Database Øving 8

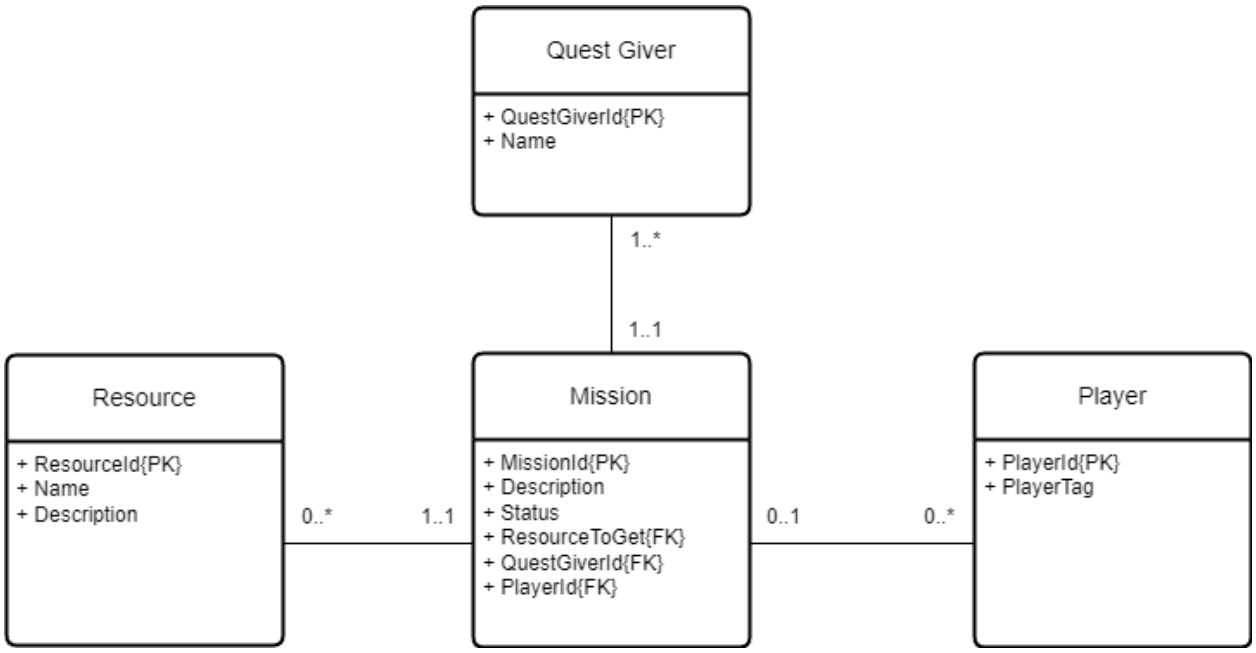
Oppgave 1.1

Den tenkte problemstillingen baserer seg på ideen om et quest system i et spill. Tanken er at man har en liste bestående av forskjellige oppdrag. Hvert oppdrag i listen blir definert ut fra en auto incrimenterende id, beskrivelse, status, ressurs som må samles, hvem som har gitt oppdraget og hvilken spiller som har tatt på seg oppdraget. ResourceToGet, QuestGiverId og PlayerId i mission listen er alle satt opp som fremmednøkler som kobler tabellen opp mot tre andre tabeller i databasen. De tre andre tabellene er Quest Giver, Player og Resource. Quest giver inneholder infoen som er viktig å vite om den som har gitt ut oppdraget. I det tenkte tilfellet her setter vi en QuestGiverId og et navn. Player tabellen er satt opp på samme måte som Quest Giver, med en PlayerId og en PlayerTag(Navn). Resource er en liste over forskjellige typer tenkte ressurser, hvor hver ressurs består av en ressurs id, navn og en forklaring på hva ressursen er.

Målet for databasen er å sette opp et oversiktlig system som lar en tenkt spiller få oversikt over alle tilgjengelige oppdrag, samt hvem de må snakke med oppdraget samt hvilke ressurser som må samles inn for å gjennomføre oppdraget. Noen begrensninger vil forekomme med den tenkte løsningen. For eksempel vil det ikke være mulig for flere spillere å ta på seg samme oppdrag.

Oppgave 1.2

Tenkt database illustrert med ER-diagram:



Oversatt til relasjonsform:

Resource(ResourceID, Name, Description)
QuestGiver(QuestGiverID, Name)
Player(PlayerID, PlayerTag)
Mission(MissionID, Description, Status, ResourceToGet*, QuestGiverID*, PlayerID*)

SQL spørringer:

```
SELECT mission.*, player.PlayerTag FROM mission
LEFT JOIN
Player ON mission.playerID = player.PlayerID
```

MissionID	Description	Status	ResourceToGet	QuestGiverID	PlayerID	PlayerTag
1	Gather the scattered pieces of an ancient mirror	Active	1	1	1	Lumyno
2	Gather lumber for the towns storage	Completed	2	2	1	Lumyno
3	Gather preccious gemstones	Innactive	4	1	*NULL*	*NULL*
4	Go fishing	Active	3	2	2	Jormungander

Spørring som skriver ut listen over alle oppdrag kombinert med spiller listen for å vise navnet til spilleren som har tatt på seg oppdraget.

```
SELECT mission.*, player.PlayerTag, questgiver.Name FROM mission
LEFT JOIN
Player ON mission.playerID = player.PlayerID
LEFT JOIN
QuestGiver ON mission.questgiverID = questgiver.questgiverID
```

MissionID	Description	Status	ResourceToGet	QuestGiverID	PlayerID	PlayerTag	Name
1	Gather the scattered pieces of an ancient mirror	Active	1	1	1	Lumyno	Robert
2	Gather lumber for the towns storage	Completed	2	2	1	Lumyno	Laura
3	Gather preccious gemstones	Innactive	4	1	*NULL*	*NULL*	Robert
4	Go fishing	Active	3	2	2	Jormungander	Laura

Samme som forrige spørring bare at vi her i tillegg kombinerer med listen over QuestGivers for å kunne se oppdraget, hvem som er på det og hvem som har gitt det ut.

```
SELECT mission.*, player.PlayerTag, questgiver.Name, resource.Name, resource.Description
FROM mission
LEFT JOIN
Player ON mission.playerID = player.PlayerID
LEFT JOIN
QuestGiver ON mission.questgiverID = questgiver.questgiverID
LEFT JOIN
Resource ON mission.ResourceToGet = resource.resourceID
```

MissionID	Description	Status	ResourceToGet	QuestGiverID	PlayerID	PlayerTag	Name	Name	Description
1	Gather the scattered pieces of an ancient mirror	Active	1	1	1	Lumyno	Robert	Mirror Pieces	Small jagged pieces of a mirror
2	Gather lumber for the towns storage	Completed	2	2	1	Lumyno	Laura	Wood	A small piece of lumber
3	Gather preccious gemstones	Innactive	4	1	*NULL*	*NULL*	Robert	Gemstone	Highly valuable
4	Go fishing	Active	3	2	2	Jormungander	Laura	Fish	Its a fish

Samme som de to foregående, bare at det her også kobles med resource listen. Får både navn på ressursen og beskrivelsen dens fra ressurs tabellen.

Oppsummering:
Databasen opplevdes som veldig lett å bruke og det var helt uproblematisk å skrive spørringer opp mot databasen. Det kan likevel tenkes at visse forbedringer og videreutvikling kunne vært gjort. For eksempel kan ikke en spiller i den nåværende databasen bli belønnet med noen ting for å utføre oppdragene. En potensiell videreutvikling kunne da vært å legge til en tabell til som kunne representert inventaret til en spiller, på denne måten kunne man endret beholdningen til en gitt ressurs ved fullførte oppdrag.

Oppgave 1.3

XML kan for eksempel brukes til lagring av navnene til spillerne. For å vise denne funksjonaliteten legger vi først til en spiller til i listen, men denne gangen skriver vi navnet til spilleren på XML form.

INSERT INTO player VALUES (0, '<spiller><navn>TESTE NAVN</navn></spiller>')

	PlayerID	PlayerTag
Edit Copy Delete	1	Lumyno
Edit Copy Delete	2	Jormungander
Edit Copy Delete	3	AndyGamer
Edit Copy Delete	6	<spiller><navn>TESTE NAVN</navn></spiller>

Har nå da en liste bestående av de samme spillerne som Istad, men har i tillegg nå en spiller med navnet lagret på XML form. Videre kan vi da kalle på MySQL spørringen extract value for å få ut navnet på denne spilleren.

SELECT ExtractValue(playertag, '/spiller/navn') AS playerTag FROM player

playerTag
TESTE NAVN

Ser at vi nå ved å kalle på extract value har fått ut navnet på den spilleren som vi la inn med navnet på XML form.

De eventuelle fordelene og ulempene med å lagre dataene på XML kan for eksempel være at det er uavhengig av plattform, noe som lar alle lese dataene, enten de sitter på en windows eller linux maskin. I tillegg tillater xml en hierarkisk struktur, noe som lar oss se forholdene mellom forskjellige datatyper. En ulempe med å benytte XML kan for eksempel være at det fører til en høyere kompleksitet i dataene. Denne kompleksiteten kan også føre til et større overhead når dataene skal prosesseres.

Oppgave 1.4

Når det kommer til å evaluere om databasen hadde vært bedre egnet om den var stilt om til NoSQL form, er det en rekke faktorer som må tas i betraktning. For eksempel må man ta i betraktning skalerbarhetskravene til databasen. I den egendefinerte databasen i denne oppgaven har vi sett for oss et system for å holde styr på oppdrag man kan få i et spill, og da spesielt MMOs ettersom det er tenkt at det er flere forskjellige spillere som kan ta på seg oppdrag. I en MMO vil det være nødvendig å utvide og endre på disse oppdragene innenfor korte tidsfrister noe som NoSQL hadde hjulpet med. Enda et viktig argument for å bytte over til NoSQL omhandler fleksibiliteten og modulariteten en får av det. Om man tenker seg at spillet blir kjørt på servere som spillere kobler opp mot, er det også mulig å tenke at det kan være ønskelig å legge til flere maskiner i systemet om man er i en periode hvor mange spillere vil være aktive samtidig. En kan i tillegg da tenke at det også vil kunne være gunstig å redusere antall maskiner i perioder hvor det er mindre spillaktivitet. Alt i alt ville det dermed sannsynligvis vært bedre å bytte over til en NoSQL løsning.