Tarea 1: Base De Datos

Implementar una base de datos en SQL en Oracle e implementar consultas, analizar optimización.

Nombres: Mario González Galdames – Luis Inostroza Flores

Rut: 20171019-7 – 20306020-3

Carrera: Ingeniería Civil Informática

Docente: Dra. Angélica Urrutia Sepúlveda

Introducción

Oracle es uno de los sistemas de bases de datos más completos, destacando principalmente entre sus características su soporte de transacciones, estabilidad escalabilidad y soporte Multiplataforma.

Podríamos definir a Oracle como una herramienta cliente/servidor para la gestión de Bases de Datos que se usa principalmente en grandes empresas, diseñado para que las organizaciones puedan controlar y gestionar grandes volúmenes de contenidos no estructurados en un único repositorio con el objetivo de reducir los costes y los riesgos asociados a la pérdida de información.

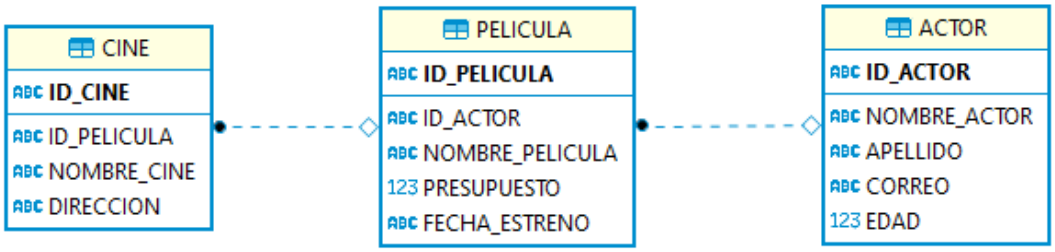
Es por estas y otras características que Oracle ocupa el primer lugar en la categoría de las bases de datos y el séptimo lugar a nivel mundial de las compañías de tecnologías de la información. La tecnología Oracle se encuentra prácticamente en muchas industrias del mundo y en las oficinas de 98 de las 100 empresas Fortune 100.

Es por esto que en este trabajo vamos a utilizar este motor de base de datos para analizar distintos grupos de secuencias de datos, realizando diferentes tipos de consultas y utilizando las diferentes teorías de manejos de datos vistas en clases, con el fin de realizar un estudio en los tiempos de respuestas de estas consultas y graficarlas para medir la eficiencia de cada una de estas.

El orden de muestra de información será: presentación del problema, código implementado, tiempos de muestra y gráficos con resultados

Creación de base de datos

El modelo de base de datos que se utilizó para hacer el análisis correspondiente está compuesto por 3 tablas, las cuales van a representar la información de las películas que brinda un cine en específico, se utilizó la aplicación DBeaver como editor de texto. El modelo relacional sería el siguiente:



Código de implementación en SQL:

```sql
CREATE TABLE ACTOR(
    ID_ACTOR VARCHAR2(20) NOT NULL,
    NOMBRE_ACTOR VARCHAR2(100),
    APELLIDO VARCHAR2(100),
    CORREO VARCHAR2(100),
    EDAD NUMBER
);

CREATE TABLE PELICULA(
    ID_PELICULA VARCHAR2(20) NOT NULL,
    ID_ACTOR VARCHAR2(20),
    NOMBRE_PELICULA VARCHAR2(100),
    PRESUPUESTO NUMBER,
    FECHA_ESTRENO VARCHAR2(50)
);

CREATE TABLE CINE(
    ID_CINE VARCHAR2(20) NOT NULL,
    ID_PELICULA VARCHAR2(20),
    NOMBRE_CINE VARCHAR2(100),
    DIRECCION VARCHAR2(200)
);

ALTER TABLE ACTOR ADD CONSTRAINT PK_ACTOR PRIMARY KEY (ID_ACTOR);
ALTER TABLE PELICULA ADD CONSTRAINT PK_PELICULA PRIMARY KEY (ID_PELICULA);
ALTER TABLE CINE ADD CONSTRAINT PK_CINE PRIMARY KEY (ID_CINE);

ALTER TABLE PELICULA ADD CONSTRAINT FK_PELICULA FOREIGN KEY (ID_ACTOR) REFERENCES ACTOR(ID_ACTOR);
ALTER TABLE CINE ADD CONSTRAINT FK_CINE FOREIGN KEY (ID_PELICULA) REFERENCES PELICULA(ID_PELICULA);

ALTER SYSTEM FLUSH BUFFER_CACHE;
ALTER SYSTEM FLUSH SHARED_POOL;
```

Consultas

A continuación, se mostrarán los códigos correspondientes a las consultas de cada sección propuesta por el docente.

a) Consultas Simples de 1 tabla:

```
                                                    --CONSULTAS SIMPLES--
--AND--
SELECT EDAD FROM ACTOR WHERE EDAD < 50 AND EDAD > 30;
--or--
SELECT EDAD FROM ACTOR WHERE EDAD >60 OR EDAD = 20;
--like--
SELECT NOMBRE_ACTOR FROM ACTOR WHERE NOMBRE_ACTOR LIKE 'D%';
--NOT LIKE--
SELECT APELLIDO FROM ACTOR WHERE APELLIDO NOT LIKE 'A%';
--LENGTH--
SELECT NOMBRE_ACTOR, APELLIDO, EDAD FROM ACTOR WHERE LENGTH(NOMBRE_ACTOR) = 6;
--IN--
SELECT * FROM ACTOR WHERE ID_ACTOR IN (1,20,32,45,70,87);
```

b) Consultas de Más de una tabla (join):

```
                                                --CONSULTAS DOBLES--
--MUESTRE TODOS LOS ACTORES QUE GRABARON UNA PELICULA CUYO PRESUPUESTO ERA DE 30.000.000 Y 50.000.000
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, P.PRESUPUESTO
FROM ACTOR a
JOIN PELICULA p
ON p.ID_ACTOR = a.ID_ACTOR
WHERE p.PRESUPUESTO BETWEEN 30000000 AND 50000000
GROUP BY A.ID_ACTOR ,A.NOMBRE_ACTOR , A.APELLIDO, P.PRESUPUESTO;

--MUESTRE EN QUE CINE DAN CADA PELICULA--
SELECT P.ID_PELICULA , P.NOMBRE_PELICULA, C.NOMBRE_CINE
FROM PELICULA p
JOIN CINE c
ON c.ID_PELICULA = p.ID_PELICULA
GROUP BY p.ID_PELICULA , p.NOMBRE_PELICULA , c.NOMBRE_CINE;

--MUESTRE TODOS LOS ACTORES QUE GRABARON UNA PELICULA CUYO PRESUPUESTO ES MAYOR AL PROMEDIO DEL PRESUPUESTO DE TODAS LAS PELICULAS
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, P.PRESUPUESTO
FROM ACTOR a
JOIN PELICULA p
ON p.ID_ACTOR = a.ID_ACTOR
WHERE p.PRESUPUESTO > (SELECT AVG(p.PRESUPUESTO) FROM PELICULA p)
GROUP BY A.ID_ACTOR ,A.NOMBRE_ACTOR , A.APELLIDO, P.PRESUPUESTO;

--MUESTRE TODOS LOS ACTORES QUE GRABARON UNA PELICULA CUYO PRESUPUESTO ES MENOR AL PROMEDIO DEL PRESUPUESTO DE TODAS LAS PELICULAS
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, P.PRESUPUESTO
FROM ACTOR a
JOIN PELICULA p
ON p.ID_ACTOR = a.ID_ACTOR
WHERE p.PRESUPUESTO < (SELECT AVG(p.PRESUPUESTO) FROM PELICULA p)
GROUP BY A.ID_ACTOR ,A.NOMBRE_ACTOR , A.APELLIDO, P.PRESUPUESTO;
```

c) Consultas Usando Group By:

```
                                                    --CONSULTAS GROUP BY--
--TOTAL ACTORES POR SU EDAD --
SELECT EDAD , COUNT(*) AS TOTAL_ACTOR FROM ACTOR GROUP BY EDAD;
--TOTAL ACTORES POR SU NOMBRE--
SELECT NOMBRE_ACTOR , COUNT(*) AS TOTAL_ACTOR FROM ACTOR GROUP BY NOMBRE_ACTOR;
--PROMEDIO DE EDAD ACTORES SEGUN SU NOMBRE--
SELECT NOMBRE_ACTOR , AVG(EDAD) AS PROMEDIO_EDAD FROM ACTOR GROUP BY NOMBRE_ACTOR ORDER BY AVG(EDAD);
--PROMEDIO DE ID DE ACTOR SEGUN SU NOMBRE--
SELECT NOMBRE_ACTOR , AVG(ID_ACTOR) AS PROMEDIO_ID FROM ACTOR GROUP BY NOMBRE_ACTOR ORDER BY AVG(ID_ACTOR);
```

d) Consultas usando Select anidado:

```
                                                --CONSULTAS ANIDADAS--
--CONSULTA ACTORES POR SOBRE EL PROMEDIO DE EDAD--
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, A.EDAD FROM ACTOR A WHERE A.EDAD > (SELECT AVG(EDAD) FROM ACTOR A);

--CONSULTA ACTORES POR DEBAJO DEL PROMEDIO DE EDAD--
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, A.EDAD FROM ACTOR A WHERE A.EDAD < (SELECT AVG(EDAD) FROM ACTOR A);

--EDAD DEL ACTOR SOBRE EL PROMEDIO Y SU ID Y DE LA PELICULA MENORES A 10--
SELECT A.ID_ACTOR, A.NOMBRE_ACTOR, A.APELLIDO, A.EDAD FROM ACTOR A WHERE A.EDAD > (SELECT AVG(EDAD) FROM ACTOR A)
AND A.ID_ACTOR NOT IN (SELECT P.ID_PELICULA FROM PELICULA P WHERE P.ID_PELICULA < 10);

--OBTENGA LAS PELICULAS QUE GRABÓ DENNY MALONE--
SELECT * FROM PELICULA p
WHERE ID_ACTOR = (SELECT ID_ACTOR FROM ACTOR a WHERE UPPER(NOMBRE_ACTOR)='DENNY' AND UPPER(APELLIDO) = 'MALONE');
```

Medición de tiempos

1) **500 DATOS**

A- CONSULTAS SIMPLES

AND (ACTOR): 17ms

OR (ACTOR): 16ms

LIKE (ACTOR): 13ms

NOT LIKE (ACTOR): 25ms

LENGTH (ACTOR): 15ms

IN (ACTOR): 14ms

B- CONSULTAS MULTITABLA

JOIN 1 (ACTOR, PELICULA): 23ms

JOIN 2 (ACTOR, PELICULA): 30ms

JOIN 3 (ACTOR, PELICULA): 25ms

JOIN 4 (ACTOR, PELICULA): 25ms

C- CONSULTAS GROUP BY

GROUP BY 1 (ACTOR): 13ms

GROUP BY 2 (ACTOR): 15ms

GROUP BY 3 (ACTOR): 17ms

GROUP BY 4 (ACTOR): 20ms

D- CONSULTAS ANIDADAS

SELECT 1 (ACTOR): 19ms

SELECT 2 (ACTOR, PELICULA): 29ms

SELECT 3 (ACTOR, PELICULA): 29ms

SELECT 5 (ACTOR): 17ms

2) 5000 DATOS

A- CONSULTAS SIMPLES

AND (ACTOR): 34ms

OR (ACTOR): 35ms

LIKE (ACTOR): 21 ms

NOT LIKE (ACTOR): 85ms

LENGTH (ACTOR): 29ms

IN (ACTOR): 17ms

B- CONSULTAS MULTITABLA

JOIN 1 (ACTOR, PELICULA): 115ms

JOIN 2 (ACTOR, PELICULA): 100 ms

JOIN 3 (ACTOR, PELICULA): 103 ms

JOIN 4 (ACTOR, PELICULA): 82 ms

C- CONSULTAS GROUP BY

GROUP BY 1 (ACTOR): 15 ms

GROUP BY 2 (ACTOR): 19 ms

GROUP BY 3 (ACTOR): 20 ms

GROUP BY 4 (ACTOR): 21 ms

D- CONSULTAS ANIDADAS

SELECT 1 (ACTOR): 48 ms

SELECT 2 (ACTOR, PELICULA): 83 ms

SELECT 3 (ACTOR, PELICULA): 92 ms

SELECT 5 (ACTOR): 22 ms

3) 50000 DATOS

A- CONSULTAS SIMPLES

AND (ACTOR): 322 ms

OR (ACTOR): 301 ms

LIKE (ACTOR): 133 ms

NOT LIKE (ACTOR): 636 ms

LENGTH (ACTOR): 270 ms

IN (ACTOR): 70 ms

B- CONSULTAS MULTITABLA

JOIN 1 (ACTOR, PELICULA): 981 ms

JOIN 2 (ACTOR, PELICULA): 893 ms

JOIN 3 (ACTOR, PELICULA): 514 ms

JOIN 4 (ACTOR, PELICULA): 533 ms

C- CONSULTAS GROUP BY

GROUP BY 1 (ACTOR): 50 ms

GROUP BY 2 (ACTOR): 60 ms

GROUP BY 3 (ACTOR): 56 ms

GROUP BY 4 (ACTOR): 64 ms

D- CONSULTAS ANIDADAS

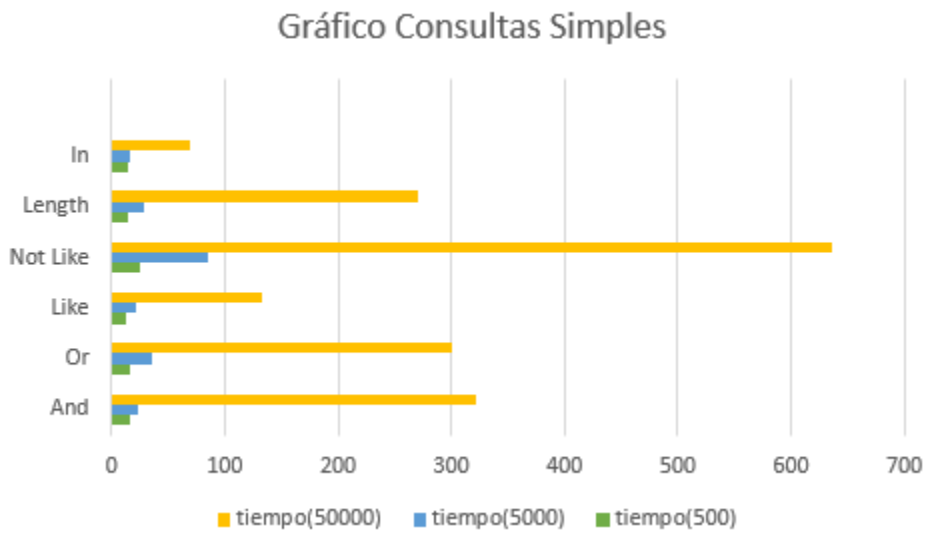
SELECT 1 (ACTOR): 475 ms

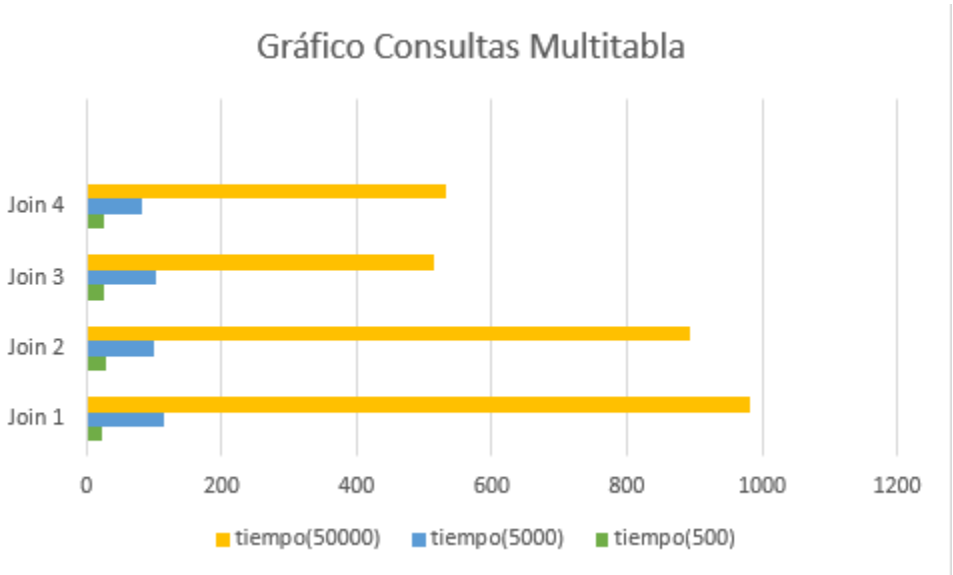
SELECT 2 (ACTOR, PELICULA): 692 ms

SELECT 3 (ACTOR, PELICULA): 610 ms

SELECT 4 (ACTOR): 302 ms

Resultados gráficos





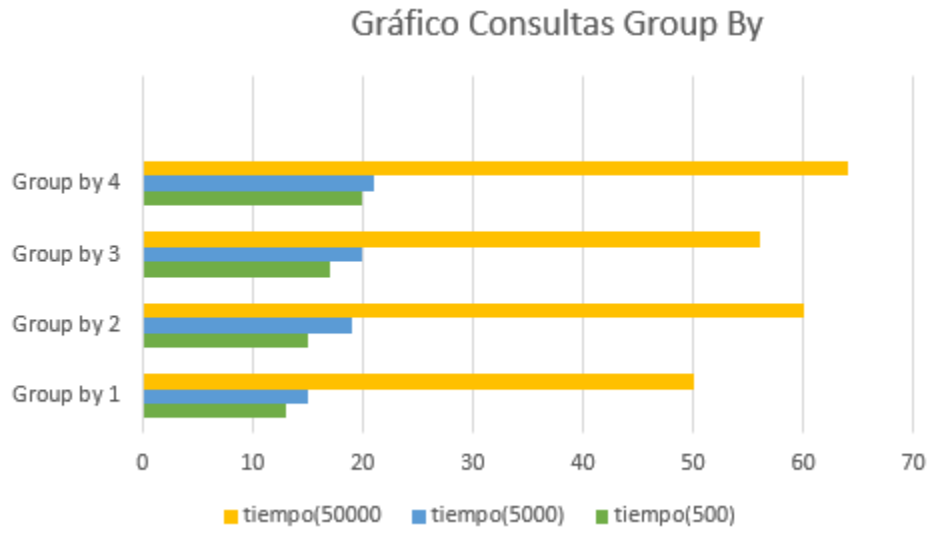


Gráfico Consultas Group By
Group by 4
Group by 3
Group by 2
Group by 1
0
10
20
30
40
50
60
70
tiempo(50000
tiempo(5000)
tiempo(500)


Gráfico Consultas Select Anidado
Select 4
Select 3
Select 2
Select 1
0
100
200
300
400
500
600
700
800
tiempo(50000)
tiempo(5000)
tiempo(500)

Conclusión

Al momento de añadir los datos, los tiempos se comportaron tal cual como se esperaba, ya que al ir aumentando la cantidad de datos, aumentó también la cantidad de tiempo necesaria para que estos se unieran a la base de datos.

En cuanto a los resultados, podemos observar que los datos de tipo de consulta, ya sea una simple o una anidada, por ejemplo, se comportan de igual manera , es decir, sus datos y gráfica van de manera ascendente regularmente, se mueven de una manera continua, por lo que podemos deducir que el tiempo se calculó correctamente en cada uno de los parámetros.

De manera global, creemos que los puntos y/o objetivos a considerar fueron cumplidos en su totalidad, generando datos, código y su respectiva monitorización a lo largo del informe. Mientras que si consideramos el código en si, podríamos haber mejorado un poco en rebajar líneas de comandos, pero en esta ocasión no fue necesario hacerlo.