Project Team #25 – Lego Database

Team Members: Jeremy Sloan, Bowen Young

PROJECT PROPOSAL

Content, Scope, and Objectives

LEGO is a popular toy brand that has been entertaining children and adults for decades. With its diverse range of themes and products, LEGO has captured the imaginations of millions of people worldwide. To better understand the LEGO products, we propose to build a LEGO Database using MySQL.

The objective of this project is to create a relational database of LEGO products using MySQL. This database will provide a comprehensive list of LEGO products, including their names, themes, product descriptions, and release dates. By analyzing this database, we can identify trends and patterns in LEGO product releases over the years.

We will be using the LEGO Database from Kaggle, which includes information on more than 10,000 LEGO products. The data set is comprehensive and includes details such as product names, themes, product descriptions, release dates, and pricing information.

We will use MySQL, the leading global database management system that operates using Structured Query Language (SQL), to build the LEGO Database. The database will be designed using the relational model, which will ensure that data is organized efficiently and can be accessed easily. We will also implement appropriate normalization techniques to eliminate data redundancy and ensure data consistency.

By building the LEGO Database using MySQL, we can gain insights into the LEGO product lines and their trends over the years. This database will help LEGO enthusiasts and researchers understand the evolution of the LEGO brand and its products. Using MySQL, we can organize, manage, manipulate, and erase data in a structured way that aligns perfectly with our

project requirements. We believe that this project will contribute significantly to the LEGO community and be a valuable resource for future research.

Sprint 0

PROJECT ENVIRONMENT

For this project, we employ MySQL, the leading global database management system. MySQL is a relational database management system that operates using Structured Query Language (SQL). The application caters to a range of tasks such as data warehousing, e-commerce, and logging. Using MySQL, we can organize, manage, manipulate, and erase data in a structured way that aligns perfectly with our project requirements.

HIGH LEVEL REQUIREMENTS

Initial user roles

User Role	Description
Lego Enthusiasts	Individuals who are interested in the Lego sets and products.
Lego Store Employees	Users who work for the LEGO company and have access to the company's database system for various purposes such as inventory management, sales analysis, and reporting.
Lego Set Collectors	Individuals who collect Lego sets and need access to the database to keep track of their collection.
Administrators	Users responsible for managing the LEGO database system and ensuring its security and integrity.

Initial user story descriptions

Story ID	Story description
US-1	As a Lego enthusiast, I want to be able to search for sets by set number, name, theme, or year of release so that I can find the sets I am interested in.
US-2	As a Lego enthusiast, I want to be able to see a detailed description of a set including the number of pieces, minifigures included, and a picture so that I can make an informed decision about purchasing the set.
US-3	As an employee, I want to be able to view real-time inventory levels and receive notifications when stock levels fall below a certain threshold so that I can efficiently manage inventory and ensure product availability.
US-4	As an administrator, I want to be able to create and manage user accounts and assign roles and permissions to ensure the security and integrity of the database system.
US-5	As an employee, I want to be able to generate reports on sales trends, product popularity, and other relevant metrics to better understand customer behavior and improve sales strategies.
US-6	As a Lego Set Collector, I want to be able to add sets to my collection in the database so that I can keep track of which sets I own.
US-7	As a Lego Set Collector, I want to be able to see a list of all the sets I own, along with the set number, name and year of release so that I can keep track of my collection.

HIGH LEVEL CONCEPTUAL DESIGN

Entities:

- Sets
- Parts
- Colors
- Inventories
- Inventory_Parts
- Inventory_Sets
- Themes
- Part_Categories

Relationships:

- A set can have multiple parts.
- A part can have multiple colors.
- An inventory can have multiple sets and parts.
- An inventory can have multiple inventory parts.
- An inventory can have multiple inventory sets.
- An inventory part belongs to a part and an inventory.
- An inventory set belongs to a set and an inventory.
- Each set has one theme.
- Each part has one part_category.

Sprint 1

REQUIREMENTS

Refine the user stories that you made in previous sprint. List your updated user stories and any notes you wish to include in decreasing order of priority and highlight the stories chosen for Sprint 1. *There is no need to show your story refinement process - just the list of updated stories suffices.* Use the format shown below.

Story ID	**Story description**
US1	As a <role>, I want to <need/feature> so that <reason/benefit>
...	...

CONCEPTUAL DESIGN

Include your detailed conceptual design here. Use the format shown below.

Entity: **Entity1**
Attributes:
    attr1_a
    attr1_b [composite]
        part_1
        part_2

Entity: **Entity2**
Attributes:
    attr2_a

attr2_b [multi-valued]
attr2_c [derived]

Relationship: **Entity1** relationship-phrase **Entity2**
Cardinality: <One/Many> to <One/Many>
Participation:
Entity1 has <partial/total> participation
Entity2 has <partial/total> participation

LOGICAL DESIGN

Include your logical design here. Use the format shown below.

Table: **Table1**
Columns:
pk_1
column_1a
column_1b

Justification (if needed)

Table: **Table2**
Columns:
pk_2
column_2a
column_2b [foreign key; references **pk_1** of **Table1**]

Justification (if needed)

SQL QUERIES

List at least **three** SQL queries that perform data retrievals relevant to the features chosen in the current sprint. For each query, paste a **screenshot** of the output, as shown through database management tool.

Sprint 2

REQUIREMENTS

Refine the user stories that you made in previous sprint. List your updated user stories in decreasing order of priority. Highlight the stories for which database design was completed in Sprint 1 in one color. Highlight the updated/new stories chosen for Sprint 2 in a different color. *There is no need to explicitly show your story refinement process*. Use the format shown below.

Story ID	**Story description**
US1	As a <role>, I want to <need/feature> so that <reason/benefit>
...	...

CONCEPTUAL DESIGN

Include your complete updated conceptual design here. Use the format shown below.

Entity: **Entity1**
Attributes:
- attr1_a
- attr1_b [composite]
  - part_1
  - part_2

Entity: **Entity2**
Attributes:
- attr2_a
- attr2_b [multi-valued]
- attr2_c [derived]

Relationship: **Entity1** relationship-phrase **Entity2**
Cardinality: <One/Many> to <One/Many>
Participation:
- Entity1 has <partial/total> participation
- Entity2 has <partial/total> participation

LOGICAL DESIGN WITH NORMAL FORM IDENTIFICATION

Include your complete updated logical design here. Use the format shown below.

Table: **Table1**

Columns:

pk_1

column_1a

column_1b

Justification of primary key (if needed)

Highest normalization level: <1NF/2NF/3NF/BCNF>
Justification (if below BCNF):

Table: **Table2**

Columns:

pk_2

column_2a

column_2b [foreign key; references **pk_1** of **Table1**]

Justification of primary key (if needed)

Highest normalization level: <1NF/2NF/3NF/BCNF>
Justification (if below BCNF):

SQL QUERIES

Refine your SQL queries that you designed in the previous sprint if in need. List at least **three** SQL queries that perform data retrievals relevant to the features chosen in the current sprint. For each query, paste a **screenshot** of the output, as shown through your user interface.

Sprint 3

REQUIREMENTS

Refine the user stories that you made in previous sprint. List your updated user stories in decreasing order of priority. Highlight the stories that were completed in Sprint 1 in one color. Highlight the stories that were completed in Sprint 2 in a different color. Highlight the updated/new stories chosen for Sprint 3, if any, in a third color. *There is no need to explicitly show your story refinement process*. Use the format shown below.

Story ID	**Story description**
US1	As a <role>, I want to <need/feature> so that <reason/benefit>
...	...

CONCEPTUAL DESIGN

Include your complete updated conceptual design here. Use the format shown below.

Entity: **Entity1**
Attributes:
- attr1_a
- attr1_b [composite]
  - part_1
  - part_2

Entity: **Entity2**
Attributes:
- attr2_a
- attr2_b [multi-valued]
- attr2_c [derived]

Relationship: **Entity1** relationship-phrase **Entity2**
Cardinality: <One/Many> to <One/Many>
Participation:
- Entity1 has <partial/total> participation
- Entity2 has <partial/total> participation

LOGICAL DESIGN WITH HIGHEST NORMAL FORMS AND INDEXES

Include your complete updated logical design here. Use the format shown below.

Table: **Table1**
Columns:
pk_1
column_1a
column_1b

Justification of primary key (if needed)

Highest normalization level: <1NF/2NF/3NF/BCNF>
Justification (if below BCNF):

Indexes:
Index #: <type (clustered/non-clustered)>
Columns: <ordered list of columns forming the index>
Justification:

Table: **Table2**
Columns:
pk_2
column_2a
column_2b [foreign key; references **pk_1** of **Table1**]

Justification of primary key (if needed)

Highest normalization level: <1NF/2NF/3NF/BCNF>
Justification (if below BCNF):

Indexes:
Index #: <type (clustered/non-clustered)>
Columns: <ordered list of columns forming the index>
Justification:

VIEWS AND STORED PROGRAMS

List the views relevant to your application here. Use the format specified below.

View: <name of view>
Goal: <1-2 sentence description of what the view contains and what its purpose is (e.g., why and what user(s) would use it, etc.)>

List the stored programs relevant to your application thus far here. Use the format specified below for the different kinds of stored programs. Note: if you do not have a particular type of stored program in your application, just leave that part out.

Stored procedure: <name of procedure>
Parameters: <list of parameters, specifying IN/OUT/INOUT for each>
Goal: <1-2 sentence description of what the stored procedure does>

Stored function: <name of function>
Parameters: <list of parameters>
Goal: <1-2 sentence description of what the stored function does and what it returns>

Trigger: <type of trigger> on <table name>
Goal: <1-2 sentence description of what the trigger does>

Event: <type of event>
Goal: <1-2 sentence description of what the event does>