Data Management Phase 2

Eric Falkenberg, Harry Longwell, Dave Studin

November 4, 2015

Submission information. Our database was initialized on a MySQL instance that was run locally on one of our laptops. The java code found along with this writeup can interface with a mysql server instance given there is a user by the name of t_user in the database and that t_user has privileges to act on the database 'test'. Along with this, t_user is expected to have the password 'hunter2'.

username t_user

password hunter2

database test

The given java code will use the above credentials to jack into any locally running MySQL server. It will then take the database located at test and (given the 'populate' argument) will wipe the database clean of any relations. Once this has been accomplished, the application will then start to load dummy data into the database. For more information about these queries you can view the actual source queries at schema.inf and data.inf respectively. When run, the code will output the results of the 5 queries our team has devised for this phase of the assignment. In this source code, these queries can be found inside DatabaseConnector.java but for the sake of readability, I will copy them here as well.

```
1)
(SELECT appearance FROM Characters WHERE name='Alex')
UNION
(SELECT e.appearance FROM ((Equips JOIN Equipment AS e ON
equipment=e.name) JOIN Characters as a ON player=a.name)
WHERE a.name=player and player='Alex');

2)
SELECT c.name, m.name, city FROM (Characters AS c JOIN Owns
ON c.name=player) JOIN Mounts AS m ON m.id=mount;

3)
SELECT COUNT(id) FROM (Tracks JOIN Quests ON quest=id) JOIN
Characters ON name=player WHERE player='Mark';

4)
SELECT e.name FROM (Equips JOIN Equipment AS e ON
equipment=name) JOIN Characters AS c ON player=c.name
WHERE c.level >= e.required_level AND
(c.class=e.required_class OR e.required_class IS NULL)
AND player='Alex';

5)
SELECT id, title  FROM (Characters JOIN Tracks ON name=player)
JOIN Quests ON quest=id WHERE level >= required_level AND
(class=required_class OR required_class IS NULL) AND player='Joe';
```

The relational algebra is as follows:

1)

$\pi_{appearance}(\sigma_{name=Alex}(Characters))$

$\cup$

$\pi_{Equipment.appearance}(\sigma_{player=Alex \wedge player=Characters.name}((\sigma_{Equips.equipment=Equipment.name}(Equipment \times Equips)) \times Characters)$

2)

$\pi_{Characters.name,Mounts.name,city}(\sigma_{Mount.id=mount}(\sigma_{Characters.name=player}(Characters \times Owns)) \times Mounts)$

3)

$\pi_{count(id)}(\sigma_{name=player \wedge player=Mark}(\sigma_{quest=id}(Tracks \times Quests)) \times Characters)$

Four and five follow the same format as 2 and 3. They each perform two joins and do the final filtering in the outermost sigma. Then the selection process is done via the projection operator.