Taller sobre optimización de consultas

Preparando el contenedor de docker

Descargue una imagen de postgres

docker pull postgres

Arranque un nuevo contenedor a partir de esta imagen. Por facilidad nombrelo ***some-postgres***

docker run --name some-postgres -e POSTGRES_PASSWORD=mysecretpassword -d postgres

Si quiere puede conectar el puerto del contenedor con un puerto local

docker run --name some-postgres -p 5432:5432 -e POSTGRES_PASSWORD=mysecretpassword -d postgres

Descargue la base de datos dvdrental http://www.postgresqltutorial.com/postgresql-sample-database/

Descomprima el archivo .zip y copie el archivo dvdrental.rar sobre el contenedor some-postgres

docker cp dvdrental.rar some-postgres:./

Conectese sobre el contenedor

docker exec -i -t some-postgres /bin/bash

Crear la nueva base de datos en postgres e importe los datos en ella

psql -h localhost -U postgres

> CREATE DATABASE dvdrental;

>\q

 # pg_restore -U postgres -d dvdrental ./dvdrental.tar

Ahora puede conectarse nuevamente a postgres y comenzar el taller

psql -h localhost -U postgres

Conociendo nuestra base de datos

\l; lista las bases de datos
\c musica; Cambia a la base de datos musica
\dt; lista de relaciones
\d tabla; Describe la relación tabla

Listar el número de páginas y tuplas de una tabla

SELECT relpages, reltuples FROM pg_class WHERE relname = 'rental';

¿Cúal es el tamaño de cada bloque?

Usando EXPLAIN para ver el plan de ejecución de una consulta

EXPLAIN SELECT COUNT(*) FROM rental, payment WHERE amount < 5;
EXPLAIN SELECT COUNT(*) FROM rental, payment WHERE rental.rental_id = payment.rental_id AND amount < 5;

Ejercicios

Considere las siguientes 2 consultas

SELECT COUNT(*) FROM rental RIGHT JOIN payment ON rental.rental_id = payment.rental_id WHERE amount < 5; **count** **-------** **10978** **(1 row)**
SELECT COUNT(*) FROM rental LEFT JOIN payment ON rental.rental_id = payment.rental_id WHERE amount < 5; **count** **-------** **10978** **(1 row)**

Vemos sus planes de ejecución

Aggregate (cost=979.87..979.88 rows=1 width=8) **-> Hash Join (cost=510.99..952.42 rows=10980 width=0)** **Hash Cond: (payment.rental_id = rental.rental_id)** **-> Seq Scan on payment (cost=0.00..290.45 rows=10980 width=4)** **Filter: (amount < '5'::numeric)** **-> Hash (cost=310.44..310.44 rows=16044 width=4)** **-> Seq Scan on rental (cost=0.00..310.44 rows=16044 width=4)**
Aggregate (cost=317.90..317.91 rows=1 width=8) **-> Seq Scan on payment (cost=0.00..290.45 rows=10980 width=0)** **Filter: (amount < '5'::numeric)**

1. Explique la diferencia en tiempo si el resultado es el mismo

2. Realice una consulta que muestre la lista de clientes con la cantidad de peliculas que alquiraron y la cantidad de dinero que gastaron en la tienda por los alquileres.

3. Muestre cuanto dinero ha generado cada una de las categorias de las peliculas

4. Ahora muestre esta misma tabla pero filtrando por las categorias “Sports” y “Drama”.

Mire el orden en que se realizan los JOIN y diga si es el mismo orden propuesto por su consulta SQL. ¿Dónde se realizan los filtros?

5. Ahora muestre esta misma tabla pero filtrando por las categorias “Sports” y “Drama” para las peliculas que duren mas de 100 minutos

Compare este plan de ejecución con el plan anterior. ¿Cúales son las principales diferencias?¿Dónde se realizan los filtros?

SELECT email, count(title) as count, sum(amount) as amount FROM customer, rental, inventory, film, payment WHERE rental.customer_id = customer.customer_id AND rental.rental_id = payment.rental_id AND inventory.inventory_id = rental.inventory_id AND inventory.film_id = film.film_id GROUP BY email ORDER BY count DESC;

EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON) SELECT email, count(title) as count, sum(amount) as amount FROM customer, rental, inventory, film, payment WHERE rental.customer_id = customer.customer_id AND rental.rental_id = payment.rental_id AND inventory.inventory_id = rental.inventory_id AND inventory.film_id = film.film_id GROUP BY email ORDER BY count DESC;

psql -d dvdrental -U postgres -t -A -F"," -c "EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON) SELECT email, count(title) as count, sum(amount) as amount FROM customer, rental, inventory, film, payment WHERE rental.customer_id = customer.customer_id AND rental.rental_id = payment.rental_id AND inventory.inventory_id = rental.inventory_id AND inventory.film_id = film.film_id GROUP BY email ORDER BY count DESC;" > query.csv

psql -d dvdrental -U postgres -t -A -F"," -c "EXPLAIN (ANALYZE, COSTS, VERBOSE, BUFFERS, FORMAT JSON) SELECT email, count(title) as count, sum(amount) as amount FROM customer, inventory, film, (SELECT amount, rental.rental_id as rental_id, rental.customer_id as customer_id, inventory_id FROM rental JOIN payment ON rental.rental_id = payment.rental_id) as rp WHERE rp.customer_id = customer.customer_id AND inventory.inventory_id = rp.inventory_id AND inventory.film_id = film.film_id GROUP BY email ORDER BY count DESC;" > query2.csv

docker cp some-postgres:./query.csv ./

http://tatiyants.com/postgres-query-plan-visualization/