Mòdul 2:
Base de dades

EXEMPLE:
massive multiplayer online (MMO)

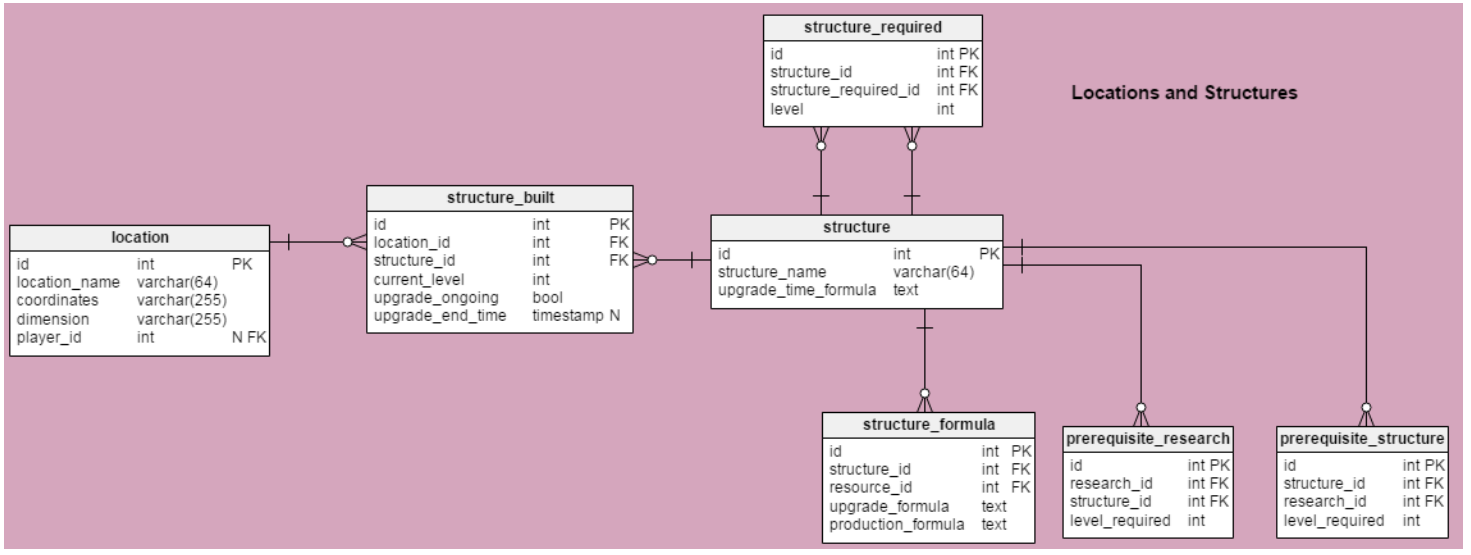
Introducció

- Until the Internet became widespread, most of us played computer games by ourselves, usually against AI opponents. It was fun, but as soon as you realized how the gameplay mechanics worked, the game lost most of its magic.
- The development of the Internet moved games online. Now, we can play against human opponents and test our skills against theirs. No more rote gameplay!
- Then massive multiplayer online (MMO) games emerged and changed everything. Thousands of players found themselves in the same game universes, competing over resources, negotiating, trading and fighting. To make such games possible, , a database structure was needed that could store all the relevant information.

https://vertabelo.com/blog/mmo-games-and-database-design/

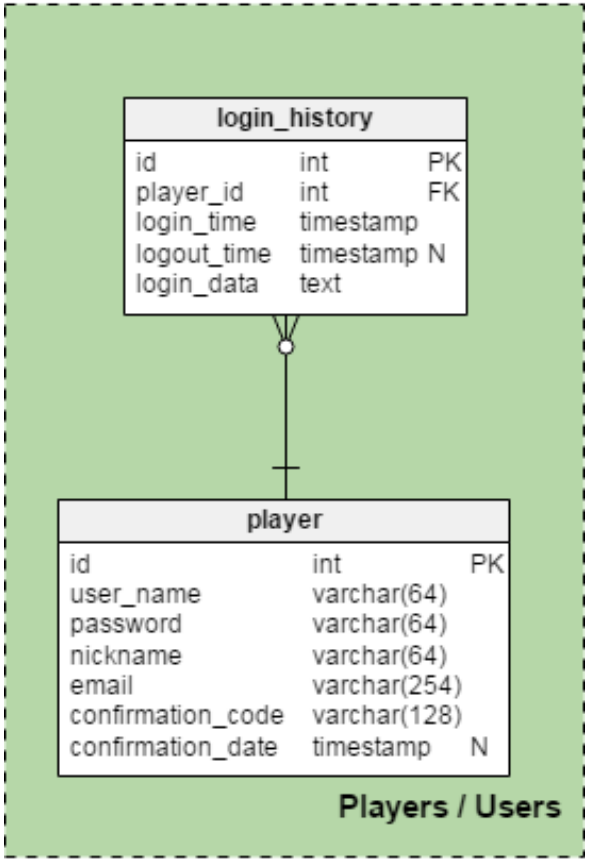
Introducció a MMO

- MMO games are set in different universes, are visually different, and use more or less different gameplay options. Still, some ideas are the same. Players compete for locations, fight for them, and form alliance with (and against) other players. They build structures, collect resources, and research technologies. They build units (such as warriors, tanks, traders, etc.) and use them to trade with allies or to fight with opponents.
- All of that needs to be supported in a **database**.



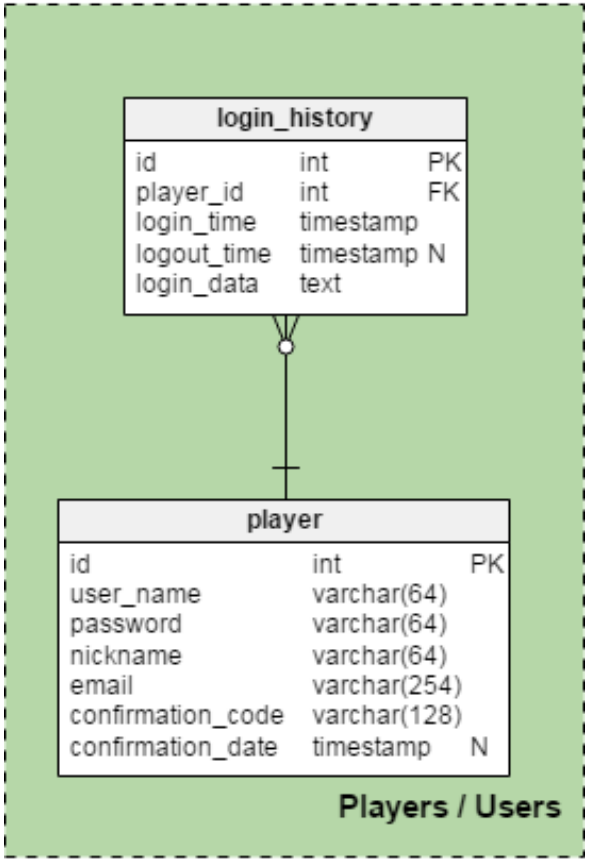
MMO – players (I)

- The ***player*** table contains a list of all registered players taking part in a game instance. We will store the players' usernames, passwords, and screen names. These will be stored in the *user_name*, *password*, and *nickname* attributes respectively.
- New users will need to provide an email address during registration. We will update the *confirmation_date* attribute when the user verifies their email address. So, this table has three unique keys: *user_name*, *nickname* and *email*.



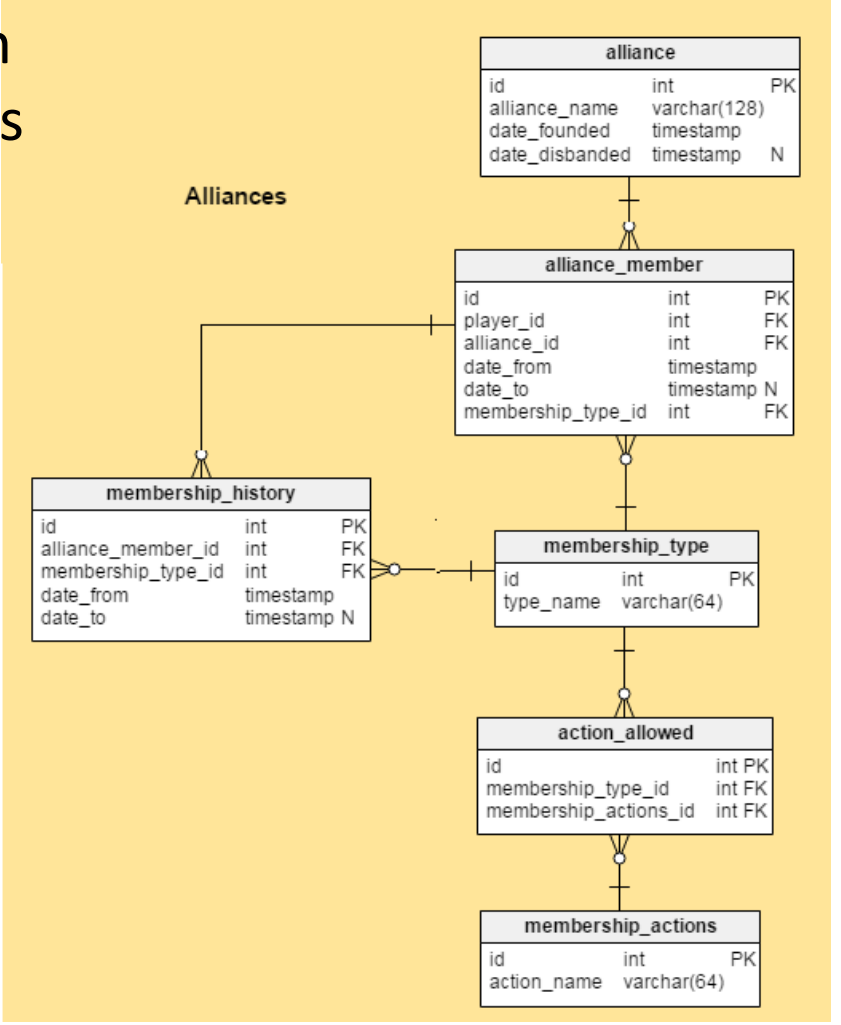
MMO – players (I)

- Each time a user logs in, a new record is added in the ***login_history*** table. All of the attributes in this table are self-explanatory.
- The *logout_time* is specific. It can be NULL when the user's current session is active or when users quit the game (without logging out) due to technical problems.
- In the *login_data* attribute, we'll store login details like a player's geographic location, IP address, and the device and browser they use.



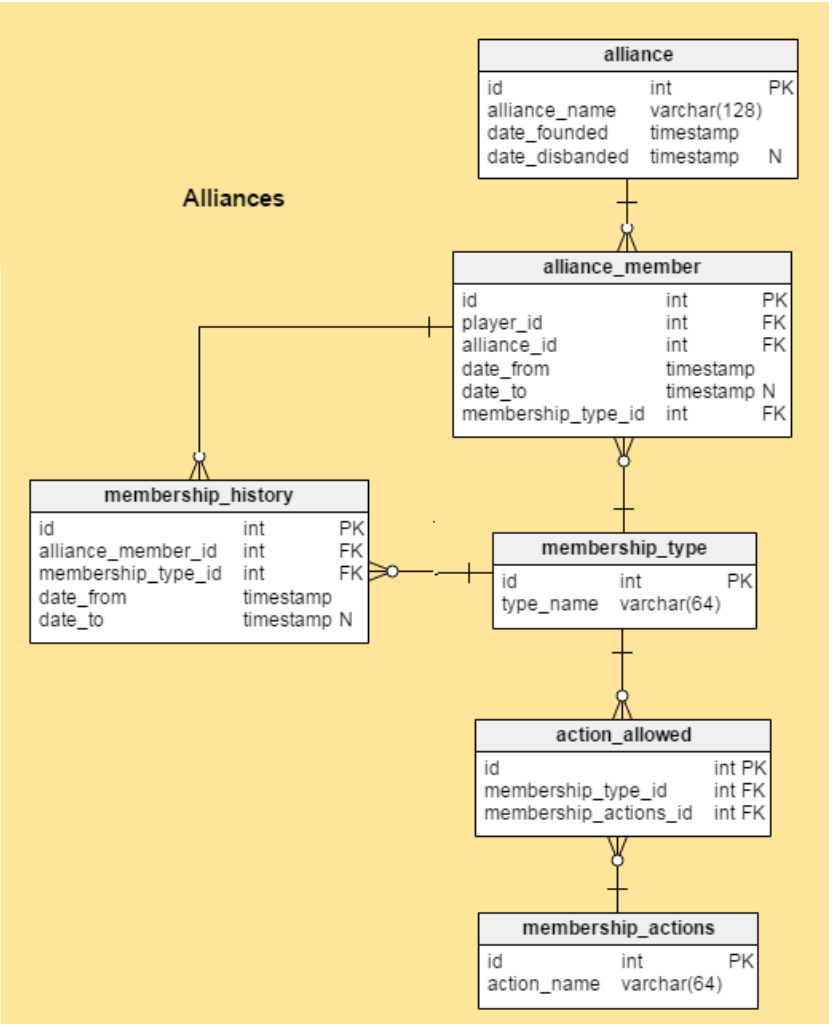
MMO – alliances (I)

- Most MMO games let us cooperate with other players. One of the standard forms of player cooperation is the alliance.
- The ***alliance*** table stores basic information about game alliances. Each one has a unique *alliance_name* that we'll store. We'll also have a field, *date_founded*, that stores when the alliance was founded. If an alliance is disbanded, we'll store that information in the *date_disbanded* attribute.
- The ***alliance_membe***r table relates players with alliances. Players may join and leave the same alliance more than once. Because of this, the *player_id – alliance_id* pair is not a unique key.



MMO – alliances (II)

- We'll keep information regarding when a player joins the alliance and when (if) they leave in the *date_from* and *date_to* fields. The *membership_type_id* attribute stores the current level of players' rights in the alliance, it is a reference to the ***membership_type*** dictionary (tabke),
- Players' rights in an alliance can change over time. The ***membership_actions***, ***membership_type*** and ***actions_allowed*** tables together define all possible rights for alliance members.
- To sum up: the values stored in these tables are defined by during the initial setup; they'll change only if we introduce new options.



MMO – alliances (III)

- The ***membership_history table*** stores all data regarding players' roles or rights within an alliance, including the range when these rights were valid. (For example, he could have "novice" permissions for a month, and then "full membership" from that point on.) The *date_to* attribute is NULLable because currently active rights have not ended yet.
- The ***membership_actions*** dictionary contains a list of every action players can make in an alliance. Each action has its' own action_name and game logic is built around these names. We can expect values like "view members list", "view members' statuses" and "send message" here.