Chris D’Angelo (cd2665) & Xin Du (xd2137)
February 11, 2013

Introduction to Databases - Project 1.1

Application Description

Basic Description:
FilmDB.com is a web application that that provides its users the ability to browse information about the movies they enjoy, rate them and receive recommendations. Users can browse movies by title, by actor or crew member. Users can find information about movies including, quotes, when the movie is playing in theaters, and where the movie is available to rent if available. Users also can keep track of when they have watched a movie and give any movie a rating. Users befriend other users and may peruse what movies they have seen and rated. Users can also rate a cast or crew member. The application can provide the user with recommendations based on the user’s friends and ratings.

Web Front End Details:
We intend to build a web front end for this database. A summary of the UX design will be as follows. After a user creates a profile they will sign in. They will then see a profile page which will contain the following feature categories: Movies I’ve Seen, Movies I’ve Rated, Friends Ratings, Recommendations. The user may also follow a link to browse movies. This brings the user to a page of movie titles listed alphabetically. From here they can choose a movie and be brought to a new page which contains the following information: Movie Details, Cast and Crew in the Movie, Friends Ratings, Where’s it Playing, Rentable, and Quotes. The user can rate the movie from this page as well. From the “alphabetical list of movies” page the user has the option to click on a link for an alphabetical listing of Cast and Crew Members. The user may click on any Cast and Crew Member to bring up a page of their work history. On this page they can rate the actor, see friend’s ratings of the actor, or click on one of the movies in their work history to view the movie detail page.

Entity / Relationship Details:
Users can only provide one rating to a cast/crew member. Movies must be one or more genre types. Movies can have any number of cast/crew members. A cast/crew member must have worked on at least one movie. A movie can have any number of quotes. A quote can only be in one movie it cannot be in multiple movies. A quote cannot exist without being associated with a movie. A movie can be watched many times by a user or not. A movie can be watched by any number of people. A movie can be rated only once by each user. In other words, a user can only give one rating to a single movie but that rating may be edited later. A movie can be rated by multiple people. A person can rate any number of movies. When a movie is playing at a movie theater it can play any number of times at any number of theaters. A movie theater cannot play two movies at the same date and time. When a movie is available for rent or streaming it can be available at any number of services. A service can provide any number of movies.

We intend to populate the movie database from plain text files provided by www.imdb.com/interfaces. User data we will generate randomly using a tool such as http://www.generatedata.com/.

Contingency Plan

If there is need to simplify the above plans the following features and their consequent entities and relationships will be removed: Users will not be able to rate movie, befriend other users, or receive recommendations. Users will not be able to look up where the movie is playing currently or where it is available to rent.

Entity Relationship Diagram

<< ATTACHED AT END OF DOCUMENT >>

Relational Schema in SQL

```
CREATE TABLE movies
      (mid CHAR(20),
      title CHAR(40),
      synopsis CHAR(2000),
      runningTime CHAR(40),
      country CHAR(40),
      language CHAR(40),
      releaseDate DATE,
      PRIMARY KEY (mid))

CREATE TABLE theaters
      (tid INTEGER,
      theaterName CHAR(40),
      address CHAR(100),
      city CHAR(40),
      state CHAR(20),
      zip INTEGER,
      PRIMARY KEY (tid))

CREATE TABLE rentalservice
      (rid INTEGER,
      serviceName CHAR(40),
      sURL CHAR(100),
      address CHAR(100),
      city CHAR(40),
      state CHAR(20),
      zip INTEGER,
      description CHAR(2000),
      PRIMARY KEY (rid))
```

```
CREATE TABLE castcrew
      (cid INTEGER,
      lastname CHAR(40),
      firstname CHAR(40),
      gender Integer,
      dob DATE,
      trivia CHAR(2000),
      PRIMARY KEY (cid))

/* A quote must be apart of only one movie. A quote cannot
 * exist unless it is apart of some movie
 *
 * This table encapsulates the entity and relationship on the diagram
 */
CREATE TABLE quotes
      (qid INTEGER,
      characterName CHAR(40),
      quote CHAR(2000),
      mid CHAR(20) NOT NULL,
      PRIMARY KEY(qid)
      FOREIGN KEY (mid) REFERENCES movies)

CREATE TABLE genres
      (genreName CHAR(20),
      PRIMARY KEY (genreName))

CREATE TABLE users
      (email CHAR(40),
      firstname CHAR(40),
      lastname CHAR(40),
      Password CHAR (20),
      PRIMARY KEY (email))

/* A theater cannot have a showing at the same date and time
 *
 * If theater is deleted, remove playing at info. If movie
 * is deleted the info can remain
 */
CREATE TABLE playingat
      (date DATE,
      time TIME,
      tid INTEGER,
      mid INTEGER,
      PRIMARY KEY (tid, date, time),
      FOREIGN KEY (mid) REFERENCES movies,
      FOREIGN KEY (tid) REFERENCES theaters
            ON DELETE CASCADE)
```

```
/* If rental service or movie is deleted, remove rentalavailfrom
 * info. If movie is deleted the info can remain
 *
 * priceQty is meant to signify “night” in the example
 * expression “$3/night”
 *
 * A rental service can only rent the same movie once over the same
 * period of time.
 */
CREATE TABLE rentavailablefrom
      (availFrom DATE,
      availTo DATE,
      price REAL,
      directurl CHAR(100),
      priceQty CHAR(100),
      terms CHAR(2000),
      rid INTEGER,
      mid INTEGER,
      PRIMARY KEY (rid, mid, availFrom, availTo),
      FOREIGN KEY (mid) REFERENCES movies,
      FOREIGN KEY (rid) REFERENCES rentalservice
            ON DELETE CASCADE)

/* If a user or movie is deleted the rated information should
 * be removed.
 *
 * User can give only one rating per movie
 */
CREATE TABLE rated
      (rating REAL,
      comments CHAR(2000),
      date DATE,
      mid INTEGER,
      email CHAR(40),
      PRIMARY KEY (mid, email),
      FOREIGN KEY (mid) REFERENCES movies
            ON DELETE CASCADE,
      FOREIGN KEY (email) REFERENCES users
            ON DELETE CASCADE)

/* If a user or movie is deleted the watched information should
 * be removed.
 *
 * A user can watch a movie multiple times and a movie can
 * be watched by multiple people so there is no unique restriction
 * here
 */
CREATE TABLE watched
```

```
      (date DATE,
      mid INTEGER,
      email CHAR(40),
      FOREIGN KEY (mid) REFERENCES movies
            ON DELETE CASCADE,
      FOREIGN KEY (email) REFERENCES users
            ON DELETE CASCADE)

/* It is impossible with only the SQL Schema to enforce that
 * all newly created Cast & Crew Members can only be created
 * when they have been apart of a movie production. This will
 * be enforced at the application level.
 *
 * An actor or crew member can have many roles on a movie
 * therefore there is no unique constraint
 *
 * When a movie is deleted an actor can’t have worked on it
 * so delete that relationship information.
 * If an actor is deleted the movie should still remain.
 */
CREATE TABLE workedin
      (charName CHAR(100),
      jobCategory CHAR(100),
      billingNum INTEGER,
      jobTitle CHAR(20),
      mid INTEGER,
      cid INTEGER,
      FOREIGN KEY (mid) REFERENCES movies
            ON DELETE CASCADE,
      FOREIGN KEY (cid) REFERENCES castcrew))

/* Again it is impossible with only the SQL to Schema to enforce
 * our design. A movie must be one or more genres. This will be
 * enforced at the application level
 *
 * When a movie is deleted its genre assignments should be
 * removed.
 *
 * A movie cannot be assigned the same genre twice.
 */
CREATE TABLE typeof
      mid INTEGER,
      genreName CHAR(20),
      (PRIMARY KEY (mid, genreName),
      FOREIGN KEY (mid) REFERENCES movies
            ON DELETE CASCADE,
      FOREIGN KEY (genreName) REFERENCES genres)
```

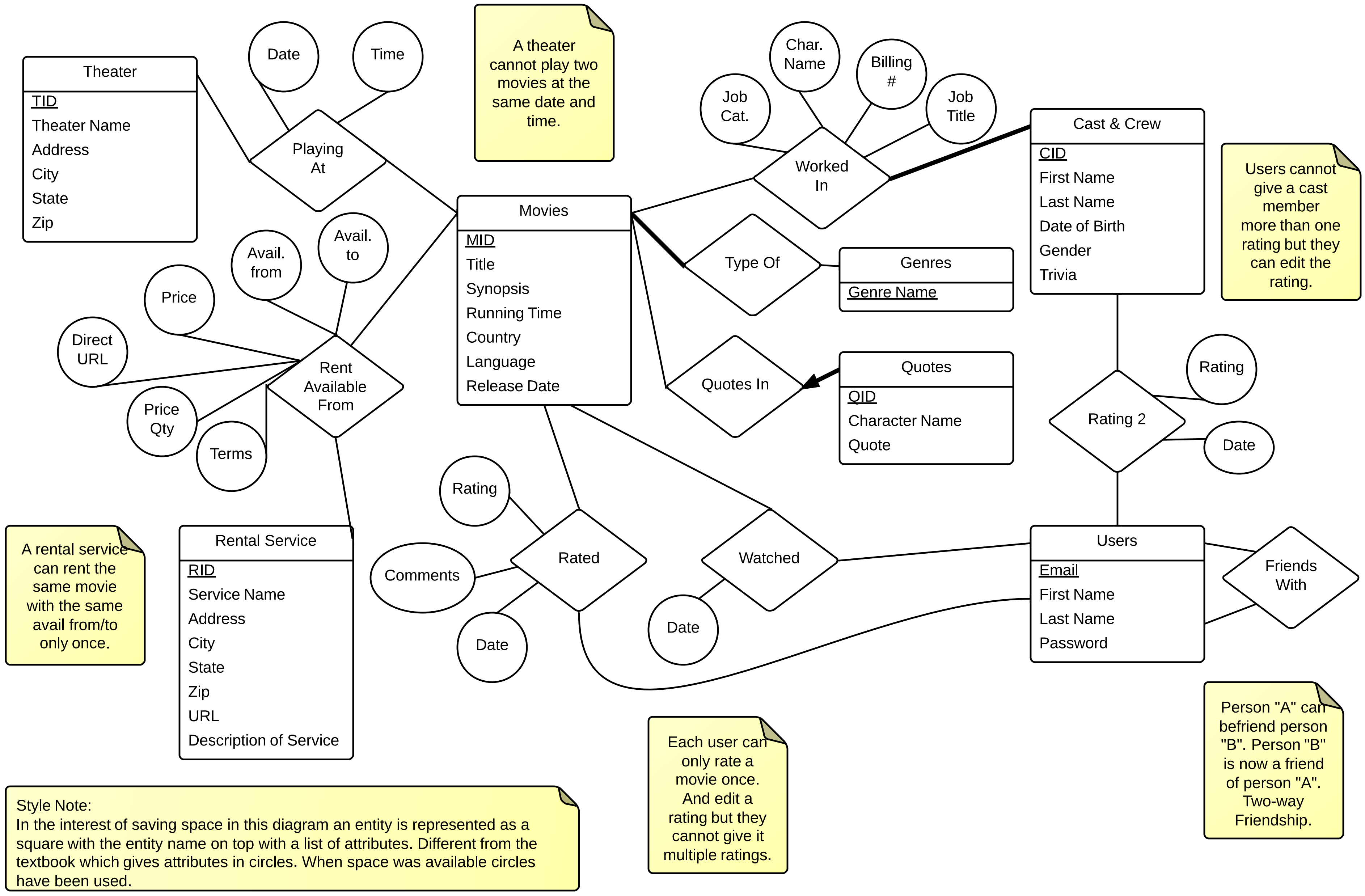
```
/* If user is deleted or the cast member is deleted
 * the corresponding ratings should be deleted
 */
CREATE TABLE rated2
      (rating REAL,
      date DATE,
      cid INTEGER,
      email CHAR(20),
      PRIMARY KEY (cid, email),
      FOREIGN KEY (cid) REFERENCES castcrew
            ON DELETE CASCADE,
      FOREIGN KEY (email) REFERENCES users
            ON DELETE CASCADE)

/* If either user is deleted the relationship should be
 * deleted
 */
CREATE TABLE friendswith
      friendsAEmail CHAR(20),
      friendsBEmail CHAR(20),
      PRIMARY KEY (friendsAEmail, friendsBEmail),
      FOREIGN KEY (friendsAEmail) REFERENCES users (email)
            ON DELETE CASCADE,
      FOREIGN KEY (friendsBEmail) REFERENCES users (email)
            ON DELETE CASCADE)
```

filmdb.com

Entity Relationship Diagram

Chris D'Angelo (cd2665) & Xin Du (xd2137)
February 17, 2013



A theater cannot play two movies at the same date and time.

Users cannot give a cast member more than one rating but they can edit the rating.

A rental service can rent the same movie with the same avail from/to only once.

Each user can only rate a movie once. And edit a rating but they cannot give it multiple ratings.

Person "A" can befriend person "B". Person "B" is now a friend of person "A". Two-way Friendship.

Style Note:
In the interest of saving space in this diagram an entity is represented as a square with the entity name on top with a list of attributes. Different from the textbook which gives attributes in circles. When space was available circles have been used.