**Iteración 4**

Camilo Otálora, María Gabriela Torres

Universidad de los Andes, Bogotá, Colombia

{c.otalora, [mg.torres}@uniandes.edu.co](mailto:mg.torres%7d@uniandes.edu.co)

Tabla de contenido

[Elementos fundamentales que hacen parte de Alohandes 2](#_Toc41402255)

[Roles de usuario: 2](#_Toc41402256)

[Entidades de negocio 2](#_Toc41402257)

[Funcionalidades principales 3](#_Toc41402258)

[Reglas de negocio 3](#_Toc41402259)

[Diseño y construcción de la base de datos 4](#_Toc41402260)

[UML 4](#_Toc41402261)

[Modelo relacional 4](#_Toc41402262)

[Índices 7](#_Toc41402263)

[Implementación de los requerimientos funcionales 8](#_Toc41402264)

[Sentencias 8](#_Toc41402265)

[Resultados Logrados 10](#_Toc41402266)

[Construcción de la aplicación, ejecución de pruebas y análisis de resultados 11](#_Toc41402267)

[Proceso para la carga masiva de datos 11](#_Toc41402268)

[Escenario de pruebas 12](#_Toc41402269)

[Análisis de eficiencia 20](#_Toc41402270)

# Elementos fundamentales que hacen parte de Alohandes

## Roles de usuario:

En Alohandes existen dos tipos de usuarios, los operadores y los clientes. Los primeros ofrecen un servicio de alojamiento de acuerdo con su tipo, (empresa, persona natural, miembro de la comunidad, vecino, o empresa de vivienda universitaria) y los clientes realizan reservas para utilizar las ofertas de alojamiento dadas por los operadores.

## Entidades de negocio

Las entidades de negocio del sistema son las siguientes:

Cliente: Persona natural que realiza reservas y utiliza las ofertas de alojamiento de un operador.

Oferta: Propuesta que realiza un operador en la que pone a disposición un bien inmueble para ser ocupado por un cliente en unas fechas establecidas.

Reserva: Acción realizada por un cliente para poder comprar o arrendar un inmueble.

Operador: Es el actor encargado de ofrecer un servicio de alojamiento. Los diferentes tipos de operadores se pueden encontrar a continuación.

Hotel: Empresa constituida para brinda servicios de hotelería. Tiene recepción 24 horas y diferentes tipos de habitaciones.

Hostal: Empresa constituida para brindar servicios de hotelería, que tiene un horario de apertura y cierre y ofrecen sus servicios en habitaciones compartidas.

Persona natural: Persona que cumple con sus obligaciones a título personal, es decir no representa legalmente a una empresa o una institución, se representa a sí misma.

Miembro comunidad: Persona natural directamente relacionada con la universidad, este puede ser un estudiante, profesor, empleado, egresado o familiar de un estudiante.

Vecino: Persona natural que habitan en el sector de la universidad.

Empresa de viviendo universitaria: Es una empresa que posee un edificio de vivienda universitaria y ofrece sus habitaciones en sus ofertas de alojamiento.

Apartamento: Bienes inmuebles que pertenecen a un miembro de la comunidad universitaria y se alquilan en su totalidad a los clientes.

Vivienda: Lugar de residencia de una persona natural que tiene un vínculo con la institución educativa.

Habitación: En algunos de los inmuebles se retan las habitaciones de este. Los tipos de habitaciones se exponen a continuación.

Habitación de un hotel: Es una habitación que pertenece a un hotel. Estas habitaciones se clasifican en tres categorías: estándar, semisuite y suite. El precio varía dependiendo del tipo, la capacidad, la ubicación y los servicios.

Habitación de un hostal: Es una habitación que pertenece a un hostal.

Habitación de una vivienda: Es una habitación que pertenece a la vivienda de una persona natural. El contrato de alojamiento de estas habitaciones debe ser de mínimo un mes.

Habitación de una vivienda universitaria: Es una habitación ofrecida por una empresa de vivienda universitaria. Estas habitaciones se pueden alquilar por mínimo una noche, y tienen opciones mensuales y semestrales. Pueden ser individuales y compartidas y debe ofrecer los servicios presentes en la vivienda universitaria.

## Funcionalidades principales

Las funcionalidades principales de Alohandes son: registrar a los operadores de un alojamiento, registrar las propuestas de alojamiento, registrar a las personas habilitadas para utilizar los servicios, registrar una reserva, cancelar una reserva, retirar una oferta de alojamiento, mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido, mostrar las 20 ofertas más populares, mostrar el índice de ocupación de cada una de las ofertas de alojamiento registradas, mostrar los alojamientos disponibles en un rango de fechas que cumplen con un conjunto de requerimientos de dotación o servicios, mostrar el uso de Alohandes para cada tipo de usuario de la comunidad y mostrar el uso de Alohandes para un usuario dado.

## Reglas de negocio

La implementación de Alohandes cuenta con las siguientes reglas de negocio:

1) Todos los alojamientos deben estar relacionados con Uniandes y los clientes deben tener algún vínculo con la institución por razones de seguridad.

2) Una persona no puede reservar más de un alojamiento en un mismo día.

3) Un alojamiento no acepta reservas que superen su capacidad.

4) El alojamiento en vivienda universitaria sólo puede ser utilizado por estudiantes, profesores, empleados y profesores visitantes.

5) Una oferta sólo se puede retirar del sistema a partir del siguiente día de finalización de la última reserva vigente.

6) EL sistema no debe permitir reservas posteriores a una fecha de retiro programada de una oferta.

7) Si una vivienda no tiene reservas vigentes esta puede ser retirada de forma inmediata.

8) Si un cliente cancela una reserva antes del tiempo mínimo se cobra un gasto administrativo del 10%.

9) Si el cliente cancela una reserva entre el tiempo límite y el día anterior a la fecha de llagad se le cobra el 30% de la reserva.

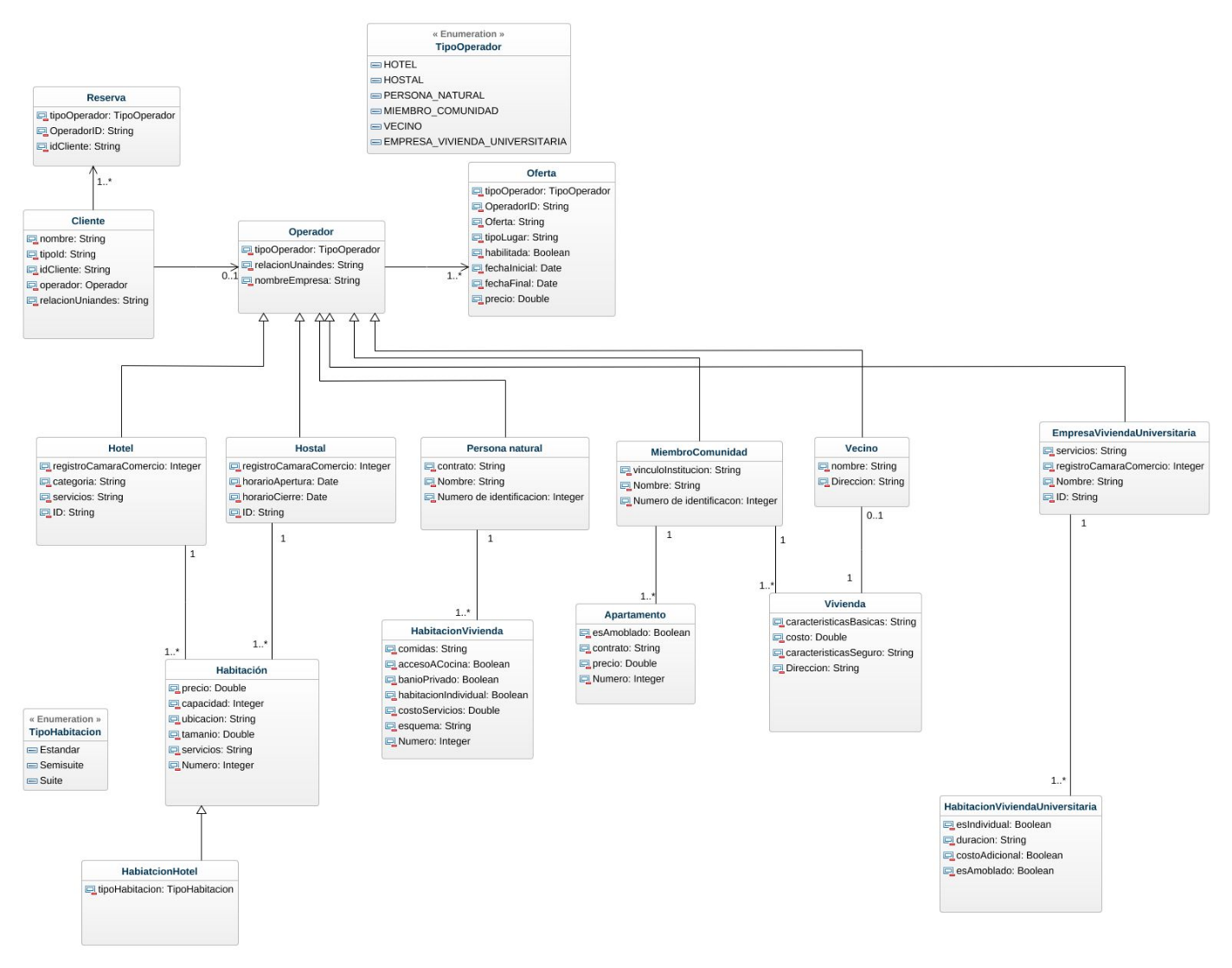
10) Si el cliente decide no utilizar el inmueble o si lo desocupa antes de tiempo se le cobra el 50% del valor contratado.

11) Si al cliente se le cobra por días el tiempo límite se fija en 3 días.

12) Si al cliente se le cobra por semanas o meses el tiempo límite se fija en una semana.

# Diseño y construcción de la base de datos

## UML



**Figura 1.** Modelo UML

## Modelo relacional

Hotel

|  |  |  |
| --- | --- | --- |
| ID | Registro cámara de comercio | Categoría |
| Pk | NN | NN |

**Tabla1**. Hotel

Hostal

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Registro cámara de comercio | Apertura | Cierre |
| Pk | NN | NN | NN |

**Tabla2**. Hostal

Persona natural

|  |  |  |
| --- | --- | --- |
| Contrato | Número de identificación | Nombre |
| NN | PK | NN |

**Tabla3**. Persona natural

Miembro comunidad

|  |  |  |
| --- | --- | --- |
| Vinculo | Nombre | Número de identificación |
| NN | NN | PK |

**Tabla4**. Miembro comunidad

Vecino

|  |  |
| --- | --- |
| Nombre | Dirección |
| PK | PK |

**Tabla5**. Vecino

Empresa vivienda universitaria

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Nombre | Registro cámara de comercio | Tiene restaurante | Tiene salas de estudio | Salas esparcimiento | Gimnasio |
| PK | NN | NN | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) |

**Tabla6**. Empresa vivienda universitaria

Habitación

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Precio | Capacidad | Ubicación | Tamaño | Numero |
| NN | NN | NN | NN | PK |

**Tabla7**. Habitación

Habitación vivienda universitaria

|  |  |  |  |
| --- | --- | --- | --- |
| Numero | Individual | Duración | Menaje |
| PK | NN, CHECK (1, 0) | NN | NN |

**Tabla8**. Habitación vivienda universitaria

Habitación vivienda

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Numero | Esquema | costoServicios | Habitación individual | Baño privado | Acceso a cocina | Comidas |
| PK | NN | NN | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) |

**Tabla 9**. Habitación vivienda

Apartamento

|  |  |  |  |
| --- | --- | --- | --- |
| Numero | Contrato | Pecio | EsAmoblado |
| PK | NN | NN | NN |

**Tabla 10**. Apartamento

Vivienda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dirección | Costo | Seguro | numero habitaciones | menaje |
| PK | NN | PK | NN | NN, CHECK (1, 0) |

**Tabla 11**. Vivienda

Operador

|  |  |  |
| --- | --- | --- |
| Tipo | ID | Nombre |
| PK | PK | PK |

**Tabla 12**. Operador

Cliente

|  |  |  |  |
| --- | --- | --- | --- |
| Dirección | TipoID | ID | Relación Uniandes |
| PK | PK | PK | NN |

**Tabla13**. Cliente

Oferta

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID Operador | NumOferta | IDLugar | TipoLugar | Habilitado | fechaInicial | FechaFinal | Precio |
| PK, FK | PK | NN | NN | Check(1,0) | NN | NN | NN |

**Tabla 14**. Oferta

Reservas

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID Cliente | ofertaID | ID Operador | Cantidad de Habitaciones | Fecha final | Fecha inicial | ID Cliente | ofertaID | ID Operador | Cantidad de Habitaciones |
| FK, PK | PK, FK | FK | NN | NN | NN | FK, PK | PK, FK | FK | NN |

**Tabla 15**. Reservas

ServiciosHotel

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| HotelID | Restaurante | Piscina | Parqueadero | Wifi | TV | 24h |
| Pk, FK | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) |

**Tabla 16**. ServiciosHotel

ServiciosHabitacion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HabitacionID | bañera | yacuzzi | sala | cocineta |
| Pk, FK | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) | NN, CHECK (1, 0) |

**Tabla 17**. ServiciosHabitacion

GananciaOperador

|  |  |
| --- | --- |
| OperadorID | Ganancia |
| Pk, FK | NN |

**Tabla18**. GananciaOperador

En el diseño establecido previamente se realizaron algunos cambios para mejorar el desempeño de la aplicación. Para manejar de mejor manera las ofertas con las cuales opera Alohandes se añadieron tres atributos en la tabla correspondiente. Estos son los atributos *fechaInicial, fechaFinal* y *precio.*  Teniendo en cuenta que varios de los requerimientos que se deben añadir para esta iteración involucran analizar los datos de las ofertas con las que opera la aplicación, especialmente en los que se debe consultar el consumo en Alohandes (RFC10, RFC11 Y RFC12).

# Índices

Como el id del cliente es una llave primaria ya se tiene un índice sobre este atributo. Con los datos iniciales el mayor costo de las tres sentencias es 7. Si se crea un índice, ya sea en los atributos de fina inicia o fecha final el costo de cada consulta se mantiene igual (5,6 y 4) para la consulta que ordena por tipo de cliente, la que ordena por oferta y por tipo de alojamiento respectivamente.

En el caso de la consulta asociada al requerimiento 11, si se añade un índice compuesto en la tabla reservas sobre los atributos fechaInicial y fechaFinal el costo baja una unidad.

En el caso del requerimiento 12, si se añade un índice en el atributo cantidad de la tabla reserva el costo de la consulta no cambia. Sin embargo, para la segunda consulta de este requerimiento, si se crea un índice en el atributo ganancia de la tabla gananciaoperador el costo de la consulta disminuye en 3 unidades.

Por último, en el caso del requerimiento 13, en el caso de las tres consultas cuando se crea un índice en los atributos diferentes a las llaves primarias el costo de las consultas se mantiene igual.

# Implementación de los requerimientos funcionales

## Sentencias

**RF10**

-- Cliente

SELECT \*

FROM cliete, reservas

WHERE reservas.idcliente=cliente.id

AND reservas.fechainicial >= '1-DIC-2018'

AND reservas.fechafinal <='31-DIC-2020'

;

-- oferta

SELECT \*

FROM CLIENTE, oferta, reservas

WHERE reservas.idcliente=cliente.id

AND reservas.fechainicial >= '1-DIC-2018'

AND reservas.fechafinal <='31-DIC-2020'

ORDER BY RESERVAS.IDCLIENTE

;

--Tipo de alojamiento

SELECT OFERTA.TIPOLUGAR, cliente.id, cliente.direccion, cliente.tipoId, cliente.relacionuniandes

FROM OFERTA, CLIENTE, RESERVAS

WHERE reservas.idcliente=cliente.id

;

**RF11**

CREATE INDEX indiceFecha ON reservas(fechainicial,fechafinal);

DROP INDEX indiceFecha;

SELECT \*

from

CLIENTE, RESERVAS

WHERE

reservas.idcliente = cliente.id

AND

reservas.fechainicial >= '1-DIC-2018'

AND

reservas.fechafinal <='31-DIC-2020'

;

**RF12**

CREATE INDEX indiceganancia ON gananciaoperador(ganancia);

DROP INDEX indiceganancia;

--DATOS OFERTA

SELECT \* FROM OFERTA, RESERVAS

WHERE RESERVAS.OFERTAID=OFERTA.NUMOFERTA

ORDER BY RESERVAS.CANTIDAD

;

--DATOS OPERADORES

SELECT operador.id, operador.nombre, operador.tipo

FROM OPERADOR, GANANCIAOPERADOR

WHERE OPERADOR.ID= gananciaoperador.idoperador

ORDER BY gananciaoperador.ganancia

;

**RF13**

--Mes

select \* from Cliente, reservas, oferta

WHERE oferta.numoferta=RESERVAS.OFERTAID

AND

RESERVA.IDCLIENTE=CLIENTE.ID

;

--Costoso

SELECT \* FROM CLIENTE, OFERTA, RESERVAS

WHERE oferta.numoferta=RESERVAS.OFERTAID

AND

RESERVAS.IDCLIENTE=CLIENTE.ID

AND

OFERTA.PRECIO>150

;

--Suite

SELECT \* FROM CLIENTE, OFERTA, RESERVAS

WHERE oferta.numoferta=RESERVAS.OFERTAID

AND RESERVAS.IDCLIENTE=CLIENTE.ID

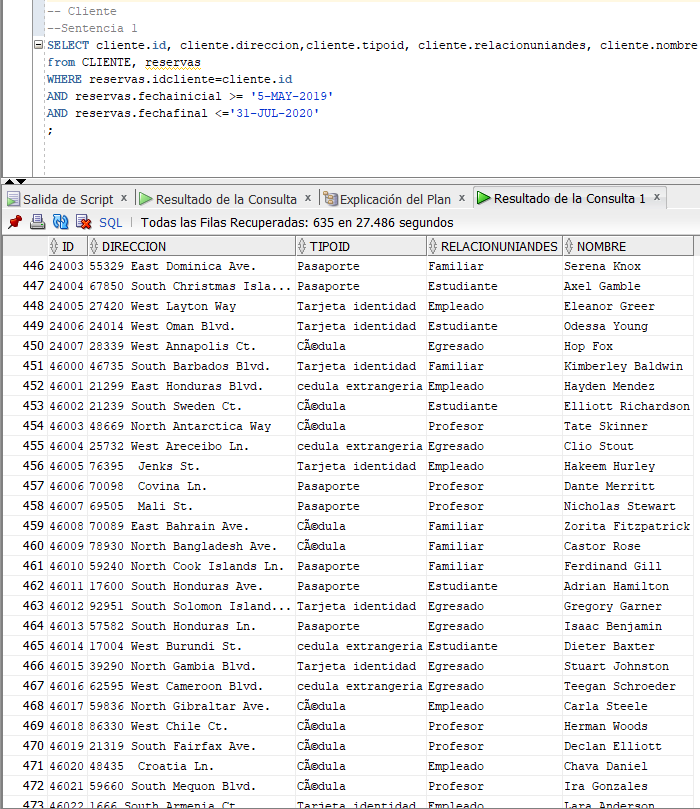
AND OFERTA.TIPOLUGAR='suite'

;

## Resultados Logrados

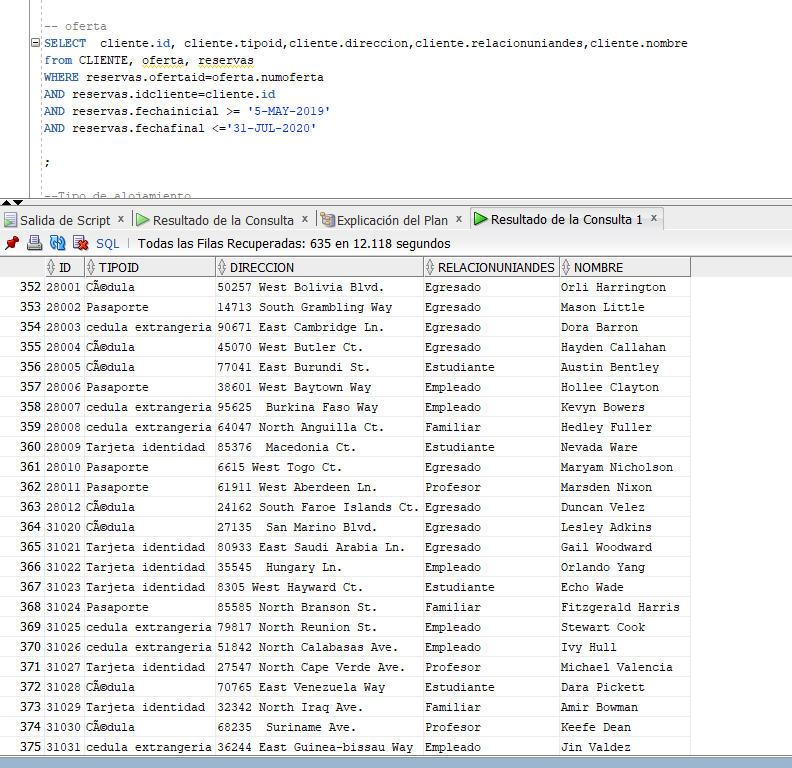
Ejecución de las consultas del RFC10:

Sentencia 1: mostrar datos de un cliente en un rango de fechas.

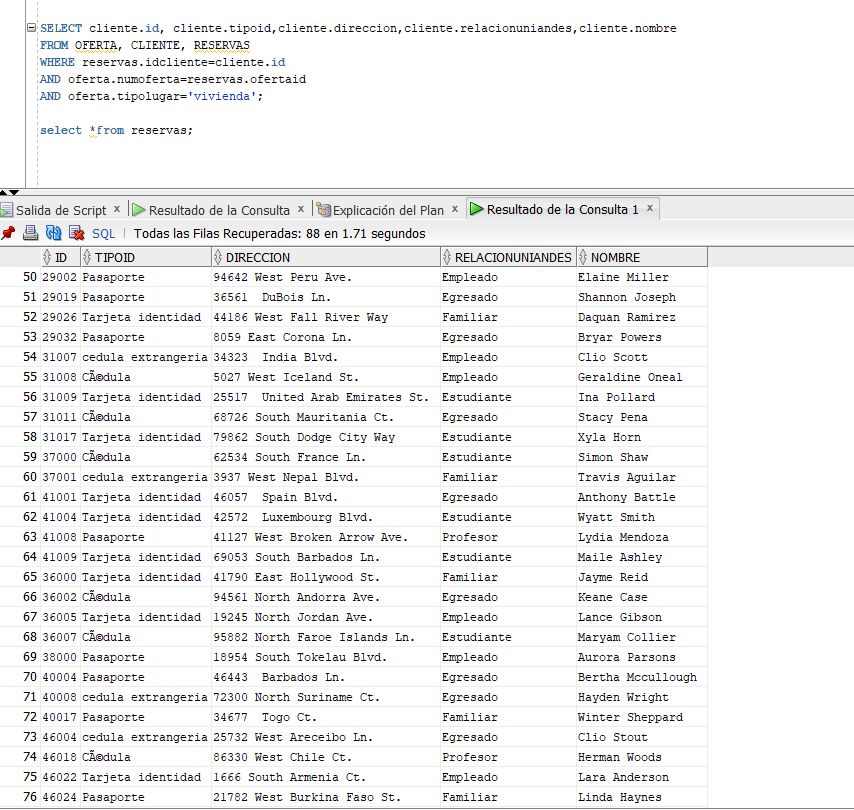


Sentencia 2: Mostrar los datos de un cliente por la oferta

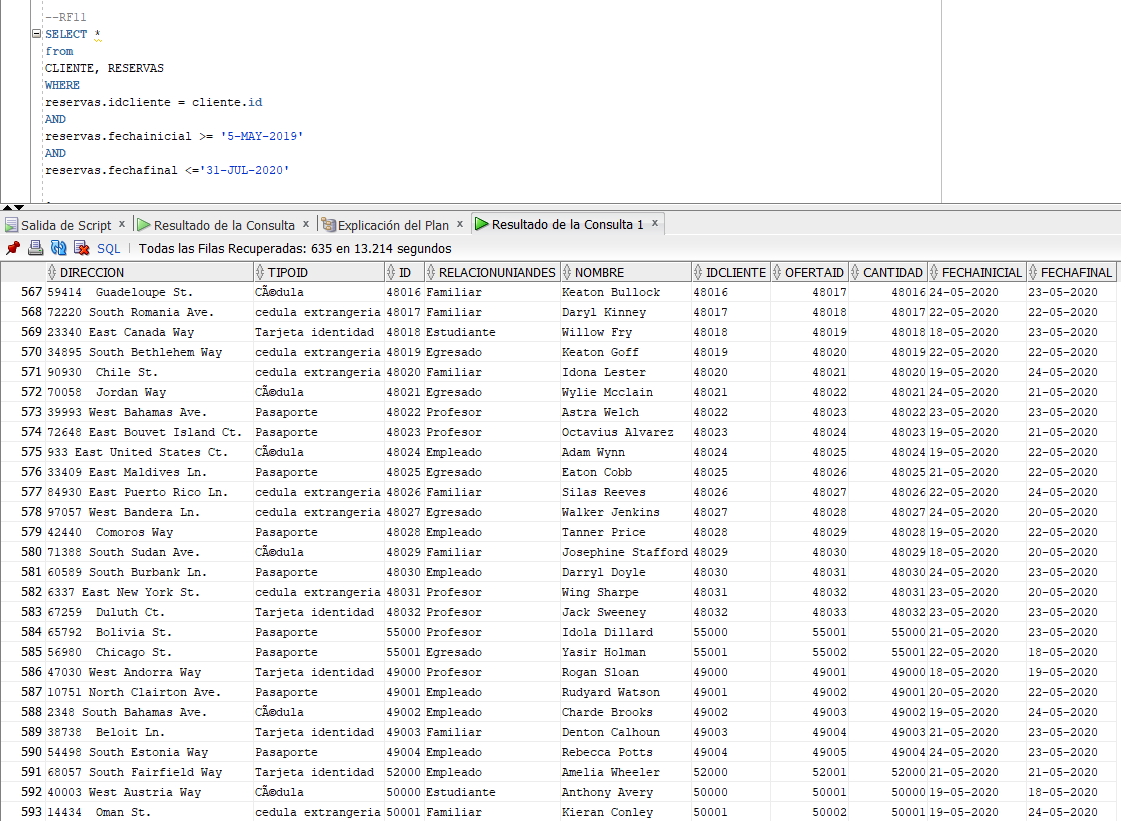
Buscar datos de un cliente por una oferta:



Sentencia 3: Buscar datos de un cliente por tipo de alojamiento



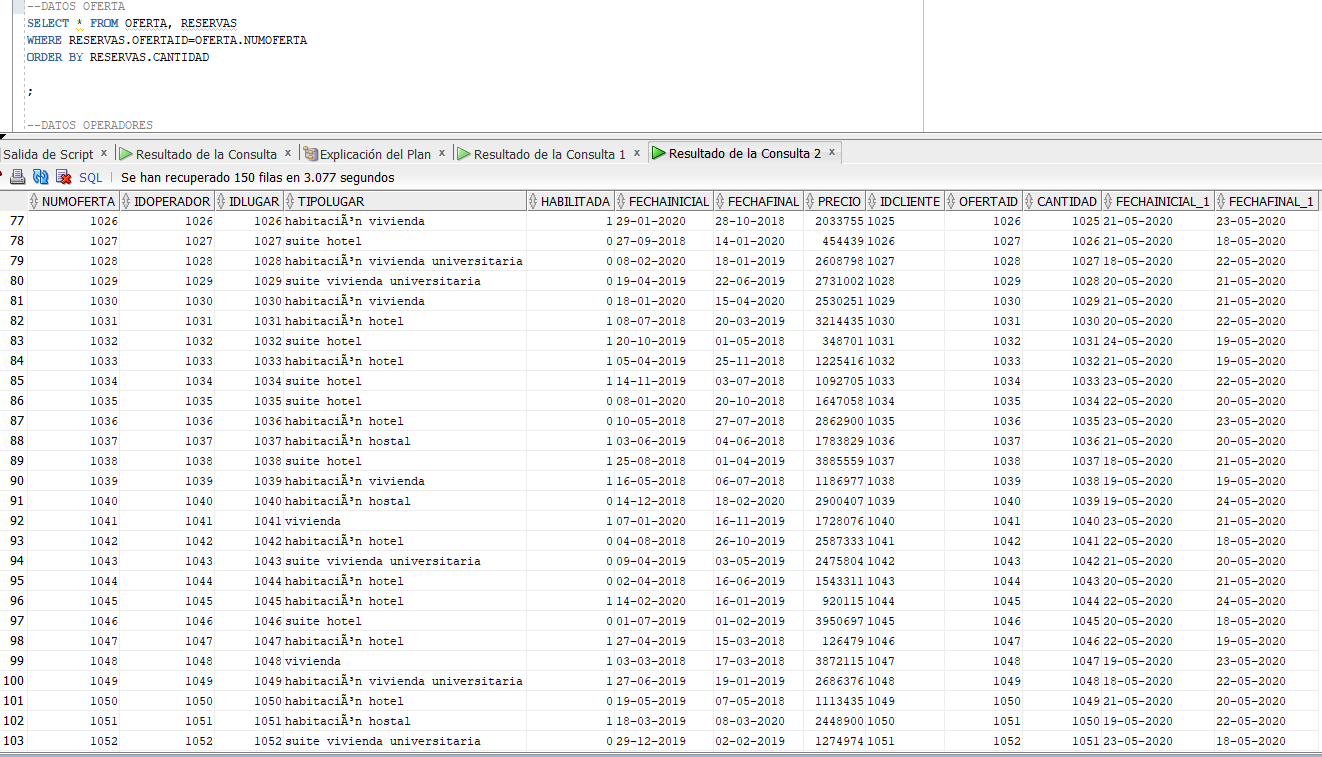
RFC11



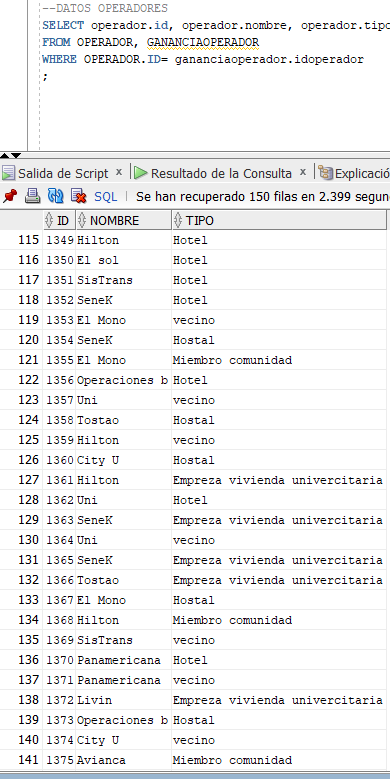
RFC12

Buscar la información de un cliente por el tipo de alojamiento que visita

Sentencia 1: Obetener datos por oferta

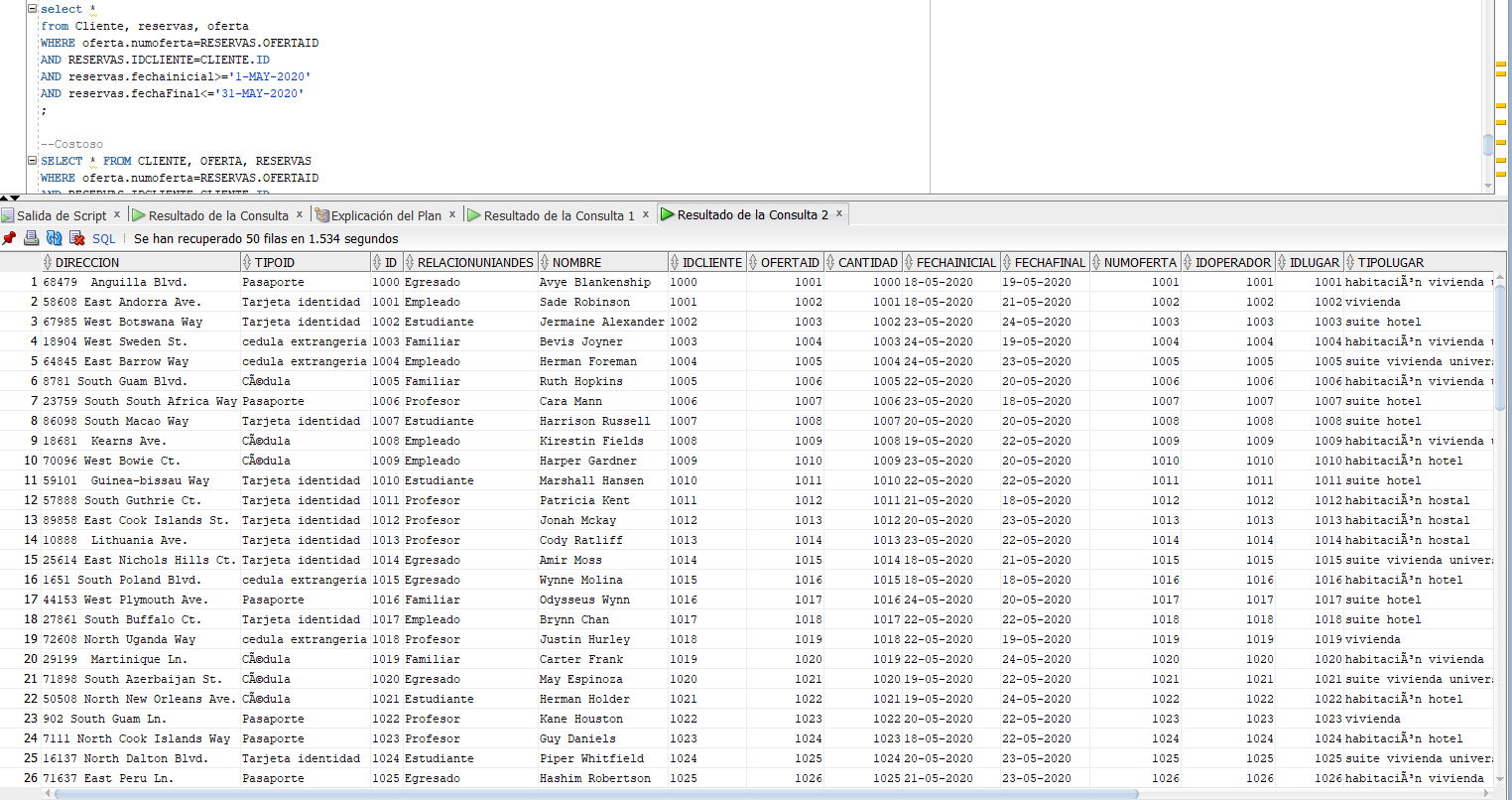


Sentencia 2: Obtener datos por operador

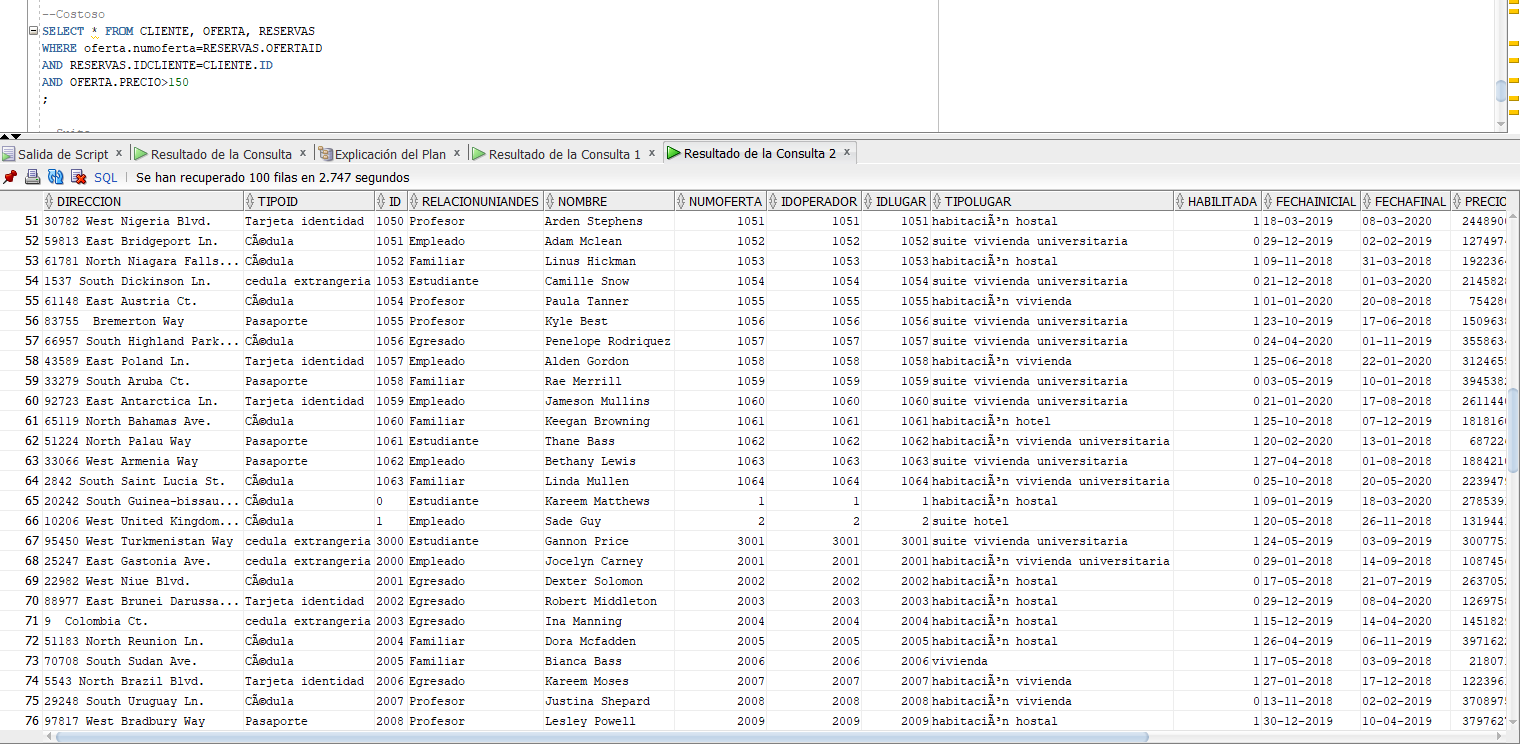


RFC13

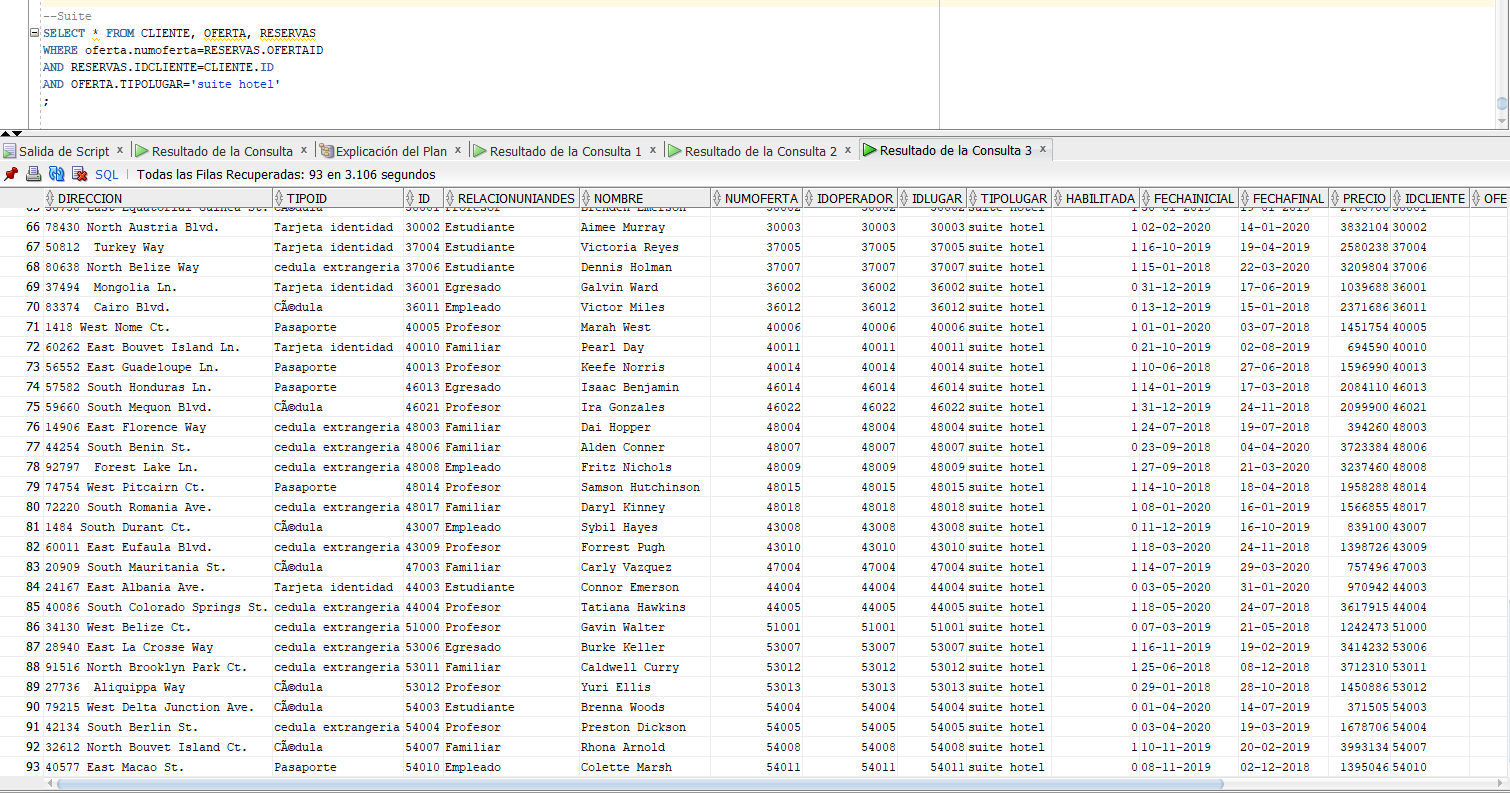
Sentencia 1: consultar por mes



Sentencia 2: consultar por costo



Sentencia 3 : consultar clientes que reservan suites

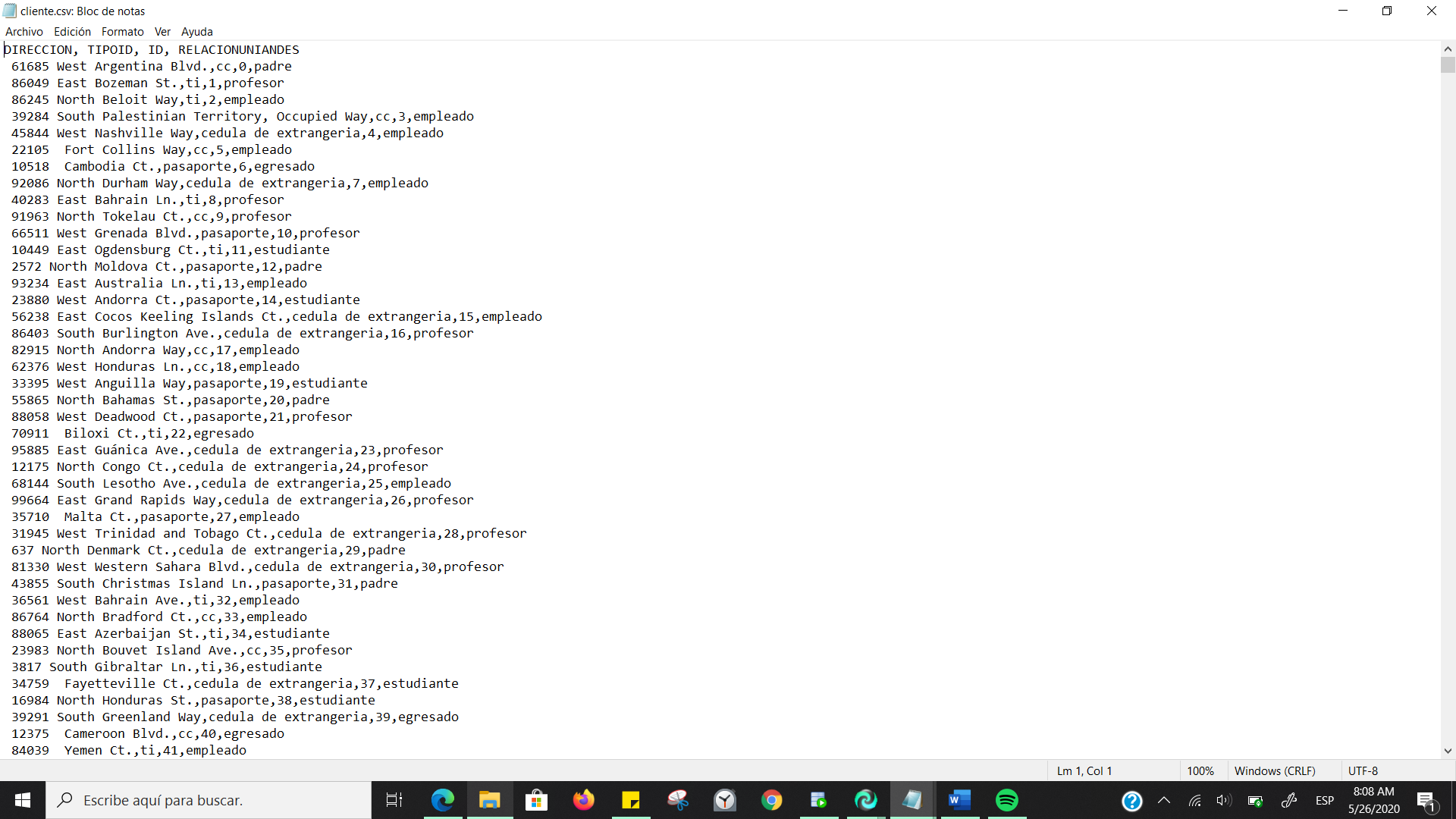


# Construcción de la aplicación, ejecución de pruebas y análisis de resultados

## Proceso para la carga masiva de datos

El proceso a seguir para cumplir con la cantidad de datos solicitada fue el siguiente:

1. Por medio del generador Spawner se configuraron las características de cada atributo de una tabla.
2. Se generó un archivo separado por comas con el formato presentado en la figura 2.



**Figura 2.** Formato csv para la carga de cliente

1. Se importó el archivo .csv de cada tabla en sql developer por medio de la opción *importar datos* como se puede observar en la figura 3.

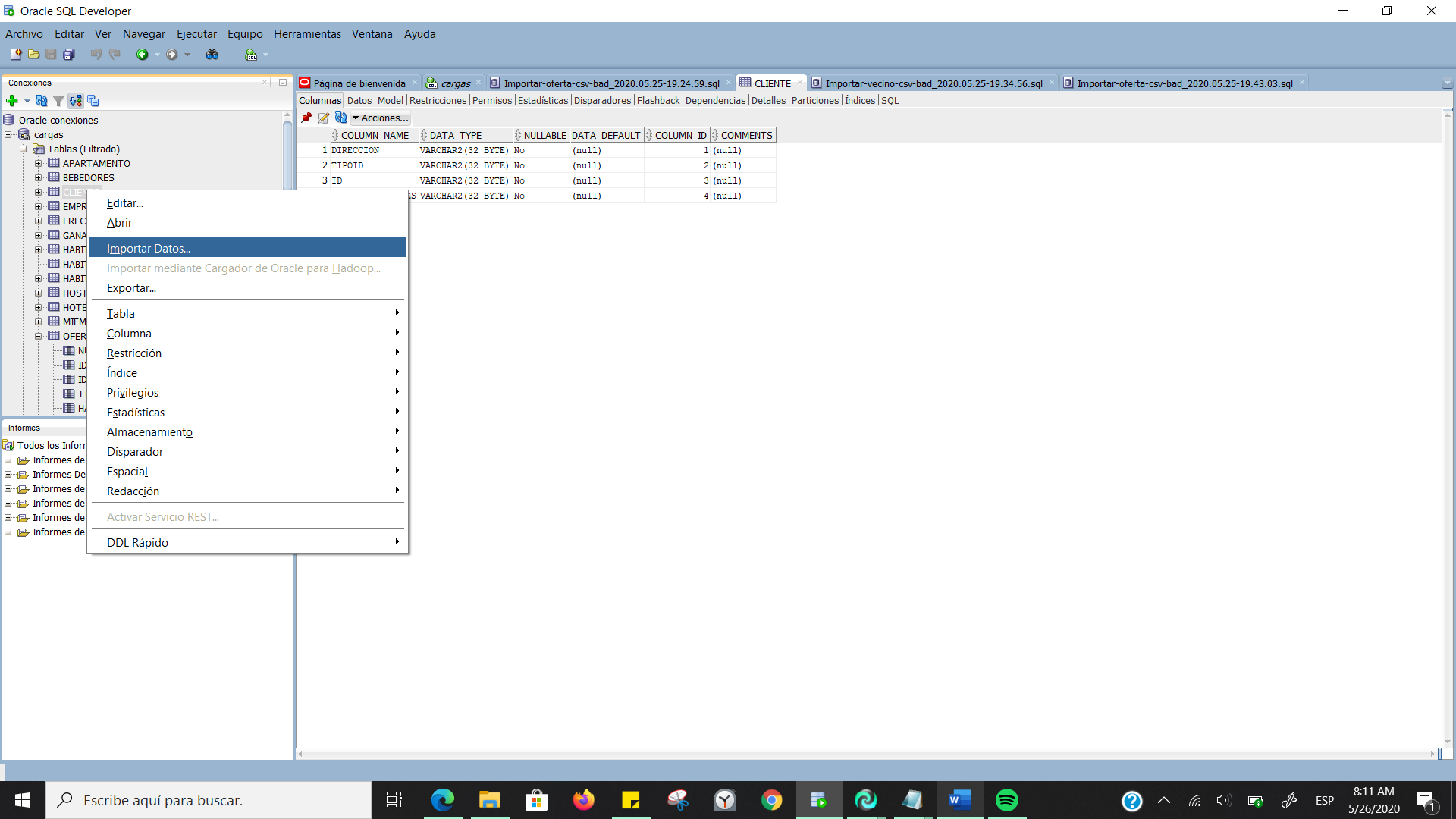


Figura 3. Importación de datos

1. Por último, se ajustó el formato de los campos que contenían fechas en los archivos que contenían estos tipos de dato.

Este proceso se realizó para todas las tablas de la base de datos de Alohandes. Se cargaron 56000 datos por cada tabla, para un total de alrededor de 1008000 datos, debido a que algunas tuplas no se insertaron debido a que no cumplían con las restricciones de longitud de algunos atributos que contenían cadenas de caracteres.

En promedio, cada archivo tardó 20 segundos en cargarse a su respectiva tabla. Esta fue la forma más eficiente de cargarlos debido a que si la carga se realizaba manualmente, es decir generando 56000 sentencias de insert y luego corriéndolas desde un script, se cargaban aproximadamente 20000 tuplas en una hora.

## Escenario de pruebas

Sentencias sql utilizadas

**--RF10**

**-- Cliente**

**--Sentencia 1**

**SELECT cliente.id, cliente.direccion,cliente.tipoid, cliente.relacionuniandes**

**from CLIENTE, reservas**

**WHERE reservas.idcliente=cliente.id**

**AND reservas.fechainicial >= '5-MAY-2019'**

**AND reservas.fechafinal <='31-JUL-2020'**

**;**

**-- oferta**

**SELECT cliente.id, cliente.tipoid,cliente.direccion,cliente.relacionuniandes,cliente.nombre**

**from CLIENTE, oferta, reservas**

**WHERE reservas.ofertaid=oferta.numoferta**

**AND reservas.fechainicial >= '5-MAY-2019'**

**AND reservas.fechafinal <='31-JUL-2020'**

**;**

**--Tipo de alojamiento**

**SELECT cliente.id, cliente.tipoid,cliente.direccion,cliente.relacionuniandes,cliente.nombre**

**FROM OFERTA, CLIENTE, RESERVAS**

**WHERE reservas.idcliente=cliente.id**

**AND oferta.numoferta=reservas.ofertaid**

**AND oferta.tipolugar='hotel';**

**select \*from reservas;**

**--RF11**

**SELECT \***

**from**

**CLIENTE, RESERVAS**

**WHERE**

**reservas.idcliente = cliente.id**

**AND**

**reservas.fechainicial >= '5-MAY-2019'**

**AND**

**reservas.fechafinal <='31-JUL-2020'**

**;**

**--RF12**

**--RF 12**

**--DATOS OFERTA**

**SELECT \* FROM OFERTA, RESERVAS**

**WHERE RESERVAS.OFERTAID=OFERTA.NUMOFERTA**

**AND oferta.numoferta=6790**

**ORDER BY RESERVAS.CANTIDAD**

**;**

**--DATOS OPERADORES**

**SELECT operador.id, operador.nombre, operador.tipo**

**FROM OPERADOR, GANANCIAOPERADOR**

**WHERE OPERADOR.ID= gananciaoperador.idoperador**

**AND gananciaoperador.idoperador=5678;**

**--RF13**

**--Mes**

**select \***

**from Cliente, reservas, oferta**

**WHERE oferta.numoferta=RESERVAS.OFERTAID**

**AND RESERVAS.IDCLIENTE=CLIENTE.ID**

**AND reservas.fechainicial>='1-MAY-2020'**

**AND reservas.fechaFinal<='31-MAY-2020'**

**;**

**--Costoso**

**SELECT \* FROM CLIENTE, OFERTA, RESERVAS**

**WHERE oferta.numoferta=RESERVAS.OFERTAID**

**AND RESERVAS.IDCLIENTE=CLIENTE.ID**

**AND OFERTA.PRECIO>150**

**;**

**--Suite**

**SELECT \* FROM CLIENTE, OFERTA, RESERVAS**

**WHERE oferta.numoferta=RESERVAS.OFERTAID**

**AND RESERVAS.IDCLIENTE=CLIENTE.ID**

**AND OFERTA.TIPOLUGAR='suite'**

**;**

Selectividad:

Planes de consulta:

Rf10:

Sentencia 1:

|  |  |
| --- | --- |
| Valores sentencia versión 1 | Fecha inicial 1: '1-DIC-2018'  Fecha final 1: '31-DIC-2020' |
| Valores sentencia versión 2 | Fecha inicial 2: '5-MAY-2019'  Fecha final 1: '31-JUL-2020' |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Tiempo 1 | 2.119s |
| Tiempo2 | 2.011s |

Sentencia2

|  |  |
| --- | --- |
| Valores sentencia versión 1 | Fecha inicial 1: '1-DIC-2018'  Fecha final 1: '31-DIC-2020' |
| Valores sentencia versión 2 | Fecha inicial 2: '5-MAY-2019'  Fecha final 1: '31-JUL-2020' |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Tiempo 1 | 1.013s |
| Tiempo 2 | 18.414 s |

Sentencia 3:

|  |  |
| --- | --- |
| Valores sentencia versión 1 | Vivienda |
| Valores sentencia versión 2 | Hotel |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Tiempo 1 | 4.915s |
| Tiempo 2 | 1.065s |

RF11

|  |  |
| --- | --- |
| Valores sentencia versión 1 | Fecha inicial 1: '1-DIC-2018'  Fecha final 1: '31-DIC-2020' |
| Valores sentencia versión 2 | Fecha inicial 2: '5-MAY-2019'  Fecha final 1: '31-JUL-2020' |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Tiempo 1 | 14.811s |
| Tiempo 2 | 47.326s |

RF12

|  |  |
| --- | --- |
| Valores sentencia versión 1 | oferta.numoferta=6790 |
| Valores sentencia versión 2 | Id del operador=5678 |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Tiempo 1 | 1.757s |
| Tiempo 2 | 0.958s |

RF13

Sentencia 1:

|  |  |
| --- | --- |
| Cliente que ha reservado en el último mes | Fecha inicial 1: '1-MAY-2020'  Fecha final 1: '31-MAY-2020' |
| Cliente que reserva ofertas costas | Precio es mayor a 150 |
| Cliente que reserva una suite | TipoLugar=suite |
| Plan de consulta 1 |  |
| Plan de consulta 2 |  |
| Plan de consulta 3 |  |
| Tiempo 1 | 1.214s |
| Tiempo 2 | 1.166s |
| Tiempo 3 | 1.412s |

# Análisis de eficiencia

La complejidad de los planes de ejecución que teníamos presupuestados era menos compleja en términos de cantidad de operaciones, que los planes ejecutados por Oracle, que se pueden observar en las imágenes de la sección de pruebas. Esto se debe a que no tuvimos en cuenta algunos factores para hacer join de todas las tablas implicadas en la consulta.

En el caso del RFC10 pensamos en hacer primero los accesos a las tablas de la consulta y como máximo tres hash joins para juntar las tablas de cliente y reservas, cliente y operaciones y cliente, reservas y oferta. Sin embargo, en el plan de Oracle surgen otras operaciones como nested loops que no habíamos tenido en cuenta.

En el caso del requerimiento RFC11 consideramos utilizar nested loops para acceder a las tablas. Sin embargo, en el plan de consulta de Oracle se incluye un hash join al inicio y después sí ejecuta los nested loops.

Para el RFC12, pensamos cargar las 2 tablas y hacer un join de manera que se viera el más solicitado y el menos solicitado., mientras que oracle lo resuelve con varios nested loops.

Para el RFC13 , lo pensamos ejecutar con un hash join de cliente y la reserva que ha hecho de manera que se clasificara para ver el cliente ideal dependiendo de la característica de este(haber hecho una reserva en el último mes, hospedarse en alojamientos de más de $150 o alquilar suites).Sin embargo, Oracle utiliza nested loops.

Al comparar nuestros planes de consulta con los planes resultantes propuestos por Oracle, podemos afirmar que los nuestros tenían menos operaciones por realizar por lo que el costo podría ser menor, sin embargo, con nuestras propuestas se podrían crear inconsistencias en las respuestas, por lo que preferimos adherirnos al plan propuesto por Oracle después de crear los respectivos índices.

# Análisis del Proceso

|  |  |
| --- | --- |
| Ejecución de consultas | |
| Oracle | Aplicación |
| * Mas rápido * Mayor capacidad * Menos limitaciones con la memoria * Oracle da 635 tuplas máximo por consulta | * Lento * Limitaciones con la memoria |