## Análisis y diseño de la aplicación

Después de leer los nuevos requerimientos, se analizó su impacto en la aplicación, se concluyó que ningún modelo tenía que ser modificado para esta iteración puesto que no se consideró necesario. Los nuevos requerimientos funcionales son solo de consulta y la arquitectura de la aplicación en la iteración 2 permite que estos nuevos requerimientos y los anteriores sean lo suficientemente eficientes.

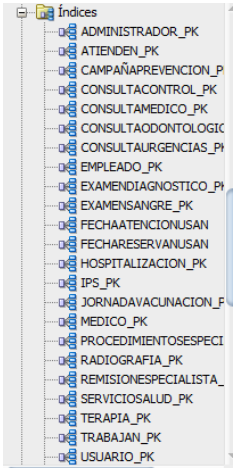
## Diseño físico (Índices)

Para justificar la selección de estos índices, se calculó la selectividad del campo FECHARESERVA y de FECHAATENCION. Se determinó que, para un año específico, la selectividad de ambos campos es de 1/365, el número de días. Esta es una buena selectividad, debido que es menor al 20%, y ello teniendo solo en cuenta la de un año, cuando en la BD hay alrededor de 30 años ingresados. El tipo de índice de ambos es de Árbol B+ secundario, pues solo tendrá apuntadores a los datos del árbol B+ principal.

```
-----------------------------------------------------Indices-----------------------------------------
CREATE INDEX "ISIS2304B101920"."FECHARESERVANUSAN" ON "ISIS2304B101920"."USAN" ("FECHARESERVA")
 PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS NOLOGGING
 STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "TBSPROD" ;

 CREATE INDEX "ISIS2304B101920"."FECHAATENCIONUSAN" ON "ISIS2304B101920"."USAN" ("FECHAATENCION")
 PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS NOLOGGING
 STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "TBSPROD" ;
```

Los índices creados por Oracle son los siguientes:

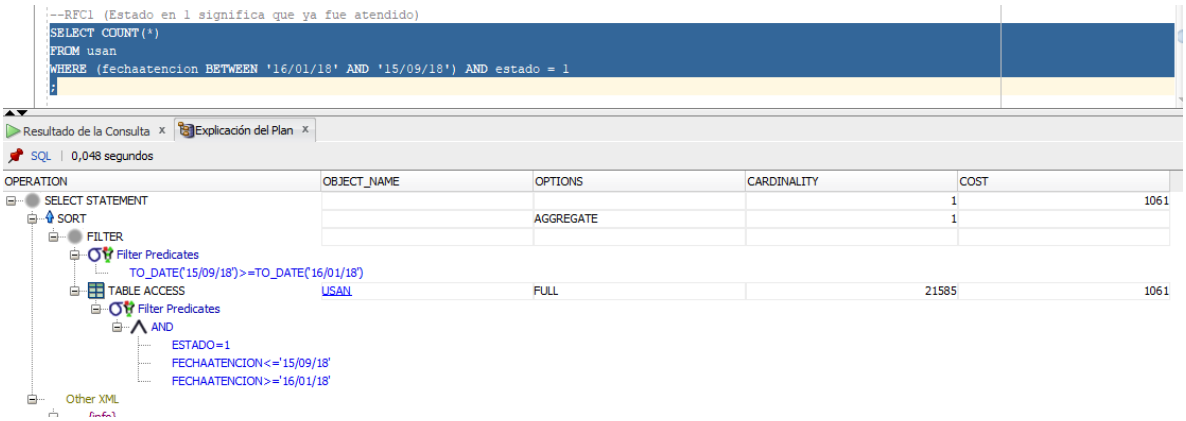



Estos pueden justificarse porque todos son de tipo Primary Key, lo que indica que son índices primarios y fueron creados por Oracle debido a su naturaleza como PK. Tales índices si ayudan al rendimiento de los RF, dado que son usados de manera frecuente para las consultas solicitadas.

## Planes de ejecución de Requerimientos

### RFC1

**El tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas especifico para pruebas de las consultas. El plan sugerido es Joins, que concuerda por el sugerido por Oracle, pues usa rangos.**



```
--RFC1 (Estado en 1 significa que ya fue atendido)
SELECT COUNT(*)
FROM usan
WHERE (fechaatencion BETWEEN '16/01/18' AND '15/09/18') AND estado = 1
;
```

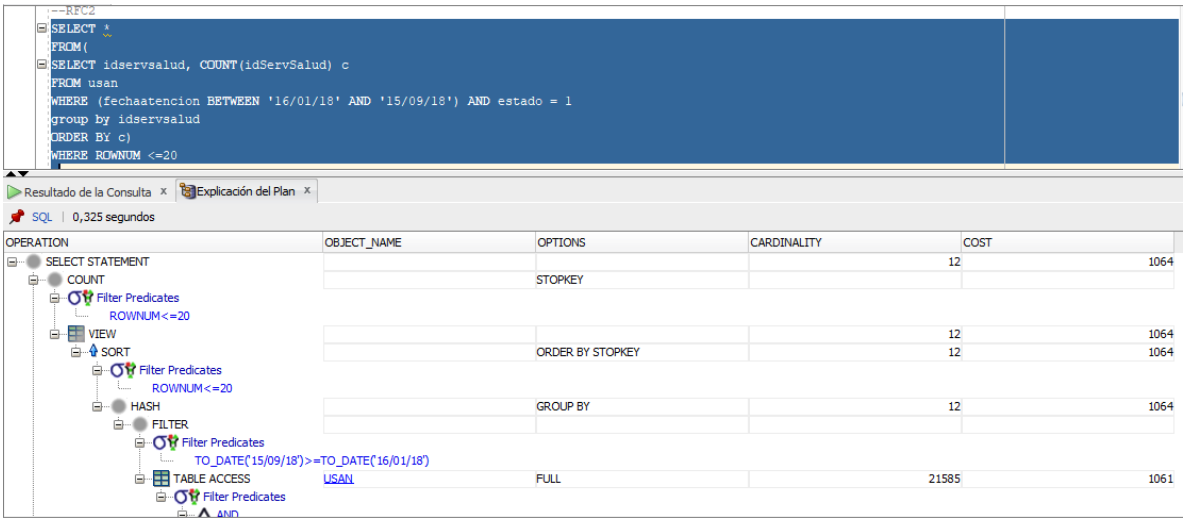
Resultado de la Consulta | Explicación del Plan

SQL | 0,048 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 1061 |
| SORT | | AGGREGATE | 1 | |
| FILTER | | | | |
| Filter Predicates | | | | |
| TO_DATE('15/09/18')>=TO_DATE('16/01/18') | | | | |
| TABLE ACCESS | USAN | FULL | 21585 | 1061 |
| Filter Predicates | | | | |
| AND | | | | |
| ESTADO=1 | | | | |
| FECHAATENCION<='15/09/18' | | | | |
| FECHAATENCION>='16/01/18' | | | | |
| Other XML | | | | |

### RFC2

**El tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas específico para pruebas de las consultas. El plan sugerido es Joins, que concuerda por el sugerido por Oracle, pues usa rangos.**



```
--RFC2
SELECT *
FROM(
SELECT idservsalud, COUNT(idServSalud) c
FROM usan
WHERE (fechaatencion BETWEEN '16/01/18' AND '15/09/18') AND estado = 1
group by idservsalud
ORDER BY c)
WHERE ROWNUM <=20
```

Resultado de la Consulta | Explicación del Plan

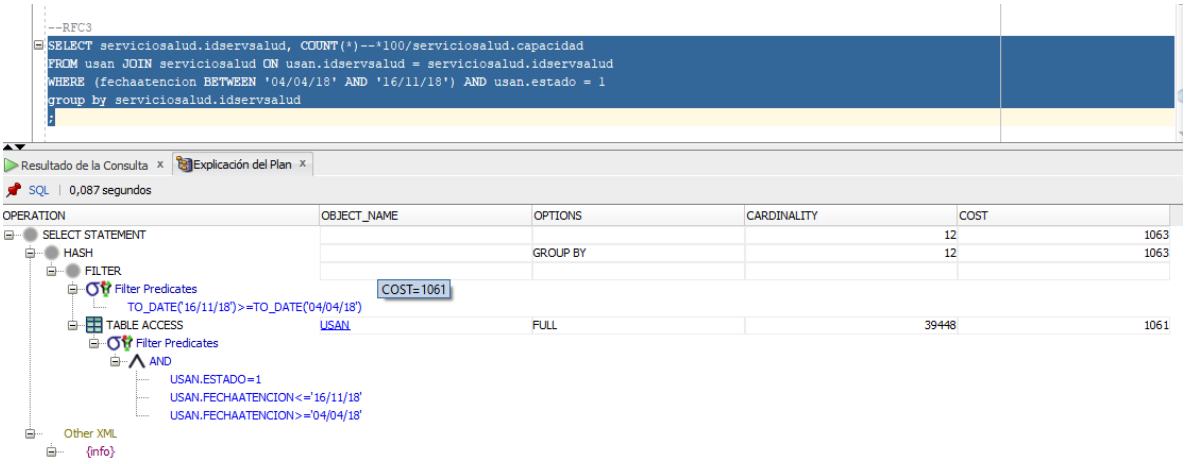
SQL | 0,325 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 12 | 1064 |
| COUNT | | STOPKEY | | |
| Filter Predicates | | | | |
| ROWNUM<=20 | | | | |
| VIEW | | | 12 | 1064 |
| SORT | | ORDER BY STOPKEY | 12 | 1064 |
| Filter Predicates | | | | |
| ROWNUM<=20 | | | | |
| HASH | | GROUP BY | 12 | 1064 |
| FILTER | | | | |
| Filter Predicates | | | | |
| TO_DATE('15/09/18')>=TO_DATE('16/01/18') | | | | |
| TABLE ACCESS | USAN | FULL | 21585 | 1061 |
| Filter Predicates | | | | |
| AND | | | | |

### RFC3

**El tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas especifico para pruebas de las**

**consultas. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**



```
--RFC3
SELECT serviciosalud.idservsalud, COUNT(*)--*100/serviciosalud.capacidad
FROM usan JOIN serviciosalud ON usan.idservsalud = serviciosalud.idservsalud
WHERE (fechaatencion BETWEEN '04/04/18' AND '16/11/18') AND usan.estado = 1
group by serviciosalud.idservsalud
;
```

Resultado de la Consulta | Explicación del Plan

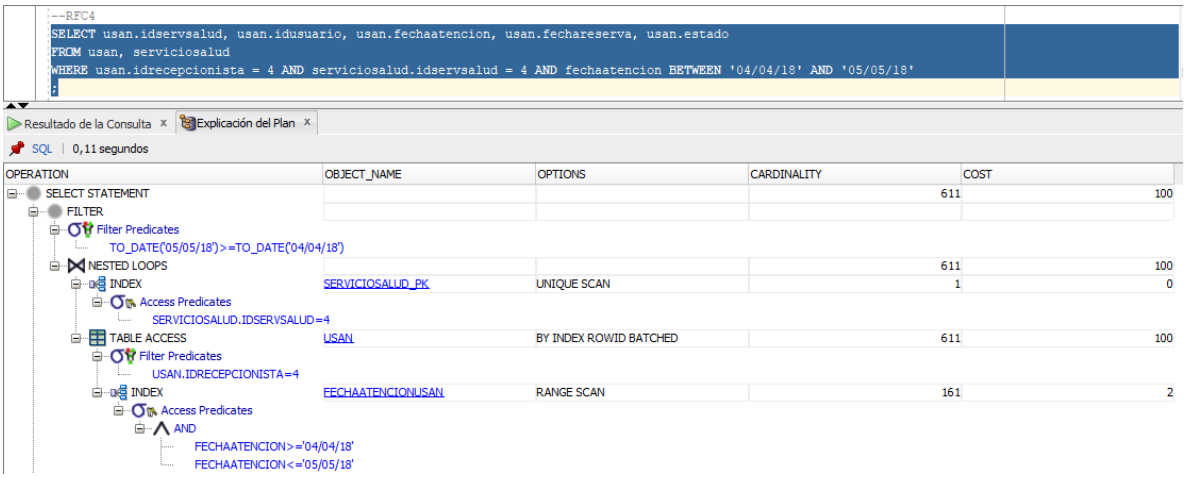
SQL | 0,087 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 12 | 1063 |
| HASH | | GROUP BY | 12 | 1063 |
| FILTER | | | | |
| Filter Predicates | | | | |
| TO_DATE('16/11/18')>=TO_DATE('04/04/18') | | | | |
| TABLE ACCESS | USAN | FULL | 39448 | 1061 |
| Filter Predicates | | | | |
| AND | | | | |
| USAN.ESTADO=1 | | | | |
| USAN.FECHAATENCION<='16/11/18' | | | | |
| USAN.FECHAATENCION>='04/04/18' | | | | |
| Other XML | | | | |
| {info} | | | | |

COST=1061

**RFC4**

**El tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas específico para pruebas de las consultas. El plan sugerido es Joins, que concuerda por el sugerido por Oracle, pues usa rangos.**



```
--RFC4
SELECT usan.idservsalud, usan.idusuario, usan.fechaatencion, usan.fechareserva, usan.estado
FROM usan, serviciosalud
WHERE usan.idrecepcionista = 4 AND serviciosalud.idservsalud = 4 AND fechaatencion BETWEEN '04/04/18' AND '05/05/18'
;
```

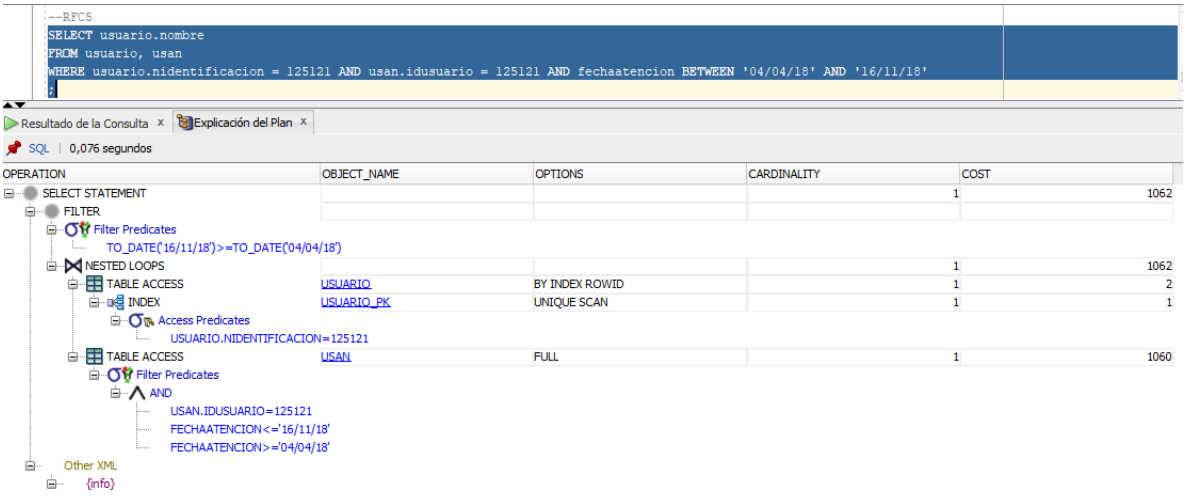
Resultado de la Consulta | Explicación del Plan

SQL | 0,11 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 611 | 100 |
| FILTER | | | | |
| Filter Predicates | | | | |
| TO_DATE('05/05/18')>=TO_DATE('04/04/18') | | | | |
| NESTED LOOPS | | | 611 | 100 |
| INDEX | SERVICIOSALUD_PK | UNIQUE SCAN | 1 | 0 |
| Access Predicates | | | | |
| SERVICIOSALUD.IDSERVSALUD=4 | | | | |
| TABLE ACCESS | USAN | BY INDEX ROWID BATCHED | 611 | 100 |
| Filter Predicates | | | | |
| USAN.IDRECEPCIONISTA=4 | | | | |
| INDEX | FECHAATENCIONUSAN | RANGE SCAN | 161 | 2 |
| Access Predicates | | | | |
| AND | | | | |
| FECHAATENCION>='04/04/18' | | | | |
| FECHAATENCION<='05/05/18' | | | | |

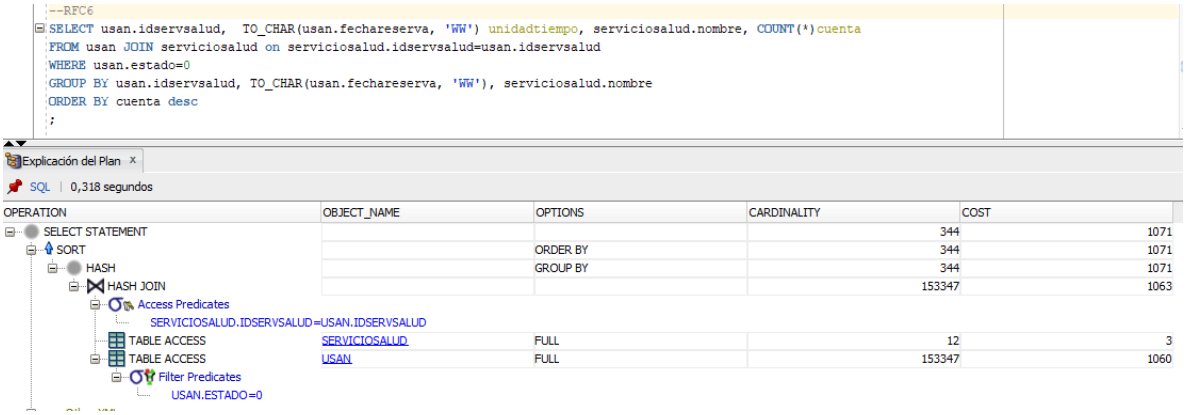
**RFC5**

**El tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas especifico para pruebas de las consultas. El plan sugerido es Joins, que concuerda por el sugerido por Oracle, pues usa rangos.**




## RFC6

**El tamaño de la respuesta cambiará dependiendo del id de servicio de salud usado y del estado buscado. Aquí también se insertaron datos cargados sobre varios id de servicios para comprobar funcionamientos. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**




## RFC7

**El tamaño de la respuesta cambiará dependiendo del id de servicio de salud usado y del estado buscado. Aquí también se insertaron datos cargados sobre varios id de servicios para comprobar funcionamientos. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**

```
--RFC7
SELECT *
FROM usuario,
    (
    SELECT COUNT(*) c,  usuario.nidentificacion n
    FROM usan, usuario,
        (
        SELECT COUNT(DISTINCT(usan.idservsalud)) co, usan.idusuario
        FROM usan group by usan.idusuario
        ) aux2
    WHERE usan.idusuario = usuario.nidentificacion AND usan.estado=1  AND aux2.co >3
    GROUP BY usuario.nidentificacion
    ) aux
WHERE aux.c > 11 and usuario.nidentificacion=aux.n
```

Explicación del Plan

SQL | 0,497 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 344 | 1071 |
| SORT | | ORDER BY | 344 | 1071 |
| HASH | | GROUP BY | 344 | 1071 |
| HASH JOIN | | | 153347 | 1063 |
| Access Predicates | | | | |
| SERVICIOSALUD.IDSERVSALUD=USAN.IDSERVSALUD | | | | |
| TABLE ACCESS | SERVICIOSALUD | FULL | 12 | 3 |
| TABLE ACCESS | USAN | FULL | 153347 | 1060 |
| Filter Predicates | | | | |

## RFC8

**El tamaño de la respuesta cambiará dependiendo del id de servicio de salud usado y del estado buscado. Aquí también se insertaron datos cargados sobre varios id de servicios para comprobar funcionamientos. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**

```
--RFC8
SELECT serviciosalud.idservsalud, serviciosalud.nombre, COUNT(*), TO_CHAR(usan.fechareserva, 'WW')
FROM usan, serviciosalud
WHERE  usan.idservsalud = serviciosalud.idservsalud AND EXTRACT (YEAR FROM usan.fechareserva) = EXTRACT (YEAR FROM SYSDATE) -1
group by serviciosalud.idservsalud, serviciosalud.nombre, TO_CHAR(usan.fechareserva, 'WW')
HAVING COUNT(*)<3
;
```

Explicación del Plan

SQL | 0,325 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 16 | 1072 |
| FILTER | | | | |
| Filter Predicates | | | | |
| COUNT(*)<3 | | | | |
| HASH | | GROUP BY | 16 | 1072 |
| HASH JOIN | | | 107507 | 1069 |
| Access Predicates | | | | |
| USAN.IDSERVSALUD=SERVICIOSALUD.IDSERVSALUD | | | | |
| TABLE ACCESS | SERVICIOSALUD | FULL | 12 | 3 |
| TABLE ACCESS | USAN | FULL | 107507 | 1066 |
| Filter Predicates | | | | |
| EXTRACT(YEAR FROM INTERNAL_FUNCTION(USAN.FECHARESERVA))=EXTRACT(YEAR FROM SYSDATE@!)-1 | | | | |
| Other XML | | | | |

COST=1066

RFC9 E**l tamaño de la respuesta cambiará dependiendo del tamaño del rango de fechas usado. Además se generaron datos cargados sobre un rango de fechas especifico para pruebas de las consultas. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**

```
--RFC9----------------------------Falta ordenar en Java la adicion segun lo que desee el usuario.
--Se tiene que ir armando la sentencia pedazo a pedazo
SELECT *
FROM usuario JOIN usan ON usan.idusuario = usuario.nidentificacion, serviciosalud
WHERE usan.idservsalud = 1 AND usan.fechaatencion>'04/04/18' AND usan.fechaatencion<'04/05/19'
AND serviciosalud.idservsalud=usan.idservsalud AND serviciosalud.idips = 7007
ORDER BY serviciosalud.idservsalud
;
```

SQL | 0,16 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 4809 | 652 |
| FILTER | | | | |
| Filter Predicates | | | | |
| TO_DATE('04/05/19')>TO_DATE('04/04/18') | | | | |
| HASH JOIN | | | 4809 | 652 |
| Access Predicates | | | | |
| AND | | | | |
| SERVICIOSALUD.IDSERVSALUD=USAN.IDSERVSALUD | | | | |
| USAN.IDUSUARIO=USUARIO.NIDENTIFICACION | | | | |
| TABLE ACCESS | USAN | FULL | 4809 | 480 |
| Filter Predicates | | | | |
| AND | | | | |
| USAN.IDSERVSALUD=1 | | | | |
| USAN.FECHAATENCION<'04/05/19' | | | | |
| USAN.FECHAATENCION>'04/04/18' | | | | |
| NESTED LOOPS | | | 100015 | 172 |
| TABLE ACCESS | SERVICIOSALUD | BY INDEX ROWID | 1 | 1 |
| Filter Predicates | | | | |
| SERVICIOSALUD.IDIPS=7007 | | | | |
| INDEX | SERVICIOSALUD_PK | UNIQUE SCAN | 1 | 0 |
| Access Predicates | | | | |
| SERVICIOSALUD.IDSERVSALUD=1 | | | | |

**RFC10 El tamaño de la respuesta dependerá en cuantos usuarios no han usado servicios de salud. Para ello se crearon usuarios que no figuran en la tabla usan, para comprobar las sentencias. El plan sugerido es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**

```
--Usuarios que no usan ningun servicio
SELECT count(*)
FROM usuario LEFT JOIN usan ON usuario.nidentificacion = usan.idusuario
WHERE usan.idusuario is NULL;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,05 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 582 |
| SORT | | AGGREGATE | 1 | |
| FILTER | | | | |
| Filter Predicates | | | | |
| USAN.IDUSUARIO IS NULL | | | | |
| HASH JOIN | | OUTER | 85938 | 582 |
| Access Predicates | | | | |
| USUARIO.NIDENTIFICACION=USAN.IDUSUARIO(+) | | | | |
| INDEX | USUARIO_PK | FAST FULL SCAN | 100015 | 102 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |

```
--Servicios de salud que no son usados
SELECT count(*)
FROM serviciosalud LEFT JOIN usan ON serviciosalud.idservsalud = usan.idservsalud
WHERE usan.idservsalud is NULL;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,067 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 481 |
| SORT | | AGGREGATE | 1 | |
| FILTER | | | | |
| Filter Predicates | | | | |
| USAN.IDSERVSALUD IS NULL | | | | |
| HASH JOIN | | OUTER | 1 | 481 |
| Access Predicates | | | | |
| SERVICIOSALUD.IDSERVSALUD=USAN.IDSERVSALUD(+) | | | | |
| INDEX | SERVICIOSALUD_PK | FULL SCAN | 12 | 1 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |
| info type="db_version" | | | | |

```
--IPS que no son usadas en ningun servicio de salud
SELECT count(*)
FROM ips LEFT JOIN (serviciosalud JOIN usan ON usan.idservsalud = serviciosalud.idservsalud) ON ips.idips =serviciosalud.idips
WHERE serviciosalud.idips IS NULL;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,043 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 1 | 484 |
| SORT | | AGGREGATE | 1 | |
| HASH JOIN | | ANTI | 3 | 484 |
| Access Predicates | | | | |
| IPS.IDIPS=SERVICIOSALUD.IDIPS | | | | |
| INDEX | IPS_PK | FULL SCAN | 3 | 1 |
| VIEW | SYS.null | | 12 | 483 |
| HASH JOIN | | SEMI | 12 | 483 |
| Access Predicates | | | | |
| USAN.IDSERVSALUD=SERVICIOSALUD.IDSERVSALUD | | | | |
| TABLE ACCESS | SERVICIOSALU | FULL | 12 | 3 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |

COST=479

**RFC11 El plan sugerido para todos es Hash, pues usa equijoins, que concuerda con el sugerido por Oracle.**

**El servicio más usado por semana.**

```
--El servicio mas usado por semana
SELECT *
FROM(
SELECT semana, aux.idservsalud, MIN(cuenta) over (partition by semana ORDER BY cuenta desc )as maximo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(usan.idservsalud) cuenta, usan.idservsalud
    FROM usan
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), usan.idservsalud
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM=1
    ;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,058 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 344 | 495 |
| VIEW | | | 344 | 495 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 344 | 495 |
| VIEW | | | 344 | 494 |
| SORT | | ORDER BY | 344 | 494 |
| HASH | | GROUP BY | 344 | 494 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |
| info type="db_version" | | | | |
| 12.1.0.2 | | | | |
| info type="parse_schema" | | | | |

COST=479

EL servicio menos usado por semana

```
--El servicio menos usado por semana
SELECT *
FROM (
SELECT semana, aux.idservsalud, MAX(cuenta) over (partition by semana ORDER BY cuenta  )as minimo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY sema
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(usan.idservsalud) cuenta, usan.idservsalud
    FROM usan
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), usan.idservsalud
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM = 1
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,049 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 344 | 495 |
| VIEW | | | 344 | 495 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 344 | 495 |
| VIEW | | | 344 | 494 |
| SORT | | ORDER BY | 344 | 494 |
| HASH | | GROUP BY | 344 | 494 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |

El usuario que mas usa servicios por semana

```
--El usuario que mas usa servicios por semana
SELECT *
FROM (
SELECT semana, aux.idusuario, MIN(cuenta) over (partition by semana ORDER BY cuenta  desc)as maximo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY se
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(usan.idusuario) cuenta, usan.idusuario
    FROM usan
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), usan.idusuario
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM = 1
    ;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,159 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 286387 | 5545 |
| VIEW | | | 286387 | 5545 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 286387 | 5545 |
| VIEW | | | 286387 | 3265 |
| SORT | | ORDER BY | 286387 | 3265 |
| HASH | | GROUP BY | 286387 | 3265 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |

El usuario que menos usa servicios por semana

```
--El usuario que menos usa servicios por semana
SELECT *
FROM (
SELECT semana, aux.idusuario, MAX(cuenta) over (partition by semana ORDER BY cuenta  )as minimo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY semana)
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(usan.idusuario) cuenta, usan.idusuario
    FROM usan
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), usan.idusuario
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM = 1
    ;
```

Resultado de la Consulta | Explicación del Plan

SQL | 0,053 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 286387 | 5545 |
| VIEW | | | 286387 | 5545 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 286387 | 5545 |
| VIEW | | | 286387 | 3265 |
| SORT | | ORDER BY | 286387 | 3265 |
| HASH | | GROUP BY | 286387 | 3265 |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |

La ips menos usada por semana

```
--El ips menos usado por semana
SELECT *
FROM (
SELECT semana, aux.idips, MAX(cuenta) over (partition by semana ORDER BY cuenta  )as minimo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY semana) AS
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(ips.idips) cuenta, ips.idips
    FROM usan join serviciosalud on usan.idservsalud = serviciosalud.idservsalud join ips on serviciosalud.idips = ips.idips
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), ips.idips
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM = 1
    ;
```

Resultado de la Consulta | Rastreo Automático | Explicación del Plan

SQL | 0,055 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 282 | 498 |
| VIEW | | | 282 | 498 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 282 | 498 |
| VIEW | | | 282 | 497 |
| SORT | | ORDER BY | 282 | 497 |
| HASH | | GROUP BY | 282 | 497 |
| HASH JOIN | | | 305025 | 483 |
| Access Predicates | | | | |
| USAN.IDSERVSALUD=SERVICIOSALUD.IDSERVSALUD | | | | |
| TABLE ACCESS | SERVICIOSALUD | FULL | 12 | 3 |
| Filter Predicates | | | | |
| SERVICIOSALUD.IDIPS IS NOT NULL | | | | |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |

La ips mas usada por semana

```
--El ips mas usado por semana
SELECT *
FROM (
SELECT semana, aux.idips, MIN(cuenta) over (partition by semana ORDER BY cuenta desc )as maximo, ROW_NUMBER() OVER (PARTITION BY semana ORDER BY semana
FROM (SELECT TO_CHAR(usan.fechareserva, 'WW') semana, COUNT(ips.idips) cuenta, ips.idips
    FROM usan join serviciosalud on usan.idservsalud = serviciosalud.idservsalud join ips on serviciosalud.idips = ips.idips
    GROUP BY TO_CHAR(usan.fechareserva, 'WW'), ips.idips
    ORDER BY semana, cuenta DESC) aux
ORDER BY semana)
WHERE ROW_NUM = 1
    ;
```

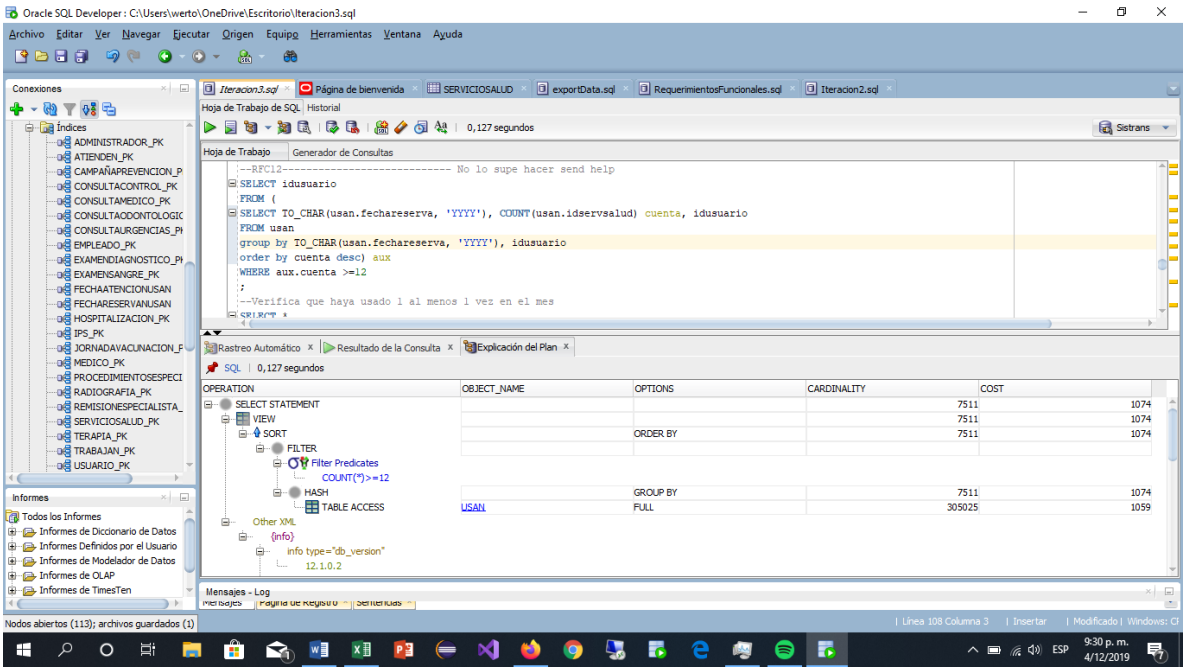
Resultado de la Consulta | Rastreo Automático | Explicación del Plan

SQL | 0,064 segundos

| OPERATION | OBJECT_NAME | OPTIONS | CARDINALITY | COST |
|---|---|---|---|---|
| SELECT STATEMENT | | | 282 | 498 |
| VIEW | | | 282 | 498 |
| Filter Predicates | | | | |
| ROW_NUM=1 | | | | |
| WINDOW | | SORT | 282 | 498 |
| VIEW | | | 282 | 497 |
| SORT | | ORDER BY | 282 | 497 |
| HASH | | GROUP BY | 282 | 497 |
| HASH JOIN | | | 305025 | 483 |
| Access Predicates | | | | |
| USAN.IDSERVSALUD=SERVICIOSALUD.IDSERVSALUD | | | | |
| TABLE ACCESS | SERVICIOSALUD | FULL | 12 | 3 |
| Filter Predicates | | | | |
| SERVICIOSALUD.IDIPS IS NOT NULL | | | | |
| TABLE ACCESS | USAN | FULL | 305025 | 479 |
| Other XML | | | | |
| {info} | | | | |
| info type="db version" | | | | |

RFC12

**El plan sugerido es Joins, que concuerda por el sugerido por Oracle, pues usa rangos.**

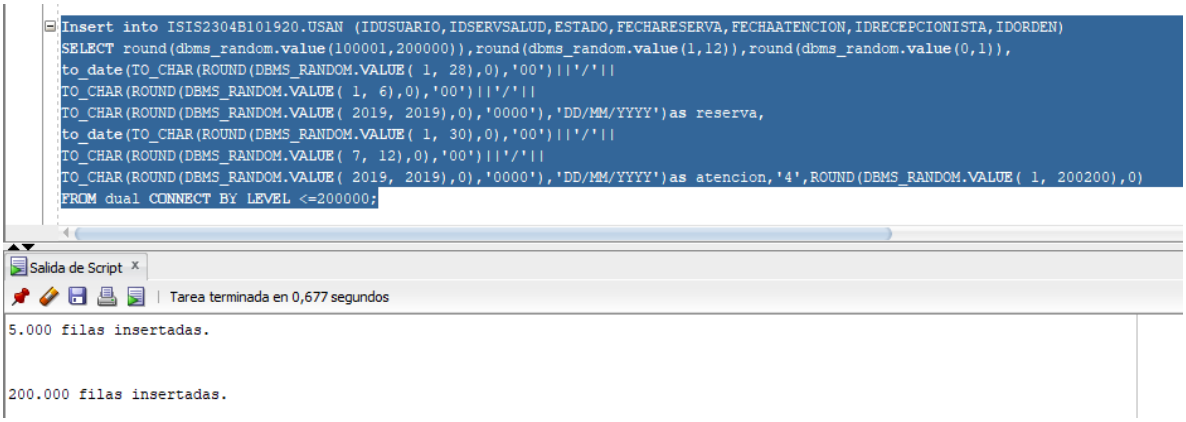



## Análisis del proceso de optimización

La diferencia de consultas realizadas o delgadas al SMBD es que Oracle ya tiene históricos de cómo se comportarán los datos y tiene automatizaciones y ventajas del manejo de grandes volúmenes de datos como lo son por ejemplo los índices. Por ello, consultas que en la aplicación se delegarían a while o for, son mucho mas optimizadas en Oracle por sus estructuras de datos como el Árbol B+.

## Carga de datos

La carga de datos se hizo desde SQLDeveloper son sentencias que hacen Selects a la tabla de datos de prueba DUAL y a la cual se le asignaban datos estratégicamente seleccionados. La cantidad de datos se ajustaba mediante el comando CONNECT BY LEVEL <=x, donde x representa la cantidad de datos que se desean insertar. A continuación se presenta un ejemplo en la tabla USAN:



Como se puede ver es eficiente, pues demora <1 segundo insertar 200 mil datos, y además es fácil elegir que datos se quieren insertar, para poder hacer casos de prueba.