Technical Report Project Databases

Hendrik De Bruyn
Marouan El Marnissi
Elias El Bouzidi
Yassine El Abdellati
Sinem Erdön
Samuel van Nimwegen

2023-2024

1 Introduction

This is the technical report of group 5. In this all the features of our game will be discussed together with a explanation of our database diagram.

2 Farms and Mines

If the player is in a settlement, he can buy and place farms and mines. Once the farm or mine has been placed it has a building time, after that time the building becomes available. Once it is available and you click on the building, you can see the stats: production per farm or mine and the max storage it can hold. You can see a start farm or mine button if you click on that it starts the farm or mine. This will take some time, after it is done it will add the made resource in the storage. Once there is available resource to collect, the collect button becomes active so you can click it to collect it. You can also like any other building upgrade it, this will increase the production and storage. You can also delete the building, deleting the building does not give your resources back.

3 Map

When the player is on the main menu of their planet, they can press the button with the sword icon to go to the attack map. When looking at the attack map, the player will be able to zoom the map and move around to scout for players to attack. Every planet that is owned by a player will be displayed on the map. When the player clicks on an opponent's planet, it will show the opponent's name, what race they belong to, the amount of building materials and food they have stored in the settlements and the combat score of the planet. They will then have the option to press a button in the bottom right to send their troop to attack that planet. After a certain amount of time has passed based on the distance between the planet where the player is attacking from and the opponent's planet, the player will be able to start the combat.

4 Combat

After the player has sent it's troops to attack a planet and the troops have arrived, he will be able to commence combat. Every round, the player's and opponent's troops will fight and one of the troops, friendly or enemy, will die. Whether one of the player's troops dies or that of the enemy is completely random since the game decision making is completely RNG based. The players will however be able to manipulate the RNG by training certain types of troops that have special effect in combat that are listed when training them in the barracks. This allows the player to have an edge by spending more resources on stronger troops. Every turn the player will have the option to choose which type of troop he wants to send to fight in that turn. The opponent will always choose the strongest troop they have available. Any of the opponent's troops that die during combat will be revived afterwards, the player's troops however do not get revived after dying meaning they have to train a new set of troops before being able to attack another player. Battle ends when either player runs out of troops to send into battle or if the attacking player chooses to stop attacking, perhaps to save troops if they think they're going to lose. There will also be an auto-play option for the player that makes it so the turns play themselves out. When in auto-play mode, the strongest available unit will be chosen to send into battle.

5 Races / Achievements

The player can create and join races. They can also earn achievements.
In order to browse races and achievements, the player can simply tap on the profile icon. In this menu, the first menu item allows the player to create game options such as "Create Alliance" and "Join Alliance". These features allow the player to create their own races or join races created by the other players.
Furthermore, players participating in the same race are able to communicate with each other via the chat box.
As players progress through the game, they have the opportunity to earn achievements which enables them to collect resources. These achievements are accessible through the second menu item on the profile menu.

6 Database diagram explanation

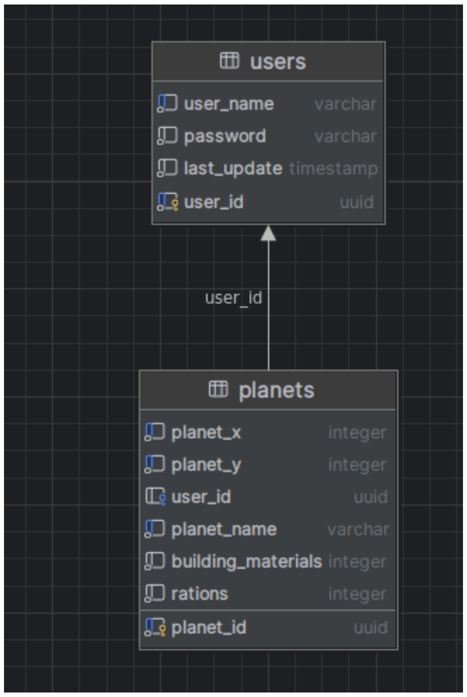
Here the complete database diagram is explained that is currently implemented.

6.1 Users

Every user has a user-name, password and user-id. The user-name always unique. This is so there can never be multiple users with the same username. The user-id is a UUID of python type. This is a number consisting of 32 Hexadecimal numbers. This is so the chance of it being duplicate are practically 0. For reference, if you would generate a billion UUIDs per second for 100 years you would have around a 50% chance of getting a duplicate. There also is a last-update time. This is the last timestamp all the user's items have been updated.

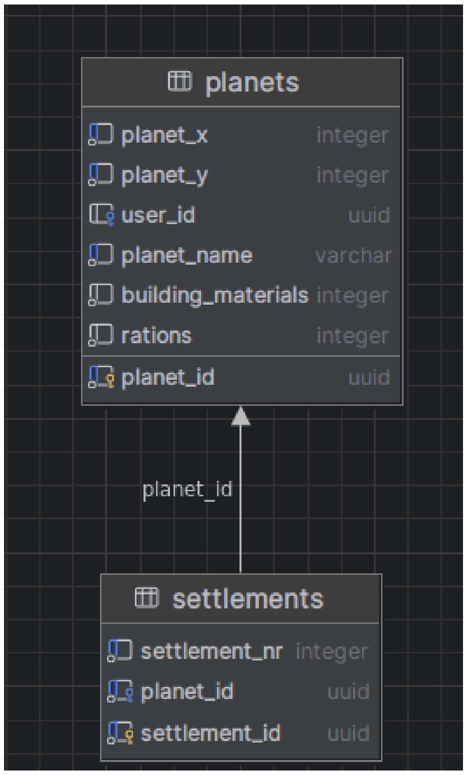
6.2 Planets

A User can have multiple planets. All these planets have unique coordinates in the universe. It also has a user-id if the planet is owned by a user. This means it can only have a single user as owner. The planet also has a unique planet name. The planet also has building materials and rations which are saved in the database. The primary key of a planet is again a UUID which is the planet-id.



6.3 Settlements

A planet can have up to 4 settlements. These are places where you can start building. Every settlement has a number (in order of unlocking) and a planet-id. The primary key is a settlement-id which is, just like all the other id's in this database, a UUID.



6.4 Buildings

A settlement can have multiple buildings. The settlement has a grid with spaces where buildings can be. This is why every building has a settlement-id and a grid-position. A building also has a level. The construction-time-left is a variable that is stored in order to see whether a building is under construction. If the value is not 0, it means the building is under construction. The type column is used for SQLAlchemy polymorphism. It contains a string with the name of its subclass, with this SQLAlchemy knows exactly which columns to put together to get the full requested building.

6.4.1 Farms

Farms make food (rations) at a consistent rate. It can store resources until these are collected. It also stores the time it has left to gather resources, since in this game, it can only collect resources for a certain time-slot before stopping. It also has a building-id since it is a building which is its primary key.

6.4.2 Mines

These work exactly the same as farms and have the same columns except that they make building materials instead of rations.

6.4.3 Town-Halls

This structure has no extra items except the building-id primary key. This is because the only thing that is important is its level. With this level you can later unlock new buildings.

6.4.4 Barracks

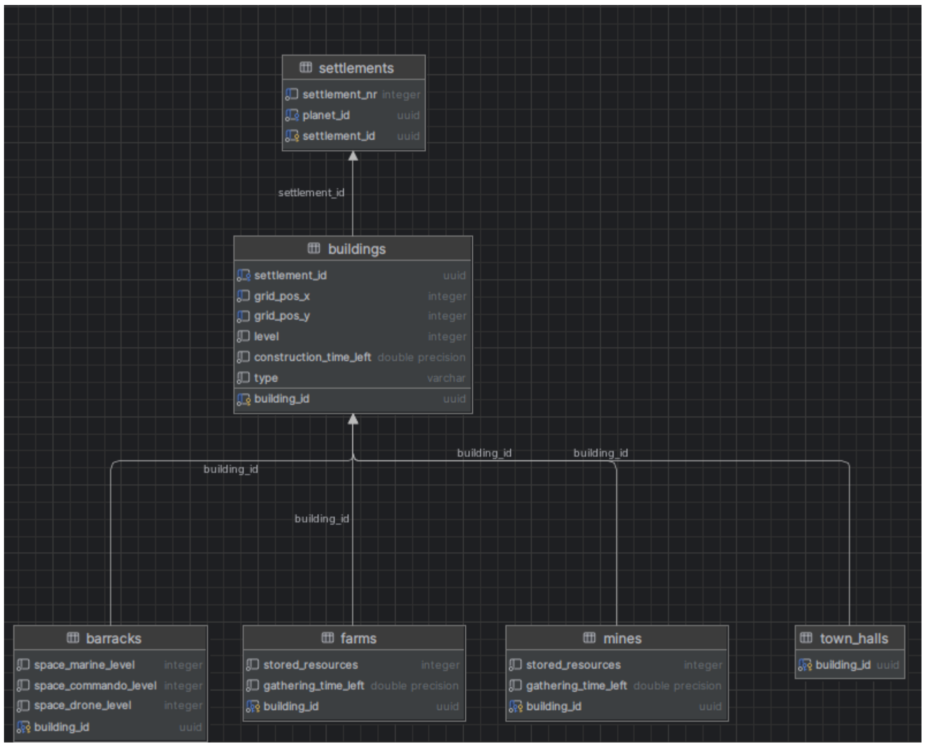
This building is used to train and keep units. It keeps a level for each unit in the database, this is the level at which these units are trained. It also has a building-id just like all the other buildings which is its primary key. More about units in the Attack-Unit section.

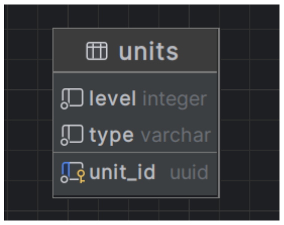
6.5 Units

Units have a level, a type and a unit-id which again is a UUID. There are right now only 1 type of unit, but there probably will come more later.

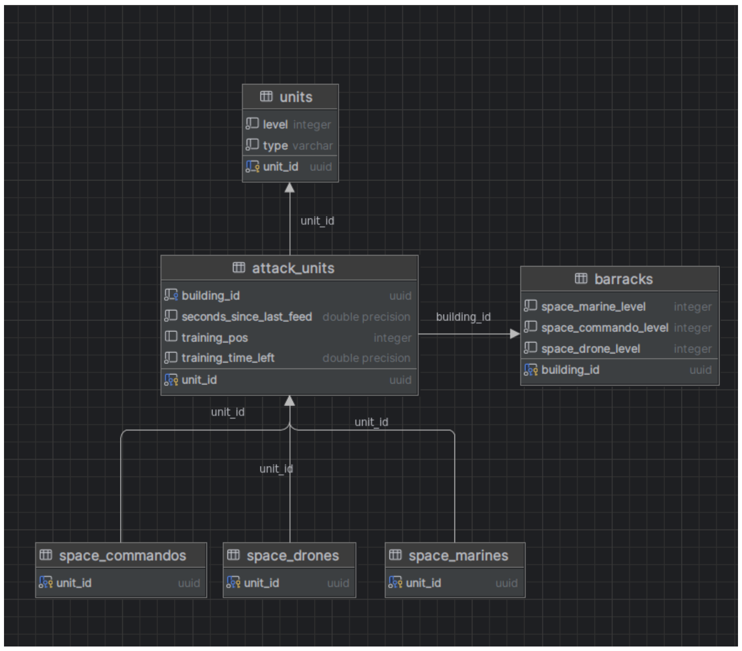
6.6 Attack-Units

Attack-units constitute the sole type of units in the system at present. These units necessitate storage within a Barrack building, hence each possesses a unique building-id key. Furthermore, maintenance requires regular feeding, indicated by the seconds-since-last-feed column, representing the elapsed time since their last feeding session. Feeding intervals occur hourly to sustain their functionality. The training-position attribute denotes the unit's priority within the training queue: a value of 1 signifies active training, while higher values denote a waiting state. Conversely, a value of 0 or None implies absence from the training queue. Additionally, a column tracks the remaining time, in seconds, required for a unit's training completion; this countdown initiates upon assignment of a priority of 1. Each unit is uniquely identified by a unit-id, represented as a UUID.



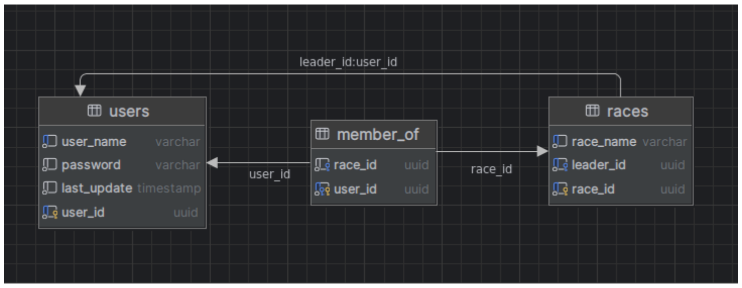


There exist three distinct unit types: Space Marines, Space Commandos, and Space Drones. While each type exhibits differing statistics, these characteristics are encapsulated within their respective class definitions and are not stored in the database, as they remain consistent across all instances of a particular unit type.



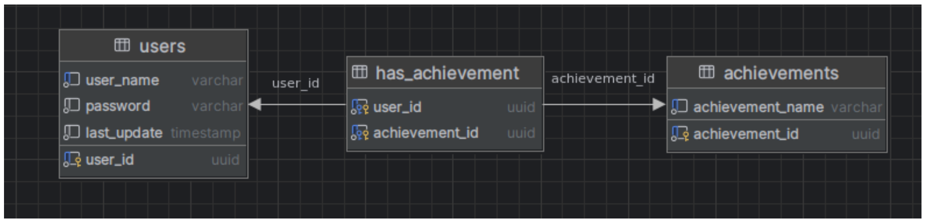
6.7 Races

A user can be part of a race. This race is a group of users, therefore a user can be part of a race and a race has multiple users. A user also has a leader-id which can never be null because there always has to be a leader.



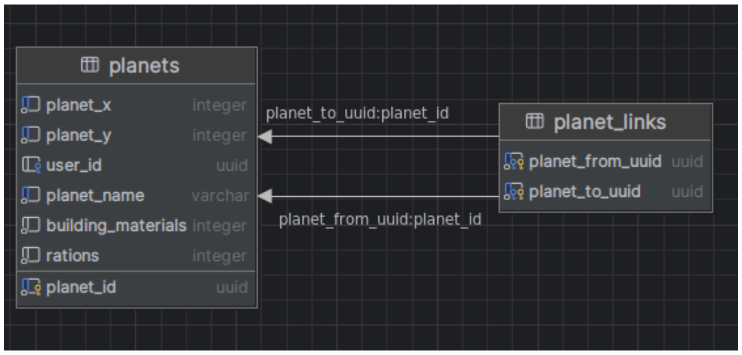
6.8 Achievements

A user can have multiple achievements, but multiple achievements can also have the same user. This is a many-to-many relationship and therefore there is a has-achievement table. Each achievement has a name and an id. The primary key of the has-achievement table is a combination of the 2 id's of the user and the achievement.



6.9 Planet-Links

Multiple planets can be linked. This happens by the planet-links relationship that has a primary key of 2 planet-ids.


planets
planet_x integer
planet_y integer
user_id uuid
planet_name varchar
building_materials integer
rations integer
planet_id uuid
planet_to_uuid:planet_id
planet_from_uuid:planet_id
planet_links
planet_from_uuid uuid
planet_to_uuid uuid