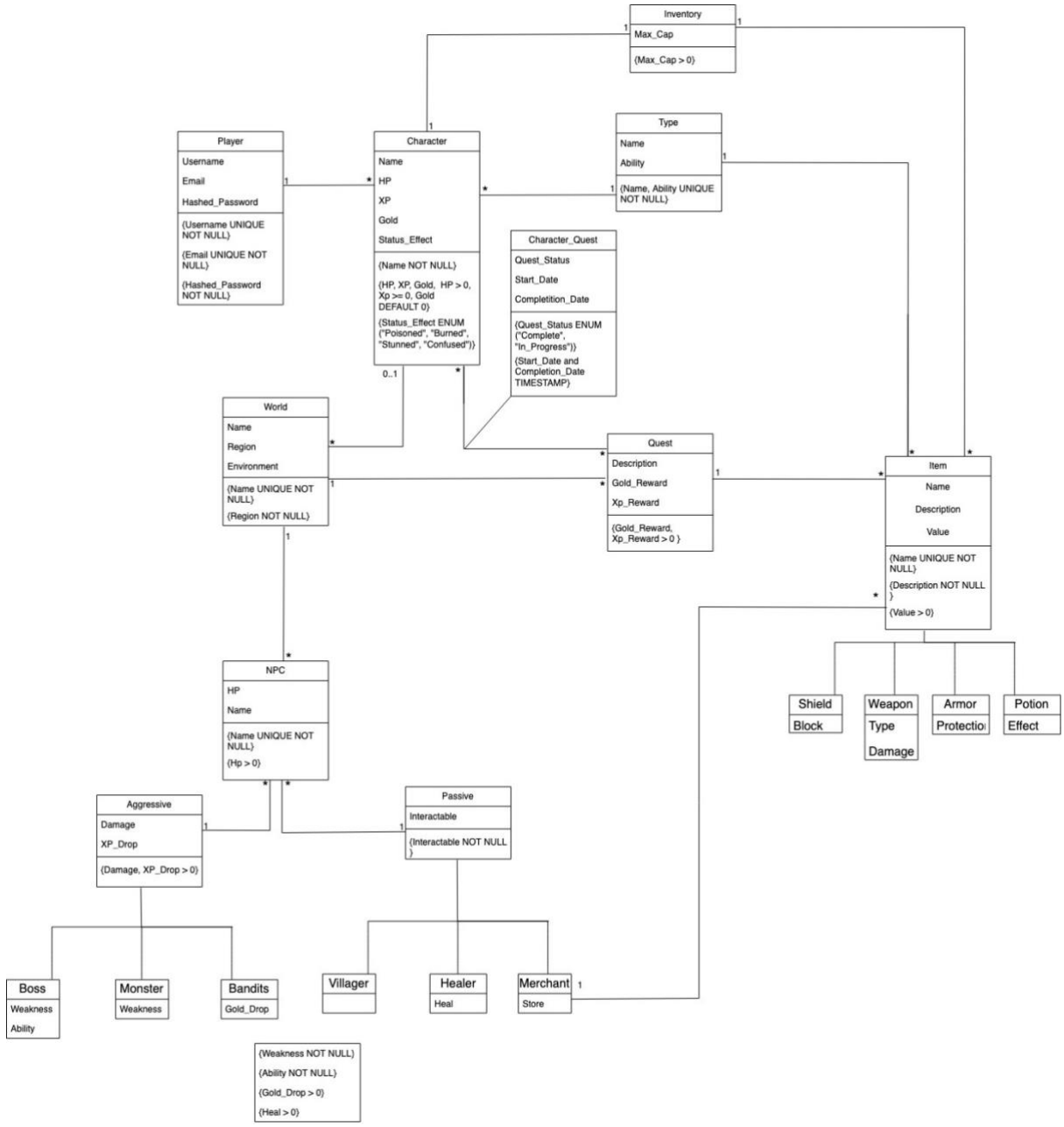
Bases de Dados - LEIC 23/24

Project – Submission 2

A. Refine Conceptual Model

Using the feedback received after the first Project Submission, we altered some things about our conceptual model. Firstly, we added the appropriate constraints to the attributes that needed them, we also altered the “Slots” attribute, creating a new attribute called max_capacity, that defines the number of items an individual inventory can have, the final result we arrived at is as follows:



B.2-4 Define the relational schema

Utilizing our refined Conceptual Model, we defined our relational schema as follows:

Player (IdPlayer, Username, Email, Hashed_Password)

Character (IdCharacter, Name, HP, XP, Gold, Status_Effect, IdPlayer->Player, IdType->Type, IdInventory->Inventory)

Inventory (IdInventory, Max_Cap)

Character_Quest (IdCharacter->Character, IdQuest->Quest, Quest_Status, Start_Date, Completition_Date)

Type (IdType, Name, Ability)

Quest (IdQuest, Description, Gold_Reward, Xp_Reward, IdWorld -> world)

Item (IdItem, Name, Description, Value, IdInventory->Inventory, IdQuest->Quest, IdType->Type)

Shield (IdItem->Item, Block)

Weapon (IdItem->Item, Type, Damage)

Armor (IdItem->Item, Protection)

Potion (IdItem->Item, Effect)

World (IdWorld, Name, Region, Environment, IdCharacter->Character)

NPC (IdNpc, Hp, Name, IdWorld->World, IdAggressive->Aggressive, IdPassive->Passive)

Aggressive (IdAggressive, Damage, Xp_Drop)

Boss (IdAggressive->Aggressive, Weakness, Ability)

Monster (IdAggressive->Aggressive, Weakness)

Bandits (IdAggressive->Aggressive, Gold_Drop)

Passive (IdPassive, Interactable)

Villager (IdPassive-> Passive)

Healer (IdPassive-> Passive, Heal)

Merchant (IdMerchant->Passive, Store)

MerchantItem (IdItem->Item, IdMerchant->Merchant)

After AI implementation, asking ChatGPT to help us improve our relational model, we arrived at our final model as follows:

Player (IdPlayer, Username, Email, Hashed_Password)

Character (IdCharacter, Name, HP, XP, Gold, Status_Effect, IdPlayer->Player, IdType->Type, IdInventory->Inventory)

Inventory (IdInventory, Max_Cap)

Character_Quest (IdCharacter->Character, IdQuest->Quest, Quest_Status, Start_Date, Completition_Date)

Type (IdType, Name, Ability)

Quest (IdQuest, Description, Gold_Reward, Xp_Reward, IdWorld -> world)

Item (IdItem, Name, Description, Value, IdInventory->Inventory, IdQuest->Quest, IdType->Type)

Shield (IdItem->Item, Block)

Weapon (IdItem->Item, Type, Damage)

Armor (IdItem->Item, Protection)

Potion (IdItem->Item, Effect)

World (IdWorld, Name, Region, Environment, IdCharacter->Character)

NPC (IdNpc, Hp, Name, IdWorld->World, IdAgressive->Aggressive, IdPassive->Passive)

Aggressive (IdAgressive, Damage, Xp_Drop)

Boss (IdBoss, IdAggressive ->Aggressive, Weakness, Ability)

Monster (IdMonster, IdAggressive ->Aggressive, Weakness)

Bandits (IdBandits, IdAggressive ->Aggressive, Gold_Drop)

Passive (IdPassive, Interactable)

Villager (IdVillager, IdPassive ->Passive)

Healer (IdHealer, IdPassive ->Passive, Heal)

Merchant (IdMerchant, IdPassive ->Passive, Store)

MerchantItem (IdItem->Item, IdMerchant->Merchant)

The AI gave us the following changes:

Boss (IdBoss, IdAggressive ->Aggressive, Weakness, Ability)

Monster (IdMonster, IdAggressive ->Aggressive, Weakness)

Bandits (IdBandits, IdAggressive ->Aggressive, Gold_Drop)

Villager (IdVillager, IdPassive ->Passive)

Healer (IdHealer, IdPassive ->Passive, Heal)

Merchant (IdMerchant, IdPassive ->Passive, Store)

MerchantItem (IdItem ->Item, IdMerchant ->Merchant)

ChatGPT changed the relations on the passive NPC's and the aggressive NPC's , by adding a foreign key that references the type of each NPC. (Ex: Healer (IdHealer , IdPassive ->Passive, Heal)) Here you can see that in the second argument there is a foreign key referencing the type of the healer class (Passive or Aggressive) and the primary key (which previously was the IdPassive) to an unique primary key to each NPC type.

Also, in the Merchant class, the AI separated the IdMerchant which was the primary key and the foreign key in two IDs, the IdMerchant which is the primary key and the IdPassive which is the foreign key.

In the MerchantItem relation the AI withdrew the primary key from the IdMerchant which was a primary key together with the IdItem.

We only applied the first two changes that we have mentioned, since we felt that having IdMerchant associated as a primary key with each merchant item would be the correct implementation, because the merchant that sells each item is also unique to that item.

C.1-6 Functional Dependencies and Normal Form Analysis

- idPlayer -> Username, Email, Hashed_password

- idInventory -> Max_Cap

- idType -> Name, Ability

- idCharacter -> idQuest, Quest_Status, Start_Date, Completition_Date

Case Shield:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Block

Case Weapon:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Damage, (Weapon)Type

Case Potion:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Effect

Case Armor:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Block

- idWorld -> Name, Region, Environment, Character

- idCharacter -> Name, HP, XP, Gold, Status_effect, Player, Type, Inventory

- idNpc -> HP, Name, World, Passive, Agressive

Case Boss:

- idAgressive -> Damage, Xp_Drop, Weakness, Ability

Case Monster:

- idAgressive -> Damage, Xp_Drop, Weakness

Case Bandit

- idAgressive -> Damage, Xp_Drop, Gold_Drop

Case Healer:

- idPassive -> interactable, Heal

Case Merchant:

- idPassive -> interactable, Store, Item

Case Villager:

- idPassive -> interactable

- IdItem, IdMerchant -> PRIMARY KEY (IdItem, IdMerchant)

The primary key is the factor that determines functionaly all the atributes of each relation (there are some other examples), in this case we have the keys: idPlayer, idInventory, idType, idQuest, idItem, idWorld, idCharacter, idNpc, idAggressive, idPassive

Also, we know that from a set atributes that contains one primary key that this key can be called a superkey.

An example of this is the Player class. A player_id is unique, so it's impossible to exist two or more tuples that have the same key but different atributtes. So, player_id is a key and superkey of that class.

By analysing individually, the other functional dependencies we can analyze if there are violations of the 3rd Normal Form or Boyce-Codd Normal Form.

3rd Normal Form: Is considered a violation if a non-trivial functional dependency X->Y does not follow the following rules: X being a superkey or all elements of Y have to be a part of any key.

BCNF - Is considered a violation of the Boyce-Codd Normal Form if, for a non-trivial relation, X is not a superkey.

Let's analyse each class:

1. Player:

- idPlayer -> Username, Email, Hashed_password

This relation is not violating BCNF and 3rd Normal Form, because idPlayer is a superkey since an idPlayer only references one tuple in that table.

2. Inventory:

- idInventory -> Max_Cap

This relation is not violating BCNF and 3rd Normal Form, because idInventory is a superkey since an idPlayer only references one tuple in that table.

3. Type:

- idType -> Name, Ability

Again, as the previous relations.

4. Character_Quest:

- idCharacter -> Quest, Quest_Status, Start_Date, Completition_Date

5. Item:

Case Shield:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Block

Case Weapon:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Damage, (Weapon)Type

Case Potion:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Effect

Case Armor:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Block

6. World:

- idWorld -> Name, Region, Environment, Character

7. Character:

- idCharacter -> Name, HP, XP, Gold, Status_effect, Player, Type, Inventory

8. NPC:

- idNpc -> HP, Name, World, Passive

- idNpc -> HP, Name, World, Agressive

None of these relations violate any principle since idItem, idWorld, idCharacter and idNpc are superkeys to each table.

AI tool made the following changes:

- IdPlayer -> Username, Email, Hashed_Password
- IdInventory -> Max_Cap
- IdType -> Name, Ability
- IdCharacter -> IdQuest, Quest_Status, Start_Date, Completition_Date

Case Shield:

- IdItem -> Name, Description, Value, IdInventory, IdQuest, IdType, Block

Case Weapon:

- IdItem -> Name, Description, Value, IdInventory, IdQuest, IdType, Damage, Type

Case Potion:

- IdItem -> Name, Description, Value, IdInventory, IdQuest, IdType, Effect

Case Armor:

- IdItem -> Name, Description, Value, IdInventory, IdQuest, IdType, Protection
- IdWorld -> Name, Region, Environment, IdCharacter
- IdCharacter -> Name, HP, XP, Gold, Status_effect, IdPlayer, IdType, IdInventory
- IdNpc -> HP, Name, IdWorld, IdAggressive, IdPassive

Case Boss:

- IdAggressive -> Damage, Xp_Drop, Weakness, Ability

Case Monster:

- IdAggressive -> Damage, Xp_Drop, Weakness

Case Bandit:

- IdAggressive -> Damage, Xp_Drop, Gold_Drop

Case Healer:

- IdPassive -> Interactable, Heal

Case Merchant:

- IdPassive -> Interactable, Store

Case Villager:

- IdPassive -> Interactable
- IdItem, IdMerchant -> PRIMARY KEY (IdItem, IdMerchant)

Instead of referencing, for example a relation which Inventory corresponds to idInventory, it changes that value to the id that the value references. Example:

AI redefined:

Case Shield:

- `IdItem -> Name, Description, Value, IdInventory, IdQuest, IdType, Block`

Our version:

Case Shield:

- idItem -> Name, Description, Value, Inventory, Quest, Type, Block

D.4 SQLite Database Creation

After implementing our project in SQLite, we asked ChatGPT for input, it didn't provide much help, besides explaining our features and providing suggestions of more features to implement, (it told us to utilize triggers, something we were not supposed to do in this project), to use security measures in Hashed_Password (which is redundant since the password is already a hash) so we didn't alter anything. It did help us in identifying an error that we had in the order we dropped the tables, so it was valuable in that.

E3. Data Loading

When it came to time to populate, we asked ChatGPT for help when it came to implementation, but it proved itself pretty useless in that regard, since it only suggested that we check for consistency errors and data integrity, it suggested optimized bulk insertion, and that we omit the values when we want to use default values, all things we already were doing. But, since it's a very powerful creative tool when it comes to creating objects, we asked to to give us examples of characters, worlds, items, and, in this regard, it proved quite useful in helping us in a task that would be otherwise, much more tedious. So we implemented a basic database in populate1 and fleshed it out much more in populate2.

F2-3. Generative AI Integration

The main conclusion we arrived at using AI Integration is that AI tools were best used at the beginning of the project, when it came to conversions into Relational Models, and at helping us with creative input when it came to time populate our SQL database and it helped us avoid redundancy. Overall, it proves itself as a great tool to have by your side, but by no means is it a substitute to our work, there's not much it could do when it came to the creation of the tables since our create1.sql already translated our Conceptual model and Relational Model correctly into SQL and the suggestions it provided were mostly related to implementing SQL into other interfaces.

GRUPO-102

Bruno Moreira:202107143
João Proença: 202207835
Guilherme Santos:202105090