Liyanwei

Basic Tasks

1. Definition of Terms

Candidate Key: An attribute or a group of attributes that can uniquely identify a tuple (a row of data) in a relation.

Composite Key: A key composed of multiple attributes that together can uniquely identify a tuple in a relation.

Foreign Key: An attribute in one relation that references the primary key of another relation, used to establish a connection between two relations.

Functional Dependency: In a relational schema, if the value of attribute A determines the value of attribute B uniquely, then attribute B is said to be functionally dependent on attribute A, denoted as A → B.

2. Integrity Rules/Constraints in the Relational Model

Entity Integrity: The primary attributes (i.e., the components of the primary key) in a relation cannot have null values. This ensures that each row of data is uniquely identifiable in the table.

Referential Integrity: If a foreign key FK in relation R references the primary key PK of relation S, then for each tuple in relation R, the value of FK is either null or equal to the PK value of some tuple in relation S. This ensures the consistency and correctness of data and avoids invalid references.

User-Defined Integrity: Constraints defined by users according to specific business requirements, such as data types and value ranges of attributes.

3. Violation of Relational Integrity Rules in the Tables

Table 1 Film

Entity Integrity: No violation. The primary keys filmNo and directorNo have no null values and uniquely identify tuples.

Referential Integrity: No violation. The directorNo in the Film table corresponds to the directorNo in the Director table.

Table 2

Entity Integrity: No violation. The primary key is not clear, but there are no null values and no identification problems for duplicate rows (assuming Supplier and PartNo together form the primary key; otherwise, there may be an entity integrity issue).

Referential Integrity: No violation (no foreign keys involved).

Table 3

Entity Integrity: There may be an issue. Some records have a null value for Artist. If Song is the primary key and is required to completely determine a tuple, then it violates entity integrity.

Referential Integrity: No violation (no foreign keys involved).

Medium Tasks

4. Analysis of the Project - Employee Relation Table

a. Anomalies

Insertion Anomaly: If only partial project information is known, a new project cannot be inserted because the table requires employee-related information to be provided simultaneously.

Deletion Anomaly: If a project is deleted, employee information may be lost because the employee information and project information are stored in the same table.

Modification Anomaly: When modifying an employee's department information, multiple places may need to be updated, which is prone to data inconsistency.

b. Derivation of 1NF

Make each field in the table atomic and indivisible. The original table already basically satisfies 1NF, but some composite attributes can be further clarified as single attributes (if there are composite attribute situations).

c. Derivation of 2NF

On the basis of satisfying 1NF, eliminate partial functional dependencies. For this table, ProjectCode determines ProjectTitle, ProjectManager, ProjectBudget, etc., which are project-related information, and EmployeeNo determines EmployeeName, DepartmentNo, etc., which are employee-related information. Therefore, the table needs to be split into a project table (Project: ProjectCode, ProjectTitle, ProjectManager, ProjectBudget), an employee table (Employee: EmployeeNo, EmployeeName, DepartmentNo, DepartmentName, HourlyRate), and an association table (ProjectEmployee: ProjectCode, EmployeeNo) to record the relationship between projects and employees, eliminating partial functional dependencies.

d. Derivation of 3NF

On the basis of satisfying 2NF, eliminate transitive functional dependencies. Since DepartmentNo determines DepartmentName, in the employee table, DepartmentName has a transitive functional dependency on EmployeeNo. Therefore, the department name needs to be separated from the employee table and placed in a separate department table (Department: DepartmentNo, DepartmentName).

5. Normalization of Bakery Order Data

a. Anomalies

Insertion Anomaly: If only partial customer information is known, a new record may not be inserted. For example, if the Acc.No. of a customer is not known, new order information may not be inserted.

Update Anomaly: Updating a certain piece of customer information may require updates in multiple places, which is prone to data inconsistency. For example, if a customer's address changes, it may need to be updated in multiple order records.

Deletion Anomaly: Deleting a customer order may lead to the loss of customer information.

b. Derivation of 1NF

Split the composite attributes in the table into single attributes to ensure that each attribute is atomic. For example, Date can be split into Year, Month, and Day (if more precise time recording is required), and it is made clear that each row of data corresponds to an order detail (one order may contain multiple details, each corresponding to a product).

c. Derivation of 2NF

Eliminate partial functional dependencies. Because customer information determines customer order information, the table needs to be split into a customer table (Customer: Customer, Address), an order table (Order: Order No., Acc.No., Customer, Date), and an order detail table (OrderDetail: Order No., Item, Qty., Price, Total Cost). The Customer field in the Customer table is related to the Customer fields in the Order table and the OrderDetail table, and the Order No. field in the Order table is related to the Order No. field in the OrderDetail table.

d. Derivation of 3NF

Eliminate transitive functional dependencies. The current table structure already largely satisfies 3NF, but if there are transitive dependencies (for example, an attribute of a customer determines another attribute, and this attribute in turn determines an attribute in the order table), these attributes need to be further separated into separate tables.

6. Analysis of Hospital-Related Tables

a. Schema of Each Table

Patient Table

PatientNo (primary key), Surname, FirstNname

Admission Table

PatientNo (foreign key, references Patient table's PatientNo), Admitted, Discharged, Ward

Doctor Table

DoctorNo (primary key), Surname, FirstName, Ward

Ward Table

Ward (primary key), WardName, DoctorNo - InCharge (foreign key, references Doctor table's DoctorNo)

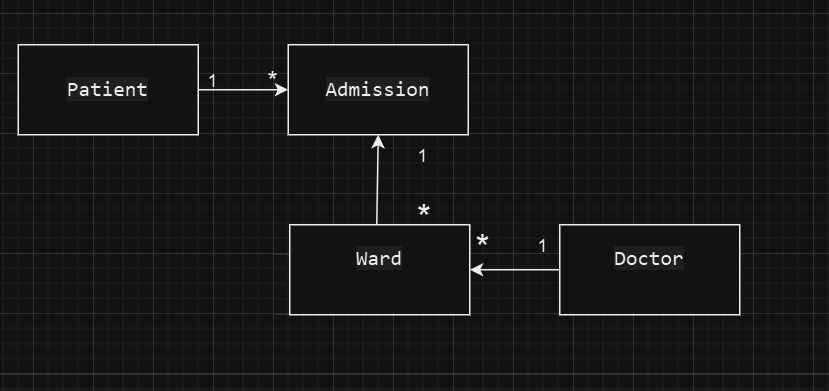
b. Relationships Between Tables and Key Linkages

The Patient table and the Admission table have a one-to-many relationship through PatientNo. One patient can have multiple admission records.

The Doctor table and the Ward table have a many-to-one relationship through DoctorNo. Multiple doctors may work in one ward, but one ward has only one person in charge.

The Ward field in the Admission table has a referential relationship with the Ward in the Ward table.

c. Generation of ER Diagram



Advanced Tasks

7. Functional Dependencies and Normal Form Derivation

a. Table 1: Catering

Functional Dependencies

Order No. → Account No., Customer, Address, Date

Account No. → Customer, Address (there may be redundancy because Customer and Address may also depend on Order No.)

Item → Quantity, Item Price

Normal Form Derivation

1NF: The original table already satisfies 1NF as each field is atomic.

2NF: To eliminate partial functional dependencies, the table can be split into an order table (Order: Order No., Account No., Customer, Address, Date), a product table (Item: Item, Quantity, Item Price), and an association table (OrderItem: Order No., Item) to record the relationship between orders and products.

3NF: The current structure satisfies 3NF as there are no transitive functional dependencies.

b. Table 2: Student Records

Functional Dependencies

Student No → Name, Course, Course Duration

Course → Module No, Module name, Lecturer

Normal Form Derivation

1NF: The original table basically satisfies 1NF, but for course module information, it can be further split to ensure atomicity. For example, each course module can be recorded as a separate row instead of multiple modules in one cell.

2NF: To eliminate partial functional dependencies, the table can be split into a student table (Student: Student No, Name, Course, Course Duration), a course table (Course: Course, Module No, Module name, Lecturer), and an association table (StudentCourse: Student No, Course) to record the relationship between students and courses.

3NF: The current structure satisfies 3NF as there are no transitive functional dependencies.

8. Normalization of Car Rental Company Data

a. Functional Dependencies

Report ID, Reporting Period → Branch Code, Branch Name, Car Plate Nr, Car Type, Bill Nr, Bill Date, Penalty, Final Bill, Supervisor ID, Supervisor Name

Branch Code → Branch Name

Car Plate Nr → Car Type

Bill Nr → Bill Date, Penalty, Final Bill

b. Normal Form Derivation

1NF: Each field in the original table is atomic, satisfying 1NF.

2NF:

Split out a branch table (Branch: Branch Code, Branch Name).

A vehicle table (Car: Car Plate Nr, Car Type).

A bill table (Bill: Bill Nr, Bill Date, Penalty, Final Bill).

An association table (ReportBranchCarBill: Report ID, Reporting Period, Branch Code, Car Plate Nr, Bill Nr) to record the relationship between reports, branches, vehicles, and bills.

3NF:

Split out a supervisor table (Supervisor: Supervisor ID, Supervisor Name).

In the association table ReportBranchCarBill, make Supervisor ID a foreign key and associate it with the Supervisor table to eliminate transitive functional dependencies.

9.

a. Database Life Cycle and Models

Requirement Analysis

Clearly identify that the M70 marine service company needs to record key information including the type of marine equipment, maintenance time, maintenance tasks for each boat, engineer information, and task execution details. At the same time, it is also necessary to consider the relevant information of contractors for specific tasks.

Conceptual Design - Logical Model

Determine the entities as follows:

Boat: Has the basic identification and related attributes of a boat.

Engineer: Contains personal information and skills-related attributes of an engineer.

Contractor: Records relevant information and skills of a contractor.

Task: Covers various types of maintenance tasks and related descriptions.

Equipment: Describes the type of equipment on a boat.

Relationship Settings:

The relationship between boat and engineer is many-to-many. That is, a boat may have multiple engineers for maintenance, and an engineer may also participate in the maintenance of multiple boats.

The relationship between boat and task is many-to-many. A boat may have multiple maintenance tasks, and a task may also be executed on multiple boats.

The relationship between engineer and task is many-to-many. An engineer may perform multiple tasks, and a task may also be completed by multiple engineers.

The relationship between equipment and boat is many-to-one. A boat has multiple types of equipment, but a type of equipment belongs to only one boat.

The relationship between boat and contractor is many-to-many. A boat may require multiple contractors to complete specific tasks, and a contractor may also provide services for multiple boats.

Physical Design

Boat Table

BoatID (Primary Key): Used to uniquely identify each boat.

BoatName: The name of the boat.

EquipmentType: The main equipment type on the boat.

Engineer Table

EngineerID (Primary Key): Uniquely identifies each engineer.

EngineerName: The name of the engineer.

Skills: The skills possessed by the engineer.

Contractor Table

ContractorID (Primary Key): Distinguishes different contractors.

ContractorName: The name of the contractor.

ContractorSkills: The specific skills possessed by the contractor.

Task Table

TaskID (Primary Key): Identifies different tasks.

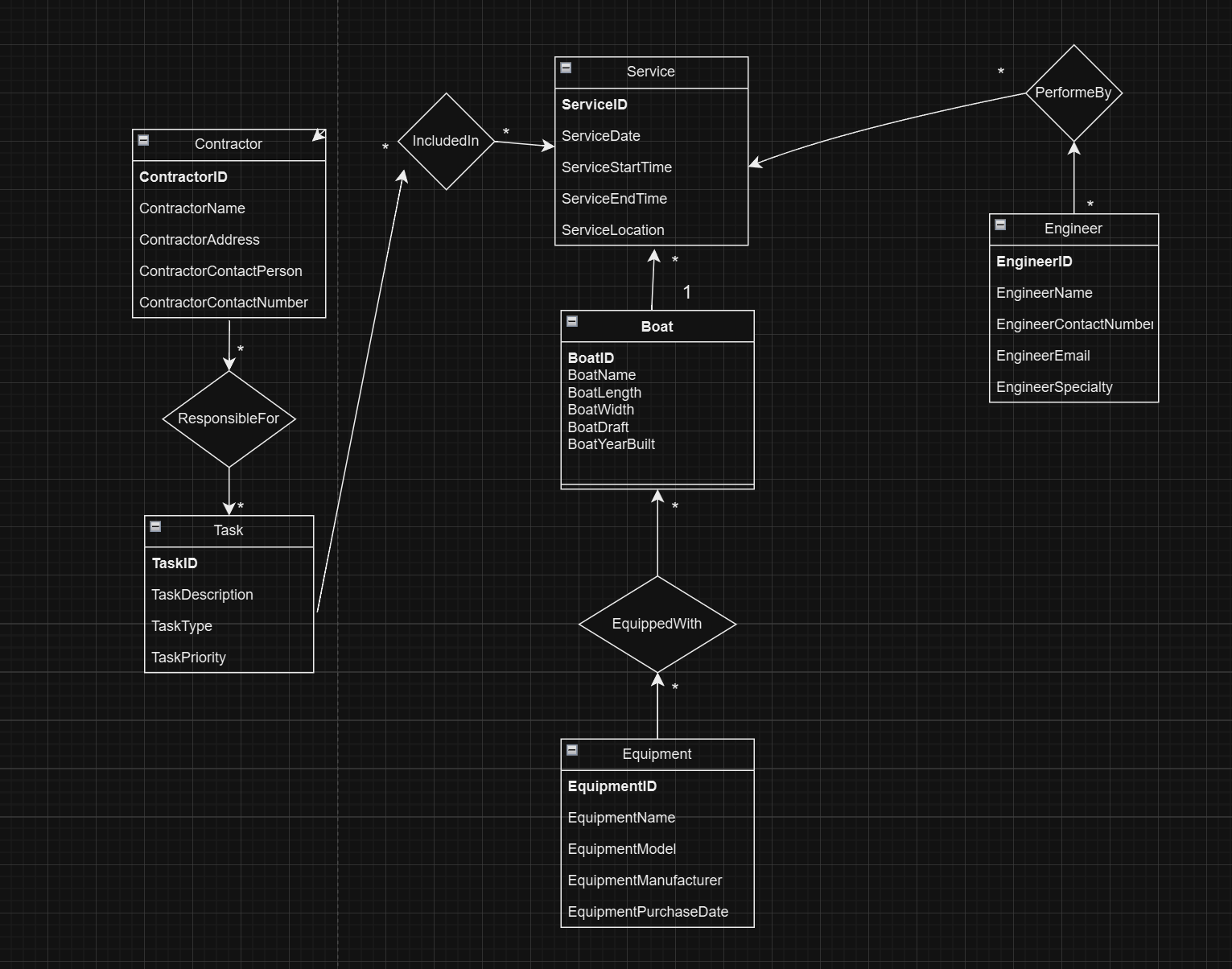
TaskType: The type of task, such as service, software upgrade, repair, safety inspection, etc.

Equipment Table

EquipmentID (Primary Key): Uniquely identifies each type of equipment.

EquipmentType: The type of equipment.

b. ER Diagram Generation

c. Functional Dependencies

Boat → EquipmentType: A boat determines its main equipment type.

Engineer → Skills: An engineer determines the skills he or she possesses.

Task → TaskType: A task determines its own type (such as service, software upgrade, etc.).

Contractor → ContractorSkills: A contractor determines its own specific skills.

d. Example of Data Population

Boat Table

Insert data: BoatID is 1, BoatName is "Boat A", EquipmentType is "Engine".

Engineer Table

Insert data: EngineerID is 1, EngineerName is "John", Skills is "Repair".

Task Table

Insert data: TaskID is 1, TaskType is "Repair", BoatID is 1, EngineerID is 1.

Contractor Table

Insert data: ContractorID is 1, ContractorName is "ABC Contractor", ContractorSkills is "Special Skill".

Equipment Table

Insert data: EquipmentID is 1, EquipmentType is "Engine", BoatID is 1.

e. Normalization

1NF

Each field in the original table structure is atomic and satisfies 1NF. For example, fields such as BoatName and EquipmentType are single, indivisible values.

2NF

For many-to-many relationships, create association tables: - The `BoatEngineer` table (`BoatID`, `EngineerID`): Records the relationship between boats and engineers, ensuring data integrity and consistency and eliminating partial functional dependencies. - The `BoatTask` table (`BoatID`, `TaskID`): Used to record the association between boats and tasks. - The `EngineerTask` table (`EngineerID`, `TaskID`): Represents the relationship between engineers and tasks. - The `BoatContractor` table (`BoatID`, `ContractorID`): Records the connection between boats and contractors. The data in each table depends only on the primary key. For example, in the `BoatEngineer` table, `BoatID` and `EngineerID` together serve as the primary key, and other data do not have partial functional dependencies on the primary key.

3NF

Check for transitive functional dependencies: In the current structure, for example, the `Skills` in the `Engineer` table does not determine the attributes in other tables, and there are no transitive dependency relationships. The current structure largely satisfies 3NF, but it needs to be further checked and adjusted according to the actual data situation. If new dependency relationships are discovered during subsequent data maintenance and update processes, it may be necessary to further optimize the table structure.