*Kevice // Kevin Bryan* 🙃

# ***ERD***

1)

## Object

- Attribute_1
- Attribute_2
- Dst.

## Users

- UserID
- First Name
- Last Name
- School
- Address
- Email
- PhoneNumber
- Location
- DateofBirth
- Gender

## Pages

- PageID
- PageName
- PageContent

## PageLikes

- UserID
- PageID
- PageLikeID

## Friends

- FriendID
- UserID

## Posts

- UserID
- PostID
- PostContent
- PostDate

## PostLikes

- PostID
- UserID
- PostLikeID

## Photos

- PhotoID
- PostID
- ImageContent

## Shares

- PostID

- UserID
- ShareID

## Comments

- CommentDate
- PostID
- CommentID
- UserID
- CommentContent

## ComentLikes

- CommentID
- UserID
- CommentLikesID

2)

1 User can have 0 to many friends. Master is user, child is friends

1 User can have 0 to many page likes, master is user, child is page likes

1 Page can have 0 to many page likes, master is page, child is page likes

1 User can have 0 to many posts, master is user, child is posts

1 Post can have 0 to many comments, master is post, child is comment

1 Post can have 0 to many photos, master is post, child is photos

1 Post can have 0 to many postlikes, master is post, child is postlikes

1 Comment can have 0 to many commentlikes, comment is master, commentlikes is child

3)

***User***

First Name cannot be NULL, Last name can be null in case user may only have 1 Name

UserID cannot be NULL, since it is PK and is use for searching the data

Email cannot be NULL and must contain '@'

Phone Number cannot be NULL and can only be 8+ digit and starts with '08'

Email & PhoneNumber cannot be NULL because it is essential for a user to have that for logging in

Gender must be either male/female so it is varchar[6]

DOB must be in YYYY/MM/DD format, where MM is <= 12 and DD is <= 31

*Kevice // Kevin Bryan* 🙃

FirstName, LastName, School, Address, Email, Location is in varchar[255] as we can’t determine the length

UserID is in char[5] as it is in the format USXXX where X is 0-9

PhoneNumber is also in varchar of length [10] as it is the max digit, it is in varchar not int because it starts with 0 which is 08-xxx-xx so it must be in varchar.

DOB is stored in the datatype date

Constraint:

PK = UserID karena yang paling unique

***Page Likes***

UserID & PageID cannot be NULL

PageID is in char[5] as it is in the format PGXXX where X is 0-9

PageLikeID isi in char[6]

Constraint:

PK = PageLikeID

FK1 = UserID

FK2 = PageID

Disini membuat PK baru yaitu pagelike id yaitu id yang mereprentasikan hubungan FK1 dan FK2, sebenarnya ingin mencoba membuat Composite Key tetapi tidak bisa di SQL

***Pages***

PageID cannot be NULL

PageContent must contain https://, as the content is stored in url format

PageConent is in varchae[255] as we cant determine how long the url will be

Constraint:

PK = PageID

***Friends***

FriendID and UserID cannot be NULL

FriendID is in char[5] as it is in the format FRXXX where X is 0-9

Constraint:

PK = FriendID

FK = UserID

### ***Posts***

UserID cannot be NULL

PostDate must be in YYYY/MM/DD format, where MM is <= 12 and DD is <= 31

PostDate is in date data type

PostID is in char[5] as it is in the format POXXX where X is 0-9

PK = PostID

FK = UserID

### ***PostLikes***

PostID & UserID cannot be NULL

Constraint:

PK = PostLikeID

FK1 = UserID

FK2 = PageID

Disini membuat PK baru yaitu PostLikeID yaitu id yang mereprentasikan hubungan FK1 dan FK2, sebenarnya ingin mencoba membuat Composite Key tetapi tidak bisa di SQL

### ***Photos***

PhotoID & PostID cannot be NULL

Image Content must contain https://, as the content is stored in url format

Imagecontent is in the varchar [255] datatype as we cant determine the length

PhotoID is in char[5] as the it is in PHXXX format

PK = PhotoID

FK = PostID

### ***Shares***

PostID and UserID cannot be NULL

Constraint:

PK = ShareID

FK1 = PostID

FK2 = UserID

Disini membuat PK baru yaitu ShareID yaitu id yang mereprentasikan hubungan FK1 dan FK2, sebenarnya ingin mencoba membuat Composite Key tetapi tidak bisa di SQL

***Comments***

PostID, CommentID, UserID cannot be NULL

CommentDate must be in YYYY/MM/DD format, where MM is <= 12 and DD is <= 31

CommentDate is in date datatype

CommentID is in char[5] data type as it is in the format COXXX

CommentContent is in varchar[255] as we can’t determine the length of the url

***CommentLikes***

CommentID & UserID can’t be NULL

Constraint:

PK = CommentLikeID

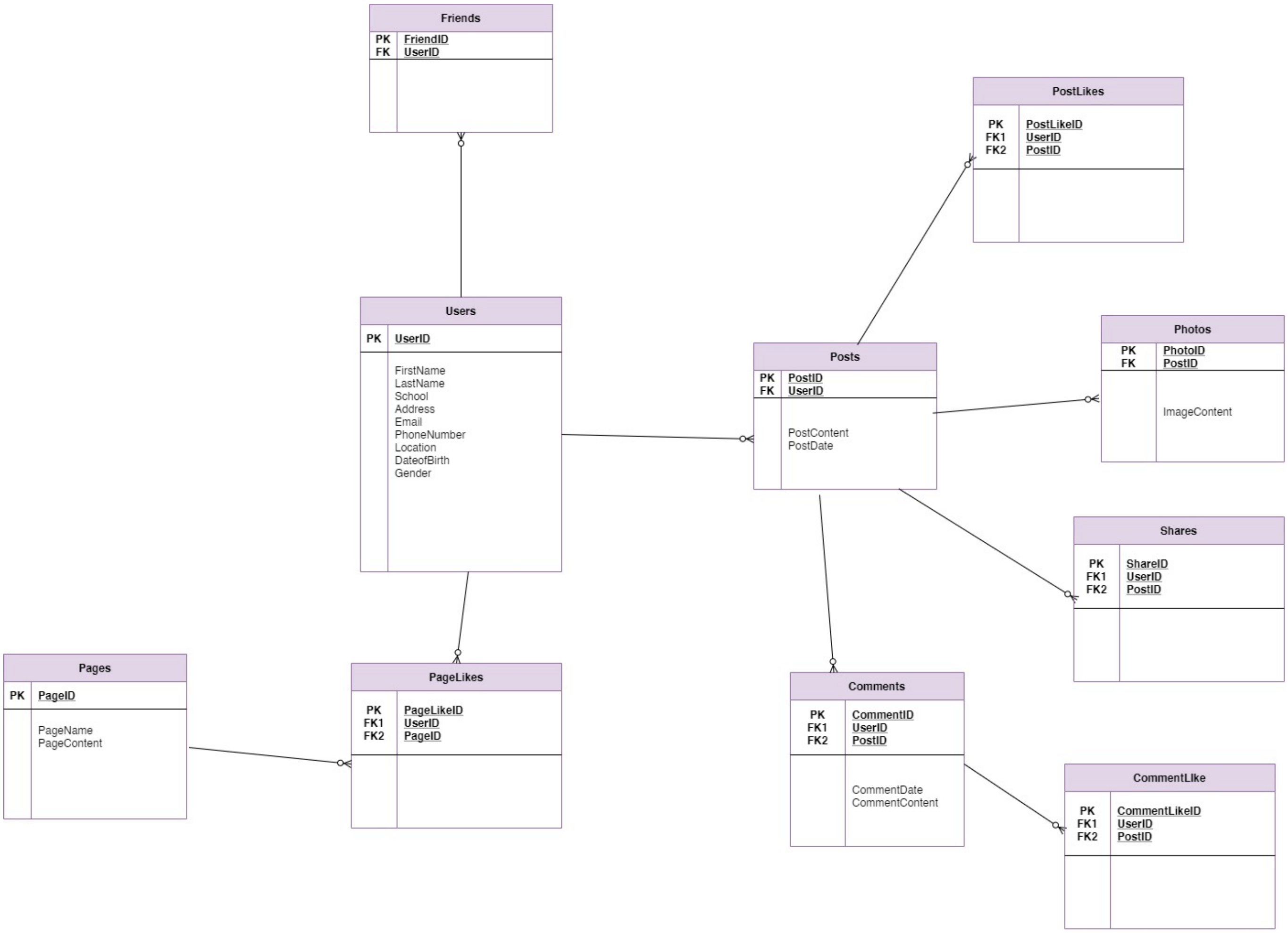
FK1 = UserID

FK2 = CommentID

Disini membuat PK baru yaitu CommentLike ID yaitu id yang mereprentasikan hubungan FK1 dan FK2, sebenarnya ingin mencoba membuat Composite Key tetapi tidak bisa di SQL

*Kevice // Kevin Bryan*

4.



*Kevice // Kevin Bryan* 🙃

## ***DDL***

1. Data integrity is the accuracy, completeness, and consistency of data and it refers to the safety of the data in regards to regulatory compliance and security. Data integrity is maintained by a collection of processes, rules, and standards during the design phase.

   Data integrity is important to maintain because it ensures recoverability and searchability, traceability and connectivity. Protecting the validity and accuracy of a data increases the stability and performance while improving reusability and maintainability

2. Primary Key is an attribute that uniquely identifies that object, it cannot be a duplicate, for example in BINUS each student has their own unique NIM and this is the primary key of the object Mahasiswa.

   Foreign key is an attribute that creates a relationship between 2 object, the purpose is to maintain the data integrity and allow navigation, for example in an app like digital market there is transaction report, in transaction report there will also be the Product ID to know what product is being sold, the Product ID is the FK that connects the object Product and TransactionReport.

   A composite key is a combination of 2 or mote columns in a table that can be used to uniquely identify each row in the table, the uniqueness is guaranteed when the two attributes or column is combined, an example is for PostLikes in Instagram, it is unique only when the PostID and UserID who likes the post is combine.

3. BEGIN TRAN

   Marks the starting point of a local transaction (acts as a savepoint).

   COMMIT

   Ends the transaction

   ROLLBACK

   Rolls back the transaction that have been done to before the savepoint.

   EXAMPLE:

   Begin Tran

   Update Heroes

   Set HeroID = ‘N’

   Where HeroID = ‘M’

   ROLLBACK

   So, when we run begin tran the HeroID will change to M if we do not rollback or shoot to only commit or end it. But if we run Rollback after running begin tran , the HeroID will revert back to M.