ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS

Laboratorio de:

Materia: Fundamentos de Bases de Datos

Práctica No.: LABORATORIO PRÁCTICO - TÓPICO 4

Tema: ESTRUCTURA DE UNA BD RELACIONAL - LENGUAJE DDL

TABLA DE CONTENIDOS

1. Objetivos
2. Requisitos
3. Caso de Estudio
4. Desarrollo Paso a Paso
5. Script Completo Oracle

OBJETIVOS

Objetivo General

Comprender y aplicar los comandos del Lenguaje de Definición de Datos (DDL) para crear, modificar y eliminar objetos en una base de datos Oracle, estableciendo correctamente las estructuras, restricciones e integridad referencial.

Objetivos Específicos

1. **Crear usuarios y esquemas** en Oracle con permisos apropiados
2. **Definir tablas** con tipos de datos Oracle apropiados
3. **Implementar restricciones de integridad** (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL, DEFAULT)
4. **Modificar estructuras existentes** mediante ALTER TABLE
5. **Crear y gestionar índices** para optimización de consultas
6. **Utilizar secuencias** para auto-incremento
7. **Eliminar objetos** de base de datos de forma segura (DROP, TRUNCATE)

REQUISITOS

Marco teórico:
Conceptos de normalización (1FN, 2FN, 3FN)

Forma Normal	Requisito Teórico Principal	Problema que Resuelve	Dependencia Funcional que Elimina
1NF	Los dominios de todos los atributos deben ser **atómicos**.	Grupos repetidos, valores multivaluados y atributos anidados.	(No aplica a DF, sino a la estructura del atributo).
2NF	- Estar en 1NF +	**Dependencias Parciales** (un atributo no-	Dependencias

	- Todo atributo **no-clave** debe tener una **dependencia funcional completa** de la clave primaria.	clave depende de *parte* de una clave primaria compuesta).	parciales.
3NF	- Estar en 2NF + - Ningún atributo **no-clave** puede tener una **dependencia transitiva** de la clave primaria.	**Dependencias Transitivas** (un atributo no-clave depende de *otro* atributo no-clave: PK -> NoClave_A -> NoClave_B).	Dependencias transitivas.

Modelo Entidad-Relación

Modelo conceptual (Entidad – Relación)
Su función principal es representar las entidades del dominio del negocio, contado con: sus atributos esenciales y relaciones entre los atributos, sin preocuparse sobre la tecnología o detalles de implementación.
El fin de esta función es reflejar el "mundo real" la situación en la que los usuarios y analistas estén de acuerdo con el modelo, siendo el puente entre los desarrolladores y el cliente.
Para representar este modelo se utilizan diagramas ER (Entity-Relationship) .

Conceptos Clave de Bases de Datos

A continuación, se detallan los conceptos que solicitaste.

Cardinalidad de Relaciones

La cardinalidad describe la relación numérica entre dos entidades en una base de datos.

Cardinalidad	Nombre	Descripción	Ejemplo
1:1	Uno a Uno	Una instancia de la Entidad A solo puede estar relacionada con una instancia de la Entidad B, y viceversa.	Usuario y PerfilDeUsuario. Un usuario tiene un solo perfil, y un perfil pertenece a un solo usuario.
1:N	Uno a Muchos	Una instancia de la Entidad A puede estar relacionada con muchas instancias de la Entidad B, pero una instancia de la Entidad B solo puede estar relacionada con una instancia de la Entidad A.	Cliente y Pedido. Un cliente puede tener muchos pedidos, pero un pedido pertenece a un solo cliente.
N:N	Muchos a Muchos	Una instancia de la Entidad A puede estar relacionada con	Estudiante y Curso. Un estudiante puede inscribirse en muchos

		muchas instancias de la Entidad B, y viceversa. Este tipo de relación generalmente requiere una tabla intermedia (tabla de unión) para implementarse.	cursos, y un curso puede tener muchos estudiantes.

Claves Primarias y Foráneas

Clave Primaria (Primary Key - PK)

Es una columna (o conjunto de columnas) que identifica de forma **única** cada fila en una tabla. No puede contener valores nulos (NULL) y sus valores no deben repetirse en toda la tabla. Es el identificador principal de un registro.

Clave Foránea (Foreign Key - FK)

Es una columna (o conjunto de columnas) en una tabla que establece un **enlace** con la clave primaria de otra tabla (o la misma). Se utiliza para mantener la integridad referencial de los datos, asegurando que un valor en la tabla "hija" exista en la tabla "padre".

Tipos de Datos Básicos

Requisitos Técnicos

Tipo de Dato	Descripción	Ejemplos
INT (o INTEGER)	Números enteros, sin decimales.	1, 100, -45
VARCHAR(n)	Cadena de texto de longitud variable, donde n es el máximo número de caracteres.	"Hola", "Juan Pérez"
CHAR(n)	Cadena de texto de longitud fija, donde n es el número exacto de caracteres.	"S", "MX"
DECIMAL(p, s) (o NUMERIC)	Números exactos con decimales. p es la precisión (total de dígitos) y s es la escala (dígitos después del punto decimal).	123.45, 99.99
FLOAT (o REAL)	Números de punto flotante (decimales aproximados).	10.5e3
DATE	Almacena una fecha (Año, Mes, Día).	2025-11-13
DATETIME (o TIMESTAMP)	Almacena una fecha y hora (Año, Mes, Día, Hora, Minuto, Segundo).	2025-11-13 23:30:00
BOOLEAN (o BIT)	Almacena valores de verdadero o falso.	true, false, 1, 0
TEXT (o CLOB)	Almacena cadenas de texto de gran longitud.	Un artículo de blog, una descripción larga.
BLOB	Almacena datos binarios de gran tamaño (Binary Large Object).	Imágenes, archivos de audio, PDFs.

Software requerido:

- Oracle Database 11g o superior
- SQL Developer o SQL*Plus
- Acceso con privilegios SYSDBA o DBA

Material de Apoyo

- Diagrama ER del caso de estudio
- Diccionario de datos
- Documentación de sintaxis DDL de Oracle

CASO DE ESTUDIO

Sistema de Gestión Académica Universitaria

Descripción del Dominio

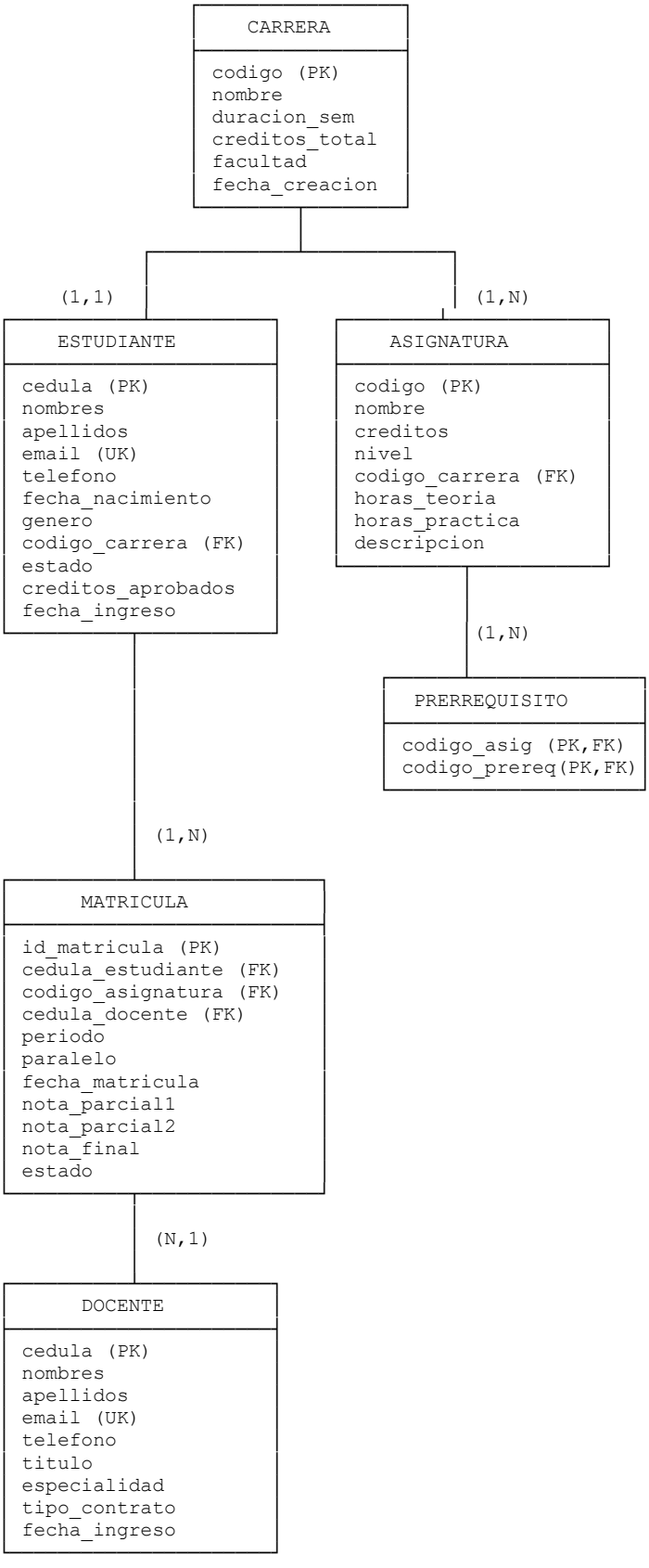
La Escuela Politécnica Nacional requiere un sistema para gestionar la información académica de sus estudiantes, docentes, carreras y asignaturas. El sistema debe permitir:

- Registrar información de estudiantes matriculados en diferentes carreras
- Gestionar el catálogo de asignaturas organizadas por carrera y nivel
- Asignar docentes a asignaturas en diferentes periodos académicos
- Registrar las matrículas de estudiantes en asignaturas
- Almacenar las calificaciones finales de cada estudiante
- Controlar prerrequisitos entre asignaturas

Reglas de Negocio

1. Cada estudiante pertenece a una única carrera
2. Una asignatura pertenece a una carrera específica y tiene un nivel definido
3. Una asignatura puede tener múltiples prerrequisitos
4. Un docente puede impartir múltiples asignaturas en diferentes periodos
5. Un estudiante puede matricularse en múltiples asignaturas por periodo
6. Las notas finales deben estar entre 0 y 10
7. Se considera aprobada una asignatura con nota >= 7.0
8. Los estados de matrícula válidos son: CURSANDO, APROBADO, REPROBADO, RETIRADO
9. Un estudiante no puede matricularse dos veces en la misma asignatura en el mismo periodo
10. El periodo académico tiene formato: YYYY-NS (ejemplo: 2024-1S, 2024-2S)

DIAGRAMA ENTIDAD-RELACIÓN


CARRERA
codigo (PK)
nombre
duracion_sem
creditos_total
facultad
fecha_creacion
(1,1)
(1,N)
ESTUDIANTE
cedula (PK)
nombres
apellidos
email (UK)
telefono
fecha_nacimiento
genero
codigo_carrera (FK)
estado
creditos_aprobados
fecha_ingreso
ASIGNATURA
codigo (PK)
nombre
creditos
nivel
codigo_carrera (FK)
horas_teoria
horas_practica
descripcion
(1,N)
PRERREQUISITO
codigo_asig (PK,FK)
codigo_prereq(PK,FK)
(1,N)
MATRICULA
id_matricula (PK)
cedula_estudiante (FK)
codigo_asignatura (FK)
cedula_docente (FK)
periodo
paralelo
fecha_matricula
nota_parcial1
nota_parcial2
nota_final
estado
(N,1)
DOCENTE
cedula (PK)
nombres
apellidos
email (UK)
telefono
titulo
especialidad
tipo_contrato
fecha_ingreso

Leyenda:

PK = Primary Key (Clave Primaria)

FK = Foreign Key (Clave Foránea)

UK = Unique Key (Clave Única)

(1,1) = Cardinalidad uno a uno

(1,N) = Cardinalidad uno a muchos

(N,M) = Cardinalidad muchos a muchos

DICCIONARIO DE DATOS

Tabla: CARRERA

Campo	Tipo de Dato Oracle	Restricciones	Descripción
codigo	VARCHAR2(10)	PK, NOT NULL	Código único de carrera (ej: ING-SIS)
nombre	VARCHAR2(100)	NOT NULL, UNIQUE	Nombre completo de la carrera
duracion_semestres	NUMBER(2)	NOT NULL, CHECK (8-12)	Duración en semestres
creditos_totales	NUMBER(3)	NOT NULL, CHECK (>0)	Total de créditos para graduarse
facultad	VARCHAR2(100)	NOT NULL	Facultad a la que pertenece
fecha_creacion	DATE	DEFAULT SYSDATE	Fecha de creación del registro

Tabla: ESTUDIANTE

Campo	Tipo de Dato Oracle	Restricciones	Descripción
cedula	VARCHAR2(10)	PK, NOT NULL, CHECK (10 dígitos)	Cédula de identidad ecuatoriana
nombres	VARCHAR2(50)	NOT NULL	Nombres del estudiante
apellidos	VARCHAR2(50)	NOT NULL	Apellidos del estudiante
email	VARCHAR2(100)	NOT NULL, UNIQUE	Correo institucional
telefono	VARCHAR2(15)	NULL	Teléfono de contacto
fecha_nacimiento	DATE	NOT NULL, CHECK (edad>=16)	Fecha de nacimiento
genero	CHAR(1)	NOT NULL, CHECK ('M','F','O')	Género del estudiante
codigo_carrera	VARCHAR2(10)	FK → Carrera, NOT NULL	Carrera del estudiante
estado	VARCHAR2(20)	DEFAULT 'ACTIVO', CHECK	Estado actual del estudiante
creditos_aprobados	NUMBER(3)	DEFAULT 0, CHECK (>=0)	Créditos acumulados
fecha_ingreso	DATE	DEFAULT SYSDATE	Fecha de ingreso a la EPN

Valores válidos para estado: ACTIVO, INACTIVO, GRADUADO, RETIRADO

Tabla: ASIGNATURA

Campo	Tipo de Dato Oracle	Restricciones	Descripción
codigo	VARCHAR2(10)	PK, NOT NULL	Código único (ej: BD-101)
nombre	VARCHAR2(100)	NOT NULL	Nombre de la asignatura

creditos	NUMBER(1)	NOT NULL, CHECK (1-8)	Número de créditos
nivel	NUMBER(2)	NOT NULL, CHECK (1-10)	Nivel o semestre
codigo_carrera	VARCHAR2(10)	FK → Carrera, NOT NULL	Carrera a la que pertenece
horas_teoria	NUMBER(2)	DEFAULT 0, NOT NULL	Horas teóricas semanales
horas_practica	NUMBER(2)	DEFAULT 0, NOT NULL	Horas prácticas semanales
descripcion	CLOB	NULL	Descripción y objetivos

Restricción adicional: horas_teoria + horas_practica > 0

Tabla: DOCENTE

Campo	Tipo de Dato Oracle	Restricciones	Descripción
cedula	VARCHAR2(10)	PK, NOT NULL, CHECK (10 dígitos)	Cédula de identidad
nombres	VARCHAR2(50)	NOT NULL	Nombres del docente
apellidos	VARCHAR2(50)	NOT NULL	Apellidos del docente
email	VARCHAR2(100)	NOT NULL, UNIQUE	Correo institucional
telefono	VARCHAR2(15)	NULL	Teléfono de contacto
titulo	VARCHAR2(50)	NOT NULL	Título académico máximo
especialidad	VARCHAR2(100)	NULL	Área de especialización
tipo_contrato	VARCHAR2(20)	DEFAULT 'TIEMPO_COMPLETO'	Tipo de contrato
fecha_ingreso	DATE	DEFAULT SYSDATE	Fecha de ingreso

Valores válidos para tipo_contrato: TIEMPO_COMPLETO, MEDIO_TIEMPO, HORA_CLASE

Tabla: MATRICULA

Campo	Tipo de Dato Oracle	Restricciones	Descripción
id_matricula	NUMBER(10)	PK, SEQUENCE	ID único generado
cedula_estudiante	VARCHAR2(10)	FK → Estudiante, NOT NULL	Estudiante matriculado
codigo_asignatura	VARCHAR2(10)	FK → Asignatura, NOT NULL	Asignatura matriculada
cedula_docente	VARCHAR2(10)	FK → Docente, NOT NULL	Docente asignado
periodo	VARCHAR2(10)	NOT NULL, CHECK (formato)	Periodo académico
paralelo	CHAR(1)	NOT NULL, CHECK (A-Z)	Paralelo
fecha_matricula	DATE	DEFAULT SYSDATE	Fecha de matrícula
nota_parcial1	NUMBER(4,2)	NULL, CHECK (0-10)	Primera nota parcial
nota_parcial2	NUMBER(4,2)	NULL, CHECK (0-10)	Segunda nota parcial
nota_final	NUMBER(4,2)	NULL, CHECK (0-10)	Nota final
estado	VARCHAR2(20)	DEFAULT 'CURSANDO'	Estado de la matrícula

Restricciones adicionales:

- UNIQUE (cedula_estudiante, codigo_asignatura, periodo)
- Si estado='CURSANDO' → nota_final IS NULL
- Si estado IN ('APROBADO','REPROBADO') → nota_final IS NOT NULL
- Formato periodo: `^\d{4}-[12]S$` (ej: 2024-1S)

Tabla: PRERREQUISITO

Campo	Tipo de Dato Oracle	Restricciones	Descripción
codigo_asignatura	VARCHAR2(10)	PK, FK → Asignatura	Asignatura que requiere
codigo_prerrequisito	VARCHAR2(10)	PK, FK → Asignatura	Asignatura prerrequisito

Restricción adicional: codigo_asignatura ≠ codigo_prerrequisito

---

DESARROLLO DE LA PRACTICA

PASO 1: Crear Usuario y Esquema en Oracle

```
-- ============================================================
-- CREACIÓN DE USUARIO/ESQUEMA - ORACLE
-- ============================================================
-- Conectarse como SYSDBA
-- sqlplus / as sysdba

-- Paso 1.1: Verificar y eliminar usuario si existe
DROP USER gestion_academica CASCADE;
```

```
SQL> DROP USER gestion_academica CASCADE;
DROP USER gestion_academica CASCADE
          *
ERROR en lÝnea 1:
ORA-01918: el usuario 'GESTION_ACADEMICA' no existe
```

En primera instancia no existe

```
-- Paso 1.2: Crear nuevo usuario
CREATE USER gestion_academica IDENTIFIED BY EPN2024Secure
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
PROFILE DEFAULT;
```

```
SQL> show user
USER es "SYSTEM"
SQL> CREATE USER gestion_academica IDENTIFIED BY EPN2024Secure
  2  DEFAULT TABLESPACE USERS
  3  TEMPORARY TABLESPACE TEMP
  4  PROFILE DEFAULT;
```

```
-- Paso 1.3: Otorgar privilegios de sistema
GRANT CONNECT TO gestion_academica;
GRANT RESOURCE TO gestion_academica;
GRANT CREATE VIEW TO gestion_academica;
GRANT CREATE SEQUENCE TO gestion_academica;
GRANT CREATE SYNONYM TO gestion_academica;
GRANT CREATE TRIGGER TO gestion_academica;
GRANT CREATE PROCEDURE TO gestion_academica;
```

```
SQL> GRANT CONNECT TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT RESOURCE TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT CREATE VIEW TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT CREATE SEQUENCE TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT CREATE SYNONYM TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT CREATE TRIGGER TO gestion_academica;

Concesi¾n terminada correctamente.

SQL> GRANT CREATE PROCEDURE TO gestion_academica;

Concesi¾n terminada correctamente.
```

```
-- Paso 1.4: Otorgar cuota en tablespace
ALTER USER gestion_academica QUOTA UNLIMITED ON USERS;
```

```
SQL> ALTER USER gestion_academica QUOTA UNLIMITED ON USERS;

SQL>
```

```
-- Paso 1.5: Verificar usuario creado
SELECT username, account_status, default_tablespace, created
FROM dba_users
WHERE username = 'GESTION_ACADEMICA';
```

```
SQL> SELECT username, account_status, default_tablespace, create
d FROM dba_users
  2  WHERE username = 'GESTION_ACADEMICA';

USERNAME
--------------------------------------------------------------------
----------------
ACCOUNT_STATUS                   DEFAULT_TABLESPACE             C
REATED
-------------------------------- ------------------------------
--------
GESTION_ACADEMICA
OPEN                             USERS                          0
7/11/25
```

```
-- Paso 1.6: Conectarse como el nuevo usuario
CONN gestion_academica/EPN2024Secure;
```

```
SQL> CONN gestion_academica/EPN2024Secure;
Conectado.
```

```
-- Paso 1.7: Verificar conexión actual
SELECT USER FROM DUAL;
SELECT SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA') AS esquema_actual FROM DUAL;
```

```
SQL> SELECT USER FROM DUAL;

USER
------------------------------------------------------------------
----------------
GESTION_ACADEMICA

SQL> SELECT SYS_CONTEXT('USERENV', 'CURRENT_SCHEMA') AS esquema_
actual FROM DUAL;

ESQUEMA_ACTUAL
------------------------------------------------------------------
----------------
GESTION_ACADEMICA
```

PASO 2: Crear Tabla CARRERA

```
-- ========================================================
-- TABLA CARRERA - ORACLE
-- ========================================================

-- Eliminar tabla si existe (para pruebas)
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Carrera CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/
```

```
SQL> BEGIN
  2  EXECUTE IMMEDIATE 'DROP TABLE Carrera CASCADE CONSTRAINTS';
 EXCEPTION
  3  WHEN OTHERS THEN
  4  IF SQLCODE != -942 THEN RAISE; END IF; END;
  5  /

Procedimiento PL/SQL terminado correctamente.
```

```
-- Crear tabla CARRERA
CREATE TABLE Carrera (
   codigo VARCHAR2(10) NOT NULL,
   nombre VARCHAR2(100) NOT NULL,
   duracion_semestres NUMBER(2) NOT NULL,
   creditos_totales NUMBER(3) NOT NULL,
   facultad VARCHAR2(100) NOT NULL,
   fecha_creacion DATE DEFAULT SYSDATE NOT NULL,
   -- Restricción de clave primaria
   CONSTRAINT pk_carrera PRIMARY KEY (codigo),
   -- Restricción de unicidad
   CONSTRAINT uk_carrera_nombre UNIQUE (nombre),
   -- Restricciones de validación
   CONSTRAINT chk_carrera_duracion
      CHECK (duracion_semestres BETWEEN 8 AND 12),
   CONSTRAINT chk_carrera_creditos
      CHECK (creditos_totales > 0)
```

```
);
```

```
SQL> CREATE TABLE Carrera (
  2  codigo VARCHAR2(10) NOT NULL, nombre VARCHAR2(100) NOT NULL
,
  3  duracion_semestres NUMBER(2) NOT NULL, creditos_totales NUM
BER(3) NOT NULL, facultad VARCHAR2(100) NOT NULL,
  4  fecha_creacion DATE DEFAULT SYSDATE NOT NULL,
  5  -- Restricción de clave primaria
  6  CONSTRAINT pk_carrera PRIMARY KEY (codigo),
  7  -- Restricción de unicidad
  8  CONSTRAINT uk_carrera_nombre UNIQUE (nombre),
  9  -- Restricciones de validación
 10  CONSTRAINT chk_carrera_duracion
 11  CHECK (duracion_semestres BETWEEN 8 AND 12),
 12  CONSTRAINT chk_carrera_creditos CHECK (creditos_totales > 0
)
 13  );

Tabla creada.
```

```
-- Agregar comentarios a la tabla
COMMENT ON TABLE Carrera IS
   'Catálogo de carreras ofertadas por la institución';
COMMENT ON COLUMN Carrera.codigo IS
   'Código único de la carrera (ej: ING-SIS, ING-CIV)';
COMMENT ON COLUMN Carrera.nombre IS
```

```
    'Nombre completo de la carrera';
COMMENT ON COLUMN Carrera.duracion_semestres IS
    'Duración de la carrera en semestres académicos';
COMMENT ON COLUMN Carrera.creditos_totales IS
    'Total de créditos necesarios para graduarse';
COMMENT ON COLUMN Carrera.facultad IS
    'Facultad a la que pertenece la carrera';
```

```
SQL> COMMENT ON TABLE Carrera IS
  2  'Catálogo de carreras ofertadas por la institución';

Comentario creado.

SQL> COMMENT ON COLUMN Carrera.codigo IS
  2  'Código único de la carrera (ej: ING-SIS, ING-CIV)'; COMMEN
T ON COLUMN Carrera.nombre IS
  3
SQL> 'Nombre completo de la carrera';
SP2-0734: inicio "'Nombre co..." de comando desconocido - resto
de la lÝnea ignorado.
SQL> COMMENT ON COLUMN Carrera.duracion_semestres IS 'Duración d
e la carrera en semestres académicos';

Comentario creado.

SQL> COMMENT ON COLUMN Carrera.creditos_totales IS 'Total de cré
ditos necesarios para graduarse';

Comentario creado.

SQL> COMMENT ON COLUMN Carrera.facultad IS 'Facultad a la que pe
rtenece la carrera';

Comentario creado.
```

```
-- Verificar creación
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

SQL>
```

```
-- Verificar restricciones
SELECT constraint_name, constraint_type, search_condition
FROM user_constraints
WHERE table_name = 'CARRERA'
ORDER BY constraint_type;
```

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1	SYS_C007671	C	"NOMBRE" IS NOT NULL
2	SYS_C007672	C	"DURACION_SEMESTRES" IS NOT NULL
3	SYS_C007673	C	"CREDITOS_TOTALES" IS NOT NULL
4	SYS_C007670	C	"CODIGO" IS NOT NULL
5	SYS_C007674	C	"FACULTAD" IS NOT NULL
6	CHK_CARRERA_CREDITOS	C	creditos_totales > 0
7	SYS_C007675	C	"FECHA_CREACION" IS NOT NULL
8	CHK_CARRERA_DURACION	C	duracion_semestres BETWEEN 8 AND 12
9	PK_CARRERA	P	(null)
10	UK_CARRERA_NOMBRE	U	(null)

PASO 3: Crear Tabla ESTUDIANTE

```
-- ================================================
-- TABLA ESTUDIANTE - ORACLE
```

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

-- Eliminar tabla si existe
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Estudiante CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/

-- Crear tabla ESTUDIANTE
CREATE TABLE Estudiante (
   cedula VARCHAR2(10) NOT NULL,
   nombres VARCHAR2(50) NOT NULL,
   apellidos VARCHAR2(50) NOT NULL,
   email VARCHAR2(100) NOT NULL,
   telefono VARCHAR2(15),
   fecha_nacimiento DATE NOT NULL,
   genero CHAR(1) NOT NULL,
   codigo_carrera VARCHAR2(10) NOT NULL,
   estado VARCHAR2(20) DEFAULT 'ACTIVO' NOT NULL,
   creditos_aprobados NUMBER(3) DEFAULT 0 NOT NULL,
   fecha_ingreso DATE DEFAULT SYSDATE NOT NULL,
   -- Clave primaria
   CONSTRAINT pk_estudiante PRIMARY KEY (cedula),
   -- Unicidad de email
   CONSTRAINT uk_estudiante_email UNIQUE (email),
   -- Validación de cédula (10 dígitos ecuatorianos)
   CONSTRAINT chk_estudiante_cedula
      CHECK (REGEXP_LIKE(cedula, '^\d{10}$')),
   -- Validación de género
   CONSTRAINT chk_estudiante_genero
      CHECK (genero IN ('M', 'F', 'O')),
   -- Validación de estado
   CONSTRAINT chk_estudiante_estado
      CHECK (estado IN ('ACTIVO', 'INACTIVO', 'GRADUADO', 'RETIRADO')),
   -- Validación de créditos
```

```
    CONSTRAINT chk_estudiante_creditos
        CHECK (creditos_aprobados >= 0 AND creditos_aprobados <= 300),
    -- Validación de edad (mayor de 16 años) a través de trigger
    -- Clave foránea a Carrera
    CONSTRAINT fk_estudiante_carrera
        FOREIGN KEY (codigo_carrera)
        REFERENCES Carrera(codigo)  );
```

```
  4  telefono VARCHAR2(15), fecha_nacimiento DATE NOT NULL, gene
ro CHAR(1) NOT NULL,
  5  codigo_carrera VARCHAR2(10) NOT NULL,
  6  estado VARCHAR2(20) DEFAULT 'ACTIVO' NOT NULL,
  7  creditos_aprobados NUMBER(3) DEFAULT 0 NOT NULL, fecha_ingr
eso DATE DEFAULT SYSDATE NOT NULL,
  8  -- Clave primaria
  9  CONSTRAINT pk_estudiante PRIMARY KEY (cedula),
 10  -- Unicidad de email
 11  CONSTRAINT uk_estudiante_email UNIQUE (email),
 12  -- Validación de cédula (10 dígitos ecuatorianos)
 13  CONSTRAINT chk_estudiante_cedula
 14  CHECK (REGEXP_LIKE(cedula, '^\d{10}$')),
 15  -- Validación de género
 16  CONSTRAINT chk_estudiante_genero CHECK (genero IN ('M', 'F'
, 'O')),
 17  -- Validación de estado
 18  CONSTRAINT chk_estudiante_estado
 19  CHECK (estado IN ('ACTIVO', 'INACTIVO', 'GRADUADO', 'RETIRA
DO')),
 20  -- Validación de créditos
 21  CONSTRAINT chk_estudiante_creditos
 22  CHECK (creditos_aprobados >= 0 AND creditos_aprobados <= 30
0),
 23  -- Validación de edad (mayor de 16 años) a través de trigge
r
 24  -- Clave foránea a Carrera
 25  CONSTRAINT fk_estudiante_carrera FOREIGN KEY (codigo_carrer
a) REFERENCES Carrera(codigo) );

Tabla creada.
```

```
-- Índices para optimizar búsquedas frecuentes
CREATE INDEX idx_estudiante_apellidos
    ON Estudiante(apellidos);

CREATE INDEX idx_estudiante_carrera
    ON Estudiante(codigo_carrera);

CREATE INDEX idx_estudiante_estado
    ON Estudiante(estado);
```

```
SQL> CREATE INDEX idx_estudiante_apellidos ON Estudiante(apellid
os);

=ndice creado.

SQL>
SQL> CREATE INDEX idx_estudiante_carrera ON Estudiante(codigo_ca
rrera);

=ndice creado.

SQL>
SQL> CREATE INDEX idx_estudiante_estado ON Estudiante(estado);

=ndice creado.
```

```
-- Índice compuesto para búsquedas combinadas
CREATE INDEX idx_estudiante_apellidos_carrera
    ON Estudiante(apellidos, codigo_carrera);
```

```
SQL>
SQL> CREATE INDEX idx_estudiante_estado ON Estudiante(estado);

=ndice creado.

SQL> CREATE INDEX idx_estudiante_apellidos_carrera ON Estudiante
(apellidos, codigo_carrera);

=ndice creado.
```

```
-- Agregar comentarios
```

```
COMMENT ON TABLE Estudiante IS
    'Información personal y académica de estudiantes matriculados en la institución';
COMMENT ON COLUMN Estudiante.cedula IS
    'Cédula de identidad ecuatoriana (10 dígitos)';
COMMENT ON COLUMN Estudiante.email IS
    'Correo electrónico institucional único';
COMMENT ON COLUMN Estudiante.creditos_aprobados IS
    'Total de créditos acumulados hasta la fecha';
COMMENT ON COLUMN Estudiante.estado IS
    'Estado actual del estudiante: ACTIVO, INACTIVO, GRADUADO, RETIRADO';
```

```
SQL> COMMENT ON TABLE Estudiante IS
  2  'Información personal y académica de estudiantes matriculad
os en la institución';

Comentario creado.

SQL> COMMENT ON COLUMN Estudiante.cedula IS 'Cédula de identidad
 ecuatoriana (10 dígitos)';

Comentario creado.

SQL> COMMENT ON COLUMN Estudiante.email IS 'Correo electrónico i
nstitucional único';

Comentario creado.

SQL> COMMENT ON COLUMN Estudiante.creditos_aprobados IS 'Total d
e créditos acumulados hasta la fecha';

Comentario creado.

SQL> COMMENT ON COLUMN Estudiante.estado IS
  2  'Estado actual del estudiante: ACTIVO, INACTIVO, GRADUADO,
RETIRADO';

Comentario creado.

SQL>
```

-- Validar que la edad sea al menos 16 años

```
CREATE OR REPLACE TRIGGER trg_validar_edad
BEFORE INSERT OR UPDATE ON Estudiante
FOR EACH ROW
BEGIN
  -- Validar que la edad sea al menos 16 años
  IF MONTHS_BETWEEN(SYSDATE, :NEW.fecha_nacimiento) / 12 < 16 THEN
    RAISE_APPLICATION_ERROR(-20001, 'El estudiante debe tener al menos 16 años.');
  END IF;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER trg_validar_edad
  2  BEFORE INSERT OR UPDATE ON Estudiante
  3  FOR EACH ROW
  4  BEGIN
  5    -- Validar que la edad sea al menos 16 años
  6    IF MONTHS_BETWEEN(SYSDATE, :NEW.fecha_nacimiento) / 12 <
16 THEN
  7      RAISE_APPLICATION_ERROR(-20001, 'El estudiante debe ten
er al menos 16 años.');
  8    END IF;
  9  END;
 10  /

Disparador creado.
```

```
-- Verificar creación
DESC Estudiante;
```

```
SQL> DESC Estudiante
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(100)
 TELEFONO                                           VARCHAR2(15)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE

SQL>
```

PASO 4: Crear Tabla ASIGNATURA

sql

```
-- ============================================================
-- TABLA ASIGNATURA - ORACLE
-- ============================================================

-- Eliminar tabla si existe
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Asignatura CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/
```

```
SQL> BEGIN
  2  EXECUTE IMMEDIATE 'DROP TABLE Asignatura CASCADE CONSTRAINTS'; EXCEPTION
  3  WHEN OTHERS THEN
  4  IF SQLCODE != -942 THEN RAISE; END IF; END;
  5  /

Procedimiento PL/SQL terminado correctamente.
```

```
-- Crear tabla ASIGNATURA
CREATE TABLE Asignatura (
    codigo VARCHAR2(10) NOT NULL,
    nombre VARCHAR2(100) NOT NULL,
    creditos NUMBER(1) NOT NULL,
    nivel NUMBER(2) NOT NULL,
    codigo_carrera VARCHAR2(10) NOT NULL,
    horas_teoria NUMBER(2) DEFAULT 0 NOT NULL,
    horas_practica NUMBER(2) DEFAULT 0 NOT NULL,
    descripcion CLOB,
    -- Clave primaria
    CONSTRAINT pk_asignatura PRIMARY KEY (codigo),
    -- Unicidad de nombre por carrera
    CONSTRAINT uk_asignatura_nombre_carrera
        UNIQUE (nombre, codigo_carrera),
    -- Validación de créditos
    CONSTRAINT chk_asignatura_creditos
        CHECK (creditos BETWEEN 1 AND 8),
    -- Validación de nivel
    CONSTRAINT chk_asignatura_nivel
        CHECK (nivel BETWEEN 1 AND 10),
    -- Validación de horas
    CONSTRAINT chk_asignatura_horas
        CHECK (horas_teoria + horas_practica > 0),
    -- Clave foránea a Carrera con CASCADE
    CONSTRAINT fk_asignatura_carrera
        FOREIGN KEY (codigo_carrera)
        REFERENCES Carrera(codigo)
        ON DELETE CASCADE
);
```

```
SQL> CREATE TABLE Asignatura (
  2  codigo VARCHAR2(10) NOT NULL, nombre VARCHAR2(100) NOT NULL
,
  3  creditos NUMBER(1) NOT NULL, nivel NUMBER(2) NOT NULL,
  4  codigo_carrera VARCHAR2(10) NOT NULL, horas_teoria NUMBER(2
) DEFAULT 0 NOT NULL, horas_practica NUMBER(2) DEFAULT 0 NOT NUL
L, descripcion CLOB,
  5  -- Clave primaria
  6  CONSTRAINT pk_asignatura PRIMARY KEY (codigo),
  7  -- Unicidad de nombre por carrera
  8  CONSTRAINT uk_asignatura_nombre_carrera UNIQUE (nombre, cod
igo_carrera),
  9  -- Validación de créditos
 10  CONSTRAINT chk_asignatura_creditos CHECK (creditos BETWEEN
1 AND 8),
 11  -- Validación de nivel
 12  CONSTRAINT chk_asignatura_nivel CHECK (nivel BETWEEN 1 AND
10),
 13  -- Validación de horas
 14  CONSTRAINT chk_asignatura_horas
 15  CHECK (horas_teoria + horas_practica > 0),
 16  -- Clave foránea a Carrera con CASCADE
 17  CONSTRAINT fk_asignatura_carrera FOREIGN KEY (codigo_carrer
a) REFERENCES Carrera(codigo) ON DELETE CASCADE
 18  );

Tabla creada.
```

```
-- Índices
CREATE INDEX idx_asignatura_carrera ON Asignatura(codigo_carrera);
CREATE INDEX idx_asignatura_nivel ON Asignatura(nivel);
CREATE INDEX idx_asignatura_carrera_nivel ON Asignatura(codigo_carrera, nivel);

-- Comentarios
COMMENT ON TABLE Asignatura IS
   'Catálogo de asignaturas organizadas por carrera y nivel';
COMMENT ON COLUMN Asignatura.codigo IS
   'Código único de asignatura (ej: BD-101, MAT-201)';
COMMENT ON COLUMN Asignatura.creditos IS
   'Número de créditos académicos que otorga la asignatura';
```

```
COMMENT ON COLUMN Asignatura.nivel IS
  'Nivel o semestre en el que se cursa normalmente';
```

```
SQL> COMMENT ON TABLE Asignatura IS
  2  'Catálogo de asignaturas organizadas por carrera y nivel';

Comentario creado.

SQL> COMMENT ON COLUMN Asignatura.codigo IS 'Código único de asi
gnatura (ej: BD-101, MAT-201)';

Comentario creado.

SQL> COMMENT ON COLUMN Asignatura.creditos IS
  2  'Número de créditos académicos que otorga la asignatura';

Comentario creado.

SQL> COMMENT ON COLUMN Asignatura.nivel IS 'Nivel o semestre en
el que se cursa normalmente';

Comentario creado.
```

```
-- Verificar creación
DESC Asignatura;
```

```
SQL> DESC Asignatura
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CODIGO                                    NOT NULL VARCHAR2(10)
 NOMBRE                                    NOT NULL VARCHAR2(100)
 CREDITOS                                  NOT NULL NUMBER(1)
 NIVEL                                     NOT NULL NUMBER(2)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 HORAS_TEORIA                              NOT NULL NUMBER(2)
 HORAS_PRACTICA                            NOT NULL NUMBER(2)
 DESCRIPCION                                        CLOB
```

PASO 5: Crear Tabla DOCENTE

sql

```
-- ========================================================
-- TABLA DOCENTE - ORACLE
-- ========================================================

-- Eliminar tabla si existe
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Docente CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/

-- Crear tabla DOCENTE
CREATE TABLE Docente (
   cedula VARCHAR2(10) NOT NULL,
   nombres VARCHAR2(50) NOT NULL,
   apellidos VARCHAR2(50) NOT NULL,
   email VARCHAR2(100) NOT NULL,
   telefono VARCHAR2(15),
   titulo VARCHAR2(50) NOT NULL,
   especialidad VARCHAR2(100),
   tipo_contrato VARCHAR2(20) DEFAULT 'TIEMPO_COMPLETO' NOT NULL,
   fecha_ingreso DATE DEFAULT SYSDATE NOT NULL,
   -- Clave primaria
   CONSTRAINT pk_docente PRIMARY KEY (cedula),
   -- Unicidad de email
   CONSTRAINT uk_docente_email UNIQUE (email),
   -- Validación de cédula
   CONSTRAINT chk_docente_cedula
      CHECK (REGEXP_LIKE(cedula, '^\d{10}$')),
   -- Validación de tipo de contrato
   CONSTRAINT chk_docente_tipo_contrato
      CHECK (tipo_contrato IN ('TIEMPO_COMPLETO', 'MEDIO_TIEMPO', 'HORA_CLASE'))
);
```

```
SQL> CREATE TABLE Docente (
  2  cedula VARCHAR2(10) NOT NULL, nombres VARCHAR2(50) NOT NULL,
  3  apellidos VARCHAR2(50) NOT NULL, email VARCHAR2(100) NOT NULL,
  4  telefono VARCHAR2(15),
  5  titulo VARCHAR2(50) NOT NULL,
  6  especialidad VARCHAR2(100),
  7  tipo_contrato VARCHAR2(20) DEFAULT 'TIEMPO_COMPLETO' NOT NULL,
  8  fecha_ingreso DATE DEFAULT SYSDATE NOT NULL,
  9  -- Clave primaria
 10  CONSTRAINT pk_docente PRIMARY KEY (cedula),
 11  -- Unicidad de email
 12  CONSTRAINT uk_docente_email UNIQUE (email),
 13  -- Validación de cédula
 14  CONSTRAINT chk_docente_cedula
 15  CHECK (REGEXP_LIKE(cedula, '^\d{10}$')),
 16  -- Validación de tipo de contrato
 17  CONSTRAINT chk_docente_tipo_contrato
 18  CHECK (tipo_contrato IN ('TIEMPO_COMPLETO', 'MEDIO_TIEMPO', 'HORA_CLASE'))
 19  );

Tabla creada.
```

```
-- Índices
CREATE INDEX idx_docente_apellidos ON Docente(apellidos);
```

```
CREATE INDEX idx_docente_especialidad ON Docente(especialidad);
CREATE INDEX idx_docente_tipo_contrato ON Docente(tipo_contrato);
```

```
SQL>
SQL> CREATE INDEX idx_docente_especialidad ON Docente(especialidad);

=ndice creado.

SQL>
SQL> CREATE INDEX idx_docente_tipo_contrato ON Docente(tipo_contrato);

=ndice creado.
```

```
-- Comentarios
COMMENT ON TABLE Docente IS
   'Información de docentes de la institución';
COMMENT ON COLUMN Docente.titulo IS
   'Título académico máximo obtenido (Licenciado, Magíster, PhD, etc)';
COMMENT ON COLUMN Docente.tipo_contrato IS
   'Tipo de contrato: TIEMPO_COMPLETO, MEDIO_TIEMPO, HORA_CLASE';
```

```
SQL> COMMENT ON TABLE Docente IS
  2  'Información de docentes de la institución';

Comentario creado.

SQL> COMMENT ON COLUMN Docente.titulo IS
  2  'Título académico máximo obtenido (Licenciado, Magíster, PhD, etc)';

Comentario creado.

SQL> COMMENT ON COLUMN Docente.tipo_contrato IS
  2  'Tipo de contrato: TIEMPO_COMPLETO, MEDIO_TIEMPO, HORA_CLASE';

Comentario creado.
```

```
-- Verificar creación
DESC Docente;
```

```
SQL> DESC Docente
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(100)
 TELEFONO                                           VARCHAR2(15)
 TITULO                                    NOT NULL VARCHAR2(50)
 ESPECIALIDAD                                       VARCHAR2(100)
 TIPO_CONTRATO                             NOT NULL VARCHAR2(20)
 FECHA_INGRESO                             NOT NULL DATE
```

PASO 6: Crear Tabla PRERREQUISITO

sql

```sql
-- ============================================================
-- TABLA PRERREQUISITO - ORACLE
-- ============================================================

-- Eliminar tabla si existe
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Prerrequisito CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/

-- Crear tabla PRERREQUISITO
CREATE TABLE Prerrequisito (
    codigo_asignatura VARCHAR2(10) NOT NULL,
    codigo_prerrequisito VARCHAR2(10) NOT NULL,
    -- Clave primaria compuesta
    CONSTRAINT pk_prerrequisito
        PRIMARY KEY (codigo_asignatura, codigo_prerrequisito),
    -- Claves foráneas con CASCADE
    CONSTRAINT fk_prereq_asignatura
        FOREIGN KEY (codigo_asignatura)
```

```
    REFERENCES Asignatura(codigo)
    ON DELETE CASCADE,
  CONSTRAINT fk_prereq_prerrequisito
    FOREIGN KEY (codigo_prerrequisito)
    REFERENCES Asignatura(codigo)
    ON DELETE CASCADE,
  -- Una asignatura no puede ser prerrequisito de sí misma
  CONSTRAINT chk_prereq_no_circular
    CHECK (codigo_asignatura != codigo_prerrequisito)
);
```

```
SQL> CREATE TABLE Prerrequisito (
  2  codigo_asignatura VARCHAR2(10) NOT NULL, codigo_prerrequisi
to VARCHAR2(10) NOT NULL,
  3  -- Clave primaria compuesta
  4  CONSTRAINT pk_prerrequisito
  5  PRIMARY KEY (codigo_asignatura, codigo_prerrequisito),
  6  -- Claves foráneas con CASCADE
  7  CONSTRAINT fk_prereq_asignatura FOREIGN KEY (codigo_asignat
ura) REFERENCES Asignatura(codigo) ON DELETE CASCADE,
  8  CONSTRAINT fk_prereq_prerrequisito FOREIGN KEY (codigo_prer
requisito) REFERENCES Asignatura(codigo) ON DELETE CASCADE,
  9  -- Una asignatura no puede ser prerrequisito de sí misma
 10  CONSTRAINT chk_prereq_no_circular
 11  CHECK (codigo_asignatura != codigo_prerrequisito)
 12  );

Tabla creada.
```

```
-- Índice para búsquedas inversas (qué asignaturas tienen X como prerrequisito)
CREATE INDEX idx_prereq_codigo_prereq
  ON Prerrequisito(codigo_prerrequisito);
```

```
SQL> CREATE INDEX idx_prereq_codigo_prereq ON Prerrequisito(codi
go_prerrequisito);

=ndice creado.

SQL>
```

```
-- Comentarios
COMMENT ON TABLE Prerrequisito IS
    'Relación de prerrequisitos entre asignaturas';
COMMENT ON COLUMN Prerrequisito.codigo_asignatura IS
    'Asignatura que requiere el prerrequisito';
COMMENT ON COLUMN Prerrequisito.codigo_prerrequisito IS
    'Asignatura que es prerrequisito';
```

```
SQL> COMMENT ON TABLE Prerrequisito IS 'Relación de prerrequisit
os entre asignaturas';

Comentario creado.

SQL> COMMENT ON COLUMN Prerrequisito.codigo_asignatura IS 'Asign
atura que requiere el prerrequisito';

Comentario creado.

SQL> COMMENT ON COLUMN Prerrequisito.codigo_prerrequisito IS 'As
ignatura que es prerrequisito';

Comentario creado.
```

```
-- Verificar creación
DESC Prerrequisito;
```

```
SQL> DESC Prerrequisito
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ------------
----------------
 CODIGO_ASIGNATURA                         NOT NULL VARCHAR2(10)
 CODIGO_PRERREQUISITO                      NOT NULL VARCHAR2(10)
```

PASO 7: Crear Tabla MATRICULA con Secuencia y Trigger

sql

```
-- ============================================================
-- TABLA MATRICULA CON SECUENCIA - ORACLE
-- ============================================================
```

```
-- Eliminar objetos si existen
BEGIN
  EXECUTE IMMEDIATE 'DROP SEQUENCE seq_matricula';
EXCEPTION
  WHEN OTHERS THEN NULL;
END;
/

BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE Matricula CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN
    IF SQLCODE != -942 THEN RAISE; END IF;
END;
/

-- Paso 7.1: Crear secuencia para ID de matrícula
CREATE SEQUENCE seq_matricula
   START WITH 1
   INCREMENT BY 1
   MAXVALUE 9999999
   MINVALUE 1
   NOCACHE
   NOCYCLE
   ORDER;
```

```
SQL> CREATE SEQUENCE seq_matricula START WITH 1
  2  INCREMENT BY 1
  3  MAXVALUE 9999999
  4  MINVALUE 1 NOCACHE NOCYCLE ORDER;

Secuencia creada.
```

```
PROMPT Secuencia seq_matricula creada

-- Paso 7.2: Crear tabla MATRICULA
CREATE TABLE Matricula (
   id_matricula NUMBER(10) NOT NULL,
```

```
    cedula_estudiante VARCHAR2(10) NOT NULL,
    codigo_asignatura VARCHAR2(10) NOT NULL,
    cedula_docente VARCHAR2(10) NOT NULL,
    periodo VARCHAR2(10) NOT NULL,
    paralelo CHAR(1) NOT NULL,
    fecha_matricula DATE DEFAULT SYSDATE NOT NULL,
    nota_parcial1 NUMBER(4,2),
    nota_parcial2 NUMBER(4,2),
    nota_final NUMBER(4,2),
    estado VARCHAR2(20) DEFAULT 'CURSANDO' NOT NULL,
    -- Clave primaria
    CONSTRAINT pk_matricula PRIMARY KEY (id_matricula),
    -- Unicidad: un estudiante no puede matricularse dos veces en la misma asignatura en el mismo periodo
    CONSTRAINT uk_matricula_estudiante_asig_periodo
        UNIQUE (cedula_estudiante, codigo_asignatura, periodo),
    -- Claves foráneas
    CONSTRAINT fk_matricula_estudiante
        FOREIGN KEY (cedula_estudiante)
        REFERENCES Estudiante(cedula)
        ON DELETE CASCADE,
    CONSTRAINT fk_matricula_asignatura
        FOREIGN KEY (codigo_asignatura)
        REFERENCES Asignatura(codigo) ,
    CONSTRAINT fk_matricula_docente
        FOREIGN KEY (cedula_docente)
        REFERENCES Docente(cedula)
        ON DELETE SET NULL,
    -- Validación de formato de periodo (YYYY-NS: 2024-1S, 2024-2S)
    CONSTRAINT chk_matricula_periodo
        CHECK (REGEXP_LIKE(periodo, '^\d{4}-[12]S$')),
    -- Validación de paralelo (A-Z)
    CONSTRAINT chk_matricula_paralelo
        CHECK (REGEXP_LIKE(paralelo, '^[A-Z]$')),
    -- Validación de notas (0-10 o NULL)
    CONSTRAINT chk_matricula_nota1
        CHECK (nota_parcial1 IS NULL OR (nota_parcial1 BETWEEN 0 AND 10)),
    CONSTRAINT chk_matricula_nota2
        CHECK (nota_parcial2 IS NULL OR (nota_parcial2 BETWEEN 0 AND 10)),
    CONSTRAINT chk_matricula_notafinal
```

```
    CHECK (nota_final IS NULL OR (nota_final BETWEEN 0 AND 10)),
  -- Validación de estado
  CONSTRAINT chk_matricula_estado
    CHECK (estado IN ('CURSANDO', 'APROBADO', 'REPROBADO', 'RETIRADO')),
  -- Restricción compleja: si está CURSANDO no debe tener nota final
  -- Si está APROBADO o REPROBADO debe tener nota final
  CONSTRAINT chk_matricula_estado_nota
    CHECK (
      (estado = 'CURSANDO' AND nota_final IS NULL) OR
      (estado IN ('APROBADO', 'REPROBADO') AND nota_final IS NOT NULL) OR
      (estado = 'RETIRADO')
    )
);
```

```
 19  CONSTRAINT chk_matricula_paralelo
 20  CHECK (REGEXP_LIKE(paralelo, '^[A-Z]$')),
 21  -- Validación de notas (0-10 o NULL)
 22  CONSTRAINT chk_matricula_nota1
 23  CHECK (nota_parcial1 IS NULL OR (nota_parcial1 BETWEEN 0 AN
D 10)),
 24  CONSTRAINT chk_matricula_nota2
 25  CHECK (nota_parcial2 IS NULL OR (nota_parcial2 BETWEEN 0 AN
D 10)),
 26  CONSTRAINT chk_matricula_notafinal
 27  CHECK (nota_final IS NULL OR (nota_final BETWEEN 0 AND 10))
,
 28  -- Validación de estado
 29  CONSTRAINT chk_matricula_estado
 30  CHECK (estado IN ('CURSANDO', 'APROBADO', 'REPROBADO', 'RET
IRADO')),
 31  -- Restricción compleja: si está CURSANDO no debe tener not
a final
 32  -- Si está APROBADO o REPROBADO debe tener nota final
 33  CONSTRAINT chk_matricula_estado_nota CHECK (
 34  (estado = 'CURSANDO' AND nota_final IS NULL) OR
 35  (estado IN ('APROBADO', 'REPROBADO') AND nota_final IS NOT
NULL) OR
 36  (estado = 'RETIRADO')
 37  )
 38  );

Tabla creada.
```

```
-- Paso 7.3: Crear trigger para auto-incremento
CREATE OR REPLACE TRIGGER trg_matricula_id
BEFORE INSERT ON Matricula
FOR EACH ROW
BEGIN
   -- Si no se proporciona id_matricula, usar la secuencia
   IF :NEW.id_matricula IS NULL THEN
      SELECT seq_matricula.NEXTVAL INTO :NEW.id_matricula FROM DUAL;
   END IF;
END;
/
```

PROMPT Trigger trg_matricula_id creado

```
SQL> CREATE OR REPLACE TRIGGER trg_matricula_id BEFORE INSERT ON
 Matricula
  2  FOR EACH ROW BEGIN
  3  -- Si no se proporciona id_matricula, usar la secuencia
  4  IF :NEW.id_matricula IS NULL THEN
  5  SELECT seq_matricula.NEXTVAL INTO :NEW.id_matricula FROM DU
AL; END IF;
  6  END;
  7  /

Disparador creado.
```

```
SQL> PROMPT Trigger trg_matricula_id creado
Trigger trg_matricula_id creado
```

```
-- Paso 7.4: Crear índices para optimización
CREATE INDEX idx_matricula_estudiante
    ON Matricula(cedula_estudiante);

CREATE INDEX idx_matricula_asignatura
    ON Matricula(codigo_asignatura);

CREATE INDEX idx_matricula_periodo
    ON Matricula(periodo);

CREATE INDEX idx_matricula_docente
    ON Matricula(cedula_docente);

CREATE INDEX idx_matricula_estado
    ON Matricula(estado);

-- Índice compuesto para consultas frecuentes
CREATE INDEX idx_matricula_periodo_estado
    ON Matricula(periodo, estado);
```

```
SQL>
SQL> CREATE INDEX idx_matricula_estado ON Matricula(estado);

=ndice creado.

SQL>
SQL> -- Índice compuesto para consultas frecuentes
SQL> CREATE INDEX idx_matricula_periodo_estado ON Matricula(peri
odo, estado);

=ndice creado.
```

```
-- Comentarios
COMMENT ON TABLE Matricula IS
    'Registro de matrículas de estudiantes en asignaturas por periodo académico';
COMMENT ON COLUMN Matricula.id_matricula IS
    'Identificador único generado automáticamente por secuencia';
COMMENT ON COLUMN Matricula.periodo IS
    'Periodo académico en formato YYYY-NS (ej: 2024-1S, 2024-2S)';
COMMENT ON COLUMN Matricula.paralelo IS
    'Paralelo o grupo de la asignatura (A, B, C, etc)';
COMMENT ON COLUMN Matricula.estado IS
    'Estado de la matrícula: CURSANDO, APROBADO, REPROBADO, RETIRADO';
```

```
SQL> COMMENT ON TABLE Matricula IS
  2  'Registro de matrículas de estudiantes en asignaturas por p
eriodo académico';

Comentario creado.

SQL> COMMENT ON COLUMN Matricula.id_matricula IS 'Identificador
único generado automáticamente por secuencia';

Comentario creado.

SQL> COMMENT ON COLUMN Matricula.periodo IS
  2  'Periodo académico en formato YYYY-NS (ej: 2024-1S, 2024-2S
)';

Comentario creado.

SQL> COMMENT ON COLUMN Matricula.paralelo IS 'Paralelo o grupo d
e la asignatura (A, B, C, etc)';

Comentario creado.

SQL> COMMENT ON COLUMN Matricula.estado IS
  2  'Estado de la matrícula: CURSANDO, APROBADO, REPROBADO, RET
IRADO';

Comentario creado.
```

```
-- Verificar creación
DESC Matricula;
```

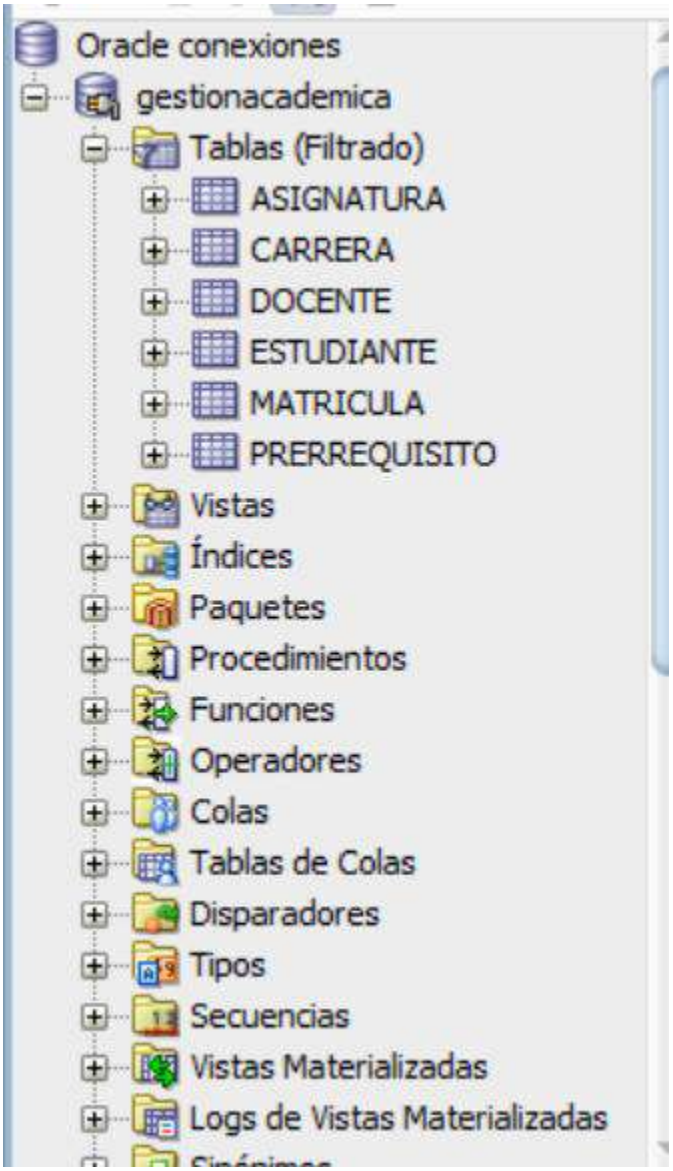
```
SQL> DESC Matricula
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ------------
----------------
 ID_MATRICULA                              NOT NULL NUMBER(10)
 CEDULA_ESTUDIANTE                         NOT NULL VARCHAR2(10)
 CODIGO_ASIGNATURA                         NOT NULL VARCHAR2(10)
 CEDULA_DOCENTE                            NOT NULL VARCHAR2(10)
 PERIODO                                   NOT NULL VARCHAR2(10)
 PARALELO                                  NOT NULL CHAR(1)
 FECHA_MATRICULA                           NOT NULL DATE
 NOTA_PARCIAL1                                      NUMBER(4,2)
 NOTA_PARCIAL2                                      NUMBER(4,2)
 NOTA_FINAL                                         NUMBER(4,2)
 ESTADO                                    NOT NULL VARCHAR2(20)
```

```
-- Verificar secuencia
SELECT sequence_name, last_number, increment_by, cache_size
FROM user_sequences
WHERE sequence_name = 'SEQ_MATRICULA';
```

```
SQL> SELECT sequence_name, last_number, increment_by, cache_size FROM user_sequences
  2  WHERE sequence_name = 'SEQ_MATRICULA';

SEQUENCE_NAME
--------------------------------------------------------------------------------
LAST_NUMBER INCREMENT_BY CACHE_SIZE
----------- ------------ ----------
SEQ_MATRICULA
          1            1          0
```

Trabajo hecho hasta ahora:


Oracle conexiones
gestionacademica
Tablas (Filtrado)
ASIGNATURA
CARRERA
DOCENTE
ESTUDIANTE
MATRICULA
PRERREQUISITO
Vistas
Índices
Paquetes
Procedimientos
Funciones
Operadores
Colas
Tablas de Colas
Disparadores
Tipos
Secuencias
Vistas Materializadas
Logs de Vistas Materializadas

```
SQL> desc tab
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 TNAME                                     NOT NULL VARCHAR2(128)
 TABTYPE                                            VARCHAR2(13)
 CLUSTERID                                          NUMBER

SQL> select tname from tab;

TNAME
--------------------------------------------------------------------------------
CARRERA
ESTUDIANTE
ASIGNATURA
DOCENTE
PRERREQUISITO
MATRICULA

6 filas seleccionadas.
```

	A	B	C
1	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
2	SYS_C007671	C	"NOMBRE" IS NOT NULL
3	SYS_C007672	C	"DURACION_SEMESTRES" IS NOT NULL
4	SYS_C007673	C	"CREDITOS_TOTALES" IS NOT NULL
5	SYS_C007670	C	"CODIGO" IS NOT NULL
6	SYS_C007674	C	"FACULTAD" IS NOT NULL
7	CHK_CARRERA_CREDITOS	C	creditos_totales > 0
8	SYS_C007675	C	"FECHA_CREACION" IS NOT NULL
9	CHK_CARRERA_DURACION	C	duracion_semestres BETWEEN 8 AND 12
10	PK_CARRERA	P	
11	UK_CARRERA_NOMBRE	U	
12			
13			

Archivo Excel generado

PASO 8: Modificaciones con ALTER TABLE

```
-- ================================================
-- MODIFICACIONES CON ALTER TABLE - ORACLE
-- ================================================

PROMPT ======================================
    PROMPT EJEMPLOS DE MODIFICACIONES CON ALTER TABLE
    PROMPT ======================================
```

```
-- 8.1: AGREGAR COLUMNAS
PROMPT Agregando columnas...

-- Agregar una columna simple
ALTER TABLE Estudiante ADD direccion VARCHAR2(200);
```

```
SQL> ALTER TABLE Estudiante ADD direccion VARCHAR2(200);

Tabla modificada.
```

```
-- Agregar múltiples columnas con valores por defecto
ALTER TABLE Estudiante ADD (
    ciudad VARCHAR2(50) DEFAULT 'Quito',
    provincia VARCHAR2(50) DEFAULT 'Pichincha'
);
```

```
SQL> ALTER TABLE Estudiante ADD (
  2  ciudad VARCHAR2(50) DEFAULT 'Quito', provincia VARCHAR2(50) DEFAULT 'Pichincha'
  3  );

Tabla modificada.
```

```
PROMPT Columnas agregadas exitosamente

-- 8.2: MODIFICAR COLUMNAS EXISTENTES

-- Cambiar tamaño de columna
ALTER TABLE Estudiante MODIFY email VARCHAR2(150);
```

```
SQL> ALTER TABLE Estudiante MODIFY email VARCHAR2(150);

Tabla modificada.
```

```
-- Cambiar tipo de dato y agregar NOT NULL
ALTER TABLE Estudiante MODIFY telefono VARCHAR2(20);
```

```
SQL> ALTER TABLE Estudiante MODIFY telefono VARCHAR2(20);

Tabla modificada.
```

```
-- Modificar valor por defecto
ALTER TABLE Estudiante MODIFY ciudad DEFAULT 'Quito';
```

```
SQL> ALTER TABLE Estudiante MODIFY ciudad DEFAULT 'Quito';

Tabla modificada.
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CEDULA	VARCHAR2(10 BYTE)	No	(null)	1	C,dula de identidad ecuatoriana (10 d¡gitos)
2	NOMBRES	VARCHAR2(50 BYTE)	No	(null)	2	(null)
3	APELLIDOS	VARCHAR2(50 BYTE)	No	(null)	3	(null)
4	EMAIL	VARCHAR2(150 BYTE)	No	(null)	4	Correo electr¤nico institucional £nico
5	TELEFONO	VARCHAR2(20 BYTE)	Yes	(null)	5	(null)
6	FECHA_NACIMIENTO	DATE	No	(null)	6	(null)
7	GENERO	CHAR(1 BYTE)	No	(null)	7	(null)
8	CODIGO_CARRERA	VARCHAR2(10 BYTE)	No	(null)	8	(null)
9	ESTADO	VARCHAR2(20 BYTE)	No	'ACTIVO'	9	Estado actual del estudiante: ACTIVO, INACTIVO, GRADUADO, RETIRADO
10	CREDITOS_APROBADOS	NUMBER(3,0)	No	0	10	Total de cr,ditos acumulados hasta la fecha
11	FECHA_INGRESO	DATE	No	SYSDATE	11	(null)
12	DIRECCION	VARCHAR2(200 BYTE)	Yes	(null)	12	(null)
13	CIUDAD	VARCHAR2(50 BYTE)	Yes	'Quito'	13	(null)
14	PROVINCIA	VARCHAR2(50 BYTE)	Yes	'Pichincha'	14	(null)

```
SQL> DESC estudiante
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
 DIRECCION                                          VARCHAR2(200)
 CIUDAD                                             VARCHAR2(50)
 PROVINCIA                                          VARCHAR2(50)
```

-- 8.3: AGREGAR RESTRICCIONES A TABLAS EXISTENTES

Ejemplos

ALTER TABLE nombre_de_tabla

ADD CONSTRAINT nombre_constraint tipo_de_constraint (columna1, columna2, ...);

PRIMARY KEY

ALTER TABLE empleados

ADD CONSTRAINT pk_empleados_id PRIMARY KEY (id_empleado);

FOREIGN KEY

ALTER TABLE pedidos

ADD CONSTRAINT fk_pedidos_cliente FOREIGN KEY (id_cliente)

REFERENCES clientes(id_cliente);

NOT NULL

ALTER TABLE empleados

MODIFY (nombre NOT NULL);

Fin ejemplos

```
-- Agregar restricción CHECK
ALTER TABLE Asignatura
ADD CONSTRAINT chk_asignatura_creditos_horas
   CHECK (creditos * 16 >= horas_teoria + horas_practica);
```

```
SQL> ALTER TABLE Asignatura
  2  ADD CONSTRAINT chk_asignatura_creditos_horas
  3  CHECK (creditos * 16 >= horas_teoria + horas_practica);

Tabla modificada.
```

```
-- Agregar restricción UNIQUE
ALTER TABLE Docente
ADD CONSTRAINT uk_docente_telefono UNIQUE (telefono);
```

```
Tabla modificada.

SQL> ALTER TABLE Docente
  2  ADD CONSTRAINT uk_docente_telefono UNIQUE (telefono);

Tabla modificada.
```

```
-- 8.4: ELIMINAR RESTRICCIONES
-- Eliminar restricción CHECK
ALTER TABLE Estudiante
DROP CONSTRAINT chk_estudiante_edad;
```

```
SQL> ALTER TABLE Estudiante
  2  DROP CONSTRAINT chk_estudiante_edad;
DROP CONSTRAINT chk_estudiante_edad
                *
ERROR en lÝnea 2:
ORA-02443: No se puede borrar la restricci¾n - restricci¾n no existente
```

(no existe esta restricción en el documento)

```
-- Eliminar restricción UNIQUE
ALTER TABLE Docente
DROP CONSTRAINT uk_docente_telefono;
```

```
SQL> ALTER TABLE Docente
  2  DROP CONSTRAINT uk_docente_telefono;

Tabla modificada.
```

```
-- 8.5: RENOMBRAR COLUMNAS
-- Renombrar columna
ALTER TABLE Estudiante
RENAME COLUMN telefono TO celular;
```

```
SQL> ALTER TABLE Estudiante
  2  RENAME COLUMN telefono TO celular;

Tabla modificada.
```

```
SQL> desc Estudiante
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 CELULAR                                            VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
 DIRECCION                                          VARCHAR2(200)
 CIUDAD                                             VARCHAR2(50)
 PROVINCIA                                          VARCHAR2(50)
```

```
-- Revertir cambio
ALTER TABLE Estudiante
RENAME COLUMN celular TO telefono;
```

```
SQL> ALTER TABLE Estudiante
  2  RENAME COLUMN celular TO telefono;

Tabla modificada.

SQL> desc Estudiante
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
 DIRECCION                                          VARCHAR2(200)
 CIUDAD                                             VARCHAR2(50)
 PROVINCIA                                          VARCHAR2(50)
```

```
-- 8.6: ELIMINAR COLUMNAS
-- Eliminar una columna
ALTER TABLE Estudiante DROP COLUMN provincia;
```

```
SQL> ALTER TABLE Estudiante DROP COLUMN provincia;

Tabla modificada.

SQL> desc Estudiante
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
 DIRECCION                                          VARCHAR2(200)
 CIUDAD                                             VARCHAR2(50)
```

```
-- Eliminar múltiples columnas
ALTER TABLE Estudiante DROP (ciudad, direccion);
```

```
SQL> ALTER TABLE Estudiante DROP (ciudad, direccion);

Tabla modificada.

SQL> desc Estudiante
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
```

-- 8.7: RENOMBRAR TABLAS

-- RENAME Estudiante TO Alumno;

```
SQL> RENAME Estudiante TO Alumno;

Nombre de tabla cambiado.

SQL> desc Alumno
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE
```

-- RENAME Alumno TO Estudiante;

```
SQL> RENAME Alumno TO Estudiante;

Nombre de tabla cambiado.

SQL> desc Estudiante
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 CEDULA                                    NOT NULL VARCHAR2(10)
 NOMBRES                                   NOT NULL VARCHAR2(50)
 APELLIDOS                                 NOT NULL VARCHAR2(50)
 EMAIL                                     NOT NULL VARCHAR2(150)
 TELEFONO                                           VARCHAR2(20)
 FECHA_NACIMIENTO                          NOT NULL DATE
 GENERO                                    NOT NULL CHAR(1)
 CODIGO_CARRERA                            NOT NULL VARCHAR2(10)
 ESTADO                                    NOT NULL VARCHAR2(20)
 CREDITOS_APROBADOS                        NOT NULL NUMBER(3)
 FECHA_INGRESO                             NOT NULL DATE

SQL>
```

```
-- 8.8: DESHABILITAR/HABILITAR RESTRICCIONES
-- Deshabilitar restricción
ALTER TABLE Matricula DISABLE CONSTRAINT chk_matricula_estado_nota;
```

```
SQL> ALTER TABLE Matricula DISABLE CONSTRAINT chk_matricula_estado_nota;

Tabla modificada.
```

```
-- Habilitar restricción
ALTER TABLE Matricula ENABLE CONSTRAINT chk_matricula_estado_not
```

```
SQL> ALTER TABLE Matricula ENABLE CONSTRAINT chk_matricula_estado_nota;

Tabla modificada.
```

```
-- 8.9: AGREGAR COLUMNA COMPUTED (VIRTUAL)
-- Oracle soporta columnas virtuales
ALTER TABLE Matricula ADD (
    promedio_parciales NUMBER(4,2)
    GENERATED ALWAYS AS ((nota_parcial1 + nota_parcial2) / 2) VIRTUAL
);
```

```
SQL> ALTER TABLE Matricula ADD ( promedio_parciales NUMBER(4,2)
  2  GENERATED ALWAYS AS ((nota_parcial1 + nota_parcial2) / 2) VIRTUAL
  3  );

Tabla modificada.
```

```
SQL> desc Matricula
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 ID_MATRICULA                              NOT NULL NUMBER(10)
 CEDULA_ESTUDIANTE                         NOT NULL VARCHAR2(10)
 CODIGO_ASIGNATURA                         NOT NULL VARCHAR2(10)
 CEDULA_DOCENTE                            NOT NULL VARCHAR2(10)
 PERIODO                                   NOT NULL VARCHAR2(10)
 PARALELO                                  NOT NULL CHAR(1)
 FECHA_MATRICULA                           NOT NULL DATE
 NOTA_PARCIAL1                                      NUMBER(4,2)
 NOTA_PARCIAL2                                      NUMBER(4,2)
 NOTA_FINAL                                         NUMBER(4,2)
 ESTADO                                    NOT NULL VARCHAR2(20)
 PROMEDIO_PARCIALES                                 NUMBER(4,2)
```

PASO 9: Scripts de Verificación

```
-- ======================================================
-- SCRIPTS DE VERIFICACIÓN - ORACLE
-- ======================================================
-- 9.1: Ver todas las tablas del usuario
SELECT table_name, num_rows
FROM user_tables
ORDER BY table_name;
```

Hoja de Trabajo | Generador de Consultas

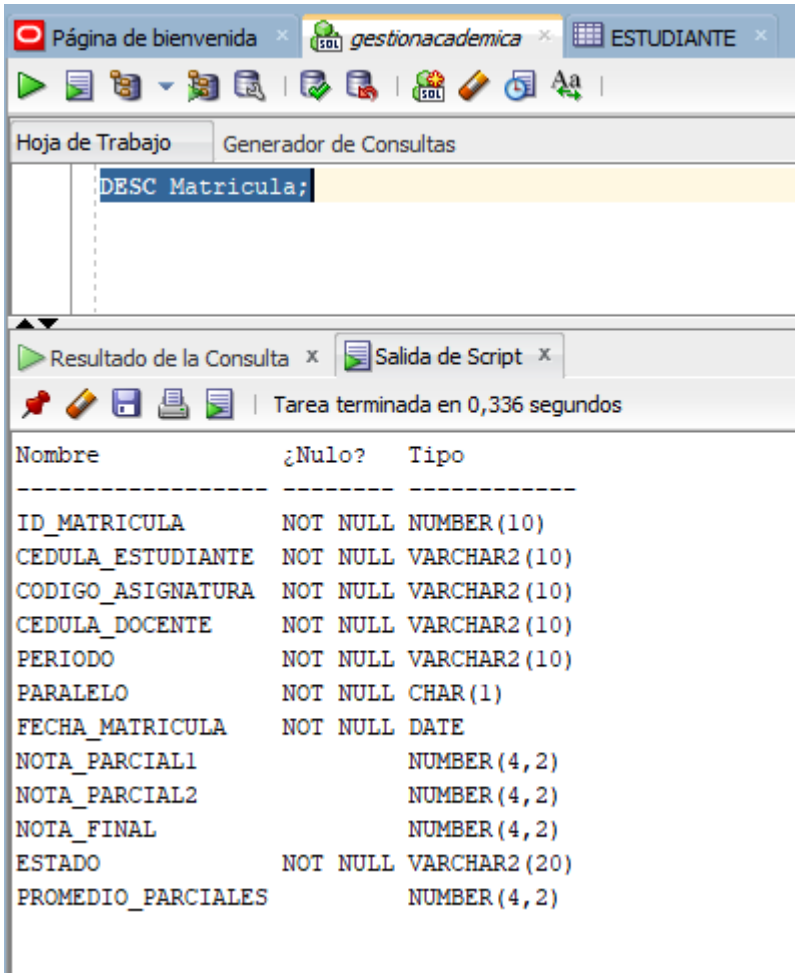
```
SELECT table_name, num_rows FROM user_tables
ORDER BY table_name;
```

Resultado de la Consulta

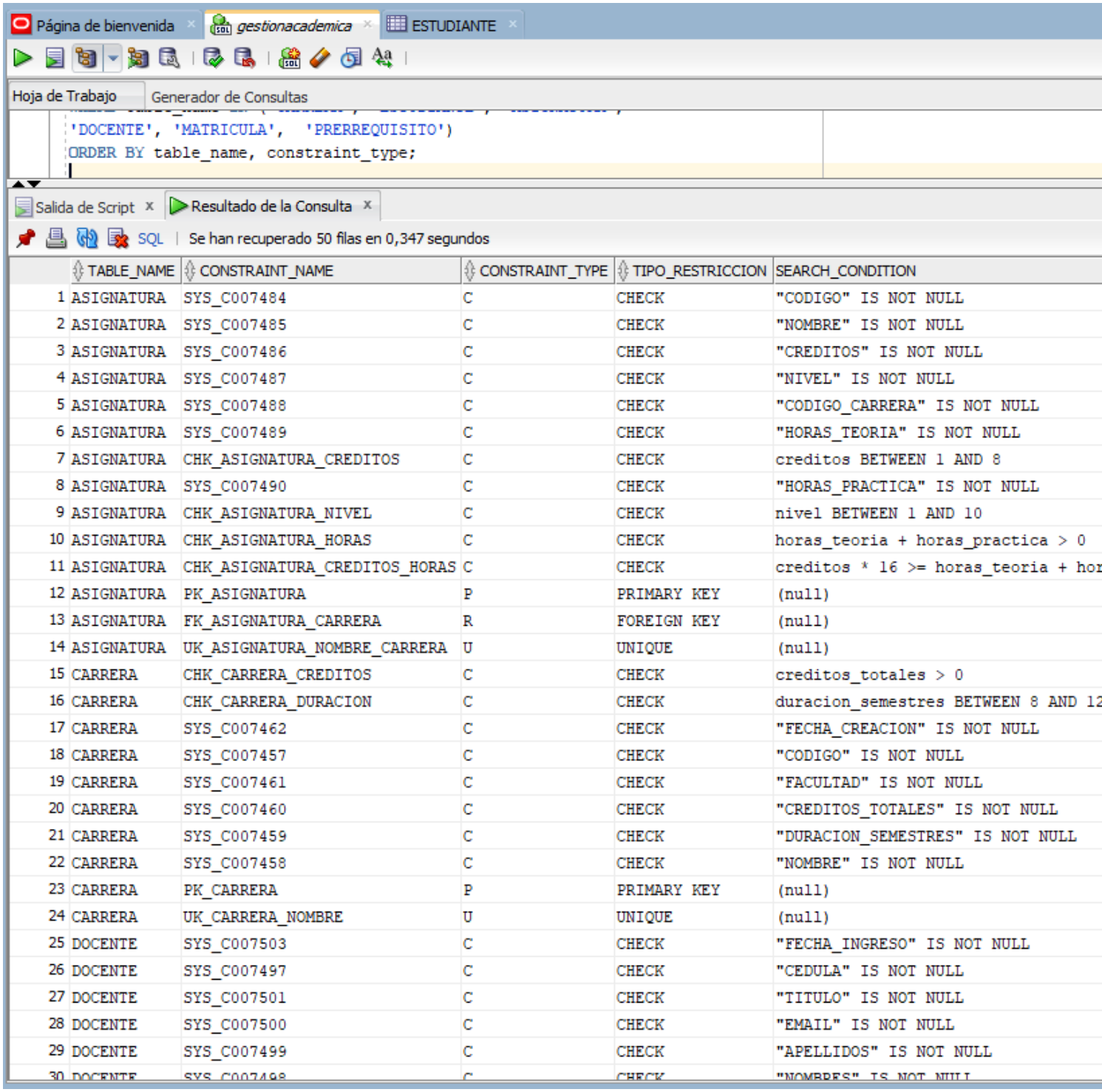
SQL | Todas las Filas Recuperadas: 6 en 0,038 segundos

	TABLE_NAME	NUM_ROWS
1	ASIGNATURA	(null)
2	CARRERA	(null)
3	DOCENTE	(null)
4	ESTUDIANTE	(null)
5	MATRICULA	(null)
6	PRERREQUISITO	(null)

```
-- 9.2: Ver estructura de una tabla específica
DESC Matricula;
```




```
-- 9.3: Ver todas las restricciones
SELECT
    table_name,
    constraint_name,
    constraint_type,  CASE
    constraint_type
        WHEN 'P' THEN 'PRIMARY KEY'
        WHEN 'R' THEN 'FOREIGN KEY'
        WHEN 'U' THEN 'UNIQUE' WHEN
        'C' THEN 'CHECK'
    END AS tipo_restriccion, search_condition
FROM user_constraints
WHERE table_name IN ('CARRERA', 'ESTUDIANTE', 'ASIGNATURA',
'DOCENTE', 'MATRICULA',  'PRERREQUISITO')
ORDER BY table_name, constraint_type;
```



Página de bienvenida | gestionacademica | ESTUDIANTE

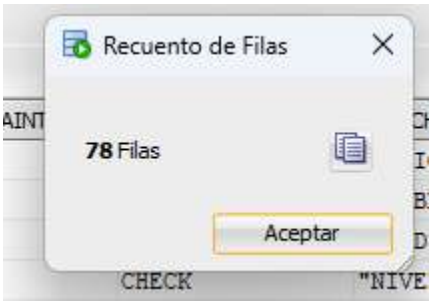
Hoja de Trabajo | Generador de Consultas

```
'DOCENTE', 'MATRICULA',  'PRERREQUISITO')
ORDER BY table_name, constraint_type;
```

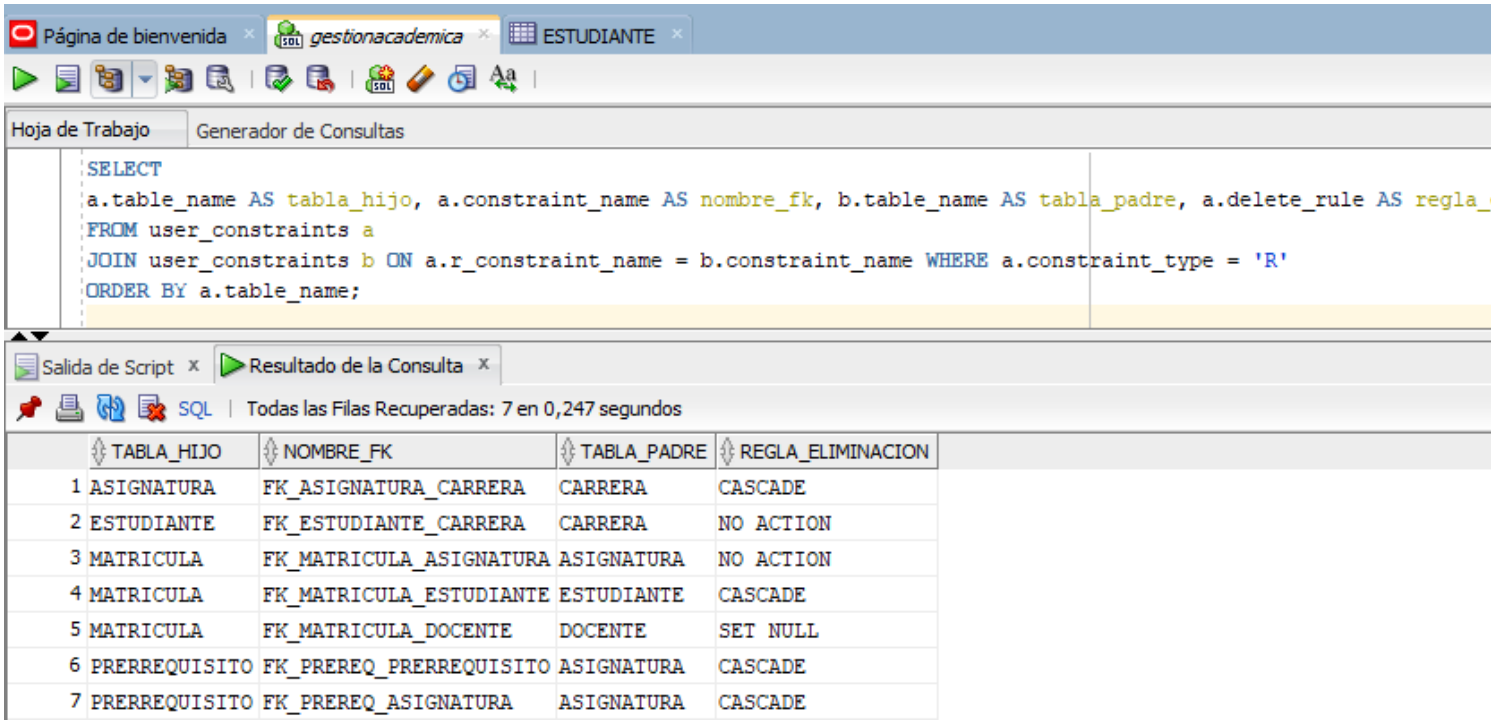
Salida de Script | Resultado de la Consulta

SQL | Se han recuperado 50 filas en 0,347 segundos

	TABLE_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE	TIPO_RESTRICCION	SEARCH_CONDITION
1	ASIGNATURA	SYS_C007484	C	CHECK	"CODIGO" IS NOT NULL
2	ASIGNATURA	SYS_C007485	C	CHECK	"NOMBRE" IS NOT NULL
3	ASIGNATURA	SYS_C007486	C	CHECK	"CREDITOS" IS NOT NULL
4	ASIGNATURA	SYS_C007487	C	CHECK	"NIVEL" IS NOT NULL
5	ASIGNATURA	SYS_C007488	C	CHECK	"CODIGO_CARRERA" IS NOT NULL
6	ASIGNATURA	SYS_C007489	C	CHECK	"HORAS_TEORIA" IS NOT NULL
7	ASIGNATURA	CHK_ASIGNATURA_CREDITOS	C	CHECK	creditos BETWEEN 1 AND 8
8	ASIGNATURA	SYS_C007490	C	CHECK	"HORAS_PRACTICA" IS NOT NULL
9	ASIGNATURA	CHK_ASIGNATURA_NIVEL	C	CHECK	nivel BETWEEN 1 AND 10
10	ASIGNATURA	CHK_ASIGNATURA_HORAS	C	CHECK	horas_teoria + horas_practica > 0
11	ASIGNATURA	CHK_ASIGNATURA_CREDITOS_HORAS	C	CHECK	creditos * 16 >= horas_teoria + hor
12	ASIGNATURA	PK_ASIGNATURA	P	PRIMARY KEY	(null)
13	ASIGNATURA	FK_ASIGNATURA_CARRERA	R	FOREIGN KEY	(null)
14	ASIGNATURA	UK_ASIGNATURA_NOMBRE_CARRERA	U	UNIQUE	(null)
15	CARRERA	CHK_CARRERA_CREDITOS	C	CHECK	creditos_totales > 0
16	CARRERA	CHK_CARRERA_DURACION	C	CHECK	duracion_semestres BETWEEN 8 AND 12
17	CARRERA	SYS_C007462	C	CHECK	"FECHA_CREACION" IS NOT NULL
18	CARRERA	SYS_C007457	C	CHECK	"CODIGO" IS NOT NULL
19	CARRERA	SYS_C007461	C	CHECK	"FACULTAD" IS NOT NULL
20	CARRERA	SYS_C007460	C	CHECK	"CREDITOS_TOTALES" IS NOT NULL
21	CARRERA	SYS_C007459	C	CHECK	"DURACION_SEMESTRES" IS NOT NULL
22	CARRERA	SYS_C007458	C	CHECK	"NOMBRE" IS NOT NULL
23	CARRERA	PK_CARRERA	P	PRIMARY KEY	(null)
24	CARRERA	UK_CARRERA_NOMBRE	U	UNIQUE	(null)
25	DOCENTE	SYS_C007503	C	CHECK	"FECHA_INGRESO" IS NOT NULL
26	DOCENTE	SYS_C007497	C	CHECK	"CEDULA" IS NOT NULL
27	DOCENTE	SYS_C007501	C	CHECK	"TITULO" IS NOT NULL
28	DOCENTE	SYS_C007500	C	CHECK	"EMAIL" IS NOT NULL
29	DOCENTE	SYS_C007499	C	CHECK	"APELLIDOS" IS NOT NULL
30	DOCENTE	SYS_C007498	C	CHECK	"NOMBRES" IS NOT NULL



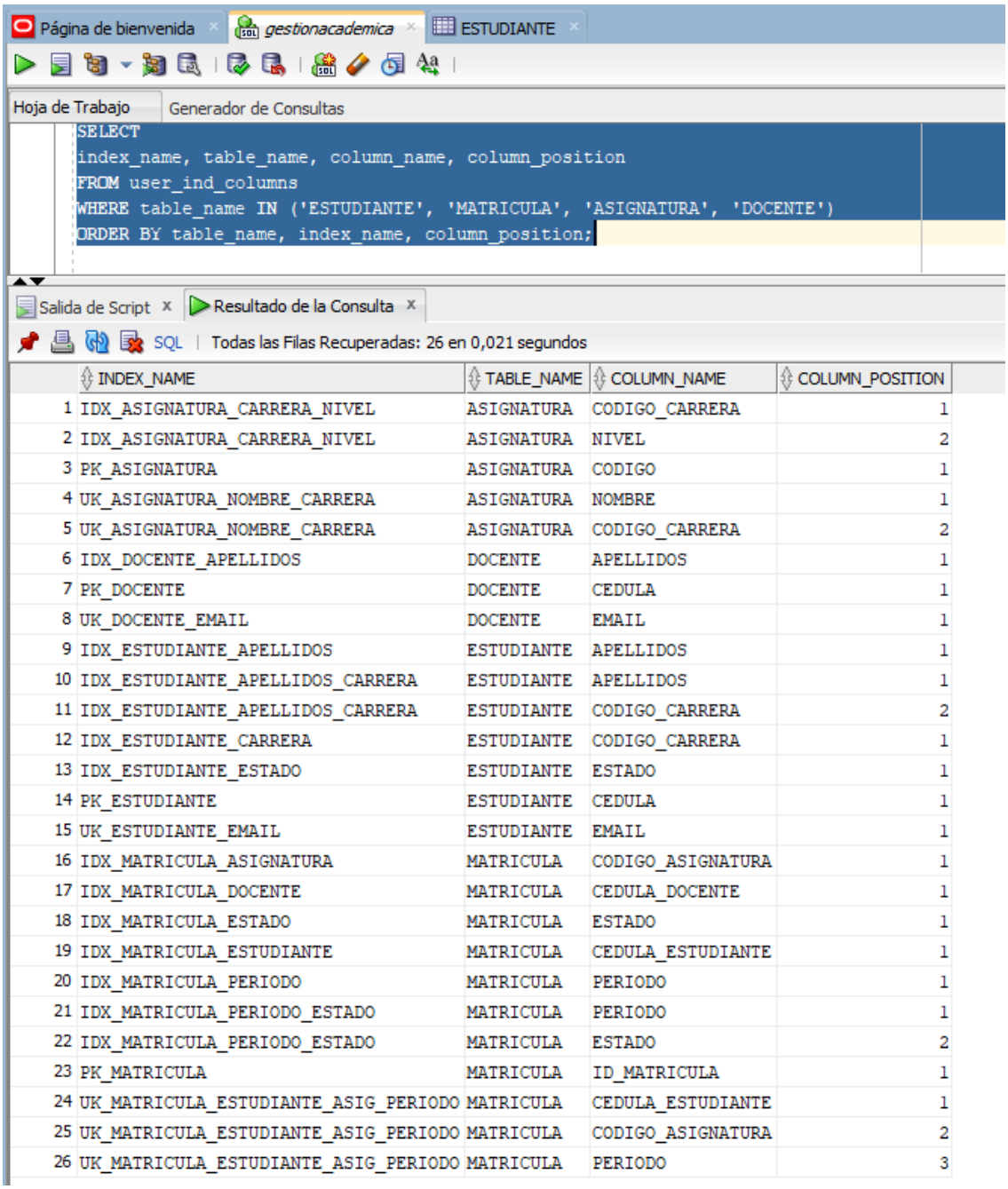

```
-- 9.4: Ver claves foráneas con sus relaciones
PROMPT Relaciones de claves foráneas:
SELECT
    a.table_name AS tabla_hijo,
    a.constraint_name AS nombre_fk,
    b.table_name AS tabla_padre, a.delete_rule
    AS regla_eliminacion
FROM user_constraints a
JOIN user_constraints b ON a.r_constraint_name = b.constraint_name WHERE
a.constraint_type = 'R'
ORDER BY a.table_name;
```



	TABLA_HIJO	NOMBRE_FK	TABLA_PADRE	REGLA_ELIMINACION
1	ASIGNATURA	FK_ASIGNATURA_CARRERA	CARRERA	CASCADE
2	ESTUDIANTE	FK_ESTUDIANTE_CARRERA	CARRERA	NO ACTION
3	MATRICULA	FK_MATRICULA_ASIGNATURA	ASIGNATURA	NO ACTION
4	MATRICULA	FK_MATRICULA_ESTUDIANTE	ESTUDIANTE	CASCADE
5	MATRICULA	FK_MATRICULA_DOCENTE	DOCENTE	SET NULL
6	PRERREQUISITO	FK_PREREQ_PRERREQUISITO	ASIGNATURA	CASCADE
7	PRERREQUISITO	FK_PREREQ_ASIGNATURA	ASIGNATURA	CASCADE

```
-- 9.5: Ver todos los índices
PROMPT Índices creados:
SELECT
    index_name,
```

```
    table_name,
    column_name,
    column_position
FROM user_ind_columns
WHERE table_name IN ('ESTUDIANTE', 'MATRICULA', 'ASIGNATURA', 'DOCENTE')
ORDER BY table_name, index_name, column_position;
```

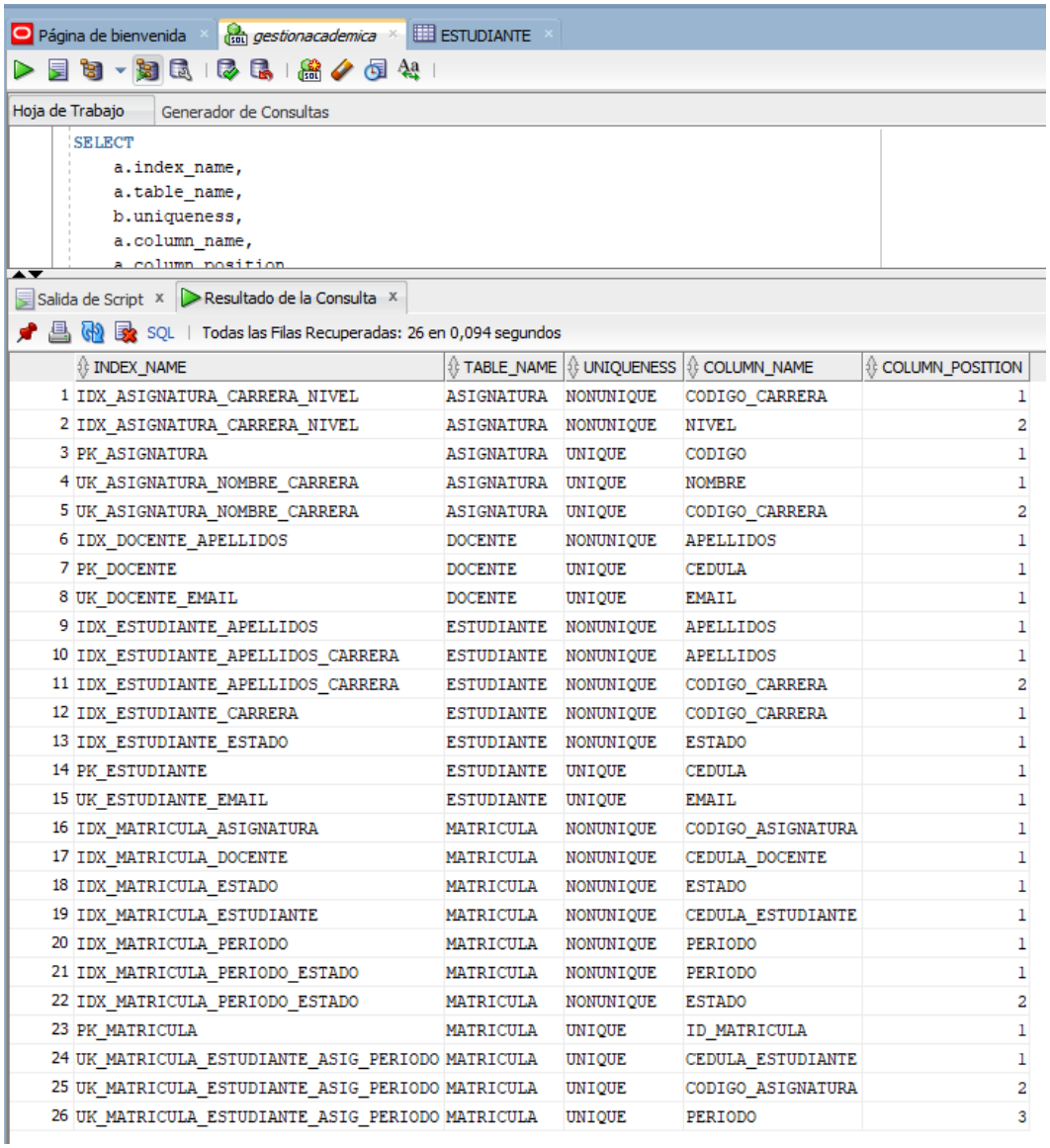


	INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION
1	IDX_ASIGNATURA_CARRERA_NIVEL	ASIGNATURA	CODIGO_CARRERA	1
2	IDX_ASIGNATURA_CARRERA_NIVEL	ASIGNATURA	NIVEL	2
3	PK_ASIGNATURA	ASIGNATURA	CODIGO	1
4	UK_ASIGNATURA_NOMBRE_CARRERA	ASIGNATURA	NOMBRE	1
5	UK_ASIGNATURA_NOMBRE_CARRERA	ASIGNATURA	CODIGO_CARRERA	2
6	IDX_DOCENTE_APELLIDOS	DOCENTE	APELLIDOS	1
7	PK_DOCENTE	DOCENTE	CEDULA	1
8	UK_DOCENTE_EMAIL	DOCENTE	EMAIL	1
9	IDX_ESTUDIANTE_APELLIDOS	ESTUDIANTE	APELLIDOS	1
10	IDX_ESTUDIANTE_APELLIDOS_CARRERA	ESTUDIANTE	APELLIDOS	1
11	IDX_ESTUDIANTE_APELLIDOS_CARRERA	ESTUDIANTE	CODIGO_CARRERA	2
12	IDX_ESTUDIANTE_CARRERA	ESTUDIANTE	CODIGO_CARRERA	1
13	IDX_ESTUDIANTE_ESTADO	ESTUDIANTE	ESTADO	1
14	PK_ESTUDIANTE	ESTUDIANTE	CEDULA	1
15	UK_ESTUDIANTE_EMAIL	ESTUDIANTE	EMAIL	1
16	IDX_MATRICULA_ASIGNATURA	MATRICULA	CODIGO_ASIGNATURA	1
17	IDX_MATRICULA_DOCENTE	MATRICULA	CEDULA_DOCENTE	1
18	IDX_MATRICULA_ESTADO	MATRICULA	ESTADO	1
19	IDX_MATRICULA_ESTUDIANTE	MATRICULA	CEDULA_ESTUDIANTE	1
20	IDX_MATRICULA_PERIODO	MATRICULA	PERIODO	1
21	IDX_MATRICULA_PERIODO_ESTADO	MATRICULA	PERIODO	1
22	IDX_MATRICULA_PERIODO_ESTADO	MATRICULA	ESTADO	2
23	PK_MATRICULA	MATRICULA	ID_MATRICULA	1
24	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	CEDULA_ESTUDIANTE	1
25	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	CODIGO_ASIGNATURA	2
26	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	PERIODO	3

-- : Mostrar campo uniqueness

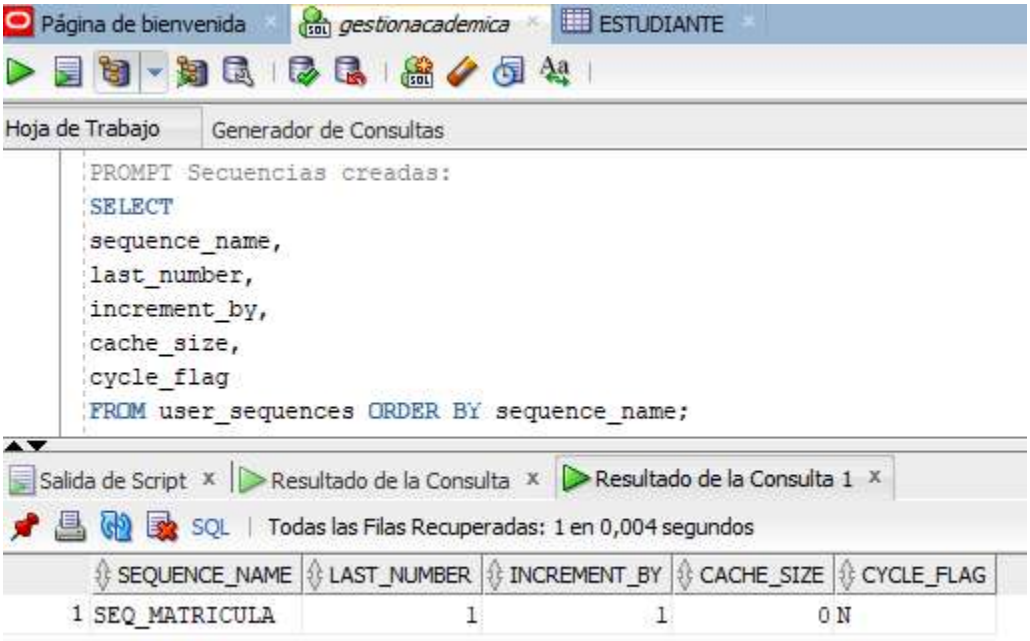
```
SELECT
```

```
    a.index_name,
    a.table_name,
    b.uniqueness,
    a.column_name,
    a.column_position
FROM
    user_ind_columns a
    JOIN user_indexes b ON a.index_name = b.index_name
WHERE
    a.table_name IN ('ESTUDIANTE', 'MATRICULA', 'ASIGNATURA', 'DOCENTE')
ORDER BY
    a.table_name,
    a.index_name,
    a.column_position;
```

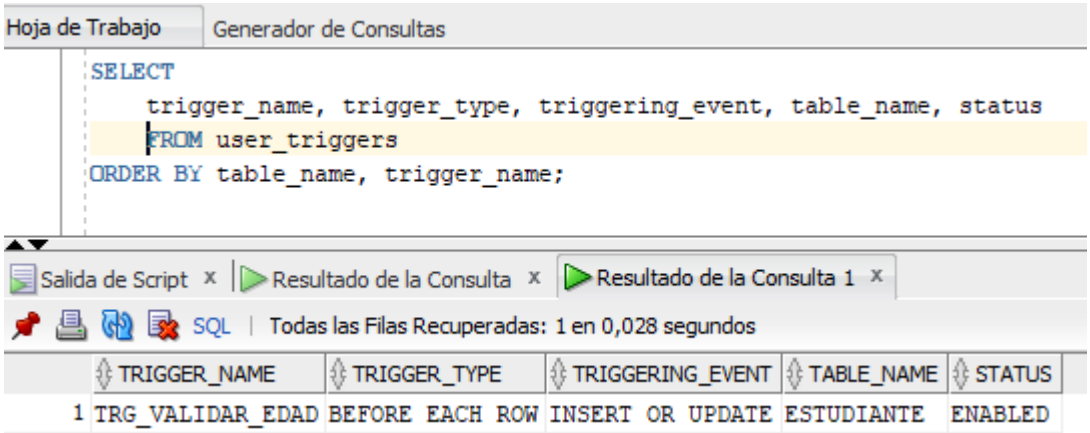


	INDEX_NAME	TABLE_NAME	UNIQUENESS	COLUMN_NAME	COLUMN_POSITION
1	IDX_ASIGNATURA_CARRERA_NIVEL	ASIGNATURA	NONUNIQUE	CODIGO_CARRERA	1
2	IDX_ASIGNATURA_CARRERA_NIVEL	ASIGNATURA	NONUNIQUE	NIVEL	2
3	PK_ASIGNATURA	ASIGNATURA	UNIQUE	CODIGO	1
4	UK_ASIGNATURA_NOMBRE_CARRERA	ASIGNATURA	UNIQUE	NOMBRE	1
5	UK_ASIGNATURA_NOMBRE_CARRERA	ASIGNATURA	UNIQUE	CODIGO_CARRERA	2
6	IDX_DOCENTE_APELLIDOS	DOCENTE	NONUNIQUE	APELLIDOS	1
7	PK_DOCENTE	DOCENTE	UNIQUE	CEDULA	1
8	UK_DOCENTE_EMAIL	DOCENTE	UNIQUE	EMAIL	1
9	IDX_ESTUDIANTE_APELLIDOS	ESTUDIANTE	NONUNIQUE	APELLIDOS	1
10	IDX_ESTUDIANTE_APELLIDOS_CARRERA	ESTUDIANTE	NONUNIQUE	APELLIDOS	1
11	IDX_ESTUDIANTE_APELLIDOS_CARRERA	ESTUDIANTE	NONUNIQUE	CODIGO_CARRERA	2
12	IDX_ESTUDIANTE_CARRERA	ESTUDIANTE	NONUNIQUE	CODIGO_CARRERA	1
13	IDX_ESTUDIANTE_ESTADO	ESTUDIANTE	NONUNIQUE	ESTADO	1
14	PK_ESTUDIANTE	ESTUDIANTE	UNIQUE	CEDULA	1
15	UK_ESTUDIANTE_EMAIL	ESTUDIANTE	UNIQUE	EMAIL	1
16	IDX_MATRICULA_ASIGNATURA	MATRICULA	NONUNIQUE	CODIGO_ASIGNATURA	1
17	IDX_MATRICULA_DOCENTE	MATRICULA	NONUNIQUE	CEDULA_DOCENTE	1
18	IDX_MATRICULA_ESTADO	MATRICULA	NONUNIQUE	ESTADO	1
19	IDX_MATRICULA_ESTUDIANTE	MATRICULA	NONUNIQUE	CEDULA_ESTUDIANTE	1
20	IDX_MATRICULA_PERIODO	MATRICULA	NONUNIQUE	PERIODO	1
21	IDX_MATRICULA_PERIODO_ESTADO	MATRICULA	NONUNIQUE	PERIODO	1
22	IDX_MATRICULA_PERIODO_ESTADO	MATRICULA	NONUNIQUE	ESTADO	2
23	PK_MATRICULA	MATRICULA	UNIQUE	ID_MATRICULA	1
24	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	UNIQUE	CEDULA_ESTUDIANTE	1
25	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	UNIQUE	CODIGO_ASIGNATURA	2
26	UK_MATRICULA_ESTUDIANTE_ASIG_PERIODO	MATRICULA	UNIQUE	PERIODO	3

```
-- 9.6: Ver secuencias
PROMPT Secuencias creadas:
 SELECT
    sequence_name,
    last_number,
    increment_by,
    cache_size,
    cycle_flag
FROM user_sequences ORDER
BY sequence_name;
```




```
-- 9.7: Ver triggers
PROMPT Triggers creados:
SELECT
    trigger_name,
    trigger_type,
    triggering_event,
    table_name, status
FROM user_triggers
ORDER BY table_name, trigger_name;
```



Hoja de Trabajo | Generador de Consultas

```
SELECT
    trigger_name, trigger_type, triggering_event, table_name, status
    FROM user_triggers
ORDER BY table_name, trigger_name;
```

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

SQL | Todas las Filas Recuperadas: 2 en 0,003 segundos

	TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_NAME	STATUS
1	TRG_VALIDAR_EDAD	BEFORE EACH ROW	INSERT OR UPDATE	ESTUDIANTE	ENABLED
2	TRG_MATRICULA_ID	BEFORE EACH ROW	INSERT	MATRICULA	ENABLED

```
-- 9.8: Ver comentarios de tablas
PROMPT Comentarios de tablas:
SELECT
   table_name,comments
      FROM user_tab_comments
      WHERE table_name IN ('CARRERA', 'ESTUDIANTE', 'ASIGNATURA', 'DOCENTE',
      'MATRICULA', 'PRERREQUISITO')
      ORDER BY table_name;
```

Hoja de Trabajo | Generador de Consultas

```
WHERE table_name IN ('CARRERA', 'ESTUDIANTE', 'ASIGNATURA', 'DOCENTE',
'MATRICULA', 'PRERREQUISITO')
ORDER BY table_name;
```

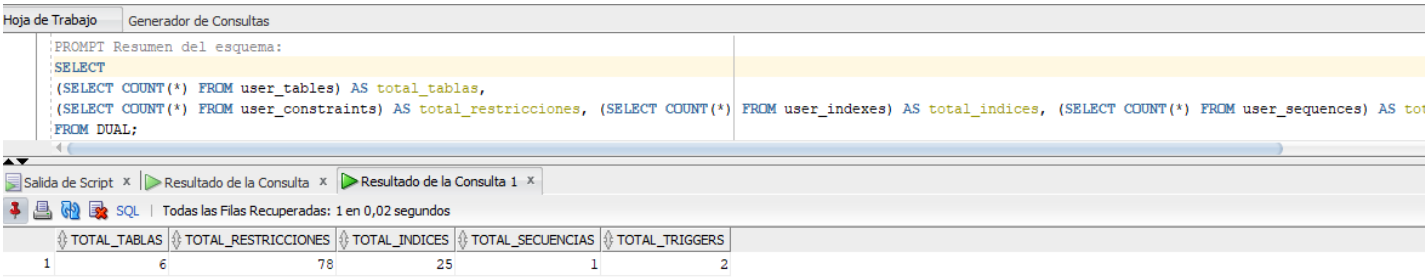
Salida de Script | Resultado de la Consulta | Resultado de la Consulta 1

SQL | Todas las Filas Recuperadas: 6 en 0,014 segundos

	TABLE_NAME	COMMENTS
1	ASIGNATURA	Cat logo de asignaturas organizadas por carrera y nivel
2	CARRERA	Cat logo de carreras ofertadas por la instituci◆n
3	DOCENTE	Informaci◆n de docentes de la instituci◆n
4	ESTUDIANTE	Informaci◆n personal y acad,mica de estudiantes matriculados en la instituci◆n
5	MATRICULA	Registro de matr¡culas de estudiantes en asignaturas por periodo acad,mico
6	PRERREQUISITO	Relaci◆n de prerrequisitos entre asignaturas

```
-- 9.9: Estadísticas del esquema
PROMPT Resumen del esquema:
SELECT
   (SELECT COUNT(*) FROM user_tables) AS total_tablas,
   (SELECT COUNT(*) FROM user_constraints) AS
```

```
    total_restricciones, (SELECT COUNT(*) FROM user_indexes) AS
    total_indices, (SELECT COUNT(*) FROM user_sequences) AS
    total_secuencias, (SELECT COUNT(*) FROM user_triggers) AS
    total_triggers
FROM DUAL;
```



PASO 10: Eliminar y Vaciar Objetos (DROP y TRUNCATE)

```
-- ============================================================
-- PARTE A: COMANDO DROP (Eliminar Objetos)
-- ============================================================

PROMPT Ejemplos de DROP - Eliminar objetos de la BD

-- 10.1: DROP TABLE - Eliminar una tabla
PROMPT 1. DROP TABLE - Eliminar tabla completa

-- Crear tabla de prueba
CREATE TABLE
    TablaTemporalPrueba (
    id NUMBER
    PRIMARY KEY,
    descripcion VARCHAR2(100)
);
```

```
SQL> CREATE TABLE TablaTemporalPrueba ( id NUMBER PRIMARY KEY,
  2  descripcion VARCHAR2(100)
  3  );

Tabla creada.
```

```
-- Insertar datos
INSERT INTO TablaTemporalPrueba VALUES (1, 'Registro 1');
```

```
INSERT INTO TablaTemporalPrueba VALUES (2,
'Registro 2');
 COMMIT;
```

```
Disparador creado.

SQL> CREATE TABLE TablaTemporalPrueba ( id NUMBER PRIMARY KEY,
  2  descripcion VARCHAR2(100)
  3  );

Tabla creada.

SQL> INSERT INTO TablaTemporalPrueba VALUES (1, 'Registro 1');

1 fila creada.

SQL> INSERT INTO TablaTemporalPrueba VALUES (2, 'Registro 2');

1 fila creada.

SQL>  COMMIT;

Confirmaci¾n terminada.
```

```
-- Verificar que existe
SELECT COUNT(*) FROM TablaTemporalPrueba;
```

```
SQL> SELECT COUNT(*) FROM TablaTemporalPrueba;

  COUNT(*)
----------
         2

SQL>
```

```
-- Eliminar la tabla (estructura y datos)
DROP TABLE TablaTemporalPrueba;
```

```
SQL> DROP TABLE TablaTemporalPrueba;

Tabla borrada.
```

```
-- Intentar consultar (debe dar error)
-- SELECT * FROM TablaTemporalPrueba;
```

```
SQL> SELECT * FROM TablaTemporalPrueba;
SELECT * FROM TablaTemporalPrueba
              *
ERROR en lÝnea 1:
ORA-00942: la tabla o vista no existe
```

```
PROMPT Tabla eliminada con DROP TABLE

-- 10.2: DROP TABLE CASCADE CONSTRAINTS
PROMPT 2. DROP TABLE CASCADE CONSTRAINTS

-- Crear tablas con relaciones
CREATE
   TABLE
   TablaPadre ( id
   NUMBER
   PRIMARY
   KEY, nombre
   VARCHAR2(5
   0));
```

```
SQL> CREATE TABLE TablaPadre ( id NUMBER PRIMARY KEY, nombre VARCHAR2(50));

Tabla creada.
```

```
   CREATE TABLE TablaHija (
      id NUMBER PRIMARY KEY,
      id_padre
      NUMBER,
      descripcion
      VARCHAR2(1
      00),
      CONSTRAINT fk_hija_padre FOREIGN KEY (id_padre) REFERENCES TablaPadre(id)
   );
```

```
SQL> CREATE TABLE TablaHija (
  2  id NUMBER PRIMARY KEY,
  3  id_padre NUMBER, descripcion VARCHAR2(100),
  4  CONSTRAINT fk_hija_padre FOREIGN KEY (id_padre) REFERENCES TablaPadre(id)
  5  );

Tabla creada.
```

```
-- Intentar eliminar tabla padre (da error por FK)
-- DROP TABLE TablaPadre;
```

```
SQL> DROP TABLE TablaPadre;
DROP TABLE TablaPadre
           *
ERROR en lÝnea 1:
ORA-02449: claves ·nicas/primarias en la tabla referidas por claves ajenas
```

```
-- Eliminar con CASCADE CONSTRAINTS (elimina las FK que referencian)
DROP TABLE TablaPadre CASCADE CONSTRAINTS;
```

```
SQL> DROP TABLE TablaPadre CASCADE CONSTRAINTS;

Tabla borrada.
```

```
-- La tabla hija sigue existiendo pero sin la FK
DESC TablaHija;
```

```
SQL> DROP TABLE TablaPadre CASCADE CONSTRAINTS;

Tabla borrada.

SQL> DESC TablaHija;
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 ID_PADRE                                           NUMBER
 DESCRIPCION                                        VARCHAR2(100)
```

```
-- Limpiar
DROP TABLE TablaHija;
```

```
SQL> DROP TABLE TablaHija;

Tabla borrada.
```

```
PROMPT Tabla eliminada con CASCADE CONSTRAINTS

-- 10.3: DROP TABLE PURGE (Sin papelera de reciclaje)
PROMPT 3. DROP TABLE con PURGE

CREATE TABLE
    TablaConPurge (
    id NUMBER
    PRIMARY
    KEY, dato
    VARCHAR2(10
    0)
);
```

```
SQL> CREATE TABLE TablaConPurge ( id NUMBER PRIMARY KEY, dato VARCHAR2(100)
  2  );

Tabla creada.
```

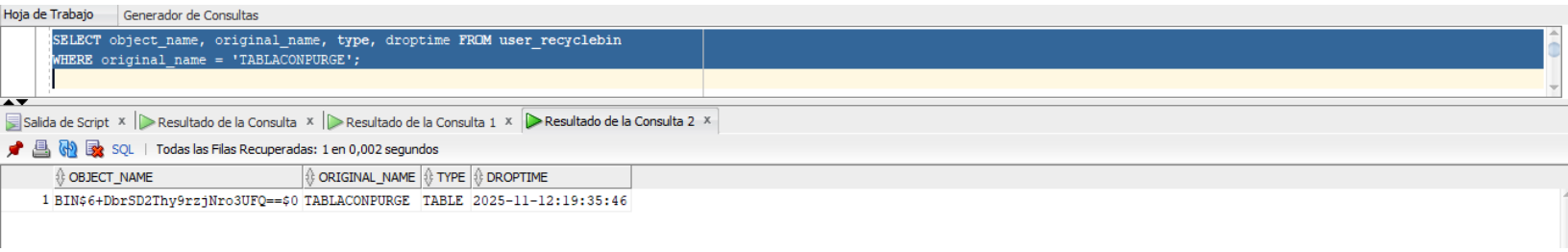
```
-- DROP normal (va a la papelera de reciclaje)
DROP TABLE TablaConPurge;
```

```
SQL> DROP TABLE TablaConPurge;

Tabla borrada.

SQL>
```

```
-- Ver papelera de reciclaje
SELECT object_name, original_name, type,
droptime FROM user_recyclebin
WHERE original_name = 'TABLACONPURGE';
```




```
-- Recuperar tabla de la papelera
FLASHBACK TABLE TablaConPurge TO BEFORE DROP;
```

```
SQL> FLASHBACK TABLE TablaConPurge TO BEFORE DROP;

Flashback terminado.

SQL> desc TablaConPurge
 Nombre                                    ¬Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 DATO                                               VARCHAR2(100)
```

```
-- Ahora eliminar permanentemente con PURGE
DROP TABLE TablaConPurge PURGE;
```

```
SQL> DROP TABLE TablaConPurge PURGE;

Tabla borrada.
```

```
-- Verificar que no está en papelera
SELECT COUNT(*) FROM user_recyclebin WHERE original_name =

'TABLACONPURGE';
```

```
SQL> SELECT COUNT(*) FROM user_recyclebin WHERE original_name = 'TABLACONPURGE';

  COUNT(*)
----------
         0
```

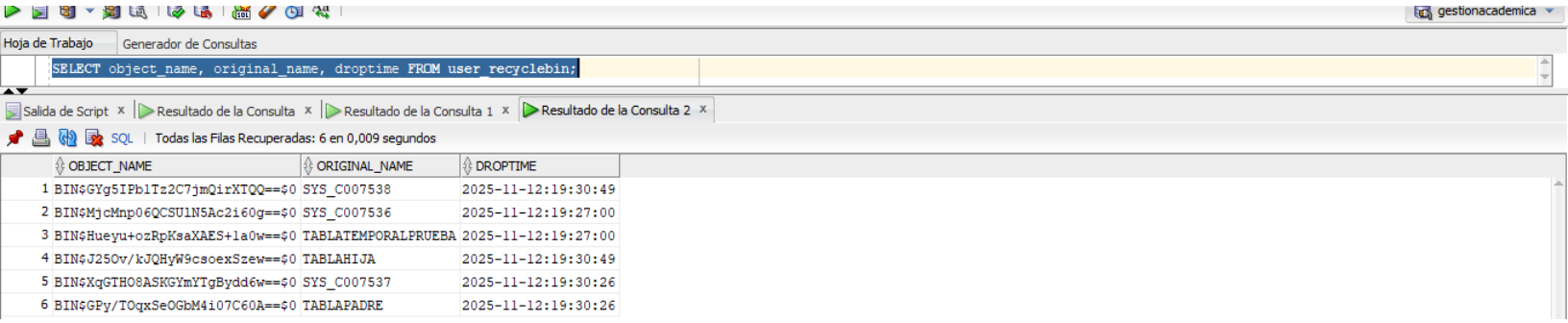
PROMPT DROP con PURGE elimina permanentemente

-- 10.4: Vaciar la papelera de reciclaje

PROMPT 4. Vaciar papelera de reciclaje

-- Ver contenido de papelera

SELECT object_name, original_name, droptime FROM user_recyclebin;

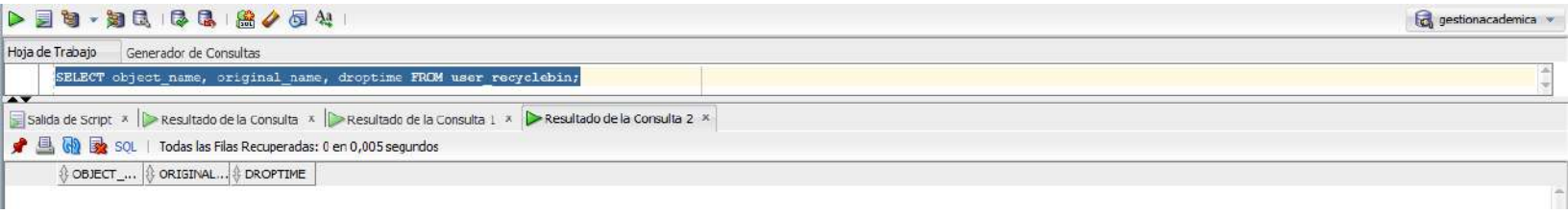


-- Vaciar papelera completamente

PURGE RECYCLEBIN;

```
SQL> PURGE RECYCLEBIN;

Papelera de reciclaje depurada.
```



-- 10.5: DROP SEQUENCE

PROMPT 5. DROP SEQUENCE - Eliminar secuencia

CREATE SEQUENCE seq_prueba_drop START WITH 1 INCREMENT BY 1;

```
SQL> CREATE SEQUENCE seq_prueba_drop START WITH 1 INCREMENT BY 1;

Secuencia creada.
```

```
-- Verificar que existe
SELECT sequence_name FROM user_sequences WHERE sequence_name = 'SEQ_PRUEBA_DROP';
```

```
SQL> SELECT sequence_name FROM user_sequences WHERE sequence_name = 'SEQ_PRUEBA_DROP';

SEQUENCE_NAME
--------------------------------------------------------------------------------
SEQ_PRUEBA_DROP
```

```
-- Eliminar secuencia
DROP SEQUENCE

seq_prueba_drop;
```

```
SQL> DROP SEQUENCE seq_prueba_drop;

Secuencia borrada.
```

```
-- 10.6: DROP INDEX
PROMPT 6. DROP INDEX - Eliminar índice

CREATE TABLE
    TablaConIndice (
    id NUMBER
    PRIMARY KEY,
    columna1
    VARCHAR
    2(50),
    columna2
    VARCHAR
    2(50)
);
```

```
SQL> SELECT sequence_name FROM user_sequences WHERE sequence_name = 'SEQ_PRUEBA

SEQUENCE_NAME
--------------------------------------------------------------------------------
SEQ_PRUEBA_DROP

SQL> DROP SEQUENCE seq_prueba_drop;

Secuencia borrada.

SQL> CREATE TABLE TablaConIndice ( id NUMBER PRIMARY KEY,
  2  columna1 VARCHAR2(50), columna2 VARCHAR2(50)
  3  );

Tabla creada.
```

```
-- Crear índice
CREATE INDEX idx_prueba_drop ON TablaConIndice(columna1);
```

```
SQL> CREATE INDEX idx_prueba_drop ON TablaConIndice(columna1);

=ndice creado.
```

```
-- Verificar índice
SELECT index_name, table_name FROM user_indexes WHERE index_name = 'IDX_PRUEBA_DROP';
```

```
SQL> SELECT index_name, table_name FROM user_indexes WHERE index_name = 'IDX_PRUEBA_DROP
';

INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
IDX_PRUEBA_DROP
TABLACONINDICE
```

```
-- Eliminar índice
DROP INDEX idx_prueba_drop;
```

```
SQL> DROP INDEX idx_prueba_drop;

=ndice borrado.
```

```
-- Limpiar tabla
DROP TABLE
```

TablaConIndice PURGE;

```
SQL> DROP TABLE TablaConIndice PURGE;

Tabla borrada.
```

-- 10.7: DROP VIEW

PROMPT 7. DROP VIEW - Eliminar vista

```
CREATE VIEW
vista_estudiantes_activos AS
SELECT cedula, nombres,
apellidos, email FROM Estudiante
WHERE estado = 'ACTIVO';
```

```
SQL> CREATE VIEW vista_estudiantes_activos AS SELECT cedula, nombres, apellidos, email F
ROM Estudiante
  2  WHERE estado = 'ACTIVO';

Vista creada.
```

-- Verificar vista

```
SELECT view_name FROM user_views WHERE view_name = 'VISTA_ESTUDIANTES_ACTIVOS';
```

```
SQL> SELECT view_name FROM user_views WHERE view_name = 'VISTA_ESTUDIANTES_ACTIVOS';

VIEW_NAME
--------------------------------------------------------------------------------
VISTA_ESTUDIANTES_ACTIVOS
```

-- Eliminar vista

```
DROP VIEW  vista_estudiantes_activos;
```

```
SQL> DROP VIEW  vista_estudiantes_activos;

Vista borrada.
```

-- 10.8: DROP CONSTRAINT

PROMPT 8. ALTER TABLE DROP CONSTRAINT - Eliminar restricción

```
CREATE TABLE
```

```
    TablaConConstraint (
    id NUMBER
    PRIMARY KEY,
    email VARCHAR2(100),
    edad NUMBER,
    CONSTRAINT uk_email UNIQUE
    (email), CONSTRAINT chk_edad
    CHECK (edad >= 18)
);
```

```
SQL> CREATE TABLE TablaConConstraint ( id NUMBER PRIMARY KEY,
  2  email VARCHAR2(100),
  3  edad NUMBER,
  4  CONSTRAINT uk_email UNIQUE (email), CONSTRAINT chk_edad CHECK (edad >= 18)
  5  );

Tabla creada.
```

```
-- Ver restricciones
SELECT constraint_name,
constraint_type FROM
user_constraints
WHERE table_name = 'TABLACONCONSTRAINT';

-- Eliminar restricción UNIQUE
ALTER TABLE TablaConConstraint DROP CONSTRAINT uk_email;
```

```
SQL> SELECT constraint_name, constraint_type FROM user_constraints
  2  WHERE table_name = 'TABLACONCONSTRAINT';

CONSTRAINT_NAME
--------------------------------------------------------------------------------
C
-
CHK_EDAD
C

SYS_C007544
P

UK_EMAIL
U
```

```
-- Eliminar restricción CHECK
ALTER TABLE TablaConConstraint DROP CONSTRAINT chk_edad;
```

```
SQL> ALTER TABLE TablaConConstraint DROP CONSTRAINT chk_edad;

Tabla modificada.
```

```
-- Limpiar
DROP TABLE

TablaConConstraint PURGE;
```

```
SQL> DROP TABLE TablaConConstraint PURGE;

Tabla borrada.
```

```
-- ============================================================
-- PARTE B: COMANDO TRUNCATE (Vaciar Tablas)
-- ============================================================

PROMPT ========================================
PROMPT COMANDO TRUNCATE - Vaciar tablas
PROMPT ========================================

-- 10.9: TRUNCATE TABLE básico
PROMPT 9. TRUNCATE TABLE - Vaciar tabla manteniendo estructura

-- Crear tabla con datos
CREATE TABLE
    TablaParaTruncate (
    id NUMBER
    PRIMARY KEY,
    descripcion
    VARCHAR2(100),
    fecha_creacion DATE
    DEFAULT SYSDATE
);
```

```
SQL> CREATE TABLE TablaParaTruncate ( id NUMBER PRIMARY KEY,
  2  descripcion VARCHAR2(100), fecha_creacion DATE DEFAULT SYSDATE
  3  );

Tabla creada.
```

```
-- Insertar datos
INSERT INTO TablaParaTruncate (id, descripcion) VALUES (1, 'Registro 1'); INSERT INTO TablaParaTruncate (id, descripcion) VALUES (2, 'Registro 2'); INSERT INTO TablaParaTruncate (id, descripcion) VALUES (3, 'Registro 3'); INSERT INTO TablaParaTruncate (id, descripcion) VALUES (4, 'Registro 4'); INSERT INTO TablaParaTruncate (id, descripcion) VALUES (5, 'Registro 5'); COMMIT;
```

```
SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (1, 'Registro 1');

1 fila creada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (2, 'Registro 2');

1 fila creada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (3, 'Registro 3');

1 fila creada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (4, 'Registro 4');

1 fila creada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (5, 'Registro 5');

1 fila creada.

SQL> COMMIT;

Confirmaci¾n terminada.

SQL>
```

```
-- Verificar datos
SELECT COUNT(*) AS total_registros FROM TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) AS total_registros FROM TablaParaTruncate;

TOTAL_REGISTROS
---------------
              5
```

```
-- TRUNCATE elimina todos los datos pero mantiene la estructura
TRUNCATE TABLE TablaParaTruncate;
```

```
SQL> TRUNCATE TABLE TablaParaTruncate;

Tabla truncada.
```

```
SQL> desc TablaParaTruncate
 Nombre                                    ┐Nulo?   Tipo
 ----------------------------------------- -------- ----------------------------
 ID                                        NOT NULL NUMBER
 DESCRIPCION                                        VARCHAR2(100)
 FECHA_CREACION                                     DATE
```

```
-- Verificar que la tabla existe pero sin datos
SELECT COUNT(*) AS total_registros FROM
TablaParaTruncate; DESC TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) AS total_registros
  2  FROM TablaParaTruncate;

TOTAL_REGISTROS
---------------
              0
```

```
PROMPT TRUNCATE eliminó datos pero mantuvo estructura

-- 10.10: TRUNCATE vs DELETE - Comparación
PROMPT 10. TRUNCATE vs DELETE - Comparación práctica

-- Recrear datos
INSERT INTO TablaParaTruncate (id, descripcion)
VALUES (1, 'Registro A');
INSERT INTO TablaParaTruncate (id, descripcion)
VALUES (2, 'Registro B');
INSERT INTO TablaParaTruncate (id, descripcion)
VALUES (3, 'Registro C');
COMMIT;
```

```
SQL> DESC TablaParaTruncate;INSERT INTO TablaParaTruncate (id, descripcion)
SP2-0565: Identificador no vßlido.
SQL> VALUES (1, 'Registro A');
SP2-0734: inicio "VALUES (1,..." de comando desconocido - resto de la lÝnea ig
SQL> INSERT INTO TablaParaTruncate (id, descripcion) VALUES (2, 'Registro B');

1 fila creada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion) VALUES (3, 'Registro C');

1 fila creada.

SQL> COMMIT;

Confirmaci¾n terminada.

SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (1, 'Registro A');

1 fila creada.
```

```
-- Estadísticas antes de TRUNCATE
SELECT COUNT(*) FROM TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) FROM TablaParaTruncate;

  COUNT(*)
----------
         3
```

```
-- TRUNCATE: Rápido, no genera UNDO, no se puede hacer ROLLBACK
TRUNCATE TABLE TablaParaTruncate;
```

```
SQL> TRUNCATE TABLE TablaParaTruncate;

Tabla truncada.
```

```
-- Verificar
SELECT COUNT(*) FROM TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) FROM TablaParaTruncate;

  COUNT(*)
----------
         0
```

```
-- Recrear para comparar con DELETE
```

INSERT INTO TablaParaTruncate (id, descripcion) VALUES (1, 'Registro X');

```
SQL> INSERT INTO TablaParaTruncate (id, descripcion) VALUES (1, 'Registro X');

1 fila creada.
```

```
INSERT INTO TablaParaTruncate (id, descripcion) VALUES (2,
'Registro Y'); COMMIT;
```

```
SQL> INSERT INTO TablaParaTruncate (id, descripcion)
  2  VALUES (2, 'Registro Y');

1 fila creada.

SQL> COMMIT;
```

```
-- DELETE: Más lento, genera UNDO, se puede hacer ROLLBACK
DELETE FROM TablaParaTruncate;
```

```
SQL> DELETE FROM TablaParaTruncate;

2 filas suprimidas.
```

```
-- Verificar
SELECT COUNT(*) FROM TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) FROM TablaParaTruncate;

  COUNT(*)
----------
         0
```

```
-- Como no hemos hecho COMMIT, podemos hacer ROLLBACK
ROLLBACK;
```

```
SQL> ROLLBACK;

Rollback terminado.
```

```
-- Los datos vuelven después de ROLLBACK
SELECT COUNT(*) FROM TablaParaTruncate;
```

```
SQL> SELECT COUNT(*) FROM TablaParaTruncate;

  COUNT(*)
----------
         2
```

```
PROMPT Diferencia demostrada: DELETE permite ROLLBACK, TRUNCATE no

-- 10.11: TRUNCATE con tablas relacionadas
PROMPT 11. TRUNCATE con Foreign Keys

-- Crear tablas padre e hija
CREATE TABLE
    TruncatePadre (
    id NUMBER
    PRIMARY
    KEY,
    nombre VARCHAR2(50)
);

CREATE
    TABLE
    TruncateHija (
    id NUMBER
    PRIMARY
    KEY,
    id_padre
    NUMBER,
    descripcion
    VARCHAR2(1
    00),
    CONSTRAINT fk_truncate FOREIGN KEY (id_padre) REFERENCES TruncatePadre(id)
);
```

```
SQL> CREATE TABLE TruncatePadre ( id NUMBER PRIMARY KEY,
  2  nombre VARCHAR2(50)
  3  );

Tabla creada.

SQL>
SQL> CREATE TABLE TruncateHija ( id NUMBER PRIMARY KEY,
  2  id_padre NUMBER, descripcion VARCHAR2(100),
  3  CONSTRAINT fk_truncate FOREIGN KEY (id_padre) REFERENCES TruncatePadre(id
  4  );
```

```
-- Insertar datos
INSERT INTO TruncatePadre VALUES (1, 'Padre 1');
INSERT INTO TruncatePadre VALUES (2, 'Padre 2');
INSERT INTO TruncateHija VALUES (1, 1, 'Hija 1-1');
INSERT INTO TruncateHija VALUES (2, 1, 'Hija 1-2');
INSERT INTO TruncateHija VALUES (3, 2, 'Hija 2-1');
COMMIT;
```

```
SQL> INSERT INTO TruncatePadre VALUES (1, 'Padre 1');

1 fila creada.

SQL> INSERT INTO TruncatePadre VALUES (2, 'Padre 2');

1 fila creada.

SQL> INSERT INTO TruncateHija VALUES (1, 1, 'Hija 1-1');

1 fila creada.

SQL> INSERT INTO TruncateHija VALUES (2, 1, 'Hija 1-2');

1 fila creada.

SQL> INSERT INTO TruncateHija VALUES (3, 2, 'Hija 2-1');

1 fila creada.

SQL>  COMMIT;

Confirmaci¾n terminada.
```

```
-- Intentar TRUNCATE en tabla padre (ERROR por FK)
-- TRUNCATE TABLE TruncatePadre;
-- ERROR: ORA-02266: unique/primary keys referenced by enabled foreign keys

-- Deshabilitar FK temporalmente
ALTER TABLE TruncateHija DISABLE CONSTRAINT fk_truncate;

-- Ahora sí se puede hacer TRUNCATE
TRUNCATE TABLE TruncatePadre;

-- Habilitar FK nuevamente
ALTER TABLE TruncateHija ENABLE CONSTRAINT fk_truncate;

-- Limpiar
DROP TABLE TruncateHija
PURGE;
DROP TABLE TruncatePadre
PURGE;

PROMPT TRUNCATE requiere deshabilitar FK primero

-- 10.12: TRUNCATE con reset de secuencias
PROMPT 12. TRUNCATE y reinicio de secuencias

CREATE SEQUENCE seq_truncate_test START WITH 1 INCREMENT

BY 1;

CREATE TABLE TablaTruncateSeq (
    id NUMBER DEFAULT seq_truncate_test.NEXTVAL
    PRIMARY KEY, dato VARCHAR2(100)
);

-- Insertar usando secuencia
INSERT INTO TablaTruncateSeq (dato) VALUES ('Dato 1');
INSERT INTO TablaTruncateSeq (dato) VALUES ('Dato 2');
INSERT INTO TablaTruncateSeq (dato) VALUES ('Dato 3');
COMMIT;
```

```
-- Ver último ID
SELECT MAX(id) FROM TablaTruncateSeq;

-- Ver valor actual de secuencia
SELECT seq_truncate_test.CURRVAL FROM DUAL;
-- TRUNCATE no reinicia la secuencia
TRUNCATE TABLE TablaTruncateSeq;

-- Insertar nuevo dato
INSERT INTO TablaTruncateSeq (dato) VALUES ('Dato
Nuevo'); COMMIT;

-- El ID continúa desde donde estaba la secuencia (4)
SELECT id, dato FROM TablaTruncateSeq;

-- Para reiniciar secuencia hay que eliminarla y recrearla
DROP SEQUENCE seq_truncate_test;
CREATE SEQUENCE seq_truncate_test START WITH 1 INCREMENT BY 1;

-- Limpiar
DROP TABLE TablaTruncateSeq PURGE;

PROMPT TRUNCATE no reinicia secuencias automáticamente

-- Limpiar tabla de ejemplos
DROP TABLE TablaParaTruncate PURGE;

-- ============================================================
-- PARTE C: COMPARACIÓN DROP vs TRUNCATE vs DELETE
-- ============================================================

PROMPT
====================================
=== PROMPT COMPARACIÓN: DROP vs
TRUNCATE vs DELETE PROMPT
====================================
```

==

```
-- Crear tabla de demostración
CREATE TABLE
    ComparacionComandos (
    id NUMBER PRIMARY
    KEY,
    comando
    VARCHAR2(20),
    descripcion
    VARCHAR2(200)
);

-- Insertar información comparativa
INSERT INTO ComparacionComandos VALUES (1, 'DELETE', 'Elimina filas. Permite WHERE. Genera UNDO. Permite ROLLBACK. Más lento.');

INSERT INTO ComparacionComandos VALUES (2, , 'TRUNCATE', 'Elimina todas las filas. No permite WHERE. No genera UNDO. No permite ROLLBACK, Rapido');

INSERT INTO ComparacionComandos VALUES (3, , 'DROP', 'Elimina la tabla completa (estructura y datos). No se puede recuperar sin FLASHBACK.');
```

```
COMMIT;

-- Mostrar comparación
SELECT * FROM ComparacionComandos ORDER BY id;

-- ============================================================
-- CUADRO COMPARATIVO DETALLADO
-- ============================================================

/*
```

CARACTERÍSTICA	DELETE	TRUNCATE	DROP
Tipo de comando	DML	DDL	DDL
Qué elimina	Filas (datos)	Todas las filas	Tabla completa (estructura + datos)
Estructura	Se mantiene	Se mantiene	Se elimina
Cláusula WHERE	Sí	No	No
ROLLBACK	Sí	No	No
COMMIT	Requerido	Automático	Automático
Genera UNDO	Sí	No	No
Velocidad	Lento	Rápido	Rápido
Triggers	Se disparan	No se disparan	No se disparan
Con Foreign Key	Funciona	Requiere deshabilitar FK	Requiere CASCADE
Espacio	No libera inmediatamente	Libera inmediatamente	Libera totalmente
High Water Mark	No cambia	Resetea	N/A
Recuperación	ROLLBACK antes de COMMIT	No (sin backup)	FLASHBACK (si no PURGE)

CUÁNDO USAR CADA UNO:

DELETE:

✓ Necesitas eliminar filas específicas con WHERE
✓ Necesitas posibilidad de ROLLBACK
✓ Quieres que se disparen triggers
✓ Trabajas con pocas filas

TRUNCATE:
✓ Necesitas vaciar completamente una tabla
✓ Tabla tiene millones de registros
✓ No necesitas ROLLBACK
✓ Quieres liberar espacio inmediatamente
✓ Rendimiento es crítico

DROP:
✓ Ya no necesitas la tabla
✓ Vas a recrear la tabla con estructura diferente
✓ Limpieza final de objetos de prueba
✓ Deseas eliminar estructura y datos

*/

```
-- Limpiar tabla de comparación
DROP TABLE ComparacionComandos PURGE;

PROMPT ========================================
PROMPT Sección DROP y TRUNCATE completada
PROMPT ========================================
```

PASO 11: Ejemplos Prácticos de DROP y TRUNCATE

sql

```
-- ================================================
-- EJEMPLOS PRÁCTICOS - DROP Y TRUNCATE
-- ================================================

PROMPT
=====================================
=== PROMPT CASOS PRÁCTICOS DE USO
PROMPT =====================================

-- CASO 1: Limpieza de datos de prueba con TRUNCATE
PROMPT CASO 1: Limpiar datos de prueba manteniendo estructura

-- Crear tabla de logs de prueba
CREATE
   TABLE
   LogsPrueba (
   id NUMBER
   PRIMARY
   KEY,
   fecha_log DATE
   DEFAULT SYSDATE,
   mensaje
   VARCHAR2(500)
);

-- Simular carga de datos de prueba
BEGIN
   FOR i IN 1..1000 LOOP
      INSERT INTO LogsPrueba
      (id, mensaje) VALUES (i,
      'Log de prueba número ' || i);
   END LOOP;
   COMMIT;
END;
/

SELECT COUNT(*) AS total_logs FROM LogsPrueba;
```

```
-- Limpiar rápidamente todos los datos de prueba
TRUNCATE TABLE LogsPrueba;

-- La tabla queda lista para nuevas pruebas
SELECT COUNT(*) FROM LogsPrueba;

PROMPT Datos de prueba limpiados con TRUNCATE

-- CASO 2: Eliminar tabla temporal con DROP
PROMPT CASO 2: Eliminar completamente tabla temporal
```

```
-- La tabla ya no se necesita
DROP TABLE LogsPrueba

PURGE; PROMPT Tabla

temporal eliminada con DROP

-- CASO 3: Recreación de tabla con DROP y CREATE
PROMPT CASO 3: Recrear tabla con nueva estructura

-- Tabla antigua
CREATE TABLE
    ConfiguracionVieja (
    id NUMBER
    PRIMARY KEY,
    parametro
    VARCHAR
    2(50), valor
    VARCHAR
    2(100)
);

-- Eliminar y recrear con nueva estructura
DROP TABLE ConfiguracionVieja PURGE;

CREATE TABLE
    Configuracion (
    id NUMBER
    PRIMARY
    KEY,
    categoria
    VARCHAR
    2(50),
    parametro
    VARCHAR
    2(50), valor
    VARCHAR
```

```
    2(200),
    tipo_dato VARCHAR2(20),
    fecha_modificacion DATE DEFAULT SYSDATE,
    CONSTRAINT chk_tipo_dato CHECK (tipo_dato IN ('STRING', 'NUMBER', 'BOOLEAN', 'DATE'))
);

PROMPT Tabla recreada con nueva estructura

-- CASO 4: Mantenimiento periódico con TRUNCATE
PROMPT CASO 4: Mantenimiento de tabla de auditoría

CREATE TABLE
    AuditoriaAccesos (
    id NUMBER
    PRIMARY KEY,
    usuario
    VARCHA
    R2(50),
    fecha_acc
    eso
    DATE,
    accion
    VARCHA
    R2(100)
);

-- Simular datos de auditoría
```

```
INSERT INTO AuditoriaAccesos VALUES (1, 'user1', SYSDATE-30,
'LOGIN'); INSERT INTO AuditoriaAccesos VALUES (2, 'user2',
SYSDATE-25, 'QUERY'); INSERT INTO AuditoriaAccesos VALUES (3,
'user1', SYSDATE-20, 'UPDATE'); COMMIT;

-- Antes de truncar, respaldar datos importantes (si es necesario)
CREATE TABLE AuditoriaAccesos_Backup AS SELECT * FROM AuditoriaAccesos;

-- Limpiar tabla de auditoría (mantenimiento mensual)
TRUNCATE TABLE

AuditoriaAccesos; PROMPT

Auditoría limpiada, backup creado

-- Limpiar ejemplos
DROP TABLE AuditoriaAccesos PURGE;
DROP TABLE
AuditoriaAccesos_Backup PURGE;
DROP TABLE Configuracion
PURGE;

-- CASO 5: Recuperación desde papelera de reciclaje
PROMPT CASO 5: Recuperar tabla eliminada accidentalmente

CREATE TABLE
   DatosImportantes (
   id NUMBER
   PRIMARY KEY,
   informacion VARCHAR2(200)
);

INSERT INTO DatosImportantes VALUES (1,
'Dato crítico 1'); INSERT INTO DatosImportantes
VALUES (2, 'Dato crítico 2'); COMMIT;

-- Eliminación accidental (sin PURGE)
```

```
DROP TABLE DatosImportantes;

-- ¡Ups! Era importante, recuperar de papelera
SHOW RECYCLEBIN;

-- Recuperar tabla
FLASHBACK TABLE DatosImportantes TO BEFORE DROP;

-- Verificar datos recuperados
SELECT * FROM DatosImportantes;
```

```
    PROMPT Tabla recuperada exitosamente desde papelera

    -- Limpiar
    DROP TABLE DatosImportantes PURGE;

    PROMPT ========================================
    PROMPT Ejemplos prácticos completados
PROMPT =
```

EJERCICIOS PRÁCTICOS

Ejercicio 1: Creación y Modificación de Tablas

Objetivo: Practicar la creación de tablas con restricciones y su posterior modificación en Oracle.

Instrucciones:

1. Crear una nueva tabla llamada `Aula` con la siguiente estructura:

```
CREATE TABLE Aula (
        codigo VARCHAR2(10) NOT NULL,
        nombre VARCHAR2(50) NOT NULL,
        edificio VARCHAR2(50) NOT NULL,
        capacidad NUMBER(3) NOT NULL,
        tipo_aula VARCHAR2(20) DEFAULT 'NORMAL' NOT NULL,
        tiene_proyector NUMBER(1) DEFAULT 0 NOT NULL,
        CONSTRAINT pk_aula PRIMARY KEY (codigo),
        CONSTRAINT chk_aula_capacidad CHECK (capacidad > 0 AND capacidad <= 200),
        CONSTRAINT chk_aula_tipo CHECK (tipo_aula IN ('NORMAL', 'LABORATORIO', 'AUDITORIO')),
        CONSTRAINT chk_aula_proyector CHECK (tiene_proyector IN (0, 1))
    );
```

```
SQL> CREATE TABLE Aula (
  2  codigo VARCHAR2(10) NOT NULL, nombre VARCHAR2(50) NOT NULL,
 edificio VARCHAR2(50) NOT NULL,
  3  capacidad NUMBER(3) NOT NULL,
  4  tipo_aula VARCHAR2(20) DEFAULT 'NORMAL' NOT NULL,
  5  tiene_proyector NUMBER(1) DEFAULT 0 NOT NULL,
  6  CONSTRAINT pk_aula PRIMARY KEY (codigo),
  7  CONSTRAINT chk_aula_capacidad CHECK (capacidad > 0 AND capa
cidad <= 200),
  8  CONSTRAINT chk_aula_tipo CHECK (tipo_aula IN ('NORMAL', 'LA
BORATORIO', 'AUDITORIO')),
  9  CONSTRAINT chk_aula_proyector CHECK (tiene_proyector IN (0,
 1))
 10  );

Tabla creada.
```

2. Agregar las siguientes columnas:

Estas columnas ya existen, por lo que se crea una con “Prueba” de prefijo

```
ALTER TABLE Aula ADD (
    piso NUMBER(2) DEFAULT 1 NOT NULL,
    estado VARCHAR2(20) DEFAULT 'DISPONIBLE'
);
```

```
SQL>
SQL> ALTER TABLE Aula ADD (
  2  pisoPrueba NUMBER(2) DEFAULT 1 NOT NULL,
  3  estadoPrueba VARCHAR2(20) DEFAULT 'DISPONIBLE'
  4  );

Tabla modificada.
```

3. Modificar la tabla `Asignatura` para agregar relación con Aula:

```
ALTER TABLE Asignatura ADD codigo_aula VARCHAR2(10);

ALTER TABLE Asignatura
ADD CONSTRAINT fk_asignatura_aula
    FOREIGN KEY (codigo_aula)
    REFERENCES Aula(codigo);
```

```
SQL> ALTER TABLE Asignatura
  2  ADD CONSTRAINT fk_asignatura_aula
  3  FOREIGN KEY (codigo_aula) REFERENCES Aula(codigo);

Tabla modificada.
```

4. Crear restricción CHECK compleja:

```
ALTER TABLE Aula
ADD CONSTRAINT chk_aula_auditorio
    CHECK (tipo_aula != 'AUDITORIO' OR capacidad >= 100);
```

```
SQL> ALTER TABLE Aula
  2  ADD CONSTRAINT chk_aula_auditorio
  3  CHECK (tipo_aula != 'AUDITORIO' OR capacidad >= 100);

Tabla modificada.
```

Ejercicio 2: Trabajo con Secuencias y Triggers

Objetivo: Crear secuencias y triggers personalizados.

Instrucciones:

1. Crear una tabla `HistorialAcademico`:

```
CREATE TABLE HistorialAcademico (
    id_historial NUMBER(10) NOT NULL,
    cedula_estudiante VARCHAR2(10) NOT NULL,
```

```
    periodo VARCHAR2(10) NOT NULL,
    promedio_periodo NUMBER(4,2),
    creditos_periodo NUMBER(2),
    creditos_acumulados NUMBER(3),
    fecha_registro DATE DEFAULT SYSDATE,
    CONSTRAINT pk_historial PRIMARY KEY (id_historial),
    CONSTRAINT fk_historial_estudiante
      FOREIGN KEY (cedula_estudiante)
      REFERENCES Estudiante(cedula)
      ON DELETE CASCADE
);
```

```
SQL> CREATE TABLE HistorialAcademico (
  2  id_historial NUMBER(10) NOT NULL,
  3  cedula_estudiante VARCHAR2(10) NOT NULL,
  4  periodo VARCHAR2(10) NOT NULL,
  5  promedio_periodo NUMBER(4,2),
  6  creditos_periodo NUMBER(2),
  7  creditos_acumulados NUMBER(3),
  8  fecha_registro DATE DEFAULT SYSDATE,
  9  CONSTRAINT pk_historial PRIMARY KEY (id_historial), CONSTRA
INT fk_historial_estudiante
 10  FOREIGN KEY (cedula_estudiante) REFERENCES Estudiante(codig
o_estudiante) ON DELETE CASCADE
 11  );

Tabla creada.
```

2. Crear secuencia:

```
CREATE SEQUENCE seq_historial
  START WITH 1
  INCREMENT BY 1
  NOCACHE;
```

```
SQL> CREATE SEQUENCE seq_historial START WITH 1
  2  INCREMENT BY 1 NOCACHE;

Secuencia creada.
```

3. Crear trigger de auto-incremento:

```
CREATE OR REPLACE TRIGGER trg_historial_id
BEFORE INSERT ON HistorialAcademico
FOR EACH ROW
BEGIN
   IF :NEW.id_historial IS NULL THEN
      SELECT seq_historial.NEXTVAL INTO :NEW.id_historial FROM DUAL;
   END IF;
END;
/
```

```
SQL> CREATE OR REPLACE TRIGGER trg_historial_id BEFORE INSERT ON
 HistorialAcademico
  2  FOR EACH ROW BEGIN
  3  IF :NEW.id_historial IS NULL THEN
  4  SELECT seq_historial.NEXTVAL INTO :NEW.id_historial FROM DU
AL; END IF;
  5  END;
  6  /

Disparador creado.
```

4. Probar el trigger insertando datos

```
INSERT INTO HistorialAcademico (
   cedula_estudiante,
   periodo,
   promedio_periodo,
   creditos_periodo,
   creditos_acumulados
) VALUES (
   '123',
   '2024-01',
   8.5,
   18,
   54
);
```

```
SQL> INSERT INTO HistorialAcademico (
  2      cedula_estudiante,
  3      periodo,
  4      promedio_periodo,
  5      creditos_periodo,
  6      creditos_acumulados
  7  ) VALUES (
  8      '123',
  9      '2024-01',
 10      8.5,
 11      18,
 12      54
 13  );
INSERT INTO HistorialAcademico (
*
ERROR en lÝnea 1:
ORA-02291: restricci¾n de integridad (SYSTEM.FK_HISTORIAL_ESTUDI
ANTE) violada -
clave principal no encontrada
```

INSERT INTO Estudiante (

codigo_estudiante,

nombre_estudiante,

apellido_estudiante,

cedula

) VALUES (

'123',

'Juan',

'Perez',

'1234567890'

);

```
SQL> INSERT INTO Estudiante (
  2      codigo_estudiante,
  3      nombre_estudiante,
  4      apellido_estudiante,
  5      cedula
  6  ) VALUES (
  7      '123',
  8      'Juan',
  9      'Perez',
 10      '1234567890'
 11  );

1 fila creada.
```

Ejercicio 3: Validaciones Complejas

Objetivo: Implementar validaciones de negocio complejas.

Instrucciones:

1. Crear tabla `Horario`:

```
CREATE TABLE Horario (
    id_horario NUMBER(10) NOT NULL,
    codigo_asignatura VARCHAR2(10) NOT NULL,
    codigo_aula VARCHAR2(10) NOT NULL,
    dia_semana VARCHAR2(10) NOT NULL,
    hora_inicio VARCHAR2(5) NOT NULL,
    hora_fin VARCHAR2(5) NOT NULL,
    periodo VARCHAR2(10) NOT NULL,
    CONSTRAINT pk_horario PRIMARY KEY (id_horario),
    CONSTRAINT fk_horario_asignatura
        FOREIGN KEY (codigo_asignatura)
        REFERENCES Asignatura(codigo),
    CONSTRAINT fk_horario_aula
        FOREIGN KEY (codigo_aula)
        REFERENCES Aula(codigo),
```

```
    CONSTRAINT chk_horario_dia CHECK (dia_semana IN
      ('LUNES', 'MARTES', 'MIERCOLES', 'JUEVES', 'VIERNES', 'SABADO')),
    CONSTRAINT chk_horario_horas CHECK (hora_fin > hora_inicio),
    CONSTRAINT uk_horario_aula_dia_hora UNIQUE
      (codigo_aula, dia_semana, hora_inicio, periodo)
);
```

```
SQL> CREATE TABLE Horario (
  2  id_horario NUMBER(10) NOT NULL, codigo_asignatura VARCHAR2(
10) NOT NULL, codigo_aula VARCHAR2(10) NOT NULL, dia_semana VARC
HAR2(10) NOT NULL, hora_inicio VARCHAR2(5) NOT NULL, hora_fin VA
RCHAR2(5) NOT NULL,
  3  periodo VARCHAR2(10) NOT NULL,
  4  CONSTRAINT pk_horario PRIMARY KEY (id_horario), CONSTRAINT
fk_horario_asignatura
  5  FOREIGN KEY (codigo_asignatura) REFERENCES Asignatura(codig
o),
  6  CONSTRAINT fk_horario_aula FOREIGN KEY (codigo_aula) REFERE
NCES Aula(codigo),
  7  CONSTRAINT chk_horario_dia CHECK (dia_semana IN
  8  ('LUNES', 'MARTES', 'MIERCOLES', 'JUEVES', 'VIERNES', 'SABA
DO')),
  9  CONSTRAINT chk_horario_horas CHECK (hora_fin > hora_inicio)
,
 10  CONSTRAINT uk_horario_aula_dia_hora UNIQUE (codigo_aula, di
a_semana, hora_inicio, periodo)
 11  );

Tabla creada.
```

2. Crear índices apropiados

```
-- Índice para la clave foránea de asignatura (mejora rendimiento en JOINs)
CREATE INDEX idx_horario_asignatura ON Horario(codigo_asignatura);

-- Índice para la clave foránea de aula (mejora rendimiento en JOINs)
CREATE INDEX idx_horario_aula ON Horario(codigo_aula);

-- Índice compuesto para búsquedas por periodo y día (consultas comunes)
CREATE INDEX idx_horario_periodo_dia ON Horario(periodo, dia_semana);
```

```sql
-- Índice para búsquedas por rango de horas
CREATE INDEX idx_horario_horas ON Horario(hora_inicio, hora_fin);

-- Índice para consultas por asignatura y periodo
CREATE INDEX idx_horario_asig_periodo ON Horario(codigo_asignatura, periodo);
```

```
=ndice creado.

SQL>
SQL> -- Índice para la clave foránea de aula (mejora rendimiento
 en JOINs)
SQL> CREATE INDEX idx_horario_aula ON Horario(codigo_aula);

=ndice creado.

SQL>
SQL> -- Índice compuesto para búsquedas por periodo y día (consu
ltas comunes)
SQL> CREATE INDEX idx_horario_periodo_dia ON Horario(periodo, di
a_semana);

=ndice creado.

SQL>
SQL> -- Índice para búsquedas por rango de horas
SQL> CREATE INDEX idx_horario_horas ON Horario(hora_inicio, hora
_fin);

=ndice creado.

SQL>
SQL> -- Índice para consultas por asignatura y periodo
SQL> CREATE INDEX idx_horario_asig_periodo ON Horario(codigo_asi
gnatura, periodo);

=ndice creado.
```

Análisis de resultados:

Conclusiones y recomendaciones:

Dentro del laboratorio se cumplió exitosamente todos los objetivos, haciendo una aplicación práctica del lenguaje DDL en la base de datos Oracle, implementando un sistema de gestión académica completo, configurando un esquema de base de datos relacional normalizado, incluyendo todas las entidades del ámbito académico, aplicando correctamente las formas normales y estableciendo sus respectivas restricciones de integridad robustas.

Bibliografía:

[1] "Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database," Guru99. [En línea]. Disponible en: https://www.guru99.com/database-normalization.html. Consultado el: 30 de oct. de 2025.

[2] Oracle Corporation, "Oracle Database SQL Language Reference 19c," Oracle Documentation, 2023. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/