Solución de Laboratorio N° 6 - SQL DML en Oracle Database

Realizar las siguientes operaciones:

1. Los siguientes atributos son obligatorios:
   • NOMBRE (en todas las tablas),
   • APELLIDO1 en EMPLEADOS,
   • PRESUPUESTO en DEPARTAMENTOS,
   • SALARIO en HISTORIAL_SALARIAL y
   • SALARIO_MIN y SALARIO_MAX en TRABAJOS.
   Al crear las tablas correspondientes especificar la opción NOT NULL.
   Si la tabla ya estuviese creada:
   ALTER TABLE nombre_tabla ADD nombre_campo TIPO NOT NULL;

   Tabla DEPARTAMENTOS

```
CREATE TABLE departamentos (
  depto_id     NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  nombre       VARCHAR2(100) NOT NULL,
  presupuesto  NUMBER(12,2) NOT NULL,
  CONSTRAINT ux_departamentos_nombre UNIQUE (nombre)
);
```

   Tabla TRABAJOS

```
CREATE TABLE trabajos (
  trabajo_id  NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  nombre       VARCHAR2(100) NOT NULL,
  salario_min  NUMBER(12,2) NOT NULL,
  salario_max  NUMBER(12,2) NOT NULL,
  CONSTRAINT ux_trabajos_nombre UNIQUE (nombre),
  CONSTRAINT chk_salario_range CHECK (salario_min <= salario_max)
);
```

Tabla UNIVERSIDADES

```
CREATE TABLE universidades (
  universidad_id NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  nombre         VARCHAR2(200) NOT NULL
);
```

Tabla EMPLEADOS

```
CREATE TABLE empleados (
  dni        VARCHAR2(20) PRIMARY KEY,        -- usaremos DNI como PK
inicial
  nombre     VARCHAR2(100) NOT NULL,          -- obligatorio en todas las
tablas
  apellido1  VARCHAR2(100) NOT NULL,          -- obligatorio en
EMPLEADOS
  apellido2  VARCHAR2(100),
  sexo       CHAR(1) CHECK (sexo IN ('H','M')),
  fecha_nac  DATE,
  direc1     VARCHAR2(100),
  ciudad     VARCHAR2(100),
  cod_postal VARCHAR2(15)
);
```

Tabla UNIVERSIDADES ya creada, por tanto:

Tabla ESTUDIOS (vincula empleado - universidad)

```
CREATE TABLE estudios (
  estudio_id     NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  empleado_dni   VARCHAR2(20) NOT NULL,
  universidad_id NUMBER,
  nombre         VARCHAR2(200) NOT NULL,
  CONSTRAINT fk_estudios_empleado FOREIGN KEY (empleado_dni)
REFERENCES empleados(dni),
```

```
  CONSTRAINT fk_estudios_universidad FOREIGN KEY (universidad_id)
REFERENCES universidades(universidad_id)
);
```

Tabla HISTORIA_LABORAL

```
CREATE TABLE historia_laboral (
  hl_id          NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  empleado_dni   VARCHAR2(20) NOT NULL,
  trabajo_id     NUMBER,
  fecha_inicio   DATE NOT NULL,
  fecha_fin      DATE,
  depto_id       NUMBER,
  supervisor_dni VARCHAR2(20),
  CONSTRAINT fk_hl_empleado FOREIGN KEY (empleado_dni)
REFERENCES empleados(dni),
  CONSTRAINT fk_hl_trabajo FOREIGN KEY (trabajo_id) REFERENCES
trabajos(trabajo_id),
  CONSTRAINT fk_hl_depto FOREIGN KEY (depto_id) REFERENCES
departamentos(depto_id),
  CONSTRAINT fk_hl_supervisor FOREIGN KEY (supervisor_dni)
REFERENCES empleados(dni)
);
```

Tabla HISTORIAL_SALARIAL

```
CREATE TABLE historial_salarial (
  hs_id          NUMBER GENERATED BY DEFAULT AS IDENTITY
PRIMARY KEY,
  empleado_dni  VARCHAR2(20) NOT NULL,
  salario       NUMBER(12,2) NOT NULL,
  fecha_inicio  DATE NOT NULL,
  fecha_fin     DATE,
  CONSTRAINT fk_hs_empleado FOREIGN KEY (empleado_dni)
REFERENCES empleados(dni)
```

);

```
Table DEPARTAMENTOS creado.

Table TRABAJOS creado.

Table UNIVERSIDADES creado.

Table EMPLEADOS creado.

Table ESTUDIOS creado.

Table HISTORIA_LABORAL creado.

Table HISTORIAL_SALARIAL creado.
```

2. El atributo SEXO en EMPLEADOS sólo puede tomar los valores H para hombre y M para mujer.

   Está incluido en la tabla empleados con:

   sexo CHAR(1) CHECK (sexo IN ('H','M')),

```
-- Tabla EMPLEADOS
CREATE TABLE empleados (
  dni        VARCHAR2(20) PRIMARY KEY,
  nombre     VARCHAR2(100) NOT NULL,
  apellido1  VARCHAR2(100) NOT NULL,
  apellido2  VARCHAR2(100),
  sexo       CHAR(1) CHECK (sexo IN ('H','M')),
  fecha_nac  DATE,
  direcl     VARCHAR2(100),
  ciudad     VARCHAR2(100),
  cod_postal VARCHAR2(15)
);
```

3. Dos DEPARTAMENTOS no se llaman igual. Dos TRABAJOS tampoco.

   Para esto se usó UNIQUE:

   - CONSTRAINT ux_departamentos_nombre UNIQUE (nombre)

```
-- Tabla DEPARTAMENTOS
CREATE TABLE departamentos (
  depto_id     NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre       VARCHAR2(100) NOT NULL,
  presupuesto  NUMBER(12,2) NOT NULL,
  CONSTRAINT ux_departamentos_nombre UNIQUE (nombre)
);
```

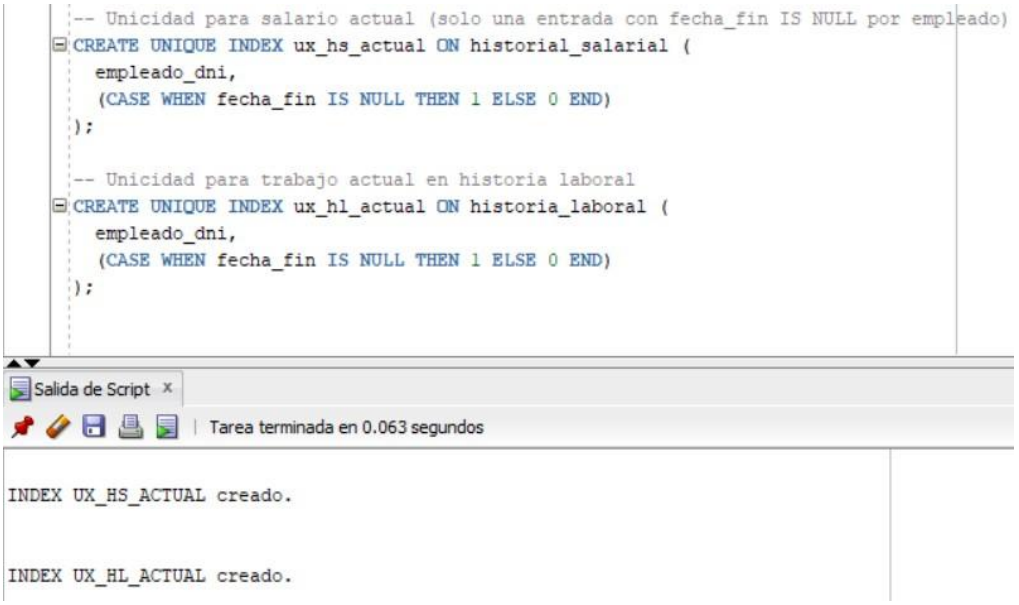
   - CONSTRAINT ux_trabajos_nombre UNIQUE (nombre),

```
-- Tabla TRABAJOS
CREATE TABLE trabajos (
  trabajo_id   NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  nombre       VARCHAR2(100) NOT NULL,
  salario_min  NUMBER(12,2) NOT NULL,
  salario_max  NUMBER(12,2) NOT NULL,
  CONSTRAINT ux_trabajos_nombre UNIQUE (nombre),
  CONSTRAINT chk_salario_range CHECK (salario_min <= salario_max)
);
```

4. Cada empleado tiene un solo salario en cada momento. También, cada empleado tendrá asignado un solo trabajo en cada momento.

```
CREATE UNIQUE INDEX ux_hs_actual ON historial_salarial (
  empleado_dni,
  (CASE WHEN fecha_fin IS NULL THEN 1 ELSE 0 END)
);

CREATE UNIQUE INDEX ux_hl_actual ON historia_laboral (
  empleado_dni,
  (CASE WHEN fecha_fin IS NULL THEN 1 ELSE 0 END)
);
```



Estos índices garantizan que por empleado: solo una fila con fecha_fin IS NULL (es decir, un único salario/trabajo activo).

5. Se ha de mantener la regla de integridad de referencia y pensar una clave primaria para cada tabla.

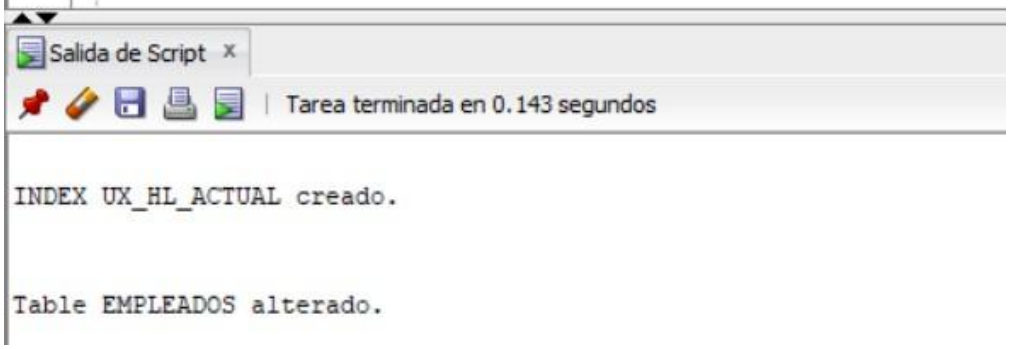
   Ya las definimos:

   - PK’s nombradas implícitamente con PRIMARY KEY en dni, depto_id, trabajo_id, hs_id, etc.
   - FK’s definidas en DDL anterior (fk_*).

6. Agregue a la tabla empleados los campos de teléfono y celular para tener como ubicar rápidamente al empleado.

   ```
   ALTER TABLE empleados ADD (
    telefono VARCHAR2(20),
    celular  VARCHAR2(20)
   );
   ```

```
ALTER TABLE empleados ADD (
  telefono VARCHAR2(20),
  celular  VARCHAR2(20)
);
```



7. Inserte las siguientes filas (las columnas que no aparecen se considerarán nulas).

Empleados				
NOMBRE	**APELLIDO1**	**APELLIDO2**	**DNI**	**SEXO**
Sergio	Palma	Entrena	111222	H
Lucia	Ortega	Plus	222333	M

Historial_Laboral					
EMPLEADO_DNI	**TRAB_COD**	**FECHA_INICIO**	**FECHA_FIN**	**DPTO_COD**	**SUPERVISOR_DNI**
111222		16/06/96		222333	

- Insertar empleados primero (porque historia_laboral tiene FK hacia empleados).

Hoja de Trabajo | Generador de Consultas

```
INSERT INTO empleados (dni, nombre, apellido1, apellido2, sexo)
VALUES ('111222', 'Sergio', 'Palma', 'Entrena', 'H');

INSERT INTO empleados (dni, nombre, apellido1, apellido2, sexo)
VALUES ('222333', 'Lucia', 'Ortega', 'Plus', 'M');

COMMIT;
```

- Insertamos un departamento y un trabajo.

Hoja de Trabajo | Generador de Consultas

```
INSERT INTO departamentos (nombre, presupuesto)
VALUES ('Ventas', 100000.00);

INSERT INTO trabajos (nombre, salario_min, salario_max)
VALUES ('Comercial', 800.00, 3000.00);

COMMIT;
```

- Insertar la fila solicitada en historia_laboral.

Hoja de Trabajo | Generador de Consultas

```
INSERT INTO historia_laboral (empleado_dni, trabajo_id, fecha_inicio, fecha_fin, depto_id, supervisor_dni)
VALUES (
    '111222',
    1,
    TO_DATE('16/06/1996', 'DD/MM/YYYY'),
    NULL,
    1,
    '222333'
);

COMMIT;
```

8. ¿Qué ocurre si se modifica esta última fila de historial_laboral asignándole al empleado 111222 un supervisor que no existe en la tabla de empleados?

Usemos como ejemplo:
INSERT INTO historia_laboral (..., supervisor_dni) VALUES (..., '999999');

```
Error que empieza en la línea: 2 del comando :
INSERT INTO historia_laboral (empleado_dni, trabajo_id, fecha_inicio, supervisor_dni)
VALUES ('111222', 1, SYSDATE, '999999')
Informe de error -
ORA-02291: restricción de integridad (SYS.FK_HL_SUPERVISOR) violada - clave principal no encontrada

https://docs.oracle.com/error-help/db/ora-02291/

More Details :
https://docs.oracle.com/error-help/db/ora-02291/
```

Conclusión: No puedes asignar un supervisor_dni que no exista en empleados.

9. Borre una universidad de la tabla de UNIVERSIDADES ¿Qué le sucede a la restricción de clave ajena de la tabla ESTUDIOS? Altere la definición de la tabla para que se mantenga la restricción, aunque se borre una universidad.

   Si intento borrar una fila de universidades que tiene estudios referenciando su universidad_id, Oracle lanza:

ORA-02292: integrity constraint (EMPRESA.FK_ESTUDIOS_UNIVERSIDAD) violated - child record found

Es decir, no deja borrar el padre si existen hijos.

Entonces, modificamos la FK para que, al borrar la universidad, la FK en estudios se ponga a NULL (ON DELETE SET NULL):

1. Primero drop la constraint que creamos:

```
ALTER TABLE estudios DROP CONSTRAINT fk_estudios_universidad;
```

2. Re-crear la constraint con ON DELETE SET NULL:

```
ALTER TABLE estudios
  ADD CONSTRAINT fk_estudios_universidad
  FOREIGN KEY (universidad_id)
  REFERENCES universidades(universidad_id)
  ON DELETE SET NULL;
```

10. Añada una restricción que obligue a que las personas que hayan introducido la CIUDAD deban tener el campo COD_POSTAL a NOT NULL. ¿Qué ocurre con las filas ya introducidas?

```
ALTER TABLE empleados ADD CONSTRAINT chk_ciudad_codpostal
    CHECK (ciudad IS NULL OR cod_postal IS NOT NULL);
```

Si ya existen filas que violan la condición (tienen ciudad no nula y cod_postal nulo), Oracle rechazará la ALTER TABLE con ORA-02290 (check constraint violated) para las filas existentes.

11. Añada un nuevo atributo VALORACIÓN en la tabla de EMPLEADOS que indique de 1 a 10 la valoración que obtuvo el empleado en su entrevista de trabajo al iniciar su andadura en la empresa. Ponga el valor por defecto 5 para ese campo.

ALTER TABLE empleados ADD (
  valoracion NUMBER(2) DEFAULT 5

);

ALTER TABLE empleados
ADD CONSTRAINT chk_valoracion CHECK (valoracion BETWEEN 1 AND 10);

```
ALTER TABLE empleados ADD CONSTRAINT chk_valoracion
    CHECK (valoracion BETWEEN 1 AND 10);
```

```
ALTER TABLE empleados ADD (
    valoracion NUMBER(2) DEFAULT 5
);
```

El DEFAULT 5 hace que nuevas filas sin valor explicitado tomen 5.

12. Elimine la restricción de que el atributo NOMBRE de la tabla EMPLEADOS no puede ser nulo.

ALTER TABLE empleados MODIFY (nombre NULL);

```
ALTER TABLE empleados MODIFY (nombre NULL);
```

13. Modificar el tipo de datos de DIREC1 de la tabla EMPLEADOS a cadena de caracteres de 40 como máximo.

ALTER TABLE empleados MODIFY (direc1 VARCHAR2(40));

```
ALTER TABLE empleados MODIFY (direc1 VARCHAR2(40));
```

14. ¿Podría modificar el tipo de datos del atributo FECHA_NAC de la tabla EMPLEADOS Y convertirla a tipo cadena?

Sí, pero no puedes simplemente hacer un ALTER TABLE empleados MODIFY (fecha_nac VARCHAR2(20)) si existen datos.

15. Cambiar la clave primaria de EMPLEADOS al NOMBRE y los dos APELLIDOS.

    1. Asegurar NOT NULL:
       ALTER TABLE empleados MODIFY (nombre NOT NULL);
       ALTER TABLE empleados MODIFY (apellido1 NOT NULL);
       ALTER TABLE empleados MODIFY (apellido2 NOT NULL);

2. Comprobar duplicados:
   SELECT nombre, apellido1, apellido2, COUNT(*) cnt
   FROM empleados
   GROUP BY nombre, apellido1, apellido2
   HAVING COUNT(*) > 1;
3. Quitar la PK actual:
   SELECT constraint_name FROM user_constraints WHERE
   table_name='EMPLEADOS' AND constraint_type='P';
   ALTER TABLE empleados DROP CONSTRAINT EMP_PK;
4. Añadir la nueva PK compuesta:
   ALTER TABLE empleados
    ADD CONSTRAINT pk_empleados_nombre_apellidos PRIMARY
   KEY (nombre, apellido1, apellido2);

16. Crear una nueva tabla llamada INFORMACIÓN UNIVERSITARIA que tenga el NOMBRE y los dos APELLIDOS (en un solo atributo) de todos los EMPLEADOS junto con la UNIVERSIDAD donde estudiaron. Cárguela con los datos correspondientes.

```
-- Crear tabla información universitaria
CREATE TABLE info_universitaria (
 id_info NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY
KEY,
 nombre_apellidos VARCHAR2(300) NOT NULL,
 universidad_nombre VARCHAR2(200)
);

-- Insertar datos (suponiendo que 'estudios' y 'universidades' están llenos)
INSERT INTO info_universitaria (nombre_apellidos, universidad_nombre)
SELECT
 e.nombre || ' ' || e.apellido1 || ' ' || NVL(e.apellido2,'') AS nombre_apellidos,
 u.nombre AS universidad_nombre
FROM estudios s
JOIN empleados e ON s.empleado_dni = e.dni
```

```
LEFT JOIN universidades u ON s.universidad_id = u.universidad_id;

COMMIT;
```

17. Crear una vista llamada NOMBRE_EMPLEADOS con el NOMBRE y los dos APELLIDOS (en un solo atributo) de todos los EMPLEADOS que son de Málaga.

```
CREATE OR REPLACE VIEW nombre_empleados AS
SELECT nombre || ' ' || apellido1 || ' ' || NVL(apellido2,'') AS
nombre_completo
FROM empleados
WHERE ciudad = 'Málaga';
```

18. Crear otra vista llamada INFORMACION_EMPLEADOS con el NOMBRE y los dos APELLIDOS (en un solo atributo) y EDAD (no fecha de nacimiento) de todos los EMPLEADOS.

```
CREATE OR REPLACE VIEW informacion_empleados AS
SELECT
  dni,
  nombre || ' ' || apellido1 || ' ' || NVL(apellido2,'') AS nombre_completo,
TRUNC(MONTHS_BETWEEN(SYSDATE, fecha_nac) / 12) AS edad
FROM empleados;
```

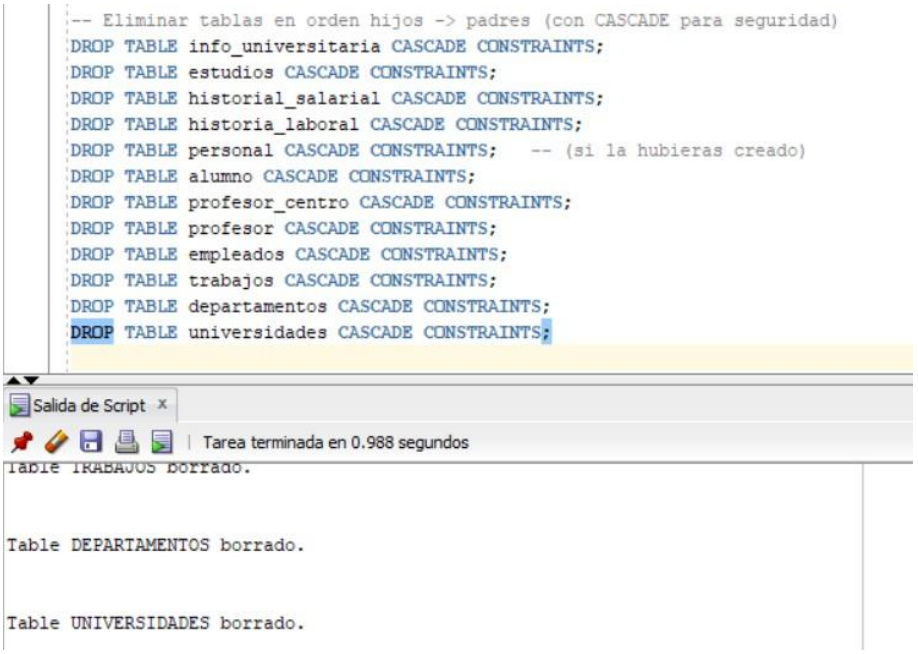
19. Crear otra vista sobre la anterior llamada INFORMACION_ACTUAL que dispone de toda la información de INFORMACION_EMPLEADOS junto con el SALARIO que está cobrando en este momento.

```
CREATE OR REPLACE VIEW informacion_actual AS
SELECT ie.dni,
      ie.nombre_completo,
      ie.edad,
      hs.salario AS salario_actual
```

```
FROM informacion_empleados ie
LEFT JOIN (
  SELECT empleado_dni, salario
  FROM historial_salarial
  WHERE fecha_fin IS NULL
) hs ON ie.dni = hs.empleado_dni;
```

20. Borrar todas las tablas. ¿Hay que tener en cuenta las claves ajenas a la hora de borrar las tablas?

- Sí. Debes borrar en orden inverso de dependencia (hijos antes que padres) o usar CASCADE CONSTRAINTS.



- Verificación final: