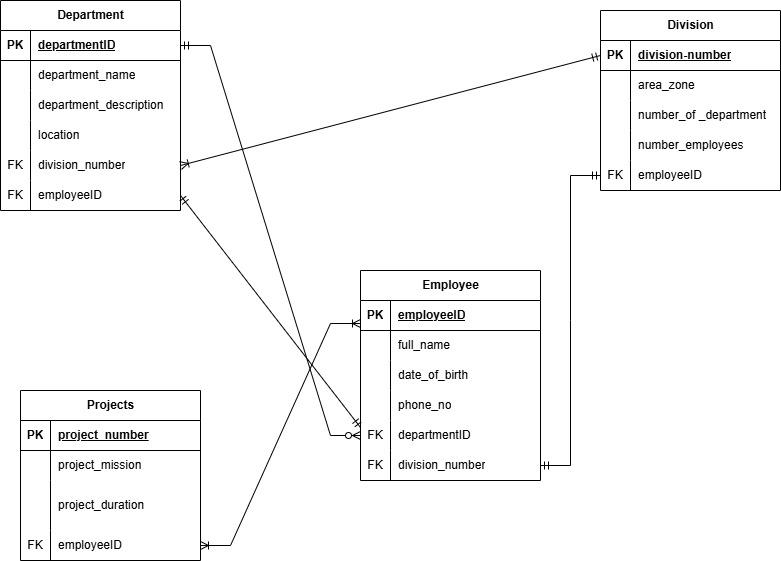
**Question 1a.**

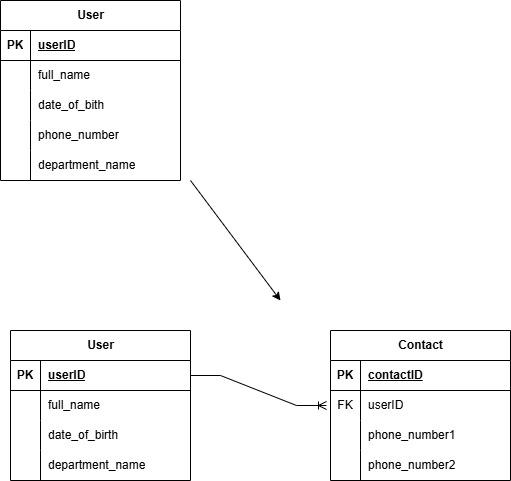


**Question 1b**.

In database design multi-valued attributes are those attributes with more than one value. For instance a ‘User’ can have multiple phone numbers and email addresses and this makes database design complicated which also breaks the 1NF rule of atomicity.

To resolve this the designer has to separate the phone number from the User’s table and create a different table named ‘Contact’. This entity has attributes (contactID, userID, phone\_number1, phone\_number2). Also, ‘phone\_number2’ can be Nullable because not everyone has two phone numbers.

Let’s consider this example:



**Question 1c.**

The 4 rules a database must obey to be considered as a relational database is as follow:

1. **Information rule:** A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.
2. **Guaranteed access rule:** Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.
3. **Systematic Treatment of null values:** This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.
4. **Active/dynamic online based catalog based on relational model:** It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

**Question 2a**

Objectives of database design is as follows;

1. Avoid redundant data: It should have different fields and minimize redundant data. The table should always have a Primary Key that would be a unique id.
2. Faultless information: The database should adhere to the required standard in providing meaningful information for the organization.
3. Data integrity: This ensures that the values are valid and faultless.
4. Modify: The database developed should be worked upon with the conventions and standards, so that it can be easily modified whenever the need arises.

Implementation phases entails that the setting up of the database at the stage will be in accordance with the logical schema This will include the specification of an appropriate storage schema, security enforcement, external schema and so on. Implementation is heavily influenced by the choice of available DBMSs, database tools and operating environment.

**Question 2b**

Four issues that must be addressed during the implementation stage;

1. Resistance to change
2. Lack of clear communication
3. Resources constraint
4. Lack of accountability

**Question 2c**



**Question 3a**

CREATE TABLE ticket(

Ticket\_no SERIAL PRIMARY KEY NOT NULL,

Source VARCHAR(50) NOT NULL,

Destination VARCHAR(100) NOT NULL,

Departure\_time TIME NOT NULL

);

**Question 3b**

CREATE TABLE buses(

Plate\_number VARCHAR(50) PRIMARY KEY NOT NULL,

Brand VARCHAR(50) NOT NULL,

Model VARCHAR(50) NOT NULL,

No\_of\_seater INT NOT NULL

);

CREATE TABLE ticket(

Ticket\_no SERIAL PRIMARY KEY NOT NULL,

Source VARCHAR(50) NOT NULL,

Destination VARCHAR(100) NOT NULL,

Departure\_time TIME NOT NULL,

Plate\_number VARCHAR(50) REFERENCES buses(plate\_number)

);

**Question 3ci**

**Top-down design method:** This approach involves starting with the general idea of what is required for the system to function and then work your way downward to more specific details on how the system will interact. This process involves identifying the entities involved and defining the possible attributes.

**Bottom-top design method**: The bottom-up approach begins with the specific details and moves up to the general. This is done by first identifying the data elements (items) and then grouping them together in data sets. In other words, this method first identifies the attributes, and then groups them to form entities.

**Question 3cii**

**Centralized**: The design can be carried out and represented in a somewhat simple database. Centralized design is typical of a simple or small database and can be successfully done by a single database administrator or by a small design team. This person or team will define the problems, create the conceptual design, verify the conceptual design with the user views, and define system processes and data constraints to ensure that the design complies with the organization's goals. That being said, centralized design is not limited to small companies. Even large companies can operate within the simple database environment.

**Decentralized:** Decentralized design might best be used when the data component of the system has a large number of entities and complex relations upon which complex operations are performed. This is also likely to be used when the problem itself is spread across many operational sites and the elements are a subset of the entire data set. In large and complex projects a team of carefully selected designers are employed to get the job done. This is commonly accomplished by several teams that work on different subsets or modules of the system. Conceptual models are created by these teams and compared to the user views, processes, and constraints for each module. Once all the teams have completed their modules they are all put aggregated into one large conceptual model.

**Question 4a**

**Entity Integrity:** Entity integrity is a design concept that states every record must have its own unique and independent unit. For example in a customer table each row should represent a unique customer. Absence of this causes data corruption.

So, the primary key plays a huge role as a unique identifier that guarantees entity integrity within a database.

CREATE TABLE customer(

Customer\_id SERIAL PRIMARY KEY,

Name VARCHAR(50),

Email VARCHAR(50)

);

**Referential Integrity:** Referential data integrity is achieved through the use of foreign keys, which link records in different tables. For instance, think about a business that has a Customers table and an Orders table. The CustomerID in the Orders table acts as a foreign key, referencing the CustomerID in the Customers table. This ensures every order is linked to a valid customer.

CREATE TABLE Customer (

CustomerID SERIAL PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

CREATE TABLE Orders (

OrderID SERIAL PRIMARY KEY,

OrderDate DATE,

Amount INT,

CustomerID INT REFERENCES Customer (CustomerID)

);

**Question 4b**

First is to Convert to 1NF of atomicity. Here we,ve to create two tables ( Course, Content).

Course Table:

| **Course\_id** | **Course** | **Price** |
| --- | --- | --- |
| 1 | Programming | $15 |
| 2 | Web development | $25 |
| 3 | Database | $50 |

Course Content table:

| **content\_id** | **content** | **course\_id** |
| --- | --- | --- |
| 1 | java | 1 |
| 2 | c++ | 1 |
| 3 | python | 1 |
| 4 | cobol | 1 |
| 5 | HTML | 2 |
| 6 | PHP | 2 |
| 7 | ASP | 2 |
| 8 | MySql | 3 |
| 9 | Oracle | 3 |
| 10 | Access | 3 |

CREATE TABLE course(

course\_id SERIAL PRIMARY KEY NOT NULL,

course VARCHAR(50) NOT NULL,

price INT NOT NULL

);

INSERT INTO course(course, price)

VALUES('Programming', 15),

('Web development', 25),

('Database', 50);

SELECT \* FROM course;

CREATE TABLE course\_content(

content\_id SERIAL PRIMARY KEY NOT NULL,

content\_name VARCHAR(50) NOT NULL,

course\_id INT REFERENCES course(course\_id)

);

drop table course\_content;

INSERT INTO course\_content(content\_name, course\_id )

VALUES('Java', 1),

('C++', 1),

('Python', 1),

('Cobol', 1),

('HTML', 2),

('PHP', 2),

('ASP', 2),

('MySql', 1),

('Oracle', 1),

('Access', 1);

SELECT \* FROM course\_content;

-- joining tables

SELECT \* FROM course

JOIN course\_content ON course\_content.course\_id = course\_content.course\_id;

--find all programming or Database courses that are less than $40