AUTO-RENTA

RUBEN DARIO ORTIZ ORTEGA

P1

PEDRO FELIPE GÓMEZ BONILLA

CAMPUSLANDS
ARTEMIS
RUTA JAVA
FLORIDABLANCA
2024

TABLA DE CONTENIDO

1. **INTRODUCCIÓN**
2. **CASO DE ESTUDIO**
    - **Laboratorio**
    - **Consideraciones**
3. **INSTALACIÓN GENERAL**
    - **Requisitos Previos**
        - ➔ ***Paso 1: Instalación de MySQL Server***
        - ➔ ***Paso 2: Instalación de MySQL Workbench***
        - ➔ ***Paso 3: Importación de la Base de Datos***
        - ➔ ***Paso 4: Configuración y Verificación***
        - ➔ ***Paso 5: Documentación de Usuarios y Accesos***
4. **PLANIFICACIÓN**
    - ***Análisis de Requisitos***
    - ***Diseño Conceptual***
    - ***Diseño Lógico***
    - ***Implementación de la Base de Datos***
    - ***Optimización y Ajustes de Rendimiento***
    - ***Implementación de Procedimientos y Disparadores (Triggers)***
    - ***Seguridad y Gestión de Usuarios***
    - ***Documentación y Entrega***
    - ***Pruebas y Validación***
    - ***Implementación y Mantenimiento Continuo***
5. **EJECUCIÓN**
    - **Diagrama Conceptual**
    - **Diagrama Lógico**
    - **Diagrama Físico**
    - **Normalización**
    - **Diagrama E-R**
    - **Triggers**
    - **Eventos**
    - **Usuarios y Accesos**
    - **Referencias**

1. INTRODUCCIÓN

En el entorno empresarial actual, la eficiencia y la precisión en la gestión de información son elementos cruciales para el éxito operativo. En este contexto, se presenta el desafío de desarrollar un sistema de información para Auto-Rent, una empresa dedicada al alquiler de vehículos con una proyección de expansión significativa a nivel nacional. Este proyecto requiere el diseño de una base de datos robusta que soporte las operaciones diarias de la empresa, asegurando una gestión integral de sucursales, empleados, clientes y vehículos.

Auto-Rent opera actualmente con 5 sucursales ubicadas estratégicamente en diferentes ciudades, gestionando una flota diversa de vehículos que varían en tipo, modelo, capacidad y especificaciones técnicas. Su modelo de negocio permite a los clientes alquilar vehículos en una sucursal y devolverlos en otra, ofreciendo además descuentos variables a lo largo del año en función del tipo de vehículo y la duración del alquiler.

El presente documento detalla los requisitos y especificaciones para el diseño y desarrollo de una base de datos que respalde estos procesos operativos y que sirva como fundamento para la implementación de tres aplicativos de software: un sistema de gestión interno, un aplicativo web para clientes y una aplicación móvil para dispositivos Android. Estos aplicativos están diseñados para mejorar la experiencia tanto de los empleados de Auto-Rent como de sus clientes, facilitando desde la gestión interna de sucursales y vehículos hasta la consulta de disponibilidad, alquiler y seguimiento de historiales de alquileres.

El objetivo principal es garantizar que la base de datos y los aplicativos desarrollados no solo cumplan con los requisitos funcionales esperados, sino que también optimicen la eficiencia operativa y mejoren la satisfacción del cliente, contribuyendo así al crecimiento sostenido y la expansión planificada de Auto-Rent en el mercado nacional.

Este documento proporciona un marco integral para la planificación, implementación y evaluación del sistema de información propuesto, destacando la importancia de una arquitectura sólida y escalable que asegure la viabilidad a largo plazo de las soluciones tecnológicas propuestas para Auto-Rent.

2. CASO DE ESTUDIO

LABORATORIO

Situación problema

La empresa donde usted trabaja ha sido contratada para desarrollar un sistema de información para una empresa de alquiler de vehículos llamada Auto-Rent, y usted ha sido designado para diseñar una base de datos para ese sistema de información.
Auto-Rent cuenta con 5 sucursales en diferentes ciudades y se proyecta a expandirse a otras ciudades del país y cuenta con una flota propia de vehículos de diferentes tipos, modelos (año), capacidad, etc. Los clientes de Auto-Rent podrán alquilar un vehículo en una sucursal y entregarlo en otra sucursal.
Auto-Rent ofrece descuentos sobre diferentes tipos de vehículos a lo largo del año.
Los valores de alquiler dependen del tipo de vehículo (sedán, compacto, camioneta platón, camioneta lujo, deportivo, etc) y se cobran por días y/o semanas. Por ejemplo, si uno alquila un vehículo por 9 días, el valor cotizado será de 1 semana y 2 días. Si un cliente entrega el vehículo pasada la fecha de entrega contratada, se cobrarán los días adicionales con un incremento del 8%.

La base de datos deberá cumplir almacenar la siguiente información:

- Sucursales: ciudad y dirección donde se ubica, teléfono fijo, celular y correo electrónico.
- Empleados: sucursal donde labora, cédula, nombres, apellidos, dirección y ciudad de residencia, celular y correo electrónico.
- Clientes: cédula, nombres, apellidos, dirección y ciudad de residencia, celular y correo electrónico.
- Vehículos: tipo de vehículo, placa, referencia, modelo, puertas, capacidad, sunroof, motor, color.
- Alquileres: vehículo, cliente, empleado, sucursal y fecha de salida, sucursal y fecha de llegada, fecha esperada de llegada, valor de alquiler por semana, valor de alquiler por día, porcentaje de descuento, valor cotizado y valor pagado.

Requerimientos de los aplicativos de software

El departamento de desarrollo de software deberá desarrollar 3 aplicativos de software:

- Software de gestión: esta herramienta será utilizada por los empleados de Auto-Rent para gestión interna de sucursales, vehículos y empleados, y también podrá ser utilizada por los empleados que atienden al público.
- Aplicativo web para clientes, debe ofrecer las funciones:
- Registro (signup)
- Inicio de sesión (login).
- Consulta de disponibilidad de vehículos para alquiler por tipo de vehículo, rango de precios de alquiler y fechas de disponibilidad.
- Alquiler de vehículos.
- Consulta de historial de alquileres.
- Aplicativo para Android: debe ofrecer las mismas funciones del aplicativo web para clientes.

Consideraciones

Se espera que usted implemente la lógica de negocio de las siguientes operaciones para los aplicativos de software de clientes:

- Registro (signup).
- Inicio de sesión (login).
- Consulta de disponibilidad de vehículos para alquiler por tipo de vehículo, rango de precios de alquiler y fechas de disponibilidad.
- Alquiler de vehículos.
- Consulta de historial de alquileres.

Usted deberá proveer cuentas de usuario para los aplicativos de software previendo las necesidades que cada aplicativo tendrá.

3. INSTALACIÓN GENERAL

Requisitos Previos

Antes de comenzar con la instalación de la base de datos, asegúrate de tener los siguientes requisitos cumplidos:

- Sistema Operativo: Compatible con MySQL y MySQL Workbench.
- MySQL Server: Debe estar instalado y configurado correctamente.
- MySQL Workbench: Debe estar instalado en la máquina desde donde se realizará la administración y diseño de la base de datos.

Paso 1: Instalación de MySQL Server

1. Descarga el instalador de MySQL Server desde el sitio oficial de MySQL (https://dev.mysql.com/downloads/mysql/).
2. Ejecuta el instalador descargado y sigue las instrucciones en pantalla para completar la instalación.
3. Durante la instalación, configura la contraseña del usuario root de MySQL y el puerto de conexión (generalmente es el puerto 3306).

Paso 2: Instalación de MySQL Workbench

1. Descarga el instalador de MySQL Workbench desde el sitio oficial de MySQL (https://dev.mysql.com/downloads/workbench/).
2. Ejecuta el instalador descargado y sigue las instrucciones en pantalla para completar la instalación.

3. Una vez instalado, abre MySQL Workbench para asegurarte de que puedes conectarte al servidor MySQL local usando las credenciales de usuario root configuradas en el paso anterior.

Paso 3: Importación de la Base de Datos

1. Abre MySQL Workbench y conecta al servidor MySQL local.
2. En MySQL Workbench, utiliza la funcionalidad de importación para cargar la estructura de la base de datos proporcionada. Por lo general, esto implica:
    - Abrir el archivo SQL que contiene las sentencias CREATE TABLE y otros comandos de definición de la base de datos.
    - Ejecutar el script SQL para crear todas las tablas, índices y relaciones necesarias.

Paso 4: Configuración y Verificación

1. Después de importar la estructura de la base de datos, verifica que todas las tablas y relaciones se hayan creado correctamente en MySQL Workbench.
2. Asegúrate de que los datos de muestra, si los hubiera, se hayan cargado correctamente y puedan ser consultados.
3. Verifica que los usuarios y permisos necesarios estén configurados adecuadamente para los diferentes aplicativos de software (gestión interna, aplicativo web para clientes y aplicación móvil).

Paso 5: Documentación de Usuarios y Accesos

1. Documenta los usuarios y contraseñas que se deben utilizar para acceder a la base de datos desde cada aplicación.
2. Incluye instrucciones claras sobre cómo configurar las conexiones a la base de datos desde los aplicativos de software (dirección IP, puerto, nombre de la base de datos, usuario y contraseña).

4. PLANIFICACIÓN

1.Análisis de Requisitos

- Se revisaron detalladamente todos los requisitos funcionales y no funcionales descritos en la introducción.
- Se identificaron las entidades principales (sucursales, empleados, clientes, vehículos, alquileres) y sus atributos.
- Define las relaciones entre las entidades y los tipos de cardinalidad (uno a uno, uno a muchos, muchos a muchos).

2. Diseño Conceptual

- Se creó un modelo de datos conceptual utilizando diagramas de entidad-relación (ERD) que representan todas las entidades y sus relaciones.
- Define las restricciones de integridad referencial necesarias para mantener la coherencia de los datos.
- Revisa el diseño conceptual con los stakeholders para obtener feedback y validar los requisitos.

3. Diseño Lógico

- Se convierte el modelo conceptual en un modelo de datos lógico utilizando MySQL Workbench u otra herramienta de diseño de bases de datos.
- Define las tablas específicas y los tipos de datos para cada atributo.
- Añade claves primarias y secundarias según sea necesario para asegurar la integridad de los datos.

4. Implementación de la Base de Datos

- Utiliza SQL para crear todas las tablas, índices y relaciones definidas en el diseño lógico.
- Implementa las restricciones de integridad referencial, como claves foráneas y reglas de eliminación y actualización.
- Carga los datos de muestra si es necesario para realizar pruebas iniciales y demostraciones.

5. Optimización y Ajustes de Rendimiento

- Revisa el diseño de la base de datos para asegurar que esté optimizado para las consultas típicas del sistema.
- Considera la creación de índices adicionales según sea necesario para mejorar el rendimiento de consultas frecuentes.
- Realiza pruebas de rendimiento para identificar y resolver posibles cuellos de botella en la base de datos.

6. Implementación de Procedimientos y Disparadores (Triggers)

- Define procedimientos almacenados y disparadores que automatizan tareas recurrentes o aplican lógica de negocio compleja directamente en la base de datos.

- Asegúrate de que los procedimientos y disparadores sean seguros y eficientes en términos de rendimiento.

7. Seguridad y Gestión de Usuarios

- Establece políticas de seguridad para controlar el acceso a la base de datos y sus objetos.
- Crea usuarios y roles con los permisos mínimos necesarios para cada aplicación y usuario final.
- Implementa copias de seguridad regulares y un plan de recuperación ante desastres para proteger los datos críticos.

8. Documentación y Entrega

- Documenta todo el diseño de la base de datos, incluyendo el modelo ER, el diseño lógico y la implementación física en MySQL.
- Proporciona manuales de usuario y administrador que expliquen cómo utilizar y mantener la base de datos.
- Entrega la base de datos y la documentación completa a los stakeholders y equipos de desarrollo de software para la integración con los aplicativos de software.

9. Pruebas y Validación

- Realiza pruebas exhaustivas para verificar que la base de datos cumple con todos los requisitos funcionales y no funcionales especificados.
- Asegúrate de que los aplicativos de software interactúen correctamente con la base de datos en todas las funciones principales (registro, inicio de sesión, alquiler de vehículos, consultas, etc.).

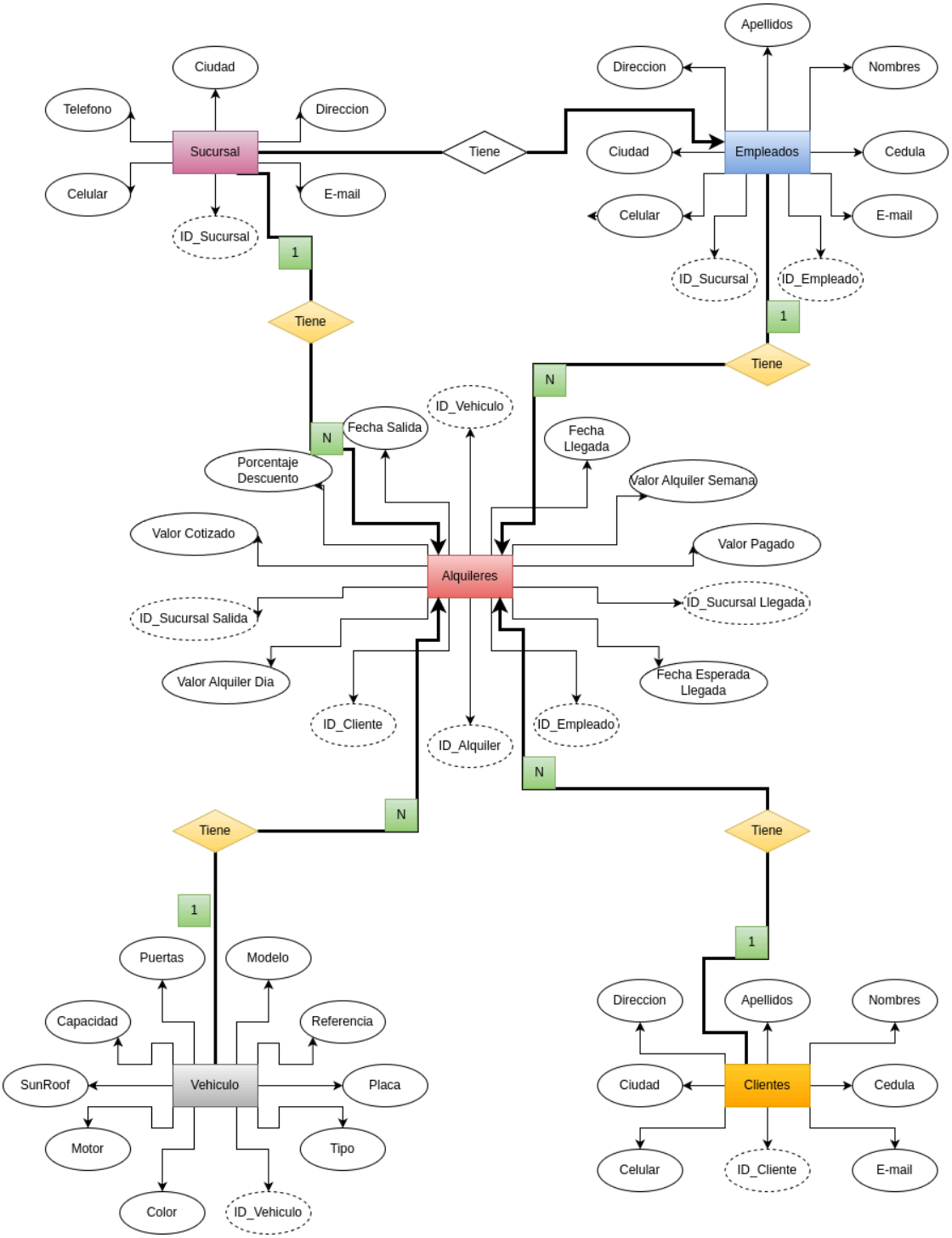
10. Implementación y Mantenimiento Continuo

- Implementa la base de datos en el entorno de producción según el plan de implementación acordado.
- Establece un proceso de monitoreo continuo para supervisar el rendimiento y la integridad de la base de datos.
- Planifica actualizaciones y mejoras periódicas según sea necesario para mantener la base de datos en óptimas condiciones.

5. EJECUCIÓN

DIAGRAMA CONCEPTUAL

Un modelo conceptual es una representación de un sistema, hecho de la composición de conceptos que se utilizan para ayudar a las personas a conocer, comprender o simular un tema que representa el modelo, incluye las entidades importantes y las relaciones entre ellos. También es un conjunto de conceptos. Algunos modelos son objetos físicos; por ejemplo, un modelo de juguete que se puede ensamblar y se puede hacer que funcione como el objeto que representa.



MODELO LÓGICO

Un modelo lógico de datos es un modelo que no es específico de una base de datos que describe aspectos relacionados con las necesidades de una organización para recopilar datos y las relaciones entre estos aspectos.

Un modelo lógico contiene representaciones de entidades y atributos, relaciones, identificadores exclusivos, subtipos y supertipos y restricciones entre relaciones. Un modelo lógico también puede contener objetos de modelo de dominio o referirse a uno o varios modelos de dominio o de glosario. Una vez definidas las relaciones y los objetos lógicos en un modelo lógico de datos, utilice el área de trabajo para transformar el modelo lógico en una representación física específica de la base de datos en forma de modelo físico de datos.

1. Sucursal (
- idSucursal INT PRIMARY KEY AUTO_INCREMENT,
- ciudad VARCHAR(100),
- direccion VARCHAR(255),
- telefonoFijo VARCHAR(20),
- celular VARCHAR(20),
- correoElectronico VARCHAR(100));

2. Empleado (
- idEmpleado INT PRIMARY KEY AUTO_INCREMENT,
- idSucursal INT,
- cedula VARCHAR(20) UNIQUE,
- nombres VARCHAR(100),
- apellidos VARCHAR(100),
- direccion VARCHAR(255),
- ciudadResidencia VARCHAR(100),
- celular VARCHAR(20),
- correoElectronico VARCHAR(100),
- FOREIGN KEY (idSucursal) REFERENCES Sucursal(idSucursal));

3. Cliente (
- idCliente INT PRIMARY KEY AUTO_INCREMENT,
- cedula VARCHAR(20) UNIQUE,
- nombres VARCHAR(100),
- apellidos VARCHAR(100),
- direccion VARCHAR(255),
- ciudadResidencia VARCHAR(100),
- celular VARCHAR(20),
- correoElectronico VARCHAR(100));

4. Vehículo (

- idVehiculo INT PRIMARY KEY AUTO_INCREMENT,
- tipo VARCHAR(50),
- placa VARCHAR(20) UNIQUE,
- referencia VARCHAR(100),
- modelo INT,
- puertas INT,
- capacidad INT,
- sunroof ENUM (‘Sí’, ‘No’)
- motor VARCHAR(50),
- color VARCHAR(50));

5. Alquiler (

- idAlquiler INT PRIMARY KEY AUTO_INCREMENT,
- idVehiculo INT,
- idCliente INT,
- idEmpleado INT,
- idSucursalSalida INT,
- idSucursalLlegada INT,
- fechaSalida DATE,
- fechaLlegada DATE,
- fechaEsperadaLlegada DATE,
- valorAlquilerSemana DECIMAL(10, 2),
- valorAlquilerDia DECIMAL(10, 2),
- porcentajeDescuento DECIMAL(5, 2),
- valorCotizado DECIMAL(10, 2),
- valorPagado DECIMAL(10, 2),
- FOREIGN KEY (idVehiculo) REFERENCES Vehículo(idVehiculo),
- FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente),
- FOREIGN KEY (idEmpleado) REFERENCES Empleado(idEmpleado),
- FOREIGN KEY (idSucursalSalida) REFERENCES Sucursal(idSucursal),
- FOREIGN KEY (idSucursalLlegada) REFERENCES Sucursal(idSucursal));

6. Usuario (

- idUsuario INT PRIMARY KEY AUTO_INCREMENT,
- idCliente INT,
- idEmpleado INT,
- username VARCHAR(50) UNIQUE,
- password VARCHAR(255),
- tipoUsuario ENUM('Cliente', 'Empleado'),
- FOREIGN KEY (idCliente) REFERENCES Cliente(idCliente),
- FOREIGN KEY (idEmpleado) REFERENCES Empleado(idEmpleado));

NORMALIZACIÓN DEL MODELO LÓGICO

“La normalización de datos es un proceso utilizado en la gestión y organización de bases de datos que consiste en estructurar los datos para reducir la redundancia y mejorar la integridad de los datos. Esto se logra dividiendo una base de datos en tablas más pequeñas y vincularlos mediante relaciones definidas. El objetivo principal es eliminar la duplicación de datos y garantizar que la base de datos sea más eficiente y fácil de mantener.

La normalización generalmente sigue una serie de pasos conocidos como formas normales, cada una con sus propias reglas y criterios. Algunas de las formas normales más comunes son:

1. ***Primera forma normal (1NF):** Elimina los grupos repetitivos y asegura que cada campo contiene sólo valores atómicos.*
2. ***Segunda forma normal (2NF)**: Asegura que cada tabla sólo contenga datos que se relacionan con la clave primaria, eliminando la dependencia parcial.*
3. ***Tercera forma normal (3NF):** Elimina las dependencias transitivas, garantizando que los campos no clave dependen solo de la clave primaria.”*

Para normalizar las tablas proporcionadas, debemos organizar los datos de manera que no haya redundancia innecesaria y se cumplan las formas normales. A continuación, te guiaré a través del proceso de normalización paso a paso.

Primera Forma Normal (1FN)
Una tabla está en 1FN si cumple con los siguientes requisitos:
1. Todos los campos deben contener valores atómicos (indivisibles).
2. Debe existir una clave primaria única para identificar cada fila.

Segunda Forma Normal (2FN)
Una tabla está en 2FN si:
1. Está en 1FN.
2. Todos los atributos no clave dependen completamente de la clave primaria.

Tercera Forma Normal (3FN)
Una tabla está en 3FN si:
1. Está en 2FN.
2. Todos los campos no clave son dependientes de la clave primaria, de toda la clave y no de otros campos no clave.

Las tablas proporcionadas están diseñadas según principios de normalización de bases de datos, asegurando que cada tabla cumple con las formas normales adecuadas. Aunque se han sugerido ajustes para mejorar la estructura y reducir la redundancia, el diseño actual ya está normalizado, garantizando la eficiencia y la integridad de los datos en el sistema.

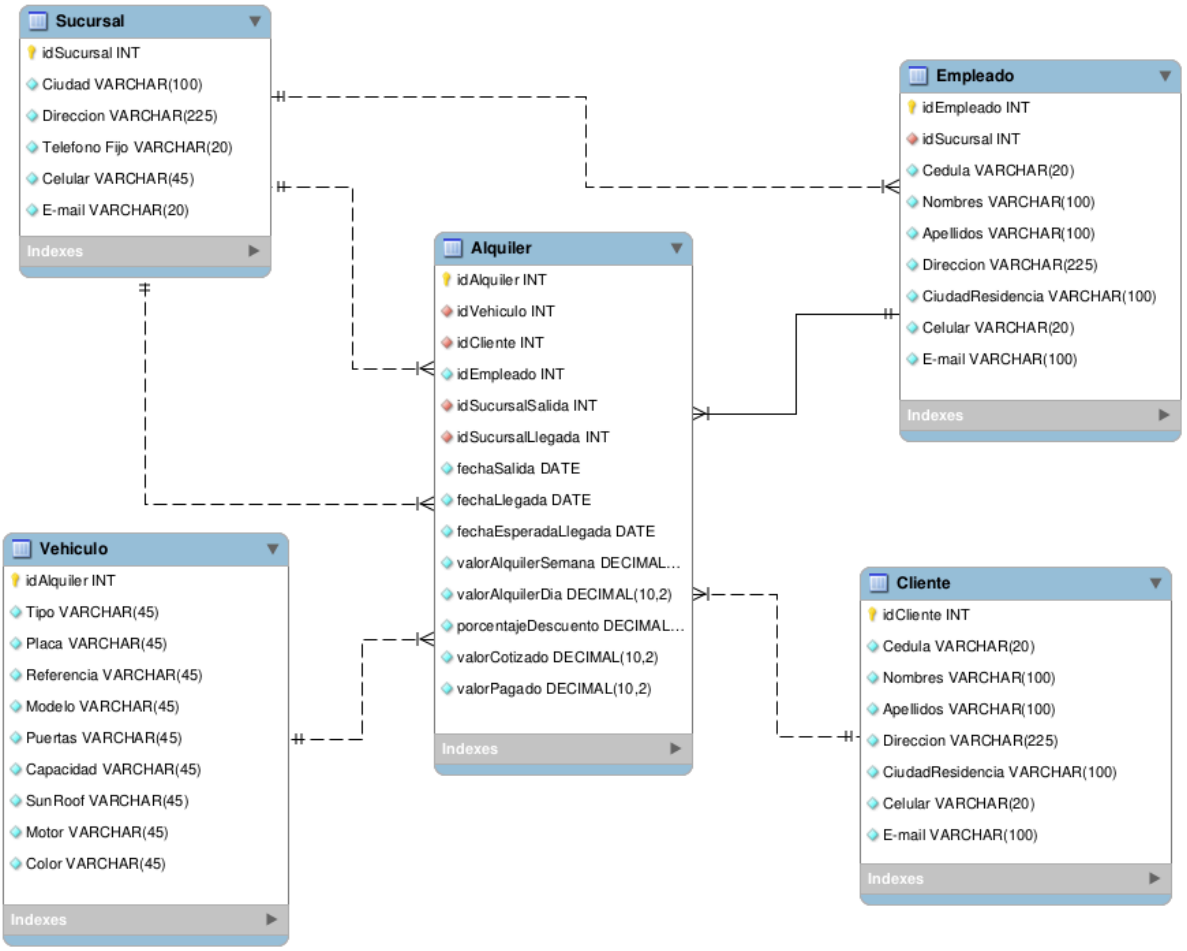
MODELO FÍSICO:

Los modelos de datos físicos proporcionan información detallada que ayuda a los administradores y desarrolladores de bases de datos a implementar la lógica empresarial en una base de datos física. Estos modelos ofrecen atributos adicionales no especificados en un modelo de datos lógico, como disparadores, procedimientos almacenados y tipos de datos. Dado que asignan los elementos de datos a una base de datos real, los modelos de datos físicos deben cumplir con las restricciones específicas de la plataforma, como las convenciones de nomenclatura y el uso de palabras reservadas.

El siguiente diagrama muestra el modelo de datos físico para Auto-Rent:

DIAGRAMA Entidad Relación (E-R)

Un diagrama entidad relación (también conocido como diagrama ER o diagrama ERD o simplemente ERD) muestra cómo interactúan las entidades (personas, objetos y conceptos). Estos modelos conceptuales de datos ayudan a desarrolladores y diseñadores a visualizar las relaciones entre elementos clave del software.



Relaciones

- Sucursal (1) <--> (N) Empleado
- Sucursal (1) <--> (N) Alquiler (Salida)
- Sucursal (1) <--> (N) Alquiler (Llegada)
- Empleado (1) <--> (N) Alquiler
- Cliente (1) <--> (N) Alquiler
- Vehículo (1) <--> (N) Alquiler

TRIGGERS

Un trigger, también conocido como disparador (Por su traducción al español) es un conjunto de sentencias SQL las cuales se ejecutan de forma automática cuando ocurre algún evento que modifique a una tabla. Pero no me refiero a una modificación de estructura, no, me refiero a una modificación en cuando a los datos almacenados, es decir, cuando se ejecute una sentencia INSERT, UPDATE o DELETE.

A diferencia de una función o un store procedure, un trigger no puede existir sin una tabla asociada.

1. ***Trigger para Actualizar el Valor Pagado en Alquileres:***

```
DELIMITER //


CREATE TRIGGER ActualizarValorPagado

BEFORE INSERT ON Alquiler

FOR EACH ROW

BEGIN

    DECLARE valor_total DECIMAL(10, 2);


    -- Calcular el valor total pagado (por ejemplo, basado en ValorCotizado y
PorcentajeDescuento)

    SET valor_total = NEW.ValorCotizado - (NEW.ValorCotizado *
(NEW.PorcentajeDescuento / 100));


    -- Asignar el valor calculado a ValorPagado

    SET NEW.ValorPagado = valor_total;

END //


DELIMITER ;
```

Explicación del Trigger:

- **BEFORE INSERT ON Alquiler**: Indica que este trigger se activa antes de insertar un nuevo registro en la tabla Alquiler.
- **FOR EACH ROW**: Especifica que el trigger se ejecuta para cada fila afectada por la operación (en este caso, inserción).
- **DECLARE valor_total DECIMAL(10, 2);**: Declara una variable local valor_total para almacenar el cálculo del valor pagado.
- **SET valor_total = ...**: Calcula el valor total pagado, usando el nuevo registro (NEW) que está siendo insertado.
- **SET NEW.ValorPagado = valor_total;**: Asigna el valor calculado a la columna ValorPagado del nuevo registro.

Este trigger asegura que cada vez que se inserte un nuevo registro en la tabla Alquiler, el campo ValorPagado se calcule automáticamente según la lógica definida.

2. ***Trigger para Registrar Operaciones de Modificación en Empleados:***

```
CREATE TABLE Empleado_Auditoria (
    OperacionID INT AUTO_INCREMENT PRIMARY KEY,
    EmpleadoID INT,
    Operacion VARCHAR(100),
    FechaOperacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //

CREATE TRIGGER RegistrarModificacionEmpleado
AFTER UPDATE ON Empleado
FOR EACH ROW
BEGIN
    INSERT INTO Empleado_Auditoria (EmpleadoID, Operacion)
    VALUES (OLD.ID, CONCAT('Actualización de datos de empleado ID ', OLD.ID));
END //
```

DELIMITER ;

Explicación del Trigger:

- **AFTER UPDATE ON Empleado**: Indica que este trigger se activa después de que se actualice un registro en la tabla Empleado.
- **FOR EACH ROW**: Especifica que el trigger se ejecuta para cada fila afectada por la operación (en este caso, actualización).
- **INSERT INTO Empleado_Auditoria ...**: Inserta un registro en la tabla Empleado_Auditoria cada vez que se actualice un registro en Empleado.
- **OLD.ID**: Se refiere al valor antiguo del campo ID de Empleado, antes de la actualización.

Este trigger registra automáticamente en la tabla Empleado_Auditoria cada vez que se actualiza un registro en Empleado, proporcionando un registro histórico de todas las modificaciones realizadas.

Consideraciones Finales

- *Los triggers son poderosos pero deben usarse con precaución para no afectar el rendimiento de la base de datos.*
- *Asegúrate de probar los triggers en un entorno de desarrollo antes de implementarlos en producción.*
- *Los ejemplos proporcionados son específicos para MySQL y pueden variar ligeramente en otros sistemas de gestión de bases de datos.*

EVENTOS:

Un *evento* no es más que una tarea la cual se ejecuta de forma automática en un momento previamente programado. Si eres un usuarios Linux puedes ver los eventos cómo los *cron jobs* .

Los *eventos* nos permitirán a nosotros cómo administradores de base de datos programar ciertas tareas las cuales queremos que se ejecuten de forma periódica o en un momento en concreto, de tal manera que podamos automatizar ciertos procesos.

1. **Evento para Actualizar el Estado de Alquileres:**

```
CREATE EVENT ActualizarEstadoAlquileres

ON SCHEDULE EVERY 1 DAY

STARTS CURRENT_TIMESTAMP

DO

    UPDATE Alquiler

    SET Estado = 'Vencido'

    WHERE FechaEsperadaLlegada < CURDATE() AND Estado != 'Finalizado';
```

Explicación del Evento:

- **CREATE EVENT ActualizarEstadoAlquileres**: Crea un nuevo evento llamado ActualizarEstadoAlquileres.
- **ON SCHEDULE EVERY 1 DAY**: Especifica que el evento se ejecuta todos los días (EVERY 1 DAY).
- **STARTS CURRENT_TIMESTAMP**: Indica que el evento comienza a ejecutarse desde el momento actual (CURRENT_TIMESTAMP).
- **DO ...**: Define las acciones que el evento debe realizar.
- **UPDATE Alquiler ...**: Actualiza la tabla Alquiler.
- **SET Estado = 'Vencido'**: Establece el estado de los alquileres como "Vencido".
- **WHERE FechaEsperadaLlegada < CURDATE() AND Estado != 'Finalizado'**: Condiciones para actualizar los registros: la fecha esperada de llegada debe ser anterior a la fecha actual (CURDATE()) y el estado no debe ser "Finalizado".

Este evento se ejecutará automáticamente todos los días y actualizará los alquileres cuya fecha esperada de llegada haya pasado, marcándolos como "Vencido".

2. **Evento para Generar Reporte de Empleados:**

```
CREATE EVENT GenerarReporteEmpleados

ON SCHEDULE EVERY 1 WEEK

STARTS CURRENT_TIMESTAMP + INTERVAL 1 DAY
```

DO

INSERT INTO ReporteEmpleados (SucursalID, EmpleadosActivos, FechaGeneracion)

SELECT e.SucursalID, COUNT(e.ID) AS EmpleadosActivos, CURDATE() AS FechaGeneracion

FROM Empleado e

WHERE e.Estado = 'Activo'

GROUP BY e.SucursalID;

Explicación del Evento:

- **CREATE EVENT GenerarReporteEmpleados**: Crea un nuevo evento llamado GenerarReporteEmpleados.
- **ON SCHEDULE EVERY 1 WEEK**: Especifica que el evento se ejecuta cada semana (EVERY 1 WEEK).
- **STARTS CURRENT_TIMESTAMP + INTERVAL 1 DAY**: Indica que el evento comienza a ejecutarse un día después del momento actual (CURRENT_TIMESTAMP), para asegurar que se ejecute en un día específico de la semana.
- **DO ...**: Define las acciones que el evento debe realizar.
- **INSERT INTO ReporteEmpleados ...**: Inserta datos en la tabla ReporteEmpleados.
- **SELECT ...**: Consulta para generar el reporte. En este caso, cuenta el número de empleados activos (Estado = 'Activo') por cada SucursalID y lo inserta en la tabla ReporteEmpleados.
- **GROUP BY e.SucursalID**: Agrupa los resultados por SucursalID.

Este evento se ejecutará automáticamente cada semana, generando un reporte de empleados activos por sucursal en la tabla ReporteEmpleados.

Consideraciones Finales

- *Asegúrate de tener los permisos adecuados para crear eventos en tu base de datos.*
- *Los eventos son útiles para automatizar tareas frecuentes pero deben ser diseñados y probados cuidadosamente.*
- *Los ejemplos proporcionados son específicos para MySQL. Si estás utilizando otro sistema de gestión de bases de datos, los comandos y la sintaxis pueden variar.*

Estos eventos te permitirán automatizar tareas como la actualización de estados o la generación de informes, basados en la lógica y la estructura de datos que hemos discutido previamente.

USUARIOS Y ACCESOS

1. ***Usuario con Accesos Completos:***

Este usuario tendrá acceso completo a todas las tablas y operaciones en la base de datos.

```
CREATE USER 'usuario_admin'@'localhost' IDENTIFIED BY 'contraseña_admin';
GRANT ALL PRIVILEGES ON *.* TO 'usuario_admin'@'localhost';
FLUSH PRIVILEGES;
```

Explicación:

- CREATE USER 'usuario_admin'@'localhost' IDENTIFIED BY 'contraseña_admin';: Crea un usuario llamado usuario_admin con la contraseña contraseña_admin.
- GRANT ALL PRIVILEGES ON *.* TO 'usuario_admin'@'localhost';: Concede todos los permisos (ALL PRIVILEGES) sobre todas las bases de datos y tablas (*.*) al usuario usuario_admin desde localhost.
- FLUSH PRIVILEGES;: Recarga los privilegios para que los cambios surtan efecto inmediatamente.

2. ***Usuario con Accesos Limitados***

Este usuario tendrá acceso solo a ciertas tablas y operaciones específicas en la base de datos.

```
CREATE USER 'usuario_restringido'@'localhost' IDENTIFIED BY 'contraseña_restringido';

GRANT SELECT, INSERT, UPDATE ON basededatos.Alquiler TO 'usuario_restringido'@'localhost';

GRANT SELECT ON basededatos.Cliente TO 'usuario_restringido'@'localhost';

FLUSH PRIVILEGES;
```

Explicación:

- CREATE USER 'usuario_restringido'@'localhost' IDENTIFIED BY 'contraseña_restringido';: Crea un usuario llamado usuario_restringido con la contraseña contraseña_restringido.
- GRANT SELECT, INSERT, UPDATE ON basededatos.Alquiler TO 'usuario_restringido'@'localhost';: Concede permisos de SELECT, INSERT y UPDATE sobre la tabla Alquiler en la base de datos basededatos al usuario usuario_restringido desde localhost.

- GRANT SELECT ON basededatos.Cliente TO 'usuario_restringido'@'localhost';: Concede permisos de SELECT sobre la tabla Cliente en la base de datos basededatos al usuario usuario_restringido desde localhost.
- FLUSH PRIVILEGES;: Recarga los privilegios para que los cambios surtan efecto inmediatamente.

Notas Adicionales:

- *Asegúrate de reemplazar basededatos con el nombre real de tu base de datos.*
- *Puedes ajustar los permisos (SELECT, INSERT, UPDATE, DELETE, etc.) y las tablas según los requisitos específicos de acceso para cada usuario.*
- *Es una buena práctica limitar los permisos de los usuarios según las operaciones que realmente necesiten realizar para mejorar la seguridad de la base de datos.*

REFERENCIAS:

https://dev.mysql.com/blog-archive/mysql-8-0-when-to-use-utf8mb3-over-utf8mb4/

https://www.linkedin.com/in/rub%C3%A9n-ortiz-1913932b7/