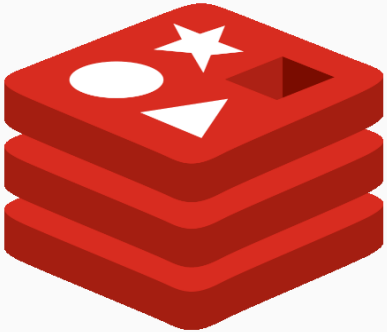
Redis

Modul 346, BBZW

Patrick Bucher



redis

Abbildung 1: Redis: **Re**mote **Di**ctionary **S**ervice

Arten von Datenspeichern

- **Strukturierte** Daten: Relationale Datenbanken
  - PostgreSQL
  - MySQL
  - Microsoft SQL Server
  - Oracle
  - sqlite
- **Unstrukturierte** Daten: Dateisystem, BLOB-Storage
  - Amazon S3
    - Minio
  - Azure Blob Storage
- **Halbstrukturierte** Daten: NoSQL-Datenbanken
  - MongoDB
  - CouchDB
  - InfluxDB
  - **Redis**

Redis ist ein Key-Value-Store

Mit einem Key-Value-Store können Werte anhand eines eindeutigen Schlüssels nachgeschaut werden:

Schlüssel	Wert
balance	25471.93
127.0.0.1	localhost
ipv4	195.347.52.9
Joe Doe	+019425287164
started	2021-12-29T19:35:12.15.632+00:00

Beispiele: DNS, Telefonbuch, ARP-Tabelle, Wörterbuch, Lexikon

Redis ist eine Art *Nachschlagewerk*!

Redis speichert Datenstrukturen

Redis unterstützt Datentypen, die Sie evtl. aus dem Programmierunterricht kennen:

Programmiersprachen	**Redis**
Primitive Datentypen	Strings
Strings	Strings
Arrays/Listen	Listen
Maps	Hashes
Sets	Sets

siehe auch Redis-Datentypen

Mit Redis kann man **Datenstrukturen** abspeichern.

Redis ist eine *Map*!

Datenstrukturen: Arrays (Wiederholung)

Ein *Array* verwendet Indizes von $[0..n[$:

key	0	1	2
value	13.75	25.45	99.95

Abbildung 2: Ein Array

Datenstrukturen: Maps (Verallgemeinerung)

Eine *Map* verwendet arbiträre Indizes:

key	"foo"	"bar"	"qux"
value	13.75	25.45	99.95

Abbildung 3: Eine Map

Andere Bezeichnungen: *Dictionary* (Python), *Hash* (Ruby), *Table* (Lua)

Redis ist eine *Map*!

Listen

Eine *Liste* hat keine Indizes, sondern speichert Nachfolger:

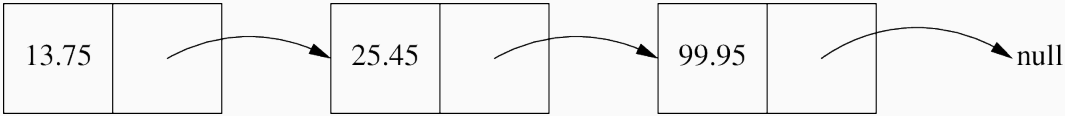


Abbildung 4: Eine (einfach verkettete) Liste

- Listen und Arrays können praktisch das gleiche machen.
- Listen und Arrays haben aber ihre Vor- und Nachteile.

Listen vs. Arrays

Operationen auf Listen bzw. Arrays mit n Elementen benötigen mehr oder weniger Schritte:

Operation	**Schritte (Array)**	**Schritte (Liste)**
Zugriff auf bestimmtes Element	1	durchschnittlich: $n/2$
vorne anfügen	n	1
hinten anfügen	1 oder $n+1$	1 oder $n+1$

- hinten anfügen bei Array
  - 1 Schritt, wenn es hinten noch Platz hat
  - $n+1$ Schritte, wenn das Array vergrössert werden muss (Kopieren der Einträge)
- hinten anfügen bei Liste
  - 1 Schritt, wenn ein Verweis auf das Ende gespeichert wird
  - $n+1$ Schritte, wenn das Ende zuerst gesucht werden muss

Sets (Mengen)

Ein Set ist eine *ungeordnete* Ansammlung von *eindeutigen* Elementen.

- Beispiele
  - $A = \{1, 2, 3, 4, 5\}$
  - $B = \{2, 4, 6, 8, 10\}$
- Auf Sets können *Mengenoperationen* angewendet werden
  - Vereinigungsmenge: $A \cup B = \{1, 2, 3, 4, 5, 6, 8, 10\}$
  - Schnittmenge: $A \cap B = \{2, 4\}$
  - Differenz: $A - B = \{1, 3, 5\}$ bzw. $B - A = \{6, 8, 10\}$

Die Schlüssel einer Map sind ein Set.

Redis unterstützt viele Mengenoperationen!

Art der Datenspeicherung

Redis kann Daten auf verschiedene Arten speichern:

- **RDB** (Redis Database): Datenbank als **Zustand** in einer Datei
  - Vorteile: kompakt, schnell, ideal für Backups
  - Nachteile: sichert Daten nur periodisch
- **AOF** (Append Only File): Datenbank als **Transaktionslog** in einer Datei
  - Vorteile: nachvollziehbar, sicher
  - Nachteile: grösser, langsamer
- **AOF** & **RDB**: Datenbank wird doppelt abgespeichert
  - Vorteile: sehr sicher
  - Nachteile: Performance schlechter, benötigt mehr Speicherplatz
- gar nicht (In-Memory-Datenbank): Datenbank wird nur im Memory gehalten
  - Voreteile: extrem schnell
  - Nachteile: Datenverlust bei Serviceunterbruch

Installation

Debian

```
$ sudo apt install redis
```

Installiert und startet den Redis-Server mit Hilfswerkzeugen (z.B. `redis-cli`).

Kostenloses Cloud-Angebot

- Registrierung mit E-Mail-Adresse, Google- oder GitHub-Account
- lokales `redis-cli` wird weiterhin benötigt

Online-Demo von Railway.app

- grafisches Interface

Verwendung

```
$ redis-cli
127.0.0.1:6379> PING
PONG
127.0.0.1:6379> SET name John
OK
127.0.0.1:6379> KEYS *
1) "name"
127.0.0.1:6379> GET name
"John"
127.0.0.1:6379> DEL name
(integer) 1
127.0.0.1:6379> EXISTS name
(integer) 0
```

Befehle

Redis kennt über 400 Befehle (Stand: 27.11.2022).

- Das Präfix richtet sich nach der Datenstruktur, auf welcher der Befehl operiert:
  - List: `L` bzw. `R` für Operationen am linken bzw. rechten Listenende
  - Sets: `S`
  - Hashes: `H`
  - Sorted Sets: `Z`
- Die Befehle haben keine, einen oder mehrere (teils optionale) Parameter:
  - `FLUSHALL`: keine Parameter
  - `GET key`: ein Parameter
  - `SET key value`: zwei Parameter

Befehle: Grundlegende Verwendung

- `PING`: Verbindung testen (gibt `PONG` aus, wenn Verbindung steht)
- `HELP`: Hilfe ausgeben, z.B. zu einem Befehl
  - `HELP PING`
- `AUTH`: Interaktive Authentifizierung mit Passwort
- `FLUSHALL`: Löscht **alle** Einträge
- `KEYS`: Schlüssel gemäss Muster anzeigen
  - `KEYS *`: listet alle Schlüssel auf
- `EXISTS`: Prüft, ob ein Schlüssel existiert
- `TYPE`: Gibt den Datentyp des Werts von einem Schlüssel aus
- `SAVE`: Persistente Speicherung forcieren

Befehle: Einfache Werte

- `SET`: Ein Schlüssel/Wert-Paar definieren
  - `MSET`: Mehrere Schlüssel/Wert-Paare gleichzeitig definieren
- `GET`: Wert anhand eines Schlüssels auslesen
- `DEL`: Eintrag entfernen
- `RENAME`: Schlüssel umbenennen

Strukturierte Schlüsselnamen

Schlüsselnamen können gemäss einer Konvention strukturiert werden:

```
SET lucerne.name Luzern
SET lucerne.population 81592

SET employee:1234:name Dilbert
SET employee:1234:position Engineer
```

Schlüssel zum gleichen “Feld” auslesen:

```
GET employee:*:name
```

Schlüssel zum gleichen “Datensatz” auslesen:

```
GET employee:1234:*
```

Befehle: Hashes

Ein **Hash** speichert Schlüssel/Wert-Paare ab und ist mit einer `map` oder `struct` in Go vergleichbar, erlaubt aber keine Verschachtelung.

- `HSET`: Definiert einen Hash mit Schlüssel/Wert-Paaren
- `HGET`: Gibt ein Feld zu einem Hash aus
- `HGETALL`: Gibt alle Feldnamen und -Werte zu einem Hash aus
- `HKEYS`: Gibt die Feldnamen zu einem Hash aus
- `HVALS`: Gibt die Werte zu einem Hash aus
- `HDEL`: Löscht ein Feld von einem Hash (aber nicht den Hash selber)
- `HSETNX`: Setzt ein Feld von einem Hash, sofern es noch nicht definiert ist

Die Struktur von einem zusammengesetzten Objekt muss nicht über den Namen codiert werden.

Hashes: Mitarbeiterverwaltung

```
HSET employee.dilbert
    id 715
    name Dilbert
    position Engineer
    salary 125000
    hired 1992

HGET employee.dilbert position
"Engineer"

KEYS employee.*
1) "employee.dilbert"
```

Bonus: Ausgabe von CSV und JSON

```
$ redis-cli --csv HGETALL employee.dilbert

"id","715","name","Dilbert","position","Engineer","salary","125000","hired","1992"

$ redis-cli --json HGETALL employee.dilbert

{
  "id": "715",
  "name": "Dilbert",
  "position": "Engineer",
  "salary": "125000",
  "hired": "1992"
}
```

Links

- Redis.io: OpenSource-Software
  - Documentation: Offizielle Dokumentation
  - Commands: Befehlsübersicht
  - Redis Data Types: Datentypen
  - Redis CLI: Kommandozeileninterface
  - Clients: Sprachanbindungen
    - Go Redis: Client-Library für Go
- Redis.com: kommerzielles Angebot
  - Redis Enterprise Cloud: Cloud-Angebot
  - Try Free: Kostenloses Cloud-Angebot zum Einstieg