Andrew M. Calhoun

Aaron Ray Schwartz

Final Project Sheet

December 6th, 2015

Outline:

Gaming is one of the most popular entertainment mediums today and being able to quickly access and create data lists is necessary, especially for collectors. By consolidating the availability, release date, and even reviews of a game, it can become quite easy to track games that are to the taste of the user.

This would be useful for helping people either find gifts for their friends and loved ones or do research into games that might be appropriate or inappropriate for their children and family.

Database Outline:

**Game** - includes game ids, titles, publisher, release date, related review ids, and rating ids. Rating and review ids are foreign keys constrant. The Game id is the primary key constraint. This contains the most pertinent information relating to the games.

**Review** - game\_id constraint foreign key to connect to the game table, contains the review\_id, star\_rating, and text\_description.

**Genre** - contains genre\_id, genre name, description, genre\_id is the primary key constraint.

**Platform -** platform\_id, name, manufacturer,

**Rating -** rating\_id (primary key),

**Game and Platform** - contains game\_id and platform\_id, part of a many to many relationship, each is a foreign key and primary key.

**Game and Genre** - contains game\_id and genre\_id, part of a many to many relationship, each is a foreign key and primary key.

Database Schema

GET FINALIZED DIAGRAM FROM AARON

GET SCHEMA CHART

ER Diagram of the Database:

Table Creation Queries

CREATE TABLE `genre`(

`genre\_id` int(11) PRIMARY KEY AUTO\_INCREMENT,

`name` varchar(255),

`characteristics` text

) ENGINE = 'innoDB';

CREATE TABLE `platform`(

`platform\_id` int(11) PRIMARY KEY AUTO\_INCREMENT,

`name` varchar(255),

`manufacturer` varchar(255),

`generation` varchar(255)

) ENGINE = 'innoDB';

CREATE TABLE `review`(

`review\_id` int(11) PRIMARY KEY AUTO\_INCREMENT,

`star\_rating` float(2,1),

`text\_review` text,

`game\_id` int(11)

) ENGINE = 'innoDB';

CREATE table `rating`(

`rating\_id` int(11) PRIMARY KEY AUTO\_INCREMENT,

`title` varchar(255),

`description` text

### `game\_id` int(11)

### This column is not needed since each rating can apply to many games and we already have the game referencing the rating.

) ENGINE = 'innoDB';

CREATE TABLE `game`(

`game\_id` int(11) PRIMARY KEY AUTO\_INCREMENT,

`title` varchar(255),

`publisher` varchar(255),

`release\_date` date NOT NULL,

`rating\_id` int(11),

`review\_id` int(11),

FOREIGN KEY(`review\_id`) REFERENCES `review`(`review\_id`),

FOREIGN KEY(`rating\_id`) REFERENCES `rating`(`rating\_id`)

-- Move these to a few alter statements to get the table double linked up.

) ENGINE = 'innoDB';

CREATE TABLE `game\_platform`(

`platform\_id` int(11),

`game\_id` int(11),

PRIMARY KEY (`platform\_id`, `game\_id`),

FOREIGN KEY (`platform\_id`) REFERENCES `platform`(`platform\_id`),

FOREIGN KEY (`game\_id`) REFERENCES `game`(`game\_id`)

) ENGINE = 'innoDB';

CREATE TABLE `game\_genre` (

`genre\_id` int(11),

`game\_id` int(11),

PRIMARY KEY (`genre\_id`, `game\_id`),

FOREIGN KEY (`genre\_id`) REFERENCES `genre`(`genre\_id`),

FOREIGN KEY (`game\_id`) REFERENCES `game`(`game\_id`)

) ENGINE = 'innoDB';

General Use Queries:

# Find games that are action oriented games.

-- Select the name, publisher, and release date of a game.

SELECT \* FROM game;

SELECT \* FROM rating;

SELECT \* FROM platform;

SELECT \* FROM review;

SELECT \* FROM genre;

SELECT title, publisher, release\_date FROM game;

SELECT title, publisher, release\_date FROM game WHERE title = [game\_title];

SELECT title, publisher, release\_date FROM game WHERE publisher = [game\_publisher];

SELECT title, publisher, release\_date FROM game WHERE release\_date = [date];

SELECT g.title, g.publisher, g.release\_date, r.title FROM games g INNER JOIN rating r (g.rating\_id = r.rating\_id);

SELECT g.title, g.publisher FROM game g INNER JOIN game\_genre gg ON (gg.game\_id = g.game\_id) INNER JOIN genre ge ON (gg.genre\_id = ge.genre\_id) WHERE ge.name = [GENRE];

SELECT g1.title FROM (SELECT g.title, g.publisher, ge.name FROM ((game g INNER JOIN game\_genre gg ON (gg.game\_id = g.game\_id)) INNER JOIN genre ge ON (ge.genre\_id = gg.genre\_id)) WHERE(ge.name = "[GENRE1]")) as g1

INNER JOIN (SELECT g.title, g.publisher, ge.name FROM ((game g INNER JOIN game\_genre gg ON (gg.game\_id = g.game\_id)) INNER JOIN genre ge ON (ge.genre\_id = gg.genre\_id)) WHERE(ge.name = "[GENRE2]")) as g2 WHERE g1.title = g2.title;

SELECT g.title, g.publisher, ge.name FROM genre ge INNER JOIN game\_genre gg ON ( gg.genre\_id = ge.genre\_id ) INNER JOIN game g ON ( g.game\_id = gg.game\_id );

SELECT g.title, p.name FROM game g INNER JOIN game\_platform gp ON gp.game\_id = g.game\_id INNER JOIN platform p ON p.platform\_id = gp.platform\_id;

SELECT p1.title from (SELECT g.title, p.name FROM game g INNER JOIN game\_platform gp ON gp.game\_id = g.game\_id INNER JOIN platform p ON p.platform\_id = gp.platform\_id WHERE p.name = [SYSTEM1]) AS p1 INNER JOIN

(SELECT g.title, p.name FROM game g INNER JOIN game\_platform gp ON gp.game\_id = g.game\_id INNER JOIN platform p ON p.platform\_id = gp.platform\_id WHERE p.name = "[SYSTEM2]") AS p2 ON p1.title = p2.title;

SELECT g.title, g.release\_date FROM game g WHERE g.game\_id NOT IN (

SELECT g.game\_id FROM game g INNER JOIN game\_platform gp ON gp.game\_id = g.game\_id INNER JOIN platform p ON p.platform\_id = gp.platform\_id WHERE p.name = [SYSTEM]

);

SELECT g.title AS "Game Title", g.publisher AS "Game Publisher", ge.name as "GENRE" FROM game g INNER JOIN game\_genre gg ON gg.game\_id = g.game\_id

INNER JOIN genre ge ON ge.genre\_id = gg.genre\_id

WHERE ge.name = [GENRE\_NAME] AND g.publisher = [COMPANY\_NAME];

SELECT g.title, r.title FROM game g INNER JOIN rating r ON ( r.rating\_id = g.rating\_id ) WHERE g.rating\_id <= [RATING\_LEVEL];

SELECT g.title, r.title FROM game g INNER JOIN rating r ON ( r.rating\_id = g.rating\_id ) WHERE g.rating\_id > [RATING\_LEVEL];

SELECT g.title, r.title FROM game g INNER JOIN rating r ON ( r.rating\_id = g.rating\_id ) WHERE g.rating\_id >= [RATING\_LEVEL];

SELECT g.title, r.title FROM game g INNER JOIN rating r ON ( r.rating\_id = g.rating\_id ) WHERE g.rating\_id = [RATING\_LEVEL];

INSERT INTO platform(name, manufacturer, generation)

VALUE ([PRODUCT\_NAME], [MANUFACTURER], [GENERATION]);

INSERT INTO rating(title, description) VALUE ([TITLE], [DESCRIPTION]);

INSERT INTO game(title, publisher, release\_date, rating\_id)

VALUE ([TITLE], [PUBLISHER], [RELEASE\_DATE], [RATING\_ID]);

INSERT INTO `game\_platform`(`platform\_id`, `game\_id`)

VALUE ([X],[Y]), ([X],[Y]), ([X],[Y])...;

INSERT INTO `game\_genre`(`genre\_id`, `game\_id`)

VALUE ([X],[Y]), ([X],[Y]), ([X],[Y])...;

INSERT INTO review(star\_rating, text\_review, game\_id)

VALUE([RATING], [REVIEW], [GAME\_ID]);

UPDATE game g SET g.review\_id=[REVIEW\_ID] WHERE g.game\_id=[GAME\_ID];

UPDATE game g SET g.review\_id=[REVIEW\_ID] WHERE g.title = [TITLE];

DELETE \* FROM game WHERE title = [TITLE];

DELETE \* FROM game WHERE release\_date = [RELEASE\_DATE];

DELETE \* FROM game WHERE publisher = [PUBLISHER];

DELETE \* FROM platform WHERE name = [PLATFORM];

DELETE \* FROM platform WHERE generation = [GENERATION];

DELETE \* FROM platform WHERE manufacturer = [MANUFACTURER];

DELETE \* FROM genre WHERE name = [GENRE];