Final Project Report

Stefon Miller

[SMM248@pitt.edu](mailto:SMM248@pitt.edu)

INFOSCI 1022: Database Management Systems

Professor Babichenko

4/9/2020

Table of Contents

3……………………………………………………………………………………. Introduction/Abstract

4…………………………………………………………………………….. UML-compliant E-R Model

5 – 6….…………………………………………………………………………………... Business Rules

7 – 9………………………………………………………………………. Entity/Attribute Descriptions

10 – 12…………………………………………………………………………………. List of Questions

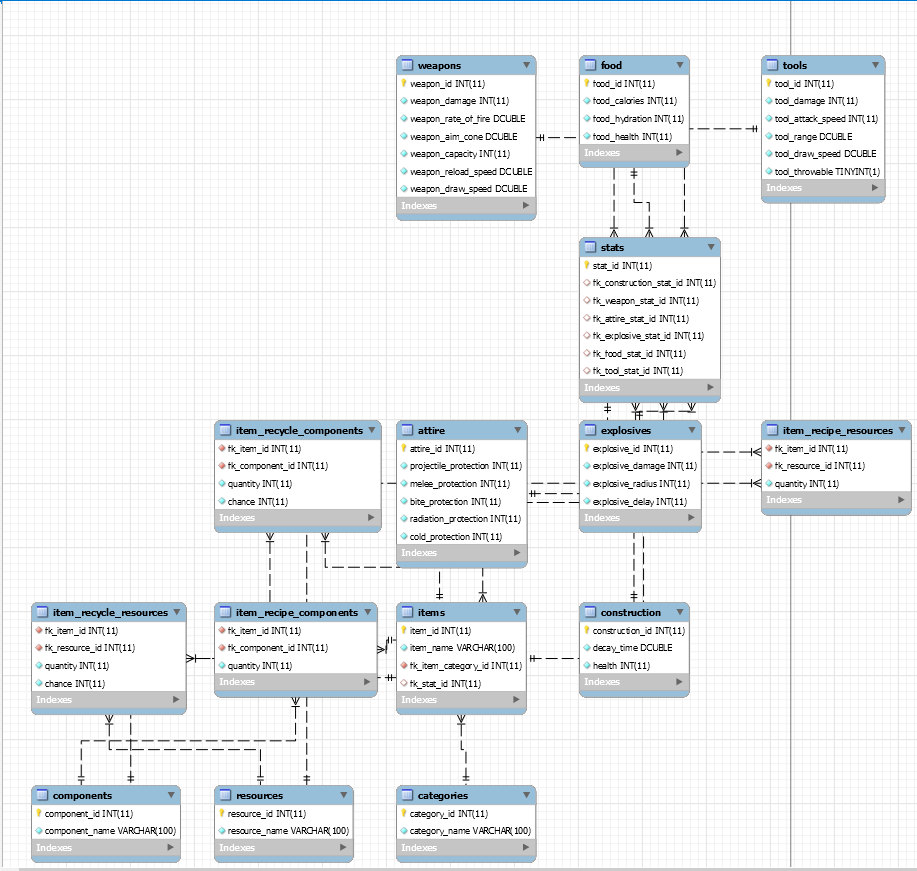
13……………………………………………………………………………………………… Conclusion

Introduction/Abstract

The project I have developed is a database containing item information for an online video game called Rust. Rust is a survival sandbox game in which you construct tools and weapons to fight other players. Items in Rust are constructed using resources and components. Resources can be harvested from the environment(wood from trees, stone from rocks, etc.) while components are found in barrels and boxes that can be looted by the player. Additionally, certain areas on the map have recyclers that players can use to break down their items into raw materials. Recycling an item will return a set amount of resources and give a chance of returning components. Items can also have stats depending on the item type, for example weapons can have damage statistics while armor can have protection values. These values are also incorporated into the database.

There already exists a website detailing all the crafting costs/recycling yields of items in Rust, along with other information. Before starting this project, I was looking to utilize this website to create a Discord bot that would return information about item as it is tedious to navigate to that website and lookup an item whenever I wanted information on it. I attempted to use web scraping to retrieve data from the site, as it does not provide an API and has no external database. However, the website has almost no discernable pattern to its layout that would allow for efficient web scraping and it appears as though each item and its corresponding information is hard coded into the HTML of the page. After dozens of hours attempting to find a pattern to return information reliably to varying degrees of success, I decided I would use this project as an opportunity to enter all the items in Rust into a database. This would allow for more structured data that is more receptive to retrieval and manipulation. This database and by extension the Discord bot would be useful to anybody who uses Discord and would like to have data on Rust items without using an external program/website.

UML-Compliant E-R Model



Business Rules

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity 1 | Entity 2 | Cardinality on Entity 1 Side | Cardinality on Entity 2 side | Business Rule(s) |
| categories | items | 1 | 1..\* | An item has exactly 1 category, but a category can apply to many items |
| items | Item\_recycle\_components | 1 | 0..\* | An item can yield 0 to multiple components types when recycled |
| items | Item\_recycle\_resources | 1 | 0..\* | An item can yield 0 to multiple resource types when recycled |
| items | Item\_recipe\_components | 1 | 0..\* | An item recipe can contain 0 to multiple component types |
| items | Item\_recipe\_resources | 1 | 0..\* | An item recipe can contain 0 to multiple resource types |
| components | Item\_recycle\_components | 1 | 0..\* | A component can recycled from multiple items |
| components | Item\_recipe\_components | 1 | 0..\* | A resource can be recycled from multiple items |
| resources | Item\_recycle\_resources | 1 | 0..\* | A resource can be required for multiple items |
| resources | Item\_recipe\_resources | 1 | 0..\* | A component can be required for multiple resources |
| items | stats | 1..\* | 0,1 | An item can have at most 1 stat description, but a stat description can describe multiple items |
| construction | stats | 0,1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |
| weapons | stats | 0,1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |
| attire | stats | 0,1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |
| food | stats | 0,1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |
| tools | stats | 0,1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |
| explosives | stats | 1 | 1..\* | Each stat entry has exactly 1 type(Weapon, food, tool, etc.), but a stat type can correspond to more than 1 entry |

Entity / Attribute Descriptions

|  |  |  |
| --- | --- | --- |
| Weapons | | |
| weapon\_id(pk) | INT | ID of the particular weapon stat combination |
| weapon\_damage | INT | Base damage of the weapon |
| weapon\_rate\_of\_fire | DOUBLE | Rate of fire of the weapon in rounds per minute |
| weapon\_aim\_cone | DOUBLE | Weapon’s aim cone in degrees |
| weapon\_capacity | INT | How many rounds the weapon holds in a magazine |
| weapon\_reload\_speed | DOULBE | How long it takes to reload the weapon in seconds |
| weapon\_draw\_speed | DOUBLE | How long it takes to pull the weapon out |

|  |  |  |
| --- | --- | --- |
| Food | | |
| food\_id(pk) | INT | ID of the particular food stat combination |
| food\_calories | INT | How many calories the food restores |
| food\_hydration | INT | How much hydration the food provides |
| food\_health | INT | How much health regeneration the food grants |

|  |  |  |
| --- | --- | --- |
| Tools | | |
| tool\_id(pk) | INT | ID of the particular tool stat combination |
| tool \_damage | INT | Base damage of the tool |
| tool \_attack\_speed | INT | How many times you can swing the tool in a minute |
| tool \_range | DOUBLE | Attack range of the tool |
| tool \_draw\_speed | DOUBLE | How long it takes to pull the tool out |
| tool \_throwable | TINYINT(1) | Can you throw the tool |

|  |  |  |
| --- | --- | --- |
| Stats | | |
| stat\_id(pk) | INT | Id of the particular stat |
| fk\_construction\_stat\_id(fk) | INT | Construction stat of item |
| fk\_weapon\_stat\_id(fk) | INT | Weapon stat of item |
| fk\_attire\_stat\_id(fk) | INT | Attire stat of item |
| fk\_explosive\_stat\_id(fk) | INT | Explosive stat of item |
| fk\_food\_stat\_id(fk) | INT | Food stat of item |
| fk\_tool\_stat\_id(fk) | INT | Tool stat of item |
| Explosives | | |
| explosive\_id(pk) | INT | Id of the particular explosive stat combination |
| explosive\_damage | INT | Base damage of the explosive |
| explosive\_radius | INT | Radius of the explosion in meters |
| explosive\_delay | INT | Delay between deploying the explosive and it going off in seconds |

|  |  |  |
| --- | --- | --- |
| Attire | | |
| attire\_id(pk) | INT | ID of the particular armor stat combination |
| projectile\_protection | INT | Percent of damage from projectiles the armor absorbs |
| melee\_protection | INT | Percent of damage from melee the armor absorbs |
| bite\_protection | INT | Percent of damage form animals the armor absorbs |
| radiation\_protection | INT | Percent of radiation the armor prevents |
| cold\_protection | INT | Percent of cold damage the armor prevents |

|  |  |  |
| --- | --- | --- |
| Construction | | |
| construction\_id(pk) | INT | Id of the particular construction stat combination |
| decay\_time | INT | How many hours it takes the item to decay |
| health | INT | How many hit points the item has |

|  |  |  |
| --- | --- | --- |
| Categories | | |
| category\_id(pk) | INT | Id of the category |
| category\_name | VARCHAR(100) | Name of the category |

|  |  |  |
| --- | --- | --- |
| Resources | | |
| resource\_id(pk) | INT | Id of the resource |
| resource\_name | VARCHAR(100) | Name of the resource |

|  |  |  |
| --- | --- | --- |
| Components | | |
| component\_id(pk) | INT | Id of the component |
| component\_name | VARCHAR(100) | Name of the component |

|  |  |  |
| --- | --- | --- |
| Items | | |
| item\_id(pk) | INT | ID of the item |
| item\_name | VARCHAR(100) | Name of the item |
| fk\_item\_category\_id(fk) | INT | ID of the item’s category |
| fk\_stat\_id(fk) | INT | ID of the item’s stats |

|  |  |  |
| --- | --- | --- |
| Item\_recycle\_components | | |
| fk\_item\_id | INT | ID of the item |
| fk\_component\_id | INT | ID of the component yielded from recycling |
| quantity | INT | Quantity of component yielded from recycling |
| chance | INT | Percent chance of yielding the component |

|  |  |  |
| --- | --- | --- |
| Item\_recycle\_resources | | |
| fk\_item\_id | INT | ID of the item |
| fk\_resource\_id | INT | ID of the resource yielded from recycling |
| quantity | INT | Quantity of resource yielded from recycling |
| chance | INT | Percent chance of yielding the resource |

|  |  |  |
| --- | --- | --- |
| Item\_recipe\_components | | |
| fk\_item\_id | INT | ID of the item |
| fk\_component\_id | INT | ID of the component needed for crafting the item |
| quantity | INT | Quantity of the component needed |

|  |  |  |
| --- | --- | --- |
| Item\_recipe\_resources | | |
| fk\_item\_id | INT | ID of the item |
| fk\_resource\_id | INT | ID of the resource needed for crafting the item |
| quantity | INT | Quantity of the resource needed |

List of questions

1. What components are currently in the game?
   1. SELECT \* FROM components;
   2. This query is useful if you are trying to get a list of all components to organize the items in your base
2. What resources do I need to craft a hoodie?
   1. SELECT item\_name, resource\_name, quantity FROM items

INNER JOIN item\_recipe\_resources

ON item\_id = fk\_item\_id

INNER JOIN resources

ON resource\_id = fk\_resource\_id

WHERE item\_name = 'Hoodie';

* 1. This query is useful for determining what resources you need to craft a certain item

1. Show me all weapons in the game
   1. SELECT \* FROM weapons

LEFT JOIN stats

ON fk\_weapon\_stat\_id = weapon\_id;

* 1. This query can be used to filter all items in the database to just those with a certain stat category. This can be used to display all weapons, clothing, tools, etc.

4a. How many metal fragments do I need to make a semi-automatic rifle and a pair of boots?

1. SELECT SUM(quantity)

FROM item\_recipe\_resources

WHERE fk\_item\_id IN (9, 11)

AND fk\_resource\_id = 3;

1. This query is essentially a crafting calculator and is one of the most useful features of the database. It can be used to determine what/how many materials you need to craft a certain item(s). Additionally, in this case it can be used to tell how much of a specific resource you need to craft a certain item(s)

4b. How many items require metal fragments to craft?

* 1. SELECT COUNT(fk\_item\_id)

FROM item\_recipe\_resources

WHERE fk\_resource\_id = 10;

* 1. This query can be used to determine how many items require a certain resource/component to craft. This can be used to determine what resources/components are most important to stock up on

1. How many of each item are in the database?
   1. SELECT COUNT(item\_id), category\_name

FROM items JOIN categories

ON fk\_item\_category\_id = category\_id

GROUP BY fk\_item\_category\_id;

* 1. This query is useful for determining the distribution of items in the game, and whether the game needs more tools, clothes, etc.

1. Show me all armor with over 5% cold protection
   1. SELECT item\_name, cold\_protection

FROM items JOIN stats

ON fk\_stat\_id = stat\_id

JOIN attire

ON fk\_attire\_stat\_id = attire\_id

GROUP BY cold\_protection

HAVING cold\_protection > 5;

* 1. In Rust, if your armor’s cold protection is too low you will take damage in cold environments. This query is useful for determining which armor you should take into said environments

1. Show me all pieces of armor, with the most protective pieces at the top
   1. SELECT item\_name, projectile\_protection

FROM items JOIN stats

ON fk\_stat\_id = stat\_id

JOIN attire

ON fk\_attire\_stat\_id = attire\_id

ORDER BY projectile\_protection DESC;

* 1. This query is useful for when picking a loadout of clothes pieces to wear.

1. Which weapon does the most damage?
   1. SELECT item\_name, weapon\_damage

FROM items JOIN stats

ON fk\_stat\_id = stat\_id

JOIN weapons

ON fk\_weapon\_stat\_id = weapon\_id

ORDER BY weapon\_damage DESC

LIMIT 1;

* 1. This query is useful for determining which weapon(s) are worth crafting

1. Which items can I recycle for high quality metal?
   1. SELECT item\_name FROM item\_recycle\_resources

JOIN items

ON fk\_item\_id = item\_id

WHERE fk\_resource\_id IN

(SELECT resource\_id FROM resources WHERE resource\_name = 'High Quality Metal');

* 1. This query is extremely useful for determining what items to recycle if you need a specific resource.

1. What is the total crafting cost of a semi-automatic rifle?
   1. SELECT item\_name, component\_name, irc.quantity, resource\_name, irr.quantity

FROM items JOIN item\_recipe\_components AS irc

ON item\_id = irc.fk\_item\_id

JOIN components

ON irc.fk\_component\_id = component\_id

JOIN item\_recipe\_resources AS irr

ON item\_id = irr.fk\_item\_id

JOIN resources

ON irr.fk\_resource\_id = resource\_id

WHERE item\_id = 9;

* 1. This query is a full crafting calculator for any item in the game

Conclusion

In conclusion, this project was a great opportunity for me to gain insight on exactly how I want to design the database for my Discord bot. The hardest part for me was setting up the relationship between the items table and the corresponding stat tables. Because a weapon, tool, etc is an item it was hard to figure out how to relate the tables. In the future I will change this relationship to more of an inheritance one but for now I am satisfied with how it works. I really like the way I created the relationships for crafting and recycling items, although it uses a lot of joins. I think that at the very least learning how to set up recipes/recycling yields for the items will greatly help in developing a full database for my Discord bot in the future.