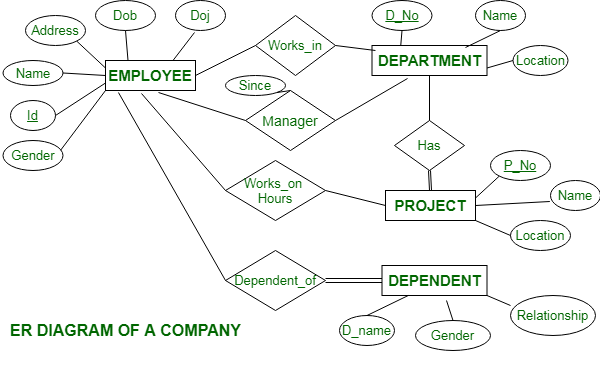
**ASSIGMNENT 1** karthikavula76@gmail.com

**RDBMS and SQL**

1.Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.

Ans:



1. **Identify Entities**: Determine the key components of the business scenario that need to be represented in the database. These are typically the nouns in the scenario description, such as Customer, Order, Product, etc.
2. **Define Relationships**: Establish how the entities interact with each other. Common relationships include one-to-one, one-to-many, and many-to-many.
3. **Specify Attributes**: For each entity, list the details that need to be recorded. For example, a Customer entity might have attributes like CustomerID, Name, Address, etc.
4. **Determine Cardinality**: This defines the numerical aspects of the relationships between entities, such as how many orders can be associated with a single customer.
5. **Normalization**: Organize the data to reduce redundancy and improve data integrity. This involves creating tables that satisfy the rules of each normal form up to the third normal form (3NF).

Here’s a simplified example based on a hypothetical online bookstore:

* **Entities**: Customer, Order, Book
* **Relationships**:
  + A Customer can place multiple Orders (one-to-many)
  + An Order can contain multiple Books (many-to-many, which requires a junction table like OrderDetails)
* **Attributes**:
  + Customer: CustomerID, Name, Email
  + Order: OrderID, Date, CustomerID (foreign key)
  + Book: ISBN, Title, Author, Price
* **Cardinality**:
  + One Customer can have zero or many Orders
  + One Order can have one or many Books
* **Normalization**:
  + 1NF: Eliminate repeating groups; ensure each field contains only atomic values
  + 2NF: Remove partial dependencies; no non-prime attribute should depend on part of a composite key
  + 3NF: Remove transitive dependencies; attributes should depend only on the primary key

2.Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

Ans:

Database Schema for a Library System

Table:

MemberFields:

MemberID INT PRIMARY KEY AUTO\_INCREMENT

Name VARCHAR(100) NOT NULL

Email VARCHAR(100) NOT NULL UNIQUE

Phone VARCHAR(15) NOT NULL UNIQUE

Address VARCHAR(255) NOT NULL

JoinDate DATE NOT NULL CHECK (JoinDate <= CURRENT\_DATE)

**Table: Book**

Fields:

BookID INT PRIMARY KEY AUTO\_INCREMENT

Title VARCHAR(255) NOT NULL

Author VARCHAR(100) NOT NULL

ISBN VARCHAR(13) NOT NULL UNIQUE CHECK (LENGTH(ISBN) = 13)

Publisher VARCHAR(100) NOT NULL

YearPublished INT NOT NULL CHECK (YearPublished >= 1450 AND YearPublished <= YEAR(CURDATE()))

Category VARCHAR(100) NOT NULL

**Table: Loan**

Fields:

LoanID INT PRIMARY KEY AUTO\_INCREMENT

MemberID INT NOT NULL

BookID INT NOT NULL

LoanDate DATE NOT NULL CHECK (LoanDate <= CURRENT\_DATE)

DueDate DATE NOT NULL CHECK (DueDate >= LoanDate)

ReturnDate DATE

Constraints:

FOREIGN KEY (MemberID) REFERENCES Member(MemberID)

FOREIGN KEY (BookID) REFERENCES Book(BookID)

CHECK (ReturnDate IS NULL OR ReturnDate >= LoanDate)

**Table: Fine**

Fields:

FineID INT PRIMARY KEY AUTO\_INCREMENT

LoanID INT NOT NULL

Amount DECIMAL(5, 2) NOT NULL CHECK (Amount >= 0)

FineDate DATE NOT NULL CHECK (FineDate <= CURRENT\_DATE)

Constraints:

FOREIGN KEY (LoanID)

REFERENCES Loan(LoanID)

**Table: Author**

Fields:

AuthorID INT PRIMARY KEY AUTO\_INCREMENT

Name VARCHAR(100) NOT NULL UNIQUE

Bio TEXT

BirthDate DATE CHECK (BirthDate <= CURRENT\_DATE)

DeathDate DATE CHECK (DeathDate IS NULL OR DeathDate >= BirthDate)

Table: BookAuthor (To handle the many-to-many relationship between Books and Authors)

Fields:

BookAuthorID INT PRIMARY KEY AUTO\_INCREMENT

BookID INT NOT NULL

AuthorID INT NOT NULL

Constraints:

KEY (BookID) REFERENCES Book(BookID)

FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID)

UNIQUE (BookID, AuthorID)

Relationships and Constraints:

A Member can have multiple Loans, but each Loan is associated with one Member.

A Book can be loaned multiple times, hence it can appear in multiple Loans, but each Loan refers to one Book.

Fines are associated with Loans, and each fine is related to one specific Loan.

Books can have multiple Authors, and Authors can write multiple Books. This many-to-many relationship is managed by the BookAuthor table.

SQL Statement to Create the TABLE.

CREATE TABLE Member (

    MemberID INT PRIMARY KEY AUTO\_INCREMENT,

    Name VARCHAR(100) NOT NULL,

    Email VARCHAR(100) NOT NULL UNIQUE,

    Phone VARCHAR(15) NOT NULL UNIQUE,

    Address VARCHAR(255) NOT NULL,

    JoinDate DATE NOT NULL CHECK (JoinDate <= CURRENT\_DATE)

);

CREATE TABLE Book (

    BookID INT PRIMARY KEY AUTO\_INCREMENT,

    Title VARCHAR(255) NOT NULL,

    Author VARCHAR(100) NOT NULL,

    ISBN VARCHAR(13) NOT NULL UNIQUE CHECK (LENGTH(ISBN) = 13),

    Publisher VARCHAR(100) NOT NULL,

    YearPublished INT NOT NULL CHECK (YearPublished >= 1450 AND YearPublished <= YEAR(CURDATE())),

    Category VARCHAR(100) NOT NULL

);

CREATE TABLE Loan (

    LoanID INT PRIMARY KEY AUTO\_INCREMENT,

    MemberID INT NOT NULL,

    BookID INT NOT NULL,

    LoanDate DATE NOT NULL CHECK (LoanDate <= CURRENT\_DATE),

    DueDate DATE NOT NULL CHECK (DueDate >= LoanDate),

    ReturnDate DATE,

    FOREIGN KEY (MemberID) REFERENCES Member(MemberID),

    FOREIGN KEY (BookID) REFERENCES Book(BookID),

    CHECK (ReturnDate IS NULL OR ReturnDate >= LoanDate)

);

CREATE TABLE Fine (

    FineID INT PRIMARY KEY AUTO\_INCREMENT,

    LoanID INT NOT NULL,

    Amount DECIMAL(5, 2) NOT NULL CHECK (Amount >= 0),

    FineDate DATE NOT NULL CHECK (FineDate <= CURRENT\_DATE),

    FOREIGN KEY (LoanID) REFERENCES Loan(LoanID)

);

CREATE TABLE Author (

    AuthorID INT PRIMARY KEY AUTO\_INCREMENT,

    Name VARCHAR(100) NOT NULL UNIQUE,

    Bio TEXT,

    BirthDate DATE CHECK (BirthDate <= CURRENT\_DATE),

    DeathDate DATE CHECK (DeathDate IS NULL OR DeathDate >= BirthDate)

);

CREATE TABLE BookAuthor (

    BookAuthorID INT PRIMARY KEY AUTO\_INCREMENT,

    BookID INT NOT NULL,

    AuthorID INT NOT NULL,

    FOREIGN KEY (BookID) REFERENCES Book(BookID),

    FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),

    UNIQUE (BookID, AuthorID)

);

3.Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

Ans:  **ACID Properties**:

* **Atomicity**: A transaction is either fully executed or not executed at all. It follows the “all or nothing” rule. If any part of the transaction fails, the entire transaction is rolled back.
* **Consistency**: The database remains in a consistent state before and after the transaction. Integrity constraints are maintained.
* **Isolation**: Multiple transactions can occur concurrently without affecting each other. Changes made by one transaction are not visible to others until committed.

[**Durability**: Once a transaction is committed, its changes are permanently stored in the database, even in the event of system failure](https://www.geeksforgeeks.org/acid-properties-in-dbms/)

**SQL Statements for Locking and Isolation Levels**:

Select \* from tab

Create table accounts (accounted int primary key, balance decimal(10, 2) not null check(balance >= 0), name varchar(20));

Insert into accounts (accounted , balance) values (1, 2000.00);

Select \* from accounts;

Alter table accounts add name varchar(20);

Insert into accounts (accounted , balance, name) values (2, 2500.00, ‘karthik’);

Alter table accounts add city varchar(20);

Insert into accounts (accounted , balance, name, city) values (2, 2500.00, ‘karthik’, ‘Hyd’);

Update accounts set balance = balance – 100 where accountid = 1;

Update accounts set balance = balance +1250 where accountid = 3;

Select \* from accounts;

4.Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.

Ans:

Below are the SQL statements to create a new database, tables, modify table structures, and remove a redundant table based on a hypothetical library schema:

Create table Books ( bookid int primary key, name varchar(20), email varchar(20), author varchar(20));

Insert into Books values (101, ‘karthik’, ‘[karthik@123.com](mailto:karthik@123.com)’, ‘classmate’);

Insert into Books values (102, ‘ravi’, ‘[ravi@243.com](mailto:ravi@243.com)’, ‘risinsun’);

Insert into Books values(103, ‘chandu’, ‘[ch@134.com](mailto:ch@134.com)’, ‘sport’);

Insert into Books values(104, ‘viddu’, ‘[viddu@143.com](mailto:viddu@143.com)’, ‘srividya’);

Select \* from Books;

Create table members (memberid, city varchar(20), joindate varchar(10));

Insert into members values (2306156, ‘hyd’, ‘1993-12-07’);

Insert into members values (233567, ‘Mumbai, ‘1999-11-09’);

Insert into members values (234598, ‘Ap’, ‘2003-05-02’);

Select \* from members;

## modifying Books table to add a colum###

Alter table Books add location varchar(50);

Alter table Books add State varchar(50);

## modifying members table to add a colum###

Alter table members add amount varchar(20);

Alter table members add Title varchar(20);

###Drop members table ##

Drop table members;

5.Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

Ans

Creating an index on database table can improve query performance, especially for ready-heavy operations. Here a using SQl creating a table called “EMPLOYEE”

Create table employee ( empid primary key, name varchar(20), department varchar(20), salary number (10, 2));

Insert into employees (empid, name, department, salary) values (1, ‘karthik’ , ‘IT’, 45000);

Insert into employees (empid, name, department, salary) values (2, ‘viddu’ , ‘medical’, 35000);

Insert into employees (empid, name, department, salary) values (3, ‘bunny’ , ‘Drive’, 24000);

Insert into employees (empid, name, department, salary) values (4, ‘raju’ , ‘HR’, 23000);

Create index empid\_name ON employees(name);

Explain plan for select \* from employees where name = ‘karthik’;

Select \* from table employees

Drop Index empid\_name;

6. Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operations to load data from an external source.

Ans:

INSERT New Records

Insert into Books values (101, ‘karthik’, ‘[karthik@123.com](mailto:karthik@123.com)’, ‘classmate’);

Insert into Books values (102, ‘ravi’, ‘[ravi@243.com](mailto:ravi@243.com)’, ‘risinsun’);

Insert into Books values(103, ‘chandu’, ‘[ch@134.com](mailto:ch@134.com)’, ‘sport’);

Insert into Books values(104, ‘viddu’, ‘[viddu@143.com](mailto:viddu@143.com)’, ‘srividya’);

Insert into Books values(105, ‘sri’, ‘[vsri@143.com](mailto:vsri@143.com)’, ‘Bird’);

Alter table Books set year varchar(20);

UPDATE New Records:

Update Books set year = 1995 where bookid = 102;

Update Books set year = 1999 where bookid = 103;

Update Books set year = 1997 where bookid = 105;

DELETE Records:

Delete from Books where bookid = ‘102’;

BULK INSERT:

Bulk insert Books from ‘c:\path\to\books.csv’

With (field terminator = ‘,’,

Row terminator = ‘\n’, fitrstrow = 2);