SISTEMAS INFORMÁTICOS
Memoria práctica 4

Muuuvie
Web de venta de películas

Marta García Marín
Jesús Daniel Franco López
Grupo 1402
Pareja 12

Índice

Introducción **3**

Optimización **3**

- Ejercicio A: Índices 3
- Ejercicio B: Preparar consultas 3
- Ejercicio C: Cambiar la forma de realizar una consulta 3
- Ejercicio D: Generación de estadísticas 4

Transacciones y deadlocks **5**

- Ejercicio E: Estudio de transacciones 5
- Ejercicio F: Estudio de bloqueos y deadlocks 5

Seguridad **5**

- Ejercicio G: Acceso indebido a un sitio web 5
- Ejercicio H: Acceso indebido a información 5

Lista de ficheros adjuntos **6**

Referencias **6**

Introducción

SI USAMOS ALGUNO MÁS LO AÑADIMOOSSS

ANALYZE: recopila estadísticas sobre el contenido de las bases de datos y las guarda, permitiendo el uso de estas en un futuro para reducir tiempos y costes. Ejecuta las consultas suministradas y muestras los tiempos de ejecución.
EXPLAIN: muestra el plan de ejecución para la consulta requerida, indicando cómo serán recorridas las tablas, así como los algoritmos utilizados para la unión de estas. Además, muestra el coste inicial (para que se muestre la primera fila) y el coste total (para que se muestren todas las filas solución). No se ejecuta la sentencia realmente.
EXPLAIN y ANALYZE se suelen utilizar juntos, mostrando así el plan de ejecución y el resultado real de la sentencia. Además permite guardar estadísticas de estas ejecuciones para mejorar las búsquedas en el futuro.
CREATE INDEX: construye un índice en las columnas indicadas de una tabla concreta, mejorando así el rendimiento de la base de datos a la hora de acceder a los valores de estas columnas.

Optimización

Ejercicio A: Índices

En primer lugar, mediremos el rendimiento de una consulta SQL y estudiaremos distintos planes de ejecución, comparándolos entre sí, mediante el uso del comando EXPLAIN. Para ello, hemos creado el fichero *"clientesDistintos.sql"* (adjunto a esta memoria de prácticas). En la consulta, hemos mostrados el número de clientes distintos que tienen pedidos en un mes dado y hemos creado distintos índices aplicados a las columnas o atributos X, Y y Z. Así, la Tabla 1 muestra los principales resultados obtenidos:

	Consulta Base	Índice en columna X	Índice en columna Y	Índice en columna X e Y
Coste total	XX	XX	XX	XX
Número total de operaciones	XX	XX	XX	XX

Tabla 1. Rendimiento del uso de un índice sobre una consulta SQL.

En general, los resultados obtenidos muestran que la creación de un índice en la columna Y mejora el rendimiento de la consulta en XXXX veces (en función del coste total). Así, la consulta Y tuvo un número total de operaciones de XX, mientras que la consulta base tuvo un número total de operaciones de XX contra la base de datos. Como conclusión, podemos

decir que el uso de índices acelera las búsquedas y mejora el rendimiento para los resultados obtenidos.

Ejercicio B: Preparar consultas

….. un apartado parecido al anterior, podéis jugar con tablas, gráficas o comentarios. Pero usad un lenguaje directo donde se haga referencia a los datos que mostráis en la tabla y donde además se presente la tabla como he hecho en el primer párrafo del punto anterior.

Ejercicio C: Cambiar la forma de realizar una consulta
Para observar los resultados de este ejercicio basta con ejecutar **cat ejercicioC.sql | psql -U alumnodb si1**, donde se llega a las siguientes conclusiones:

	Consulta 1	Consulta 2	Consulta 3
Coste inicial	3961.65	4537.41	0.00
Coste total	4490.81	4539.41	4640.83
Planning time	0.672 ms	0.295 ms	0.138 ms
Execution time	142.053 ms	185.285 ms	171.006 ms

Tabla 1. Generación de estadísticas sobre 3 consultas diferentes.

- El coste inicial en las 2 primeras alternativas supera los 3500 milisegundos, mientras que en la segunda alternativa es de 0 milisegundos, lo que indica que esta consulta muestra resultados nada más comenzar la ejecución.
- El tiempo de planificación es bastante más alto en la primera alternativa, pero el tiempo de ejecución total se reduce considerablemente en esta, lo que significa que esta alternativa es más rápida pero más difícil de encontrar.
- La segunda alternativa se lleva a cabo con dos Seq Scans que ocurren al mismo tiempo (al mismo nivel), por lo que una ejecución en paralelo sería beneficiosa
- La tercera alternativa realiza dos Subquerys, también al mismo nivel, lo que sería ventajoso en la ejecución en paralelo.

Ejercicio D: Generación de estadísticas
Para probar este ejercicio basta con ejecutar el fichero **countStatus.sql** una vez creada e inicializada la base de datos, donde se mostraran los diferentes planes de consulta antes de la creación de los índices, tras la creación de estos y una vez ejecutada la sentencia analyze sobre orders. Para que su visualización sea más clara, hemos dividido los resultados en dos tablas diferentes.

	Consulta 1 base	Consulta 2 base	Consulta 1 índice	Consulta 2 índice
Coste total	3507.18	3961.66	1496.53	1498.80
Planning time	0.385 ms	0.098 ms	0.177	0.064
Execution time	97.252 ms	179.109 ms	0.060 ms	131.581 ms

Tabla 1. Rendimiento del uso de índices sobre 3 consultas diferentes.

Observamos que el tiempo de ejecución y el coste se reducen considerablemente tras la creación de los índices en el campo status de orders. Además, el árbol de ejecución también cambia, pasando en ambas consultas de un Seq Scan on orders a un Bitmap Heap Scan on orders y un Bitmap Index Scan on i.

	Consulta 1	Consulta 2	Consulta 3	Consulta 4
Coste total	7.26	4211.82	2315.34	2930.86
Planning time	0.080	0.054	0.103	0.135
Execution time	0.032	127.801	24.838	46.576

Tabla 1. Rendimiento del analyze sobre 4 consultas diferentes.

Observamos que en la primera consulta el tiempo de ejecución y el coste se reducen hasta casi ser nulos, esto se debe a que al analizar la tabla orders se guarda la cantidad de elementos que tienen null, por lo que no habría que realizar la consulta como tal, si no que solo hay que acceder a este dato.
El resto de las consultas tienen unos valores más elevados, siendo el coste de la segunda casi el doble que el de las dos últimas y con un tiempo de ejecución bastante superior. En cambio, el tiempo de planificación es casi la mitad en este caso. No se muy bien porque es pero la 2 hace Gather, Partial aggregate y Parallel Seq Scan, y las otras 2 hacen Bitmap Heap Scan y Bitmap Index, que parece ser que es más sencillo
Los costes iniciales no los hemos tenido en cuenta ya que difieren de los costes finales por millonésimas, lo que implica que ninguna consulta muestra resultados nada más empezar su ejecución, si no que todos se muestran al final.

Transacciones y deadlocks

Ejercicio E: Estudio de transacciones
hfuewiñb
Ejercicio F: Estudio de bloqueos y deadlocks
guqwieñfcwqbgu

Seguridad

Ejercicio G: Acceso indebido a un sitio web
hfuewiñb
Ejercicio H: Acceso indebido a información
guqwieñfcwqbgu

Lista de ficheros adjuntos

Archivos sql:

- *clientesDistintos.sql*: Contiene las sentencias ejecutadas en las mediciones del ejercicio A.
- *ejercicioC.sql*: Contiene las tres alternativas para la consulta propuesta en el ejercicio C.
- *countStatus.sql*: Contiene las consultas, los índices y las sentencias ANALYZE del ejercicio D.
- updPromo.sql: Contiene las sentencias ejecutadas para el ejercicio F

Archivos base de la app:

- *routes.py*:
- *database.py*:

Referencias

- Información sobre el planificador de PostgresSQL: https://www.postgresql.org/docs/9.5/index.html https://www.w3schools.com/sql/
- Información spobre vulnerabilidades en aplicaciones https://www.owasp.org/index.php/Main_Page
- Diapositivas de moodle
- Stack Overflow para resolver algunas dudas: https://es.stackoverflow.com/