Laboratorio 9: Fragmentación con PostgreSQL

Prof. Heider Sanchez
ACLs: Ana María Accilio, Sebastián Loza

Introducción

Contexto del Caso

Imagina que eres parte del equipo de ingeniería de datos de una gran empresa multinacional con más de 100,000 empleados. El departamento de Recursos Humanos ha reportado que las consultas a la base de datos de empleados están siendo cada vez más lentas, especialmente cuando necesitan:

- Generar reportes por departamentos específicos
- Analizar tendencias de contratación por años
- Realizar análisis salariales

Como ingeniero de datos, tu misión es optimizar estas consultas implementando técnicas de fragmentación horizontal en PostgreSQL. Esta técnica permitirá dividir las tablas grandes en fragmentos más pequeños y manejables, mejorando significativamente el rendimiento de las consultas.

Objetivos de Laboratorio

- Implementar fragmentación horizontal para mejorar el rendimiento de consultas frecuentes
- Comparar el rendimiento entre tablas normales y fragmentadas
- Analizar diferentes estrategias de fragmentación según casos de uso específicos

Dataset

Trabajaremos con un subconjunto de datos de empleados que incluye información sobre:

- Datos personales y laborales de empleados
- Histórico de salarios

```sql
CREATE TABLE employees (
  emp_no int,
  birth_date date,
  first_name varchar(14),
  last_name varchar(16),
  gender character(1),
  hire_date date,
  dept_no varchar(5),
  from_date date
);

CREATE TABLE salaries (
  emp_no int,
  salary int ,
  from_date date,
  to_date date
);
```

Los datos necesarios se encuentran en el archivo `data2.zip` . Antes de cargar los datos, asegúrese de realizar las modificaciones necesarias para que sean compatibles con PostgreSQL.

P1. (6 pts) Optimización de Consultas por Departamento

El equipo de RRHH necesita generar reportes departamentales frecuentemente y ha notado que las consultas son especialmente lentas cuando filtran por departamento específico. Para solucionar esto, se le solicita implementar una fragmentación por lista (PARTITION BY LIST) basada en departamentos.

1. Crear una tabla fragmentada `employees1` por el atributo `dept_no` :
2. Analizar la distribución de empleados por departamento.
3. Crear tres fragmentos balanceados basados en el análisis anterior.
4. Cargar los datos en la tabla fragmentada `employees1` .
5. Analice los resultados que se obtienen al ejecutar una misma consulta en el atributo `dept_no` en ambas tablas y coloque los tiempos en la tabla comparativa:

Consulta	Sin Fragmentación	Con Fragmentación	Mejora (%)
WHERE dept_no = 'd005'			
WHERE dept_no = 'd004'			
WHERE dept_no = 'd007'			

Usar:

```sql
SET enable_partition_pruning = on;
```

P2. (7 pts) Fragmentación con `PARTITION BY RANGE`

El departamento de Planificación Estratégica necesita analizar las tendencias de contratación por períodos específicos para optimizar sus procesos de reclutamiento. Las consultas que involucran rangos de fechas de contratación están tomando demasiado tiempo, especialmente cuando analizan períodos específicos.

Se le solicita implementar una fragmentación por rango (RANGE) sobre el año de contratación para optimizar estas consultas frecuentes:

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

```
CREATE TABLE employees2 (
    -- Mismos campos que employees
    PRIMARY KEY (emp_no, hire_date)
) PARTITION BY RANGE (date_part('year', hire_date));
```

2. Analizar la distribución de contrataciones por año.
3. Crear tres fragmentos balanceados basados en el análisis anterior con respecto al año de contratación, (por ejemplo, vector de particionamiento `[1988, 1994]`).
4. Cargar los datos en la tabla fragmentada `employees2` .
5. Elaborar el cuadro de comparación de costos para tres consultas por rango diferentes sobre el atributo `hire_date` . Cuidar que la query acceda a una sola partición.

Consulta	Sin Fragmentación	Con Fragmentación	Mejora (%)
Query 1			
Query 2			
Query 3			

6. Aplique el índice B-tree sobre el atributo `hire_date` en ambas tablas. Note que el índice aplicado sobre la tabla particionada es un índice distribuido, por ende se crea también en las particiones. Vuelva a ejecutar las consultas, analice el resultado y anote los tiempos. ¿Hay mejoras significativas en la tabla particionada? (Tal vez requiera particionar en la fecha completa).

Consulta	Sin Fragmentación	Con Fragmentación	Mejora (%)
Query 1			
Query 2			
Query 3			

Responder:

¿Las mejoras son significativas? Considerar:

- Tiempo de respuesta
- Uso de recursos

P3. (7 pts) Fragmentación con dos atributos

El departamento de Compensaciones necesita realizar análisis complejos que combinan históricos de contratación y rangos salariales. Las consultas actuales que involucran tanto fechas de contratación como salarios están resultando extremadamente lentas.

Se le solicita implementar una estrategia de fragmentación doble que nos permita optimizar consultas que involucran tanto períodos de contratación como rangos salariales.

1. Crear la tabla `employees3` para ser particionada en dos atributos.

2. 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`.

3. Considerar un segundo predicado de consulta sobre el atributo `salary`. Proponer un vector de particionamiento lo más equitativo posible.

   Estrategias de Implementación:

   - **Opción 1: Fragmentación Anidada:** Primero fragmentar la tabla en el atributo `hire_date`. Luego aplicar una subfragmentación sobre cada partición en el atributo `salary`.

     ```
     -- Primer nivel: fragmentación por año
     CREATE TABLE employees3 PARTITION BY RANGE (date_part('year', hire_date));

     -- Segundo nivel: subfragmentación por salario
     CREATE TABLE emp_1988 PARTITION OF employees3
         FOR VALUES FROM (1988) TO (1989)
         PARTITION BY RANGE (salary);
     ```

   - **Opción 2: Fragmentación Compuesta** Crear la tabla `employees3` indicando ambos atributos en la partición:

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

4. Cargar los datos en la tabla fragmentada `employees3`.

5. Mostrar el plan de ejecución y anotar los tiempos para tres consultas que incluyan ambos atributos: `hire_date` y `salary`.

Consulta	Sin Fragmentación (ms)	Con Fragmentación (ms)	Mejora (%)
Query 1			
Query 2			
Query 3			

   Ejemplo de consultas:

   - Query 1: Empleados contratados en 1990 con salario > 50000
   - Query 2: Promedio salarial por año de contratación
   - Query 3: Distribución salarial por período

Responder:

- ¿Qué estrategia de fragmentación resultó más efectiva?
- ¿Cómo se comportan las consultas que acceden a múltiples fragmentos?

Entregable:

Subir el informe en formato PDF con la respectiva evidencia de los planes de ejecución, y adjuntar el script de BD (debe ser replicable).