

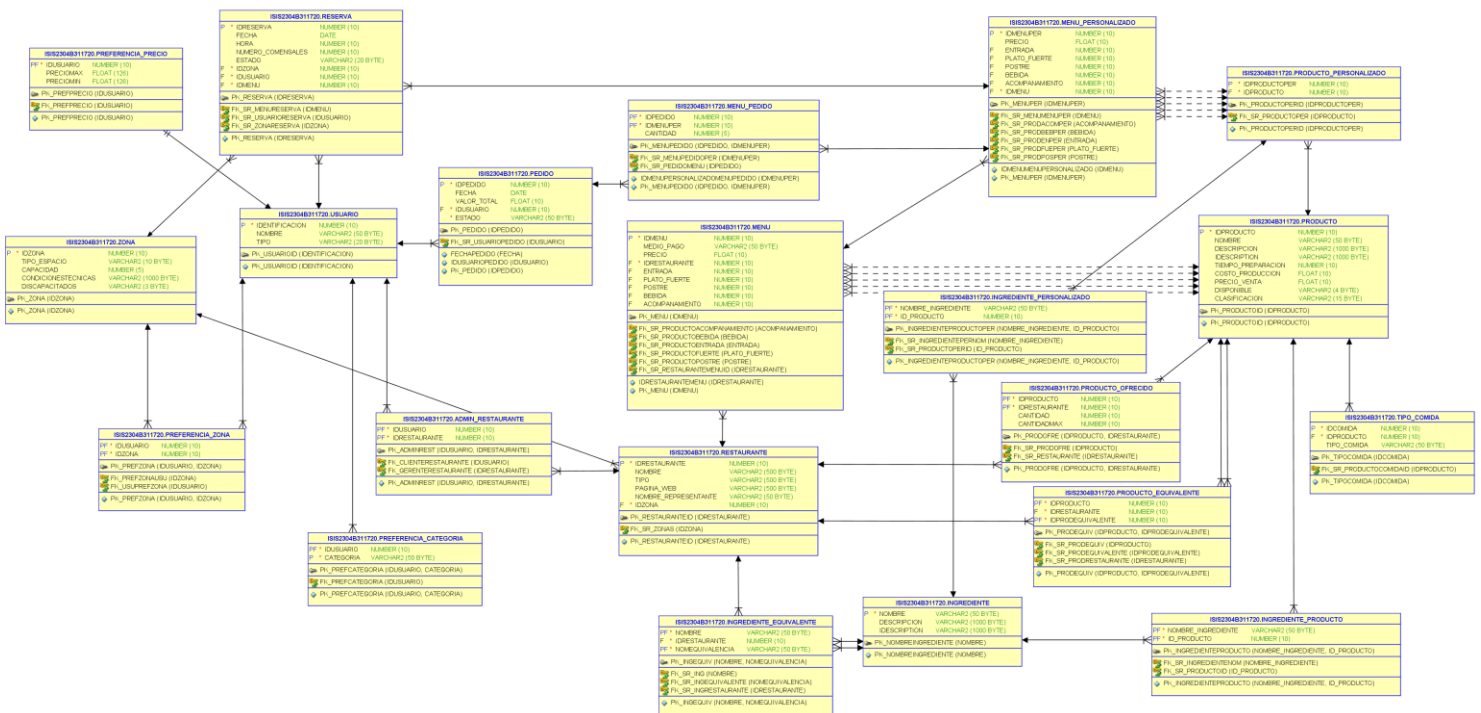
# Sistemas transaccionales - Caso de estudio Rotond Andes

## Iteración 4

1-Algunos de los cambios realizados con respecto a las iteraciones pasadas para cumplir con los objetivos de la nueva fueron:

- ✓ Se agregó una nueva tabla que relaciona un administrador con un restaurante para cumplir con una de las condiciones del requerimiento nueve. Es decir, para asociar un administrador con los restaurantes que administra el mismo y que solamente pueda consultar el consumo del restaurante del cual es administrador.
- ✓ Se extendió la capacidad de las descripciones de productos e ingredientes.
- ✓ Se agregó un nuevo rol de usuario llamado AdminRestaurante, el cual representa un administrador de uno o más restaurantes.
- ✓ En las demás tablas no hicimos ningún cambio adicional.

A continuación se tiene el diagrama relacional actual.



## 2-Diseño Físico

-> Justificar la selección de índices para cada uno de los requerimientos funcionales, indicar cuál es el tipo de índice utilizado (B+, Hash, Primario o secundario) Agregar costos.

## Selección de índices

**Requerimiento 9:** Para el requerimiento nueve se utilizaron los siguientes índices:

- ✓ Índice en el id de un Pedido de la tabla menú pedido.\*
- ✓ Índice en el id de restaurante de la tabla menú.
- ✓ Índice en el id de menú de la tabla menú personalizado.
- ✓ Índice en el id de menú personalizado en la tabla menú pedido.
- ✓ Índice la identificación del usuario.\*
- ✓ Índice en el id de usuario de la tabla pedido.
- ✓ Índice en el id de pedido de la tabla pedido.\*

**Nota:** Índices marcados con \* son creados automáticamente por SQLdeveloper por ser o formar parte de la llave primaria.

Los índices fueron creados de la siguiente manera:

```
CREATE INDEX idPedido ON MENU_PEDIDO(IDPEDIDO);
CREATE INDEX idMenu ON MENU(IDRESTAURANTE);
CREATE INDEX idMenuMenuPersonalizado ON MENU_PERSONALIZADO (IDMENU);
CREATE INDEX idMenuPersonalizadoMenuPedido ON MENU_PEDIDO(IDMENUPER);
CREATE INDEX identificacionUsuario ON USUARIO(IDENTIFICACION);
CREATE INDEX idUsuarioPedido ON PEDIDO(IDUSUARIO);
CREATE INDEX idPedidoEnPedido ON PEDIDO(IDPEDIDO);
```

Después se llegó a esta configuración para mejorar la eficiencia.

A continuación se muestran los índices que se planearon para el desarrollo del requerimiento número nueve (en rojo). Los que inician con PK\_NAME fueron creados automáticamente por SQL developer (en verde).

INDEX_NAME	STATUS	COLUMNS	COMMENTS	UNIQUENESS	DISTINC...	NUM_ROWS	LAST_ANA...	LAST_DD...	CREATED
1 FECHAPEDIDO	VALID	1 (null)		NONUNIQUE	86	100	21/11/17	21/11/17	21/11/17
2 IDMENUMENUPERSONALIZADO	VALID	1 (null)		NONUNIQUE	68	100	21/11/17	21/11/17	21/11/17
3 IDMENUPERSONALIZADOMENUPEDIDO	VALID	1 (null)		NONUNIQUE	41	50	21/11/17	21/11/17	21/11/17
4 IDRESTAURANTEMENU	VALID	1 (null)		NONUNIQUE	20	100	21/11/17	21/11/17	21/11/17
5 IDUSUARIOPEDIDO	VALID	1 (null)		NONUNIQUE	20	100	21/11/17	21/11/17	21/11/17
6 PK_ADMINREST	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
7 PK_INGEQUIV	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
8 PK_INGREDIENTEPRODUCTO	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
9 PK_INGREDIENTEPRODUCTOPER	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
10 PK_MENU	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
11 PK_MENUPEDIDO	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
12 PK_MENUPER	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
13 PK_NOMBREINGREDIENTE	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
14 PK_PEDIDO	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
15 PK_PREFCATEGORIA	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
16 PK_PREFPRECIO	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
17 PK_PREFZONA	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
18 PK_PRODEQUIV	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
19 PK_PRODOFRE	VALID	2 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
20 PK_PRODUCTOID	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
21 PK_PRODUCTOPERID	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
22 PK_RESERVA	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
23 PK_RESTAURANTEID	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
24 PK_TIPOCOMIDA	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
25 PK_USUARIOID	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17
26 PK_ZONA	VALID	1 (null)		UNIQUE	(null)	(null)	(null)	21/11/17	21/11/17

Los índices fueron creados por SQL developer sobre las llaves primarias de la tabla, porque la selectividad de estos es igual a la mejor selectividad posible. Para probar esto, analicemos lo siguiente:

Selectividad = 1 / Cantidad\_de\_filas\_con\_valores\_distintos = 1/100000 (Las llaves primarias son únicas)  
Mejor\_Selectividad = 1 / 100000 (El número de llaves es el mismo que el de filas de la tabla)

La selectividad calculada para este ejemplo es de 0.00001 y la mejor selectividad es de 0.00001. Entre más se acerque la selectividad calculada a la mejor selectividad es más eficiente escoger dicho índice, y mucho mejor si es la misma como en el caso anterior. Los índices mejoraron el rendimiento porque en la consulta del requerimiento nueve, se realizan varios joins sobre las llaves primarias, entonces la creación de estos índices permite una consulta más óptima pues en este caso los índices disminuyen el costo de búsqueda, entonces se puede decir que la decisión que tomó SQL developer para asignar índices en las llaves primarias mejoró el rendimiento de la búsqueda.

**Requerimiento 10:** Para el requerimiento diez se utilizaron los siguientes índices:

- ✓ Índice en el id de un Pedido de la tabla menú pedido.\*
- ✓ Índice en el id de restaurante de la tabla menú.
- ✓ Índice en el id de menú de la tabla menú personalizado.
- ✓ Índice en el id de menú personalizado en la tabla menú pedido.
- ✓ Índice la identificación del usuario.\*
- ✓ Índice en el id de usuario de la tabla pedido.
- ✓ Índice en el id de pedido de la tabla pedido.\*

**Nota:** Índices marcados con \* son creados automáticamente por SQLdeveloper por ser o formar parte de la llave primaria.

```
Select Usuario.Identificacion,Usuario.Nombre From usuario left join
(((menu join menu_personalizado on menu.idmenu = menu_personalizado.idMenu and menu.idrestaurante = 2)
join menu_pedido on menu_personalizado.idmenuper = menu_pedido.idmenuper) join pedido on pedido.idpedido = menu_pedido.idpedido
and pedido.fecha between '01/12/1999' and '31/12/2019') on Usuario.Identificacion = pedido.Idusuario where pedido.Idusuario IS NULL;
```

Los índices fueron creados de la siguiente manera:

```
CREATE INDEX idPedido ON MENU_PEDIDO(IDPEDIDO);
CREATE INDEX idMenu ON MENU(IDRESTAURANTE);
CREATE INDEX idMenuMenuPersonalizado ON MENU_PERSONALIZADO (IDMENU);
CREATE INDEX idMenuPersonalizadoMenuPedido ON MENU_PEDIDO(IDMENUPER);
CREATE INDEX identificacionUsuario ON USUARIO(IDENTIFICACION);
CREATE INDEX idUsuarioPedido ON PEDIDO(IDUSUARIO);
CREATE INDEX idPedidoEnPedido ON PEDIDO(IDPEDIDO);
```

Después se llegó a esta configuración para lograr una mejor eficiencia.

```
CREATE INDEX idPedido ON MENU_PEDIDO(IDMENUPER,IDPEDIDO);
CREATE INDEX idMenuPer ON MENU(IDRESTAURANTE);
CREATE INDEX idMenu ON MENU_PERSONALIZADO(entrada,plato_fuerte,postre,bebida,acompanamiento);
CREATE INDEX idUsuarioPedido ON PEDIDO (IDUSUARIO,ESTADO,Fecha);
```

Dado que el requerimiento nueve y diez son similares, se puede observar que los índices escogidos para el requerimiento diez son los mismos del requerimiento nueve. Se escogieron estos índices para lograr una mejor eficiencia porque son los principales atributos que se involucran en las dos consultas.

- Requerimiento 11:** Para el requerimiento diez se utilizaron los siguientes índices:
- ✓ Índice en el id de Índice en el id de pedido en la tabla menú pedido.\*
  - ✓ Índice en el id de menú personalizado menú pedido.
  - ✓ Índice en el id de menú de la menú personalizado.

**Nota:** Índices marcados con \* son creados automáticamente por SQLdeveloper por ser o formar parte de la llave primaria.

Los índices se crearon de la siguiente manera:

```
CREATE INDEX idPedido ON MENU_PEDIDO(IDPEDIDO);
CREATE INDEX idMenuPer ON MENU_PEDIDO(IDMENUPER);
CREATE INDEX idMenu ON MENU_PERSONALIZADO(IDMENU);
```

Dado que el requerimiento once implica la utilización de los ids de los pedidos, de los menús personalizados y de los menús en general, se decidió crear los anteriores índices porque en la sentencia SQL que responde al requerimiento se hacen varios joins sobre estos atributos, entonces, de esta forma se disminuiría el costo de la consulta.

La información de los índices creados para este requerimiento es la siguiente:

En la tabla menú pedido:

INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1 ISIS2304B161720	IDPEDIDO	NONUNIQUE	VALID	NORMAL	N	NO	{null}	NO	IDPEDIDO
2 ISIS2304B161720	PK_MENUPEDIDO	UNIQUE	VALID	NORMAL	N	NO	{null}	NO	IDPEDIDO, IDMENUPER

En la tabla menú personalizado:

INDEX_OWNER	INDEX_NAME	UNIQUENESS	STATUS	INDEX_TYPE	TEMPORARY	PARTITIONED	FUNCIDX_STATUS	JOIN_INDEX	COLUMNS
1 ISIS2304B161720	IDMENU	NONUNIQUE	VALID	NORMAL	N	NO	{null}	NO	IDMENU
2 ISIS2304B161720	PK_MENUPER	UNIQUE	VALID	NORMAL	N	NO	{null}	NO	IDMENUPER

## Requerimiento 9:

### Escenario de prueba:

Se desea consultar la identificación y el nombre de los usuarios que al menos hicieron un pedido (que fue entregado) en el restaurante con id 268 en el rango de fechas '01/01/2005' y '31/12/2006'. Dicha consulta se realiza con la siguiente **sentencia SQL1**:

```
SELECT Usuario.Identificacion, Usuario.Nombre, Usuario.tipo FROM usuario join
(((menu join menu_personalizado on menu.idmenu = menu_personalizado.idMenu and menu.idrestaurante = 268)
join menu_pedido on menu_personalizado.idmenuper = menu_pedido.idmenuper) join pedido on pedido.idpedido = menu_pedido.idpedido
and pedido.fecha between '01/01/2005' and '31/12/2006' and pedido.estado = 'Entregado') on Usuario.Identificacion = pedido.Idusuario;
```

Donde se tienen tres parámetros que son: el id del restaurante al cual se desea consultar el consumo, y las dos fechas que representan el rango por el que se desea hacer la consulta.

La anterior sentencia SQL retorna como respuesta los siguientes 20 usuarios.

SQL | Todas las Filas Recuperadas: 20 en 2,588 segundos

	IDENTIFICACION	NOMBRE	TIPO
1	1234199509	Persona 199508	Cliente
2	1234199510	Persona 199509	Cliente
3	1234199559	Persona 199558	Cliente
4	1234199560	Persona 199559	Cliente
5	1234199609	Persona 199608	Cliente
6	1234199610	Persona 199609	Cliente
7	1234199659	Persona 199658	Cliente
8	1234199660	Persona 199659	Cliente
9	1234199709	Persona 199708	Cliente
10	1234199710	Persona 199709	Cliente
11	1234199759	Persona 199758	Cliente
12	1234199760	Persona 199759	Cliente
13	1234199809	Persona 199808	Cliente
14	1234199810	Persona 199809	Cliente
15	1234199859	Persona 199858	Cliente
16	1234199860	Persona 199859	Cliente
17	1234199909	Persona 199908	Cliente
18	1234199910	Persona 199909	Cliente
19	1234199959	Persona 199958	Cliente
20	1234199960	Persona 199959	Cliente

Para demostrar el funcionamiento de la anterior consulta veamos el siguiente ejemplo con algunos datos que se tenían en la tabla:

- 1) Se tienen 200.000 Usuarios registrados en la base de datos. Teniendo en cuenta la tabla anterior, vamos a observar uno de los 20 usuarios mostrados. El usuario con id 1234199509 con nombre “Persona 199508”.

	IDENTIFICACION	NOMBRE	TIPO
1	1234199509	Persona 199508	Cliente

- 2) Vamos a consultar los pedidos que se fueron realizados por dicho usuario en dicho rango de fechas en.

```
SELECT PEDIDO.IDPEDIDO, PEDIDO.FECHA, PEDIDO.IDUSUARIO, PEDIDO.ESTADO, menu.idRESTAURANTE
FROM MENU_PERSONALIZADO join MENU_PEDIDO on MENU_PERSONALIZADO.idmenuper = MENU_PEDIDO.idMenuper join PEDIDO on pedido.idpedido = menu_pedido.idpedido
JOIN MENU ON Menu_personalizado.idmenu = MENU.IDMENU
WHERE IDUSUARIO = 1234199509 and ESTADO = 'Entregado' AND FECHA between '01/01/2005' and '31/12/2006' ;
```

	IDPEDIDO	FECHA	IDUSUARIO	ESTADO	IDRESTAURANTE
1	598526	09/09/05	1234199509	Entregado	268

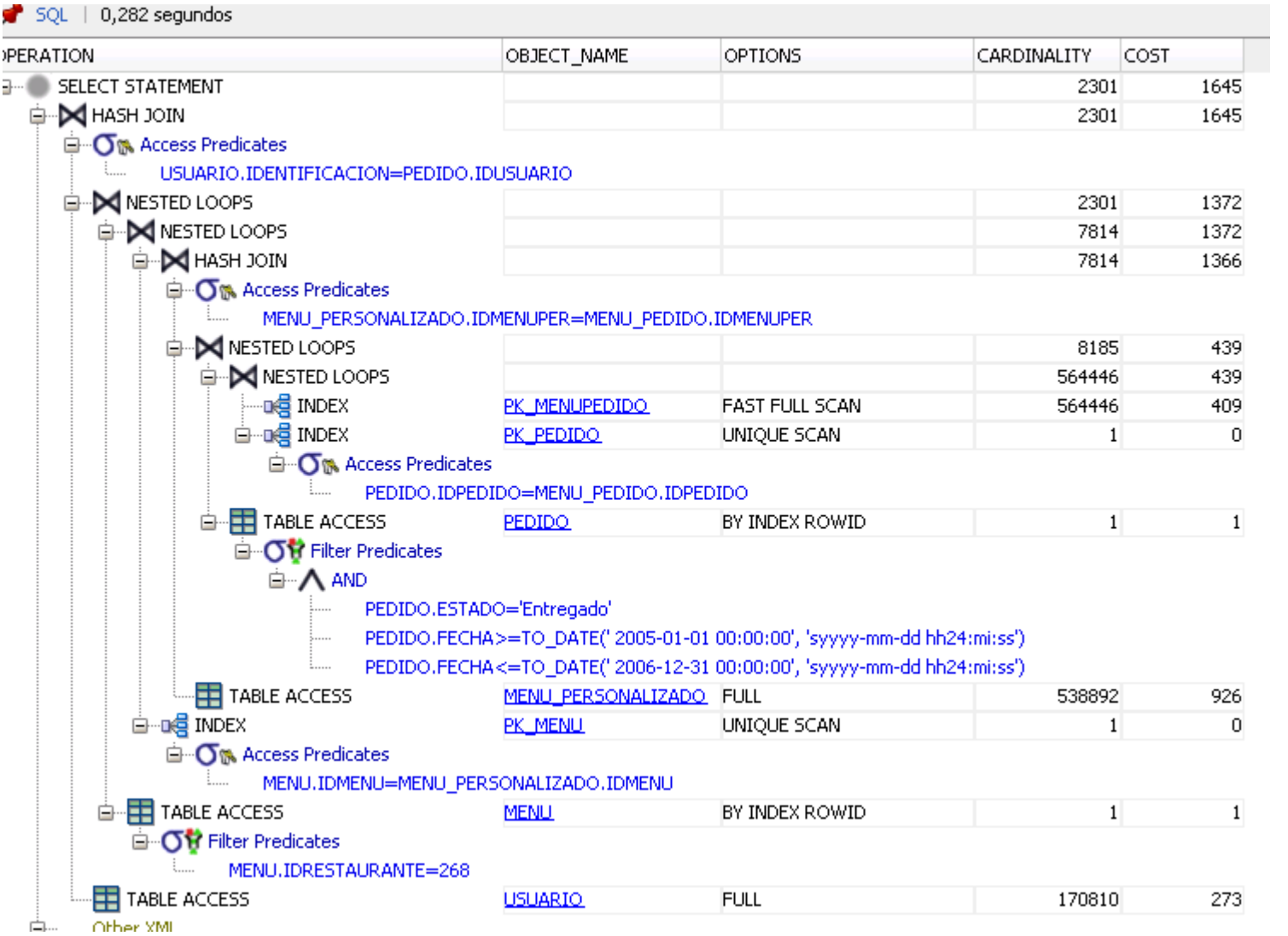
Se puede observar que se realizaron 1 pedidos (en estado entregado) dentro del rango de fechas '01/05/2017' y '31/12/2017' por dicho usuario en el restaurante con id 268. Se pueden concluir que dicho usuario hizo al menos un pedido en el restaurante con id 268 en el rango de fechas establecido, resultado que se evidencia en la respuesta de la sentencia que se planteó para la solución, es decir debería retornar el usuario con id 1234199509 y efectivamente se encuentra en la respuesta.

Distribución de Los datos

Se tiene: 200.000 usuarios, 200.000 productos, aproximadamente 600.000 pedidos, 800 restaurantes, y 600.000 menús.

Plan de consulta generado por SQL Developer

Se puede observar que en el plan que ejecutó SQL developer en primer lugar se utiliza un hash join, varios nested loops, y finalmente el acceso a las tablas por medio de los índices que SQL developer genera de forma automática. Se puede observar que el costo de la operación es de 1645.



Tiempo obtenido con la ejecución del plan

El tiempo que se tardó en ejecutar la sentencia y consultar las dos tuplas fue de 0.282 Segundos. 0,282 segundos Y el tiempo de la consulta es alreder de 4 segundos.

## Requerimiento 10:

### Escenario de prueba:

Se desea consultar la identificación, el nombre y el tipo de los usuarios que no hicieron pedidos o que hicieron pedidos pero ninguno fue entregado, en el restaurante con id 268 en el rango de fechas '01/01/2005' y '31/12/2006'. Dicha consulta se realiza con la siguiente **sentencia SQL2**:

```
SELECT Usuario.Identificacion,Usuario.Nombre FROM usuario left join
(((menu join menu_personalizado on menu.idmenu = menu_personalizado.idMenu and menu.idrestaurante = 268)
join menu_pedido on menu_personalizado.idmenuper = menu_pedido.idmenuper) join pedido on pedido.idpedido = menu_pedido.idpedido
and pedido.fecha between '01/01/2005' and '31/12/2006') on Usuario.Identificacion = pedido.Idusuario where pedido.Idusuario IS NULL;
```

Que retorna como respuesta:

	IDENTIFICACION	NOMBRE
1	1234943	Persona 942
2	1234944	Persona 943
3	1234945	Persona 944
4	1234946	Persona 945
5	1234947	Persona 946
6	1234948	Persona 947
7	1234949	Persona 948
8	1234950	Persona 949
9	1234951	Persona 950
10	1234952	Persona 951
11	1234953	Persona 952
12	1234954	Persona 953
13	1234955	Persona 954
14	1234956	Persona 955
15	1234957	Persona 956

En realidad son demasiados usuarios para mostrarlos en pantalla, sin embargo el resultado de esta consulta debería ser la opuesta a la anterior, pues en la anterior se muestran los usuarios que al menos hicieron un pedido, entonces en esta se deberían mostrar los que no hicieron ningún pedido.

### Para demostrar el funcionamiento de la consulta:

- 1) Se debe tener en cuenta que la primera consulta nos dio como resultado un solo usuario con id 1234199509, el cual solamente realizó un pedido el cual le fue entregado. Entonces, el resultado de esta consulta debería ser todos los usuarios restantes. Para probar esto, vamos a consultar desde la tabla que nos dio como respuesta la sentencia SQL2 si existe un registro del usuario con id 1234199509, usando la siguiente sentencia SQL:

```
SELECT Usuario.Identificacion,Usuario.Nombre FROM usuario left join
(((menu join menu_personalizado on menu.idmenu = menu_personalizado.idMenu and menu.idrestaurante = 268)
join menu_pedido on menu_personalizado.idmenuper = menu_pedido.idmenuper) join pedido on pedido.idpedido = menu_pedido.idpedido
and pedido.fecha between '01/01/2005' and '31/12/2006') on Usuario.Identificacion = pedido.Idusuario where pedido.Idusuario IS NULL
AND USUARIO.identificacion = 1234199509;
```

Que da como resultado:

IDENTIFICACION	NOMBRE
----------------	--------

Es decir, que el resultado es coherente, pues no debería existir un registro del usuario con id 1234199509, ya que el hizo al menos un pedido en ese rango de fechas entonces no debería aparecer en la anterior consulta.

### Distribución de Los datos

Se tiene: 200.000 usuarios, 200.000 productos, aproximadamente 600.000 pedidos, 800 restaurantes, y 600.000 menús.

### Plan de consulta generado por SQL Developer

Se puede observar que en el plan que ejecutó SQL developer en primer lugar se utiliza un hash join, varios nested loops, y finalmente el acceso a las tablas por medio de los índices que SQL developer genera de forma automática. Se puede observar que el costo de la operación es de 2704.



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			170810	2704
HASH JOIN		RIGHT ANTI	170810	2704
Access Predicates	USUARIO.IDENTIFICACION=PEDIDO.IDUSUARIO			
VIEW	SYS.null		726	2430
NESTED LOOPS			726	2430
NESTED LOOPS			1544	2430
HASH JOIN			1544	2429
Access Predicates	MENU_PERSONALIZADO.IDMENUPER=MENU_PEDIDO.IDMENUPER			
HASH JOIN			1544	2019
Access Predicates	MENU.IDMENU=MENU_PERSONALIZADO.IDMENU			
TABLE ACCESS	MENU	FULL	1056	1092
Filter Predicates	MENU.IDRESTAURANTE=268			
TABLE ACCESS	MENU_PERSONALIZADO	FULL	538892	926
INDEX	PK_MENU_PEDIDO	FAST FULL SCAN	564446	409
INDEX	PK_PEDIDO	UNIQUE SCAN	1	0
Access Predicates	PEDIDO.IDPEDIDO=MENU_PEDIDO.IDPEDIDO			
TABLE ACCESS	PEDIDO	BY INDEX ROWID	1	1
Filter Predicates	AND PEDIDO.FECHA>=TO_DATE(' 1999-12-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss') PEDIDO.FECHA<=TO_DATE(' 2019-12-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
TABLE ACCESS	USUARIO	FULL	170810	273
Other XML				

Tiempo obtenido con la ejecución del plan

El tiempo que se tardó en ejecutar la sentencia y consultar las dos tuplas fue de 0.282 Segundos. 1,427 segundos . El tiempo de la consulta fue de masomenos 5 segundos.

Requerimiento 11: En el requerimiento 11 se desea consultar el restaurante y el producto más frecuentado y vendido respectivamente.

Para desarrollar este requerimiento utilizamos la siguiente sentencia SQL:

```
With t1 as (SELECT PRODUCTO.idproducto, to_char(pedido.fecha,'DAY') as numero,count(PRODUCTO.idproducto) as otronumero FROM Producto JOIN (
(PEDIDO JOIN MENU_PEDIDO ON PEDIDO.IDPEDIDO = MENU_PEDIDO.IDPEDIDO)
JOIN MENU_PERSONALIZADO ON MENU_PERSONALIZADO.IDMENUPER = MENU_PEDIDO.IDMENUPER) ON Producto.idproducto = Menu_Personalizado.ENTRADA OR
Producto.idproducto = Menu_Personalizado.PLATO_FUERTE OR
Producto.idproducto = Menu_Personalizado.POSTRE OR
Producto.idproducto = Menu_Personalizado.BEBIDA OR
Producto.idproducto = Menu_Personalizado.ACOMPAÑAMIENTO group by PRODUCTO.idproducto,to_char(pedido.fecha,'DAY') order by count(PRODUCTO.idproducto) desc)

select t1.numero, max(t1.otronumero)
from t1
group by t1.numero;
```

Que da como respuesta lo siguiente:

Salida de Script | Explicación del Plan | Resultado de la Cons

Todas las Filas Recuperadas: 7 en 15,193 segund

NUMERO	MAX(T1.OTRONUMERO)
1 DOMINGO	5
2 JUEVES	5
3 VIERNES	5
4 MARTES	5
5 LUNES	5
6 SÁBADO	5
7 MIÉRCOLES	5





Requerimiento 12:

Sentencia utilizada:

```
select usuario.* from usuario join pedido on pedido.idusuario = usuario.identificacion join MENU_PEDIDO
on MENU_PEDIDO.IDPEDIDO = pedido.IDPEDIDO left join (select IDMENUPER from
MENU_PERSONALIZADO where ACOMPANAMIENTO is null)t1 on t1.IDMENUPER =
MENU_PEDIDO.IDMENUPER;
```

Análisis de eficiencia

**Requerimiento 9:** Para analizar la eficiencia después de haber establecido los índices vamos a ejecutar la sentencia SQL del primer requerimiento para compararla con la consulta sin índices. Al ejecutar la sentencia que planteamos para el requerimiento 9 después de haber creado los índices, se tiene un costo de 1401 que es menor al de la consulta ejecutada sin índices. Se puede observar, que los índices creados afectaron un poco de manera positiva el costo de la primera consulta. Además, hubo una disminución en el tiempo de la consulta pues antes era de más o menos 4 segundos y con índices este tiempo se redujo a 0.156 segundos.

 SQL | Todas las Filas Recuperadas: 20 en 0,156 segundos

	IDENTIFICACION	NOMBRE	TIPO
1	1234199509	Persona 199508	Cliente
2	1234199510	Persona 199509	Cliente
3	1234199511	Persona 199510	Cliente



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			24	1401
NESTED LOOPS			24	1401
NESTED LOOPS			24	1401
NESTED LOOPS			24	1377
HASH JOIN			1544	1354
Access Predicates	MENU_PERSONALIZADO.IDMENUPER=MENU_PEDIDO.IDMENUPER			
NESTED LOOPS			1544	1354
STATISTICS COLLECTOR				
HASH JOIN			1544	943
Access Predicates	MENU.IDMENU=MENU_PERSONALIZADO.IDMENU			
TABLE ACCESS	MENU	BY INDEX ROWID BATCHED	926	16
INDEX	IDMENUPER	RANGE SCAN	750	5
Access Predicates	MENU.IDRESTAURANTE=268			
TABLE ACCESS	MENU_PERSONALIZADO	FULL	538892	926
INDEX	IDPEDIDO	RANGE SCAN	1	409
Access Predicates	MENU_PERSONALIZADO.IDMENUPER=MENU_PEDIDO.IDMENUPER			
INDEX	PK_MENU PEDIDO	FAST FULL SCAN	564446	409
TABLE ACCESS	PEDIDO	BY INDEX ROWID	1	1
Filter Predicates				
AND	PEDIDO.ESTADO='Entregado'			
	PEDIDO.FECHA>=TO_DATE(' 2005-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
	PEDIDO.FECHA<=TO_DATE(' 2006-12-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
INDEX	PK_PEDIDO	UNIQUE SCAN	1	0
Access Predicates	PEDIDO.IDPEDIDO=MENU_PEDIDO.IDPEDIDO			
INDEX	PK_USUARIOID	UNIQUE SCAN	1	0
Access Predicates	USUARIO.IDENTIFICACION=PEDIDO.IDUSUARIO			
TABLE ACCESS	USUARIO	BY INDEX ROWID	1	1
Other XML				
{info}				
info type="db_version"				
12.1.0.2				

**Requerimiento 10:** Para analizar la eficiencia después de haber establecido los índices vamos a ejecutar la sentencia SQL del segundo requerimiento para compararla con la consulta sin índices. Al ejecutar la sentencia que planteamos para el requerimiento 10 después de haber creado los índices, se tiene un costo de 1651 que es menor al de la consulta ejecutada sin índices. Se puede observar, que los índices creados afectaron un poco de manera positiva el costo de la primera consulta. Además, hubo una disminución en el tiempo de la consulta pues antes era de más o menos 5 segundos y con índices este tiempo se redujo a 0.798 segundos.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			170810	1651
HASH JOIN		RIGHT ANTI	170810	1651
Access Predicates	USUARIO.IDENTIFICACION=PEDIDO.IDUSUARIO			
VIEW	SYS.null		73	1377
NESTED LOOPS			73	1377
NESTED LOOPS			1544	1377
HASH JOIN			1544	1354
Access Predicates	MENU_PERSONALIZADO.IDMENUPER=MENU_PEDIDO.IDMENUPER			
NESTED LOOPS			1544	1354
STATISTICS COLLECTOR				
HASH JOIN			1544	943
Access Predicates	MENU.IDMENU=MENU_PERSONALIZADO.IDMENU			
TABLE ACCESS	MENU	BY INDEX ROWID BATCHED	926	16
INDEX	IDMENUPER	RANGE SCAN	750	5
Access Predicates	MENU.IDRESTAURANTE=268			
TABLE ACCESS	MENU_PERSONALIZADO	FULL	538892	926
INDEX	IDPEDIDO	RANGE SCAN	1	409
Access Predicates	MENU_PERSONALIZADO.IDMENUPER=MENU_PEDIDO.IDMENUPER			
INDEX	PK_MENU PEDIDO	FAST FULL SCAN	564446	409
INDEX	PK_PEDIDO	UNIQUE SCAN	1	0
Access Predicates	PEDIDO.IDPEDIDO=MENU_PEDIDO.IDPEDIDO			
TABLE ACCESS	PEDIDO	BY INDEX ROWID	1	1
Filter Predicates				
AND	PEDIDO.FECHA>=TO_DATE(' 2005-01-01 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
AND	PEDIDO.FECHA<=TO_DATE(' 2006-12-31 00:00:00', 'yyyy-mm-dd hh24:mi:ss')			
TABLE ACCESS	USUARIO	FULL	170810	273

El tiempo en el que se desplego la respuesta del requerimiento número diez fue de 0.798 segundos, que considerablemente disminuyó en comparación con la consulta sin índices.

Se han recuperado 50 filas en 0,798 segundos

IDENTIFICACION	NOMBRE
1	1234943 Persona 942
2	1234944 Persona 943

Documentar como fue el proceso de carga de datos, como se realizó como se logró el volumen de datos esperado.

El proceso para cargar datos:

Se escribió un programa en eclipse que genera una cantidad bastante considerable de datos para realizar la prueba de eficiencia, a continuación se muestra una captura de pantalla del código:

```
import java.io.FileWriter;

public class main {

    public static void main(String[] args) throws IOException {

        FileWriter usuarioF = null;
        PrintWriter usuarioW = null;

        FileWriter ingredienteF = null;
        PrintWriter ingredienteW = null;

        FileWriter zonaF = null;
        PrintWriter zonaW = null;

        // FileWriter prefPrecioF = null;
        // PrintWriter prefPrecioW = null;
        //
        // FileWriter prefZonaF = null;
        // PrintWriter prefZonaW = null;
        //
        // FileWriter prefCategoriaF = null;
        // PrintWriter prefCategoriaW = null;

        FileWriter restauranteF = null;
        PrintWriter restauranteW = null;

        FileWriter productoF = null;
        PrintWriter productoW = null;

        FileWriter prodOfF = null;
        PrintWriter prodOfW = null;

        FileWriter ingProvF = null;
        PrintWriter ingProvW = null;

        FileWriter prodPerF = null;
        PrintWriter prodPerW = null;

        FileWriter adminF = null;
        PrintWriter adminW = null;

        FileWriter menuF = null;
        PrintWriter menuW = null;

        FileWriter MENU_PERSONALIZADOF = null;
        PrintWriter MENU_PERSONALIZADOW = null;

        FileWriter PEDIDOF = null;
        PrintWriter PEDIDOW = null;

        FileWriter MENU_PEDIDOF = null;
        PrintWriter MENU_PEDIDOW = null;

        String discapacitado[] = new String[2];
        discapacitado[0] = "Si";
        discapacitado[1] = "No";

        String espacio[] = new String[2];
        espacio[0] = "Abierto";
        espacio[1] = "Cerrado";
```

```

58 estado[2] = "En espera";
59
60 String clasificacion[] = {"Entrada,Fuerte,Postre,Bebida,Acompañamiento".split(",")};
61
62 String frase = "Pienso que éste es el peligro de llevar un diario: se exagera todo, uno está al acecho, forzando continuamente la verdad.Diario."
63 • "El que vive solo ni siquiera sabe qué es contar; lo verosímil desaparece al mismo tiempo que los amigos.Me admira cómo se puede mentir poniendo a la razón de parte de uno."
64 • "No tengo costumbre de contar lo que me sucede, por eso me resulta difícil encontrar la sucesión de los acontecimientos, no distingo lo que es importante.Los objetos no deberían tocar, puesto que no viven."
65 • "Uno los usa, los pone en su sitio, vive entre ellos; son útiles, nada más.y a mí me tocan; es insoportable.Tengo miedo de entrar en contacto con ellos como si fueran animales vivos."
66 • "Ahora veo; recuerdo mejor lo que sentí el otro día, a la orilla del mar, cuando tenía el guijarro. Era una especie de repugnancia dulzona."
67 • "Qué desagradable era proceder del guijarro, estoy seguro pesaba del guijarro a mis manos. Si, es eso, es eso; una especie de náusea en las manos."
68 • "Tienes por la tarde.Lentos, perezosos, fastidiados, los hechos se acomodan en rigor al orden que yo quiero dárles; pero éste sigue siendo exterior a ellos."
69 • "Tengo la impresión de hacer un trabajo puramente imaginativo. Además, estoy seguro de que los personajes de una novela parecerían más verdaderos; en todo caso, serían más agradables."
70 • "Me ilumina por dentro una luz empobrecedora.Tal vez sea imposible comprender el propio rostro.O acaso es porque soy un hombre solo Los que viven en sociedad han aprendido a mirarse en los espejos, tal como los
71 • "Yo no tengo amigos; por eso es mi carne tan desnuda Si, es como la naturaleza sin los hombres.Las cinco y media.Otra vez la soledad, la Náusea."
72 • "Entonces me dio la Náusea: me dejó caer en el asiento, ni siquiera sabía dónde estaba; veía girar lentamente los colores a mi alrededor; tenía ganas de vomitar."
73 • "Y desde entonces la Náusea no me ha abandonado, me posee.La Náusea no está en mí; la siento allí en la pared, en los tirantes, en todas partes a mi alrededor."
74 • "Es una sola cosa con el café, soy yo quien está en ella.Sobre todo la manera brusca de arrojarse hacia adelante, como un acantilado contra el mar.Comienzo a calentarme, a sentirme feliz."
75 • "Todavía no es nada extraordinario, es una pequeña dicha de Náusea: se despliega en el fondo del charco viscoso, en el fondo de nuestro tiempo.La música horada esas formas vagas y las traspasa.Qué extraño, qué d
76 • "Nada puede interrumpirla y todo puede quebrantarla.Cuando la voz se elevó en silencio, sentí que mi cuerpo se endurecía; y la Náusea se desvaneció.Todos las masas blandas que se mueven espontáneamente.
77 • "La Náusea, se ha quedado allí, en la luz amarilla. Soy feliz; este frío es tan puro, tan pura la noche; no soy yo mismo una onda de aire helado? No tener ni sangre, ni linfa, ni carne."
78 • "Deslizarse por este largo canal hacia aquella palidez.Ser sólo frío.Yo no puedo recibir de estas soledades trágicas nada más que un poco de pureza vacía.Me ilumíné; comprendo el método del Autodidacto; se instr
79 • "Veo el porvenir. Está allí en la calle, apenas más palido que el presente. Qué necesidad tiene de realizarse qué ganará con ello.A veces acierto a pronunciar en mi relato esos hermosos nombres que se leen en lo
80 • "Provocan en mí imágenes nuevas, como las que concebí, según sus lecturas, las personas que nunca han viajado; sueño basándome en palabras, eso es todo.Señor, creo que la aventura puede definirse así: un aconte
81 • "Qué cosas alcanzaría si mi propia vida constituyera la materia de la melodía.Sábado, mediodía.Para que el suceso más trivial se convierta en aventura, es necesario y suficiente contarlo. Esto es lo que engaña a
82
83 frase = frase.replace(",","");
84 frase = frase.replace(".", "");
85 String frases[] = frase.split("\n");
86 System.out.println("Tamaño de frases: " + frases.length);
87
88 usuarioW = new PrintWriter(usuarioF = new FileWriter("C:/Usuario.txt"));
89 ingredienteW = new PrintWriter(ingredienteF = new FileWriter("C:/O2Ingrediente.txt"));
90 zonaW = new PrintWriter(zonaF = new FileWriter("C:/O3Zona.txt"));
91 restauranteW = new PrintWriter(restauranteF = new FileWriter("C:/O4Restaurante.txt"));
92 productoW = new PrintWriter(productoF = new FileWriter("C:/O8Producto.txt"));
93 prodOfW = new PrintWriter(prodOfF = new FileWriter("C:/O9ProdOf.txt"));
94 ingProvW = new PrintWriter(ingProvF = new FileWriter("C:/O10IngProv.txt"));
95 adminW = new PrintWriter(adminF = new FileWriter("C:/O11AdminRest.txt"));
96 menuW = new PrintWriter(menuF = new FileWriter("C:/O12Menu.txt"));
97 MENU_PERSONALIZADOW = new PrintWriter(MENU_PERSONALIZADOF = new FileWriter("C:/O13MENU_PERSONALIZADO.txt"));
98 PEDIDOW = new PrintWriter(PEDIDOF = new FileWriter("C:/O14PEDIDO.txt"));
99 MENU_PEDIDOW = new PrintWriter(MENU_PEDIDOF = new FileWriter("C:/O15MENU_PEDIDO.txt"));
100
101 //
102 ingPerW = new PrintWriter(ingPerF = new FileWriter("C:/O12IngPer.txt"));
103 prodPerW = new PrintWriter(prodPerF = new FileWriter("C:/O12ProdPer.txt"));
104
105 int idZona = 1;
106 int idRest = 1;
107 int idMenuPer = 1;
108 int idPedido = 1;
109
110 String restaurante = "IdRestaurante,Nombre,Tipo,Pagina_Web,Nombre_Representante,IdZona";
111 String usuario = "IdIDENTIFICACION,Nombre,Tipo";
112 String zona = "IdZona,Tipo,Estado,IdCapacidad,CondicionesTécnicas,IdDiscapacitados";
113 String ingrediente = "Nombre,Descripcion,IdDescripcion";
114 String producto = "IdProducto,Nombre,Descripcion,IdDescripcion,Tiempo_Preparacion,Costo_Produccion,Precio_Venta,Disponible,Clasificacion";
115 String prodOf = "IdProducto,IdRestaurante,cantidad,cantidadMax";
116 String ingProd = "Nombre_Ingrediente,Id_producto";
117 String adminRest = "Idusuario,IdRestaurante";
```

Usuario.javaRotondTM.javamain.java

```
123 String ingrediente = "Nombre,Descripcion,IDdescripcion";
124 String producto = "IdProducto,Nombre,Descripcion,IDdescripcion,Tiempo_Preparacion,Costo_Produccion,Precio_Venta,Disponible,Clasificacion";
125 String prodOf = "IdProducto,idRestaurante,cantidad,cantidadMax";
126 String ingProd = "Nombre_Ingrediente,Id_producto";
127 String adminRest = "Idusuario,idRestaurante";
128
129 String menu = "IdMenu,Medio_Pago,Precio,IdRestaurante,Entrada,Plato_Fuerte,Postre,Bebida,Acompanamiento";
130 String MENU_PERSONALIZADO = "IdMenuPer,Precio,Entrada,Plato_Fuerte,Postre,Bebida,Acompanamiento,IdMenu";
131 String PEDIDO = "IdPedido,Fecha,Valor_Total,Idusuario,Estado";
132 String MENU_PEDIDO = "IdPedido,IdMenuPer,cantidad";
133
134 // String IngPer = "Nombre_Ingrediente,Id_producto";
135 String ProdPer = "IdProductoPer,IdProducto";
136
137 restaurantw.println(restaurant);
138 usuariow.println(usuario);
139 zonaw.println(zona);
140 ingredientew.println(ingrediente);
141 productow.println(producto);
142 prodOfw.println(prodOf);
143 ingProdW.println(ingProd);
144 adminw.println(adminRest);
145
146 menuw.println(menu);
147 MENU_PERSONALIZADOW.println(MENU_PERSONALIZADO);
148 PEDIDOW.println(PEDIDO);
149 MENU_PEDIDOW.println(MENU_PEDIDO);
150
151 // ingPerW.println(IngPer);
152 prodPerW.println(ProdPer);
153
154 for (int i = 0; i < 200000; i++) {
155     if (i % 250 == 0) {
156
157         usuario = "1234" + (i + 1) + ",Persona " + i + ",AdminRestaurante";
158         usuariow.println(usuario);
159
160         restaurante = "" + idRest + ",Restaurante" + i + ",Tipo " + i % 301 + ",Rest" + idRest + "" + i
161             + ".RotondalLosAlpes.com,Persona " + i + ", " + idZona + "";
162         restaurantew.println(restaurante);
163         idRest++;
164
165         if (i % 500 == 0) {
166             if (i % 3 == 0) {
167                 zona = "" + idZona + "," + espacio[1] + ",15" + i / 1500 + "," + frases[i % 204] + ","
168                     + discapacitado[0] + "";
169                 zonaw.println(zona);
170                 idZona++;
171             } else {
172                 zona = "" + idZona + "," + espacio[0] + ",15" + i / 1500 + "," + frases[i % 204] + ","
173                     + discapacitado[1] + "";
174                 zonaw.println(zona);
175                 idZona++;
176             }
177         }
178         adminRest = "1234" + (i + 1) + ", " + idRest;
179         adminw.println(adminRest);
180
181     } else {
182
183         usuario = "1234" + (i + 1) + ",Persona " + i + ",Cliente";
```