

PROYECTO FINAL - BASES DE DATOS

CARRENTALX

Por: Enrique Reyes Baute

Índice

Índice.....	2
Introducción.....	3
Sobre el Proyecto de CarRentalX.....	3
Análisis del Enunciado.....	3
Modelo Conceptual.....	4
Entidades y Atributos del sistema.....	4
Relaciones y Cardinalidad del sistema.....	5
Modelo Relacional.....	6
Las Tablas y sus Datos.....	6
Relaciones entre las Tablas.....	8
Implementación en MySQL.....	9
Inserción de Datos en las Tablas.....	11
Consultas Propuestas.....	11
Consulta Simple.....	11
Consulta con uso de Filtros.....	12
Consulta con uso de Funciones.....	12
Consulta con JOIN.....	13
Consultas de Modificación de las Tablas.....	13
Ampliación de la Base de Datos.....	14
Vistas y Triggers.....	15
Vistas en MySQL.....	15
Triggers en MySQL.....	16
Pruebas y Validación.....	16
Conclusión.....	17
Anexos.....	18

Introducción

Sobre el Proyecto de CarRentalX

En este informe se presenta el contenido detallado sobre el proyecto final de la asignatura de Bases de Datos de 1º de DAM. El objetivo y finalidad de este proyecto ha sido el de investigar, practicar y desarrollar un modelo de bases de datos simple pero funcional sobre un sistema de alquiler de coches.

En este proyecto se han volcado las diferentes etapas y procesos a los que debe someterse una base de datos antes de hacerse. Con modelos conceptuales se vuelca toda la información que se requeriría almacenar y se plasma en un diagrama para tenerlo limpio y claro, con un modelo relacional se establece las relaciones entre los elementos y finalmente el modelo físico es la culminación de los modelos predecesores para que finalmente en este modelo no haya problemas mayores.

Finalmente con una base de datos lista se le pueden empezar a realizar inserciones, consultas simples, complejas, la aplicaciones de triggers e incluso la modificación a posteriori de la base de datos.

Análisis del Enunciado

Este documento y proyecto sigue las pautas establecidas del repositorio original dado por la docente de la asignatura. Una estructura en directorios “recursos”, “scripts” y “capturas”, en los que se almacenan sus respectivos datos, tales como los scripts generados por el alumno, capturas del proyecto o recursos externos que haya podido aprovechar. De esta manera, siguiendo el orden establecido de los diferentes puntos y objetivos a conseguir, se ha generado y terminado el proyecto.

Modelo Conceptual

Como bien sabemos el modelo conceptual es una forma simplificada y carente de tecnicismos que nos ayuda a ver con claridad qué es lo que necesitamos.

Para este proyecto se ha elegido de las tres posibles opciones el de CarRentalX, un sistema de alquiler de coches. Por lo que hay que aclarar los *tipos de entidades, atributos y relaciones* que se van a tener en cuenta.

Entidades y Atributos del sistema

- **Vehículos:** Es la Entidad principal en la cual nos basamos
 - ID: Un identificador
 - Matricula
 - Kilometraje
 - Tipo de vehículo: Un familiar, una pickup, un compacto...
 - Estado: Disponible, reservado, alquilado o en mantenimiento
 - Precio diario: Cuanto se cobra por el alquiler diario
 - ID Sucursal: La sucursal en la que se encuentra
- **Cliente:** La Entidad que realiza el mayor número de acciones
 - ID: Un identificador
 - Nombre
 - Apellido
 - Email: Primera forma de contacto
 - Teléfono: Segunda forma de contacto
 - Tipo de Identificador: DNI o pasaporte, para saber si son locales o extranjeros.
 - Identificación gubernamental: Su número de dni o pasaporte
 - Licencia de Conducción
- **Reserva:** La Entidad que mantiene relacionadas al resto
 - ID: Un identificador
 - Fecha de Inicio
 - Fecha de Finalización
 - Estado de la Reserva: El cómo se encuentra la reserva, si está tramitando, cancelada, confirmada o finalizada.
 - Id del Cliente: El cliente que ha realizado la reserva
 - Id del Vehiculo: El vehiculo que quiere el cliente

- **Empleado:** La Entidad que trabaja en la empresa
 - ID: Un identificador
 - Nombre
 - Apellido
 - Email: Primera forma de contacto
 - Teléfono: Segunda forma de contacto
 - ID Sucursal: Sucursal en la que trabaja

- **Sucursal:** La Entidad que gestiona los vehículos y tiene empleados
 - ID: Un identificador
 - Nombre
 - Dirección
 - Email: Primera forma de contacto
 - Teléfono: Segunda forma de contacto

Relaciones y Cardinalidad del sistema

Ya tenemos los elementos del sistema, pero debemos saber cómo se relacionan entre ellos para un correcto funcionamiento.

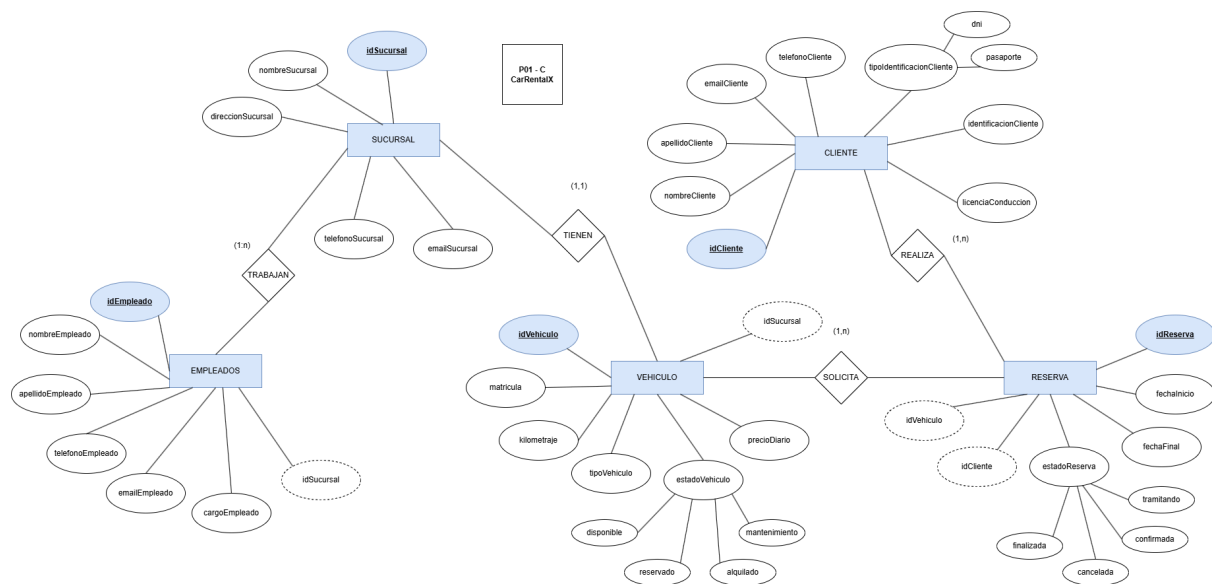
- **CLIENTE - RESERVA:** Un cliente *realiza* una o más reservas, ya que podría solicitar varios vehículos.

- **RESERVA - VEHÍCULO:** Una reserva solicita uno o más vehículos, pues varios clientes podrían querer reservar el mismo vehículo.

- **SUCURSAL - VEHÍCULO:** Una sucursal tiene un vehículo.

- **EMPLEADO - SUCURSAL:** Uno o más empleados trabajan en una sucursal.

Todos estos datos quedarían plasmados en un diagrama tal como el siguiente **[La imagen está en capturas]**:



Ahora con el modelo conceptual preparado y listo, solo nos queda empezar con el modelo relacional.

Modelo Relacional

Aquí en el modelo relacional nos toca traducir el modelo conceptual a un sistema en el cual se definirán tablas con columnas y datos. Esto no tiene mayor misterio, ya que con la interfaz gráfica de *MySQL Workbench* podemos hacer una representación al pie de la letra de nuestro modelo conceptual de manera sencilla, incluso corrigiendo posibles errores que surjan a la hora de implementar.

Las Tablas y sus Datos

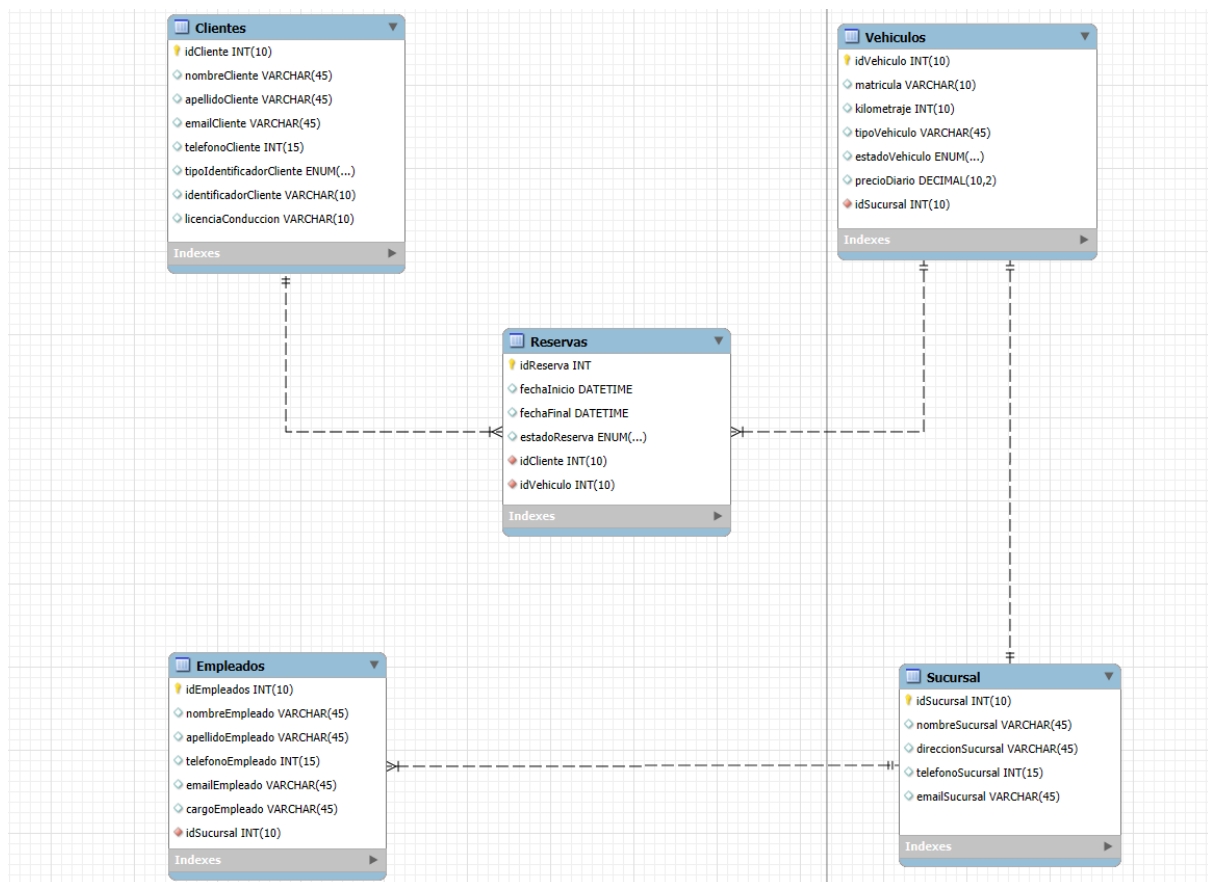
- **Vehículos:** La Tabla de los Vehículos
 - **idVehiculo:** INT (10)
 - Matricula: VARCHAR (10)
 - Kilometraje: INT (10)
 - TipoVehiculo: VARCHAR (45)
 - EstadoVehiculo: ENUM ('Disponible', 'reservado', 'alquilado', 'mantenimiento')
 - PrecioDiario: DECIMAL (10, 2)
 - **idSucursal:** INT (10)

- **Cientes:** Tabla de los Clientes
 - **idCliente:** INT (10)
 - NombreCliente: VARCHAR (45)
 - ApellidoCliente: VARCHAR (45)
 - EmailCliente: VARCHAR (45)
 - TeléfonoCliente: INT (15)
 - TipoidentificadorCliente: ENUM ('DNI','Pasaporte')
 - IdentificadorCliente: VARCHAR (10)
 - LicenciaConducción: VARCHAR (10)
- **Reserva:** Tabla de las Reservas
 - **idReserva:** INT (10)
 - FechaInicio: DATETIME
 - FechaFinal: DATETIME
 - EstadoReserva: ENUM ('tramitando', 'cancelada', 'confirmada', 'finalizada')
 - **IdCliente:** INT (10)
 - **IdVehiculo:** INT (10)
- **Empleado:** La Entidad que trabaja en la empresa
 - **idEmpleado:** INT (10)
 - NombreEmpleado: VARCHAR (45)
 - ApellidoEmpleado: VARCHAR (45)
 - EmailEmpleado: VARCHAR (45)
 - TeléfonoEmpleado: INT (15)
 - **idSucursal:** INT (10)
- **Sucursal:** La Entidad que gestiona los vehículos y tiene empleados
 - **idSucursal:** INT (10)
 - NombreSucursal: VARCHAR (45)
 - DirecciónSucursal: VARCHAR (45)
 - EmailSucursal: VARCHAR (45)
 - TeléfonoSucursal: INT (15)

Relaciones entre las Tablas

- Cada **cliente** puede hacer una o más **reservas**, por lo que nos queda una relación de uno a muchos [1:n].
- Cada **reserva** puede solicitar uno o más **vehículos**, por lo que nos queda una relación de uno a muchos [1:n].
- Cada **sucursal** tiene un único tipo de **vehículo**, por lo que tenemos uno a uno [1:1].
- Cada **empleado** (pueden ser varios) trabajan en una **sucursal**, por lo que queda como una relación muchos a uno [n:1].

De esta forma al plasmar los datos en el MySQL Workbench nos queda el siguiente Modelo Relacional **[La imagen está en capturas]**:



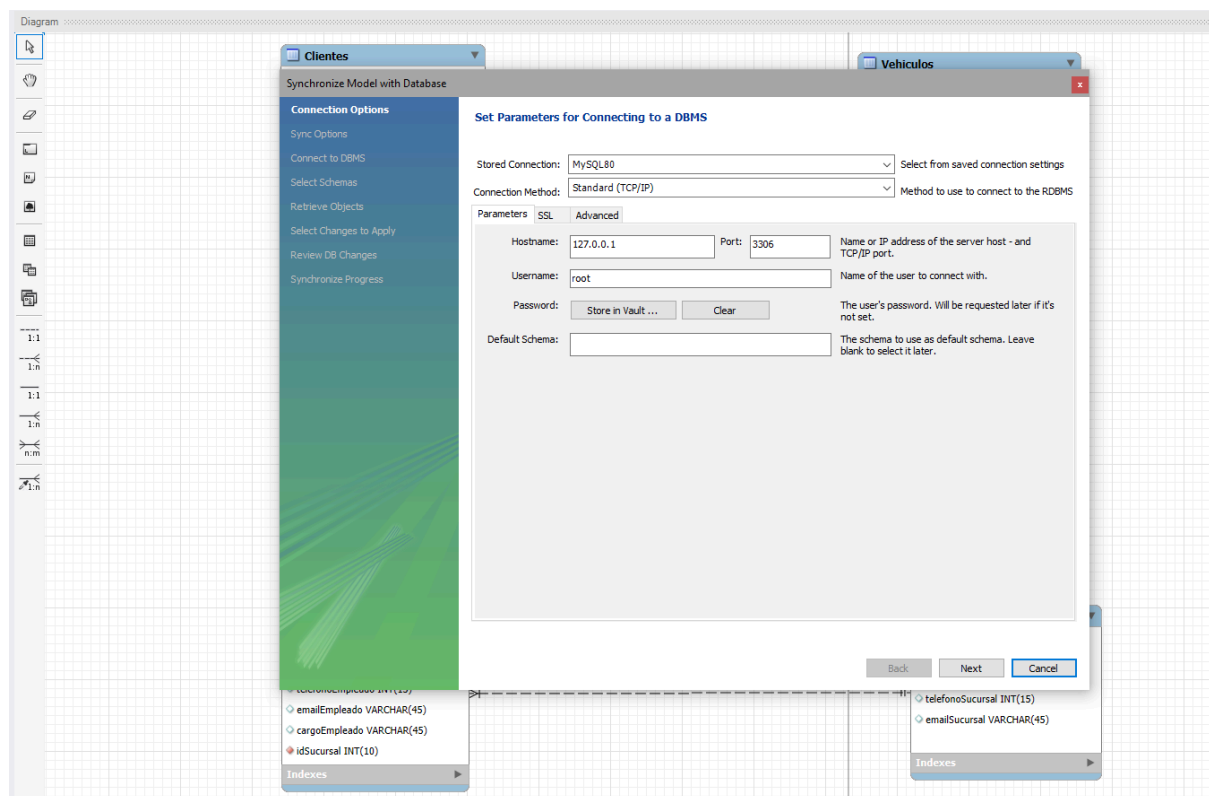
Los diferentes datos que se han empleado, deberían prevenir ciertos problemas, como podría ser los tipo DECIMAL en los precios diarios del

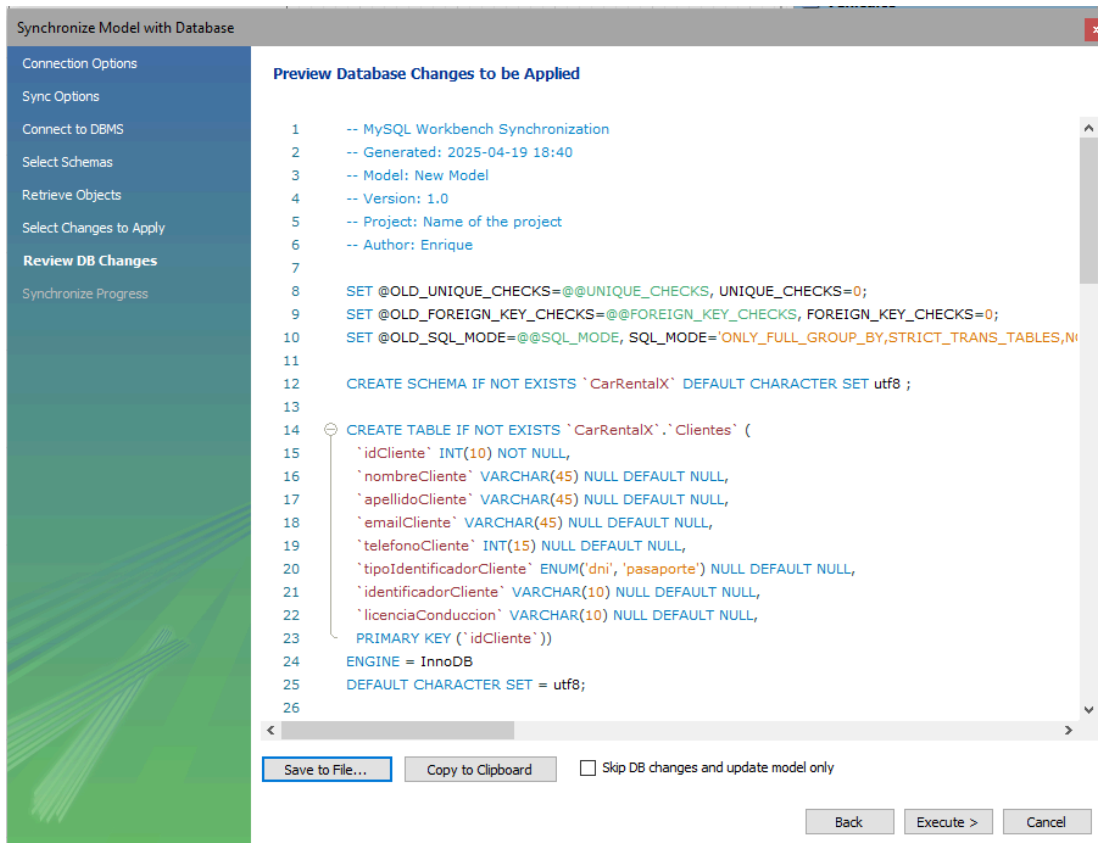
alquiler para los costos decimales y que no salgan redondeados, o los ENUM con los posibles estados en los que se pueda encontrar el vehículo o del tipo de cliente para una posible aplicación de descuento según su procedencia.

Implementación en MySQL

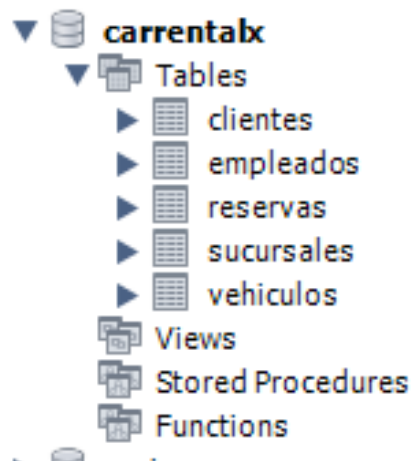
Finalmente, con el modelo relacional preparado, podemos empezar a generar nuestro modelo físico utilizando las herramientas que proporciona *MySQL Workbench*.

Con el modelo relacional en pantalla seleccionamos la opción de *Sincronizar el Modelo*, el cual nos lleva a un asistente el cual nos guía y permite crear e incluso generar un código SQL de la base de datos.





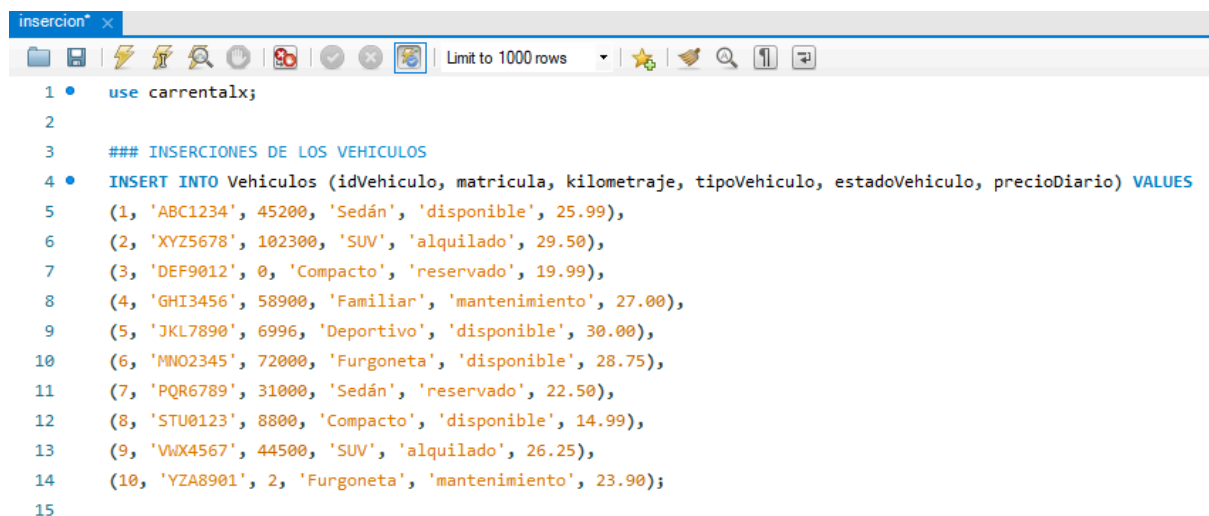
Finalmente, al entrar como root en el sistema, nos sale en la banda izquierda toda la estructura que planeamos y generamos:



Inserción de Datos en las Tablas

Con la base de datos recién creada podemos finalmente abastecer las tablas. Para ellos debemos generar un script que lo haga, mediante la palabra clave *INSERT INTO* podemos decirle en qué tabla queremos introducir los datos y cuales de estos serán.

A continuación una muestra de cómo se ha hecho la inserción de la tabla vehículos. Si se desea indagar más, el script está en su respectiva carpeta con el nombre de **insercion.sql**



```
1 • use carrentalx;
2
3   ### INSERCIONES DE LOS VEHICULOS
4 • INSERT INTO Vehiculos (idVehiculo, matricula, kilometraje, tipoVehiculo, estadoVehiculo, precioDiario) VALUES
5   (1, 'ABC1234', 45200, 'Sedán', 'disponible', 25.99),
6   (2, 'XYZ5678', 102300, 'SUV', 'alquilado', 29.50),
7   (3, 'DEF9012', 0, 'Compacto', 'reservado', 19.99),
8   (4, 'GHI3456', 58900, 'Familiar', 'mantenimiento', 27.00),
9   (5, 'JKL7890', 6996, 'Deportivo', 'disponible', 30.00),
10  (6, 'MNO2345', 72000, 'Furgoneta', 'disponible', 28.75),
11  (7, 'PQR6789', 31000, 'Sedán', 'reservado', 22.50),
12  (8, 'STU0123', 8800, 'Compacto', 'disponible', 14.99),
13  (9, 'VWX4567', 44500, 'SUV', 'alquilado', 26.25),
14  (10, 'YZA8901', 2, 'Furgoneta', 'mantenimiento', 23.90);
15
```

De la misma manera se abastecen el resto de tablas. Ahora nos toca consultar la información que se ha introducido en las tablas.

Consultas Propuestas

La información que se ha introducido en las tablas puede ser o bien simple y fácil de manejar, o bien complicada y amplia. Por lo que es ideal saber cuales son las formas de manejar y consultar las diferentes tablas para poder acceder a la información deseada de manera eficiente.

Consulta Simple

Si deseamos ver el contenido total de una tabla, para tener una idea general de los datos que tratamos o para detección de errores, se deberá emplear la siguiente sentencia: *SELECT * FROM TABLA_DESEADA*

Resultando en algo como lo siguiente:

Result Grid		Filter Rows:		Edit:		Export/Import:	
	idVehiculo	matricula	kilometraje	tipoVehiculo	estadoVehiculo	precioDiario	
▶	1	ABC1234	45200	Sedán	disponible	25.99	
	2	XYZ5678	102300	SUV	alquilado	29.50	
	3	DEF9012	0	Compacto	reservado	19.99	
	4	GHI3456	58900	Familiar	mantenimiento	27.00	
	5	JKL7890	6996	Deportivo	disponible	30.00	
	6	MNO2345	72000	Furgoneta	disponible	28.75	
	7	PQR6789	31000	Sedán	reservado	22.50	
	8	STU0123	8800	Compacto	disponible	14.99	
	9	VWX4567	44500	SUV	alquilado	26.25	
	10	YZA8901	2	Furgoneta	mantenimiento	23.90	
*	NULL	NULL	NULL	NULL	NULL	NULL	

Consulta con uso de Filtros

Si deseamos ver contenido de una forma concreta o buscar ciertos datos y que se nos muestran de maneras concretas de una tabla, el uso de palabras clave como las consultas con WHERE, ORDER BY, BETWEEN o LIKE son ideales para filtrar el contenido.

idVehiculo	kilometraje
3	0
10	2
5	6996
8	8800
7	31000
9	44500
1	45200
NULL	NULL

Consulta con uso de Funciones

Si queremos no buscar un dato de la tabla, sino una recapitulación, una media o un agrupamiento de los diferentes datos, podemos usar las funciones de SQL para esta tarea. Con las palabras clave de SUM, AVG o GROUP BY para conseguirlo.

	EXTRANJEROS
▶	7

Consulta con JOIN

A veces nos interesa ver la información que relaciona diferentes tablas al mismo tiempo. Para eso uso JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN... Nos viene perfecto, pues nos permite combinar información de varias tablas en una misma consulta.

	idReserva	idVehiculo	precioDiario	diasAlquiler	totalRecaudado
▶	1	3	19.99	4	79.96

Consultas de Modificación de las Tablas

Finalmente podemos encontrar que los datos introducidos inicialmente o bien, o tienen que ser modificados, o el dato de la tabla en sí es insuficiente y tiene que cambiar. Para lograr esto debemos usar una consulta con UPDATE para cambiar la información que se introdujo en la tabla, o si queremos cambiar el tipo de dato de la tabla a un tipo de dato distinto hay que hacerlo con UPDATE precedido por ALTER TABLE, para alterar de esta manera a la tabla en sí misma.

```
56      ### Cambiar el tipo de Vehiculo a uno más interesante
57 •    UPDATE Vehiculos SET tipoVehiculo = 'Máquina del Tiempo'
58      WHERE idVehiculo = 2;
59
```

Result Grid Filter Rows: <input type="text"/> Edit: Export/Import:						
	idVehiculo	matricula	kilometraje	tipoVehiculo	estadoVehiculo	precioDiario
▶	1	ABC1234	45200	Sedán	disponible	25.99
▶	2	XYZ5678	102300	Máquina del Tiempo	alquilado	29.50

Ampliación de la Base de Datos

Tras una serie de consultas se ha llegado a la conclusión de que la base de datos que tenemos montada requiere el almacenaje de un nuevo tipo de entidad, siendo este el de los **pagos** que se realizan al finalizar una reserva.

Por lo que tras preparar todos los atributos que se requieren para esta nueva tabla, se ha generado el siguiente script de la creación de la tabla y de las inserciones convenientes de las reservas anteriores.

```
### CREAR LA NUEVA TABLA DE PAGOS
> CREATE TABLE pagos (
    idPago INT(10) NOT NULL AUTO_INCREMENT,
    idReserva INT(10) NOT NULL,
    fechaPago DATETIME NOT NULL,
    formaPago VARCHAR(45),
    cantidadPago DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (idPago),
    FOREIGN KEY (idReserva) REFERENCES reservas(idReserva)
);

### INSERCIONES DE LA NUEVA TABLA DE PAGOS
INSERT INTO pagos (idPago, idReserva, fechaPago, formaPago, cantidadPago) VALUES
(DEFAULT, 1, '2025-03-05 11:00:00', 'tarjeta', 100.00),
(DEFAULT, 2, '2025-03-15 10:30:00', 'transferencia', 125.00),
(DEFAULT, 3, '2025-04-07 12:00:00', 'efectivo', 140.00),
(DEFAULT, 5, '2025-04-20 09:45:00', 'sueños y esperanzas', 160.00),
(DEFAULT, 10, '2025-04-21 12:15:00', 'bizum', 180.00);
```

Vistas y Triggers

Vistas en MySQL

Cuando está haciendo múltiples consultas a la base de datos, rápidamente se encontrará con el problema de tener que estar escribiendo largas sentencias para hacer selecciones a información de forma reiterada. Para solucionar esto se pueden usar las **vistas**, siendo estas unas tablas virtuales que sirven como atajo para aligerar las largas sentencias en las búsquedas de SQL.

```
## Vista para ver los clientes con reservas de vehículos disponibles
CREATE VIEW vista_clientes_reservas_vehiculos_disponibles AS
SELECT
  c.idCliente, c.nombreCliente, c.apellidoCliente,
  r.idReserva, r.estadoReserva, r.fechaInicio, r.fechaFinal,
  v.idVehiculo, v.tipoVehiculo, v.estadoVehiculo
FROM clientes c
JOIN reservas r ON c.idCliente = r.idCliente
JOIN vehiculos v ON r.idVehiculo = v.idVehiculo
WHERE v.estadoVehiculo = 'disponible';
```

Ahora si hacemos un `SELECT * FROM [NOMBRE DE LA TABLA]`, nos saldrá la información que hemos especificado en la creación de la vista.

	idCliente	nombreCliente	apellidoCliente	idReserva	estadoReserva	fechaInicio	fechaFinal	idVehiculo	tipoVehiculo	estadoVehiculo
▶	4	Miguel	Ruiz	4	tramitando	2025-04-08 08:00:00	2025-04-12 10:00:00	1	Sedán	disponible
	10	Daniel	Ortega	11	cancelada	2025-04-02 10:00:00	2025-04-03 10:00:00	1	Sedán	disponible
	2	Carlos	Gómez	2	finalizada	2025-03-10 09:30:00	2025-03-15 09:30:00	5	Deportivo	disponible
	1	Laura	Martínez	12	finalizada	2025-04-04 10:00:00	2025-04-06 10:00:00	5	Deportivo	disponible
	6	Javier	López	6	cancelada	2025-03-20 16:00:00	2025-03-22 10:00:00	6	Furgoneta	disponible
	11	Sophie	Müller	13	tramitando	2025-04-10 10:00:00	2025-04-14 10:00:00	6	Furgoneta	disponible
	7	Patricia	Fernández	8	confirmada	2025-04-10 13:00:00	2025-04-13 10:00:00	8	Compacto	disponible
	14	Rubén	Molina	18	finalizada	2025-04-05 08:00:00	2025-04-08 08:00:00	8	Compacto	disponible

Basta con hacer las tablas de las consultas más frecuentes que solemos hacer y tendríamos el trabajo de la búsqueda de información de la base de datos más simplificada.

Triggers en MySQL

Cuando se desea automatizar o crear ciertos métodos de seguridad en la base de datos los triggers son ideales. Siendo estos una porción de código que se activa al hacer determinadas acciones, como por ejemplo actualizar el salario de un empleado de forma automática si trabajas en X sucursal.

```
## Trigger para asignar automaticamente el salario cor
CREATE TRIGGER trigger_salario_sucursal_tenerife
BEFORE INSERT ON Empleados
FOR EACH ROW
> BEGIN
>   IF NEW.idSucursal = 2 THEN
>       SET NEW.salario = 33000.99;
-   END IF;
- END $$
```

Pruebas y Validación

A lo largo de todo el proyecto se ha ido probando y verificando la integridad de lo que se ha hecho. En los scripts del proyecto se puede generar, insertar, consultar, modificar... todos los elementos que hemos visto a lo largo del proyecto sin ningún problema, llegando a probar que todo lo que se ha dado en la asignatura ha sido plasmado en el proyecto.

Si se desea replicar los pasos dados en el informe, en la carpeta están todos los scripts y se deberían ejecutar en el orden en el que se presenta en el propio informe.

Conclusión

La creación de una base de datos desde cero es un trabajo arduo y en el que se debe ser muy meticuloso y seguir los pasos que hemos seguido para poder llegar a obtener un resultado satisfactorio.

Con RentalCarX hemos buscado los elementos necesarios y los atributos que se deberían tener en cuenta en el modelo, plasmándolo en el modelo conceptual. Posteriormente en el modelo relacional se entra un poco en los tecnicismo y se vuelca todo lo creado en el conceptual en la interfaz gráfica de MySQL Workbench para, en una última instancia, generar el modelo físico.

La inserción de elementos con el uso de scripts ha sido exitosa y hemos podido visualizar esta información insertada en las tablas mediante diferentes consultas y de diferentes formas, pudiendo extraer la información de distintas maneras.

Llegamos a ampliar el modelo de la base de datos a posteriori de su creación con una nueva tabla con sus propios datos y con sus relaciones con otras tablas ya preexistentes.

Finalmente al tener cierta soltura con la base de datos hemos hallado las consultas más usuales que solemos hacer, por lo que con la ayuda de las vistas hemos creado una forma de ahorrarnos bastante tiempo entre consultas, para finalmente culminar con la creación de triggers, unos bloques de código que se activan al hacer ciertas acciones con la base de datos, esto con el fin de que nos ayuden a automatizar y prevenir ciertos errores.

Anexos

Todo el contenido que se ha generado en el informe y del proyecto están en el repositorio de GitHub del siguiente [link](#).

Como se ha especificado en las instrucciones del proyecto, la información está dividida en los directorios establecidos.