DBL – 02 Lab

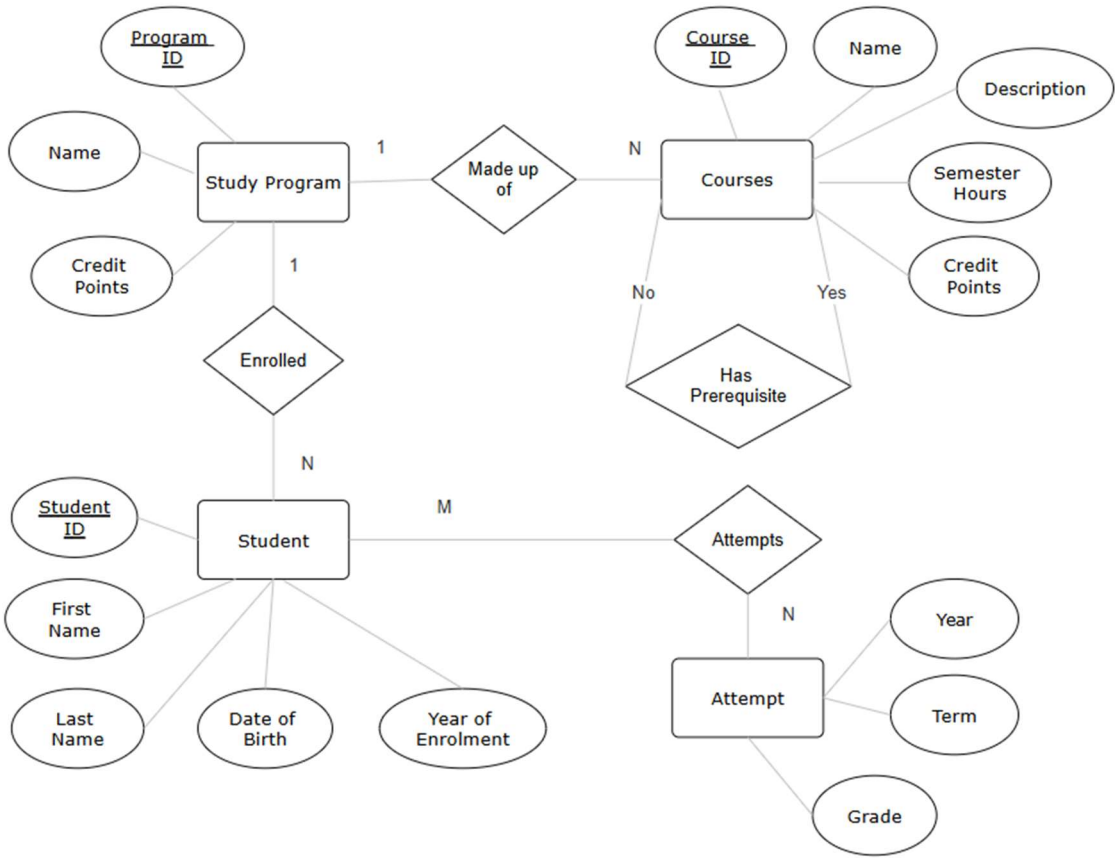
Team - 12

Thilakraj Soundararajan - 2705370

Mir Md Redwon Sagar - 2613747

Assignment 1: Student Information System

Task 1 & 2: ERM in Chen notation and MC notation



Task 3. Name two semantic integrity requirements, which make sense for the described model, but cannot be described in the Entity-Relationship-Diagram (e.g., a student cannot attempt a course he/she has already passed).

- A student can only attempt courses that are part of their enrolled study program. Enforcing that students don't take courses outside of their curriculum is a rule the ERD cannot enforce alone

- A student must complete all prerequisites before attempting a course. The ERD can show "prerequisite" relationships but cannot enforce the logic (e.g., checking completion before enrollment).

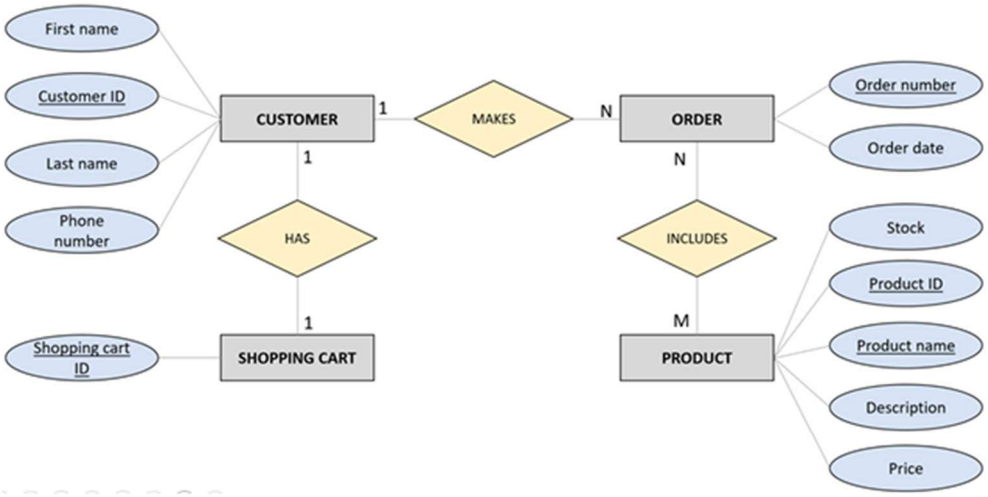
Task 4. Can you think of adding some additional composite, multivalued, or derived attributes in this example?

- Number of credits earned → Sum of passed course credits for a student
- GPA or Average Grade → Calculated from all course grades attempted

Assignment 2: ERD for an Online Shopping Platform

2. Assignment 2: ERD for an Online Shopping Platform

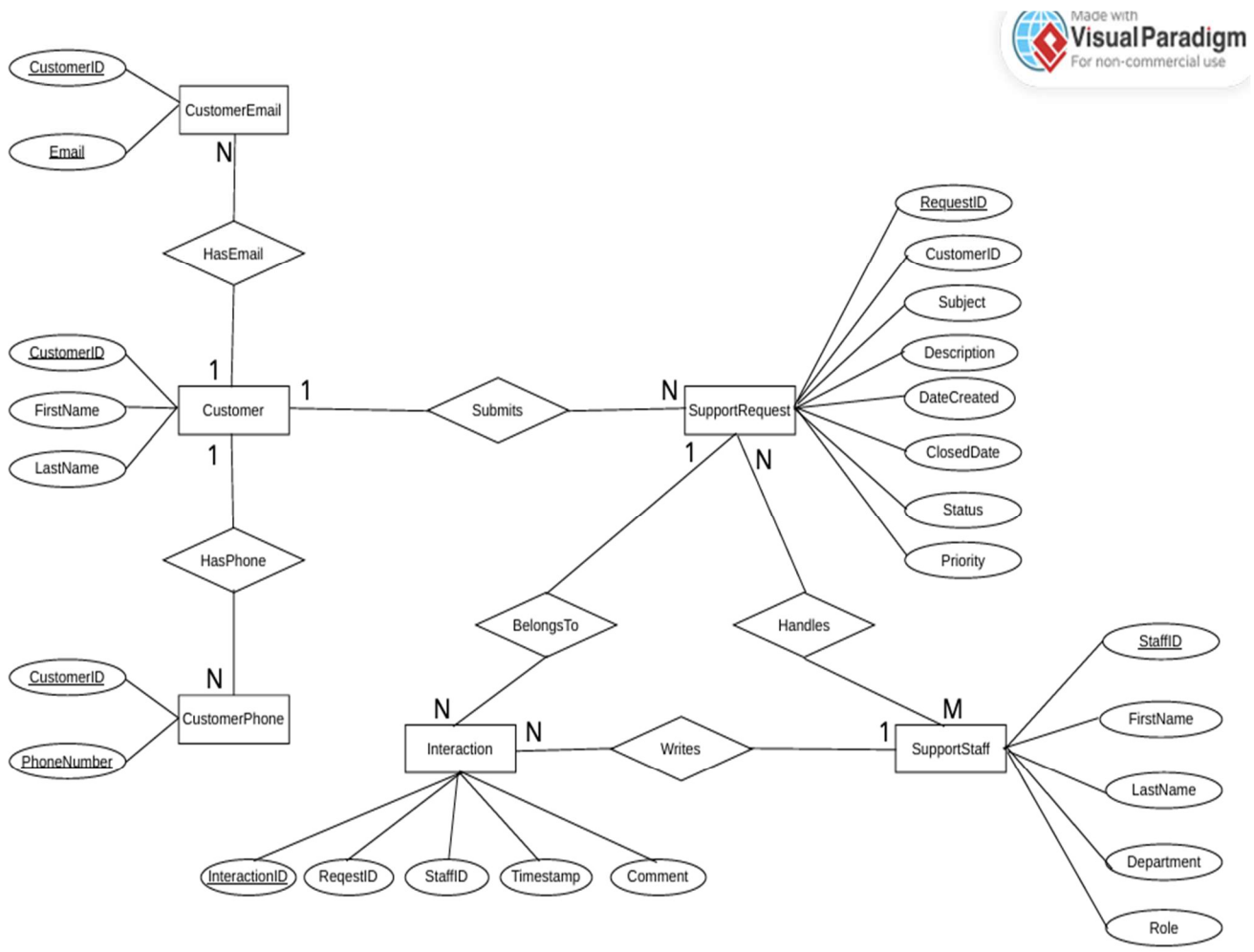
The following entity-relationship-diagram is given. Assume that the database is filled with data according to the ERM. Decide whether the following statements are either true (T), false (F), or undecidable/maybe (U). "U" is used for statements that can be either T or F depending on the stored data. Evaluate the statements based exclusively on the ERM and the restrictions it contains.



Number	Statement	Answer
01	A customer can have multiple shopping carts.	False
02	A product can be included in multiple orders.	True
03	Each shopping cart is associated with a specific order.	Unsure
04	An order can at most contain only one product.	False
05	A customer can place multiple orders.	True
06	Each order has a unique order number.	True
07	Each order must be associated with a customer.	True
08	A customer can place an order without a shopping cart.	Unsure
09	There are some customers who have placed more then ten orders.	Unsure
10	Every product can be contained several times in the same order.	Unsure
11	A product can be uniquely identified by the combination of the Product name and Product ID.	False

Assignment 3: ERD for a Technology Support Company

Task 1 & 3: ERM in Chen and MC notation



Task 2: Can you think of adding some additional composite, multivalued, or derived attributes in this example?

- In the support ticket management system, a composite attribute like Customer Address (Street, City, Postal Code) and a derived attribute like Ticket Resolution Time (calculated from Resolution Date minus Creation Date) can be added to enhance data organization and analysis.

Task 4: Relational Schema

```sql
-- Table to store customer details
CREATE TABLE Customer (
    CustomerID SERIAL PRIMARY KEY,              -- Unique ID for each customer (auto-increment)
    FirstName VARCHAR(50) NOT NULL,             -- Customer's first name (required)
    LastName VARCHAR(50) NOT NULL               -- Customer's last name (required)
);

-- Table to store multiple phone numbers per customer
CREATE TABLE CustomerPhone (
    CustomerID INT NOT NULL REFERENCES Customer(CustomerID),  -- Link to Customer table
    PhoneNumber VARCHAR(20) NOT NULL,                         -- Phone number
    PRIMARY KEY (CustomerID, PhoneNumber)                     -- Composite key to avoid duplicates
);

-- Table to store multiple email addresses per customer
CREATE TABLE CustomerEmail (
    CustomerID INT NOT NULL REFERENCES Customer(CustomerID),  -- Link to Customer table
    Email VARCHAR(100) NOT NULL,                              -- Email address
    PRIMARY KEY (CustomerID, Email)                           -- Composite key
);

-- Table to store support staff information
CREATE TABLE SupportStaff (
    StaffID SERIAL PRIMARY KEY,             -- Unique ID for staff (auto-increment)
    FirstName VARCHAR(50) NOT NULL,         -- First name (required)
    LastName VARCHAR(50) NOT NULL,          -- Last name (required)
    Department VARCHAR(50) NOT NULL,        -- Department (e.g., IT, Networking)
    Role VARCHAR(50) NOT NULL               -- Role (e.g., 1st Level Support)
);
```

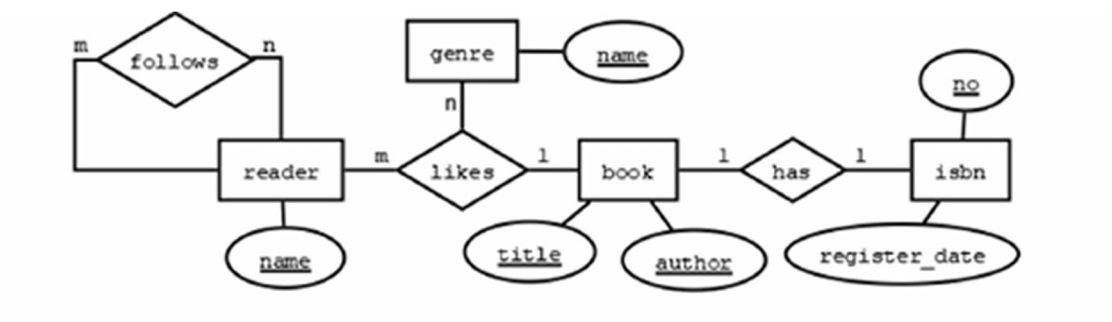
```sql
-- Table to store support requests (tickets)
CREATE TABLE SupportRequest (
    RequestID SERIAL PRIMARY KEY,                               -- Unique ticket ID
    CustomerID INT NOT NULL REFERENCES Customer(CustomerID),    -- Link to customer
    Subject VARCHAR(100) NOT NULL,                              -- Short title of the request
    Description TEXT,                                           -- Full description
    DateCreated DATE NOT NULL DEFAULT CURRENT_DATE,             -- Auto-fills current date
    ClosedDate DATE,                                            -- When the request was closed
    Status VARCHAR(20) NOT NULL CHECK (Status IN ('Open', 'In Progress', 'Closed')),  -- Ticket status
    Priority VARCHAR(20) NOT NULL CHECK (Priority IN ('Low', 'Medium', 'High'))       -- Ticket priority
);

-- M:N link table for which staff handles which requests
CREATE TABLE HandledRequest (
    StaffID INT NOT NULL REFERENCES SupportStaff(StaffID),      -- Staff assigned to handle
    RequestID INT NOT NULL REFERENCES SupportRequest(RequestID), -- Request being handled
    PRIMARY KEY (StaffID, RequestID)                            -- Prevent duplicate pairings
);

-- Table to store all interactions (comments/messages) on support requests
CREATE TABLE Interaction (
    InteractionID SERIAL PRIMARY KEY,                           -- Unique message ID
    RequestID INT NOT NULL REFERENCES SupportRequest(RequestID),-- Related support request
    StaffID INT NOT NULL REFERENCES SupportStaff(StaffID),      -- Staff who wrote it
    Timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,     -- When the comment was made
    Comment TEXT NOT NULL                                       -- The message itself
);
```

Assignment 4: RM for Favorite Books

Transform given the ERD to an equivalent relational model.



```sql
-- ENTITY: Reader
-- Each reader is uniquely identified by their name.
-- A reader can follow other readers and like books.
CREATE TABLE Reader (
    ReaderName VARCHAR PRIMARY KEY  -- Primary Key: unique reader name
);

-- ENTITY: Book
-- A book is uniquely identified by a combination of its title and author.
-- Composite primary key is used because both title and author are underlined in the ERD.
CREATE TABLE Book (
    Title VARCHAR NOT NULL,         -- Book title
    Author VARCHAR NOT NULL,        -- Book author
    PRIMARY KEY (Title, Author)     -- Composite key (Title + Author)
);

-- ENTITY: Genre
-- Genre is a simple entity with a single attribute: name.
CREATE TABLE Genre (
    GenreName VARCHAR PRIMARY KEY   -- Primary Key: unique genre name
);

-- ENTITY: ISBN
-- Each book has one ISBN number with a register date.
-- Note: ISBN is separate from Book, connected via a 1:1 relationship.
CREATE TABLE ISBN (
    No VARCHAR PRIMARY KEY,         -- ISBN number (Primary Key)
    RegisterDate DATE               -- Date when the ISBN was registered
);



-- RELATIONSHIP: Has (1:1 between Book and ISBN)
-- This table links each Book (identified by Title and Author) to exactly one ISBN number.
-- Since the Book entity uses a composite primary key (Title, Author), we use the same here.
-- ISBNNo is not part of the primary key because it's already unique and referenced as a foreign key.

CREATE TABLE Has (
    Title VARCHAR,                                      -- Book title (part of composite key)
    Author VARCHAR,                                     -- Book author (part of composite key)
    ISBNNo VARCHAR UNIQUE,                              -- ISBN number (must be unique, 1:1 mapping)

    PRIMARY KEY (Title, Author),                        -- Composite key from the Book table
    FOREIGN KEY (Title, Author) REFERENCES Book(Title, Author), -- Link to Book table
    FOREIGN KEY (ISBNNo) REFERENCES ISBN(No)            -- Link to ISBN table
);
```

```sql


-- RELATIONSHIP: Follows (Reader ↔ Reader) [M:N Self-Relationship]
-- A reader can follow many other readers, and be followed by many.
-- This table tracks who follows whom.
CREATE TABLE Follows (
    Follower VARCHAR REFERENCES Reader(ReaderName),   -- Reader who follows
    Following VARCHAR REFERENCES Reader(ReaderName),  -- Reader being followed
    PRIMARY KEY (Follower, Following)                 -- Composite key
);

-- RELATIONSHIP: Likes (Ternary Relationship: Reader ↔ Genre ↔ Book)
-- A reader likes a book in the context of a genre.
-- This is a 3-way relationship requiring a ternary junction table.
CREATE TABLE Likes (
    ReaderName VARCHAR REFERENCES Reader(ReaderName),   -- Reader who likes
    GenreName VARCHAR REFERENCES Genre(GenreName),      -- Genre of the book liked
    Title VARCHAR,                                      -- Book title (part of PK & FK)
    Author VARCHAR,                                     -- Book author (part of PK & FK)
    PRIMARY KEY (ReaderName, GenreName, Title, Author), -- Composite key for uniqueness
    FOREIGN KEY (Title, Author) REFERENCES Book(Title, Author)
);

```