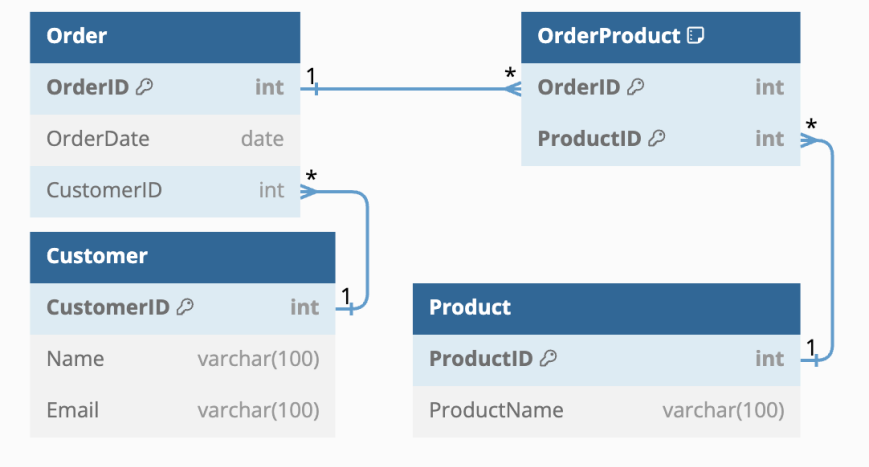
SEPTEMBER 7, 2024

HOMEWORK 1 B

CS 457 B

UMAIR DADA
2O281 - SFBU

1. **Data Models**
    a. **Conceptual Data Models**
        i. A high-level data model that concentrates on the core entities and relationships but does not provide any implementation specifics. Mostly employed in the draft or early stages of a database, it shows the overall structure and focuses on the broad picture.
        ii. Example:
            1. Entities – Customer, Product, Order.
            2. Relationship: A customer can place multiple orders; one order can contain multiple products.
    b. **Physical Data Models**
        i. A low-level detailed model that demonstrates how data should be kept in a database. Essentially, it specifies what tables shall be included, as well as the columns, indexes, constraints, relationships across tables, and data types for each column.
        ii. Example –



c. Representational Data Model

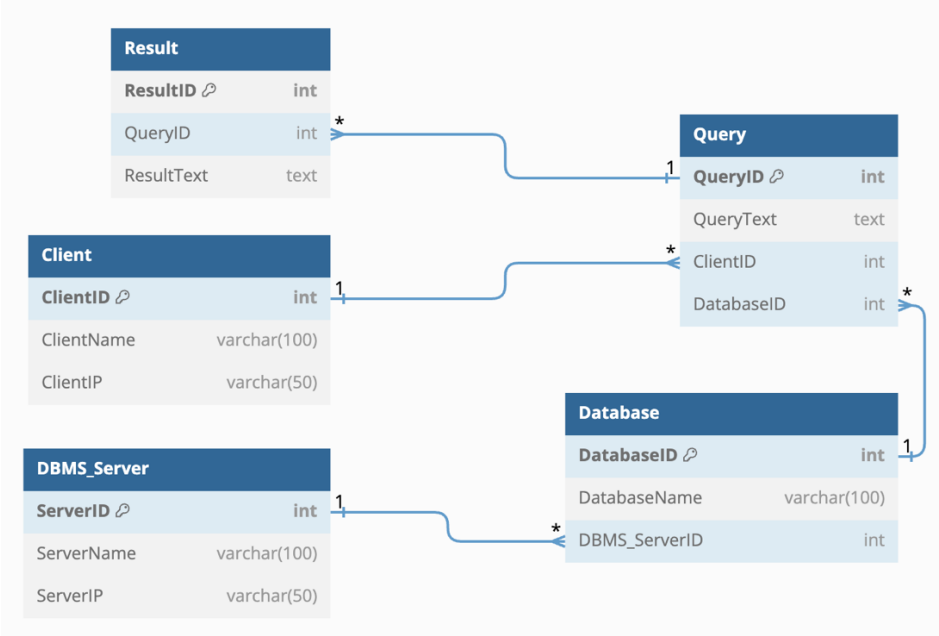
i. A model providing a more thorough representation of the database. It is mostly used to represent the conceptual component of a database, rather than its actual storage or structure. A theoretical concept that is really implemented in a physical data model.

ii. Example:

1. Entities:
    a. Customer (CustomerID, Name, Email)
    b. Order (OrderID, OrderDate, CustomerID)
    c. Product (ProductID, ProductName)
    d. OrderProduct (OrderID, ProductID)
2. Relationships:
    a. One-to-many: A customer may have many orders.
    b. Many-to-many: A single Order can contain numerous Products, and a single Product can be included in several Orders.

2. Diagram of a DBMS architecture:

a.



b. Attributes and examples:

	Attributes	Examples
Client	ClientID, ClientName, ClientIP	ClientID: 1, ClientName: MobileApp, ClientIP: 192.168.1.10
DBMS Server	ServerID, ServerName, ServerIP	ServerID: 1, ServerName: MySQLServer, ServerIP: 192.168.1.20
Database	DatabaseID, DatabaseName, DBMS_ServerID	DatabaseID: 1, DatabaseName: CustomerDB, DBMS_ServerID: 1
Query	QueryID, QueryText, ClientID, DatabaseID	QueryID: 1, QueryText: SELECT * FROM Customers, ClientID: 1, DatabaseID: 1
Result	ResultID, QueryID, ResultText	ResultID: 1, QueryID: 1, ResultText: [{CustomerID: 1, Name: "Umair"}]

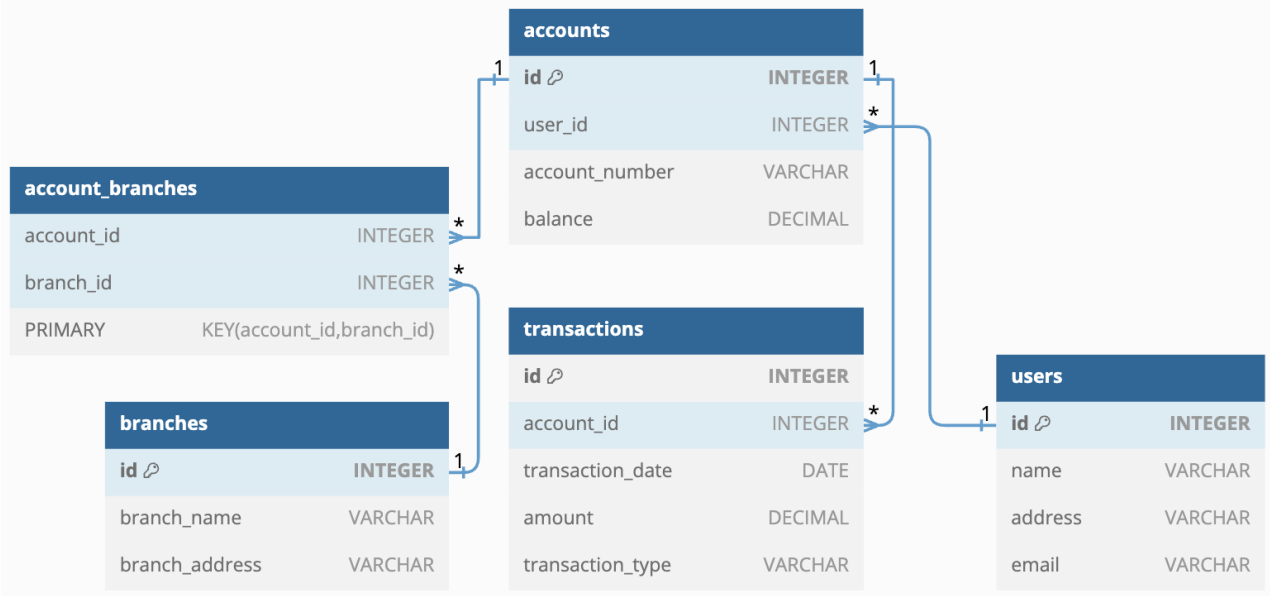
c. Summary: When the Client sends a query to the DBMS Server, The DBMS Server will process the query on the Database which then will return the Result to the Client.

3. Whats the difference.
    a. Difference:
        i. A Relational Data Model arranges the data into a proper table with rows and columns, the relationships for the data are made through shared entries between tables, but in a Object Data Model, the data is more of an object that could hold the data such as attributes and the behavior like the methods, this way allows more complex relationships and enclose the data within the object itself. The Relational Data Model is focused on how the data is being structured where as the Object Data Model leans toward real-world entities with inherent behavior.

    b. Example:
        i. Main Idea – There is a database for a library with tables for "Books" (with columns like Title, Author, ISBN) and "Members" (with columns like Name, Address). Here how to link a book to a member.
        ii. Relational Data Model
            1. You would require a separate table like ‘books_borrowed’ with foreign keys that would reference the ‘Book’ and Member’ tables, this way it will show who borrowed which book (vice-versa).
        iii. Object Data Model
            1. In an object model, a "Book" object would include attributes such as Title, Author, and ISBN, as well as methods like "isAvailable()" and "borrowToMember(memberObject)". A "Member" object may include methods like "borrowBook(bookObject)" or "returnBook(bookObject)", allowing for more direct interaction between related objects.

4. Schema diagram for the database in a bank account holder system

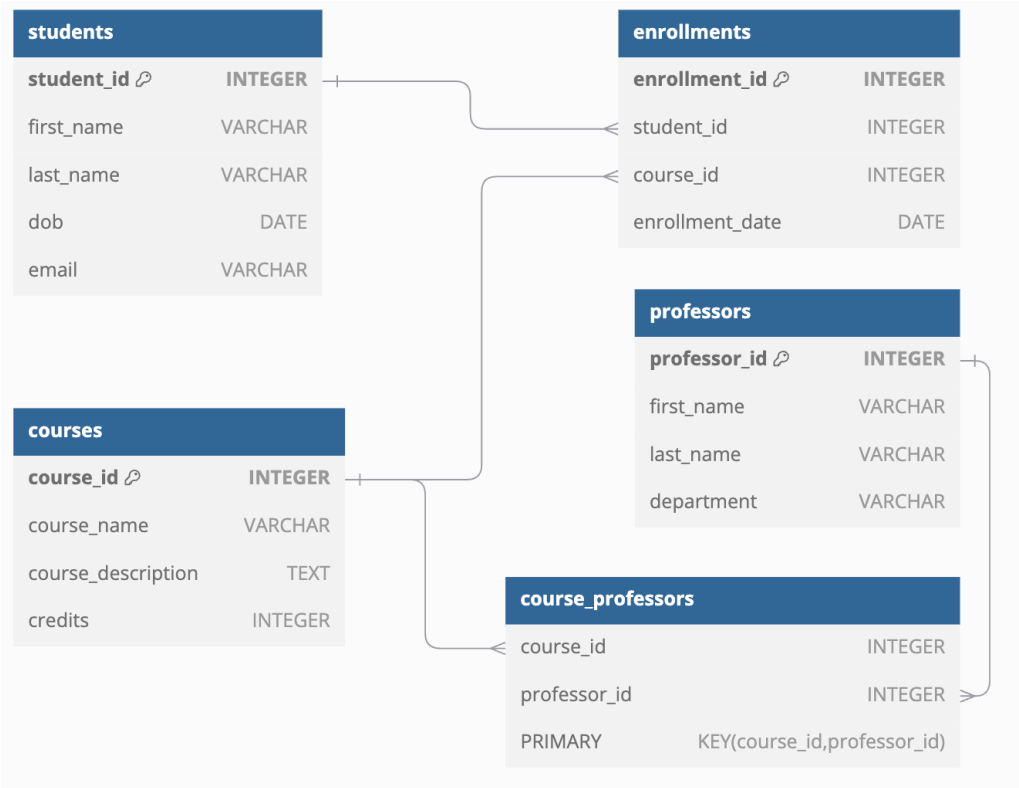
    a. Schema



    b. The structure of a bank account holder system includes several key tables: ‘users’, which contain personal details such as `id`, `name`, `address`, and `email`; ‘accounts’, which hold account-specific details like `id`, `user id`, `account number`, and `balance`; ‘transactions’, which record transaction information like `id`, `account id`, `transaction date`, `amount`, and `type`; ‘branches’, which store details about different bank branches such as `id`, `name`, and `address`; and ‘account_branches’, which is used to link accounts to branches by using `account id` and `branch id`. This schema effectively captures the fundamental elements of a banking system by organizing user accounts, their transactions, and their association with specific branches.

5. Three Schema architecture for a Student Enrolment system in a university.

   a. Schema



   b. Explanation

      i. External Schema represents the views that different users of the system might need.

         1. The student can view their personal information, courses they've been enrolled or their grades.
         2. The professor can view the courses they have been assigned and the list of students in that course.
         3. Administration can see all the student's enrollments and each course details and the lecturer assigned.

      ii. Conceptual Schema defines the overall logical structure of the database, abstracting from the physical details. It includes entities like students, courses, enrollments, and professors

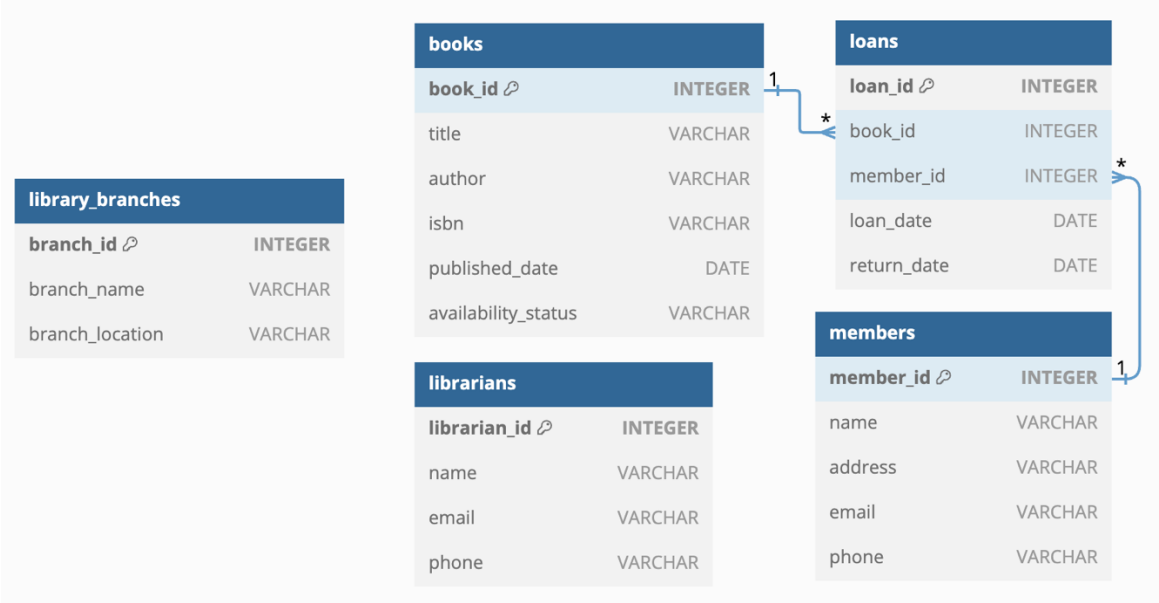
         1. Students can enroll in multiple Courses (via the enrollments table).
         2. Professors can teach multiple Courses (via the course_professors table).

iii. Internal Schema:

1. The internal schema is same as physical schema. It's a low level and pretty much concerns how the data is stored physically. The physical level is used to describe complex low-level data structures in detail.

6. Two-Tier client-server architecture for a Library management system.

a. Schema



b. Client

i. The Client Tier is the user interface (UI) through which library staff (librarians) and users interact with the system. Clients are usually using desktop, web-based, or mobile applications

ii. Examples:

1. Librarians can add, update, and remove books, as well as manage loans and member information.
2. Members can search for books, view available books, and check their loan status.

c. Server

i. The server tier is where the database and business logic reside. In this design, the database server stores the library's data and responds to client requests for data storage, retrieval, and changes.

ii. That database contains:

1. Books: Information about all books available in the library (ID, title, author, ISBN, and availability status).
2. Members: Information about members (ID, name, contact information).
3. Loans: Keeps track of which books are borrowed to which members and when they become due.
4. Librarians: Information on the library staff who manage the system.
5. Library Branches: Information about the different library locations (ID, branch name, and location).

d. Architecture

i. The client sends queries to the server, such as adding a new book or checking a book's availability.
ii. The server processes these requests, interacts with the database, and returns an appropriate response to the client.
iii. The server handles the business logic, and the database server guarantees that data is saved and retrieved efficiently.

Book Exercises:

1. **(2.12)**
    a. Enrollment Office Client: They can enter information that mirror the enlistment of understudies in areas of courses, and later enter the grades of the understudies.

        i. Applications can include:
            1. Register an understudy in a part of a course
            2. Check whether an understudy who is enrolled in a course has the proper essential courses
            3. Remove a student from a course section
            4. Add an understudy to a segment of a course
            5. Enter the understudy grades for a segment
        ii. Application software engineers can compose various canned exchanges for the enrollment office end-clients, giving them either structures and menus, or with a parametric interface.

    b. Admission Office Client: The primary application is to enter recently acknowledged understudies into the information base. It can include the same applications as enrollement office.

    c. Records Office Client: The fundamental application is to print understudy records. Application software engineers can compose a canned exchange utilizing a report generator utility to print the record of an understudy in a recommended design. The specific understudy can be distinguished by name or government managed retirement number. Generating would be an additional use. at the conclusion of each semester, grade slips for all students who have completed courses during that semester. Once more, this application could be customized utilizing a report utility for a generator.

2. **(2.14)**
    a. 2.5.4 Three-Tier Client/Server Architecture for Web Application is the most ideal decision. The Client comprises of Web UI. All of the rules and regulations pertaining to the reservation process and the issuance of tickets are contained in the application logic on the Web Server, while the database management system (DBMS) is on the Database Server.

    b. 2.5.1 Centralized DBMS Architecture wouldn't work since the UI and data set server are on various machines for an web based framework.

    c. 2.5.2 Basic Client/Server Design and 2.5.3 Two-Tier Client/Server Engineering would work on the off chance that the Business Logiv can dwell on a server other than the DBMS Server. In general, if the business logic is located on the DBMS Server, it will overload the server. In the event that the business logic were to live on the web client, it will burden the communication network too a conceivably thin client.

3. (2.15)

a. Course - The "CourseNumber" includes both the department code and a unique course number within that department. While this doesn't account for situations where a department might offer several "Special Topics" courses with the same number but different titles, we're choosing to ignore that for now. One option would be to combine the CourseNumber with the CourseName, but this approach is more prone to errors, like mistyping, when entering the data.

b. Prerequisite - The combination of CourseNumber and PrerequisiteNumber

c. Section - The "SectionIdentifier" is assumed to be unique for each section. However, if we consider that a SectionIdentifier is only unique within a specific course during a particular term (for example, section 2 of CS101), then we would need to make it a combination of SectionIdentifier, CourseNumber, Semester, and Year to ensure it remains unique.

d. Grade Report - The "StudentNumber" and "SectionIdentifier" combination should be unique. Based on the assumption mentioned earlier, the SectionIdentifier will change if a student takes the same course or a different course in a different term. This ensures that the same course taken in different terms is treated as a distinct entry.