ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Institute of Information Systems
Alexandre de Spindler
Christoph Zimmerli

globis

Object-Oriented Databases

Spring 2012

Exercise 1

In this exercise we will develop an application for managing video clips, similar to YouTube. The application offers the facilities to add users, videos and comments such that they can be retrieved and viewed later on. We will use Hibernate as the persistence technology.

Exercise 1.1 *Hibernate*

Go to the Hibernate web site [1] and get yourself familiar with using Hibernate with Java. A good way of getting started is working through the first chapter of the Hibernate Reference Documentation which takes you step by step into using Hibernate to make Java objects persistent. As suggested in that tutorial, you may use HSQL as a relational database.

Exercise 1.2 *Design*

The scenario for our version of YouTube is modeled in Figure 1. Users can register with a nickname and e-mail address. These users can then *a*) author (upload and edit) their own videos and *b*) make comments on any video on the site (own or from other user). A user's activities (authoring and commenting) will be shown in his activity stream, ordered by date. Each video can either be standard definition or high definition (but not both). All data in the scenario should be persistent in the database, except for the session a user has while he is logged in.

For illustration of the requirements that must be met by your implementation, we provide some example code in Listing 1 which shows a typical interaction scenario between a client application and your database. This code covers basic functionality only – you may enrich your implementation with additional features.

In order to facilitate the client side interaction, we suggest the use of a `StorageManager` class which encapsulates all persistence mechanisms including creating, opening and closing a database instance. The storage manager must also support some forms of making domain objects persistent as well as to retrieve particular or all objects of a given type.

Design this object-oriented database that allows another Java application to manage users, videos and comments as indicated in Listing 1. Decide about the Java classes you want to use and specify their attributes and methods. You can do this for instance using a UML class diagram.

Exercise 1.3 *Implementation*

Implement the YouTube database as you have designed it in the previous task. A good starting point for the implementation is the tutorial in Chapter 1 of the Hibernate documentation mentioned before. Additionally, you can download an archive from the course website. This archive includes:

- An example for class `StorageManager`.
- A class `MainApplication` containing the usage scenario in Listing 1.
- An Ant script for starting and stopping HSQLDB and running the application.
- Configuration files for Hibernate.

[1] `http://hibernate.org/`

- All necessary libraries (Hibernate, HSQLDB and their respective dependencies).

Using this archive, you have basically two open points to address:

- Implement classes to match the application model and the usage scenario.
- Specify the mapping for Hibernate.

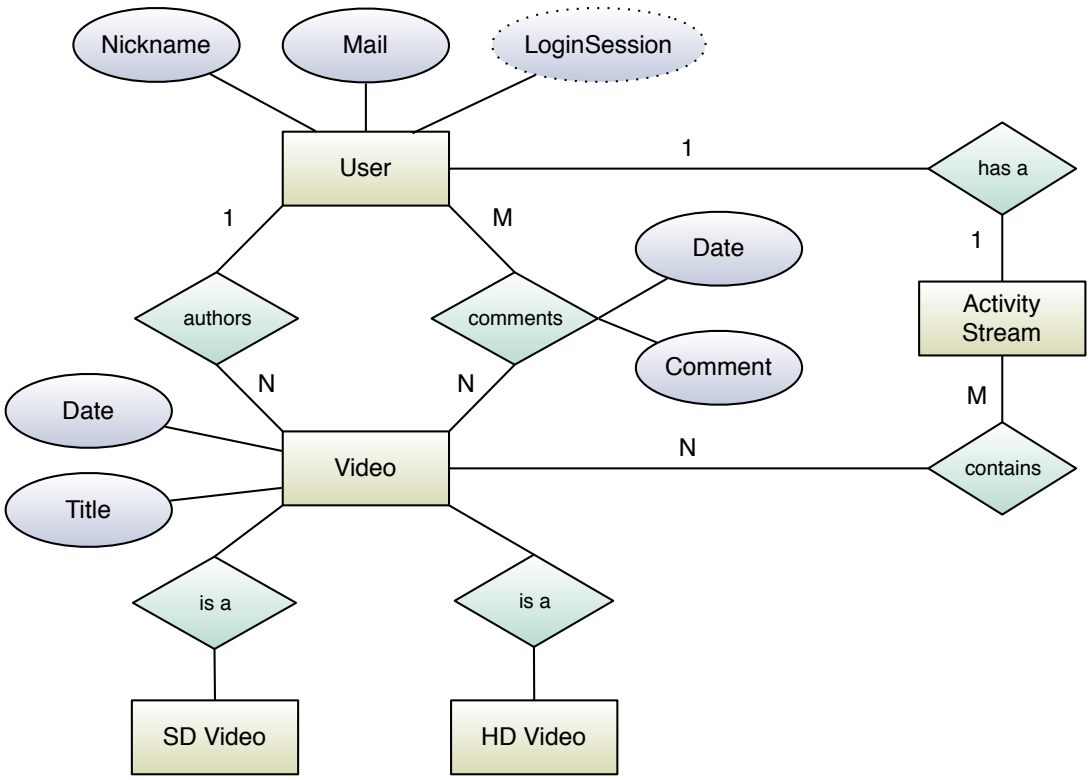

Figure 1: Simple model of the YouTube application domain.

Exercise 1.4 *Queries*

For verification of your persistence solution and getting familiar with querying facilities offered by hibernate, you should implement and run the following queries and assert their results' correctness:

- Show the titles of all videos in the database.
- Show the titles of all videos authored by a given user.
- Show the titles of all videos authored by user one that user two commented on.
- Show the titles of all videos that received comments during the last week.
- Show the title of the video that received the most comments.
- Show all users that authored HD videos.

```
storageManager.beginTransaction();

User userOne = new User("Dogg", "dogg@animals.net");
storageManager.save(userOne);
Video videoOne = new SDVideo("My Dogs");
storageManager.save(videoOne);
userOne.author(videoOne);
Comment commentOne = new Comment("I love my dogs, they are the best!",
   videoOne, userOne);
storageManager.save(commentOne);

User userTwo = new User("Kitty", "kitty@animals.net");
storageManager.save(userTwo);
Comment commentTwo = new Comment("Dogs are soooo lame! " +
   "My cats are way cooler!", videoOne, userTwo);
storageManager.save(commentTwo);

Comment commentThree = new Comment("I need to see that, before I " +
   "believe your bold claims!", videoOne, userOne);
storageManager.save(commentThree);

Video videoTwo = new HDVideo("Awesome Cat Action");
storageManager.save(videoTwo);
userTwo.author(videoTwo);
Comment commentFour = new Comment("There you go: " +
   "<link-to:Awesome_Cat_Action>", videoOne, userTwo);
storageManager.save(commentFour);

Comment commentFive = new Comment("Woah, those are really cool cats! " +
   "And even in HD!", videoTwo, userOne);
storageManager.save(commentFive);

Video videoThree = new SDVideo("No Comment");
storageManager.save(videoThree);
userTwo.author(videoThree);

storageManager.commitTransaction();
```

Listing 1: Example application code.