



---

# Advanced Data Management

## Redis Intro

---

Peyer Lars

Berger Adrian

Version 1.0

9. November 2019

<b>1</b>	<b>Key Value Store</b>	<b>2</b>
1.1	Definition . . . . .	2
1.2	Vorteile und Nachteile . . . . .	2
1.2.1	Vorteile . . . . .	2
1.2.2	Nachteile . . . . .	2
1.3	Einsatzgebiete . . . . .	2
1.4	In-Memory . . . . .	2
1.5	Top 5 Key Value Stores . . . . .	3
<b>2</b>	<b>Redis</b>	<b>4</b>
2.1	Charakterisierung . . . . .	4
2.1.1	Plattformen, welche Redis verwenden . . . . .	4
2.2	Installation . . . . .	4
2.3	Datentypen . . . . .	5
2.4	Abfragesprache . . . . .	5
2.4.1	Transaktionen . . . . .	6
2.5	Benchmarks . . . . .	6
2.6	Persistenz . . . . .	6
2.7	Replikation . . . . .	7
2.8	Optimierungsmöglichkeiten . . . . .	7
2.9	GUI . . . . .	7
2.10	API . . . . .	7
2.11	Massenload . . . . .	7
<b>3</b>	<b>Ressourcen</b>	<b>8</b>

# 1 Key Value Store

## 1.1 Definition

Ein Key Value Store basiert auf einer Tabelle mit genau zwei Spalten:

In der einen befindet sich ein eindeutiges Identifikationsmerkmal, der Schlüssel (Key), in der anderen der Wert (Value). Somit wird jeder Wert eindeutig einem Schlüssel zugewiesen. Welche Datentypen im Wert gespeichert werden können, ist abhängig vom verwendeten Key-Value Store. Es handelt sich somit um eine NoSQL Datenbank.

## 1.2 Vorteile und Nachteile

### 1.2.1 Vorteile

- Geschwindigkeit
- Skalierbarkeit
- Simple Model, einfach verständlich

### 1.2.2 Nachteile

- Für Daten mit vielen relationalen Abhängigkeiten sehr komplex
- Keine Lookup Optimierungen für das Suchen

## 1.3 Einsatzgebiete

Überall wo schnelle Zugriffszeiten bei großen Datenmengen benötigt werden, eignen sich Key Value Stores. Typische Einsatzgebiete sind deshalb Warenkörbe in Onlineshops oder das Speichern von Session-Daten.

## 1.4 In-Memory

Bei vielen Key Value Stores handelt es sich um In-Memory Datenbanken. Im Gegensatz zu herkömmlichen On-Disk Datenbanken verwenden diese anstelle des ROM, den Arbeitsspeicher eines Computers.

	Disk	SSD
Funktion	Ist eine rotierenden Magnetscheibe	Speichermedium ist ein Flash-Speicher
Stärken	Günstig	Effizient
Grenzen		

Disk und SSD - Charakteristiken, Stärken und Grenzen Die Bedeutung von In Memory Datenbanken

## 1.5 Top 5 Key Value Stores

Gemäss db-engines.com (Stand 27.10.2019) sind die folgenden Datenbanksysteme führend:

1. Redis
2. Amazon DynamoDB
3. Microsoft Azure Cosmos DB
4. Memcached
5. Hazelcast

## 2 Redis

### 2.1 Charakterisierung

Redis ist ein Open Source, In-Memory Datenbanksystem, das als vorallem als Datenbank, Cache und Message Broker verwendet werden kann.

Redis ist BSD lizenziert und stellt somit keine Anforderungen an die Weiterverteilung der Software.

Der Name Redis entstand als Abkürzung für remote dictionary server.

Entwickelt wurde Redis 2009 von Salvatore Sanfilippo in C und gehört mittlerweile zu VMWare. Die aktuelle Version (Stand 5. November 2019) ist 5.0.5, wobei die aktuelle LTS Version 3.2.11 ist. Redis ist single-threaded und lässt sich einfach replizieren.

#### 2.1.1 Plattformen, welche Redis verwenden

Redis erfreut sich grosser Beliebtheit und ist laut DB-Engines.com die verbreitetste Key-Value Datenbank. Einige der grossen Kunden sind etwa Twitter (Real-Time Delivery), Github (Speicherung von Routing Information) und StackOverflow (2-Level Cache).

Mehr Infos unter: <https://redis.io/topics/whos-using-redis>

### 2.2 Installation

Redis lässt sich leicht als tar.gz herunterladen und installieren:

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
make install
```

Falls gewünscht lässt sich Redis auch mittels Docker installieren.

Mehr Infos unter: <https://redis.io/topics/quickstart>

## 2.3 Datentypen

Redis unterstützt die folgenden Datentypen:

Typ	Beschreibung
String	Eine beliebige Zeichenfolge
List	Entspricht einem Array, die Elemente werden über einen positiven, ganzzahligen, null-basierten Index angesprochen
Hash	Entspricht einem assoziativem Array, die Elemente bestehen hier wiederum aus Schlüssel-Werten Paaren.
Set	Wie List, jedoch kann jedes Element nur einmal vorkommen darf.
Sorted Set	Entspricht einem Set, nur dass zusätzlich der Index des Elements explizit mit angegeben wird.
Bitmaps	Speicherung von Bits
HyperLogLogs	Probabilistische Datenstruktur, mit der sich die Kardinalität einer Datenmenge sehr speichereffizient bestimmen lässt

Mehr Infos unter: <https://redis.io/topics/data-types>

## 2.4 Abfragesprache

Für String lässt sich ein Speichern und Abrufen mit `get` und `set` erreichen (alle Beispiele mit Python Redis):

```
import redis
r = redis.Redis()
r.set("msg:hello", "Hello Redis!!!")
msg = r.get("msg:hello")
# prints b'Hello Redis!!!'
```

Für jeden Datentyp gibt es solche Schreib- und Lesebefehle. Zudem gibt es diverse weitere Befehle (z.B. zum Zählen der Elemente). Nachfolgend ein Beispiel für ein Set:

```
mountainSet = "Berge"
# Add elements to the Redis set
r.sadd(mountainSet, "Eiger")
r.sadd(mountainSet, "Moench")
r.sadd(mountainSet, "Jungfrau")

print("Anzahl_Elmente: ")
print(r.scard(mountainSet))
print(r.smembers(mountainSet))
# prints Anzahl_Elmente: 3 {b'Eiger', b'Moench', b'Jungfrau'}
```

Eine Liste aller Abfragen findet sich hier: <https://www.cheatography.com/tasjaevan/cheat-sheets/redis/>

### 2.4.1 Transaktionen

Eine Transaktion in Redis besteht aus einem Block von Befehlen, die zwischen MULTI und EXEC (oder DISCARD für Rollback) platziert werden. Sobald ein MULTI gefunden wurde, werden die Befehle auf dieser Verbindung nicht ausgeführt - sie stehen in der Warteschlange (und der Anrufer erhält die Antwort QUEUED). Wenn eine EXEC angetroffen wird, werden sie alle in einer einzigen Transaktion angewendet (d.h. ohne dass andere Verbindungen Zeit zwischen den Operationen erhalten). Wenn anstelle von EXEC eine DISCARD gesehen wird, wird alles weggeworfen. Da die Befehle innerhalb der Transaktion in der Warteschlange stehen, können Sie innerhalb der Transaktion keine Entscheidungen treffen.

Sämtliche Informationen dazu finden sich hier: <https://redis.io/topics/transactions>

## 2.5 Benchmarks

Redis bietet das Dienstprogramm redis-benchmark, das laufend Befehle von Clients simuliert und gleichzeitig Gesamtabfragen sendet. Damit kann die Performance von Redis getestet und visualisiert werden. Anleitungen und Beispiele dazu finden sich hier: <https://redis.io/topics/benchmarks>

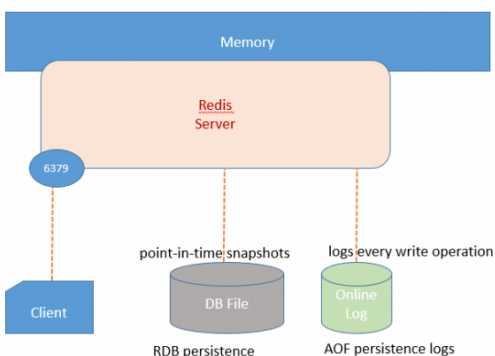
## 2.6 Persistenz

Redis bietet generell zwei Varianten der sicheren Datenspeicherung:

RDB (Snapshots): Zu bestimmten Zeitpunkten (z.B. jede volle Stunde) wird eine vollständige Kopie der im Speicher befindlichen Daten erstellt. Wenn z.B. zwischen zwei Snapshots die Stromversorgung verlieren geht, gehen die Daten aus der Zeit zwischen dem letzten Snapshot und dem Crash verloren.

AOF: Logt jede vom Server empfangene Schreiboperation und führt diese beim Serverstart erneut ab zur Rekonstruktion des ursprünglichen Datensatzes.

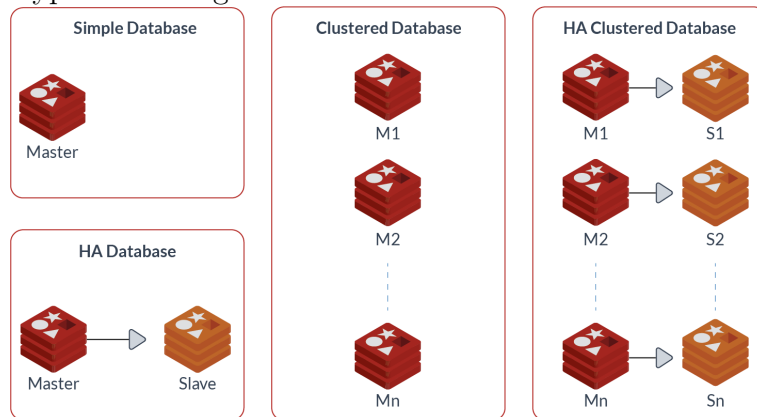
Grundsätzlich lassen sich RDB und AOF einzeln aktivieren. Somit wäre auch keine Persistenz möglich, in dem beides deaktiviert wird. Die beste Persistenz wird erreicht, in dem beides aktiviert wird. Grundsätzlich handelt es sich hierbei um eine Frage zwischen Datensicherheit und Leistung.



Weitere Vor- und Nachteile finden sich hier: <https://redis.io/topics/persistence>

## 2.7 Replikation

### Typisches Design



Verteilte Datenhaltung, Skalierbarkeit, Protokolle, Architektur <https://redis.io/topics/replication>

## 2.8 Optimierungsmöglichkeiten

<https://redis.io/topics/memory-optimization>

## 2.9 GUI

<https://redislabs.com/blog/so-youre-looking-for-the-redis-gui/>

## 2.10 API

<https://redis.io/topics/modules-intro>  
mit Demo

## 2.11 Massenload

<https://redis.io/topics/mass-insert>



### 3 Ressourcen

[https://en.wikipedia.org/wiki/Key-value\\_database](https://en.wikipedia.org/wiki/Key-value_database)

<https://db-engines.com/de/ranking/key-value+store>

<https://www.aerospike.com/what-is-a-key-value-store/>

<https://www.ionos.de/digitalguide/hosting/hosting-technik/key-value-store/>

<https://redis.io/>

<https://redislabs.com>

[https://www.pipperr.de/dokuwiki/doku.php?id=nosql:redis\\_overview](https://www.pipperr.de/dokuwiki/doku.php?id=nosql:redis_overview)

<https://wikipedia.org/wiki/Redis>