Steam Backlog Manager Database Design

Krisztián Köves

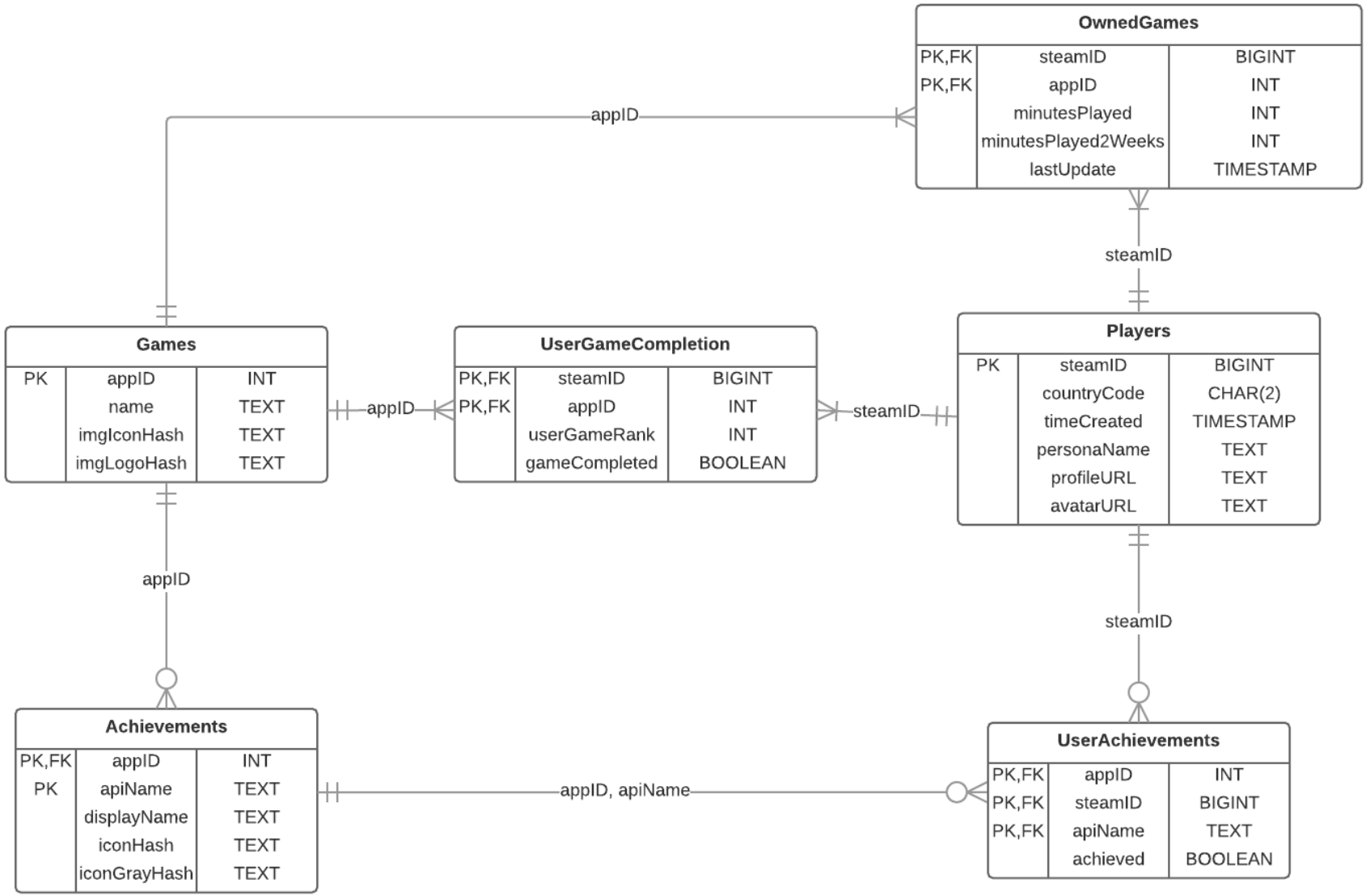
Table of Contents

Executive Summary

This is a database design for Steam Backlog Manager, a web application that helps people with large video game libraries on Steam "manage" their backlogs by deciding what games they want to play, and in what order. The data is all pulled from the Steam Web API, and refreshed whenever the user logs in to the web application.

This database design was implemented and tested on PostgreSQL 9.5.1, which was released on February 11th, 2016.

Entity-Relationship Diagram



Players Table

This table stores the users of the web application, as well as some basic information to show on their home page.

```
CREATE TABLE IF NOT EXISTS Players (
  steamID     BIGINT    NOT NULL UNIQUE,
  countryCode CHAR(2)   NOT NULL,
  timeCreated TIMESTAMP NOT NULL,
  personaName TEXT      NOT NULL,
  profileURL  TEXT      NOT NULL,
  avatarURL   TEXT      NOT NULL,

  PRIMARY KEY (steamID)
);
```

Functional Dependencies:

steamID → countryCode, timeCreated, personaName, profileURL, avatarURL

Sample Data:

steamID	countryCode	timeCreated	personaName	profileURL	avatarURL
76561198055529452	'US'	1324858286	'sKKy'	…	…
76561197972495328	'US'	1063407589	'Robin'	…	…

Games Table

This table contains basic information about Steam games, so that the website does not have to fetch them from the Steam API every time a user requests data from the site. The full logo and icon URLs are generated from a consistent base URL (on Steam servers), the game's appID, and a hash that identifies each image.

```
CREATE TABLE IF NOT EXISTS Games (
  appID       INT UNIQUE NOT NULL,
  name        TEXT NOT NULL,
  imgIconHash TEXT,
  imgLogoHash TEXT,

  PRIMARY KEY (appID)
);
```

Functional Dependencies:

`appID` → `name`, `imgIconHash`, `imgLogoHash`

Sample Data:

appID	name	imgIconHash	imgLogoHash
220	'Half-Life 2'	…	…
400	'Portal'	…	…
620	'Portal 2'	…	…
72850	'The Elder Scrolls V: Skyrim'	…	…
22370	'Fallout 3 - Game of the Year Edition'	…	…

OwnedGames Table

This linking table stores the relationships between players and games, as well as how much time players have spent on each game. The `lastUpdate` field is stored to check "freshness" of the data, and refresh data when the user signs in to the website.

```
CREATE TABLE IF NOT EXISTS OwnedGames (
  steamID              BIGINT NOT NULL,
  appID                INT NOT NULL,
  minutesPlayed        INT NOT NULL,
  minutesPlayed_2Weeks INT NOT NULL,
  lastUpdate           TIMESTAMP NOT NULL,

  PRIMARY KEY(steamID, appID),
  FOREIGN KEY(steamID)
      REFERENCES Players(steamID),
  FOREIGN KEY(appID)
      REFERENCES Games(appID)
);
```

Functional Dependencies:

(steamID, appID) → minutesPlayed, minutesPlayed_2Weeks, lastUpdate

Sample Data:

steamID	appID	minutesPlayed	minutesPlayed_2Weeks	lastUpdate
76561198055529452	220	1618	0	'2016-04-26 11:00:00.00'
76561198055529452	400	5087	0	'2016-04-26 11:00:00.00'
76561198055529452	620	6435	0	'2016-04-26 11:00:00.00'
76561198055529452	72850	20413	0	'2016-04-26 11:00:00.00'
76561198055529452	22370	1898	0	'2016-04-26 11:00:00.00'

UserGameCompletion Table

This linking table stores users' preferences of the order in which they will play their games, and whether or not they have finished those games.

```
CREATE TABLE IF NOT EXISTS UserGameCompletion (
  steamID       BIGINT NOT NULL,
  appID         INT NOT NULL,
  userGameRank  INT NOT NULL,
  gameCompleted BOOLEAN NOT NULL,

  PRIMARY KEY(steamID, appID),
  FOREIGN KEY(steamID)
      REFERENCES Players(steamID),
  FOREIGN KEY(appID)
      REFERENCES Games(appID)
);
```

Functional Dependencies:

(steamID, appID) → userGameRank, gameCompleted

Sample Data:

steamID	appID	userGameRank	gameCompleted
76561198055529452	72850	1	TRUE
76561198055529452	220	2	TRUE
76561198055529452	620	3	TRUE
76561198055529452	400	4	TRUE
76561198055529452	22370	5	FALSE

Achievements Table

This table stores game achievements, when available for each game. They are stored along with the display name, description, and icons that users will already recognize from Steam.

```
CREATE TABLE IF NOT EXISTS Achievements (
  appID        INT  NOT NULL,
  apiName      TEXT NOT NULL UNIQUE,
  displayName  TEXT NOT NULL,
  description  TEXT,
  iconHash     TEXT NOT NULL,
  iconGrayHash TEXT NOT NULL,

  PRIMARY KEY(appID, apiName),
  FOREIGN KEY(appID)
      REFERENCES Games(appID)
);
```

Functional Dependencies:

(appID, apiName) → displayName, description, iconHash, iconGrayHash

Sample Data:

appID	apiName	displayName	Description	iconHash	iconGrayHash
220	‘HL2_GET_CROWBAR’	‘Trusty Hardware’	…	…	…
400	‘PORTAL_GET_PORTALGUNS’	‘Lab Rat’	…	…	…
620	‘ACH.WAKE_UP’	‘You Monster’	…	…	…
72850	‘NEW_ACHIEVEMENT_8_0’	‘Dragonslayer’	…	…	…

UserAchievements Table

This linking table stores users' game achievement status for each game's achievement.

```
CREATE TABLE IF NOT EXISTS UserAchievements (
  steamID  BIGINT  NOT NULL,
  appID    INT     NOT NULL,
  apiName  TEXT    NOT NULL,
  achieved BOOLEAN NOT NULL,

  PRIMARY KEY(steamID, appID, apiName),
  FOREIGN KEY(steamID)
      REFERENCES Players(steamID),
  FOREIGN KEY(appID)
      REFERENCES Games(appID),
  FOREIGN KEY(apiName)
      REFERENCES Achievements(apiName)
);
```

Functional Dependencies:

(steamID, appID, apiName) → achieved

Sample Data:

steamID	**appID**	**apiName**	**achieved**
76561198055529452	220	'HL2_GET_CROWBAR'	TRUE
76561198055529452	400	'PORTAL_GET_PORTALGUNS'	TRUE
76561198055529452	620	'ACH.WAKE_UP'	TRUE
76561198055529452	72850	'NEW_ACHIEVEMENT_8_0'	FALSE

MostPlayedGames View

This view outputs a list of games, ordered by the number of people who play each game.

```
CREATE OR REPLACE VIEW MostPlayedGames AS
  SELECT DISTINCT o.appID, g.name, t.playerCount
  FROM (SELECT o.appID, COUNT(o.appID) AS playerCount
        FROM Players p,
             OwnedGames o
        WHERE o.steamID = p.steamID
        GROUP BY o.appID
       ) t,
       OwnedGames o,
       Games g
   WHERE t.appID = g.appID
     AND t.appID = o.appID
   ORDER BY playerCount DESC;
```

Sample Output:

appID	name	playerCount
220	‘Half-Life 2’	2
400	‘Portal’	2
620	‘Portal 2’	2
72850	‘The Elder Scrolls V: Skyrim’	2
22370	‘Fallout 3 – Game of the Year Edition’	1

UnfinishedGames Report

This report generates a list of users, showing how many unfinished games each user has in their account.

```
CREATE OR REPLACE VIEW UnfinishedGames AS
  SELECT steamID, COUNT(*) AS unfinishedGamesCount
  FROM UserGameCompletion
  WHERE gameCompleted = FALSE
  GROUP BY steamID;
```

Sample Output:

steamID	unfinishedGamesCount
76561198055529452	1
76561197972495328	1

Stored Procedures

userUnfinishedGames(bigint, refcusor)

This function returns a list of unfinished games in a user's account, for the web application to display to the user.

```
CREATE OR REPLACE FUNCTION unfinishedGames(bigint, refcursor) RETURNS refcursor AS
$$
DECLARE
    userSteamID bigint    := $1;
    resultset   REFCURSOR := $2;
BEGIN
    OPEN resultset FOR
        SELECT *
        FROM UserGameCompletion
        WHERE steamID = userSteamID
          AND gameCompleted = FALSE;
    RETURN resultset;
END;
$$
LANGUAGE plpgsql;
```

Sample Output:

steamID	appID	userGameRank	gameCompleted
76561198055529452	22370	5	FALSE

Security

There are only two user groups needed for this database: one for the web application to use, which can select, insert, and update records; and an admin user, who is the only one that can delete records from the database, for security reasons.

```
CREATE ROLE webapp_admin;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO admin;

CREATE ROLE webapp_user;
GRANT SELECT, INSERT, UPDATE
ON ALL TABLES
IN SCHEMA PUBLIC
TO webapp_user;
```

Implementation Notes

Known problems:

The main problem I would encounter in fully implementing this database is that of inconsistent source data from the Steam Web API. For example:

- Some games are duplicated on Steam for different versions of the game (e.g., "Game of the Year" editions)
- No icon or logo images for some games
- Duplicated achievement API names, but this is a unique field in my database design
    - Saw this with Bethesda games in particular

Another design aspect that could be a problem in the future is the `steamID` field – I set it to type `BIGINT`, since steam user IDs are currently stored as 64-bit numbers. Valve could change this in the future, so it would be wiser to store the `steamID` field as data type `TEXT`.

Future enhancements:

If I were to improve upon this database design in the future, I would like to track more than just Steam games – I could add other game distribution platforms, such as Origin, Humble Bundle, uPlay, and GOG. However, this is dependent on each platform having a publicly-accessible web API. Here are some other features that I would like to implement:

- Give users more ways to track their progress than just hours, such as tracking by achievements that are earned as a player progresses through a game's story
- More game status markers than just "completed", such as – "not interested in game," "tried game, but not interested," etc.
- Ranking options by genre and Steam Community tags