KUZEY KIBRIS KAMPÜSÜ
ODTÜ
METU
NORTHERN CYPRUS CAMPUS

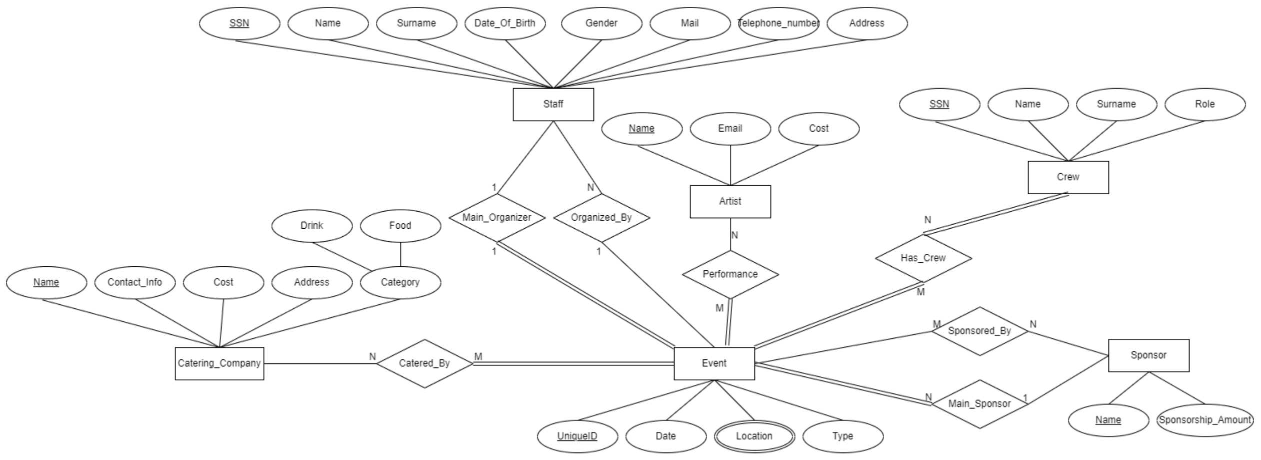
CNG 351 Assignment 1

Team Members:

Name: Muhammad Somaan
ID: 2528404
Name: Daniil Belousov
ID: 2491827

Use Case 1
MENTOR (Metu EveNT OrganiseR)

Design:



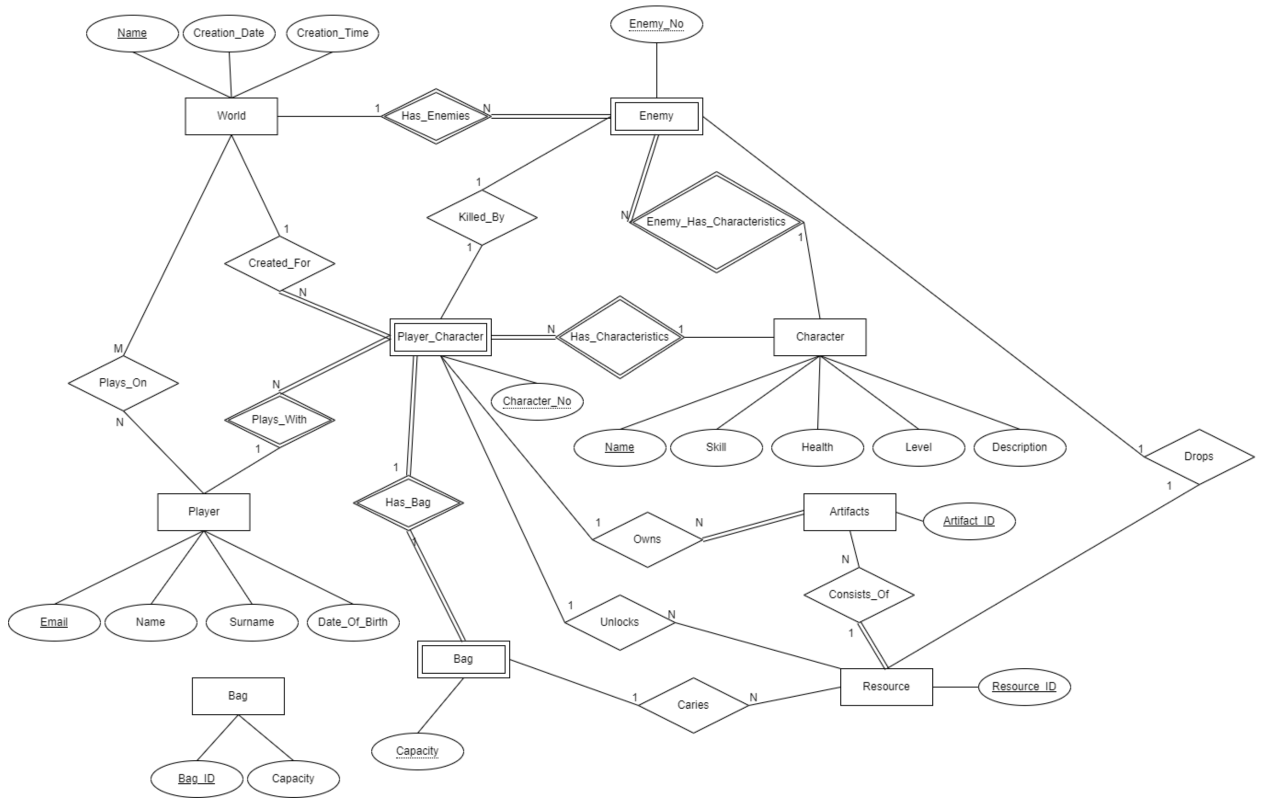
Assumptions:

- Organizer and Crew primary attribute is SSN as ssn is unique for every person.
- For Sponsor, Artist and Catering_Company entity we are making Name as primary attribute, making the assumption that every sponsor’s, artist’s and catering’s name will be unique.
- Assuming that sponsors will be companies, so the name will be unique.
- Assuming that artist's name will be the stage name, which is unique to every artist, such as “50 Cent”, and “Eminem”.
- Main_Organizer relation has partial participation from Organizers entity side since not all organizers are going to be the main organizer for an event.
- 1 to 1 relation for Main_Organizer considering that only one organizer is the main organizer for a single event.

- Partial participation in Organized_By relation, as it is possible only 1 staff organizes the event, in that case, it will be the main organizer.
- Main_Sponsor relation has partial participation from Sponsors side since not all sponsors are going to be the main sponsor for an event.
- Many to 1 relation for Main_Sponsor considering that one sponsor can be the main organizer for multiple events.
- Partial participation in Sponsored_By relation, as it is possible only 1 sponsor sponsors the event, in that case, it will be the main sponsor.
- Artist has partial participation as we may have stored an artist information, but the artist is not performing in any event.
- Catering_Company has partial participation as we may have stored catering company information, but it is not catering any event.
- Catering_Company has many to many relation as it is possible for a single catering company to cater to multiple events.

Use Case 2
Survive

Design:



Assumptions:

- Email attribute is the primary attribute for Player entity as email is unique to every person.
- Player Character is a weak entity that is identified by multiple strong entities, Player and Character. It is so because a player can have characters for different worlds, but a character can also be used by enemies.

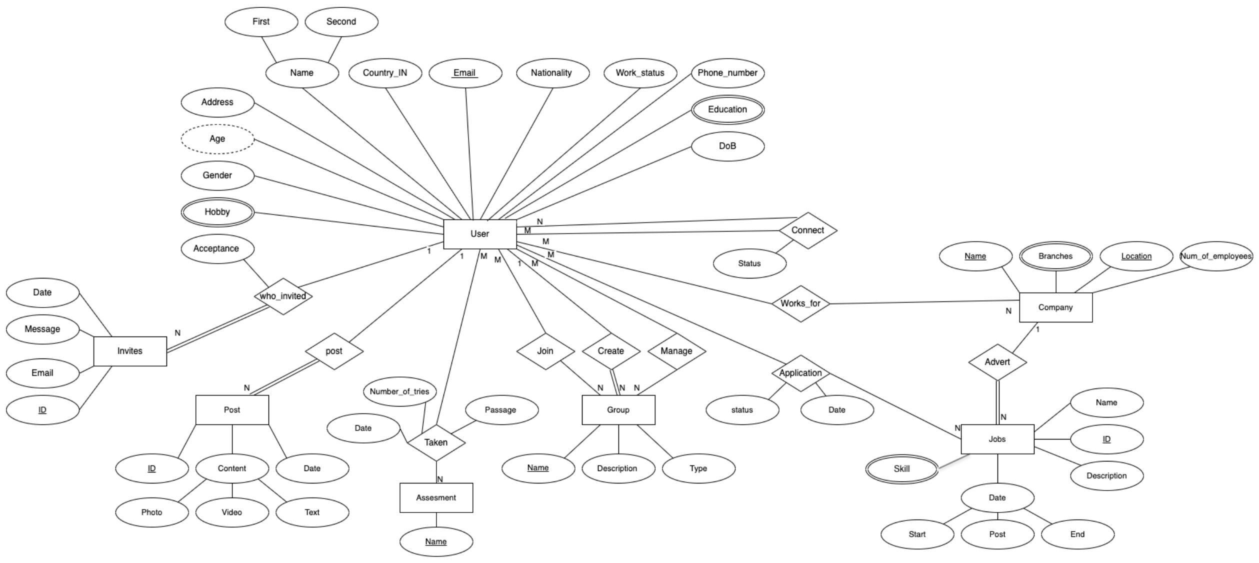
- Player Character entity has a partial key Character_No, assuming that data is logged for all player characters that the player made for each world, character_no would be an integer that stores the creation number of the player character. For example, if a player is playing for the second time, the character number will be 2 for the new world that the player plays on.
- Plays_On relation has many to many relation assuming that the player can play on multiple worlds and that the data related to a world such as players joined is not deleted when a game ends.
- Plays_With relation has 1 to many relation as 1 player may have many characters to play with on different worlds.
- Plays_With relation has partial participation on player side, as it is possible a player has not joined a world yet meaning no player character created yet.
- Has_Characteristics relation has 1 to many relation as 1 character may have many player characters by different players, or even the same player but on a different world, but a player character is only 1 character.
- Has_Characteristics relation has partial participation on character side, as it is possible a character is not used by any player to create a player character.
- Created_For relation has 1 to many relation as 1 world may have many player characters by different players, assuming that only one character for one world per player restriction is applied.
- Created_For relation has partial participation on world side, as it is possible a newly created world is not joined by any player for it to have a player character.
- Enemy is weak entity as it cannot be identified by it’s own, and is identified by the character it is and the world it is in.
- Enemy entity has Enemy_No as a partial key, assuming that enemies will have a numbering system to them, that 1st enemy created will have Enemy_No as 1, then 2nd enemy will have it as 2 and so on.
- Enemy_Has_Characteristics relation has 1 to many relation as 1 character may have many enemies.

- Enemy_Has_Characteristics relation has partial participation on character side, as it is possible a character is not used by any enemy.
- Bag is a weak entity with identifying relationship with Player_Character, assuming that in the requirements by “sharing bags” statement means that a bag will always have an owner, but the owner can change mid-game.
- Has_Bag is a 1 to 1 relation as 1 character can only hold 1 bag, and 1 bag can only have 1 owner at a time.
- Has_Bag has full participation from both sides, as a character needs to have a bag at all times according to the above assumption, and similarly a bag needs to have an owner at all times.
- Caries is a 1 to many relation, because a bag can carry multiple resources, it is also partial on both side because it is possible that the bag is empty, similarly it is possible for a resource to be on ground and not picked by anyone yet.
- Resource is assumed entity for storing all the dropped resources that may be picked by a character, it’s primary key is assumed to be Resource_ID which is going to be unique for every resource that is dropped by enemy.
- Drops is a 1 to 1 relation as 1 enemy can only drop 1 resource, and since a resource is uniquely identified it can only be dropped by 1 enemy.
- Artifacts is an assumed entity with Artifact_ID as its primary key, which will be unique for every artifact unleashed.
- Consists_Of relation is 1 to many relation because, 1 resource may contain many artifacts, but 1 unique artifact can only come from one resource.
- Consists_Of has total participation from Resource entity as a resource must have artifacts within it when unlocked.
- Unlocks relation is 1 to many relation because, a player can unlock many resources but a unique resource can only be unlocked by a single player.
- Unlocks has partial participation on both side since it is possible that a player does not unlock a resource.

- Owns is 1 to many relation as a single player can own multiple unique artifacts but a unique artifact can be owned by multiple players.
- Owns have full participation with Artifacts entity as when an artifact exist it is due to when a player unlocks a resource but the resource must be present in character’s bag, only then the artifact is unleashed, and that player is associated with it.
- Owns relation have partial participation with Player_Character entity since it is possible that a player haven’t unlocked any resource yet and have no artifacts.

Use Case 3
FindJob

Design:



Assumptions:

For this database we assume the only possible memberships are user and company entities, however they have different required information. Each user will be having a personal unique id to be distinguishable but companies will just have the name and the location to be distinguished as we assume no two companies can have the same name and location.

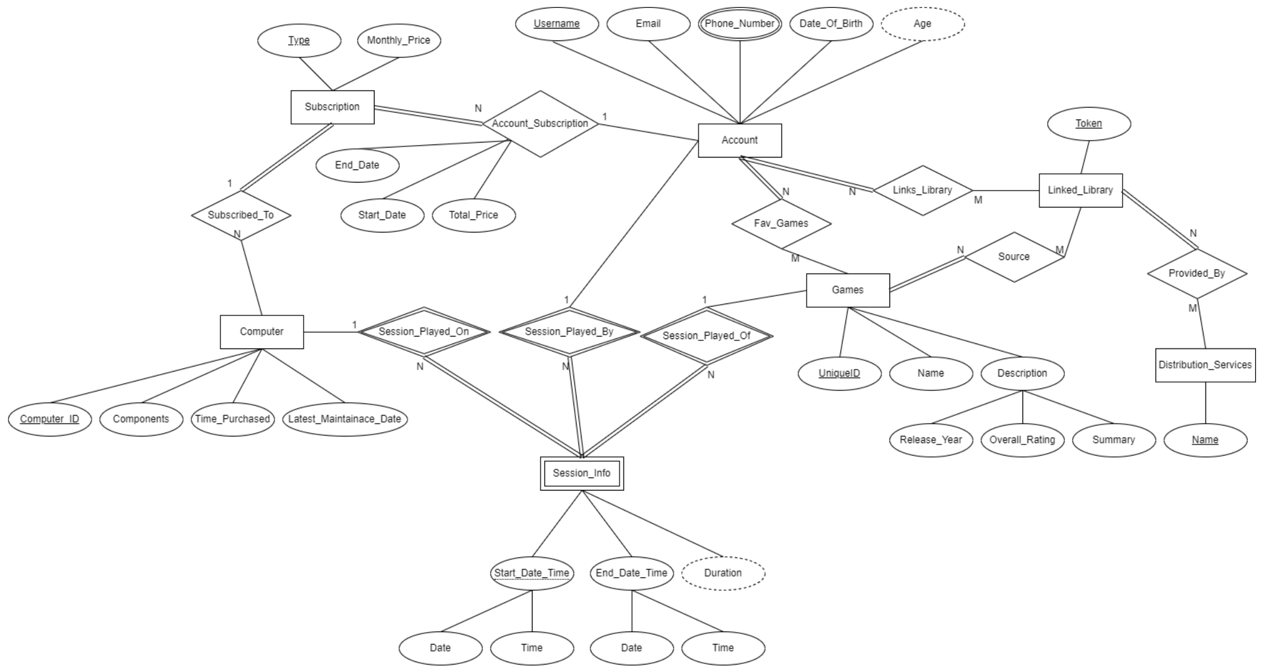
- Users: Each user will be having a personal unique email to be distinguishable as there may be situations when most of the other information will be the same, so email will be used as the primary key.
- Invites: the invitation id is unique and can not be replicated
- Post: the post id is unique and can not be replicated

- Assessment: we assume that each assessment has a unique identifying name
- Group: its id is unique and can not be replicated
- Jobs: each one should have a specific id and no two jobs can have the same id ever
- Compony: we assume companies can have the same names (if they are in different counties and do not work internationally), that is why we use a composite key of name and location as we assume no two companies can have same name and location of the main office
- Connect relationship assumes that many users can have many connections but if the user do not want to have connections or have not made any yet it is also possible to have 0 connections, however if we have a connection we need to keep the status of it, if it is rejected or approved
- Who_invited relationship: we assume that there are users who could not have invited anyone or there could be users with multiple sent invitations. this relation also stores the information if the invitation was accepted. If an invitation is sent, it must participate fully as an invitation must have a User entity associated with it.
- Taken relationship: we assume users can take as many assessments as they want and the assessments could be taken by any amount of people. In this relationship, we store the number of tries as a user can take a limited amount of them in a given time interval which is also stored in Date so that if the user fails all the tries we would know when we can renew the number of tries.
- Post relationship: here we assume that users can have as many posts as they want but the post can have only one author. if there is a post, it must have an author but not all the users must have posts.
- Join relation we assume that people can join any amount of groups and groups can have any amount of participants
- Create relation we assume that a user can create any amount of groups but the group can have only one creator. However, if there is a group it must have a creator but the users may not have any groups created by them

- Manage relation we assume that users can manage any amount of groups, and the groups can have any amount of managers. Group doesn't need to have anyone to manage it (partial participation) as we assume that the creator holds the power to manage the group.
- Application relationship: here we assume that people can apply for any amount of jobs and jobs can have any amount of applicants, but there might be people who did not apply for any jobs and there could be jobs that did not receive any applications. this relationship should also keep tread when the applicant sent the application and the status of the application if it was submitted, saved, or incomplete.
- For the advert relationship: the company may advertise any amount of jobs but if the job is there it must have only 1 company that provides it
- works_for relationship: we assume that users can work for many companies (freelancer or advisor, etc.) and company may have any amount of workers

Use Case 4
NCCloud

Design:



Assumptions:

- In Distribution_Services entity it is assumed that each will have a unique name, as it most likely a company name which is unique.
- In Source relation Games entity takes full participation because if there is a game, then it has to be associated to a library.
- In Source relation Linked_Library takes partial participation as it is possible that an account's library has no games in it, eg. no games bought on steam.
- Source is many to many relation because it is possible for a game to exist in different libraries and also possible for a single library to hold different games.

- Linked_Library entity holds the library that was provided by the distribution service for their account.
- Provided_By has many to many relation as a one single distribution service will provide libraries for different users and it is possible for a single user to link libraries from different distribution services.
- In Fav_Games, Games entity takes partial participation as it is possible that a game may not be in any account's favorite list.
- Assuming that 1 account can have multiple subscriptions, maybe different dates, maybe different types that give access to better computers.
- For Account_Subscription relation, the total price attribute is not derived, as it is possible that a discount may be given to those with a longer period of subscription, for example, 1-year subscription.
- In Subscription entity type is chosen to be the primary key, as it makes sense to have different unique types, eg. student, premium, premium plus.
- Subscribed_To is 1 to many relation, assuming that only multiple computers are allocated to the subscriber, in case if one is being used by someone else, a subscriber can use another allocated computer.
- Subscribed_To relation has partial participation from Computer side, as NCCCloud does not want all the computers to be allocated to subscribers, but full participation from Subscriber side, as a subscriber must have a computer allocated to it.
- Session_Info is a weak identity, identified by Game, Computer and Account, as Session_Info itself doesn't have a unique key.
- Session_Played_On, Session_Played_Of and Session_Played_By all relations are 1 to many, as many sessions can be played on a single computer, many sessions can be played by a single account, and many sessions can be played of a single game.
- Session_Played_On has partial participation from Computer as it is possible a computer is never been used.
- Session_Played_Of has partial participation from Games as it is possible a game has never been played.

- Session_Played_By has partial participation from Account as it is possible that an account has not yet played any game.