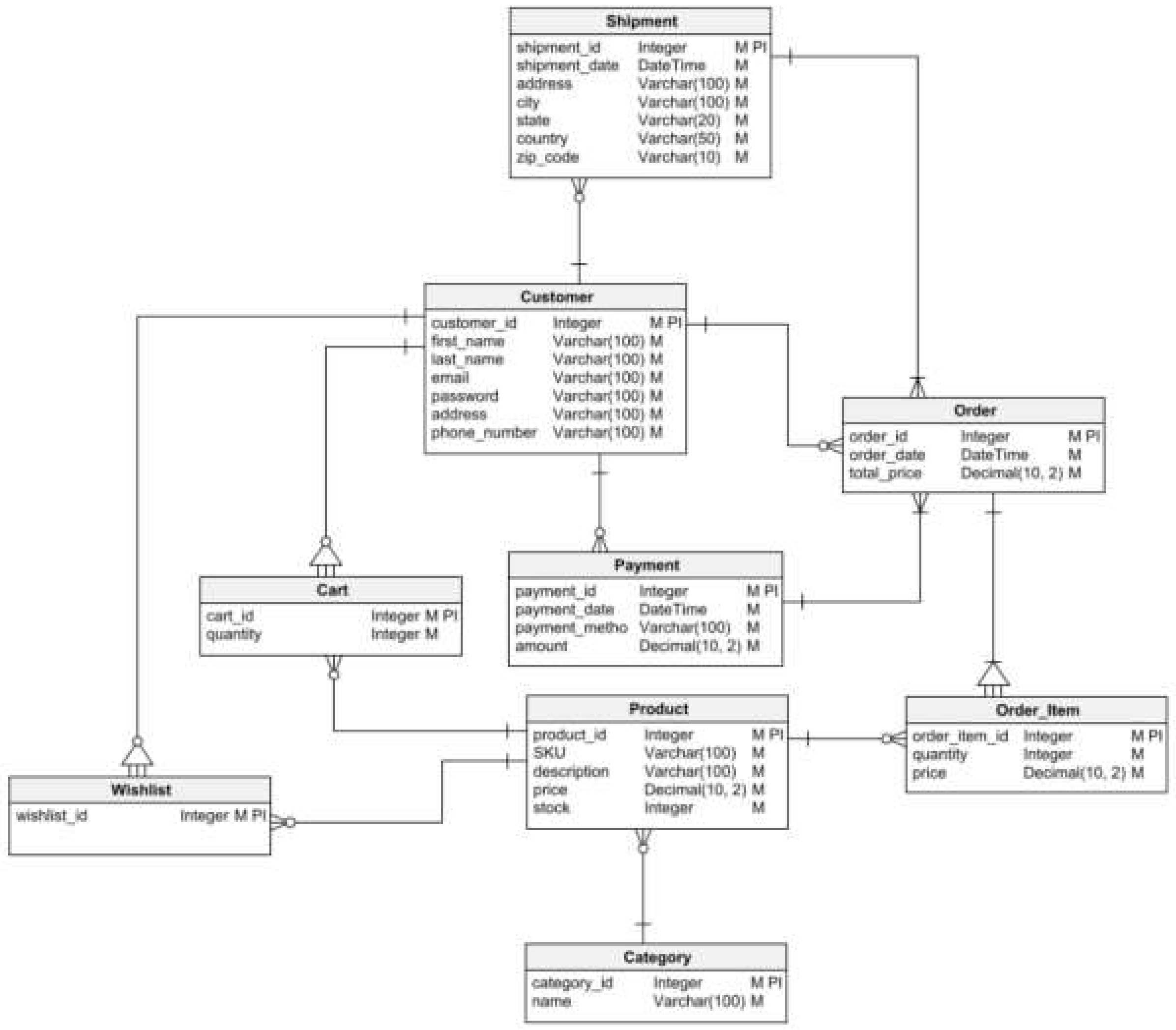
RDBMS ASSIGNMENTS(1-7) 21-05

Assignment 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.

Solution:

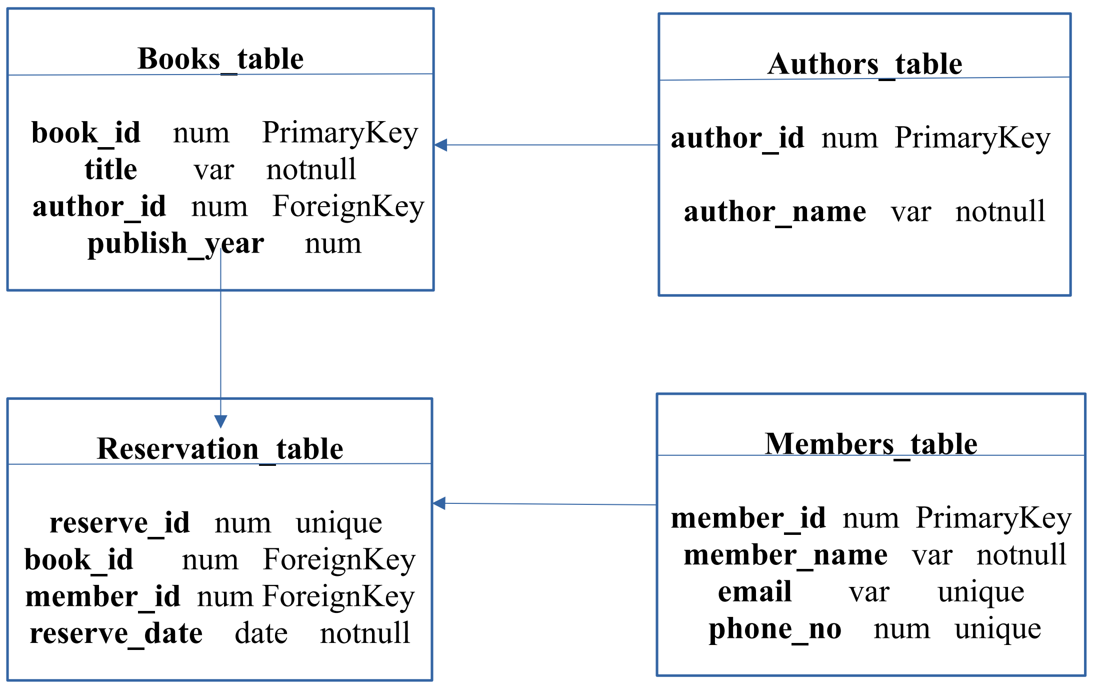


Explanation:

This ER diagram reflects proper normalization up to the third normal form by organizing entities and their attributes in a way that minimizes redundancy and dependency. Each entity has its own primary key, and relationships between entities are clearly defined with appropriate cardinality.

Assignment 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.

Database Table Design:



Explanation:

- Authors table stores information about authors.
- Books table stores information about books, Each book is associated with one author.
- Members table stores information about library members.
- Reservation table tracks the Reservation made by members, Each Reservation is associated with one book and one member.
- The **primary keys (author_id, book_id, member_id)** uniquely identify each record in their respective tables. The **foreign keys (author_id, book_id, member_id)** establish relationships between tables, ensuring referential integrity.

Assignment 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.

ACID properties of a transaction:

Atomicity: This property ensures that a transaction is either completed in its entirety or not at all.
if any part of the transaction fails, the entire transaction is rolled back, leaving the database in its original state.

Consistency: This property ensures that the database remains in a valid state before and after the transaction. Even though the data might change during the transaction, it must adhere to all constraints, triggers, and rules defined in the database schema.

Isolation: This property ensures that multiple transactions can occur concurrently without interfering with each other. Each transaction should be isolated from others until it is complete, preventing transactions from seeing each other's intermediate states.

Durability: This property guarantees that once a transaction is committed, its changes are permanent and will not be lost, even in the event of a system failure.

SQL Commands:

```
CREATE TABLE Accounts (
   Account_id NUMBER PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   Balance DECIMAL(10,2));

INSERT INTO Accounts  VALUES (1, ‘Ravi’,1000.00);
INSERT INTO Accounts  VALUES (2, ‘Siva’, 500.00);

BEGIN TRANSACTION;

UPDATE Accounts SET balance = balance - 200.00 WHERE Account_id = 1;

UPDATE Accounts SET balance = balance + 200.00 WHERE Account_id = 2;

COMMIT;

SELECT * FROM Accounts;
```

Results:

Account_id	**Name**	**Balance**
1	Ravi	800.00
2	Siva	700.00

Different isolation levels:

- **READ UNCOMMITTED:** Allows dirty reads, meaning transactions can see uncommitted changes made by other transactions.
- **READ COMMITTED:** Prevents dirty reads but allows non-repeatable reads and phantom reads.
- **REPEATABLE READ:** Prevents dirty reads and non-repeatable reads but allows phantom reads.
- **SERIALIZABLE:** Prevents dirty reads, non-repeatable reads, and phantom reads by serializing transactions.

Assignment 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.

SQL Commands:

```
CREATE DATABASE LibraryDatabase;

CREATE TABLE Authors (
   author_id NUMBER(10) PRIMARY KEY,
   author_name VARCHAR(50) NOT NULL);

CREATE TABLE Books (
   book_id NUMBER(10) PRIMARY KEY,
   title VARCHAR(50) NOT NULL,
   author_id  REFERENCES Authors(author_id),
   publication_year NUMBER);

CREATE TABLE Members (
   member_id NUMBER(10) PRIMARY KEY,
   member_name VARCHAR(50) NOT NULL,
   email VARCHAR(20) UNIQUE,
   phone_no NUMBER(10)  UNIQUE);

CREATE TABLE Reservations (
   reservation_id NUMBER(10) PRIMARY KEY,
   book_id REFERENCES Books(book_id),
   member_id REFERENCES Members(member_id),
   reservation_date DATE NOT NULL);

ALTER TABLE Books
ADD COLUMN language VARCHAR(50);

ALTER TABLE Authors
MOTIFY author_id NUMBER(15);

ALTER TABLE Members
DROP COLUMN email;

DROP TABLE Reservations;
```

Assignment 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.

SQL Commands:

```
CREATE TABLE Employees (
    empID NUMBER(2) PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Department VARCHAR(50),
    Salary NUMBER NOT NULL);

INSERT INTO Employees VALUES (1, ‘Ravi’, ‘IT’, 60000);
INSERT INTO Employees VALUES  (2, ‘Shan’, ‘IT’, 55000);
INSERT INTO Employees VALUES  (3, ‘Siva’, ‘Sales’, 62000);
INSERT INTO Employees VALUES  (4, ‘Suriya', ‘Marketing’, 58000);

CREATE INDEX idxDepartment ON Employees (Department);
SELECT * FROM Employees WHERE Department = ‘IT’;
DROP INDEX idxDepartment;
```

Results:

empID	**Name**	**Department**	**Salary**
1	Ravi	IT	60000
2	Shan	IT	55000

Explanation:

When we create an index on a column, the database system creates a separate data structure that organizes the values of that column in a sorted manner. This allows the database engine to quickly locate rows based on the indexed column's values.

Dropping the index removes the sorted data structure associated with the Department column. Without the index, the database engine will resort to scanning the entire table sequentially to locate rows based on the Department column. This can result in slower query performance, especially for large tables, as the database has to search through every row to find the relevant ones.

Assignment 6: Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

SQL Commands:

```
CREATE USER ‘ravi’@’localhost’ IDENTIFIED BY ‘ravi@1234’;

GRANT SELECT, INSERT, UPDATE ON productdatabase * TO ‘ravi’@’localhost’;

REVOKE INSERT ON productdatabase * FROM ‘ravi’@’localhost’;

DROP USER ‘ravi’@’localhost’;
```

Explanation:

- CREATE USER **'ravi'@'localhost'** IDENTIFIED BY **'ravi@1234'** creates a new database user named **'ravi'** with the password **'ravi@1234'**.
- GRANT SELECT, INSERT, UPDATE ON productdatabase * TO **'ravi'@'localhost'** grants SELECT, INSERT, and UPDATE privileges on all tables within the **'productdatabase'** database to the **'ravi'** user.
- REVOKE INSERT ON productdatabase * FROM **'ravi'@'localhost'** revokes the INSERT privilege from the **'ravi'** user on all tables within the **'productdatabase'** database.
- DROP USER **'ravi'@'localhost'** drops the 'ravi' user from the database.

Assignment 7: 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.

SQL Commands:

```
// Insert new records
INSERT INTO books (title, author, publication_year) VALUES (‘Who moved my cheese’, ‘Spencer Jhonson’, 1997);

// Update existing record
UPDATE books SET publication_year = 1998 WHERE title = ‘Who moved my cheese’;

// Delete existing record
DELETE FROM books WHERE publication_year < 2000;

// Bulk insert
BULK INSERT books
FROM ‘C:\path\to\myfile\bookdetails.csv’
WITH
(
   FIELDTERMINATOR = ‘,’ ,
   ROWTERMINATOR = ‘\n’,
   FIRSTROW = 2
);
```