DBMS - 1

Class Assignment 1

SUBMITTED BY:
SHAHZANEER AHMED

REGISTRATION NUMBER:
SP21-BCS-087

SUBMITTED TO:
Dr. Basit Raza

DATE OF SUBMISSION:
October 20, 2022

// Question No 1

Suppose a small database is being built for customers for placing orders through specific agent. For that database designer has designed a database schema. The database contains data about customers, agents, and orders. It contains only three relations:

- The first relation titled *Agents* stores information related to agents. It stores *agent code*, the *name, working area, commission, phone number,* and the *country* for agents. All the attributes are mandatory. In addition, *agent code* should always start with A and commission data type is *real*.
- The second relation titled *Customer* stores information of customer. It stores *Customer code, name, city, working area, country, grade (value ranges from 0-5), opening amount (should be >3000), receive amount, payment amount, outstanding amount,* and *agent code*. *Customer code* should start with *C%*. Since *Customer* contains attributes from *Agent's* relation too, so it is expected that any changes (Update, Delete) made in *Agents* will automatically be reflected here.
- The third relation titled *Orders* stores information about orders. It stores *order number, order amount, advance amount, order date, customer code, agent code* and *order description,* of order placed. Since *Orders* contains attributes from *Agents* and *Customer* relations too, so it is expected that any changes (Update, Delete) made in *Agents* and *Customer* will automatically be set to *NULL* here.

Identify the primary keys in each of the above relations. Also identify all other integrity constraints mentioned by the designer that must be imposed while writing a DDL script.

ANSWER:

- The primary key in Agent's relation will be agent_code because it is not only unique but also will not be null in the table because it is the thing that identifies each agent different from each other. The constraints mentioned in the Agent's relation are as follows:
  - **Agent's code** should always be char or string type and it should always start with character **A%**.
  - The **commission** attribute in the Agent's relation should always be real data type.

```
CREATE TABLE Agents (
agent_code varchar(255) NOT NULL CHECK(agent_code LIKE 'A%'),
agent_name varchar(255) NOT NULL,
working_area varchar(255) NOT NULL,
commission REAL NOT NULL,
```

phone_no varchar(255) NOT NULL,
PRIMARY KEY (agent_code));

- The primary key in Customer relation will be customer_code because it is not only unique but also will not be null in the table because it is the thing that identifies each customer different from each other. The constraints mentioned in the customer relation are as follows:
  - **Customer code** should always be char or string type and it should always start with character **C%**.
  - The **grade** attribute in the customer relation should always be int data type and its values should only lie between 0-5.
  - The **opening_amount** attribute in the customer relation should always be greater than 5000.
  - As customer relation is containing the agent_code as it's foreign key then any CRUD operation which occurred in the Agent's relation table should be reflected in the customer table.

```
CREATE TABLE Customer (
customer_code varchar(255) NOT NULL CHECK(customer_code LIKE 'C%'),
cust_name varchar(255),
cust_city varchar(255),
working_area varchar(255),
cust_country varchar(255),
grade int CHECK(grade >= 0 and grade <= 5),
opening_amt int CHECK(opening_amt > 3000),
receive_amt int,
payment_amt int,
outstanding_amt int,
phone_no varchar(255),
agent_code varchar(255) NOT NULL REFERENCES agents(agent_code) ON DELETE CASCADE, PRIMARY KEY (customer_code)
);
```

- The primary key in Orders relation will be order_number because it is not only unique but also will not be null in the table because it is the thing that identifies each order different from each other. The constraints mentioned in the Orders relation are as follows:

  - As Orders relation is containing the agent_code and customer_code as its foreign key then any CRUD operation which occurred in the Agent's relation or in the Customer relation table should be reflected in the customer table like if any update or delete operation occurred in the above tables then automatically order is set to be null valued.

```
CREATE TABLE Orders (
order_number int NOT NULL PRIMARY KEY,
 ord_amount real,
 advance_amount real,
ord_date DATE,
cust_code varchar(255) NOT NULL REFERENCES customer(customer_code) ON DELETE SET NULL,
 agent_code varchar(255) NOT NULL REFERENCES agents(agent_code) ON DELETE SET NULL, ord_description varchar(255) );
```

// Question No 2

Consider the following instances of each of the above relations following the same schema as described in Question 1. For each of the following modifications, what changes will be made to the database and specify if they will violate any constraint? If yes, which constraint?

a) Inserting (A005, Abey, New York, 0.15, 044-26578911) in *Agents*
Explanation:
By inserting the above instances in the agents table will add a new record in it and it does not violate any constraint.

b) Inserting (013, Leena, London, 0.12, 078-225278911) in *Agents*
Explanation:
By inserting the above instance in the agents table that will make some changes are the agent_code and the phone_number and it does violate the constraints that is the agent_code is not starting with the A% although the phone_number has no constraint,

but it is odd from any other phone number in the agents table because it is 13 number and rest of them are 12 number.

c) Inserting (C00024, Jhon, New York, New York, USA, 6, 5000, 4000, 2000, 1000, 2000, 222222, A005) into *Customer*
Explanation:
This change will not be made as it violates the constraints. It has more than required column values and grade should be between 0 to 5.

d) Inserting (C00025, Abel, New York, New York, USA, 3, 1000, 2000, 4000, 1000, 2000, 444444, A0017) into *Customer*
Explanation:
This change will not be made as it violates the constraints. It has more than required column values and opening amount should be greater than 3000.

e) Updating (Agent number only A006) in *Agents* to (A0016)
Explanation:
Row will be updated in Agent's relation as it is following all constraints. Values in others against it will be set to null.

f) Updating (*Customer code* C0006) in *Customer* to (C00026)
Explanation:
Row will be updated in Customer relation as it is following all constraints. Values in order will be set to null against it.

g) Deleting (where *agent code* = A004) from *Agents*
Explanation:
By deleting agent A004, the above instances with the agent_code table will be removed, and the change it will make is that the customer table where agent_code is the foreign key will be set to null and the order table relation where agent_code and customer_code is the foreign key will be set to null and it does not violate any constraint.

// Question No 3

Write relational algebra queries over the database of Question 1. You are going to use the same instance of the database depicted in Question 2.

a. Find all the Agents who have commission greater than 0.13.
Query:
$(\sigma \text{ commision} > 0.13(\text{Agents}))$

b. Find Agent code and name of all agents who are working in Bangalore.

Query:

π agent_code,agent_name(σ working_area='Bangalore'(Agents))

c. Find all customers who have grade equal to 2.

Query:

(σ grade=2(Customers))

d. Find Agent name for customer C00007.

Query:

π agent_name(σ cust_code='C00007'(Customers×Agents))

e. Find customer code, customer name and city for all customers who are in USA and having outstanding amount greater than 3000

Query:

π cust_code,cust_name,cust_city(σ cust_country='USA' ∧ outstanding_amt>3000(Customers))

f. Find agent name and customer name for order number 200108.

Query:

π agent_name,cust_name(σ order_no=200108((Customers×Agents)⋈Order))

g. Find count of customers assigned to each agent.

Query:

agent_code π COUNT agent_code (Customer) select agent_code, count(agent_code) as customer_count from customer GROUP BY agent_code;

h. Display all orders between 01-01-2008 and 01-05-2008. (Date format is DD-MM-YY)

Query:

σ ord_date > date('01-01-08') ∧ ord_date < date('01-05-08') (Order)