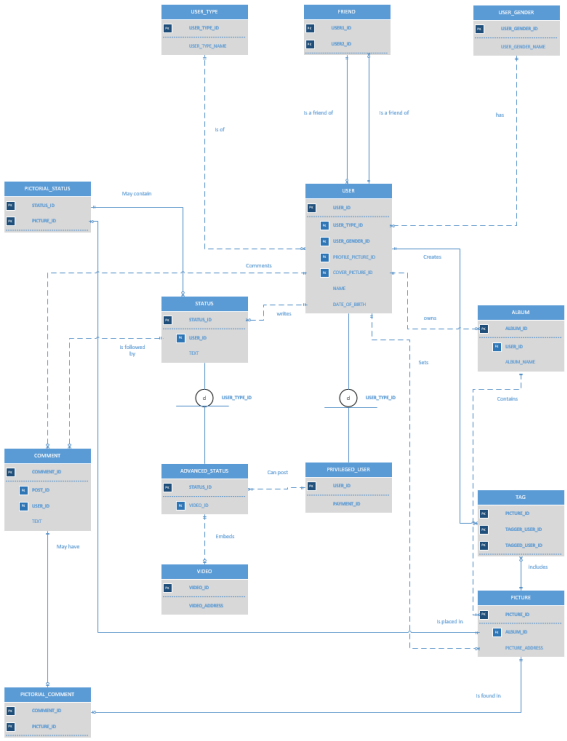
# CSCI 585 Homework 1 – Sbook




**(Please Copy paste this image in Paint and zoom it to read clearly)**

## Design Choices:

The designed system has following entities:

1. **USER**:
    a. Contains all the details for a user's profile on the social network.
    b. Attributes such as user's type, gender, profile picture and cover picture id are all foreign keys from their respective tables. Besides these, attributes for user name and user's date of birth also exist.
    c. **Assumptions**:
        i. User cannot exist without a type or gender. Hence type id and gender id are made REQUIRED attributes.
        ii. User can exist without a profile or a cover picture. Hence those attributes are not REQUIRED attributes.
    d. **Sub type:** PRIVILEGED_USER. Discriminator attribute is USER_TYPE_ID.
2. USER_TYPE, USER_GENDER:
    a. These are created as separate entities because I want to limit their values to be a small fixed set of specific strings. It is like a "enumeration" in Java.
        i. For example, if user type is kept as a string attribute in User table, it would be possible to enter any string in there. But I want to limit user's type to only 2 values – "Regular" and "Privileged". Hence, I store these 2 strings in a separate table and refer to their ids in the user table
        ii. Let's say USER_TYPE_ID = 0 then USER_TYPE_NAME = "Regular" and USER_TYPE_ID = 1 then USER_TYPE_NAME = "Privileged".
        iii. Similarly, GENDER table also contains constant strings for every possible gender type and its id is used in USER table to identify as user's gender.
3. PRIVILEGED_USER:
    a. This is a sub type of user. Discriminator attribute is USER_TYPE_ID.

    b. It has all the characteristics of a regular user, plus an additional attribute called PAYMENT_ID (REQUIRED). This represents payment conformation id, which is generated when a regular user makes onetime payment to change his/her user type to “Privileged”.
4. ALBUM:
    a. Contains information about a picture album.
    b. Since every album must belong to one and only one user, user id is a foreign key (REQUIRED) here.
5. PICTURE:
    a. Contains information about a picture.
    b. Since every picture must belong to one and only one album, album id is a foreign key (REQUIRED) here.
    c. Since pictures are stored remotely, picture table contains an attribute to store the picture’s address on the remote server
6. STATUS:
    a. Contains information about the status posted by a user
    b. Since every status must belong to one and only one user, user’s id is a foreign key (REQUIRED) here.
    c. Status also contains an attribute to store actual status text.
    d. **Sub type:** ADVANCED_STATUS.
7. ADVANCED_STATUS:
    a. This is a sub type of Status. This status can be posted by only “Privileged user”.
    b. It has all the characteristics of a normal status, plus an additional attribute called VIDEO_ID. This is because advanced status can contain 0 or 1 video.
8. VIDEO:
    a. Contains information about a video
    b. Since every video must belong to one and only one post, post’s id is a foreign key (REQUIRED) here.
    c. Just like a picture, video is also stored remotely. So, I have included an attribute to store video’s address on the remote server.
9. COMMENT:
    a. Contains information about a comment made by a user on a post.
    b. Since every comment must belong to one and only one post and user, both their keys are foreign keys (REQUIRED) here.
    c. It also contains an attribute to store actual comment text.
10. TAG:
    a. Contains information about when a user tags some other
    b. It is a
    c. TAGGER_USER_ID: This is id of user who tagged some other user
    d. TAGGED_USER_ID: This is id of user who got tagged
    e. PICTURE_ID: Picture in which the tagging was done
    f. **Composite Primary Key**:
        i. Tag must belong to a picture and two users.
        ii. None of these combinations (1 pic + 2 users) should repeat.
        iii. Combination of these foreign keys is enough to identify a tag uniquely.
        iv. Hence, all their ids are taken together to form a composite primary key.
    g. **Weak relationship:**
        i. Since Tag makes its own primary key from primary keys of other tables (USER, PICTURE), following relationships are weak (Identifying):
            1. TAG – USER
            2. TAG – PICTURE
11. PICTORIAL_POST and PICTORIAL_COMMENT:
    a. Connector tables between PICTURE – POST and PICTURE – COMMENT respectively.
    b. Whenever there is a picture in a status post or a comment, an entry is made in these tables with respective picture, comment, post id

c. These are created because these relationships are optional. For ex: A status post may or may not contain a picture. Similarly, comment may or may not contain a picture. If we do not have these connector entities, there may be a lot of NULL values in POST, PICTURE or COMMENT tables.
d. **Composite Primary Key:**
    i. Both these tables make up their primary keys from primary keys of the tables they are connecting.
    ii. Combination of primary keys of constituent tables is enough to identify a record uniquely
e. **Weak relationship:**
    i. Since both these tables derive their primary keys from the tables they connect, following relationships are weak (Identifying):
        1. POST – PICTURE
        2. COMMENT – PICTURE

12. FRIEND:
    a. Contains information about friendship between two users.
    b. Whenever any two users become friends, an entry is made in this table, with their ids.
    c. **Composite Primary Key:**
        i. User ids of both users must be present.
        ii. Combination of the two user ids must not repeat.
        iii. Combination of the two user ids is sufficient to uniquely identify a friendship.
    d. **Weak relationship:**
        i. Since primary keys of both users make up the primary key for a record in this table, relationship between this table and user table is weak (Identifying).