Cars Online Rental App

Coursepaper

Grin Polina

2 year student

C# major

2024

Content

Buisness description **3**

OLTP description **3**

OLAP and ETL description**6**

Loading data from CSV files**7**

Buisness description

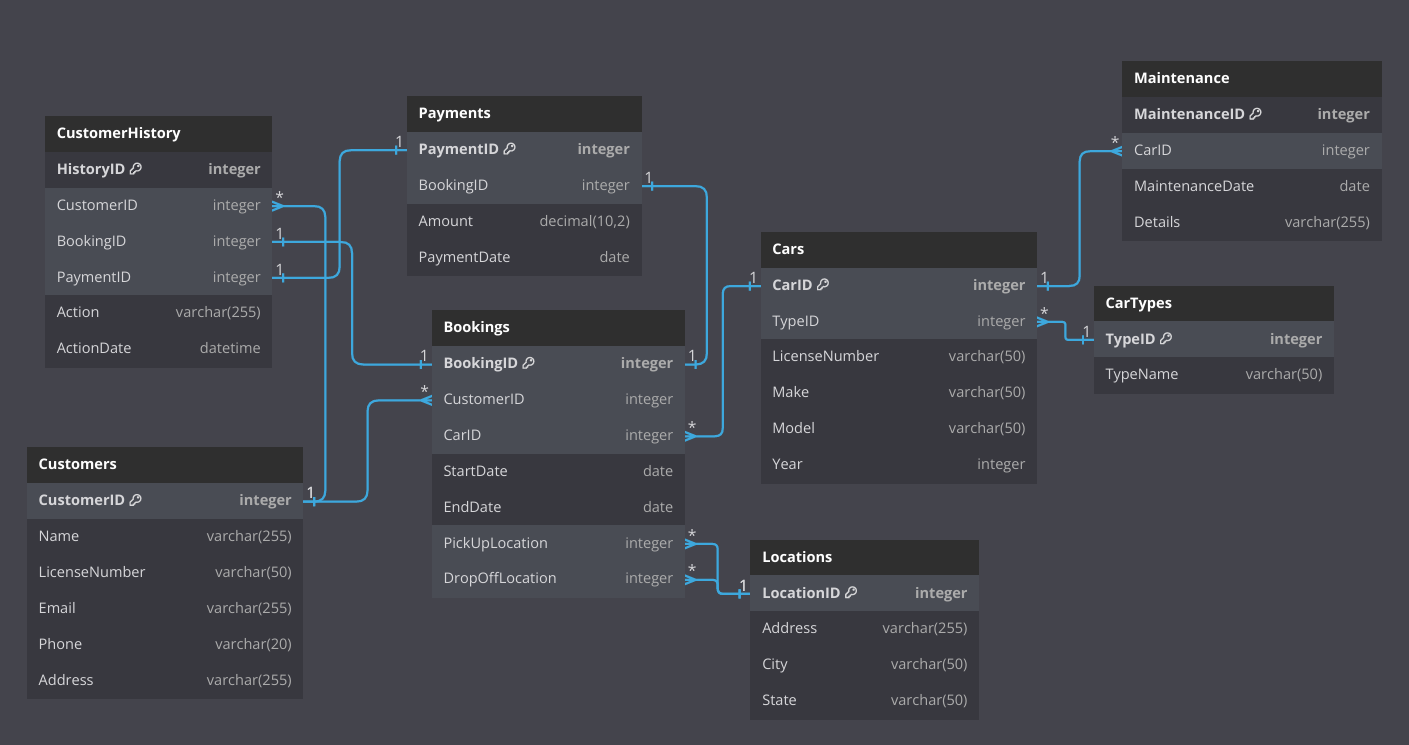
This database is designed for managing a car rental service. It helps store and manage information about customers, cars, car types, locations, bookings, payments, and maintenance activities. The database supports both operational transactions (OLTP) for everyday business operations and analytical processing (OLAP) for reporting and analysis.

The database performs the following functions:

* Stores and manages customer information.
* Keeps track of different car types and individual car details.
* Records booking details, including customer information, car details, start and end dates, and pickup/drop-off locations.
* Processes and records payments for bookings.
* Manages maintenance records for cars.
* Maintains a history of customer actions for auditing and reporting purposes.

OLTP description

ER-diagram



Tables:

* **Customers**: Stores customer information such as name, email, phone number, address, and driving license number. Each customer can make multiple bookings.
* **CarTypes**: Stores information about different car types, such as Sedan and SUV.
* **Cars**: Stores information about individual cars, including the car type, license number, make, model, and year. Each car can have multiple maintenance records and bookings.
* **Locations**: Stores information about different rental locations, including address, city, and state.
* **Bookings**: Stores booking information, including the customer, car, start and end dates, and pickup/drop-off locations. Each booking can generate multiple payments and have related history records.
* **Payments**: Stores payment information for bookings, including the amount paid and the payment date.
* **Maintenance**: Stores maintenance records for cars, including the maintenance date and details.
* **CustomerHistory**: Stores a history of customer actions, including booking and payment action

**Relationships between tables:**

* One customer can make several bookings (one-to-many relationship between **Customers** and **Bookings**).
* One customer can have several history records (one-to-many relationship between **Customers** and **CustomerHistory**).
* One car can be associated with multiple bookings and maintenance records (one-to-many relationship between **Cars** and **Bookings/Maintenance**).
* One booking can have one payment (one-to-one relationship between **Bookings** and **Payments**).
* One booking and payment can have one history record (one-to-one relationship between **Bookings/Payments** and **CustomerHistory**).
* One location can be used in several bookings (one-to-many relationship between **Locations** and **Bookings**).
* Several cars can have one car type (one-to-many relationship betweeen **CarTypes** and **Cars**).

Indexes on tables:

* **idx\_customer\_email**: Index on Customers(Email) to speed up searches by email.
* **idx\_car\_license**: Index on Cars(LicenseNumber) to speed up searches by car license number.
* **idx\_booking\_dates**: Index on Bookings(StartDate, EndDate) to speed up searches by booking dates.
* **idx\_payment\_date**: Index on Payments(PaymentDate) to speed up searches by payment date.

Roles and permissions:

* **RentalAdmin**: Full access (SELECT, INSERT, UPDATE, DELETE) to all tables.
* **RentalUser**: Read-only access (SELECT) to all tables.

Queries:

* Customer Information Query (**GetCustomerInformation**)

Retrieves information about a specific customer.

* Available Cars Query (**GetAvailableCars**)

Finds cars that are not booked within a specific date range.

* Booking Details Query (**GetBookingDetails**)

Retrieves details of a specific booking, including customer name and car details

Functions:

* **CreateBooking**

Creates a new booking and updates the customer history.

* **ProcessPayment**

Processes a payment and updates the customer history.

* **UpdateBookingTrigger**

Trigger to update customer history when a booking is updated.

OLAP and ETL description

Tables:

* **DimCustomer**: Dimension table for customers. Stores information such as customer ID, name, email, phone, address, and license number, along with start and end dates and a flag indicating if the record is current.
* **DimCarType**: Dimension table for car types. Stores car type ID and name.
* **DimCar**: Dimension table for cars. Stores car ID, type ID, license number, make, model, and year.
* **DimDate**: Dimension table for dates. Stores various date parts such as year, quarter, month, day, and week, along with a flag indicating if the date is a weekend.
* **DimLocation**: Dimension table for locations. Stores location ID, address, city, and state.
* **FactBookings**: Fact table for bookings. Stores booking ID, customer key, car key, start and end date keys, and pickup/drop-off location keys.
* **FactPayments**: Fact table for payments. Stores payment ID, booking key, amount, and payment date key.

ETL process:  
 The ETL process involves extracting data from the OLTP schema, transforming it as needed, and loading it into the OLAP schema.

Description:

* Extract: Data is extracted from the OLTP tables.
* Transform: Necessary transformations are applied, such as converting dates to dimensional keys, and handling slowly changing dimensions (SCD) to maintain historical accuracy.
* Load: The transformed data is loaded into the OLAP dimension and fact tables, ensuring that the data warehouse is populated with accurate and comprehensive information for analysis.

Steps:

1. Extract and transform customer data to maintain historical changes and populate the DimCustomer table.
2. Load car types into the DimCarType table and details into the DimCar.
3. Load location details into the DimLocation table.
4. Generate and load date dimensions into the DimDate table.
5. Extract booking data and load it into the FactBookings table.
6. Extract payment data and load it into the FactPayments table.

Loading data from CSV files

The process of loading data from CSV files into the database is handled by a custom function (in file Load-data.sql) designed to ensure data integrity and efficiency. Below is an explanation of how this function works:

Purpose:  
The load\_data function is designed to automate the process of loading data from CSV files into the database tables. It creates temporary tables to hold the data temporarily, validates the data, and then inserts it into the main tables, ensuring no duplicates or errors.

Steps involved:

1. **Creation of Temporary Tables**:

* For each target table (e.g., Customers, CarTypes, Cars, etc.), a corresponding temporary table is created.
* These temporary tables have the same structure as the main tables but exist only for the duration of the data loading process.

1. **Loading Data into Temporary Tables**:

* The CSV data is loaded into these temporary tables using the COPY command, which is highly efficient for bulk data loading.
* The COPY command reads the CSV file and populates the temporary table with its contents.

1. **Data Validation and Insertion into Main Tables**:

* After loading the data into temporary tables, the function performs validation to ensure data integrity.
* For example, it checks for duplicate entries and ensures that no empty rows are inserted.
* Using a LEFT JOIN and WHERE condition, it inserts data from the temporary tables into the main tables only if the records do not already exist in the main tables.

1. **Handling Foreign Keys and Relationships**:

* The function handles foreign keys and relationships by ensuring that references to other tables are correctly mapped.
* It ensures that the foreign key constraints are maintained during the data insertion process.

1. **Cleanup**:

* Once the data has been successfully inserted into the main tables, the temporary tables are dropped.
* This cleanup step ensures that no temporary data remains in the database, maintaining its cleanliness and efficiency.