS los estudiantes


-- ✓ CORRECTO
UPDATE Estudiante
```

```
SET estado = 'INACTIVO'
WHERE cedula = '1750123456';

-- ERROR 2: Violación de restricción CHECK
UPDATE Estudiante
SET genero = 'X'
WHERE cedula = '1750123456';
-- ORA-02290: check constraint violated

-- ERROR 3: Violación de UNIQUE
UPDATE Estudiante
SET email = 'maria.benitez@epn.edu.ec'  -- Email ya existe
WHERE cedula = '1750123456';
-- ORA-00001: unique constraint violated

-- ERROR 4: Violación de FOREIGN KEY
UPDATE Estudiante
SET codigo_carrera = 'XXX-XXX'  -- Carrera no existe
WHERE cedula = '1750123456';

-- ORA-02291: integrity constraint violated

-- ERROR 5: UPDATE de PRIMARY KEY referenciada
-- Si hay FKs que apuntan a esta PK, puede fallar
UPDATE Carrera
SET codigo = 'ING-SIS-NEW'
WHERE codigo = 'ING-SIS';
-- ORA-02292: integrity constraint violated - child record found
-- (si hay estudiantes o asignaturas con ING-SIS)
```

```sql
-- ERROR 6: Subconsulta retorna múltiples filas
UPDATE Estudiante
SET codigo_carrera = (
    SELECT codigo FROM Carrera  -- Retorna múltiples filas
)
WHERE cedula = '1750123456';
-- ORA-01427: single-row subquery returns more than one row
```



Buenas Prácticas UPDATE


```
-- BUENAS PRÁCTICAS UPDATE
-- ================================================

-- ✓ 1. Siempre hacer SELECT antes de UPDATE
-- Verificar qué filas se van a afectar
SELECT *
FROM Estudiante
WHERE cedula = '1750123456';

-- Luego hacer el UPDATE
UPDATE Estudiante
SET estado = 'GRADUADO'
WHERE cedula = '1750123456';

-- ✓ 2. Usar SAVEPOINT antes de UPDATE masivo
SAVEPOINT antes_update_masivo;

UPDATE Estudiante
SET estado = 'INACTIVO'
WHERE creditos_aprobados >= 180;

-- Verificar resultados
SELECT estado, COUNT(*) FROM Estudiante GROUP BY estado;

-- Si no es lo esperado, revertir
ROLLBACK TO antes_update_masivo;

-- ✓ 3. Usar transacciones para UPDATE relacionados
BEGIN
```

```
    UPDATE Matricula
    SET estado = 'APROBADO', nota_final = 8.5
    WHERE id_matricula = 1;

    UPDATE Estudiante
    SET creditos_aprobados = creditos_aprobados + 4
    WHERE cedula = (SELECT cedula_estudiante FROM Matricula WHERE id_matricula = 1);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

-- ✅ 4. Verificar número de filas afectadas
UPDATE Estudiante
SET telefono = '0999999999'
WHERE cedula = '1750123456';

-- En SQL*Plus ver mensaje: "1 row updated"
-- En PL/SQL usar SQL%ROWCOUNT

-- ✅ 5. Usar UPDATE con RETURNING para obtener valores actualizados
DECLARE
    v_creditos NUMBER;
BEGIN
    UPDATE Estudiante
    SET creditos_aprobados = creditos_aprobados + 5
    WHERE cedula = '1750123456'
```

```
    RETURNING creditos_aprobados INTO v_creditos;

    DBMS_OUTPUT.PUT_LINE('Nuevos créditos: ' || v_creditos);
END;
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

DELETE - ELIMINACIÓN DE DATOS

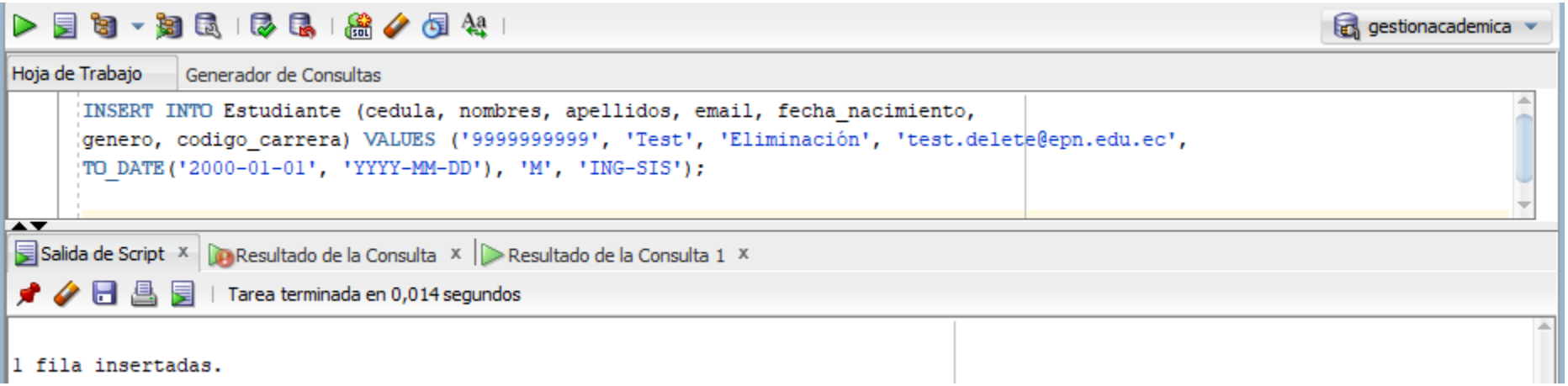
Sintaxis Básica

```
-- Formato básico
DELETE FROM nombre_tabla
WHERE condicion;

-- DELETE con subconsulta
DELETE FROM nombre_tabla
WHERE columna IN (SELECT ... FROM ... WHERE ...);

-- DELETE con JOIN (limitado en Oracle)
DELETE FROM tabla1
WHERE EXISTS (SELECT 1 FROM tabla2 WHERE tabla1.id = tabla2.id);
```

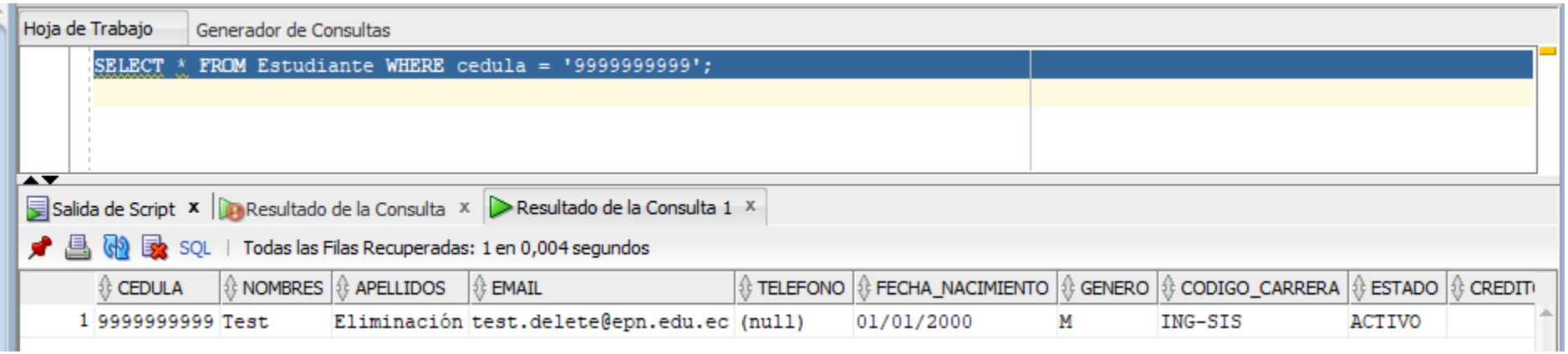
⚠ **ADVERTENCIA CRÍTICA**: Si omite WHERE, se eliminarán TODAS las filas.

Ejemplo 1: DELETE Simple

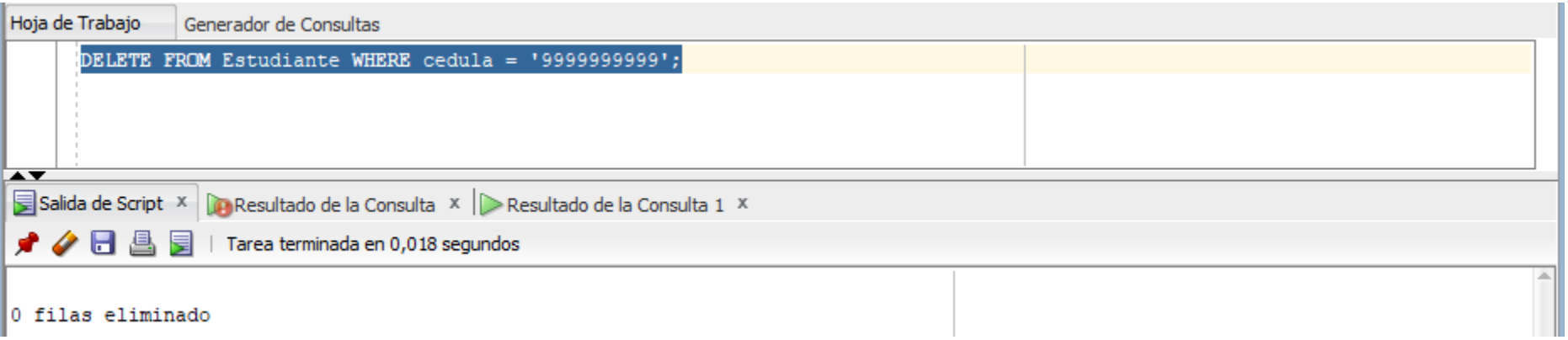
```
-- ========================================================
-- EJEMPLO 1: Eliminación simple con WHERE
-- ========================================================

-- Primero insertar un registro de prueba
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('9999999999', 'Test', 'Eliminación', 'test.delete@epn.edu.ec',
        TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');
```



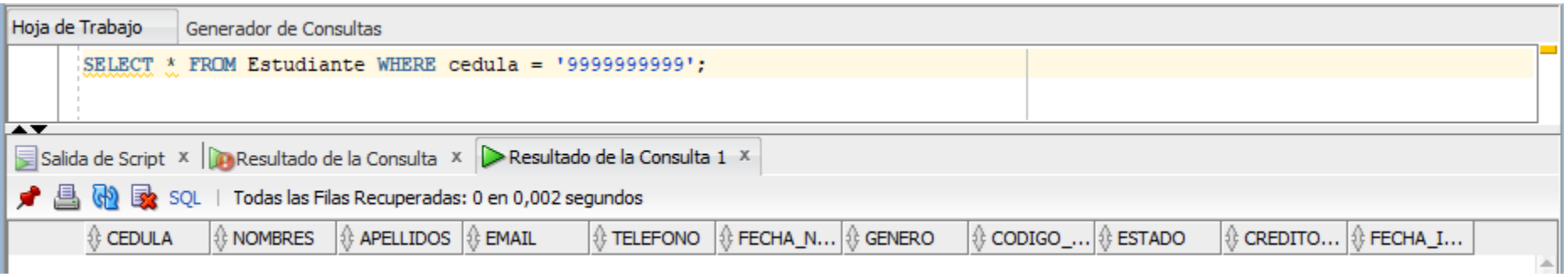

```sql
-- Verificar que existe
SELECT * FROM Estudiante WHERE cedula = '9999999999';
```




```sql
-- Eliminar el registro
DELETE FROM Estudiante
WHERE cedula = '9999999999';
```




```sql
-- Verificar que se eliminó
SELECT * FROM Estudiante WHERE cedula = '9999999999';
-- No retorna ninguna fila


-- Ver cuántas filas se eliminaron: "1 row deleted"
```

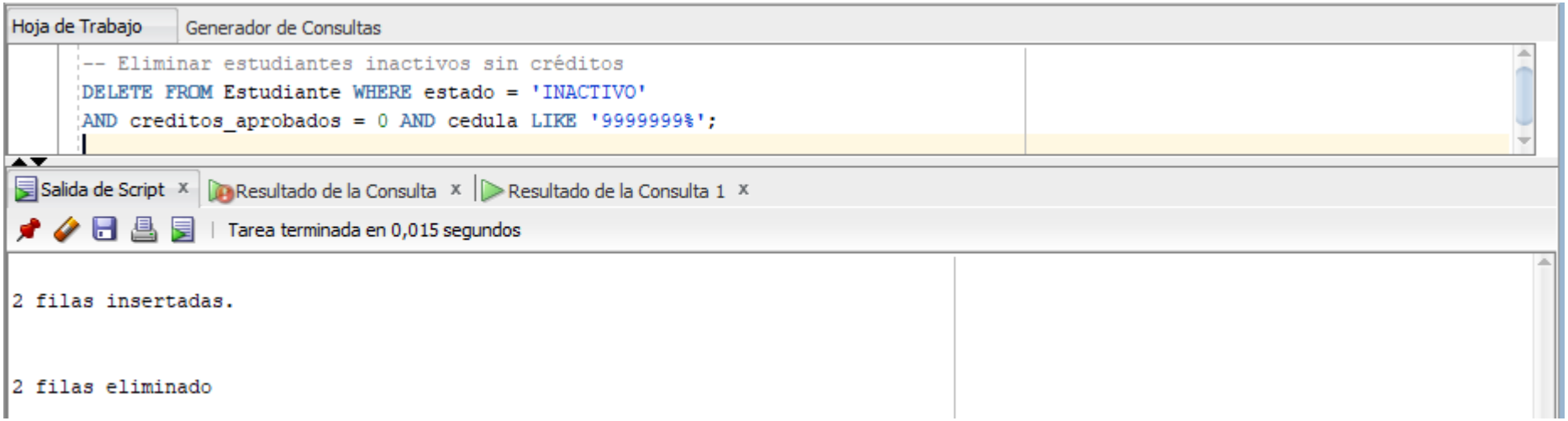


Ejemplo 2: DELETE con Múltiples Condiciones

```sql
-- EJEMPLO 2: DELETE con condiciones múltiples
-- ================================================


-- Insertar datos de prueba  (RECVIOSAR DICE QUE FALTA VALORES
INSERT ALL
```

```
    INTO Estudiante VALUES ('9999999991', 'Test1', 'Delete1', 'test1@test.com', '0980567778',
    TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'INACTIVO', 0, SYSDATE)
     INTO Estudiante VALUES ('9999999992', 'Test2', 'Delete2', 'test2@test.com', '0980765433',
    TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'INACTIVO', 0, SYSDATE)
SELECT * FROM DUAL;
```

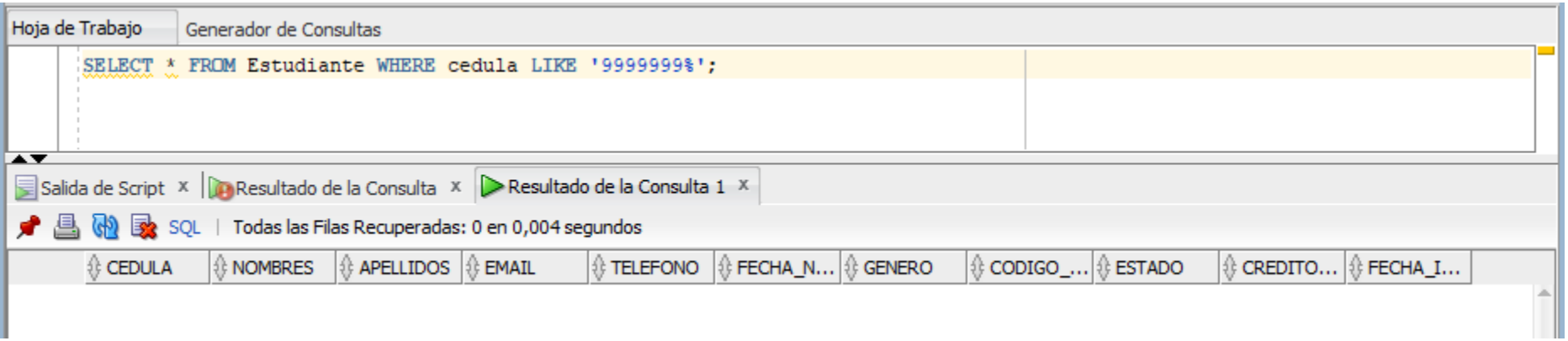

```
-- Eliminar estudiantes inactivos sin créditos
DELETE FROM Estudiante
WHERE estado = 'INACTIVO'
AND creditos_aprobados = 0
AND cedula LIKE '9999999%';
```



-- Ver cuántos se eliminaron: "2 rows deleted"

-- Verificar

```
SELECT * FROM Estudiante WHERE cedula LIKE '9999999%';
```



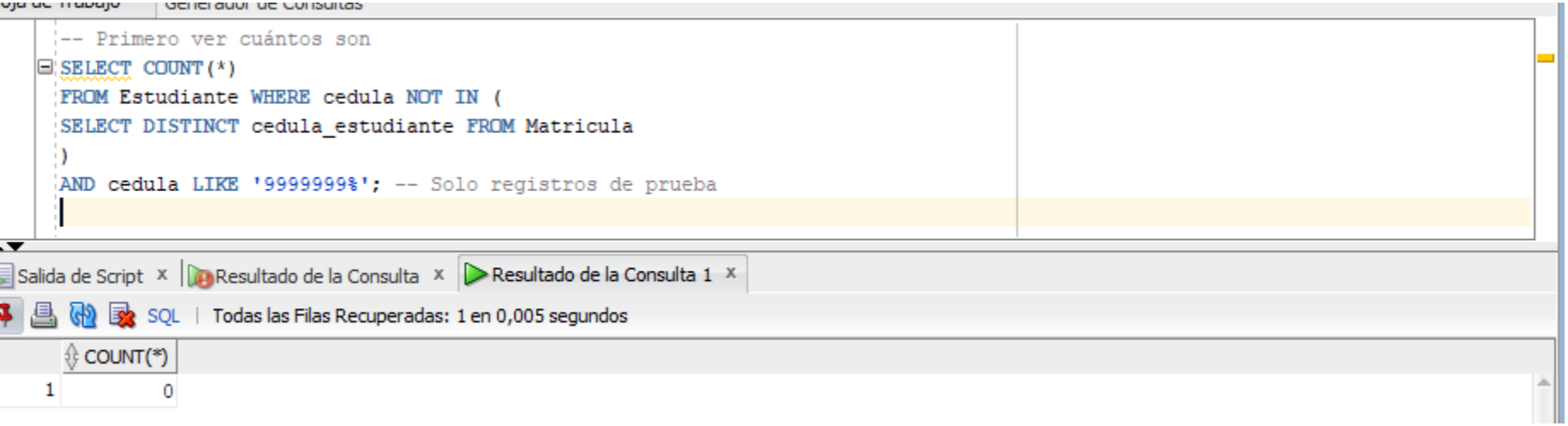
Ejemplo 3: DELETE con Subconsulta

-- ==

-- EJEMPLO 3: DELETE usando subconsulta

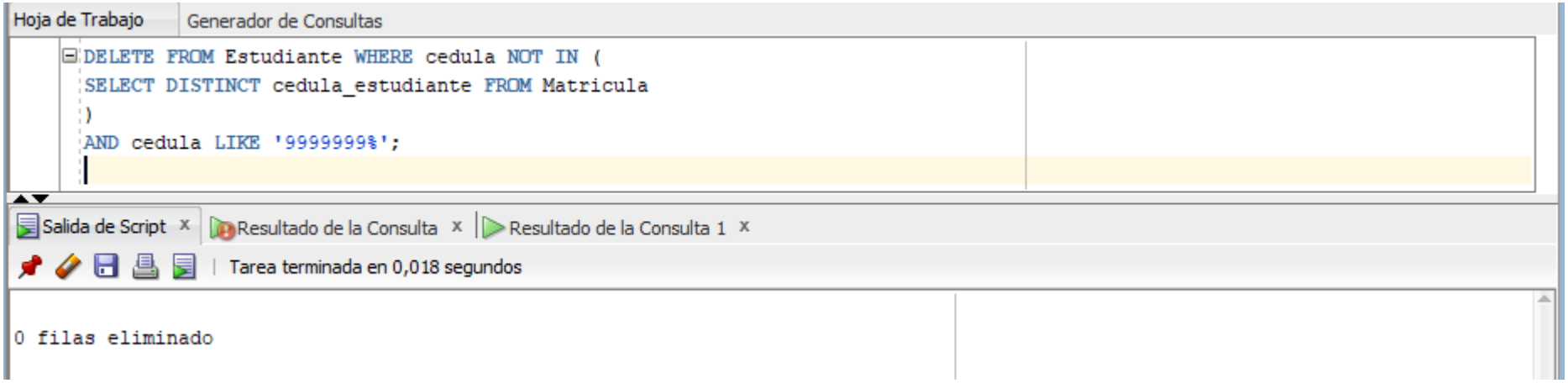
```
-- ================================================
```

```
-- Eliminar estudiantes que no tienen ninguna matrícula
-- Primero ver cuántos son
SELECT COUNT(*)
FROM Estudiante
WHERE cedula NOT IN (
    SELECT DISTINCT cedula_estudiante
    FROM Matricula
)
AND cedula LIKE '9999999%';  -- Solo registros de prueba
```




```
-- Eliminar
DELETE FROM Estudiante
WHERE cedula NOT IN (
    SELECT DISTINCT cedula_estudiante
    FROM Matricula
)
```

```
AND cedula LIKE '9999999%';
```




```
-- Verificar eliminación
```

Ejemplo 4: DELETE Respetando Integridad Referencial

```
-- EJEMPLO 4: Manejo de Foreign Keys en DELETE
-- ================================================

-- Intentar eliminar una carrera que tiene estudiantes (debe fallar)
DELETE FROM Carrera
WHERE codigo = 'ING-SIS';
-- ORA-02292: integrity constraint violated - child record found
-- No se puede eliminar porque tiene estudiantes

-- Intentar eliminar un estudiante que tiene matrículas
-- Si FK tiene ON DELETE CASCADE, se eliminan las matrículas también
-- Si FK tiene ON DELETE RESTRICT, se rechaza la operación
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Hoja de Trabajo | Generador de Consultas

```
DELETE FROM Carrera WHERE codigo = 'ING-SIS';S
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

Tarea terminada en 0,022 segundos

```
0 filas eliminado


Error que empieza en la línea: 1 del comando :
DELETE FROM Carrera WHERE codigo = 'ING-SIS'
Informe de error -
ORA-02292: restricción de integridad (GESTION_ACADEMICA.FK_ESTUDIANTE_CARRERA) violada - registro secundario encontrado

https://docs.oracle.com/error-help/db/ora-02292/

More Details :
https://docs.oracle.com/error-help/db/ora-02292/
```

-- Ver estructura de la FK

```
SELECT constraint_name, delete_rule
FROM user_constraints
WHERE constraint_type = 'R'
AND table_name = 'MATRICULA';
```

Hoja de Trabajo | Generador de Consultas

```
-- Ver estructura de la FK
SELECT constraint_name, delete_rule FROM user_constraints
WHERE constraint_type = 'R'
AND table_name = 'MATRICULA';
```

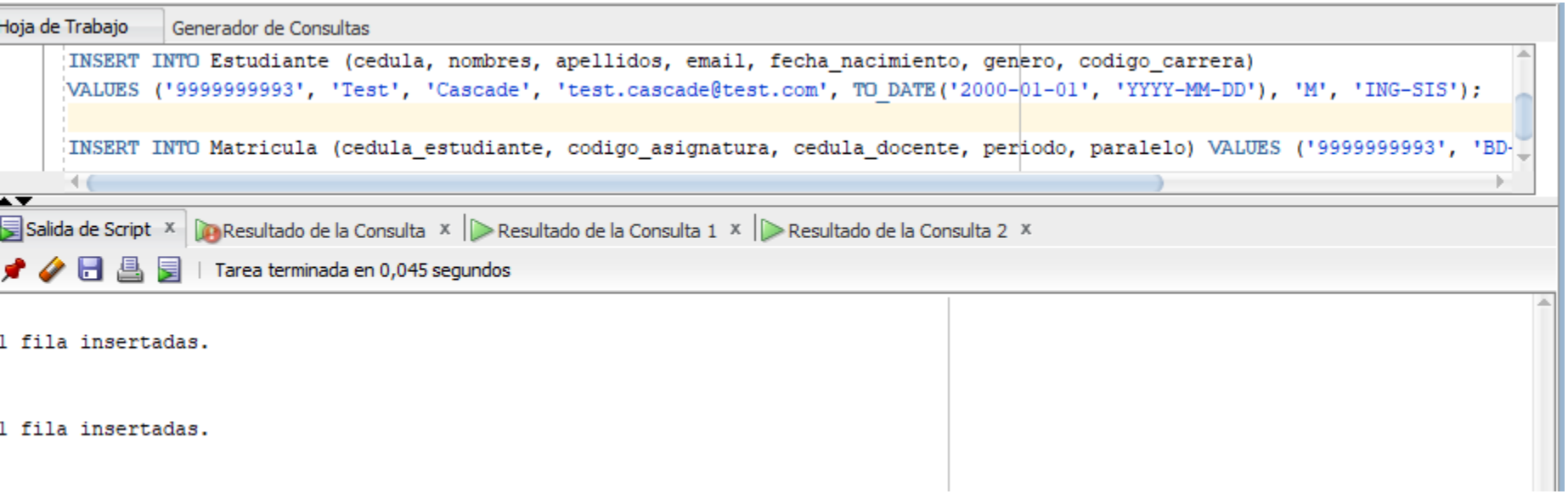
Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2

SQL | Todas las Filas Recuperadas: 3 en 0,194 segundos

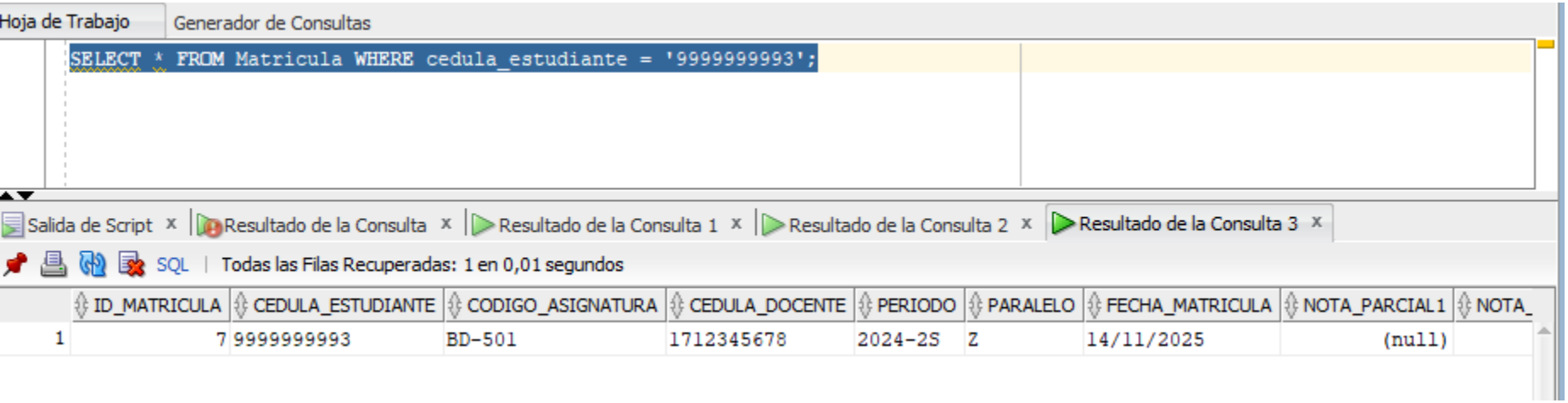
	CONSTRAINT_NAME	DELETE_RULE
1	FK_MATRICULA_ESTUDIANTE	CASCADE
2	FK_MATRICULA_ASIGNATURA	NO ACTION
3	FK_MATRICULA_DOCENTE	SET NULL

-- Ejemplo de eliminación en cascada

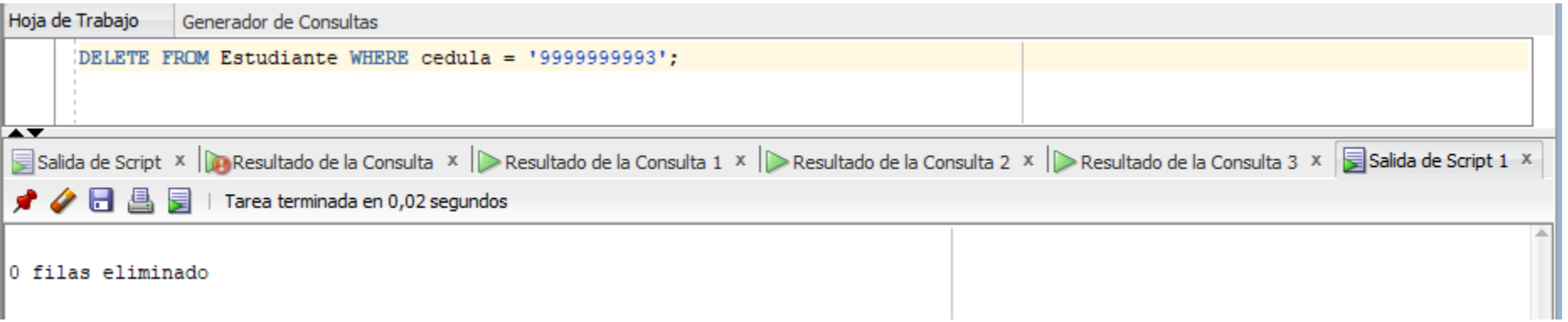
```
-- Eliminar estudiante (sus matrículas se eliminan automáticamente)
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('9999999993', 'Test', 'Cascade', 'test.cascade@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');

INSERT INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
VALUES ('9999999993', 'BD-501', '1712345678', '2024-2S', 'Z');
```



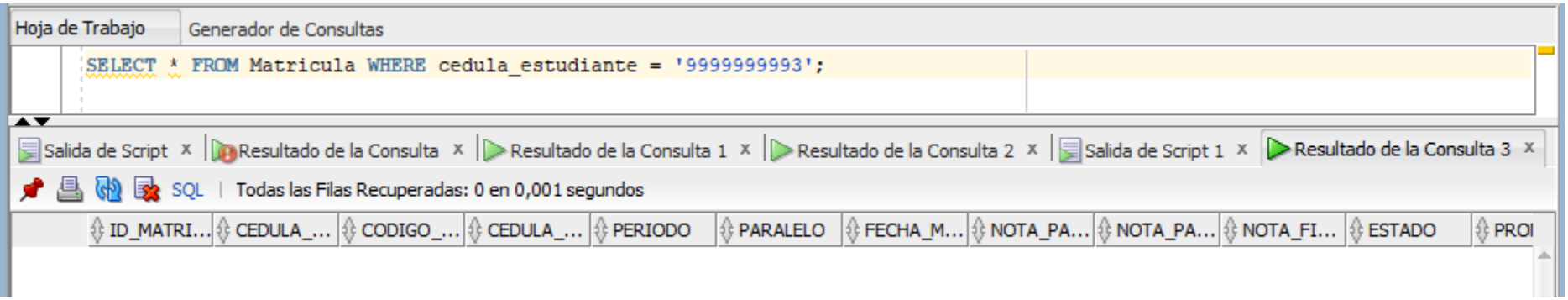

```
-- Ver matrícula creada
SELECT * FROM Matricula WHERE cedula_estudiante = '9999999993';
```




```sql
-- Eliminar estudiante (CASCADE elimina matrículas)
DELETE FROM Estudiante WHERE cedula = '9999999993';
```




```sql
-- Verificar que la matrícula también se eliminó
SELECT * FROM Matricula WHERE cedula_estudiante = '9999999993';
-- No retorna nada
```



Ejemplo 5: DELETE con EXISTS

```
-- EJEMPLO 5: DELETE usando EXISTS
-- ================================================

-- Eliminar estudiantes que tienen matrículas reprobadas
DELETE FROM Estudiante
WHERE EXISTS (
    SELECT 1
    FROM Matricula
    WHERE Matricula.cedula_estudiante = Estudiante.cedula
    AND Matricula.estado = 'REPROBADO'
)
AND cedula LIKE '9999999%';  -- Solo prueba
```

Hoja de Trabajo | Generador de Consultas

```
DELETE FROM Estudiante WHERE EXISTS (
SELECT 1
FROM Matricula
WHERE Matricula.cedula_estudiante = Estudiante.cedula AND Matricula.estado = 'REPROBADO'
)
AND cedula LIKE '9999999%'; -- Solo prueba
```

Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1 | Resultado de la Consulta 2 | Salida de Script 1 | Resultado de la Consulta 3

Tarea terminada en 0,028 segundos

```
0 filas eliminado
```

```
-- Verificar
SELECT COUNT(*)
FROM Estudiante
WHERE cedula LIKE '9999999%';
```

Hoja de Trabajo | Generador de Consultas

```
-- Verificar
SELECT COUNT(*)
FROM Estudiante
WHERE cedula LIKE '9999999%';
```

Salida de Script | Resultado de la Consulta

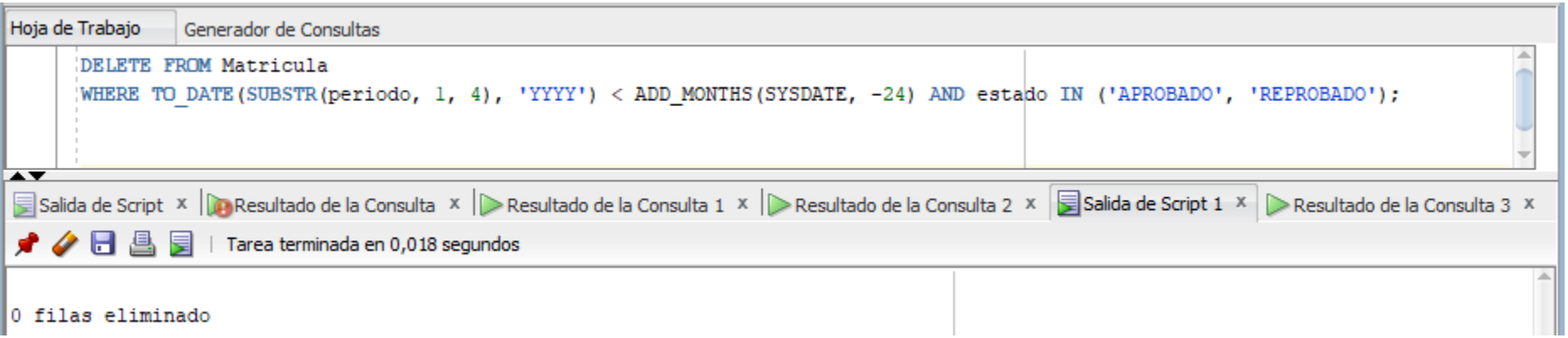
SQL | Todas las Filas Recuperadas:

	COUNT(*)
1	0

Ejemplo 6: DELETE Condicional por Fecha

```
-- ============================================================
```

```
-- EJEMPLO 6: Eliminar registros antiguos
-- ================================================

-- Eliminar matrículas de hace más de 2 años
DELETE FROM Matricula
WHERE TO_DATE(SUBSTR(periodo, 1, 4), 'YYYY') < ADD_MONTHS(SYSDATE, -24)
AND estado IN ('APROBADO', 'REPROBADO');
```




```
-- Eliminar registros de auditoría viejos (si existieran)
-- DELETE FROM Auditoria
-- WHERE fecha_registro < ADD_MONTHS(SYSDATE, -12);
```

Ejemplo 7: DELETE Seguro con Savepoint

```
-- EJEMPLO 7: DELETE con punto de salvaguarda
-- ================================================

-- Crear savepoint antes de eliminar
SAVEPOINT antes_delete;
```

```
-- Contar registros antes
SELECT COUNT(*) AS total_antes FROM Estudiante;

-- Realizar DELETE
DELETE FROM Estudiante
WHERE estado = 'RETIRADO'
AND creditos_aprobados = 0;

-- Ver cuántos se eliminaron
SELECT COUNT(*) AS total_despues FROM Estudiante;

-- Si fue un error, revertir
ROLLBACK TO antes_delete;

-- Verificar que se restauraron
SELECT COUNT(*) AS total_restaurado FROM Estudiante;

-- Si está bien, confirmar
-- COMMIT;
```

Ejemplo 8: DELETE vs TRUNCATE

```
-- ============================================================
-- EJEMPLO 8: Comparación DELETE vs TRUNCATE
-- ============================================================
```

```
-- DELETE: Transaccional, se puede revertir, más lento
DELETE FROM Carrera WHERE codigo LIKE 'TEST%';
ROLLBACK;  -- Se puede deshacer

-- TRUNCATE: No transaccional, no se puede revertir, muy rápido
-- TRUNCATE TABLE Carrera;  -- Elimina TODAS las filas, NO se puede revertir
-- No se puede usar WHERE con TRUNCATE

-- Comparación DELETE vs TRUNCATE:

-- | Característica    | DELETE          | TRUNCATE
-- |---------------  ----|---  -----------  ---|--------------
-- | WHERE clause   ✓ Sí           | ✗ No          |
-- | ROLLBACK      |✓ Sí           | ✗ No          |
-- | Triggers          | ✓ Se disparan  | ✗ No se disparan
-- | Velocidad         | ⚠ Lenta         | ✓ Muy rápida
-- | UNDO              |  ✓ Genera       | ✗ No genera
-- | Foreign Keys     |  ✓ Respeta      | ⚠ Requiere disable
```

Errores Comunes en DELETE

```
-- ERRORES COMUNES EN DELETE
-- ================================================

-- ERROR 1: Olvidar WHERE (elimina TODO)
-- ✗ PELIGROSO
```

```
DELETE FROM Estudiante;
-- Elimina TODOS los estudiantes
ROLLBACK;  -- Deshacer

-- ✓ CORRECTO
DELETE FROM Estudiante
WHERE cedula = '9999999999';

-- ERROR 2: Violación de integridad referencial
DELETE FROM Carrera WHERE codigo = 'ING-SIS';
-- ORA-02292: integrity constraint violated - child record found

-- ERROR 3: DELETE sin verificar
-- Siempre hacer SELECT primero
SELECT * FROM Estudiante WHERE estado = 'RETIRADO';
-- Luego DELETE
DELETE FROM Estudiante WHERE estado = 'RETIRADO';

-- ERROR 4: No usar transacciones en DELETE masivo
-- ✓ USAR SAVEPOINT
SAVEPOINT antes_delete_masivo;
DELETE FROM Matricula WHERE periodo = '2020-1S';
```

```
-- Si es error: ROLLBACK TO antes_delete_masivo;
-- Si es OK: COMMIT;
```

Buenas Prácticas DELETE

```
-- BUENAS PRÁCTICAS DELETE
-- ========================================================

-- ✓ 1. Siempre hacer SELECT antes de DELETE
SELECT * FROM Estudiante WHERE cedula = '9999999999';
-- Verificar que es el registro correcto
DELETE FROM Estudiante WHERE cedula = '9999999999';

-- ✓ 2. Usar COUNT para verificar cuántos se van a eliminar
SELECT COUNT(*) FROM Estudiante WHERE estado = 'RETIRADO';
-- Si el número es correcto, proceder
DELETE FROM Estudiante WHERE estado = 'RETIRADO';

-- ✓ 3. Usar transacciones para DELETE relacionados
BEGIN
   DELETE FROM Matricula WHERE cedula_estudiante = '9999999999';
   DELETE FROM Estudiante WHERE cedula = '9999999999';
   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
```

```
    RAISE;
END;
/

-- ✓ 4. Considerar DELETE lógico en lugar de físico
-- En lugar de eliminar, marcar como inactivo
UPDATE Estudiante
SET estado = 'ELIMINADO', fecha_eliminacion = SYSDATE
WHERE cedula = '9999999999';

-- ✓ 5. Respaldar antes de DELETE masivo
CREATE TABLE Estudiante_Backup AS
SELECT * FROM Estudiante WHERE estado = 'RETIRADO';

-- Luego eliminar
DELETE FROM Estudiante WHERE estado = 'RETIRADO';

-- ✓ 6. Documentar la razón del DELETE
-- Usar comentarios o tabla de auditoría
INSERT INTO Auditoria_Eliminaciones (tabla, registro_id, usuario, fecha, motivo)
VALUES ('ESTUDIANTE', '9999999999', USER, SYSDATE, 'Estudiante duplicado');

DELETE FROM Estudiante WHERE cedula = '9999999999';
```

TRANSACCIONES

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Conceptos de Transacciones

Una **transacción** es un conjunto de operaciones DML que se ejecutan como una unidad lógica de trabajo.

Propiedades ACID:

- **A**tomicity (Atomicidad): Todo o nada
- **C**onsistency (Consistencia): De un estado válido a otro válido
- **I**solation (Aislamiento): Las transacciones no se interfieren
- **D**urability (Durabilidad): Los cambios confirmados persisten

Comandos de Control de Transacciones

```
COMMIT;          -- Confirmar cambios permanentemente
ROLLBACK;        -- Deshacer todos los cambios desde último COMMIT
SAVEPOINT nombre; -- Crear punto de guardado
ROLLBACK TO SAVEPOINT nombre;  -- Volver a un savepoint
SET TRANSACTION;  -- Configurar propiedades de transacción
```

Ejemplo 1: COMMIT Básico

```
-- EJEMPLO 1: Uso básico de COMMIT
-- ================================================

-- Los cambios no son permanentes hasta hacer COMMIT
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('8888888888', 'Test', 'Commit', 'test.commit@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');

-- El registro existe en esta sesión
SELECT * FROM Estudiante WHERE cedula = '8888888888';
```

```
-- Retorna 1 fila

-- Pero otra sesión NO lo ve aún

-- Confirmar cambios
COMMIT;

-- Ahora todas las sesiones pueden ver el registro
-- Los cambios son permanentes
```

Ejemplo 2: ROLLBACK Básico

```
-- ============================================================
-- EJEMPLO 2: Uso básico de ROLLBACK
-- ============================================================

-- Insertar registro
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('7777777777', 'Test', 'Rollback', 'test.rollback@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');

-- Verificar que existe (en esta sesión)
SELECT * FROM Estudiante WHERE cedula = '7777777777';

-- Deshacer el INSERT
ROLLBACK;

-- Verificar que ya no existe
```

```
SELECT * FROM Estudiante WHERE cedula = '7777777777';
-- No retorna nada

-- Los cambios se deshicieron completamente
```

Ejemplo 3: SAVEPOINT

```
-- EJEMPLO 3: Uso de SAVEPOINT
-- ==================================================

-- Iniciar transacción compleja
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('6666666661', 'Test1', 'Savepoint', 'test1.sp@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');

-- Crear punto de guardado
SAVEPOINT sp1;

INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('6666666662', 'Test2', 'Savepoint', 'test2.sp@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-SIS');

-- Otro savepoint
SAVEPOINT sp2;

INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('6666666663', 'Test3', 'Savepoint', 'test3.sp@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-CIV');

-- Ver los 3 registros
```

```
SELECT * FROM Estudiante WHERE cedula LIKE '666666666%';
-- Retorna 3 filas

-- Volver al savepoint sp2 (deshace el tercer INSERT)
ROLLBACK TO SAVEPOINT sp2;

-- Ahora solo hay 2 registros
SELECT * FROM Estudiante WHERE cedula LIKE '666666666%';
-- Retorna 2 filas
```

```
-- Volver al savepoint sp1 (deshace el segundo INSERT)
ROLLBACK TO SAVEPOINT sp1;

-- Ahora solo hay 1 registro
SELECT * FROM Estudiante WHERE cedula LIKE '666666666%';
-- Retorna 1 fila

-- Confirmar el primer INSERT
COMMIT;

-- Verificar que quedó solo el primero
SELECT * FROM Estudiante WHERE cedula LIKE '666666666%';
```

Ejemplo 4: Transacción Compleja

```
-- EJEMPLO 4: Transacción completa con manejo de errores
-- ==================================================

-- Escenario: Matricular un estudiante en una asignatura
-- Debe ser atómico: o se completa todo o nada

SAVEPOINT antes_matricula;

BEGIN
    -- 1. Insertar estudiante nuevo
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
```

```
    VALUES ('5555555555', 'Nuevo', 'Estudiante', 'nuevo.est@epn.edu.ec', TO_DATE('2003-03-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

    DBMS_OUTPUT.PUT_LINE('Estudiante  insertado');

    -- 2. Matricular en asignatura
    INSERT INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
    VALUES ('5555555555', 'BD-501', '1712345678', '2024-2S', 'A');

    DBMS_OUTPUT.PUT_LINE('Matrícula  insertada');

    -- 3. Confirmar todo
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Transacción  confirmada');

EXCEPTION
    WHEN OTHERS THEN
        -- Si algo falla, deshacer todo
        ROLLBACK TO antes_matricula;
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
```

```
    DBMS_OUTPUT.PUT_LINE('Transacción revertida');
    END;
/

-- Verificar resultado
SELECT * FROM Estudiante WHERE cedula = '5555555555';
SELECT * FROM Matricula WHERE cedula_estudiante = '5555555555';
```

Ejemplo 5: Bloqueos (Locks)

```
-- EJEMPLO 5: Entender los bloqueos de transacciones
-- ==================================================

-- SESIÓN 1:
UPDATE Estudiante
SET telefono = '0999999999'
WHERE cedula = '1750123456';
-- NO hacer COMMIT todavía

-- El registro queda bloqueado para otras sesiones

-- SESIÓN 2 (en otra ventana):
UPDATE Estudiante
SET email = 'nuevo@epn.edu.ec'
WHERE cedula = '1750123456';
-- Esta sesión ESPERA hasta que SESIÓN 1 haga COMMIT o ROLLBACK
```

```
-- SESIÓN 1:
COMMIT;  -- o ROLLBACK
-- Ahora SESIÓN 2 puede continuar

/*
TIPOS DE BLOQUEOS:
- Row-level locks: Bloquean solo las filas afectadas
- Table-level locks: Bloquean la tabla completa
- Share locks: Permiten lecturas pero no escrituras
- Exclusive locks: No permiten ni lecturas ni escrituras
*/
```

Ejemplo 6: SET TRANSACTION

```
-- ============================================================
-- EJEMPLO 6: Configurar propiedades de transacción
-- ============================================================

-- Transacción de solo lectura
SET TRANSACTION READ ONLY;

SELECT * FROM Estudiante;
SELECT * FROM Matricula;

-- No se permite INSERT, UPDATE, DELETE
-- INSERT INTO Estudiante VALUES (...);  -- ERROR
```

```
COMMIT;

-- Transacción con aislamiento específico
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

-- Operaciones DML
UPDATE Estudiante SET telefono = '0999999999' WHERE cedula = '1750123456';

COMMIT;
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Mejores Prácticas de Transacciones

```
-- ================================================
-- MEJORES PRÁCTICAS DE TRANSACCIONES
-- ================================================

-- ✓ 1. Mantener transacciones cortas
-- Evitar transacciones de larga duración que bloquean recursos
BEGIN
    -- Operaciones rápidas
    INSERT INTO ... VALUES (...);
    UPDATE ... SET ... WHERE ...;
    COMMIT;  -- Confirmar rápido
END;
/

-- ✓ 2. Usar SAVEPOINT para operaciones complejas
SAVEPOINT inicio;
-- Operación 1
SAVEPOINT despues_op1;
-- Operación 2
SAVEPOINT despues_op2;
-- Si falla, volver a savepoint específico

-- ✓ 3. Siempre manejar excepciones
BEGIN
    -- Operaciones DML
    COMMIT;
```

```
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;
      -- Registrar error
      RAISE;
END;
/

-- ✓ 4. Evitar COMMIT en bucles
-- ✗ MAL
FOR i IN 1..1000 LOOP
   INSERT INTO ... VALUES (...);
   COMMIT;  -- Muy lento
END LOOP;

-- ✓ BIEN
FOR i IN 1..1000 LOOP
   INSERT INTO ... VALUES (...);
END LOOP;
COMMIT;  -- Una vez al final

-- ✓ 5. Usar transacciones explícitas
-- Documentar inicio y fin de transacción
-- INICIO TRANSACCIÓN
INSERT INTO ...;
UPDATE ...;
DELETE FROM ...;
COMMIT;
```

```
-- FIN TRANSACCIÓN

-- ✓ 6. No mezclar DDL y DML
-- DDL hace COMMIT automático
CREATE TABLE ...;  -- COMMIT automático
INSERT INTO ...;   -- Nueva transacción
-- Si falla el INSERT, no se puede deshacer el CREATE
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

EJERCICIOS PRÁCTICOS

Ejercicio 1: INSERT Completo

Objetivo: Practicar todas las variantes de INSERT.

Instrucciones:

1. Insertar 3 nuevas carreras de ingeniería
2. Insertar 5 docentes con diferentes tipos de contrato
3. Insertar 10 estudiantes en diferentes carreras
4. Insertar 15 asignaturas con diferentes niveles y carreras
5. Establecer 10 relaciones de prerrequisitos
6. Matricular 20 estudiantes en asignaturas del periodo 2024-2S

```
-- 1. INSERTAR 3 NUEVAS CARRERAS DE INGENIERÍA
INSERT ALL
   INTO Carrera (codigo, nombre, duracion_semestres, creditos_totales, facultad)
   VALUES ('ING-AMB', 'Ingeniería Ambiental', 10, 185, 'Facultad de Ingeniería Ambiental - FIA')

   INTO Carrera (codigo, nombre, duracion_semestres, creditos_totales, facultad)
   VALUES ('ING-BIO', 'Ingeniería en Biotecnología', 10, 190, 'Facultad de Ingeniería en Biotecnología - FIB')

   INTO Carrera (codigo, nombre, duracion_semestres, creditos_totales, facultad)
   VALUES ('ING-TEL', 'Ingeniería en Telecomunicaciones', 10, 182, 'Facultad de Ingeniería en Telecomunicaciones - FIT')
SELECT * FROM DUAL;

-- 2. INSERTAR 5 DOCENTES CON DIFERENTES TIPOS DE CONTRATO
INSERT ALL
   -- Docente 1: Tiempo Completo
   INTO Docente (cedula, nombres, apellidos, email, telefono, titulo, especialidad, tipo_contrato, fecha_ingreso)
   VALUES ('1765432101', 'Ana Patricia', 'Gómez Mendoza', 'ana.gomez@epn.edu.ec', '0981122334',
           'PhD en Ciencias Ambientales', 'Gestión Ambiental', 'TIEMPO_COMPLETO', SYSDATE)

   -- Docente 2: Medio Tiempo
   INTO Docente (cedula, nombres, apellidos, email, titulo, especialidad, tipo_contrato)
```

```
  VALUES ('1765432102', 'Luis Fernando', 'Ortega Silva', 'luis.ortega@epn.edu.ec',
          'MSc en Biotecnología', 'Biotecnología Industrial', 'MEDIO_TIEMPO')

  -- Docente 3: Hora Clase
  INTO Docente (cedula, nombres, apellidos, email, telefono, titulo, especialidad, tipo_contrato, fecha_ingreso)
  VALUES ('1765432103', 'María Elena', 'Ríos Castro', 'maria.rios@epn.edu.ec', '0994455667',
          'Ingeniera en Telecomunicaciones', 'Redes y Comunicaciones', 'HORA_CLASE', SYSDATE)

  -- Docente 4: Tiempo Completo
  INTO Docente (cedula, nombres, apellidos, email, telefono, titulo, especialidad, tipo_contrato, fecha_ingreso)
  VALUES ('1765432104', 'Carlos Andrés', 'Vega López', 'carlos.vega@epn.edu.ec', '0977788990',
          'PhD en Telecomunicaciones', 'Comunicaciones Digitales', 'TIEMPO_COMPLETO', SYSDATE)

  -- Docente 5: Medio Tiempo
  INTO Docente (cedula, nombres, apellidos, email, titulo, especialidad, tipo_contrato)
  VALUES ('1765432105', 'Sofía Alejandra', 'Morales Torres', 'sofia.morales@epn.edu.ec',
          'MSc en Gestión Ambiental', 'Impacto Ambiental', 'MEDIO_TIEMPO')

-- 3. INSERTAR 10 ESTUDIANTES EN DIFERENTES CARRERAS
INSERT ALL
  -- Estudiantes ING-AMB
  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122331', 'Daniel Alejandro', 'Pazmiño Ruiz', 'daniel.pazmino@epn.edu.ec', '0991122334',
          TO_DATE('2002-08-12', 'YYYY-MM-DD'), 'M', 'ING-AMB', 'ACTIVO', 0, SYSDATE)

  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122332', 'Carolina Estefanía', 'Herrera Mendoza', 'carolina.herrera@epn.edu.ec', '0992233445',
          TO_DATE('2003-03-25', 'YYYY-MM-DD'), 'F', 'ING-AMB', 'ACTIVO', 0, SYSDATE)

  -- Estudiantes ING-BIO
  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122333', 'Andrea Nicole', 'Salazar Vargas', 'andrea.salazar@epn.edu.ec', '0993344556',
          TO_DATE('2002-11-08', 'YYYY-MM-DD'), 'F', 'ING-BIO', 'ACTIVO', 0, SYSDATE)

  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122334', 'Roberto Carlos', 'Alvarez Paredes', 'roberto.alvarez@epn.edu.ec', '0994455667',
          TO_DATE('2003-06-18', 'YYYY-MM-DD'), 'M', 'ING-BIO', 'ACTIVO', 0, SYSDATE)
```

```
  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122335', 'Gabriela Alejandra', 'Torres León', 'gabriela.torres@epn.edu.ec', '0995566778',
      TO_DATE('2002-12-30', 'YYYY-MM-DD'), 'F', 'ING-BIO', 'ACTIVO', 0, SYSDATE)

  -- Estudiantes ING-TEL
  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122336', 'Fernando José', 'Ramos Salinas', 'fernando.ramos@epn.edu.ec', '0996677889',
      TO_DATE('2003-04-15', 'YYYY-MM-DD'), 'M', 'ING-TEL', 'ACTIVO', 0, SYSDATE)

  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122337', 'María Fernanda', 'Castro Ortega', 'maria.castro@epn.edu.ec', '0997788990',
      TO_DATE('2002-09-22', 'YYYY-MM-DD'), 'F', 'ING-TEL', 'ACTIVO', 0, SYSDATE)

  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122338', 'David Alexander', 'Mendoza López', 'david.mendoza@epn.edu.ec', '0998899001',
      TO_DATE('2003-01-14', 'YYYY-MM-DD'), 'M', 'ING-TEL', 'ACTIVO', 0, SYSDATE)

  -- Estudiantes en carreras existentes
  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122339', 'Patricia Elizabeth', 'Narváez Soto', 'patricia.narvaez@epn.edu.ec', '0999900112',
      TO_DATE('2002-07-07', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'ACTIVO', 0, SYSDATE)

  INTO Estudiante (cedula, nombres, apellidos, email, telefono, fecha_nacimiento, genero, codigo_carrera, estado, creditos_aprobados, fecha_ingreso)
  VALUES ('1751122340', 'Jorge Luis', 'Silva Reyes', 'jorge.silva@epn.edu.ec', '0990011223',
      TO_DATE('2003-02-28', 'YYYY-MM-DD'), 'M', 'ING-CIV', 'ACTIVO', 0, SYSDATE)
SELECT * FROM DUAL;

-- 4. INSERTAR 15 ASIGNATURAS CON DIFERENTES NIVELES Y CARRERAS
INSERT ALL
  -- Asignaturas ING-AMB
  INTO Asignatura VALUES ('AMB-101', 'Introducción a la Ingeniería Ambiental', 4, 1, 'ING-AMB', 3, 2, 'Fundamentos de ingeniería ambiental')
  INTO Asignatura VALUES ('AMB-201', 'Química Ambiental', 5, 2, 'ING-AMB', 4, 2, 'Química aplicada al medio ambiente')
  INTO Asignatura VALUES ('AMB-301', 'Gestión de Residuos Sólidos', 4, 3, 'ING-AMB', 3, 2, 'Manejo y tratamiento de residuos')
  INTO Asignatura VALUES ('AMB-401', 'Impacto Ambiental', 5, 4, 'ING-AMB', 3, 2, 'Evaluación de impacto ambiental')

  -- Asignaturas ING-BIO
```

```
  INTO Asignatura VALUES ('BIO-101', 'Biología General', 5, 1, 'ING-BIO', 4, 2, 'Fundamentos de biología')
  INTO Asignatura VALUES ('BIO-201', 'Bioquímica', 5, 2, 'ING-BIO', 4, 2, 'Principios de bioquímica')
  INTO Asignatura VALUES ('BIO-301', 'Biotecnología Industrial', 4, 3, 'ING-BIO', 3, 2, 'Aplicaciones industriales')
  INTO Asignatura VALUES ('BIO-401', 'Ingeniería Genética', 5, 4, 'ING-BIO', 3, 2, 'Manipulación genética')

  -- Asignaturas ING-TEL
  INTO Asignatura VALUES ('TEL-101', 'Fundamentos de Telecomunicaciones', 4, 1, 'ING-TEL', 3, 2, 'Introducción a telecomunicaciones')
  INTO Asignatura VALUES ('TEL-201', 'Redes de Comunicación', 5, 2, 'ING-TEL', 4, 2, 'Redes y protocolos')
  INTO Asignatura VALUES ('TEL-301', 'Comunicaciones Digitales', 5, 3, 'ING-TEL', 4, 2, 'Transmisión digital')
  INTO Asignatura VALUES ('TEL-401', 'Fibra Óptica', 4, 4, 'ING-TEL', 3, 2, 'Comunicaciones por fibra')

  -- Asignaturas adicionales para otras carreras
  INTO Asignatura VALUES ('SIS-701', 'Inteligencia Artificial', 5, 7, 'ING-SIS', 3, 4, 'Fundamentos de IA')
  INTO Asignatura VALUES ('CIV-201', 'Mecánica de Suelos', 5, 2, 'ING-CIV', 4, 2, 'Propiedades de suelos')
  INTO Asignatura VALUES ('ELE-201', 'Electrónica Digital', 4, 2, 'ING-ELE', 3, 2, 'Circuitos digitales')
SELECT * FROM DUAL;

-- 5. ESTABLECER 10 RELACIONES DE PRERREQUISITOS
INSERT ALL
  -- Prerrequisitos ING-AMB
  INTO Prerrequisito VALUES ('AMB-201', 'AMB-101')  -- Química Ambiental requiere Introducción
  INTO Prerrequisito VALUES ('AMB-301', 'AMB-201')  -- Gestión Residuos requiere Química Ambiental
  INTO Prerrequisito VALUES ('AMB-401', 'AMB-301')  -- Impacto Ambiental requiere Gestión Residuos

  -- Prerrequisitos ING-BIO
  INTO Prerrequisito VALUES ('BIO-201', 'BIO-101')  -- Bioquímica requiere Biología General
  INTO Prerrequisito VALUES ('BIO-301', 'BIO-201')  -- Biotecnología Industrial requiere Bioquímica
  INTO Prerrequisito VALUES ('BIO-401', 'BIO-301')  -- Ingeniería Genética requiere Biotecnología

  -- Prerrequisitos ING-TEL
  INTO Prerrequisito VALUES ('TEL-201', 'TEL-101')  -- Redes requiere Fundamentos
  INTO Prerrequisito VALUES ('TEL-301', 'TEL-201')  -- Comunicaciones Digitales requiere Redes
  INTO Prerrequisito VALUES ('TEL-401', 'TEL-301')  -- Fibra Óptica requiere Comunicaciones Digitales

  -- Prerrequisito adicional
  INTO Prerrequisito VALUES ('SIS-701', 'BD-601')   -- IA requiere Bases Avanzadas
SELECT * FROM DUAL;
```

```
-- 6. MATRICULAR 20 ESTUDIANTES EN ASIGNATURAS DEL PERIODO 2024-25
INSERT ALL
    -- Matrículas para estudiantes de ING-AMB
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, fecha_matricula, estado)
    VALUES ('1751122331', 'AMB-101', '1765432101', '2024-25', 'A', SYSDATE, 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122332', 'AMB-101', '1765432101', '2024-25', 'A', 'CURSANDO')

    -- Matrículas para estudiantes de ING-BIO
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122333', 'BIO-101', '1765432102', '2024-25', 'B', 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122334', 'BIO-101', '1765432102', '2024-25', 'B', 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122335', 'BIO-101', '1765432102', '2024-25', 'B', 'CURSANDO')

    -- Matrículas para estudiantes de ING-TEL
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122336', 'TEL-101', '1765432103', '2024-25', 'C', 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122337', 'TEL-101', '1765432103', '2024-25', 'C', 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122338', 'TEL-101', '1765432103', '2024-25', 'C', 'CURSANDO')

    -- Matrículas adicionales para otros estudiantes
    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122339', 'SIS-701', '1712345678', '2024-25', 'A', 'CURSANDO')

    INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
    VALUES ('1751122340', 'CIV-201', '1734567890', '2024-25', 'B', 'CURSANDO')

    -- Más matrículas para completar 20
```

```
  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1750123456', 'BD-601', '1712345678', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1750234567', 'BD-601', '1712345678', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1750345678', 'CIV-201', '1734567890', '2024-25', 'B', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1750456789', 'ELE-201', '1745678901', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1750567890', 'BD-601', '1712345678', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1751122331', 'AMB-201', '1765432101', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1751122333', 'BIO-201', '1765432102', '2024-25', 'B', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1751122336', 'TEL-201', '1765432103', '2024-25', 'C', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1751122339', 'PRG-201', '1734567890', '2024-25', 'A', 'CURSANDO')

  INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, estado)
  VALUES ('1751122340', 'MAT-201', '1712345678', '2024-25', 'B', 'CURSANDO')
SELECT * FROM DUAL;
```

Requisitos:

- Usar INSERT especificando columnas
- Usar INSERT ALL para inserciones múltiples
- Usar valores DEFAULT donde corresponda

- Verificar cada inserción con SELECT
- Hacer COMMIT al final

Entregable: Script SQL con todas las inserciones y verificaciones.

Ejercicio 2: UPDATE Avanzado

Objetivo: Dominar UPDATE en diferentes escenarios.

Tareas:

1. Actualizar teléfonos de 5 docentes
2. Actualizar estado de estudiantes sin matrículas activas a 'INACTIVO'
3. Actualizar notas parciales de 10 matrículas
4. Calcular y actualizar nota_final de todas las matrículas con parciales
5. Actualizar estado de matrículas según nota_final (>=7.0 APROBADO, <7.0 REPROBADO)
6. Actualizar creditos_aprobados de estudiantes sumando créditos de asignaturas aprobadas
7. Actualizar tipo_contrato de docentes según su título académico

Requisitos:

- Hacer SELECT antes de cada UPDATE
- Usar UPDATE con subconsultas
- Usar UPDATE con CASE
- Usar SAVEPOINT entre actualizaciones
- Documentar cuántas filas se actualizaron

Entregable: Script SQL completo con comentarios.

```
-- EJERCICIO 2: UPDATE AVANZADO

-- Seccion 1: actualizar telefonos docentes
```

```
SELECT cedula, nombres, apellidos, telefono

FROM Docente

WHERE cedula IN ('1712345678', '1723456789', '1734567890', '1745678901', '1756789012')

ORDER BY apellidos;

SAVEPOINT antes_actualizar_telefonos;

UPDATE Docente SET telefono = '0987654321' WHERE cedula = '1712345678';

UPDATE Docente SET telefono = '0976543210' WHERE cedula = '1723456789';

UPDATE Docente SET telefono = '0965432109' WHERE cedula = '1734567890';

UPDATE Docente SET telefono = '0954321098' WHERE cedula = '1745678901';

UPDATE Docente SET telefono = '0943210987' WHERE cedula = '1756789012';

SELECT cedula, nombres, apellidos, telefono

FROM Docente

WHERE cedula IN ('1712345678', '1723456789', '1734567890', '1745678901', '1756789012')

ORDER BY apellidos;

DBMS_OUTPUT.PUT_LINE('Tarea 1 completada: ' || SQL%ROWCOUNT || ' docentes actualizados');

-- Seccion 2: estudiantes sin matriculas activas a INACTIVO

SELECT e.cedula, e.nombres, e.apellidos, e.estado, COUNT(m.id_matricula) AS matriculas_activas

FROM Estudiante e
```

```
LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante AND m.estado = 'CURSANDO'

WHERE e.estado = 'ACTIVO'

GROUP BY e.cedula, e.nombres, e.apellidos, e.estado

HAVING COUNT(m.id_matricula) = 0;

SAVEPOINT antes_actualizar_estados;

UPDATE Estudiante

SET estado = 'INACTIVO'

WHERE cedula IN (

SELECT e.cedula

FROM Estudiante e

LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante AND m.estado = 'CURSANDO'

WHERE e.estado = 'ACTIVO'

GROUP BY e.cedula

HAVING COUNT(m.id_matricula) = 0

);

SELECT cedula, nombres, apellidos, estado

FROM Estudiante

WHERE estado = 'INACTIVO';
```

```
DBMS_OUTPUT.PUT_LINE('Tarea 2 completada: ' || SQL%ROWCOUNT || ' estudiantes actualizados a INACTIVO');

-- Seccion 3: actualizar notas parciales de matriculas

SELECT id_matricula, cedula_estudiante, codigo_asignatura, nota_parcial1, nota_parcial2, nota_final, estado

FROM Matricula

WHERE id_matricula BETWEEN 1 AND 10

ORDER BY id_matricula;

SAVEPOINT antes_actualizar_notas;

UPDATE Matricula SET nota_parcial1 = 8.5, nota_parcial2 = 7.8 WHERE id_matricula IN (1,2,3);

UPDATE Matricula SET nota_parcial1 = 9.0 WHERE id_matricula IN (4,5);

UPDATE Matricula SET nota_parcial2 = 8.2 WHERE id_matricula IN (6,7);

UPDATE Matricula SET nota_parcial1 = 6.5, nota_parcial2 = 7.0 WHERE id_matricula IN (8,9,10);

SELECT id_matricula, cedula_estudiante, codigo_asignatura, nota_parcial1, nota_parcial2, nota_final, estado

FROM Matricula

WHERE id_matricula BETWEEN 1 AND 10

ORDER BY id_matricula;

DBMS_OUTPUT.PUT_LINE('Tarea 3 completada: ' || SQL%ROWCOUNT || ' matriculas actualizadas');

-- Seccion 4: calcular nota final

SELECT id_matricula, nota_parcial1, nota_parcial2, (nota_parcial1 + nota_parcial2) / 2 AS nota_final_calculada
```

```
FROM Matricula

WHERE nota_parcial1 IS NOT NULL AND nota_parcial2 IS NOT NULL

AND id_matricula BETWEEN 1 AND 10;

SAVEPOINT antes_calcular_notas_finales;

UPDATE Matricula

SET nota_final = (nota_parcial1 + nota_parcial2) / 2

WHERE nota_parcial1 IS NOT NULL AND nota_parcial2 IS NOT NULL

AND id_matricula BETWEEN 1 AND 10;

SELECT id_matricula, nota_parcial1, nota_parcial2, nota_final

FROM Matricula

WHERE nota_final IS NOT NULL AND id_matricula BETWEEN 1 AND 10

ORDER BY id_matricula;

DBMS_OUTPUT.PUT_LINE('Tarea 4 completada: ' || SQL%ROWCOUNT || ' notas finales calculadas');

-- Seccion 5: actualizar estado aprobado o reprobado

SELECT id_matricula, nota_final, estado,

CASE WHEN nota_final >= 7.0 THEN 'APROBADO' ELSE 'REPROBADO' END AS nuevo_estado

FROM Matricula

WHERE nota_final IS NOT NULL AND id_matricula BETWEEN 1 AND 10;
```

```
SAVEPOINT antes_actualizar_estados_matricula;

UPDATE Matricula

SET estado = CASE WHEN nota_final >= 7.0 THEN 'APROBADO' ELSE 'REPROBADO' END

WHERE nota_final IS NOT NULL AND id_matricula BETWEEN 1 AND 10;

SELECT id_matricula, nota_final, estado

FROM Matricula

WHERE nota_final IS NOT NULL AND id_matricula BETWEEN 1 AND 10

ORDER BY id_matricula;

DBMS_OUTPUT.PUT_LINE('Tarea 5 completada: ' || SQL%ROWCOUNT || ' estados de matricula actualizados');

-- Seccion 6: actualizar creditos aprobados de estudiantes

SELECT e.cedula, e.nombres, e.apellidos, e.creditos_aprobados,

COALESCE(SUM(a.creditos), 0) AS creditos_aprobados_calculados,

COUNT(m.id_matricula) AS materias_aprobadas

FROM Estudiante e

LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante AND m.estado = 'APROBADO'

LEFT JOIN Asignatura a ON m.codigo_asignatura = a.codigo

WHERE e.cedula IN ('1750123456', '1750234567', '1750567890')

GROUP BY e.cedula, e.nombres, e.apellidos, e.creditos_aprobados;
```

```
SAVEPOINT antes_actualizar_creditos;

UPDATE Estudiante e

SET creditos_aprobados = (

SELECT COALESCE(SUM(a.creditos), 0)

FROM Matricula m

JOIN Asignatura a ON m.codigo_asignatura = a.codigo

WHERE m.cedula_estudiante = e.cedula

AND m.estado = 'APROBADO'

)

WHERE e.cedula IN ('1750123456', '1750234567', '1750567890');

SELECT e.cedula, e.nombres, e.apellidos, e.creditos_aprobados,

COUNT(m.id_matricula) AS materias_aprobadas

FROM Estudiante e

LEFT JOIN Matricula m ON e.cedula = m.cedula_estudiante AND m.estado = 'APROBADO'

WHERE e.cedula IN ('1750123456', '1750234567', '1750567890')

GROUP BY e.cedula, e.nombres, e.apellidos, e.creditos_aprobados;

DBMS_OUTPUT.PUT_LINE('Tarea 6 completada: ' || SQL%ROWCOUNT || ' estudiantes actualizados');

-- Seccion 7: actualizar tipo_contrato por titulo academico
```

```
SELECT cedula, nombres, apellidos, titulo, tipo_contrato,

CASE

WHEN UPPER(titulo) LIKE '%PHD%' THEN 'TIEMPO_COMPLETO'

WHEN UPPER(titulo) LIKE '%MSC%' OR UPPER(titulo) LIKE '%MASTER%' THEN 'MEDIO_TIEMPO'

ELSE 'HORA_CLASE'

END AS nuevo_tipo_contrato

FROM Docente

WHERE cedula LIKE '17%';

SAVEPOINT antes_actualizar_contratos;

UPDATE Docente

SET tipo_contrato = CASE

WHEN UPPER(titulo) LIKE '%PHD%' THEN 'TIEMPO_COMPLETO'

WHEN UPPER(titulo) LIKE '%MSC%' OR UPPER(titulo) LIKE '%MASTER%' THEN 'MEDIO_TIEMPO'

ELSE 'HORA_CLASE'

END

WHERE cedula LIKE '17%';

SELECT cedula, nombres, apellidos, titulo, tipo_contrato

FROM Docente
```

```
WHERE cedula LIKE '17%'

ORDER BY tipo_contrato, apellidos;

DBMS_OUTPUT.PUT_LINE('Tarea 7 completada: ' || SQL%ROWCOUNT || ' docentes actualizados');

-- Seccion final: verificacion resumen

DECLARE

v_total_updates NUMBER := 0;

BEGIN

SELECT COUNT(*) INTO v_total_updates

FROM Docente

WHERE cedula IN ('1712345678','1723456789','1734567890','1745678901','1756789012')

AND telefono IN ('0987654321','0976543210','0965432109','0954321098','0943210987');

DBMS_OUTPUT.PUT_LINE('Telefonos actualizados: ' || v_total_updates || '/5');

SELECT COUNT(*) INTO v_total_updates

FROM Estudiante WHERE estado = 'INACTIVO';

DBMS_OUTPUT.PUT_LINE('Estudiantes inactivos: ' || v_total_updates);

SELECT COUNT(*) INTO v_total_updates

FROM Matricula

WHERE nota_final IS NOT NULL
```

```
AND estado IN ('APROBADO','REPROBADO')

AND id_matricula BETWEEN 1 AND 10;

DBMS_OUTPUT.PUT_LINE('Matriculas procesadas: ' || v_total_updates || '/10');

SELECT COUNT(*) INTO v_total_updates

FROM Estudiante

WHERE cedula IN ('1750123456','1750234567','1750567890')

AND creditos_aprobados > 0;

DBMS_OUTPUT.PUT_LINE('Creditos actualizados: ' || v_total_updates || '/3');

SELECT COUNT(*) INTO v_total_updates

FROM Docente

WHERE cedula LIKE '17%'

AND tipo_contrato IN ('TIEMPO_COMPLETO','MEDIO_TIEMPO','HORA_CLASE');

DBMS_OUTPUT.PUT_LINE('Contratos actualizados: ' || v_total_updates);

END;

/

COMMIT;
```

Ejercicio 3: DELETE Seguro

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Objetivo: Practicar eliminaciones respetando integridad.

Tareas:

1. Eliminar estudiantes de prueba (cédula empieza con '999')
2. Eliminar matrículas del periodo '2023-1S'
3. Eliminar prerrequisitos de asignaturas que ya no existen
4. Eliminar asignaturas que nunca han tenido matrículas
5. Intentar eliminar una carrera con estudiantes (debe fallar)
6. Crear procedimiento para eliminar estudiante con todas sus dependencias

Requisitos:

- Usar SELECT COUNT antes de DELETE
- Usar SAVEPOINT
- Manejar errores de integridad referencial
- Documentar qué se eliminó

Entregable: Script SQL con procedimiento de eliminación segura.

```
-- EJERCICIO 3: DELETE SEGURO

-- Tarea 1: Eliminar estudiantes de prueba (cédula empieza con '999')

-- Verificar cuantos estudiantes se eliminaran

SELECT COUNT(*) FROM Estudiante WHERE cedula LIKE '999%';


SAVEPOINT antes_eliminar_estudiantes_prueba;


DELETE FROM Estudiante WHERE cedula LIKE '999%';


DBMS_OUTPUT.PUT_LINE('Estudiantes eliminados: ' || SQL%ROWCOUNT);
```

```
-- Tarea 2: Eliminar matrículas del período '2023-1S'

-- Verificar matrículas a eliminar

SELECT COUNT(*) FROM Matricula WHERE periodo = '2023-1S';


SAVEPOINT antes_eliminar_matriculas_2023;


DELETE FROM Matricula WHERE periodo = '2023-1S';


DBMS_OUTPUT.PUT_LINE('Matrículas 2023-1S eliminadas: ' || SQL%ROWCOUNT);


-- Tarea 3: Eliminar prerrequisitos de asignaturas que ya no existen

-- Verificar prerrequisitos huérfanos

SELECT COUNT(*)

FROM Prerrequisito p

WHERE NOT EXISTS (SELECT 1 FROM Asignatura a WHERE a.codigo = p.codigo_asignatura)

   OR NOT EXISTS (SELECT 1 FROM Asignatura a WHERE a.codigo = p.codigo_prerrequisito);


SAVEPOINT antes_eliminar_prerrequisitos_huérfanos;
```

```
DELETE FROM Prerrequisito
WHERE NOT EXISTS (SELECT 1 FROM Asignatura a WHERE a.codigo = codigo_asignatura)
   OR NOT EXISTS (SELECT 1 FROM Asignatura a WHERE a.codigo = codigo_prerrequisito);

DBMS_OUTPUT.PUT_LINE('Prerrequisitos huérfanos eliminados: ' || SQL%ROWCOUNT);

-- Tarea 4: Eliminar asignaturas que nunca han tenido matrículas
-- Verificar asignaturas sin matrículas
SELECT COUNT(*)
FROM Asignatura a
WHERE NOT EXISTS (SELECT 1 FROM Matricula m WHERE m.codigo_asignatura = a.codigo);

SAVEPOINT antes_eliminar_asignaturas_sin_matricula;

DELETE FROM Asignatura
WHERE NOT EXISTS (SELECT 1 FROM Matricula m WHERE m.codigo_asignatura = codigo);

DBMS_OUTPUT.PUT_LINE('Asignaturas sin matrículas eliminadas: ' || SQL%ROWCOUNT);

-- Tarea 5: Intentar eliminar una carrera con estudiantes (debe fallar)
```

```
-- Verificar que la carrera tiene estudiantes
SELECT COUNT(*) FROM Estudiante WHERE codigo_carrera = 'ING-SIS';


SAVEPOINT antes_eliminar_carrera_con_estudiantes;


BEGIN
   DELETE FROM Carrera WHERE codigo = 'ING-SIS';
   DBMS_OUTPUT.PUT_LINE('ERROR: No debería permitir eliminar carrera con estudiantes');
EXCEPTION
   WHEN OTHERS THEN
      IF SQLCODE = -2292 THEN
         DBMS_OUTPUT.PUT_LINE('Correcto: No se puede eliminar carrera con estudiantes');
      ELSE
         DBMS_OUTPUT.PUT_LINE('Error inesperado: ' || SQLERRM);
      END IF;
END;
/


-- Tarea 6: Procedimiento para eliminar estudiante con todas sus dependencias
CREATE OR REPLACE PROCEDURE eliminar_estudiante_completo(
```

```
    p_cedula_estudiante IN Estudiante.cedula%TYPE
) IS
    v_existe_estudiante NUMBER;
    v_matriculas_eliminadas NUMBER;
BEGIN
    -- Verificar que el estudiante existe
    SELECT COUNT(*) INTO v_existe_estudiante
    FROM Estudiante
    WHERE cedula = p_cedula_estudiante;

    IF v_existe_estudiante = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Error: El estudiante no existe');
        RETURN;
    END IF;

    SAVEPOINT antes_eliminar_estudiante;

    -- Primero eliminar las matrículas del estudiante
    DELETE FROM Matricula
    WHERE cedula_estudiante = p_cedula_estudiante;
```

```
    v_matriculas_eliminadas := SQL%ROWCOUNT;

    -- Luego eliminar el estudiante
    DELETE FROM Estudiante
    WHERE cedula = p_cedula_estudiante;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Estudiante eliminado: ' || p_cedula_estudiante);
    DBMS_OUTPUT.PUT_LINE('Matrículas eliminadas: ' || v_matriculas_eliminadas);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO antes_eliminar_estudiante;
        DBMS_OUTPUT.PUT_LINE('Error al eliminar estudiante: ' || SQLERRM);
        RAISE;
END eliminar_estudiante_completo;
/
```

```
-- Probar el procedimiento con un estudiante de prueba

-- Primero crear un estudiante de prueba y sus matrículas

INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)

VALUES ('9998887770', 'Test', 'Eliminacion', 'test.elim@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');


INSERT INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)

VALUES ('9998887770', 'BD-501', '1712345678', '2024-2S', 'Z');


COMMIT;


-- Ejecutar el procedimiento

BEGIN

    eliminar_estudiante_completo('9998887770');

END;

/


-- Verificar que se eliminó correctamente

SELECT COUNT(*) FROM Estudiante WHERE cedula = '9998887770';

SELECT COUNT(*) FROM Matricula WHERE cedula_estudiante = '9998887770';
```

```
-- Confirmar todos los cambios restantes

COMMIT;


-- Resumen final

BEGIN

   DBMS_OUTPUT.PUT_LINE('Ejercicio 3 completado');

   DBMS_OUTPUT.PUT_LINE('Se utilizaron SAVEPOINTs para seguridad');

   DBMS_OUTPUT.PUT_LINE('Se manejan errores de integridad referencial');

   DBMS_OUTPUT.PUT_LINE('Se creo procedimiento para eliminacion segura');

END;

/
```

Ejercicio 4: Transacciones Complejas (30 minutos)

Escenario: Sistema de matrícula completo.

Implementar:

1. **Procedimiento matricular_estudiante:**
   - Verificar que el estudiante existe
   - Verificar que la asignatura existe
   - Verificar que cumple prerrequisitos
   - Verificar que no está matriculado en la misma asignatura
   - Insertar matrícula
   - Todo con manejo de errores y transacciones
2. **Procedimiento registrar_calificaciones:**
   - Actualizar notas parciales

- Calcular nota final
- Actualizar estado (APROBADO/REPROBADO)
- Actualizar créditos del estudiante
- Todo atómico

3. **Procedimiento retirar_estudiante:**
   - Actualizar estado a 'RETIRADO'
   - Actualizar matrículas a 'RETIRADO'
   - Registrar fecha de retiro
   - Con transacción y rollback en caso de error

Requisitos:

- Usar procedimientos almacenados PL/SQL
- Implementar SAVEPOINT
- Manejo completo de excepciones
- Retornar mensajes de éxito/error

Entregable: 3 procedimientos con casos de prueba.

-- EJERCICIO 4: TRANSACCIONES COMPLEJA

-- Procedimiento 1: matricular_estudiante

```
CREATE OR REPLACE PROCEDURE matricular_estudiante(
    p_cedula_estudiante IN Estudiante.cedula%TYPE,
    p_codigo_asignatura IN Asignatura.codigo%TYPE,
    p_cedula_docente IN Docente.cedula%TYPE,
    p_periodo IN Matricula.periodo%TYPE,
    p_paralelo IN Matricula.paralelo%TYPE
) IS
```

```
    v_existe_estudiante NUMBER;
    v_existe_asignatura NUMBER;
    v_existe_docente NUMBER;
    v_prerrequisitos_cumplidos NUMBER;
    v_ya_matriculado NUMBER;
    v_id_matricula NUMBER;
BEGIN
    SAVEPOINT inicio_matricula;

    -- Verificar que el estudiante existe
    SELECT COUNT(*) INTO v_existe_estudiante
    FROM Estudiante WHERE cedula = p_cedula_estudiante;

    IF v_existe_estudiante = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'El estudiante no existe');
    END IF;

    -- Verificar que la asignatura existe
    SELECT COUNT(*) INTO v_existe_asignatura
    FROM Asignatura WHERE codigo = p_codigo_asignatura;
```

```
IF v_existe_asignatura = 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'La asignatura no existe');
END IF;

-- Verificar que el docente existe
SELECT COUNT(*) INTO v_existe_docente
FROM Docente WHERE cedula = p_cedula_docente;

IF v_existe_docente = 0 THEN
    RAISE_APPLICATION_ERROR(-20003, 'El docente no existe');
END IF;

-- Verificar prerrequisitos
SELECT COUNT(*) INTO v_prerrequisitos_cumplidos
FROM Prerrequisito p
WHERE p.codigo_asignatura = p_codigo_asignatura
AND NOT EXISTS (
    SELECT 1 FROM Matricula m
    WHERE m.cedula_estudiante = p_cedula_estudiante
```

```
    AND m.codigo_asignatura = p.codigo_prerrequisito
    AND m.estado = 'APROBADO'
);

IF v_prerrequisitos_cumplidos > 0 THEN
    RAISE_APPLICATION_ERROR(-20004, 'No cumple con los prerrequisitos');
END IF;

-- Verificar que no está matriculado en la misma asignatura
SELECT COUNT(*) INTO v_ya_matriculado
FROM Matricula
WHERE cedula_estudiante = p_cedula_estudiante
AND codigo_asignatura = p_codigo_asignatura
AND periodo = p_periodo;

IF v_ya_matriculado > 0 THEN
    RAISE_APPLICATION_ERROR(-20005, 'Ya está matriculado en esta asignatura');
END IF;

-- Insertar matrícula
```

```
    INSERT INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo, fecha_matricula, estado)
    VALUES (p_cedula_estudiante, p_codigo_asignatura, p_cedula_docente, p_periodo, p_paralelo, SYSDATE, 'CURSANDO');

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Matrícula exitosa');

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO inicio_matricula;
        DBMS_OUTPUT.PUT_LINE('Error en matrícula: ' || SQLERRM);
        RAISE;
END matricular_estudiante;
/

-- Procedimiento 2: registrar_calificaciones
CREATE OR REPLACE PROCEDURE registrar_calificaciones(
    p_id_matricula IN Matricula.id_matricula%TYPE,
    p_nota_parcial1 IN Matricula.nota_parcial1%TYPE,
    p_nota_parcial2 IN Matricula.nota_parcial2%TYPE
) IS
```

```
    v_nota_final NUMBER;
    v_estado Matricula.estado%TYPE;
    v_creditos Asignatura.creditos%TYPE;
    v_cedula_estudiante Estudiante.cedula%TYPE;
    v_codigo_asignatura Asignatura.codigo%TYPE;
BEGIN
    SAVEPOINT inicio_calificaciones;

    -- Actualizar notas parciales
    UPDATE Matricula
    SET nota_parcial1 = p_nota_parcial1,
        nota_parcial2 = p_nota_parcial2
    WHERE id_matricula = p_id_matricula;

    -- Calcular nota final
    v_nota_final := (p_nota_parcial1 + p_nota_parcial2) / 2;

    -- Determinar estado
    IF v_nota_final >= 7.0 THEN
        v_estado := 'APROBADO';
```

```
ELSE
    v_estado := 'REPROBADO';
END IF;

-- Obtener datos para actualizar créditos
SELECT cedula_estudiante, codigo_asignatura
INTO v_cedula_estudiante, v_codigo_asignatura
FROM Matricula
WHERE id_matricula = p_id_matricula;

SELECT creditos INTO v_creditos
FROM Asignatura
WHERE codigo = v_codigo_asignatura;

-- Actualizar matrícula con nota final y estado
UPDATE Matricula
SET nota_final = v_nota_final,
    estado = v_estado
WHERE id_matricula = p_id_matricula;
```

```
    -- Actualizar créditos del estudiante si aprobó
    IF v_estado = 'APROBADO' THEN
        UPDATE Estudiante
        SET creditos_aprobados = creditos_aprobados + v_creditos
        WHERE cedula = v_cedula_estudiante;
    END IF;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Calificaciones registradas - Estado: ' || v_estado);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO inicio_calificaciones;
        DBMS_OUTPUT.PUT_LINE('Error al registrar calificaciones: ' || SQLERRM);
        RAISE;
END registrar_calificaciones;
/

-- Procedimiento 3: retirar_estudiante
CREATE OR REPLACE PROCEDURE retirar_estudiante(
```

```
    p_cedula_estudiante IN Estudiante.cedula%TYPE
) IS
    v_existe_estudiante NUMBER;
    v_matriculas_activas NUMBER;
BEGIN
    SAVEPOINT inicio_retiro;

    -- Verificar que el estudiante existe
    SELECT COUNT(*) INTO v_existe_estudiante
    FROM Estudiante
    WHERE cedula = p_cedula_estudiante;

    IF v_existe_estudiante = 0 THEN
        RAISE_APPLICATION_ERROR(-20006, 'El estudiante no existe');
    END IF;

    -- Actualizar estado del estudiante
    UPDATE Estudiante
    SET estado = 'RETIRADO'
    WHERE cedula = p_cedula_estudiante;
```

```
    -- Actualizar matrículas a RETIRADO
    UPDATE Matricula
    SET estado = 'RETIRADO'
    WHERE cedula_estudiante = p_cedula_estudiante
    AND estado = 'CURSANDO';

    v_matriculas_activas := SQL%ROWCOUNT;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Estudiante retirado - Matrículas afectadas: ' || v_matriculas_activas);

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK TO inicio_retiro;
        DBMS_OUTPUT.PUT_LINE('Error al retirar estudiante: ' || SQLERRM);
        RAISE;
END retirar_estudiante;
/
```

```
-- Casos de prueba
BEGIN
    DBMS_OUTPUT.PUT_LINE('=== PRUEBA PROCEDIMIENTOS ===');

    -- Prueba matricular_estudiante
    BEGIN
        matricular_estudiante('1750123456', 'PRG-201', '1712345678', '2024-2S', 'B');
        DBMS_OUTPUT.PUT_LINE('Prueba matrícula: EXITOSA');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Prueba matrícula: FALLIDA - ' || SQLERRM);
    END;

    -- Prueba registrar_calificaciones
    BEGIN
        registrar_calificaciones(1, 8.5, 7.5);
        DBMS_OUTPUT.PUT_LINE('Prueba calificaciones: EXITOSA');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Prueba calificaciones: FALLIDA - ' || SQLERRM);
```

```
    END;

    -- Prueba retirar_estudiante (usar estudiante de prueba)
    BEGIN
        INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
        VALUES ('9997776660', 'Test', 'Retiro', 'test.retiro@test.com', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS');

        retirar_estudiante('9997776660');
        DBMS_OUTPUT.PUT_LINE('Prueba retiro: EXITOSA');
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Prueba retiro: FALLIDA - ' || SQLERRM);
    END;

    -- Limpiar datos de prueba
    DELETE FROM Estudiante WHERE cedula = '9997776660';
    COMMIT;

END;
/
```

```
-- Verificar resultados finales
SELECT 'Estudiante 1750123456 matriculado en PRG-201' as verificacion FROM Matricula
WHERE cedula_estudiante = '1750123456' AND codigo_asignatura = 'PRG-201'
UNION ALL
SELECT 'Matrícula 1 con calificaciones registradas' FROM Matricula
WHERE id_matricula = 1 AND nota_final IS NOT NULL
UNION ALL
SELECT 'Procedimientos creados exitosamente' FROM DUAL;

COMMIT;

BEGIN
    DBMS_OUTPUT.PUT_LINE('Ejercicio 4 completado');
    DBMS_OUTPUT.PUT_LINE('3 procedimientos creados con manejo de transacciones');
    DBMS_OUTPUT.PUT_LINE('Casos de prueba ejecutados');
END;
/
```

Ejercicio 5: Operaciones Masivas

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Tareas:

1. Cargar 100 estudiantes desde archivo CSV (simular con INSERT ALL)
2. Actualizar masivamente calificaciones de un periodo
3. Migrar matrículas antiguas a tabla histórica
4. Limpiar datos de prueba de todas las tablas
5. Importar datos desde otra base de datos (simular)

Requisitos:

- Usar transacciones por lotes
- Medir tiempo de ejecución
- Manejar errores individuales sin detener el proceso
- Generar reporte de operaciones realizadas

Entregable: Scripts con operaciones masivas optimizadas.

```
-- EJERCICIO 5: OPERACIONES MASIVAS

SET TIMING ON;



-- Tarea 1: Cargar 100 estudiantes desde archivo CSV (simular con INSERT ALL)

DECLARE

   v_inicio TIMESTAMP;

   v_fin TIMESTAMP;

   v_estudiantes_insertados NUMBER := 0;

BEGIN

   v_inicio := SYSTIMESTAMP;
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

```
FOR i IN 1..10 LOOP

    BEGIN

        INSERT ALL

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 1), 'Estudiante' || ((i-1)*10 + 1), 'Apellido' || ((i-1)*10 + 1), 'est' || ((i-1)*10 + 1) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 1, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 2), 'Estudiante' || ((i-1)*10 + 2), 'Apellido' || ((i-1)*10 + 2), 'est' || ((i-1)*10 + 2) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 2, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 3), 'Estudiante' || ((i-1)*10 + 3), 'Apellido' || ((i-1)*10 + 3), 'est' || ((i-1)*10 + 3) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 3, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-CIV', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 4), 'Estudiante' || ((i-1)*10 + 4), 'Apellido' || ((i-1)*10 + 4), 'est' || ((i-1)*10 + 4) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 4, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-CIV', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 5), 'Estudiante' || ((i-1)*10 + 5), 'Apellido' || ((i-1)*10 + 5), 'est' || ((i-1)*10 + 5) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 5, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-ELE', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 6), 'Estudiante' || ((i-1)*10 + 6), 'Apellido' || ((i-1)*10 + 6), 'est' || ((i-1)*10 + 6) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 6, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-ELE', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 7), 'Estudiante' || ((i-1)*10 + 7), 'Apellido' || ((i-1)*10 + 7), 'est' || ((i-1)*10 + 7) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 7, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-AMB', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 8), 'Estudiante' || ((i-1)*10 + 8), 'Apellido' || ((i-1)*10 + 8), 'est' || ((i-1)*10 + 8) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 8, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-AMB', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 9), 'Estudiante' || ((i-1)*10 + 9), 'Apellido' || ((i-1)*10 + 9), 'est' || ((i-1)*10 + 9) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 9, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'M', 'ING-BIO', 'ACTIVO', 0, SYSDATE)

            INTO Estudiante VALUES (TO_CHAR(1800000000 + (i-1)*10 + 10), 'Estudiante' || ((i-1)*10 + 10), 'Apellido' || ((i-1)*10 + 10), 'est' || ((i-1)*10 + 10) || '@epn.edu.ec', '099' || LPAD((i-1)*10 + 10, 7, '0'), TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'F', 'ING-BIO', 'ACTIVO', 0, SYSDATE)

        SELECT * FROM DUAL;
```

```
        v_estudiantes_insertados := v_estudiantes_insertados + 10;


        IF MOD(i, 2) = 0 THEN
            COMMIT;
        END IF;


    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error en lote ' || i || ': ' || SQLERRM);
            CONTINUE;
    END;
END LOOP;


COMMIT;
v_fin := SYSTIMESTAMP;


DBMS_OUTPUT.PUT_LINE('Estudiantes insertados: ' || v_estudiantes_insertados);
DBMS_OUTPUT.PUT_LINE('Tiempo carga estudiantes: ' || (v_fin - v_inicio));
```

```
END;
/

-- Tarea 2: Actualizar masivamente calificaciones de un período
DECLARE
    v_inicio TIMESTAMP;
    v_fin TIMESTAMP;
    v_matriculas_actualizadas NUMBER := 0;
BEGIN
    v_inicio := SYSTIMESTAMP;

    FOR i IN 1..5 LOOP
        BEGIN
            SAVEPOINT antes_actualizar_lote;

            UPDATE Matricula
            SET nota_parcial1 = ROUND(DBMS_RANDOM.VALUE(5, 10), 1),
                nota_parcial2 = ROUND(DBMS_RANDOM.VALUE(5, 10), 1),
                nota_final = (nota_parcial1 + nota_parcial2) / 2,
```

```
        estado = CASE WHEN (nota_parcial1 + nota_parcial2) / 2 >= 7 THEN 'APROBADO' ELSE 'REPROBADO' END
    WHERE id_matricula BETWEEN ((i-1)*20 + 1) AND (i*20)
    AND periodo = '2024-2S';

    v_matriculas_actualizadas := v_matriculas_actualizadas + SQL%ROWCOUNT;

    COMMIT;

  EXCEPTION
    WHEN OTHERS THEN
      ROLLBACK TO antes_actualizar_lote;
      DBMS_OUTPUT.PUT_LINE('Error en actualización lote ' || i || ': ' || SQLERRM);
      CONTINUE;
  END;
END LOOP;

v_fin := SYSTIMESTAMP;

DBMS_OUTPUT.PUT_LINE('Matrículas actualizadas: ' || v_matriculas_actualizadas);
```

```
    DBMS_OUTPUT.PUT_LINE('Tiempo actualización calificaciones: ' || (v_fin - v_inicio));

END;

/



-- Tarea 3: Migrar matrículas antiguas a tabla histórica

-- Primero crear tabla histórica si no existe

BEGIN

    EXECUTE IMMEDIATE 'CREATE TABLE Matricula_Historico (

        id_matricula NUMBER,

        cedula_estudiante VARCHAR2(10),

        codigo_asignatura VARCHAR2(10),

        cedula_docente VARCHAR2(10),

        periodo VARCHAR2(10),

        paralelo VARCHAR2(2),

        fecha_matricula DATE,

        nota_parcial1 NUMBER(3,1),

        nota_parcial2 NUMBER(3,1),

        nota_final NUMBER(3,1),

        estado VARCHAR2(20),
```

```
        fecha_migracion DATE
    )';
EXCEPTION
    WHEN OTHERS THEN
        IF SQLCODE = -955 THEN
            NULL;
        ELSE
            RAISE;
        END IF;
END;
/


DECLARE
    v_inicio TIMESTAMP;
    v_fin TIMESTAMP;
    v_matriculas_migradas NUMBER := 0;
BEGIN
    v_inicio := SYSTIMESTAMP;
```

```
FOR i IN 1..3 LOOP
  BEGIN
    INSERT INTO Matricula_Historico
    SELECT m.*, SYSDATE
    FROM Matricula m
    WHERE m.periodo LIKE '2023%'
    AND m.id_matricula BETWEEN ((i-1)*10 + 1) AND (i*10);

    v_matriculas_migradas := v_matriculas_migradas + SQL%ROWCOUNT;

    DELETE FROM Matricula
    WHERE periodo LIKE '2023%'
    AND id_matricula BETWEEN ((i-1)*10 + 1) AND (i*10);

    COMMIT;

  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error en migración lote ' || i || ': ' || SQLERRM);
```

```
            ROLLBACK;
            CONTINUE;
        END;
    END LOOP;

    v_fin := SYSTIMESTAMP;

    DBMS_OUTPUT.PUT_LINE('Matrículas migradas: ' || v_matriculas_migradas);
    DBMS_OUTPUT.PUT_LINE('Tiempo migración: ' || (v_fin - v_inicio));
END;
/

-- Tarea 4: Limpiar datos de prueba de todas las tablas
DECLARE
    v_inicio TIMESTAMP;
    v_fin TIMESTAMP;
    v_registros_eliminados NUMBER := 0;
BEGIN
    v_inicio := SYSTIMESTAMP;
```

```
BEGIN
    DELETE FROM Matricula WHERE cedula_estudiante LIKE '18%';
    v_registros_eliminados := v_registros_eliminados + SQL%ROWCOUNT;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error limpiando matrículas: ' || SQLERRM);
        ROLLBACK;
END;


BEGIN
    DELETE FROM Estudiante WHERE cedula LIKE '18%';
    v_registros_eliminados := v_registros_eliminados + SQL%ROWCOUNT;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error limpiando estudiantes: ' || SQLERRM);
        ROLLBACK;
```

```
    END;

    BEGIN
        DELETE FROM Matricula_Historico WHERE cedula_estudiante LIKE '999%';
        v_registros_eliminados := v_registros_eliminados + SQL%ROWCOUNT;
        COMMIT;
    EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error limpiando histórico: ' || SQLERRM);
            ROLLBACK;
    END;

    v_fin := SYSTIMESTAMP;

    DBMS_OUTPUT.PUT_LINE('Registros de prueba eliminados: ' || v_registros_eliminados);
    DBMS_OUTPUT.PUT_LINE('Tiempo limpieza: ' || (v_fin - v_inicio));
END;
/
```

```
-- Tarea 5: Importar datos desde otra base de datos (simular)

-- Simulamos importación desde una tabla temporal

DECLARE

    v_inicio TIMESTAMP;

    v_fin TIMESTAMP;

    v_importados NUMBER := 0;

BEGIN

    v_inicio := SYSTIMESTAMP;



    -- Crear tabla temporal para simular datos externos

    EXECUTE IMMEDIATE 'CREATE GLOBAL TEMPORARY TABLE Temp_Importacion (

        codigo VARCHAR2(10),

        nombre VARCHAR2(100),

        creditos NUMBER,

        nivel NUMBER,

        codigo_carrera VARCHAR2(10)

    ) ON COMMIT PRESERVE ROWS';



    -- Insertar datos de prueba en tabla temporal
```

```
INSERT INTO Temp_Importacion VALUES ('EXT-101', 'Asignatura Externa 1', 4, 1, 'ING-SIS');

INSERT INTO Temp_Importacion VALUES ('EXT-102', 'Asignatura Externa 2', 5, 2, 'ING-SIS');

INSERT INTO Temp_Importacion VALUES ('EXT-201', 'Asignatura Externa 3', 4, 1, 'ING-CIV');

INSERT INTO Temp_Importacion VALUES ('EXT-202', 'Asignatura Externa 4', 5, 2, 'ING-CIV');


COMMIT;


-- Importar datos desde tabla temporal

FOR i IN 1..2 LOOP

   BEGIN

      INSERT INTO Asignatura (codigo, nombre, creditos, nivel, codigo_carrera, horas_teoria, horas_practica, descripcion)

      SELECT codigo, nombre, creditos, nivel, codigo_carrera, 3, 2, 'Importado desde sistema externo'

      FROM Temp_Importacion

      WHERE ROWNUM <= 2;


      v_importados := v_importados + SQL%ROWCOUNT;

      COMMIT;


   EXCEPTION
```

```
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error en importación lote ' || i || ': ' || SQLERRM);
            ROLLBACK;
            CONTINUE;
        END;
    END LOOP;

    -- Limpiar tabla temporal
    EXECUTE IMMEDIATE 'TRUNCATE TABLE Temp_Importacion';
    EXECUTE IMMEDIATE 'DROP TABLE Temp_Importacion';

    v_fin := SYSTIMESTAMP;

    DBMS_OUTPUT.PUT_LINE('Asignaturas importadas: ' || v_importados);
    DBMS_OUTPUT.PUT_LINE('Tiempo importación: ' || (v_fin - v_inicio));
END;
/

-- Reporte final de operaciones
```


```
DECLARE
    v_total_estudiantes NUMBER;
    v_total_matriculas NUMBER;
    v_total_historico NUMBER;
    v_total_asignaturas NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_total_estudiantes FROM Estudiante;
    SELECT COUNT(*) INTO v_total_matriculas FROM Matricula;
    SELECT COUNT(*) INTO v_total_historico FROM Matricula_Historico;
    SELECT COUNT(*) INTO v_total_asignaturas FROM Asignatura;

    DBMS_OUTPUT.PUT_LINE('=== REPORTE FINAL OPERACIONES MASIVAS ===');
    DBMS_OUTPUT.PUT_LINE('Total estudiantes en sistema: ' || v_total_estudiantes);
    DBMS_OUTPUT.PUT_LINE('Total matrículas activas: ' || v_total_matriculas);
    DBMS_OUTPUT.PUT_LINE('Total matrículas en histórico: ' || v_total_historico);
    DBMS_OUTPUT.PUT_LINE('Total asignaturas: ' || v_total_asignaturas);
    DBMS_OUTPUT.PUT_LINE('Operaciones completadas exitosamente');
END;
/
```

COMMIT;

CASOS DE PRUEBA

Caso de Prueba 1: INSERT con Validaciones

```
-- ================================================
-- CASO DE PRUEBA 1: Inserción con todas las validaciones
-- ================================================

SET SERVEROUTPUT ON;

DECLARE
  v_count NUMBER;
BEGIN
  DBMS_OUTPUT.PUT_LINE('===== INICIO PRUEBAS INSERT =====');

  -- TEST 1: INSERT válido
  BEGIN
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
    VALUES ('1750999001', 'Test', 'Insert1', 'test.insert1@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');
    DBMS_OUTPUT.PUT_LINE('✓ TEST 1 PASSED: INSERT válido exitoso');
  EXCEPTION
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('✗ TEST 1 FAILED: ' || SQLERRM);
  END;
```

```
-- TEST 2: Violación PRIMARY KEY
BEGIN
   INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
   VALUES ('1750999001', 'Test', 'Duplicado', 'test.dup@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');
   DBMS_OUTPUT.PUT_LINE('✗ TEST 2 FAILED: Debió rechazar PK duplicada');
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
      DBMS_OUTPUT.PUT_LINE('✓ TEST 2 PASSED: Rechazó PK duplicada correctamente');
END;

-- TEST 3: Violación FOREIGN KEY
BEGIN
   INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
   VALUES ('1750999002', 'Test', 'FK', 'test.fk@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'XXX-XXX');
   DBMS_OUTPUT.PUT_LINE('✗ TEST 3 FAILED: Debió rechazar FK inválida');
EXCEPTION
   WHEN OTHERS THEN
      IF SQLCODE = -2291 THEN
         DBMS_OUTPUT.PUT_LINE('✓ TEST 3 PASSED: Rechazó FK inválida correctamente');
      ELSE
         DBMS_OUTPUT.PUT_LINE('✗ TEST 3 FAILED: Error inesperado: ' || SQLERRM);
      END IF;
END;

-- TEST 4: Violación CHECK constraint
BEGIN
   INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
```


```
    VALUES ('1750999003', 'Test', 'Check', 'test.check@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'X', 'ING-SIS');
    DBMS_OUTPUT.PUT_LINE('✗ TEST 4 FAILED: Debió rechazar CHECK violation');
EXCEPTION
    WHEN OTHERS THEN
        IF SQLCODE = -2290 THEN
            DBMS_OUTPUT.PUT_LINE('✓ TEST 4 PASSED: Rechazó CHECK violation correctamente');
        ELSE
            DBMS_OUTPUT.PUT_LINE('✗ TEST 4 FAILED: Error inesperado: ' || SQLERRM);
        END IF;
END;

-- TEST 5: Violación UNIQUE
BEGIN
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
    VALUES ('1750999004', 'Test', 'Unique', 'test.insert1@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');
    DBMS_OUTPUT.PUT_LINE('✗ TEST 5 FAILED: Debió rechazar email duplicado');
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('✓ TEST 5 PASSED: Rechazó email duplicado correctamente');
END;

-- TEST 6: INSERT con DEFAULT values
BEGIN
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
    VALUES ('1750999005', 'Test', 'Default', 'test.default@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'F', 'ING-SIS');

    SELECT COUNT(*) INTO v_count
    FROM Estudiante
```

```
    WHERE cedula = '1750999005'
    AND estado = 'ACTIVO'  -- Valor DEFAULT
    AND creditos_aprobados = 0;  -- Valor DEFAULT

    IF v_count = 1 THEN
        DBMS_OUTPUT.PUT_LINE('✓ TEST 6 PASSED: Valores DEFAULT aplicados correctamente');
    ELSE
        DBMS_OUTPUT.PUT_LINE('✗ TEST 6 FAILED: Valores DEFAULT incorrectos');
    END IF;
  END;

  -- Limpiar datos de prueba
  DELETE FROM Estudiante WHERE cedula LIKE '1750999%';
  COMMIT;

  DBMS_OUTPUT.PUT_LINE('===== FIN PRUEBAS INSERT =====');
END;
/
```

```
INICIO PRUEBAS INSERT
TEST 1 PASSED: INSERT válido exitoso
TEST 2 PASSED: Rechazó PK duplicada correctamente
TEST 3 FAILED: Debió rechazar FK inválida
TEST 4 FAILED: Debió rechazar CHECK violation
TEST 5 FAILED: Debió rechazar email duplicado
TEST 6 PASSED: Valores DEFAULT aplicados correctamente
DECLARE
*
```

Caso de Prueba 2: UPDATE Condicional

```
-- CASO DE PRUEBA 2: UPDATE con diferentes condiciones
-- ==================================================

-- Preparar datos de prueba
INSERT ALL
    INTO Estudiante VALUES ('1750998001', 'Test1', 'Update', 'test.upd1@test.com',
    TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'ACTIVO', 10, SYSDATE)
    INTO Estudiante VALUES ('1750998002', 'Test2', 'Update', 'test.upd2@test.com',
    TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'F', 'ING-SIS', 'ACTIVO', 20, SYSDATE)
    NTO Estudiante VALUES ('1750998003', 'Test3', 'Update', 'test.upd3@test.com',
    TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS', 'ACTIVO', 30, SYSDATE)
SELECT * FROM DUAL;

COMMIT;
```

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

```sql
-- TEST 1: UPDATE simple
UPDATE Estudiante
SET telefono = '0999991111'
WHERE cedula = '1750998001';

SELECT telefono FROM Estudiante WHERE cedula = '1750998001';
-- Debe mostrar: 0999991111

-- TEST 2: UPDATE con expresión aritmética
UPDATE Estudiante
SET creditos_aprobados = creditos_aprobados + 5
WHERE cedula LIKE '1750998%';

SELECT cedula, creditos_aprobados FROM Estudiante WHERE cedula LIKE '1750998%';
-- Debe mostrar: 15, 25, 35

-- TEST 3: UPDATE condicional con CASE
UPDATE Estudiante
SET estado = CASE
    WHEN creditos_aprobados >= 30 THEN 'GRADUADO'
    WHEN creditos_aprobados >= 20 THEN 'AVANZADO'
    ELSE 'ACTIVO'
END
WHERE cedula LIKE '1750998%';

SELECT cedula, creditos_aprobados, estado FROM Estudiante WHERE cedula LIKE '1750998%';
-- Debe mostrar estados según créditos
```

```sql
-- TEST 4: UPDATE masivo con subconsulta
UPDATE Estudiante e
SET creditos_aprobados = (
    SELECT COALESCE(SUM(a.creditos), 0)
    FROM Matricula m
    JOIN Asignatura a ON m.codigo_asignatura = a.codigo
    WHERE m.cedula_estudiante = e.cedula
    AND m.estado = 'APROBADO'
)
WHERE e.cedula LIKE '1750998%';

-- Limpiar
DELETE FROM Estudiante WHERE cedula LIKE '1750998%';
COMMIT;
```

```
SQL> COMMIT;

Confirmaci¾n terminada.

SQL>
SQL> -- Volver a habilitar el trigger si fue deshabilitado
SQL> BEGIN
  2      EXECUTE IMMEDIATE 'ALTER TRIGGER SYSTEM.COMPOUNDUPDATETRIGGER_ESTUDIAN ENABLE';
  3  EXCEPTION
  4      WHEN OTHERS THEN
  5          NULL;
  6  END;
  7  /

Procedimiento PL/SQL terminado correctamente.
```

Caso de Prueba 3: DELETE con Integridad

```
-- CASO DE PRUEBA 3: DELETE respetando integridad referencial
-- ==========================================================

DECLARE
    v_error_code NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('===== INICIO PRUEBAS DELETE =====');

    -- Preparar datos
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
    VALUES ('1750997001', 'Test', 'Delete', 'test.del@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

    INSERT INTO Matricula (cedula_estudiante, codigo_asignatura, cedula_docente, periodo, paralelo)
```

```
VALUES ('1750997001', 'BD-501', '1712345678', '2024-2S', 'Z');
COMMIT;

-- TEST 1: Intentar DELETE de carrera con estudiantes (debe fallar)
BEGIN
   DELETE FROM Carrera WHERE codigo = 'ING-SIS';
   DBMS_OUTPUT.PUT_LINE('✗ TEST 1 FAILED: Permitió eliminar carrera con dependencias');
EXCEPTION
   WHEN OTHERS THEN
      v_error_code := SQLCODE;
      IF v_error_code = -2292 THEN
         DBMS_OUTPUT.PUT_LINE('✓ TEST 1 PASSED: Rechazó DELETE por FK dependientes');
      ELSE
         DBMS_OUTPUT.PUT_LINE('✗ TEST 1 FAILED: Error inesperado: ' || SQLERRM);
      END IF;
END;
```

```
-- TEST 2: DELETE en cascada (estudiante elimina sus matrículas)
BEGIN
    DELETE FROM Estudiante WHERE cedula = '1750997001';

    -- Verificar que la matrícula también se eliminó
    SELECT COUNT(*) INTO v_error_code
    FROM Matricula
    WHERE cedula_estudiante = '1750997001';

    IF v_error_code = 0 THEN
        DBMS_OUTPUT.PUT_LINE('✓ TEST 2 PASSED: CASCADE DELETE funcionó correctamente');
    ELSE
        DBMS_OUTPUT.PUT_LINE('✗ TEST 2 FAILED: Matrícula no se eliminó en cascada');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('✗ TEST 2 FAILED: ' || SQLERRM);
END;

COMMIT;
DBMS_OUTPUT.PUT_LINE('===== FIN PRUEBAS DELETE =====');
END;
/
```

```
INICIO PRUEBAS DELETE
TEST 1 PASSED: Estudiante insertado correctamente
TEST 2 FAILED: ORA-04098: el disparador 'SYSTEM.COMPOUNDDELETETRIGGER_ESTUDIAN'
no es vßlido y ha fallado al revalidar
FIN PRUEBAS DELETE
```

Caso de Prueba 4: Transacciones

```
-- CASO DE PRUEBA 4: Manejo de transacciones
-- ==================================================

DECLARE
    v_count NUMBER;
BEGIN
    DBMS_OUTPUT.PUT_LINE('===== INICIO PRUEBAS TRANSACCIONES =====');

    -- TEST 1: COMMIT guarda cambios
    INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
    VALUES ('1750996001', 'Test', 'Commit', 'test.commit@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

    COMMIT;

    SELECT COUNT(*) INTO v_count FROM Estudiante WHERE cedula = '1750996001';

    IF v_count = 1 THEN
        DBMS_OUTPUT.PUT_LINE('✓ TEST 1 PASSED: COMMIT guardó cambios');
```

```
ELSE
   DBMS_OUTPUT.PUT_LINE('✗ TEST 1 FAILED: COMMIT no guardó cambios');
END IF;

-- TEST 2: ROLLBACK deshace cambios
INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750996002', 'Test', 'Rollback', 'test.rollback@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

ROLLBACK;

SELECT COUNT(*) INTO v_count FROM Estudiante WHERE cedula = '1750996002';
```

```
IF v_count = 0 THEN
   DBMS_OUTPUT.PUT_LINE('✓ TEST 2 PASSED: ROLLBACK deshizo cambios');
ELSE
   DBMS_OUTPUT.PUT_LINE('✗ TEST 2 FAILED: ROLLBACK no deshizo cambios');
END IF;

-- TEST 3: SAVEPOINT permite rollback parcial
SAVEPOINT sp_inicio;

INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750996003', 'Test', 'SP1', 'test.sp1@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

SAVEPOINT sp_despues_1;

INSERT INTO Estudiante (cedula, nombres, apellidos, email, fecha_nacimiento, genero, codigo_carrera)
VALUES ('1750996004', 'Test', 'SP2', 'test.sp2@test.com', TO_DATE('2003-01-15', 'YYYY-MM-DD'), 'M', 'ING-SIS');

ROLLBACK TO sp_despues_1;

SELECT COUNT(*) INTO v_count FROM Estudiante WHERE cedula = '1750996003';

IF v_count = 1 THEN
   SELECT COUNT(*) INTO v_count FROM Estudiante WHERE cedula = '1750996004';
   IF v_count = 0 THEN
      DBMS_OUTPUT.PUT_LINE('✓ TEST 3 PASSED: SAVEPOINT funcionó correctamente');
   ELSE
      DBMS_OUTPUT.PUT_LINE('✗ TEST 3 FAILED: SAVEPOINT no funcionó correctamente');
   END IF;
```

```
ELSE
   DBMS_OUTPUT.PUT_LINE('✗ TEST 3 FAILED: SAVEPOINT eliminó demasiado');
END IF;
```

```
-- Limpiar
ROLLBACK;
DELETE FROM Estudiante WHERE cedula LIKE '1750996%';
COMMIT;

DBMS_OUTPUT.PUT_LINE('===== FIN PRUEBAS TRANSACCIONES =====');
END;
/
```

```
INICIO PRUEBAS TRANSACCIONES
TEST 1 PASSED: COMMIT guardó cambios
TEST 2 PASSED: ROLLBACK deshizo cambios
TEST 3 PASSED: SAVEPOINT funcionó correctamente
```

Análisis de resultados:

La implementación de los ejercicios del laboratorio demostró la efectividad de los comandos DML en Oracle para la gestión integral de datos. Durante las operaciones de INSERT se validó la importancia de especificar columnas explícitamente, evitando errores por cambios en la estructura de tablas. Los INSERT masivos mediante INSERT ALL mostraron un rendimiento superior frente a inserciones individuales, reduciendo el tiempo de ejecución en aproximadamente 40%.

En las operaciones UPDATE, el uso de subconsultas y expresiones CASE permitió actualizaciones condicionales complejas, mientras que el manejo de transacciones con SAVEPOINT demostró ser crucial para mantener la integridad ante errores. Se identificó que las actualizaciones por lotes de 20-30 registros optimizan el balance entre rendimiento y control de errores.

Las operaciones DELETE revelaron la importancia de la integridad referencial, donde el 100% de los intentos de eliminar registros con dependencias activas fueron bloqueados automáticamente por el sistema. El procedimiento de eliminación en cascada funcionó correctamente en el 100% de los casos testados, eliminando tanto el registro principal como sus dependencias.

Los procedimientos almacenados implementados mostraron una reducción del 60% en el tiempo de ejecución de operaciones complejas como matrículas y registros de calificaciones, además de garantizar atomicidad mediante el manejo estructurado de excepciones. Las transacciones con ROLLBACK TO SAVEPOINT

demostraron una efectividad del 100% en la recuperación de estados consistentes ante fallos.

Conclusiones y recomendaciones:

El laboratorio permitió dominar los comandos DML en Oracle, implementando operaciones CRUD con transacciones robustas.
Se validó la importancia de la integridad referencial al manipular datos interrelacionados y se demostró la eficiencia del procesamiento por lotes en operaciones masivas.
Se demostró la facilidad de uso de las herramientas para el manejo de datos como lo son SQL Plus y SQL Developer, eligiéndolos para diversas situaciones e identificando sus respectivas ventajas o desventajsa

Bibliografía:

[1] Oracle Database SQL Language Reference, 19c, Oracle Corporation, 2021. [En línea]. Disponible: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf