

## DOCUMENTO ANÁLISIS REQUERIMIENTOS FUNCIONALES DE CONSULTA ITERACIÓN 3 - GRUPO C07

### RFC 9

Dado que se quiere conocer la información de los clientes que consumieron al menos una vez un servicio determinado en un rango de fechas, se realiza un INNER JOIN entre las tablas CLIENTES y RESERVAS\_SERVICIOS sobre la identificación del cliente. Adicionalmente, se tiene en cuenta que el servicio prestado tenga un ID específico y que su fecha corresponda a un rango determinado. Además, se agrupan los datos por identificación del cliente para que se pueda hacer un COUNT y ver la cantidad de reservas de servicios que registra dicho cliente. Finalmente, para dar cumplimiento al requerimiento de que el usuario que realiza la consulta puede decidir cómo filtrar la información entonces se usa un ORDER BY para ordenar de acuerdo a la cifra de NUM\_RESERVAS.

Como se observa, el plan de ejecución comienza con un select statement dado que la operación a realizar es una consulta. Seguidamente se realiza un HASH JOIN para unir las coincidencias entre la tabla RESERVAS\_SERVICIOS Y CLIENTES. Este tipo de join es el más adecuado puesto que se está buscando coincidencias de tipo numérico y específico, específicamente, la llave primaria de CLIENTES (ID). El Access Predicate corresponde a la coincidencia entre el ID del cliente en ambas tablas. Se realiza un TABLE ACCESS FULL sobre RESERVAS\_SERVICIOS puesto que es necesario acceder a toda la tabla, teniendo en cuenta los FILTER PREDICATES, es decir, que el ID de cliente del servicio de la reserva coincida con el que se está buscando, además que la fecha de inicio y fin esté dentro del rango de fechas buscado.

```
/*  
EL WHERE DE ESTE ARCHIVO ES PARA DEMOSTRAR LOS DATOS, EN EL  
PROGRAMA SON VARIABLES.  
*/  
SELECT c.ID AS CEDULA, COUNT(r.ID) as NUM_RESERVAS  
FROM CLIENTES c INNER JOIN RESERVAS_SERVICIOS r ON c.ID =  
r.ID_CLIENTE  
WHERE r.ID_SERVICIO = 10  
AND r.HORA_APERTURA >= '23/05/2019'  
AND r.HORA_CIERRE <= '28/05/2019'  
GROUP BY c.ID  
ORDER BY NUM_RESERVAS ASC;
```

Juan David Díaz Cristancho - 201729408

Luis Miguel Gómez Londoño - 201729597

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3307	400
SORT		ORDER BY	3307	400
HASH		GROUP BY	3307	400
HASH JOIN			3349	398
Access Predicates				
C.ID=R.ID_CLIENTE				
TABLE ACCESS	RESERVAS_SERVICIOS	FULL	3349	309
Filter Predicates				
AND				
R.ID_SERVICIO=10				
R.HORA_CIERRE<=TO_DATE(' 2019-05-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
R.HORA_APERTURA>=TO_DATE(' 2019-05-23 00:00:00', 'yyyy-mm-dd hh24:mi:ss')				
INDEX	RFC91	FAST FULL SCAN	142857	88

## RFC 10

Dado que se quiere conocer únicamente todos los datos que NO se encuentran en los resultados de la consulta RFC10, basta con usar la función not in. Dado que lo que se solicita es el complemento de RFC9, el proceso de selección de índices para la consulta actual es nulo. (ver análisis RFC 9 para profundizar en selección de índices).

Como se observa, la operación es un select statement, dado que es una consulta. La agrupación (GROUP BY) se hace por una única columna que corresponde con ID del cliente. El software decide inteligentemente realizar un HASH JOIN RIGHT ANTI, es lógico usar este tipo de join puesto que se está haciendo el join por la llave primaria de CLIENTES (ID que es tipo number), que corresponde a un valor específico y no un rango, por lo cual es ideal usar Hash. Además, se usa un RIGHT ANTI puesto que se está usando la función not in(), la cual satisface la necesidad de usar todo lo opuesto a una consulta.

Para obtener los registros sobre los cuales se va a hacer el RIGHT ANTI, es decir, para obtener los resultados de RFC9. **Para suplir el RFC9:** Dado que se quiere conocer la información de los clientes que consumieron al menos una vez un servicio determinado en un rango de fechas, se realizar un INNER JOIN entre las tablas CLIENTES y RESERVAS\_SERVICIOS sobre la identificación del cliente. Adicionalmente, se tiene en cuenta que el servicio prestado tenga un ID específico y que su fecha corresponda a un rango determinado. Además, se agrupan los datos por identificación del cliente para que se pueda hacer un COUNT y ver la cantidad de reservas de servicios que registra dicho cliente. Finalmente, para dar cumplimiento al requerimiento de que el usuario que realiza la consulta puede decidir cómo filtrar la información entonces se usa un ORDER BY para ordenar de acuerdo a la cifra de NUM\_RESERVAS.

Como se observa, el plan de ejecución comienza con un select statement dado que la operación a realizar es una consulta. Seguidamente se realiza un HASH JOIN para unir las coincidencias entre la tabla RESERVAS\_SERVICIOS Y

CLIENTES. Este tipo de join es el más adecuado puesto que se está buscando coincidencias de tipo numérico y específico, específicamente, la llave primaria de CLIENTES (ID). El Access Predicate corresponde a la coincidencia entre el ID del cliente en ambas tablas. Se realiza un TABLE ACCESS FULL sobre RESERVAS\_SERVICIOS puesto que es necesario acceder a toda la tabla, teniendo en cuenta los FILTER PREDICATES, es decir, que el ID de cliente del servicio de la reserva coincida con el que se está buscando, además que la fecha de inicio y fin esté dentro del rango de fechas buscado.

```

/*
EL WHERE DE ESTE ARCHIVO ES PARA DEMOSTRAR LOS DATOS, EN EL
PROGRAMA SON VARIABLES.
*/
SELECT ID
FROM CLIENTES
WHERE ID NOT IN
(SELECT C.ID FROM CLIENTES C INNER JOIN RESERVAS_SERVICIOS R ON
C.ID = R.ID_CLIENTE
WHERE R.ID_SERVICIO =7 AND R.HORA_APERTURA >= '25/05/2018'
AND R.HORA_CIERRE <= '28/05/2019')
AND ROWNUM<126
GROUP BY ID;

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	
SELECT STATEMENT				1	401
HASH				1	401
COUNT		GROUP BY		1	401
Filter Predicates		STOPKEY			
ROWNUM<126					
HASH JOIN		RIGHT ANTI		1	400
Access Predicates					
ID=ID					
VIEW	SYS.VW_NSO_1			6715	398
HASH JOIN		SEMI		6715	398
Access Predicates					
C.ID=R.ID_CLIENTE					
TABLE ACCESS	RESERVAS_SERVICIOS	FULL		6715	309
Filter Predicates					
AND					
R.ID_SERVICIO=7					
R.HORA_CIERRE<=TO_DATE(' 2019-05-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
R.HORA_APERTURA>=TO_DATE(' 2018-05-25 00:00:00', 'yyyy-mm-dd hh24:mi:ss')					
R.ID_CLIENTE>0					
INDEX	REC91	FAST FULL SCAN		142857	89
Filter Predicates					
C.ID>0					
INDEX	REC91	FAST FULL SCAN		132	2

Fue muy difícil la creación de esta tabla ya que requirió mucha investigación en documentación y en errores comunes de ciertos casos SQL que se presentaban en este caso. Inicialmente lo que aparentaba ser muy difícil era mostrar casi 6 datos de dos tablas diferentes en una sola ejecución y script, al fin eso terminó siendo lo más fácil de hacer. En la elaboración de este script se vio inicialmente que no se podía hacer `group by` función de otra, en este caso (`max(count)`). Esto se solucionó entonces usando el pedazo con el `count` como *subquery* y crear un padre que si tuviera el `max`. El siguiente problema que se presentó fue que se quería sacar la tupla del máximo que se calculaba y SQL no permite ver la tupla de donde proviene el `max`, solo se hace el cálculo pero no registra su procedencia. Siendo así, se tenía que usar tablas/subqueries ya utilizadas nuevamente, pero al ver que los sinónimos de tablas no funcionan para las tablas padre, nos vimos obligados a copiar y pegar la tabla utilizada anteriormente, una y otra vez.

Soprendentemente este request no tomaba nada de tiempo, si bien solo realiza operaciones y subqueries no requería hacer JOINS con otras tablas si no con ella misma. El plan de ejecución consiste principalmente en HASH JOINS y FILTER PREDICATES. Entre más adentro, menos costo, nuevamente, debido a que cada vez se procesa un subquery más pequeño.

```

/*
EL WHERE DE ESTE ARCHIVO ES PARA DEMOSTRAR LOS DATOS, EN EL PROGRAMA SON
VARIABLES.
*/
SELECT p.SEMANA, p.ID_SERVICIO_MAX, p.MAXIMO, p.ID_SERVICIO_MIN, p.MINIMO,
d.ID_HABITACION_MAX, d.MAXIMO AS MAX_HABITACION, d.ID_HABITACION_MIN, d.MINIMO
AS MIN_HABITACION
FROM (SELECT w.SEMANA, w.SERVICIOS AS ID_SERVICIO_MAX, w.MAXIMO, j.SERVICIOS
AS ID_SERVICIO_MIN, j.MINIMO
FROM (SELECT z.SEMANA, l.SERVICIOS, z.MAXIMO
FROM (SELECT SEMANA, MAX(CUENTA) MAXIMO
FROM (SELECT to_char(r.HORA_APERTURA, 'WW') SEMANA, r.ID_SERVICIO SERVICIOS,
COUNT(ID_SERVICIO) CUENTA
FROM RESERVAS_SERVICIOS r
GROUP BY to_char(r.HORA_APERTURA, 'WW'), r.ID_SERVICIO
ORDER BY SEMANA ASC) tabla
GROUP BY SEMANA) z, (SELECT to_char(r.HORA_APERTURA, 'WW') SEMANA,
r.ID_SERVICIO SERVICIOS, COUNT(ID_SERVICIO) CUENTA
FROM RESERVAS_SERVICIOS r
GROUP BY to_char(r.HORA_APERTURA, 'WW'), r.ID_SERVICIO
ORDER BY SEMANA ASC) l
WHERE z.MAXIMO = l.CUENTA) w, (SELECT z.SEMANA, l.SERVICIOS, z.MINIMO
FROM (SELECT SEMANA, MIN(CUENTA) MINIMO
FROM (SELECT to_char(r.HORA_APERTURA, 'WW') SEMANA, r.ID_SERVICIO SERVICIOS,
COUNT(ID_SERVICIO) CUENTA

```

Juan David Díaz Cristancho - 201729408

Luis Miguel Gómez Londoño - 201729597

```
FROM RESERVAS_SERVICIOS r
GROUP BY to_char(r.HORA_APERTURA,'WW'), r.ID_SERVICIO
ORDER BY SEMANA ASC) tabla
GROUP BY SEMANA) z, (SELECT to_char(r.HORA_APERTURA,'WW') SEMANA,
r.ID_SERVICIO SERVICIOS, COUNT(ID_SERVICIO) CUENTA
FROM RESERVAS_SERVICIOS r
GROUP BY to_char(r.HORA_APERTURA,'WW'), r.ID_SERVICIO
ORDER BY SEMANA ASC) l
WHERE z.MINIMO = l.CUENTA) j
WHERE w.SEMANA = j.SEMANA) p, (SELECT x.SEMANA, x.HABITACIONES AS
ID_HABITACION_MAX, x.MAXIMO, n.HABITACIONES AS ID_HABITACION_MIN, n.MINIMO
FROM (SELECT v.SEMANA, m.HABITACIONES, v.MAXIMO
FROM (SELECT SEMANA, MAX(CUENTA) MAXIMO
FROM (SELECT to_char(c.FECHA_INICIO,'WW') SEMANA, c.ID_HABITACION
HABITACIONES, COUNT(ID_HABITACION) CUENTA
FROM RESERVAS_HABITACIONES c
GROUP BY to_char(c.FECHA_INICIO,'WW'), c.ID_HABITACION
ORDER BY SEMANA ASC) a
GROUP BY SEMANA) v, (SELECT to_char(c.FECHA_INICIO,'WW') SEMANA,
c.ID_HABITACION HABITACIONES, COUNT(ID_HABITACION) CUENTA
FROM RESERVAS_HABITACIONES c
GROUP BY to_char(c.FECHA_INICIO,'WW'), c.ID_HABITACION
ORDER BY SEMANA ASC) m
WHERE v.MAXIMO = m.CUENTA) x, (SELECT v.SEMANA, m.HABITACIONES, v.MINIMO
FROM (SELECT SEMANA, MIN(CUENTA) MINIMO
FROM (SELECT to_char(c.FECHA_INICIO,'WW') SEMANA, c.ID_HABITACION
HABITACIONES, COUNT(ID_HABITACION) CUENTA
FROM RESERVAS_HABITACIONES c
GROUP BY to_char(c.FECHA_INICIO,'WW'), c.ID_HABITACION
ORDER BY SEMANA ASC) a
GROUP BY SEMANA) v, (SELECT to_char(c.FECHA_INICIO,'WW') SEMANA,
c.ID_HABITACION HABITACIONES, COUNT(ID_HABITACION) CUENTA
FROM RESERVAS_HABITACIONES c
GROUP BY to_char(c.FECHA_INICIO,'WW'), c.ID_HABITACION
ORDER BY SEMANA ASC) m
WHERE v.MINIMO = m.CUENTA) n
WHERE x.SEMANA = n.SEMANA) d
WHERE p.SEMANA = d.SEMANA;
```

Juan David Díaz Cristancho - 201729408

Luis Miguel Gómez Londoño - 201729597

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY
SELECT STATEMENT			454212295
HASH JOIN			454212295
Access Predicates V.MINIMO=M.CUENTA			
VIEW			6789
SORT		GROUP BY	6789
INDEX	RFC112	FAST FULL SCAN	201145
HASH JOIN			6690415
Access Predicates V.SEMANA=V.SEMANA			
VIEW			6789
HASH		GROUP BY	6789
VIEW			6789
HASH		GROUP BY	6789
INDEX	RFC112	FAST FULL SCAN	201145
HASH JOIN			98548
Access Predicates V.MAXIMO=M.CUENTA			
HASH JOIN			1452
Access Predicates Z.SEMANA=V.SEMANA			
HASH JOIN			21
Access Predicates Z.MINIMO=L.CUENTA			
HASH JOIN			31
Access Predicates Z.SEMANA=Z.SEMANA			
HASH JOIN			46
Access Predicates Z.MAXIMO=L.CUENTA			
VIEW			68
HASH		GROUP BY	68
VIEW			68
HASH		GROUP BY	68
INERFC111		FAST FULL SCAN	200145
VIEW			68
SORT		GROUP BY	68
INDEX	RFC111	FAST FULL SCAN	200145
VIEW			68
HASH		GROUP BY	68
VIEW			68
HASH		GROUP BY	68
INDEXRFC111		FAST FULL SCAN	200145
VIEW			68
SORT		GROUP BY	68
INDEX	RFC111	FAST FULL SCAN	200145
VIEW			6789
HASH		GROUP BY	6789
VIEW			6789
HASH		GROUP BY	6789
INDEX	RFC112	FAST FULL SCAN	201145
VIEW			6789
SORT		GROUP BY	6789
INDEX	RFC112	FAST FULL SCAN	201145
Other XML			

RFC 12

Dado que se desea conocer los clientes que cumplen con alguna de varias características, es necesario, se realiza un INNER JOIN (no se hace uso explícito de INNER JOIN, pero se logra por medio de separación de comas). Específicamente se realizan 3 JOINS, el primero es de CLIENTES y RESERVAS\_HABITACIONES POR EL ID del cliente, segundo, entre GASTOS y PRODUCTOS sobre el ID del producto, finalmente, entre GASTOS y RESERVA\_HABITACION por el id de la reserva. Además, se tiene en cuenta que el cliente haya realizado check-in y check-out o que los el producto registrado en el gasto sea mayor a 300 000 o que haya consumido un servicio SPA o SALA DE REUNIÓN.

Se agrupa por el nombre del cliente y por el id del mismo, además, se ordena por la cantidad de servicios consumidos por ese cliente que cumple dichas condiciones, el cual se calcula con COUNT(R.ID).

Primero, el plan comienza con un select statement, puesto que es una consulta. Seguidamente se realiza un NESTED LOOP JOIN entre la tabla CLIENTES y el HASH JOIN resultante entre GASTOS y el NESTED LOOP JOIN entre PRODUCTOS Y RESERVAS\_HABITACIONES. Es importante mencionar que en el NESTED LOOP JOIN más interior se tiene en cuenta los Filter Predicates que revisan que las condiciones que se buscan en la consulta como el estado del check in y check out de la reserva, además del ID del producto y el valor. El HASH JOIN se realiza sobre GASTO Y RESERVAS\_HABITACIONES por ID y GASTOS y PRODUCTOS por ID de producto.

```
/*  
EL WHERE DE ESTE ARCHIVO ES PARA DEMOSTRAR LOS DATOS, EN EL  
PROGRAMA SON VARIABLES.  
*/  
  
SELECT C.NOMBRE,C.ID, COUNT(R.ID) AS SERVICIOS_CONSUMIDOS  
FROM CLIENTES C, RESERVAS_HABITACIONES R, GASTOS G, PRODUCTOS P  
WHERE C.ID=R.ID_CLIENTE  
AND G.ID_RESERVA_HABITACION=R.ID  
AND G.ID_PRODUCTO=P.ID  
AND ((R.CHECKED_IN=1 AND R.CHECKED_OUT=1) OR (P.VALOR>300000) OR  
(P.ID_SERVICIO=8 OR P.ID_SERVICIO=11))  
AND ROWNUM<126  
GROUP BY C.NOMBRE,C.ID  
ORDER BY COUNT(R.ID) DESC;
```

Juan David Díaz Cristancho - 201729408

Luis Miguel Gómez Londoño - 201729597

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				125
SORT		ORDER BY		125
HASH		GROUP BY		125
COUNT		STOPKEY		
Filter Predicates ROWNUM<126				
NESTED LOOPS				176
NESTED LOOPS				176
HASH JOIN				176
Access Predicates AND G.ID_RESERVA_HABITACION=R.ID G.ID_PRODUCTO=P.ID				
TABLE ACCESS GASTOS		FULL		245599
NESTED LOOPS				2048
TABLE ACCESS PRODUCTOS		FULL		20
TABLE ACCESS RESERVAS_HABITACIONES		FULL		11960
Filter Predicates OR AND R.CHECKED_IN=1 R.CHECKED_OUT=1 P.VALOR>300000 P.ID_SERVICIO=8 P.ID_SERVICIO=11				
INDEX REC91		RANGE SCAN		1
Access Predicates C.ID=R.ID_CLIENTE				
TABLE ACCESS CLIENTES		BY INDEX ROWID		1