VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJA

NERELIACINĖS DUOMENŲ BAZĖS

Situacijos analizė

Socialinis tinklas

Atliko: Tomas Šinkūnas
VU el. p.: tomas.sinkunas@mif.stud.vu.lt

Vadovas: Lekt. Giedrius Graževičius

Vilnius, 2022

Turinys

1. Užduotis

Jums tenka užduotis pagal formulę: *LSP numeris mod 6* : **2013297 mod 6 = 3**

Gautai situacijai įvertinkite:

- Kokį duomenų modelį pasirinksite (*reliacinį, rakto-reikšmės, dokumentų, stulpelių, grafų*)?
- Kokiomis savybėmis pasižymėti turi duomenų bazė (saugojimo duomenų modeliu (*Hash Table, B-Tree, LSM*), plėčiamumu, ACID garantijomis, ...)?
- Kokia iš žinomų duomenų bazių būtų tinkamiausia? Kokie jos trūkumai? Kokias savybes turėtų turėti ideali situacijai duomenų bazė.
- Pateikite fizinį duomenų modelį, pasirinktai duomenų bazei (stulpelius, indeksus, raktus, duomenų tipus ir t.t.). Taip pat pateikite duomenų schemos sukūrimo sakinius ir duomenų įterpimo pavyzdžius.

Situacija 3 – Socialinis tinklas.

- Socialiniame tinkle vartotojai gali talpinti įrašus.
- Kiti vartotojai po įrašais gali palikti komentarus ir “like’us”.
- Socialinis tinklas planuoja susilaukti milijonų aktyviai bendraujančių vartotojų.
- Socialiniam tinklui nėra labai svarbu ar kiekvienas vartotojas matys pačius naujausius duomenis.
- Socialinis tinklas nori būti atsparus sutrikimams ir pasižymėti dideliu pasiekiamumu.

2. Darbo eiga

2.1. Situacijos analizė

Socialiniai tinklai suteikia vartotojams galimybę susikurti savo socialinį tinklą skaitmeninėje erdvėje. Socialinis tinklas – tai mazgų (angl. *nodes*) ir ryšių (angl. *links*) grupė, kurią sudaro socialiniai subjektai, kuriuose mazgai atstovauja socialinius subjekus, tokius kaip žmonės ar organizacijos. Ryšiai atspindi jų santykius, pvz., draugystę ar prekybinius santykius. (Khan, 2015)

Socialiniai tinklai leidžia paprastai ir greitai užmegzti, plėsti ir palaikyti ryšius subjektams ir dėl to labai greitai pritraukia daugybę vartotojų. Dėl didelio naudotojų skaičiaus augimo sparčiai ir milžiniškai išaugo duomenų kiekiai, su kuriais sistema turi susitvarkyti. To pasekoje socialiniai tinklai susidūrė su didelių duomenų keliamais iššūkiais, visų pirma precedento neturinčiais operacijų kiekiais bei lūkesčiais dėl lėtos prieigos prie didžiulių duomenų rinkinių.

2004 metų pradžioje prasidėjo Facebook era, kuri iš kart tapo karšta tema, ir jau metų pabaigoje sistema turėjo daugiau nei milijoną registruotų vartotojų. Nuo pat pradžių Facebook rėmėsi reliacine MySQL duomenų baze kaip duomenų saugojimu. Jie pradėjo nuo InnoDB ir vėliau sukūrė MyRocksDB, kuri galiausiai buvo naudojama kaip MySQL duomenų bazės variklis.

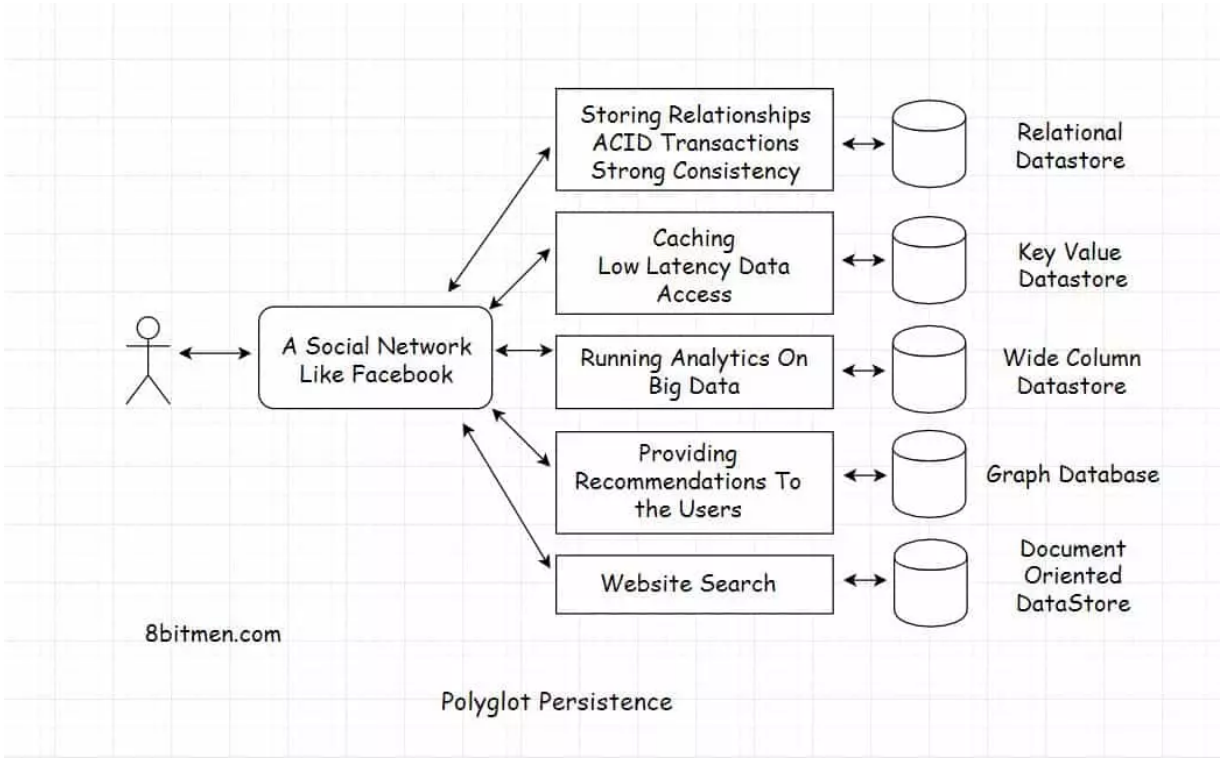
2008 metais kompanija sukūrė ir paviešino noSQL tipo duomenų bazę Cassandra, kurią vėliau pakeitė su HBase. Didžiųjų duomenų (angl. *Big Data*) valdymui Facebook šiuo metu naudoja Apache Hadoop, HBase, Hive, Apache Thrift, PrestoDB, MongoDB ir Memcached duomenų bazes, nors MySQL vis dar naudojamas struktūriniams duomenims, pvz., sienų įrašams, vartotojo informacijai, laiko juostai ir panašiai, saugoti.

Twitter savo egzistavimą pradėjo 2006 metais ir taipogi naudojo MySQL duomenų bazę. Tačiau, vartotojų skaičiui augant, kompanija sukūrė paskirstytą duomenų bazę pavadinimu Manhattan. Šiuo metu sistema naudoja kelių tipų duomenų saugojimui ir valdymui:

- MySQL ir Manhattan yra pagrindinės duomenų saugyklos, skirtos vartotojo duomenims saugoti.
- Memcache, Redis – lokaliam talpyklos saugojimui (*caching*)
- FlockDB – socialinių ryšių grafui saugoti
- MetricsDB – platformos duomenų metrikoms saugoti
- Blobstore – vaizdams, vaizdo įrašams ir dideliems dvejetainiams objektams saugoti.

Nuo 2010 pradžios Instagram socialinis tinklas sparčiai augo ir vos po 2 mėnesių turėjo vieną milijoną vartotojų. Siekdami užtikrinti tinkamą duomenų saugojimą tokiam didžiuliam vartotojų augimui, Instagram sujungė reliacinę duomenų bazę – PostgresSQL – su NoSQL duomenų bazėmis Redis ir Cassandra, taip išnaudodami kiekvienos technologijos pranašumais kiekvienu atveju.

Iš socialinių tinklų milžinų praktikos matome, kad kaip sistemos pagrindas dažniausiai yra pasirenkamos reliacinės duomenų bazės. Sistemos veikimo pagerinimui, didžiųjų duomenų saugojimui, patikimumo bei prieinamumo užtikrinimui yra papildomai naudojamos ir nereliacinės duomenų bazės. Rezultate gauname "poliglotines" sistemas, kurios naudoja skirtingus tipus skirtingoms užduotims atlikti (1 paveikslas).



1 pav. Skirtingų duomenų bazių naudojimas socialiniuose tinkluose. Šaltinis: https://scaleyourapp.com/what-database-does-facebook-use-a-1000-feet-deep-dive/

MySQL, kartu su grafų tipo duomenų bazėmės yra realizuotos vartotojų santykiams valdyti – sekti ir registruoti vartotojo veiksmus, tokius kaip kas kieno įrašą pa'like'ino, bendri draugai, draugų draugai kurie seka vieną ar kitą draugą, ir panašiai.

2.2. Duomenų modelis

Mūsų modeliuojamame socialiniame tinkle numatoma saugoti tik informaciją apie vartotojus, jų komentarus, įrašus bei like'us – tai pakankamai paprasta duomenų struktūra, atspindinti dokumentų modelio tipą. Sistema taip pat nereikalauja sudėtingų ryšių tarp esybių – mes nesiruošiame sekti ryšių tarp atskirų vartotojų, dėl to galime atsisakyti MySQL ar grafų tipo duomenų bazių.

Jeigu mūsų nagrinėjama užduotis turėtų užtikrinti ACID reikalavimus, pavyzdžiui, atlikinėtume finansines operacijas, tai MySQL duomenų bazė mums tiktų. Kita vertus, jei reikėtų greitos prieigos prie duomenų, vertėtų rinktis raktas-reikšmė Memcache arba Redis duomenų bazes. Tačiau mūsų atveju yra reikalinga greita ir globaliai prieinama duomenų bazė. Mus tenkina duomenų denormalizavimas, kuris galiausiai susinormalizuos, dėl to, manau, NoSQL sprendimas būtų tinkamiausias šiai užduočiai atlikti.

NoSQL (paprastai vadinamas „Ne tik SQL“) yra kitokia duomenų bazių sistema negu SQL, leidžianti efektyviai ir greitai apdoroti informaciją didžiuliu mastu. Kitaip tariant, tai yra infrastruktūra, kuri yra pritaikyta dideliems duomenų poreikiams, kas ir yra būdinga socialiniam tinklui.

2.3. Duomenų bazės savybės

Pagal užduotį, mūsų socialinis tinklas planuoja sulaukti milijonų aktyviai bendraujančių vartotojų, tad duomenų bazė turėtų pasižymėti lankstumu, plečiamumu ir galimybe keisti mastelį. Be to, tinklui yra svarbu būti atspariam trikiams bei turėti didelį pasiekiamumą, tad duomenų bazė turėtų pasižymėti atsparumu.

Socialiniuose tinkluose dažniausiai vyksta daugiau skaitymo operacijų negu rašymo, dėl to yra verta rinktis B+ duomenų modelį, nes jis skaitymo operacijas atlieka greičiau negu LSM modelis.

Kadangi tinkle neplanuojama daryti finansinių transakcijų, dėl to BASE modelio pilnai užtenka. Jis užtikrins sistemos prieinamumą (angl. *Basically Available*), rašymo operacijoms vientisumas nėra būtinas (angl. *Soft State*), o duomenys galiausiai susinormalizuos (angl. *Eventually Consistent*). Pagal CAP paradigmą duomenų bazė turi užtikrinti AP – turi būti prieinama (angl. *available*) ir atspari particijoms (angl. *Partition tolerance*).

2.4. Duomenų bazė

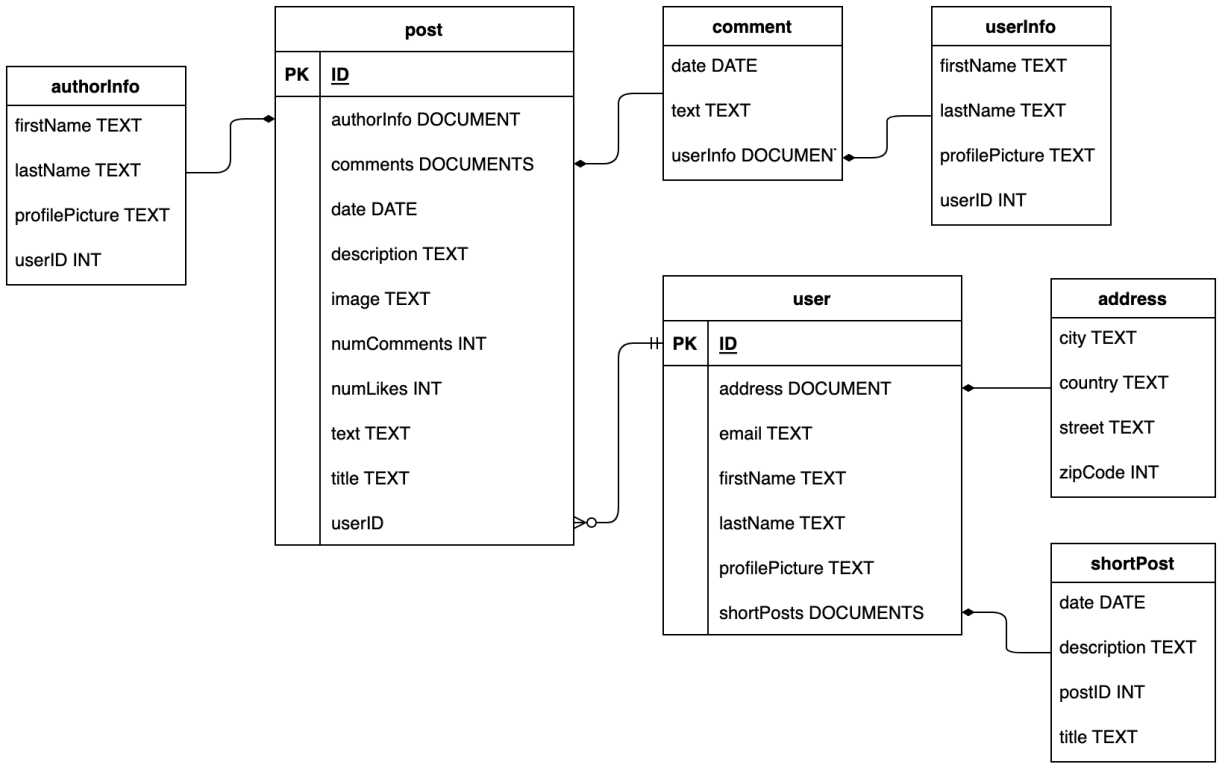
Tinkamiausia nereliacinė duomenų bazė šiai užduočiai atlikti būtų MongoDB. Ji atitinka visas reikiamas savybes (*2.3. Duomenų bazės savybės*) bei turi platų programavimo kalbų palaikymą (C, C++, C#, Go, Java, Node.js, PHP, Python, Ruby, Rust, Scala, Swift). Kaip dokumentų duomenų bazė, MongoDB leidžia saugoti tiek struktūrizuotus, tiek nestruktūrizuotus dokumentų modelius į JSON panašiu formatu.

Be to, MongoDB naudoja daug žinomų kompanijų, tarp kurių yra Forbes, Toyota, Verizon ir netgi Facebook socialinis tinklas.

Vienas iš pagrindinių trūkumų – tai nenuolatinis nuoseklumas. Gali būti situacijų, kai skirtingi vartotojai mato skirtingą informaciją. Tačiau tai ne pačios MongoDB trūkumas, o CAP teoremos, kurioje teigiama, kad paskirstytoje duomenų bazėje sutrikus tinklui, galima užtikrinti nuoseklumą arba prieinamumą, bet ne abu.

2.5. Fizinis duomenų bazės modelis

Fizinį duomenų bazės modelį sudarytų dvi kolekcijos bei 5 įterptiniai dokumentai (2 paveikslas).



2 pav. Fizinis duomenų bazės modelis

- *Post* – įrašų kolekcija,
- *User* – vartotojų kolekcija
- *authorInfo* – įdėtinis (angl. *embeded*) dokumentas, saugantis trumpą įrašo autoriaus informaciją,
- *userInfo* – įdėtinis dokumentas, saugantis trumpą komentaro autoriaus informaciją,
- *Address* – įdėtinis dokumentas
- *Comment* – įdėtinis dokumentas, saugantis įrašo komentarą
- *ShortPost* – įdėtinis dokumentas, saugantis trumpą įrašo informaciją

Nors *authorInfo* ir *userInfo* iš pažiūros saugo tą pačią informaciją, atrodytų, jog jie gali būti apjungti į vieną dokumentą. Tačiau to nevertėtų to daryti, nes jie priklauso skirtingoms kolekcijoms ir skelbimo autoriaus informacija nebūtinai turi sutapti su komentaro autoriaus informacija. Čia tik šiuo atveju taip atrodo, jog abiejose esybėse yra saugoma ta pati informacija – tai tiesiog sutapimas.

2.6. JSON duomenų struktūra

Fizinis duomenų bazės modelis, paverstas į JSON notaciją, naudojamą MongoDB duomenų bazėse, atrodytų taip:

- Address:

```
{
  city: String,
  country: String,
  street: String,
  zipCode: Number,
}
```

- AuthorInfo:

```
{
  firstName: String,
  lastName: String,
  profilePicture: String,
  userID: Number,
}
```

- Comment:

```
{
  date: Date,
  text: String,
  userInfo: UserInfo,
}
```

- Post:

```
{
  _id: Number,
  authorInfo: AuthorInfo,
  comments: Comment[],
  date: Date,
  description: String,
  image: String,
  numComments: Number,
```

```
  numLikes: Number,
  text: String,
  title: String,
  userID: Number,
}
```

- ShortPost:

```
{
  date: Date,
  description: String,
  postID: Number,
  title: String,
}
```

- User:

```
{
  _id: Number,
  address: Address,
  email: String,
  firstName: String,
  lastName: String,
  profilePicture: String,
  shortPosts: ShortPost[],
}
```

- UserInfo:

```
{
  firstName: String,
  lastName: String,
  profilePicture: String,
  userID: Number,
}
```

2.7. Užklausos

- Duomenų bazės sukūrimas:

```
const client = new MongoClient(DB_URL);
const db = client.db(DATABASE_NAME);
```

- Kolekcijos sukūrimas:

```
const collection = db.collection(COLLECTION_NAME);
```

- Naujo vartotojo sukūrimas User kolekcijoje:

```
const user = {
 address: {
   city: 'Klaipėda',
   country: 'Lithuania',
   street: 'Žuvėdrų 100',
   zipCode: 1234,
 },
 email: 'dede_vania@gmail.com',
 firstName: 'Dėdė',
 lastName: 'Vania',
 profilePicture: 'https://randomuser.me/api/portraits/man/10.jpg',
 shortPosts: [],
};

user._id = (await User.insertOne(user)).insertedId;
return user;
```

- Naujo įrašo sukūrimas Post kolekcijoje:

```
const post = {
 authorInfo: {
   firstName: user.firstName,
   lastName: user.lastName,
   profilePicture: user.profilePicture,
   userID: user._id,
 },
 comments: [],
 date: Date.now(),
 description: 'Description of the post',
 image: 'image url',
 numComments: 0,
 numLikes: 0,
 text: 'Long post text.',
 title: 'Post title',
 userID: user._id,
};

post._id = (await Post.insertOne(post)).insertedId;
return post;
```

- Įrašo įkėlimas į vartotojo porfolio:

```
return (
 await User.findOneAndUpdate(
   { _id: user._id },
   {
     $push: {
       shortPosts: {
         date: post.date,
         description: post.description,
         postID: post._id,
         title: post.title,
       },
     },
   },
   { returnDocument: 'after' }
 )
).value;
```

- Naujo komentaro sukūrimas Post kolekcijoje:

```
return (
 await Post.findOneAndUpdate(
   { _id: post._id },
   {
     $inc: { numComments: 1 },
     $push: {
       comments: {
         date: Date.now(),
         text: 'this is some awesome post. I love it!',
         userInfo: {
           firstName: user.firstName,
           lastName: user.lastName,
           profilePicture: user.profilePicture,
           userID: user._id,
         },
       },
     },
   },
   {
     returnDocument: 'after',
   }
 )
```

```
).value;
```

- ‘Like’ kiekio rodiklio atnaujinimas Post kolekcijoje:

```
return (
await Post.findOneAndUpdate(
  { _id: post._id },
  { $inc: { numLikes: 1 } },
  { returnDocument: 'after' }
)
).value;
```

- Jeigu norėtume įterpti keletą įrašų vienu metu, tai užklausa būtų tokia:

```
const result = await collection.insertMany(DATA_ARRAY);
```

3. Naudoti šaltiniai

ACID vs. BASE: Comparison of Database Transaction Models. (2020, November 25). phoenixNAP. Retrieved October 28, 2022, from phoenixnap.com/kb/acid-vs-base

Comparing LSM-Tree with B-Tree. (2022, February 1). LinkedIn. Retrieved October 28, 2022, from .linkedin.com/pulse/comparing-lsm-tree-b-tree-sameh-muhammed

Facebook Database [Updated] – A Thorough Insight Into The Databases Used @Facebook. (n.d.). Scaleyourapp. Retrieved October 28, 2022, from https://scaleyourapp.com/what-database-does-facebook-use-a-1000-feet-deep-dive/

Gašpar, D., & Mabić, M. (2017, September). NoSQL Databases as Social Networks Storage Systems.

Instagration Pt. 2: Scaling our infrastructure to multiple data centers. (n.d.). Instagram Engineering. Retrieved October 28, 2022, from https://instagram-engineering.com/instagration-pt-2-scaling-our-infrastructure-to-multiple-data-centers-5745cbad7834

Khan, G. F. (2015, July 2). Seven Layers of Social Media Analytics: Mining Business Insights from Social Media Text, Actions, Networks, Hyperlinks, Apps, Search Engine, and Location Data. Amazon.com. Retrieved October 28, 2022, from https://www.amazon.com/Seven-Layers-Social-Media-Analytics/dp/1507823207

Lo, F. (n.d.). What is Hadoop? What is NoSQL? What is MapReduce? DataJobs.com. Retrieved October 28, 2022, from https://datajobs.com/what-is-hadoop-and-nosql
What Database Does Twitter Use? - A Deep Dive. (n.d.). Scaleyourapp. Retrieved October 28, 2022, from scaleyourapp.com/what-database-does-twitter-use-a-deep-dive/

Storing Comments — MongoDB Manual. (n.d.). MongoDB手册. Retrieved November 10, 2022, from https://mongodb-documentation.readthedocs.io/en/latest/use-cases/storing-comments.html#gsc.tab=0