LEGO Visual Database

Final Project for CS464/564 Spring 2018

Elizabeth E. Esterly
The University of New Mexico
Dept. of Computer Science: CS 564
eesterly@unm.edu

Justin D. Thomas
The University of New Mexico
Dept. of Computer Science: CS 464
jthomas105@unm.edu

1 DATABASE DESIGN

In this report, we describe the contents of our database, its intended use, its basic structure, and how the tables are connected.

Further, we give details on how we meet other requirements of the project, such as number of entities and domain standards. We also include our E/R diagram and Relational Schema as Appendices to this document.

1.1 *Background on LEGOs*

LEGO *sets* are divided into *parts*. LEGO *minifigs* are also divided into *parts*, which have their own part id as well as a general part id. Minifigs are *theme* members, and can be members of multiple themes and sets. They have their own *year* of release that is independent of the year of release of a given set. For example, a Gandalf the Grey minifig that appeared alongside Saruman in a Lord of the Rings Tower of Orthanc set from 2013 appeared in 2015 with Batman in a LEGO Dimensions set that was a tie-in with the LEGO Movie. However, the Gandalf the Grey figure was released in 2012.

1.2 *Database Description*

Our database contains data describing over 10,000 LEGO sets. Sets are chronicled in a detailed manner, with their names theme membership, and year of release stored. The set contents are similarly well-described. Data is included on individual parts that comprise a set: their colors, names, and even images of parts are included and searchable by color. Minifigs and their names, parts, and theme membership (and by extension, set membership) are also included.

Data was gathered from several disparate sources. Basic data on set contents originated from a dataset available on Kaggle. Part images came from Rebrickable. A custom script was written to scrape minifig data from Bricklink. These features differentiate our database from existing LEGO databases and as such extends the capabilities on what is currently available online. Our database consists of twelve tables, listed below. See the E/R diagram and Relational schema for more detailed information on the primary and foreign keys of these tables, their cardinality, and relationships.

1) *users:* usernames, id, email, and Boolean value of admin status
2) *passwords:* passwords associated with usernames, hashed and salted
3) *favorite sets:* set numbers associated to usernames
4) *sets:* descriptions of sets with LEGO set numbers, year of release, number of parts, and the theme
5) *themes:* id and name of theme, and the parent theme (some themes are sub-themes)
6) *minifigs:* theme name that minifig has membership in, part number, description, and year of release
7) *parts:* part name, category, and number
8) *colors:* id, name code, and Boolean value of transparency status
9) *set_contents:* set id that the part is a member of, color id of the part, part number, quantity of part in the set, and Boolean value of a part's spare part status in the set
10) *minifig_contents:* minifig part number, overall part number
11) *part_categories:* part category id, part category name
12) *part_images:* images of each part, organized by color membership; multi-color parts appear in multiple colors

1.3 *Intended Use*

We intend for our database to be a resource for LEGO collectors, enthusiasts, or just those with a casual interest in LEGOs who are curious about a certain set, theme, or minifigure.

We envision three main classes of users. The first, a dedicated collector, could find images of every single part that comprised their favorite set and create an extensive catalogued inventory. The second user might be an enthusiast who needed to replace some minifigs for their set. Imagine they need a knight and an archer. There are a lot of LEGO knights, archers, and other minifigs of this type released over many years, but each belongs to only certain sets and themes. How can the collector ensure they have the right knight and archer for their set? They could search our database and find the exact minifig that belongs to their set, allowing them to complete their set accurately. And last, the casual user might be interested in looking up

LEGO sets that they played with as a child, or perhaps they want to purchase a set online but couldn't remember the name of it.

1.4 *Structure and Interaction*

Here we describe in general terms how different tables in the database interact with each other and give examples of information that can be retrieved.

- Users can search for sets by description, theme, and year.
- Users can search for themes by name.
- Users can search for minifigs by year, theme, and description (the description includes the name).
- Parts lists for sets and minifigs are recoverable by name and part number.
- In the case of minifigs, minifig parts are recoverable by both minifig part number as well as general part number.
- Additional information on each part is also searchable, for example a part's color and spare part status or quantity within a set.
- Part images are searchable by color and part number.
- Users create usernames and passwords to gain access to the database.
- Authenticated users can store their favorite sets to retrieve and compare.

1.5 *E/R Diagram*

Please see Appendix 1 at the end of this document for the ER Diagram.

1.6 *Relational Schema*

Please see Appendix 2 at the end of this document for the Relational Schema. The cardinality of the relationships is expressed in the endpoints of the lines drawn between entities.

1.7 *Number of entities*

Our group consists of two people. We use the formula given in the project specifications: $\mathbf{6 + 4(n-1)}$ to calculate that we need at least $\mathbf{6 + 4(2-1) = 10}$ entities. Our database has 12 entities; therefore, we have fulfilled the minimum entity requirement.

1.8 *Domain Standards*

We comply with SSAE 16 standards to ensure our users' password information is protected. We use PHP's *password_hash* when storing the user passwords, which uses the Blowfish encryption algorithm and salts the hash. Passwords are also stored in a separate table from usernames.

2 USER ACCESS

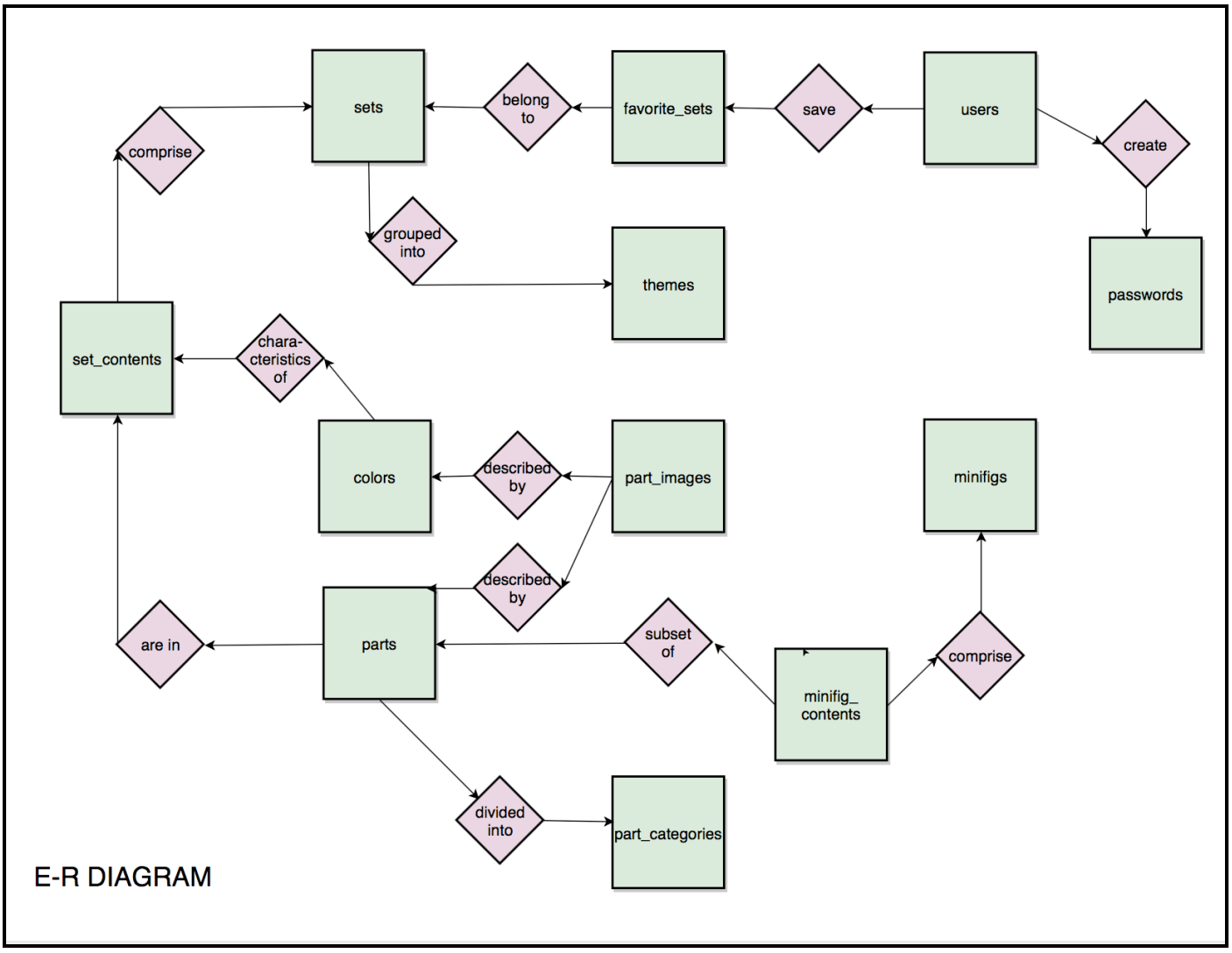
Here we describe user access to the tables and our reasoning behind granting that level of access for each table.

1) *users:* update
2) *passwords:* update
3) *favorite sets:* update
4) *sets:* access
5) *themes:* access
6) *minifigs:* access
7) *parts:* access
8) *colors:* access
9) *set_contents:* access
10) *minifig_contents:* access
11) *part_categories:* access
12) *part_images:* access

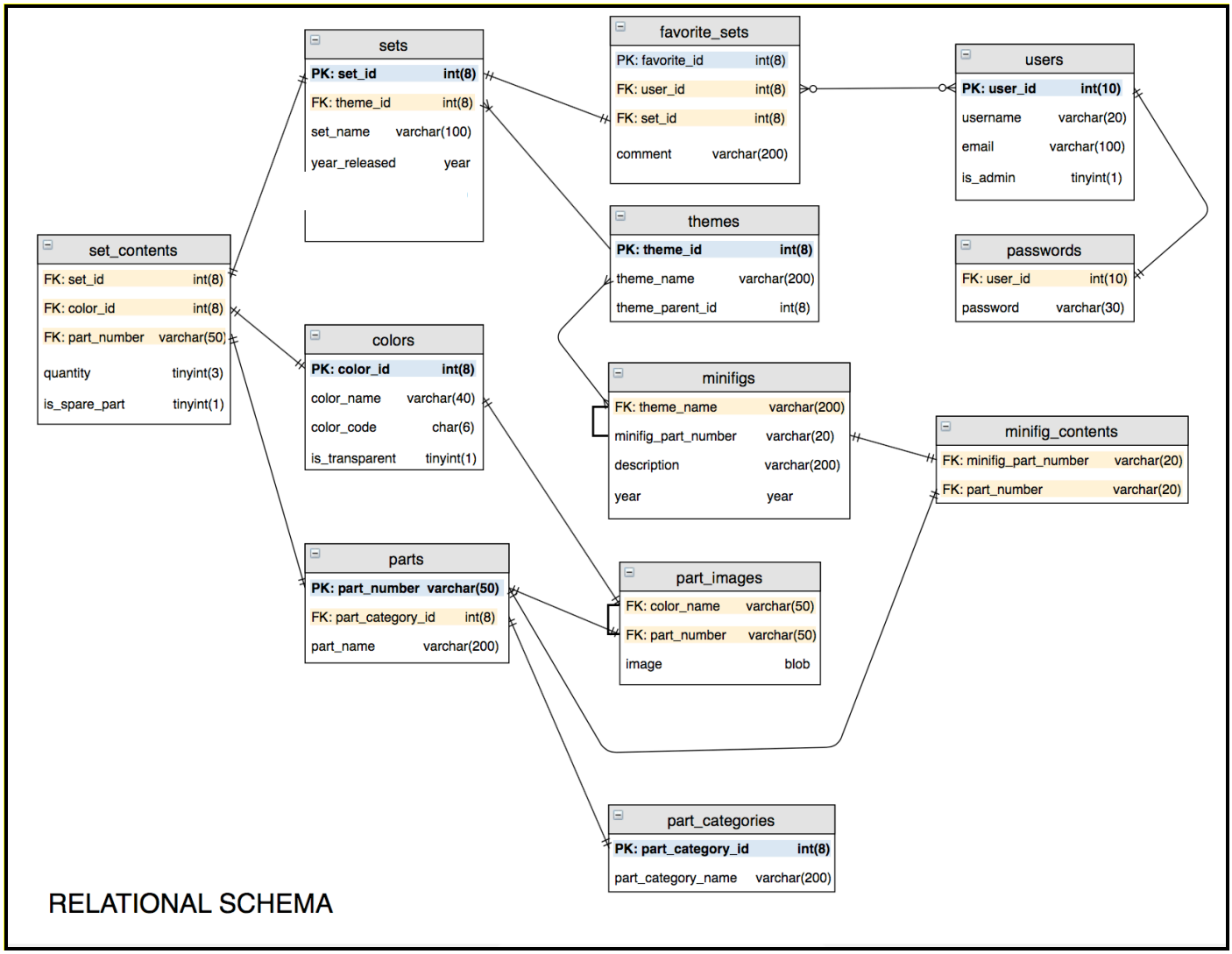
Users have update permissions for tables 1—3. Users will need to register a new username and password for themselves (tables 1 and 2), and save their favorite sets (table 3). The remainder of the tables are access-only. There is no need for users to have the ability to alter that information as this database is meant to serve as a tool for users to be able to search a consistent inventory.

3 CONTENTS LISTING

1) Esterly_Thomas_finalreport.pdf
2) students.txt
3) structure.sql
4) data.sql
5) queries.sql
6) application.zip
7)



APPENDIX A. E/R DIAGRAM.



APPENDIX B. RELATIONAL SCHEMA.