Matthew Toro
CS 340 Introduction to Databases
Assignment: Final Project
Due Date: 8/18/2017

Outline:
The topic of my final project is board games. By board games, I generally refer to most games that can be played on a table. The modern board game industry has been on the rise for the past few years so there is a lot of data on them out there now. My site will allow the user to add data to my board game database, see the results of those additions, and filter on a few select attributes. There are four separate pages that display tables and forms for each respective entity and their relationships.

Database Outline:
I have four entities in my database along with four relationships. My four entities are board games, designers, publishers, and mechanisms. Board games are, obviously, the actual games themselves. Designers are the people who design the games. Publishers are the companies who market and distribute the games. Mechanisms are the game rules included in the game or in other words, the different ways of playing the game that can be grouped under common names.

Board games consist of the attributes: name, age requirement, playing time, max players, artist, and year published. There is also an auto incrementing id attribute. Age, playing time, and max players are integers. Year published is the year data type. Lastly, name and artist are varchars. Artist could have been a separate entity entirely but there is not that much interesting information to include about them in the database so I kept them as an attribute instead.

The designers table has an auto incrementing id integer, and varchars for name and country. I debated splitting up name into first name and last name but since the designers are such a niche category, there is not much information to gain by doing so nor is there much worry for duplicate names being actual unique instances of data.

The publishers table has an auto incrementing id integer, and varchars for name, country, and website.

Lastly, the mechanisms table has an auto incrementing id integer and a varchar for the the mechanism name just referred to as mechanism for short.

There are four relationships among these entities. First, there is a many-to-many relationship between board games and designers in a table called bg_designers. This table has a foreign key reference bg_id that refers to the corresponding id in the board games table. This table also has a foreign key reference d_id that refers to the corresponding id in the designers table. The primary key for the relationship is the pairing of bg_id and d_id.

This is a many-to-many relationship because a board game could have more than one designer via either a co-designer or a re-implementation of an older game. Conversely, designers could have designed more than one board game. I debated whether there is total participation on either side of this relationship. I decided no on a few factors. One reason is because ancient games like chess, go, mancala, or mahjong no longer have a name to associate with its design unlike modern board games. The other reason is because designers may not have a board game either designed yet (they could be in the process of it, or they work as a consultant and edit rule books or something) or have not received a publishing deal yet. So in either way, either entity could exist without the other.

The second relationship is a many-to-many relationship between board games and publishers in a table called bg_publishers. This table has a foreign key reference bg_id that refers to the corresponding id in the board games table. This table also has a foreign key reference p_id that refers to the corresponding id in the publishers table. The primary key for the relationship is the pairing of bg_id and p_id.

This is a many-to-many relationship because a board game could have more than one publisher (they could have a different publisher for release in different countries). Also publishers have catalogs of all the different board games they publish so they can indeed have the license for many board games. I decided for similar reasons that this relationship does not have total participation either. Ancient games don't really have private licenses that can be sold and only distributed by that publisher or a publisher could have just started in the business and has not published anything yet.

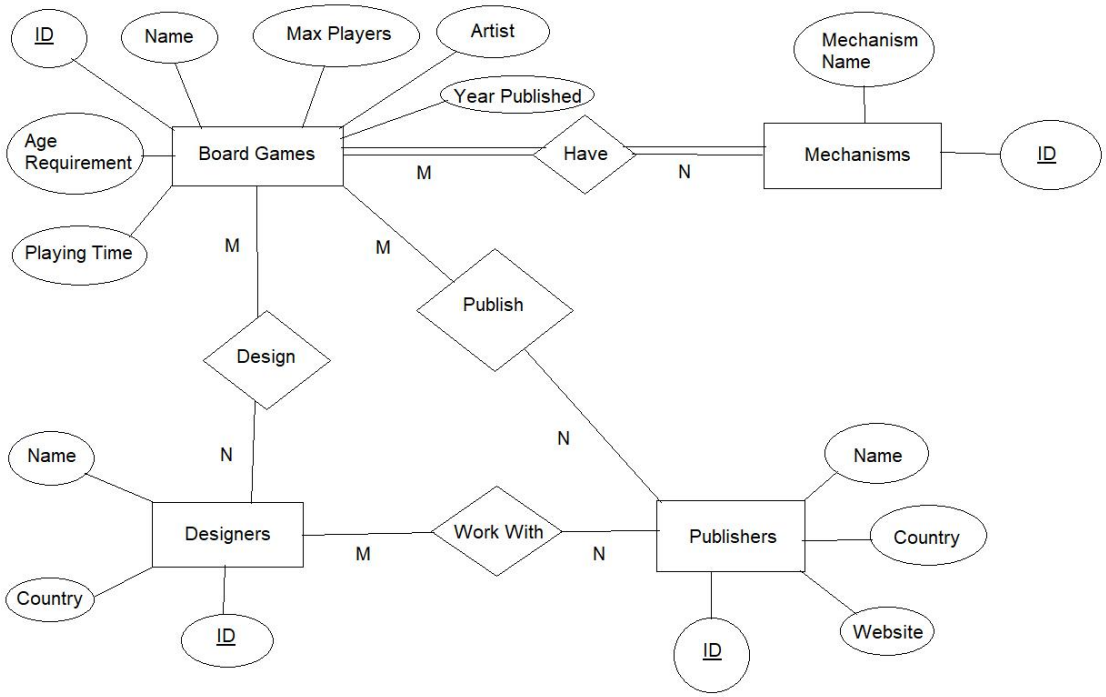
The third relationship is a many-to-many relationship between board games and mechanisms in a table called bg_mechanisms. This table has a foreign key reference bg_id that refers to the corresponding id in the board games table. This table also has a foreign key reference m_id that refers to the corresponding id in the mechanisms table. The primary key for the relationship is the pairing of bg_id and m_id.

This is many-to-many because a board game can have more than one mechanism. There could be dice rolling, hand management, and worker placement all in the same game. Vice versa, multiple games can share the same mechanisms as one game could use tile placement and another uses it as well but in a slightly different manner. This relationship does have total participation on both sides. Board games need mechanisms. They aren't a game without at least one and mechanisms don't really mean anything without a game associated with them. One could roll dice but without a rule about rolling those dice, there is not a game there.

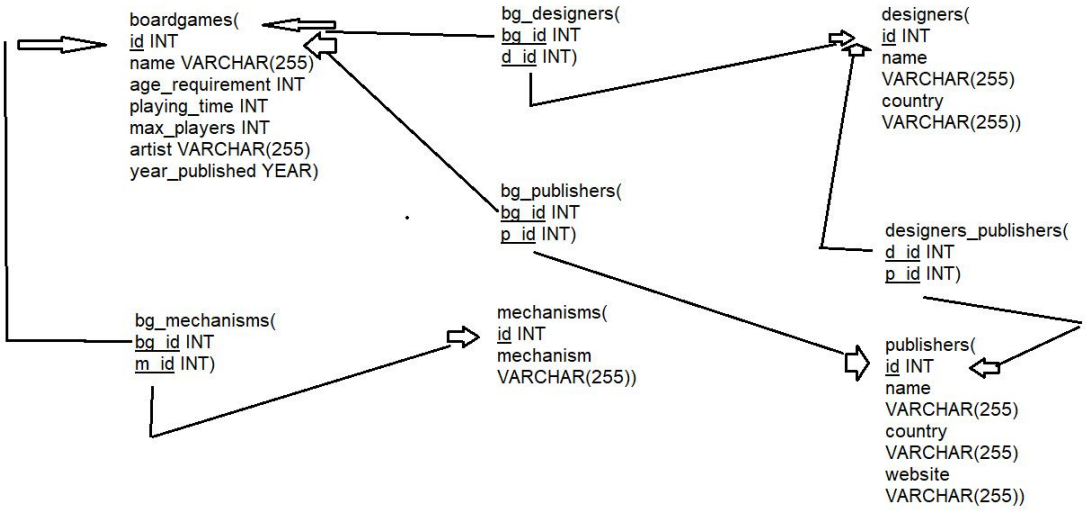
The last relationship is a many-to-many relationship between designers and publishers in a table called designers_publishers. This table has a foreign key reference d_id that refers to the corresponding id in the designers table. This table also has a foreign key reference p_id that refers to the corresponding id in the publishers table. The primary key for the relationship is the pairing of d_id and p_id.

This is many-to-many because a designer could have worked with different publishers at different times. A publisher could have also worked with different designers. However, there is not total participation on either side as designers don't need to work with publishers to design games, they only need to work with publishers if they want to publish their game and publishers don't need to work with designers to publish games, they can distribute already existing games by acquiring the licenses for those games.

ER Diagram:



Schema:



SQL Queries:

DATA DEFINITION QUERIES:

```
CREATE TABLE boardgames(
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(255) NOT NULL,
age_requirement INT,
playing_time INT,
max_players INT,
artist VARCHAR(255),
year_published YEAR NOT NULL,
CONSTRAINT UNIQUE (name)
) ENGINE=InnoDB;

CREATE TABLE designers
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(255) NOT NULL,
country VARCHAR(255),
CONSTRAINT UNIQUE (name)
) ENGINE=InnoDB;

CREATE TABLE publishers(
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
name VARCHAR(255) NOT NULL,
country VARCHAR(255),
website VARCHAR(255),
CONSTRAINT UNIQUE (name)
) ENGINE=InnoDB;

CREATE TABLE mechanisms(
id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
mechanism VARCHAR(255) NOT NULL,
CONSTRAINT UNIQUE (mechanism)
) ENGINE=InnoDB;

CREATE TABLE bg_designers(
d_id INT,
bg_id INT,
PRIMARY KEY (d_id, bg_id),
CONSTRAINT FOREIGN KEY (d_id) REFERENCES designers (id) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT FOREIGN KEY (bg_id) REFERENCES boardgames (id) ON DELETE CASCADE ON UPDATE
CASCADE) ENGINE=InnoDB;

CREATE TABLE bg_publishers(
p_id INT,
bg_id INT,
PRIMARY KEY (p_id, bg_id),
CONSTRAINT FOREIGN KEY (p_id) REFERENCES publishers (id) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT FOREIGN KEY (bg_id) REFERENCES boardgames (id) ON DELETE CASCADE ON UPDATE
CASCADE) ENGINE=InnoDB;

CREATE TABLE designers_publishers(
d_id INT,
p_id INT,
```

```
PRIMARY KEY (d_id, p_id),
CONSTRAINT FOREIGN KEY (d_id) REFERENCES designers (id) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT FOREIGN KEY (p_id) REFERENCES publishers (id) ON DELETE CASCADE ON UPDATE
CASCADE) ENGINE=InnoDB;

CREATE TABLE bg_mechanisms(
m_id INT,
bg_id INT,
PRIMARY KEY (m_id, bg_id),
CONSTRAINT FOREIGN KEY (m_id) REFERENCES mechanisms (id) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT FOREIGN KEY (bg_id) REFERENCES boardgames (id) ON DELETE CASCADE ON UPDATE
CASCADE) ENGINE=InnoDB;
```

DATA MANIPULATION QUERIES:

```
SELECT id, name, age_requirement, playing_time, max_players, artist, year_published
FROM boardgames
ORDER BY name

SELECT id, name, country
FROM designers
ORDER BY name

SELECT id, name, country, website
FROM publishers
ORDER BY name

SELECT id, mechanism
FROM mechanisms
ORDER BY mechanism

SELECT bg_id, d_id, boardgames.name AS bg_name, designers.name AS d_name
FROM bg_designers AS BGD
INNER JOIN boardgames ON boardgames.id = BGD.bg_id
INNER JOIN designers ON designers.id = BGD.d_id
ORDER BY designers.name, boardgames.name

SELECT bg_id, p_id, boardgames.name AS bg_name, publishers.name AS p_name
FROM bg_publishers AS BGP
INNER JOIN boardgames ON boardgames.id = BGP.bg_id
INNER JOIN publishers ON publishers.id = BGP.p_id
ORDER BY publishers.name, boardgames.name

SELECT bg_id, m_id, boardgames.name AS bg_name, mechanisms.mechanism AS m_name
FROM bg_mechanisms AS BGM
INNER JOIN boardgames ON boardgames.id = BGM.bg_id
INNER JOIN mechanisms ON mechanisms.id = BGM.m_id
ORDER BY boardgames.name, mechanisms.mechanism

SELECT d_id, p_id, publishers.name AS p_name, designers.name AS d_name
FROM designers_publishers AS DP
INNER JOIN designers ON designers.id = DP.d_id
INNER JOIN publishers ON publishers.id = DP.p_id
ORDER BY publishers.name, designers.name
```

```
SELECT DISTINCT year_published FROM boardgames

-- Input comes from form
SELECT id, name, age_requirement, playing_time, max_players, artist, year_published
FROM boardgames
WHERE year_published < [form.year_published]
ORDER BY name

--Input comes from form
SELECT id, name, age_requirement, playing_time, max_players, artist, year_published
FROM boardgames
WHERE year_published > [form.year_published]
ORDER BY name

-- id gets selected via a passed query parameter
SELECT id, name, age_requirement, playing_time, max_players, artist, year_published
FROM boardgames WHERE id=[req.query.id]

--input comes from a form
INSERT INTO boardgames (`name`, `age_requirement`, `playing_time`, `max_players`, `artist`,
`year_published`)
VALUES ([form.name], [form.age_requirement], [form.playing_time], [form.max_players],
[form.artist], [form.year_published])

--input comes from a form
INSERT INTO designers (`name`, `country`)
VALUES ([form.name], [form.country])

--input comes from a form
INSERT INTO publishers (`name`, `country`, `website`)
  VALUES ([form.name], [form.country], [form.website])

--input comes from a form
INSERT INTO mechanisms (`mechanism`)
VALUES ([form.mechanism])

--names come from a drop down menu
INSERT INTO bg_designers (`bg_id`, `d_id`)
VALUES ( (SELECT id FROM boardgames WHERE name=[dropdown.name]),
(SELECT id FROM designers WHERE name=[dropdown.name]) )

--names come from a drop down menu
INSERT INTO bg_publishers(`bg_id`, `p_id`)
VALUES ( (SELECT id from boardgames WHERE name=[dropdown.name]),
(SELECT id FROM publishers WHERE name=[dropdown.name]) )

--names come from a drop down menu
INSERT INTO bg_mechanisms(`bg_id`, `m_id`)
VALUES ( (SELECT id from boardgames WHERE name=[dropdown.name]),
(SELECT id FROM mechanisms WHERE mechanism=[dropdown.name]) )

--names come from a drop down menu
INSERT INTO designers_publishers(`p_id`, `d_id`)
VALUES ( (SELECT id from publishers WHERE name=[dropdown.name]),
(SELECT id FROM designers WHERE name=[dropdown.name]) )
```

```
-- id gets selected via hidden input in a form submission
DELETE FROM boardgames WHERE id = [form.id]

-- bg_id and m_id are selected via hidden input in a form submission
DELETE FROM bg_mechanisms WHERE bg_id = [form.bg_id] AND m_id = [form.m_id]

-- attributes are updated via form, id is hidden input that is not changed
UPDATE boardgames
SET name=[form.name], age_requirement=[form.age_requirement],
playing_time=[form.playing_time], max_players=[form.max_players], artist=[form.artist],
year_published=[form.year_published]
WHERE id=[form.id]
```