Sistemas Transaccionales - P1

d.trivino, i.bermudezl , je.lopeze1

Octubre de 2023

1. Caso de estudio: HotelAndes

En el caso de estudio propuesto se nos pide realizar una aplicacion que se encargue de las necesidades de negocio de HotelAndes. Estas necesidades abarcan el manejo de usuarios, y su rol dentro del negocio; Tendremos los clientes, los recepcionistas, el gerente del hotel, entre otros. Junto con la administración de las reservas de un cliente junto a su correspondiente llegada (`check-in` y salida (`check-out`) del establecimiento. Además, en HotelAndes se pueden presentar tarifas especiales por eventos, así que es algo que se debe realizar al momento de hacer la facturación. Otro detalle que también se debe tomar en cuenta es el consumo de los servicios que ofrece el hotel. Y entre estos se encuentra el restaurante, el gimnasio, piscina, etc.

2. Modelo conceptual y lógico

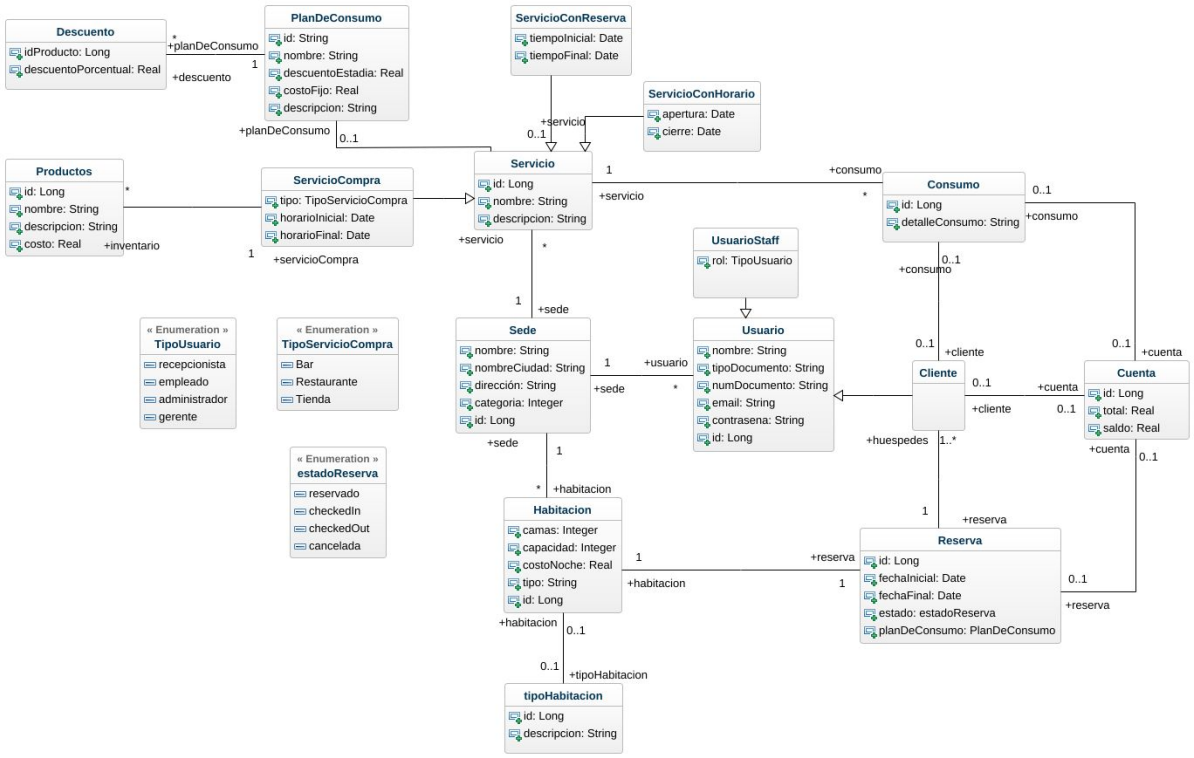

Figura 1: Modelo conceptual del caso de estudio hotelAndes

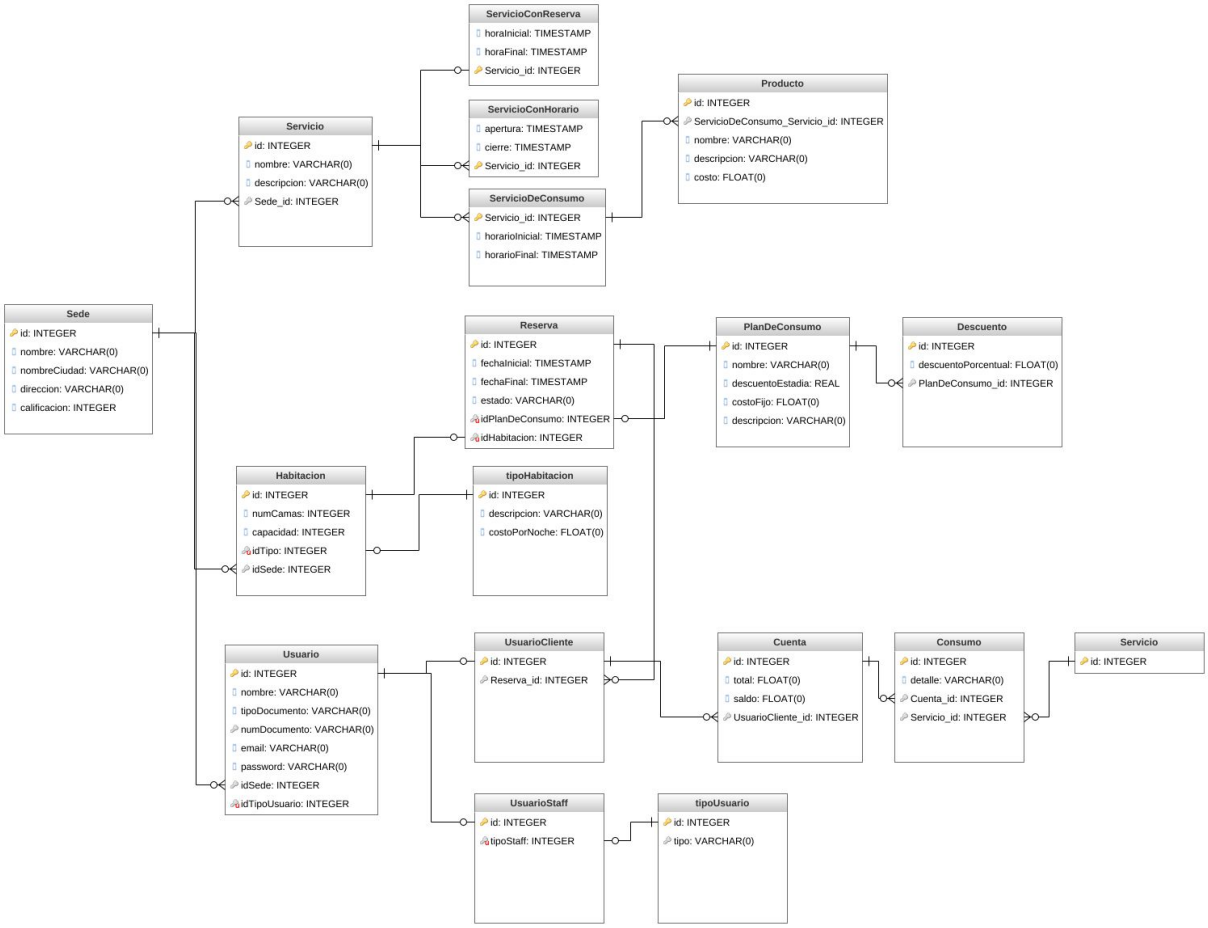

Figura 2: Boceto del modelo relacional desarrollado en genMyModel

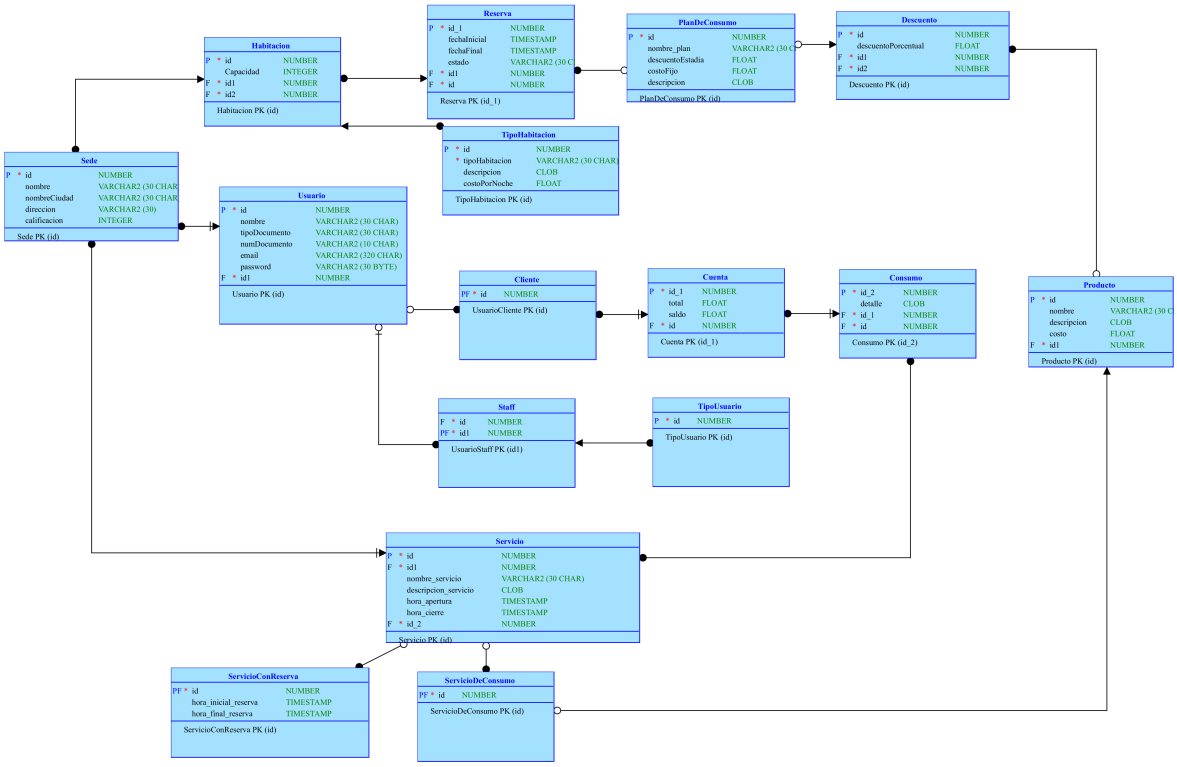

Figura 3: Modelo lógico terminado en Oracle Data Modeler

3. Diseño de la base de datos

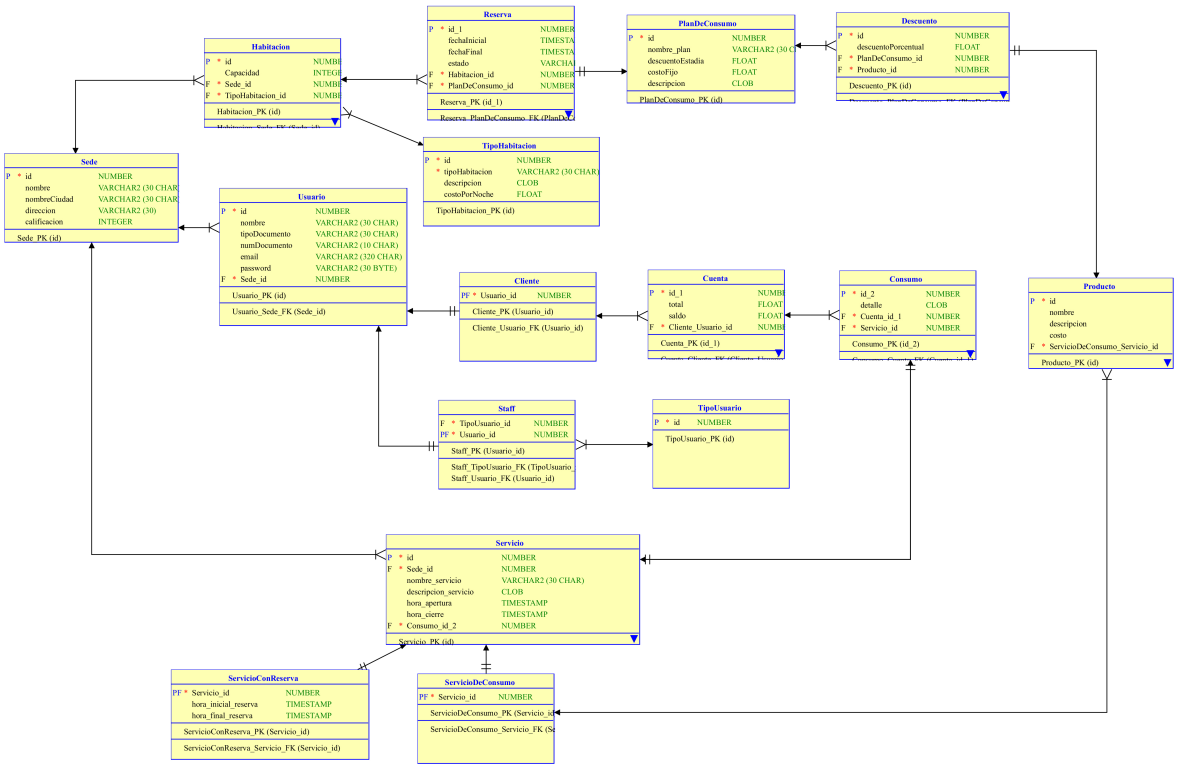

Figura 4: Modelo Relacional generado por Oracle Data Modeler

3.1. Análisis de normalización

3.1.1. Primera forma normal

Se puede observar trivialmente que todas las relaciones (tablas) cuentan únicamente con tipos de datos atómicos, y las tablas por definición solo tienen dos dimensiones (filas y columnas).

3.1.2. Segunda forma normal

Todas las relaciones abajo descritas como dependencias funcionales cumplen con el siguiente criterio: No debe haber dependencias parciales en la relación. Esto se ve porque partimos de una llave candidata compuesta en varias de las relaciones, y generalmente lo reducimos a el ÏD"de cada entidad como llave primaria.

Sede

id, nombre, nombreCiudad, direccion $\implies$ calificacion

id $\implies$ nombre, nombreCiudad, direccion, calificacion

Habitacion

sede_id, id $\implies$ capacidad, tipoHabitacion

TipoHabitacion

id, tipoHabitacion $\implies$ descripcion, costoPorNoche

id $\implies$ descripcion, costoPorNoche, tipoHabitacion

Reserva

id, fechaInicial, fechaFinal, Cliente_Usuario_id $\implies$ estado, Habitacion_id, PlanDeConsumo_id

id $\implies$ estado, Habitacion_id, PlanDeConsumo_id, fechaInicial, fechaFinal, Cliente_Usuario_id

UsuariosDeSede

sede_id, Usuario_id $\implies$ sede_id, Usuario_id

Usuario

id, email $\implies$ nombre, tipoDocumento, numDocumento, password

id $\implies$ nombre, tipoDocumento, numDocumento, password, email

TipoUsuario id, nombre_tipo $\implies$ id, nombre_tipo

id, nombre_tipo

Plan de consumo

id, nombre_plan $\implies$ descuentoEstadia, costoFijo, descripcion

id $\implies$ descuentoEstadia, costoFijo, descripcion, nombre_plan

Descuento

id, PlanDeConsumo_id, Producto_id $\implies$ descuentoPorcentual

id $\implies$ descuentoPorcentual, PlanDeConsumo_id, Producto_id

Cuenta

id, usuario_id $\implies$ total, saldo

id $\implies$ total, saldo, usuario_id

Consumo

id $\implies$ detalle, cuenta_id, servicio_id

Producto

id, nombre, servicio_id $\implies$ descripcion, costo

id $\implies$ descripcion, costo, nombre, servicio_id

Servicio

id, nombre_servicio, sede_id $\implies$ descripcion_servicio, hora_apertura, hora_cierre

id $\implies$ descripcion_servicio, hora_apertura, hora_cierre, nombre_servicio, sede_id

ReservaDeServicio

id, usuario_id, servicio_id, hora_inicial $\implies$ hora_final

id $\implies$ hora_final, usuario_id, servicio_id, hora_inicial

4. Objetivos logrados y no logrados

De los 11 requerimientos funcionales se lograron los siguientes (lograr significa implementar y probar las sentencias CRUD tanto en SQL como en Postman) Se lograron los requerimientos 1,2, 3,4,6 7 y 10. Los demas requerimientos fueron implementados, pero no han sido probados (reglas de negocio, CRUD, integridad) con SQL y postman en su enteridad.

5. Escenarios de prueba

5.1. Requerimientos funcionales

Para realizar las pruebas se utilizo SQL Developer Postman para probar las funcionalidades CRUD de cada uno de los requerimientos funcionales y entidades del modelo. Se adjunta en la carpeta DOCS del proyecto la colección de postman utilizada para probar la api y las solicitudes CRUD.

1. Respuesta : 200 OK
2. RF2 - Crear un usuario
   ïd_type": ÇC",
   ïd_number":"1111111111",
   "password":"1234",
   "name": "juan",
   .emʺail":.ºtro email",
   role_id": 1
   }
3. RF3 - Crear una habitacion

   Body enviado con POST {
   ïd": 1,
   "name": "test room",
   roomTypeId": 1
   }
   Respuesta : 200 OK

4. RF4
5. RF5
6. RF6
7. RF7
8. RF8
9. RF9
10. RF10
11. RF11