“Diseño Físico”

Juan Carlos Gloria, Mauricio Diaz
Diseño Físico Para la Iteración 3
Universidad de los Andes, Bogotá, Colombia
{jc.gloria, m.diazt}@uniandes.edu.co
Fecha de presentación: Mayo 20 de 2018

Tabla de contenido

1 Selección de índices para requerimientos funcionales de consulta

1.1 RFC 10

RFC10: Para el requerimiento funcional de consulta 10, se creará un índice B+ de las fechas de inicio de las reservas. Esto es debido a que los índices B+ son muy efectivos cuando se necesita revisar si un valor está en cierto rango. En un árbol B+, se puede encontrar cualquier valor con una complejidad de log(n), mientras que si fuera hash podría resultar en una búsqueda completa de la tabla. Por otro lado, se van a buscar reservas con un cierto id de oferta, esto implica que se necesitara encontrar las tuplas que tienen como id de oferta un valor exacto. Para este tipo de escenario, se usará un índice de hash ya que este es capaz de encontrar tuplas con un valor exacto mucho más eficiente que un árbol B+. Mientras que un árbol B+ requiere una operación de entrada/salida (I/O) por cada bloque que debe recorrer del árbol, la función de hash solo tendrá que hacer una operación de I/O por cada tupla que encuentra. Esto último es asumiendo que las tuplas que encuentre estarán en diferentes bloques (peor escenario). Hace falta mencionar los problemas que tienen los índices de hash al agregar tuplas, ya que llegaría el caso de volver a calcular la función de hash y el costo de esto es alto si se tratan de miles de datos. Como se está creando el índice de hash sobre las ofertas, no es significante la preocupación de usar función de hash ya que las inserciones de ofertas no son tan comunes en comparación con las de reservas, donde se usa árbol B+ y este si se adapta bien a inserciones.

1.2 RFC 11

RFC11: Para el requerimiento funcional de consulta 11, se está consultando el también un rango de tuplas entre dos fechas como en el requerimiento anterior, se usará el mismo índice

con árbol b+ de las fechas de las reservas. Como no se está buscando igualdad sino desigualdad en este caso, el índice de hash no servirá de mucho ya que la función de hash nos ayudará exclusivamente a encontrar igualdad. En este caso se requerirá escanear todos los datos de la tabla en este rango así que no se usará el índice de hash agregado anteriormente para el requerimiento funcional de consulta 10.

1.3 RFC 12

Para el requerimiento funcional de consulta 12, se tendrá un índice compuesto secundario de las reservas y su id de oferta, y su fecha de reserva, esto tomará mucho espacio, pero ayudara a contar de forma rápida cuántas reservas se tienen de cada oferta en cierto rango. Las tuplas de reservas se estarán reduciendo de 9 columnas a 3. Es necesario recorrer la mayoría de las tuplas debido que se están contando.

1.4 RFC 13

En este requerimiento se usaron los índices de las llaves primarias de las tablas que brinda Oracle. Esto es debido a que gran complejidad de la consulta se usa un NOT IN.

2 Índices automáticos de Oracle

Consulta: `SELECT INDEX_NAME, TABLE_NAME FROM ALL_INDEXES WHERE OWNER = 'ISIS2304A091810';`

	INDEX_NAME	TABLE_NAME
1	SYS_C00376710	RESERVAS
2	SYS_C00376702	SERVICIOS
3	SYS_C00376699	OFERTAS_APARTAMENTO
4	SYS_C00376695	OFERTAS_VIVIENDA
5	SYS_C00376688	OFERTAS_EVENTO
6	SYS_C00376681	OFERTAS_PERSONANATURAL
7	SYS_C00376678	OFERTAS_HOSTAL
8	SYS_C00376673	OFERTAS_HOTEL
9	SYS_C00376667	OPERADORES_VIVIENDA
10	SYS_C00376666	EVENTOS
11	SYS_C00376664	OPERADORES_PERSONANATURAL
12	SYS_C00376661	OPERADORES_HOSTAL
13	SYS_C00376656	OPERADORES_HOTEL
14	SYS_C00376654	PERSONAS

Estos son los índices creados automáticamente por Oracle. Se tiene un índice para cada tabla que tiene como llave primaria una sola columna (comúnmente es una columna de id). Se puede ver que la tabla de gustan, no tiene índice ya que no posee una columna que identifique cada tupla de forma única. Para esto se requeriría hacer un índice que incluya toda la información de la tupla, lo cual perdería el propósito del índice.

3 Documentación de Requerimientos

3.1 RFC10

Consulta:

```
SELECT PERSONA,NOMBRE,CORREO, AFILIACION
FROM RESERVAS INNER JOIN PERSONAS ON RESERVAS.PERSONA = PERSONAS.ID WHERE ID_OFERTA = 235090
AND (FECHA_INICIO BETWEEN TO_DATE('01/01/19', 'DD/MM/YY') AND TO_DATE('01/01/20', 'DD/MM/YY'));
```

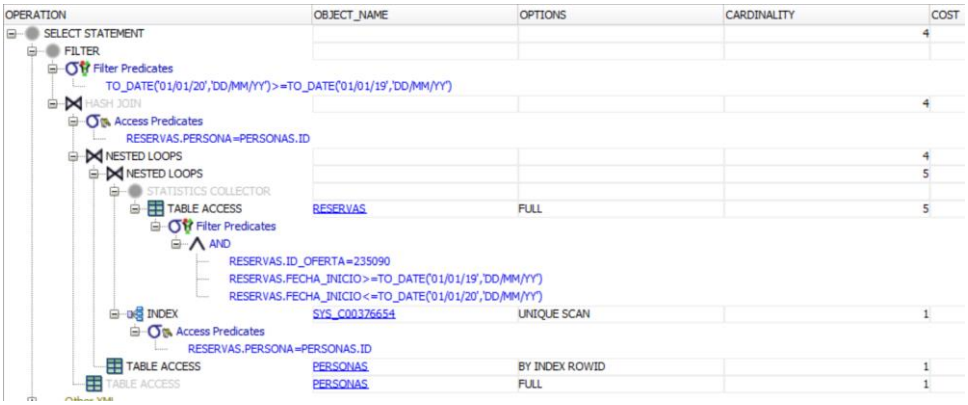
Distribución de Resultados: Debido a que el número de reservas de una oferta en cualquier rango de fecha es muy parecido en todas las ofertas (entre 1 y 5 ofertas), los resultados siempre tienen este rango de filas. Si se pone un rango de fecha muy ajustado es muy probable que no se obtengan resultados. En la siguiente imagen se puede ver un ejemplo de 11 ofertas y su número de reservas, este es el caso para todas las ofertas de la base de datos.

	COUNT(ID_OFERTA)	ID_OFERTA
121	1	5912
122	2	232437
123	1	122890
124	1	194827
125	3	82175
126	1	93400
127	2	416069
128	3	269767
129	4	235090
130	2	304730
131	2	45127
132	1	156958
133	1	86204

Parámetros Utilizados: Id de la oferta, fecha de inicio, fecha final.

El id de la oferta especifica que oferta se quiere consultar, la fecha de inicio y final indican el rango que se quiere buscar.

Plan de Ejecución de Oracle:



Tiempo de resultado promedio: 0,033 segundos

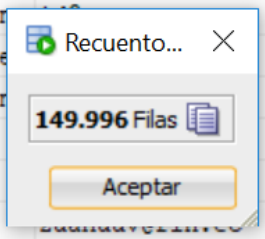
3.2 RFC11

Consulta:

```
SELECT * FROM PERSONAS WHERE ID NOT IN(
SELECT PERSONA
FROM RESERVAS INNER JOIN PERSONAS ON RESERVAS.PERSONA = PERSONAS.ID
WHERE ID_OFERTA = 235090
AND (FECHA_INICIO BETWEEN TO_DATE('01/01/19', 'DD/MM/YY') AND
TO_DATE('01/01/20', 'DD/MM/YY')));
```

Distribución de resultados: Estos resultados serán opuestos al caso del requerimiento 10, donde solo se encontraban entre 1 y 5 personas, aquí se encontrarán todos otros los miles de personas que no cumplen con una reserva de una oferta.

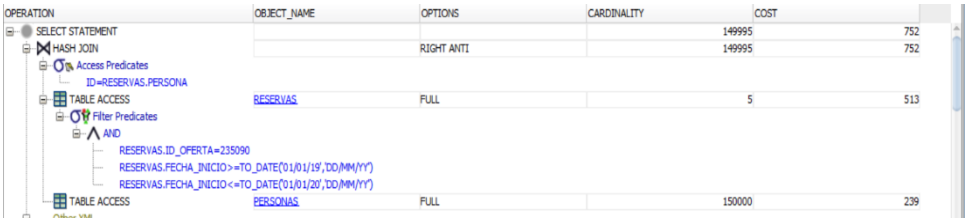
	ID	NOMBRE	CORREO	AFILIACION
1	187	Charlotte	hipre@emerib.co.uk	empleado
2	188	Christin		profesor
3	189	Theodore		empleado
4	190	Catherin		egresado
5	191	Jean		profesor
6	192	Pearl		profesor
7	193	Harold		estudiante
8	194	Dean	wauwi@izjodus.sz	profesor
9	195	Hilda	metja@vas.co	profesor
10	196	Theresa	ej@fozi.ee	profesor
11	197	Lenora	gav@ni.kz	egresado
12	198	Anthony	kaj@jute.qa	profesor
13	199	Daniel	hiwopok@uwmaczac.tv	profesor



Parámetros utilizados: id de la oferta, fecha de inicio, fecha final.

El id de la oferta especifica que oferta se quiere consultar, la fecha de inicio y final indican el rango que se quiere buscar.

Plan de Ejecución de Oracle:



Tiempo de resultado promedio: 0,033 segundos

3.3 RFC12

- **Oferta con más ocupación:**

Consulta:

```
SELECT SEMANA, DEMANDA, ID_OFERTA
FROM (SELECT t.*, row_number() OVER (PARTITION BY SEMANA ORDER BY DEMANDA DESC) as seqnum
        FROM         (SELECT         to_number(to_char(to_date(FECHA_INICIO,'DD/MM/YYYY'),'WW'))         AS SEMANA,COUNT(ID_OFERTA) AS DEMANDA,ID_OFERTA
                FROM RESERVAS
                GROUP BY ID_OFERTA,to_number(to_char(to_date(FECHA_INICIO,'DD/MM/YYYY'),'WW'))
                ORDER BY DEMANDA desc) t
     ) t
WHERE seqnum = 1;
```

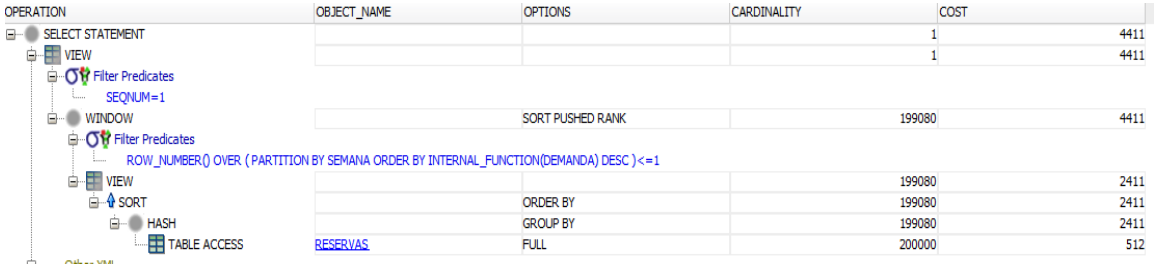
Distribución de resultados:

Debido a que una reserva de una oferta dura casi siempre más de una semana, la oferta con más demanda pocas veces va a tener más de 2 demandas en una semana y también van a haber varias con el mismo número de demandas.

	SEMANA	DEMANDA	ID_OFERTA
1	1	2	381937
2	2	2	338082
3	3	2	438899
4	4	2	174292
5	5	2	291207
6	6	2	12902
7	7	2	236818
8	8	2	177811
9	9	2	257385
10	10	2	109690
11	11	2	380594
12	12	2	190327
13	13	2	245711

Parámetros utilizados: El año que se quiere consultar, pero no es tan necesario ya que todos los datos ingresados fueron en un mismo año (2019).

Plan ejecución Oracle:



Tiempo de resultado promedio: 0.4 segundos

- **Oferta con menos ocupación:**

Consulta:

```
SELECT SEMANA, DEMANDA, ID_OFERTA
FROM (SELECT t.*, row_number() OVER (PARTITION BY SEMANA ORDER BY DEMANDA asc) as seqnum
    FROM (SELECT to_number(to_char(to_date(FECHA_INICIO,'DD/MM/YYYY'),'WW')) AS SEMANA,COUNT(ID_OFERTA) AS DEMANDA,ID_OFERTA
        FROM RESERVAS
        GROUP BY ID_OFERTA,to_number(to_char(to_date(FECHA_INICIO,'DD/MM/YYYY'),'WW'))
        ORDER BY DEMANDA asc) t
  ) t
WHERE seqnum = 1;
```

Consulta(alterna):

```
SELECT *
FROM OFERTAS
WHERE ID_OFERTA NOT IN (SELECT ID_OFERTA FROM RESERVAS);
```

Distribución de resultados:

Es lo mismo que el anterior punto, pero en orden ascendente.

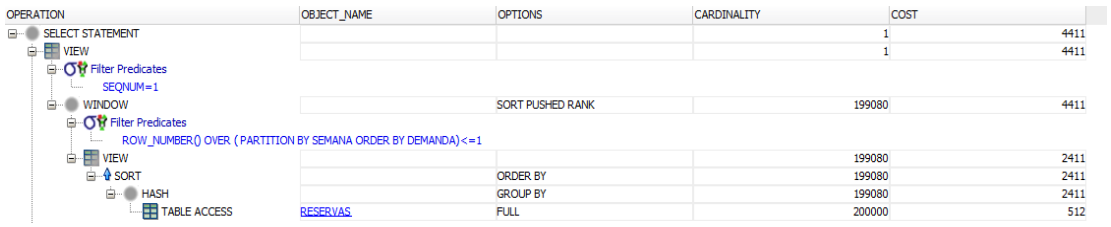
	SE...	DEMANDA	ID_OFERTA
1	1	1	406577
2	2	1	423547
3	3	1	357086
4	4	1	94569
5	5	1	62184
6	6	1	277671
7	7	1	433529
8	8	1	390165
9	9	1	44469
10	10	1	315390
11	11	1	209464
12	12	1	211371
13	13	1	6523

Alterna: Debido a que hay demasiadas ofertas, van a haber bastantes sin reservas en una semana por lo que con solo mirar que el ID de la oferta no se encuentre en la tabla de reservas en esa semana obtenemos la oferta con menos ocupación (o una de las que no tiene) para hacer el requerimiento.

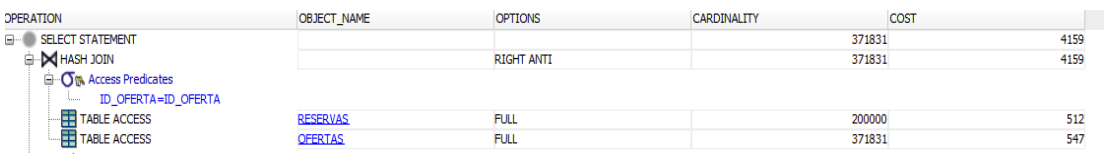
	ID_OFERTA	TIPO_OFERTA	ID_OPERADOR
1	798	OFERTAS_PERSONANATURAL	18813
2	801	OFERTAS_PERSONANATURAL	1236
3	803	OFERTAS_PERSONANATURAL	11408
4	804	OFERTAS_PERSONANATURAL	2656
5	805	OFERTAS_PERSONANATURAL	8358
6	806	OFERTAS_PERSONANATURAL	10098
7	807	OFERTAS_PERSONANATURAL	17830
8	809	OFERTAS_PERSONANATURAL	19321
9	810	OFERTAS_PERSONANATURAL	15131
10	811	OFERTAS_PERSONANATURAL	18706
11	814	OFERTAS_PERSONANATURAL	2721
12	815	OFERTAS_PERSONANATURAL	7625

Parámetros utilizados: El año que se quiere consultar, pero no es tan necesario ya que todos los datos ingresados fueron en un mismo año (2019).

Plan ejecución Oracle:



Plan ejecución Oracle (Alterno):



Tiempo de resultado promedio: 0.38 segundos.

Tiempo de resultado promedio (Alterno): 0.04 segundos

- **Operadores más solicitados:**

Consulta:

```
SELECT SEMANA, DEMANDA, ID_OPERADOR
FROM (SELECT t.*, row_number() OVER (PARTITION BY SEMANA ORDER BY DEMANDA DESC) as seqnum
      FROM (SELECT to_number(to_char(to_date(r1.FECHA_INICIO,'DD/MM/YYYY'),'WW')) AS SEMANA,COUNT(o1.ID_OPERADOR) AS DEMANDA,o1.ID_OPERADOR
            FROM(OFERTAS o1 INNER JOIN RESERVAS r1
            ON o1.ID_OFERTA = r1.ID_OFERTA)
            GROUP BY to_number(to_char(to_date(r1.FECHA_INICIO,'DD/MM/YYYY'),'WW')), o1.ID_OPERADOR
            ORDER BY DEMANDA desc)t)t
WHERE seqnum = 1;
```

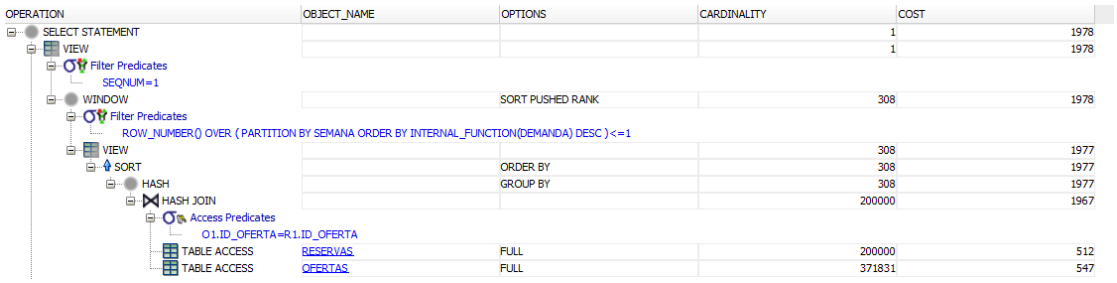
Distribución de resultados:

El resultado arrojado fue bastante prometedor ya que hubo hasta 5 reservas por Operador en una semana.

	SEMANA	DEMANDA	ID_OPERADOR
1	1	3	5744
2	2	4	9497
3	3	4	6353
4	4	3	14988
5	5	3	19700
6	6	4	13654
7	7	4	19036
8	8	4	11758
9	9	4	15000
10	10	4	14776
11	11	3	12776
12	12	4	1031
13	13	4	14519

Parámetros utilizados: ninguno

Plan ejecución Oracle:



Tiempo de resultado promedio: 0.513 segundos

- **Operadores menos solicitados:**

Consulta:

```
SELECT SEMANA, DEMANDA, ID_OPERADOR
FROM (SELECT t.*, row_number() OVER (PARTITION BY SEMANA ORDER BY DEMANDA ASC) as seqnum
    FROM (SELECT to_number(to_char(to_date(r1.FECHA_INICIO,'DD/MM/YYYY'),'WW')) AS SEMANA,COUNT(o1.ID_OPERADOR) AS DEMANDA,o1.ID_OPERADOR
            FROM(OFERTAS o1 INNER JOIN RESERVAS r1
            ON o1.ID_OFERTA = r1.ID_OFERTA)
            GROUP BY to_number(to_char(to_date(r1.FECHA_INICIO,'DD/MM/YYYY'),'WW')), o1.ID_OPERADOR
           ORDER BY DEMANDA ASC)t)t
WHERE seqnum = 1;
```

Consulta (Alterna):

```
SELECT DISTINCT ID_OPERADOR
FROM OFERTAS
WHERE ID_OPERADOR NOT IN(SELECT o1.ID_OPERADOR
            FROM(OFERTAS o1 INNER JOIN RESERVAS r1
              ON o1.ID_OFERTA = r1.ID_OFERTA))
    ORDER BY ID_OPERADOR;
```

Distribución de resultados

Es lo mismo que el anterior punto, pero en orden ascendente.

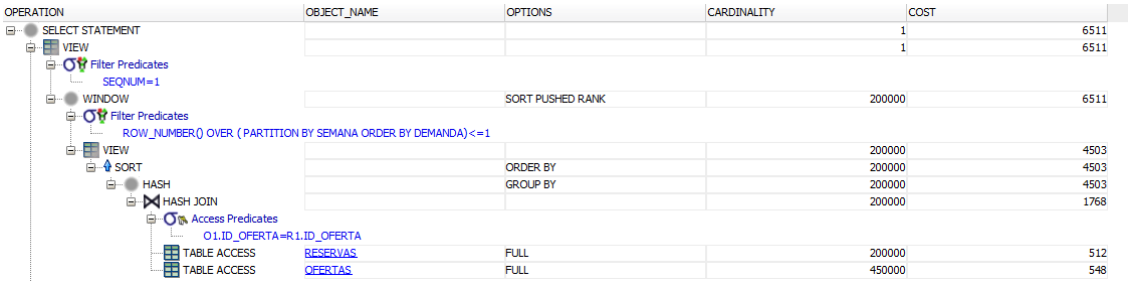
	SEMANA	DEMANDA	ID_OPERADOR
1	1	1	5267
2	2	1	15481
3	3	1	13939
4	4	1	5611
5	5	1	3800
6	6	1	6979
7	7	1	7424
8	8	1	4634
9	9	1	12599
10	10	1	16636
11	11	1	97
12	12	1	19160
13	13	1	8016

Alterna: Debido a que hay demasiadas ofertas y operadores, van a haber bastantes sin reservas en una semana, por lo que con solo mirar que el ID del operador no se encuentre en la tabla de reservas en esa semana obtenemos la oferta con menos ocupación (o una de las que no tiene) para hacer el requerimiento.

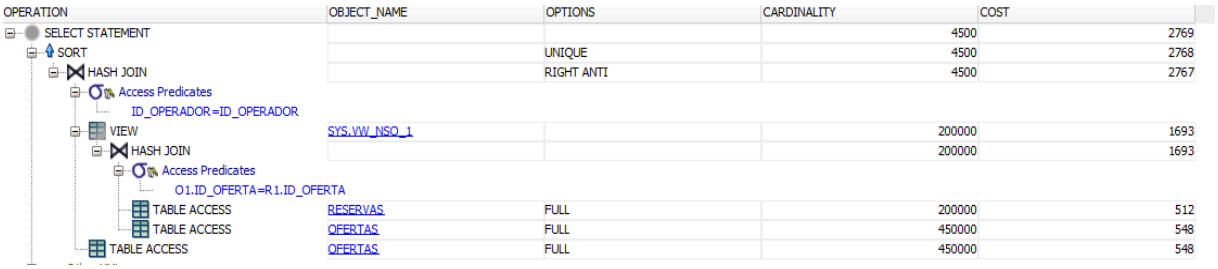
	ID_OPER...
1	1667
2	2090
3	2208
4	3580
5	3734
6	4227
7	4341
8	4392
9	5342
10	7304
11	7898
12	8023
13	8168
14	8232

Parámetros utilizados: Ninguno

Plan ejecución Oracle:



Plan ejecución Oracle (Alterno):



Tiempo de resultado promedio: 0.469 segundos

Tiempo de resultado promedio (Alterno): 0.3 segundos

3.4 RFC13

Consulta:

```
SELECT PERSONAS.ID, PERSONAS.NOMBRE, PERSONAS.CORREO, PERSONAS.AFILIACION, RESERVAS.COSTO_CALCULADO
FROM (PERSONAS INNER JOIN RESERVAS
    ON PERSONAS.ID = RESERVAS.PERSONA)
WHERE PERSONAS.ID NOT IN(SELECT PERSONA
                FROM RESERVAS
                WHERE COSTO_CALCULADO < 500000
```

```
        GROUP BY PERSONA)
ORDER BY PERSONAS.ID;
```

Distribución de resultados:

Debido a que los precios que se generaron estaban bastante altos, van a haber bastantes tuplas resultantes, pero aun así pueden haber personas que no siempre pagaron más de 150 dólares (en este caso se colocó 500000 pesos) por lo que se hizo la parte del NOT IN.

	ID	NOMBRE	CORREO	AFILIACION	COSTO_CALCULADO
1	2	Blake	vadrepnim@tudoda.mq	padre	8680544
2	3	Evelyn	feuvhaw@vu.sk	profesor	8460496
3	5	Landon	pon@anewuh.md	profesor	3322859
4	5	Landon	pon@anewuh.md	profesor	9661959
5	6	Carrie	inu@fiwifeju.tf	padre	6210010
6	7	Stella	fijem@de.ar	padre	8877473
7	9	Austin	measa@dincipoj.sg	empleado	7439461
8	10	Julia	weumfig@ticeju.co.uk	padre	3526013
9	10	Julia	weumfig@ticeju.co.uk	padre	2044810
10	10	Julia	weumfig@ticeju.co.uk	padre	2890943
11	10	Julia	weumfig@ticeju.co.uk	padre	4696144
12	11	Cora	tihoh@vis.ag	profesor	636267
13	11	Cora	tihoh@vis.ag	profesor	2893882
14	12	Arthur	pufam@ce.se	empleado	1668495
15	14	Teresa	iz@divgalho.jo	egresado	2728353
16	14	Teresa	iz@divgalho.jo	egresado	6542159
17	17	Ivan	tic@akede.sc	padre	7494594

Parámetros utilizados: ninguno

Plan ejecución Oracle:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			189502	3027
MERGE JOIN		ANTI	189502	3027
MERGE JOIN			200000	2512
TABLE ACCESS	PERSONAS	BY INDEX ROWID	150000	1122
INDEX	SYS_C00376067	FULL SCAN	150000	283
SORT		JOIN	200000	1390
Access Predicates				
PERSONAS.ID=RESERVAS.PERSONA				
Filter Predicates				
PERSONAS.ID=RESERVAS.PERSONA				
TABLE ACCESS	RESERVAS	FULL	200000	513
SORT		UNIQUE	7873	515
Access Predicates				
PERSONAS.ID=PERSONA				
Filter Predicates				
PERSONAS.ID=PERSONA				
VIEW	SYS.VW_NSO_1		7873	514
HASH		GROUP BY	7873	514
TABLE ACCESS	RESERVAS	FULL	8000	513
Filter Predicates				
COSTO_CALCULADO<500000				

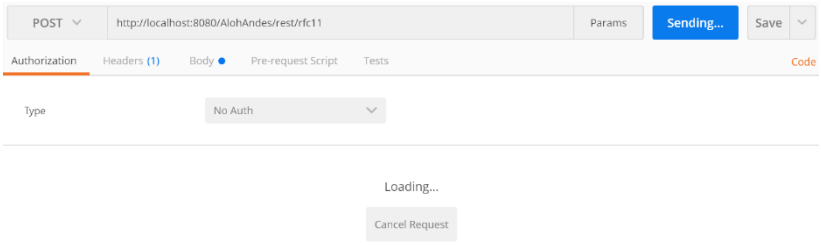
Tiempo de resultado promedio: 0.109 segundos.

4 Documentación de carga de datos

Para la carga de datos, se usó la página ConvertCSV para generar archivos CSV con datos para cada tabla. Esta página cuenta con una herramienta para generar datos de prueba. Los números de identificación de las tuplas para cada tabla de ofertas son de cierto rango, de esta forma se logra separar e identificar que tuplas pertenecen a qué tipo de oferta. Estos archivos CSV generados se importaron a Oracle SQL Developer, la cual tiene una herramienta para insertar tuplas a partir de archivos CSV.

La página para generar los datos se puede encontrar aquí: http://www.convertcsv.com/generate-test-data.htm

5 Análisis del proceso de optimización



Al realizar los joins con operaciones de java, como se muestra en la imagen de arriba, no logra terminar de recorrer todas las tuplas de las tablas. Después de dejarlo más de 10 minutos, no terminó el proceso. Esto se puede explicar debido a que java recorre todas las tablas del join tupla por tupla, requiriendo operaciones de I/O por cada tupla. Oracle por otro lado minimiza el número de accesos a bloques con la ayuda de los índices.

(3)(3) (4)