Database

By: Matthew Musich

CMPT 308

Table of Contents

Executive Summary

The Humble Bundle Database is a representation of the Humble Bundle independent game distribution service. It keeps track of all of the games in each bundle that is currently being sold. This includes what platform the game is available on, who developed each game, and ways to download the game when the bundle containing it is purchased by a user. It can also keep track if the user purchased the bundle above the average price, which gives the user additional games. This Database will make it easy to addl games, bundles, and users in the future.

Entity Relationship Diagram

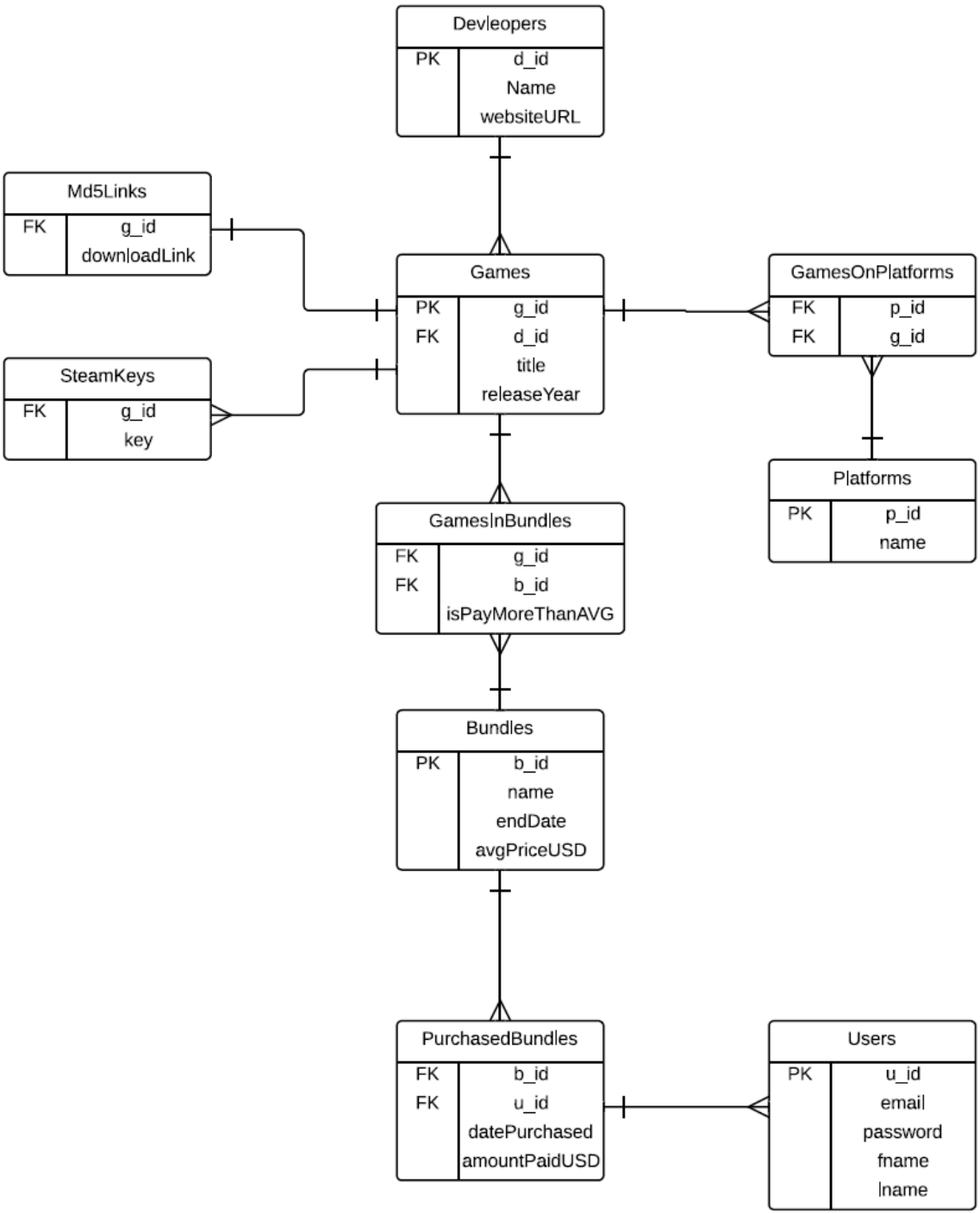

Table Descriptions:

This section describes all of the tables' description, functional dependencies, creation statements and some sample data that could be seen within the database.

Developer Table

Description

Stores the name and website of the developers whose games are within the database

Functional Dependencies

d_id -> name, websiteURL

Table Creation

```
DROP TABLE IF EXISTS Developers;
CREATE TABLE Developers (
  d_id     int not null,
  name     char (100) not null,
  websiteURL char (250),
  primary key (d_id)
);
```

Sample Data

	d_id integer	name character(100)	websiteurl character(250)
1	1	Terry Cavanag	http://distractionware.com/
2	2	Jonathan Blov	http://number-none.com/blow/
3	3	2D Boy	http://2dboy.com/
4	4	Kloonigames	http://www.kloonigames.com/blog/
5	5	Team Meat	http://supermeatboy.com/
6	6	Gaijin Games	http://gaijingames.com/
7	7	Studio Pixel	http://studiopixel.sakura.ne.jp/
8	8	Supergiant Ga	http://supergiantgames.com/
9	9	Vlambeer	http://www.vlambeer.com/
10	10	Mike Bithell	http://mikebithellgames.com/

Games Table

Description

Stores the name and release year of the games and references the Developers table for d_id.

Functional Dependencies

g_id -> title, releaseYear

Table Creation

```
DROP TABLE IF EXISTS Games;
CREATE TABLE Games (
  g_id     int not null,
  d_id     int not null REFERENCES Developers(d_id),
  title    char (100) not null,
  releaseYear int,
  primary key (g_id)
);
```

Sample Data

	g_id integer	d_id integer	title character(100)	releaseyear integer
1	1	1	Super Hexagon	2012
2	2	2	Braid	2008
3	3	3	World of Goo	2008
4	4	4	Crayon Physics Deluxe	2009
5	5	5	Super Meat Boy	2010
6	6	6	Bit.Trip Runner	2011
7	7	7	Cave Story+	2011
8	8	8	Bastion	2011
9	9	9	Wasteland Kings	2013
10	10	10	Thomas Was Alone	2012

Md5Links Table

Description

Stores the link to the download for each game.

Functional Dependencies

None

Table Creation

```
DROP TABLE IF EXISTS Md5Links;
CREATE TABLE Md5Links (
  g_id    int not null REFERENCES Games(g_id),
  downloadLink    char (250) not null
);
```

Sample Data

	g_id integer	downloadlink character(250)
1	1	http://terrycavanaghgames.com/hexagon/
2	2	https://hb1.ssl.hwcdn.net/braid full 1015.exe
3	3	http://www.worldofgoo.com/
4	4	https://hb1.ssl.hwcdn.net/crayon release55 2.exe
5	5	https://hb1.ssl.hwcdn.net/SuperMeatBoySetup.exe
6	6	https://hb1.ssl.hwcdn.net/BIT.TRIPRUNNER WebSetup.exe
7	7	https://hb1.ssl.hwcdn.net/cave story plus-windows-1356583089.zip
8	8	https://hb1.ssl.hwcdn.net/Bastion.msi
9	9	https://hb1.ssl.hwcdn.net/WASTELAND KINGS.zip
10	10	https://hb1.ssl.hwcdn.net/thomaswasalone-pc-1369347074.zip

SteamKeys Table

Description

Stores multiple keys for each associated game.

Functional Dependencies

None

Table Creation

```
DROP TABLE IF EXISTS SteamKeys;
CREATE TABLE SteamKeys (
  g_id    int not null REFERENCES Games(g_id),
  key    char (17) not null
);
```

Sample Data

	g_id integer	key character(17)
1	1	ABCDE-EFGHI-JKLMN
2	1	QWERT-EFGHI-JKLMN
3	2	TREWQ-EFGHI-JKLMN
4	2	ASDFG-EFGHI-JKLMN
5	3	GFDSA-EFGHI-JKLMN
6	3	ZXCVB-EFGHI-JKLMN
7	4	BVCXZ-EFGHI-JKLMN
8	4	POIUY-EFGHI-JKLMN
9	5	YUIOP-EFGHI-JKLMN
10	5	LKJHG-EFGHI-JKLMN
11	6	GHJKL-EFGHI-JKLMN
12	6	CVBNM-EFGHI-JKLMN
13	7	MNBVC-EFGHI-JKLMN
14	7	SDFGH-EFGHI-JKLMN
15	8	ERTYU-EFGHI-JKLMN
16	8	UYTRG-EFGHI-JKLMN
17	9	HYTGF-EFGHI-JKLMN
18	9	DRFGF-EFGHI-JKLMN
19	10	JUYSV-EFGHI-JKLMN
20	10	XTHEB-EFGHI-JKLMN

GamesOnPlatforms Table

Description

Connects each game with its multiple possibilities of each platform.

Functional Dependencies

None

Table Creation

```
DROP TABLE IF EXISTS GamesOnPlatforms;
CREATE TABLE GamesOnPlatforms (
  p_id    int not null REFERENCES Platforms(p_id),
  g_id    int not null REFERENCES Games(g_id)
);
```

Sample Data

	p_id integer	g_id integer
1	1	1
2	1	2
3	2	3
4	1	4
5	1	5
6	2	6
7	1	7
8	1	8
9	3	9
10	1	10

Platforms Table

Description

Stores all of the possible platforms that the games can be played on.

Functional Dependencies

g_id -> name

Table Creation

```
DROP TABLE IF EXISTS Platforms;
CREATE TABLE Platforms (
  p_id    int not null,
  name    char(50) not null,
  primary key (p_id)
);
```

Sample Data

	p_id integer	name character(50)
1	1	Windows
2	2	Mac
3	3	Linux

GamesInBundles Table

Description

Connects what games are contained in each bundle based on id's, and can tell you if the game is unlockable if the average price is beaten.

Functional Dependencies

None

Table Creation

```
DROP TABLE IF EXISTS GamesInBundles;
CREATE TABLE GamesInBundles (
  g_id    int not null REFERENCES Games(g_id),
  b_id    int not null REFERENCES Bundles(b_id),
  isPayMoreThanAVG    boolean not null
);
```

Sample Data

	g_id integer	b_id integer	ispaymorethanavg boolean
1	1	1	f
2	2	1	f
3	3	1	f
4	4	1	t
5	5	1	t
6	6	2	f
7	7	2	f
8	8	2	f
9	9	2	f
10	10	2	t

Bundles Table

Description

Stores each of the bundles that are currently for sale, or have been on sale in the past, containing the name, end date, and the current average price in USD.

Functional Dependencies

b_id -> name, endDate, avgPriceUSD

Table Creation

```
DROP TABLE IF EXISTS Bundles;
CREATE TABLE Bundles (
  b_id     int not null,
  name     char (100) not null,
  endDate     date not null,
  avgPriceUSD decimal (7,2) not null,
  primary key (b_id)
);
```

Sample Data

	b_id integer	name character(100)	enddate date	avgpriceusd numeric(7,2)
1	1	Humble Bundle 1	2013-12-25	6.53
2	2	Humble Bundle 2	2013-12-31	4.88

PurchasedBundles Table

Description

Shows what bundles the user has purchased based on the bundles' ids and records how much the user spent on the bundle

Functional Dependencies

None

Table Creation

```
DROP TABLE IF EXISTS PurchasedBundles;
CREATE TABLE PurchasedBundles (
  b_id    int not null REFERENCES Bundles(b_id),
  u_id    int not null REFERENCES Users(u_id),
  datePurchased    date not null,
  amountPaidUSD decimal (10,2) not null
);
```

Sample Data

	b_id integer	u_id integer	datepurchased date	amountpaidusd numeric(10,2)
1	1	1	2013-12-06	4.00
2	1	2	2013-12-05	7.00
3	1	3	2013-12-02	9.00
4	1	6	2013-12-09	10.00
5	2	5	2013-12-26	6.00
6	2	1	2013-12-28	5.00
7	2	6	2013-12-28	7.00
8	2	4	2013-12-29	8.00
9	2	7	2013-12-30	8.00
10	2	2	2013-12-31	15.00

Users Table

Description

Shows what Users exist and it stores their email, a hashed password, along with their first and last name. They are assigned an id to uniquely identify them.

Functional Dependencies

u_id -> email, password, fname, lname

Table Creation

```
DROP TABLE IF EXISTS Users;
CREATE TABLE Users (
  u_id    int not null,
  email    char (100) not null,
  password char (40),
  fname char (50),
  lname char (50),
  primary key (u_id)
);
```

Sample Data

	u_id integer	email character(100)	password character(40)	fname character(50)	lname character(50)
1	1	matthew.musich1@marist.edu	7a495904a8c0b3e6aabe27440b436c28	Matthew	Musich
2	2	testing@hotmail.com	7a495904a8c0b3e6aabe27440b436c28	Bob	Smith
3	3	emailman@yahoo.com	7a495904a8c0b3e6aabe27440b436c28	Jake	Pope
4	4	gamefan@gmail.com	7a495904a8c0b3e6aabe27440b436c28	Henry	Goodridge
5	5	myemail@aol.com	7a495904a8c0b3e6aabe27440b436c28	Mike	Pitz
6	6	imthebest@gmail.com	7a495904a8c0b3e6aabe27440b436c28	Colby	Fresh
7	7	howtoemail@hotmail.com	7a495904a8c0b3e6aabe27440b436c28	Gary	Dean

Views

This view will show the user what Steam Keys are available to them, along with the bundle name, game name, and who made the game. This view would be used the most by the user. It allows quick search for the steam keys that the user will want to use.

```
Create View AvailableSteamKeys AS
SELECT Distinct
b.name, g.title, d.name, sk.key
FROM
Bundles b, GamesInBundles gib, Games g, SteamKeys sk, Users u, Developers d,
PurchasedBundles pb
WHERE
u.u_id = pb.u_id AND
pb.b_id = b.b_id AND
b.b_id = gib.b_id AND
gib.g_id = g.g_id AND
g.g_id = sk.g_id
ORDER BY
g.title ASC
```

Reports and Queries

This query will show the games and the bundles that a specific developer has had a game in.

```
SELECT
b.name, b.name, g.title
FROM
Bundles b, GamesInBundles gib, Games g, Developers d
WHERE
b.b_id = gib.b_id AND
gib.g_id = g.g_id AND
g.d_id = d.d_id AND
d.name = ‘Terry Cavanagh’
```

This query will show the platforms that a specific game is on along with the name of the game.

```
SELECT
g.title, p.name
FROM
Games g, Platforms p, GamesOnPlatforms gop
WHERE
g.g_id = gop.g_id AND
gop.p_id = p.p_id AND
g.title = 'Super Hexagon'
```

Stored Procedures

This stored procedure will show the games that are in the specified bundle, and display their titles and the bundle they came from.

```
CREATE FUNCTION bundleGames(b.name text)
returns table (b.name text, g.title text ) as $$
SELECT
b.name, g.title
FROM
Bundle b, Games g, GamesInBundles gib
WHERE
b.b_id = gib.b_id AND
gib.g_id = g.g_id

$$ language 'sql';
select * from bundleGames(‘b.name’);
```

Triggers

This trigger will check at the time of purchase if the amount of money spent is above the average price, and will add the unlockable games to what the user has purchased.

```
Create Trigger beatTheAverage
After Insert or Update
On PurchasedBundles
For Each row
Execute Procedure beatTheAverageCheck();
```

Security

Security is very important when handling over the web transactions. In order to keep the payments secure, all transactions will be held at a 3rd party site that controls all of the security.

In addition all of the users' passwords are stored as md5 hashes which will prevent the possible leak of passwords if the database is compromised.

Implementation

This database system can easily be implemented with the create statements provided. It can also be tweaked to fit any other purchasable content based system, by adding or removing tables. More views can be made in order to display more data to the user and keep track of the library of games/bundles a user has purchased.

Known Problems

The only current issue is that there is no tables that deal with payment, because the user will be paying with a 3rd party site like PayPal. I would be unsure how to incorporate that into a new table.

Future Enhancements

The addition of other types of content other than games could be added, such as movies, soundtracks, and e-books. Along with finding a better way to store games as a user, such as a library table.