Laboratorio 11

CURSO:

Base de Datos II (CS2042)

SECCIÓN:

Teoría 3

DOCENTE:

Sanchez Enriquez, Heider Ysaias

INTEGRANTES:

- Sebastian Nieto Paz
- Matías Alonso Meneses Roncal

P1

1. Creación de la tabla fragmentada

```
CREATE TABLE employees1 (
 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
) partition by list(dept_no);
```

2. Análisis de la distribución por departamento

```
30  ) partition by list(dept_no);
31
32  select dept_no,  count(*) as total
33  from employees
34  group by dept_no order by total desc;
35
```

Data Output | Messages | Notifications

Showin

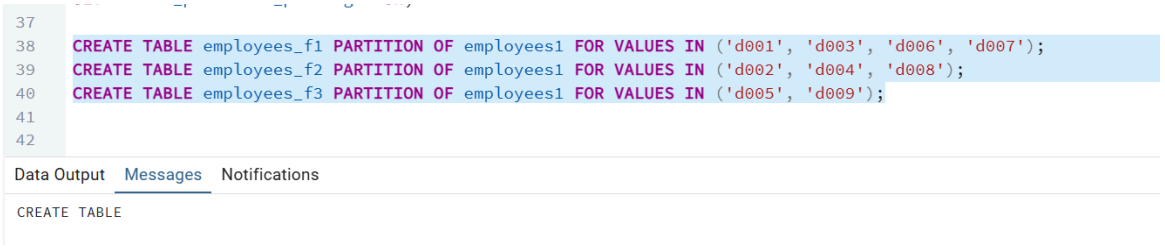
	dept_no character varying (5)	total bigint
1	d005	85707
2	d004	73485
3	d007	52245
4	d009	23580
5	d008	21126
6	d001	20211
7	d006	20117
8	d003	17786
9	d002	17346

Bajo esta suma la decisión de cómo dividir equitativamente fue hallando la suma total y dividendo entre 3 el resultado para hallar cuánto debería tener el grupo. Siguiendo esta lógica, los grupos con menor diferencia son:

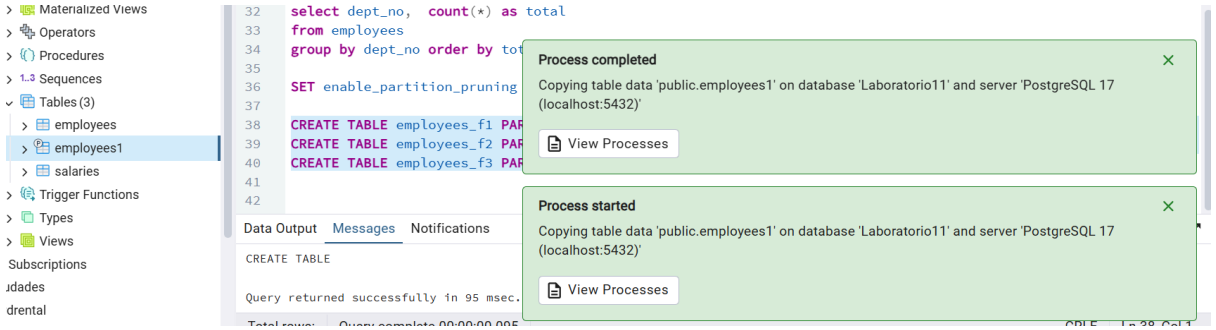
8	2	4	5	9	7	1	6	3
21126	17346	73485	85707	23580	52245	20211	20117	17786
111957			109287		110359			

F1 = 1,3,6,7
F2 = 2,4,8
F3 = 5,9

3. Creación de los fragmentos

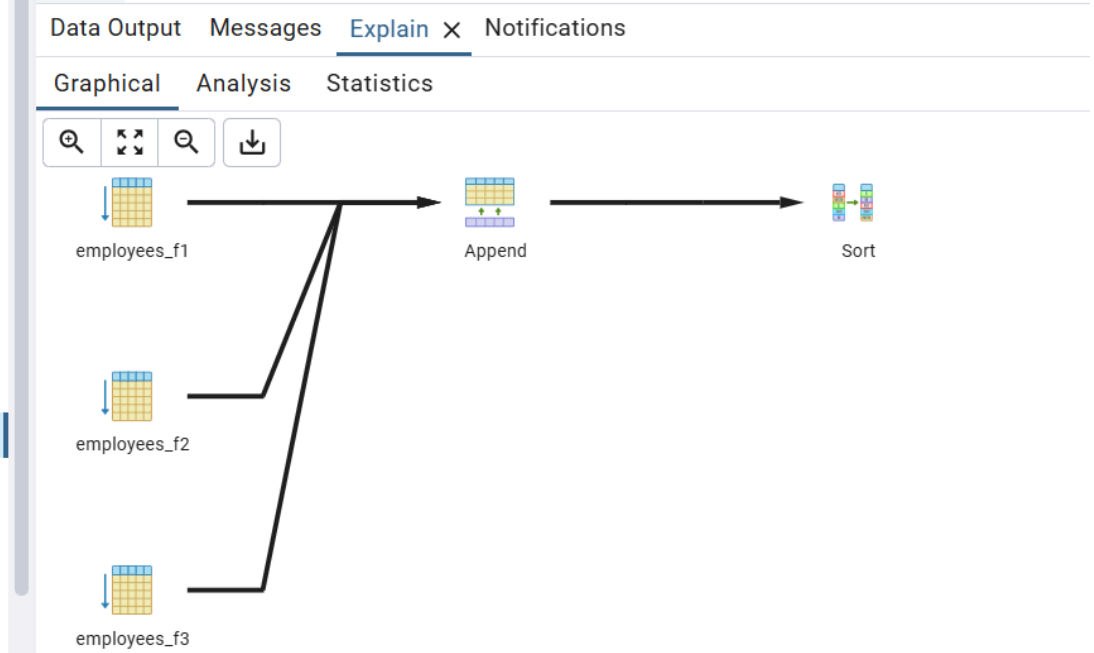


4. Carga de datos



Comprobación

```
42
43  SELECT * FROM employees1 ORDER BY dept_no;
44
45
```



5. Análisis de resultados

Query1:

--67.276ms
explain analyze
select * from employees where dept_no = 'd002'

-- 25.179ms
explain analyze
select * from employees1 where dept_no = 'd002'

Query2:

--93.406ms
explain analyze
select * from employees where dept_no = 'd003' or dept_no = 'd007'

-- 37.289 ms
explain analyze
select * from employees1 where dept_no = 'd003' or dept_no = 'd007'

Query3:

--80.238 ms
explain analyze
select * from employees where dept_no = 'd001' or dept_no = 'd005'

-- 71.033 ms
explain analyze
select * from employees1 where dept_no = 'd001' or dept_no = 'd005'

Tabla:

Consulta	Sin fragmentación	Con fragmentación	Mejora (%)
Query 1	67.276	25.179	62.573578
Query 2	93.406	37.289	60.078582
Query 3	80.238	71.033	11.47212

Si se usó
SET enable_partition_pruning = on;

P2

1. Creación de la tabla employees2

```
74
75  CREATE TABLE employees2 (
76   emp_no int,
77   birth_date date,
78   first_name varchar(14),
79   last_name varchar(16),
80   gender character(1),
81   hire_date date,
82   dept_no varchar(5),
83   from_date date
84  ) PARTITION BY RANGE (date_part('year', hire_date));
```

Data Output | Messages | Explain | Notifications

```
CREATE TABLE

Query returned successfully in 84 msec.
```

2. Analisis de distribución por año

```
88  select date_part('year', hire_date),  count(*) as total
89  from employees
90  group by date_part('year', hire_date) order by total desc;
```

Data Output | Messages | Explain | Notifications

Showing rows

	date_part double precision	total bigint
1	1986	40005
2	1985	39080
3	1987	36930
4	1988	34705
5	1989	31348
6	1990	28328
7	1991	24934
8	1992	22539
9	1993	19667
10	1994	16463
11	1995	13413
12	1996	10568
13	1997	7375
14	1998	4562

Ahora para saber los rangos de los fragmentos tendremos un objetivo por grupo de 108,894 (resultado de sumar todo y dividir entre 3)

Realizar esto nos da:

1985	39080	1		116015
1986	40005	1		
1987	36930	1		
1988	34705	2		119315
1989	31348	2		
1990	28328	2		
1991	24934	2		96273
1992	22539	3		
1993	19667	3		
1994	16463	3		
1995	13413	3		
1996	10568	3		
1997	7375	3		
1998	4562	3		
1999	1671	3		
2000	15	3		

F1 = 1985-1987
F2 = 1988-1991
F3 = 1992-2000

3. Creación de fragmentos

```
93
94   -- Fragmento 1: 1985-1987
95   CREATE TABLE employees2_f1 PARTITION OF employees2
96       FOR VALUES FROM (1985) TO (1988);
97
98   -- Fragmento 2: 1988-1991
99   CREATE TABLE employees2_f2 PARTITION OF employees2
100      FOR VALUES FROM (1988) TO (1992);
101
102  -- Fragmento 3: 1992-2000
103  CREATE TABLE employees2_f3 PARTITION OF employees2
104      FOR VALUES FROM (1992) TO (2001);
105
106
```

Data Output | Messages | Explain | Notifications

```
CREATE TABLE

Query returned successfully in 85 msec.
```

4. Carga de datos

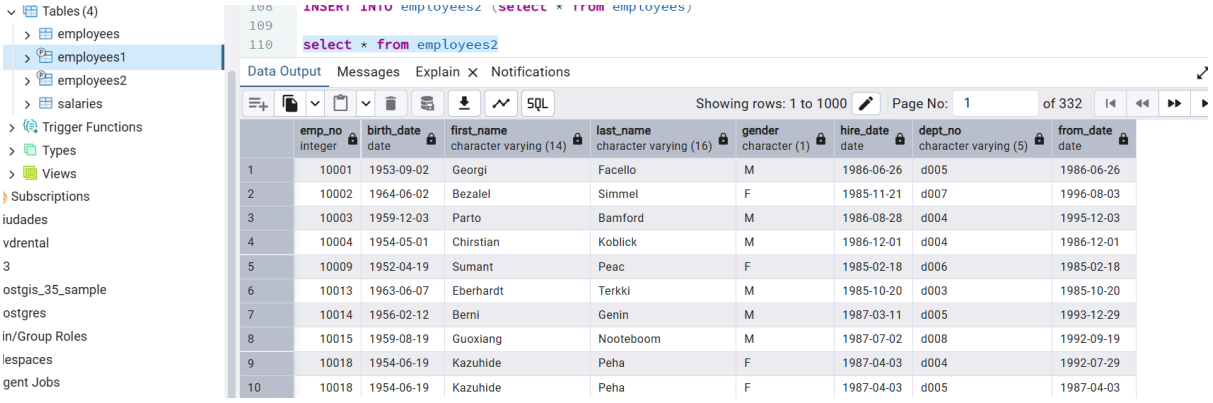
```
106
107    --Insercion de la forma sin csv
108    INSERT INTO employees2 (select * from employees)
109
110
```

Data Output | Messages | Explain | Notifications

```
INSERT 0 331603

Query returned successfully in 1 secs 585 msec
```

Comprobación



	emp_no integer	birth_date date	first_name character varying (14)	last_name character varying (16)	gender character (1)	hire_date date	dept_no character varying (5)	from_date date
1	10001	1953-09-02	Georgi	Facello	M	1986-06-26	d005	1986-06-26
2	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21	d007	1996-08-03
3	10003	1959-12-03	Parto	Bamford	M	1986-08-28	d004	1995-12-03
4	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01	d004	1986-12-01
5	10009	1952-04-19	Sumant	Peac	F	1985-02-18	d006	1985-02-18
6	10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20	d003	1985-10-20
7	10014	1956-02-12	Berni	Genin	M	1987-03-11	d005	1993-12-29
8	10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02	d008	1992-09-19
9	10018	1954-06-19	Kazuhide	Peha	F	1987-04-03	d004	1992-07-29
10	10018	1954-06-19	Kazuhide	Peha	F	1987-04-03	d005	1987-04-03

5. Analisis de Resultados

Query1:

Execution Time: 74.170 ms
explain analyze
select * from employees where date_part('year', hire_date) BETWEEN 1985 AND 1987;

Execution Time: 46.798 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1985 AND 1987;

Query2:

Execution Time: 71.488 ms
explain analyze

select * from employees where date_part('year', hire_date) BETWEEN 1988 AND 1991;

Execution Time: 47.181 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1988 AND 1991;

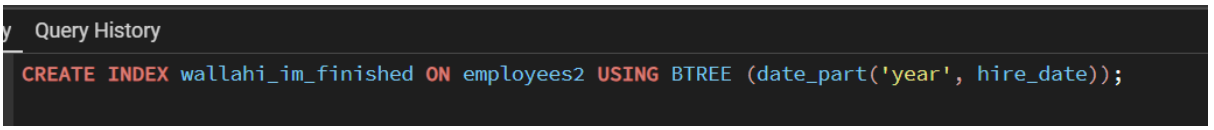
Query3:

Execution Time: 67.396 ms
explain analyze
select * from employees where date_part('year', hire_date) BETWEEN 1992 AND 2001;

Execution Time: 18.599 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1992 AND 2001;

Consulta	Sin fragmentacion	Con fragmentacion	Mejora (%)
Query 1	74.170 ms	46.798 ms	36,904%
Query 2	71.488 ms	47.181 ms	34,001%
Query 3	67.396 ms	18.599 ms	72,403%

6.



Query1:

Execution Time: 21.005 ms
explain analyze
select * from employees where date_part('year', hire_date) BETWEEN 1985 AND 1987;

Execution Time: 12.406 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1985 AND 1987;

Query2:

Execution Time: 21.152 ms
explain analyze
select * from employees where date_part('year', hire_date) BETWEEN 1988 AND 1991;

Execution Time: 13.531 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1988 AND 1991;

Query3:

Execution Time: 22.416 ms
explain analyze
select * from employees where date_part('year', hire_date) BETWEEN 1992 AND 2001;

Execution Time: 14.868 ms
explain analyze
select * from employees2 where date_part('year', hire_date) BETWEEN 1992 AND 2001;

Consulta	Sin fragmentacion	Con fragmentacion	Mejora (%)

Query 1	21.005 ms	12.406 ms	40,937%
Query 2	21.152 ms	13.531 ms	36,029%
Query 3	22.416 ms	14.868 ms	33,672%

Con la creacion de los indices en date_part('year', hire_date) , nos permitio utilizar las mismas consultas. Utilizaron un plan de ejecucion BitMap Heap Scan. Para el indice en employees, se escaneo toda la tabla, mientras que la tabla de employees2, solo realizo el scaneo en el fragmento correspondiente.

P3.
Usando la siguiente consulta:

```
WITH SalaryTertiles AS (
  SELECT
    latest_salary,
    NTILE(3) OVER (ORDER BY latest_salary) AS tertile_group
  FROM
    employees_with_salary
)
SELECT
  tertile_group,
  COUNT(*) AS frequency,
  MIN(latest_salary) AS min_salary_in_group,
  MAX(latest_salary) AS max_salary_in_group
FROM
  SalaryTertiles
GROUP BY
  tertile_group
ORDER BY
  tertile_group;
```

conseguimos los rangos de salarios para los 3 fragmentos de la tabla employees3.

tertile_group integer	frequency bigint	min_salary_in_group integer	max_salary_in_group integer
1	110535	38623	60501
2	110534	60502	75722
3	110534	75722	158220

```sql
CREATE MATERIALIZED VIEW employees_with_salary AS
SELECT
    e.emp_no,
    e.birth_date,
    e.first_name,
    e.last_name,
    e.gender,
    e.hire_date,
    e.dept_no,
    e.from_date,
    s.salary AS latest_salary
FROM employees2 e
JOIN (
    SELECT DISTINCT ON (emp_no)
        emp_no,
        salary,
        to_date
    FROM salaries
    ORDER BY emp_no, to_date DESC
) s
ON e.emp_no = s.emp_no;
```

Creamos esta vista materializada para unir los salarios con la tabla de employees, esta luego sera insertada en nuestra tabla employees3 tras generar la participación compuesta.

```sql
CREATE TABLE employees3 (
    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,
    latest_salary int
)
PARTITION BY RANGE (date_part('year', hire_date), latest_salary);

CREATE TABLE employees3_f1
  PARTITION OF employees3
  FOR VALUES FROM (1985, 38623) TO (1988, 60501);

CREATE TABLE employees3_f2
  PARTITION OF employees3
  FOR VALUES FROM (1988, 60502) TO (1992, 75722);

CREATE TABLE employees3_f3
  PARTITION OF employees3
  FOR VALUES FROM (1992, 75722) TO (2001, 158220);

INSERT INTO employees3
SELECT * FROM employees_with_salary;
```

Query1:

```
EXPLAIN ANALYZE
SELECT *
FROM employees3
WHERE date_part('year', hire_date) = 1986
  AND latest_salary BETWEEN 40000 AND 50000;
```

No fragmentado: Execution Time: 50.300 ms
Fragmentado: Execution Time: 41.415 ms

Query2:

```
EXPLAIN ANALYZE
SELECT *
FROM employees3
WHERE date_part('year', hire_date) = 1988
  AND latest_salary > 65000;
```

No Fragmentado: Execution Time: 56.632 ms
Fragmentado: Execution Time: 15.066 ms

Query3:

```
EXPLAIN ANALYZE
SELECT *
FROM employees3
WHERE date_part('year', hire_date) = 1988
  AND latest_salary > 65000;
```

No Fragmentado: Execution Time: 50.554 ms
Fragmentado: Execution Time: 5.057 ms

Consulta	Sin fragmentacion	Con fragmentacion	Mejora (%)
Query 1	50.300 ms	41.415 ms	17,664%
Query 2	56.632 ms	15.066 ms	73,396%
Query 3	50.554 ms	5.057 ms	89,996%

¿Qué estrategia de fragmentación resultó más efectiva?
Para las queries dada, resulta mas efectiva la partition compuesta ya que siempre estamos utilizando el hire_date y el salary para filtrar. Sin embargo, para consultas que solo utilizen el hire_data o solo el salary, la particion multinivel/anidada resultara ser mejor.

¿Cómo se comportan las consultas que acceden a múltiples fragmentos?
Para las consultas que utilizan multiples fragmentos, se realiza un parallel seq scan en cada una, lo cual es el mismo metodo de busqueda que se hace a la tabla sin particion. Pero en el partition se realiza un parallel append para concatenar los datos encontrados en cada particion.