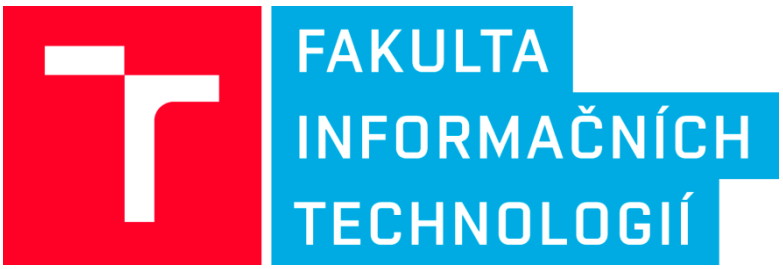


Pokročilé databázové systémy 2021/2022

Zadanie

návrh databázového systému

Patrik Trstenský, Bc (xtrste00)
Vosyka Pavel, Bc.(xvosyk00)

31.10.2021

1 Specifikace

Vytvoriť databázový systém pre sociálnu sieť, ktorá bude založená na príspevkoch, užívatelia budú môcť pridávať príspevky a komentovať ich. Tiež budú môcť príspevky hodnotiť bodmi od 0-10. Systém bude umožňovať zobrazovať najpopulárnejšie príspevky na platforme alebo príspevky od followerov.

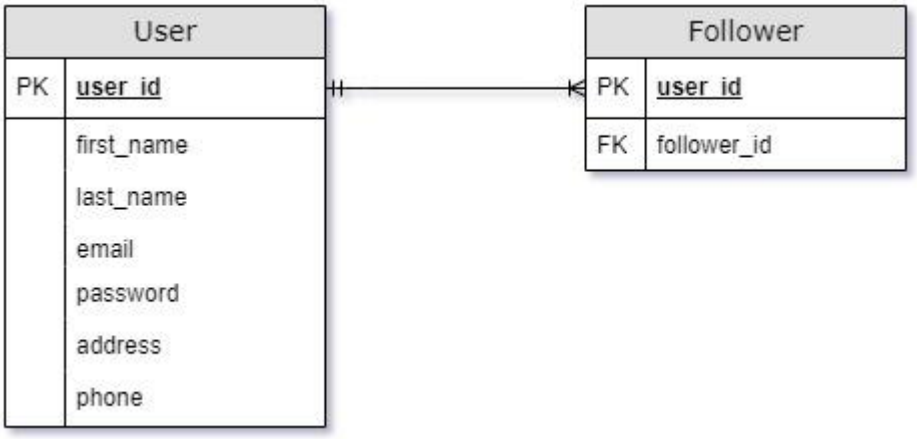
2 Analýza

Vlastnosti

vierohodnost a doba platnosti - nepotrebujeme mať čo najaktuálnejšie data
dostupnosť - chceme mať dobrou dostupnosť
miesta spotreby a užívatelia - česká republika
škálovatelnost- chceme mať dobrou škálovatelnost

Relačná databáze

Bude obsahovať tabuľku používateľov a tabuľku followerov kde používatelia budú mať svojich followerov. Relačná DB bude overovať originalitu emailovej adresy pri registrácií používateľov. Tiež bude obmedzujúce pravidlo na počet followerov ktorých môže mať používateľ (max 500 followerov, testovateľnosti).



Nerelačná databáza

Bude obsahovať USER, POST, COMMENT, RATING. Používatelia z relačnej DB budú skopírovaný do nerelačnej DB (USER) pre rýchle čítanie.

User

- user_id
- first_name
- last_name
- email - partition key
- address

- password
- phone
- followers

Post

- user_email - partition key
- time - clustering key
- title - index
- content
- file
- rating
- comment_count
- rating_count

Comment

- postID - partition key
- time - clustering key
- user_email - clustering key
- content

Rating

- postID - partition key
- value
- user_email - clustering key

Dotazy

Commands
vytvorenie užívateľa
vytvorenie príspevkov
vytvorenie komentára
vytvorenie ohodnotenia

úprava užívateľa
úprava príspevkov
úprava komentára
úprava ohodnotenia

vymazať užívateľa
vymazať príspevkov
vymazať komentára
vymazať ohodnotenia

Query
(vždy bude vracať príspevky a komentáre od najaktuálnejších)

vrátiť užívateľa
vrátiť komentáre príspevku
vrátiť zoznam followerov

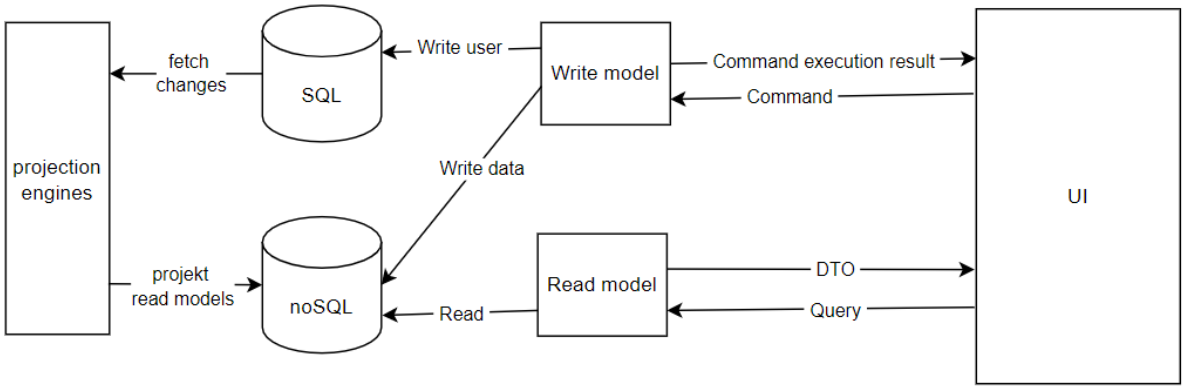
vrátiť príspevky priateľov
vrátiť moje príspevky
vrátiť príspevky podľa najlepšieho ohodnotenia
vrátiť príspevky podľa popularity (koľko obsahuje ohodnotenia)
vrátiť príspevky podľa množstva komentárov

Aktualizácia dát

Dáta vznikajú registráciou užívateľov, vytvorením príspevku, komentovaním príspevku a jeho ohodnotením. Pri pridaní/odstránení komentára musí byť aktualizovaný atribút comment_count v príspevku do ktorého komentár patrí. Podobne atribut rating_count sa aktualizuje pri pridaní/odstránení ohodnotenia pre daný príspevok. Aktualizuje sa atribút rating pri modifikácií/pridaní/odstránení, ktorý predstavuje priemerne hodnotenie príspevku.

3 Návrh

Ak sa bude jednať o zapisovanie Používateľa alebo follow používateľa najskôr ho zapíšeme do SQL DB aby sme overili podmienky, následne zapíšeme zmeny do noSQL DB. Ak však budeme chcieť zapisovať POST, COMMENT, RATING tak zapíšeme priamo do noSQL DB pretože nemáme žiadne obmedzujúce podmienky. Všetky dáta budeme čítať z noSQL DB.



4 Implementácia

spustenie

Spustenie aplikácie a cassandra databáze vykonáva docker. Spustenie pomocou príkazu **#docker-compose up**. Oracle databáza nie je súčasťou kontejneru, musí byť nakonfigurovaná samostatne v súboru **.oracleConnection**. Formát súboru je popísaný v **.oracleConnection.example**. Pro spustenie testov sa pripojte do kontajnera s aplikáciou a zadajte **./gradlew test**

Databázy

Pred použitím aplikácie je potrebné overiť či dané databázy majú vytvorené potrebné tabuľky, triggre.. Tieto DDL skripty sú definované pre Oracle databázu v súbore **oracle.sql** a pre Cassandru v **cassandra.cql**. Overovanie followerov je na úrovni relačnej databázy ktorá obsahuje trigger:

```
CREATE OR REPLACE TRIGGER max_followers
BEFORE INSERT ON soc_net_follower
FOR EACH ROW
DECLARE
      result int;
BEGIN
    SELECT COUNT(*) INTO result FROM soc_net_follower
    JOIN soc_net_user u ON (soc_net_follower.user_id = u.user_id)
    JOIN soc_net_user f ON (soc_net_follower.follower_id = f.user_id)
    WHERE u.user_id = :new.user_id;

      IF result >= 5 THEN
            raise_application_error(-20001, 'Max limit of followers!');
      END IF;
END;
/
```

Tiež sme potrebovali definovať pre cassandru špecialnu ďalšiu tabuľku **top_rating** ktorá v sebe bude držať hodnoty priemerného hodnotenia prispevku, kvôli tomu aby sme dokazali zo všetkých prispevkov zobrať ten ktorý má maximálne hodnotenie a dostali najlepšie hodnotený prispevok.

```
CREATE TABLE IF NOT EXISTS top_rating (
    user_email TEXT,
    rating DOUBLE,
    post_id UUID,
    PRIMARY KEY (post_id, rating)
) WITH CLUSTERING ORDER BY (rating DESC);
```

Potom z tejto tabuľky dostaneme jednoducho najlepšie hodnotený prispevok pomocou selectu:

```
SELECT * FROM top_rating LIMIT;
```

API

Všetky potrebné metódy na pracovanie so systémom obsahujú dve Triedy ktoré sú rozdelene podla návrhu na **QueryModel** a **WriteModel** kde **QueryModel** obsahuje všetky potrebné triedy na čítanie dát zo systému a **WriteModel** všetky metódy pre zápis do systému.

Query definované pri návrhu a ich API

`QueryModel.findUser(String email)`	vrátiť užívateľa
`QueryModel.getComments(Post post)`	vrátiť komentáre príspevku
`QueryModel.follow(User user)`	vrátiť zoznam followerov
`QueryModel.followsPosts(User user)`	vrátiť príspevky priateľov
`QueryModel.userPosts(User user)`	vrátiť moje príspevky

`QueryModel.bestRatedPosts()`
vrátiť príspevky podľa najlepšieho ohodnotenia

`QueryModel.popularPost(User user)`
vrátiť príspevky podľa popularity (koľko obsahuje ohodnotenia)

`QueryModel.mostCommentedPosts(User user)`
vrátiť príspevky podľa množstva komentárov

Testovanie

Testovanie prebieha pomocí frameworku junit.