

# “Iteración 2: – Desarrollo del Caso de Estudio de Alohandes”

Lina María Gómez Mesa, Eduardo José Herrera Alba  
Iteración 1 del Proyecto de Sistemas Transaccionales  
Universidad de los Andes, Bogotá, Colombia  
{l.gomez1, ej.herreraa}@uniandes.edu.co  
Fecha de presentación: Abril 11 de 2023

## Tabla de contenido

<b>1 Revisión del caso de estudio y documentación de resultados</b>	<b>2</b>
1.1 Roles de Usuario	2
1.2 Entidades de Negocio	2
1.3 Funcionalidades Principales	2
1.4 Reglas de Negocio	2
<b>2 Modelo Conceptual</b>	<b>2</b>
2.1 Diagrama de Clases UML	2
<b>3 Diseño de la Base de Datos</b>	<b>3</b>
3.1 Relaciones	4
3.2 Normalización del Modelo Relacional	7
<b>4 Resultados Logrados</b>	<b>8</b>
<b>5 Resultados No Logrados</b>	<b>8</b>
<b>6 Balance del Plan de Pruebas</b>	<b>8</b>
6.1 Pruebas de unicidad de tuplas	8
TipoMiembroComunidad	8
Cliente	10
Operador	11
TipoOperadorEmpresa	12
Empresa	14
TipoOperadorNatural	15
PersonaNatural	16
Contrato	18
Seguro	19
Horario	20
TipoAlojamiento	21
Alojamiento	22
Reserva	23
TipoServicioPublico	24
ServicioPublico	25
TipoServicio	26
Servicio	27
6.2 Pruebas de integridad con FK	28
Alojamiento	28
Reserva	29

6.3 FK Pruebas de integridad de acuerdo con restricciones de chequeo	31
TipoMiembroComunidad	31
Reserva	32
<b>7 Supuestos adicionales sobre las reglas de negocio encontradas en el caso de estudio</b>	<b>33</b>

## **1 Revisión del caso de estudio y documentación de resultados**

### **1.1 Roles de Usuario**

En el caso de Alohandes se identificaron los siguientes roles de usuario.

1. TipoMiembroComunidad: Incluye profesor, egresado, estudiante, empleado y padre de estudiante.
2. TipoOperadorNatural: Incluye vecino, personaNatural y miembroComunidad.
3. TipoEmpresa: Incluye hotel, hostel y viviendaUniversitaria.
4. Administrador Base de Datos

### **1.2 Entidades de Negocio**

Las entidades de negocio que se identificaron son aquellas que se incluyen en el modelo conceptual. El modelo conceptual se encuentra expuesto en la sección 2, Modelo conceptual.

### **1.3 Funcionalidades Principales**

Las funcionalidades principales del sistema se describen mediante los requerimientos funcionales en la siguiente sección, análisis y modelo conceptual, en la sección 1.

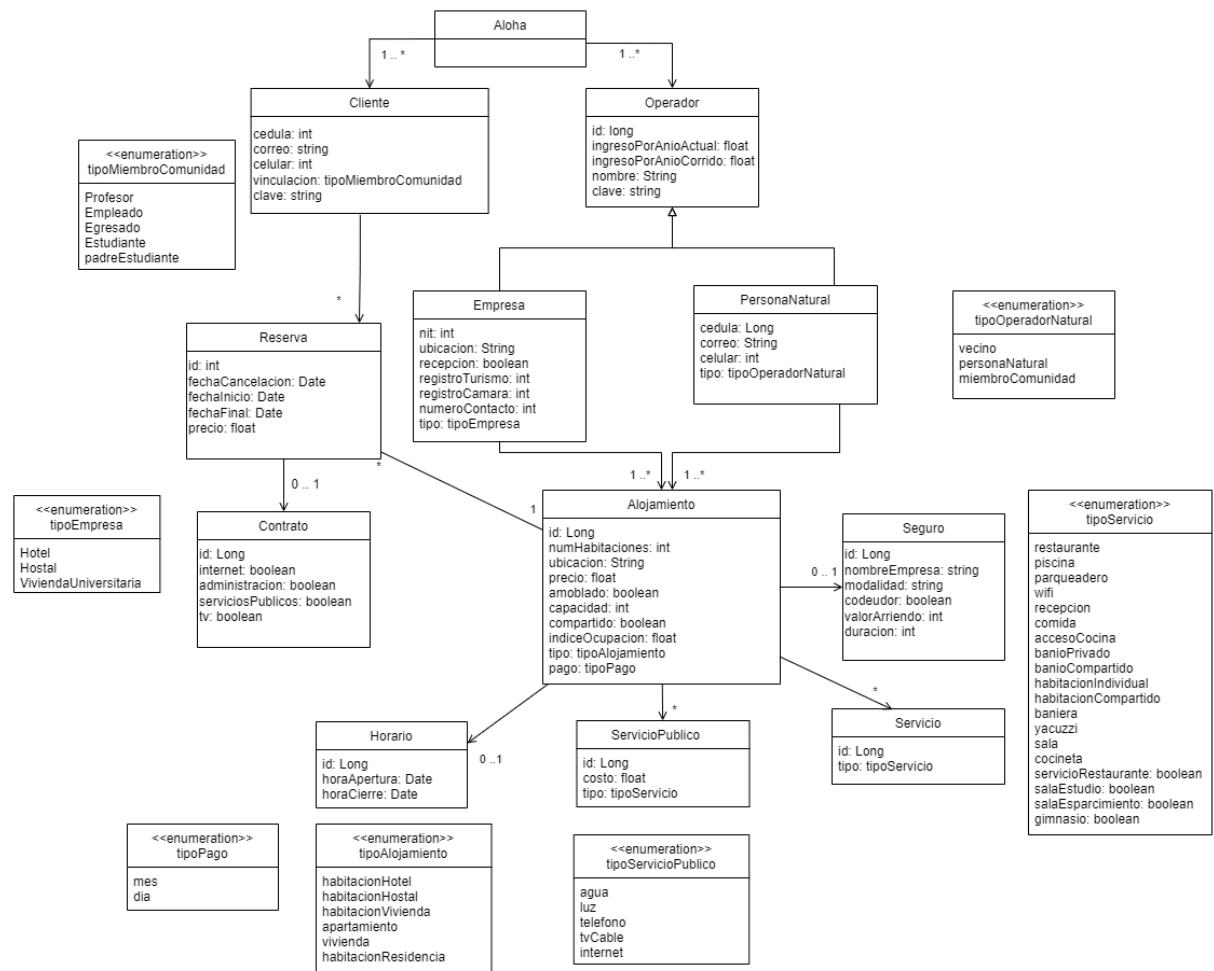
### **1.4 Reglas de Negocio**

Las reglas de negocio están incorporadas en el modelo conceptual que se realizó para la descripción de este, en la subsección 2: Modelo conceptual.

## **2 Modelo Conceptual**

### **2.1 Diagrama de Clases UML**

Se realizó la revisión del caso de estudio propuesto: Alohandes. A continuación, se muestra un diagrama de clases de acuerdo a lo identificado en el enunciado. El modelo conceptual tiene integradas las reglas de negocio mediante anotaciones, las clases propuestas que son elementos del mundo del negocio y las cardinalidades de las asociaciones.



**Figura 1.** Modelo conceptual completo de 12 clases para el sistema transaccional de AlohaAndes  
[Elaborado en Diagrams.Net]

Las enumeraciones que aparecen en el diagrama tienen valores que tiene el sistema inicialmente. Sin embargo, es posible agregar nuevos valores en el modelo relacional. Teniendo en cuenta el RNF2, todas las clases deberían ser persistentes, con excepción a Aloha, la cual es una clase de carácter conceptual.

Algunas restricciones de acuerdo al enunciado son:

1. Las empresas deben cumplir con el registro en la cámara de comercio y en la superintendencia de turismo.
2. Los clientes de Alohandes deben tener algún vínculo con la institución: estudiantes, egresados, empleados, profesores, padres de estudiantes, profesores invitados, personas registradas en eventos de Uniandes.
3. Una persona no puede reservar más de un alojamiento en un mismo día.
4. Un alojamiento no acepta reservas que superen su capacidad.
5. El alojamiento en vivienda universitaria sólo está habilitado a estudiantes, profesores, empleados y profesores visitantes.

### 3 Diseño de la Base de Datos

Se muestra a continuación el modelo de datos relacional que corresponde al modelo conceptual que se propuso. Las tablas de algunas relaciones son demasiado extensas para una

sola línea, por lo que se partieron para conservar la legibilidad. El modelo se compone de las siguientes relaciones:

### 3.1 Relaciones

**Tabla 1.** Relación de TipoMiembroComunidad

#### TipoMiembroComunidad

Id	Nombre
PK,SA	NN, CK[profesor,empleado,egresado,estudiante,padreEstudiante], ND

**Tabla 2.** Relación de Cliente

#### Cliente

Cedula	Correo	Nombre	Celular	Vinculacion	Clave
PK,UA	NN,UA	NN,UA	CK[=10], NN,UA, ND	FK[tipoMiembroCo munidad.Id],UA, NN	NN,UA

**Tabla 3.** Relación de Operador

#### Operador

Id	IngresoPorAnioActual	IngresoPorAnioCorrido	Nombre	Clave
PK, UA	CK[>0], NN, UA	CK[>0], NN,UA	NN,UA	NN,UA

**Tabla 4.** Relación de TipoOperadorEmpresa

#### TipoOperadorEmpresa

Id	Nombre
SA	NN,CK[Hotel, Hostal,ViviendaUniversitaria], UA, ND

**Tabla 5.** Relación de Empresa

#### Empresa

Nit	Ubicacion	Recepcion	RegistroTurismo	Registro Camara	NumeroContacto	Tipo
PK, FK[Operador.Id],UA	NN, UA	NN, CK[0,1],UA	NN, CK[=9],UA	NN, CK[=9],U A	NN,CK[=10] ,UA	FK[TipoOperadorEmpresa.Id], NN, UA

**Tabla 6.** Relación de TipoOperadorNatural

**TipoOperadorNatural**

Id	Nombre
PK,SA	CK[Vecino,persona Natural,miembroComunidad, ND, UA

**Tabla 7.** Relación de PersonaNatural

**PersonaNatural**

Cedula	Correo	Celular	Tipo
PK, FK[Operador.Id],UA	NN,UA	CK[=10], NN, UA	FK[TipoOperadorNatural.Id], UA

**Tabla 8.** Relación de Contrato

**Contrato**

Id	internet	administracion	serviciosPublicos	tv
PK, SA	CK[0,1], UA, NN	CK[0,1], UA, NN	CK[0,1], NN, UA	CK[0,1], UA,NN

**Tabla 9.** Relación de Seguro

**Seguro**

Id	NombreEmpresa	Modalidad	Codeudor	ValorArriendo	Duracion
PK, SA	NN, UA	NN, UA, CK[presencial, digital]	NN, CK[0,1]	NN	NN, CK[3,6,12]

**Tabla 10.** Relación de Horario

**Horario**

Id	HoraApertura	HoraCierre
PK,SA	NN, UA	NN, UA

**Tabla 11.** Relación de TipoAlojamiento

**TipoAlojamiento**

---

Id	Nombre	TipoPago
PK, SA	CK[habitacionHotel,habitacionHostal,habitacionVivienda,apartamento,vivienda,habitacionResidencia], UA, ND	CK[mes, dia]

**Tabla 12.** Relación de Alojamiento

#### Alojamiento

Id	NumHabitaciones	Ubicacion	Precio	Amoblado	Capacidad
PK	NN, CK[>0]	NN	NN, CK[>0]	NN, CK[0,1]	NN, CK[>0]

Compartido	IndiceOcupacion	Tipo	Horario	Seguro	Operador
NN, CK[0,1]	NN	FK[TipoAlojamiento.Id]	FK[Horario.Id]	FK[Contrato.Id]	FK[Seguro.Id]

**Tabla 13.** Relación de Reserva

#### Reserva

Id	fechaCancelacion	fechaInicio	fechaFinal	Alojamiento	Cliente	Contrato	Precio
PK, SA	UA	NN, UA, CK[fechaInicio < fechaFinal]	NN,UA	FK[Alojamiento.Id], UA	FK[Cliente.cedula], UA	FK[Contrato.Id], UA	NN, UA

**Tabla 14.** Relación de TipoServicioPublico

#### TipoServicio

Id	Nombre
PK,SA	CK[restaurante, piscina, ... (ver UML), gimnasio], UA

**Tabla 15.** Relación de ServicioPublico

#### ServicioPublico

---

Id	Costo	Tipo	Alojamiento
PK, SA	NN	FK[TipoServicio.Id]	FK[Alojamiento.Id]

**Tabla 16.** Relación de TipoServicio

#### TipoServicio

Id	Nombre
PK,SA	CK[restaurante, piscina, ... (ver UML), gimnasio], ND

**Tabla 17.** Relación de Servicio

#### Servicio

Id	Tipo	Alojamiento
PK, SA	FK[TipoServicio.Id]	FK[Alojamiento.Id]

### 3.2 Normalización del Modelo Relacional

El modelo relacional planteado se encuentra en el tercer nivel de normalización. A continuación, se justifica nivel por nivel:

**Primera forma normal:** Un modelo relacional de datos se encuentra en la primera forma normal si todos los atributos de cada una de las relaciones tienen dominios atómicos. El modelo planteado cumple con la primera forma normal dado que no hay atributos multivalor. Esto se puede observar en la sección 3.1 de este documento.

**Segunda Forma Normal:** Un modelo relacional de datos se encuentra en la segunda forma normal si para cada una de sus relaciones se cumple que no existen dependencias funcionales parciales desde algún atributo primo a un atributo no primo. De acuerdo a lo anterior, el modelo planteado sí cumple con la segunda forma normal porque se encuentra en primera forma normal y no existen dependencias parciales desde los atributos primos; puesto que, las llaves primarias no son compuestas y por lo tanto no hay subconjuntos propios de atributos a los cuales aplicar esta restricción.

**Tercera Forma Normal:** En tercer lugar, el modelo propuesto cumple con la tercera forma normal debido a que se encuentra en segunda forma normal y no existen dependencias transitivas entre atributos no primos. En el caso de este modelo, los conjuntos de llaves candidatas son de cardinalidad 1, solo tienen la llave primaria que, como se dijo antes, consta de solo un atributo, por lo general un número. Por ende, en este caso particular, todo atributo que no es la llave primaria es un atributo no primo.

**Forma normal de Boyce-Codd:** Un modelo relacional de datos se encuentra en la cuando no existen dependencias parciales entre atributos primos y además no hay varias llaves candidatas compuestas que no son disjuntas. En el caso de este modelo, el conjunto de llaves candidatas de cada relación consta de solo un atributo: el identificador respectivo (ya sea un id, número de identificación o NIT). Por ende, no hay dependencias parciales entre

atributos primos (pues en cada relación hay un único atributo primo, el ya mencionado identificador) y además no hay llaves candidatas compuestas del todo

#### **4 Resultados Logrados**

De forma general, se logró poblar las tablas, los requerimientos de RF1 a RF6 y los requerimientos de consulta RFC1-RFC4. Asimismo, se realizaron los escenarios de prueba en SQL Developer. Asimismo se cumplió con:

1. **Privacidad:** El sistema tiene incorporado la autenticación de usuario (ya sea cliente o empresa). Por lo que, se necesita que se haga una autenticación y especificación de la información para poder hacer los requerimientos funcionales. Esto se adicionó en Registrar una Reserva, cancelar una Reserva, adicionar un Alojamiento y Retirar una oferta de Alojamiento.

2. **Persistencia:** Dado que es una base de datos, es persistente el manejo de la información para todas las clases a excepción de Alohandes.

3. **Concurrencia:** Para este requerimiento no funcional se mencionaba que cada una de las solicitudes podían ser hechas de manera múltiple y simultánea. Esto es particularmente importante para el caso de las consultas de información y registros. La carencia de información duplicada garantiza la unicidad de la información. De esta manera, cada consulta o registro se maneja como si fuera secuencial, siguiendo el concepto de transaccionalidad *Isolation*. Además, es posible que dos usuarios distintos hagan uso de la aplicación en máquinas diferentes accediendo a la misma base de datos, y aun así podrán realizar sus operaciones de manera simultánea.

4. **Distribución:** Para la garantía de la información en la base de datos de manera centralizada, se maneja el concepto de Alohandes, que tiene conexión directa o indirecta con todas las entidades del modelo, y es a partir de donde se inicia el análisis para cada uno de los requerimientos funcionales una vez se ejecuta la autenticación de Usuario (empresa/cliente). Además, está centralizada haciendo uso del SMBD de Oracle SQL Developer.

#### **5 Resultados No Logrados**

En relación con esta iteración, se pudo haber mejorado un poco más la separación de los distintos actores y sus requerimientos funcionales. Esto se debe a que aunque se pide el usuario y contraseña antes de ejecutar la mayoría de requerimientos, aún aparece en la interfaz todos los requerimientos a la vista de otros usuarios. Por lo que los pueden visualizar aunque no ejecutar. Por ello, hubiera sido ideal mostrar únicamente los requerimientos funcionales de acuerdo al usuario que ingresa y los permisos de acceso que tiene.

#### **6 Balance del Plan de Pruebas**

A continuación se muestra el plan de pruebas. Este se dividió en tres secciones: 1) Pruebas de unicidad que se realizaron con todas las tablas, 2) pruebas de integridad con FK para las dos tablas más robustas ( Reserva y Alojamiento) y 3) Pruebas de integridad de acuerdo con restricciones de chequeo para las tablas de TipoMiembroComunidad y Reserva.



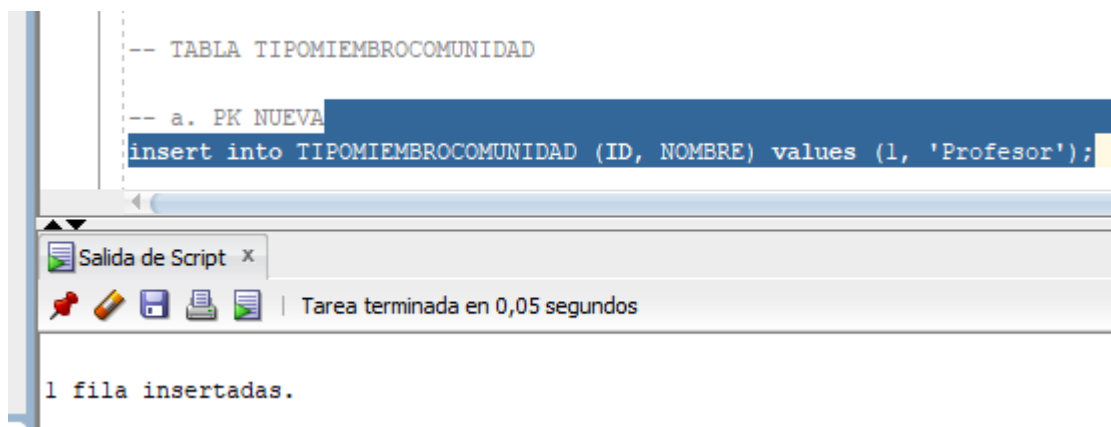
## 6.1 Pruebas de unicidad de tuplas

### TipoMiembroComunidad

```
-- TABLA TIPOMIEMBROCOMUNIDAD
-- a. PK NUEVA
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (1, 'Profesor');
-- b. PK CONOCIDA
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (1, 'Empleado');
-- c. COMPROBACIÓN
SELECT * FROM TIPOMIEMBROCOMUNIDAD;
```

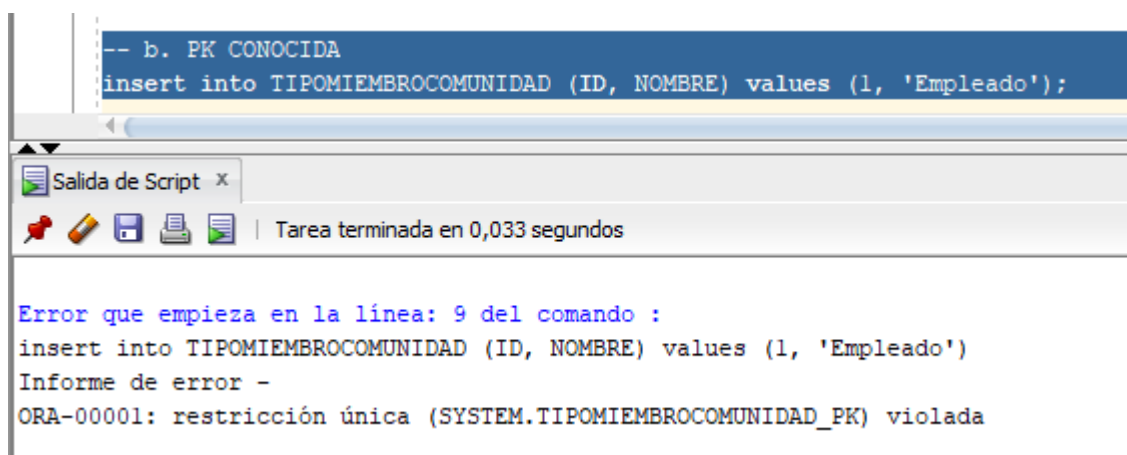
**Figura 2.** Pruebas de unicidad de tuplas para el TipoMiembroDeComunidad AlohaAndes [Elaborado en SQLDeveloper]

#### a. Llave Primaria Nueva



**Figura 3.** Pruebas de unicidad de tuplas punto a) para el TipoMiembroDeComunidad AlohaAndes [Elaborado en SQLDeveloper]

#### b. Llave Primaria Conocida



**Figura 4.** Pruebas de unicidad de tuplas punto b) para el TipoMiembroDeComunidad AlohaAndes [Elaborado en SQLDeveloper]

#### c. Comprobación

The screenshot shows the SQL Developer interface. The top pane contains the SQL query: `-- c. COMPROBACIÓN`  
`SELECT * FROM TIPOMIEMBROCOMUNIDAD;`

The bottom pane shows the execution result with two tabs: 'Salida de Script' and 'Resultado de la Consulta'. The 'Resultado de la Consulta' tab is active, displaying the following data:

ID	NOMBRE
1	1 Profesor

Below the table, it indicates 'Todas las Filas Recuperadas: 1 en 0,044 segundos'.

**Figura 5.** Pruebas de unicidad de tuplas punto c) para el TipoMiembroDeComunidad AlohaAndes [Elaborado en SQLDeveloper]

## Cliente

```
-- TABLA CLIENTE

-- a. PK NUEVA
insert into CLIENTE (CEDULA, CORREO, NOMBRE, CELULAR, VINCULACION, CLAVE) values
-- b. PK CONOCIDA
insert into CLIENTE (CEDULA, CORREO, NOMBRE, CELULAR, VINCULACION, CLAVE) values
-- c. COMPROBACIÓN
SELECT * FROM CLIENTE;
```

**Figura 6.** Pruebas de unicidad de tuplas para el Cliente [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva

The screenshot shows the SQL Developer interface. The top pane contains the SQL script:

```
-- TABLA CLIENTE

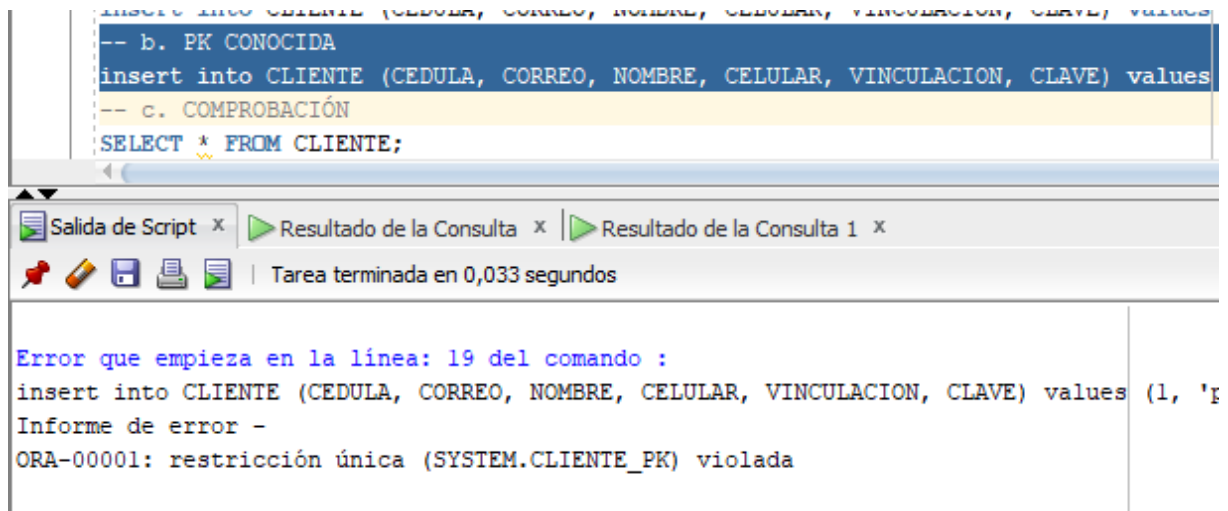
-- a. PK NUEVA
insert into CLIENTE (CEDULA, CORREO, NOMBRE, CELULAR, VINCULACION, CLAVE) values
-- b. PK CONOCIDA
insert into CLIENTE (CEDULA, CORREO, NOMBRE, CELULAR, VINCULACION, CLAVE) values
-- c. COMPROBACIÓN
SELECT * FROM CLIENTE;
```

The bottom pane shows the execution result with three tabs: 'Salida de Script', 'Resultado de la Consulta', and 'Resultado de la Consulta 1'. The 'Resultado de la Consulta 1' tab is active, displaying the message: '1 fila insertadas.'

Below the message, it indicates 'Tarea terminada en 0,019 segundos'.

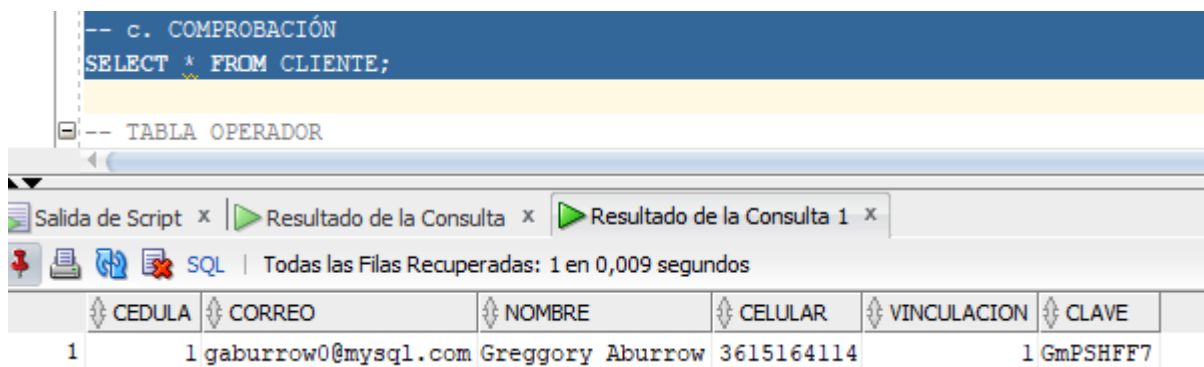
**Figura 7.** Pruebas de unicidad de tuplas punto a) para el Cliente [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida



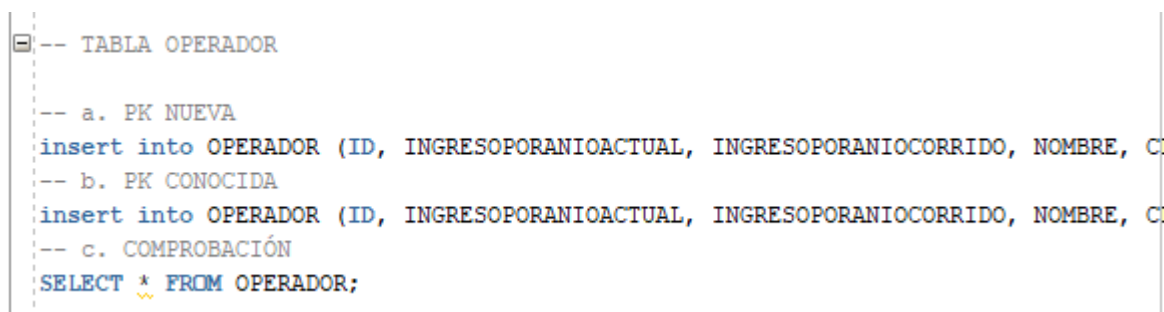
**Figura 8.** Pruebas de unicidad de tuplas punto b) para el Cliente [Elaborado en SQLDeveloper]

### c. Comprobación



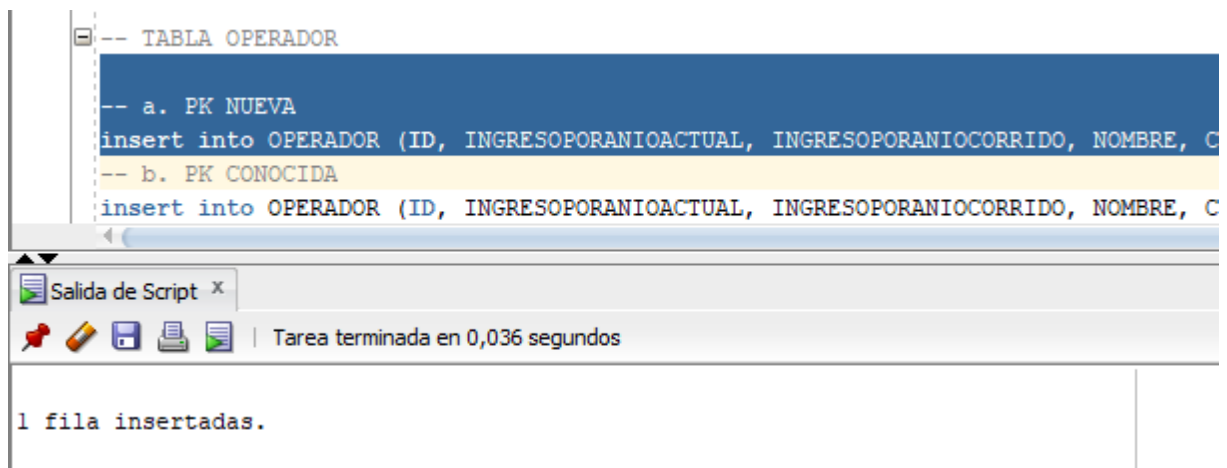
**Figura 9.** Pruebas de unicidad de tuplas punto c) para el Cliente [Elaborado en SQLDeveloper]

## Operador



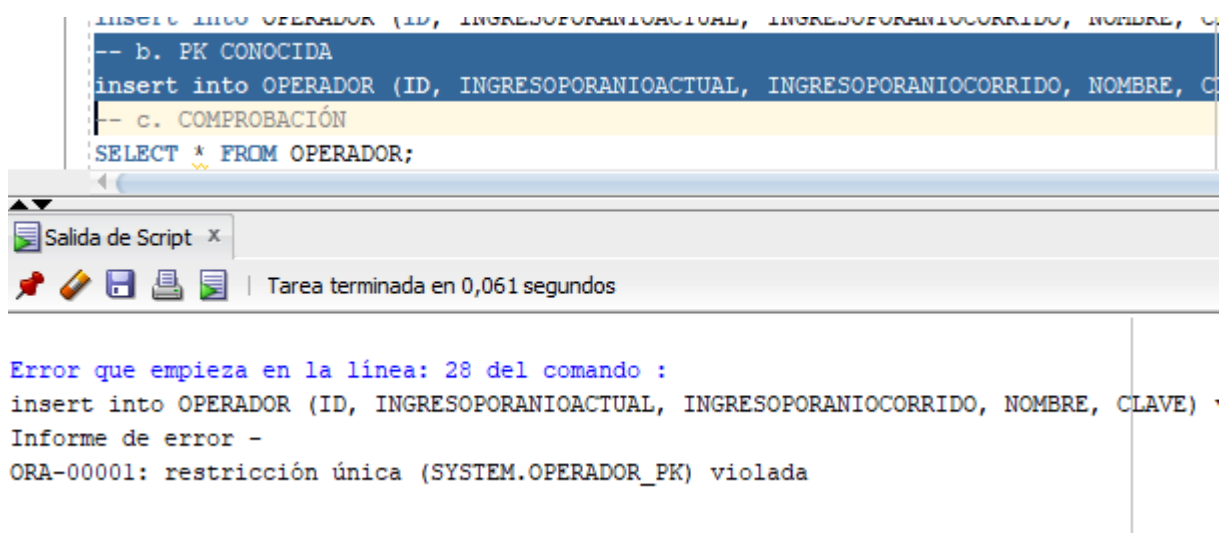
**Figura 10.** Pruebas de unicidad de tuplas para el Operador [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva



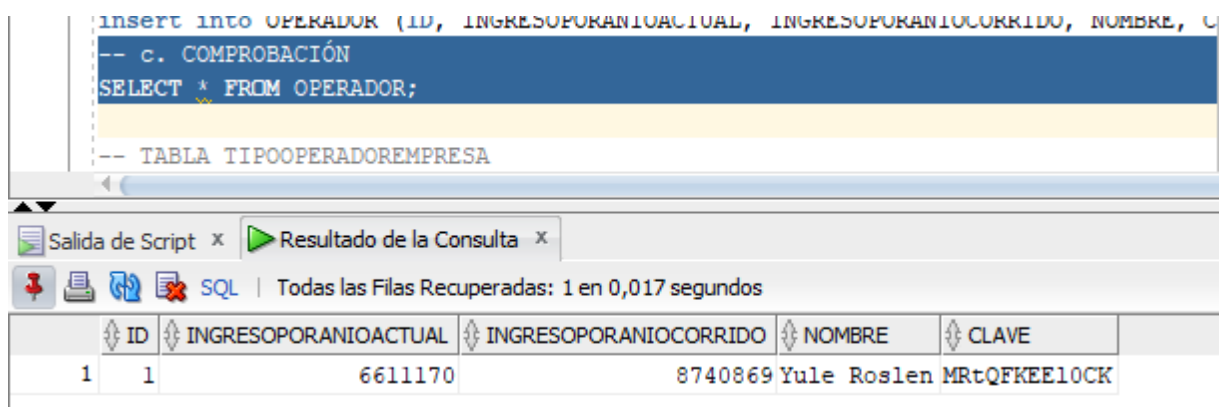
**Figura 11.** Pruebas de unicidad de tuplas para el Operador punto a [Elaborado en SQLDeveloper]

#### b. Llave Primaria Conocida



**Figura 12.** Pruebas de unicidad de tuplas para el Operador punto b [Elaborado en SQLDeveloper]

#### c. Comprobación



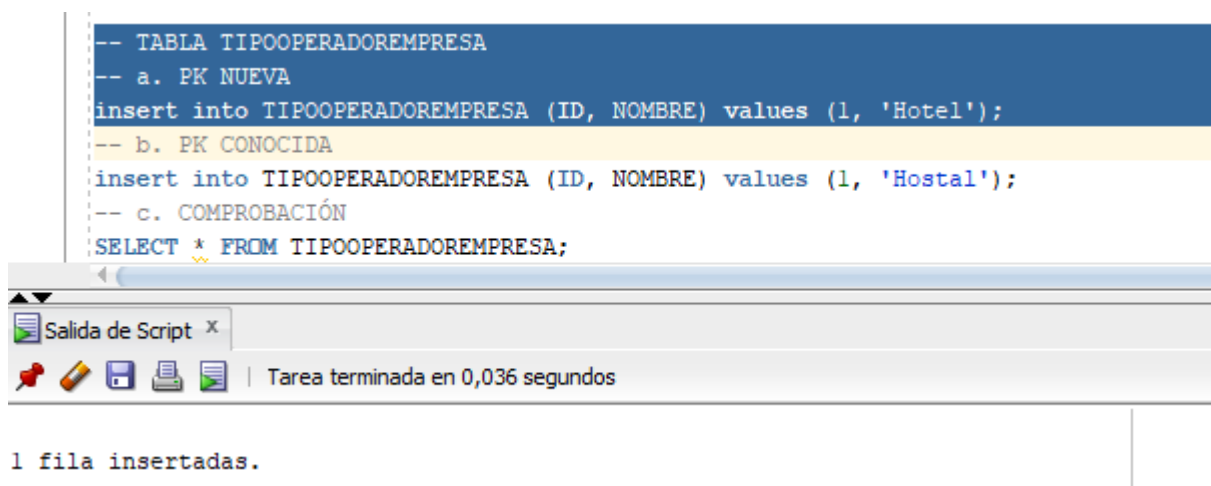
**Figura 13.** Pruebas de unicidad de tuplas para el Operador punto c [Elaborado en SQLDeveloper]

## TipoOperadorEmpresa

```
-- TABLA TIPOOPERADOREMPRESA
-- a. PK NUEVA
insert into TIPOOPERADOREMPRESA (ID, NOMBRE) values (1, 'Hotel');
-- b. PK CONOCIDA
insert into TIPOOPERADOREMPRESA (ID, NOMBRE) values (1, 'Hostal');
-- c. COMPROBACIÓN
SELECT * FROM TIPOOPERADOREMPRESA;
```

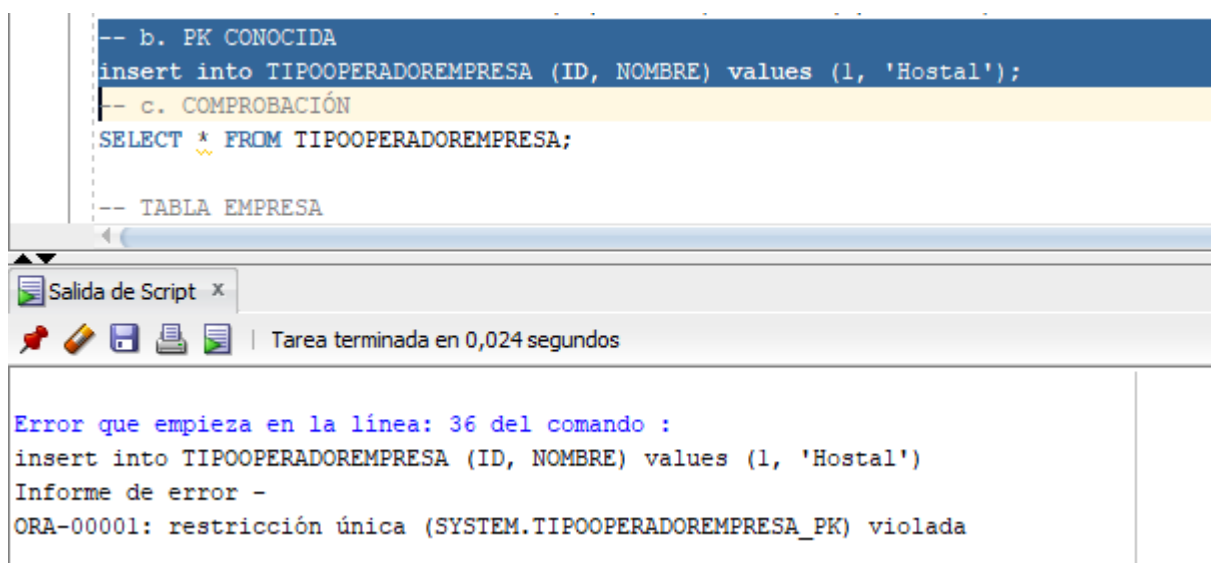
**Figura 14.** Pruebas de unicidad de tuplas para el TipoOperadorEmpresa [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva



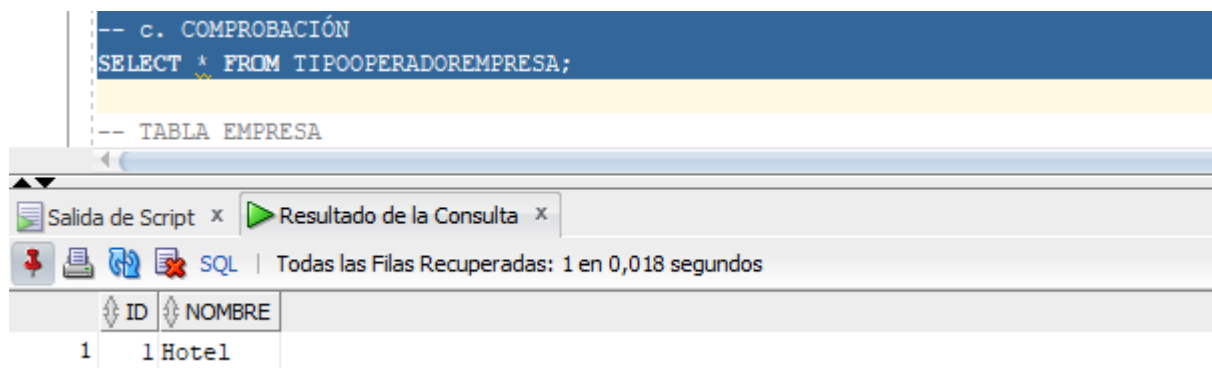
**Figura 15.** Pruebas de unicidad de tuplas para el TipoOperadorEmpresa punto a [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida



**Figura 16.** Pruebas de unicidad de tuplas para el TipoOperadorEmpresa punto b [Elaborado en SQLDeveloper]

### c. Comprobación



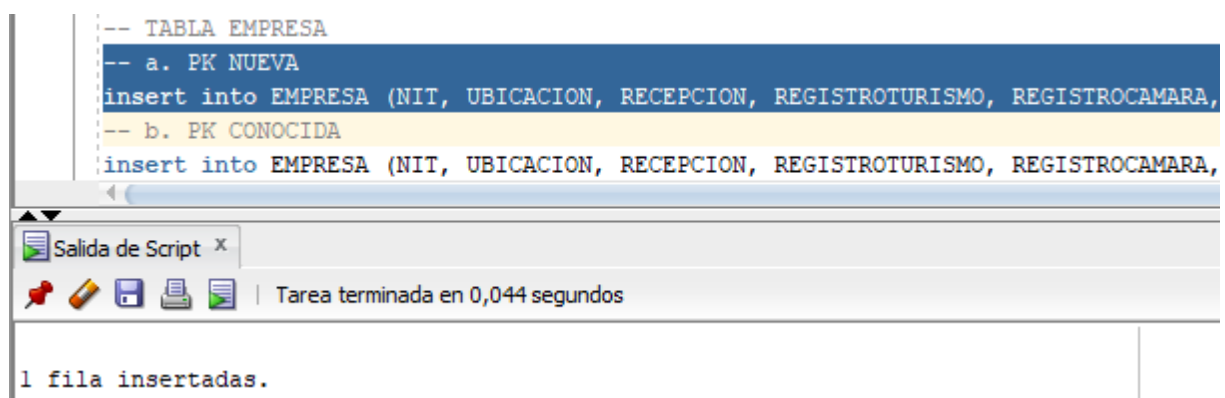
**Figura 17.** Pruebas de unicidad de tuplas para el TipoOperadorEmpresa punto c [Elaborado en SQLDeveloper]

### Empresa

```
-- TABLA EMPRESA  
-- a. PK NUEVA  
insert into EMPRESA (NIT, UBICACION, RECEPCION, REGISTROTURISMO, REGISTROCAMARA,  
-- b. PK CONOCIDA  
insert into EMPRESA (NIT, UBICACION, RECEPCION, REGISTROTURISMO, REGISTROCAMARA,  
-- c. COMPROBACIÓN  
SELECT * FROM EMPRESA;
```

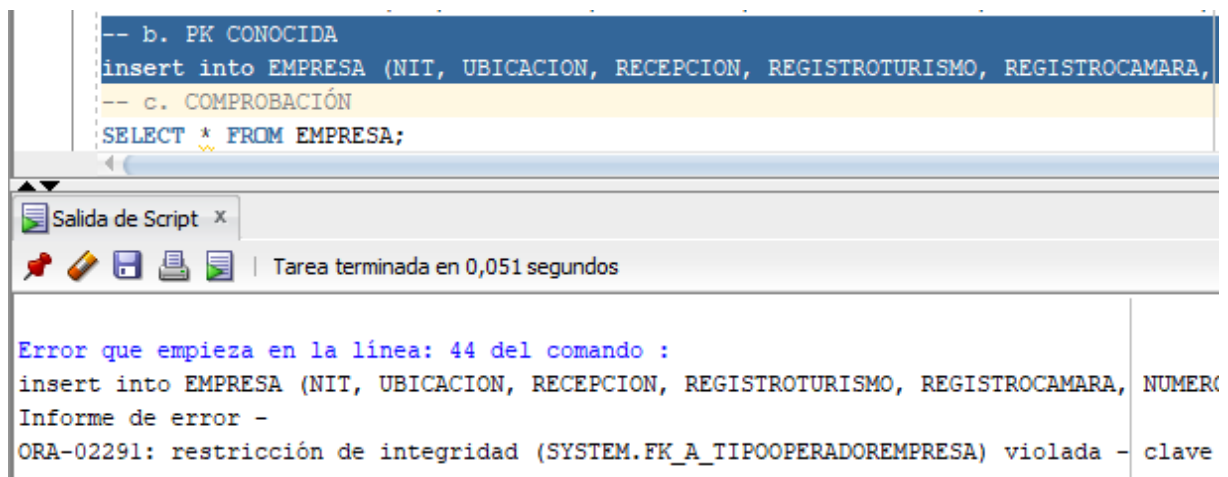
**Figura 18.** Pruebas de unicidad de tuplas para la Empresa [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva



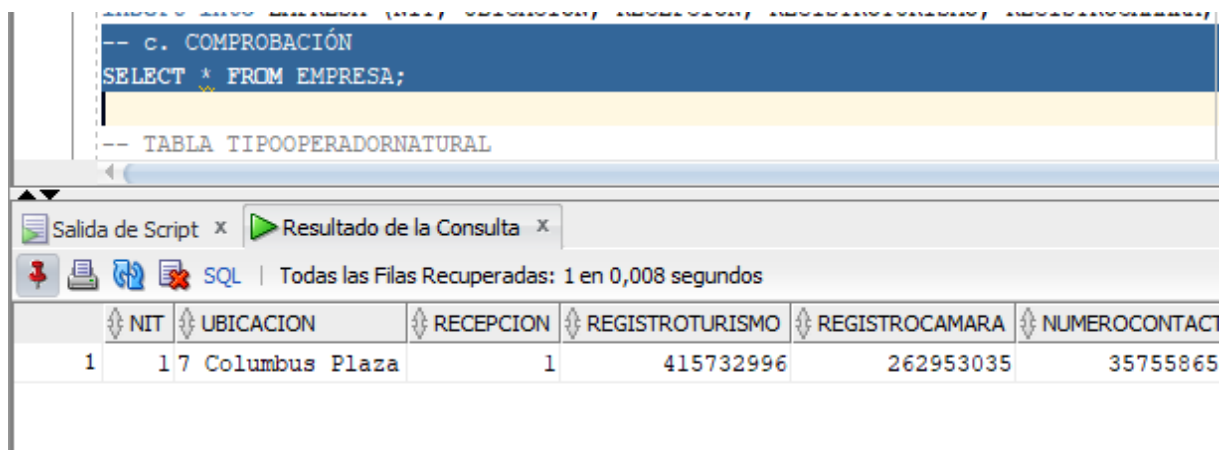
**Figura 19.** Pruebas de unicidad de tuplas para la Empresa punto a [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida



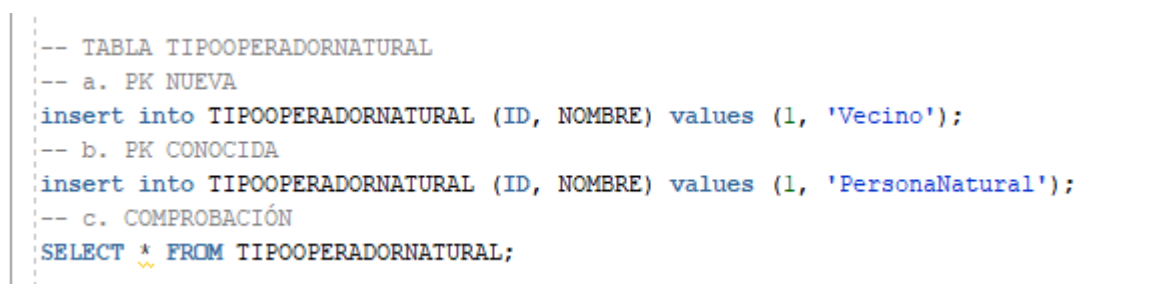
**Figura 20.** Pruebas de unicidad de tuplas para la Empresa punto b [Elaborado en SQLDeveloper]

### c. Comprobación



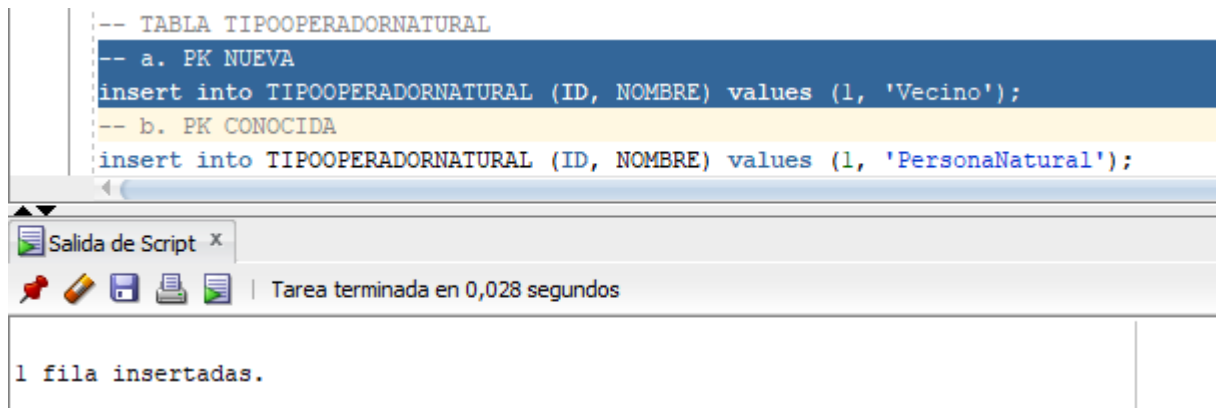
**Figura 21.** Pruebas de unicidad de tuplas para la Empresa punto c [Elaborado en SQLDeveloper]

## TipoOperadorNatural



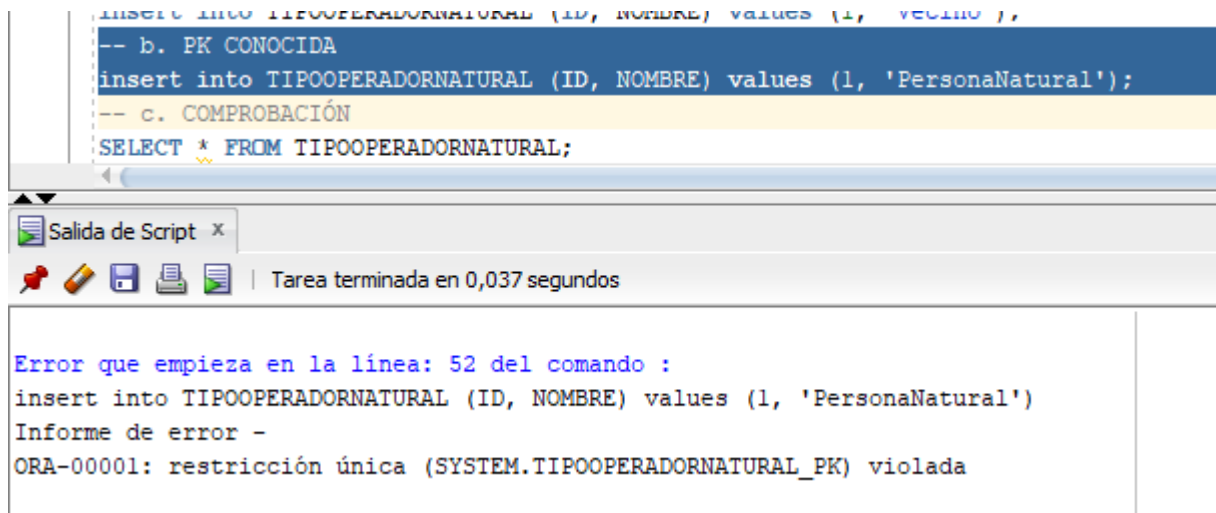
**Figura 22.** Pruebas de unicidad de tuplas para la TipoOperadorNatural [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva



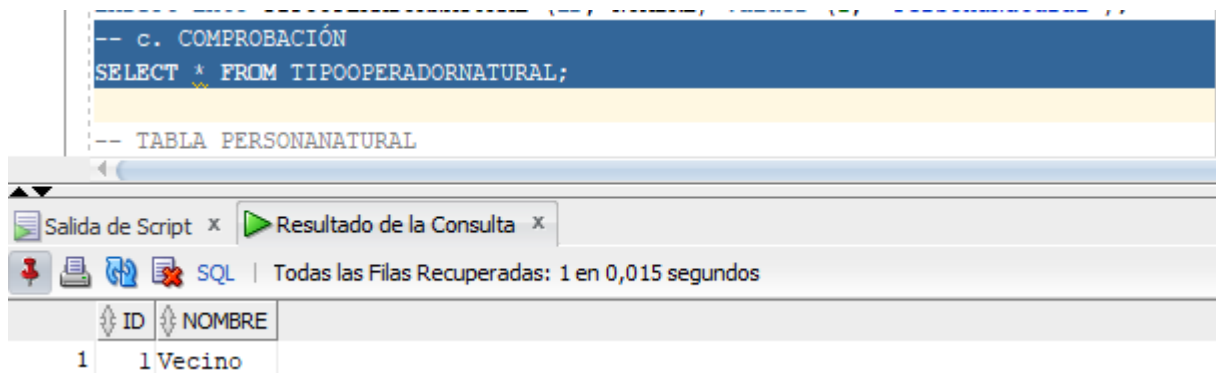
**Figura 23.** Pruebas de unicidad de tuplas para la TipoOperadorNatural punto a [Elaborado en SQLDeveloper]

#### b. Llave Primaria Conocida



**Figura 24.** Pruebas de unicidad de tuplas para la TipoOperadorNatural punto b [Elaborado en SQLDeveloper]

#### c. Comprobación



**Figura 25.** Pruebas de unicidad de tuplas para la TipoOperadorNatural punto c [Elaborado en SQLDeveloper]

PersonaNatural



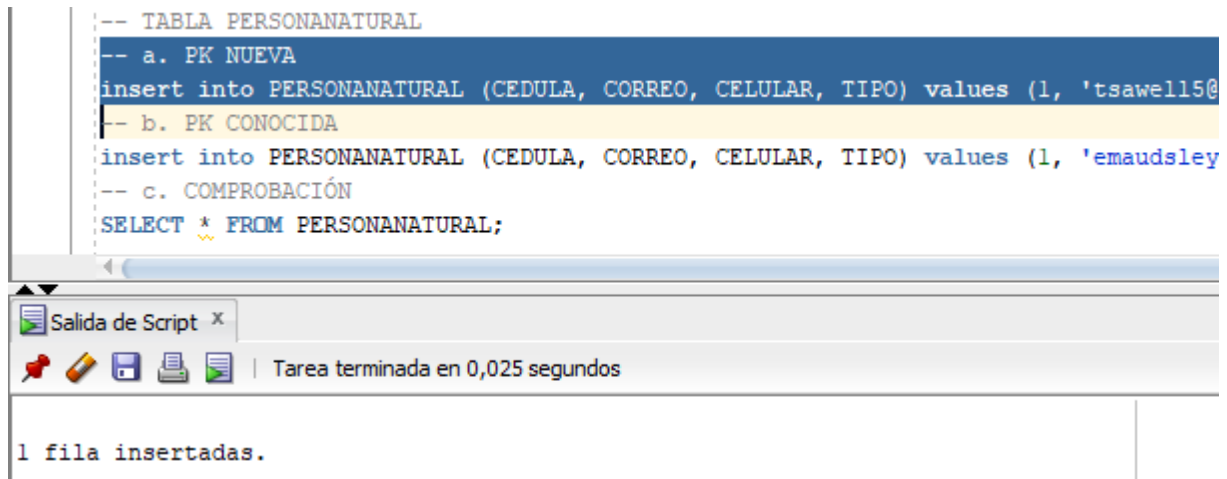
```

-- TABLA PERSONANATURAL
-- a. PK NUEVA
insert into PERSONANATURAL (CEDULA, CORREO, CELULAR, TIPO) values (1, 'tsawell15@lycos.com', 3774680631, 1);
-- b. PK CONOCIDA
insert into PERSONANATURAL (CEDULA, CORREO, CELULAR, TIPO) values (1, 'emaudsley6@privacy.gov.au', 3296144508, 2);
-- c. COMPROBACIÓN
SELECT * FROM PERSONANATURAL;

```

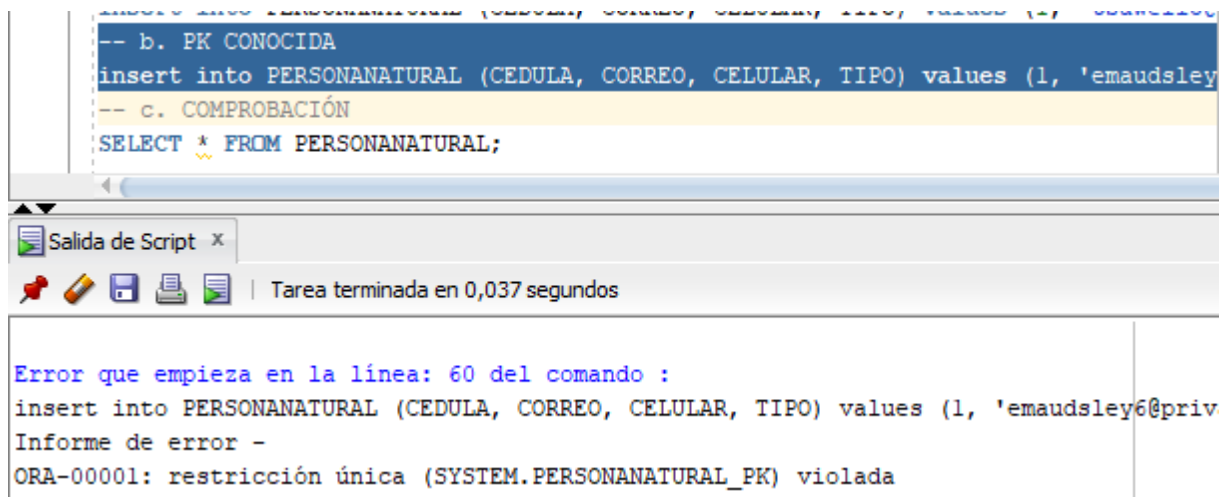
**Figura 26.** Pruebas de unicidad de tuplas para la PersonaNatural [Elaborado en SQLDeveloper]

#### a. Llave Primaria Nueva



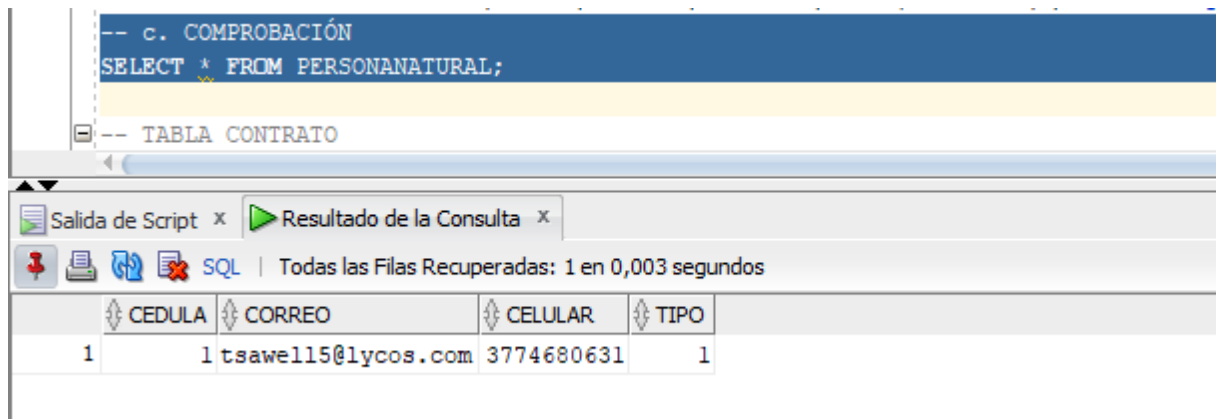
**Figura 27.** Pruebas de unicidad de tuplas para la PersonaNatural punto a [Elaborado en SQLDeveloper]

#### b. Llave Primaria Conocida



**Figura 28.** Pruebas de unicidad de tuplas para la PersonaNatural punto b [Elaborado en SQLDeveloper]

#### c. Comprobación



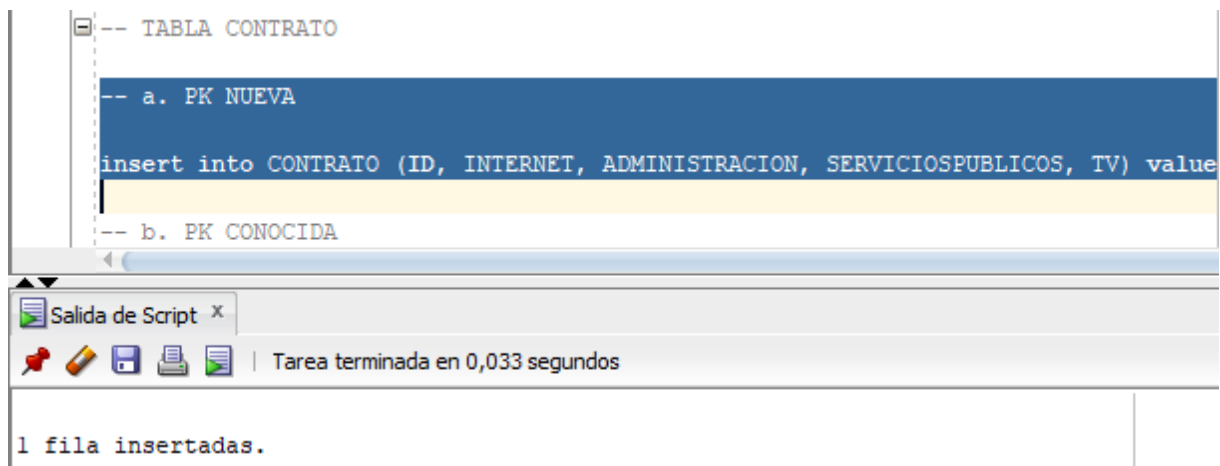
**Figura 29.** Pruebas de unicidad de tuplas para la PersonaNatural punto c [Elaborado en SQLDeveloper]

## Contrato

```
-- TABLA CONTRATO
-- a. PK NUEVA
insert into CONTRATO (ID, INTERNET, ADMINISTRACION, SERVICIOSPUBLICOS, TV) values (1, 1, 1, 1, 0);
-- b. PK CONOCIDA
insert into CONTRATO (ID, INTERNET, ADMINISTRACION, SERVICIOSPUBLICOS, TV) values (1, 0, 1, 0, 0);
-- c. COMPROBACIÓN
SELECT * FROM CONTRATO;
```

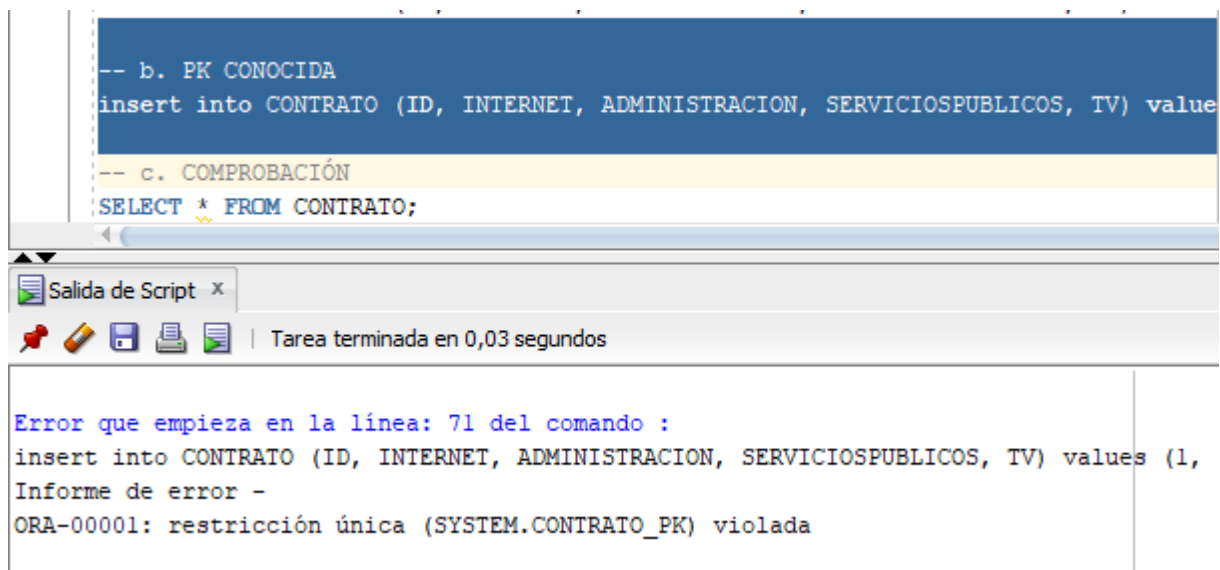
**Figura 30.** Pruebas de unicidad de tuplas para el Contrato [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva



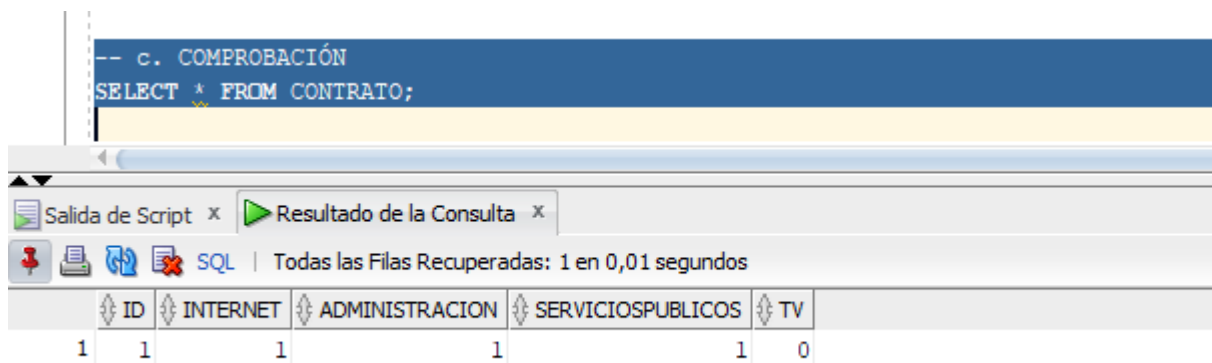
**Figura 31.** Pruebas de unicidad de tuplas para el Contrato punto a [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida



**Figura 32.** Pruebas de unicidad de tuplas para el Contrato punto b [Elaborado en SQLDeveloper]

### c. Comprobación



**Figura 33.** Pruebas de unicidad de tuplas para el Contrato punto c [Elaborado en SQLDeveloper]

## Seguro

```
-- 9. Seguro

-- a.
insert into SEGURO (ID, NOMBREEMPRESA, MODALIDAD,
CODEUDOR, VALORARRIENDO, DURACION)
values (49, 'Chatterbridge', 'digital', 0, 9202664.0, 6);

-- b.
insert into SEGURO (ID, NOMBREEMPRESA, MODALIDAD,
CODEUDOR, VALORARRIENDO, DURACION)
values (49, 'Feedmix', 'presencial', 0, 1410842.0, 12);

-- c.
SELECT * FROM SEGURO;
```

**Figura 34.** Pruebas de unicidad de tuplas para el Seguro[Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva

1 fila insertadas.

**Figura 35.** Pruebas de unicidad de tuplas para el Seguro punto a [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida

Error que empieza en la línea: 695 del comando :  
insert into SEGURO (ID, NOMBREEMPRESA, MODALIDAD,  
CODEUDOR, VALORARRIENDO, DURACION)  
values (49, 'Feedmix', 'presencial', 0, 1410842.0, 12)  
Informe de error -  
ORA-00001: restricción única (SYSTEM.SEGURO\_PK) violada

**Figura 36.** Pruebas de unicidad de tuplas para el Seguro punto b [Elaborado en SQLDeveloper]

### c. Comprobación

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x						
SQL   Todas las Filas Recuperadas: 1 en 0,003 segundos						
ID	NOMBREEMPRESA	MODALIDAD	CODEUDOR	VALORARRIENDO	DURACION	
1	49 Chatterbridge	digital	0	9202664	6	

**Figura 37.** Pruebas de unicidad de tuplas para el Seguro punto c [Elaborado en SQLDeveloper]

## Horario

```
-- 10. Horario

-- a.
insert into HORARIO (ID, HORAAPERTURA, HORACIERRE)
values (49, to_date('2022-11-13 18:09:53', 'yyyy-mm-dd hh24-mi-ss'),
to_date('2022-11-14 0:10:47', 'yyyy-mm-dd hh24-mi-ss'));

-- b.
insert into HORARIO (ID, HORAAPERTURA, HORACIERRE)
values (49, to_date('2022-03-31 21:42:03', 'yyyy-mm-dd hh24-mi-ss'),
to_date('2022-04-27 22:40:21', 'yyyy-mm-dd hh24-mi-ss'));

--c
SELECT * FROM HORARIO;
```

**Figura 38.** Pruebas de unicidad de tuplas para el Horario [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva

1 fila insertadas.

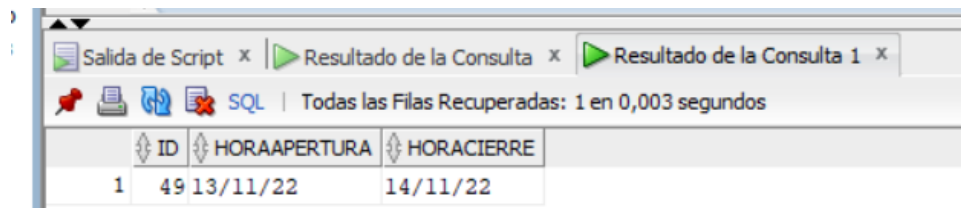
**Figura 39.** Pruebas de unicidad de tuplas para el Horario punto a [Elaborado en SQLDeveloper]

### b. Llave Primaria Conocida

```
Error que empieza en la línea: 711 del comando :
insert into HORARIO (ID, HORAAPERTURA, HORACIERRE)
values (49, to_date('2022-03-31 21:42:03', 'yyyy-mm-dd hh24-mi-ss'),
to_date('2022-04-27 22:40:21', 'yyyy-mm-dd hh24-mi-ss'))
Informe de error -
ORA-00001: restricción única (SYSTEM.HORARIO_PK) violada
```

**Figura 40.** Pruebas de unicidad de tuplas para el Horario punto b [Elaborado en SQLDeveloper]

### c. Comprobación



ID	HORAAPERTURA	HORACIERRE
49	13/11/22	14/11/22

**Figura 41.** Pruebas de unicidad de tuplas para el Horario punto c [Elaborado en SQLDeveloper]

## TipoAlojamiento

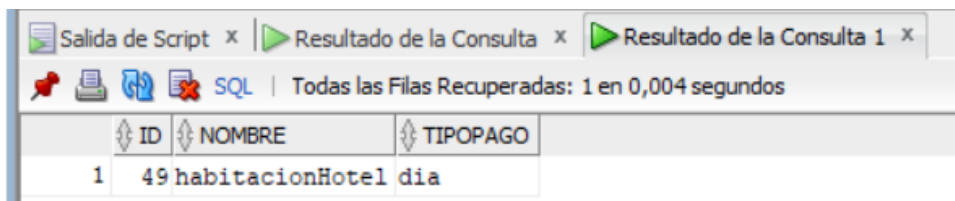
### a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 42.** Pruebas de unicidad de tuplas para el TipoAlojamiento punto a [Elaborado en SQLDeveloper]  
b. Llave Primaria Conocida

```
Error que empieza en la línea: 726 del comando :
insert into TIPOALOJAMIENTO (ID, NOMBRE, TIPOPAGO)
values (49, 'habitacionHostal', 'mes')
Informe de error -
ORA-00001: restricción única (SYSTEM.TIPOALOJAMIENTO_PK) violada
```

**Figura 43.** Pruebas de unicidad de tuplas para el TipoAlojamiento punto b [Elaborado en SQLDeveloper]  
c. Comprobación



The screenshot shows the SQL Developer interface with a query result window. The window title is 'Resultado de la Consulta 1'. The status bar indicates 'Todas las Filas Recuperadas: 1 en 0,004 segundos'. The query result is displayed in a table with three columns: ID, NOMBRE, and TIPOPAGO. The first row contains the values 1, 49habitacionHotel, and dia.

ID	NOMBRE	TIPOPAGO
1	49habitacionHotel	dia

**Figura 44.** Pruebas de unicidad de tuplas para el TipoAlojamiento punto c [Elaborado en SQLDeveloper]  
Alojamiento

```
-- 12. Alojamiento

-- a.
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (49, 18, '71074 Lyons Parkway', 1329228.6, 1, 44, 1, 97.6, 49, null, 49, 1);

-- b.
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (49, 5, '18 Dunning Drive', 1646894.0, 0, 48, 0, 94.7, 49, null, 49, 1);

-- c.
SELECT * FROM ALOJAMIENTO;
```

**Figura 45.** Pruebas de unicidad de tuplas para el Alojamiento [Elaborado en SQLDeveloper]  
a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 46.** Pruebas de unicidad de tuplas para el Alojamiento punto a [Elaborado en SQLDeveloper]  
b. Llave Primaria Conocida

```

Error que empieza en la línea: 741 del comando :
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (49, 5, '18 Dunning Drive', 1646894.0, 0, 48, 0, 94.7, 49, null, 49,1)
Informe de error -
ORA-00001: restricción única (SYSTEM.ALOJAMIENTO_PK) violada

```

**Figura 47.** Pruebas de unicidad de tuplas para el Alojamiento punto b [Elaborado en SQLDeveloper]  
c. Comprobación

ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR
1	49	18 71074 Lyons Parkway	1329228,6	1	44	1	97,6	49	(null)	49	1

**Figura 48.** Pruebas de unicidad de tuplas para el Alojamiento punto c [Elaborado en SQLDeveloper]

## Reserva

```

-- 13. Reserva

-- a.
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO,
FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO, PRECIO)
values (49, to_date('2022-06-03', 'yyyy-mm-dd'),
to_date('2022-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'),
49, 4, 4, 3365758.4);

-- b.
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO,
FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO, PRECIO)
values (49, null, to_date('2021-04-22', 'yyyy-mm-dd'),
to_date('2023-02-19', 'yyyy-mm-dd'), 49, 4, 4, 3052932.6);

-- c.
SELECT * FROM RESERVA;

```

**Figura 49.** Pruebas de unicidad de tuplas para la Reserva [Elaborado en SQLDeveloper]  
a. Llave Primaria Nueva

```

1 fila insertadas.

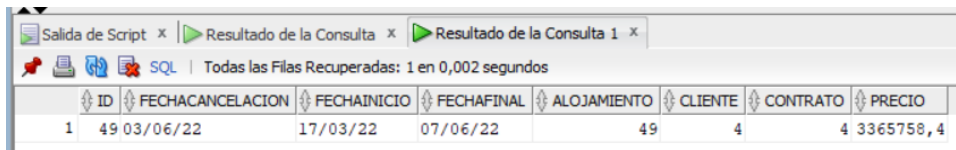
```

**Figura 50.** Pruebas de unicidad de tuplas para la Reserva punto a [Elaborado en SQLDeveloper]  
b. Llave Primaria Conocida

```
Error que empieza en la línea: 762 del comando :
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO,
FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO, PRECIO)
values (49, null, to_date('2021-04-22', 'yyyy-mm-dd'),
to_date('2023-02-19', 'yyyy-mm-dd'), 49, 4, 4, 3052932.6)
Informe de error -
ORA-00001: restricción única (SYSTEM.RESERVA_PK) violada
```

**Figura 51.** Pruebas de unicidad de tuplas para la Reserva punto b [Elaborado en SQLDeveloper]

### c. Comprobación



ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO
1	49	03/06/22	17/03/22	07/06/22	49	4	3365758,4

**Figura 52.** Pruebas de unicidad de tuplas para la Reserva punto c [Elaborado en SQLDeveloper]

## TipoServicioPublico

```
-- 14. TipoServicioPublico

-- a.
insert into TIPOSERVICIOPUBLICO (ID, NOMBRE) values (1, 'agua');

-- b.
insert into TIPOSERVICIOPUBLICO (ID, NOMBRE) values (1, 'luz');

-- c.
SELECT * FROM TIPOSERVICIOPUBLICO;
```

**Figura 53.** Pruebas de unicidad de tuplas para el TipoServicioPublico [Elaborado en SQLDeveloper]

### a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 54.** Pruebas de unicidad de tuplas para el TipoServicioPublico punto a [Elaborado en SQLDeveloper]

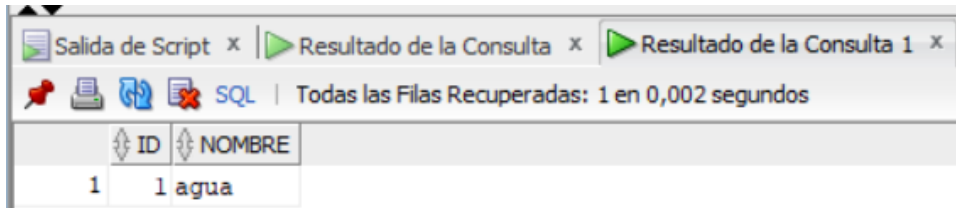
### b. Llave Primaria Conocida



```
Error que empieza en la línea: 781 del comando :
insert into TIPOSERVICIOPUBLICO (ID, NOMBRE) values (1, 'luz')
Informe de error -
ORA-00001: restricción única (SYSTEM.TIPOSERVICIOPUBLICO_PK) violada
```

**Figura 54.** Pruebas de unicidad de tuplas para el TipoServicioPublico punto b [Elaborado en SQLDeveloper]

c. Comprobación



ID	NOMBRE
1	agua

**Figura 55.** Pruebas de unicidad de tuplas para el TipoServicioPublico punto c [Elaborado en SQLDeveloper]

## ServicioPublico

```
-- 15. ServicioPublico

-- a.
insert into SERVICIOPUBLICO (ID, COSTO, TIPO, ALOJAMIENTO)
values (1, 181232.6, 1, 49);

-- b.
insert into SERVICIOPUBLICO (ID, COSTO, TIPO, ALOJAMIENTO)
values (1, 315844.1, 1, 49);

-- c.
SELECT * FROM SERVICIOPUBLICO;
```

**Figura 56.** Pruebas de unicidad de tuplas para el ServicioPublico [Elaborado en SQLDeveloper]

a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 57.** Pruebas de unicidad de tuplas para el ServicioPublico punto a [Elaborado en SQLDeveloper]

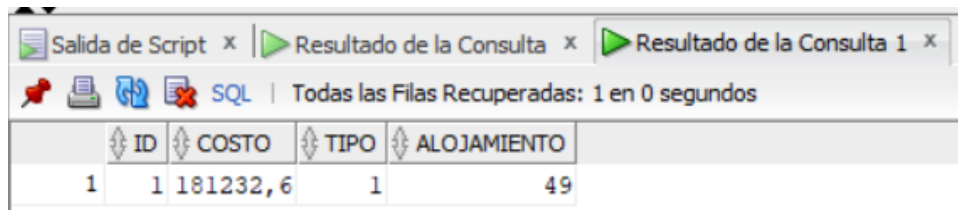
b. Llave Primaria Conocida

```

Error que empieza en la línea: 794 del comando :
insert into SERVICIOPUBLICO (ID, COSTO, TIPO, ALOJAMIENTO)
values (1, 315844.1, 1, 49)
Informe de error -
ORA-00001: restricción única (SYSTEM.SERVICIOPUBLICO_PK) violada

```

**Figura 58.** Pruebas de unicidad de tuplas para el ServicioPublico punto b [Elaborado en SQLDeveloper]  
c. Comprobación



	ID	COSTO	TIPO	ALOJAMIENTO
1	1	181232,6	1	49

**Figura 59.** Pruebas de unicidad de tuplas para el ServicioPublico punto c [Elaborado en SQLDeveloper]  
TipoServicio

```

-- 16. TipoServicio

-- a.
insert into TIPOSERVICIO (ID, NOMBRE) values (49, 'restaurante');

-- b.
insert into TIPOSERVICIO (ID, NOMBRE) values (49, 'piscina');

-- c.
SELECT * FROM TIPOSERVICIO;

```

**Figura 60.** Pruebas de unicidad de tuplas para el TipoServicio [Elaborado en SQLDeveloper]

a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 61.** Pruebas de unicidad de tuplas para el TipoServicio punto a [Elaborado en SQLDeveloper]  
b. Llave Primaria Conocida

```

Error que empieza en la línea: 808 del comando :
insert into TIPOSERVICIO (ID, NOMBRE) values (49, 'piscina')
Informe de error -
ORA-00001: restricción única (SYSTEM.TIPOSERVICIO_PK) violada

```

**Figura 62.** Pruebas de unicidad de tuplas para el TipoServicio punto b [Elaborado en SQLDeveloper]  
c. Comprobación

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x

SQL | Todas las Filas Recuperadas: 1 en 0,001 segundos

ID	NOMBRE
1	49 restaurante

**Figura 63.** Pruebas de unicidad de tuplas para el TipoServicio punto c [Elaborado en SQLDeveloper]

```
-- 17. Servicio
-- a.
insert into SERVICIO (ID, TIPO, ALOJAMIENTO) values (66, 49, 49);
-- b.
insert into SERVICIO (ID, TIPO, ALOJAMIENTO) values (66, 49, 49);
-- c.
SELECT * FROM SERVICIO;
```

**Figura 64.** Pruebas de unicidad de tuplas para el Servicio [Elaborado en SQLDeveloper]

a. Llave Primaria Nueva

```
1 fila insertadas.
```

**Figura 65.** Pruebas de unicidad de tuplas para el Servicio punto a [Elaborado en SQLDeveloper]

b. Llave Primaria Conocida

```
Error que empieza en la línea: 820 del comando :
insert into SERVICIO (ID, TIPO, ALOJAMIENTO) values (66, 49, 49)
Informe de error -
ORA-00001: restricción única (SYSTEM.SERVICIO_PK) violada
```

**Figura 66.** Pruebas de unicidad de tuplas para el Servicio punto b [Elaborado en SQLDeveloper]

c. Comprobación

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x

SQL | Todas las Filas Recuperadas: 1 en 0,002 segundos

ID	TIPO	ALOJAMIENTO
1	66	49

**Figura 67.** Pruebas de unicidad de tuplas para el Servicio punto c [Elaborado en SQLDeveloper]

## 6.2 Pruebas de integridad con FK

### Alojamiento

```
-- Alojamiento

-- a. Alojamiento con FK existente Seguro
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (200, 18, '71074 Lyons Parkway',
1329228.6, 1, 44, 1, 97.6, 49, null, 49, 1);

-- b. Alojamiento con FK no existente Seguro
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (201, 5, '18 Dunning Drive',
1646894.0, 0, 48, 0, 94.7, 49, null, 1000, 1);

-- c.
SELECT * FROM ALOJAMIENTO;

-- d. Haga pruebas de borrado de tuplas maestras y dependientes.
DELETE FROM SEGURO WHERE ID = 49; -- maestra pone problema
DELETE FROM ALOJAMIENTO WHERE ID = 200; -- dependiente no pone problema
```

**Figura 69.** Pruebas de integridad con FK para el Alojamiento [Elaborado en SQLDeveloper]

a. Inserte una tupla 1 que tenga una FK que se encuentra en la tabla referenciada

1 fila insertadas.

**Figura 70.** Pruebas de integridad con FK para el Alojamiento punto a [Elaborado en SQLDeveloper]

b. Inserte una tupla 1 que tenga una FK que no se encuentra en la tabla referenciada

```
Error que empieza en la línea: 846 del comando :
insert into ALOJAMIENTO (ID, NUMHABITACIONES, UBICACION, PRECIO, AMOBLADO,
CAPACIDAD, COMPARTIDO, INDICEOCUPACION, TIPO, HORARIO, SEGURO, OPERADOR)
values (201, 5, '18 Dunning Drive',
1646894.0, 0, 48, 0, 94.7, 49, null, 1000, 1)
Informe de error -
ORA-02291: restricción de integridad (SYSTEM.FK_SEGURO_ALOJAMIENTO) violada - clave principal no encontrada
```

**Figura 71.** Pruebas de integridad con FK para el Alojamiento punto b [Elaborado en SQLDeveloper]

c. Haga las pruebas de inserción para cada caso

ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR
1	49	18 71074 Lyons Parkway	1329228,6	1	44	1	97,6	49	(null)	49	1
2	200	18 71074 Lyons Parkway	1329228,6	1	44	1	97,6	49	(null)	49	1

**Figura 72.** Pruebas de integridad con FK para el Alojamiento punto c [Elaborado en SQLDeveloper]

d. Haga pruebas de borrado de tuplas maestras y dependientes.

Al eliminar la tupla maestra de la tabla Seguro:

```
Error que empieza en la línea: 855 del comando :
DELETE FROM SEGURO WHERE ID = 49
Informe de error -
ORA-02292: restricción de integridad (SYSTEM.FK_SEGURO_ALOJAMIENTO) violada - registro secundario encontrado
```

**Figura 73.** Pruebas de integridad con FK para el Alojamiento punto d1 [Elaborado en SQLDeveloper]

Al eliminar la tupla dependiente de la tabla Alojamiento:

1 fila eliminado

**Figura 74.** Pruebas de integridad con FK para el Alojamiento punto d2 [Elaborado en SQLDeveloper]

## Reserva

```
--a. Inserte una tupla 1 que tenga una FK que se encuentra en la tabla referenciada
-- Reserva con el CONTRATO existente
-- Reserva con Todas sus FKs existentes: CONTRATO EXISTENTE

insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (1, to_date('2022-06-03', 'yyyy-mm-dd'),
to_date('2022-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'), 2, 4, 1, 3365758.4);

--b. Inserte una tupla 1 que NO tenga una FK que se encuentra en la tabla referenciada (CONTRATO)
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (2, to_date('2022-06-03', 'yyyy-mm-dd'),
to_date('2022-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'), 2, 4, 7, 3365758.4);

--c. Haga las pruebas de inserción para cada caso
SELECT * FROM RESERVA;

--d. Haga pruebas de borrado de tuplas maestras y dependientes.

DELETE FROM CONTRATO WHERE ID= 1;
DELETE FROM RESERVA WHERE ID = 1;
```

**Figura 75.** Pruebas de integridad con FK para la Reserva [Elaborado en SQLDeveloper]

a. Inserte una tupla 1 que tenga una FK que se encuentra en la tabla referenciada.

Con el contrato existente se obtuvo:

1 fila insertadas.

**Figura 76.** Pruebas de integridad con FK para la Reserva punto a [Elaborado en SQLDeveloper]

b. Inserte una tupla 1 que tenga una FK que no se encuentra en la tabla referenciada

Con un contrato que no existe se obtuvo:

```
Error que empieza en la línea: 4 del comando :
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (2, to_date('2022-06-03', 'yyyy-mm-dd'),
to_date('2022-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'),
2, 4, 7, 3365758.4)
Informe de error -
ORA-02291: restricción de integridad (SYSTEM.FK_CONTRATO_RESERVA) violada - clave principal no encontrada
```

**Figura 77.** Pruebas de integridad con FK para la Reserva punto b [Elaborado en SQLDeveloper]

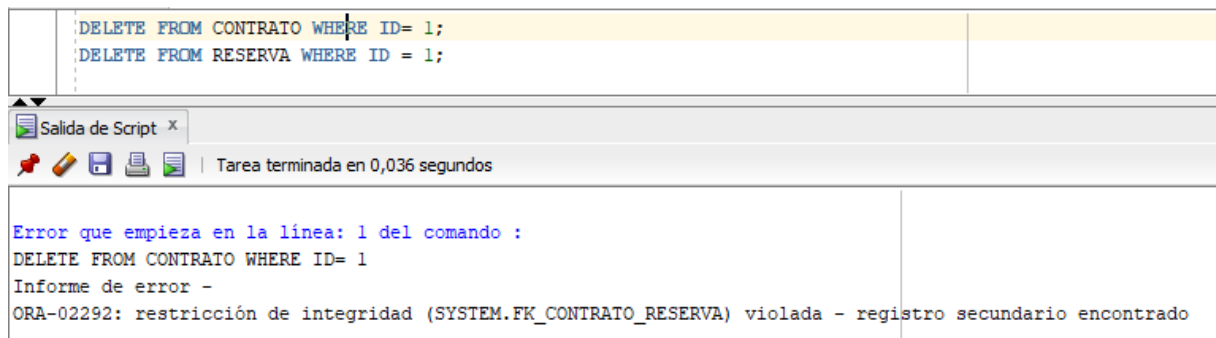
c. Haga las pruebas de inserción para cada caso

ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO
1	1 03/06/22	17/03/22	07/06/22	2	4	1	3365758,4

**Figura 78.** Pruebas de integridad con FK para la Reserva punto c [Elaborado en SQLDeveloper]

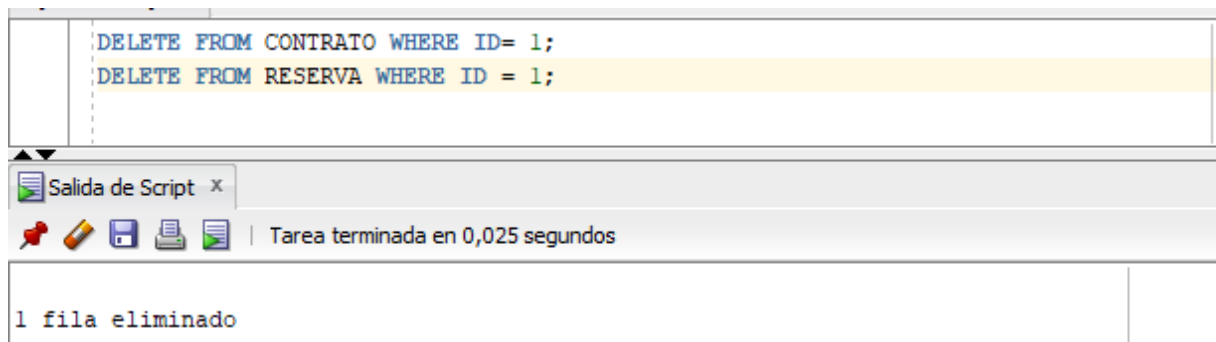
d. Haga pruebas de borrado de tuplas maestras y dependientes.

Al eliminar la tupla maestra de la tabla Contrato:



**Figura 79.** Pruebas de integridad con FK para la Reserva punto d1 [Elaborado en SQLDeveloper]

Al eliminar la tupla dependiente de la tabla Reserva:



**Figura 80.** Pruebas de integridad con FK para la Reserva punto d2 [Elaborado en SQLDeveloper]

Cuando se elimina primero contrato se elimina directamente la Reserva asociada a ese contrato. Por el contrario cuando primero se elimina reserva y luego contrato ahí se eliminan independientemente las tuplas.

## 6.3 FK Pruebas de integridad de acuerdo con restricciones de chequeo

### TipoMiembroComunidad

```
--- PRUEBAS DE INTEGRIDAD CON RESTRICCIONES DE CHEQUEO
-- a. Inserte tuplas que cumplen con las restricciones de chequeo establecidas
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (1, 'Profesor');
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (2, 'Empleado');
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (3, 'Egresado');
-- b. Inserte tuplas que violan las restricciones de chequeo establecidas
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (1, 'Casol');
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (2, 'Caso2');
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (3, 'Caso3');
-- c. Haga las pruebas de inserción y borrado correspondientes.
SELECT * FROM TIPOMIEMBROCOMUNIDAD;
```

**Figura 81.** Pruebas de integridad de acuerdo con restricciones de chequeo para TipoMiembroComunidad  
[Elaborado en SQLDeveloper]

#### a. Inserte tuplas que cumplen con las restricciones de chequeo establecidas

```
1 fila insertadas.

1 fila insertadas.

1 fila insertadas.
```

**Figura 82.** Pruebas de integridad de acuerdo con restricciones de chequeo para TipoMiembroComunidad  
punto a [Elaborado en SQLDeveloper]

#### b. Inserte tuplas que violan las restricciones de chequeo establecidas

```
Error que empieza en la línea: 8 del comando :
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (1, 'Casol')
Informe de error -
ORA-02290: restricción de control (SYSTEM.CK_NOMBRE_TIPOMIEMBROCOMUNIDAD) violada

Error que empieza en la línea: 9 del comando :
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (2, 'Caso2')
Informe de error -
ORA-02290: restricción de control (SYSTEM.CK_NOMBRE_TIPOMIEMBROCOMUNIDAD) violada

Error que empieza en la línea: 10 del comando :
insert into TIPOMIEMBROCOMUNIDAD (ID, NOMBRE) values (3, 'Caso3')
Informe de error -
ORA-02290: restricción de control (SYSTEM.CK_NOMBRE_TIPOMIEMBROCOMUNIDAD) violada
```

**Figura 83.** Pruebas de integridad de acuerdo con restricciones de chequeo para TipoMiembroComunidad  
punto b [Elaborado en SQLDeveloper]

#### c. Haga las pruebas de inserción y borrado correspondientes.

	ID	NOMBRE
1	1	Profesor
2	2	Empleado
3	3	Egresado

**Figura 84.** Pruebas de integridad de acuerdo con restricciones de chequeo para TipoMiembroComunidad punto c [Elaborado en SQLDeveloper]

## Reserva

```
-- Reserva

-- a. tuplas con fechaInicio < fechaFinal
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (123, null,
to_date('2022-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'),
49, 4, 4, 3365758.4);

insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (124, null,
to_date('2021-04-22', 'yyyy-mm-dd'), to_date('2023-02-19', 'yyyy-mm-dd'),
49, 4, 4, 3052932.6);

-- b. tuplas con fechaInicio >= fechaFinal
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (125, null,
to_date('2023-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'),
49, 4, 4, 3365758.4);

insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (126, null,
to_date('2021-04-22', 'yyyy-mm-dd'), to_date('2021-02-19', 'yyyy-mm-dd'),
49, 4, 4, 3052932.6);

-- c.
SELECT * FROM RESERVA;
```

**Figura 85.** Pruebas de integridad de acuerdo con restricciones de chequeo para Reserva [Elaborado en SQLDeveloper]

a. Inserte tuplas que cumplen con las restricciones de chequeo establecidas

```
1 fila insertadas.

1 fila insertadas.
```

**Figura 86.** Pruebas de integridad de acuerdo con restricciones de chequeo para Reserva punto a [Elaborado en SQLDeveloper]

b. Inserte tuplas que violan las restricciones de chequeo establecidas



```

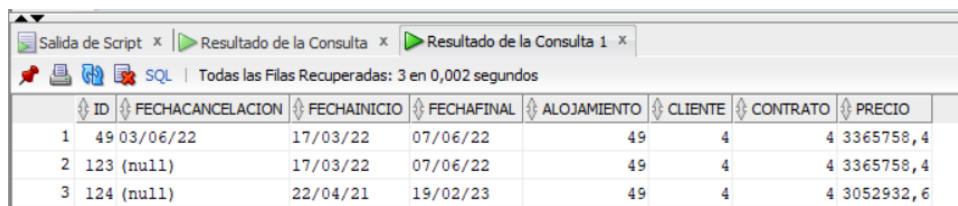
Error que empieza en la línea: 879 del comando :
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (125, null,
to_date('2023-03-17', 'yyyy-mm-dd'), to_date('2022-06-07', 'yyyy-mm-dd'),
49, 4, 4, 3365758.4)
Informe de error -
ORA-02290: restricción de control (SYSTEM.CK_MENOR_FECHAINICIOFINAL_RESERVA) violada

Error que empieza en la línea: 884 del comando :
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO,
CLIENTE, CONTRATO, PRECIO) values (126, null,
to_date('2021-04-22', 'yyyy-mm-dd'), to_date('2021-02-19', 'yyyy-mm-dd'),
49, 4, 4, 3052932.6)
Informe de error -
ORA-02290: restricción de control (SYSTEM.CK_MENOR_FECHAINICIOFINAL_RESERVA) violada

```

**Figura 87.** Pruebas de integridad de acuerdo con restricciones de chequeo para Reserva punto b [Elaborado en SQLDeveloper]

c. Haga las pruebas de inserción y borrado correspondientes.



	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO
1	49	03/06/22	17/03/22	07/06/22	49	4	4	3365758,4
2	123	(null)	17/03/22	07/06/22	49	4	4	3365758,4
3	124	(null)	22/04/21	19/02/23	49	4	4	3052932,6

**Figura 88.** Pruebas de integridad de acuerdo con restricciones de chequeo para Reserva punto c [Elaborado en SQLDeveloper]

## 7 Supuestos adicionales sobre las reglas de negocio encontradas en el caso de estudio

Con respecto al modelo relacional de la iteración 1, se hicieron los siguientes cambios también considerando las reglas del negocio:

1. Adición del atributo NOMBRE en la tabla CLIENTE.
2. Adición de una restricción de UNIQUE en el atributo NOMBRE de todas las tablas de Tipo: TipoMiembroComunidad, TipoOperadorEmpresa, TipoOperadorNatural, TipoAlojamiento, TipoServicioPublico y TipoServicio.
3. Adición de atributo pago en tabla TipoAlojamiento.

## 8 Documentación de los Requerimientos Funcionales

### 8.1 Requerimiento Funcional 4: Registrar una reserva

En la Tabla 4 se muestra el cuarto requerimiento funcional de Alohandes. Este se centra en registrar una reserva de acuerdo a las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta ... etc.

**Tabla 4.** Registrar una Reserva

<b>Nombre</b>	RF4. Registro de una reserva
---------------	------------------------------

<b>Resumen</b>	La aplicación debe permitir que los <i>clientes</i> de Alohandes puedan buscar alojamiento y reservar una estancia del de su elección. Esta debe considerar las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta, etc.
<b>Entradas</b>	
Long: Cliente.Id	
String: Cliente.Clave	
Atributos de Reserva que se pueden ver en la Tabla 8 de la sección 3 en la entidad Reserva	
Atributos de entidades que se relacionan con una Reserva que lo necesiten (Alojamiento, Servicio, ServioPúblico)	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> String: Registro exitoso	
<b>Creación de:</b> nueva tupla en relación Reserva	
<b>RNF asociados</b>	
Durabilidad (transaccionalidad): Para el cumplimiento de este requerimiento funcional se necesita persistencia. Esto se debe a que en el momento en que se genera una reserva se necesita tener la reserva vigente durante el tiempo que dure la reserva para dicho cliente y quitar el alojamiento y su disponibilidad en ese periodo.	
Isolation- Concurrencia (Enunciado): Debido a que varios usuarios pueden intentar realizar una reserva al mismo tiempo, es necesario garantizar que la aplicación maneje correctamente la concurrencia para evitar problemas de consistencia en la base de datos.	
Privacidad (enunciado): La privacidad es importante para garantizar que los usuarios solo puedan manipular y consultar la información que les es propia, evitando así posibles vulneraciones de seguridad y privacidad.	

## 8.2 Requerimiento Funcional 5: Cancelar una Reserva

**Tabla 5.** Cancelar una Reserva

<b>Nombre</b>	RF5. Cancelar una reserva
<b>Resumen</b>	La aplicación debe permitir que un cliente que ha hecho una reserva en el sistema pueda cancelar dicha reserva antes del inicio de la misma.
<b>Entradas</b>	
Long: Cliente.Id	
Long: Reserva.Id	
String: Cliente.Clave	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> String: Cancelación exitosa	
<b>Delección de:</b> tupla en relación Reserva	
<b>RNF asociados</b>	
Durabilidad (transaccionalidad): Para el cumplimiento de este requerimiento funcional se necesita persistencia. Esto se debe a que en el momento en que se cancela cierta reserva se necesita que quede guardada en la base de datos para que se pueda consultar la fecha de cancelación en futuras oportunidades.	
Privacidad: Se debe garantizar que el usuario que quiere cancelar una reserva solo pueda acceder a información propia, mas no de otros usuarios, pues se debe garantizar la privacidad de estos.	

## 8.3 Requerimiento Funcional 6: Retirar una oferta de alojamiento

En la Tabla 6 se muestra el sexto requerimiento funcional de Alohandes. Este se centra en cancelar una reserva de acuerdo a las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta ... etc.

**Tabla 6.** Retirar una oferta de Alojamiento

<b>Nombre</b>	RF6. Retirar una oferta de Alojamiento
<b>Resumen</b>	La aplicación debe permitir que los <i>operadores</i> de ALOHA puedan retirar una oferta de alojamiento que ya no esté disponible. Este requerimiento implica que la información de las ofertas retiradas se actualice correctamente en la base de datos de la aplicación, garantizando que no se muestran alojamientos que ya no están disponibles para reservar.
<b>Entradas</b>	
Long: Alojamiento.Id	
Atributos de entidades que se relacionan con la entidad Alojamiento que lo necesiten (Contrato, Seguro, Horario)	
String: Operador.Clave	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> String: Cancelación exitosa	
<b>Delección de:</b> tupla en relación Alojamiento	
<b>RNF asociados</b>	
Persistencia (enunciado): La acción de retirar la oferta de Alojamiento implica una actualización en la base de datos de la aplicación con tal de quitar la oferta y que no se puedan hacer reservas a futuro.	
Seguridad (transaccionalidad): Se requiere que los únicos permitidos para retirar ofertas de alojamiento sean los operadores de acuerdo a sus condiciones actuales y disponibilidad. Esto se debe a que estos son los responsables de publicar ofertas.	
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que los clientes realicen reservas cuando ya no está disponible el alojamiento.	

#### 8.4 Mostrar el dinero recibido por cada proveedor (RFC1)

En la Tabla 7 se muestra el séptimo requerimiento funcional de Alohandes. Este se centra en mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido.

**Tabla 7.** Mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido

<b>Nombre</b>	RF7. Mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido
<b>Resumen</b>	La aplicación debe poder mostrar el dinero que cada proveedor registrado en la base de datos ha recibido en el año actual y en el año corrido
<b>Entradas</b>	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> String: valores de dinero del año actual y del año corrido	
<b>RNF asociados</b>	
Concurrencia: se debe garantizar que varios usuarios de la base de datos puedan acceder simultáneamente a la información y no haya problemas de concurrencia	
Persistencia (transaccionalidad): La información del dinero recibido en el año corrido y en el año actual de los operadores debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de los operadores debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos.	

### 8.5 Mostrar las 20 ofertas más populares (RFC2)

En la Tabla 8 se muestra el octavo requerimiento funcional de Alohandes. Este se centra en mostrar las 20 ofertas más populares.

**Tabla 8.** Mostrar las 20 ofertas más populares

<b>Nombre</b>	RF8. (RFC2) Mostrar las 20 ofertas más populares
<b>Resumen</b>	El sistema debe permitir la operación de consulta por parte de los <i>clientes</i> y <i>operadores</i> de Alohandes. Esta se centra en consultar las 20 ofertas más populares de Alojamiento de acuerdo a la cardinalidad de la tabla Reserva.
<b>Entradas</b>	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> List<Alojamiento>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento y el operador dueño	
<b>RNF asociados</b>	
Concurrencia (enunciado): Pueden hacerse más de una consulta de este tipo al tiempo.	
Persistencia (transaccionalidad): La información de las ofertas y su popularidad debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de las ofertas y su popularidad debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos. Por lo tanto, esta debe concordar con el número de reservas que ha tenido cada alojamiento.	

### 8.6 Mostrar el Índice de Ocupación (RFC3)

En la Tabla 9 se muestra el noveno requerimiento funcional de Alohandes. Este se centra en mostrar el índice de Ocupación de cada una de las ofertas de alojamiento registradas.

**Tabla 9.** Mostrar el Índice de Ocupación de cada una de las ofertas de alojamiento registradas

<b>Nombre</b>	RF9. Mostrar el índice de ocupación de cada una de las ofertas de alojamiento registradas
<b>Resumen</b>	El sistema debe permitir la operación de consulta por parte de los <i>clientes</i> y <i>operadores</i> de Alohandes. Esta se centra en consultar el índice de ocupación de las ofertas de alojamiento registradas en el sistema.
<b>Entradas</b>	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> List<string>: valores de indiceOcupacion de todas las tuplas de la relación Alojamiento	
<b>RNF asociados</b>	
Concurrencia: se debe garantizar que varios usuarios de la base de datos puedan acceder simultáneamente a la información del índice de ocupación de los operadores y no haya problemas de concurrencia.	
Persistencia (transaccionalidad): La información del índice de ocupación de los operadores debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de los operadores debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos.	

### 8.7 Mostrar los alojamientos disponibles (RFC4)

En la Tabla 10 se muestra el décimo requerimiento funcional de Alohandes. Este se centra en mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios.

**Tabla 10.** Mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios

<b>Nombre</b>	RF10. Mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios.
<b>Resumen</b>	La aplicación debe permitir a los <i>clientes</i> buscar alojamiento según un conjunto de criterios, como la ubicación, las fechas de estancia, la capacidad, la dotación o servicios incluidos, entre otros. La aplicación debe mostrar una lista de alojamientos disponibles que cumplan con esos criterios.
<b>Entradas</b>	
Date: fechaInicial	
Date: fechaFinal	
List<Servicio> o List<ServicioPublico>	
<b>Resultados</b>	
<b>Enviar Mensaje:</b> List<Alojamiento>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento y el operador dueño (ej. Alojamiento.ubicacion, Operador.Nombre).	
<b>RNF asociados</b>	
Concurrencia: Pueden hacerse más de una consulta de este tipo al tiempo.	
Seguridad: Los usuarios deben tener permisos adecuados para acceder a la información de alojamientos y disponibilidad, y los datos deben ser protegidos contra acceso no autorizado. Es decir, personas que no sean clientes o empresas relacionadas con la Universidad de los Andes.	
Usabilidad: La interfaz de usuario debe ser clara, sencilla e intuitiva para facilitar la búsqueda de alojamientos por parte de los <i>clientes</i> .	