DB til PRJ4

Til at opbevarer vores data til spillet og dennes loginsystem, skal vi benytte en database. Hertil skal der undersøges muligheder for at finde en database som gør det muligt for at bruge meget præcise tabeller med forhold til hinanden for at kunne genskabe et spil der er blevet gemt. Så derfor undersøges der fordele og ulemper ved SQL og NoSQL, og derefter ville der kunne vælges en af deres respektive applikationer, såsom MySQL (SQL) eller MongoDB (NoSQL).

**SQL:**

- Relational  
En samling af data med forudbestemte forhold derimellem. Genstande organiseres I tabeller som holder information om objektet.

- Kendt  
SQL er en kendt databasetype som vi modtager undervisning i.

- Rigide og fastlåste skemaer  
Ikke flexible og uden mulighed for ændringer efterhånden som kravene ændres.

- Vel dokumenteret  
SQL er en velkendt og gammel teknologi, med et stort fællesskab bag den til støtte og har derfor også gjort meget arbejde på SQL.

Fungere med ACID principperne.  
- Atomicity: Hver transaktion ses som en samlet unit, derfor vil transaktion enten være fuldført komplet, eller fejle, hvis en eller flere operationer fejler.

- Consistency: Kun gyldige data kan skrives i databasen. Hvis inputtet data er ugyldigt så ville databasen returnere til, hvordan den var før transaktionen, og pga. dette kan fejl transaktioner ikke korruptere databasen.

- Isolation: Ufærdige transaktioner forbliver isoleret. Dette medfører at alle transaktioner bliver behandlet individuelt.

- Durability: Data bliver gemt af systemet selvom en transaktion fejler, og pga. dette vil data ikke blive mistet ved at f.eks. systemet crasher under transaktionen.

- Portabel:  
SQL kan bruges i programmer på PC’er, servers, bærbare, uanset OS, og embedded systemer

- Hurtige query processering  
Store mængder af data hentes hurtigt og effektivt. Operationer som indsættelse, at slette og datamanipulation, sker hurtigt og effektivt

**NoSQL:**

- Flexible datamodeller:  
NoSQL databaser har typisk flexible skemaer, som gør det nemmere at lave ændringer i databasen eftersom kravene til denne ændres.

- Horizontal scaling:  
Ved en SQL-database bliver man nødt til at migrere til en større server, når man rammer den nuværendes kapacitet. I en NoSQL database kan man skalere horizontalt, ved at tilføje mindre ekstra servers, når det bliver nødvendigt.

- Hurtigere queries  
SQL Databaser er ofte normaliseret, så et query efter et enkelt objekt, kan skulle bruge flere joins, som bliver dyrere efterhånden som tabellerne vokser.  
NoSQL data er ofte opsat efter queries. Data der skal hentes sammen, er ofte opbevaret sammen.

Kilder:

<https://yalantis.com/blog/how-to-choose-a-database/>

<https://www.ibm.com/cloud/blog/sql-vs-nosql>

<https://www.mongodb.com/nosql-explained/nosql-vs-sql>

ER diagram