

# Gatcha Impact Database prosjekt

Gruppe 6: Memory Access Violation

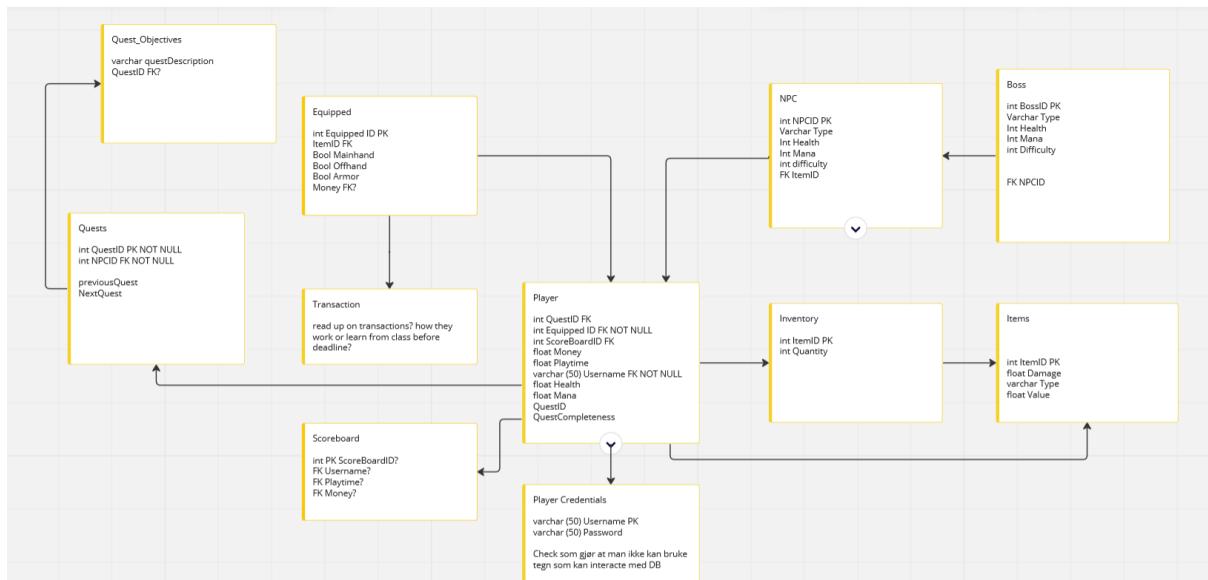
## Grovdesign

Hovedhensikten med datasystemet som helhet er å styre ulike komponenter som utfyller et MMO-spill.

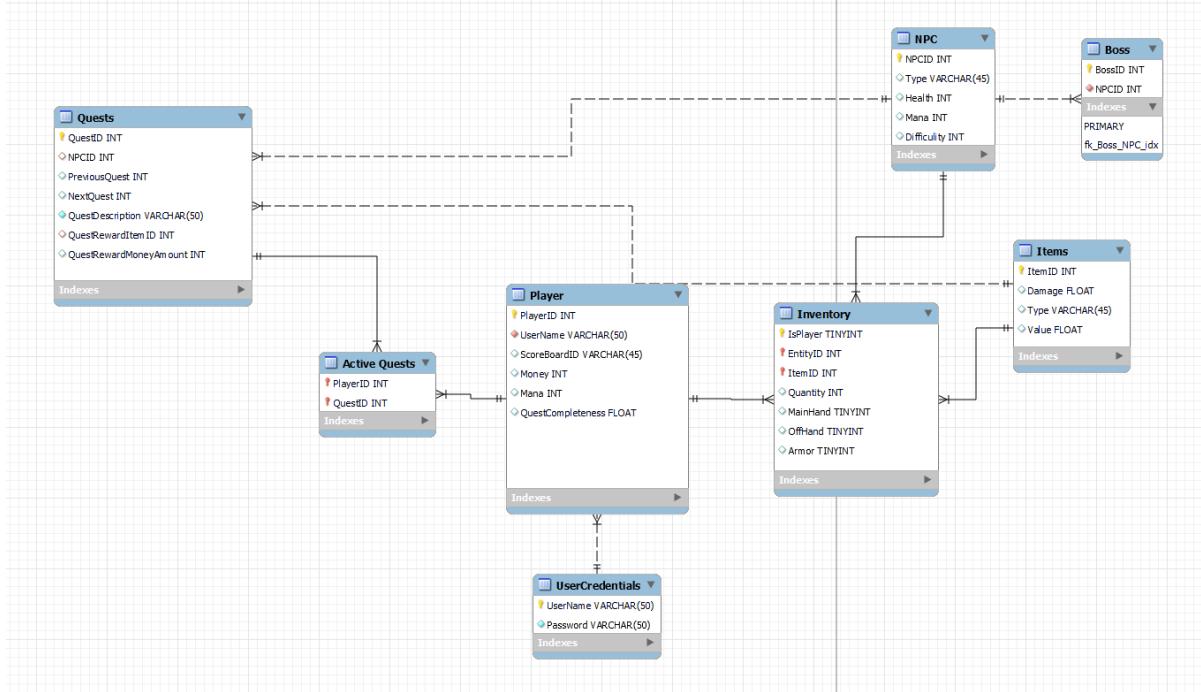
Databasen skal brukes til å holde styr på:

- Hvilke quests(oppdrag) en spiller-karakter har gjort / ikke har gjort
- Hvilke items(gjenstander) en spiller-karakter har
- Spillernes påloggingsinformasjon.
- Scoreboard(poengtabell) som inneholder hvor lenge en spiller-karakter har spilt, og antall penger.

## skisser/illustrasjoner:



# Mer detaljert design



Første ER-diagram for databasen. MySQL versjonen av diagrammet er inkludert i filen `GatchaImpactModel.mwb` som 'Concept diagram'.

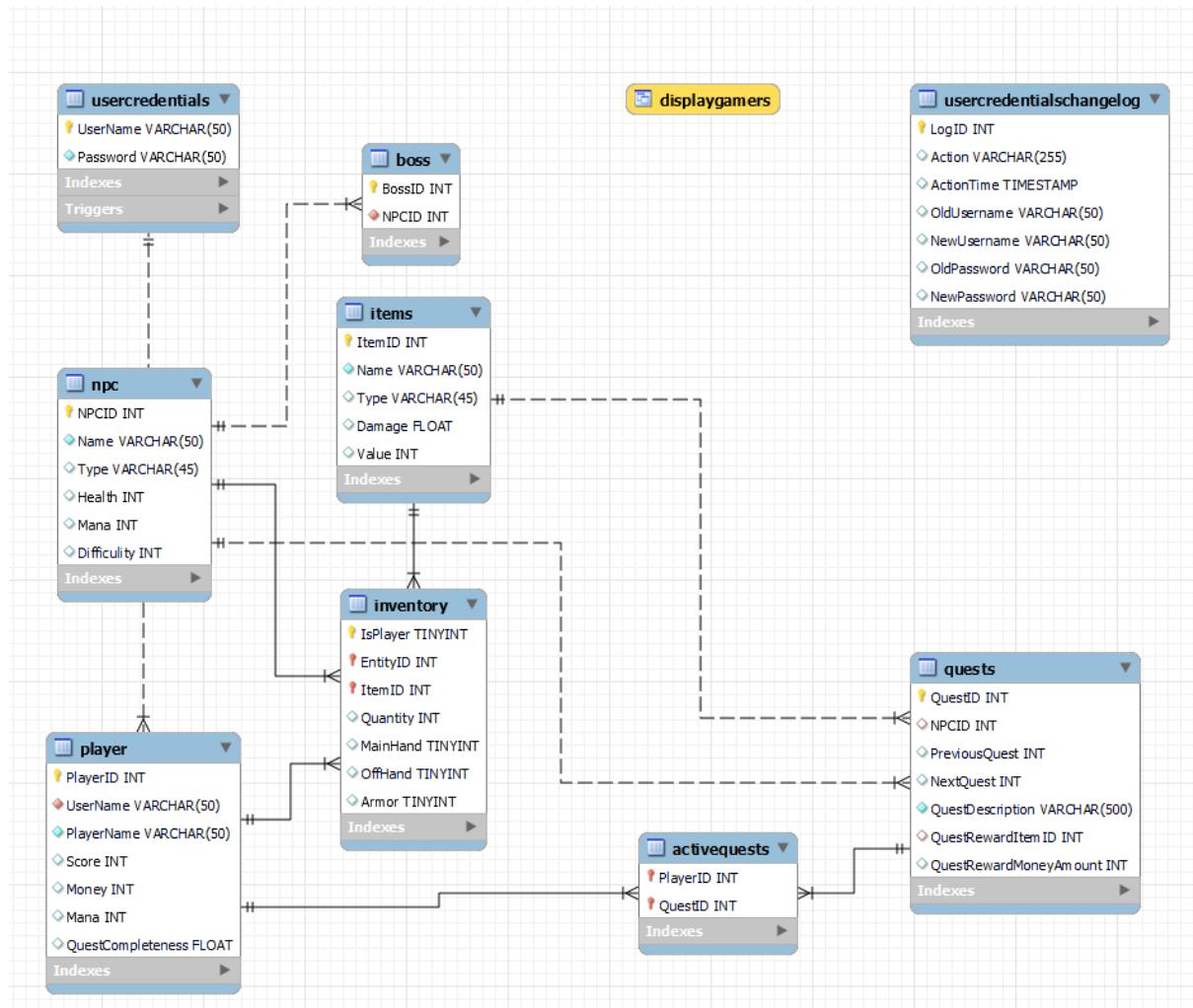
## Utdypning av tabellene:

- Player:** Representerer en av eventuelle flere spiller-karakterer til en UserCredentials rad. Derfor har vi satt UserName i player som FK for UserName i UserCredentials sin PK. I tillegg har Player PlayerID som PK. Med denne kan andre tabeller (Inventory og ActiveQuests) ha en FK kobling til Player. Player har også noen andre variabler som Score, Money, Mana og QuestCompleteness som er variabler bare relevant til spiller-karakteren sin tilstand.
- UserCredentials:** en rad representerer en User eller bruker av spillet. Holder Password og PK UserName, hvor player har en FK UserName som en kobling til dens User.
- Quests:** Quests holder alle quests i spillet, den har QuestID PK slik at ActiveQuests kan ha en kobling til den. Den har også en NPCID FK til den som ga questen, og FK QuestRewardItemID som er Item reward for å gjøre question.
- Active Quests:** Holder en FK QuestID og FK PlayerID, dette gjør at for hver Active Quest til en Player har vi en rad.
- Items:** I tillegg til sine tilstandsvariabler (Damage, Type og Value), så har den en PK ItemID, slik at andre tabellen kan ha en FK kobling til en spesifikk Item.
- Inventory:** Fungerer på samme konseptet som Quests og Active Quests. Men siden både Player og NPC kan ha en eller flere Inventory rader. Så har vi en EntityID (enten PlayerID eller NPCID) og en bool isPlayer som sier om denne inventory raden tilhører en Player eller NPC. For hver Player eller NPC sin Item har vi en rad.
- NPC:** Hver rad representerer en NPC i spillet. har en PK NPCID, som gjør at andre tabeller kan ha en FK kobling med denne. I tillegg har den andre tilstandsvariabler Type, Health, Mana og Difficulty.

- **Boss:** Representere en utvidelse av en NPC. Har en PK BossID og en FK NPCID til NPC-en som den er en utvidelse av.

## Database-implementasjonen

Fra det konseptuelle ER-diagrammet i oppgave 3, så brukte vi MySQL sin forward-engineering til å generere et skript som lager database-schematet ('GatchalImpactGenerateSchema.sql'). Vi gjorde så endringer, eller utvidelser av dette skriptet etterhvert som vi korrigerte implementasjonen.



*Endelig versjon av ER-Diagrammet, basert på den faktiske databasen etter alle endringer, views, stored procedures og triggers er lagt til. MySQL versjonen av diagrammet er inkludert i filen 'GatchalImpactModel.mwb' som 'Implementation diagram'.*

## Eksempladata, Views og Stored Procedures

Vi lagde et skript 'GenerateExampleDataSPsAndViews.sql' som legger til eksempladata i databasen, og lager 6 stored procedures, som innkapsler viktige insert, update, og select setninger.

- ‘createUser’ lar oss legge til en ny bruker med gitt brukernavn og passord.
- ‘ChangeUsername’ lar oss endre brukernavn på en gitt bruker.
- ‘ChangePassword’ lar oss endre passordet til en gitt bruker.
- ‘displayPlayerInventory’ lar oss se inventaret til en gitt spiller-karakter.
- ‘CreatePlayer’ lar oss lage en ny spiller-karakter for en gitt bruker.
- ‘createNPC’ lar oss lage en ny non-player character.

Vi lagde en view ‘DisplayGamers’, som lar oss se alle spiller-karakterene og brukeren de er knyttet til.

## Viktige SELECT, UPDATE og INSERT setninger

De fleste av de viktige UPDATE og INSERT setningene våre er innkapslet i SPene nevnt i forrige del av oppgaven. I tillegg inneholder skriptet ‘ImportantQuerries.sql’ de viktige SELECT setningene som vi har laget, som å finne alle aktive quests, finne alle spillere som driver med en gitt quest, og finne alle questene en gitt spiller driver med. ‘ImportantQuerries.sql’ inneholder også eksempler på bruk av stored procedurene som vi lagde i ‘GenerateExampleDataSPsAndViews.sql’.

## Triggere og endringslogg

Til slutt lagde vi en tabell UserCredentialsChangelog, som viser endringer gjort på tabellen UserCredentials. Endringslogg-tabellen er definert på linje 196 til 207 av ‘GatchalImpactGenerateSchema.sql’, og for å automatisk logge endringene gjort på UserCredentials-tabellen, så lagde vi tre triggere på linje 209 til 238 av ‘GatchalImpactGenerateSchema.sql’. En som logger insertions, en som logger updates, og en som logger deletions.