Algorithm Bootcamp Final Season

Theory Examination

By: Matthew Adrianus Mulyono

Entity Relationship Diagram

1. **From the above diagram, list all of the objects including its attributes! (.pdf)**

- Users:
  a. UserId char(5)
  b. FirstName varchar(255)
  c. LastName varchar(255)
  d. School varchar(255)
  e. Address varchar(255)
  f. Email varchar(255)
  g. PhoneNumber varchar(12)
  h. Location varchar(255)
  i. Gender varchar(10)
  j. DateOfBirth date
- Friends:
  a. FriendId char(5)
  b. UserId char(5)
- PageLikes:
  a. PageId char(5)
  b. UserId char(5)
- Pages:
  a. PageId char(5)
  b. PageName varchar(255)
  c. PageContent varchar(max)
- Posts:
  a. PostId char(5)
  b. UserId char(5)
  c. PostDate date
  d. PostContent varchar(max)
- PostLikes:
  a. PostId char(5)
  b. UserId char(5)
- Photos:
  a. PhotoId char(5)
  b. PostId char(5)
  c. ImageContent varchar(max)
- Shares:
  a. PostId char(5)
  b. UserId char(5)
- Comments:
  a. CommentId char(5)
  b. PostId char(5)

c. UserId char(5)
d. CommentDate date
e. CommentContent varchar(max)

- CommentLikes:
  a. CommentId char(5)
  b. UserId char(5)

2. Determine the relation between every object, specify the master and child table! (.pdf)

a. Users = Master table to Friends, PageLikes, Posts, PostLikes (through Posts), Shares (through Posts), Comments (through Posts), CommentLikes(through Posts and Comments)
  - Relation: A user can have zero to many page likes (depends on if other user like their pages), They can have 0 to many friends (again, depends on how many user wants to be friends with the user), and they can have zero to many posts.

b. Friends = Child Table to Users
  - Relation: A friend can be befriended by zero to many users.

c. PageLikes = Child Table to Users and Pages
  - Relation: Page Likes can only be owned by only one user and one page

d. Pages = Master Table to PageLikes
  - Relation: A Page can have zero to many page likes

e. Posts: Child Table to Users, Master Table to PostLikes, Photos, Shares, and Comments.
  - Relation: A Post can only be owned by one user, can have zero to many post likes, can have zero to many photos embedded to it (like in Instagram or facebook, there might be many photos in one posts), can have zero to many shares, and can have zero to many comments.

f. PostLikes: Child Table to Users and Posts
  - Relation: Postlikes are only owned by a single post.

g. Photos: Child Table to Posts
  - Relation: Photos can only be owned by a single post, eventhough different posts may have the same photos (different link so its not the same)

h. Shares: Child Table to Users and Posts
  - Relation: A Share can only be derived from a singular post (the user cant share multiple posts at the same time)

i. Comments: Child Table to Users and Posts, Master Table to Comments
  - Relation: A Comment can only be owned by a one post (although many comments might contain the same contents), and a comment can have zero to many comment likes.

j. CommentLikes: Child Table to Users and Comments
  - Relation: A Comment Like can only be owned by a single comment.

3. For each object, decide its constraint and specify the reason in detail! (.pdf)

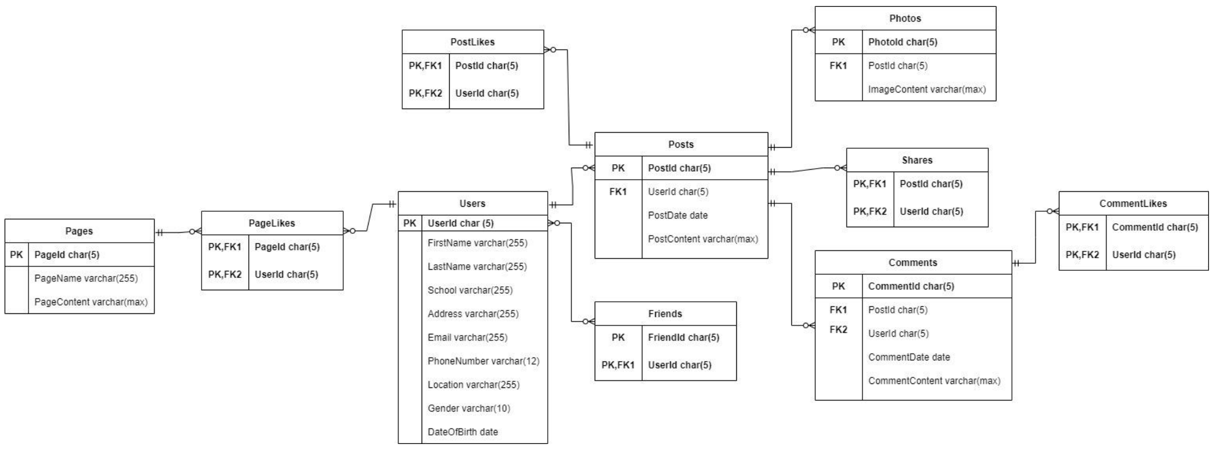
- Users:
  k. UserId char(5) Primary Key MUST BE USXXX (X is number)
  l. FirstName varchar(255) NOT NULL
  m. LastName varchar(255) NOT NULL
  n. School varchar(255) NOT NULL

o. Address varchar(255) NOT NULL
p. Email varchar(255) NOT NULL must have @
q. PhoneNumber varchar(12) NOT NULL must contain numbers
r. Location varchar(255) NOT NULL
s. Gender varchar(10) must be between Male and Female
t. DateOfBirth date NOT NULL

- Friends:
  c. FriendId char(5) must be FRXXX (X is number)
  d. UserId char(5) must be USXXX (X is number)
- PageLikes:
  c. PageId char(5) must be PGXXX (X is number)
  d. UserId char(5) must be USXXX (X is number)
- Pages:
  d. PageId char(5) must be PGXXX (X is number)
  e. PageName varchar(255) NOT NULL
  f. PageContent varchar(max)
- Posts:
  e. PostId char(5) must be POXXX(X is number)
  f. UserId char(5) must be USXXX (X is number)
  g. PostDate date NOT NULL
  h. PostContent varchar(max)
- PostLikes:
  c. PostId char(5) must be POXXX (X is number)
  d. UserId char(5) must be USXXX (X is number)
- Photos:
  d. PhotoId char(5) must be PHXXX (X is number)
  e. PostId char(5) must be POXXX (X is number)
  f. ImageContent varchar(max) NOT NULL (can be like a link to a cloud)
- Shares:
  c. PostId char(5) must be POXXX (X is number)
  d. UserId char(5) must be USXXX (X is number)
- Comments:
  f. CommentId char(5) must be COXXX (X is number)
  g. PostId char(5) must be POXXX (X is number)
  h. UserId char(5) must be USXXX (X is number)
  i. CommentDate date NOT NULL
  j. CommentContent varchar(max)
- CommentLikes:
  c. CommentId char(5) must be COXXX (X is number)
  d. UserId char(5) must be USXXX (X is number)

// The content can be null == like an empty post :)

4. **Draw the above diagram in "ERD format" which includes the data types, primary and foreign key, and relation between objects. Please choose appropriate tools, we recommend using Visual Paradigm. (.jpeg)**



The diagram will also be included in the folder

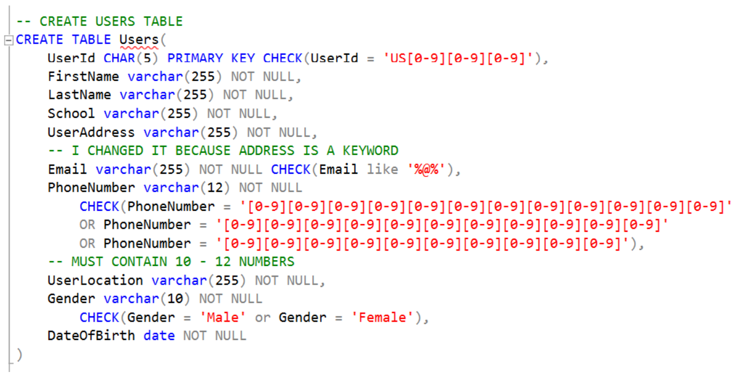
Data Definition Language

1. **Explain what data integrity is and how do we maintain it in SQL Server! (.pdf)**

   Data integrity is the accuracy and consistency of data as a database is being used. We can maintain data integrity using FOREIGN KEYS, and keyword of ON UPDATE [CASCADE|SET NULL|DEFAULT] and ON DELETE [CASCADE|SET NULL|DEFAULT]. When the user updates/deletes an element from a parent table, the child tables will not be affected unless we use these keywords, this can destroy data integrity, that is why it is necessary to add the keywords, where CASCADE means that if a data is updated/deleted, then the data in the child table will also be updated/deleted. SET NULL means when the data is updated/deleted, then the connected data in the child table will be set to null. Lastly DEFAULT means that if a data is updated/deleted, then the value of the connected data in the child table will be reset to default.

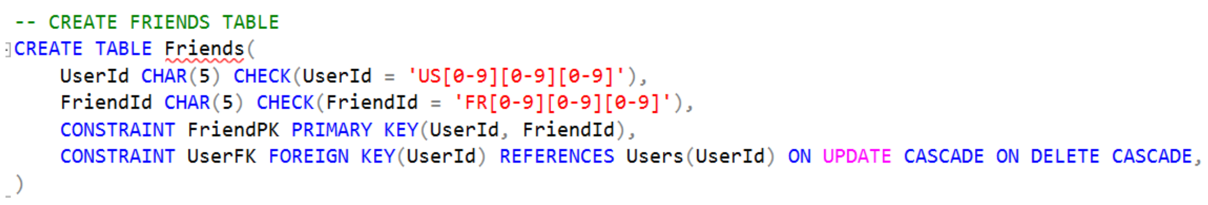
2. **Explain the difference and give example for: primary key, foreign key, and composite key! (.pdf)**

- Primary Key: An attribute that acts as a unique identifier for each row inside a table.
- Foreign Key: A key in a table that refers to other tables, on which in that other table, it is a primary key. A foreign key can be used as a composite key.
- Composite Key: A Type of Primary Keys that are composed of multiple attributes.



```
-- CREATE USERS TABLE
CREATE TABLE Users(
    UserId CHAR(5) PRIMARY KEY CHECK(UserId = 'US[0-9][0-9][0-9]'),
    FirstName varchar(255) NOT NULL,
    LastName varchar(255) NOT NULL,
    School varchar(255) NOT NULL,
    UserAddress varchar(255) NOT NULL,
    -- I CHANGED IT BECAUSE ADDRESS IS A KEYWORD
    Email varchar(255) NOT NULL CHECK(Email like '%@%'),
    PhoneNumber varchar(12) NOT NULL
        CHECK(PhoneNumber = '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
        OR PhoneNumber = '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
        OR PhoneNumber = '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
    -- MUST CONTAIN 10 - 12 NUMBERS
    UserLocation varchar(255) NOT NULL,
    Gender varchar(10) NOT NULL
        CHECK(Gender = 'Male' or Gender = 'Female'),
    DateOfBirth date NOT NULL
)
```

Primary Key Example

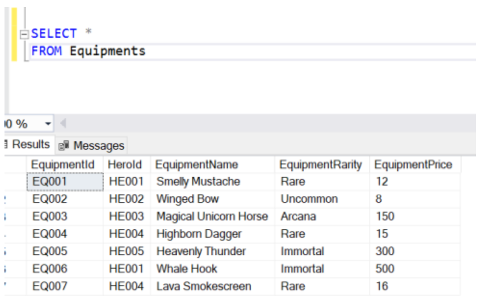


```sql
-- CREATE FRIENDS TABLE
CREATE TABLE Friends(
    UserId CHAR(5) CHECK(UserId = 'US[0-9][0-9][0-9]'),
    FriendId CHAR(5) CHECK(FriendId = 'FR[0-9][0-9][0-9]'),
    CONSTRAINT FriendPK PRIMARY KEY(UserId, FriendId),
    CONSTRAINT UserFK FOREIGN KEY(UserId) REFERENCES Users(UserId) ON UPDATE CASCADE ON DELETE CASCADE,
)
```

Composite Key and Foreign Key Examples

3. **Explain the following terms and give example: BEGIN TRAN, COMMIT, and ROLLBACK! (.pdf)**

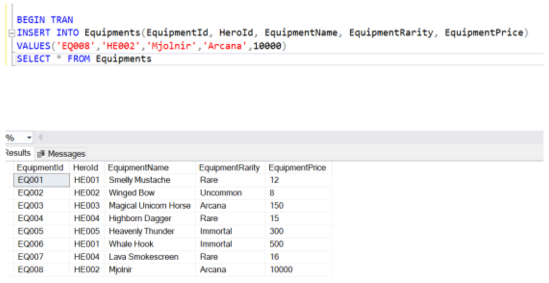
- BEGIN TRAN: This keyword is similar to setting up a checkpoint, after the user executes BEGIN TRAN, all the executed queries after executing BEGIN TRAN can be reset by executing ROLLBACK or COMMIT.
- COMMIT: This keyword saves the history of all executed queries since the user executes BEGIN TRAN, therefore they can't ROLLBACK anymore.
- ROLL BACK: This keyword returns the condition of the database back to how it was before the user execute BEGIN TRAN.



```sql
SELECT *
FROM Equipments
```

EquipmentId	HeroId	EquipmentName	EquipmentRarity	EquipmentPrice
EQ001	HE001	Smelly Mustache	Rare	12
EQ002	HE002	Winged Bow	Uncommon	8
EQ003	HE003	Magical Unicorn Horse	Arcana	150
EQ004	HE004	Highborn Dagger	Rare	15
EQ005	HE005	Heavenly Thunder	Immortal	300
EQ006	HE001	Whale Hook	Immortal	500
EQ007	HE004	Lava Smokescreen	Rare	16

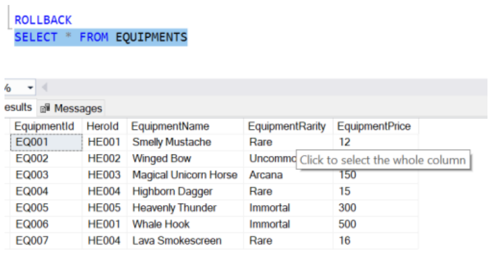
Before BEGIN TRAN



```sql
BEGIN TRAN
INSERT INTO Equipments(EquipmentId, HeroId, EquipmentName, EquipmentRarity, EquipmentPrice)
VALUES('EQ008','HE002','Mjolnir','Arcana',10000)
SELECT * FROM Equipments
```

EquipmentId	HeroId	EquipmentName	EquipmentRarity	EquipmentPrice
EQ001	HE001	Smelly Mustache	Rare	12
EQ002	HE002	Winged Bow	Uncommon	8
EQ003	HE003	Magical Unicorn Horse	Arcana	150
EQ004	HE004	Highborn Dagger	Rare	15
EQ005	HE005	Heavenly Thunder	Immortal	300
EQ006	HE001	Whale Hook	Immortal	500
EQ007	HE004	Lava Smokescreen	Rare	16
EQ008	HE002	Mjolnir	Arcana	10000

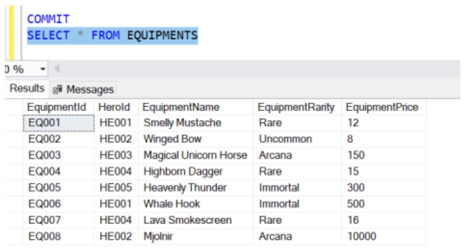
AFTER BEGIN TRAN



```sql
ROLLBACK
SELECT * FROM EQUIPMENTS
```

EquipmentId	HeroId	EquipmentName	EquipmentRarity	EquipmentPrice
EQ001	HE001	Smelly Mustache	Rare	12
EQ002	HE002	Winged Bow	Uncommon	8
EQ003	HE003	Magical Unicorn Horse	Arcana	150
EQ004	HE004	Highborn Dagger	Rare	15
EQ005	HE005	Heavenly Thunder	Immortal	300
EQ006	HE001	Whale Hook	Immortal	500
EQ007	HE004	Lava Smokescreen	Rare	16

After ROLLBACK



```sql
COMMIT
SELECT * FROM EQUIPMENTS
```

EquipmentId	HeroId	EquipmentName	EquipmentRarity	EquipmentPrice
EQ001	HE001	Smelly Mustache	Rare	12
EQ002	HE002	Winged Bow	Uncommon	8
EQ003	HE003	Magical Unicorn Horse	Arcana	150
EQ004	HE004	Highborn Dagger	Rare	15
EQ005	HE005	Heavenly Thunder	Immortal	300
EQ006	HE001	Whale Hook	Immortal	500
EQ007	HE004	Lava Smokescreen	Rare	16
EQ008	HE002	Mjolnir	Arcana	10000

AFTER COMMIT

4. **Create all of the tables above according to your answer in the previous section! (.sql)**