UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Práctica #3: Creación de Tablas Complejas y Consultas Avanzadas

PRESENTAN

Paredes Zamudio Luis Daniel
Rivera Morales David
Tapia Hernandez Carlos Alberto

PROFESOR

Erick Daniel Arroyo Martinez

ASIGNATURA

Fundamentos de Bases de Datos

27 de Agosto de 2024

Nota General: En los bloques de código insertado no se escriben palabras con acentos con la letra *ñ* ya que el paquete que se usa para la inserción del mismo no detecta los caracteres (creemos derivado de que son pocos los lenguajes de programación que existen en español y por lo mismo no hay soporte para ellos).

1. Implementación del Esquema

Se crea el esquema dado el diagrama relacional indicado, creando las tablas `Departments, Employees, Projects, Project Assignments, Salaries`

```
01 |      CREATE TABLE Departments (
02 |      department_id NUMBER PRIMARY KEY,
03 |      department_name VARCHAR2(50) NOT NULL,
04 |      budget NUMBER,
05 |      CONSTRAINT departments_budget_not_zero CHECK (budget > 0)
06 |  );
07 |
08 |  CREATE TABLE Employees (
09 |      -- Al ser un ID podemos ocupar el comando 'GENERATED BY DEFAULT AS IDENTITY'
10 |      -- para actuar como un AUTOINCREMENT, ya que las BD de Oracle, para hacer
11 |      -- la funcion de AUTOINCREMENT necesita un trigger o ocupar ese comando.
12 |      employee_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
13 |    fst_name VARCHAR2(50) NOT NULL,
14 |    lst_name VARCHAR2(50) NOT NULL,
15 |      email VARCHAR2(100) UNIQUE NOT NULL,
16 |      phone_number VARCHAR2(15),
17 |      hire_date DATE NOT NULL,
18 |      job_title VARCHAR2(50),
19 |      department_id NUMBER,
20 |      salary NUMBER,
21 |      CONSTRAINT employee_salary_not_zero CHECK (salary > 0),
22 |      FOREIGN KEY (department_id) REFERENCES departments(department_id)
23 |  );
24 |
25 |  CREATE TABLE Projects (
26 |      project_id NUMBER PRIMARY KEY,
27 |      project_name VARCHAR2(100) NOT NULL,
28 |      start_date DATE,
29 |      end_date DATE,
30 |      budget NUMBER,
31 |      CONSTRAINT project_budget_not_zero CHECK (budget >= 0)
32 |  );
33 |
34 |  CREATE TABLE Salaries (
35 |      salary_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
36 |      employee_id NUMBER,
37 |      salary_amount NUMBER CHECK (salary_amount > 0),
38 |      salary_date DATE NOT NULL,
39 |      FOREIGN KEY (employee_id) REFERENCES employees(employee_id)
40 |  );
41 |
42 |  CREATE TABLE Project_Assignments (
43 |      assignment_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
44 |      employee_id NUMBER,
45 |      project_id NUMBER,
46 |      -- el nombre 'roole' parece ser aproposito.
47 |      roole VARCHAR2(50),
48 |      hours_allocated NUMBER,
49 |      CONSTRAINT project_assigments_hours_allocated_not_zero CHECK (hours_allocated >=
       0),
50 |      FOREIGN KEY (employee_id) REFERENCES employees(employee_id),
51 |      FOREIGN KEY (project_id) REFERENCES projects(project_id)
52 |  );
```

2. Inserción de Datos

Se ingresan a la tabla los valores siguientes (provistos del archivo original `inserts.sql`).

```
01 |      -- Insertar departamentos
02 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (1, 'Human
       Resources', 50000);
03 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (2, 'Finance
       ', 75000);
04 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (3, 'IT
       Support', 100000);
05 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (4, '
       Marketing', 60000);
06 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (5, 'Sales',
        90000);
07 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (6, '
       Customer Service', 45000);
08 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (7, 'Product
        Development', 120000);
09 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (8, 'Legal',
        70000);
10 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (9, '
       Operations', 80000);
11 |  INSERT INTO Departments (department_id, department_name, budget) VALUES (10, '
       Research & Development', 95000);
12 |
13 |  -- Insertar empleados
14 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (1, 'Alice', 'Johnson', '
       alice.johnson@example.com', '123-456-7890', TO_DATE('2022-01-15', 'YYYY-MM-DD'),
        'Software Engineer', 3, 7000);
15 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (2, 'Bob', 'Smith', 'bob.
       smith@example.com', '234-567-8901', TO_DATE('2022-03-22', 'YYYY-MM-DD'), '
       Project Manager', 1, 8000);
16 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (3, 'Charlie', 'Brown', '
       charlie.brown@example.com', '345-678-9012', TO_DATE('2022-05-10', 'YYYY-MM-DD'),
        'UX Designer', 4, 6500);
17 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (4, 'Diana', 'Clark', 'diana
       .clark@example.com', '456-789-0123', TO_DATE('2022-07-18', 'YYYY-MM-DD'), 'Data
       Analyst', 6, 7200);
18 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (5, 'Edward', 'Miller', '
       edward.miller@example.com', '567-890-1234', TO_DATE('2022-09-25', 'YYYY-MM-DD'),
        'Marketing Specialist', 4, 6000);
19 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (6, 'Fiona', 'Davis', 'fiona
       .davis@example.com', '678-901-2345', TO_DATE('2022-11-30', 'YYYY-MM-DD'), '
       Customer Support', 6, 5500);
20 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (7, 'George', 'Martinez', '
       george.martinez@example.com', '789-012-3456', TO_DATE('2023-01-15', 'YYYY-MM-DD'
       ), 'Financial Analyst', 2, 7300);
21 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (8, 'Hannah', 'Wilson', '
       hannah.wilson@example.com', '890-123-4567', TO_DATE('2023-03-20', 'YYYY-MM-DD'),
        'HR Coordinator', 1, 6700);
22 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (9, 'Ian', 'Lee', 'ian.
       lee@example.com', '901-234-5678', TO_DATE('2023-05-25', 'YYYY-MM-DD'), 'Web
       Developer', 3, 7100);
23 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
       hire_date, job_title, department_id, salary) VALUES (10, 'Julia', 'Taylor', '
       julia.taylor@example.com', '012-345-6789', TO_DATE('2023-07-30', 'YYYY-MM-DD'),
       'Operations Manager', 9, 7800);
24 |
```

```
25 | -- Insertar proyectos
26 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (1, 'Website Redesign', TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('
     2024-06-30', 'YYYY-MM-DD'), 200000);
27 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (2, 'New CRM System', TO_DATE('2024-02-01', 'YYYY-MM-DD'), TO_DATE('2024-08-31'
     , 'YYYY-MM-DD'), 150000);
28 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (3, 'Mobile App Development', TO_DATE('2024-03-01', 'YYYY-MM-DD'), TO_DATE('
     2024-09-30', 'YYYY-MM-DD'), 175000);
29 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (4, 'Data Warehouse Implementation', TO_DATE('2024-04-01', 'YYYY-MM-DD'),
     TO_DATE('2024-12-31', 'YYYY-MM-DD'), 300000);
30 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (5, 'Marketing Campaign', TO_DATE('2024-05-01', 'YYYY-MM-DD'), TO_DATE('
     2024-10-31', 'YYYY-MM-DD'), 120000);
31 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (6, 'Product Launch', TO_DATE('2024-06-01', 'YYYY-MM-DD'), TO_DATE('2024-11-30'
     , 'YYYY-MM-DD'), 250000);
32 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (7, 'Employee Training Program', TO_DATE('2024-07-01', 'YYYY-MM-DD'), TO_DATE('
     2024-12-31', 'YYYY-MM-DD'), 80000);
33 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (8, 'Infrastructure Upgrade', TO_DATE('2024-08-01', 'YYYY-MM-DD'), TO_DATE('
     2024-11-30', 'YYYY-MM-DD'), 130000);
34 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (9, 'Customer Feedback System', TO_DATE('2024-09-01', 'YYYY-MM-DD'), TO_DATE('
     2025-03-31', 'YYYY-MM-DD'), 160000);
35 | INSERT INTO Projects (project_id, project_name, start_date, end_date, budget) VALUES
       (10, 'Regulatory Compliance', TO_DATE('2024-10-01', 'YYYY-MM-DD'), TO_DATE('
     2025-06-30', 'YYYY-MM-DD'), 190000);
36 |
37 | -- Insertar salarios
38 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (1,
       1, 5000, TO_DATE('2024-01-15', 'YYYY-MM-DD'));
39 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (2,
       2, 6000, TO_DATE('2024-01-15', 'YYYY-MM-DD'));
40 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (3,
       3, 5500, TO_DATE('2024-01-15', 'YYYY-MM-DD'));
41 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (4,
       4, 7000, TO_DATE('2024-01-15', 'YYYY-MM-DD'));
42 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (5,
       5, 6500, TO_DATE('2024-01-15', 'YYYY-MM-DD'));
43 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (6,
       1, 5200, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
44 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (7,
       2, 6100, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
45 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (8,
       3, 5600, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
46 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES (9,
       4, 7100, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
47 | INSERT INTO Salaries (salary_id, employee_id, salary_amount, salary_date) VALUES
     (10, 5, 6600, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
48 |
49 | -- Insertar asignaciones de proyecto
50 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (1, 1, 1, 'Lead Developer', 120);
51 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (2, 2, 1, 'Project Manager', 100);
52 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (3, 3, 2, 'UX Designer', 110);
53 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (4, 4, 3, 'Data Analyst', 130);
54 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (5, 5, 4, 'Marketing Specialist', 90);
55 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
     hours_allocated) VALUES (6, 6, 5, 'Customer Support', 80);
56 | INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
```

```
        hours_allocated) VALUES (7, 7, 6, 'Financial Analyst', 150);
57 |  INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
        hours_allocated) VALUES (8, 8, 7, 'HR Coordinator', 70);
58 |  INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
        hours_allocated) VALUES (9, 9, 8, 'Web Developer', 125);
59 |  INSERT INTO Project_Assignments (assignment_id, employee_id, project_id, roole,
        hours_allocated) VALUES (10, 10, 9, 'Operations Manager', 140);
```

3. Consultas Avanzadas

1. Selecciona los nombres y apellidos de todos los empleados que trabajan en un departamento con un presupuesto mayor a 1,000,000. **Hint:** Usa una subconsulta para filtrar los departamentos con un presupuesto superior a un millón.

```
01 |  -- Es normal que no haya resultados, pero es una query valida.
02 |  SELECT fst_name, lst_name FROM Employees WHERE department_id IN (SELECT
        department_id FROM Departments WHERE budget > 1000000);
```

2. Obtén los nombres y apellidos de los empleados que tienen un salario mayor al promedio de todos los salarios. Ordena los resultados por el salario en orden descendente. **Hint:** Utiliza la función de agregación que calcula el promedio para comparar los salarios individuales con el promedio.

```
01 |  -- Se puede hacer una consulta secundaria para obtener el promedio, redondeado
        a un entero, y luego sobre
02 |  -- ese valor aplicar el WHERE de la consulta.
03 |  SELECT fst_name, lst_name FROM Employees WHERE salary > (SELECT ROUND(AVG(
        salary)) from Employees);
```

3. Encuentra todos los proyectos que tienen un presupuesto entre 500,000 y 1,000,000. Muestra el nombre del proyecto y su presupuesto. **Hint:** Usa la cláusula para filtrar los presupuestos dentro del rango especificado.

```
01 |  -- Como el 1. , no regresa un valor ya que no existen proyectos con esas
        condiciones, pero la query
02 |  -- sigue siendo valida.
03 |  SELECT project_name, budget from Projects WHERE budget BETWEEN 500000 AND
        1000000
```

4. Inserta un nuevo empleado en la tabla `Employees` con un salario y asigna este empleado a un departamento especıífico. Hint: Asegúrate de asignar correctamente el ID del departamento en la nueva fila que insertes.

```
01 |  INSERT INTO Employees (employee_id, fst_name, lst_name, email, phone_number,
        hire_date, job_title, department_id, salary)
02 |  VALUES (11, 'Daniel', 'Paredes', 'danielparedes@ciencias.unam.mx', '
        556-937-7986', TO_DATE('2024-08-22', 'YYYY-MM-DD'), 'IT Manager', 3, 8500);
```

5. Actualiza el salario de todos los empleados cuyo tıítulo de trabajo es '`Manager`' incrementándolo en un 10 %. **Hint:** Usa una operación matemática para incrementar los salarios en función de su valor actual.

```
01 |  -- El calculo del aumento de salario se hace usando sumas y multiplicaciones y
        luego se hace el SET y el WHERE
02 |  -- para actualizar solo los que sean Managers.
03 |  UPDATE Employees SET salary = (salary + (salary*.10)) WHERE job_title LIKE '%
        Manager';
```

6. Elimina todos los registros de empleados que no tienen ningún proyecto asignado. **Hint:** Utiliza una subconsulta o un operador de comparación para identificar empleados sin asignaciones de proyectos.

```
-- Se seleccionan todos los employee_id de Employees y luego los de Project_Assignments. Los que no
-- aparezcan en Project_Assignments pero si en Employees se eliminan de Employees.
DELETE FROM Employees WHERE employee_id NOT IN (SELECT employee_id FROM Project_Assignments);
```

7. Cuenta cuántos empleados trabajan en cada departamento y muestra el nombre del departamento junto con el número de empleados. Filtra los resultados para mostrar solo los departamentos con más de 10 empleados. **Hint:** Usa la función de agregación que cuenta filas para agrupar los empleados por departamento.

```
-- Investigamos como poner "alias" para las tablas sin usar el comando AS. Esto se hace despues de
-- declarar el FROM se pueden empezar a ocupar desde antes. Por ejemplo, hacemos d.departament_name
-- y luego del comando hacemos FROM Departments d.

SELECT d.department_name, employee_count FROM Departments d,
     (SELECT department_id, COUNT(*) as employee_count FROM Employees
      GROUP BY department_id HAVING COUNT(*) > 10) e
WHERE d.department_id = e.department_id;
```

8. Encuentra el salario mínimo y máximo que reciben los empleados en el departamento de 'IT'. Muestra el tíitulo del trabajo, el salario mínimo y el salario máximo. **Hint:** Usa funciones de agregación para calcular los valores mínimo y máximo.

```
-- Se pueden "anidar" las funciones de agregacion y luego darles un alias. De esta forma, se muestran
-- ambos resultados simultaneamente y con sus alias correspondientes.
SELECT job_title, MIN(salary) AS ITSalaryMin, MAX(salary) AS ITSalaryMax FROM Employees WHERE department_ID = 3 GROUP BY job_title;
```

9. Encuentra los nombres de los proyectos que comenzaron en el año 2023 y que están en el departamento con el presupuesto más alto. **Hint:** Usa una subconsulta para obtener el presupuesto más alto y filtra los proyectos por la fecha de inicio.

```
-- Ocupamos la misma idea del 7 para los alias. Lo que se hace es seleccionar el ano y el nombre de proyecto
-- de los 'project_id' que aparezcan en la subconsulta de buscar el departamento con el presupuesto mas
-- alto. Luego, ocupamos el (creemos es un operador) 'date' para transformar la fecha '2023-01-01' a un tipo
-- reconocido por la BD de Oracle y asi hacer el filtro del ano a buscar.
SELECT DISTINCT p.project_name, p.start_date FROM Projects p WHERE p.project_id IN (
            SELECT department_id FROM Departments WHERE budget = (SELECT MAX(budget) FROM Departments)
) AND p.start_date >= date '2023-01-01';
```

10. Selecciona los nombres y apellidos de los empleados que tienen un correo electrónico que contiene el dominio `'example.com'`. Asegúrate de que el correo electrónico sea único. **Hint**: Filtra los resultados usando un patrón que coincida con el dominio `'example.com'` en la dirección de correo electrónico.

```sql
-- Basta con ocupar 'DISTINCT' y 'LIKE' para poder hacer el filtro en la tabla Employees.
SELECT DISTINCT fst_name, lst_name FROM Employees WHERE email LIKE '%@example.com';
```

11. Muestra los nombres de los empleados que están trabajando en más de un proyecto. Ordena los resultados alfabéticamente. Hint: Agrupa los resultados por empleado y filtra aquellos que tienen más de una asignación de proyecto.

```sql
-- Hacemos una consulta secundaria para agrupar los proyectos que tengan mas de un employee_id
-- asignado. Luego ordenamos por nombre los ID que aparezcan en el filtro.
SELECT fst_name, lst_name FROM Employees WHERE employee_id IN (
    SELECT employee_id FROM Project_Assignments GROUP BY employee_id HAVING COUNT(*) > 1)
ORDER BY fst_name, lst_name;
```

12. Selecciona el nombre del proyecto y el total de horas asignadas a cada proyecto. Filtra los resultados para mostrar solo los proyectos con mas de 1000 horas asignadas. **Hint:** Usa una función de agregación para sumar las horas asignadas y filtra los proyectos que cumplan con el criterio.

```sql
-- Lo que se hace es hacer una consulta secundaria para obtener el total de horas asignadas de un proyecto de
-- la tabla Project_Assignments y luego se filtra para solo considerar aquellos proyectos que tengan mas de
-- 1000 horas asignadas. Finalmente lo que se hace es devolver los nombres y las horas asignadas de aquellos ID
-- que coincidan con los que aparecen en la consulta secundaria.
SELECT p.project_name, total_hours FROM Projects p,
     (SELECT project_id, SUM(hours_allocated) as total_hours FROM Project_Assignments
      GROUP BY project_id HAVING SUM(hours_allocated) > 1000) pa
WHERE p.project_id = pa.project_id;
```

4. Justificaciones

4.1. Departments y Employees

La relación entre `Departments` y `Employees` es esencial para la estructura organizativa. La clave foránea `department_id` en `Employees` permite asociar cada empleado a un departamento específico, facilitando la gestión de recursos humanos y la organización del personal.

Justificación Técnica: Esta relación mantiene la integridad referencial y posibilita consultas sobre la distribución del personal, como el conteo de empleados por departamento o la identificación de departamentos con mayor número de empleados.

4.2. Departments y Projects

La conexión entre `Departments` y `Projects` asigna la responsabilidad de los proyectos a departamentos específicos. Cada proyecto se vincula a un único departamento mediante la clave foránea `department_id` en la tabla `Projects`.

Justificación Técnica: Esta estructura permite una gestión eficiente de recursos, alineando los proyectos con las capacidades departamentales. Facilita consultas sobre proyectos por departamento y análisis de asignación presupuestaria.

4.3. Employees y Project Assignments

La relación muchos a muchos entre `Employees` y `Projects` se implementa mediante la tabla intermedia `Project_Assignments`. Esto permite asignar múltiples empleados a diversos proyectos, registrando las horas dedicadas.

Justificación Técnica: Esta estructura flexible facilita el seguimiento detallado del tiempo invertido en proyectos, la carga de trabajo de los empleados y la distribución de recursos humanos.

4.4. Projects y Project Assignments

La relación entre `Projects` y `Project_Assignments` permite un registro detallado de las horas trabajadas en cada proyecto.

Justificación Técnica: Las claves foráneas en `Project_Assignments` aseguran la integridad de los datos y permiten análisis complejos sobre la distribución del trabajo y la eficiencia en la utilización de recursos.

4.5. Employees y Salaries

La asociación directa entre `Employees` y sus salarios es parte fundamental para la definició del estatus de empleado y la gestión financiera de recursos humanos.

Justificación Técnica: Esta relación permite un control preciso de los costos laborales, facilitando análisis financieros como comparaciones salariales, cálculo de promedios y gestión de incrementos salariales.