Vicarious Learning in Informatics
YouTute: Database Design

Neil Mayo, nmayo@inf.ed.ac.uk
Informatics, University of Edinburgh

March 29, 2008

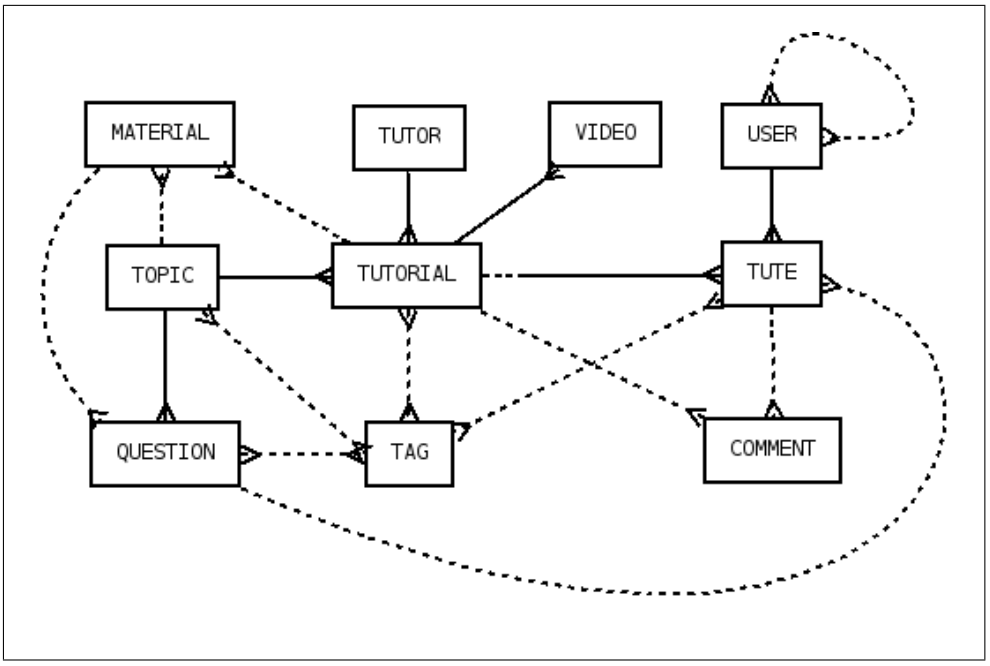
1 Introduction

This document is a record of database design, building from a model of entity relations to a normalised set of tables ready for implementation.

2 Data Model (Entities)

The data model shows entities in the data, and their relations.

Figure 1: Data model showing entity relations.



2.1 Notes

Tutorials A *tutorial* is an instance, a physical tutorial occurring on a particular date, tackling a particular *topic* and steered by a tutor. A topic may be "Asymptotic Notation", which is dealt with in

several different tutorials. Within a tutorial, we assume that the discussion can be broken down into *questions*, as experience has proven that the structure of the tutorials is quite regular.

Tutes A *tute* is a defined subsection of a tutorial. We expect tutes to reflect roughly the period of a question.

Questions A *question* is a specific question discussed within the tutorial, and is therefore related to a specific topic along with other questions. It can be tagged, related to specific pages of a material (which may pose and elaborate upon the question), and may be associated with a subsection of a tutorial discussing the topic, by associating it with a tute.

Materials Materials can be associated with tutorial topics (for example, lecture materials) or individual tutorials (for example, materials produced during the tutorial). They are predefined but we may add the capability for students to upload their own materials, which may then be tute-specific. Possible subtypes: subject materials, tutorial materials, global materials (tutor, timetable information ?).

Tags Note that a tag can be applied to a tute, or a tutorial as a whole. Possible subtypes: default, user-defined. Also tutes should inherit tutorial tags, which is possible in this design. Comments may also be applied to both tutes and tutorials. When a tag or comment is created, the id of the user who created them is also recorded.

Note: Subtypes for materials and tags may be represented internally as an attribute.

Friends The optional reflexive relation on the user entity is to define a 'friend' relation whereby users can link to other users and e.g. see their tutes. The main question here is whether a friend relationship is necessarily mutual (requiring confirmation), or one way (which will take up more rows in the table). For the moment we will allow one-way friendships, as it is simpler, and it is easy for a user to find out who calls him/her a friend.

2.2 Assumptions/Questions

We assume there is one tutor per tutorial; we could later add a reference to a second tutor.

A tutorial is associated with several (3) videos representing 'views' of the tutorial. Comments and tags however are made on the tutorial, as we assume that there is no value in allowing the videos to be treated individually.

3 Relational Structure (Intensions)

These describe the structure of the relations, showing attributes and attribute domains for each entity. Possible (undecided) attributes are greyed and italicised. Keys are also identified, to enable relations between entities in database tables. Note that many-to-many relations involve creating a third relation mapping the primary keys of each entity.

Primary keys are shown in red and underlined.

Foreign keys* are shown in blue and with an asterisk.

Relations have been normalised to 3NF.

Materials – pdf files (slides, handout, lecture notes), eventually other types (text, doc, audio?). The TutorialId is optional as a material may relate either to an individual tutorial, or generally to a tutorial topic, and thus transitively to several tutorials.
MATERIAL(Id/IDS, Name/NAMES, Type/MATERIALS, Filename/FILE-NAMES Filetype/FILE-TYPES, TopicId/TOPIC-IDS*, TutorialId/TUTORIAL-NAMES*)

Tutors giving the tutorials.
TUTOR(Name/NAMES, UserId/UNI-IDS)

Users of the interface, i.e. students (and staff).
USER(Name/NAMES, UserId/UNI-IDS, Password/PASSWORDS)

Friends linking users to other users.
FRIEND(UserId/UNI-IDS*, FriendId/UNI-IDS*)

Tutorial Topics – what is being discussed.
TOPIC(Id/TOPIC-IDS, Description/DESCRIPTIONS)

Questions within a tutorial topic. The question can optionally be linked to a material, and a particular page within a pdf material.
QUESTION(Id/QUESTION-IDS, TopicId/TOPIC-IDS*, MaterialId/MATERIAL-IDS, MaterialPage/INTEGERS)

We also provide a structure to relate questions to tutes for particular tutorials. We could relate them by adding an optional reference in the tute to a question foreign key, but in practise this relation will be much less used than tutes, so this additional table is deemed more efficient.
QUESTION_TUTE(QuestionId/QUESTION-IDS*, TuteId/TUTE-IDS*)

Tutorials – the events which are videoed. They occur within a particular week of term, deal with a topic, and are tutored by a tutor.
TUTORIAL(Id/TUTORIAL-IDS, Week/INTEGERS, TopicId/TOPIC-IDS*, TutorId/UNI-IDS*)

Videos – the videos forming the component views of the tutorials. Each video shows a particular view of a particular tutorial.
VIDEO(View/VIDEO-VIEWS, TutorialId/TUTORIAL-IDS*, Filename/FILE-NAMES)

Tutes created by users. A tute has a user-defined name and description, a start and end time within the tutorial, and is linked to the user who created it.
TUTE(Id/TUTE-IDS, Name/TUTE-NAMES, Description/DESCRIPTIONS, UserId/USER-IDS*, Start/TIMES, End/TIMES, TutorialId/TUTORIAL-IDS*)

Tags attached to Topics, Tutorials, Tutes or Questions, and used in searching for Tutorial or Tute videos. The original creator of the tag is recorded, though tags are available for everyone to use, to encourage collaboration and sharing of resources. (Instead of creator we should perhaps have a switch indicating whether the tag is default or user-defined.)
TAG(Name/TAG-NAMES, Class/TAG-CLASSES, Creator/UNI-IDS*)

Taggings – extra entities for modelling many-to-many relations with tags. The id of the user who tagged the resource is also recorded.
TOPIC_TAGGING(TopicId/TOPIC-IDS*, TagId/TAG-IDS*, CreatorId/UNI-IDS*)
TUTORIAL_TAGGING(TutorialId/TUTORIAL-IDS*, TagId/TAG-IDS*, CreatorId/UNI-IDS*)
TUTE_TAGGING(TuteId/TUTE-IDS*, TagId/TAG-IDS*, CreatorId/UNI-IDS*)
QUESTION_TAGGING(QuestionId/QUESTION-IDS*, TagId/TAG-IDS*, CreatorId/UNI-IDS*)

Comments attached to specific points in a Tute/Tutorial video and shown during playback. The comment has a start time within the tutorial, and is shown for a specified duration (a default should be supplied for this attribute). The creator of the comment is recorded along with the comment-level and the id of either a tute or tutorial.
COMMENT(Id/COMMENT-IDS, Content/DESCRIPTIONS, Start/TIMES, Duration/DURATIONS, UserId/UNI-IDS, Level/COMMENT-LEVELS, TuteId/TUTE-IDS*, TutorialId/TUTORIAL-IDS*)

Note: Most string-valued attributes will be realised with the MySQL `TEXT` type; however some attributes have distinct limits. For example, tags should be kept short so we impose a limit of 30 characters; university ids have a limit of ?? characters, and we limit the length of passwords. Input length must be checked programmatically before the value is submitted to the database, or it will be truncated..

3.1 Notes

The Material entity can be related to either a subject or a specific tutorial, therefore its relations have optional ends. Note that this modelling does not explicitly enforce that a material must be related to exactly one subject or tutorial. So theoretically we could have orphaned materials, which may or may not be desirable. A material can also be associated with both a tutorial and a subject. Representing this relationship as it stands requires referencing the key of the subject or tutorial in the material – so either of these foreign keys may be null in the database.

In recording supporting materials, should we store URLs, or IDs which allow identification of resources on a relative path with specific naming conventions? Note that these materials may eventually be added by staff via the interface.

Note that it is Tutorials and Tutes that are tagged and commented. They each represent sections of video relating to the actual videos if the tutorial. Comments should be shown only in the Tute or Tutorial video to which they are related, but we could add an option to show Tutorial-level Comments in Tute videos.

Note that we have omitted a description of a course (e.g. Inf2B), in which each of the subjects would occur. This may be added later and associated with subjects via a foreign key reference in each subject.

Note that we do not provide an explicit relation between User and Tutor, even though in some cases there will be one. We can correlate these with a query based on the UserId.

Do questions relate to more than one material, e.g. question and solution sheets?

3.2 Enumerations (of attribute domains)

MATERIALS

- slides

- handout
- lecture

FILE-TYPES

- pdf
- text
- *doc*

VIDEO-VIEWS

- class – the 'class' view
- tutor – the 'tutor' view
- smart – the smartboard

TAG-CLASSES

- default (predefined tags)
- user (user-added tags)
- *we can add more later*

COMMENT-LEVELS

- tutorial
- tute
- *subtitle* ?

Other domains:

- UNI-IDS come from the University's id scheme (staff and student), and should therefore be unique.
- TIMES is an integer representing a number of seconds after the start time of the tutorial video.
- DURATIONS is an integer representing a number of seconds.
- IDS, PASSWORDS, NAMES are limited-length text attributes
- DESCRIPTIONS, FILE-NAMES are arbitrary text attributes.

4 Sample queries

We are likely to perform the following queries among others:

- Get materials for a tutorial/tute/subject
- Get tutorials/tutes/subjects with particular tags
- Get the videos for a tutorial
- Get the tutes for a tutorial/subject
- Get the tutes for a user
- Get the comments for a tute
- Get the subject/tutor of a tute/tutorial

TODO add the queries here.

5 Database Implementation

The model will be implemented in a MySQL database. Tasks include:

- Creating tables.
- Entering/importing data.
- Setting up privilege tables for secure access.

See B for the commands to set up the tables to reflect the data model.

5.1 Enter initial data commands

Some example commands for entering initial data:

```
insert into user values ('nmayo', 'Neil', 'nmayo');
insert into tutor values ('srenals',  'Steve Renals');

insert into subject values ('inf2b_tut3', 'Asymptotic Notation');
insert into tutorial values (null, 20080128, 'inf2b_tut3', 1);

insert into video values ('face1', 1, 'nm-face1.flv');
insert into video values ('face2, 1, 'nm-face2.flv');
insert into video values ('smart', 1, 'nm-smart.flv');
```

5.2 Security

Basic Users should only be able to add to the Tute table

Tutors have user privileges plus ability to add/change/delete materials for their associated tutorials, to add default tags and add tutorials(?).

Admin can add/delete/modify any entries in running database; not add/remove/rename tables as this may affect the YouTute interface.

TODO describe commands to set appropriate access permissions

APPENDIX

A MySQL Setup Commands

Some commands to setup various aspects of MySQL.

A.1 Basic commands

Within MySQL, type an SQL expression to query the database, or use other commands to modify the tables or entries. For example:

`describe users` – show the `users` table design.

`select * from user_roles` – get all `user_roles` records.

```
insert into users values('nmayo', 'Neil', 'nmayo')
```

A.2 Creating users and setting access

(···)

A.3 Starting MySQL with protected DB

```
mysql -p -u root youtute
```

(···)

A.4 Providing controlled access for JSPs

(···)

A.5 Data Modification Commands

```
alter table tutorial drop column date;
drop table user;
drop table if exists `user`;
delete from tag where creator_id='nmayo';
```

B MySQL Commands to Setup the Data Model

Commands used for creating and populating the database have been saved as SQL scripts. They can be run like this:

```
mysql -p -u root youtute < database-model.sql
mysql -p -u root youtute < fill-database.sql
```