Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Project Report

Information Management II - CSU34041

Prathamesh Sai
3rd year Integrated Computer Science
Student ID: 19314123

School of Computer Science and Statistics,
Trinity College Dublin

Contents

1 Application Description **2**

1.1 What is Growtopia? 2

1.2 How will this database be used in the context of Growtopia? 2

2 Entity Relationship Diagram **3**

3 Mapping to Relational Schema **5**

4 Functional Dependency Diagram (for proposed relations) **6**

5 Explanation of one of the SQL code for creating one of your database tables (including any constraints). **7**

6 Explanation and SQL code for any Altering tables **7**

7 Explanation and SQL code for any Trigger operations **8**

8 Explanation and SQL code for any creation of Views **9**

9 Explanation and SQL Code for one of your commands to Populate a Tables **10**

10 Explanation and example SQL Code for retrieving information from the database (including any use of Joins and use of functions). **10**

11 Explanation and SQL Code for any Security commands (roles & permissions). **11**

12 Explanation of any additional SQL features of your choice **12**

13 References **12**

1 Application Description

For this project, we are expected to make a ER Model for information to be represented for an application of our own choice and implement it as a MySQL database. I decided to make my application based on a vending machine business in Growtopia.

1.1 What is Growtopia?

Growtopia is a sandbox MMO game with almost endless possibilities for world creation, customization and having fun with your friends [1]. Each player has their own 'GrowID', and can explore new worlds. A world can be locked with a 'World Lock', which is the main currency in the game. Users can trade items, and become wealthy by gathering resources and acquiring new worlds. There are many items such as vending machines (called a DigiVend) which you can place in worlds, which will allow other users to buy items from inside the vending machine in exchange for world locks.

Figure 1: My application is based on a vending machine business in Growtopia [1] (Game logo above) which tracks the flow of customers buying items from vending machines.

1.2 How will this database be used in the context of Growtopia?

In Growtopia, you can pair with your friends and make a small business together to gain world locks, which is the main currency in the game. I have paired with my friends to sell items on vending machines in Growtopia among our different worlds that we own. We wanted to make an application to keep track of the worlds that we own which have vending machines in them. With this, we will be able to view all the vending machine sales, revenue, and customer information from the vending machines, and we will be able to make queries on that information.

Figure 2: Me in one of my vending machine worlds with 2 vending machines in the picture, one selling fireworks and another selling a door.

2 Entity Relationship Diagram

I have made an entity relationship diagram to represent the entities within the vending machine business in Growtopia, as well as the relationships that they have with each other.

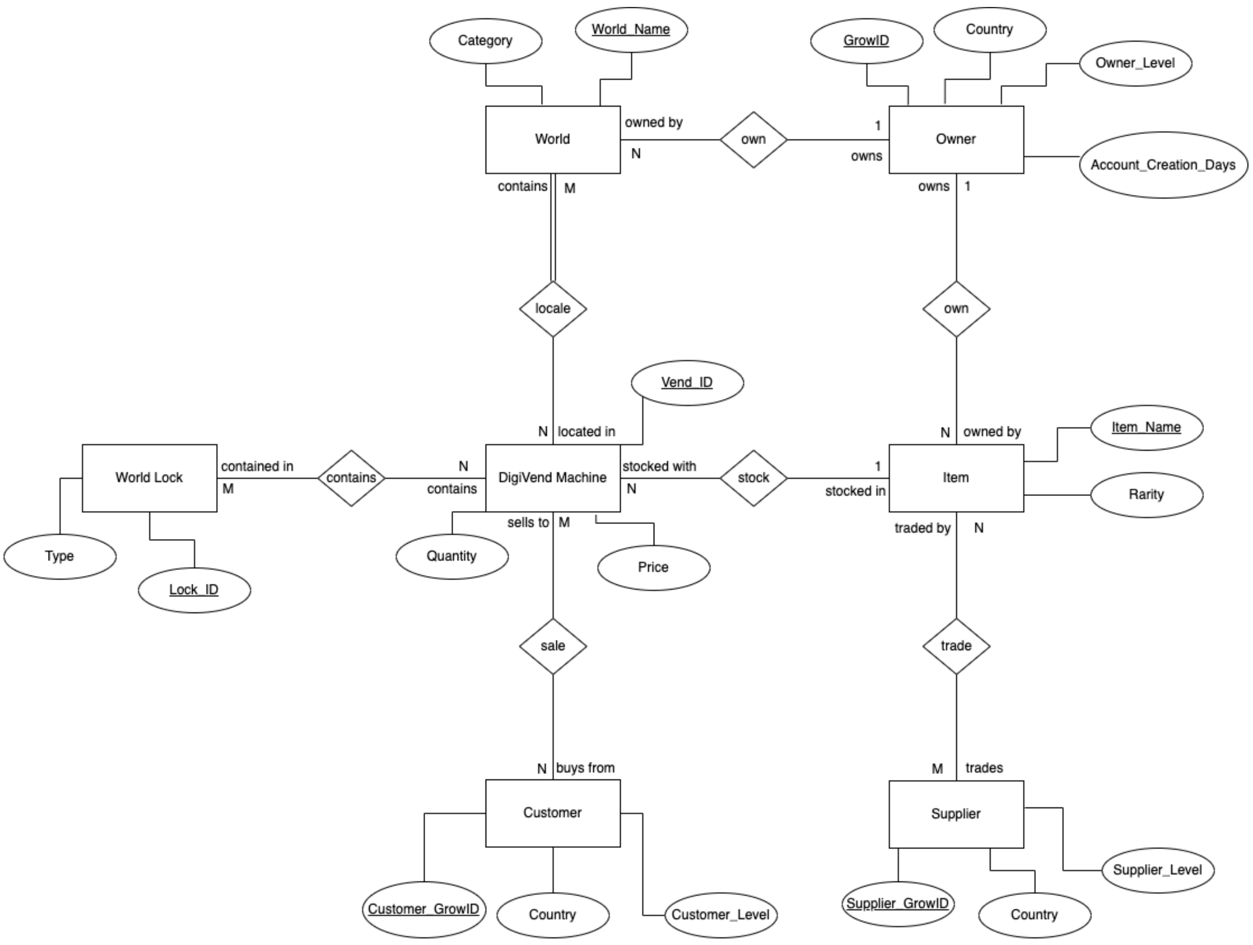

Figure 3: The entity relationship diagram for my application.

You can see above that there are 7 entities - World, Owner, World Lock, DigiVend Machine, Item, Customer, and Supplier. These seven entities allow me to describe the information model that I need for the Growtopian vending machine business. There are a number of assumptions that must be followed so that we follow the rules of the game. There will be a number of owners of the vending machines/worlds, each of which is a friend who is a member of the business. Each owner can own many Growtopian worlds, but each world can only be owned by one owner (1:N relationship). Each owner can own a number of items, but an item can only be owned by one owner (1:N relationship). Each DigiVend Machine is located in many worlds, but each world must have many vending machines (Total participation i.e. existence dependency hence there is a double line unlike the rest where there is partial participation with a single line. There is also a M:N relationship here). Each world must contain many vending machines as otherwise it the world would not fall into the scope of the vending machine business. Each DigiVend Machine can only have one item, but each item can be stored in many DigiVend Machines (1:N relationship). A DigiVend Machine contains many World Locks, but World Locks are contained in many DigiVend Machines (N:M relationship). Each customer buys from many DigiVend Machines, and each DigiVend Machine sells to many customers (M:N relationship). Each item is supplied by many suppliers, and each supplier supplies many items (M:N relationship).

The World entity has attributes such as World_Name (primary key) and category which is an optional tag you can give to worlds. The Owner entity has attributes such as GrowID (primary key), Country, Owner_Level (the level of the user in the game which ranges from 1 to 125), and Account_Creation_Days (the number of days since the user's account was made). The Item entity has attributes such as Item_Name (primary key) and Rarity (value from 1 to 363). The Supplier entity has attributes such as Supplier_GrowID (primary key), Country, and Supplier_Level. The Customer entity has attributes such as Customer_GrowID (primary key), Country, and Customer_Level. The DigiVend Machine entity has attributes such as Vend_ID (primary key), Price (how many World Locks it costs to buy the item inside), and Quantity (how many of the item inside is in stock). The World Lock entity has attributes such as Lock_ID (primary key) and type (there are types of world locks such as normal, emerald, diamond, robotic, and royal).

3 Mapping to Relational Schema

I have also mapped my entity relationship diagram to a relational schema to help me understand the layout of the database for the vending machine business in Growtopia.

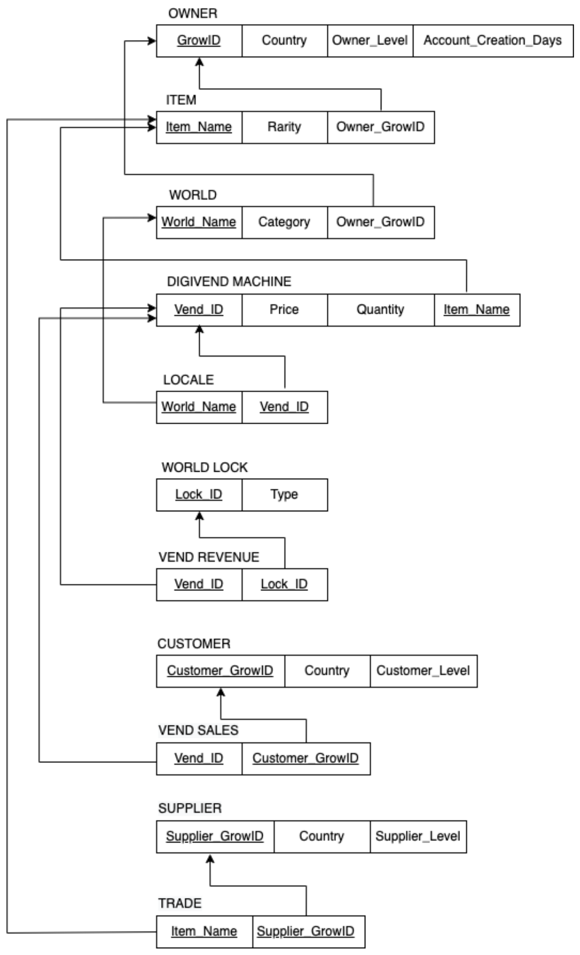

Figure 4: The mapping to a relational schema for my application.

The Owner table has a primary key of ”GrowID”. The Item table has a primary key of ”Item_Name” and a foreign key of ”Owner_GrowID” (there is a 1:N relationship between an Owner and an Item). The World table has a primary key of ”World_Name” and a foreign key of ”Owner_GrowID” (again, there is a 1:N relationship between an Owner and a World). The DigiVend Machine table has a primary key of ”Vend_ID”, and a foreign key of ”Item_Name” (there is a 1:N relationship between an Item and DigiVend Machine). Since there is a M:N relationship between a World and a DigiVend Machine, there is a new table called Locale which has a primary key of both ”World_Name” and ”Vend_ID”. The World Lock table has a primary key of ”Lock_ID”. Since there is a M:N relationship between a World Lock and a DigiVend Machine, there is a new table called Vend Revenue which has a primary key of both ”Vend_ID” and ”Lock_ID”. The Customer table has a primary key of ”Customer_ID”. Since there is a M:N relationship between a Customer and a DigiVend Machine, there is a new table called Vend Sales which has a primary key of both ”Vend_ID” and ”Customer_ID”. The Supplier table has a primary key of ”Supplier_GrowID”. Since there is a M:N relationship between a Supplier and an Item, there is a new table called Trade which has a primary key of both ”Item_Name” and ”Supplier_GrowID”.

4 Functional Dependency Diagram (for proposed relations)

I have made a functional dependency diagram which shows if one attribute determines another. This will help in understanding the logic of Growtopia and therefore developing this MySQL database for the vending machine business.

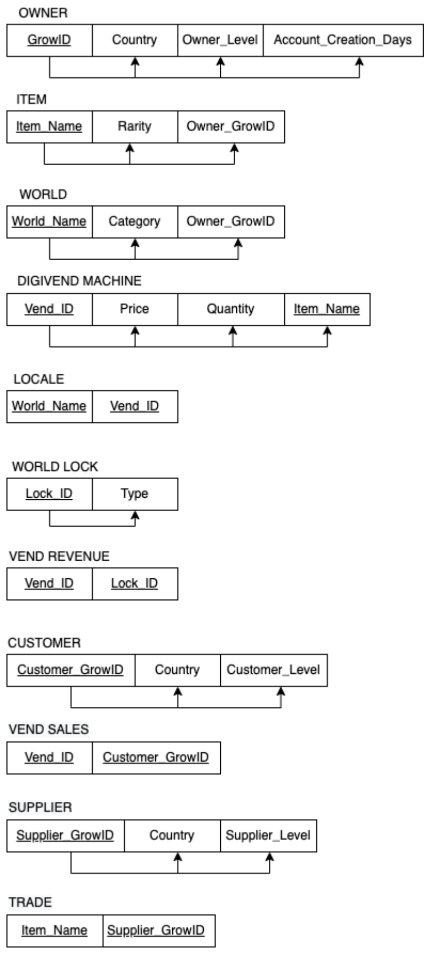

Figure 5: The functional dependency diagram for my application.

As you can see above, the table tuples depend on the primary key. The tables such as Locale, Vend Revenue, Vend Sales, and Trade do not have any arrows. Vend Revenue for example has Vend_ID and Lock_ID as the primary and also the foreign keys and has no arrows.

5 Explanation of one of the SQL code for creating one of your database tables (including any constraints).

```
CREATE TABLE DIGIVEND_MACHINE (
     Vend_ID INT NOT NULL UNIQUE,
     Price INT,
     Quantity INT,
     Item_Name VARCHAR(50),
    PRIMARY KEY (Vend_ID),
    FOREIGN KEY (Item_Name) REFERENCES ITEM (Item_Name) ON UPDATE CASCADE,
    CONSTRAINT CHK_QUANTITY CHECK (Quantity >= 0)
);
```

Listing 1: Creating a table for a DigiVend Machine.

To explain the code above, I have to first talk about the columns taken from the relational schema in section 3. Each vending machine has it's own unique Vend_ID which is an Integer to differentiate it from the other vending machines (hence it's the primary key). A DigiVend Machine can have a price which is an Integer, quantity which is an Integer and Item_Name which is set to VARCHAR(50) which are optional, and can be set as null representing the vending machine being out of stock. The DigiVend Machine's Item_Name acts as a foreign key to the Item entity which is stored inside of it. The DigiVend Machine's quantity of the item stored inside it must be greater than or equal to zero, as we cannot have a negative amount of an item in a vending machine; this acts as a constraint called 'CHK_QUANTITY'. If the quantity is 0, then the vending machine is out of stock and hence the Price and Item_Name should be null. Instead of adding another constraint for this, I have added this requirement with the help of a trigger.

6 Explanation and SQL code for any Altering tables

```
#Alter Owner table to include total hours played by the owner
ALTER TABLE OWNER ADD Total_Hours_Played INT NOT NULL;
#Alter World table to include number of DigiVend Machine's contained in it
ALTER TABLE WORLD ADD Number_Of_DigiVend_Machine INT NOT NULL;
#Alter Customer table to include Customer's Account creation in days.
ALTER TABLE CUSTOMER ADD Customer_Account_Creation_Days INT NOT NULL;
#Alter Supplier table to include Supplier's Account creation in days.
ALTER TABLE SUPPLIER ADD Supplier_Account_Creation_Days INT NOT NULL;
#Alter Trade table to include time of Trade
ALTER TABLE SUPPLIER ADD Time_Of_Trade TIME NOT NULL;
#Alter Item table to include amount of gems it will give if it is recycled
ALTER TABLE ITEM ADD Recycle_Potential INT NOT NULL;
```

Listing 2: Code to alter the tables for possible future expansion

I wanted to design the database with the possibility of expansion in the future. If another group of Growtopians want to use this database, they can use the first statement above to add a column to the Owner table. Every Growtopian can access the total number of hours they have played if they view their profile. For the database, a user can run the code above to include this data for all the owners, defaulting to 0. They could also use the last statement to add a column to the Item table to include the amount of gems it will give if it is recycled. As you can see, there is room for lots of expansion in the database, and these commands act are helpful for anyone who wants to expand this database even further than I want it to be.

7 Explanation and SQL code for any Trigger operations

```
#Define Triggers for Database
DELIMITER //
CREATE TRIGGER UpdateDigiVend BEFORE INSERT ON DIGIVEND_MACHINE
FOR EACH ROW
BEGIN
   IF NEW.QUANTITY = 0 THEN
      SET NEW.PRICE = null;
      SET NEW.Item_Name = null;
   END IF;
END//
 .
RUN;
DELIMITER ;
```

Listing 3: Trigger sets Price and Item_Name to null if Quantity is 0.

I made sure that the Quantity is greater than 0 using constraints when I was creating the table, but now I can ensure that Price and Item_Name are null when the Quantity is 0. This is because if the Quantity is 0, then the vending machine has no stock, and in Growtopia this would mean the DigiVend Machine is unavailable for use. To imitate this functionality, this trigger sees if an insert on the DigiVend Machine table occurs, and if so, it checks the value of Quantity and sets Price and Item_Name to null if it is 0.

8 Explanation and SQL code for any creation of Views

```
#Define Views for Database
CREATE VIEW ALLCUSTOMERS (
    ID,
    AccountCountryOfOrigin,
    LevelOfCustomer)
    AS SELECT Customer_GrowID, Country, Customer_Level
    FROM CUSTOMER;

CREATE VIEW ALLOWNERS (
    ID,
    AccountCountryOfOrigin,
    LevelOfOwner,
    AccountCreationInDays)
    AS SELECT GrowID, Country, Owner_Level, Account_Creation_Days
    FROM OWNER;

CREATE VIEW ALLSALES (
    VENDINGMACHINE_ID,
    CustomerID)
    AS SELECT Vend_ID, Customer_GrowID
    FROM VEND_SALES;

CREATE VIEW ALLVENDINGMACHINES (
    VENDINGMACHINE_ID,
    Price,
    Quantity,
    Item_Name)
    AS SELECT Vend_ID, Price,Quantity,Item_Name
    FROM DIGIVEND_MACHINE;

CREATE VIEW ALLWORLDS (
    Name,
    Category,
    ID)
    AS SELECT World_Name, Category, Owner_GrowID
    FROM WORLD;
```

Listing 4: Creating five views using SQL.

I made five different views which serve for different purposes. The first three are for users with the role of ”OWNER”, as they can see all the customers, all the owners, and all the sales that has happened. The last two views are for users with the role of ”CUSTOMER”, as they can see all the vending machines, and worlds which have vending machines in them. I will talk about the security aspect of the roles later on. The creation of these views using the ”CREATE VIEW” command allows helpful views to be made to be used by particular users of the database.

9 Explanation and SQL Code for one of your commands to Populate a Tables

```
INSERT INTO WORLD (World_Name, Category, Owner_GrowID)
  VALUES
     ('ZashVend', 'Shop', 'iZashGT'),
     ('JomonsStory', 'Story', 'JohnsJomon'),
     ('EligijusParkour', 'Parkour', 'EligijusSkersonas'),
     ('CriticalTrade', 'Trade', 'CriticalAnalyzer2'),
     ('NoTrollDating', 'Love', 'NoTrollGaming');
```

Listing 5: Populating the WORLD table.

A table is populated using the "INSERT" command. When populating the World table above, we must take note of the fact that there are some restrictions on the values that we are trying to insert. For example, the World_Name has to be between 1 and 24 characters long in the game. Also, any Owner_GrowID must already exist in the OWNER table's GrowID column, otherwise you will get an error. Designing the database in this way makes it effective, and gets rid of any accidental errors.

10 Explanation and example SQL Code for retrieving information from the database (including any use of Joins and use of functions).

```
#Retrieve all the customers who have bought something from the vending
machine business along with which vending machine they bought from
SELECT VEND_SALES.Vend_ID, VEND_SALES.Customer_GrowID, CUSTOMER.Country,
CUSTOMER.Customer_Level
FROM VEND_SALES
INNER JOIN CUSTOMER
ON VEND_SALES.Customer_GrowID = CUSTOMER.Customer_GrowID;
```

Listing 6: Retreiving information from the database.

We could have retrieved information separately from 2 tables. One with the customers who have bought something from one of the worlds with vending machines, and another with the sales of each of the vending machines. However, we have used "INNER JOIN" to combine them and get one table as a result which retrieves information from the database and results in a simple table that shows all the customers and which vending machine they have bought from (using their Vend_ID).

11 Explanation and SQL Code for any Security commands (roles & permissions).

```
#New roles
CREATE ROLE GROWTOPIANOWNER;
CREATE ROLE CUSTOMER;

#Set permissions for Owners
GRANT ALL ON OWNER TO GROWTOPIANOWNER;
GRANT ALL ON ITEM TO GROWTOPIANOWNER;
GRANT ALL ON WORLD TO GROWTOPIANOWNER;
GRANT ALL ON DIGIVEND_MACHINE TO GROWTOPIANOWNER;
GRANT ALL ON LOCALE TO GROWTOPIANOWNER;
GRANT ALL ON WORLD_LOCK TO GROWTOPIANOWNER;
GRANT ALL ON VEND_REVENUE TO GROWTOPIANOWNER;
GRANT ALL ON CUSTOMER TO GROWTOPIANOWNER;
GRANT ALL ON VEND_SALES TO GROWTOPIANOWNER;
GRANT ALL ON SUPPLIER TO GROWTOPIANOWNER;
GRANT ALL ON TRADE TO GROWTOPIANOWNER;
GRANT ALL ON ALLCUSTOMERS TO GROWTOPIANOWNER;
GRANT ALL ON ALLOWNERS TO GROWTOPIANOWNER;
GRANT ALL ON ALLSALES TO GROWTOPIANOWNER;


#Set permissions for Customers
GRANT SELECT ON ALLWORLDS TO CUSTOMER;
GRANT SELECT ON ALLVENDINGMACHINES TO CUSTOMER;
```

Listing 7: Creating roles and setting permissions .

There are two roles that are created, one for the Owner, and another for the Customer. The Growtopian Owner role has full permission to all the tables including the views in the database. The customer role only has read permissions on the views for seeing all the vending machines and all the worlds. By thinking about database security in this way (using roles and permissions), you can ensure database security using security protocols and access control.

12 Explanation of any additional SQL features of your choice

```
#Below command shows all triggers
show triggers;
#Below command returns the command to remove a trigger, copy it and run it
to remove a trigger
SELECT Concat('DROP TRIGGER ', Trigger_Name, ';') FROM
information_schema.TRIGGERS WHERE TRIGGER_SCHEMA = 'GrowtopiaVendingStore';
```

Listing 8: Additional commands to run.

Using the first command above, you can show all the existing triggers in the database (i.e. UpdateDigiVend in this database). Using the second command, it will return a command you can copy to delete the trigger in the database. For example, it will return "DROP TRIGGER UpdateDigiVend;" which will delete the trigger in the database called "GrowtopiaVendingStore".

13 References

1. Ubisoft. 2021. *Growtopia Home Page* [Online]. Available from:
https://www.growtopiagame.com/ [accessed 1st February 2022].

Section C is a sql file called "Growtopia-Vending-Machine-Business-CSU34041.sql" inside the "Section-C" folder. I have also added another folder called "Separated-SQL-Code-For-Database" which contains a number of sql files which can be used as well.