Diseño de la aplicación:

1. Para esta iteración el modelo planteado desde la segunda iteración no ha presentado cambios drásticos en cuanto a estructura relacional y al respaldo de persistencia de la aplicación. Los nuevos requerimientos funcionales a realizar eran únicamente de consulta por lo que no era necesario implementar modificaciones en la BD. Por el otro lado, para la iteración si fue necesario la creación de índices sobre algunas columnas de las tablas con el fin de volver el tiempo y costo de acceso más eficiente debido a la cantidad de datos utilizados.

Diseño físico:

- Justificaciones índices.
    - RF9

      En el noveno requerimiento era necesario hacer una consulta donde se involucraba nuestras tablas: compania, usuarios, funciones, boletas y espectaculos. Al incluir estas 5 tablas y cada una poblada con al menos 10000 datos era necesario imprentar índices sobre las columnas por la cuales se filtraba algún parámetro y aquellas donde se realizaba el inner o natural Join. Fue de esta manera que decidimos implementar índices B+, bitmap, primarios y compuestos. En primer lugar, utilizamos índices B+, o conocidos en Oracle Sql Developer como "normal", para aquellas columnas que son llaves primarias. Las implementaciones de índices para llaves primarias son de gran ayuda sobre todo en las búsquedas sobre predicados de igualdad (=) debido a que el IndexScan que se realiza es único, pues tanto la cardinalidad como el costo es bajo. De igual forma, tanto el costo como la cardinalidad se reducen drásticamente utilizando un índice normal pero compuesto. Estos fueron usados en la tabla espectaculo sobre las columnas id_Espectaculo e ID_Compania, se da prelación al id_espectaciulo para filtrar por predicado con el fin de realizar el hash join, y el ID_espectaculo para cumplir con el predicado de igualdad, mediante una búsqueda de rango con el índice. En cuanto a índices normales simples se encuentra el índice aplicado sobre la tabla función y la columna id_espectaculo con el fin de realizar una igualdad de predicado más eficaz. Por último se implementaron índices Bitmap para los campos ID_Funcion e ID_Silla en la tabla boleta donde vale destacar que la única llave primaria es ID_Boleta, por ende al realizar un índice bitmap sobre estos campos realiza un escaneo de rango mucho más eficiente en espacio y tiempo.

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS	COLUMN_EXPRESSION
1	ISIS2304A241720	ID_ESPEC_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPEC	(null)
2	ISIS2304A241720	INDEX_ESPEC_COMPA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPEC, ID_COMPANIA	(null)

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS	COLUMN_EXPRESSION
1	ISIS2304A241720	INDEX_FULL_BO	NONUNIQUE	VALID	BITMAP	N	NO	(null)	NO	ID_FUNCION, ID_SILLA	(null)
2	ISIS2304A241720	ID_BOLETA_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_BOLETA	(null)
3	ISIS2304A241720	ESTADO_INDEX	NONUNIQUE	VALID	BITMAP	N	NO	(null)	NO	ESTADO	(null)
4	ISIS2304A221720	INDEX_IDSILLA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_SILLA	(null)

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS	COLUMN_EXPRESSION
1	ISIS2304A241720	INDEX_IDESPEC	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPECTACULO	(null)
2	ISIS2304A241720	ID_FUNCION_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_FUNCION	(null)
3	ISIS2304A221720	INDEX_FUNC_ESPEC	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_FUNCION, ID_ESPECTACULO	(null)

    - RF10

      Debido a nuestro inicial diseño respecto a la BD, este requerimiento se lograba cumplir negando una sentencia del RF9. Esta sentencia es aquella donde se dice que el estado ='A' donde 'A' significa activo. Para este requerimiento el único cambio necesario fue estado='D' donde 'D' significa devuelto, es decir que el usuario no asistió a la función. Por ende, la explicación de índices y el plan de consulta es básicamente el mismo que el RF9.

- RF11

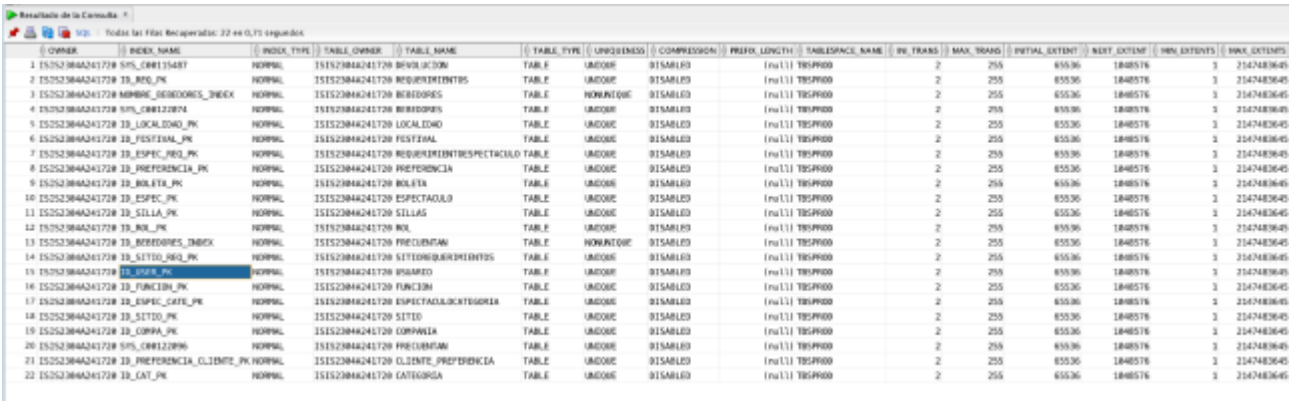
El requerimiento 11 debía mostrar la información de las boletas dependiendo de ciertos críterios dados por el usuario sobre la función. Para ello se decidío implementar índices sobre los críterios de busqueda y sobre aquellos atributos o columnas donde se realizan los joins. En cuanto a los índices implementados y utilizados para esta sentencia, se utilizó un índice compuesto de tipo normal o B+ para la tabla requerimientoEspectaculo aplicado a las columnas ID_requerimiento y ID_Espectaculo. Debido a que en la sentencia se realiza un inner join entre la tabla 5 y requerimientos donde la columna de join es id_requerimiento, la tabla 5 es el resultado de un inner join entre requerimientoEspectaculo y la tabla 4 donde la columna igualdad es esl id_espec. Por tal motivo se decidió implementar tales índices en ese orden. En la mitad de la sentencia se ejecuta un inner join entre función y espectáculo mediante el id_espec, luego se realiza un acceso a predicados en un intervalo dado. El índice aplicado para esta parte de la sentencia causa en la disminución drástica de costos de ejecución, esto se da por realizar el id compuesto entre ID_Espectaculo y hora.

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS	COLUMN_EXPRESSION
1	ISIS2304A241720	INDEX_IDESPEC	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPECTACULO	(null)
2	ISIS2304A241720	ID_FUNCION_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_FUNCION	(null)
3	ISIS2304A241720	INDEX_FUN_ESPEC_HORA	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPECTACULO, HORA	(null)
4	ISIS2304A221720	INDEX_FUNC_ESPEC	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_FUNCION, ID_ESPECTACULO	(null)

	INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS	COLUMN_EXPRESSION
1	ISIS2304A241720	INDEX_PK3	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_REQUERIMIENTO, ID_ESPECTACULO	(null)
2	ISIS2304A241720	ID_ESPEC_REQ_PK	UNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_ESPECTACULO, ID_REQUERIMIENTO	(null)
3	ISIS2304A241720	INDEX_PK2	NONUNIQUE	VALID	NORMAL	N	NO	(null)	NO	ID_REQUERIMIENTO	(null)

- RF12

Índices generados automáticamente por Oracle



	OWNER	INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS	COMPRESSION	PREFIX_LENGTH	TABLESPACE_NAME	INI_TRANS	MAX_TRANS	INITIAL_EXTENT	NEXT_EXTENT	MIN_EXTENTS	MAX_EXTENTS
1	ISIS2304A241720	SYS_C0013587	NORMAL	ISIS2304A241720	BEVOLUCION	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
2	ISIS2304A241720	ID_REQ_PK	NORMAL	ISIS2304A241720	REQUERIMIENTOS	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
3	ISIS2304A241720	NOMBRE_BEBEDORES_INDEX	NORMAL	ISIS2304A241720	BEBEDORES	TABLE	NONUNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
4	ISIS2304A241720	SYS_C0012074	NORMAL	ISIS2304A241720	BEBEDORES	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
5	ISIS2304A241720	ID_LOCALIDAD_PK	NORMAL	ISIS2304A241720	LOCALIDAD	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
6	ISIS2304A241720	ID_FESTIVAL_PK	NORMAL	ISIS2304A241720	FESTIVAL	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
7	ISIS2304A241720	ID_ESPEC_REQ_PK	NORMAL	ISIS2304A241720	REQUERIMIENTOESPECTACULO	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
8	ISIS2304A241720	ID_PREFERENCIA_PK	NORMAL	ISIS2304A241720	PREFERENCIA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
9	ISIS2304A241720	ID_BOLETA_PK	NORMAL	ISIS2304A241720	BOLETA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
10	ISIS2304A241720	ID_ESPEC_PK	NORMAL	ISIS2304A241720	ESPECTACULO	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
11	ISIS2304A241720	ID_SILLA_PK	NORMAL	ISIS2304A241720	SILLAS	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
12	ISIS2304A241720	ID_ROL_PK	NORMAL	ISIS2304A241720	ROL	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
13	ISIS2304A241720	ID_BEBEDORES_INDEX	NORMAL	ISIS2304A241720	PRECUENTAN	TABLE	NONUNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
14	ISIS2304A241720	ID_SITIO_REQ_PK	NORMAL	ISIS2304A241720	SITIOSREQUERIMIENTOS	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
15	ISIS2304A241720	ID_USUA_PK	NORMAL	ISIS2304A241720	USUARIO	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
16	ISIS2304A241720	ID_FUNCION_PK	NORMAL	ISIS2304A241720	FUNCION	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
17	ISIS2304A241720	ID_ESPEC_CATE_PK	NORMAL	ISIS2304A241720	ESPECTACULOCATEGORIA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
18	ISIS2304A241720	ID_SITIO_PK	NORMAL	ISIS2304A241720	SITIO	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
19	ISIS2304A241720	ID_COMPA_PK	NORMAL	ISIS2304A241720	COMPAÑIA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
20	ISIS2304A241720	SYS_C0012096	NORMAL	ISIS2304A241720	PRECUENTAN	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
21	ISIS2304A241720	ID_PREFERENCIA_CLIENTE_PK	NORMAL	ISIS2304A241720	CLIENTE_PREFERENCIA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645
22	ISIS2304A241720	ID_CAT_PK	NORMAL	ISIS2304A241720	CATEGORIA	TABLE	UNIQUE	DISABLED	(null)	TBSPP00	2	255	65536	1048576	1	2147483645

Cuando se crea una tabla Oracle automáticamente genera un índice único sobre la columna de la llave primaria para generar el constraint de que no se puedan repetir. Como podemos ver en la imagen hasta el momento solo existen indices para la llave primaria de cada tabla. Estos índices si mejoran el rendimiento de las sentencias ya que por ejemplo en las sentencias 1,2 y 4 debemos dar la información de los clientes que tienen ciertas características sobre sus boletas por ende después de hacer el filtro que nos permite conocer cuáles son estos clientes debemos buscarlos en la tabla de usuarios y estos índices nos ayudan a que no tengamos que buscar en toda la tabla sino llegar solamente a los índices que coincidan con lo que estamos buscando.

Documentación escenarios de prueba

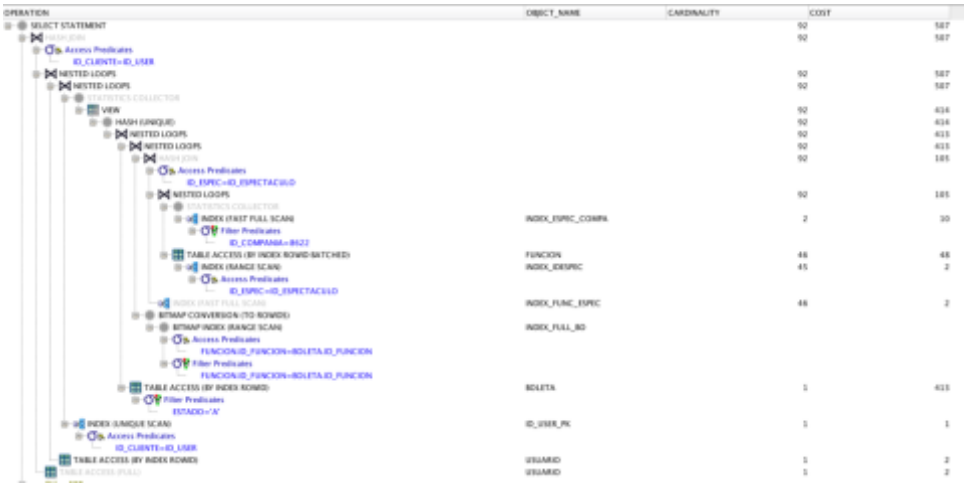
Requerimiento 9:

- Sentencia: with tabla1 as (select id_funcion as id_fun from ((select id_espec from ISIS2304A241720.espectaculo where id_compania = 19)

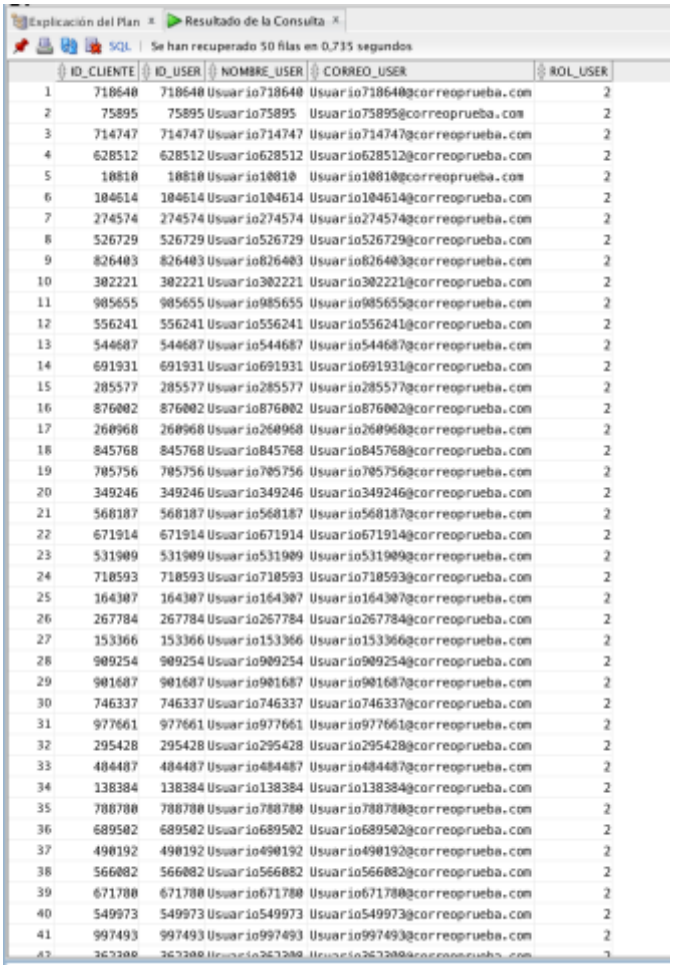
  inner join ISIS2304A241720.funcion on id_espec = id_espectaculo)), tabla2 as (select distinct id_cliente from tabla1 inner join (select * from ISIS2304A241720.boleta where estado = 'A') on id_fun = id_funcion)

  select * from (tabla2 inner join ISIS2304A241720.usuario on id_cliente = id_user);

- Distribución de los datos: Los datos estan distribuidos asimetricamente por lo cual las sentencias no tienen la misma duración. Esto quiere decir que no hay el mismo numero de boletas para todas las funciones, y que no todas tienen boletas tampoco. Por ende, dependiendo del numero de la compañía que entre por parametros los tiempos de ejecucion no seran los mismos gracias a los indices. Ademas tampoco es simetrica entre las boletas asistidas y las devueltas por lo cual esto tambien interfiere en el tiempo de ejecucion de la sentencia.
- Valores de los parámetros: id compania = 8622 y 2423 los tiempos de ejecucion son distintos, aunque el plan es el mismo debido a la asimetria de los datos
- Plan de ejecución:



- Tiempo de ejecución: t1=0,735 segundos

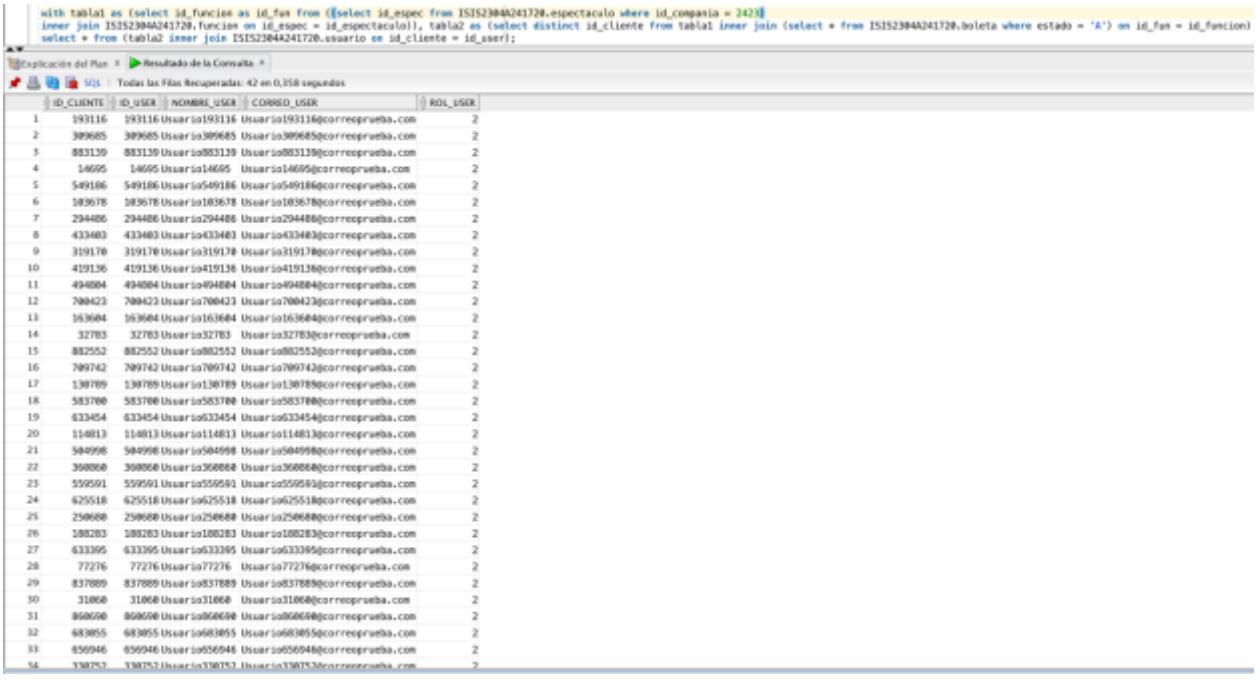


Explicación del Plan | Resultado de la Consulta

Se han recuperado 50 filas en 0,735 segundos

	ID_CLIENTE	ID_USER	NOMBRE_USER	CORREO_USER	ROL_USER
1	718640	718640	Usuario718640	Usuario718640@correoprueba.com	2
2	75895	75895	Usuario75895	Usuario75895@correoprueba.com	2
3	714747	714747	Usuario714747	Usuario714747@correoprueba.com	2
4	628512	628512	Usuario628512	Usuario628512@correoprueba.com	2
5	10810	10810	Usuario10810	Usuario10810@correoprueba.com	2
6	104614	104614	Usuario104614	Usuario104614@correoprueba.com	2
7	274574	274574	Usuario274574	Usuario274574@correoprueba.com	2
8	526729	526729	Usuario526729	Usuario526729@correoprueba.com	2
9	826403	826403	Usuario826403	Usuario826403@correoprueba.com	2
10	302221	302221	Usuario302221	Usuario302221@correoprueba.com	2
11	985655	985655	Usuario985655	Usuario985655@correoprueba.com	2
12	556241	556241	Usuario556241	Usuario556241@correoprueba.com	2
13	544687	544687	Usuario544687	Usuario544687@correoprueba.com	2
14	691931	691931	Usuario691931	Usuario691931@correoprueba.com	2
15	285577	285577	Usuario285577	Usuario285577@correoprueba.com	2
16	876002	876002	Usuario876002	Usuario876002@correoprueba.com	2
17	260968	260968	Usuario260968	Usuario260968@correoprueba.com	2
18	845768	845768	Usuario845768	Usuario845768@correoprueba.com	2
19	705756	705756	Usuario705756	Usuario705756@correoprueba.com	2
20	349246	349246	Usuario349246	Usuario349246@correoprueba.com	2
21	568187	568187	Usuario568187	Usuario568187@correoprueba.com	2
22	671914	671914	Usuario671914	Usuario671914@correoprueba.com	2
23	531909	531909	Usuario531909	Usuario531909@correoprueba.com	2
24	710593	710593	Usuario710593	Usuario710593@correoprueba.com	2
25	164307	164307	Usuario164307	Usuario164307@correoprueba.com	2
26	267784	267784	Usuario267784	Usuario267784@correoprueba.com	2
27	153366	153366	Usuario153366	Usuario153366@correoprueba.com	2
28	909254	909254	Usuario909254	Usuario909254@correoprueba.com	2
29	901687	901687	Usuario901687	Usuario901687@correoprueba.com	2
30	746337	746337	Usuario746337	Usuario746337@correoprueba.com	2
31	977661	977661	Usuario977661	Usuario977661@correoprueba.com	2
32	295428	295428	Usuario295428	Usuario295428@correoprueba.com	2
33	484487	484487	Usuario484487	Usuario484487@correoprueba.com	2
34	138384	138384	Usuario138384	Usuario138384@correoprueba.com	2
35	788780	788780	Usuario788780	Usuario788780@correoprueba.com	2
36	689502	689502	Usuario689502	Usuario689502@correoprueba.com	2
37	490192	490192	Usuario490192	Usuario490192@correoprueba.com	2
38	566082	566082	Usuario566082	Usuario566082@correoprueba.com	2
39	671780	671780	Usuario671780	Usuario671780@correoprueba.com	2
40	549973	549973	Usuario549973	Usuario549973@correoprueba.com	2
41	997493	997493	Usuario997493	Usuario997493@correoprueba.com	2

T2= 0,358 Segundos



```
with tabla1 as (select id_funcion as id_fun from ((select id_espec from ISIS2304A241720.espectaculo where id_compania = 2423)
inner join ISIS2304A241720.funcion on id_espec = id_espectaculo)), tabla2 as (select distinct id_cliente from tabla1 inner join (select * from ISIS2304A241720.boleta where estado = 'A') on id_fun = id_funcion)
select * from (tabla2 inner join ISIS2304A241720.usuario on id_cliente = id_user);
```

Explicación del Plan | Resultado de la Consulta

Todas las Filas Recuperadas: 42 en 0,358 segundos

	ID_CLIENTE	ID_USER	NOMBRE_USER	CORREO_USER	ROL_USER
1	193116	193116	Usuario193116	Usuario193116@correoprueba.com	2
2	389685	389685	Usuario389685	Usuario389685@correoprueba.com	2
3	883139	883139	Usuario883139	Usuario883139@correoprueba.com	2
4	14695	14695	Usuario14695	Usuario14695@correoprueba.com	2
5	549186	549186	Usuario549186	Usuario549186@correoprueba.com	2
6	103678	103678	Usuario103678	Usuario103678@correoprueba.com	2
7	294486	294486	Usuario294486	Usuario294486@correoprueba.com	2
8	433403	433403	Usuario433403	Usuario433403@correoprueba.com	2
9	319170	319170	Usuario319170	Usuario319170@correoprueba.com	2
10	419136	419136	Usuario419136	Usuario419136@correoprueba.com	2
11	494804	494804	Usuario494804	Usuario494804@correoprueba.com	2
12	700423	700423	Usuario700423	Usuario700423@correoprueba.com	2
13	163604	163604	Usuario163604	Usuario163604@correoprueba.com	2
14	32783	32783	Usuario32783	Usuario32783@correoprueba.com	2
15	882552	882552	Usuario882552	Usuario882552@correoprueba.com	2
16	709742	709742	Usuario709742	Usuario709742@correoprueba.com	2
17	130789	130789	Usuario130789	Usuario130789@correoprueba.com	2
18	583700	583700	Usuario583700	Usuario583700@correoprueba.com	2
19	633454	633454	Usuario633454	Usuario633454@correoprueba.com	2
20	114813	114813	Usuario114813	Usuario114813@correoprueba.com	2
21	504998	504998	Usuario504998	Usuario504998@correoprueba.com	2
22	360860	360860	Usuario360860	Usuario360860@correoprueba.com	2
23	559591	559591	Usuario559591	Usuario559591@correoprueba.com	2
24	625518	625518	Usuario625518	Usuario625518@correoprueba.com	2
25	250688	250688	Usuario250688	Usuario250688@correoprueba.com	2
26	188283	188283	Usuario188283	Usuario188283@correoprueba.com	2
27	633395	633395	Usuario633395	Usuario633395@correoprueba.com	2
28	77276	77276	Usuario77276	Usuario77276@correoprueba.com	2
29	837889	837889	Usuario837889	Usuario837889@correoprueba.com	2
30	31060	31060	Usuario31060	Usuario31060@correoprueba.com	2
31	860690	860690	Usuario860690	Usuario860690@correoprueba.com	2
32	683055	683055	Usuario683055	Usuario683055@correoprueba.com	2
33	656946	656946	Usuario656946	Usuario656946@correoprueba.com	2
34	330753	330753	Usuario330753	Usuario330753@correoprueba.com	2

Requerimiento 10

- Sentencia: with tabla1 as (select id_funcion as id_fun from ((select id_espec from ISIS2304A241720.espectaculo where id_compania = 19)

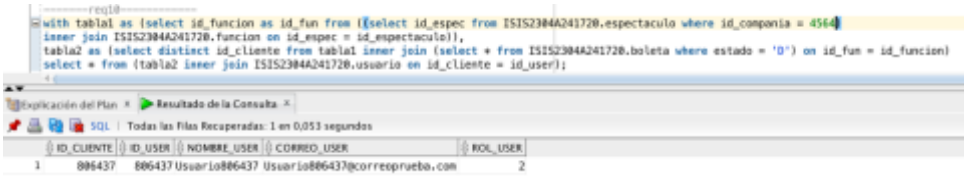
  inner join ISIS2304A241720.funcion on id_espec = id_espectaculo)),

  tabla2 as (select distinct id_cliente from tabla1 inner join (select * from ISIS2304A241720.boleta where estado = 'D') on id_fun = id_funcion)

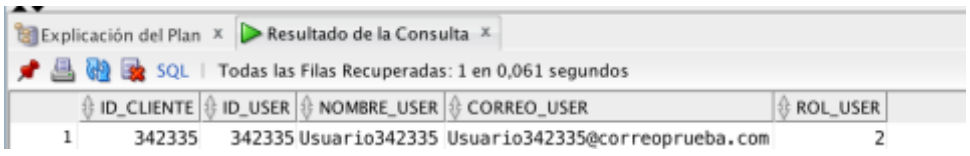
  select * from (tabla2 inner join ISIS2304A241720.usuario on id_cliente = id_user);

- Distribución de los datos: Asi como en el literal anterior los datos no estan distribuidos simetricamente en cuanto a la cantidad de boletas que tiene cada compañía por cliente esto quiere decir que dependiendo de la compania que busquemos los tiempos de ejecucion seran distintos debido a los indices que nos permiten llegar a esta informacion
- Valores de los parámetros: Para esta usamos id compania= 4564 y 1131
- Plan de ejecución:

- Tiempo de ejecución: t1=0,053 segundos

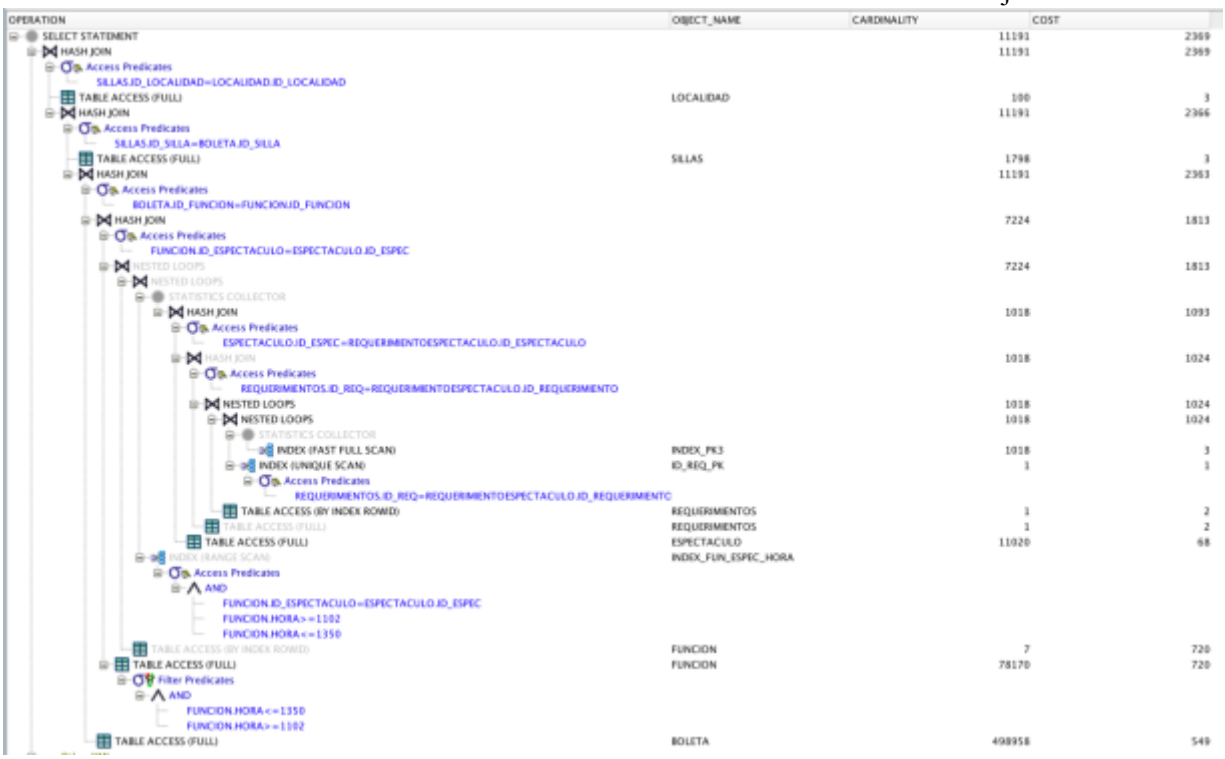


T2 = 0,061 segundos

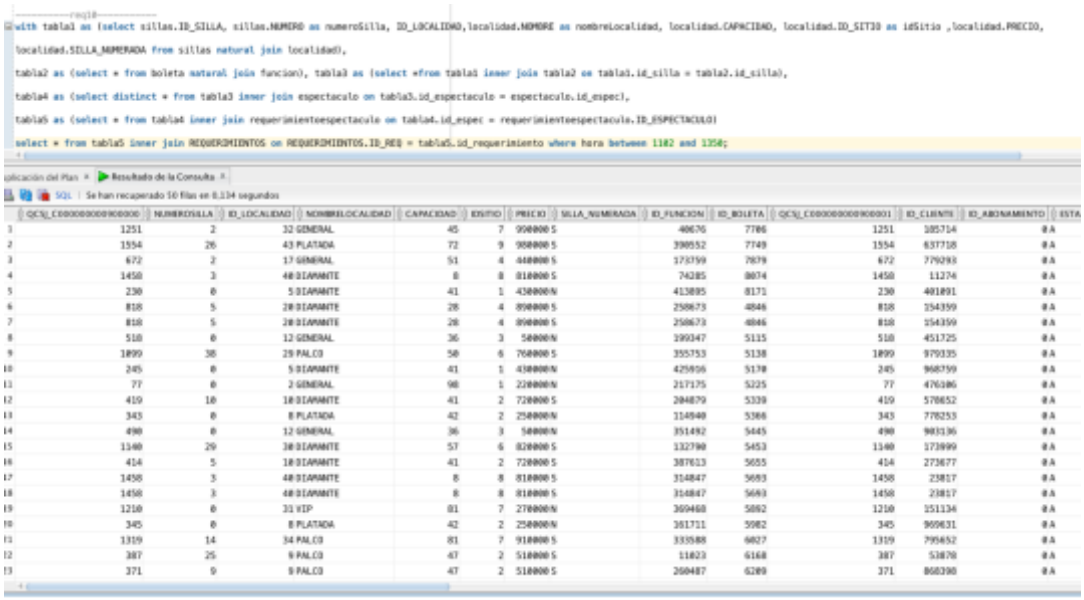


Requerimiento 11

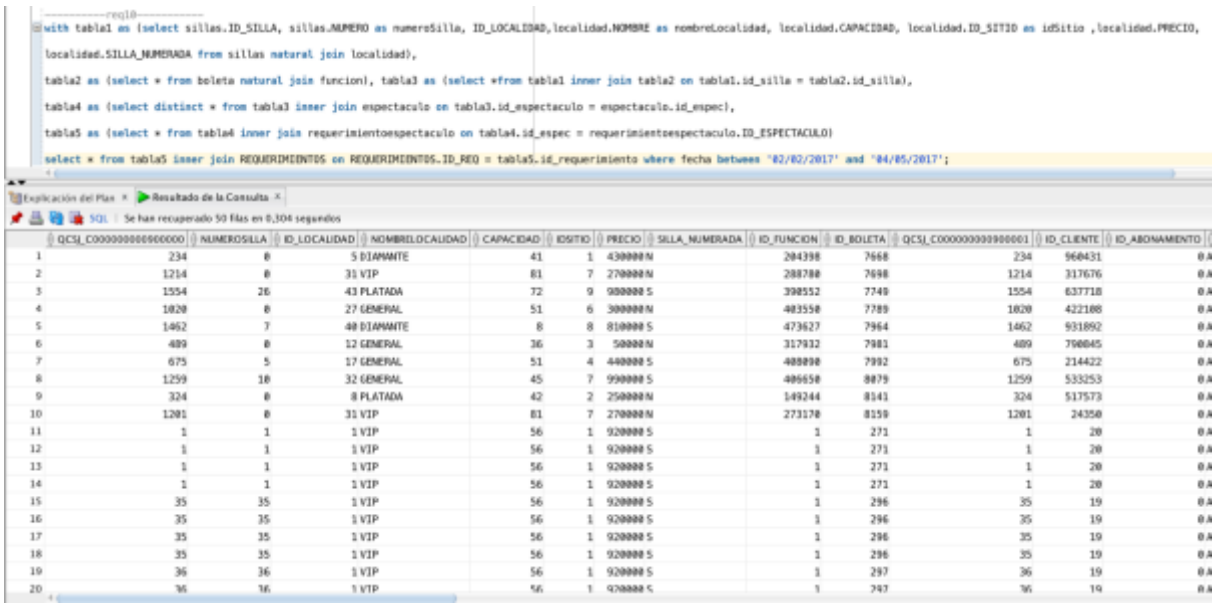
- Sentencia: with tabla1 as (select * from sillas natural join localidad), tabla2 as (select * from boleta natural join funcion), tabla3 as (select *from tabla1 inner join tabla2 on tabla1.id_silla = tabla2.id_silla), tabla4 as (select distinct * from tabla3 inner join espectaculo on tabla3.id_espectaculo = espectaculo.id_espec), tabla5 as (select * from tabla4 inner join requerimientoespectaculo on tabla4.id_espec = requerimientoespectaculo.ID_ESPECTACULO) select * from tabla5 inner join REQUERIMIENTOS on REQUERIMIENTOS.ID_REQ = tabla5.id_requerimiento where fecha between '02/02/2017' and '05/07/2017';
- Distribucion de los datos: Para este requerimiento tenemos muchos mas criterios por los cuales evaluar la sentencia dependiendo de las preferencias del cliente. Sin embargo la tabla no esta simetricamente distribuida por estos parametros. Debido a esto las sentencias no se demoraran lo mismo porque la cantidad de datos que encontrara gracias a los indices no siempre sera la misma
- Valores de los parámetros: 1350>Franja horaria >1120
- Plan de ejecución:



Tiempo de ejecución: t1=0,134 segundos



T2 = 0,304 segundos



Requerimiento 12

- Sentencia: with tabla1 as (select id_boleta, id_cliente, nombre from (select id_boleta, id_silla, id_cliente, id_localidad from ((select * from ISIS2304A241720.boleta) natural join (select * from ISIS2304A241720.sillas))) natural join ISIS2304A241720.localidad), tabla3 as(select id_boleta, id_cliente, nombre from tabla1 where nombre != 'VIP') select * from ((select id_cliente as id_user from(select distinct id_cliente, count(*) as cantidad from(Select * from (select id_cliente from tabla1 minus select id_cliente from tabla3) natural join tabla1) group by id_cliente) where cantidad >= &cantidad) natural join ISIS2304A241720.usuario);
- Distribución de los datos: Los datos de las boletas no estan simetricamente distribuidos con respecto a las localidades de las boletas ni porque un cliente compre solo un tipo de localidad por esto las diferentes cantidades que pongamos nos traeran diferentes resultados y por ende tomaran tiempos distintos
- Valores de los parámetros: cantidad = 4, cantidad = 6
- Plan de ejecución

- 

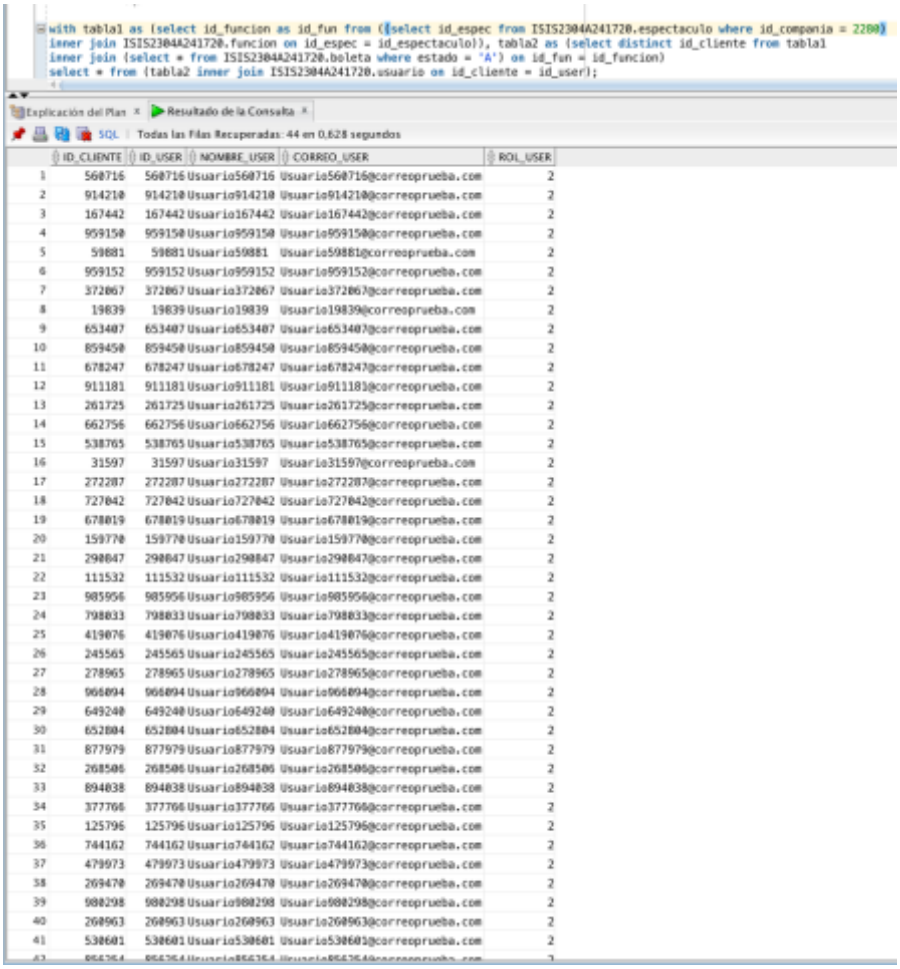
CASOS DE PRUEBA:

1. RF9:
   a. Caso de prueba 1:

      Se realizará el requerimiento funcional de consulta número 9 con el parámetro de consulta tal que el id de la compañía es el 2280. Se espera como resultado al menos una tupla con la información requerida por el requerimiento.

El plan de ejecución que creemos que sucederá es: Debido a que la sentencia tiene varias vistas de tablas (views) involucradas, se resolverán primero los select * más internos de la sentencia donde estos serían select * from espectaculo where id_compania = 2280, para esta sentencia se espera que se haga el filtro de predicado usando el índice index_espec_compa. Luego se espera que resuelva un inner join con la tabla función bajo los valores de id_espectaculo donde se pretende que use el mismo índice anteriormente descrito. Después, se espera que para realizar la sentencia que contiene la igualdad estado='A' use el índice bitmap creado sobre la columna estado para resolver la sentencia mucho más eficaz. Con el resultado de la anterior sentencia, puede ocurrir el inner join con la tabla cliente bajo el id_cliente como relación de las dos tablas. Valga aclarar que se utiliza una columna Estado para identificar si la boleta aún está activa o fue devuelta antes que se realizará la función

Resultado: 44 tuplas recuperadas en 0,620 segundos



```
with tabla1 as (select id_funcion as id_fun from ((select id_espec from ISIS2304A241720.espectaculo where id_compania = 2280)
inner join ISIS2304A241720.funcion on id_espec = id_espectaculo)), tabla2 as (select distinct id_cliente from tabla1
inner join (select * from ISIS2304A241720.boleta where estado = 'A') on id_fun = id_funcion)
select * from (tabla2 inner join ISIS2304A241720.usuario on id_cliente = id_user);
```

Explicación del Plan | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 44 en 0,628 segundos

	ID_CLIENTE	ID_USER	NOMBRE_USER	CORREO_USER	ROL_USER
1	560716	560716	Usuario560716	Usuario560716@correoprueba.com	2
2	914210	914210	Usuario914210	Usuario914210@correoprueba.com	2
3	167442	167442	Usuario167442	Usuario167442@correoprueba.com	2
4	959150	959150	Usuario959150	Usuario959150@correoprueba.com	2
5	59881	59881	Usuario59881	Usuario59881@correoprueba.com	2
6	959152	959152	Usuario959152	Usuario959152@correoprueba.com	2
7	372067	372067	Usuario372067	Usuario372067@correoprueba.com	2
8	19839	19839	Usuario19839	Usuario19839@correoprueba.com	2
9	653407	653407	Usuario653407	Usuario653407@correoprueba.com	2
10	859450	859450	Usuario859450	Usuario859450@correoprueba.com	2
11	678247	678247	Usuario678247	Usuario678247@correoprueba.com	2
12	911181	911181	Usuario911181	Usuario911181@correoprueba.com	2
13	261725	261725	Usuario261725	Usuario261725@correoprueba.com	2
14	662756	662756	Usuario662756	Usuario662756@correoprueba.com	2
15	538765	538765	Usuario538765	Usuario538765@correoprueba.com	2
16	31597	31597	Usuario31597	Usuario31597@correoprueba.com	2
17	272287	272287	Usuario272287	Usuario272287@correoprueba.com	2
18	727042	727042	Usuario727042	Usuario727042@correoprueba.com	2
19	678019	678019	Usuario678019	Usuario678019@correoprueba.com	2
20	159770	159770	Usuario159770	Usuario159770@correoprueba.com	2
21	290847	290847	Usuario290847	Usuario290847@correoprueba.com	2
22	111532	111532	Usuario111532	Usuario111532@correoprueba.com	2
23	985956	985956	Usuario985956	Usuario985956@correoprueba.com	2
24	798033	798033	Usuario798033	Usuario798033@correoprueba.com	2
25	419076	419076	Usuario419076	Usuario419076@correoprueba.com	2
26	245565	245565	Usuario245565	Usuario245565@correoprueba.com	2
27	278965	278965	Usuario278965	Usuario278965@correoprueba.com	2
28	966894	966894	Usuario966894	Usuario966894@correoprueba.com	2
29	649240	649240	Usuario649240	Usuario649240@correoprueba.com	2
30	652804	652804	Usuario652804	Usuario652804@correoprueba.com	2
31	877979	877979	Usuario877979	Usuario877979@correoprueba.com	2
32	268506	268506	Usuario268506	Usuario268506@correoprueba.com	2
33	894038	894038	Usuario894038	Usuario894038@correoprueba.com	2
34	377766	377766	Usuario377766	Usuario377766@correoprueba.com	2
35	125796	125796	Usuario125796	Usuario125796@correoprueba.com	2
36	744162	744162	Usuario744162	Usuario744162@correoprueba.com	2
37	479973	479973	Usuario479973	Usuario479973@correoprueba.com	2
38	269470	269470	Usuario269470	Usuario269470@correoprueba.com	2
39	980298	980298	Usuario980298	Usuario980298@correoprueba.com	2
40	260963	260963	Usuario260963	Usuario260963@correoprueba.com	2
41	530601	530601	Usuario530601	Usuario530601@correoprueba.com	2

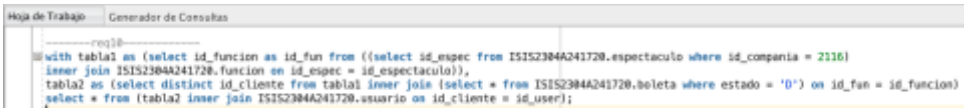
Plan de ejecución:

OPERATION	OBJECT_NAME	CARDINALITY	COST
SELECT STATEMENT		16	234
HASH JOIN		16	234
Access Predicates			
ID_CLIENTE=ID_USER			
NESTED LOOPS		16	234
NESTED LOOPS		16	234
NESTED LOOPS COLLECTION			
VIEW		16	197
HASH (UNIQUE)		16	197
NESTED LOOPS		16	196
NESTED LOOPS		16	196
HASH JOIN		16	58
Access Predicates			
ID_ESPEC=ID_ESPECTACULO			
NESTED LOOPS		16	58
STATISTICS COLLECTOR			
INDEX (FAST FULL SCAN)	INDEX_ESPEC_COMPA	1	10
Filter Predicates			
ID_COMPANIA=2280			
TABLE ACCESS (BY INDEX ROWID BATCHED)	FUNCION	16	48
INDEX (RANGE SCAN)	INDEX_IDESPEC	45	2
Access Predicates			
ID_ESPEC=ID_ESPECTACULO			
INDEX (FAST FULL SCAN)	INDEX_FUNC_ESPEC	16	2
BITMAP CONVERSION (TO ROWIDS)			
BITMAP INDEX (RANGE SCAN)	INDEX_FULL_BO		
Access Predicates			
FUNCION.ID_FUNCION=BOLETA.ID_FUNCION			
Filter Predicates			
FUNCION.ID_FUNCION=BOLETA.ID_FUNCION			
TABLE ACCESS (BY INDEX ROWID)	BOLETA	1	196
Filter Predicates			
ESTADO='A'			
INDEX (UNIQUE SCAN)	ID_USER_PK	1	1
Access Predicates			
ID_CLIENTE=ID_USER			
TABLE ACCESS (BY INDEX ROWID)	USUARIO	1	2
TABLE ACCESS (FULL)	USUARIO	1	2

De acuerdo al plan de ejecución realizado por Oracle se tiene que: en primera medida si utilizó el índice index_espec_compa para filtrar por el predicado 2280. Luego de esto, utiliza el índice creado llamado index_idEspec el cual es de tipo normal. Línea despues utiliza un índice sobre la tabla boleta el cual es un índice compuesto donde se resuelve una parte del nested loop. Por úlitmo no utiliza el índice del predicado bitmap sobre la columna estado para realizar la respectiva comparación.

2. RF10:
   a. Caso de prueba 1:

      Se realizará el requerimiento funcional de consulta número 10 con el parámetro de consulta tal que el id de la compañía es el 2116. Se espera como resultado al menos una tupla con la información requerida por el requerimiento.



      Para la resolución de esta sentencia se espera lo mismo que el anterior requerimiento funcional con la diferencia de que cambia el valor del estado en la sentencia: Debido a que la sentencia tiene varias vistas de tablas (views) involucradas, se resolverán primero los select * más internos de la sentencia donde estos serían select * from espectaculo where id_compania = 2280, para esta sentencia se espera que se haga el filtro de predicado usando el índice index_espec_compa. Luego se espera que resuelva un inner join con la tabla función bajo los valores de id_espectaculo donde se pretende que use el mismo índice anteriormente descrito. Después, se espera que para realizar la sentencia que contiene la igualdad estado='D' use el índice bitmap creado sobre la columna estado para resolver la sentencia mucho más eficaz. Con el resultado de la anterior sentencia, puede ocurrir el inner join con la tabla cliente bajo el id_cliente como relación de las dos tablas. Valga aclarar que se utiliza una columna Estado para identificar si la boleta aún está activa o fue devuelta antes que se realizará la función.

Resultado: 1 tupla en 0,089 segundos.

	ID_CLIENTE	ID_USER	NOMBRE_USER	CORREO_USER	ROL_USER
1	86826	86826	Usuario86826	Usuario86826@correoprueba.com	2

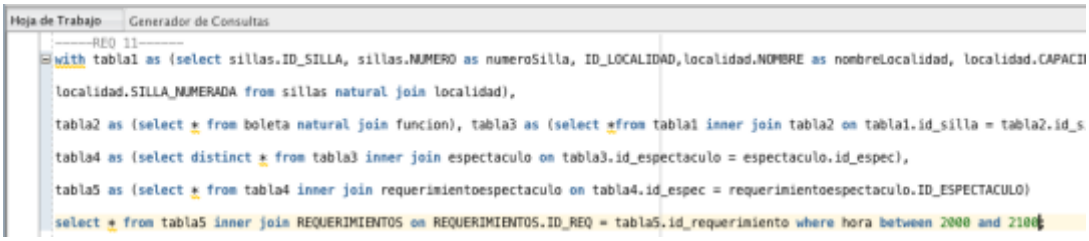
Plan de ejecución:

      De acuerdo al plan de ejecución realizado por Oracle se tiene que: en primera medida si utilizó el índice index_espec_compa para filtrar por el predicado 2116. Luego de esto, utiliza el índice creado llamado index_idEspec el cual es de tipo normal. Líneas después utiliza un índice sobre la tabla boleta el cual es un índice compuesto donde se resuelve una parte del nested loop. Por úlitmo y a diferencia del anterior requerimiento,

se utiliza el índice bitmap sobre la columna estado para encontrar todas aquellas tuplas en la cual el estado es "D".

3. RF11
   a. Caso de prueba 1.

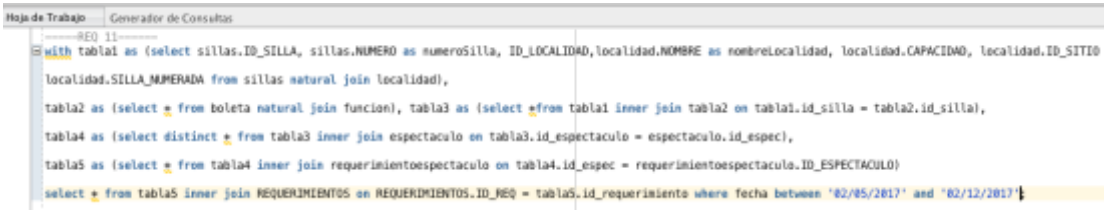
Se realizará el requerimiento número 11 haciendo un filtrado de datos por la franja horaria. Estos datos serán entre las 20 horas y las 24 horas. Se espera como resultado al menos una tupla con la información solicitada en el requerimiento.



El plan de ejecución esperado para el requerimiento funcional 11 al igual que los otros requerimientos, se inicia desarrollando las sentencias internas haciendo especial énfasis en los natural e inner joins donde se espera que utilice el índice compuesto creado para la tabla requerimientoEspectaculo. De igual forma, se espera que al momento de ejecutar la sentencia where para la columna de franja horaria, se utilice el índice compuesto creado especialmente para ese requerimiento, tal índice está formado por la hora y el id del espectáculo donde se le da prevalencia al campo de hora.

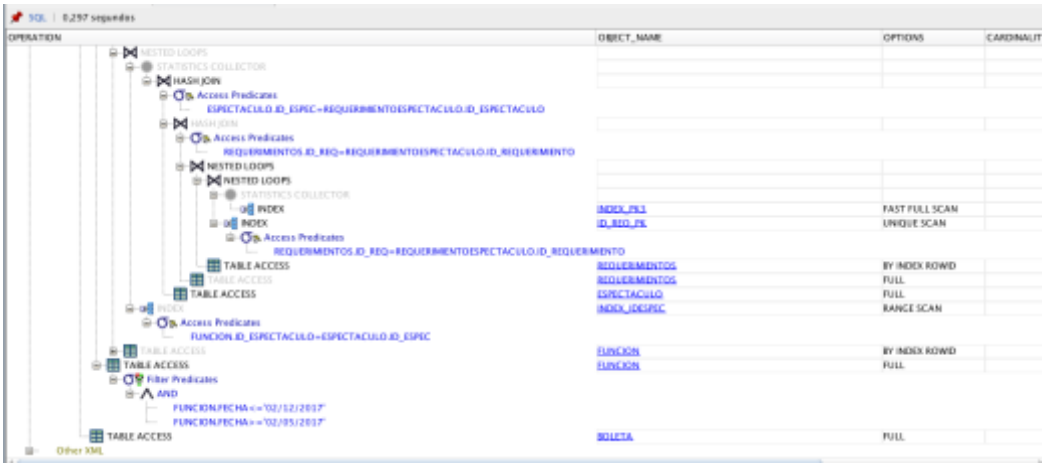
   b. Caso de prueba 2:

Se realizará el requerimiento número 11 haciendo un filtrado de datos por la Fecha. Estos datos serán entre la fecha 02/05/2017 y las 02/12/2017. Se espera como resultado al menos una tupla con la información solicitada en el requerimiento.



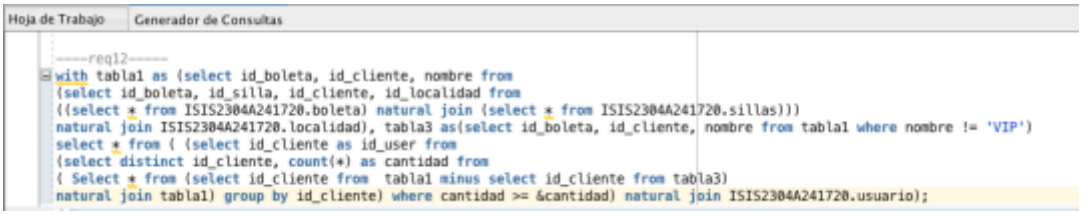
Para el plan de ejecución al igual que el caso de prueba 1, se espera que se inicia desarrollando las sentencias internas haciendo especial énfasis en los natural e inner joins donde se espera que utilice el índice compuesto creado para la tabla requerimientoEspectaculo. Al momento de ejecutar la sentencia del where con datos de

fechas, se espera que primero se organice la tabla mediante esta columna y luego realice la búsqueda dentro del rango establecido.

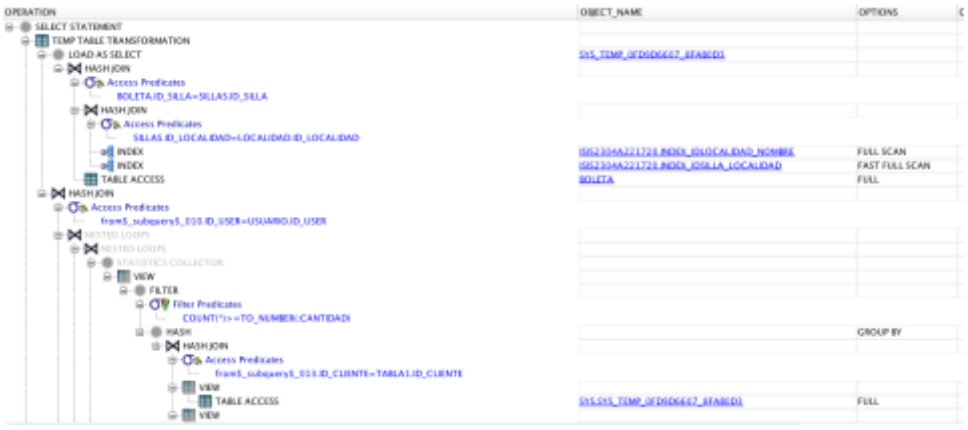


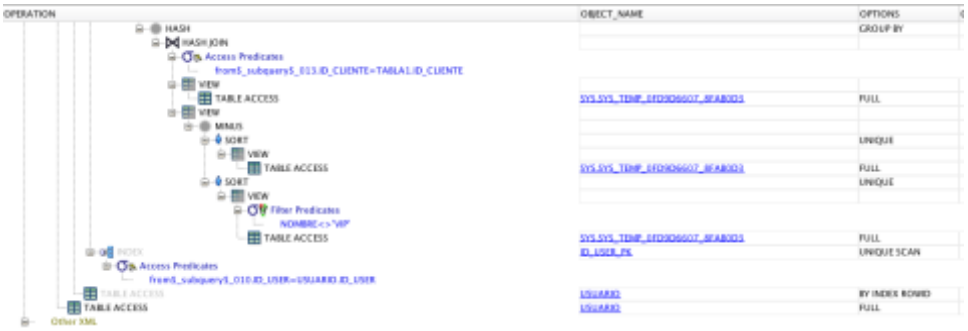
4. RF12
   a. Caso de prueba 1:

Se realizará el requerimiento número 12 de acuerdo al enunciado, tomando las boletas de los clientes las cuales no son VIP y luego "restando" este resultado de la tabla total de boletas con el fin de dejar únicamente los clientes que si compraron boleta en localidad "VIP".



El de ejecución esperado para la sentencia es: primero evaluar las sentencias donde sea necesario realizar hash join o nested loops mediante llaves primarias con índices previamente creados. Luego se espera que se evalué mediante los índices creados en la tabla localidad con las columnas id localidad y nombre con el fin de efectuar de una manera más eficaz las sentencias join mediante el id y los condicionales where con el nombre de la localidad. De igual forma se espera la ejecución de la sentencia usando el índice creado sobre la tabla silla para las columnas id silla e id localidad con el de fin de que cumplan la misma función descrita anteriormente.





POBLAMIENTO BD:

En un principio, tocó generar los datos de manera que en las tablas hubiera información coherente. Por ello, creamos un pequeño programa generador de datos. En este nosotros podíamos decidir la cantidad de datos que generaban, y en qué formato quedaba. Decidimos escribir dentro de archivos .csv debido a que nos pareció la mejor opción.

Se crearon alrededor de un millón de datos para las tablas: Boleta, Boleta_silla, Boleta_detalle y Usuarios. Las demás tablas se les generaron datos de tal manera que el manejo de la coherencia de los datos fuera fácil y rápido.

Teniendo los datos ya creados, nos dirigimos a la plataforma SQL developer donde se encuentran nuestras tablas del proyecto. Para cada tabla a la que se le generaron datos, se produjo el proceso de acceder a la opción de importación de datos de la tabla. En esta seleccionamos la ruta en donde se encuentra el archivo .csv que generó el programa anteriormente. Luego de seguir una serie de pasos dados por el Wizard de SQLDeveloper. La importanción de alrededor un millón de datos por tabla fue efectiva.

Análisis resultados:

Si trajéramos todos los datos a memoria principal realizar todas las operaciones de filtro y joins sería mucho menos eficiente y el programa ocuparía mucha más memoria. Primero deberíamos traer toda la información de las tablas y guardarlos objetos y estos en estructuras como arraylist o hash tables donde usaríamos los id como key para estas tablas. Cuando ya hayamos traído toda la información podríamos comenzar a filtrar por los criterios que necesitáramos pero tendríamos que usar estructuras temporales para guardar esta información lo cual ocuparía mucha más memoria. Además de que no podríamos filtrar y unir las tablas tan fácilmente porque primero necesitaríamos filtrar una, luego la otra y luego intentar unirlas y guardarlas en nuevos objetos que nos permitan guardar la información que estamos uniendo. Esto requeriría mucho as espacio y tiempo de ejecución cuando estamos hablando de una cantidad de datos considerable.