KUZEY KIBRIS KAMPÜSÜ
ODTÜ METU
NORTHERN CYPRUS CAMPUS

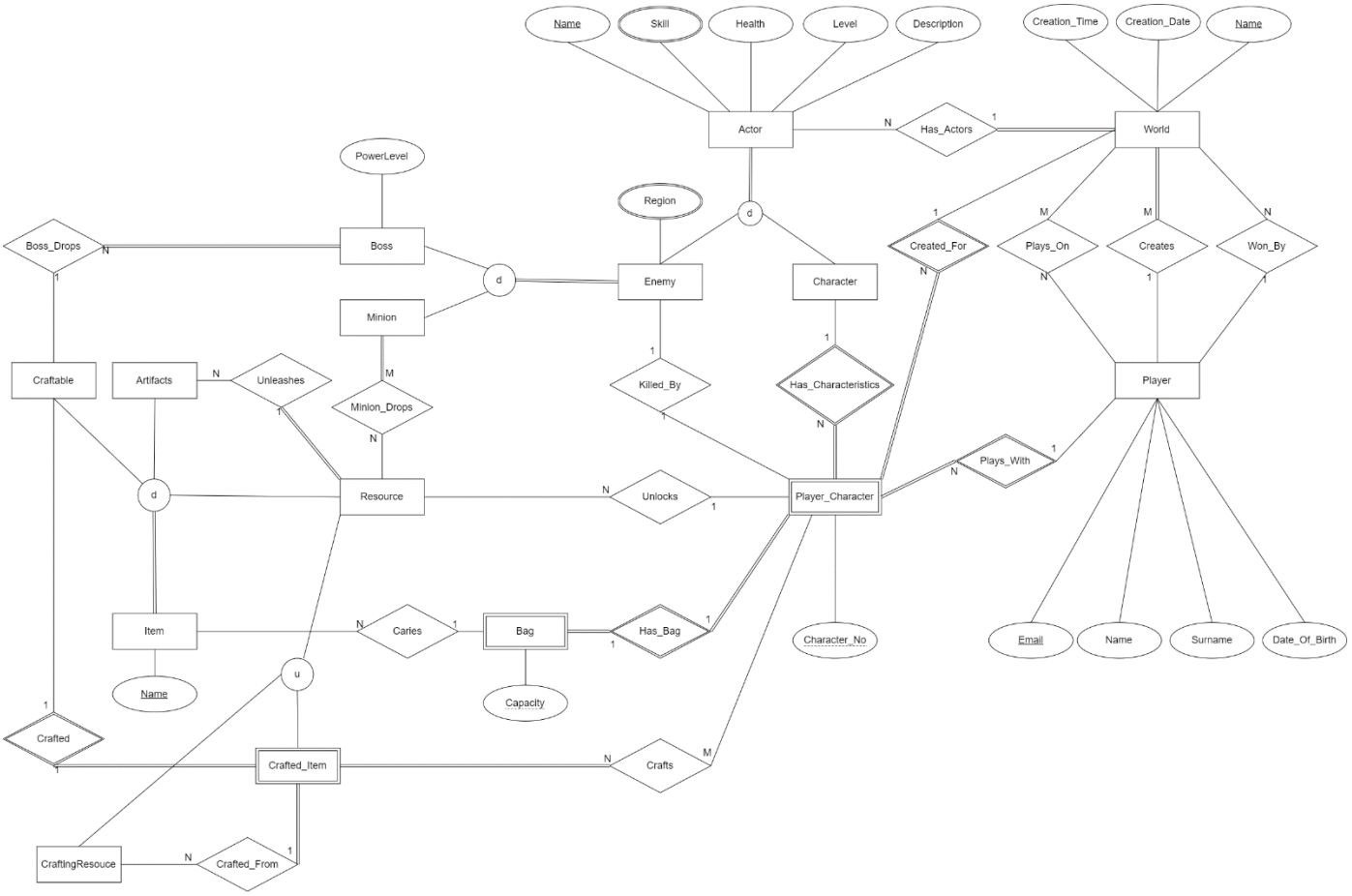
CNG 351 Assignment 2

Team Members:

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

PART 1:
Use Case 1
Survive

Design:



Assumptions:

- Email is the primary attribute for the Player entity, as email is unique to every person.
- Player Character is a weak entity identified by multiple strong entities, Player and Character. It is so because a player can have characters for different worlds.
- 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 plays for the second time, the character number will be 2 for the new world on which the player plays.

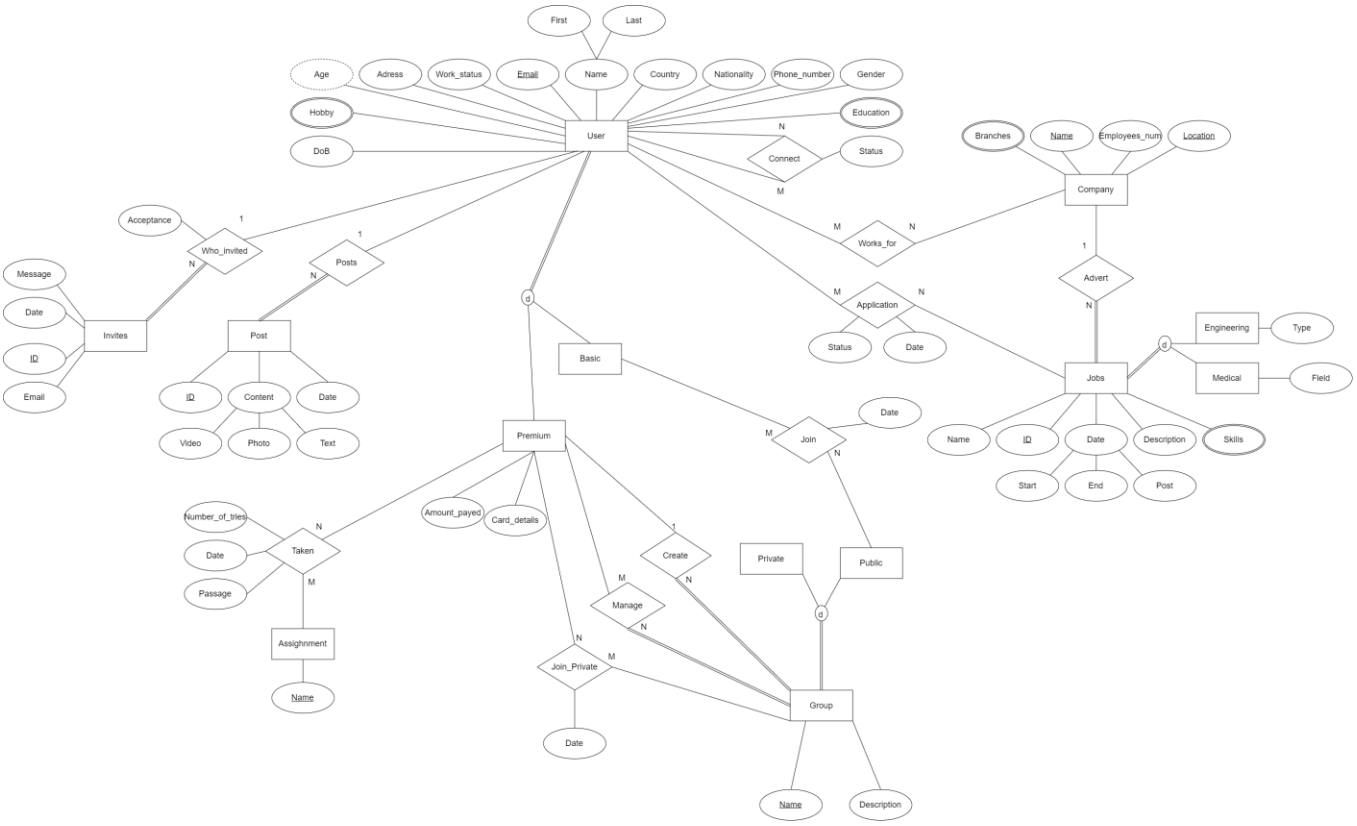
- Has_Actors relation is 1-to-many, as each world can have multiple actors.
- Has_Actors has full participation on the World side, as a world needs to have actors in it, either enemy or player characters. It has partial participation on the Actor side as it is possible for an actor to exist that is not used in that world.
- Enemy entity has Region as a multi-valued attribute, assuming that it is possible that a single enemy can spawn in multiple regions, as is commonly seen in games.
- 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.
- Creates relation is a 1-to-many relation, as one player can create multiple worlds.
- Creates relation has partial participation on the Player side as it is possible for a player never to create a world, but on the World side it is full participation, as a World needs to be created by a player.
- Won_By relation is 1-to-many relation, as one player can win in multiple worlds.
- Won_By relation has partial participation on both side, as there can be a player that has never won, but it is also possible for a World to have no winners, in case of a server crash.
- 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.
- Bag is a weak entity with identifying relationship with Player_Character, assuming that in the requirements “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 one character can only hold one bag, and one bag can only have one 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 items; it is also partial on both sides because it is possible that the bag is empty. Similarly it is possible for an item to be on ground and not picked by anyone yet.
- Boss_Drops is a many-to-1 relation as a boss can only drop 1 craftable item, but similar craftable items may be dropped by multiple bosses.
- Boss_Drops has full participation on the Boss side since a boss must drop a craftable item when it is killed, but on the other hand, there can exist a craftable item that is not dropped by any boss in that world.
- Minion_Drops is a many-to-many relation as a minion can only drop multiple resources and similar resources may be dropped by multiple minions.
- Minion_Drops has full participation on the Minion side since a minion must drop a resource when it is killed, but on the other hand, there can exist a resource that is not dropped by any minion in that world.
- Artifacts are also an item, since they can be carried by a bag, and a bag can only carry items.
- Unleashes relation is a 1-to-many relation, as 1 resource can unleash multiple artifacts when unlocked.
- Unleashes relation has full participation on the Resource side since a resource must unleash some artifacts when it is unlocked. Artifacts has partial participation in this relation assuming that there can be other ways to obtain artifacts.

- Unlocks relation is a 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 sides since it is possible that a player does not unlock a resource.
- Crafted_Item is a weak entity that is identified by a Craftable item, since a player needs to craft a craftable item to use it.
- Crafted relation is 1-to-1 identifying relation for Crafted_Item.
- Crafts relation is a many-to-many relation, as multiple players can craft multiple craftable items.
- Crafts relation has full participation on Crafted_Item side as it needs a player to craft it, but on the other hand, Player_Character has partial participation since it is possible for a player to never craft a craftable item.
- CraftingResource entity is a union of Resource and Crafted_Item, since for a craftable item to be used, it can require these 2 types of items to be crafted.
- Crafted_From relation is a 1-to-many relation, as a crafted item can be crafted from multiple resources or already crafted items.
- Crafted_From has full participation on Crafted_Item side, as a Crafted_Item needs to be crafted from crafting resources, but on the other hand, crafting resources can exist without being crafted in an item.

PART 1:
Use Case 2
FindJob

Design:



Assumptions:

- 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.
- Users is further specialized into disjointed 2 entities (premium and basic) with full participation
- 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
- Jobs is further disjoined into engineer and medical entities with partial participation, because if there is a job it might be a different type
- 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 premium 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 users can join any amount of public groups and groups can have any amount of participants

- Join_Private relation we assume that premium members can join any amount of public and private groups and groups can have any amount of participants
- Create relation we assume that a premium 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 premium 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

PART 2:
Use Case 1
MENTOR (Metu EveNT OrganiseR)

Design:

Artist

Email	Name	Price

Event

ID	Name	Date_Day	Date_Time	E-Manage [FK: Organizer: SSN]	ManageSponser [FK: Sponsor: SponID]

Social

ID [FK: Event: ID]	Type	Perform [FK: Artist: Email]

Conference

ID [FK: Event: ID]	Edition

Sport

ID [FK: Event: ID]	Category

Location

ID	Name

Organizer

SSN	Email	Address	Name	Surname	DOB	Gender

Organizer_Phone_Number

SSN [FK: Organizer: SSN]	Phone_Number

Organize

SSN [FK: Organizer: SSN]	Event_ID [FK: Event: ID]

Crew

SSN	Name

Work

SSN [FK: Crew: SSN]	Event_ID [FK: Event: ID]	Role

Sponsor

SponID	Name

Participate

SponID [FK: Sponsor: SponID]	Event_ID [FK: Event: ID]	Funding_Amount

Catering

CatID	Address	Name	Contact_Name	Contact_Surname	Contact_Phone_Number	Food_Flag	Type	Drinks_Flag	Brand

Supply

CatID [FK: Catering: CatID]	Event_ID [FK: Event: ID]	Cost

Comments:

- In the Organizer table, since it had 2 candidate keys SSN and Email, we chose SSN as the primary key.
- Organize table is the table for the relationship organize between Organizer and Event.
- Organizer_Phone_Number table stores the multivalue attribute of Phone number in Organizer table
- Work table is the table for the relationship work between Crew and Event.

PART 2:
Use Case 2
NCCCloud

Design:

User_Account

Username	Email	DoB	Age

User_Account_Phone_Number

Username [FK: User_Account: Username]	Phone_Number

Library

Connection_Token	Name	Link [FK: User_Account: Username]

Steam

Connection_Token [FK: Library: Connection_Token]	Level

EpicGames

Connection_Token [FK: Library: Connection_Token]

Game

ID	Name	Description

Library_Has_Games

Connection_Token [FK: Library: Connection_Token]	Game_ID [FK: Game: ID]

Favourite_List

Username [FK: User_Account: Username]	Name	Date

Favourite_List_Contains_Games

Username [FK: Favourite_List: Username]	Name [FK: Favourite_List: Name]	Game_ID [FK: Game: ID]

Computer

ID	Price	Time_Bought	Motherboard	GPU	CPU	Type	SSD_Capacity	Maintanence_Date

Play_Session

Username [FK: User_Account: Username]	Game_ID [FK: Game: ID]	Comp_ID [FK: Computer: ID]	Start_Date_Time	End_Date_Time	Duration

Subscription

ID	Start_Date	End_Date	Monthly_Price	Total_Price	Type	Loyalty	Premium_Type

Subscribe

Username [FK: User_Account: Username]	Subscriber_ID [FK: Subscription: ID]

Subscription_Enables_Computers

Computer_ID [FK: Computer: ID]	Subscriber_ID [FK: Subscription: ID]

Comments:

- In the User_Account table, since it had 2 candidate keys Username and Email, we chose Username as the primary key.
- User_Account_Phone_Number table stores the multivalue attribute of Phone number in Username table
- Library_Has_Games table is for the Has relation between Library and Game
- Favourite_List_Contains_Games table is for Contains relation between Favourite_List and Game
- In Computer table Type attribute is used to identify the child, if its Low Medium or High.
- SSD_Capacity attribute in Computer table can be null.
- In Subscription table Type attribute is used to identify the child, if its Free, Priority or Premium.

- Loyalty and Premium_Type attributes in Subscription table can be null.
- Premium_Type attribute is used to store the type for Premium type of Subscription.