Predavanje 2-9

Objektne baze podataka

S obzirom da se interakcija sa bazama podataka najčešće vrši koristeći programerski interfejs, tako da krajnji korisnik ne koristi bazu podataka već koristi neki program koji preko programskog interfejsa spašava i vraća podatke u bazu. Kako bi se ova interakcija izvršila, programi koriste SQL upite kako bi unijeli i vratili podatke iz baze kao i imali i ostalu interakciju.

Pisanje SQL upita, te parsiranje rezultata koristeći programski jezik može biti vrlo komplikovano. Još jedan nedostatak relacionih baza podataka je potreba za modeliranjem komplikovanih veza između tabela. Kako je tehnologija napredovala, prostor za pohranu podataka se povećao mnogostruko u odnosu na period kada su relacione baze podataka izumljene.

Prilikom izrade nekih programa, postoji potreba da se sačuvaju objekti koje program koristi. Za ovu svrhu su napravljene objektne baze podataka. One se još nazivaju dokumentne baze podataka.

U objektnim bazama podataka, podaci su spašeni kao upravo isti objekti kao u programskim jezicima i potreba za strukturom je smanjena ili uklonjena. Tako da objektne baze nemaju tabele, već imaju objekte i kolekcije objekata. Jedna vrsta objektne ili dokumentne baze je MongoDB.

MongoDB

MongoDB[5] ili jednostavno Mongo, je objektna baza podataka koja objekte spašava u kolekcije. Objekti imaju atribute i njihove vrijednosti. Vrijednosti atributa objekta mogu ponovo biti objekti.

Primjer jednog objekta u Mongu je:

```
{
   "_id": ObjectId("507f191e810c19729de860ea"),
   "ime": "Alisa",
   "prezime": "Alisovic",
   "razred": 3,
   "upis": ISODate("2021-12-19T06:01:17.171Z"),
   "predmeti": ["Matematika", "Fizika", "Hemija"],
   "prosjek": 3.14,
   "izborni_predmet": null,
   "izgled": { oci: "plave", visina: 160, boja_kose: "crna" }
}
```

Osnovna karakteristika objekata je da imaju naziv atributa, koji je string, i vrijednost atributa koji najčešće mogu biti brojevi, stringovi, datumi, null-vrijednosti, nizovi, objekti, koji će biti u fokusu ali mogu biti i drugi koji nisu spomenuti ovdje ali jesu u dokumentaciji ove baze podataka.

[5] https://www.mongodb.com/

Interakcija sa MongoDB se može vršiti kroz komandnu liniju, programski, ili koristeći Robo 3T[6] aplikaciju koja omogućava pisanje komandi. U primjerima će biti korišten Robo 3T.

Kada se koristi Robo 3T, ne koristi se mongo shell kao interfejs, već MongoDB-ov upitni jezik. Osnovni objekat nad kojim se pozivaju metode se zove db. Tako npr.:

```
db.adminCommand( { listDatabases: 1 } );
```

će vratiti listu baza u našoj instalaciji baze podataka. adminCommand je metoda za cijeli spektar administratorskih operacija za bau podataka. Osnovne interakcije poput kreiranja baze i kolekcija će se obavljati kroz Robo 3T GUI, tako da će glavni fokus biti na upitni jezik.

Tako npr. da dobavimo sve studente iz kolekcije, možemo napisati:

```
db.getCollection('studenti').find({});
```

ili

```
db.studenti.find({});
```

U prvom slučaju koristimo metodu `getCollection()`, a u drugom jednostavno koristimo atribut nad objektom `db`. Ova dva pristupa su potpuno jednaka, i drugi je tu radi lakšeg korištenja.

Kolekcija je također objekat, nad kojim se može primjeniti niz funkcija ili metoda. Umjesto komandi i klauzula kao u SQL jeziku ovdje se pozivaju funkcije da obave neku funkciju. Tako za osnovne funkcije imamo `find` `insert` i `remove`. Tako npr. za ubacivanje studenta u kolekciju, koristi se:

```
db.studenti.insert({ ime: 'Denis', prezime: 'Prezime' });
```

Da bi vidjeli da je objekat zaista unesen u kolekciju, možemo koristiti metodu `find`. Za pretragu se koristi objekat u kojem se naznače polja i njihove vrijednosti, a kao rezultat se dobiju svi objekti koji imaju ta polja koja imaju tu vrijednost.

```
db.studenti.find({ ime: 'Denis' });
```

Rezultat:

```
{
    "_id" : ObjectId("6207920285b9ad817857d8d1"),
    "ime" : "Ime",
    "prezime" : "Prezime"
}
```

[6] https://robomongo.org/

ObjectId

Primjećuje se da objekat ima dodatno polje u odnosu na objekat koji je unesen u bazu podataka. To polje je `"_id"` i predstavlja **primarni ključ**. Tip ovog ključa je ObjectId. U SQL bazama imali smo id koji je bio `INT` i uvećavao se za svaki novi unos. Međutim, u MongoDB, ObjectId je kombinacija nasumično odabranih brojeva i datuma kada je unos napravljen.

ObjectId se sastoji od 12 bajti, gdje prvih 4 označavaju broj sekundi od UNIX epohe[7] naredna 3 bajta označavaju ID mašine, a naredna 2 ID procesa. Ova kombinacija omogućava dovoljnu nasumičnost kako se ne bi desilo preklapanje. Dodatni plus je što nam prva 4 bajta daju vrijeme kada je objekat kreiran. Za vođenje evidencije kada je objekat modifikovan će biti potrebno koristiti neko drugi atribut.

Struktura

U poređenju sa SQL bazama podataka, objektne baze podataka nemaju fiksiranu strukturu. To znači da jedan objekat može imati jednu listu atributa, a drugi objekat drugu listu atributa u istoj kolekciji. Sa organizacijske tačke gledišta imati objekte sa potpuno različitim atributima, nema smisla, ali imati objekte sa istom baznom skupinom atributa, gdje neki objekti mogu imati dodatne atribute različite, može biti vrlo korisno. Tako npr. možemo dodati još neki objekat, sa drugom listom atributa:

```
db.studenti.insert({ ime: 'Drugo ime', nadimak: 'dj drugi' });

db.studenti.find({});
{
    "_id" : ObjectId("6207920285b9ad817857d8d1"),
    "ime" : "Denis",
    "prezime" : "Prezime"
},
{
    "_id" : ObjectId("6207a14885b9ad817857d8d2"),
    "ime" : "Drugo ime",
    "nadimak" : "dj drugi"
}
```

Vidimo da u istoj kolekciji imamo oba objekta ali sa različitom strukturom. Na ovaj način, eleminiše se potreba za organizacijom tabela i dizajniranjem baze tako detaljno kao što je rađeno u SQL bazama. Za brisanje objekta možemo koristiti metodu remove:

```
db.studenti.remove({ "_id" : ObjectId("6207a14885b9ad817857d8d2") });
```

Nedostatak objektnih baza je loša podrška za transakcije, te iz tog razloga, prihvaćeno je dupliranje podataka kako bi se izbjegla potreba za transakcijama.

[7] Početak UNIX epohe 00:00:00 UTC on 1 January 1970