Fragmentación con PostgresSQL

Profesor: Heider Sanchez
ACLs: Nicolás Arroyo, Mariana Capuñay, Sebastián Loza
Repositorio de Github: db2-partition-sql

Se le pide crear las siguientes tablas "employees" y "salaries" y cargar los datos del dataset "data2.zip". Talvez requiera realizar algunas modificaciones para que la data pueda ser cargado al PostgreSQL correctamente.

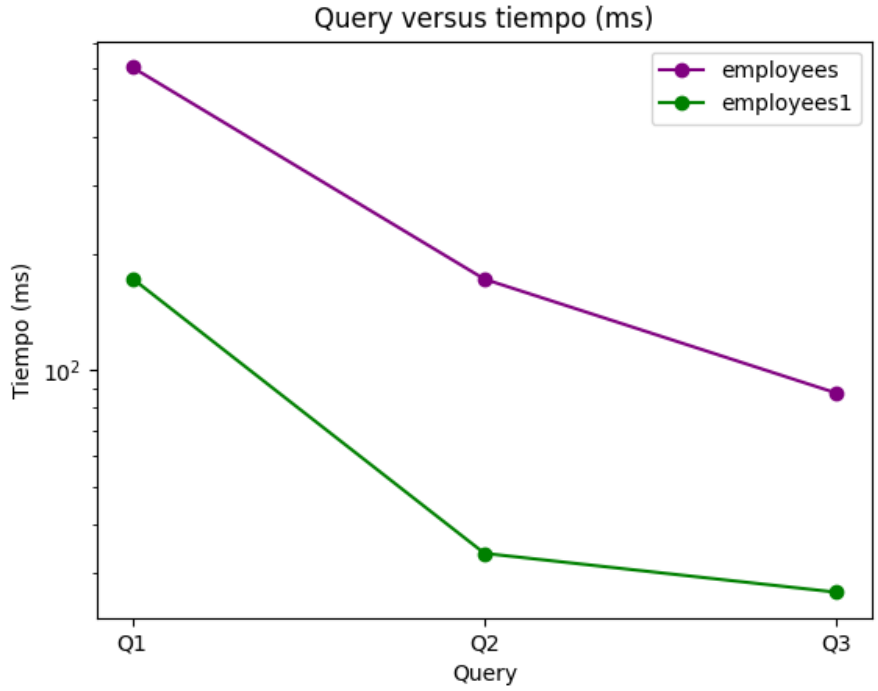
P1. Fragmentación con PARTITION BY LIST

- Crear la tabla employees1 indicando que será fragmentado por el atributo dept_no.
- Cargue los datos en la tabla employees1
- Analice los resultados que se obtienen al ejecutar una misma consulta en el atributo dept_no en ambas tablas.
- Use el comando Explain Analyze y coloque los tiempos en una tabla comparativa en (ms).

Usar `SET enable_partition_pruning = on;`

Restultados P1

	dept_no = d005	dept_no = d004	dept_no = d007
employees	605.311	171.956	87.348
employees1	172.772	33.744	26.785



Como se puede apreciar en el gráfica, los tiempos de ejecución son significativamente menores al usar la tabla con partición **employees1**. No obstante, la diferencia no llega a superar un orden de magnitud.

P2. Fragmentación con PARTITION BY RANGE

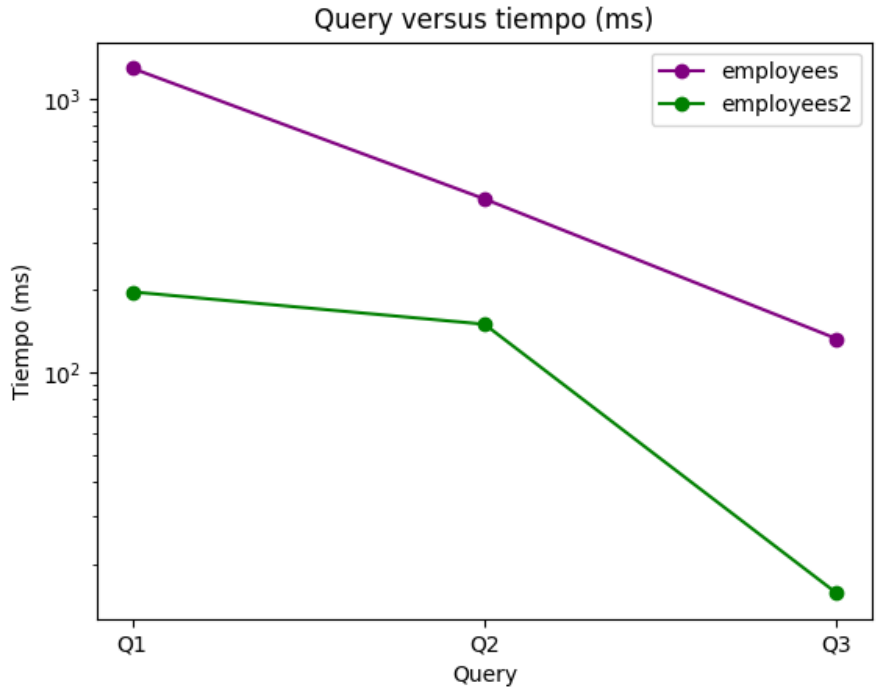
- Crear la tabla employees2 indicando que será fragmentado por rango sobre el año de la fecha de contrato:

  PARTITION BY RANGE (date_part('year', hire_date))

- Realizar la fragmentación con respecto al año (vector: [1988, 1994]).
- Cargue los datos en la tabla employees2.
- Elabore el cuadro de comparación de costos para tres consulta por rango diferentes sobre el atributo hire_date. Cuidar que la query acceda a una sola partición (ms).

Restultados P2 sin índice

	Query 1	Query 2	Query 3
employees	1288.744	430.376	133.116
employees2	197.354	150.429	15.734



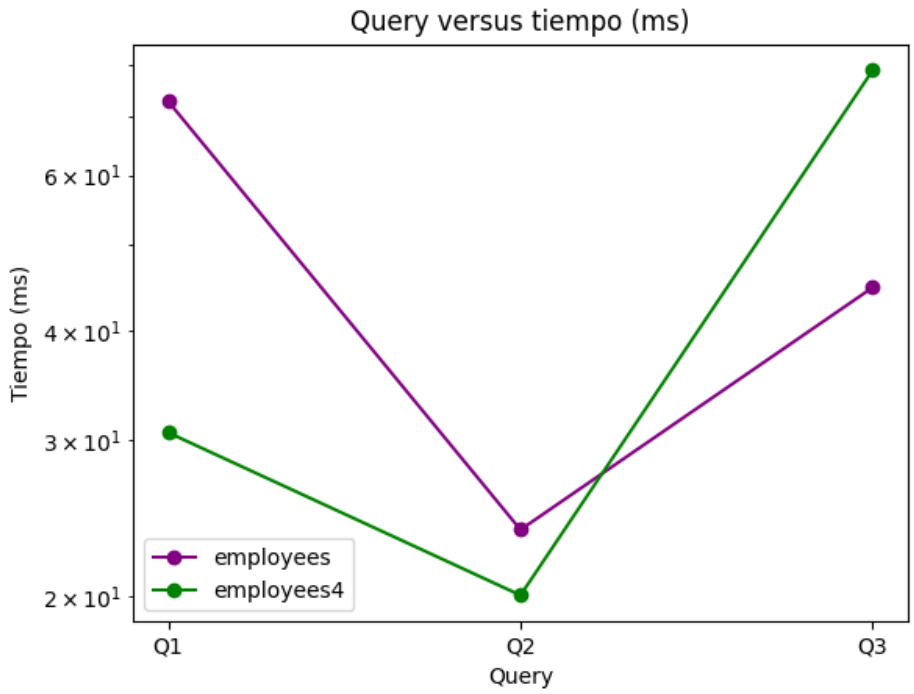
Como se puede apreciar en la gráfica, nuevamente se obtienen tiempos significativamente mejores para todas las consultas.

P2 con índice

- Aplique el índice btree sobre el atributo hire_date en ambas tablas. Note usted que el índice aplicado sobre la tabla particionada es un índice distribuido. Vuelva a ejecutar las consultas y anote los tiempos.
- Como se solicita aplicar el índice sobre el atributo hire_date, para que la partición siga siendo utilizada, es necesario que esta también esté en función de toda la fecha, no solo el año. Por ello, se crea una nueva table employees4 siguiendo un esquema de partición por el atributo hire_date.

Resultados P2 con índice

	Query 1	Query 2	Query 3
employees	72.751	23.780	44.762
employees4	30.643	20.042	78.900



Observando la gráfica, lo primero que se nota es que los valores en el eje Y están mucho más cercanos, de modo que la escala logarítmica acaba siendo prácticamente lineal. Esto significa que la diferencia entre los tiempos de ejecución (entre *employees* y *employees4*) es mucho menor que en los casos anteriores. Por otra parte, si bien la tendencia vista en los

2 grupos de resultados anteriores se mantiene para las consultas 1 y 2, para la consulta 3, la búsqueda sobre toda la tabla indexada resulta más eficiente que sobre la partición. Esto significa que el overhead de seleccionar la partición correcta es mayor a $lg_k(n) - lg_k(m)$ donde n es la cantidad total de registros y m es la cantidad de registros en la partición seleccionada.

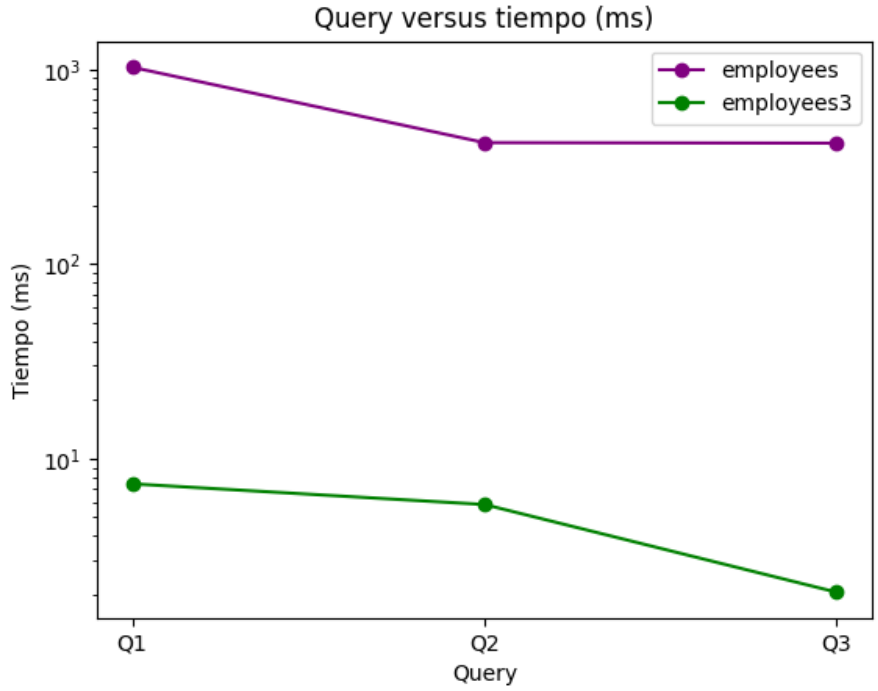
P3. Fragmentación con dos atributos.

- Crear la tabla employees3 para ser particionada en dos atributos.
- Agregar la columna "salary" a la tabla employees para guardar el último salario de cada empleado. Obtener el valor del salario desde la tabla "salaries".
- Considerar un segundo predicado de consulta sobre el atributo salary
  - Proponer un vector de particionamiento lo más equitativo posible.
- Hay dos opciones de fragmentación con dos atributos:
  - Opción 1: Primero fragmentar la tabla en el atributo hire_date. Luego aplicar una sub fragmentación sobre cada partición en el atributo salary.
  - Opción 2: Crear la tabla employees3 indicando ambos atributos en la partición: PARTITION BY RANGE (date_part('year', hire_date), salary
- Mostrar el plan de ejecución con Explain Analyze para tres consultas que incluya ambos atributos, hire_data y salary.

Resultados P3

Se eligió la opción de fragmentar primero por el atributo hire_date y luego por el atributo salary. Asimismo, cabe indicar que se formularon las consultas para que solo se acceda a una subpartición.

	Query 1	Query 2	Query 3
employees	1025.544	420.391	417.729
employees3	7.426	5.814	2.059



A partir de la tabla y la gráfica, se concluye que, siempre y cuando se garantice que la consulta acceda a una única partición, el uso de una tabla fragmentada con particiones es mucho más eficiente que el uso de la tabla original, llegando a tener una diferencia de tiempo de ejecución de hasta 2 órdenes de magnitud.