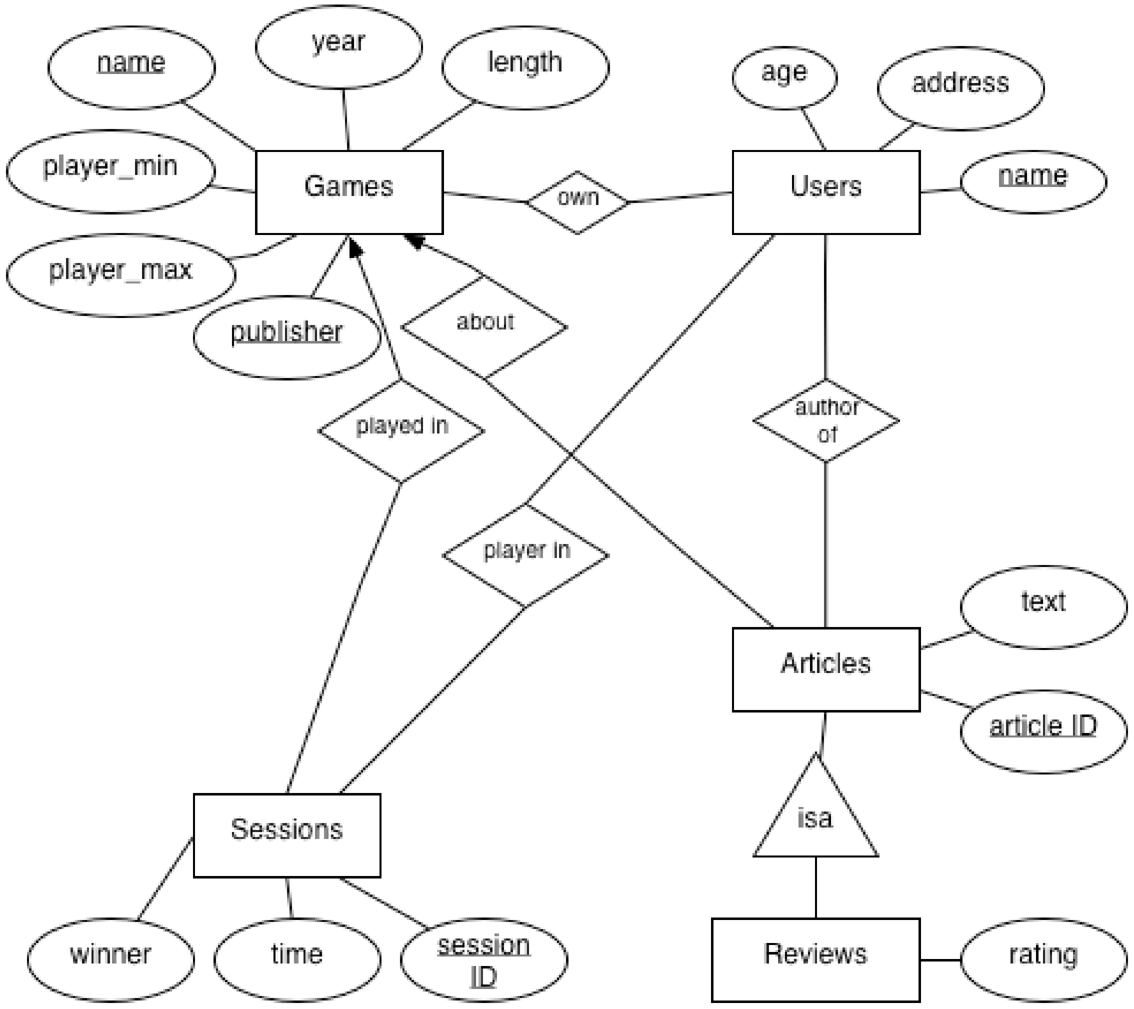
Robert Liguori
Ross Andrews
Game Database Final Write-up

1: Background information
This database exists to store information about board games, players, and sessions of game play. The database can be used to track which player owns what games, and who won what sessions of which game. The exact specifications of the project are as follows.

2: E/R diagram / ODL Schema



Ensure that the number of players in a session falls between player_min and player_max

3: Relations from E/R diagram / Relations from ODL Schema

Entity set relations:
Games(gameName, publisher, year, length, player_min, player_max)
Users(playerName, address, age)
Articles(article_ID, text)
Sessions(session_ID, time, winner)

Relationship relations:
Own(playerName, gameName, publisher)
About(article_ID, gameName, publisher)
Played_in(session_ID, gameName, publisher)
Author_of(article_ID, playerName)
Player_in(session_ID, playerName)

Inheritance relations:
Reviews(article_ID, rating)

Relations generated from ODL schema:

Games(gameName,publisher,year,length,player_min,player_max)
Users(userName,address,age)
Reviews(articleID,rating)

// These next two incorporate the keys for Games, to handle the many-one relationships:
Sessions(sessionID,publisher,gameName,year,month,day,winner)
Articles(articleID,gameName,publisher,text)

player_in(userName,sessionID)
author_of(articleID,userName)
own(userName,gameName,publisher)

4: FDs / MDs (and normalization)
Entity set relations:
Games(gameName, publisher, year, length, player_min, player_max)
Users(playerName, address, age)
Articles(article_ID, text)
Sessions(session_ID, time, winner)

Derived FDs:
GameName, publisher -> GameName, publisher, year, length, player_min, player_max
playerName -> playerName, address, age
article_ID -> article_ID, text
session_ID -> session_ID, time, winner

Relationship relations:
Own(playerName, gameName, publisher)
About(article_ID, gameName, publisher)
Played_in(session_ID, gameName, publisher)
Author_of(article_ID, playerName)
Player_in(session_ID, playerName)

Derived FDs:
Because all elements of the relations are part of each relation's key, all FDs are trivial.

Inheritance relations:
Reviews(article_ID, rating)

Derived FDs:
article_ID -> rating

A relation is in BCNF iff the left-hand sides of all FDs for the relation contains a superkey. The left-hand sides of all above FDs contain a superkey for their relation. Therefore, they are all in BCNF.

If a relation is in BCNF, it is in 3NF. All of the above relations are in BCNF. Therefore, they are in 3NF.

The right-hand sides of all FDs are the set of all attributes of the relation. Therefore, all the MDs derivable from the FDs are trivial. A relation is in 4NF if every non-trivial MD contains a superkey for its relation in its left side. Since there are no non-trivial MD, all of these nonexistent relations contain a superkey on their left-hand sides, and therefore, all of the above relations are in 4NF.

5: SQL Schema / example table entries

```
CREATE TABLE Games
(
      gameName VARCHAR(255),
      publisher VARCHAR(255),
      year INT,
      length_min INT,
      length_max INT,
      player_min INT,
      player_max INT,
      primary key(gameName,publisher)
) type="InnoDB";


Create TABLE Users
(
      userName varchar(255) PRIMARY KEY,
      address TEXT,
      age INT
) type="InnoDB";
```

```
CREATE TABLE Sessions
(
      sessionID int PRIMARY KEY,
      gameName VARCHAR(255),
      publisher VARCHAR(255),
      winner VARCHAR(255),
      time_start INT,
      time_end INT,
      foreign key (winner) references Users(userName),
      foreign key (gameName,publisher) references
Games(gameName,publisher)
) type="InnoDB";


CREATE TABLE Articles
(
      articleID INT PRIMARY KEY,
      gameName VARCHAR(255),
      publisher VARCHAR(255),
      text text,
      foreign key(gameName,publisher) references
Games(gameName,publisher)
) type="InnoDB";


CREATE TABLE Reviews
(
      articleID int primary key,
      rating double
            CHECK (rating >= 0.0 AND rating <= 100.0),
      foreign key (articleID) references Articles(articleID)
) type="InnoDB";


CREATE TABLE player_in
(
      sessionID int,
      userName varchar(255),
      foreign key (sessionID) references Sessions(sessionID),
      foreign key (userName) references Users(userName)
) type="InnoDB";


CREATE TABLE author_of
(
      userName varchar(255),
      articleID int,
      foreign key (userName) references Users(userName),
      foreign key (articleID) references Articles(articleID)
) type="InnoDB";
```

```
CREATE TABLE own
(
      userName varchar(255),
      gameName VARCHAR(255),
      publisher VARCHAR(255),
      foreign key (userName) references Users(userName),
      foreign key (gameName,publisher) references
Games(gameName,publisher)
) type="InnoDB";
```

6: Queries / indexes

Look up one person's collection:
This is supposed to return all the data for all the games owned by one person (in this case, "ehall").

```
select
      Games.*
from
      Games,own
where
      Games.gameName=own.gameName and Games.publisher=own.publisher
      and own.userName="ehall";

(Truncating the output, and showing a row at a time)
*************************** 1. row ***************************
  gameName: Monkeys on the Moon
 publisher: Eight Foot Llama
      year: 2002
length_min: 60
length_max: 60
player_min: 2
player_max: 4
*************************** 2. row ***************************
  gameName: 25 Words or Less
 publisher: Winning Moves
      year: 1996
length_min: 60
length_max: 60
player_min: 4
player_max: 4
```

Look up the game(s) owned by the most people:

```
select
      Games.*,count(userName) as num_owners
from
      own, Games
where
      Games.gameName=own.gameName and Games.publisher=own.publisher
group by
      gameName,publisher
having
      count(userName)=
      (select count(userName) from own group by gameName,publisher
order by count(userName) desc limit 1);

*************************** 1. row ***************************
  gameName: Checkers
 publisher: Public Domain
      year: 1150
length_min: 30
length_max: 30
player_min: 2
player_max: 2
num_owners: 24
```

Look up the game(s) with the most plays:
(This is the game with the most sessions involving it)

```
select
        Games.*,count(Sessions.sessionID) as num_plays
from
        Sessions,Games
where
        Games.gameName=Sessions.gameName and
Games.publisher=Sessions.publisher
group by
        Sessions.gameName,Sessions.publisher
having
        count(Sessions.sessionID)=
        (select count(*) from Sessions group by gameName,publisher
order by count(*) desc limit 1);

(Truncated to one row)
*************************** 1. row ***************************
  gameName: Atilla
 publisher: Rio Grande Games
      year: 2000
length_min: 45
length_max: 45
player_min: 2
player_max: 5
 num_plays: 2
```

Look up the most recently played game (game from the session with the highest end time):

```
select
        Games.*,Sessions.time_start,Sessions.time_end
from
        Sessions,Games
where
        Games.gameName=Sessions.gameName and
Games.publisher=Sessions.publisher
order by
        time_end desc
limit 1;

*************************** 1. row ***************************
  gameName: Bohnanza
 publisher: Rio Grande Games
      year: 1997
length_min: 45
length_max: 45
player_min: 2
player_max: 7
time_start: 7
  time_end: 12
```

Look up the game(s) with the highest average rating:

```
select
        Games.*,round(avg(Reviews.rating),2) as averageRating
from
        Games,Reviews,Articles
where
        Games.gameName=Articles.gameName and
Games.publisher=Articles.publisher
        and Reviews.articleID=Articles.articleID
group by
        Games.gameName,Games.publisher
having
        avg(Reviews.rating)=
        (select avg(rating) from Reviews,Articles where
Reviews.articleID=Articles.articleID group by gameName,publisher order
by avg(rating) desc limit 1);

*************************** 1. row ***************************
     gameName: Quo Vadis?
    publisher: AMIGO Spiel
         year: 1992
   length_min: 45
   length_max: 45
   player_min: 3
   player_max: 5
averageRating: 0.95
```

Indexes:
Creating an index on (gameName, publisher) in the tables Games and own may be worthwhile, as these two attributes form a key for each game. In addition, both gameName and publisher are both strings, and as such have the potential to slow down comparison. A query on a large number of game records with similar names and publishers would be improved by the creation of such an index. Since users should only have access to modify who owns which games, and not to what games the database knows about, the performance hit caused by the creation of this index would not significantly affect the database's functionality.

7: Division of Labor

The initial segments of the project were collaborations, with both team memebers contributing more or less equally. Steps that could be evenly subdivided often were. The vast majority of the actual database code was produced by Ross Andrews, while the test data for the database and write-ups were generally produced by Robert Liguori.

8: Real-world utility

Ultimately, there will never be an enormous market for real-life uses of the game database. Resources such as http://www.boardgamegeek.com/ already exist and already have a great deal of data pre-loaded into them. Since the utility of a database such as this is dependent on the amount of data pre-loaded, and this database has very little chance of attracting as many users as the existing game database solutions, it seems that this project, were it to be released publicly, would permanently lag behind existing solutions in popularity. However, the game database does have a few advantages over boardgamegeek.com. One is its flexibility; any user who wishes to use the software would be free to download a copy and modify it to their taste. Advanced users who wished to execute arbitrary SQL queries would certainly get more mileage out of our solution than an existing one. Another similar issue is the one of security; users who wished to track tournament statistics could use the game database on a secure machine, and thusly ensure that their data is secure.
In conclusion, it appears that the game database may get some use in the real world, but only by a very narrow class of users.