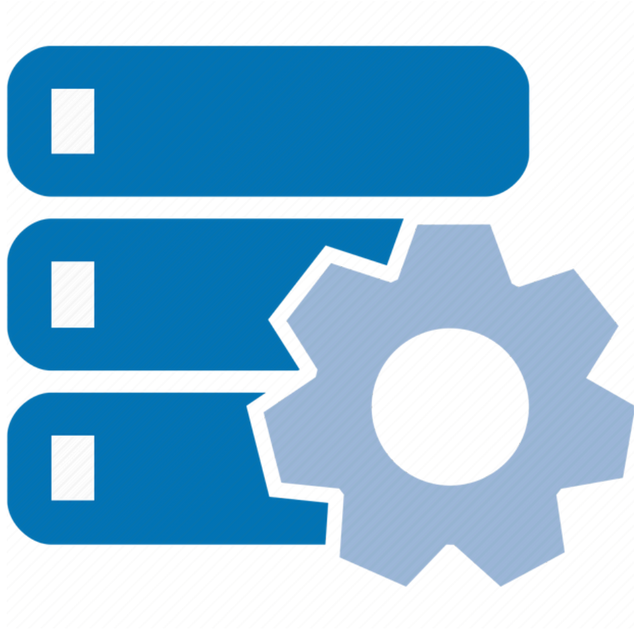
Administriranje baza podataka

Key-value baza podataka



Smerovi: SRT, KOT

Predmetni profesor: Dr Dušan Stefanović, prof. struk. studija

Predmetni asistenti: Nevena Minić, Nikola Vukotić

ODSEK NIŠ
AKADEMIJA
TEHNIČKO-VASPITAČKIH
STRUKOVNIH STUDIJA

Cilj vežbe

1. **Razumevanje koncepta key-value modela i strukturiranja podataka:** Upoznavanje sa osnovnim principima ključ-vrednost (key-value) modela, gde se podaci čuvaju kao parovi ključeva i vrednosti, bez fiksne šeme. Razumevanje načina na koji se brzo pristupa podacima na osnovu ključa i kako se podaci organizuju u memoriji.
2. **Kreiranje i upravljanje key-value bazama podataka (Redis):** Razumevanje kako koristiti Redis kao in-memory key-value bazu, uključujući rad sa osnovnim tipovima podataka (stringovi, liste, skupovi, hash-evi, z-setovi), kao i komande za rad s podacima. Upoznavanje sa radom pub/sub mehanizmom.
3. **Razumevanje prednosti i kada koristiti key-value baze podataka:** Upoznavanje sa slučajevima kada je key-value baza pogodnija od relacionih baza, posebno za aplikacije koje zahtevaju izuzetno brz pristup podacima, keširanje rezultata, brojanje događaja u realnom vremenu, upravljanje sesijama, ili rad sa podacima koji se brzo menjaju. Razumevanje Redis optimizacija i načina upotrebe kao cache-layer ili glavna baza u određenim scenarijima.

Key-value stores

- Key-value paradigma za pristup podacima obezbeđuje dobre performanse i dostupnost podataka
- Jedna vrednost, jedan ključ, nema duplikata, izuzetno brzo
- Skaliranje ogromnih količina podataka
- Projektovane da podnesu velika opterećenja
- Ključevi i vrednosti mogu biti kompleksni objekti
- Konzistentnost podataka se može primeniti samo na operacije koje se odnose na jedan ključ
- Bazira se na hash tabeli – algoritmu koji omogućava da se za ključ brzo pronađe odgovarajuća vrednost.
- U distribuiranim sistemima, podaci se mogu raspodeliti po više čvorova, ali pristup je uvek preko ključa.

- Key-value store je jednostavan model baze podataka u kojem se podaci čuvaju kao parovi:
  - Ključ (Key) → jedinstveni identifikator
  - Vrednost (Value) → podatak (bilo kog oblika)
- Može se zamisliti kao digitalni rečnik:
  - 🔑 **username_123** → 📦 "**Ana Jovanović**"
  - 🔑 **cart_001** → 📦 ["**jogurt**", "**mleko**"]
- Redis je jedan od najpoznatijih Key-value sistema

Key-value stores - prednosti

1. ⚡ **Efikasna pretraga po ključu**
   - Pristup je trenutan: tačan ključ je poznat, pa nema potrebe za traganjem po svim redovima. Minimalne latencije čitanja i pisanja.
   - Npr: **GET user:001** je trenutna operacija.
2. 🌍 **Distribucija u klasterima**
   - Lako se širi na više servera: Redis klaster deli podatke po ključevima.
   - Npr: **korisnici A–F** na čvoru **1**, **G–L** na čvoru **2**...
3. 🧩 **Fleksibilna šema**
   - Svaki ključ može imati drugačiju strukturu podataka.
   - Nema potrebe za fiksnim kolonama kao u SQL:

Ključ	Vrednost
`user:001`	`{ ime: "Ana", email: "ana@mail.com" }`
`product:45`	`{ naziv: "Telefon", cena: 350 }`

4. 💾 **Memorijska efikasnost**
   - Čuvaju se samo popunjeni atributi.
   - Nema praznih kolona kao u SQL-u (NULL vrednosti).
5. 🛑 **Keširanje rezultata iz SQL-a**
   - Redis se često koristi za keširanje rezultata iz sporijih relacijskih upita.
   - Npr. **GET user:123** čuva podatke koji su izvučeni iz SQL baze.

Key-value stores - nedostaci

1. ❌ **Nema kompleksnih upita**

```
HSET user:001 ime "Ana" grad "Niš"
HSET user:002 ime "Marko" grad "Beograd"
HSET user:003 ime "Ivana" grad "Niš"
```

   - Nemoguće je postaviti upit: “Pronađi sve korisnike iz Niša“ – jer je potrebno znati tačne ključeve
   - Rešenje: koristiti druge baze ili indeksirati to aplikativno
2. ❌ **Nema stranih ključeva**
   - Ne može se reći: “Ovaj proizvod pripada ovom korisniku direktno”
   - Aplikacija mora sama da poveže te podatke
3. ❌ **Nema standardizovanog jezika**
   - Nema SQL jezika – koriste se komande (npr. SET, GET, ZADD...)
   - Svaka baza može imati svoj API (Redis, Riak, itd.)
4. 📉 **Ograničeno trajno skladištenje (RAM + periodični disk)**

Redis

Redis (REmote DIctionary Server) je visoko-performantna open-source baza podataka zasnovana na **key/value** modelu, koja funkcioniše kao **in-memory data structure server** – čuva podatke u RAM-u radi izuzetno brzog pristupa i podržava bogat spektar struktura vrednosti.

🔍 **Karakteristike:**

- 🛠️ Razvila: **Redis Labs / Salvatore Sanfilippo (antirez)**
- 📅 Godina nastanka: **2009**
- 💻 Jezik implementacije: **ANSI C** (visoka portabilnost)
- 📄 Licenca: **BSD (open-source)**
- 🧠 Ime: **Redis = REmote DIctionary Server**
- 🏆 Najpopularnija Key/Value baza (i šire: koristi se i za caching, messaging, rang liste itd.)
- 🌐 Zvanični sajt: https://redis.io/
- 📚 Dokumentacija: https://redis.io/docs/latest/

Redis

🔢 Arhitektura i ponašanje:

- 💾 **In-memory storage**: Svi podaci se primarno čuvaju u memoriji (RAM) → vrhunske performanse
- 🧱 **Perzistencija (opciona)**:
  - 🗂️ Snapshotting (RDB) – periodično čuvanje celokupnog stanja na disk
  - 📜 Append-only File (AOF) – zapisivanje svake operacije, komande na disk (kao log)
- 🧮 **Jedan proces, jedna nit**: Redis server radi na jednoj niti, ali ekstremno brz
- 🔁 **Sinhronizacija memorije i diska**: po default-u svake 2 sekunde (podesivo)
- ⚠️ **Potencijalni gubitak podataka**: ako nije bilo sinhronizacije pre rušenja, podaci iz RAM-a mogu biti izgubljeni

🔑 Ključevi u Redis-u:

- ✅ **Binary safe** – bilo koji niz bajtova je validan ključ (npr. string, čak i binarni podaci)
- 🔒 **Preporuke za ključeve:**
  - ✅ Maksimalna veličina: 512MB
  - 🧠 Preporučeno: čitljivi stringovi (user:123, session:456)
  - ⛔ Izbegavati predugačke ključeve (usporava pretragu)
  - ⛔ Izbegavati prekratke ključeve (loša čitljivost)

Redis - tipovi podataka

Tip	Opis i upotreba
📝 **String**	Najosnovniji tip: tekst, broj, JSON, tokeni...
📜 **List**	Redosledna kolekcija – koristi se za redove i stekove
🎯 **Set**	Neuređeni skupovi jedinstvenih elemenata
📊 **Sorted Set (ZSet)**	Skup sa score vrednostima – koristi se za rang-liste
🧠 **Hash**	Slično JSON-u ili rečniku – kolekcija ključ-vrednost parova unutar jednog ključa
🧮 **Bitmap**	Za binarne operacije – DA/NE tip podataka po indeksima
🧮 **HyperLogLog**	Statistički model za brojanje jedinstvenih elemenata sa minimalnom memorijom
📢 **Pub/Sub**	Mehanizam za slanje i primanje poruka između servisa

Redis - tipovi podataka i operacije

🧵 1. **String (najjednostavniji tip podataka u Redis-u)**

- ✅ Binary-safe
- 📏 Maksimalna veličina: **512 MB**
- 📌 Tipične operacije:
  - SET – postavlja vrednost
  - GET – pribavlja vrednost
  - INCR, INCRBY – atomično inkrementiranje broja
  - DECR, DECRBY – atomično dekrementiranje
  - GETSET – menja vrednost i vraća prethodnu
  - MSET, MGET – rad sa više ključeva odjednom

📋 2. **Lista (List)**

- 🧬 Implementirana kao **lančana lista stringova**
- ✅ Efikasne operacije na početku i kraju liste
- 🧭 Redosled dodatih elemenata se čuva
- 📌 Tipične operacije:
  - LPUSH / RPUSH – dodavanje elemenata levo/desno
  - LPOP / RPOP – uzimanje i uklanjanje elementa
  - LRANGE – vraća podlistu
  - LTRIM – čuva samo deo liste
  - LLEN – broj elemenata u listi
  - LREM – briše po vrednosti i broju ponavljanja
  - LINSERT – umetanje pre/posle elementa

Redis - tipovi podataka i operacije

🎯 3. **Set (Skup)**

- 🧩 Neuređena kolekcija jedinstvenih stringova
- ❌ Duplikati nisu dozvoljeni
- 📌 Tipične operacije:
  - SADD – dodaj element(e)
  - SMEMBERS – svi članovi skupa
  - SISMEMBER – da li element postoji?
  - SREM – uklanjanje iz skupa
  - SRANDMEMBER, SPOP – slučajan element (sa ili bez uklanjanja)
  - SINTER, SUNION, SDIFF – skupovne operacije (presek, unija, razlika)

📊 4. **Hash (Haš tabela)**

- 🔐 Kolekcija **atribut/vrednost** parova (kao JSON objekat)
- 📌 Pogodno za predstavljanje objekata
- 📌 Tipične operacije:
  - HSET, HGET – set/get pojedinačnih atributa
  - HMSET, HMGET – rad sa više atributa
  - HLEN – broj atributa
  - HKEYS, HVALS, HGETALL – sve ključ/vrednost kombinacije

Redis - tipovi podataka i operacije

🧮 5. **Sorted Set (ZSet)**

- 📐 Kolekcija jedinstvenih elemenata sa score vrednostima (float)
- 📈 Automatski sortiran po score
- 📌 Tipične operacije:
  - ZADD – dodavanje ili ažuriranje elemenata sa score-om
  - ZREM – uklanjanje elemenata
  - ZRANGE – vraćanje po opsegu (npr. top 10)
  - Ako score isti → koristi se leksikografski redosled

🔢 6. **Bitmap**

- ⚙️ Nije poseban tip, već skup bit-operacija nad stringovima
- 🎯 Koristi se za efikasno čuvanje true/false vrednosti
- 🧠 1 string = do 2^{32} bitova!
- 📌 Tipične operacije:
  - SETBIT, GETBIT – postavi/pribavi bit
  - BITOP – bit-operatorska logika (AND, OR, XOR)
  - BITCOUNT – broj 1 bitova
  - BITPOS – pozicija prvog 0 ili 1

🧠 7. **Operacije nezavisne od tipa:**

- EXISTS – da li ključ postoji?
- DEL – obriši ključ i njegovu vrednost
- TYPE – vrati tip podatka pridruženog ključu

⏱️ 8. **Timeout / Isticanje ključeva**

- ✅ Redis omogućava vremenski ograničene ključeve
- 📌 Operacije:
  - EXPIRE – postavi timeout u sekundama
  - PEXPIRE – timeout u milisekundama
  - PERSIST – uklanja prethodno definisan timeout

📡 **Redis Publish/Subscribe (Pub/Sub)**

- 📤 Publisher šalje poruke u kanal
- 📥 Subscriber prima poruke iz kanala

📌 Komande:

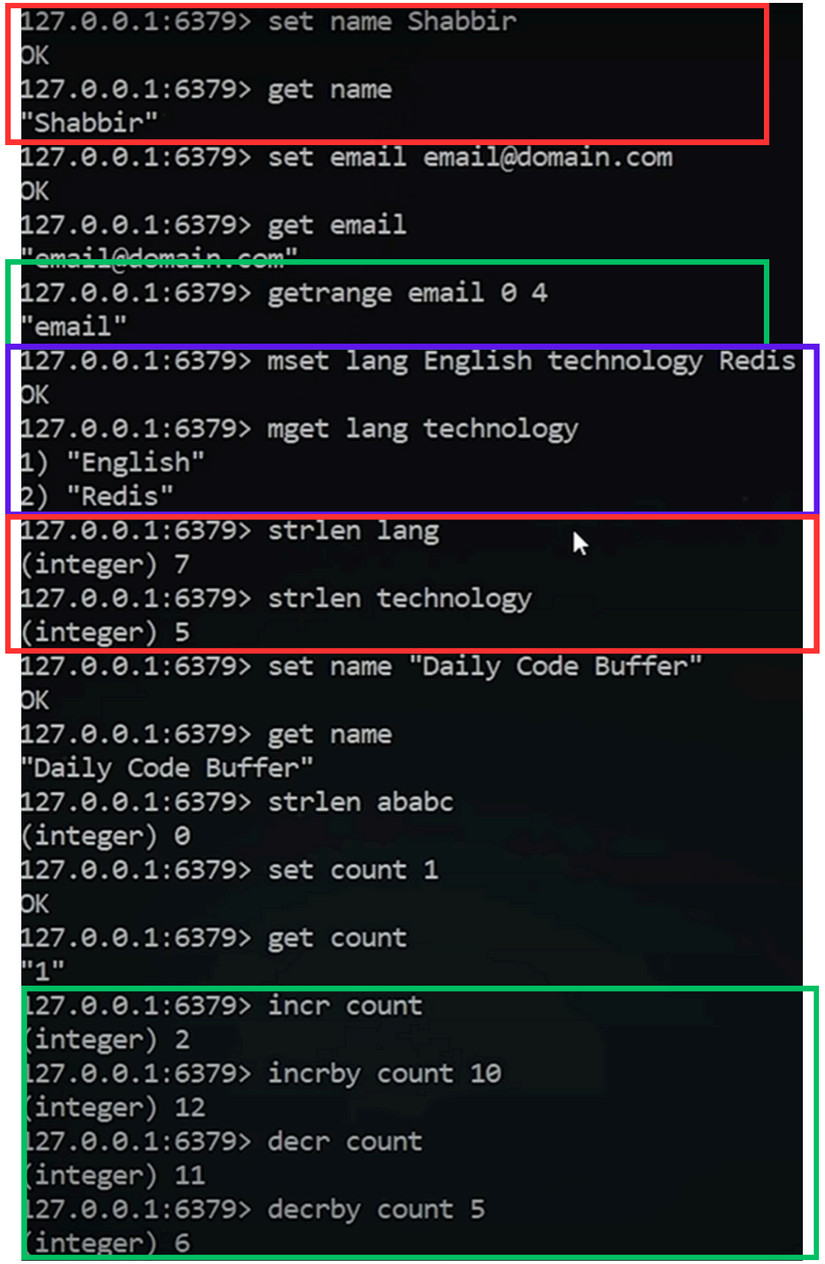
- PUBLISH channel message
- SUBSCRIBE channel
- UNSUBSCRIBE channel

📦 Korisno za:

- Real-time chat sisteme
- Notifikacije
- Event-based komunikaciju među servisima

Redis - string

- ***Namena: Keširanje, brojači i jednostavno skladištenje parova ključ-vrednost.***
- ***Primer: Čuvanje podataka o sesiji korisnika ili brojača pregleda stranice.***



set *- Postavljanje ključa “name”*
get *- Vračanje vrednosti ključa name*

getrange email 0 4 *- Vraća prva 5 karaktera vrednosti ključa “email”*

mset *- Postavljanje vrednosti više ključeva odjednom (“lang”, “technology”)*
mget *- Vraćanje vrednosti više ključeva*

strlen *- Vraća broj karaktera vrednosti ključa*

incr *- Povećava vrednost ključa za 1*
decr *- Smanjuje vrednist ključa za 1*
incrby count 10 *- Povećava vrednost ključa za 10*
decrby count 5 *- Smanjuje vrednost ključa za 5*

Redis - list

lpush *- Dodavanje elementa levo ("India" u "country")*

lrange 0 -1 *- Vraćanje svih elemenata u listi*

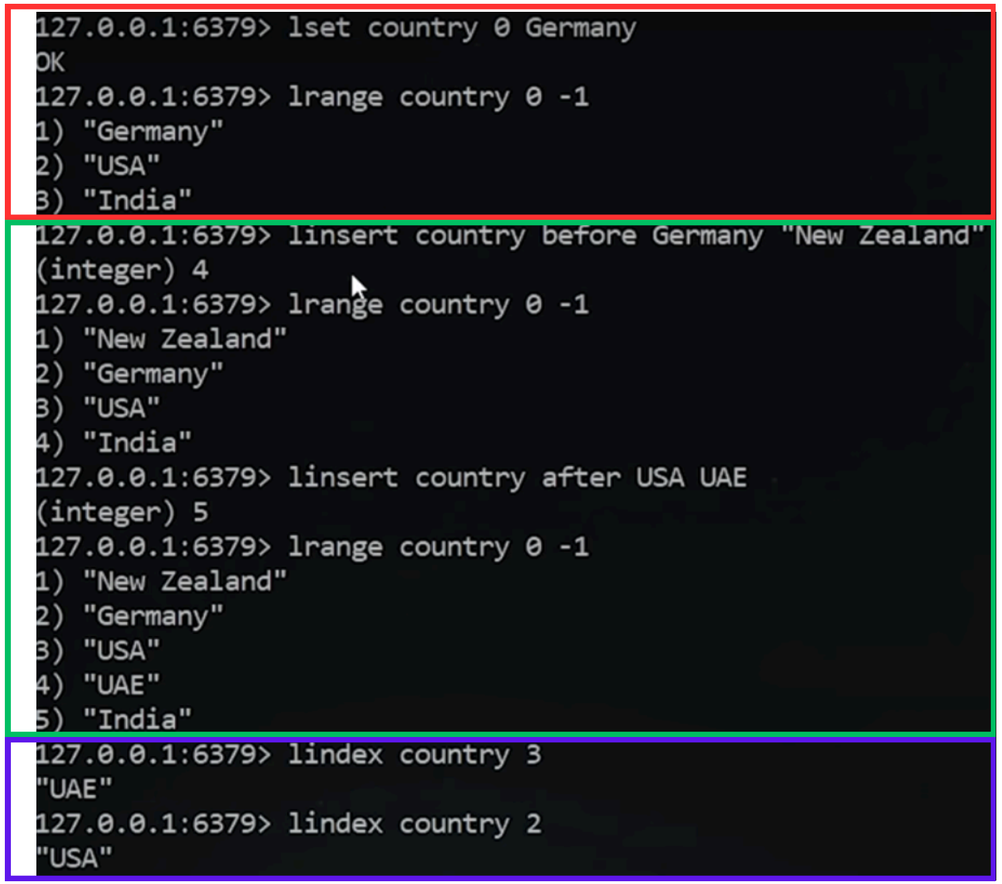
rpush *- Dodavanje elementa desno ("Australia" u "country")*

llen *- Vraća broj elemenata u listi*

lpop *- Uzimanje i uklanjanje prvog elementa iz liste*
rpop *- Uzimanje i uklanjanje poslednjeg elementa iz liste*

Redis - list

🔹 ***Namena: Redovi čekanja (queues), stekovi (stacks) i uređene kolekcije.***
🔹 ***Primer: Red zadataka za obradu ili prikaz poslednjih aktivnosti korisnika.***



lset 0 *- Postavljanje vrednosti prvog elementa u listi*

linsert *- Umetanje elemenata (BEFORE/AFTER)*

lindex *- Vraća vrednost elementa s određenim indeksom*

Redis - set & zset

set

```
> sadd myset 1 2 3
(integer) 3
> smembers myset
1. 3
2. 1
3. 2

> sismember myset 3
(integer) 1
> sismember myset 30
(integer) 0
```

```
> sadd news:1000:tags 1 2 5 77
(integer) 4

> sadd tag:1:news 1000
(integer) 1
> sadd tag:2:news 1000
(integer) 1
> sadd tag:5:news 1000
(integer) 1
> sadd tag:77:news 1000
(integer) 1

> smembers news:1000:tags
1. 5
2. 1
3. 77
4. 2
```

vešću (news) sa ID-jem 1000 dodeljujemo tagove s ID-jevima 1, 2, 5 i 77

tagu sa ID-jem 1 dodeljujemo da pripada vešću s ID-jem 1000, itd...

🔹 ***Namena: Kolekcije jedinstvenih elemenata.***
🔹 ***Primer: Tagovi na vestima, jedinstveni posetioci, ili bilo koja situacija gde su potrebne bez duplikata vrednosti.***

1940 je score po kome se sortiraju elementi, a "Alan Kay" je value

zset

Vraćeni su elementi sortirani po score-u

```
> zadd hackers 1940 "Alan Kay"
(integer) 1
> zadd hackers 1957 "Sophie Wilson"
(integer 1)
> zadd hackers 1953 "Richard Stallman"
(integer) 1
> zadd hackers 1949 "Anita Borg"
(integer) 1
> zadd hackers 1965 "Yukihiro Matsumoto"
(integer) 1
> zadd hackers 1914 "Hedy Lamarr"
(integer) 1
> zadd hackers 1916 "Claude Shannon"
(integer) 1
> zadd hackers 1969 "Linus Torvalds"
(integer) 1
> zadd hackers 1912 "Alan Turing"
(integer) 1
```

```
> zrange hackers 0 -1
1) "Alan Turing"
2) "Hedy Lamarr"
3) "Claude Shannon"
4) "Alan Kay"
5) "Anita Borg"
6) "Richard Stallman"
7) "Sophie Wilson"
8) "Yukihiro Matsumoto"
9) "Linus Torvalds"
```

🔹 ***Namena: Uređene kolekcije sa skorovima (bodovima).***
🔹 ***Primer: Rang liste, sistemi za ocenjivanje, vremenski zasnovani događaji.***

Redis - publish/subscribe

🔹 ***Namena: Slanje i prijem poruka u realnom vremenu između delova aplikacije.***
🔹 ***Primer: Chat aplikacije – korisnik pošalje poruku, svi pretplaćeni odmah je dobijaju.***

Subscriber 1

```
127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "New Breaking News"
```

Subscriber 2

```
127.0.0.1:6379> subscribe news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "New Breaking News"
```

Publisher

```
127.0.0.1:6379> publish news "New Breaking News"
(integer) 2
127.0.0.1:6379>
```

1) *Subscriberi su se pretplatili na kanal “news”*
2) *Subscriberi čekaju na poruke*
3) *Publisher objavljuje poruku “New Breaking News” kanalu “news”*
4) Oba subscribera dobijaju porulu “New Breaking News”