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 constraints. 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, and generation that the
Due to the highly variable nature of customizable platforms such as PCs, Macs, tablets, and phones, this particular column was given a varchar rather than a int or float. This is due to the fact that many PCs can have what could be considered 7th gen-equivalent motherboards/processors, and 8th /9th –equivalent gen graphical processing units – ie, an Intel i7-2600k processor, but a Geforce 970GTX graphics card.

Rating - rating_id (primary key), title, description.

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.

Table Relationships:
Game-Review - The game and review tables have a one-to-one relationship managed by a "game_id" and a "review_id". Each table includes these keys which makes them accessible by search queries. For the purposes of our database, a game that is not linked to a review is considered an incomplete entry and will not show up in a search. It will however show up under the "Genre", "Platform", and "Rating System" tabs since that data is provided.

Game-Rating - The game and rating tables use a one-to-many relationship where one rating can apply to multiple games. For example, both Assassin's Creed and Call of Duty: Black Ops III are rated M and only M. This relationship is managed by a "rating_id" that is stored in the game table.

Game-Platform - The game and platform tables have a many-to-many relationship since each game can be on multiple platforms, and each platform can have multiple games. A "game_platform" table that includes the primary keys of both the game and platform was created to manage this relationship.

Game-Genre - The game and genre tables also have a many-to-many relationship, and is managed in the same way that the game-platform relationship is.

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)
FOREIGN KEY(‘game_id`) REFERENCES game(‘game_id);
) ENGINE = 'innoDB';
```

```
CREATE table `rating`(
`rating_id` int(11) PRIMARY KEY AUTO_INCREMENT,
`title` varchar(255),
`description` text
) 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`)
) 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:

INSERT Queries:

```
INSERT INTO game(title, publisher, release_date, rating_id) VALUES ([title], [publisher],
[release_date], [rating_id]);
        /*Note: the user will specify the desired rating (E, T, M, etc.) and the program will pass
        the rating_id to the query*/
```

```
INSERT INTO game_genre (genre_id, game_id) VALUES ([genre_id], game_id);
        /*Note: the user will specify the desired genre (Action, Adventure, etc.) and the program
        will pass the genre_id to the query.  Game_id is retrieved using a previously executed
        SELECT query*/
INSERT INTO game_platform (platform_id, game_id) VALUES ([platform_id], game_id);
        /*Note: the user will specify the desired platform (PC, N64, etc.) and the program will
        pass the platform_id to the query.  Game_id is retrieved using a previously executed
        SELECT query*/
INSERT INTO genre (name, characteristics) VALUES ([genre_name], [genre_characteristics]);
INSERT INTO platform (name, manufacturer, generation) VALUES ([platform_name],
[platform_manufacturer], [platform_generation]);
INSERT INTO rating (title, description) VALUES ([rating_title], [rating_description]);
INSERT INTO review (star_rating, text_review, game_id) VALUES ([star_rating], [text_review],
[game_id]);
        /*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.)
        and the program will pass the game_id to the query*/
```

SELECT Queries:

```
SELECT * from game;
SELECT * FROM genre;
SELECT * from platform;
SELECT * FROM rating;
SELECT * FROM review WHERE review_id=[review_id];
        /*Note: the user will specify the desired review and the program will pass the rating_id to
        the query*/
SELECT * FROM review INNER JOIN game ON (game.review_id = review.review_id);
SELECT game_id FROM game WHERE title=[game_title];
SELECT review_id FROM review WHERE game_id=[game_id]);
SELECT review_id FROM game WHERE review_id =[review_id]);
        /*Note: the user will specify the desired review and the program will pass the rating_id to
        the query*/
SELECT * FROM game WHERE rating_id=[rating_id];
        /*Note: the user will specify the desired rating (T, M, etc.) and the program will pass the
        rating_id to the query*/
SELECT * FROM game_genre WHERE game_id=[game_id];
        /*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.)
        and the program will pass the game_id to the query*/
SELECT * FROM game_platform WHERE game_id=[game_id];
        /*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.)
        and the program will pass the game_id to the query*/
SELECT g.game_id, g.title AS gamename, g.publisher, g.release_date, r.title FROM game g
INNER JOIN rating r ON (r.rating_id = g.rating_id) ORDER BY game_id ASC;
SELECT g.game_id, g.title FROM game g;
```

SELECT * FROM review WHERE game_id=[game_id]
/*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.) and the program will pass the game_id to the query*/
SELECT * FROM review r INNER JOIN game g ON g.game_id = r.game_id
SELECT * FROM genre WHERE genre_id=[genre_id];
/*Note: the user will specify the desired genre (Action, Driving, etc.) and the program will pass the genre_id to the query*/
SELECT g.title AS gTitle, g.publisher, g.release_date, r.title AS rTitle FROM game g INNER JOIN rating r ON (r.rating_id = g.rating_id) INNER JOIN game_platform gp ON (g.game_id = gp.game_id) INNER JOIN platform p ON (p.platform_id = gp.platform_id) WHERE p.platform_id=platform_id
SELECT g.title as gTitle, r.title as rTitle FROM game g INNER JOIN rating r ON (g.rating_id = r.rating_id) WHERE g.rating_id=rating_id;
SELECT game.title AS gTitle, game.publisher, game.release_date, rating.title AS rTitle, review.star_rating FROM game INNER JOIN review ON game.review_id=review.review_id INNER JOIN rating ON game.rating_id=rating.rating_id INNER JOIN game_genre ON game.game_id=game_genre.game_id INNER JOIN genre ON game_genre.genre_id=genre.genre_id INNER JOIN game_platform ON game.game_id=game_platform.game_id INNER JOIN platform ON game_platform.platform_id=platform.platform_id WHERE [user specified criteria];
/*Note: the part of the query before the "WHERE" is the same no matter what criteria is being searched. anything after the "WHERE" is depends on what specifically is being searched. For example, if the user is searching for a specific game it would read "WHERE game.title=[game_title];". If the user searches for genre and platform it will read "WHERE genre.name=[genre_name] AND platform.name=[platform_name]".*/

DELETE Queries:
DELETE FROM game WHERE game.game_id=[game_id];
/*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.) and the program will pass the game_id to the query*/
DELETE FROM game_genre WHERE game_id=[game_id];
/*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.) and the program will pass the game_id to the query*/
DELETE FROM game_platform WHERE game_id=[game_id];
/*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.) and the program will pass the game_id to the query*/
DELETE FROM review WHERE game_id=[game_id];
/*Note: the user will specify the desired game (Donkey Kong 64, Assassin's Creed, etc.) and the program will pass the game_id to the query*/
DELETE FROM genre WHERE genre.genre_id=[genre_id];
/*Note: the user will specify the desired genre (Action, Driving, etc.) and the program will pass the genre_id to the query*/
DELETE FROM platform WHERE platform.platform_id=[platform_id];

/*Note: the user will specify the desired platform (PC, N64, etc.) and the program will pass the platform_id to the query.*/

DELETE FROM rating WHERE rating_id=[rating_id];

/*Note: the user will specify the desired rating (T, M, etc.) and the program will pass the rating_id to the query*/

DELETE rating_id FROM game WHERE rating_id=[rating_id];

/*Note: the user will specify the desired rating (T, M, etc.) and the program will pass the rating_id to the query*/

DELETE FROM review where review_id=[review_id];

/*Note: the user will specify the desired review and the program will pass the rating_id to the query*/

UPDATE Queries:

UPDATE game SET review_id=review_id WHERE game_id=game_id;

/*Note: both review_id and game_id are obtained using previously executed SELECT queries.*/

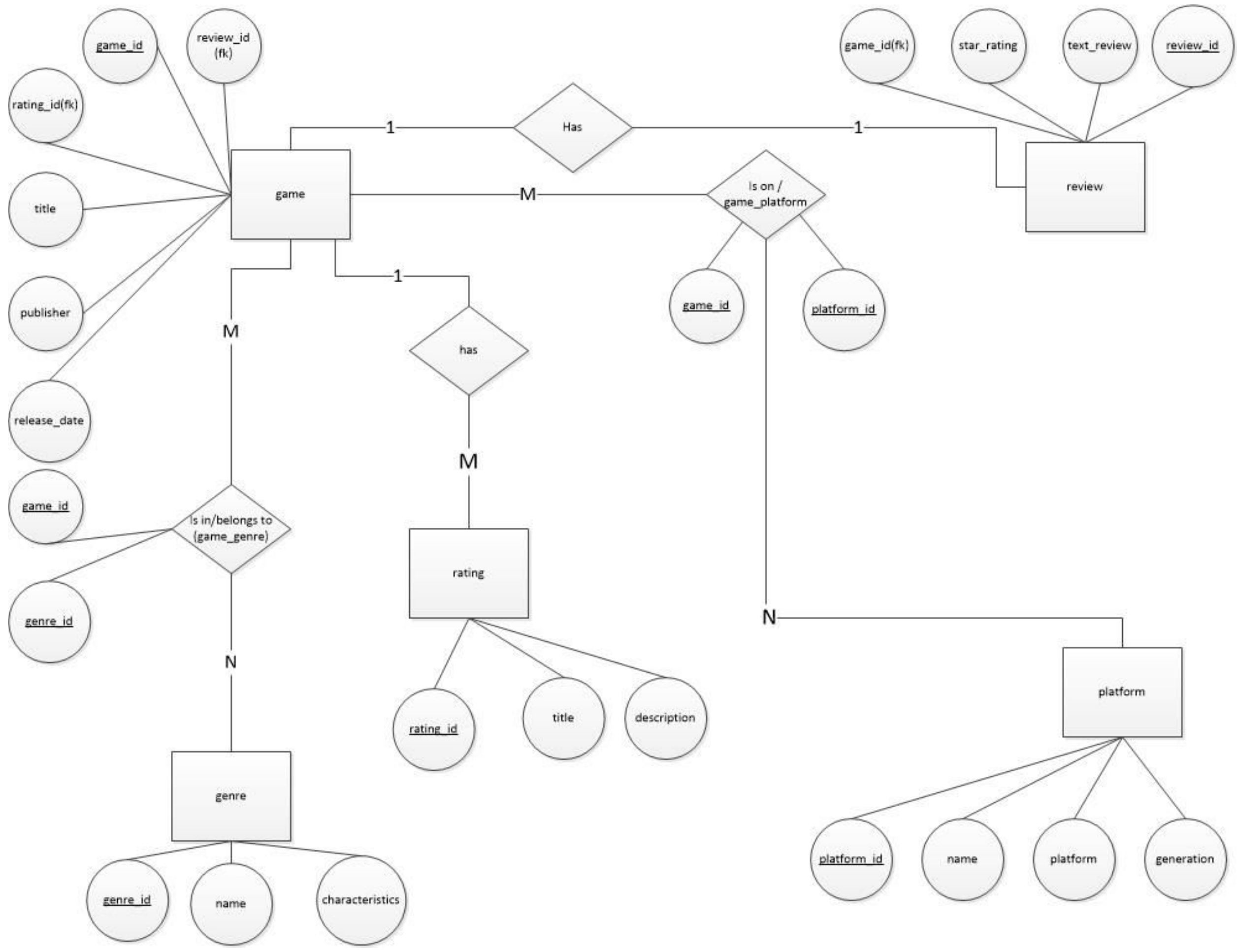
UPDATE game_id SET review_id = 'NULL' WHERE review_id=[review_id];

/*Note: the user will specify the desired review and the program will pass the rating_id to the query*/

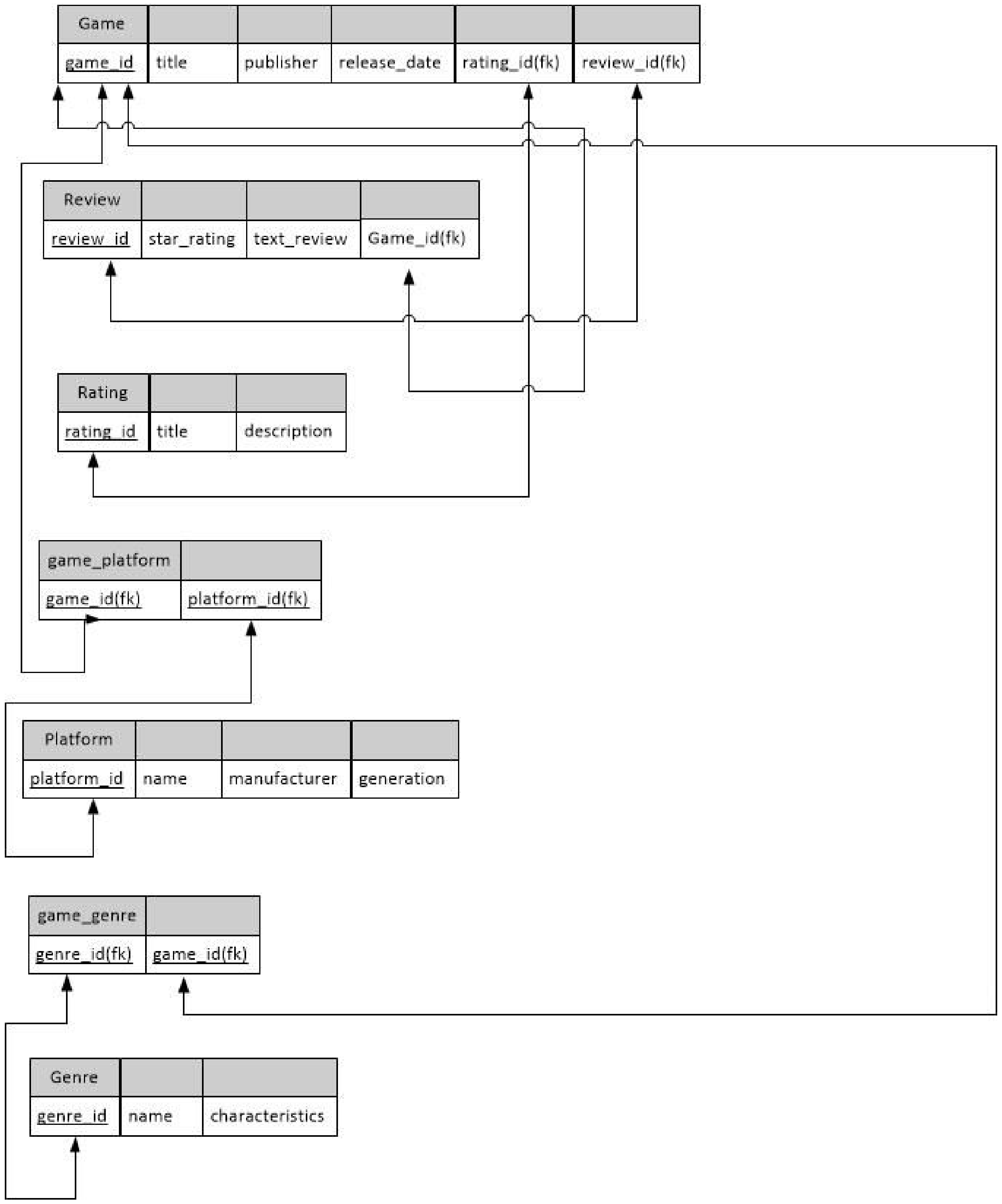
Utility Queries:

SET foreign_key_checks = 0;
SET foreign_key_checks = 1;

PROJECT DIAGRAM - RELATIONSHIPS AND TABLES



PROJECT GENERAL ER DIAGRAM


Game
game_id
title
publisher
release_date
rating_id(fk)
review_id(fk)
Review
review_id
star_rating
text_review
Game_id(fk)
Rating
rating_id
title
description
game_platform
game_id(fk)
platform_id(fk)
Platform
platform_id
name
manufacturer
generation
game_genre
genre_id(fk)
game_id(fk)
Genre
genre_id
name
characteristics