# **Final Report on Car Rental DBMS Design and Implementation**

**Table of contents:**

1. Introduction
2. Application Description
3. Requirements

* Functional
* Non-functional

1. Schema Design
2. Normalization

* Functional Dependencies
* 3NF and BCNF Verification

1. Query Development

* Simple Queries
* Advanced Queries

1. Application Implementation

* Unix Shell Integration
* View Design

1. Concluding Remarks

## **1. Introduction**

This report details the design and implementation of a database management system (DBMS) for a car rental service. The DBMS optimizes record-keeping for customer details, vehicle inventory, rental transactions, and related payments and maintenance. By ensuring efficient, secure, and scalable operations, the system addresses the data management challenges faced by car rental businesses.

## **2. Application Description**

The car rental DBMS encompasses the following:

* Managing customer data: personal details, licenses, and contact information.
* Tracking vehicle inventory: details such as make, model, year, and status.
* Recording rental transactions: linking customers and cars with transaction details.
* Handling payments: capturing amounts, dates, and methods.
* Monitoring maintenance activities: documenting service details and costs.

## **3. Requirements**

### Functional Requirements

* Manage customer data, including personal and license details.
* Maintain vehicle inventory, including availability and maintenance history.
* Track rental transactions and link them to customers and cars.
* Handle payments associated with rentals.
* Generate reports based on rentals, payments, and maintenance.

### Non-functional Requirements

* Scalability: The system must perform efficiently as data volume increases.
* Security: Ensure data integrity and protection, especially for sensitive customer and payment details.

## **4. Schema Design**

### Core Tables and Their Purpose

1. Customers
   * Stores customer details with Customer\_ID as the primary key.
   * Alternate key: License\_Number.
2. Cars
   * Maintains vehicle inventory with Car\_ID as the primary key.
   * Candidate keys: License\_Plate, VIN.
3. Rental\_Transactions
   * Links customers and cars with Rental\_ID as the primary key.
   * Tracks start/end dates, total costs, and rental status.
4. Payments

* Tracks payments linked to rentals with Payment\_ID as the primary key.

1. Car\_Maintenance

* Records maintenance activities for vehicles with Maintenance\_ID as the primary key.

1. Locations

* Stores branch information with Location\_ID as the primary key.

## **5. Normalization**

### *5.1 Functional Dependencies*

* Customers: Customer\_ID → First\_Name, Last\_Name, Email, Phone\_Number, License\_Number, Address
* Cars: Car\_ID → Make, Model, Year, License\_Plate, VIN, Daily\_Rental\_Price, Availability\_Status
* Rental\_Transactions: Rental\_ID → Customer\_ID, Car\_ID, Rental\_Start\_Date, Rental\_End\_Date, Total\_Cost, Status
* Payments: Payment\_ID → Rental\_ID, Payment\_Amount, Payment\_Date, Payment\_Method
* Car\_Maintenance: Maintenance\_ID → Car\_ID, Maintenance\_Date, Description, Maintenance\_Cost
* Locations: Location\_ID → Location\_Name, Address, Phone\_Number

### *5.2 3NF Verification*

* Each table is in 3NF as all non-key attributes are fully functionally dependent on the primary key, and there are no transitive dependencies.

### *5.3 BCNF Verification*

* Customers Table: Both Customer\_ID and License\_Number are candidate keys.
* Cars Table: Car\_ID, License\_Plate, and VIN are candidate keys.
* Other tables were confirmed as BCNF since every determinant is a candidate key.

## **6. Query Development**

### *6.1 Simple Queries*

* Distinct Last Names of Customers:
* SELECT DISTINCT Last\_Name FROM Customers ORDER BY Last\_Name;
* Available Cars by Make:
* SELECT Make, COUNT(\*) AS NumberOfCars FROM Cars WHERE Availability\_Status ='Available' GROUP BY Make;

### *6.2 Advanced Queries*

* Find Customers with No Active Rentals:
* SELECT First\_Name, Last\_Name FROM Customers c WHERE EXISTS (SELECT 1 FROMRental\_Transactions r WHERE r.Customer\_ID = c.Customer\_ID) AND NOT EXISTS (SELECT1 FROM Rental\_Transactions r WHERE r.Customer\_ID = c.Customer\_ID AND r.Status ='Active');
* Average Maintenance Cost by Location:
* SELECT l.Location\_Name, COUNT(cm.Maintenance\_ID) AS Maintenance\_Count, AVG(cm.Maintenance\_Cost) AS Avg\_Cost FROM Locations l JOIN Cars c ON l.Location\_ID = c.Location\_ID JOIN Car\_Maintenance cm ON c.Car\_ID = cm.Car\_ID GROUP BYl.Location\_Name;

**7. Application Implementation**

### *7.1 Unix Shell Integration*

* A Unix shell script-based interface allows users to interact with the database via a menu-driven UI.
* Key operations include creating, populating, querying tables, and viewing results.

### *7.2 Views*

* **Customer Rental Summary View:** CREATE VIEW Customer\_Rental\_Summary AS SELECT c.First\_Name, c.Last\_Name, COUNT(r.Rental\_ID) AS Total\_Rentals FROM Customers c LEFT JOIN Rental\_Transactions r ON c.Customer\_ID = r.Customer\_ID GROUP BY c.First\_Name, c.Last\_Name;
* **Cars Maintenance View:** CREATE VIEW Cars\_Maintenance\_View AS SELECT c.Make, c.Model, cm.Description, cm.Maintenance\_Date, cm.Maintenance\_Cost FROM Cars c JOIN Car\_Maintenance cm ONc.Car\_ID = cm.Car\_ID;

## **8. Conclusion**

The car rental DBMS was successfully designed and implemented with the following highlights:

* All tables comply with 3NF and BCNF.
* Queries cover a range of requirements, from simple to advanced, enabling operational insights.
* The Unix shell integration provides a text-based UI, ensuring ease of use for database operations.
* The Unix shell also provides algorithms for transforming tables into 3NF and BCNF