SYMBIOSIS INSTITUTE OF TECHNOLOGY (SIT)

Constituent of Symbiosis International (Deemed University), Pune

(Established under Section 3 of the UGC Act of 1956 vide notification number F-9-12/2001-U-3 of the Government of india)
Re-Accredited by NAAC with 'A' Grade

॥वसुधैव कुटुम्बकम्॥

A DBMS Project Report on OLA CABS DATABASE

Submitted by

Sakshi Joshi(20070122199)
Pratham Suryawanshi(20070122201)
Akshansh Sourabh(20070122202)
Aditya Khaladkar(21070122509)

UNDER THE GUIDANCE OF

Dr. Pooja Bagane

Department of Computer Science
SYMBIOSIS INSTITUTE OF TECHNOLOGY,PUNE

Index

Introduction

Ola is India's largest mobility platform and one of the world's largest ride-hailing companies, serving more than 250 cities in India, Australia, New Zealand, and the United Kingdom. The Ola app connects customers to drivers and a wide range of vehicles, including bikes, auto-rickshaws, metered taxis, and cabs, allowing hundreds of millions of consumers and over 1.5 million driver-partners convenience and transparency.

This service requires a massive amount of data to run smoothly, thus, necessitating the use of a Database Management System. The Ola Cab database organises all the necessary information about the customer who books the ride, the driver partners, employees, vehicles and booking. The correct source and destination information, driver and customer names and contact information, fare price, OTP, booking id (CRN number), and vehicle details are essential for the effective operation of the Ola Cab application and constitute the foundation of the database.

Problem Statement

Through our database, where we will keep the information of the passenger who books the journey, the driver partners, staff, vehicles, and booking, we want to comprehend Ola Cabs' data organization issues and present a clear image of their data and services. Earlier, it was fairly challenging to hunt out for appropriate transportation when we were in a rush and at a reasonable rate. Ola eliminates this issue and links you to what is best for you in a fairly low amount of time. Additionally, it helped fleet operators do business and improved drivers' earnings by making it easier for them to reach customers. The goal of our project is to build a database of application that address this issue and comprehend it at a fundamental level.

While designing this database, we have not considered the Ola rental services.

Functional Requirements

- Every user needs to be authenticated. The user must enter personal information such as their name (first and last), phone number (on which they receive an OTP), date of birth (which establishes their age), and an optional password. After registering, each user is given a unique UserID.
- A user can either be a customer or a driver partner.
- Customer information, such as email addresses, favourite destinations, emergency contacts, profile type (personal or corporate) is maintained for every customer.
- When a customer books a trip, he is expected to give specific details, like the name of the rider (whether it is for him or for someone else) and their contact details to contact him directly. Along with the ride type (city trip or outstation), the customer should also specify the preferred vehicle type while making a reservation.
- The customer who books the trip makes payment the payment can either be done through ola wallet, cash, card, or UPI. In case of card payment card details like card number, card holder name, CVV, expiry date is asked and in case of UPI, UPI ID is required.
- The customer provides all the trip details for a booking, including the start and destination addresses and any stops. The customer should also specify day, date and time of the trip. Each trip has a trip booking id or CRN, and the customer is given an estimate of the cost. A partner can be a fleet operator who owns the vehicles and hires the drivers, or it can be the driver himself.
- Every partner must enter their preferred communication language, complete address (street, city, state, pin code) Aadhar card number, Pan card number, bank account number, and account holder name. The joining date is also stored.
- A driver drives the vehicle and confirms the trip booking. Each driver is assigned a driver id, and his ratings are saved. He must provide his name, licence number and update his status (offline- not ready to accept booking, online- ready to accept bookings) Once the driver who drives vehicle confirms the trip booking, the customer is given the precise cost and OTP.
- The details of vehicles like vehicle number, vehicle type, vehicle brand company, model, colour, chassis number, RC number and insurance number are to be provided by vehicle owners. Vehicle owner can either be a driver or fleet operator.
- Vehicle owners must provide information such as vehicle number, vehicle type, vehicle brand company, model, colour, chassis number, RC number, and insurance number.

Entities, Relationships and Attributes

Entities	**Relationships**	**Entities**
Customer	A customer can book N trips by specifying details of his requirement like preferred vehicle type, ride type and who the ride is for ie. Ride user (Name and Phone). A customer may or may not book trip based on the details.	Trip
Customer & Trip (Aggregation)	Customer who books the trip must make a payment.	Payment
Driver & Vehicle (Aggregation)	A driver who drives the vehicle can confirm N trips. As a result of confirmation exact cost and OTP is provided to the customer. Each trip may or may not be confirmed.	Trip.
Driver	A driver must drive a vehicle	Vehicle
Fleet operator	One fleet operator hires N drivers. A driver may or may not be hired by the fleet operator.	Driver
Fleet Operator	A fleet operator or a driver can own multiple vehicles. A vehicle may or may not be owned by fleet operator or driver.	Vehicles

Attributes:

I) **User**

- UserID
- Name (Composite)
    - First Name
    - Last Name
- Phone no.
- DOB
- Age (Derived)
- Password (Optional)
- Customer

II) **Customer**

- Profile Type
- Email id
- Favourite Locations (Multivalued)
- Emergency Contacts (Multivalued)

III) **Trip**

- CRN
- Start Address
- Destination Address
- Stops (Multivalued)
- Approximate Cost
- Date Time (Composite)
    - Date
    - Time
    - Day (Derived)

IV) **Books (Relationship)**

- Ride user (Composite)
    - Name
    - Phone No
- Ride Type
- Preferred Vehicle Type

V) **Payment**

- Payment Method (Composite)
  - Ola Wallet
  - Cash
  - Card (Composite)
    - Card Holder Name
    - Card No
    - CVV
    - Expiry date
  - UPI (Composite)
    - ID

VI) **Confirms (Relationship)**

- Cost
- OTP

VII) **Partner**

- Date of Joining
- Preferred Language,
- Bank Details (Composite)
  - Account Holder Name
  - Bank Account Number
- Address (Composite)
  - Street
  - City
  - State
  - Pin code
- Pan Card No
- Aadhar card No

VIII) **Fleet Operator**

- Operator Id

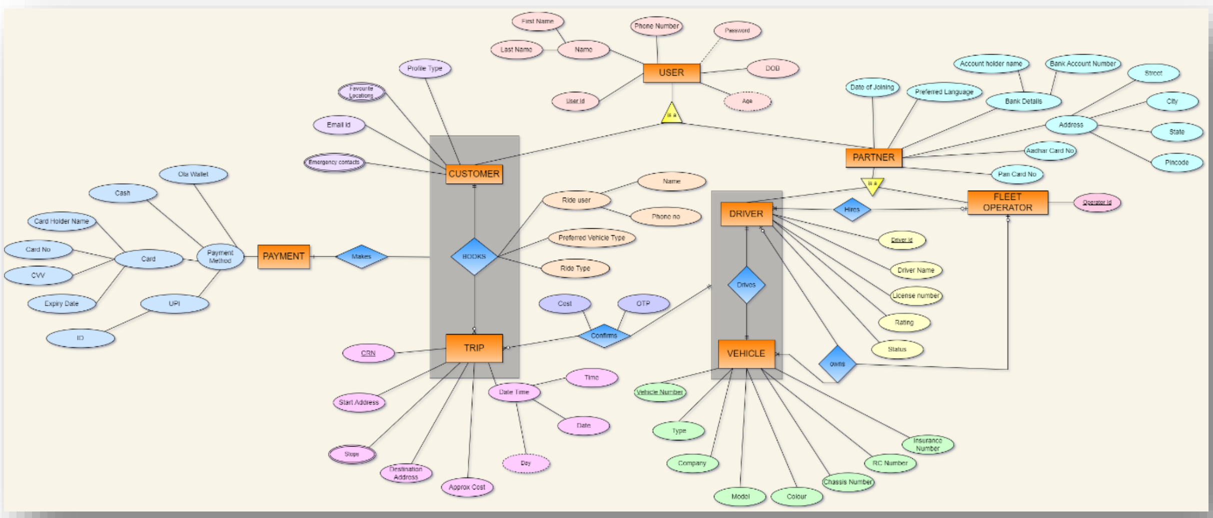
IX) **Driver**

- DriverID
- Driver Name
- License No
- Rating
- Status

X) Vehicle

- Vehicle No
- Type
- Company
- Model
- Colour
- Chassis No
- RC No
- Insurance No

E-R /EER Diagram



Relational Schema:

1. User(UserID, First Name, Last Name, Phone no., DOB, Password)
2. Customer(UserID, Profile Type, Email id)
3. Customer_Emergency_Contact(UserID, Contact no)
4. Customer_Favourite_Location(UserID, Location)
5. Books(UserID, TripID, Name, Phone No., Ride Type, Preferred Vehicle Type)
6. Trip(TripID, DriverID, VehicleNo, Start Address, Destination Address, Time, Date, Approx Cost)
7. Trip_Stops(TripID, Stops)
8. Confirms(TripID, DriverID, Vehicle No, Cost, OTP)
9. Driver(DriverID, UserID, OperatorID, Vehicle No, Driver Name, License No, Rating, Status)
10. Vehicle (Vehicle No, OperatorID, DriverID, Type, Company, Model, Colour, Chassis No, RC No, Insurance No)
11. Partner(UserID, Date of Joining, Preferred Language, Account Holder Name, Bank Account No, Street, City, State, Pin code, Pan Card No, Aadhar card No)
12. Fleet Operator(OpeartorID, UserID)
13. Payment(UserID, TripID, Ola wallet, UPI ID, Cash, Card Holder Name, Card No, CVV, Expiry date)

1. User

UserID	First Name	Last Name	Phone No.	DOB	Password

- Primary Key: UserID
- Foreign Key: NULL
- Candidate Key: UserID, First Name, Last Name
- Alternate Keys: First Name, Last Name

2. Customer

UserID	Profile Type	Email ID

- Primary Key: UserID
- Foreign Key: NULL
- Candidate Key: UserID, Email ID
- Alternate Keys: Email ID

3. Customer_Emergency_Contact

UserID	Contact No

- Primary Key: UserID
- Foreign Key: NULL
- Candidate Key: UserID, Contact no
- Alternate Keys: Contact no

4. Customer_Favourite_Location

UserID	Location

- Primary Key: UserID
- Foreign Key: NULL
- Candidate Key: UserID
- Alternate Keys: NULL

5. Books

UserID	TripID	Name	Phone No.	Ride Type	Preferred Vehicle Type

- Primary Key: UserID, TripID
- Foreign Key: NULL
- Candidate Key: UserID, Trip ID
- Alternate Keys: NULL

6. Trip

TripID	*DriverID*	*VehicleNo.*	S. Address	D. Address	Time	Date	App. Cost

- Primary Key: TripID
- Foreign Key: DriverID, VehicleNo
- Candidate Key: NULL
- Alternate Keys: NULL

7. Trip_Stops

<u>TripID</u>	Stops

- Primary Key: TripID
- Foreign Key: NULL
- Candidate Key: TripID
- Alternate Keys: NULL

8. Confirms

<u>TripID</u>	<u>DriverID</u>	<u>Vehicle No.</u>	Cost	OTP

- Primary Key: TripID, DriverID, Vehicle No
- Foreign Key: NULL
- Candidate Key: TripID
- Alternate Keys: NULL

9. Driver

<u>DriverID</u>	*UserID*	*OperatorID*	*Vehicle No*	Driver Name	License No.	Rating	Status

- Primary Key: Driver Id
- Foreign Key: UserID, OperatorID
- Candidate Key: DriverID, Driver Name, License Number.
- Alternate Keys: Driver Name, License Number

10. Vehicle

<u>Vehicle No.</u>	*OperatorID*	*Driver ID*	Type	Company	Model	Colour	Chassis No	RC No	Insurance No

- Primary Key: OperatorID, DriverID
- Foreign Key: NULL
- Candidate Key: Vehicle No
- Alternate Keys: NULL

11. Partner

<u>UserID</u>	DOJ	P. Lang	Acc Holder Name	Bank Ac.	Street	City	State	Pin Code	Pan Card No.	Aadhar No.

- Primary Key: UserID
- Foreign Key: NULL
- Candidate Key: UserID, Aadhar Card Number, PAN Card Number, Bank Acc Number
- Alternate Keys: Aadhar Card Number, PAN Card Number, Bank Acc Number.

12. Fleet Operator

<u>OperatorID</u>	<u>UserID</u>

- Primary Key: OperatorID, UserID
- Foreign Key: NULL
- Candidate Key: OperatorID, UserID
- Alternate Keys: NULL

13. Payment

<u>UserID</u>	<u>TripID</u>	Ola wallet	UPI ID	Cash	Card Holder Name	Card No.	CVV	Expiry date

- Primary Key: UserID, TripID
- Foreign Key: NULL
- Candidate Key: UserID, TripID
- Alternate Keys: NULL

Codd's Rules

- Information Rule: All information in the database is to be represented in one and only one way, namely by values in column positions within rows of *tables*.
    - This rule *is followed* in our database design. Our ola database is implemented in the form of tables.

- Guaranteed Access Rule: All data must be accessible. This rule is essentially a restatement of the fundamental requirement for *primary keys*.
    - This rule *is followed* in our database as every data value can be accessed logically using SQL Queries.

- Systematic Treatment of NULL values: *NULL values* in the database only correspond to missing, unknown or not applicable values.
    - This rule *is followed* in our database. The NULL value in the database implemented is data missing in our database implemented through SQL.

- Active Online Catalogue: Structure of database must be stored in an *online catalogue* which can be queried by authorized users.
    - This rule *is not followed* in our database as we have not created any online catalogue for our database. It is stored in individual machines.

- Comprehensive Data Sub-Language Rule: A database can only be accessed using a language having a linear syntax that supports data definition, data manipulation and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation
    - This rule *is followed* in our database. An SQL Client will be used to implement this database. Our database design supports data definition operations (including view definitions), data manipulation operations (update as well as retrieval), security and integrity constraints, and transaction management operations.

- View Updating Rule: Different views created for various purposes should be automatically updatable by the system.
    - This rule *is not followed* in our database. Although updating the view will update the table used for creating it, it is not recommended by most of the database. Hence this rule is not used in most of the databases.

- High level insert, update and delete rule: Relational Model should support *insert*, *delete*, *update* etc. Operations at each level of relations. Also, set operations like Union, intersection and minus should be supported.
    - This rule *is followed* in our database. High-level operations are allowed in our database and one single query can be used to update, insert or delete a set of records.

- Physical data independence: Any modification in the physical location of a table should not enforce modification at application level.
    - This rule *is followed* our database as Any change done would be absorbed by the mapping between the conceptual and internal levels in our SQL Client.

- Logical data independence: Any modification in logical or conceptual schema of a table should not enforce modification at application level. For example, merging of two tables into one should not affect application accessing it which is difficult to achieve.
    - This rule *is not followed* in our database. In an ideal scenario, this is difficult to achieve since all the logical and user views are tied togethso strongly that they will be almost the same.

- Integrity Independence: A database must be independent of the application that uses it. All its integrity constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.
    - This rule *is not followed* in our database.

- Distribution Independence: Distribution of data over various locations is not visible to end-users.
    - This rule *is not followed* in our database. Since we are using the MySQL community edition, this condition is violated.

- Non-Subversion Rule: If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.
    - This rule *is followed* in our database. Users can only access data using SQL and no other lower-level language in SQL. So, the query written in the database by the user will be converted in the same low-level syntax and not any other language which will change the data integrity.