Valg af database.

Intro:

Vi er to personer i gruppen (exposee og integrator). Jeg (exposee) skal oprette en database, der inkluderer en bruger med forskellige rettigheder til databasen.

Databasen indeholder tre tabeller: users, teachers, og students. Jeg har tildelt integratoren brugere med følgende rettigheder:

1) Kan ikke læse q
2) Kan kun læse
3) Kan læse og skrive

Integratoren skal manipulere disse data i tabellen for at se, om brugerrettighederne er opfyldt.

Diskussion om valg af database:

Jeg vil begynde med at diskutere, hvorfor jeg har valgt denne specifikke database til at løse opgaven. Ordet 'granularity' betyder, at data er opdelt i mindre komponenter eller detaljer.
In-memory database: Redis DB

En Redis-database er en 'in-memory' database, hvilket betyder, at vores data gemmes i RAM og ikke lokalt på vores maskine. Dette indebærer, at vores data forsvinder, så snart jeg lukker min server ned. Dette kan potentielt skabe problemer, især når man arbejder med vigtige data.

Dog er det en god database, da den er ekstremt hurtig og kan bruges i projekter, hvor det ikke er nødvendigt at gemme data, men blot have det hurtigt tilgængeligt. Databasen bruger en key-value-opdeling, der deler data ned i mindre komponenter.

Mere specifikt eksempel: Lad os sige, at Redis-databasen indeholdt to nøgler: “skole1” og “elev1”. “skole1” skal indeholde nogle værdier såsom skolenavn, lokation, og elev1 skal indeholde navn og efternavn samt skolenavn.

For at kunne skabe en relation mellem skole og elev kræver det, at vi sætter en relation mellem de to nøgler. Dette kan godt lade sig gøre, men Redis er en "in-memory db", ikke en relationel database. Redis understøtter ikke den slags data. Til gengæld har vi en relationel database, der understøtter den slags datastruktur: MySQL-database.
Relationel database: MySQL DB

En MySQL-database er en database, der opdeler data i tabeller og kolonner. Disse tabeller af data kan have relationer til hinanden. For eksempel, hvis man har en tabel A, der repræsenterer 'skoler', og en anden tabel B, der repræsenterer 'elever', kan 'skole'-tabel indeholde mange 'elever'.

Grunden til, at dette bliver simplere, er at "elev"-tabellen har en foreign key, f.eks. "skole_id," der refererer til primærnøglen i "skole"-tabellen. Dermed opstår der en forbindelse mellem disse to tabeller, hvilket betyder, at hver elev i "elev"-tabellen er associeret med en specifik skole i "skole"-tabellen.

Konklusion:

In-memory-databaser som Redis er velegnede til situationer, hvor data skal være tilgængeligt konstant uden afbrydelser, som f.eks. på Twitter. I denne opgave, hvor dataen ikke behøver konstant adgang, men kun når der er behov for det, er en relationel database en bedre løsning sammenlignet med Redis.

Guide til integrator

Brugernavn	Password	Retigheder
school_admin	1234	CRUD
school_student	1234	Only read
student_parent	1234	Can not read

1. Forbind mysql serveren med en af ovenstående brugere
    a. Port: 3307
    b. IP: 165.22.70.135
    c. Du kan også connecte igennem terminalen brug denne command:
        i. mysql -h 165.22.70.135 -P 3307 -u [username] -p
            1. -h = host
            2. -P = port

3. -u = username
4. -p = prompt for password - behold tomt!

EER - diagram

Jeg vil gerne have at brugernes rolle, også er i overensstemmelse med hvad deres retigheder er. Opgaven siger at det skal være så granulær som muligt, derfor har jeg lavet deres retigheder helt ned på tabel level.

Sql-query til at grant en anden bruger retigheder:
GRANT SELECT (kolonne1, kolonne2….) ON table_name TO 'username'@'host';

- Bruger 1: school_admin skal betragtes som en rektor (CRUD, på users)
- Bruger 2: shcool_student skal betragtes som en elev (Read på users)
- Bruger 3: student_parnet skal betragtes som en forældre til eleven (no access)