Práctica 06

Sobre optimización de consultas aplicando indexación, desnormalización y buenas prácticas

PLANTILLA DE RESPUESTAS

PARTE 1: PROCESAMIENTO DE CONSULTAS

Ejercicio 1

```
SELECT *
FROM empleados e
WHERE e.apellidos like 'A%'
AND e.apellidos like 'B%'
AND e.apellidos like 'C%';
```

1) La consulta no tiene ningún error sintáctico

2) La consulta posee errores semánticos, esto debido al uso del operador AND ya que, si el listado no posee el primer valor, descartara los demás y no generara la lista

3) Consulta corregida:

```
SELECT *
FROM empleados e
WHERE e.apellidos like 'A%'
OR e.apellidos like 'B%'
OR e.apellidos like 'C%';
```

Ejercicio 2

```
SELECT *
FROM empleados
WHERE fecha_ing > '01-ENE-1998'
AND fecha_ing > '15-MAY-2001'
OR genero = 'M';
```

1) El enunciado solo generaría resultados en base a la condición (genero = ‘M’) ya que no podría encontrar una fecha mayor a '01-ENE-1998' que a su vez sea mayor que '15-MAY-2001'

Ejercicio 3

```
SELECT email, count(cedula)
FROM empleados
WHERE genero = 'M'
GROUP BY email
HAVING count(cedula) > 1;
```

1) La consulta no presentara ningún dato, ya que al agrupar en base al campo email, nos dará una lista con datos únicos, que luego con la cláusula Having evaluara en la lista los que contengan mas de 1 cedula, lo que sería imposible ya que cedula es un campo único por lo que no se cumpliría esa condición

Ejercicio 4

```
SELECT e.*
FROM empleados e, cargos c
WHERE e.idcargo = c.idcargo
 AND c.cargo = 'FISCALIZADOR';
```

1) $\pi cedula(empleados \bowtie idcargo \sigma cargo='FISCALIZADOR'(cargos))$

Ejercicio 5

Determine el costo en términos de operaciones I/O para planes de ejecución A, B y C que se indican más adelante. Para ello suponga las siguientes estadísticas:

- editoriales: 200
- libros: 10000
- libros del 2016 con más de 500 páginas: 250
- libros publicados en Argentina el 2016 con más de 500 páginas: 20

$$\pi_{l.titulo}\left(\sigma_{l.idedt = e.idedt \text{ and } l.anio = 2016 \text{ and } l.numpaginas > 500 \text{ and } e.pais = 'Argentina'}\left(\left(\rho_{l}\, libros\right) \times \left(\rho_{e}\, editoriales\right)\right)\right)$$

OPERACION	**LECTURAS**	**ESCRITURAS**
Unión Libros x Editoriales	10.000 + 200 = 10.200	10.000 * 200 = 2.000.000
Filtrado	2.000.000	20
Proyección	20	
TOTAL	2.010.220	2.000.020
LECTURAS Y ESCRITURAS	4.010.240	

Plan B:

$$\pi_{l.titulo}\left(\sigma_{e.pais = 'Argentina'}\left(\left(\sigma_{l.anio = 2016 \text{ and } l.numpaginas > 500}\left(\rho_{l}\, libros\right)\right) \bowtie_{l.idedt = e.idedt} \left(\rho_{e}\, editoriales\right)\right)\right)$$

OPERACION	**LECTURAS**	**ESCRITURAS**
Filtrado de Libros	10.000	250
Combinar (Libros x Editoriales)	250 + 200 = 450	250
Filtrado libros Argentina	250 + 20 = 270	20
Proyección	20	
TOTAL	10.740	520
LECTURAS Y ESCRITURAS	11.260	

Plan C:

$$\pi_{l.titulo}\left(\sigma_{l.anio = 2016 \text{ and } l.numpaginas > 500 \text{ and } e.pais = 'Argentina'}\left(\left(\rho_{l}\ libros\right) \bowtie_{l.idedt = e.idedt}\left(\rho_{e}\ editoriales\right)\right)\right)$$

Operación	**Lecturas**	**Escrituras**
Combinar (Libros x Editoriales)	10.000 + 200 = 10.200	10.000
Filtrado	10.000	20
Proyección	20	
TOTAL	20.220	10.020
LECTURAS Y ESCRITURAS	30 240	

PARTE 2: BUENAS PRÁCTICAS SQL

Ejercicio 6

Evalúe las siguientes expresiones condicionales, y diga cómo se podría mejorar su formulación:

a) NOT (NUM_PAG < 80 OR EDITORIAL = 'LIMUSA')

Aplicando las Leyes de Morgan, la cual nos dice que :

NOT(A OR B)≡(NOT A) AND (NOT B)

lo cual nos dan como resultado:

NOT (NUMPAG<80 OR EDITORIAL='LIMUSA')≡(NOT NUMPAG<80) AND (NOT EDITORIAL='LIMUSA')

La cual al simplificarla obtenemos que :

NUMPAG≥80 AND EDITORIAL ≠'LIMUSA' // resultado

b) NOT (SUELDO < 800) AND COD_POSTAL = '110401'

NOT (SUELDO<800) equivale a SUELDO ≥ 800

Por lo tanto:

SUELDO≥800 AND CODPOSTAL='110401' // resultado

PARTE 3: DESNORMALIZACIÓN

Ejercicio 7

Técnica de desnormalización

Se aplicaría la técnica de desnormalización de Combinación de tablas con asociación 1:1

Tablas desnormalizadas

ProyectoIntegrantes(Codigo, Id, Denominación, Año, Nombre, Rol, Empresa)

Consulta

```
SELECT p.denomicacion,
p.nombre,
p.rol
FROM proyectos_integrantes
```

Ejercicio 8

```
SELECT idMatricula, fecha, estudiante, periodoAcademico,
       costoCurso+tasaAdministrativa–descuento AS "ImporteTotal"
FROM matriculas;
```

Técnica de desnormalización

Se aplicaría la técnica de desnormalización Inclusión de atributos derivados

Tablas desnormalizadas

Matriculas(idMatricula, fecha, estudiante, periodoAcademico, costoCurso, tasaAdministrativa, descuento, ImporteTotal**)**

Consulta

```
SELECT idMatricula,
fecha, estudiante,
periodoAcademico,
ImporteTotal
FROM matriculas
```

Ejercicio 9

Dadas las siguientes tablas.

PROVINCIAS

CODPROV <PK>	NOMPROV	EXTENSION	REGION	PTELEF	CODCAP <FK>
01	Azuay	8628	Centro Sur	7	CUE
06	Chimborazo	6487	Centro	3	RIO
09	Guayas	17139	Litoral	4	GYE
11	Loja	11027	Sur	7	LOJ
12	Los Rios	6254	Litoral	5	BAB
13	Manabi	18400	Pacífico	5	POR
16	Pastaza	29520	Centro	6	PUY
17	Pichincha	9612	Centro Norte	2	UIO

CAPITALES

CODCAP <PK>	NOMCAP	POBLACION	ALTURA
BAB	Babahoyo	153776	8
CUE	Cuenca	505585	2550
GYE	Guayaquil	2526927	4
LOJ	Loja	206834	2200
POR	Portoviejo	280586	53
PUY	Puyo	33557	924
RIO	Riobamba	223586	2750
UIO	Quito	2239141	2800

Suponga que en el 90% de los casos en los que consultamos información de provincias incluimos también
los datos de su capital, por ejemplo:

```
SELECT p.nomprov "Provincia",
       p.region "Region",
       c.nomcap "Capital",
       c.altura "Altura Capital (msnm)"
FROM provincias p, capitales c
WHERE p.codcap = c.codcap;
```

Técnica de desnormalización

Se aplicaría la técnica de desnormalización Combinación de tablas con asociación 1:1

Tablas desnormalizadas

proyectos_integrantes(CODPROV, NOMPROV, EXTENSION, REGION, CODCAP, NOMCAP, POBLACION, ALTURA)

Consulta Final:

```
SELECT
codprov,
nomprov AS "Provincia",
region AS "Región",
nomcap AS "Capital",
altura AS "Altura Capital (msnm)"
FROM
provincias;
```

Ejercicio 10

Dadas las siguientes tablas.

PAISES

ID_PAIS (PK)	NOMBRE_PAIS	CAPITAL	ID_CONTINENTE (FK)
1	ECUADOR	QUITO	C1
2	FRANCIA	PARIS	C2
3	JAPON	TOKIO	C3
4	CHINA	PEKIN	C3
5	BRASIL	BRASILIA	C1
6	ITALIA	ROMA	C2

CONTINENTES

ID_CONTINENTE (PK)	NOMBRE_CONTINENTE
C1	AMERICA
C2	EUROPA
C3	ASIA
C4	AFRICA

Suponga que casi siempre requerimos consultar la información de países con el nombre del continente, así:

```
SELECT
    PAI.ID_PAIS,
    PAI.NOMBRE_PAIS,
    PAI.CAPITAL,
    CON.NOMBRE_CONTINENTE
FROM
    PAISES PAI INNER JOIN CONTINENTES CON
       ON PAI.ID_CONTINENTE = CON.ID_CONTINENTE;
```

Técnica de desnormalización

Se aplicaría la técnica de desnormalización Duplicidad de atributos que no forman parte de la clave en asociaciones 1:N

Tablas desnormalizadas

proyectos_integrantes(ID_PAIS, NOMBRE_PAIS, CAPITAL, ID_CONTINENTE, NOMBRE_CONTINENTE)

Consulta

```
SELECT
   ID_PAIS,
   NOMBRE_PAIS,
   CAPITAL,
   NOMBRE_CONTINENTE
FROM
   PAISES;
```

PARTE 4: PREGUNTAS REPASO

Pregunta 1

1. Discuta cuándo los índices pueden mejorar la eficiencia del sistema de base de datos.

Consultas de búsqueda:

- **Consultas con WHERE**: Cuando las consultas filtran datos usando cláusulas WHERE, los índices en las columnas mencionadas pueden acelerar la búsqueda. Por ejemplo,

  ```
  SELECT *

  FROM empleados

  WHERE cedula = '1234567890'
  ```

  será más rápido si cedula está indexada.

- **Consultas con rangos**: Para consultas que buscan un rango de valores, como

  ```
  SELECT *

  FROM productos

  WHERE precio BETWEEN 10 AND 20
  ```

  los índices en la columna precio pueden mejorar el rendimiento

Pregunta 2

2. Investigue 3 buenas prácticas SQL, adicionales a las que constan el material subido a Canvas. Descríbalas usando algún ejemplo.

1. **Utilizar alias claros**

El uso de aliases claros y consistentes en tus consultas SQL mejora la legibilidad y facilita el mantenimiento del código. Los aliases deben ser cortos pero descriptivos, evitando confusiones con otros nombres de columnas o tablas.

-- Mala práctica: aliases poco claros

```
SELECT a.cedula, b.cargo
FROM empleados a
JOIN cargos b ON a.idcargo = b.idcargo
WHERE b.cargo = 'FISCALIZADOR';
```

```
-- Buena práctica: aliases claros y consistentes
SELECT e.cedula, c.cargo
FROM empleados e
JOIN cargos c ON e.idcargo = c.idcargo WHERE c.cargo = 'FISCALIZADOR';
```

2. Limitar el uso de funciones en las condiciones WHERE

Evita usar funciones en las condiciones de WHERE, especialmente aquellas que requieren cálculos en cada fila de la tabla. Esto puede impedir que se utilicen índices eficientemente y ralentizar el rendimiento de la consulta.

```
-- Mala práctica: uso de función en la condición WHERE
SELECT *
FROM empleados
WHERE YEAR(fecha_ing) = 2023;

-- Buena práctica: cálculo de valores de comparación fuera de la función
SELECT *
FROM empleados
WHERE fecha_ing >= '2023-01-01' AND fecha_ing < '2024-01-01';
```

3. Utiliza Comentarios para Documentar Consultas Complejas

Cuando se escribe consultas SQL complejas que involucren múltiples tablas, uniones o lógica complicada, usar comentarios para explicar la intención y el propósito de la consulta facilitara la comprensión para otros desarrolladores que revisen o mantengan el código en el futuro.

```
-- Buena práctica: uso de comentarios para explicar la consulta

SELECT e.cedula, e.apellidos, e.nombres, c.cargo
FROM empleados e
JOIN cargos c ON e.idcargo = c.idcargo
WHERE c.cargo = 'FISCALIZADOR'

-- Esta consulta obtiene la información de los empleados que tienen el cargo de fiscalizador.
```

Pregunta 3

3. ¿Bajo qué circunstancias considera que sería conveniente aplicar desnormalización en una base de datos? Usar algún ejemplo.

Cuando se realizan consultas complejas que implican múltiples uniones y agregaciones, la desnormalización puede reducir el tiempo de respuesta al evitar operaciones costosas de JOIN y cálculos repetitivos.

Base de Datos de una Tienda:

```
CREATE TABLE productos (
   id_producto INT PRIMARY KEY,
   nombre VARCHAR(100),
   descripcion TEXT,
   precio DECIMAL(10, 2),
   categoria_id INT,
   FOREIGN KEY (categoria_id) REFERENCES categorias(id_categoria)
);

CREATE TABLE pedidos (
   id_pedido INT PRIMARY KEY,
   fecha_pedido DATE,
   cliente_id INT,
   FOREIGN KEY (cliente_id) REFERENCES clientes(id_cliente)
);

CREATE TABLE clientes (
   id_cliente INT PRIMARY KEY,
   nombre VARCHAR(50),
   apellidos VARCHAR(50),
   direccion VARCHAR(100),
   email VARCHAR(100) UNIQUE
);
```

Supongamos que necesitamos frecuentemente mostrar detalles del producto junto con información del pedido y del cliente al realizar consultas de historial de pedidos o para generar informes detallados.

Una consulta normal para obtener los detalles de un pedido podría ser así:

```
SELECT p.id_producto, p.nombre AS nombre_producto, p.precio,

        c.id_cliente, c.nombre AS nombre_cliente, c.apellidos,

        pe.id_pedido, pe.fecha_pedido

FROM productos p

JOIN pedidos pe ON p.id_producto = pe.id_producto

JOIN clientes c ON pe.cliente_id = c.id_cliente

WHERE pe.id_pedido = 12345;
```

Esta consulta implicaria un uso de recursos considerable, toamdno en cuenta que se le llegaria a usar multiples veces, y esto podria alentar nuestra base de datos

Solucion

Usando la tecnica de la desnormalizacion podriamos mejorar el tiempo de ejecucion de sta cosnulta optimizando asi recursos necesarios para nuestra base de datos

Ejemplo de desnormalizacion

```
CREATE VIEW vista_pedidos_con_detalle AS

SELECT pe.id_pedido, pe.fecha_pedido,
    p.id_producto, p.nombre AS nombre_producto, p.precio,
    c.id_cliente, c.nombre AS nombre_cliente, c.apellidos

FROM pedidos pe

JOIN productos p ON pe.id_producto = p.id_producto
JOIN clientes c ON pe.cliente_id = c.id_cliente;
```

Y gracias a esta desnormalizacion podriamos tener consultas faciles de implementar a nivel de recursos y tiempo, como es:

```
SELECT pe.id_pedido, pe.fecha_pedido

FROM vista_pedidos_con_detalle

WHERE id_pedido = 12345;
```