Iteración No. 4 Sistemas Transaccionales

Juan David Serrano 201632249
Maria Alejandra Escalante 201631008

Febrero 2019

Índice

1. Modelos y Analisis **2**
1.1. Modelo Conceptual . . . 2
1.2. Modelo de Datos . . . 3
1.3. Ajustes a modelos actual . . . 5

2. Diseño aplicación **6**
2.1. Diseño Físico . . . 6
2.1.1. Documentación diseño físico . . . 6
2.1.2. Documentación RFC . . . 6

3. Contracción aplicación, ejecución pruebas y análisis de resultados **14**
3.1. Documentación de carga de datos . . . 14
3.2. Ajuste clases, clases SQL y transacciones (Persistencia) . . . 15

4. Proceso optimización **15**

5. Resultados Logrados **15**

6. Resultados no logrados **15**

7. Balance del plan de pruebas **15**

1. Modelos y Analisis

1.1. Modelo Conceptual

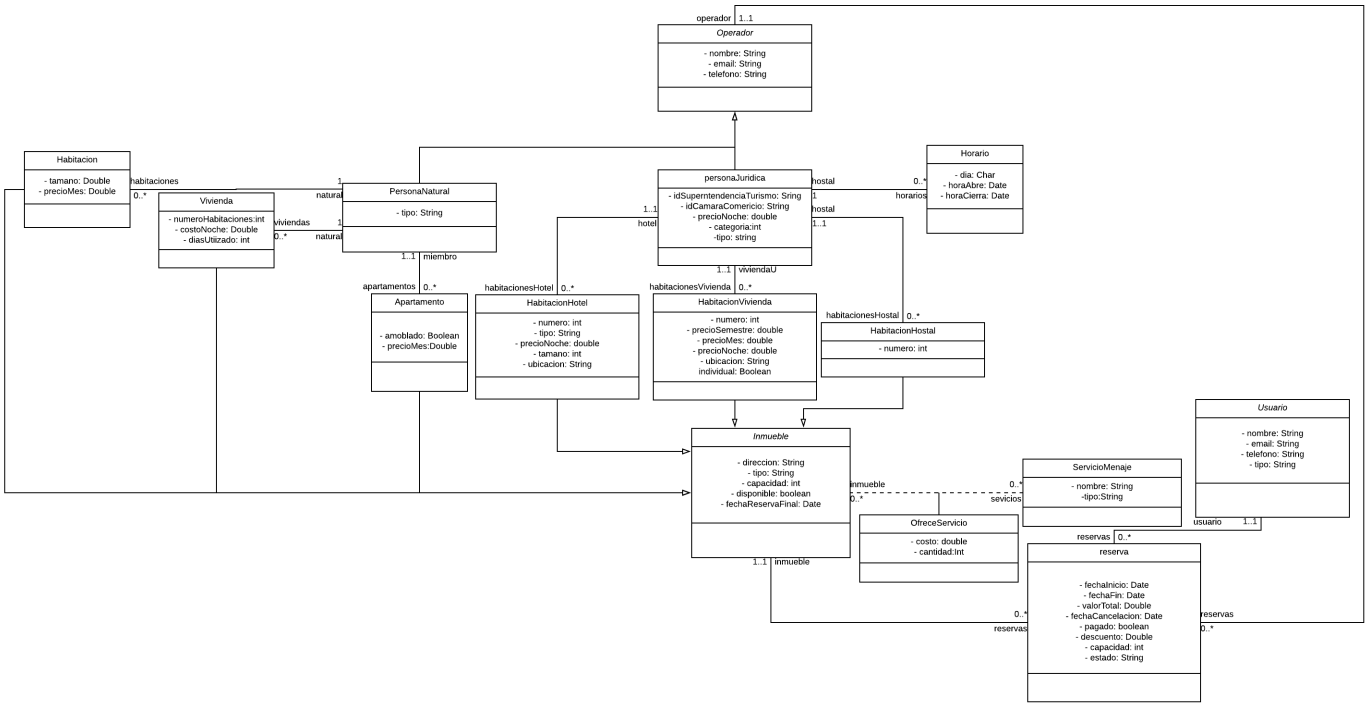


Figura 1: Modelo Conceptual

Supuestos adicionales sobre las reglas de negocio encontradas en el caso de estudio Por aplicación se revisa que se cumplan las siguientes cosas

1. Un alojamiento no acepta reservas que superen su capacidad
2. Solo se pueden arrendar inmuebles disponibles
3. Que un invitado solo pueda arrendar vivienda
4. Habitación reserva mínima es de un mes
5. En apto el contrato mínimo es de un mes
6. En vivienda verificar que los días utilizados no superen un mes
7. una persona no puede reservar mas de un alojamiento en un mismo día
8. el alojamiento de vivienda universitaria solo esta habilitado a estudiantes, profesores, empleados y profesor visitante
9. Para usos cobrados por semanas o meses, el tiempo limite está fijado en una semana.
10. Horario únicamente de hostal
11. calcular valor de reserva

12. revisar que todo menaje tiene cantidad y todo servicio tiene un costo asociado
13. precio noche es de hostal únicamente
14. categoría es de hotel únicamente

1.2. Modelo de Datos

La única modificación al esquema de la base de datos fue incorporar un atributo adicional a la tabla reserva para que fuera posible indicar si esta pertenecía o no a una reserva colectiva. Adicionalmente en el esquema en SQL se modificaron las secuencia para que existieran 4: para inmueble, para usuario, para operador, para reserva y para reserva colectiva.

operador			
id	nombre	email	telefono
SA PK	NN	NN ND	NN
NUMBER	VARCHAR2(40)	VARCHAR2(40)	VARCHAR2(15)

usuario				
id	nombre	email	telefono	tipo
SA PK	NN	NN ND	NN	NN $CK_{tipo\_usuario}$
NUMBER	VARCHAR2(40)	VARCHAR2(40)	VARCHAR2(15)	VARCHAR2(40)

$tipo\_usuario = [profesor, profesor\_invitado, estudiante, egresado, empleado, padre\_estudiante, invitado]$

persona_juridica					
id	id_superintendencia_turismo	id_camara_comercio	categoria	precio_noche	tipo
$FK_{operador.id}$ PK	NN ND	NN ND	NULL	NULL $CK_{\mathbb{R}\geq 0}$	NN $CK_{[hotal,hostal,vivienda]}$
NUMBER	NUMBER	NUMBER	NUMBER(1)	NUMBER	VARCHAR2(9)

horario			
id_hostal	dia	hora_abre	hora_cierre
PK $FK_{persiona\_juridica.id}$	NN PK	NN	NN
NUMBER	VARCHAR2(1)	DATE	DATE

habitacion_hotel						
id	id_hotel	numero	tipo	precio_noche	tamanho	ubicacion
$FK_{inmueble.id}$ PK	$FK_{persona\_juridica.id}$ NN	NN	NN CK_{tipo}	NN $CK_{\mathbb{R}\geq 0}$	NN $CK_{\mathbb{R}\geq 0}$	NN
NUMBER	NUMBER	NUMBER(5)	VARCHAR2(9)	NUMBER(10)	number(5)	VARCHAR2(40)

$tipo = [estandar, semisuite, suite]$

habitacion_vivienda							
id	id_vivienda	numero	precio_semestre	precio_mes	precio_noche	ubicacion	individual
$FK_{inmueble.id}$ PK	$FK_{persona\_juridica.id}$ NN	NN $CK_{\mathbb{R}\geq 0}$	NN $CK_{\mathbb{R}\geq 0}$	NN $CK_{\mathbb{R}\geq 0}$	NN	NN $CK_{\mathbb{R}\geq 0}$	NN $CK_{[Y,N]}$
NUMBER	NUMBER	NUMBER (5)	NUMBER (60)	NUMBER (30)	NUMBER (10)	VARCHAR2 (40)	VARCHAR (1)

habitacion_hostal		
id	id_hostal	numero
$FK_{inmueble.id}$ PK	$FK_{persona\_juridica.id}$ NN	NN
NUMBER	NUMBER	NUMBER(5)

inmueble					
id	direccion	tipo	capacidad	disponible	fecha_reserva_final
PK SA	NN	NN $CK_{tipo_inmueble}$	NN	NN	NULL
NUMBER	VARCHAR2(40)	VARCHAR(12)	NUMBER(2)	VARCHAR (1)	DATE

$tipo_inmueble = [vivienda, habitacion, apartamento, habitacion\_hotel, habitacion\_hostal, habitacion\_vivienda]$

servicio_menaje	
nombre	tipo
PK UA	NN $CK_{[menaje,servicio]}$
VARCHAR2(40)	VARCHAR2(8)

ofrece_servicio			
id_servicio_menaje	id_inmueble	costo	cantidad
PK $FK_{servicio\_menaje.nombre}$	PK $FK_{inmueble.id}$	NULL $CK_{\mathbb{R}\geq 0}$	NULL $CK_{\mathbb{N}\geq 0}$
VARCHAR2(40)	NUMBER	NUMBER(9,2)	NUMBER(2)

reserva					
id	fecha_inicio	fecha_fin	valor_total	fecha_cancelacion	pagado
SA PK	NN	NN	NN	NN SA	NN
NUMBER	DATE	DATE	NUMBER	DATE	NUMBER

Continuando la anterior tabla

reserva						
descuento	capacidad	estado	id_operador	id_usuario	id_inmueble	idColectiva
NN	NN	NN $CK_{0,1}$	$FK_{operador.id}$ NN	FK $_{usuario.id}$ NN	$FK_{inmueble.id}$ NN	NULL
NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER

persona_natural	
id	tipo
PK $FK_{entidad.id}$	NN
NUMBER	VARCHAR2(40)

habitacion			
id	tamano	precioMes	id_persona
PK $FK_{inmueble.id}$	NN	NN	$FK_{persona\_natural.id}$
NUMBER	NUMBER	NUMBER	NUMBER

vivienda				
id	numero_habitaciones	costo_noche	dias_utilizado	id_persona
PK $FK_{inmueble.id}$	NN	NN	NN	$FK_{persona\_natural.id}$
NUMBER	NUMBER	NUMBER	NUMBER	NUMBER

apartamento			
id	Amoblado	precioMes	id_persona
PK $FK_{inmueble.id}$	NN $CK_{0,1}$	NN	$FK_{persona\_natural.id}$
NUMBER	NUMBER	NUMBER	NUMBER

1.3. Ajustes a modelos actual

Lo que concierne al diseño estructural de los modelos existentes no se les realizo ningún cambio adicional para cumplir con los Requerimientos Funcionales de Consulta (RFC) o lo Requerimientos No funcionales (RNF) de esta iteración 4.

2. Diseño aplicación

2.1. Diseño Físico

2.1.1. Documentación diseño físico

	INDEX_NAME
1	UNUSUARIOEMAIL
2	UNPJIDSUPER
3	UNPJIDCOMERCIO
4	UNOPERADOREMAIL
5	PKVIVIENDA
6	PKUSUARIO
7	PKSERVICIOMENAJE
8	PKRESERVA
9	PKPERSONANATURAL
10	PKPERSONAJURIDICA
11	PKOPERADOR
12	PKOFRECESERVICIO

Figura 2: Indices

13	PKINMUEBLE
14	PKHORARIO
15	PKHABITACIONVIVIENDA
16	PKHABITACIONHOTEL
17	PKHABITACIONHOSTAL
18	PKHABITACION
19	PKAPTO

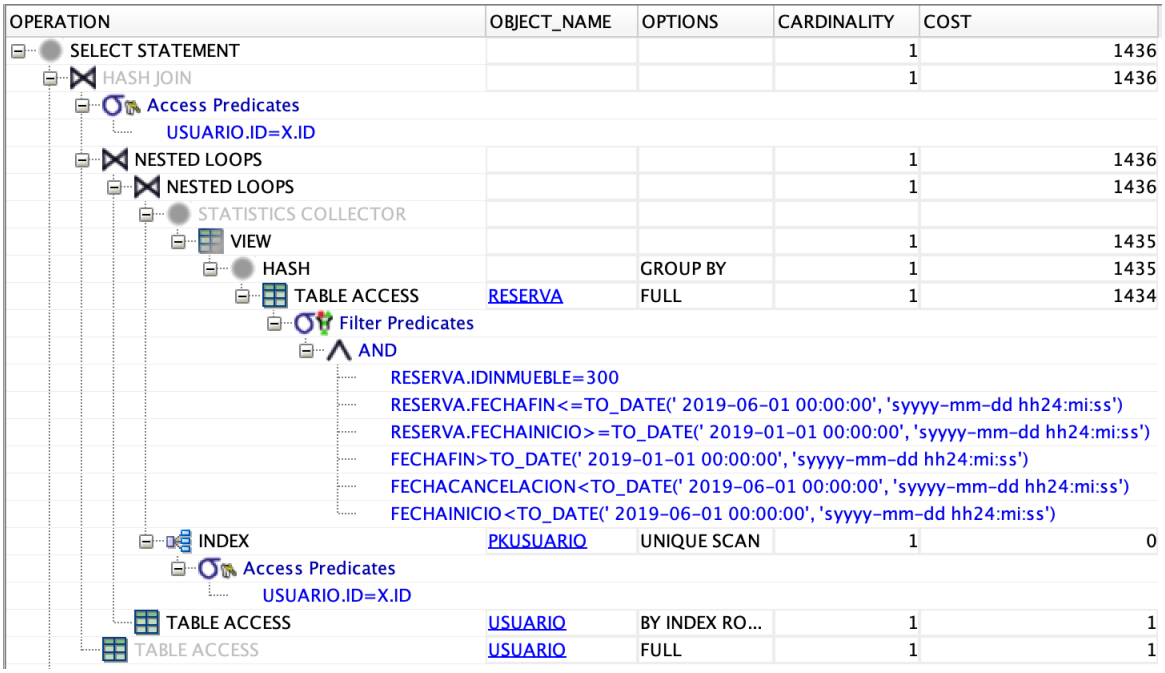
Figura 3: Indices

2.1.2. Documentación RFC

1. RFC10

   - Sentencias SQL

```
SELECT USUARIO.*
FROM
(SELECT USUARIO.ID
FROM USUARIO
INNER JOIN RESERVA ON USUARIO.ID = RESERVA.IDUSUARIO
WHERE IDINMUEBLE = P1
AND FECHAINICIO BETWEEN  P2 AND P3
AND FECHAFIN BETWEEN P2 AND P3
```



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	1436
HASH JOIN			1	1436
Access Predicates				
USUARIO.ID=X.ID				
NESTED LOOPS			1	1436
NESTED LOOPS			1	1436
STATISTICS COLLECTOR				
VIEW			1	1435
HASH		GROUP BY	1	1435
TABLE ACCESS	RESERVA	FULL	1	1434
Filter Predicates				
AND				
RESERVA.IDINMUEBLE=300				
RESERVA.FECHAFIN<=TO_DATE(' 2019-06-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
RESERVA.FECHAINICIO>=TO_DATE(' 2019-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
FECHAFIN>TO_DATE(' 2019-01-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
FECHACANCELACION<TO_DATE(' 2019-06-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
FECHAINICIO<TO_DATE(' 2019-06-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')				
INDEX	PKUSUARIO	UNIQUE SCAN	1	0
Access Predicates				
USUARIO.ID=X.ID				
TABLE ACCESS	USUARIO	BY INDEX RO...	1	1
TABLE ACCESS	USUARIO	FULL	1	1

Figura 4: Plan de consulta RFC10

```
GROUP BY USUARIO.ID) X
INNER JOIN USUARIO ON USUARIO.ID = X.ID;
```

- Distribución datos
  Los reservas se generaron entre el 01/01/2019 y el 31/01/2030, teniendo en cuenta que se poblaron cerca de 542,000 reservas y que se poblaron 100,000 inmuebles. A partir de esto y tomando los rangos de fechas como días la probabilidad de que una reserva pertenezca a ese rango dado un inmueble sigue la siguiente ecuación:
  $P_{RFC10} = \frac{1}{100{,}000} * \frac{\Delta dias}{4046}$
  Aunque el rango fuera muy grande, dado que las funciones de selección aleatoria usadas en la generación de datos tienen una distribución uniforme como máximo la consulta solo podría arrojar $1/100,000$ de las reservas totales.
- Valores parámetros
  Los parámetros de mayor importancia para esta consulta son el identificador del inmueble/oferta de alojamiento que se quiere consultar, la fecha de inicio del rango de fecha y la fecha de fin del rango. Sin embargo, dependiendo del rol de quien ejecuta el RFC también es necesario el identificador del usuario. Adicionalmente, los criterios de ordenamiento que podrían ser usados para entregar los resultados son el identificador del usuario, el nombre del usuario y el email.
- Planes de consulta obtenidos Oracle
  El plan de consulta generado de forma automática por Oracle es el que se muestra a continuación.
- Tiempos de ejecución de planes

- Escenarios de prueba
  Como escenario de prueba se eligió eligió el caso en que el inmueble era el 300 y el rango de fechas se encontraba entre el 01/01/2019 y el 01/01/2030. Para este caso de prueba se generaron 4 usuarios.

2. RFC11

- Sentencias SQL

```
SELECT *
FROM USUARIO
WHERE USUARIO.ID NOT IN (
SELECT USUARIO.ID
FROM USUARIO
INNER JOIN RESERVA ON USUARIO.ID = RESERVA.IDUSUARIO
WHERE IDINMUEBLE = 1? AND FECHAINICIO BETWEEN 2? AND 3? AND FECHAFIN BETWEEN 2? AND 3?
GROUP BY USUARIO.ID);
```

- Distribución datos
  Como esta operación es la negada de la anterior la probabilidad del evento sera 1/PRFC10 y la cantidad de datos sera muy grande (prácticamente todos los usuarios) ya que es mas probable que un usuario no tenga una reserva sobre un inmueble especifico. $P_{RFC10} = \frac{1}{100{,}000} * \frac{\Delta dias}{4046}$
  Aunque el rango fuera muy grande, dado que las funciones de selección aleatoria usadas en la generación de datos tienen una distribución uniforme como máximo la consulta solo podría arrojar $1/100,000$ de las reservas totales.
- Valores parámetros
  Los parámetros son el id del inmueble/oferta de alojamiento el cual va a ser consultado y la fecha de inicio y fin entre las cuales se quiere saber los usuarios que no tienen reservas en dicho inmueble. Esto es si lo consulta un administrador. Dado elc aso que lo consulte un su usuario este deberá proporcionar su id para limitar la consulta por ese parámetro también
- Planes de consulta obtenidos Oracle

Figura 5: Plan de Consulta en oracle RFC11

- Tiempos de ejecución de planes
  El tiempo de ejecución del plan de Oracle fue de 0.235 segundos para el escenario de prueba que se muestra a continuacion.
- Escenarios de prueba
  Se realizo una prueba con el inmueble 300 y el rango de fecha entre el primero de enero de 2019 y el primero de diciembre de 2030. resultaron 99995 usuarios. los primeros son

1	2679	Antonia	Antonia2678@hotmail.com	8468291264	Profesor Invitado
2	2680	Valeria	Valeria2679@hotmail.com	8468291264	Invitado
3	2681	David	David2680@hotmail.com	8468291264	Profesor Invitado
4	2682	Guillermo	Guillermo2681@hotmail.com	8468291264	Invitado
5	2683	Luis Eduardo	Luis Eduardo2682@hotmail.com	8468291264	Egresado
6	2684	Ricardo	Ricardo2683@hotmail.com	8468291264	Profesor Invitado
7	2685	Daniel	Daniel2684@hotmail.com	8468291264	Profesor
8	2686	Alberto	Alberto2685@hotmail.com	8468291264	Profesor Invitado
9	2687	Fernando	Fernando2686@hotmail.com	8468291264	Invitado
10	2688	Tania	Tania2687@hotmail.com	8468291264	Egresado

Figura 6: Resultado RFC11

3. RFC12
   El RFC12 se compone de 4 consultas SQL. Debido a que los datos que se solicitaban no siempre eran de la misma índole entonces descompuso en 4 sentencias.

   - Sentencias SQL
     La primera de las consultas da la información da inmueble con mayor tasa de ocupación por semana. Es importante resaltar que las semanas se identificaron por la fecha de inicio (domingo) de cada una de ellas.

```
SELECT INMUEBLE.*, SEMANA
FROM
(SELECT INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY') AS SEMANA,
AVG(RESERVA.CAPACIDAD/INMUEBLE.CAPACIDAD) AS TASAOCUPACION
FROM INMUEBLE
INNER JOIN RESERVA ON RESERVA.IDINMUEBLE = INMUEBLE.ID
GROUP BY INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')) X
INNER JOIN INMUEBLE ON INMUEBLE.ID =X.ID
WHERE (SEMANA, TASAOCUPACION) IN (SELECT SEMANA, MAX(TASAOCUPACION)
FROM
(SELECT INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY') AS SEMANA,
AVG(RESERVA.CAPACIDAD/INMUEBLE.CAPACIDAD) AS TASAOCUPACION
FROM INMUEBLE
INNER JOIN RESERVA ON RESERVA.IDINMUEBLE = INMUEBLE.ID
GROUP BY INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY'))
GROUP BY SEMANA);
```

     Le segunda consulta es identifica a la primera solo que en vez de dar los inmuebles con mayor tasa de ocupación por semana da los de menor tasa de ocupación.

```
SELECT INMUEBLE.*, SEMANA
```

```
FROM (SELECT INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY') AS SEMANA,
AVG(RESERVA.CAPACIDAD/INMUEBLE.CAPACIDAD) AS TASAOCUPACION
FROM INMUEBLE
INNER JOIN RESERVA ON RESERVA.IDINMUEBLE = INMUEBLE.ID
GROUP BY INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')) X
INNER JOIN INMUEBLE ON X.ID = INMUEBLE.ID
WHERE (SEMANA, TASAOCUPACION) IN (SELECT SEMANA, MIN(TASAOCUPACION)
FROM
(SELECT INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY') AS SEMANA,
AVG(RESERVA.CAPACIDAD/INMUEBLE.CAPACIDAD) AS TASAOCUPACION
FROM INMUEBLE
INNER JOIN RESERVA ON RESERVA.IDINMUEBLE = INMUEBLE.ID
GROUP BY INMUEBLE.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY'))
GROUP BY SEMANA);
```

La tercera consulta genera los operadores mas solicitados por semana (se tomo como mas solicitas como con mayor numero de reservas a inmuebles de su propiedad en esa semana)

```
SELECT OPERADOR.*, SEMANA
FROM (SELECT OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')AS SEMANA, COUNT(*) AS C
FROM OPERADOR
INNER JOIN RESERVA ON OPERADOR.ID = RESERVA.IDOPERADOR
GROUP BY OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')) X
INNER JOIN OPERADOR ON OPERADOR.ID = X.ID
WHERE (SEMANA, C) IN (SELECT SEMANA, MAX(C)
FROM
(SELECT OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')AS SEMANA, COUNT(*) AS C
FROM OPERADOR
INNER JOIN RESERVA ON OPERADOR.ID = RESERVA.IDOPERADOR
GROUP BY OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY'))
GROUP BY SEMANA);
```

La cuarta consulta es identica a la tercera solo que en vez de entregar los operadores mas solicitad muestra los menos solicitados.

```
SELECT OPERADOR.*, SEMANA
FROM (SELECT OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')AS SEMANA, COUNT(*) AS C
FROM OPERADOR
INNER JOIN RESERVA ON OPERADOR.ID = RESERVA.IDOPERADOR
GROUP BY OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')) X
INNER JOIN OPERADOR ON OPERADOR.ID = X.ID
WHERE (SEMANA, C) IN (SELECT SEMANA, MIN(C)
FROM
(SELECT OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY')AS SEMANA, COUNT(*) AS C
FROM OPERADOR
INNER JOIN RESERVA ON OPERADOR.ID = RESERVA.IDOPERADOR
GROUP BY OPERADOR.ID, TRUNC(RESERVA.FECHAINICIO, 'DAY'))
GROUP BY SEMANA);
```

- Distribución datos
  La distribución de los datos depende de la consulta. Para las primeras dos consultas se quiere saber cuantas resevra por usuarios hay en una semana

- Valores parámetros
  Esta consulta no posee parámetros ya que entrega tuplas fijas para todo el tiempo de operación de AlohAndes.
- Planes de consulta obtenidos Oracle
- Tiempos de ejecución de planes

  a) .36 segundos
  b)
  c)
  d)
- Escenarios de prueba
  El escenario de prueba es la ejecucion misma del metodo porque no solicita parámetros para su ejecución

4. RFC13

- Sentencias SQL

```
select usuario.id
from usuario
where usuario.id in (

select usuario.id
from usuario
inner join reserva on reserva.idusuario=usuario.id
where reserva.valortotal>=19500

) or usuario.id in (
select usuario.id
from usuario
inner join reserva on reserva.idusuario=usuario.id
inner join inmueble on reserva.idinmueble=inmueble.id
inner join habitacionhotel on habitacionhotel.id=inmueble.id
where inmueble.tipo='Habitacion Hotel' and habitacionhotel.tipo='Suite'
) or usuario.id in (
SELECT id
FROM
(SELECT ID, COUNT(*) AS C
FROM
(SELECT USUARIO.ID, TRUNC(RESERVA.FECHAINICIO,'MONTH')
FROM USUARIO
INNER JOIN RESERVA ON USUARIO.ID = RESERVA.IDUSUARIO
INNER JOIN INMUEBLE ON RESERVA.IDINMUEBLE = INMUEBLE.ID
GROUP BY USUARIO.ID, TRUNC(RESERVA.FECHAINICIO,'MONTH'))
GROUP BY ID)
)
group by usuario.id;
```

- Distribución datos
  La distribución de datos para esta sentencia depende de la cantidad de habitaciones hotel y la cantidad de ellas que son suites que podríamos asumir son al rededor de 5500 y de la cantidad de reservas sobre ellas. Por el otro requisito también depende del costo de las reservas y la cantidad de las mismas.
- Valores parámetros
  Esta consulta no tiene parámetros
- Planes de consulta obtenidos Oracle

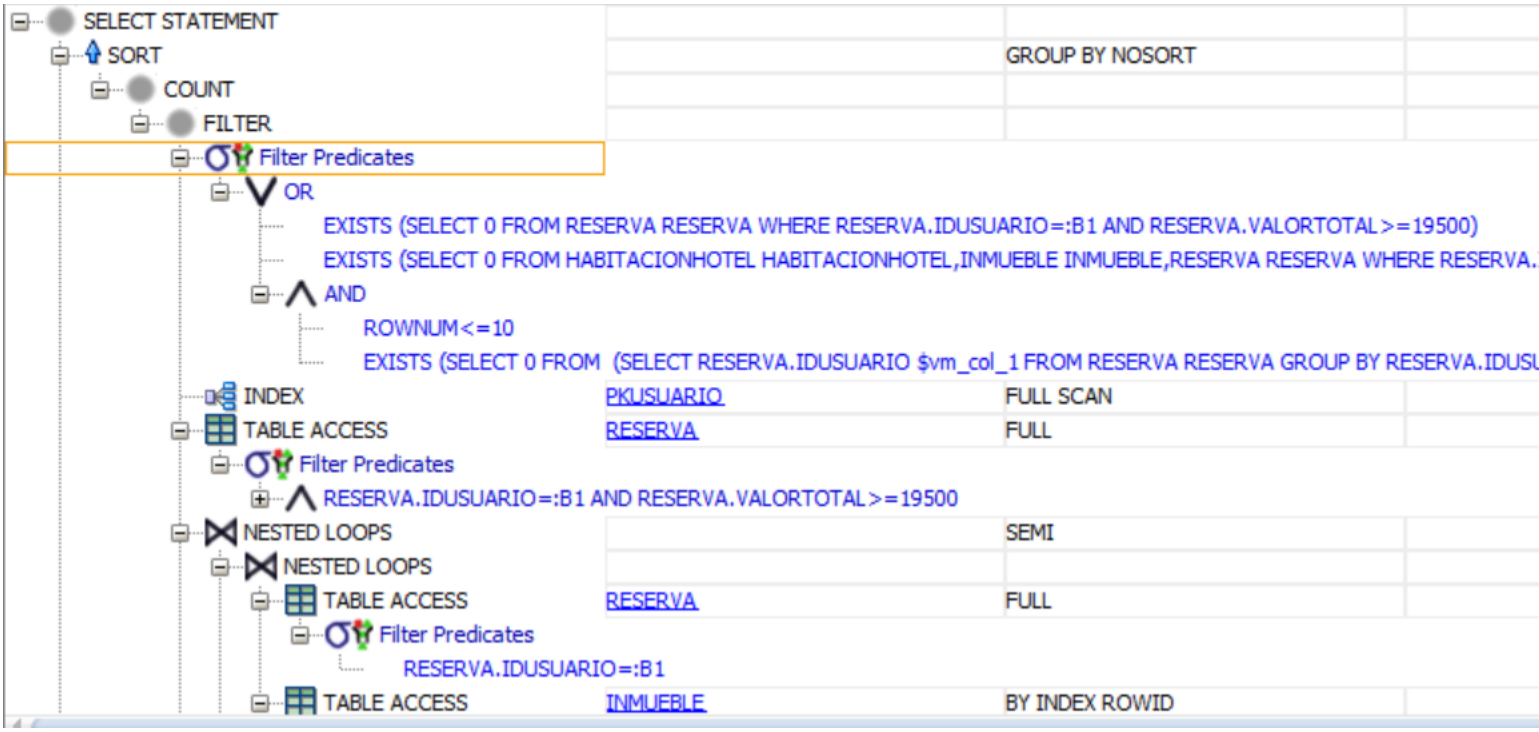


Figura 7: Plan oracle parte 1

Figura 8: Plan oracle parte 2

Figura 9: Plan oracle parte 3

- Tiempos de ejecución de planes
  el tiempo de ejecucion del plan es 0.36 segundos
- Escenarios de prueba
  Al intentar correr la consulta en sql se obtuvieron los siguientes resultados

	ID
1	21316
2	50457
3	36409
4	65061
5	61895
6	73015
7	33094
8	61351
9	60167
10	19704
11	26921
12	17602
13	44889
14	2886
15	9926
16	83589
17	36282

Figura 10: resultados consulta 13

3. Contracción aplicación, ejecución pruebas y análisis de resultados

3.1. Documentación de carga de datos

Los datos para la carga se crearon usando un script de python. En dicho archivo se aseguraba que se cumplieran las reglas de negocio para la creación y para cada tabla se creaba un archivo csv. La cantidad de datos elegida fue la siguiente

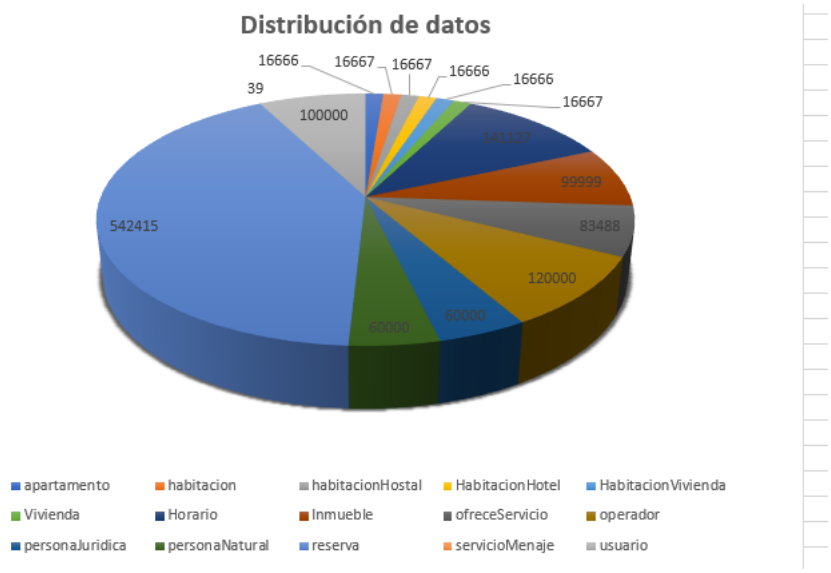


Figura 11: Distribución

Se eligieron estos valores para asegurar una distribución equitativa entre la cantidad de usuarios y de inmuebles y para que la clase reserva tuviese suficientes datos como para probar la eficiencia de las consultas. La carga de los archivo csv se hizo usando sqlloader .

3.2. Ajuste clases, clases SQL y transacciones (Persistencia)

En la interfaz se agregaron en las consultas 10 y 11 la opción de entrar como usuario o administrador. También, en las mismas consultas se habilito la opción de ordenar por los parámetros que el usuario seleccionara. Se agregaron las nuevas consultas pero no se hicieron ajustes en la estructura del proyecto.

4. Proceso optimización

Se intento crear indices sobre ciertas columnas y crear indices de join sin exito. No se realizo la optimización de las consultas

5. Resultados Logrados

En el desarrollo de la interacción 4 se logro:

1. Poblar la base de datos
2. Realización de consultas

6. Resultados no logrados

No se logro hacer la optimización de las consultas

7. Balance del plan de pruebas

Se realizaron las pruebas de todos los requerimientos en java