Instrucciones extendidas entrega 4

Mauricio Monsalve

19 de junio

1 Del informe

1.1 Objetivos

Evaluar la eficiencia lograda en la base de datos al usar diferentes planes de índices para un conjunto de consultas críticas.

1.2 Requerimientos

Para esta entrega, es necesario:

1. Definir tres consultas complicadas: que incluyan al menos tres tablas en la evaluación y tengan agrupación y/o consultas anidadas. Las consultas deben ser costosas de evaluar (ojo con las condiciones WHERE).
2. Ser capaces de llenar tablas hasta con 10.000 elementos (cada una de las tablas importantes de las consultas) dinámicamente. O sea, variar el contenido de las tablas. Esto se puede hacer con un script externo o con copias de las tablas con una cierta cantidad de datos en ellas. Por ejemplo, SELECT ... FROM A,B,C WHERE... requiere A, B, C, por lo que se podrían tener A1000, B1000, C1000 como tablas con 1000 registros, A5000, B5000, C5000 con 5000 registros, etc.
3. Definir planes de índice con sólo un tipo de índice (variable). Deberán usar planes sin índices, con índices B+ y con índices de hashing.

1.3 Experimentos

En el trabajo será necesario realizar experimentos del rendimiento de la base de datos ante distintos planes de índices. En particular,

- Para cada consulta:
  - Por cada plan (sin índices, B+ y hashing):
    - Obtener los tiempos de consulta con 100, 1000, 5000 y 10000 registros por tabla.

Nótese que cada experimento debe realizarse varias veces para absorber los rendimientos variables del sistema. Así, el tiempo promedio por experimento debiera suavizar rendimientos extraños de sistema.

El número de veces que se repite un experimento se deja a criterio, pero debiera ser superior a las 10 veces.

Nótese que, sin contar las veces que se repite el experimento, hay que realizar $3 \times 3 \times 4 = 36$ experimentos distintos. Se recomienda usar un script para automatizar el experimento.

1.4 Estructura del informe

En el informe se deben presentar las consultas importantes, su significado y la justificación de su costo (que sean algo más de un simple join natural entre tres tablas). Luego se debe indicar qué atributos tendrán índices y por qué. Después de esto se detallarán los resultados experimentales.

Los resultados experimentales se presentarán como gráficos de N v/s tiempo, donde N es el tamaño de cada tabla (100, 1000, 5000 y 10000). Esto significa que habrán tres gráficos por consulta, significando un total de nueve gráficos.

La presentación del informe será considerada. Nótese que deberá haber coherencia entre el plan de índices y los resultados experimentales; un buen plan de índices logrará resultados coherentes con la teoría en cuanto al costo de las consultas.

Una vez terminados los gráficos, se deberán presentar las conclusiones. Contraste la teoría con la práctica en términos de aproximación de costos y vea cuál es el beneficio real (cuantitativo) del uso de índices. Vea cómo un índice supera a otro en determinados casos.

2 La parte técnica

2.1 EXPLAIN ANALYZE o cómo evaluar planes de consulta

En PostgreSQL, la instrucción EXPLAIN permite estimar el tiempo que tardará una consulta. Por ejemplo, EXPLAIN SELECT * FROM A; entrega el tiempo estimado de evaluación de la consulta. Adicionalmente, EXPLAIN VERBOSE SELECT * FROM A; entrega el plan de consulta elegido por el optimizador de PostgreSQL. Esto puede ser bastante importante para determinar cómo PostgreSQL usará los índices y con qué algoritmo realizará la evaluación. Pero la estimación no es algo que deba ser usado en el experimento, es sólo una guía.

Para entregar el tiempo preciso de la evaluación de una consulta, se puede usar la instrucción EXPLAIN ANALYZE. El modificador ANALYZE indica que es necesario que la consulta se realice en la práctica, evitando que se entregue una estimación en los tiempos. De esta manera, los tiempos reales de evaluación de consulta serán presentados.

2.2 El atributo CURRENT_TIME o de cómo ingresar tiempos a una tabla

PostgreSQL tiene un atributo implícito llamado *current_time* que indica la hora actual en el formato hora:minutos:segundos, los segundos teniendo decimales: es un atributo de tipo *time*. La consulta SELECT current_time; entrega la hora actual.

¿Cómo aplicar esto a los experimentos? Agregando los tiempos a una tabla. Por ejemplo:

```
CREATE TABLE tiempos1 (tiempos TIME);
INSERT INTO tiempos1 VALUES(CURRENT_TIME);
SELECT ... ;                          (consulta complicada, 1ª vez)
SELECT ... ;                          (consulta complicada, 2ª vez)
SELECT ... ;                          (consulta complicada, 3ª vez)
SELECT ... ;                          (consulta complicada, 4ª vez)
 ...
SELECT ... ;                          (consulta complicada, 10ª vez)
INSERT INTO tiempos1 VALUES(CURRENT_TIME);
```

En la tabla *tiempos1* habrán quedado registrados los tiempos que tardó el experimento. De esto, el tiempo promedio de consulta se puede calcular fácilmente mediante $\frac{T_f - T_0}{10}$. Una misma tabla se puede ocupar para la misma serie de experimentos, pero considerando otras cantidades de elementos por tabla.

Tenga especial cuidado con considerar los tiempos de inserción de elementos en tablas. Estos datos NO se deben considerar pues no forman parte de los SELECT que se piden evaluar.

2.3 Cómo copiar valores de una tabla a otra

Suponga que tiene la tabla Clientes pero que necesita diferentes cantidades de datos para esta tabla. Para eso, suponga que ya dispone de Clientes500, Clientes1000, etc. ¿Cómo copiar los datos entre tablas?

La solución simple es usar INSERT INTO. Esta consulta posee una sintaxis muy cómoda para insertar datos resultantes de un SELECT. En el caso particular de la tarea, se puede hacer lo siguiente:

```
CREATE TABLE Clientes( ... );
INSERT INTO Clientes SELECT * FROM Clientes500;
 ...
 (aqu\'i ud realiza el experimento)
 ...
DELETE FROM Clientes;       (sin WHERE porque deseamos eliminar todo)
INSERT INTO Clientes SELECT * FROM Clientes1000;
 ...
 (otro experimento)
 ...
```

Este tratamiento facilita la creación de un gran script para automatizar el experimento.

2.4 Automatización de experimentos

Se recomienda que cada bloque de experimentos se haga de manera separada para evitar reconocer demasiado tarde la presencia de errores en los scripts. Además, esto facilita la creación de scripts más grandes basados en scripts más pequeños.

Suponga que desea automatizar la consulta "SELECT COMPLICADO" que usar las tablas A, B y C. Naturalmente, ud. dispone de A500, B500, C500, A1000, B1000, etc. Suponga que desea automatizar la consulta sin índices, entoces puede usar un script de la siguiente forma:

```
 ---------- experimento_1_1.txt ----------
DELETE FROM A; DELETE FROM B; DELETE FROM C;
INSERT INTO A SELECT * FROM A500;
INSERT INTO B SELECT * FROM B500;
INSERT INTO C SELECT * FROM C500;
\i consulta_1.txt
DELETE FROM A; DELETE FROM B; DELETE FROM C;
INSERT INTO A SELECT * FROM A1000;
INSERT INTO B SELECT * FROM B1000;
INSERT INTO C SELECT * FROM C1000;
\i consulta_1.txt
DELETE FROM A; DELETE FROM B; DELETE FROM C;
INSERT INTO A SELECT * FROM A5000;
INSERT INTO B SELECT * FROM B5000;
INSERT INTO C SELECT * FROM C5000;
\i consulta_1.txt
DELETE FROM A; DELETE FROM B; DELETE FROM C;
INSERT INTO A SELECT * FROM A10000;
INSERT INTO B SELECT * FROM B10000;
INSERT INTO C SELECT * FROM C10000;
\i consulta_1.txt
```

En el ejemplo anterior, el archivo *consulta_1.txt* puede tener la siguiente forma:

```
 ---------- consulta_1.txt ----------
INSERT INTO tiempos1 VALUES(CURRENT_TIME);
SELECT COMPLICADO;                          (consulta complicada, 1ª vez)
SELECT COMPLICADO;                          (consulta complicada, 2ª vez)
 ...
SELECT COMPLICADO;                          (consulta complicada, 10ª vez)
INSERT INTO tiempos1 VALUES(CURRENT_TIME);
```

Así, la tabla tiempos1 registrará todos los experimentos realizados para una consulta y un plan de índices específico. Para otra consulta, se puede usar otro archivo, digamos *consulta_2.txt* o *consulta_3.txt*, y naturalmente usar otro *experimento_n_m.txt*. Lo mismo vale para otro plan de índices sobre las tablas especificadas.

Para ejecutar el experimento en PostgreSQL, desde el terminal psql se puede escribir $\backslash i$ *experimento_n_m.txt*. De esta manera, los experimentos se pueden automatizar en gran manera (y dejarse corriendo durante una noche, por ejemplo). Pero mucho ojo con verificar que los archivos estén bien escritos y que el experimento está realizando la labor correcta para evitar la pérdida de tiempo.

2.5 Cómo generar tablas Tabla1000, Tabla5000, etc.

Estas tablas pueden ser generadas mendiante scripts generados por lenguajes de programación (como Perl, Java, C, etc.).

Una alternativa más cómoda puede ser tener tablas con palabras y números (ejemplo PALS y NUMS). Luego se puede usar INSERTO INTO ... SELECT * FROM PALS, PALS, NUMS, PALS; para una tabla con atributos (varchar, text, integer, varchar), por ejemplo. Si es necesario tener especial respecto por llaves foráneas, se pueden usar los atributos de las tablas referenciadas ya creadas para generar la nueva tabla. Y para las llaves primarias, es importante lograr su unicidad (la estrategia a usar puede depender del caso). En particular, si las llaves primarias son numéricas, se puede usar una consulta de la forma SELECT count(*),A1.* FROM A as A1, A as A2 WHERE A1>A2; siendo > alguna relación de orden (usando lógica) entre registros la misma tabla.

También es posible partir de la tabla más grande, por ejemplo A10000. Luego es posible crear tablas más pequeñas (A500, A1000, A5000) mediante consultas SELECT restrictivas en el WHERE.

Si las llaves foráneas son demasiado problemáticas, remuévalas. Pero no nunca remueva la restricción de llave primaria.