|  |  |
| --- | --- |
| **INFO 210 — Database Management Systems Assignment – Milestone 2** |  |

**1. ER diagram modification and description**

1. We add a partial key (aid) to announcements so that it can be looked up by group (since announcements is a weak entity)

2. we add other\_uid to friend list table because the friend be added need to be store in the table.

# 

**2. Relational Schema**

The following relational schema is presented in bold, and the description of its constraints will be presented in [...] directly below it.

**user (uid: INTEGER, email: CHAR, password: CHAR, profile: CHAR)**

**posts (pid: INTEGER, title: CHAR, content: CHAR, datetime: DATETIME, uid: INTEGER)**

[uid is the foreign key reference to user and uid is not null. (uid is cascade)]

**comments (cid: INTEGER, content: CHAR, datetime: DATETIME, uid: INTEGER, pid: INTEGER)**

[Respectively, uid and pid are foreign key reference to user and posts and both are not null.(both uid and pid is cascade) ]

**admin (uid: INTEGER, permission\_level: INTEGER)**

[uid is the foreign key reference to user.(on deleted no action)]

**advertisement (id: INTEGER, content: CHAR, uid: INTEGER)**

[uid is the foreign key reference to admin.( on deleted no action)]

**categories (cat\_id: INTEGER, cat\_name: CHAR, cat\_desc: CHAR)**

**manage (uid: INTEGER, cat\_id: INTEGER)**

[uid and cat\_id are also foreign key reference to admins and categories respectively.(uid and cat\_id is cascade)]

**subscribe (uid: INTEGER, cat\_id: INTEGER)**

[uid and cat\_id are also foreign key reference to users and categories respectively.( uid and cat\_id is cascade)]

**classify (pid: INTEGER, cat\_id: INTEGER)**

[pid and cat\_id are also foreign key reference to posts and categories respectively.

(pid and cat\_id is cascade) ]

**groups ( gid: INTEGER, des: CHAR)**

**create ( uid:INTEGER, gid:INTEGER)**

[uid and gid are foreign key reference to admin and posts respectively. (uid and gid is cascade)]

**join ( gid: INTEGER，uid: TEGER)**

[uid and gid are also foreign key reference to users and groups respectively.(uid and gid is cascade)]

**announcement (gid: INTEGER, aid: INTEGER, title: CHAR, content: CHAR)**

[aid is the partial key of announcements and gid is the primary key of groups. The two key combines the primary key of announcements. gid is not null. (on delete cascade)]

**Back list (from:INTEGER, to:INTEGER)**

[Both from and to (reference to the user) comprise the primary key. (both is cascade)]

**Friend list(uid: INTEGER, other\_uid: INTEGER)**

[Both uid and other\_uid (reference to user) comprise the primary key. (both is cascade)]

**3.** **Explanation**

**I. 1 to N relations**

Posts: Posts and users have a N-to-1 relationship (because one post could only be owned by one user). So we should add uid of user as foreign key to the posts schema and uid should not be null. And the primary key pid is auto incrementally. Plus, If the user is deleted, the corresponding post would be deleted.

Comments: Comments has a N-to-1 relationship with posts and user ( because one comment could only be owned by one user and be attached to only one post, while a user could add multiple comments and a post could have multiple comments.) And comments are total participations of both two relations. So we should add uid reference to user and pid reference to posts as foreign keys to the schema and they should not be null. Plus, the primary key cid is auto incrementally. If a post or a user is deleted，the corresponding comment got deleted.

Announcements: It is a weak entity and it is n to 1 with the group entity (because each group can generate more than one announcement). The partial key of announcements (aid) depends on the groups, because the aid is defined by groups. So, the primary key for announcements is composed of aid and the gid of groups (gid should not be null, and when a group is deleted, the group's announcements are also deleted).

**II. N to N relations**

Subscribe: Subscribe is n to n relationship (because one category could be subscribed by multiple user and one user could subscribe multiple categories). So both uid of user and cat\_id of categories comprise the primary key. In addition, if the user or category is deleted, the corresponding data would be deleted.

Manage: Same as subscribe, uid and cat\_id comprise its primary. However, Its uid is admin's uid. Plus, if the admin or category is deleted, the corresponding data would be deleted.

Join: Join is n to n relationship (because many user could join in one group and a user could join in multiple group). So Both gid of groups and uid of user consist the primary key.

Classify: It is n to n relationship, because multiple posts could be classified under one or more category while one category could have many posts. What’s more, posts are total participations in this relation, every posts must be classified. So pid of posts and cat\_id of categories are primary key and cat\_id should not be null. Plus, If the post or category is deleted, the corresponding relation would be deleted

Category&group: Both of the primary key is generated auto incrementally which serve as an unique identifier.

Friend\_list: It is n to n relationship (one user could add multiple user into friend list and one user could be added by multiple user into friend list). So both uid and other\_uid are primary key. Besides, If a user in one of the column is deleted, the data would be deleted.

Black\_list: It is n to n relationship (one user could add multiple user into black list and one user could be added by multiple user into black list). So from and to which store the uid of user are primary key. Besides, If the user in one of the column is deleted, the data would be deleted

**III. ISA**

Admin: It requires a separate schema. Although admin inherits from user, admin obviously cannot coverage user in its entirety. So it inherits user's primary key uid and has its own attribute, permission\_level.