**Mount Royal University** **Winter 2013**
**COMP 2521** **Instructor: Al Fedoruk**
**Databases I** **Apr 1, 2013**

# Assignment 5

**Database Application**

**Due: Apr 12, 2013**

## Purrr

I have come up with a new idea for a social media website called Purrr. When you first log on to the website the song *Soft Kitty* plays for you — but that is not important.

Users sign up for the site and can post short updates about their lives called Meows. Each Meow must be no more than nine words long, because cats have nine lives, get it? Each word can be as long as it needs to be.

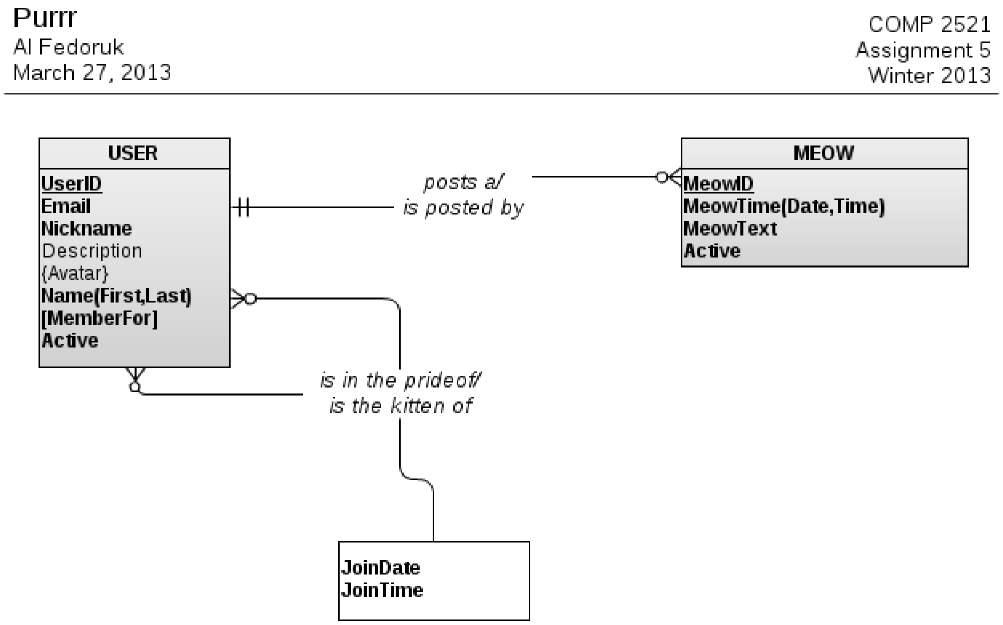
A user may join another user's Pride, enabling them to see all of that user's Meows. When you are in another's Pride they can and will refer to you as one of their Kittens.

Some of the functions that the database will need to provide to the user interface are:

- Retrieve the profile of a user.
- Retrieve all the Meows for a given user.
- Retrieve the Pride(s) that a user belongs to.
- Retrieve a list of users who are in a given user's Pride.
- Add a new user.
- Add a user to a Pride.
- Add a Meow for a user.
- Deactivate a user.

For this assignment you will be implementing some of these functions.

### Conceptual Model



### *Notes and Assumptions*

**User** entity instances represent the registered users of the system. Users can be edited, but never deleted. A user can go 'inactive'.

- A unique `UserID` is generated for each registered user.
- Each user must provide a valid email address. The email address is only for Purrr administrators to use and will not be displayed.
- All users must provide a contact name. The name is only for Purrr administrators to use and will not be displayed.
- A user can optionally provide a text description of themselves. For example a user may enter: 'Calgary cat lover Xtra-ordinairre!'.
- A user may upload one or more avatar image files. The path to the file(s) is stored. The most current image is displayed beside the users `Meows`.
- The date that the user joined is stored and the `MemberFor` field is used in the profile to display how long the member has been meowing.
- The `Active` field is a boolean, true if this member is active, false if not.

**Meow** Instances of the `Meow` entity type represent the updates posted by the users.

Meows cannot be edited or deleted once they are posted.

- Each `Meow` is assigned an integer sequential ID number.
- The date and time that the `Meow` was posted is recorded.
- The `MeowText` can be as long as required, but can only contain nine words. Words are delimited by whitespace or any of the punctuation characters: :;,.
- When a user is deactivated, all of his/her Meows are marked as with FALSE in the `Active` field.

**Pride/Kittens** The `Pride/Kitten` relationship indicates which users prides each user belongs to. A user can belong to many prides and a pride can have many users (kittens).

A user can add or delete pride/kitten relationships at will. When a user is deactivated, all of his/her pride/kitten relationships are deleted.

## Logical Model

Transforming the conceptual model above, the following relations are produced:

USER( UserID, Email, NickName, NameFirst, NameLast, Description, DateJoined, Active)
PK: UserID

AVATAR (UserID, Avatar, AddedDate, AddedTime)
PK: UserID, Avatar)
FK:UserID ← USER.UserID

MEOW( MeowID, UserID, MeowTime, MeowDate, MeowText, Active)
PK: MeowID FK:UserID ← USER.UserID

PRIDE( UserID, KittenID, JoinDate, JoinTime)
PK: UserID, KittenID
FK:UserID ← USER.UserID
FK:KittenID ← USER.UserID

***Domains***

| Table | Field | Null? | Type | Domain |
|---|---|---|---|---|
| USER | UserID | PK | INT | |
| USER | Email | N | String | Must be a valid, unique email. |
| USER | Nikname | N | String | Any unique string, maximum length 20. |
| USER | Description | Y | String | Any text, maximum length 1000. |
| USER | NameFirst | N | String | Any text, maximum length 100. |
| USER | NameLast | N | String | Any text, maximum length 100. |
| USER | DateJoined | N | Date | The date the user joined. |
| USER | Active | N | Boolean | True if user is active. |
| AVATAR | UserID | PK | | Foreign Key to USER. |
| AVATAR | Avatar | PK | String | A relative path name to an image file. |
| AVATAR | AddedDate | N | Date | Date this image was added. |
| AVATAR | AddedTime | N | Time | Time this image was added. |
| MEOW | MeowID | PK | Integer | Generated key. |
| MEOW | UserID | N | | Foreign Key to USER. |
| MEOW | MeowDate | N | Date | Foreign Key to USER. |
| MEOW | MeowTime | N | Time | Foreign Key to USER. |
| MEOW | MeowText | N | String | Text of the Meow. Must be nine words or less, words are delimited by whitespace or any of the punctuation characters: ()[] :; , .$% <> |
| MEOW | Active | N | Boolean | False if the user who posted this Meow is inactive, true otherwise. |
| PRIDE | UserID | PK | | Foreign Key to USER |
| PRIDE | KittenID | PK | | Foreign Key to USER |
| PRIDE | JoinDate | N | Date | Date user joined this Pride |
| PRIDE | JoinTime | N | Time | Time user joined this Pride |

### Implementation of Tables

The script `comp2521A5.sql` does a base-line implementation of the tables described above. Some notes:

- All instances where a DATE and TIME value are being stored have been implemented as a single DATETIME value.

## Your Tasks

1. Create an insert trigger to properly date/time stamp a `Meow` and to ensure that the meow text has between one and nine words in it. A function to count words would be a good idea. Meows cannot be updated or deleted once they are inserted so no other triggers are needed.

2. Create an insert and an update trigger to ensure that the email entered in the user table is in a valid format.

3. Create an insert trigger to date/time stamp the PRIDE table. Entries in the PRIDE table can be deleted, but not updated.

4. Create views

   (a) USERS which lists the public profile for all active users. This should include the NikName, the Description if present, the path to the users Avatar, the number of kittens the user has, the number of prides the user belongs to and the number of Meows the user has posted.

   (b) PRIDES to list all of the prides that a user belongs to. This view should have two fields: ME, KITTEN which are the `NikName`s of the users in a relationship. The view can then be used to quickly list all of the prides that a user belongs to and all of the kittens that a user has.

   (c) TIMELINE should contain three fields: NIKNAME, DATETIME, TEXT and be ordered by date. Create a query that retrieves the timeline for a particular user. Meows from inactive users should not be displayed.

5. Create a procedure to deactivate a user. It should take one parameter, the users NikName. If the NikName is not valid, an error should be raised.

6. Testing. Create test data that demonstrates that all of your code is working.

## Deliverables

Your deliverable is a singe `sql` script names A5. This script should be extraordinarily well documented.