THEORY EXAMINATION

A. ERD

1. Berikut list table dan atributnya:
    1) Users
        a. User ID
        b. First Name
        c. Last Name
        d. School
        e. Address
        f. Email
        g. Phone Number
        h. Location
        i. Date of Birth
        j. Gender

    2) Friends
        a. Friend ID
        b. User ID

    3) Pages
        a. Page ID
        b. Page Name
        c. Page Content

    4) Page Likes
        a. Page ID
        b. User ID

    5) Posts
        a. Post ID
        b. Post Content

    c. Post Date
    d. User ID

6) Post Likes
    a. Post ID
    b. User ID

7) Photos
    a. Photo ID
    b. Image Content
    c. Post ID

8) Shares
    a. Post ID
    b. User ID

9) Comments
    a. Comment ID
    b. Comment Date
    c. Comment Content
    d. Post ID
    e. User ID

10) Comment Likes
    a. Comment ID
    b. User ID

2. Format “master-child”.
    1) Users-Friends
    2) Users-Page Likes
    3) Users-Posts

4) Pages-Page Likes
5) Posts-Post Likes
6) Posts-Photos
7) Posts-Shares
8) Posts-Comments
9) Comments-Comment Likes

3. List table
    1) Users
        a. User ID (PK, char(5), not null)

           User ID menjadi PK karena harus unik dan sebagai pengenal dari table Users. Char panjang 5 karena User ID berbentuk 'USXXX'. Not null karena PK tidak boleh kosong.
        b. First Name (varchar, not null)

           Berbentuk varchar karena nama depan dari seseorang tidak bisa ditebak panjangnya (tidak pasti). Not null karena setiap orang pasti memiliki nama, setidaknya nama depan.
        c. Last Name (varchar, nullable)

           Berbentuk varchar karena nama belakang dari seseorang tidak bisa ditebak panjangnya (tidak pasti). Nullable karena bisa saja namanya hanya terdiri dari satu kata.
        d. School (varchar, nullable)

           Berbentuk varchar karena nama sekolah seseorang tidak bisa ditebak panjangnya (tidak pasti). Nullable karena mungkin saja dia tidak bersekolah.
        e. Address (varchar, nullable)

           Berbentuk varchar karena alamat seseorang tidak bisa ditebak panjangnya (tidak pasti). Nullable karena untuk menghargai privasi masing-masing user sehingga tidak harus diisi.

        f. Email (varchar, not null, check like '%@%')

Berbentuk varchar karena email seseorang tidak bisa ditebak panjangnya (tidak pasti). Not null karena email dibutuhkan sebagai *contact person* si user. Kemudian terdapat check untuk memastikan bahwa yang diinput oleh user merupakan sebuah email.

g. Phone Number (varchar, not null, check like '+62%')
   Berbentuk varchar karena nomor telepon seseorang tidak bisa ditebak panjangnya (tidak pasti). Not null karena nomor telepon diperlukan juga untuk mengontak si user selain dari email. Kemudian terdapat check untuk memastikan bahwa yang diinput oleh user merupakan sebuah nomor telepon dari Indonesia.
h. Location (varchar, nullable)
   Berbentuk varchar karena lokasi seseorang tidak bisa ditebak panjangnya (tidak pasti). Nullable karena untuk menghargai privasi si user. Jadi, tidak diharuskan untuk mengisi lokasi.
i. Date of Birth (date, not null)
   Berbentuk date karena berisi sebuah tanggal lahir.
j. Gender (varchar, not null, check gender like 'Male' or 'Female')
   Berbentuk varchar karena panjang karakter jenis kelamin berbeda. Not null karena dibutuhkan untuk mengetahui jenis kelaminnya. Kemudian terdapat check untuk memastikan input dari user merupakan 'Male' atau 'Female'.

2) Friends
   a. Friend ID (PK, char(5), not null)
      Friend ID menjadi PK karena dia harus unique dan dapat menjadi elemen pengenal dari table Friends. Char(5) karena bentuk ID nya 'FRXXX'. Not null karena sebuah Primary Key tidak boleh NULL.

   b. User ID (FK dari Users)
      User ID menjadi FK karena atribut ini diambil dari table Users.

3) Pages
    a. Page ID (PK, char(5), not null, check Page ID like 'PG%')
       Page ID menjadi PK karena dia harus unique dan dapat menjadi elemen pengenal dari table Pages. Char(5) karena bentuk ID nya 'PGXXX'. Not null karena sebuah Primary Key tidak boleh NULL.
    b. Page Name (varchar, not null, unique)
       Berbentuk varchar karena nama dari sebuah halaman tidak bisa ditebak panjangnya (tidak pasti). Not null karena sebuah halaman harus mempunyai nama sebagai tanda pengenal. Unique karena sebuah page tidak boleh sama namanya.
    c. Page Content (varchar, not null, unique)
       Berbentuk varchar karena konten di dalam sebuah halaman tidak bisa ditebak panjangnya (tidak pasti). Not null karena sebuah halaman tidak mungkin kosong (NULL). Unique karena sebuah konten di dalam sebuah halaman pasti berbeda.

4) Page Likes
    a. Page ID (PK, FK dari Pages)
       Page ID menjadi FK karena atribut ini diambil dari table Pages dan atribut ini dibutuhkan karena setiap page, tak terkecuali Page Likes pasti memiliki Page ID.
    b. User ID (FK dari Users)
       User ID menjadi FK karena atribut ini diambil dari table Users dan atribut ini dibutuhkan karena setiap page likes pasti dimiliki oleh seorang user.

5) Posts
    a. Post ID (PK, char(5), not null)

Post ID menjadi PK karena dia harus unique dan dapat menjadi elemen pengenal dari table Posts. Char(5) karena bentuk ID nya 'PSXXX'. Not null karena sebuah Primary Key tidak boleh NULL.

b. Post Content (varchar, not null, unique)

Berbentuk varchar karena nama isi konten dari sebuah post tidak bisa ditebak panjangnya (tidak pasti). Not null karena sebuah konten tidak mungkin kosong. Unique karena konten dari sebuah post tidak boleh sama, kalo sama plagiarism.

c. Post Date (date, not null)

Berbentuk date untuk menunjukkan tanggal post tersebut diunggah. Not null karena sebuah post pasti memiliki tanggal unggahannya kapan.

d. User ID (FK dari Users)

FK karena User ID diambil dari table Users.

6) Post Likes

a. Post Likes ID (PK, char(5), not null)

Post Likes ID ada sebagai pengenal dan PK untuk table Post Likes karena Post ID dan User ID bisa ter-duplicate jika satu user melakukan like sebanyak lebih dari satu kali pada post yang berbeda ataupun sebaliknya. Char (5) karena berbentuk 'PLXXX'.

b. Post ID (FK dari Posts)

FK karena Post ID diambil dari table Users.

c. User ID (FK dari Users)

FK karena User ID diambil dari table Users.

7) Photos

a. Photo ID (PK, char(5), not null, check Photo ID like 'PH%')

Photo ID menjadi PK karena dia harus unique dan dapat menjadi elemen pengenal dari table Photos. Char(5) karena bentuk ID nya 'PHXXX'. Not null karena sebuah Primary Key tidak boleh NULL.

b. Image Content (BLOB, nullable)

Image Content berbentuk BLOB karena menyimpan data foto. Nullable karena dalam setiap posts tidak diharuskan untuk mencantumkan foto di dalamnya.

c. Post ID (FK dari Posts)

FK karena diambil dari table Posts. Karena photos ini ada di dalam sebuah posts, maka Post ID dibutuhkan sebagai sekaligus penghubung antara table Photos dan Posts.

8) Shares

a. Share ID (PK, char(5), not null)

Share ID ada sebagai pengenal dan PK untuk table shares karena Post ID dan User ID bisa ter-duplicate jika satu user melakukan share sebanyak lebih dari satu kali, baik pada post yang sama atau berbeda. Char (5) karena berbentuk 'SHXXX'.

b. Post ID (FK dari Posts)

FK karena diambil dari table Posts. Karena shares ini berhubungan dengan sebuah posts, maka Post ID dibutuhkan sekaligus sebagai penghubung antara table Shares dan Posts.

c. User ID (FK dari User)

FK karena diambil dari table User. Karena Shares ini pasti dilakukan oleh seorang user, maka User ID dibutuhkan sekaligus sebagai penghubung antara table Photos dan Posts.

9) Comments

a. Comment ID (PK, char(5), not null)

Photo ID menjadi PK karena dia harus unique dan dapat menjadi elemen pengenal dari table Photos. Char(5) karena bentuk ID nya 'CMXXX'. Not null karena sebuah Primary Key tidak boleh NULL.

b. Comment Date (date, not null)

   Berbentuk date karena menampung tanggal. Not null karena tanggal dari sebuah komen dibutuhkan untuk keperluan data/history.

c. Comment Content (varchar, not null)

   Berbentuk varchar karena nama isi konten dari sebuah comment tidak bisa ditebak panjangnya (tidak pasti). Not null karena dalam sebuah komen tidak boleh kosong isinya.

d. User ID (FK dari Users)

   FK karena diambil dari table User. Karena Comment ini pasti dilakukan oleh seorang user, maka User ID dibutuhkan sekaligus sebagai penghubung antara table Photos dan Posts.

10)Comment Likes

a. Comment Like ID (PK, char(5), not null)

   Comment Like ID ada sebagai pengenal dan PK untuk table Comment Likes karena Comment ID dan User ID bisa ter-duplicate jika satu user melakukan like ke berbagai comment dan sebaliknya. Char (5) karena berbentuk 'CLXXX'.

b. Comment ID (FK dari Comments)

   Comment ID dibutuhkan karena sebuah like membutuhkan penanda yang menunjuk pada sebuah comment.

c. User ID (FK dari Users)

   User ID dibutuhkan karena sebuah like pasti muncul dari sebuah User.

B. DDL

1. Integritas data adalah pemeliharaan, dan jaminan, keakuratan dan konsistensi data selama siklus hidup data tersebut dan merupakan aspek penting untuk desain, implementasi, dan penggunaan sistem apa pun yang menyimpan, memproses, atau mengambil data.
   Sumber: https://en.wikipedia.org/wiki/Data_integrity

2. Primary key → Harus Unique dan Not Null. Contoh: NIK KTP.

   Foreign key → Merupakan Primary Key yang "dipinjam" ke table lain dan berguna untuk menghubungkan table tersebut dengan table yang "dipinjam" Primary Key nya. Contoh: Sebuah Post pasti dibuat oleh seorang user, maka table Post akan mengambil User ID dari table User.

   Composite key → Merupakan gabungan dari dua attribut yang berbeda. Composite key berguna agar membentuk Primary Key yang benar-benar unique.

3. BEGIN TRAN → Merupakan tanda mulainya sebuah transaksi, dimana kita bisa melakukan proses delete, update, dll. Dengan syntax ini, kita dapat menggunakan fungsi COMMIT dan ROLLBACK. Syntax iniberguna seperti sistem checkpoint. Contohnya ada sebuah table dengan atribut name. Atribut ini ingin diubah dari 'Budi' ke 'Bada'. Maka sebelum melakukan perubahan kita akan menggunakan syntax BEGIN TRAN sebagai checkpoint untuk berjaga-jaga jika ada kesalahan dalam mengubah data.

   Commit → Merupakan syntax yang berguna untuk menyimpan perubahan yang telah dilakukan setelah BEGIN TRAN. Sebagai contoh, setelah melakukan perubahan setelah BEGIN TRAN dan kita sudah yakin apa yang diubah sudah benar, maka kita tinggal menggunakan syntax COMMIT untuk menyimpan perubahan teresebut.

ROLLBACK → Merupakan syntax yang berguan untuk meng-*undo* perubahan yang telah dilakukan setelah BEGIN TRAN. Sebagai contoh, setelah melakukan perubahan setelah BEGIN TRAN dan ternyata setelah dicek terdapat kesalahan, maka kita tinggal menggunakan syntax ROLLBACK untuk meng-*undo* perubahan tersebut.