Iteración 4

Gabriela Vargas, Santiago Ramírez
202013830 - 201910908

Departamento de Ingeniería de sistemas y computación, Universidad de los Andes

28 de mayo de 2023

Diseño físico

Selección de índices

RFC10 CONSULTAR CONSUMO EN ALOHANDES

Para este requerimiento, se quería conocer la información de los usuarios que realizaron al menos una reserva de una determinada oferta de alojamiento en un rango de fechas. Para cumplir este requerimiento, se necesitaba llamar tres tablas que son alojamiento, reserva y usuario. En alojamiento, se podía ver que el índice que oracle creo fue el siguiente:

Alohandes | ALOHAMIENTO

Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	SYSTEM	ALOHAMIENTO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

En reserva, el índice creado fue el siguiente:

Alohandes | RESERVA

Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	SYSTEM	RESERVA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

Y, por último, en usuario, el índice creado fue el siguiente:

Alohandes | USUARIO

Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	SYSTEM	USUARIO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID

Con estos índices, la respuesta en segundos que nos tomaba realizar este requerimiento era de:

Salida de Script | Resultado de la Consulta

Tarea terminada en 0,189 segundos

```
        ID NOMBRE
---------- ------------------------------------------------------------------
         1 Juan Perez
         1 Juan Perez
         5 Luisa Fernandez


        ID NOMBRE
---------- ------------------------------------------------------------------
         1 Juan Perez
         1 Juan Perez
         5 Luisa Fernandez
```

Con los índices que nosotros creamos, la respuesta arrojaba el siguiente tiempo:

Salida de Script

Tarea terminada en 0,076 segundos

```
        ID NOMBRE
---------- ------------------------------------------------------------------
         1 Juan Perez
         1 Juan Perez
         5 Luisa Fernandez


        ID NOMBRE
---------- ------------------------------------------------------------------
         1 Juan Perez
         1 Juan Perez
         5 Luisa Fernandez
```

Al comparar los tiempos, los índices creados hicieron que se ahorrara en un 58 % el tiempo de este requerimiento. Los índices que se crearon (los cuales ayudaron para todos los requerimientos) fueron los siguientes:

Alohandes | RESERVA

Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	SYSTEM	RESERVA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID
2	SYSTEM	IDX_RESERVA_FECHA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	FECHA
3	SYSTEM	IDX_RESERVA_IDUSUARIO	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDUSUARIO
4	SYSTEM	IDX_RESERVA_IDALOHAMIENTO	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	IDALOHAMIENTO

Alohandes | USUARIO

Columnas | Datos | Model | Restricciones | Permisos | Estadísticas | Disparadores | Flashback | Dependencias | Detalles | Particiones | Índices | SQL

Acciones...

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1	SYSTEM	USUARIO_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID
2	SYSTEM	IDX_USUARIO_NOMBRE	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	NOMBRE

Como se puede ver en la imagen, se crearon con las siguientes sentencias en SQL:

```
CREATE INDEX idx_usuario_id ON USUARIO(id);
CREATE INDEX idx_reserva_idalohamiento ON RESERVA(idalohamiento);
CREATE INDEX idx_reserva_fecha ON RESERVA(fecha);
CREATE INDEX idx_reserva_idusuario ON RESERVA(idusuario);
```

- (idx_usuario_id) en la tabla USUARIO para la columna id: Este índice se crea en la columna id de la tabla USUARIO y facilita la búsqueda y recuperación rápida de usuarios por su ID. Es útil en consultas que involucran la tabla USUARIO y requieren búsquedas o uniones basadas en el ID del usuario.
- (idx_reserva_idalohamiento) en la tabla RESERVA para la columna idalohamiento: Este índice se crea en la columna idalohamiento de la tabla RESERVA. Ayuda a mejorar el rendimiento de consultas que involucran la tabla RESERVA y requieren búsquedas o uniones basadas en el ID del alojamiento.
- (idx_reserva_fecha) en la tabla RESERVA para la columna fecha: Este índice se crea en la columna fecha de la tabla RESERVA y mejora la eficiencia de las búsquedas basadas en fechas. Permite acelerar consultas que filtran o agrupan registros en función de rangos de fechas.
- (idx_reserva_idusuario) en la tabla RESERVA para la columna idusuario: Este índice se crea en la columna idusuario de la tabla RESERVA. Ayuda a mejorar el rendimiento de consultas que involucran la tabla RESERVA y requieren búsquedas o uniones basadas en el ID del usuario.

Además, todos estos índices serían tipo árbol balanceado (B-Tree) ya que mejoran la eficiencia de los requerimientos, son soportes para rangos de búsqueda (en este caso, rangos de fechas) y su buena capacidad en inserciones y eliminaciones.
La sentencia de SQL para este requerimiento es la siguiente:

```
SELECT u.*
FROM USUARIO u INNER JOIN RESERVA r ON u.id = r.idusuario
INNER JOIN ALOHAMIENTO a ON r.idalohamiento = a.id
WHERE r.fecha BETWEEN '01/01/2023' AND '01/02/2023'
ORDER BY u.nombre;
```

```
SELECT u.*
FROM USUARIO u INNER JOIN RESERVA r ON u.id = r.idusuario
INNER JOIN ALOHAMIENTO a ON r.idalohamiento = a.id
WHERE r.fecha BETWEEN '01/01/2023' AND '01/02/2023'
ORDER BY a.id;
```

RFC11 CONSULTAR CONSUMO EN ALOHANDES – RFC10-V2

Para este requerimiento, se quería conocer la información de los usuarios QUE NO realizaron al menos una reserva de una determinada oferta de alojamiento en un rango de fechas. Este requerimiento va de la mano con el RFC10

por lo que no se va a colocar todas las explicaciones realizadas anteriormente, solo se colocará los tiempos que nos arrojó. De acuerdo con lo anterior, el tiempo sin índices nos dio el siguiente:

```
Salida de Script
Tarea terminada en 0,123 segundos

        ID NOMBRE
---------- ------------------------------------------------------------------
         4 Ana Torres
         2 Maria Rodriguez
         3 Pedro Gomez


        ID NOMBRE
---------- ------------------------------------------------------------------
         2 Maria Rodriguez
         3 Pedro Gomez
         4 Ana Torres
```

Y con los índices fue de:

```
Salida de Script
Tarea terminada en 0,078 segundos

        ID NOMBRE
---------- ------------------------------------------------------------------
         4 Ana Torres
         2 Maria Rodriguez
         3 Pedro Gomez


        ID NOMBRE
---------- ------------------------------------------------------------------
         2 Maria Rodriguez
         3 Pedro Gomez
         4 Ana Torres
```

Al comparar los tiempos, los índices creados hicieron que se ahorrara en un 80 % el tiempo de este requerimiento. La sentencia de SQL para este requerimiento es la siguienta:

```
SELECT u.*
FROM USUARIO u
WHERE NOT EXISTS (SELECT 1 FROM RESERVA r
INNER JOIN ALOHAMIENTO a ON r.idalohamiento = a.id WHERE r.idusuario = u.id AND r.fecha
BETWEEN '01/01/2023' AND '01/02/2023')
ORDER BY u.nombre;
```

```
SELECT u.*
FROM USUARIO u
WHERE NOT EXISTS (SELECT 1 FROM RESERVA r
INNER JOIN ALOHAMIENTO a ON r.idalohamiento = a.id
WHERE r.idusuario = u.id AND r.fecha
BETWEEN '01/01/2023' AND '01/02/2023')
ORDER BY a.id;
```

RFC12 CONSULTAR FUNCIONAMIENTO

Para este requerimiento, se quería mostrar, para cada semana del año, la oferta de alojamiento con más ocupación, la oferta de alojamiento con menos ocupación, los operadores más solicitados y los operadores menos solicitados. El tiempo sin los índices fue el siguiente:

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 11 en 0,027 segundos

	SEMANA	ALOJAMIENTO_MAX_RESERVAS	ALOJAMIENTO_MIN_RESERVAS
1	02	3	3
2	05	1	1
3	06	1	1
4	08	4	4
5	09	3	3
6	11	5	5
7	16	3	3
8	18	3	3
9	19	2	2
10	22	3	3
11	52	1	1

Y con los índices fue de:

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 11 en 0,006 segundos

	SEMANA	ALOJAMIENTO_MAX_RESERVAS	ALOJAMIENTO_MIN_RESERVAS
1	02	3	3
2	05	1	1
3	06	1	1
4	08	4	4
5	09	3	3
6	11	5	5
7	16	3	3
8	18	3	3
9	19	2	2
10	22	3	3
11	52	1	1

Al comparar los tiempos, los índices creados hicieron que se ahorrara 74 % el tiempo de este requerimiento.

```
SELECT semana, MAX(alojamiento_reservas) AS alojamiento_max_reservas,
MIN(alojamiento_reservas) AS alojamiento_min_reservas
FROM (SELECT semana, idalohamiento AS alojamiento_reservas, ROW_NUMBER() OVER
(PARTITION BY semana ORDER BY contador_reservas DESC) AS rn_max, ROW_NUMBER()
OVER (PARTITION BY semana ORDER BY contador_reservas ASC)
AS rn_min FROM (SELECT TO_CHAR(r.fecha, 'IW')
AS semana, r.idalohamiento, COUNT(*)
AS contador_reservas FROM RESERVA r
GROUP BY TO_CHAR(r.fecha, 'IW'), r.idalohamiento) r)
WHERE rn_max = 1 OR rn_min = 1
GROUP BY semana;
```

RFC13 CONSULTAR LOS BUENOS CLIENTES

Para este requerimiento se quieren mostrar los buenos clientes son de tres tipos: aquellos que hacen reservas en AlohAndes al menos una vez al mes, aquellos que siempre reservan alojamientos costosos y aquellos que siempre reservan suites. El tiempo sin los índices fue el siguiente:

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0,101 segundos

	ID	NOMBRE	CEDULA	EDAD	CELULAR	VINCULACION
1	5	Luisa Fernandez	1357902468	28	3151357902	Empleado

Y con los índices fue de:

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 1 en 0,02 segundos

	ID	NOMBRE	CEDULA	EDAD	CELULAR	VINCULACION
1	5	Luisa Fernandez	1357902468	28	3151357902	Empleado

Al comparar los tiempos, los índices creados hicieron que se ahorrara 70 % el tiempo de este requerimiento.

```
WITH meses AS (SELECT ADD_MONTHS(TO_DATE('01/01/2023', 'DD/MM/YYYY'), LEVEL - 1)
AS fecha FROM DUAL CONNECT BY LEVEL <= MONTHS_BETWEEN(TRUNC(SYSDATE, 'MM'),
TO_DATE('01/01/2023', 'DD/MM/YYYY')) + 1)
SELECT u.*
FROM USUARIO u
WHERE (SELECT COUNT(DISTINCT EXTRACT(MONTH FROM r.fecha))
FROM meses m LEFT JOIN RESERVA r ON u.id = r.idusuario
AND TRUNC(r.fecha, 'MM') = TRUNC(m.fecha, 'MM')
WHERE r.idusuario IS NOT NULL) = (SELECT COUNT(*) FROM meses);
```

Escenarios de prueba

Para los escenarios de prueba, los realizamos en el momento de poblar las tablas, algunos ejemplos pueden ser los siguientes:

```
INSERT INTO PERSONANATURAL VALUES (1, 'Gabriela', '1000000', 17);
INSERT INTO PERSONANATURAL VALUES (2, 'Maria Gomez', '987654321', 30);
INSERT INTO PERSONANATURAL VALUES (3, 'Pedro Rodriguez', '456789123', 40);
INSERT INTO APARTAMENTO VALUES(1, 'Calle 1, Ciudad 1', 1000000, 900000, 120, 1, 'SI');
INSERT INTO APARTAMENTO VALUES(2, 'Calle 2, Ciudad 2', 750000, 0, 90, 2, 'NO');
INSERT INTO APARTAMENTO VALUES(3, 'Calle 3, Ciudad 3', 1200000, 1000000, 150, 3, 'SI');
INSERT INTO COMUNIDAD VALUES (1, 'Comunidad 1', 35);
INSERT INTO COMUNIDAD VALUES (2, 'Comunidad 2', 28);
INSERT INTO COMUNIDAD VALUES (3, 'Comunidad 3', 42);
```

Estos son algunos datos ya que poblamos nuestra base de datos con 1.000.005 datos. El script de sql que se subió al repositorio tiene algunos datos ya que la carga masiva de datos se generó con un archivo csv y se importo los datos por sqlLoader. El orden que se colocó en el script es como nos ingresaron los datos sin problema. A continuación, se puede ver que si ingresaron las filas con las que se intentó:

```
1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.

1 fila insertadas.
```

Referencias

[1] Sistrans (2023) *ITERACIÓN 4 DISEÑO FÍSICO Y OPTIMIZACIÓN DE CONSULTAS* Recuperado de https://bloqueneon.uniandes.edu.co//content/enforced/171153-202310$_ISIS2304_01/Proyecto/ST-Pry-It4.pdf$