Documentación Iteración No. 3

Juan P. Correa Puerta, Nicolás F. Jaramillo Páez
Universidad de los Andes, Bogotá, Colombia
Sistemas Transaccionales, ISIS-2304 (2)
Profesor: German Enrique Bravo Córdoba
Fecha de presentación: mayo 20 de 2019

Contenido

1 Diseño de la aplicación

1.1 Análisis de modelos

Con base en la retroalimentación de la Iteración 2, no se resaltó ningún fallo en el modelo UML. Con esto en mente, decidimos conservar el mismo diagrama con el que se venía trabajando, tal y como se muestra a continuación:

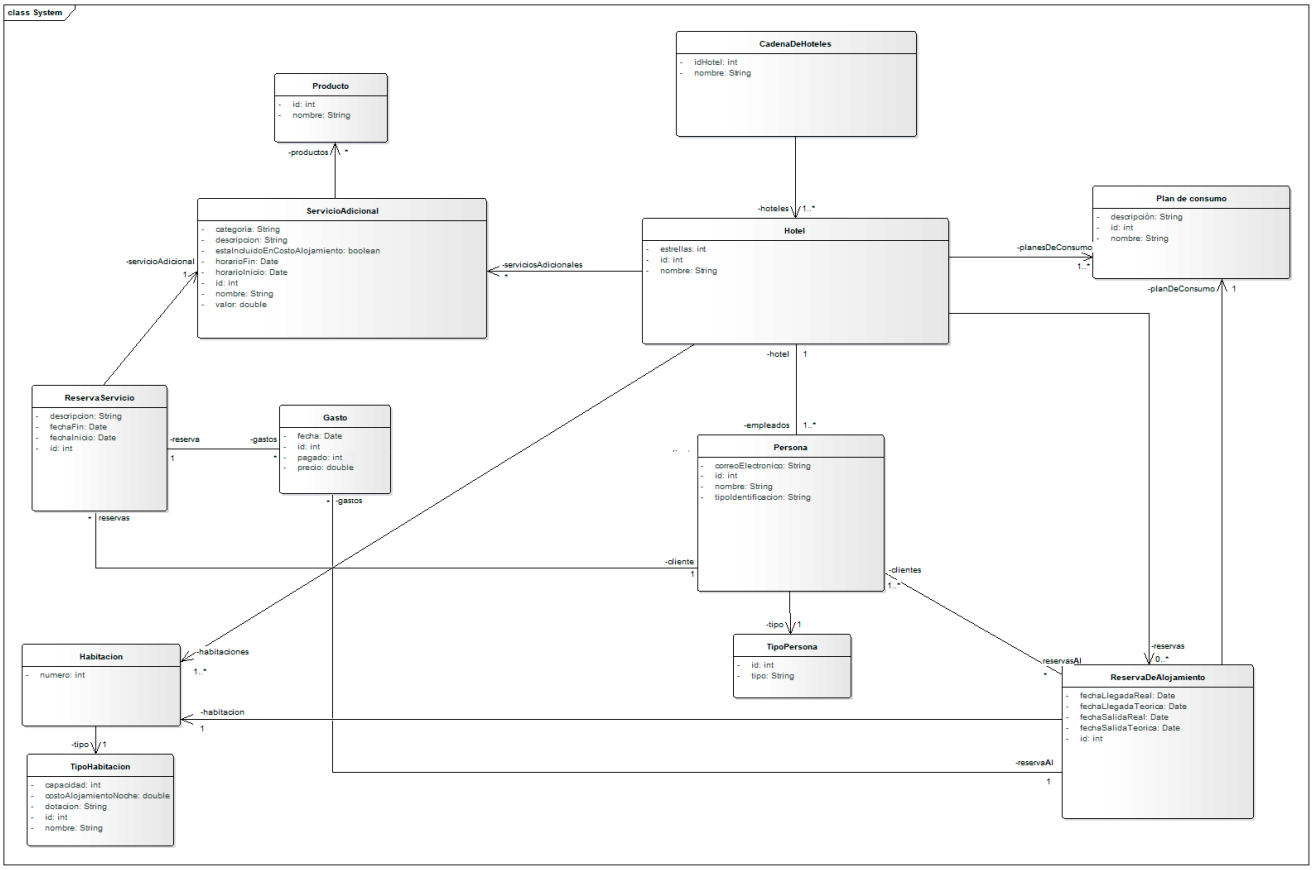


Figura 1. Modelo UML presentado en la Iteración 2

De la misma manera, el modelo relacional planteado en la Iteración 2 permaneció sin cambios:

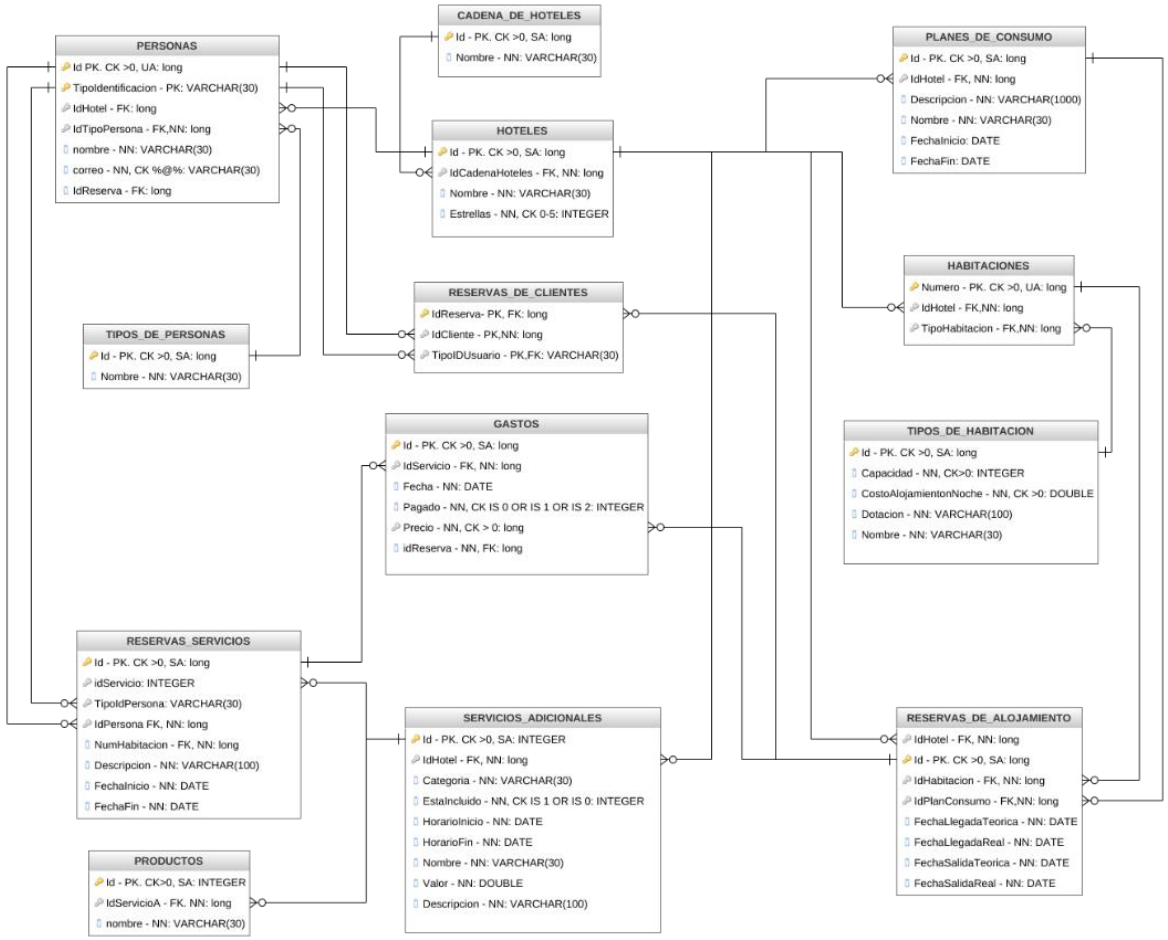


Figura 2. Modelo relacional presentado en la Iteración 2

1.2 Diseño físico

1.2.1 Índices creados automáticamente por Oracle

Gracias a Oracle se generaron automáticamente índices por la llave primaria de cada una de las tablas. Estos índices son constantemente utilizados en las consultas sql, y en el caso de la presente iteración, se usan especialmente los de `RESERVAS_DE_ALOJAMIENTO`, `RESERVAS_DE_CLIENTES` y `PERSONAS`, pues en los diversos requerimientos se realizan `INDEX FULL SCAN`, `INDEX RANGE SCAN` e `INDEX UNIQUE SCAN` sobre estos. El tamaño para todos es el mismo, iniciando con 65.536 bytes y donde la siguiente tupla requiere 1’048.576 bytes adicionales. Adicionalmente, la cantidad máxima de tuplas es 2.147’483.645. El mantenimiento se realizará cada vez que ocurra una inserción o borrado de una tupla. Para algunas tablas esto casi no sucede, como lo es en el caso de la tabla `PERSONAS`. No

obstante, en otras tablas el mantenimiento es intensivo, como por ejemplo en RESERVAS_DE_ALOJAMIENTO. Finalmente, todos los índices son hash primarios:

```
--------------------------------------------------------
--  DDL for Index IDRESERVASSERVICIOS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."IDRESERVASSERVICIOS_PK" ON "ISIS2304B181910"."RESERVAS_SERVICIOS" ("ID")
  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" ;

--------------------------------------------------------
--  DDL for Index IDHOTELES_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."IDHOTELES_PK" ON "ISIS2304B181910"."CADENA_DE_HOTELES" ("ID")
  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" ;

--------------------------------------------------------
--  DDL for Index IDGATOS_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."IDGATOS_PK" ON "ISIS2304B181910"."GASTOS" ("ID")
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" ;

--------------------------------------------------------
--  DDL for Index NUMEROHABITACIONES_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."NUMEROHABITACIONES_PK" ON "ISIS2304B181910"."HABITACIONES" ("NUMERO")
  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" ;
```

```
--------------------------------------------------------
--  DDL for Index IDHOTELESHOTELES_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."IDHOTELESHOTELES_PK" ON "ISIS2304B181910"."HOTELES" ("ID")
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" ;

--------------------------------------------------------
--  DDL for Index IDPERSONAS_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."IDPERSONAS_PK" ON "ISIS2304B181910"."PERSONAS" ("ID", "TIPOIDENTIFICACION")
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" ;

--------------------------------------------------------
--  DDL for Index IDDELPRODUCTO_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."IDDELPRODUCTO_PK" ON "ISIS2304B181910"."PRODUCTOS" ("ID")
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" ;

--------------------------------------------------------
--  DDL for Index IDPLANES_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."IDPLANES_PK" ON "ISIS2304B181910"."PLANES_DE_CONSUMO" ("ID")
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" ;
```

```
--------------------------------------------------------
--  DDL for Index IDRESERVASALOJAMIENTO_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."IDRESERVASALOJAMIENTO_PK" ON "ISIS2304B181910"."RESERVAS_DE_ALOJAMIENTO" ("ID") 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" ;

--------------------------------------------------------
--  DDL for Index RESERVAS_DE_CLIENTES_PK
--------------------------------------------------------

CREATE UNIQUE INDEX "ISIS2304B181910"."RESERVAS_DE_CLIENTES_PK" ON "ISIS2304B181910"."RESERVAS_DE_CLIENTES" ("IDRESERVA", "IDUSUARIO", "TIPOIDUSUARIO") 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" ;

--------------------------------------------------------
--  DDL for Index NUMEROSERVICIOSADICIONALES_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."NUMEROSERVICIOSADICIONALES_PK" ON "ISIS2304B181910"."SERVICIOS_ADICIONALES" ("ID") 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" ;

--------------------------------------------------------
--  DDL for Index IDTIPOSHABITACION_PK
--------------------------------------------------------

  CREATE UNIQUE INDEX "ISIS2304B181910"."IDTIPOSHABITACION_PK" ON "ISIS2304B181910"."TIPOS_DE_HABITACION" ("ID") 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" ;
```

```
--------------------------------------------------------
--  DDL for Index IDTIPOSPERSONAS_PK
--------------------------------------------------------
CREATE UNIQUE INDEX "ISIS2304B181910"."IDTIPOSPERSONAS_PK" ON
"ISIS2304B181910"."TIPOS_DE_PERSONAS" ("ID") 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" ;
```

1.2.2 Sentencias SQL e índices:

RFC9 Consultar consumo en HotelAndes: Se quiere conocer la información de los clientes que consumieron al menos una vez un determinado servicio del hotel, en un rango de fechas. Los resultados deben ser clasificados según un criterio deseado por quien realiza la consulta. En la clasificación debe ofrecerse la posibilidad de agrupamiento y ordenamiento de las respuestas según los intereses del usuario que consulta como, por ejemplo, por los datos del cliente, por fecha y número de veces que se utilizó el servicio. Esta operación está disponible para el recepcionista del hotel, el gerente del hotel y también para los organizadores de eventos.

Para poder cumplir con que el usuario agrupe y ordene por cantidad de veces que se consumió el servicio, fechas en las que se consumió el servicio, entre otros ordenamientos, se dividió este requerimiento en dos consultas, una en la que se necesitan las fechas y otra en la que se necesitan agrupar los resultados sin tener en cuenta las fechas:

Tipo 1: Se pide agrupar por un criterio que no es fechas

```
SELECT COUNT(*) AS APARICIONES, per.TIPOIDENTIFICACION, per.ID,
per.NOMBRE, per.CORREO
FROM PERSONAS per, RESERVAS_SERVICIOS resSer
WHERE per.TIPOIDENTIFICACION = resSer.TIPOIDENTIFICACION
    AND resSer.IDSERVICIO = <ID>
    AND per.ID = resSer.IDTIPOPERSONA
    AND resSer.FECHAINICIO >= 'DD/MM/AAAA'
    AND resSer.FECHAFIN <= 'DD/MM/AAAA'
GROUP BY per.TIPOIDENTIFICACION, per.ID, per.NOMBRE, per.CORREO
ORDER BY <Criterio> <Orden>
```

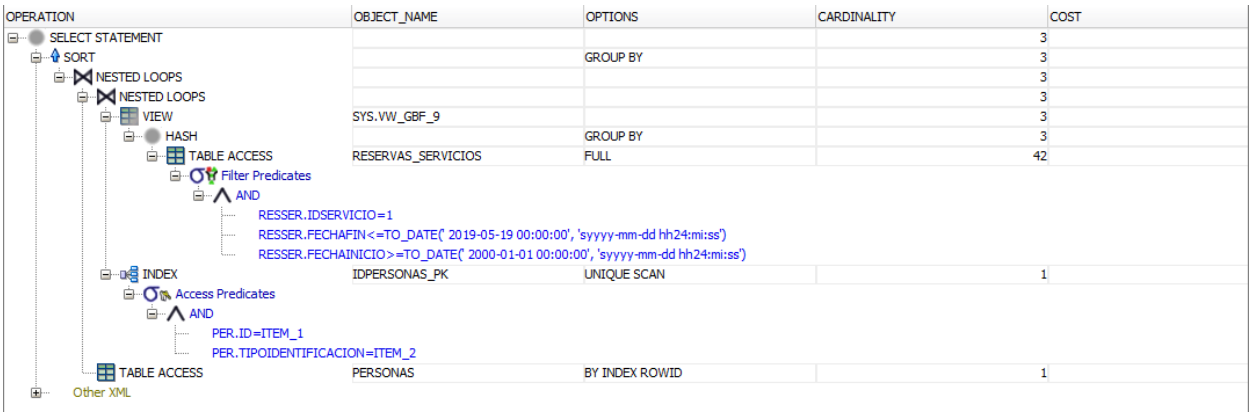


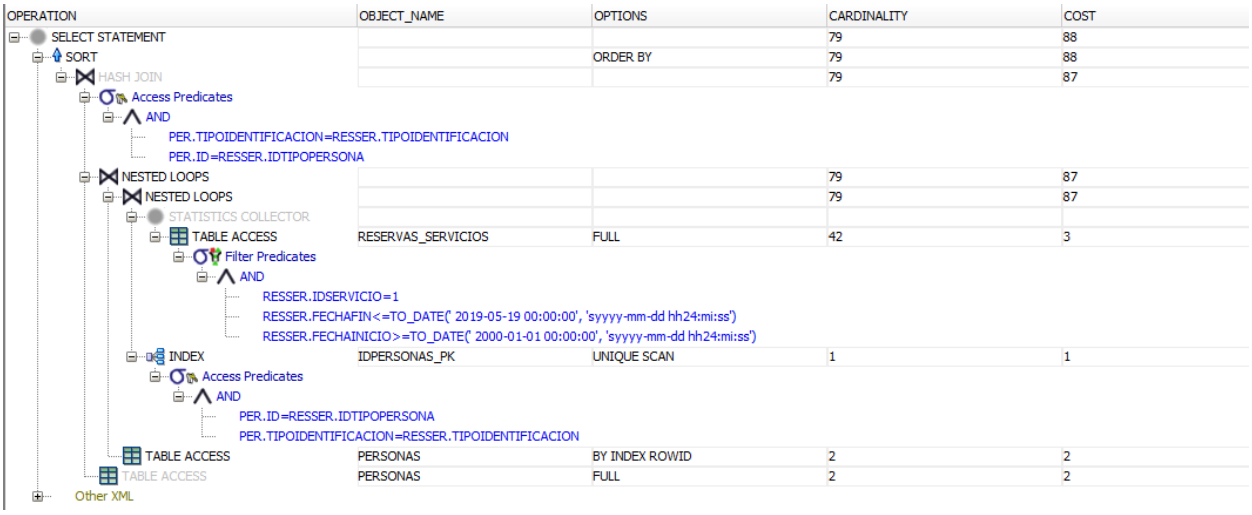
Figura 3. Plan de ejecución propuesto por Oracle para el RFC9 sin fechas

En el plan de ejecución propuesto por Oracle, se accede primero a la tabla RESERVAS_SERVICIOS y se filtran las condiciones estipuladas en la consulta, posteriormente se pasan a un hash para ordenarlas por el criterio seleccionado. Después, esta información se contrasta con la tabla PERSONAS mediante un NESTED LOOP para finalmente ordenarlo y retornar la consulta deseada.

El tiempo de ejecución es de 721 ms y la selectividad basada en la cardinalidad es de 42%

Tipo 2: Se pide agrupar por un criterio que sí relaciona fechas

```
SELECT resSer.FECHAINICIO, resSer.FECHAFIN, per.TIPOIDENTIFICACION,
per.ID, per.NOMBRE, per.CORREO
FROM PERSONAS per, RESERVAS_SERVICIOS resSer
WHERE per.TIPOIDENTIFICACION = resSer.TIPOIDENTIFICACION
    AND resSer.IDSERVICIO = <ID>
    AND per.ID = resSer.IDTIPOPERSONA
    AND resSer.FECHAINICIO >= 'DD/MM/AAAA'
    AND resSer.FECHAFIN <= 'DD/MM/AAAA'
ORDER BY resSer.<CuálFecha> <Orden>
```



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			79	88
SORT		ORDER BY	79	88
HASH JOIN			79	87
Access Predicates				
AND				
PER.TIPOIDENTIFICACION=RESSER.TIPOIDENTIFICACION				
PER.ID=RESSER.IDTIPOPERSONA				
NESTED LOOPS			79	87
NESTED LOOPS			79	87
STATISTICS COLLECTOR				
TABLE ACCESS	RESERVAS_SERVICIOS	FULL	42	3
Filter Predicates				
AND				
RESSER.IDSERVICIO=1				
RESSER.FECHAFIN<=TO_DATE(' 2019-05-19 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESSER.FECHAINICIO>=TO_DATE(' 2000-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	IDPERSONAS_PK	UNIQUE SCAN	1	1
Access Predicates				
AND				
PER.ID=RESSER.IDTIPOPERSONA				
PER.TIPOIDENTIFICACION=RESSER.TIPOIDENTIFICACION				
TABLE ACCESS	PERSONAS	BY INDEX ROWID	2	2
TABLE ACCESS	PERSONAS	FULL	2	2
Other XML				

Figura 4. Plan de ejecución propuesto por Oracle para el RFC9 con fechas

Debido a la similitud con la consulta anterior, se observa cómo se realizan los mismos filtros iniciales sobre RESERVAS_SERVICIOS, con la diferencia de que ahora se requiere un nuevo acceso a PERSONAS y RESERVAS_SERVICIOS para poder almacenar la información de las fechas en la que se consumieron los servicios. Es interesante ver cómo ahora Oracle se vale de un Hash Join, probablemente por la gran cantidad de información que está presente y por la forma en que se selecciona la información (IDs únicos). La selectividad según la cardinalidad es del 79% y el tiempo de ejecución es inferior a los 100 ms.

> *RFC10 Consultar consumo en HotelAndes (RFC9 V2):* Se quiere conocer la información de los clientes que NO consumieron ninguna vez un determinado servicio del hotel, en un rango de fechas. Los resultados deben ser clasificados según un criterio deseado por quien realiza la consulta. En la clasificación debe ofrecerse la posibilidad de agrupamiento y ordenamiento de las respuestas según los intereses del usuario que consulta como, por ejemplo, por los datos del cliente, por fecha y número de veces que se utilizó el servicio. Esta operación está disponible para el recepcionista del hotel, el gerente del hotel y también para los organizadores de eventos.

Este requerimiento se solucionó de manera similar al requerimiento anterior, por lo que también se divide en dos partes, una consulta donde se necesitan las fechas y otra en donde se agrupa sin tenerlas presentes:

Tipo 1: Se pide agrupar por un criterio que no es fechas

```
SELECT res.TIPOIDENTIFICACION, res.ID, res.NOMBRE, res.CORREO
```

```
FROM PERSONAS res
WHERE res.ID NOT IN ( SELECT per.ID
FROM PERSONAS per, RESERVAS_SERVICIOS resSer
WHERE per.TIPOIDENTIFICACION = resSer.TIPOIDENTIFICACION
    AND resSer.IDSERVICIO = <ID>
    AND per.ID = resSer.IDTIPOPERSONA
    AND resSer.FECHAINICIO >= 'DD/MM/AAAA'
    AND resSer.FECHAFIN <= 'DD/MM/AAAA'
  GROUP   BY   per.TIPOIDENTIFICACION,   per.ID,   per.NOMBRE,
per.CORREO )
ORDER BY <Criterio> <Orden>
```

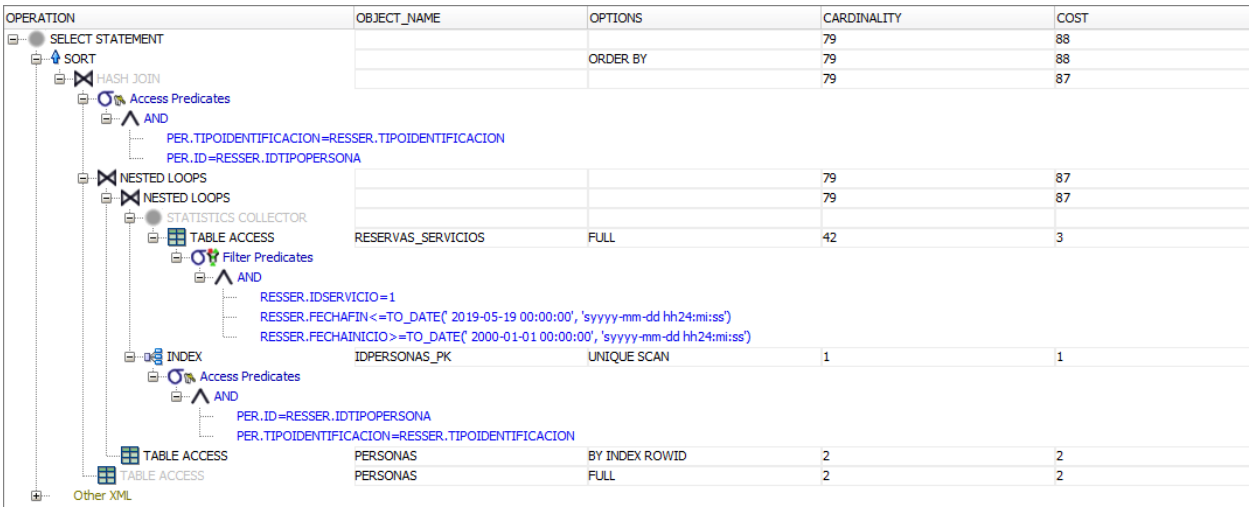


Figura 5. Plan de ejecución propuesto por Oracle para el RFC10 sin fechas

Esta transacción opera de la misma manera que su homónima en el RFC9, con la diferencia de que ahora se necesita volver a comparar con PERSONAS para poder extraer los que NO cumplen lo solicitado. Los tiempos de ejecución son inferiores a 420 ms.

Tipo 2: Se pide agrupar por un criterio que implica fechas

```
SELECT  conResSer.FECHAINICIO,  conResSer.FECHAFIN,
conPer.TIPOIDENTIFICACION, conPer.ID, conPer.NOMBRE,
conPer.CORREO
FROM PERSONAS conPer, RESERVAS_SERVICIOS conResSer
WHERE conPer.TIPOIDENTIFICACION = conResSer.TIPOIDENTIFICACION
   AND conPer.ID = conResSer.IDTIPOPERSONA
   AND conPer.ID NOT IN ( SELECT per.ID
FROM PERSONAS per, RESERVAS_SERVICIOS resSer
```

```
WHERE per.TIPOIDENTIFICACION = resSer.TIPOIDENTIFICACION
   AND resSer.IDSERVICIO = <ID>
   AND per.ID = resSer.IDTIPOPERSONA
   AND resSer.FECHAINICIO >= 'DD/MM/AAAA'
   AND resSer.FECHAFIN <= 'DD/MM/AAAA' )
ORDER BY conResSer.<CuálFecha> <Orden>
```

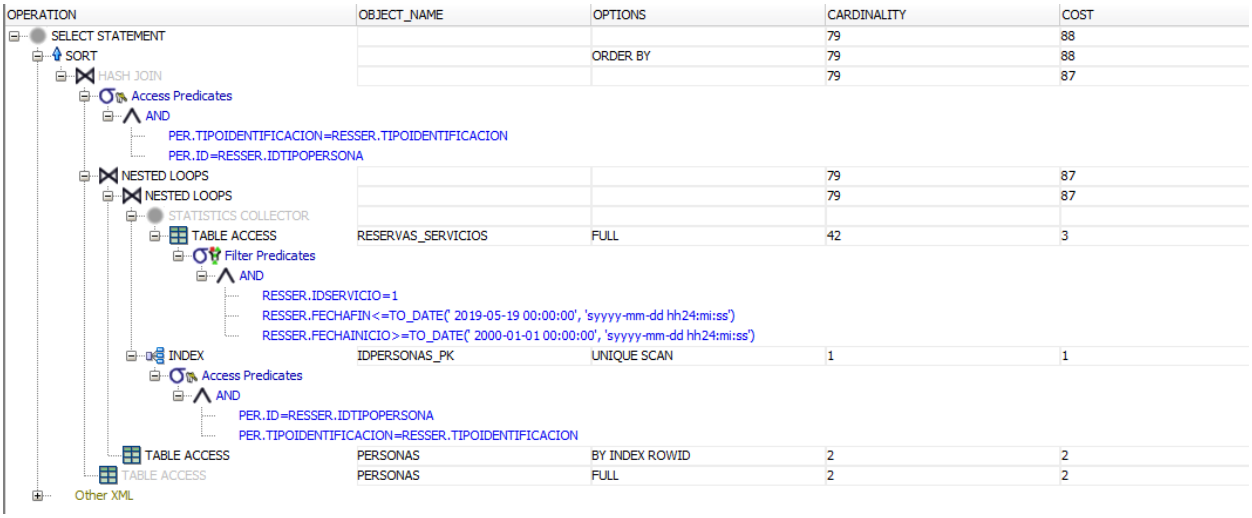


Figura 6. Plan de ejecución propuesto por Oracle para el RFC10 con fechas

Nuevamente Oracle actúa de manera similar que en el requerimiento RFC9 con fechas, sólo que ahora se hace un Hash Join entre la consulta RFC9 y la tabla PERSONAS. En tiempo de ejecución tenemos 61 ms y una selectividad de 79% según la cardinalidad.

> *RFC11 Consultar funcionamiento:* Muestra, para cada semana del año (sábado a sábado), el servicio más consumido, el servicio menos consumido, las habitaciones más solicitadas y las habitaciones menos solicitadas. Las respuestas deben ser sustentadas por el detalle de las reservas y consumos correspondiente. Esta operación es realizada el gerente general de HotelAndes.

Tipo 1: Muestra, para cada semana del año (sábado a sábado), el servicio más consumido y el servicio menos. En Java se realiza un for de 1 a 53 donde la variable es I en la consulta.

Escenarios: Este requerimiento se ve afectado únicamente por la tabla de reservas servicios, es decir, esta es una tabla que no está uniformemente distribuida y que posee un índice primario. Sin embargo, los filtros más importantes dependen de su fecha. Por lo que se ve afectado a medida que incrementa el número de estos, sin embargo, el número de tuplas en las demás tablas es irrelevante.

Índices: No se utiliza ningún índice para esta consulta, sin embargo, podría llegar a ser útil un compuesto por el id y la fecha de inicio.

Tiempo de ejecución: Este oscila entre 350 y 399 milisegundos.

Selectividad: 2%

```
select   idservicio
from reservas_servicios
where  to_char(to_date(fechainicio,'DD/MM/YYYY'),'ww') = I
group by to_char(to_date(fechainicio,'DD/MM/YYYY'),'ww'),idservicio
order                                                          by
(to_char(to_date(fechainicio,'DD/MM/YYYY'),'ww')),count(idservicio)  desc;
```

El plan de ejecución propuesto por Oracle es el siguiente:

En él, Oracle realiza los *filter predicates* mientras hace un *table access* para posteriormente agrupar los datos mediante un hash y luego los organiza.

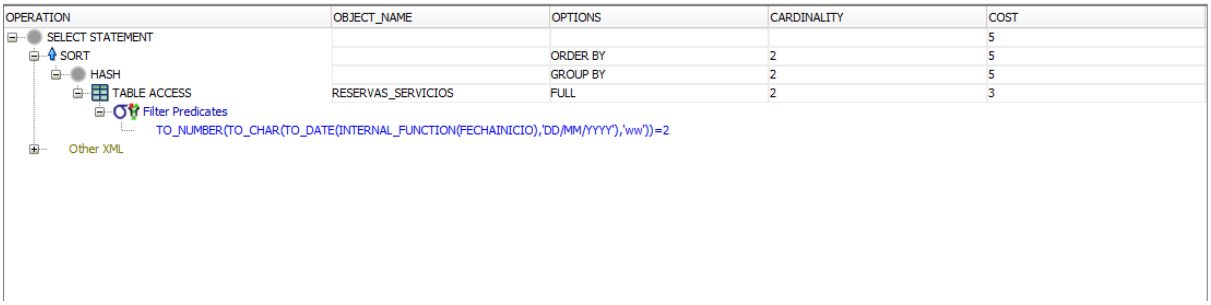


Figura 7. Plan de ejecución propuesto por Oracle

Tipo 2: Muestra, para cada semana del año (sábado a sábado), las habitaciones más solicitadas y las habitaciones menos solicitadas. En Java se realiza un for de 1 a 53 donde la variable es I en la consulta.

Escenarios: Este requerimiento se ve afectado únicamente por la tabla de reservas de alojamiento, es decir, esta es una tabla que no está uniformemente distribuida y que posee un índice primario. Sin embargo, los filtros más importantes dependen de su fecha. Por lo que se ve afectado a medida que incrementa el número de estos, sin embargo, el número de tuplas en las demás tablas es irrelevante.

Índices: No se utiliza ningún índice para esta consulta, sin embargo, podría llegar a ser útil un compuesto por el id y la fecha de llegada teórica.

Tiempo de ejecución: Este oscila entre 350 y 399 milisegundos.

Selectividad: 1%

```
select   idhabitacion
from reservas_de_alojamiento
where  to_char(to_date(FECHALLEGADATEORICA,'DD/MM/YYYY'),'ww') = I
group by
to_char(to_date(FECHALLEGADATEORICA,'DD/MM/YYYY'),'ww'),idhabitacion
order by
(to_char(to_date(FECHALLEGADATEORICA,'DD/MM/YYYY'),'ww')),count(idha
bitacion)  desc
```

El plan de ejecución propuesto por Oracle es el siguiente:

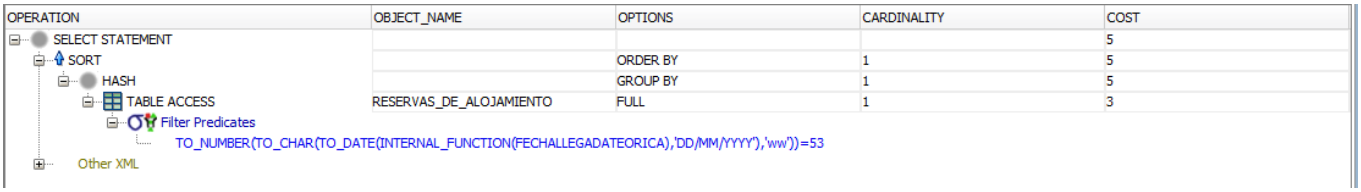


Figura 8. Plan de ejecución propuesto por Oracle

En él, Oracle realiza los *filter predicates* mientras hace un *table access* para posteriormente agrupar los datos mediante un *hash* y luego los organiza.

RFC12 Consultar los buenos clientes: Los buenos clientes son de tres tipos: aquellos que realizan estancias (las estancias están delimitadas por un check in y su respectivo check out) en HotelAndes al menos una vez por trimestre, aquellos que siempre consumen por lo menos un servicio costoso (Entiéndase como costoso, por ejemplo, con un precio mayor a $300.000.oo) y aquellos que en cada estancia consumen servicios de SPA o de salones de reuniones con duración mayor a 4 horas. Esta consulta retorna toda la información de dichos clientes, incluyendo aquella que justifica su calificación como buenos clientes. Esta operación es realizada únicamente por el gerente general de HotelAndes.

Tipo 1: Muestra los buenos clientes que realizan estancias (las estancias están delimitadas por un check in y su respectivo check out) en HotelAndes al menos una vez por trimestre.

Escenarios: En este requerimiento están en juego las tablas PERSONAS RESERVAS_DE_ALOJAMIENTO Y RESERVAS_DE_ALOJAMIENTO. Estas son tablas que no están uniformemente distribuidas y que poseen un índice primario. El aumento de datos en la tabla PERSONAS no aumenta drásticamente la consulta, sin embargo su lo hace el aumento de tuplas en RESERVAS_DE_ALOJAMIENTO y RESERVAS_DE_ALOJAMIENTO ya que sobre estas se hace el primer filtro y luego si lo relaciona con PERSONAS.

Índices: La consulta realiza un INDEX RANGE SCAN sobre PERSONAS, un INDEX UNIQUE SCAN sobre PERSONAS y un INDEX FULL SCAN sobre RESERVAS_DE_CLIENTES. Esto agiliza mucho el proceso ya que los índices ya existen, sin embargo, el mantenimiento del de RESERVAS_DE_CLIENTES es muy costos. Esto es más rápido que traer toda la tabla para realizar estos accesos.

Tiempo de ejecución: entre 50 y 100 milisegundos.

Selectividad: 16%

```
select personas.id,personas.tipoidentificacion,nombre,correo
from personas
inner join (
select id
FROM(
select
to_char(to_date(reservas_de_alojamiento.fechallegadateorica,'DD/MM/YYYY'),
'Q'),count(personas.id),personas.id
from (( reservas_de_clientes
inner join reservas_de_Alojamiento
on reservas_de_clientes.idreserva = reservas_de_alojamiento.id)
inner join personas
on personas.id = reservas_de_clientes.idusuario AND
personas.tipoidentificacion = reservas_de_clientes.tipoidusuario)
where reservas_de_alojamiento.fechallegadateorica >= '18/05/18'
and
to_char(to_date(reservas_de_alojamiento.fechallegadateorica,'DD/MM/YYYY'),
'Q') =
```

```
to_char(to_date(reservas_de_alojamiento.fechasalidateorica,'DD/MM/YYYY'),'
Q')
group                                                                    by
to_char(to_date(reservas_de_alojamiento.fechallegadateorica,'DD/MM/YYYY'),
'Q'),personas.id
having count(*)>0)
group by id
having count(*)=4) resp
on personas.id = resp.id;
```

El plan de ejecución propuesto por Oracle es el siguiente:

Donde primero realiza un `access("PERSONAS"."ID"="RESP"."ID"),`

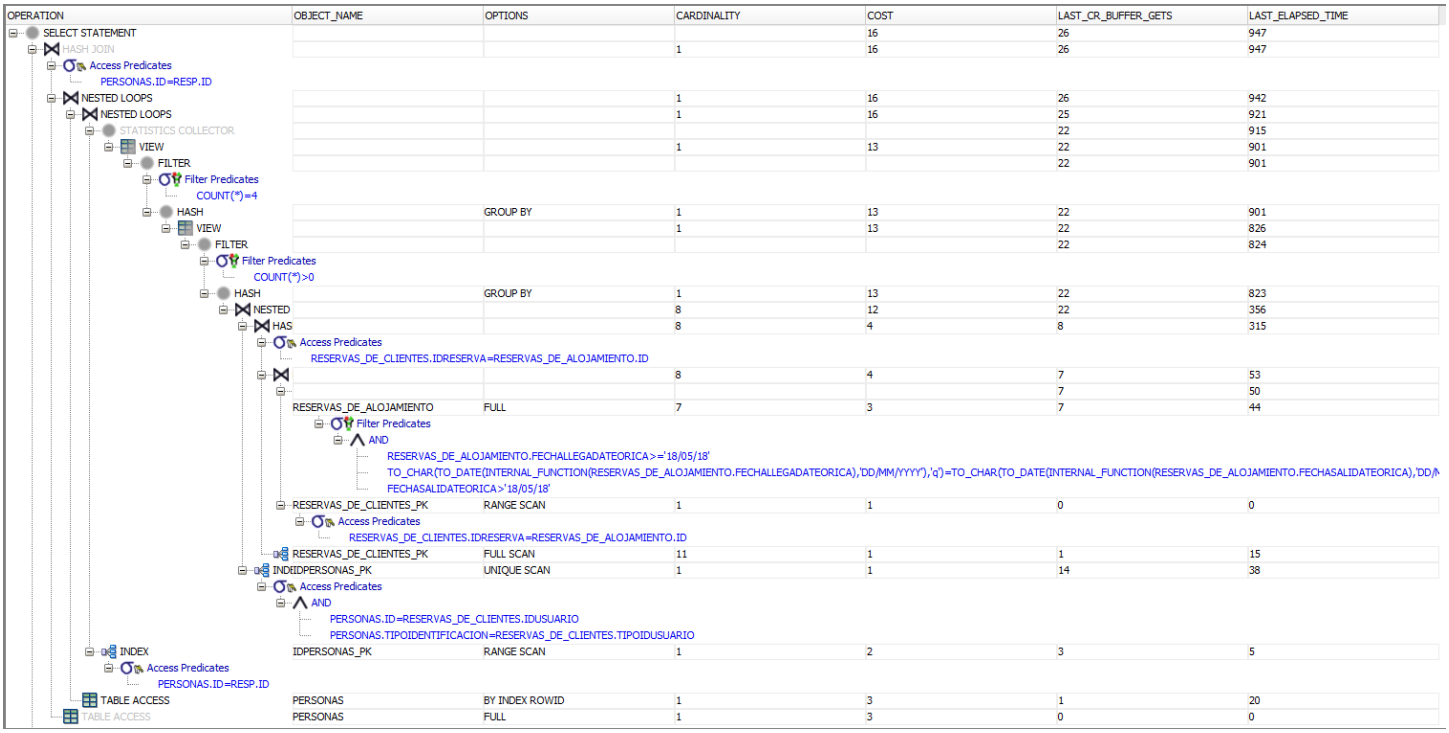
luego un `access("PERSONAS"."ID"="RESERVAS_DE_CLIENTES"."IDUSUARIO" AND "PERSONAS"."TIPOIDENTIFICACION"="RESERVAS_DE_CLIENTES"."TIPOIDUSUARIO"),`

después `filter(("RESERVAS_DE_ALOJAMIENTO"."FECHALLEGADATEORICA">='18/05/18' AND TO_CHAR(TO_DATE(INTERNAL_FUNCTION("RESERVAS_DE_ALOJAMIENTO"."FECHALLEGADATEORICA"),'DD/MM/YY YY'),'q')=TO_CHAR(TO_DATE(INTERNAL_FUNCTION ("RESERVAS_DE_ALOJAMIENTO"."FECHASALIDATEORICA"), 'DD/MM/YYYY'),'q') AND "FECHASALIDATEORICA">'18/05/18'))`

más tarde

`access("RESERVAS_DE_CLIENTES"."IDRESERVA"="RESERVAS_DE_ALOJAMIENTO"."ID")`

y finalmente `filter(COUNT(*)=4) y - filter(COUNT(*)>0)`



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
SELECT STATEMENT				16	26	947
HASH JOIN			1	16	26	947
Access Predicates						
PERSONAS.ID=RESP.ID						
NESTED LOOPS			1	16	26	942
NESTED LOOPS			1	16	25	921
STATISTICS COLLECTOR					22	915
VIEW			1	13	22	901
FILTER					22	901
Filter Predicates						
COUNT(*)=4						
HASH		GROUP BY	1	13	22	901
VIEW			1	13	22	826
FILTER					22	824
Filter Predicates						
COUNT(*)>0						
HASH		GROUP BY	1	13	22	823
NESTED			8	12	22	356
HAS			8	4	8	315
Access Predicates						
RESERVAS_DE_CLIENTES.IDRESERVA=RESERVAS_DE_ALOJAMIENTO.ID						
			8	4	7	53
					7	50
	RESERVAS_DE_ALOJAMIENTO	FULL	7	3	7	44
Filter Predicates						
AND						
RESERVAS_DE_ALOJAMIENTO.FECHALLEGADATEORICA>='18/05/18'						
TO_CHAR(TO_DATE(INTERNAL_FUNCTION(RESERVAS_DE_ALOJAMIENTO.FECHALLEGADATEORICA),'DD/MM/YYYY'),'q')=TO_CHAR(TO_DATE(INTERNAL_FUNCTION(RESERVAS_DE_ALOJAMIENTO.FECHASALIDATEORICA),'DD/M						
FECHASALIDATEORICA>'18/05/18'						
	RESERVAS_DE_CLIENTES_PK	RANGE SCAN	1	1	0	0
Access Predicates						
RESERVAS_DE_CLIENTES.IDRESERVA=RESERVAS_DE_ALOJAMIENTO.ID						
	RESERVAS_DE_CLIENTES_PK	FULL SCAN	11	1	1	15
INDEX	IDPERSONAS_PK	UNIQUE SCAN	1	1	14	38
Access Predicates						
AND						
PERSONAS.ID=RESERVAS_DE_CLIENTES.IDUSUARIO						
PERSONAS.TIPOIDENTIFICACION=RESERVAS_DE_CLIENTES.TIPOIDUSUARIO						
INDEX	IDPERSONAS_PK	RANGE SCAN	1	2	3	5
Access Predicates						
PERSONAS.ID=RESP.ID						
TABLE ACCESS	PERSONAS	BY INDEX ROWID	1	3	1	20
TABLE ACCESS	PERSONAS	FULL	1	3	0	0

Figura 9. Plan de ejecución propuesto por Oracle

Tipo 2: Muestra los buenos clientes que siempre consumen por lo menos un servicio costoso (Entiéndase como costoso, por ejemplo, con un precio mayor a $300.000.oo).

Escenarios: Esta consulta depende de las tablas `PERSONAS, RESERVAS_DE_CLIENTES, RESERVAS_DE_ALOJAMIENTO Y GASTOS`. Principalmente la eficiencia de la consulta va a depender de `RESERVAS_DE_CLIENTES, RESERVAS_DE_ALOJAMIENTO Y GASTOS` que es de donde se hacen la mayoría de los filtros, un aumento en los números de estos causaría una reducción en el tiempo, sin embargo, un aumento en PERSONAS, no la afectaría tanto dado que con esta solo se coteja al final.

Índices: La consulta realiza: `INDEX RANGE SCAN` sobre `PERSONAS, INDEX UNIQUE SCAN` sobre `PERSONAS, INDEX FULL SCAN` sobre `RESERVAS_DE_CLIENTES, INDEX UNIQUE SCAN` sobre `PERSONAS, INDEX FULL SCAN` sobre `RESERVAS_DE_CLIENTES` y un `INDEX UNIQUE SCAN` sobre `IDRESERVASALOJAMIENTO`. Los índices son realmente útiles y aceleran el proceso de gran manera, sin estos recorrer por memoria todas estas operaciones tardaría 4 veces más.

Tiempo de ejecución: entre 50 y 100 milisegundos.

Selectividad: 11%

```
select personas.id,personas.tipoidentificacion,nombre,correo
```

```
from personas
inner join (
select a.id
from (
select personas.id,count(personas.id) as cA
from (( reservas_de_clientes
inner join reservas_de_Alojamiento
on reservas_de_clientes.idreserva = reservas_de_alojamiento.id)
inner join personas
on personas.id = reservas_de_clientes.idusuario AND
personas.tipoidentificacion = reservas_de_clientes.tipoidusuario)
GROUP BY personas.id ) A
inner join (
select resp.id, count(resp.id) as cB
from(
select personas.id
from ((( reservas_de_clientes
inner join reservas_de_Alojamiento
on reservas_de_clientes.idreserva = reservas_de_alojamiento.id)
inner join personas
on personas.id = reservas_de_clientes.idusuario AND
personas.tipoidentificacion = reservas_de_clientes.tipoidusuario)
inner join gastos
on  reservas_de_Alojamiento.id = gastos.idreserva)
where gastos.fecha between reservas_de_alojamiento.fechallegadateorica and
reservas_de_alojamiento.fechasalidateorica
and gastos.precio > 300000
group by reservas_de_Alojamiento.id,personas.id) resp
group by resp.id) B
on a.id = b.id and a.ca = b.cb) C
on personas.id = c.id
```

El plan de ejecución propuesto por Oracle es el siguiente:

Primero hace un access("RESERVAS_DE_ALOJAMIENTO"."ID"="GASTOS"."IDRESERVA")

Luego un
`filter(("GASTOS"."FECHA"<="RESERVAS_DE_ALOJAMIENTO"."FECHASALIDATEORICA" AND "GASTOS"."FECHA">="RESERVAS_DE_ALOJAMIENTO"."FECHALLEGADATEORICA"))`

Más tarde un `filter("GASTOS"."PRECIO">300000)`

Posteriormente `filter("GASTOS"."PRECIO">300000)`

Después `access("PERSONAS"."ID"="ITEM_2" AND "PERSONAS"."TIPOIDENTIFICACION"="ITEM_1")`

Además, `access("PERSONAS"."ID"="A"."ID")`

Y finalmente un `access("A"."ID"="B"."ID" AND "A"."CA"="B"."CB")`

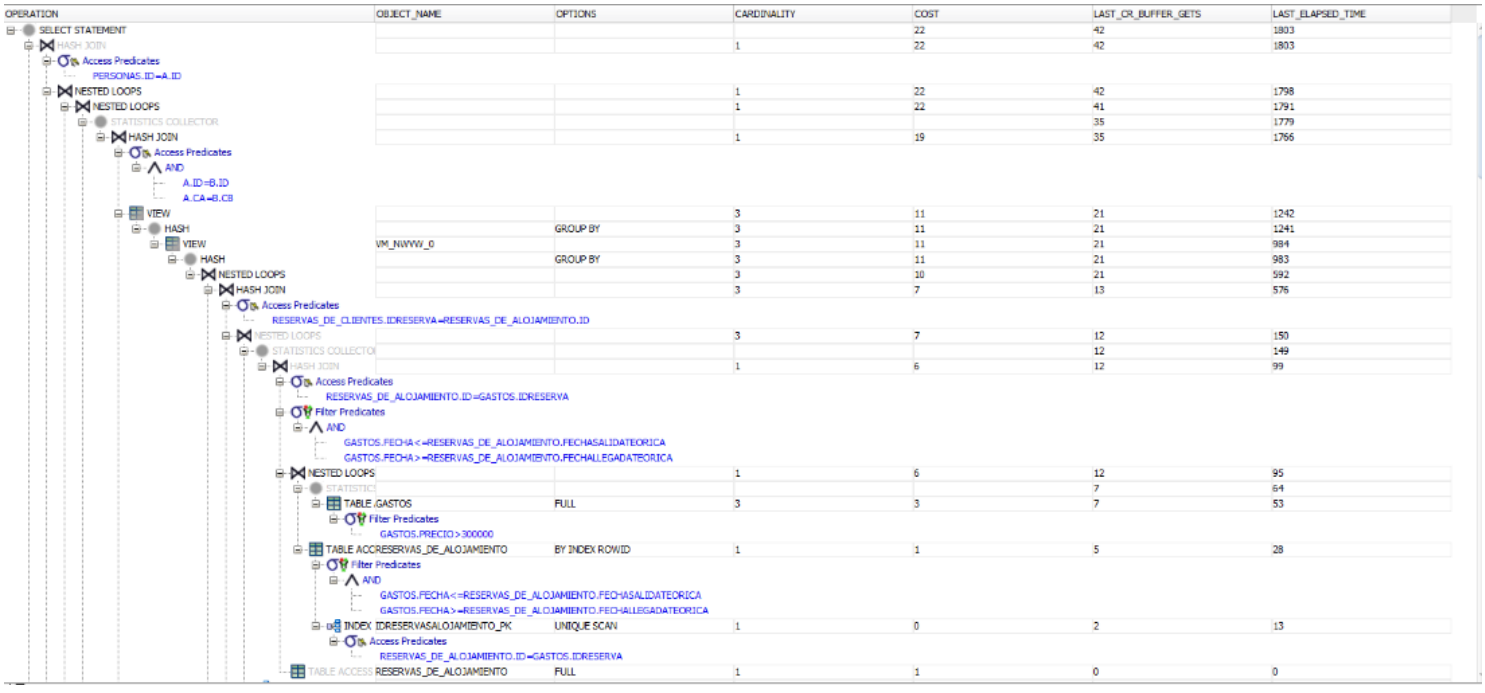


Figura 10. Plan de ejecución propuesto por Oracle

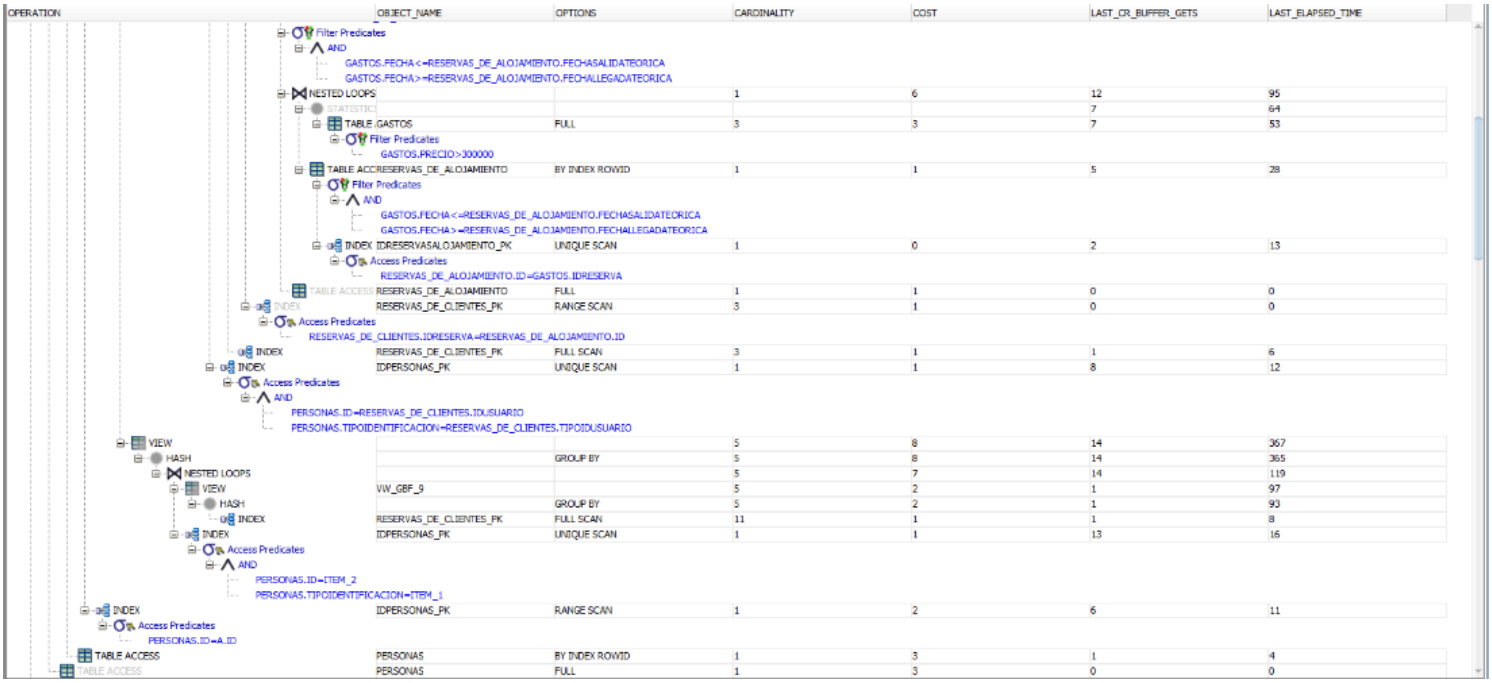


Figura 11. Plan de ejecución propuesto por Oracle

Tipo 3: Muestra los buenos clientes que en cada estancia consumen servicios de SPA o de salones de reuniones con duración mayor a 4 horas.

Escenarios: Esta consulta depende de las tablas PERSONAS, RESERVAS_DE_CLIENTES, RESERVAS_DE_ALOJAMIENTO Y GASTOS. Principalmente la eficiencia de la consulta va a depender de RESERVAS_DE_CLIENTES, RESERVAS_DE_ALOJAMIENTO Y GASTOS que es de donde se hacen la mayoría de los filtros, un aumento en los números de estos causaría una reducción en el tiempo, sin embargo, un aumento en PERSONAS, no la afectaría tanto dado que con esta solo se coteja al final.

Índices: La consulta realiza: `INDEX RANGE SCAN` sobre `PERSONAS`, `INDEX UNIQUE SCAN` sobre `PERSONAS`, `INDEX FULL SCAN` sobre `RESERVAS_DE_CLIENTES`, `INDEX UNIQUE SCAN` sobre `PERSONAS`, `INDEX FULL SCAN` sobre `RESERVAS_DE_CLIENTES` y un `INDEX UNIQUE SCAN` sobre `IDRESERVASALOJAMIENTO`. Los índices son realmente útiles y aceleran el proceso de gran manera, sin estos recorrer por memoria todas estas operaciones tardaría 4 veces más.

Tiempo de ejecución: entre 50 y 100 milisegundos.

Selectividad: 11%

```
select personas.id,personas.tipoidentificacion,nombre,correo
from personas
```

```
inner join (
select a.id
from (
select personas.id,count(personas.id) as cA
from (( reservas_de_clientes
inner join reservas_de_Alojamiento
on reservas_de_clientes.idreserva = reservas_de_alojamiento.id)
inner join personas
on personas.id = reservas_de_clientes.idusuario AND
personas.tipoidentificacion = reservas_de_clientes.tipoidusuario)
GROUP BY personas.id ) A
inner join (
select resp.id, count(resp.id) as cB
from(
select personas.id
from ((( reservas_de_clientes
inner join reservas_de_Alojamiento
on reservas_de_clientes.idreserva = reservas_de_alojamiento.id)
inner join personas
on personas.id = reservas_de_clientes.idusuario AND
personas.tipoidentificacion = reservas_de_clientes.tipoidusuario)
inner join gastos
on  reservas_de_Alojamiento.id = gastos.idreserva)
where gastos.fecha between reservas_de_alojamiento.fechallegadateorica and
reservas_de_alojamiento.fechasalidateorica
and (gastos.idservicio = 12 or gastos.idservicio = 13)
group by reservas_de_Alojamiento.id,personas.id) resp
group by resp.id) B
on a.id = b.id and a.ca = b.cb) C
on personas.id = c.id
```

El plan de ejecución propuesto por Oracle es el siguiente:

Primero hace un `access("RESERVAS_DE_ALOJAMIENTO"."ID"="GASTOS"."IDRESERVA")`

Luego un `filter(("GASTOS"."FECHA"<="RESERVAS_DE_ALOJAMIENTO"."FECHASALIDATEORICA" AND "GASTOS"."FECHA">="RESERVAS_DE_ALOJAMIENTO"."FECHALLEGADATEORICA"))`

Mas tarde un `filter("GASTOS"."PRECIO">300000)`

Posteriormente `filter("GASTOS"."PRECIO">300000)`

Después `access("PERSONAS"."ID"="ITEM_2" AND "PERSONAS"."TIPOIDENTIFICACION"="ITEM_1")`

Además, `access("PERSONAS"."ID"="A"."ID")`

Y finalmente un `access("A"."ID"="B"."ID" AND "A"."CA"="B"."CB")`

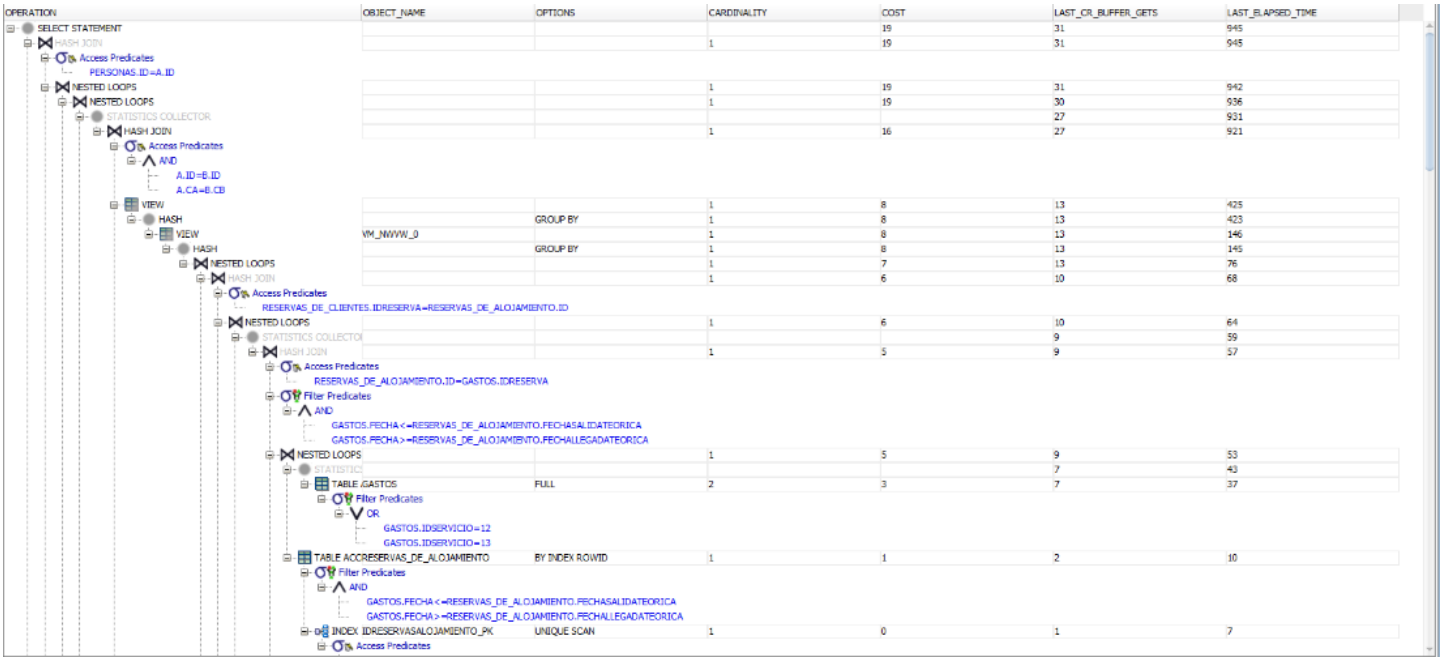
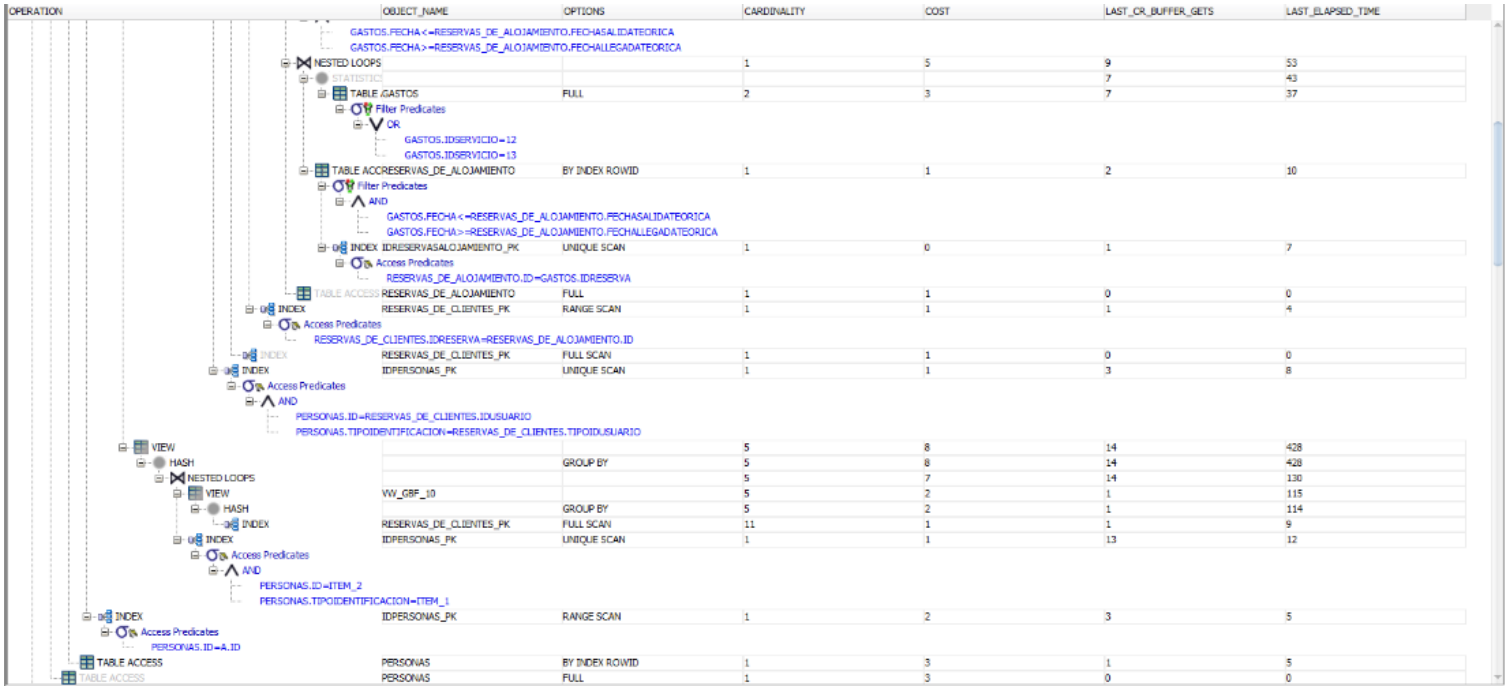


Figura 12. Plan de ejecución propuesto por Oracle



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_CR_BUFFER_GETS	LAST_ELAPSED_TIME
GASTOS.FECHA<=RESERVAS_DE_ALOJAMIENTO.FECHASALIDATEORICA						
GASTOS.FECHA>=RESERVAS_DE_ALOJAMIENTO.FECHALLEGADATEORICA						
NESTED LOOPS			1	5	9	53
STATISTICS					7	43
TABLE ACCESS	GASTOS	FULL	2	3	7	37
Filter Predicates						
OR						
GASTOS.IDSERVICIO=12						
GASTOS.IDSERVICIO=13						
TABLE ACCESS	RESERVAS_DE_ALOJAMIENTO	BY INDEX ROWID	1	1	2	10
Filter Predicates						
AND						
GASTOS.FECHA<=RESERVAS_DE_ALOJAMIENTO.FECHASALIDATEORICA						
GASTOS.FECHA>=RESERVAS_DE_ALOJAMIENTO.FECHALLEGADATEORICA						
INDEX	IDRESERVASALOJAMIENTO_PK	UNIQUE SCAN	1	0	1	7
Access Predicates						
RESERVAS_DE_ALOJAMIENTO.ID=GASTOS.IDRESERVA						
TABLE ACCESS	RESERVAS_DE_ALOJAMIENTO	FULL	1	1	0	0
INDEX	RESERVAS_DE_CLIENTES_PK	RANGE SCAN	1	1	1	4
Access Predicates						
RESERVAS_DE_CLIENTES.IDRESERVA=RESERVAS_DE_ALOJAMIENTO.ID						
INDEX	RESERVAS_DE_CLIENTES_PK	FULL SCAN	1	1	0	0
INDEX	IDPERSONAS_PK	UNIQUE SCAN	1	1	3	8
Access Predicates						
AND						
PERSONAS.ID=RESERVAS_DE_CLIENTES.IDUSUARIO						
PERSONAS.TIPOIDENTIFICACION=RESERVAS_DE_CLIENTES.TIPOIDUSUARIO						
VIEW			5	8	14	428
HASH		GROUP BY	5	8	14	428
NESTED LOOPS			5	7	14	130
VIEW	VW_GBF_10		5	2	1	115
HASH		GROUP BY	5	2	1	114
INDEX	RESERVAS_DE_CLIENTES_PK	FULL SCAN	11	1	1	9
INDEX	IDPERSONAS_PK	UNIQUE SCAN	1	1	13	12
Access Predicates						
AND						
PERSONAS.ID=ITEM_2						
PERSONAS.TIPOIDENTIFICACION=ITEM_1						
INDEX	IDPERSONAS_PK	RANGE SCAN	1	2	3	5
Access Predicates						
PERSONAS.ID=A.ID						
TABLE ACCESS	PERSONAS	BY INDEX ROWID	1	3	1	5
TABLE ACCESS	PERSONAS	FULL	1	3	0	0

Figura 13. Plan de ejecución propuesto por Oracle