

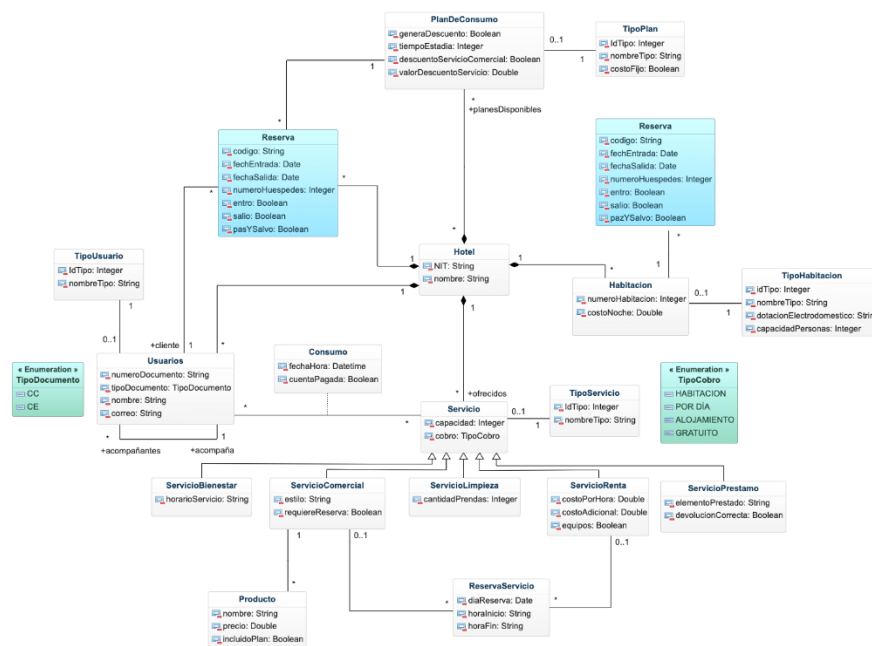
DOCUMENTACIÓN DEL PROYECTO – INFORME ENTREGA 2

ANÁLISIS

Cómo punto de partida del proyecto, se tomaron los dos diagramas establecidos para la primera entrega. Bajo estos diagramas, se implementó un proceso de normalización dadas las observaciones realizadas en la primera iteración del proyecto. Adicional a este proceso, se realizaron algunos cambios sobre el diagrama de clases UML independientes a la normalización. Estos dos procesos son descritos a continuación.

CAMBIOS PREVIOS

A continuación, se exhibe el diagrama de clases UML que estuvo vigente hasta el final de la primera entrega. Estos cambios responden parcialmente a los nuevos requerimientos, otros fueron implementados por cuestiones de simplificación.



Los cambios implementados, independientes a la normalización fueron:

- Se eliminaron las clases “Servicio limpieza”, “Servicio préstamo” y “Servicio bienestar”, esto se debe al hecho de que estas clases añadían un nivel de detalle que no era solicitado en los requerimientos del cliente, luego, no era necesario mantenerlas.
- Se suprimieron de la clase reserva los atributos “Entro”, “Salió” y “Paz y salvo”. En su lugar, se añadió el atributo “Estado” que corresponde a una enumeración que contempla cuatro posibles valores: “Por entrar”, “entro”, “Salió con deuda” y “Salió sin deuda”.
- Para evitar hacer uso del patrón fachada se eliminó la clase “hotel”, esto es particularmente razonable considerando que solo se está considerando un hotel (El hotel de los Andes).
- A la clase de asociación “consumo” se le añadió el atributo “Hora” y el atributo “Registrado por”, quien describe al empleado que registró el consumo.

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



- De la clase “Plan de consumo” se eliminó el atributo booleano que indicaba si un plan de descuento implementaba descuento en algún servicio comercial. Esto se justifica en la redundancia del atributo, pues esta información era proporcionada por el atributo “Valor descuento servicio”
- Para propósitos de la segunda entrega, se añadió el atributo “precio” a servicio.
- De la clase “Servicio renta” se eliminó el atributo booleano que indicaba si la zona rentada incluía en su precio el alquiler de equipos de sonido. Esto se justifica en la redundancia del atributo, pues esta información era proporcionada por el atributo “Costo adicional” (Si los equipos están incluidos, no hay costo adicional.
- Para facilitar la comunicación entre relaciones, se eliminó de la clase producto el atributo booleano “Incluido en plan”, en su lugar, se le añadió a la clase “Plan de consumo” una lista de productos.

NORMALIZACIÓN

En este apartado se incluirá tanto la descripción de las normalizaciones realizadas como ciertas aclaraciones en algunas relaciones para justificar el que no se hayan normalizado. Para el proceso de normalización, se tomaron en cuenta las siguientes representaciones entidad relación. Las demás relaciones no contaban con dependencias funcionales complejas, por lo que se omite su mención en este documento.

Reservas_servicio

id_reserva_servicio	día_reserva	hora_inicio	hora_fin	servicio_comercial	servicio_renta	cliente
NN, PK	NN	NN	NN	FK servicios_comerciales.tipo	FK servicios_renta.tipo	NN, FK usuarios.num_documento

Dependencias funcionales

Id reserva servicio

→ *Día reserva, hora inicio, hora fin, servicio comercial, servicio renta, cliente*

Para esta primera relación, se hace la aclaración de que la hora inicio y la hora fin no se determinan funcionalmente entre ellas, esto se justifica en el supuesto de que cada reserva de servicio tiene una duración arbitraria. Al establecer esta información, se evita una dependencia transitiva o problemas con la forma normal Boyce-codd.

Tipos_habitacion

id_tipo	nombre	dotacion	capacidad
NN, PK	NN	NN	NN

Dependencias funcionales

Id tipo → *nombre, dotacion, capacidad*

dotacion → *capacidad*

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Siendo “Id tipo” la única llave candidata de la relación, se encuentra dependencia a un atributo no primo por parte de otro no primo. Esta dependencia era, en general, confusa, por lo que se decidió implementar una nueva relación de nombre “Dotación”, que tuviese una relación de asociación con la relación de “Tipos habitación”. Los anteriores cambios se pueden visualizar de la siguiente manera.

Dotaciones

id_dotacion	nombre
NN, PK	NN

Dependencias funcionales

Id dotacion → nombre

Dotaciones_incluidas

tipo_habitacion	dotacion	cantidad
PK, FK tipos_habitacion.id_tipo	PK, FK dotaciones.id_dotacion	NN

Dependencias funcionales

tipo habitacion, dotacion → cantidad

Acompañantes

num_documento	tipo_documento	reserva
NN, PK	NN, CK[CC, CE, TI]	NN, PK, FK reservas.cod_reserva

Dependencias funcionales

numero documento, reserva → tipo documento

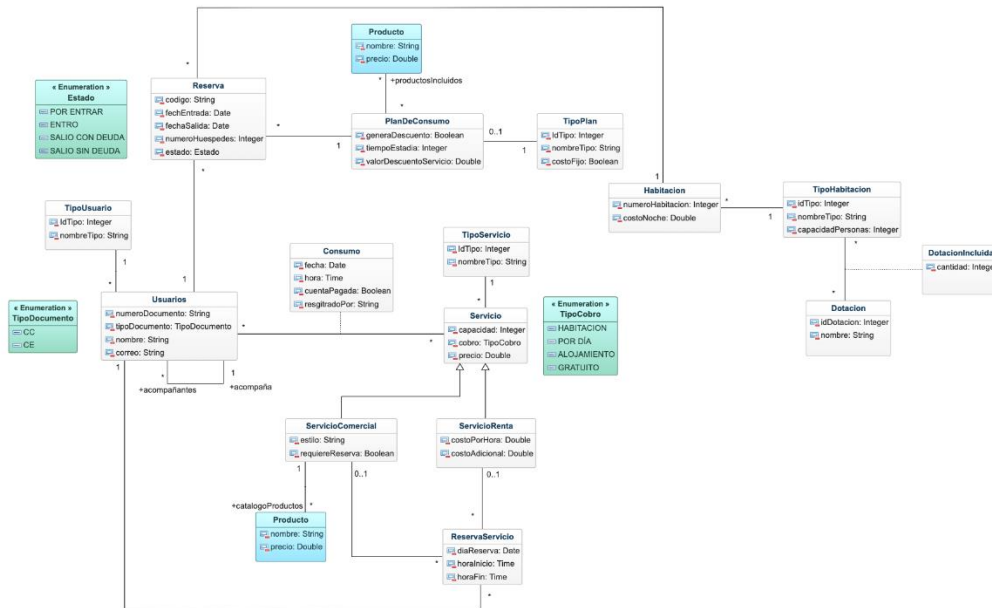
En la relación acompañantes, el atributo “tipo documento” puede tomar un valor adicional a los establecidos en la tabla de usuarios, este valor, “tarjeta de identidad” evita la existencia de una dependencia funcional parcial entre el atributo primo “número de documento” y “tipo de documento”.

Como se puede apreciar, no se realizó un proceso muy exhaustivo de normalización, pues el diagrama de clases fue pensado y modificado con el propósito de evitar problemas con las formas normales.

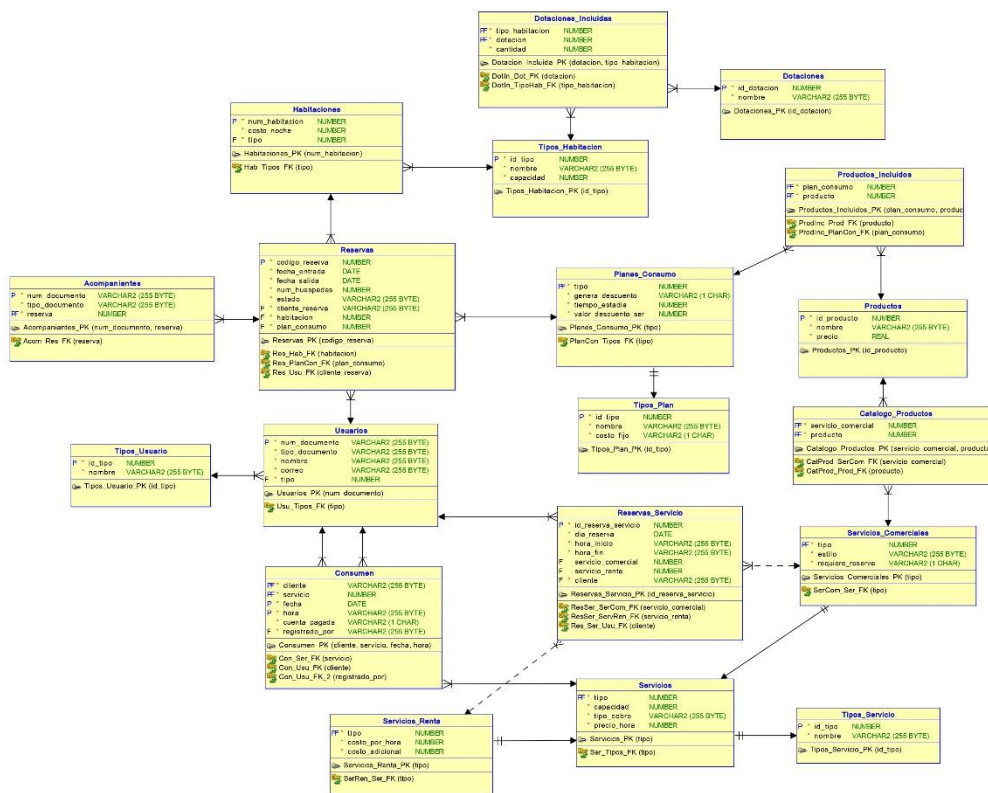
DIAGRAMAS ACTUALIZADOS

Con los cambios previos y el proceso de normalización se obtienen el diagrama de clases UML y el diagrama relacional actualizados, los cuales se exhiben a continuación. Se omite mostrar el diagrama entidad relación debido a su similitud con el modelo relacional (Pues fue generado con la herramienta de ingeniería reversa); sin embargo, este puede ser visualizado en el repositorio del grupo.

Diagrama de clases UML



Modelo relacional



IMPACTO DE LOS NUEVOS REQUERIMIENTOS

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Afortunadamente, los nuevos requerimientos no supusieron una cantidad de cambios sustancial o de magnitud crítica. Cada requerimiento involucró una cierta cantidad de modificaciones al Front-end de la aplicación, además de la creación de nuevas consultas, este es un impacto ya considerado por lo que no se toma como un reto a la implementación. Más allá de la inclusión de unos pocos atributos a ciertas clases, la aplicación soportó de manera satisfactoria la expansión.

DISEÑO DE LA APLICACIÓN

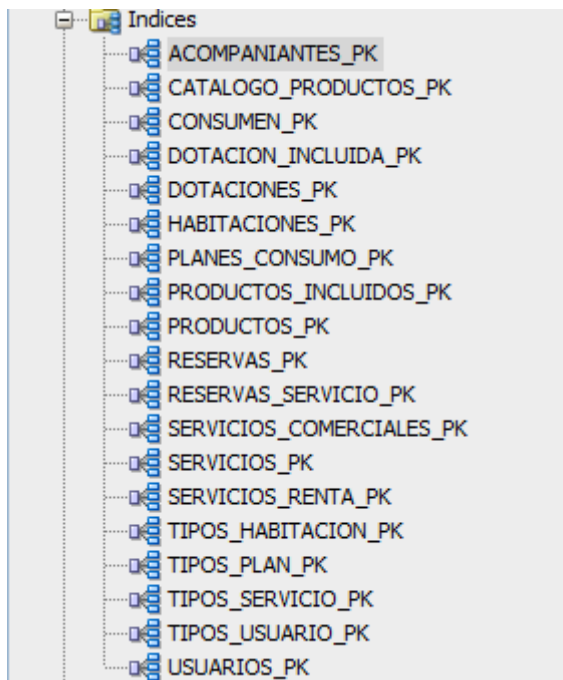
Tras haber corregido y mejorado los diagramas y la composición de la aplicación se da paso a la segunda parte del proyecto; el diseño de la aplicación. En primer lugar, se establecieron ciertos índices con propósitos de optimización de las consultas debido a la inminente población masiva de la base de datos. Por otro lado, se tiene el corazón del funcionamiento, las consultas SQL que responden a los nuevos requerimientos funcionales. En este apartado será posible apreciar un análisis de estos dos componentes.

ÍNDICES

Para cada requerimiento funcional, se explora la posibilidad de incluir o no un índice, junto con los detalles que esto abarcaría en caso de considerarse su inclusión. Finalmente, se mencionará los índices generados automáticamente por Oracle y se discutirá su aporte a los requerimientos funcionales.

La explicación detallada de los índices escogidos, su justificación y sus especificaciones puede ser encontrada en el documento de índices.

ÍNDICES GENERADOS AUTOMÁTICAMENTE POR ORACLE



Todos los índices generados por defecto son índices primarios ubicados sobre la llave primaria, lo que quiere decir que estos índices son los que establecen el orden de las tuplas en su respectiva relación. En los planes de consumo que serán discutidos por cada requerimiento será posible

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



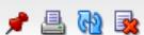
apreciar su contribución al funcionamiento de las consultas. Para conocer quienes son específicamente los atributos que constituyen las llaves primarias a las cuales estos índices se refieren dirigirse al modelo relacional.

DISEÑO DE CONSULTAS

Para cada uno de los requerimientos funcionales de consulta se documentarán distintos aspectos de interés. Entre estos aspectos se encuentra la sentencia SQL empleada, los planes de consulta, distribución de los datos, tiempos, análisis de eficiencia, entre otros aspectos.

RFC1 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Mostrar el dinero recolectado por servicios en cada habitación en el último año corrido																						
Sentencias SQL																							
<pre>--RFC1 SELECT H.num_habitacion, SUM(NVL(S.precio, 0)) AS dinero FROM habitaciones H LEFT JOIN reservas R ON R.habitacion = H.num_habitacion LEFT JOIN consumen C ON C.cliente = R.cliente_reserva LEFT JOIN servicios S ON S.tipo = C.servicio WHERE (C.fecha BETWEEN (SELECT TO_DATE('01/01/' EXTRACT(YEAR FROM SYSDATE)) FROM DUAL) AND (SELECT SYSDATE FROM DUAL))OR R.habitacion IS NULL GROUP BY H.num_habitacion;</pre>																							
Distribución de los datos																							
Se decidió que todas las reservas, consumos y reservas de servicio serían realizadas durante el último año corrido (del 28/10/22 al 28/10/23), por lo que no es posible comparar el resultado de la query con años anteriores.																							
Efecto de los índices	<div> SQL Todas las Filas Recuperadas: 15 en 0,676 segundos</div> <table><thead><tr><th>NUM_HABITACION</th><th>DINERO</th></tr></thead><tbody><tr><td>1</td><td>305 198370244</td></tr><tr><td>2</td><td>105 180340278</td></tr><tr><td>3</td><td>104 177520232</td></tr><tr><td>4</td><td>203 169490320</td></tr><tr><td>5</td><td>204 142520174</td></tr><tr><td>6</td><td>302 131920274</td></tr><tr><td>7</td><td>303 119600266</td></tr><tr><td>8</td><td>103 177840200</td></tr><tr><td>9</td><td>101 118840272</td></tr><tr><td>10</td><td>202 145840164</td></tr></tbody></table> <p>Se completo la consulta en muy buen tiempo.</p>	NUM_HABITACION	DINERO	1	305 198370244	2	105 180340278	3	104 177520232	4	203 169490320	5	204 142520174	6	302 131920274	7	303 119600266	8	103 177840200	9	101 118840272	10	202 145840164
NUM_HABITACION	DINERO																						
1	305 198370244																						
2	105 180340278																						
3	104 177520232																						
4	203 169490320																						
5	204 142520174																						
6	302 131920274																						
7	303 119600266																						
8	103 177840200																						
9	101 118840272																						
10	202 145840164																						

Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



PLAN_TABLE_OUTPUT	
1	Plan hash value: 105620428
2	
3	
4	-----
4	Id Operation Name Rows Bytes Cost (%CPU) Time
5	-----
6	0 SELECT STATEMENT 1 332 7 (15) 00:00:01
7	1 HASH GROUP BY 1 332 7 (15) 00:00:01
8	* 2 FILTER
9	3 NESTED LOOPS OUTER 15 4980 6 (0) 00:00:01
10	* 4 HASH JOIN OUTER 15 4590 6 (0) 00:00:01
11	* 5 HASH JOIN OUTER 15 2325 5 (0) 00:00:01
12	6 INDEX FULL SCAN HABITACIONES_PK 15 195 1 (0) 00:00:01
13	7 TABLE ACCESS FULL RESERVAS 1 142 4 (0) 00:00:01
14	8 INDEX FULL SCAN CONSUMEN_PK 1 151 1 (0) 00:00:01
15	9 TABLE ACCESS BY INDEX ROWID SERVICIOS 1 26 0 (0) 00:00:01
16	* 10 INDEX UNIQUE SCAN SERVICIOS_PK 1 0 (0) 00:00:01
17	11 FAST DUAL 1 2 (0) 00:00:01
18	12 FAST DUAL 1 2 (0) 00:00:01
19	-----
20	
21	Predicate Information (identified by operation id):
22	-----
23	
24	2 - filter("R"."HABITACION" IS NULL OR "C"."FECHA">= (SELECT
25	TO_DATE('01/01/' TO_CHAR(EXTRACT(YEAR FROM SYSDATE@!))) FROM "SYS"."DUAL" "DUAL") AND
26	"C"."FECHA"<= (SELECT SYSDATE@! FROM "SYS"."DUAL" "DUAL"))
27	4 - access("C"."CLIENTE" (+)= "R"."CLIENTE_RESERVA")
28	5 - access("R"."HABITACION" (+)= "H"."NUM_HABITACION")
29	10 - access("S"."TIPO" (+)= "C"."SERVICIO")

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1 7 % de uso de CPU sin índices.

Análisis de eficacia

Plan de consulta obtenido con los índices.

PLAN_TABLE_OUTPUT	
1	Plan hash value: 1156974543
2	
3	
4	-----
4	Id Operation Name Rows Bytes Cost (%CPU) Time
5	-----
6	0 SELECT STATEMENT 1 332 4 (25) 00:00:01
7	1 HASH GROUP BY 1 332 4 (25) 00:00:01
8	* 2 FILTER
9	3 NESTED LOOPS OUTER 15 4980 3 (0) 00:00:01
10	* 4 HASH JOIN OUTER 15 4590 3 (0) 00:00:01
11	5 NESTED LOOPS OUTER 15 2325 2 (0) 00:00:01
12	6 INDEX FULL SCAN HABITACIONES_PK 15 195 1 (0) 00:00:01
13	7 TABLE ACCESS BY INDEX ROWID BATCHED RESERVAS 1 142 1 (0) 00:00:01
14	* 8 INDEX RANGE SCAN IDX_HABITACION_RESERVAS 1 0 (0) 00:00:01
15	9 INDEX FULL SCAN CONSUMEN_PK 1 151 1 (0) 00:00:01
16	10 TABLE ACCESS BY INDEX ROWID SERVICIOS 1 26 0 (0) 00:00:01
17	* 11 INDEX UNIQUE SCAN SERVICIOS_PK 1 0 (0) 00:00:01
18	12 FAST DUAL 1 2 (0) 00:00:01
19	13 FAST DUAL 1 2 (0) 00:00:01
20	-----
21	
22	Predicate Information (identified by operation id):
23	-----
24	
25	2 - filter("R"."HABITACION" IS NULL OR "C"."FECHA">= (SELECT TO_DATE('01/01/' TO_CHAR(EXTRACT(YEAR FROM
26	SYSDATE@!))) FROM "SYS"."DUAL" "DUAL") AND "C"."FECHA"<= (SELECT SYSDATE@! FROM "SYS"."DUAL" "DUAL"))
27	4 - access("C"."CLIENTE" (+)= "R"."CLIENTE_RESERVA")
28	8 - access("R"."HABITACION" (+)= "H"."NUM_HABITACION")
29	11 - access("S"."TIPO" (+)= "C"."SERVICIO")

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

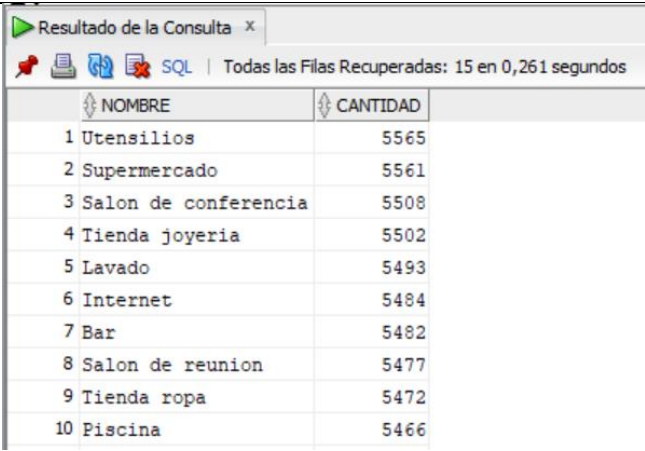
Leidy J. Lozano

Se redujo en 3% el uso de CPU de la consulta.



RFC2 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Mostrar los 20 servicios más populares.																						
Sentencias SQL																							
<pre>--RFC2 SELECT TS.nombre, COALESCE(COUNT(C.servicio), 0) cantidad FROM tipos_servicio TS LEFT JOIN consumen C ON C.servicio = TS.id_tipo AND C.fecha BETWEEN :fecha_inicio AND :fecha_fin INNER JOIN servicios S ON S.tipo = TS.id_tipo GROUP BY TS.nombre ORDER BY cantidad DESC, TS.nombre;</pre>																							
Distribución de los datos																							
Los servicios son constantes en todo el año corrido, luego, no se pueden comparar con años anteriores.																							
Efecto de los índices	<div><p>Resultado de la Consulta x</p><p>Todas las Filas Recuperadas: 15 en 0,261 segundos</p><table><thead><tr><th>NOMBRE</th><th>CANTIDAD</th></tr></thead><tbody><tr><td>1 Utensilios</td><td>5565</td></tr><tr><td>2 Supermercado</td><td>5561</td></tr><tr><td>3 Salon de conferencia</td><td>5508</td></tr><tr><td>4 Tienda joyeria</td><td>5502</td></tr><tr><td>5 Lavado</td><td>5493</td></tr><tr><td>6 Internet</td><td>5484</td></tr><tr><td>7 Bar</td><td>5482</td></tr><tr><td>8 Salon de reunion</td><td>5477</td></tr><tr><td>9 Tienda ropa</td><td>5472</td></tr><tr><td>10 Piscina</td><td>5466</td></tr></tbody></table></div> <p>Se logró un buen tiempo para la consulta en los rasgos de fechas de 23/04/23 – 10/07/23. Independiente a este rango se debería obtener un tiempo similar.</p>	NOMBRE	CANTIDAD	1 Utensilios	5565	2 Supermercado	5561	3 Salon de conferencia	5508	4 Tienda joyeria	5502	5 Lavado	5493	6 Internet	5484	7 Bar	5482	8 Salon de reunion	5477	9 Tienda ropa	5472	10 Piscina	5466
NOMBRE	CANTIDAD																						
1 Utensilios	5565																						
2 Supermercado	5561																						
3 Salon de conferencia	5508																						
4 Tienda joyeria	5502																						
5 Lavado	5493																						
6 Internet	5484																						
7 Bar	5482																						
8 Salon de reunion	5477																						
9 Tienda ropa	5472																						
10 Piscina	5466																						

Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 1

PLAN_TABLE_OUTPUT									
1	Plan hash value: 1054282566								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		14	2478	4 (50)	00:00:01		
7	1	SORT ORDER BY		14	2478	4 (50)	00:00:01		
8	2	HASH GROUP BY		14	2478	4 (50)	00:00:01		
9	* 3	HASH JOIN OUTER		14	2478	2 (0)	00:00:01		
10	4	NESTED LOOPS		14	2170	1 (0)	00:00:01		
11	5	NESTED LOOPS		14	2170	1 (0)	00:00:01		
12	6	INDEX FULL SCAN	SERVICIOS_PK	14	182	1 (0)	00:00:01		
13	* 7	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1		0 (0)	00:00:01		
14	8	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0 (0)	00:00:01		
15	* 9	INDEX FULL SCAN	CONSUMEN_PK	1	22	1 (0)	00:00:01		
16	-----								
17									
18	Predicate Information (identified by operation id):								
19	-----								
20									
21	3 - access("C"."SERVICIO"(+)="TS"."ID_TIPO")								
22	7 - access("S"."TIPO"="TS"."ID_TIPO")								
23	9 - access("C"."FECHA"(>)=:FECHA_INICIO AND "C"."FECHA"(<)=:FECHA_FIN)								
24	filter("C"."FECHA"(>)=:FECHA_INICIO AND "C"."FECHA"(<)=:FECHA_FIN)								

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

4% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT									
1	Plan hash value: 1375721009								
2									
3	-----								
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5	-----								
6	0	SELECT STATEMENT		14	2478	4 (50)	00:00:01		
7	1	SORT ORDER BY		14	2478	4 (50)	00:00:01		
8	2	HASH GROUP BY		14	2478	4 (50)	00:00:01		
9	3	NESTED LOOPS OUTER		14	2478	2 (0)	00:00:01		
10	4	NESTED LOOPS		14	2170	1 (0)	00:00:01		
11	5	INDEX FULL SCAN	SERVICIOS_PK	14	182	1 (0)	00:00:01		
12	6	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0 (0)	00:00:01		
13	* 7	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1		0 (0)	00:00:01		
14	* 8	TABLE ACCESS BY INDEX ROWID BATCHED	CONSUMEN	1	22	1 (0)	00:00:01		
15	* 9	INDEX RANGE SCAN	IDX_SERVICIO	1		0 (0)	00:00:01		
16	-----								
17									
18	Predicate Information (identified by operation id):								
19	-----								
20									
21	7 - access("S"."TIPO"="TS"."ID_TIPO")								
22	8 - filter("C"."FECHA" (+)>=:FECHA_INICIO AND "C"."FECHA" (+)<=:FECHA_FIN)								
23	9 - access("C"."SERVICIO" (+)="TS"."ID_TIPO")								
24									

Se hizo uso de índices en la consulta, pero no se redujo el uso de CPU.

RFC3 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Nombre del requerimiento	Mostrar el índice de ocupación de cada una de las habitaciones del hotel
---------------------------------	--

Sentencias SQL

```
SELECT H.num_habitacion, ROUND(
    (SUM(
        CASE
            WHEN R.fecha_salida >= (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AND R.fecha_entrada <= (SELECT SYSDATE FROM DUAL)
            THEN
                LEAST(R.fecha_salida, (SELECT SYSDATE FROM DUAL)) -
                GREATEST(R.fecha_entrada, (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL))
            ELSE 0
        END
    ) / 365) * 100, 2) AS porcentaje
FROM habitaciones H
LEFT JOIN reservas R ON R.habitacion = H.num_habitacion
GROUP BY (H.num_habitacion);
```

Distribución de los datos

No es posible cambiar la distribución de datos en relación a otros años, ya que se mide la ocupación durante un único año corrido (El año vigente)

Efecto de los índices

Resultado de la Consulta x

Todas las Filas Recuperadas: 15 en 0,016 segundos

	NUM_HABITACION	PORCENTAJE
1	101	295,22
2	102	95,89
3	103	104,93
4	104	101,25
5	105	93,78
6	201	89,04
7	202	83,56
8	203	94,52
9	204	72,6
10	205	82,19
11	301	89,67
12	302	94,52
13	303	79,45
14	304	74,87
15	305	101,37

Se realizó la consulta en muy buen tiempo.

Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 1

PLAN_TABLE_OUTPUT

1 Plan hash value: 2439101390

2

3

4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |

5

6 | 0 | SELECT STATEMENT | | | 1 | 44 | 14 (8) | 00:00:01 |

7 | 1 | FAST DUAL | | | 1 | | 2 (0) | 00:00:01 |

8 | 2 | FAST DUAL | | | 1 | | 2 (0) | 00:00:01 |

9 | 3 | FAST DUAL | | | 1 | | 2 (0) | 00:00:01 |

10 | 4 | FAST DUAL | | | 1 | | 2 (0) | 00:00:01 |

11 | 5 | SORT GROUP BY NOSORT | | | 1 | 44 | 14 (8) | 00:00:01 |

12 | 6 | MERGE JOIN OUTER | | 15 | 660 | 6 (17) | 00:00:01 |

13 | 7 | INDEX FULL SCAN | HABITACIONES_PK | 15 | 195 | 1 (0) | 00:00:01 |

14 | * 8 | SORT JOIN | | 1 | 31 | 5 (20) | 00:00:01 |

15 | 9 | TABLE ACCESS FULL | RESERVAS | 1 | 31 | 4 (0) | 00:00:01 |

16

17

18 Predicate Information (identified by operation id):

19 -----

20

21 8 - access("R"."HABITACION" (+)="H"."NUM_HABITACION")

22 filter("R"."HABITACION" (+)="H"."NUM_HABITACION")

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

14% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT									
1 Plan hash value: 365301076									
2									
3 -----									
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		
5 -----									
6	0	SELECT STATEMENT		1	44	10 (0)	00:00:01		
7	1	FAST DUAL		1		2 (0)	00:00:01		
8	2	FAST DUAL		1		2 (0)	00:00:01		
9	3	FAST DUAL		1		2 (0)	00:00:01		
10	4	FAST DUAL		1		2 (0)	00:00:01		
11	5	SORT GROUP BY NOSORT		1	44	10 (0)	00:00:01		
12	6	NESTED LOOPS OUTER		15	660	2 (0)	00:00:01		
13	7	INDEX FULL SCAN	HABITACIONES_PK	15	195	1 (0)	00:00:01		
14	8	TABLE ACCESS BY INDEX ROWID	RESERVAS	1	31	1 (0)	00:00:01		
15	* 9	INDEX RANGE SCAN	IDX_HABITACION_RESERVAS	1		0 (0)	00:00:01		
16 -----									
17									
18 Predicate Information (identified by operation id):									
19 -----									
20									
21 9 - access("R"."HABITACION" (+)="H"."NUM_HABITACION")									
22									

Se redujo en un 4% el uso de CPU de la consulta.

RFC4 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Mostrar los servicios que cumplen con cierta característica.
--------------------------	--

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



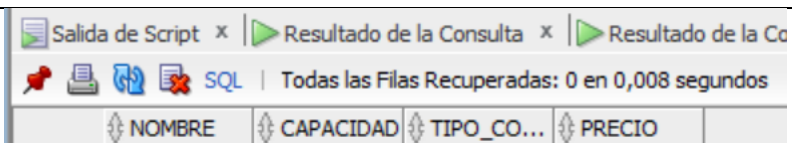
Sentencias SQL

```
SELECT DISTINCT TS.nombre, S.capacidad, S.tipo_cobro, S.precio
FROM tipos_servicio TS
LEFT JOIN consumen C ON C.servicio = TS.id_tipo
INNER JOIN servicios S ON S.tipo = TS.id_tipo
WHERE ((:precio_menor IS NULL AND :precio_mayor IS NULL) OR (S.precio BETWEEN :precio_menor AND :precio_mayor))
AND ((:fecha_inicio IS NULL AND :fecha_fin IS NULL) OR (C.fecha BETWEEN :fecha_inicio AND :fecha_fin))
AND ((:registrado_por IS NULL) OR (:registrado_por = :registrado_por))
AND ((:tipo_menor IS NULL AND :tipo_mayor IS NULL) OR (TS.id_tipo BETWEEN :tipo_menor AND :tipo_mayor));
```

Distribución de los datos

Debido a la densidad de datos, es difícil seleccionar dos conjuntos de datos que manejen distintas distribuciones de datos.

Efecto de los índices



Planes de consulta del requerimiento

Planes de consulta obtenidos

Plan de consulta 1

```
PLAN_TABLE_OUTPUT
1 Plan hash value: 119949606
2
3
4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
5
6 | 0 | SELECT STATEMENT | | | 1 | 461 | 7 (29) | 00:00:01 |
7 | 1 | HASH UNIQUE | | | 1 | 461 | 7 (29) | 00:00:01 |
8 | 2 | NESTED LOOPS | | | 1 | 461 | 6 (17) | 00:00:01 |
9 | 3 | NESTED LOOPS | | | 1 | 461 | 6 (17) | 00:00:01 |
10 |* 4 | FILTER | | | | | | |
11 | 5 | MERGE JOIN OUTER | | | 1 | 293 | 6 (17) | 00:00:01 |
12 | 6 | TABLE ACCESS BY INDEX ROWID | TIPOS_SERVICIO | 1 | 142 | 1 (0) | 00:00:01 |
13 |* 7 | INDEX FULL SCAN | TIPOS_SERVICIO_PK | 1 | | 1 (0) | 00:00:01 |
14 |* 8 | SORT JOIN | | | 1 | 151 | 5 (20) | 00:00:01 |
15 | 9 | TABLE ACCESS FULL | CONSUMEN | 1 | 151 | 4 (0) | 00:00:01 |
16 |* 10 | INDEX UNIQUE SCAN | SERVICIOS_PK | 1 | | 0 (0) | 00:00:01 |
17 |* 11 | TABLE ACCESS BY INDEX ROWID | SERVICIOS | 1 | 168 | 0 (0) | 00:00:01 |
18
19
20 Predicate Information (identified by operation id):
21 -----
22
23 4 - filter((:FECHA_INICIO IS NULL AND :FECHA_FIN IS NULL OR "C"."FECHA">=:FECHA_INICIO AND
24 "C"."FECHA"<=:FECHA_FIN) AND (:REGISTRADO_POR IS NULL OR
25 "C"."REGISTRADO_POR"=:REGISTRADO_POR))
26 7 - filter(:TIPO_MENOR IS NULL AND :TIPO_MAYOR IS NULL OR
27 "TS"."ID_TIPO">=:TO_NUMBER(:TIPO_MENOR) AND "TS"."ID_TIPO"<=:TO_NUMBER(:TIPO_MAYOR))
28 8 - access("C"."SERVICIO"(:)= "TS"."ID_TIPO")
29 filter("C"."SERVICIO"(:)= "TS"."ID_TIPO")
30 10 - access("S"."TIPO"=: "TS"."ID_TIPO")
31 11 - filter("S"."PRECIO">=:TO_NUMBER(:PRECIO_MENOR) AND
32 "S"."PRECIO"<=:TO_NUMBER(:PRECIO_MAYOR) OR :PRECIO_MENOR IS NULL AND :PRECIO_MAYOR IS NULL)
```

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

7% de uso de CPU sin índices.

Análisis de eficacia

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano







PLAN_TABLE_OUTPUT							
1	Plan hash value: 4272542361						
2							
3	-----						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5	-----						
6	0	SELECT STATEMENT		1	461	6 (17)	00:00:01
7	1	HASH UNIQUE		1	461	6 (17)	00:00:01
8	* 2	FILTER					
9	3	NESTED LOOPS OUTER		1	461	5 (0)	00:00:01
10	4	NESTED LOOPS		1	310	4 (0)	00:00:01
11	* 5	TABLE ACCESS FULL	SERVICIOS	1	168	4 (0)	00:00:01
12	6	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0 (0)	00:00:01
13	* 7	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1		0 (0)	00:00:01
14	8	TABLE ACCESS BY INDEX ROWID BATCHED	CONSUMEN	1	151	1 (0)	00:00:01
15	* 9	INDEX RANGE SCAN	IDX_SERVICIO	1		0 (0)	00:00:01
16	-----						
17							
18	Predicate Information (identified by operation id):						
19	-----						
20							
21	2	filter((:FECHA_INICIO IS NULL AND :FECHA_FIN IS NULL OR "C"."FECHA">=:FECHA_INICIO AND					
22		"C"."FECHA"<=:FECHA_FIN) AND (:REGISTRADO_POR IS NULL OR "C"."REGISTRADO_POR"=:REGISTRADO_POR))					
23	5	filter("S"."PRECIO">=TO_NUMBER(:PRECIO_MENOR) AND "S"."PRECIO"<=TO_NUMBER(:PRECIO_MAYOR) OR					
24		:PRECIO_MENOR IS NULL AND :PRECIO_MAYOR IS NULL)					
25	7	access("S"."TIPO"="TS"."ID_TIPO")					
26		filter(:TIPO_MENOR IS NULL AND :TIPO_MAYOR IS NULL OR "TS"."ID_TIPO">=TO_NUMBER(:TIPO_MENOR)					
27		AND "TS"."ID_TIPO"<=TO_NUMBER(:TIPO_MAYOR))					
28	9	access("C"."SERVICIO"("+)"TS"."ID_TIPO")					

Se redujo en un 1% el uso de CPU de la consulta.

RFC5 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Mostrar el consumo en hotelandes por un usuario dado, en un rango de fechas indicado.														
Sentencias SQL															
<pre>SELECT C.cliente, TS.nombre as servicio, C.fecha, C.cuenta_pagada, C.hora, C.registrado_por FROM consumen C INNER JOIN tipos_servicio TS ON TS.id_tipo = C.servicio WHERE C.cliente = :cliente AND (C.fecha BETWEEN :fecha_inicio AND :fecha_fin);</pre>															
Distribución de los datos															
Es difícil comparar con distintos clientes, debido a la cardinalidad de la relación usuarios los servicios consumidos por cada uno son reducidos, luego, no se obtiene mucha diferencia para observar una distribución.															
Efecto de los índices	<div>Resultado de la Consulta x</div> <div>    SQL Todas las Filas Recuperadas: 1 en 0,019 segundos</div> <table><thead><tr><th></th><th>CLIENTE</th><th>SERVICIO</th><th>FECHA</th><th>CUENTA_PAGADA</th><th>HORA</th><th>REGISTRADO_POR</th></tr></thead><tbody><tr><td>1</td><td>7118132732</td><td>Internet</td><td>13/03/23</td><td>Y</td><td>05:34:32</td><td>5760989203</td></tr></tbody></table>		CLIENTE	SERVICIO	FECHA	CUENTA_PAGADA	HORA	REGISTRADO_POR	1	7118132732	Internet	13/03/23	Y	05:34:32	5760989203
	CLIENTE	SERVICIO	FECHA	CUENTA_PAGADA	HORA	REGISTRADO_POR									
1	7118132732	Internet	13/03/23	Y	05:34:32	5760989203									

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Se cumple el requisito no funcional de tiempo para el requerimiento para un rango entre 28/10/22 – 20/07/23, para el cliente que se aprecia en la imagen.

Planes de consulta del requerimiento

Planes de consulta obtenidos

Plan de consulta 1

PLAN_TABLE_OUTPUT							
1 Plan hash value: 408584496							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	555	1 (0)	00:00:01
7	* 1	FILTER					
8	2	NESTED LOOPS		1	555	1 (0)	00:00:01
9	3	NESTED LOOPS		1	555	1 (0)	00:00:01
10	4	TABLE ACCESS BY INDEX ROWID BATCHED	CONSUMEN	1	413	1 (0)	00:00:01
11	* 5	INDEX RANGE SCAN	CONSUMEN_PK	1		1 (0)	00:00:01
12	* 6	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1		0 (0)	00:00:01
13	7	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0 (0)	00:00:01
14							
15							
16 Predicate Information (identified by operation id):							
17							
18							
19	1 - filter(TO_DATE(:FECHA_FIN)>=TO_DATE(:FECHA_INICIO))						
20	5 - access("C"."CLIENTE"=:CLIENTE AND "C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN)						
21	filter("C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN)						
22	6 - access("TS"."ID_TIPO"="C"."SERVICIO")						

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

1% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT							
1 Plan hash value: 408584496							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	555	1 (0)	00:00:01
7	* 1	FILTER					
8	2	NESTED LOOPS		1	555	1 (0)	00:00:01
9	3	NESTED LOOPS		1	555	1 (0)	00:00:01
10	4	TABLE ACCESS BY INDEX ROWID BATCHED	CONSUMEN	1	413	1 (0)	00:00:01
11	* 5	INDEX RANGE SCAN	CONSUMEN_PK	1		1 (0)	00:00:01
12	* 6	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1		0 (0)	00:00:01
13	7	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0 (0)	00:00:01
14							
15							
16 Predicate Information (identified by operation id):							
17							
18							
19	1 - filter(TO_DATE(:FECHA_FIN)>=TO_DATE(:FECHA_INICIO))						
20	5 - access("C"."CLIENTE"=:CLIENTE AND "C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN)						
21	filter("C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN)						
22	6 - access("TS"."ID_TIPO"="C"."SERVICIO")						

No se hizo uso de índices en la consulta, por lo que no se redujo el uso de CPU.

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



RFC6 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Analizar la operación de hotelandes
Sentencias SQL	
<pre>--RFC6.1 WITH Fechas (fecha_reserva) AS (SELECT MIN(fecha_entrada) FROM reservas UNION ALL SELECT fecha_reserva + 1 FROM Fechas WHERE fecha_reserva + 1 <= (SELECT MAX(fecha_salida) FROM reservas)) SELECT fecha_reserva AS fecha, COUNT(*) AS cantidad FROM Fechas INNER JOIN reservas ON fecha_reserva BETWEEN fecha_entrada AND fecha_salida GROUP BY (fecha_reserva) ORDER BY cantidad DESC, fecha_reserva DESC FETCH FIRST 10 ROWS ONLY; --RFC6.2 SELECT C.fecha, COUNT(C.fecha) AS cantidad FROM consumen C GROUP BY (C.fecha) ORDER BY cantidad DESC, C.fecha DESC FETCH FIRST 10 ROWS ONLY; --RFC6.3 WITH Fechas (fecha_reserva) AS (SELECT MIN(fecha_entrada) FROM reservas UNION ALL SELECT fecha_reserva + 1 FROM Fechas WHERE fecha_reserva + 1 <= (SELECT MAX(fecha_salida) FROM reservas)) SELECT fecha_reserva, COUNT(*) AS cantidad FROM Fechas INNER JOIN reservas ON fecha_reserva BETWEEN fecha_entrada AND fecha_salida GROUP BY (fecha_reserva) ORDER BY cantidad ASC, fecha_reserva DESC FETCH FIRST 10 ROWS ONLY;</pre>	
Distribución de los datos	
La operación del hotel es constante para todo el año corrido, no se tienen más años con los cuales comparar.	
Efecto de los índices	

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Resultado de la Consulta x		Resultado de la Consulta 1 x		Resultado de la Consulta 2 x	
Todas las Filas Recuperadas: 10 en 0,027 segundos					
	FECHA		CANTIDAD		
1	13/03/23		28		
2	13/12/22		28		
3	18/07/23		27		
4	15/03/23		27		
5	17/02/23		27		
6	15/12/22		27		
7	14/12/22		27		
8	02/07/23		26		
9	01/07/23		26		
10	14/03/23		26		

Resultado de la Consulta x		Resultado de la Consulta 1 x		Resultado de la Consulta 2 x	
Todas las Filas Recuperadas: 10 en 0,049 segundos					
	FECHA		CANTIDAD		
1	21/11/22		717		
2	02/07/23		711		
3	09/06/23		705		
4	22/07/23		702		
5	16/12/22		693		
6	29/03/23		692		
7	11/02/23		691		
8	18/01/23		691		
9	01/09/23		689		
10	16/08/23		688		

Resultado de la Consulta x		Resultado de la Consulta 1 x		Resultado de la Consulta 2 x	
Todas las Filas Recuperadas: 10 en 0,029 segundos					
	FECHA_RESERVA		CANTIDAD		
1	01/11/23		2		
2	31/10/23		5		
3	30/10/23		7		
4	29/10/23		7		
5	08/11/22		8		
6	28/10/22		8		
7	10/08/23		9		
8	29/10/22		9		
9	04/09/23		11		
10	18/06/23		11		

Las tres consultas cumplieron satisfactoriamente el estándar de tiempo esperado.

Planes de consulta del requerimiento

Planes de consulta obtenidos

Leidy J. Lozano



```

6 PLAN_TABLE_OUTPUT
7
8 1 Plan hash value: 4048453059
9
10
11 -----
12
13 4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
14 -----
15 6 | 0 | SELECT STATEMENT | | | 10 | 570 | 44 (10) | 00:00:01 |
16 7 | 1 | SORT ORDER BY | | | 10 | 570 | 44 (10) | 00:00:01 |
17 8 |* 2 | VIEW | | | 10 | 570 | 43 (7) | 00:00:01 |
18 9 |* 3 | WINDOW SORT PUSHED RANK | | | 1 | 27 | 43 (7) | 00:00:01 |
19 10 | 4 | HASH GROUP BY | | | 1 | 27 | 43 (7) | 00:00:01 |
20 11 | 5 | MERGE JOIN | | | 1 | 27 | 42 (5) | 00:00:01 |
21 12 | 6 | SORT JOIN | | | 2 | 18 | 37 (3) | 00:00:01 |
22 13 | 7 | VIEW | | | 2 | 18 | 36 (0) | 00:00:01 |
23 14 | 8 | UNION ALL (RECURSIVE WITH) BREADTH FIRST | | | | | |
24 15 | 9 | SORT AGGREGATE | | | 1 | 9 | | |
25 16 | 10 | TABLE ACCESS FULL | RESERVAS | 3 | 27 | 4 (0) | 00:00:01 |
26 17 |* 11 | RECURSIVE WITH PUMP | | | | | |
27 18 | 12 | SORT AGGREGATE | | | 1 | 9 | | |
28 19 | 13 | TABLE ACCESS FULL | RESERVAS | 3 | 27 | 4 (0) | 00:00:01 |
29 20 |* 14 | FILTER | | | | | |
30 21 |* 15 | SORT JOIN | | | 3 | 54 | 5 (20) | 00:00:01 |
31 22 | 16 | TABLE ACCESS FULL | RESERVAS | 3 | 54 | 4 (0) | 00:00:01 |
32 -----
33
34 25 Predicate Information (identified by operation id):
35 -----
36
37 2 - filter("from$_subquery$_007"."rowlimit_$$_rownumber"<=10)
38
39 3 - filter(ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC
40 ,INTERNAL_FUNCTION("FECHAS"."FECHA_RESERVA") DESC )<=10)
41
42 11 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS"
43 "RESERVAS"))
44
45 14 - filter("FECHA_RESERVA"<="FECHA_SALIDA")
46
47 15 - access(INTERNAL_FUNCTION("FECHA_RESERVA")>=INTERNAL_FUNCTION("FECHA_ENTRADA"))
48 filter(INTERNAL_FUNCTION("FECHA_RESERVA")>=INTERNAL_FUNCTION("FECHA_ENTRADA"))

```

```

1 PLAN_TABLE_OUTPUT
2
3 -----
4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
5 -----
6 | 0 | SELECT STATEMENT | | | | | |
7 | 1 | SORT ORDER BY | | 10 | 570 | 4 (75) | 00:00:01 |
8 |* 2 | VIEW | | 10 | 570 | 3 (67) | 00:00:01 |
9 |* 3 | WINDOW SORT PUSHED RANK | | 1 | 9 | 3 (67) | 00:00:01 |
10 | 4 | HASH GROUP BY | | 1 | 9 | 3 (67) | 00:00:01 |
11 | 5 | INDEX FULL SCAN | CONSUMEN_PK | 1 | 9 | 1 (0) | 00:00:01 |
12 -----
13
14 Predicate Information (identified by operation id):
15 -----
16
17 2 - filter("from$_subquery$002"."rowlimit_$$_rownumber"<=10)
18 3 - filter(ROW_NUMBER() OVER ( ORDER BY COUNT(*) DESC
19 ,INTERNAL_FUNCTION("C"."FECHA") DESC )<=10)

```



```

1 PLAN_TABLE_OUTPUT
2
3
4 Plan hash value: 4048453059
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
10
```

44% de uso de CPU sin índices.

```

6 @PLAN_TABLE_OUTPUT
7 Plan hash value: 1430319473
8
9 -----
10 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time
11 -----
12 | 0 | SELECT STATEMENT | | | | |
13 | 1 | SORT ORDER BY | | | | |
14 | * 2 | VIEW | | 10 | 570 | 13 (24) | 00:00:01
15 | * 3 | WINDOW SORT PUSHED RANK | | 1 | 27 | 12 (17) | 00:00:01
16 | 4 | HASH GROUP BY | | 1 | 27 | 12 (17) | 00:00:01
17 | 5 | NESTED LOOPS | | 1 | 27 | 10 (0) | 00:00:01
18 | 6 | INDEX FULL SCAN | IDX_FECHA_ENTRADA_SALIDA | 1 | 18 | 1 (0) | 00:00:01
19 | * 7 | VIEW | | 1 | 9 | 9 (0) | 00:00:01
20 | 8 | UNION ALL (RECURSIVE WITH) BREADTH FIRST | | | | |
21 | 9 | SORT AGGREGATE | | 1 | 9 | |
22 | 10 | INDEX FULL SCAN (MIN/MAX) | IDX_FECHA_ENTRADA | 1 | 9 | 1 (0) | 00:00:01
23 | * 11 | RECURSIVE WITH PUMP | | | | |
24 | 12 | SORT AGGREGATE | | 1 | 9 | |
25 | 13 | INDEX FULL SCAN | IDX_FECHA_ENTRADA_SALIDA | 1 | 9 | 1 (0) | 00:00:01
26 -----
27 Predicate Information (identified by operation id):
28 -----
29
30 2 - filter("from_subquery5_007"."rowlimit_66_rownumber"<=10)
31
32 3 - filter(ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC ,INTERNAL_FUNCTION("FECHAS"."FECHA_RESERVA") DESC )<=10)
33
34 7 - filter("FECHA_RESERVA">="FECHA_ENTRADA" AND "FECHA_RESERVA"<<"FECHA_SALIDA")
35
36 11 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS" "RESERVAS"))
37
38 -----

```

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



PLAN_TABLE_OUTPUT							
1 Plan hash value: 288802498							
2							
3 -----							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5 -----							
6	0	SELECT STATEMENT		10	570	3 (67)	00:00:01
7	1	SORT ORDER BY		10	570	3 (67)	00:00:01
8	* 2	VIEW		10	570	2 (50)	00:00:01
9	* 3	WINDOW SORT PUSHED RANK		1	9	2 (50)	00:00:01
10	4	SORT GROUP BY NOSORT		1	9	2 (50)	00:00:01
11	5	INDEX FULL SCAN	IDX_FECHA	1	9	1 (0)	00:00:01
12 -----							
13							
14 Predicate Information (identified by operation id):							
15 -----							
16							
17 2 - filter("from\$_subquery\$_002"."rowlimit_\$\$_rownumber"<=10)							
18 3 - filter(ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC							
19 ,INTERNAL_FUNCTION("C"."FECHA") DESC)<=10)							
20							

PLAN_TABLE_OUTPUT									
1 Plan hash value: 1430319473									
2									
3									

4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time		

5									
6	0	SELECT STATEMENT		10	570	13 (24)	00:00:01		
7	1	SORT ORDER BY		10	570	13 (24)	00:00:01		
8	* 2	VIEW		10	570	12 (17)	00:00:01		
9	* 3	WINDOW SORT PUSHED RANK		1	27	12 (17)	00:00:01		
10	4	HASH GROUP BY		1	27	12 (17)	00:00:01		
11	5	NESTED LOOPS		1	27	10 (0)	00:00:01		
12	6	INDEX FULL SCAN	IDX_FECHA_ENTRADA_SALIDA	1	18	1 (0)	00:00:01		
13	* 7	VIEW		1	9	9 (0)	00:00:01		
14	8	UNION ALL (RECURSIVE WITH) BREADTH FIRST		1	9	1 (0)	00:00:01		
15	9	SORT AGGREGATE		1	9	1 (0)	00:00:01		
16	10	INDEX FULL SCAN (MIN/MAX)	IDX_FECHA_ENTRADA	1	9	1 (0)	00:00:01		
17	* 11	RECURSIVE WITH PUMP		1	9	1 (0)	00:00:01		
18	12	SORT AGGREGATE		1	9	1 (0)	00:00:01		
19	13	INDEX FULL SCAN	IDX_FECHA_ENTRADA_SALIDA	1	9	1 (0)	00:00:01		

21									
22 Predicate Information (identified by operation id):									
23 -----									
24									
25 2 - filter("from\$_subquery\$_007"."rowlimit_\$\$_rownumber"<=10)									
26 3 - filter(ROW_NUMBER() OVER (ORDER BY COUNT(*),INTERNAL_FUNCTION("FECHAS"."FECHA_RESERVA") DESC)<=10)									
27 7 - filter("FECHA_RESERVA">="FECHA_ENTRADA" AND "FECHA_RESERVA"<="FECHA_SALIDA")									
28 11 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS" "RESERVAS"))									

Las consultas 1 y 3 tuvieron una reducción de uso de CPU de 31%, mientras que la consulta 2n tuvo una reducción de 1%.

RFC7 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del
requerimiento

Encontrar los buenos clientes

Sentencias SQL

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



```
--RFC7
WITH Fechas (fecha_reserva) AS (
    SELECT MIN(fecha_entrada)
    FROM reservas
    UNION ALL
    SELECT fecha_reserva + 1
    FROM Fechas
    WHERE fecha_reserva + 1 <= (SELECT MAX(fecha_salida) FROM reservas)
)
SELECT U.num_documento AS cliente, ROUND(COUNT(DISTINCT CONCAT(U.num_documento, fecha_reserva))/7, 2) AS semanas,
FROM Fechas
INNER JOIN reservas R ON fecha_reserva BETWEEN R.fecha_entrada AND R.fecha_salida
INNER JOIN usuarios U ON R.cliente_reserva = U.num_documento
INNER JOIN consumen C ON C.cliente = U.num_documento
INNER JOIN servicios S ON S.tipo = C.servicio
INNER JOIN Suma_Consumo SC ON U.num_documento = SC.cliente
WHERE U.tipo = 1
GROUP BY (U.num_documento, SC.suma_consumo)
HAVING COUNT(DISTINCT CONCAT(U.num_documento, fecha_reserva)) >= 14 OR SC.suma_consumo > 1500000;
```

Distribución de los datos

Los buenos clientes son los mismos a lo largo del año corrido vigente, no hay información de años anteriores para hacer la comparación.

Efecto de los índices

Resultado de la Consulta x

Todas las Filas Recuperadas: 3 en 0,178 segundos

	CLIENTE	SEMANAS	SUMA_CONSUMO
1	1008	4,57	1400000
2	1007	4,57	1800000
3	1006	52,29	2750016

El requerimiento cumple con el requerimiento no funcional de eficiencia

Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 1

PLAN_TABLE_OUTPUT						
1 Plan hash value: 1848759086						
2						
3						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU) Time
5						
6	0	SELECT STATEMENT		1	275	43 (3) 00:00:01
7	1	FILTER				
8	2	HASH GROUP BY		1	275	43 (3) 00:00:01
9	3	VIEW	VW_DAG_0	1	275	43 (3) 00:00:01
10	4	HASH GROUP BY		1	569	43 (3) 00:00:01
11	5	NESTED LOOPS		1	569	42 (0) 00:00:01
12	6	NESTED LOOPS SEMI		1	560	6 (0) 00:00:01
13	7	NESTED LOOPS		1	431	5 (0) 00:00:01
14	8	MERGE JOIN CARTESIAN		1	289	5 (0) 00:00:01
15	9	VIEW		1	142	1 (0) 00:00:01
16	10	HASH GROUP BY		1	168	1 (0) 00:00:01
17	11	NESTED LOOPS		1	168	1 (0) 00:00:01
18	12	NESTED LOOPS		1	168	1 (0) 00:00:01
19	13	INDEX FULL SCAN	CONSUMEN_FK	1	142	1 (0) 00:00:01
20	14	INDEX UNIQUE SCAN	SERVICIOS_PK	1		0 (0) 00:00:01
21	15	TABLE ACCESS BY INDEX ROWID	SERVICIOS	1	26	0 (0) 00:00:01
22	16	BUFFER SORT		1	147	5 (0) 00:00:01
23	17	TABLE ACCESS FULL	RESERVAS	1	147	4 (0) 00:00:01
24	18	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	142	0 (0) 00:00:01
25	19	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0) 00:00:01
26	20	INDEX RANGE SCAN	CONSUMEN_FK	1	129	1 (0) 00:00:01
27	21	VIEW		1	9	36 (0) 00:00:01
28	22	UNION ALL (RECURSIVE WITH) BREADTH FIRST				
29	23	SORT AGGREGATE		1	9	
30	24	TABLE ACCESS FULL	RESERVAS	1	9	4 (0) 00:00:01
31	25	RECURSIVE WITH PUMP				
32	26	SORT AGGREGATE		1	9	
33	27	TABLE ACCESS FULL	RESERVAS	1	9	4 (0) 00:00:01
34						
35						
36 Predicate Information (identified by operation id):						
37						
38						
39 1 - filter(COUNT("ITEM_1")>=14 OR "ITEM_3">15000000)						
40 14 - access("S"."TIPO"="C"."SERVICIO")						
41 18 - filter("U"."TIPO"=1)						
42 19 - access("R"."CLIENTE_RESERVA"="U"."NUM_DOCUMENTO")						
43 filter("U"."NUM_DOCUMENTO"="SC"."CLIENTE")						
44 20 - access("C"."CLIENTE"="U"."NUM_DOCUMENTO")						
45 21 - filter("FECHA_RESERVA"="R"."FECHA_ENTRADA" AND "FECHA_RESERVA"<="R"."FECHA_SALIDA")						
46 25 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS"						
47 "RESERVAS"))						

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

43% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT						
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU) Time
5						
6	0	SELECT STATEMENT		1	151	24 (5) 00:00:01
7	1	FILTER				
8	2	HASH GROUP BY		1	151	24 (5) 00:00:01
9	3	VIEW	VW_DAG_0	1	151	24 (5) 00:00:01
10	4	HASH GROUP BY		1	185	24 (5) 00:00:01
11	5	NESTED LOOPS		1	185	23 (0) 00:00:01
12	6	NESTED LOOPS		1	185	23 (0) 00:00:01
13	7	NESTED LOOPS SEMI		1	43	23 (0) 00:00:01
14	8	NESTED LOOPS		1	38	23 (0) 00:00:01
15	9	NESTED LOOPS		1	30	22 (0) 00:00:01
16	10	VIEW		2	18	18 (0) 00:00:01
17	11	UNION ALL (RECURSIVE WITH) BREADTH FIRST				
18	12	SORT AGGREGATE		1	8	
19	13	INDEX FULL SCAN (MIN/MAX)	IDX_FECHA_ENTRADA	1	8	2 (0) 00:00:01
20	14	RECURSIVE WITH PUMP				
21	15	SORT AGGREGATE		1	8	
22	16	INDEX FULL SCAN (MIN/MAX)	IDX_FECHA_SALIDA	1	8	2 (0) 00:00:01
23	17	TABLE ACCESS BY INDEX ROWID BATCHED	RESERVAS	1	21	2 (0) 00:00:01

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



PLAN_TABLE_OUTPUT															
24	*	18		INDEX RANGE SCAN		IDX_FECHA_ENTRADA_SALIDA		1		1	(0)		00:00:01		
25	*	19		TABLE ACCESS BY INDEX ROWID		USUARIOS		1		8		1	(0)		00:00:01
26	*	20		INDEX UNIQUE SCAN		USUARIOS_PK		1				0	(0)		00:00:01
27	*	21		INDEX RANGE SCAN		IDX_CLIENTE_CONSUMEN		5		25		0	(0)		00:00:01
28	*	22		INDEX UNIQUE SCAN		SUMA_CONSUMO_PK		1		1		0	(0)		00:00:01
29		23		TABLE ACCESS BY INDEX ROWID		SUMA_CONSUMO		1		142		0	(0)		00:00:01
30	-----														
31															
32	Predicate Information (identified by operation id):														
33	-----														
34															
35	1	-	filter	(COUNT("ITEM_1")>=14 OR "ITEM_3">15000000)											
36	14	-	filter	(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS" "RESERVAS"))											
37	18	-	access	("FECHA_RESERVA"<="R"."FECHA_SALIDA" AND "FECHA_RESERVA">="R"."FECHA_ENTRADA")											
38			filter	("FECHA_RESERVA"<="R"."FECHA_SALIDA")											
39	19	-	filter	("U"."TIPO"=1)											
40	20	-	access	("R"."CLIENTE_RESERVA"="U"."NUM_DOCUMENTO")											
41	21	-	access	("C"."CLIENTE"="U"."NUM_DOCUMENTO")											
42	22	-	access	("U"."NUM_DOCUMENTO"="SC"."CLIENTE")											
43															

Se redujo en un 19% el uso de la CPU de la consulta con la ayuda de los índices.

RFC8 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA BÁSICO.

Información básica del requerimiento

Nombre del requerimiento	Encontrar los servicios que no tienen mucha demanda
Sentencias SQL	
<pre>WITH Semanas (inicio_semana, fin_semana) AS (SELECT (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AS inicio_semana, (SELECT SYSDATE - INTERVAL '1' YEAR + 7 FROM DUAL) AS fin_semana FROM DUAL UNION ALL SELECT fin_semana, fin_semana + 7 FROM Semanas WHERE fin_semana <= (SELECT SYSDATE - 7 FROM DUAL)) SELECT nombre, SUM(menos_tres) semanas FROM (SELECT S.inicio_semana, S.fin_semana, TS.nombre, CASE WHEN COUNT(C.servicio) < 3 THEN 1 ELSE 0 END AS menos_tres FROM Semanas S LEFT JOIN tipos_servicio TS ON 1=1 LEFT JOIN consumen C ON C.fecha BETWEEN S.inicio_semana AND S.fin_semana AND TS.id_tipo = C.servicio GROUP BY S.inicio_semana, S.fin_semana, TS.nombre) GROUP BY (nombre) HAVING SUM(menos_tres) = 52; --RFC9 SELECT U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre tipo, COUNT(U.num_documento) veces_consumo, C.fecha FROM usuarios U INNER JOIN consumen C ON C.cliente = U.num_documento INNER JOIN tipos_usuario TU ON TU.id_tipo = U.tipo WHERE (C.servicio = :servicio) AND (:num_documento IS NULL OR U.num_documento = :num_documento) AND (:tipo_documento IS NULL OR U.tipo_documento = :tipo_documento) AND (:nombre IS NULL OR U.nombre = :nombre) AND (:correo IS NULL OR U.correo = :correo) AND ((:fecha_inicio IS NULL AND :fecha_fin IS NULL) OR (C.fecha BETWEEN :fecha_inicio AND :fecha_fin)) AND (U.tipo = 1) GROUP BY U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre, C.fecha HAVING (:veces_consumo IS NULL AND COUNT(U.num_documento) >= 1) OR COUNT(U.num_documento) >= :veces_consumo;</pre>	

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

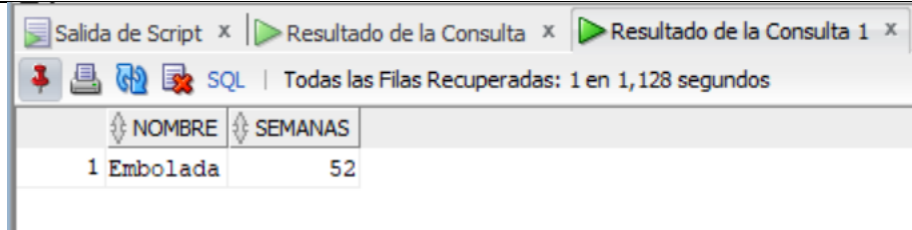
Leidy J. Lozano



Distribución de los datos

Solo se tiene información de un año corrido, por lo que no es posible comparar con años anteriores para observar cómo cambia la distribución de datos.

Efecto de los índices



NOMBRE	SEMANAS
1 Embolada	52

A pesar de la presencia de los índices, el requerimiento supera levemente el tiempo esperado.

Planes de consulta del requerimiento

Planes de consulta obtenidos

Plan de consulta 1

```

1 Plan hash value: 2463305089
2
3
4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
5 -----
6 | 0 | SELECT STATEMENT | | | 58 | 7656 | 36 (3) | 00:00:01 |
7 | * 1 | HASH GROUP BY | | | 58 | 7656 | 36 (3) | 00:00:01 |
8 | 2 | VIEW | | | 58 | 7656 | 36 (3) | 00:00:01 |
9 | 3 | HASH GROUP BY | | | 58 | 10556 | 36 (3) | 00:00:01 |
10 | * 4 | HASH JOIN RIGHT OUTER | | | 58 | 10556 | 35 (0) | 00:00:01 |
11 | 5 | INDEX FULL SCAN | CONSUMEN_PK | 1 | 22 | 1 (0) | 00:00:01 |
12 | 6 | MERGE JOIN OUTER | | | 58 | 9280 | 34 (0) | 00:00:01 |
13 | 7 | VIEW | | 2 | 36 | 26 (0) | 00:00:01 |
14 | 8 | UNION ALL (RECURSIVE WITH) BREADTH FIRST | | | | | | |
15 | 9 | FAST DUAL | | 1 | | 2 (0) | 00:00:01 |
16 | 10 | FAST DUAL | | 1 | | 2 (0) | 00:00:01 |
17 | 11 | FAST DUAL | | 1 | | 2 (0) | 00:00:01 |
18 | * 12 | RECURSIVE WITH PUMP | | | | | | |
19 | 13 | FAST DUAL | | 1 | | 2 (0) | 00:00:01 |
20 | 14 | BUFFER SORT | | 29 | 4118 | 34 (0) | 00:00:01 |
21 | 15 | VIEW | VW_LAT_6F8E1EB6 | 29 | 4118 | 4 (0) | 00:00:01 |
22 | 16 | TABLE ACCESS FULL | TIPOS_SERVICIO | 29 | 4118 | 4 (0) | 00:00:01 |
23
24
25 Predicate Information (identified by operation id):
26
27
28 1 - filter(SUM("MENOS_TRES")=52)
29 4 - access("ITEM_1"="C"."SERVICIO"(+))
30 filter("C"."FECHA"(>)= "S"."INICIO_SEMANA" AND "C"."FECHA"(>)= "S"."FIN_SEMANA")
31 12 - filter("FIN_SEMANA"<= (SELECT SYSDATE!-7 FROM "SYS"."DUAL" "DUAL"))
32

```

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

36% de uso de CPU sin índices.

Análisis de eficacia

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



PLAN_TABLE_OUTPUT									
1 Plan hash value: 2463305089									
2									
3									
4		Id		Operation		Name		Rows	
5								Bytes	
6		0		SELECT STATEMENT				58	
7		* 1		HASH GROUP BY				58	
8		2		VIEW				58	
9		3		HASH GROUP BY				58	
10		* 4		HASH JOIN RIGHT OUTER				58	
11		5		INDEX FULL SCAN		CONSUMEN_PK		1	
12		6		MERGE JOIN OUTER				58	
13		7		VIEW				2	
14		8		UNION ALL (RECURSIVE WITH) BREADTH FIRST					
15		9		FAST DUAL				1	
16		10		FAST DUAL				1	
17		11		FAST DUAL				1	
18		* 12		RECURSIVE WITH PUMP					
19		13		FAST DUAL				1	
20		14		BUFFER SORT				29	
21		15		VIEW		VW_LAT_6F8E1EB6		29	
22		16		TABLE ACCESS FULL		TIPOS_SERVICIO		29	
23									

No se hizo uso de índices en la consulta, por lo que no se redujo el uso de CPU.

RFC9 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA AVANZADA.

Información básica del requerimiento

Nombre del requerimiento	Consultar consumo en hotelandes
Sentencias SQL	
<pre>SELECT U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre tipo, COUNT(U.num_documento) veces_consumo, C.fecha FROM usuarios U INNER JOIN consumen C ON C.cliente = U.num_documento INNER JOIN tipos_usuario TU ON TU.id_tipo = U.tipo WHERE (C.servicio = :servicio) AND (:num_documento IS NULL OR U.num_documento = :num_documento) AND (:tipo_documento IS NULL OR U.tipo_documento = :tipo_documento) AND (:nombre IS NULL OR U.nombre = :nombre) AND (:correo IS NULL OR U.correo = :correo) AND ((:fecha_inicio IS NULL AND :fecha_fin IS NULL) OR (C.fecha BETWEEN :fecha_inicio AND :fecha_fin)) AND (U.tipo = 1) GROUP BY U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre, C.fecha HAVING (:veces_consumo IS NULL AND COUNT(U.num_documento) >= 1) OR COUNT(U.num_documento) >= :veces_consumo;</pre>	
Distribución de los datos	
No hay datos de más años.	
Efecto de los índices	Debido a la especificidad del requerimiento, es más fácil ver su funcionamiento con ayuda del front. Sin embargo, si se corre con los siguientes parámetros: -lan Schmitt, CC 1558336267, desde 28/10/22 hasta 28/10/23, servicio 1, consumido dos veces, correo lauren01@gmail.com . Se obtiene el siguiente tiempo.

Entrega 2 – Optimización de consultas





Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Salida de Script x | Resultado de la Consulta x | Resultado de la Consulta 1 x | Resultado de la Consulta x

    SQL | Todas las Filas Recuperadas: 0 en 0,305 segundos

NUM_DO...	TIPO_DO...	NOMBRE	CORREO	TIPO	VECES_C...	FECHA
-----------	------------	--------	--------	------	------------	-------

Planes de consulta del requerimiento

Planes de consulta obtenidos	
Plan de consulta 1	<pre> PLAN_TABLE_OUTPUT 1 Plan hash value: 3397006984 2 3 ----- 4 Id Operation Name Rows Bytes Cost (%CPU) Time 5 ----- 6 0 SELECT STATEMENT 1 822 2 (50) 00:00:01 7 * 1 FILTER 8 2 HASH GROUP BY 1 822 2 (50) 00:00:01 9 3 NESTED LOOPS 1 822 1 (0) 00:00:01 10 4 NESTED LOOPS 1 822 1 (0) 00:00:01 11 5 NESTED LOOPS 1 293 1 (0) 00:00:01 12 6 TABLE ACCESS BY INDEX ROWID TIPOS_USUARIO 1 142 0 (0) 00:00:01 13 * 7 INDEX UNIQUE SCAN TIPOS_USUARIO_PK 1 0 (0) 00:00:01 14 * 8 INDEX FULL SCAN CONSUMEN_PK 1 151 1 (0) 00:00:01 15 * 9 INDEX UNIQUE SCAN USUARIOS_PK 1 0 (0) 00:00:01 16 * 10 TABLE ACCESS BY INDEX ROWID USUARIOS 1 529 0 (0) 00:00:01 17 ----- 18 19 Predicate Information (identified by operation id): 20 ----- 21 22 1 - filter(:VECES_CONSUMO IS NULL AND COUNT(*)>=1 OR COUNT(*)>=TO_NUMBER(:VECES_CONSUMO)) 23 7 - access("TI"."ID_TIPO"=1) 24 8 - access("C"."SERVICIO"=TO_NUMBER(:SERVICIO)) 25 filter("C"."SERVICIO"=TO_NUMBER(:SERVICIO) AND ("C"."FECHA">=FECHA_INICIO AND 26 "C"."FECHA"<=FECHA_FIN OR :FECHA_INICIO IS NULL AND :FECHA_FIN IS NULL)) 27 9 - access("C"."CLIENTE"="U"."NUM_DOCUMENTO") 28 filter(:NUM_DOCUMENTO IS NULL OR "U"."NUM_DOCUMENTO"=:NUM_DOCUMENTO) 29 10 - filter("U"."TIPO"=1 AND (:TIPO_DOCUMENTO IS NULL OR 30 "U"."TIPO_DOCUMENTO"=:TIPO_DOCUMENTO) AND (:NOMBRE IS NULL OR "U"."NOMBRE"=:NOMBRE) AND 31 (:CORREO IS NULL OR "U"."CORREO"=:CORREO)) </pre>
	Tiempos obtenidos en los planes de consulta
	Tiempo plan de consulta 1 2% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT	
1	Plan hash value: 3397006984
2	
3	-----
4	Id Operation Name Rows Bytes Cost (%CPU) Time
5	-----
6	0 SELECT STATEMENT 1 822 2 (50) 00:00:01
7	* 1 FILTER
8	2 HASH GROUP BY 1 822 2 (50) 00:00:01
9	3 NESTED LOOPS 1 822 1 (0) 00:00:01
10	4 NESTED LOOPS 1 822 1 (0) 00:00:01
11	5 NESTED LOOPS 1 293 1 (0) 00:00:01
12	6 TABLE ACCESS BY INDEX ROWID TIPOS_USUARIO 1 142 0 (0) 00:00:01
13	* 7 INDEX UNIQUE SCAN TIPOS_USUARIO_PK 1 0 (0) 00:00:01
14	* 8 INDEX FULL SCAN CONSUMEN_PK 1 151 1 (0) 00:00:01
15	* 9 INDEX UNIQUE SCAN USUARIOS_PK 1 0 (0) 00:00:01
16	* 10 TABLE ACCESS BY INDEX ROWID USUARIOS 1 529 0 (0) 00:00:01
17	-----

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

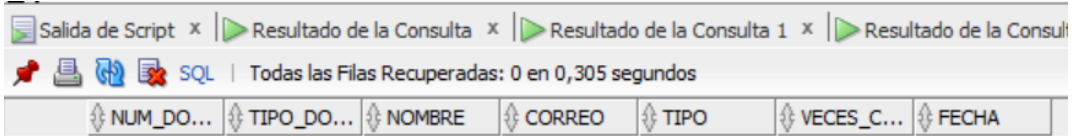
Leidy J. Lozano

No se hizo uso de índices en la consulta, por lo que no se redujo el uso de CPU.



RFC10 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA AVANZADA.

Información básica del requerimiento

Nombre del requerimiento	Consultar consumo en hotelandes – RFC9-v2
Sentencias SQL	
<pre>SELECT U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre tipo FROM usuarios U INNER JOIN tipos_usuario TU ON TU.id_tipo = U.tipo LEFT JOIN (SELECT * FROM consumen C WHERE C.servicio = :servicio AND ((:fecha_inicio IS NULL AND :fecha_fin IS NULL) OR (C.fecha BETWEEN :fecha_inicio AND :fecha_fin))) ClientesConsumo ON U.num_documento = ClientesConsumo.cliente WHERE ClientesConsumo.cliente IS NULL AND (:num_documento IS NULL OR U.num_documento = :num_documento) AND (:tipo_documento IS NULL OR U.tipo_documento = :tipo_documento) AND (:nombre IS NULL OR U.nombre = :nombre) AND (:correo IS NULL OR U.correo = :correo) AND (U.tipo = 1);</pre>	
Distribución de los datos	
No hay datos para más años.	
Efecto de los índices	<p>Debido a la especificad del requerimiento, es más fácil ver su funcionamiento con ayuda del front. Sin embargo, si se corre con los siguientes parámetros: -lan Schmitt, CC 1558336267, desde 28/10/22 hasta 28/10/23, lauren01@gmail.com, servicio 1. Se obtiene el siguiente tiempo.</p> 

Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 1

PLAN_TABLE_OUTPUT							
1 Plan hash value: 289275943							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	673	10 (0)	00:00:01
7	1	NESTED LOOPS ANTI		1	673	10 (0)	00:00:01
8	2	NESTED LOOPS		1	671	9 (0)	00:00:01
9	3	TABLE ACCESS BY INDEX ROWID	TIPOS_USUARIO	1	142	0 (0)	00:00:01
10	4	INDEX UNIQUE SCAN	TIPOS_USUARIO_PK	1		0 (0)	00:00:01
11	5	TABLE ACCESS FULL	USUARIOS	1	529	9 (0)	00:00:01
12	6	VIEW PUSHED PREDICATE		1	2	1 (0)	00:00:01
13	7	INDEX RANGE SCAN	CONSUMEN_PK	1	151	1 (0)	00:00:01
14							
15							
16 Predicate Information (identified by operation id):							
17							
18							
19 4 - access("TU"."ID_TIPO"=1)							
20 5 - filter("U"."TIPO"=1 AND (:NUM_DOCUMENTO IS NULL OR							
21 "U"."NUM_DOCUMENTO"=:NUM_DOCUMENTO) AND (:TIPO_DOCUMENTO IS NULL OR							
22 "U"."TIPO_DOCUMENTO"=:TIPO_DOCUMENTO) AND (:NOMBRE IS NULL OR "U"."NOMBRE"=:NOMBRE) AND							
23 (:CORREO IS NULL OR "U"."CORREO"=:CORREO))							
24 7 - access("C"."CLIENTE"="U"."NUM_DOCUMENTO" AND "C"."SERVICIO"=TO_NUMBER(:SERVICIO))							
25 filter("C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN OR :FECHA_INICIO IS							
26 NULL AND :FECHA_FIN IS NULL)							

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1 10% de uso de CPU sin índices.

Análisis de eficacia

PLAN_TABLE_OUTPUT							
1 Plan hash value: 1205531919							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	673	2 (0)	00:00:01
7	1	NESTED LOOPS ANTI		1	673	2 (0)	00:00:01
8	2	NESTED LOOPS		1	671	1 (0)	00:00:01
9	3	TABLE ACCESS BY INDEX ROWID	TIPOS_USUARIO	1	142	0 (0)	00:00:01
10	4	INDEX UNIQUE SCAN	TIPOS_USUARIO_PK	1		0 (0)	00:00:01
11	5	TABLE ACCESS BY INDEX ROWID BATCHED	USUARIOS	1	529	1 (0)	00:00:01
12	6	INDEX RANGE SCAN	IDX_TIPO	4		0 (0)	00:00:01
13	7	VIEW PUSHED PREDICATE		1	2	1 (0)	00:00:01
14	8	INDEX RANGE SCAN	CONSUMEN_PK	1	151	1 (0)	00:00:01
15							
16							
17 Predicate Information (identified by operation id):							
18							
19							
20 4 - access("TU"."ID_TIPO"=1)							
21 5 - filter((:NUM_DOCUMENTO IS NULL OR "U"."NUM_DOCUMENTO"=:NUM_DOCUMENTO) AND (:TIPO_DOCUMENTO							
22 IS NULL OR "U"."TIPO_DOCUMENTO"=:TIPO_DOCUMENTO) AND (:NOMBRE IS NULL OR "U"."NOMBRE"=:NOMBRE)							
23 AND (:CORREO IS NULL OR "U"."CORREO"=:CORREO))							
24 6 - access("U"."TIPO"=1)							
25 8 - access("C"."CLIENTE"="U"."NUM_DOCUMENTO" AND "C"."SERVICIO"=TO_NUMBER(:SERVICIO))							
26 filter("C"."FECHA">=:FECHA_INICIO AND "C"."FECHA"<=:FECHA_FIN OR :FECHA_INICIO IS NULL AND							
27 :FECHA_FIN IS NULL)							
28							

Se redujo en un 8% el porcentaje de uso de la CPU para la consulta.

RFC11 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA AVANZADA.

Información básica del requerimiento

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Nombre del requerimiento

Consultar funcionamiento

Sentencias SQL

```
WITH Semanas (inicio_semana, fin_semana) AS (  
    SELECT (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AS inicio_semana,  
           (SELECT SYSDATE - INTERVAL '1' YEAR + 7 FROM DUAL) AS fin_semana  
    FROM DUAL  
    UNION ALL  
    SELECT fin_semana, fin_semana + 7  
    FROM Semanas  
    WHERE fin_semana <= (SELECT SYSDATE - 7 FROM DUAL)  
)  
SELECT inicio_semana, fin_semana, nombre  
FROM (  
    SELECT TS.nombre, S.inicio_semana, S.fin_semana, TS.id_tipo,  
           ROW_NUMBER() OVER (PARTITION BY S.inicio_semana, S.fin_semana ORDER BY COUNT(TS.id_tipo) DESC) AS ranking  
    FROM consumen C  
    INNER JOIN tipos_servicio TS ON TS.id_tipo = C.servicio  
    INNER JOIN Semanas S ON C.fecha BETWEEN S.inicio_semana AND S.fin_semana  
    GROUP BY TS.nombre, S.inicio_semana, S.fin_semana, TS.id_tipo  
) RankingSemanas  
WHERE ranking = 1;  
  
--RFC11.2  
WITH Semanas (inicio_semana, fin_semana) AS (  
    SELECT (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AS inicio_semana,  
           (SELECT SYSDATE - INTERVAL '1' YEAR + 7 FROM DUAL) AS fin_semana  
    FROM DUAL  
    UNION ALL  
    SELECT fin_semana, fin_semana + 7  
    FROM Semanas  
    WHERE fin_semana <= (SELECT SYSDATE - 7 FROM DUAL)  
)  
SELECT inicio_semana, fin_semana, nombre  
FROM (  
    SELECT TS.nombre, S.inicio_semana, S.fin_semana, TS.id_tipo,  
           ROW_NUMBER() OVER (PARTITION BY S.inicio_semana, S.fin_semana ORDER BY COUNT(TS.id_tipo) ASC) AS ranking  
    FROM consumen C  
    INNER JOIN tipos_servicio TS ON TS.id_tipo = C.servicio  
    INNER JOIN Semanas S ON C.fecha BETWEEN S.inicio_semana AND S.fin_semana  
    GROUP BY TS.nombre, S.inicio_semana, S.fin_semana, TS.id_tipo  
) RankingSemanas  
WHERE ranking = 1;
```

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



```
--RFC11.3
WITH Semanas (inicio_semana, fin_semana) AS (
    SELECT (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AS inicio_semana,
           (SELECT SYSDATE - INTERVAL '1' YEAR + 7 FROM DUAL) AS fin_semana
    FROM DUAL
    UNION ALL
    SELECT fin_semana, fin_semana + 7
    FROM Semanas
    WHERE fin_semana <= (SELECT SYSDATE - 7 FROM DUAL)
),
Fechas (fecha_reserva) AS (
    SELECT MIN(fecha_entrada)
    FROM reservas
    UNION ALL
    SELECT fecha_reserva + 1
    FROM Fechas
    WHERE fecha_reserva + 1 <= (SELECT MAX(fecha_salida) FROM reservas)
)
SELECT inicio_semana, fin_semana, num_habitacion
FROM (
    SELECT H.num_habitacion, S.inicio_semana, S.fin_semana,
           ROW_NUMBER() OVER (PARTITION BY S.inicio_semana, S.fin_semana ORDER BY COUNT(H.num_habitacion) DESC) AS ranking
    FROM habitaciones H
    INNER JOIN reservas R ON R.habitacion = H.num_habitacion
    INNER JOIN Fechas F ON F.fecha_reserva BETWEEN R.fecha_entrada AND R.fecha_salida
    INNER JOIN Semanas S ON F.fecha_reserva BETWEEN S.inicio_semana AND S.fin_semana
    GROUP BY H.num_habitacion, S.inicio_semana, S.fin_semana
) RankingSemanas
WHERE ranking = 1;

WITH Semanas (inicio_semana, fin_semana) AS (
    SELECT (SELECT SYSDATE - INTERVAL '1' YEAR FROM DUAL) AS inicio_semana,
           (SELECT SYSDATE - INTERVAL '1' YEAR + 7 FROM DUAL) AS fin_semana
    FROM DUAL
    UNION ALL
    SELECT fin_semana, fin_semana + 7
    FROM Semanas
    WHERE fin_semana <= (SELECT SYSDATE - 7 FROM DUAL)
),
Fechas (fecha_reserva) AS (
    SELECT MIN(fecha_entrada)
    FROM reservas
    UNION ALL
    SELECT fecha_reserva + 1
    FROM Fechas
    WHERE fecha_reserva + 1 <= (SELECT MAX(fecha_salida) FROM reservas)
)
SELECT inicio_semana, fin_semana, num_habitacion
FROM (
    SELECT H.num_habitacion, S.inicio_semana, S.fin_semana,
           RANK() OVER (PARTITION BY S.inicio_semana, S.fin_semana ORDER BY COUNT(H.num_habitacion) ASC) AS ranking
    FROM habitaciones H
    INNER JOIN reservas R ON R.habitacion = H.num_habitacion
    INNER JOIN Fechas F ON F.fecha_reserva BETWEEN R.fecha_entrada AND R.fecha_salida
    INNER JOIN Semanas S ON F.fecha_reserva BETWEEN S.inicio_semana AND S.fin_semana
    GROUP BY H.num_habitacion, S.inicio_semana, S.fin_semana
) RankingSemanas
WHERE ranking = 1;
```

Distribución de los datos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Solo se conoce el funcionamiento de un único año

Efecto de los índices

Salida de Script	Resultado de la Consulta	Resultado de la Consulta
SQL	Se han recuperado 50 filas en 1,385 segundos	
INICIO_SEMANA	FIN_SEMANA	NOMBRE
1 08/11/22	15/11/22	Restaurante
2 15/11/22	22/11/22	Planchado
3 22/11/22	29/11/22	Salon de conferencia
4 29/11/22	06/12/22	Spa
5 06/12/22	13/12/22	Supermercado

Salida de Script	Resultado de la Consulta	Resultado de la Consulta 1
SQL	Se han recuperado 50 filas en 1,238 segundos	
INICIO_SEMANA	FIN_SEMANA	NOMBRE
1 08/11/22	15/11/22	Planchado
2 15/11/22	22/11/22	Tienda ropa
3 22/11/22	29/11/22	Utensilios
4 29/11/22	06/12/22	Salon de reunion
5 06/12/22	13/12/22	Lavado
6 13/12/22	20/12/22	Spa
7 20/12/22	27/12/22	Salon de conferencia

Salida de Script	Resultado de la Consulta	Resultado de la Consulta 1
SQL	Se han recuperado 50 filas en 0,219 segundos	
INICIO_SEMANA	FIN_SEMANA	NUM_HABITACION
1 08/11/22	15/11/22	101
2 15/11/22	22/11/22	101
3 22/11/22	29/11/22	101
4 29/11/22	06/12/22	101
5 06/12/22	13/12/22	102

Salida de Script	Resultado de la Consulta	Resultado de la Consulta 1
SQL	Se han recuperado 50 filas en 0,212 segundos	
INICIO_SEMANA	FIN_SEMANA	NUM_HABITACION
1 08/11/22	15/11/22	203
2 15/11/22	22/11/22	205
3 15/11/22	22/11/22	204
4 22/11/22	29/11/22	305
5 29/11/22	06/12/22	102

Las dos primeras consultas superan un poco el tiempo esperado, las dos ultimas cumplen el requisito de eficiencia.

Planes de consulta del requerimiento

Planes de consulta obtenidos

Leidy J. Lozano

Plan de consulta 1

```

1 PLAN_TABLE_OUTPUT
2
3
4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |
5 -----
6 | 0 | SELECT STATEMENT | | | | | |
7 |* 1 | VIEW | | | | | |
8 |* 2 | WINDOW SORT PUSHED RANK | | | | | |
9 | 3 | HASH GROUP BY | | | | | |
10 | 4 | NESTED LOOPS | | | | | |
11 | 5 | NESTED LOOPS | | | | | |
12 | 6 | INDEX FULL SCAN | CONSUMEN_PK | | | | |
13 | 7 | TABLE ACCESS BY INDEX ROWID | TIPOS_SERVICIO | | | | |
14 |* 8 | INDEX UNIQUE SCAN | TIPOS_SERVICIO_PK | | | | |
15 |* 9 | VIEW | | | | | |
16 | 10 | UNION ALL (RECURSIVE WITH) BREADTH FIRST | | | | | |
17 | 11 | FAST DUAL | | | | | |
18 | 12 | FAST DUAL | | | | | |
19 | 13 | FAST DUAL | | | | | |
20 |* 14 | RECURSIVE WITH PUMP | | | | | |
21 | 15 | FAST DUAL | | | | | |
22 -----
23
24 Predicate Information (identified by operation id):
25 -----
26
27 1 - filter("RANKING"<1)
28 2 - filter(ROW_NUMBER() OVER ( PARTITION BY "S"."INICIO_SEMANA", "S"."FIN_SEMANA" ORDER BY COUNT(*) DESC
29 )<1)
30 8 - access("IS"."ID_TIPO"="C"."SERVICIO")
31 9 - filter("C"."FECHA">="S"."INICIO_SEMANA" AND "C"."FECHA"<="S"."FIN_SEMANA")
32 14 - filter("FIN_SEMANA"<= (SELECT SYSDATE@1-7 FROM "SYS"."DUAL"))

```

Plan de consulta 2

```

24 PLAN_TABLE_OUTPUT
25
26 Plan hash value: 1588149916
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
10
```

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 3

PLAN_TABLE_OUTPUT							
1 Plan hash value: 2680259668							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	44	68 (3)	00:00:01
7	* 1	VIEW		1	44	68 (3)	00:00:01
8	* 2	WINDOW SORT PUSHED RANK		1	58	68 (3)	00:00:01
9	3	HASH GROUP BY		1	58	68 (3)	00:00:01
10	4	NESTED LOOPS		1	58	66 (0)	00:00:01
11	5	NESTED LOOPS		1	40	40 (0)	00:00:01
12	6	TABLE ACCESS FULL	RESERVAS	1	31	4 (0)	00:00:01
13	* 7	VIEW		1	9	36 (0)	00:00:01
14	8	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
15	9	SORT AGGREGATE		1	9		
16	10	TABLE ACCESS FULL	RESERVAS	1	9	4 (0)	00:00:01
17	* 11	RECURSIVE WITH PUMP					
18	12	SORT AGGREGATE		1	9		
19	13	TABLE ACCESS FULL	RESERVAS	1	9	4 (0)	00:00:01
20	* 14	VIEW		1	18	26 (0)	00:00:01
21	15	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
22	16	FAST DUAL		1		2 (0)	00:00:01
23	17	FAST DUAL		1		2 (0)	00:00:01
24	18	FAST DUAL		1		2 (0)	00:00:01
25	* 19	RECURSIVE WITH PUMP					
26	20	FAST DUAL		1		2 (0)	00:00:01
27							
28							
29 Predicate Information (identified by operation id):							
30							
31							
32 1 - filter("RANKING"<=1)							
33 2 - filter(ROW_NUMBER() OVER (PARTITION BY "S"."INICIO_SEMANA", "S"."FIN_SEMANA" ORDER BY							
34 COUNT(*) DESC)<=1)							
35 7 - filter("F"."FECHA_RESERVA">="R"."FECHA_ENTRADA" AND "F"."FECHA_RESERVA"<="R"."FECHA_SALIDA")							
36 11 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS"							
37 "RESERVAS"))							
38 14 - filter("F"."FECHA_RESERVA">="S"."INICIO_SEMANA" AND "F"."FECHA_RESERVA"<="S"."FIN_SEMANA")							
39 19 - filter("FIN_SEMANA"<= (SELECT SYSDATE@!-7 FROM "SYS"."DUAL" "DUAL"))							
40							

Plan de consulta 4

PLAN_TABLE_OUTPUT							
1 Plan hash value: 2680259668							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	44	68 (3)	00:00:01
7	* 1	VIEW		1	44	68 (3)	00:00:01
8	* 2	WINDOW SORT PUSHED RANK		1	58	68 (3)	00:00:01
9	3	HASH GROUP BY		1	58	68 (3)	00:00:01
10	4	NESTED LOOPS		1	58	66 (0)	00:00:01
11	5	NESTED LOOPS		1	40	40 (0)	00:00:01
12	6	TABLE ACCESS FULL	RESERVAS	1	31	4 (0)	00:00:01
13	* 7	VIEW		1	9	36 (0)	00:00:01
14	8	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
15	9	SORT AGGREGATE		1	9		
16	10	TABLE ACCESS FULL	RESERVAS	1	9	4 (0)	00:00:01
17	* 11	RECURSIVE WITH PUMP					
18	12	SORT AGGREGATE		1	9		
19	13	TABLE ACCESS FULL	RESERVAS	1	9	4 (0)	00:00:01
20	* 14	VIEW		1	18	26 (0)	00:00:01
21	15	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
22	16	FAST DUAL		1		2 (0)	00:00:01
23	17	FAST DUAL		1		2 (0)	00:00:01
24	18	FAST DUAL		1		2 (0)	00:00:01
25	* 19	RECURSIVE WITH PUMP					
26	20	FAST DUAL		1		2 (0)	00:00:01
27							
28							
29 Predicate Information (identified by operation id):							
30							
31							
32 1 - filter("RANKING"<=1)							
33 2 - filter(RANK() OVER (PARTITION BY "S"."INICIO_SEMANA", "S"."FIN_SEMANA" ORDER BY							
34 COUNT(*)<=1)							
35 7 - filter("F"."FECHA_RESERVA">="R"."FECHA_ENTRADA" AND "F"."FECHA_RESERVA"<="R"."FECHA_SALIDA")							
36 11 - filter(INTERNAL_FUNCTION("FECHA_RESERVA")+1<= (SELECT MAX("FECHA_SALIDA") FROM "RESERVAS"							
37 "RESERVAS"))							
38 14 - filter("F"."FECHA_RESERVA">="S"."INICIO_SEMANA" AND "F"."FECHA_RESERVA"<="S"."FIN_SEMANA")							
39 19 - filter("FIN_SEMANA"<= (SELECT SYSDATE@!-7 FROM "SYS"."DUAL" "DUAL"))							
40							

Tiempos obtenidos en los planes de consulta

Tiempo plan de consulta 1

29% de uso de CPU sin índices.

Tiempo plan de consulta 2

29% de uso de CPU sin índices.

PLAN_TABLE_OUTPUT									
1 Plan hash value: 1588149916									
2									
3									
4	Id	Operation	Name	Rows	Bytes	Cost	(%CPU)	Time	
5									
6	0	SELECT STATEMENT		1	160	29	(7)	00:00:01	
7	1	VIEW		1	160	29	(7)	00:00:01	
8	2	WINDOW SORT PUSHED RANK		1	182	29	(7)	00:00:01	
9	3	HASH GROUP BY		1	182	29	(7)	00:00:01	
10	4	NESTED LOOPS		1	182	27	(0)	00:00:01	
11	5	NESTED LOOPS		1	164	1	(0)	00:00:01	
12	6	INDEX FULL SCAN	CONSUMEN_PK	1	22	1	(0)	00:00:01	
13	7	TABLE ACCESS BY INDEX ROWID	TIPOS_SERVICIO	1	142	0	(0)	00:00:01	
14	8	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	1	0	0	(0)	00:00:01	
15	9	VIEW		1	18	26	(0)	00:00:01	
16	10	UNION ALL (RECURSIVE WITH) BREADTH FIRST							
17	11	FAST DUAL		1		2	(0)	00:00:01	
18	12	FAST DUAL		1		2	(0)	00:00:01	
19	13	FAST DUAL		1		2	(0)	00:00:01	
20	14	RECURSIVE WITH PUMP							
21	15	FAST DUAL		1		2	(0)	00:00:01	
22									

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



PLAN_TABLE_OUTPUT							
1 Plan hash value: 2935339722							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	44	40 (8)	00:00:01
7	* 1	VIEW		1	44	40 (8)	00:00:01
8	* 2	WINDOW SORT PUSHED RANK		1	58	40 (8)	00:00:01
9	3	HASH GROUP BY		1	58	40 (8)	00:00:01
10	4	NESTED LOOPS		1	58	38 (3)	00:00:01
11	5	MERGE JOIN		1	40	12 (9)	00:00:01
12	6	TABLE ACCESS BY INDEX ROWID	RESERVAS	1	31	2 (0)	00:00:01
13	7	INDEX FULL SCAN	IDX_FECHA_ENTRADA	1		1 (0)	00:00:01
14	* 8	FILTER					
15	* 9	SORT JOIN		2	18	10 (10)	00:00:01
16	10	VIEW		2	18	9 (0)	00:00:01
17	11	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
18	12	SORT AGGREGATE		1	9		
19	13	INDEX FULL SCAN (MIN/MAX)	IDX_FECHA_ENTRADA	1	9	1 (0)	00:00:01
20	* 14	RECURSIVE WITH PUMP					
21	15	SORT AGGREGATE		1	9		
22	16	INDEX FULL SCAN	IDX_FECHA_ENTRADA_SALIDA	1	9	1 (0)	00:00:01
23	* 17	VIEW		1	18	26 (0)	00:00:01
24	18	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
25	19	FAST DUAL		1		2 (0)	00:00:01
26	20	FAST DUAL		1		2 (0)	00:00:01
27	21	FAST DUAL		1		2 (0)	00:00:01
28	* 22	RECURSIVE WITH PUMP					
29	23	FAST DUAL		1		2 (0)	00:00:01
30							

PLAN_TABLE_OUTPUT							
1 Plan hash value: 2935339722							
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	44	40 (8)	00:00:01
7	* 1	VIEW		1	44	40 (8)	00:00:01
8	* 2	WINDOW SORT PUSHED RANK		1	58	40 (8)	00:00:01
9	3	HASH GROUP BY		1	58	40 (8)	00:00:01
10	4	NESTED LOOPS		1	58	38 (3)	00:00:01
11	5	MERGE JOIN		1	40	12 (9)	00:00:01
12	6	TABLE ACCESS BY INDEX ROWID	RESERVAS	1	31	2 (0)	00:00:01
13	7	INDEX FULL SCAN	IDX_FECHA_ENTRADA	1		1 (0)	00:00:01
14	* 8	FILTER					
15	* 9	SORT JOIN		2	18	10 (10)	00:00:01
16	10	VIEW		2	18	9 (0)	00:00:01
17	11	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
18	12	SORT AGGREGATE		1	9		
19	13	INDEX FULL SCAN (MIN/MAX)	IDX_FECHA_ENTRADA	1	9	1 (0)	00:00:01
20	* 14	RECURSIVE WITH PUMP					
21	15	SORT AGGREGATE		1	9		
22	16	INDEX FULL SCAN	IDX_FECHA_ENTRADA_SALIDA	1	9	1 (0)	00:00:01
23	* 17	VIEW		1	18	26 (0)	00:00:01
24	18	UNION ALL (RECURSIVE WITH) BREADTH FIRST					
25	19	FAST DUAL		1		2 (0)	00:00:01
26	20	FAST DUAL		1		2 (0)	00:00:01
27	21	FAST DUAL		1		2 (0)	00:00:01
28	* 22	RECURSIVE WITH PUMP					
29	23	FAST DUAL		1		2 (0)	00:00:01
30							

Se redujo en un 24% el uso de CPU para las consultas 3 y 4.

RFC12 – DOCUMENTACIÓN REQUERIMIENTO FUNCIONAL DE CONSULTA AVANZADA.

Información básica del requerimiento

Nombre del requerimiento	Consultar los clientes excelentes
Sentencias SQL	

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



```
WITH Trimestres AS (
    SELECT
        TO_CHAR(ADD_MONTHS(SYSDATE, -12)) AS t1_inicio,
        TO_CHAR(ADD_MONTHS(SYSDATE, -9)) AS t2_inicio,
        TO_CHAR(ADD_MONTHS(SYSDATE, -6)) AS t3_inicio,
        TO_CHAR(ADD_MONTHS(SYSDATE, -3)) AS t4_inicio
    FROM DUAL
)
SELECT U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre AS tipo, COUNT(DISTINCT CASE
    WHEN R.fecha_salida BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
    WHEN R.fecha_salida BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
    WHEN R.fecha_salida BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
    WHEN R.fecha_salida BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
END) AS cuenta_reservas,
COUNT(DISTINCT CASE
    WHEN C.fecha BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
    WHEN C.fecha BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
    WHEN C.fecha BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
    WHEN C.fecha BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
END) AS cuenta_consumo,
COUNT(DISTINCT CASE
    WHEN ReS.dia_reserva BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
    WHEN ReS.dia_reserva BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
    WHEN ReS.dia_reserva BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
    WHEN ReS.dia_reserva BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
END) AS cuenta_r_servicios
FROM usuarios U
INNER JOIN tipos_usuario TU ON TU.id_tipo = U.tipo
INNER JOIN reservas R ON R.cliente_reserva = U.num_documento
INNER JOIN consumen C ON C.cliente = U.num_documento
INNER JOIN tipos_servicio TS ON TS.id_tipo = C.servicio
INNER JOIN reservas_servicio ReS ON ReS.cliente = U.num_documento
INNER JOIN Trimestres T ON (((R.fecha_entrada BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3)) AND (R.fecha_salida BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3)))
    OR ((R.fecha_entrada BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3)) AND (R.fecha_salida BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3)))
    OR ((R.fecha_entrada BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3)) AND (R.fecha_salida BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3)))
    OR ((R.fecha_entrada BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3)) AND (R.fecha_salida BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3))))
    AND (C.fecha BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3)
    OR C.fecha BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3)
    OR C.fecha BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3)
    OR C.fecha BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3))
    AND (ReS.dia_reserva BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3)
    OR ReS.dia_reserva BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3)
    OR ReS.dia_reserva BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3)
    OR ReS.dia_reserva BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3))
311 INNER JOIN servicios S ON C.servicio = S.tipo
312 WHERE S.precio > 300000 AND (TO_NUMBER(SUBSTR(ReS.hora_fin, 1, 2)) - TO_NUMBER(SUBSTR(ReS.hora_inicio, 1, 2)) > 4) AND U.tipo = 1
313 GROUP BY U.num_documento, U.tipo_documento, U.nombre, U.correo, TU.nombre
314 HAVING
315 COUNT(DISTINCT CASE
316 WHEN R.fecha_salida BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
317 WHEN R.fecha_salida BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
318 WHEN R.fecha_salida BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
319 WHEN R.fecha_salida BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
320 END) = 4
321 AND
322 COUNT(DISTINCT CASE
323 WHEN C.fecha BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
324 WHEN C.fecha BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
325 WHEN C.fecha BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
326 WHEN C.fecha BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
327 END) = 4
328 AND
329 COUNT(DISTINCT CASE
330 WHEN ReS.dia_reserva BETWEEN t1_inicio AND ADD_MONTHS(t1_inicio, 3) THEN 1
331 WHEN ReS.dia_reserva BETWEEN t2_inicio AND ADD_MONTHS(t2_inicio, 3) THEN 2
332 WHEN ReS.dia_reserva BETWEEN t3_inicio AND ADD_MONTHS(t3_inicio, 3) THEN 3
333 WHEN ReS.dia_reserva BETWEEN t4_inicio AND ADD_MONTHS(t4_inicio, 3) THEN 4
334 END) = 4
335 ORDER BY U.num_documento;
```

Distribución de los datos

Solo se tiene la información de un único año, luego solo se obtienen los clientes de ese año en específico.

Efecto de los índices

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x Resultado de la Consulta 2 x Resultado de la Consulta 3 x Resultado de la Consulta 4 x						
Todas las Filas Recuperadas: 3 en 0,919 segundos						
NUM_DOCUMENTO	TIPO_DOCUMENTO	NOMBRE	CORREO	TIPO	CUENTA_RESERVAS	CUENTA_CONSUMO
1 1006	CC	Laura	l.restrepop@uniandes.edu.co	Cliente	4	4
2 1007	CC	Catalina	c.espitia@uniandes.edu.co	Cliente	4	4
3 1008	CC	Sergio	s.cifuentes@uniandes.edu.co	Cliente	4	4

La consulta no cumple el requerimiento de eficiencia, pero lo hace por una diferencia de tiempo realmente pequeña (0,119 s)

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Planes de consulta del requerimiento

Planes de consulta obtenidos

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



Plan de consulta 1

PLAN_TABLE_OUTPUT

1 Plan hash value: 503306589

2

3

4 | Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |

5

6 | 0 | SELECT STATEMENT | | | | | 13 (16) | 00:00:01 |

7 | 1 | SORT ORDER BY | | | | | 13 (16) | 00:00:01 |

8 | 2 | FILTER | | | | | | |

9 | 3 | SORT GROUP BY | | | | | 13 (16) | 00:00:01 |

10 | 4 | HASH JOIN | | | | | 11 (0) | 00:00:01 |

11 | 5 | NESTED LOOPS SEMI | | | | | 7 (0) | 00:00:01 |

12 | 6 | NESTED LOOPS SEMI | | | | | 7 (0) | 00:00:01 |

13 | 7 | NESTED LOOPS | | | | | 7 (0) | 00:00:01 |

14 | 8 | NESTED LOOPS | | | | | 6 (0) | 00:00:01 |

15 | 9 | NESTED LOOPS | | | | | 6 (0) | 00:00:01 |

16 | 10 | NESTED LOOPS | | | | | 2 (0) | 00:00:01 |

17 | 11 | FAST DUAL | | | | | 2 (0) | 00:00:01 |

18 | 12 | TABLE ACCESS BY INDEX ROWID | TIPOS_USUARIO | 1 | 142 | 0 (0) | 00:00:01 |

19 | 13 | INDEX UNIQUE SCAN | TIPOS_USUARIO_PK | 1 | | 0 (0) | 00:00:01 |

20 | 14 | TABLE ACCESS FULL | RESERVAS_SERVICIO | 1 | 396 | 4 (0) | 00:00:01 |

21 | 15 | TABLE ACCESS BY INDEX ROWID | USUARIOS | 1 | 529 | 0 (0) | 00:00:01 |

22 | 16 | INDEX UNIQUE SCAN | USUARIOS_PK | 1 | | 0 (0) | 00:00:01 |

23 | 17 | INDEX RANGE SCAN | CONSUMEN_PK | 1 | 151 | 1 (0) | 00:00:01 |

24 | 18 | TABLE ACCESS BY INDEX ROWID | SERVICIOS | 1 | 26 | 0 (0) | 00:00:01 |

25 | 19 | INDEX UNIQUE SCAN | SERVICIOS_PK | 1 | | 0 (0) | 00:00:01 |

26 | 20 | INDEX UNIQUE SCAN | TIPOS_SERVICIO_PK | 29 | 377 | 0 (0) | 00:00:01 |

27 | 21 | TABLE ACCESS FULL | RESERVAS | 1 | 147 | 4 (0) | 00:00:01 |

28

29

30 Predicate Information (identified by operation id):

31

32

33 2 - filter(COUNT(DISTINCT CASE WHEN ("R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))))

34 AND "R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))),3)) THEN 1 WHEN

35 ("R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9)))) AND

36 "R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))),3)) THEN 2 WHEN

37 ("R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6)))) AND

38 "R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))),3)) THEN 3 WHEN

39 ("R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3)))) AND

40 "R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))),3)) THEN 4 END)=4 AND

41 COUNT(DISTINCT CASE WHEN ("C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12)))) AND

42 ("C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))),3)) THEN 1 WHEN

43 ("C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9)))) AND

44 ("C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))),3)) THEN 2 WHEN

45 ("C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6)))) AND

46 ("C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))),3)) THEN 3 WHEN

47 ("C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3)))) AND

48 ("C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))),3)) THEN 4 END)=4 AND COUNT(DISTINCT

49 CASE WHEN ("RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12)))) AND

50 ("RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))),3)) THEN 1 WHEN

51 ("RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9)))) AND

52 ("RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))),3)) THEN 2 WHEN

53 ("RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6)))) AND

54 ("RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))),3)) THEN 3 WHEN

55 ("RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3)))) AND

56 ("RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))),3)) THEN 4 END)=4)

57 4 - access("R"."CLIENTE_RESERVA"="U"."NUM_DOCUMENTO")

58 13 - access("TU"."ID_TIPO"=1)

59 14 - filter(TO_NUMBER(SUBSTR("RES"."HORA_FIN",1,2))-TO_NUMBER(SUBSTR("RES"."HORA_INICIO",1,2))>4

60 AND ("RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12)))) AND

61 "RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))),3) OR

62 "RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9)))) AND

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



	63	"RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))), 3) OR
	64	"RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))) AND
	65	"RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))), 3) OR
	66	"RES"."DIA_RESERVA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))) AND
	67	"RES"."DIA_RESERVA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))), 3))
	68	15 - filter("U"."TIPO"=1)
	69	16 - access("RES"."CLIENTE"="U"."NUM_DOCUMENTO")
	70	17 - access("C"."CLIENTE"="U"."NUM_DOCUMENTO")
	71	filter("C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))) AND
	72	"C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))), 3) OR
	73	"C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))) AND
	74	"C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))), 3) OR
	75	"C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))) AND
	76	"C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))), 3) OR
	77	"C"."FECHA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))) AND
	78	"C"."FECHA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))), 3))
	79	18 - filter("S"."PRECIO">300000)
	80	19 - access("C"."SERVICIO"="S"."TIPO")
	81	20 - access("TS"."ID_TIPO"="C"."SERVICIO")
	82	21 - filter("R"."FECHA_ENTRADA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))) AND
	83	"R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))) AND
	84	"R"."FECHA_ENTRADA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))), 3) AND
	85	"R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-12))), 3) OR
	86	"R"."FECHA_ENTRADA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))) AND
	87	"R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))) AND
	88	"R"."FECHA_ENTRADA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))), 3) AND
	89	"R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-9))), 3) OR
	90	"R"."FECHA_ENTRADA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))) AND
	91	"R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))) AND
	92	"R"."FECHA_ENTRADA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))), 3) AND
	93	"R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-6))), 3) OR
	94	"R"."FECHA_ENTRADA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))) AND
	95	"R"."FECHA_SALIDA">=TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))) AND
	96	"R"."FECHA_ENTRADA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))), 3) AND
	97	"R"."FECHA_SALIDA"<=ADD_MONTHS(TO_CHAR(ADD_MONTHS(SYSDATE@!, (-3))), 3))
	98	
Tiempos obtenidos en los planes de consulta		
<i>Tiempo plan de consulta 1</i>	13 % de uso de CPU sin índices.	

Análisis de eficacia

PLAN_TABLE_OUTPUT							
1	Plan hash value: 3101922544						
2							
3							
4	Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
5							
6	0	SELECT STATEMENT		1	1404	13 (16)	00:00:01
7	1	SORT ORDER BY		1	1404	13 (16)	00:00:01
8	* 2	FILTER					
9	3	SORT GROUP BY		1	1404	13 (16)	00:00:01
10	* 4	HASH JOIN		1	1404	11 (0)	00:00:01
11	5	NESTED LOOPS SEMI		1	1257	7 (0)	00:00:01
12	6	NESTED LOOPS SEMI		1	1244	7 (0)	00:00:01
13	7	NESTED LOOPS		1	1218	7 (0)	00:00:01
14	8	NESTED LOOPS		1	1067	6 (0)	00:00:01
15	9	NESTED LOOPS		1	538	6 (0)	00:00:01
16	10	NESTED LOOPS		1	142	2 (0)	00:00:01
17	11	FAST DUAL		1		2 (0)	00:00:01
18	12	TABLE ACCESS BY INDEX ROWID	TIPOS_USUARIO	1	142	0 (0)	00:00:01
19	* 13	INDEX UNIQUE SCAN	TIPOS_USUARIO_PK	1		0 (0)	00:00:01
20	* 14	TABLE ACCESS FULL	RESERVAS_SERVICIO	1	396	4 (0)	00:00:01
21	* 15	TABLE ACCESS BY INDEX ROWID	USUARIOS	1	529	0 (0)	00:00:01
22	* 16	INDEX UNIQUE SCAN	USUARIOS_PK	1		0 (0)	00:00:01
23	* 17	INDEX RANGE SCAN	CONSUMEN_PK	1	151	1 (0)	00:00:01
24	* 18	TABLE ACCESS BY INDEX ROWID BATCHED	SERVICIOS	1	26	0 (0)	00:00:01
25	* 19	INDEX RANGE SCAN	IDX_PRECIO	1		0 (0)	00:00:01
26	* 20	INDEX UNIQUE SCAN	TIPOS_SERVICIO_PK	29	377	0 (0)	00:00:01
27	* 21	TABLE ACCESS FULL	RESERVAS	1	147	4 (0)	00:00:01
28							

No se redujo el tiempo de CPU a pesar de que se utilizaron los índices en la consulta.

En esta sección, se describirá superficialmente el proceso de diseño, generación y población masiva. Para lograr el volumen de datos utilizados se hizo uso de las librerías Faker y random de Python, además de una corta colaboración de ChatGPT. Haciendo uso de los proveedores personalizados de faker, se crearon funciones dentro de un proveedor enfocado al hotel de los Alpes, estas funciones permitían diligenciar aleatoriamente los campos solicitados en las tablas.

En la captura es posible apreciar varias funciones de las anteriormente mencionadas, entre las más relevantes se encuentran

```
def numero_identidad(self):
    return f.Unique.bathify(text="#####")
def tipo_documento_usuario(self):
    return random.choice(['CC', 'CE'])
def tipo_documento_acompaniante(self):
    return random.choice(['CC', 'CE', 'TI'])
def fecha_inicial_reserva(self):
    fecha_inicio = f.date_between_dates(datetime.date(year=2022, month=10, day=28), datetime.date(year=2023, month=10, day=28))
    return fecha_inicio
def plan_consumo(self):
    return f.random_int(min=1, max=5)
```

- La función de generación de números de identidad, que generaba un entero aleatorio de 10 cifras.
- La función de generación de fechas de reserva, que generaba un objeto del tipo datetime entre las fechas ingresadas por parámetro.
- La función de planes de consumo hace uso del método random int de faker, si bien no se aprecia en las fotos, este fue usado para generar los id de otras relaciones.
- En colaboración de la librería random, un método que permitía escoger un tipo de documento aleatorio.

Incluso con todas las cosas mencionadas anteriormente, es necesario recalcar que las capacidades de generación de los proveedores por defecto de faker es limitada, por lo que se tuvo que recorrer a ChatGPT para la generación de cosas más específicas, en particular nombres de cosas.

```
f = Faker(['en_US'])
dotacion_list = ['Not-so-flat Screen', 'Simple bed', 'Coffee machine', 'Wool sheets', 'Mini refrigerator', 'Flat-screen TV', 'Microwave oven']
productos_list = ['Apples', 'Bananas', 'Oranges', 'Strawberries', 'Blueberries', 'Raspberries', 'Grapes', 'Pineapples', 'Watermelons', 'Candies']
floors_rooms = [101, 102, 103, 104, 105, 201, 202, 203, 204, 205, 301, 302, 303, 304, 305]
```

Tanto la lista de dotaciones como la de productos es limitada y relativamente pequeña en comparación a los volúmenes de datos generados por faker, esto debido que la IA tenía menor capacidad de generación única de cadenas de texto (En general, no superaba 150 nombres sin empezar a repetir información) Por último, y como dato curioso, en esta imagen se aprecia que, para la información que aplica (por ejemplo, nombres), Faker se basó geográficamente en estados unidos. Esto se hizo con el propósito de mantener un estilo más o menos constante en este tipo de datos.

Finalmente, con el proveedor configurado y demás datos creados, fue posible empezar a escribir el archivo de población en formato txt. Este proceso fue realizado con simples ciclos for que escribían línea a línea la información requerida.

Entrega 2 – Optimización de consultas

Laura M. Restrepo

Karen T. Vera

Leidy J. Lozano



```
file.write('--Usuarios'+'\n')
usuarios = 'INSERT INTO usuarios (num_documento, tipo_documento, nombre, correo, tipo) VALUES ('
num_identidad = []
num_empleado_2 = []

for _ in range(50000):
    num_id = f.numero_identidad()
    num_identidad.append(num_id)
    entry = (usuarios+""+str(num_id)+""+f.tipo_documento_usuario()+""+f.name()+""+f.ascii_free_email()+""+str(f.random_int(
```

En esta foto, también se logran apreciar funcionalidades de los proveedores básicos de faker, como `f.name()`, que genera un nombre y un apellido. De esta captura es posible resaltar la lista a la cual se añaden los documentos de identidad, si bien la mayoría de datos es aleatorio (y en varias ocasiones, sin sentido), se respetaron las reglas de negocio, por eso:

- Los números de identidad creados por faker eran añadidos a una lista, luego, en la creación de reservas de servicio o consumos, se seleccionaba un número al azar de esta lista para reflejar a la persona que había reservado o consumido el servicio, asegurando que fuese un usuario previamente creado.
- La finalización de reservas siempre debe ser superior a su inicio, por eso, tras obtener una fecha inicial para la reserva, simplemente se añadían cinco días a esa fecha para simbolizar la fecha final. Esta tarea no se delegó a faker debido a la posibilidad de que se generaran incoherencias o fechas problemáticas (Una reserva que dure meses, bloqueando innecesariamente la habitación)
- Algo similar se hizo con las horas de finalización de las reservas de los servicios, solo que estas tenían cierta variedad de duraciones.
- Adicionalmente, se crearon unos pocos usuarios con el rol de empleado. De esta lista se tomaron aleatoriamente sus elementos para simbolizar el atributo de “Registrado por”, presente en consumos. Esto no quiere decir que haya solo unos pocos empleados, solo que se tomaron exclusivamente de esta lista porque se tenía la certeza de que eran empleados, es posible que haya cientos o miles de empleados que se generaron con la población masiva de la relación usuarios, pero no se consideraron.
- El estado de una reserva se definía según una función que comparaba la información con la fecha del presente día. Debido a que los datos llegan hasta el 28 de octubre, ya todas las reservas han finalizado.

Tras considerar estos supuestos, se considera terminado el proceso de diseño y población de la base.

ELEMENTOS COMPLEMENTARIOS

Para finalizar, es importante mencionar que la información presente en este informe se apoya y se complementa con otros documentos adicionales, los cuales pueden ser encontrados en la carpeta *docs* del repositorio, y son los siguientes:

- Modelos en formato fuente, PDF y PNG
- Archivos SQL (Esquemas, poblaciones y consultas)
- Documentación de los índices.
- Documentación de los requerimientos funcionales
- Documentación del proyecto de SW
- Documento de Excel con el esquema de contribución.