**Base de Datos 2**

Mini Tp

# **Alumno: Espinosa Lucas**

# **Profesor: Bisso Ignacio**

## **Fecha de Entrega: 22/04/2017**

##### 1 Armar un shell script llamado carga\_inicial.sh que cree las tablas ALUMNO, MATERIA y CURSA y les cargue 10000 alumnos, 20 materias y entre 0 y 4 inscripciones por alumno al azar respectivamente. El script debe borrar las tablas si existen previamente (ver DROP IF EXISTS).

##### 2 Armar un shell script con SQL embebido llamado query1.sh que ejecute el query de la guia de laboratorio que devuelve los nombres de los alumnos que cursan 3 materias usando un subquery correlacionado.

##### 3 Armar un shell script con SQL embebido lamado query2.sh que ejecute el query de la guia de laboratorio que devuelve los nombres de los alumnos que cursan 3 materias sin usar un subquery correlacionado

##### 4 Ejecutar los scripts de 2 y 3 anteponiendo el comando time para medir el tiempo de ejecución, comparar y explicar brevemente la causa de la duración de cada query.

##### 5 Es posible acelerar alguno de los queries ? En caso de ser posible implementar la mejora en un tercer shell script (query3.sh) y explicar cómo se logró la mejora y el tiempo resultante.

##### Tiempo de query1.sh: real 0m6.530s

##### Tiempo de query2.sh: real 0m0.065s

##### Query con SubConsulta Correlacionada.

##### El incremento en el tiempo de respuesta en la consulta del Query correlacionado se debe a que por cada registro candidato de la tabla de alumnos se realiza la subconsulta, si este registro candidato califica para la subconsulta, este se incluye en el conjunto de resultados, por lo que en este caso realiza 10.000 veces la subconsulta, debido a esto es que el tiempo de respuesta de la consulta del Query correlacionado es más lento. pero se encontró una ventaja en esta consulta. Al buscar los legajos que contienen 0 materias asignadas, nos devuelve el resultado de la misma, gracias al funcionamiento de la consulta correlacionada explicada anteriormente.

##### 

Query sin Subconsulta Correlacionada

Como se observa hay una gran diferencia en el tiempo de respuesta de la consulta no correlacionada,

la cual es mas rapida que la consulta con Subquery Correlacionado debido a que el motor de la base de datos analiza y realiza la consulta de la manera más eficiente buscando los registro que califiquen con las condiciones dadas. Pero se encontró una desventaja, al realizar la consulta por los legajos que contienen 0 materias asignadas el resultado fue 0, lo cual no es correcto (en todos los demas casos el resultado fue el mismo que con la consulta correlacionada).

Optimización de Consultas

La forma de optimizar las consultas es con índices, en este caso como la tabla cursa contiene a las columnas Legajo y Código las cuales por separado tienen datos repetidos pero agrupadas los registros son únicos podría utilizarse estas 2 columnas como Primary Key con lo cual tendríamos una forma de indexar registros , pero al no estar ordenado sólo sería útil si se buscar un registro específico y se conoce el índice de antemano. Con lo cual la mejor opción es utilizar la columna de código que tiene 20 valores diferentes y al no tener un orden, cada valor distinto de esta columna utilizara una matriz de punteros a la posición de los registros que contengan este valor en la tabla cursa, es decir solo realizará la búsqueda en la matriz optimizando así los tiempos de respuesta .

**Código Generado**

*carga\_inicial.sh*

*echo* *sudo service postgresql restart* *psql lucasespinosa <CrearTablas.sql* *psql minitp <materia.sql* *psql minitp <alumnos.sql* *psql minitp <funcionCursa.sql*

**Función Para Cargar Cursa**

*CREATE OR REPLACE FUNCTION public.CargarCursa() RETURNS void AS*

*$BODY$*

*DECLARE*

*LEG int;*

*CONT int;*

*MAT VARCHAR(7);*

*LEGMAT int;*

*BEGIN*

*FOR I IN 1..8500 LOOP*

*LEG := (SELECT legajo FROM alumno ORDER BY random() LIMIT 1);*

*CONT := (SELECT COUNT(\*) FROM cursa WHERE cursa.legajo = LEG);*

*MAT := (SELECT m2.codigo FROM materia AS m2 ORDER BY random() LIMIT 1);*

*LEGMAT := (SELECT COUNT(\*) FROM cursa WHERE cursa.legajo = LEG AND cursa.codigo = MAT);*

*IF CONT <4 AND LEGMAT = 0*

*THEN*

*INSERT INTO cursa (legajo,codigo) VALUES (LEG,MAT);*

*END IF;*

*END LOOP;*

*END;$BODY$*

*LANGUAGE plpgsql VOLATILE*

*COST 100;*

*ALTER FUNCTION public.CargarCursa()*

*OWNER TO lucasespinosa;*

###### *SELECT CargarCursa();*

#### **En esta Función se controla que ningún legajo tenga más de cuatro materias asignada y que a ningún legajo se le asigne más de una vez la misma materia.**

## **Query Correlacionado**

SELECT \* FROM alumno WHERE

(SELECT COUNT(\*) FROM cursa WHERE alumno.legajo = cursa.legajo)=4;

## **Query No Correlacionado**

SELECT c.legajo, a.nombre,a.apellido from alumno AS a, cursa AS c WHERE c.legajo = a.legajo

GROUP BY c.legajo, a.nombre, a.apellido HAVING COUNT(\*)=4