1. Introduction

The Odyssey Database is designed to ensure data integrity and efficient management of operations. The design process begins with creating an Entity-Relationship (ER) diagram to visually represent entities, attributes, and relationships within the system.

Normalization is applied from the First Normal Form (1NF) to the Fifth Normal Form (5NF) to reduce redundancy and prevent anomalies. The resulting database schema defines the structure, specifying tables, columns, data types, and relationships through primary and foreign keys.

This documentation provides a clear guide to the ER diagram, normalization process, and the resulting database schema, facilitating effective data management and retrieval.

2. ER-Diagram

The Entity-Relationship (ER) diagram is a vital tool in the database design process, providing a visual representation of the database's structure. It illustrates the entities within the system, their attributes, and the relationships between them. The ER diagram helps in understanding the overall layout of the database and serves as a blueprint for creating the actual database schema.

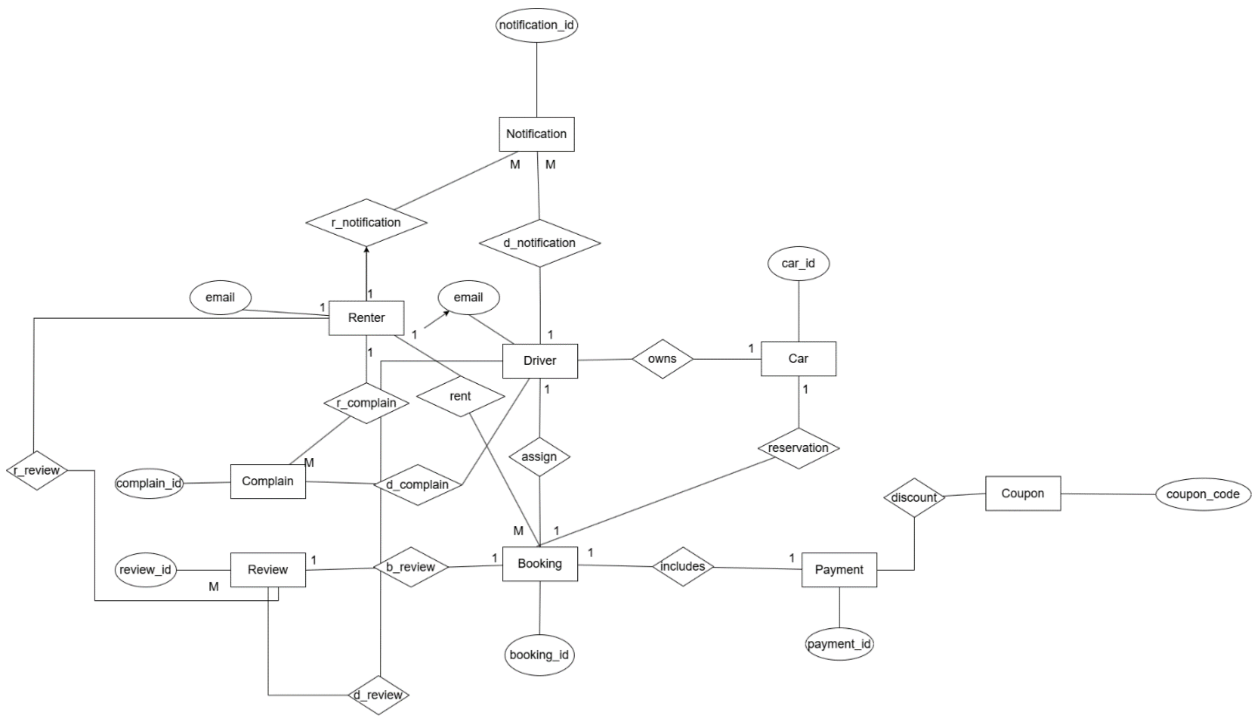


Figure 1:ER-Diagram

2.1 Entities and Attributes

1. **Renter**
   - Attributes: renter_id (Primary Key), email
2. **Driver**
   - Attributes: driver_id (Primary Key), email
3. **Notification**
   - Attributes: notification_id (Primary Key)
4. **Car**
   - Attributes: car_id (Primary Key)
5. **Complain**
   - Attributes: complain_id (Primary Key)
6. **Review**
   - Attributes: review_id (Primary Key)
7. **Booking**
   - Attributes: booking_id (Primary Key)
8. **Payment**
   - Attributes: payment_id (Primary Key)
9. **Coupon**
   - Attributes: coupon_code (Primary Key)

2.2 Relationships

1. **Renter - Notification**
   - Relationship: r_notification
   - Cardinality: One-to-Many (A renter can have many notifications, and a notification can be associated with many renters)
2. **Driver - Notification**
   - Relationship: d_notification
   - Cardinality: One-to-Many (A driver can have many notifications, and a notification can be associated with many drivers)
3. **Driver - Car**
   - Relationship: owns
   - Cardinality: One-to-One (A driver owns one car, and a car is owned by one driver)
4. **Renter - Complain**
   - Relationship: r_complain
   - Cardinality: One-to-Many (A renter can make many complaints, and a complaint is made by one renter)
5. **Driver - Complain**
   - Relationship: d_complain
   - Cardinality: One-to-Many (A driver can receive many complaints, and a complaint is associated with one driver)
6. **Renter - Review**

   - Relationship: r_review
   - Cardinality: One-to-Many (A renter can give many reviews, and a review is given by one renter)
7. **Driver - Review**
   - Relationship: d_review
   - Cardinality: One-to-Many (A driver can receive many reviews, and a review is associated with one driver)
8. **Booking - Review**
   - Relationship: b_review
   - Cardinality: One-to-Many (A booking can have many reviews, and a review is associated with one booking)
9. **Driver - Booking**
   - Relationship: assign
   - Cardinality: One-to-Many (A driver can be assigned many bookings, and a booking is assigned to one driver)
10. **Renter - Booking**
    - Relationship: rent
    - Cardinality: One-to-Many (A renter can make many bookings, and a booking is made by one renter)
11. **Car - Booking**
    - Relationship: reservation
    - Cardinality: One-to-Many (A car can have many bookings, and a booking is for one car)
12. **Booking - Payment**
    - Relationship: includes
    - Cardinality: One-to-One (A booking includes one payment, and a payment is for one booking)
13. **Coupon - Payment**
    - Relationship: discount
    - Cardinality: One-to-Many (A coupon can be used for many payments, and a payment can use one coupon)

The ER diagram helps identify the primary and foreign keys that establish relationships between the tables, ensuring referential integrity and efficient data retrieval. By analyzing the ER diagram, we can ensure that the database structure aligns with the real-world requirements of the agricultural management system.

Normalization is a systematic approach to organizing data in a database to minimize redundancy and ensure data integrity. It involves decomposing tables into smaller, well-structured tables without losing information, by applying a series of rules known as normal forms. Starting with the First Normal Form (1NF), which ensures atomicity and the elimination of repeating groups, the process moves through the Second Normal Form (2NF) to eliminate partial dependencies, and the Third Normal Form (3NF) to remove transitive dependencies. Further normalization includes Boyce-Codd Normal Form (BCNF) for stricter rules ensuring all determinants are candidate keys, Fourth Normal Form (4NF) to eliminate multi-valued dependencies, and Fifth Normal Form (5NF) to address join dependencies. This hierarchical process ensures that the database is efficient, consistent, and free from anomalies, facilitating robust data management and retrieval.

3.Normalization

Normalization is a systematic approach to organizing data in a database to minimize redundancy and ensure data integrity. It involves decomposing tables into smaller, well-structured tables without losing information, by applying a series of rules known as normal forms. Starting with the First Normal Form (1NF), which ensures atomicity and the elimination of repeating groups, the process moves through the Second Normal Form (2NF) to eliminate partial dependencies, and the Third Normal Form (3NF) to remove transitive dependencies. Further normalization includes Boyce-Codd Normal Form (BCNF) for stricter rules ensuring all determinants are candidate keys, Fourth Normal Form (4NF) to eliminate multi-valued dependencies, and Fifth Normal Form (5NF) to address join dependencies. This hierarchical process ensures that the database is efficient, consistent, and free from anomalies, facilitating robust data management and retrieval.

Normalization process

Step 1: Identify the Problem with the Current Table

Your initial review table has the following columns:

- review_id
- rating
- feedback

The table violates normalization rules, specifically:

- **1NF (First Normal Form)**: Ensures that the table contains atomic (indivisible) values and each column contains values of a single type.
- **2NF (Second Normal Form)**: Ensures that the table is in 1NF and all non-key columns are fully dependent on the primary key.
- **3NF (Third Normal Form)**: Ensures that the table is in 2NF and all columns are directly dependent on the primary key, not on other non-key columns.

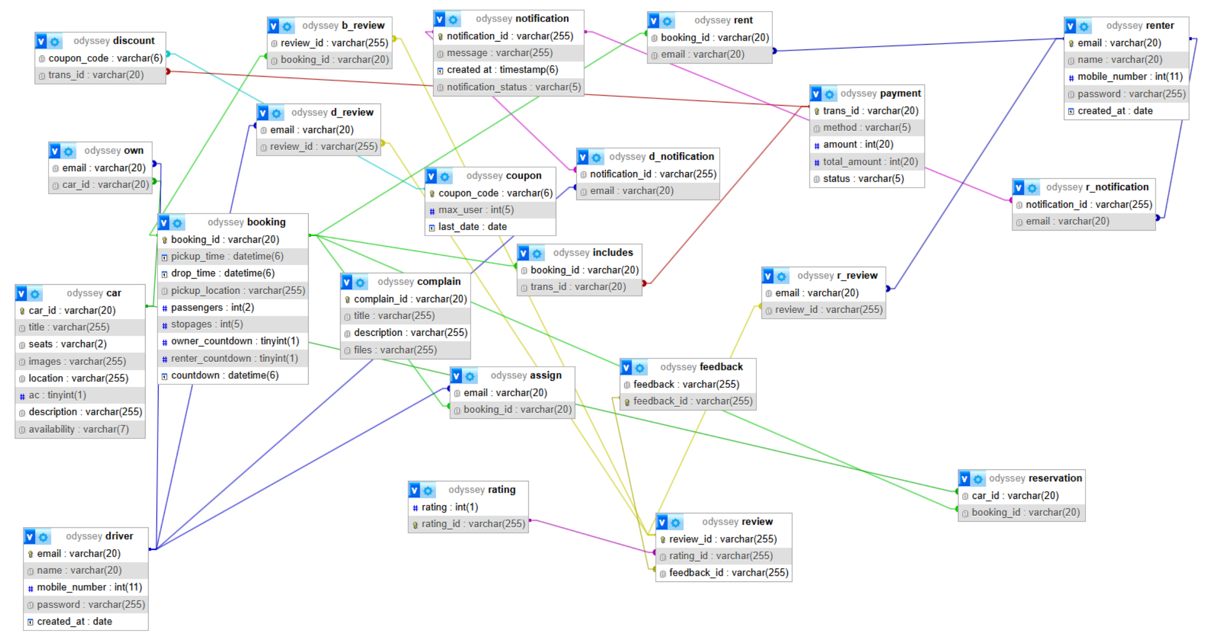
Step 2: Apply 1NF (First Normal Form)

Ensure Atomicity: Ensure each column contains atomic values, which means no repeating groups or arrays. The original table already seems to follow this, but we can further refine it.

Step 3: Apply 2NF (Second Normal Form)

Remove Partial Dependencies: Create separate tables to ensure that non-key columns are fully functionally dependent on the primary key.

4. Database Schema of Odessey after Normalization



5. Conclusion

Creating a database schema from an Entity-Relationship (ER) diagram using normalization is crucial for designing efficient and reliable databases. The process begins with converting the ER diagram into an initial schema by defining tables, primary keys, and foreign keys. Normalization then refines this schema through stages: First Normal Form (1NF) eliminates repeating groups and ensures atomicity; Second Normal Form (2NF) removes partial dependencies; and Third Normal Form (3NF) eliminates transitive dependencies. This structured approach minimizes redundancy, ensures data integrity, and optimizes performance. Ultimately, combining ER diagrams with normalization results in a well-organized, scalable database design that supports robust data management and retrievals.