



# Moderne Datenbanken

## Key-Value Datenbanken

# Agenda

|   |                                      |    |
|---|--------------------------------------|----|
| 1 | Beispiel: Datenanalyse               | 2  |
| 2 | Marktrecherche Key-Value Datenbanken | 14 |
| 3 | Key-Value Datenbank Redis            | 19 |



Fachhochschule  
Dortmund  
University of Applied Sciences and Arts

Abmelden

Benutzer: saatz Letzte Stand: Hallo saatz Zur Zeit Online:

Home

Übungen

Suche

Kontakt

## 0.1 Senkrechter Wurf

{Aufgabe}

Ein Ball mit der Masse 100g wird von einem Turm aus 28m Höhe senkrecht nach oben mit einer Anfangsgeschwindigkeit von 7 m/s geworfen.

1. ) Stellen Sie die Bewegungsgleichungen auf, d.h. die Funktionen für den Ort, die Geschwindigkeit und die Beschleunigung des Balles.
2. ) Berechnen Sie die erreichte Wurfhöhe.
3. ) Berechnen Sie die Zeit die verstreicht bis der Ball seine maximale Höhe erreicht hat.

check die Ergebnisse, speichere aktuelle Zustand und gehe weiter...

Link zur Vorlesung:

Weiter »

Link zum Praktikum:

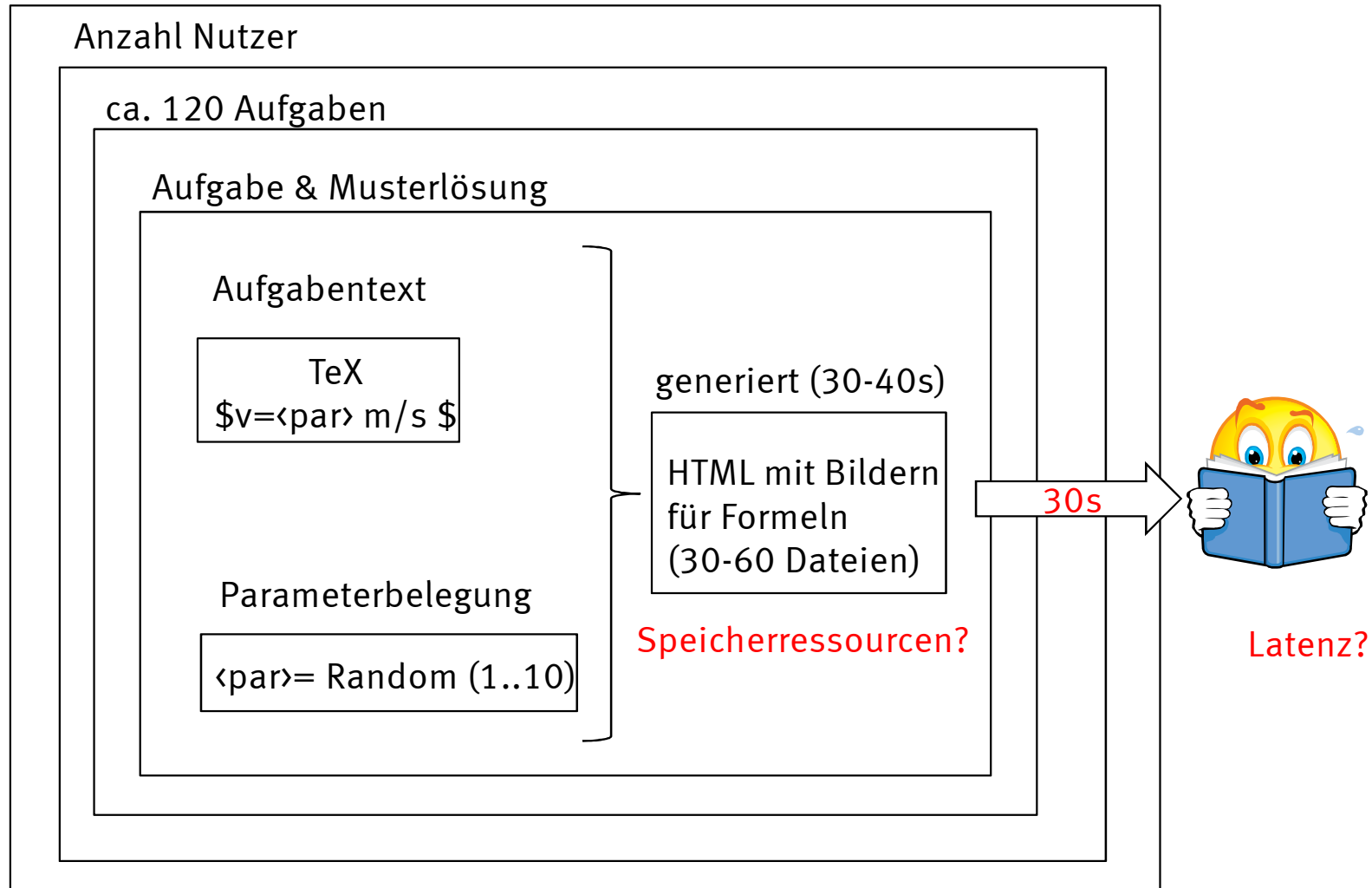
Weiter »

Formelsammlung (.pdf)

Weiter »

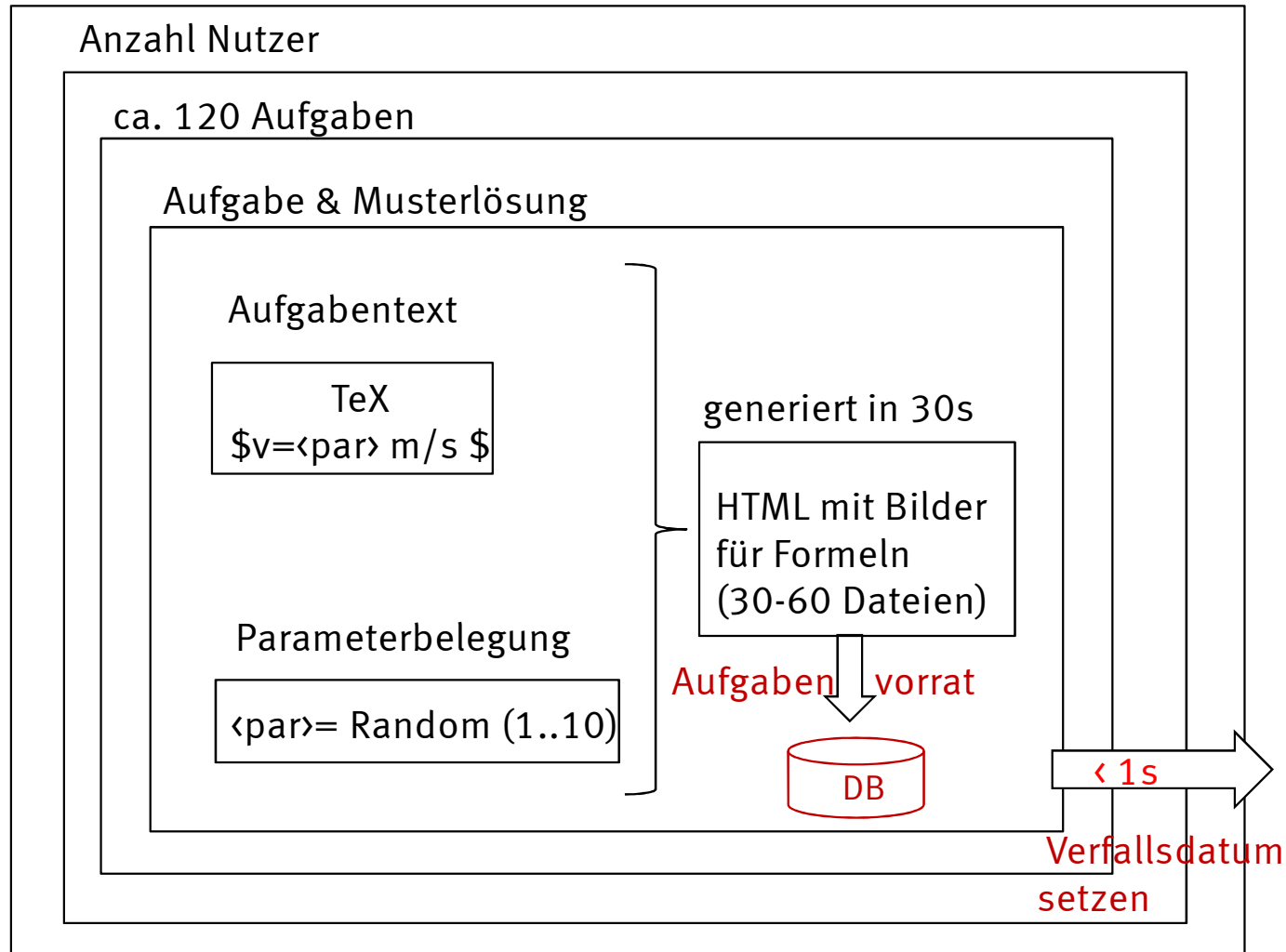
# Beispiel: Physik-Aufgabentrainer

Semester



# Beispiel: Physik-Aufgabentrainer

Semester



| Auswahlkriterium               | Gewicht | DB 1 | DB 2 | DB 3 |
|--------------------------------|---------|------|------|------|
| Kriterium 1                    | 2       | 1    | 2    | 2    |
| Kriterium 2                    | 5       | 2    | 3    | 2    |
| Kriterium 3                    | 3       | 3    | 1    | 0    |
| Summe<br>(Gewicht * Bewertung) | 10      | 21   | 22   | 14   |

Handlungsempfehlung

zu evaluierende Datenbanken



Evaluation der Datenbanken

Erstellung Evaluationsprototypen

Aufstellung Bewertungskriterien inkl. Gewichtung (z.B. Usability)

Vergleich der Testergebnisse



Handlungsempfehlung Datenbankenauswahl

# Datenbank-Auswahlkriterien

| Anforderung                             |                                                                                                                                                                                                                                                                                                                         | Gewichtung |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| Datenanalyse                            | <ul style="list-style-type: none"> <li>– Datenart</li> <li>– Daten-/Speichermodell</li> <li>– Datennavigation</li> <li>– Datenmenge</li> <li>– Datenkomplexität</li> </ul>                                                                                                                                              |            |
| Transaktionsmodell                      | <ul style="list-style-type: none"> <li>– klassische Transaktionsmodelle</li> <li>– CAP-Abwägung</li> </ul>                                                                                                                                                                                                              |            |
| Performance-Aspekte                     | <ul style="list-style-type: none"> <li>– Performance-Dimensionen</li> <li>– Scale-up versus Scale-out</li> </ul>                                                                                                                                                                                                        |            |
| Abfrageanforderungen                    |                                                                                                                                                                                                                                                                                                                         |            |
| Architektur                             | <ul style="list-style-type: none"> <li>– Verteilungsarchitektur</li> <li>– Datenzugriffsmuster (Data Access Patterns)</li> </ul>                                                                                                                                                                                        |            |
| weitere nicht-funktionale Anforderungen | <ul style="list-style-type: none"> <li>– Replikation</li> <li>– Refactoring</li> <li>– Support</li> <li>– Know-how &amp; Einfachheit</li> <li>– Unternehmensvorgaben</li> <li>– Datenbankvielfalt</li> <li>– Sicherheit</li> <li>– Backup/Restore</li> <li>– Ausfallsicherheit</li> <li>– Lizenz/Open Source</li> </ul> |            |




Informationen zu relationalen und NoSQL Datenbankmanagement

Home | DB-Engines Ranking | S

Featured Products: [DataStax](#)

**DB-Engines**

DB-Engines ist eine Initiative zur Sammlung Präsentation von Informationen zum Thema Datenbankmanagementsysteme (DBMS). Ne

Tabelle entnommen aus [EFHB 2010, S. 283]

1. Welche Datenarten sollen gespeichert werden?
  - Domain-Daten, Log-Daten, Event-Daten, Message-Daten, ...
  - Unternehmenskritische Daten (Finanzdaten...), Business-Daten (Aufträge, ...)
  - Metadaten, Temporäre Daten, Session-Daten
  - Geographische Daten
2. Welches ist das geeignete Daten- und Speichermodell?
  - In welcher Form liegen die Daten vor?
  - Ist eine strikte Trennung des Datenmodells von der Anwendung erforderlich?
  - Müssen die Daten strukturiert vorliegen?
  - Ist ein Schema erforderlich?
  - Wie sind die Daten geeignet zu speichern? (Relational, Dokumentorientiert, ...)
3. Wie viele Beziehung bestehen zwischen den Daten?
  - Wie intensiv müssen die Daten durch die Anwendung verknüpft werden?
    - Bsp: Geographische Daten → viele Joins vs. Session-Daten → wenige Joins
4. Wie groß ist die zu speichernde Datenmenge?
  - Perspektive für die nächsten Jahren?
5. Wie komplex sind die zu speichernden Daten?
  - Werden durch den Anwender neue komplexe Datentypen definiert?
  - Können O/R-Mapper die Datenkomplexität abbilden?



Jedem DBS liegt ein **Datenmodell** (Konzept) zugrunde, das festlegt,

- welche **Eigenschaften** die Datenelemente besitzen,
- welche **Struktur die Datenelemente** besitzen dürfen,  
(z.B. Relationen, Tupel ...)
- welche **Konsistenzbedingungen** einzuhalten sind,  
(Integritätsbedingungen zur weiteren Einschränkung der Struktur)
- welche **Operationen** zum Speichern, Suchen, Ändern und Löschen von Datenelementen existieren.

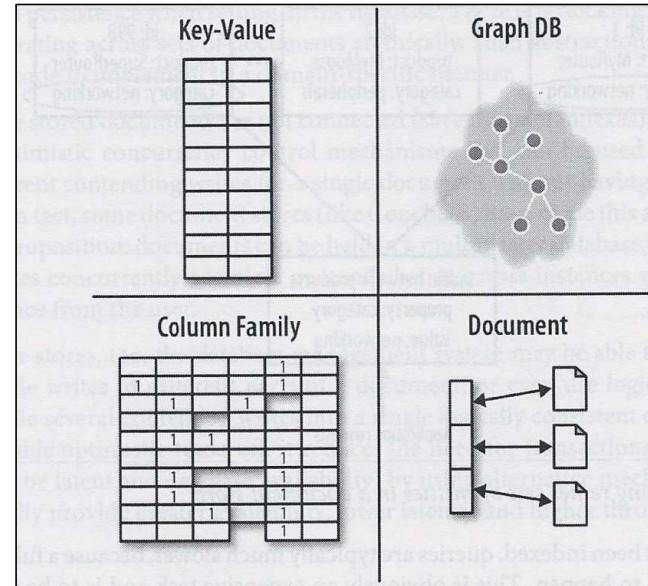
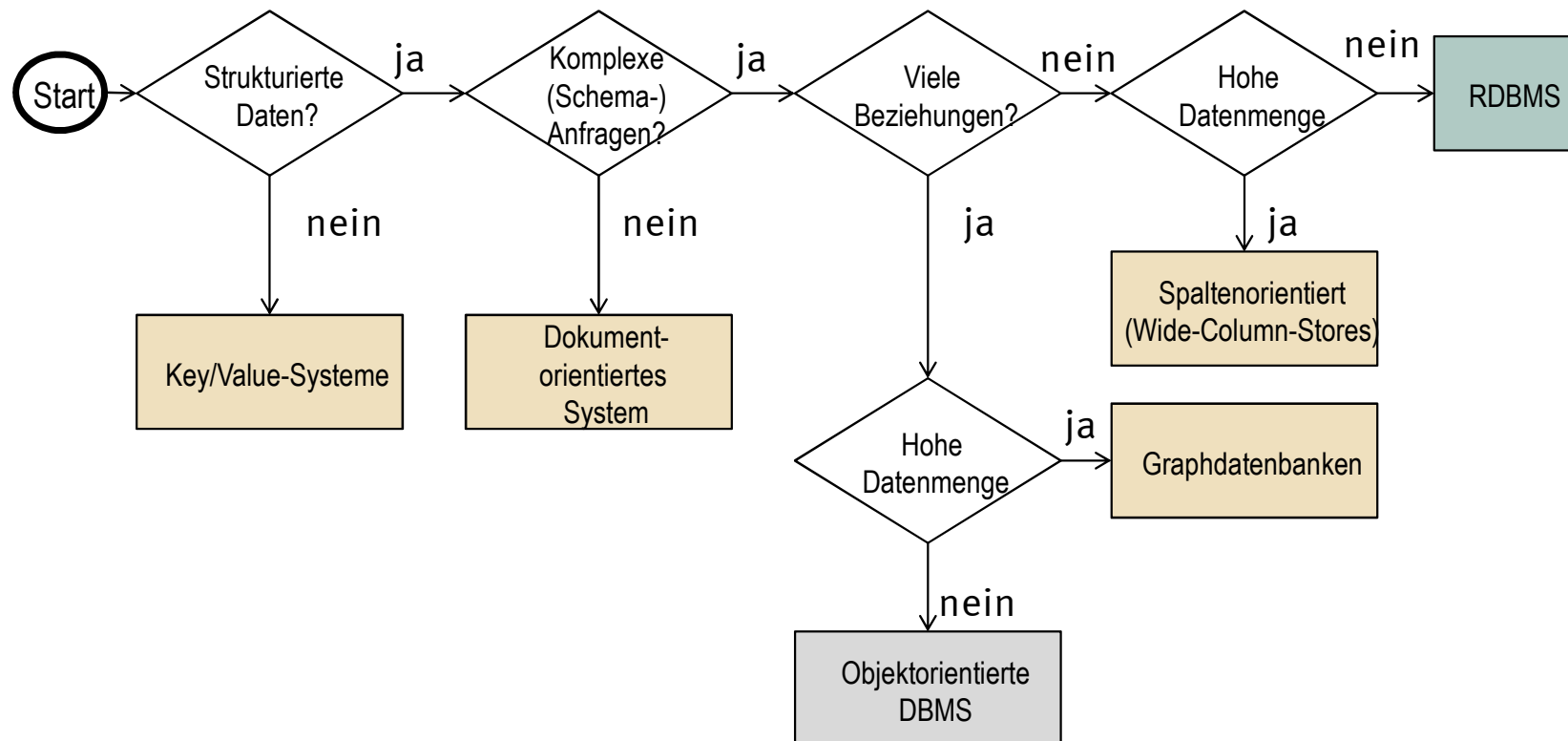


Abb. Entnommen aus Graph Databases, I. robinson, J. Webber & E. Eifrem, O'Reilly (2013)

# Entscheidungskriterien Datenanalyse (Vereinfacht)



Welche Datenbank passt zu dem Anwendungsbeispiel?

| Anwendungsfall                                                                                                   | NoSQL-Datenbank                                         |
|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| Viele Beziehungen/Graphstruktur                                                                                  | Neo4j, Sones etc.                                       |
| Individuelle Replikation, Offline- oder mobiler Einsatz                                                          | CouchDB                                                 |
| Viele große unstrukturierte Daten                                                                                | Amazon S3, Hadoop                                       |
| Sehr viele strukturierte Daten                                                                                   | HBase, Amazon Simple DB, Google- und MS Azure Datastore |
| Hochverfügbare Systeme                                                                                           | Cassandra, Riak, Membase                                |
| Unzusammenhängende Daten, statistische Daten, mehr <i>writes</i> als <i>reads</i> , hohe Performance per Instanz | Redis, MongoDB                                          |
| Flüchtige Daten                                                                                                  | memcache und deren Derivate                             |

Tabelle entnommen aus [EFHB 2010, S. 281]

# Auswahlkriterien ermitteln und gewichten

| Auswahlkriterium                | Muss-Kriterium | Gewicht |
|---------------------------------|----------------|---------|
| Datenmodell                     |                |         |
| Datentypisierung                |                |         |
| Persistierung<br>(Server/Cloud) |                |         |
| Komplexe Anfragen               |                |         |
| Transaktionskonzept             |                |         |
| Konsistenz-Konzept              |                |         |
| Replikation                     |                |         |
| Rechtemanagement                |                |         |
| Lizenz                          |                |         |
| Performanz – Schreiben          |                |         |
| Performanz – Lesen              |                |         |

# Agenda

|   |                                      |    |
|---|--------------------------------------|----|
| 1 | Beispiel: Datenanalyse               | 2  |
| 2 | Marktrecherche Key-Value Datenbanken | 14 |
| 3 | Key-Value Datenbank Redis            | 19 |

- Typische Anwendungsfälle
  - Cache für den schnellen Zugriff auf Daten, z.B. können die Daten direkt über einzelne PUT- und GET-Operationen angesprochen werden.
  - Session-Information von Webanwendungen, z.B. kann das gesamte Sessionobjekt in einem Schlüssel-Wertpaar abgespeichert werden.
  - Nutzerprofile und -vorlieben, z.B. Kundennummer, Kundenname, Sprache, Zeitzone, Produktvorlieben, usw.
  - Warenkorbbinhalte
  - Publish/Subscribe Messaging
- Nicht geeignet ...
  - zur Abbildung von Beziehungen zwischen Daten(mengen) oder
  - zur Abbildung von Verknüpfungen (Links) zwischen Daten oder
  - zum Suchen von Schlüsseln auf der Basis von Eigenschaften der zugehörigen Werte (Ausnahme: Riak) oder
  - um gleichzeitig mehrere Schlüssel zu ändern oder
  - um Schlüssel-Wert-Paare innerhalb eines Transaktionskontext einzufügen (Ausnahme: Redis).

- Datenmodell:
  - Schema aus Schlüssel-Wert Paaren
  - Schlüssel aufteilbar in Namensräume und Datenbanken
  - Werte können sein:
    - Zeichenketten
    - Geodaten
    - Hash-Maps
    - Mengen oder Listen
    - Publish/Subscribe
- Vorteile
  - Sehr einfaches Datenmodell
  - Schnelle und effiziente Datenverwaltung
- Nachteile
  - Geringere Abfragemächtigkeit
  - Abhängigkeit von der Mächtigkeit der API bei der Erstellung komplexer Anfragen
- Datenbank-Beispiele
  - Amazon-Dynamo und Amazon S3
  - **Redis**, Voldemort, Scalaris



**DB-Engines**

Informationen zu relationalen und NoSQL Datenbankmanagementsystemen

Home | DB-Engines Ranking | Systeme | Enzyklopädie | Blog

Featured Products: [DataStax](#) [Couchbase](#) [AllegroGraph](#) [Redis](#) [Neo4j](#)

**DB-Engines**

DB-Engines ist eine Initiative zur Sammlung und Präsentation von Informationen zum Thema Datenbankmanagementsysteme (DBMS). Neben den bewährten relationalen DBMS bilden Systeme und Konzepte aus dem stark wachsenden NoSQL-Sektor einen Schwerpunkt.

Das DB-Engines Ranking ist eine Rangliste der aktuellen Popularität einzelner Systeme, die monatlich aktualisiert wird.

In der Übersicht über [Datenbankmanagementsysteme](#) sind die wichtigsten Eigenschaften zahlreicher Systeme dargestellt. Dies ist einfach abgefragt und auch für einen übersichtlichen Vergleich geeignet.

In der [Datenbank-Enzyklopädie](#) werden

**Die letzten Neuigkeiten**

[MariaDB strengthens its position in the open source RDBMS market](#)  
5. April 2018, Matthias Gelbmann

[PostgreSQL is the DBMS of the Year 2017](#)  
2. Januar 2018, Paul Andlinger, Matthias Gelbmann

**NoSQL**

Your Ultimate Guide to the Non-Relational Universe! [including News Feeds]

**NoSQL DEFINITION:**Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source and horizontally scalable**.

[www.db-engines.com/de](http://www.db-engines.com/de)

<http://nosql-database.org/>

**LIST OF NOSQL DATABASES [currently >225]**



# Eingrenzung der Datenbank-Kandidaten

| Auswahlkriterium                | KO/<br>Gewicht | Redis                       | Amazon DynamoDB                    | Memcache                             |
|---------------------------------|----------------|-----------------------------|------------------------------------|--------------------------------------|
| Datenmodell                     |                | Key-Value                   | Dokument & Key-Value               | Key-Value                            |
| Datentypisierung                |                | teilweise                   | ja                                 | nein                                 |
| Persistierung<br>(Server/Cloud) |                | ja (Server)                 | ja (Cloud)                         | nein (Server)                        |
| Anfragesprache                  |                | nein                        | nein                               | nein                                 |
| Transaktions-<br>konzept        |                | Optimistic Locking          | nein                               | nein                                 |
| Konsistenzkonzept               |                | Eventuell                   | Eventuell/Unmittelbar<br>(konfig.) | nein                                 |
| Replikation                     |                | Master-Slave                | ja                                 | nein                                 |
| Rechte                          |                | einfacher<br>Passwortschutz | Rollen & Rechte pro<br>Benutzer    | Passwortschutz<br>Berechtigungslayer |
| Lizenz                          |                | Open Source                 | Kommerziell                        | Open Source                          |
| <b>Summe</b>                    |                |                             |                                    |                                      |

Welche Anforderungen können noch nicht bewertet werden?

# Agenda

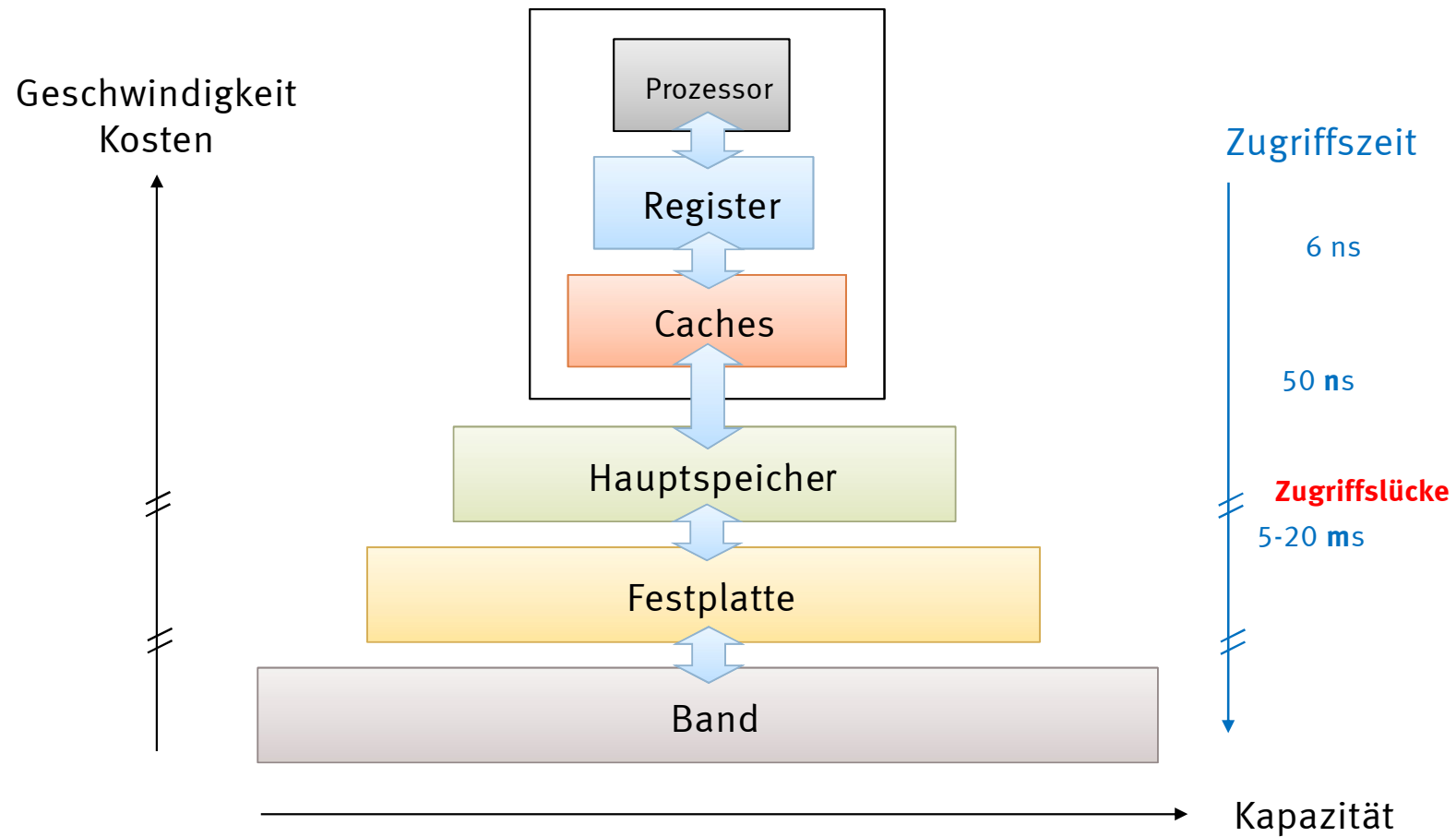
|     |                                      |    |
|-----|--------------------------------------|----|
| 1   | Beispiel: Datenanalyse               | 2  |
| 2   | Marktrecherche Key-Value Datenbanken | 14 |
| 3   | Key-Value Datenbank Redis            | 19 |
| 3.1 | Redis - Befehle (Auswahl)            | 26 |
| 3.2 | JavaSchnittstelle JRedis             | 37 |
| 3.3 | Beispiel: Evaluation                 | 41 |

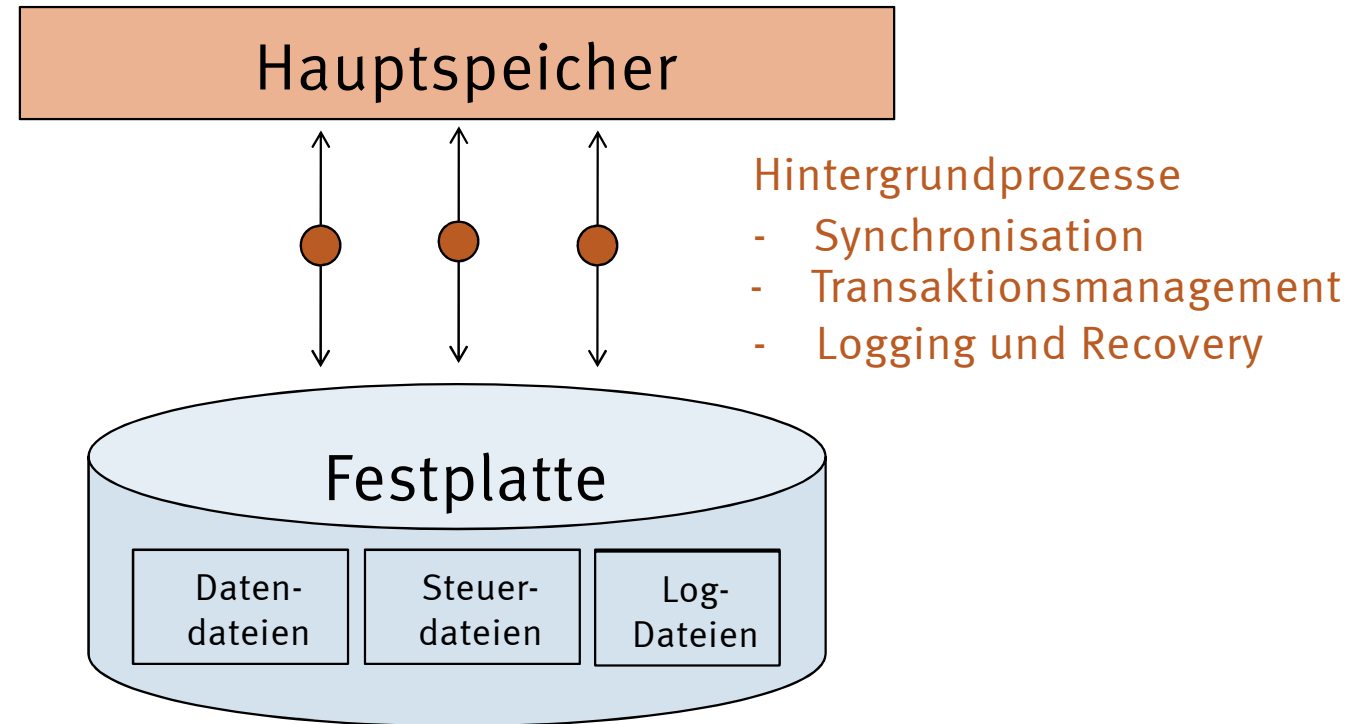


<http://redis.io/>

- Redis zählt zu den **Schlüssel-Wert-Speichern** (Key-Value Datenbank)
  - Key: Zeichenkette
  - Values: Datentypen Zeichenkette, **Listen, Mengen und Hashmaps**
  - Kapazität der Collection-Datentypen: maximal  $2^{32}$  (~ 4 Milliarden) Elemente pro Schlüssel
  
- Redis ist eine **InMemory-Datenbank**:
  - Alle Daten werden im Hauptspeicher gehalten
  - Performance:
    - Bis zu ca. 100.000 Schreibvorgänge pro Sekunde
    - Bis zu ca. 80.000 Lesevorgänge pro Sekunde
    - Schreibvorgänge sind schneller als Lesevorgänge

## Klassische Speicherhierarchie





Alle Daten werden im Hauptspeicher gehalten



# Master-Slave Architektur – Redis

Prof. Dr. I. M. Saatz

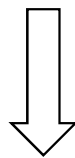
Moderne Datenbanken

Fachbereich Informatik

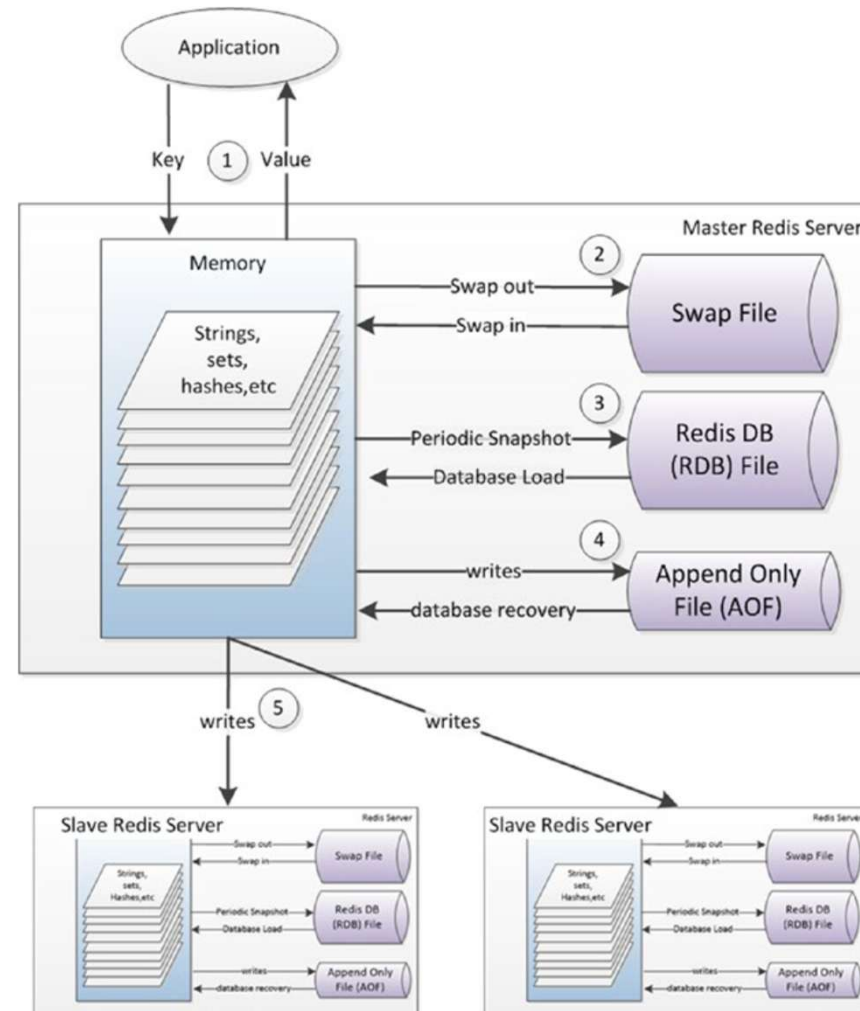
23

Bei der Master-Slave Architektur wird ein Knoten als Master-Knoten ausgezeichnet. Dessen Daten werden auf die abhängigen (Slave-) Knoten repliziert.

Änderung  
der Master-Daten



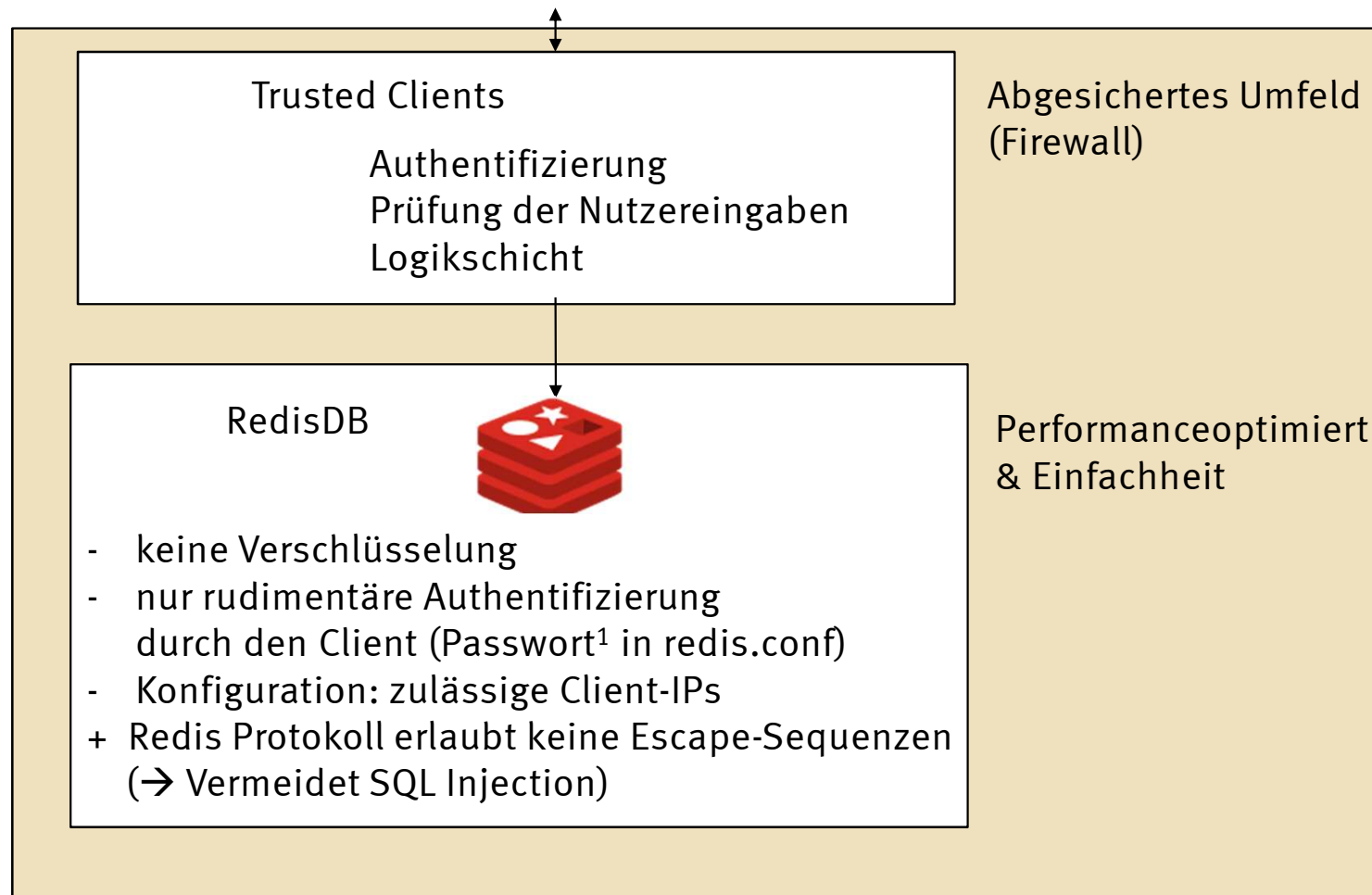
Propagation der  
Änderungen auf  
die Slaves erforderlich



aus: G. Harrison, Next Generation Databases, S. 94

Durch Konfigurationseinstellungen soll eine Balance zwischen Dauerhaftigkeit und Geschwindigkeit erreicht werden.

- Konfiguration des verfügbaren Hauptspeichers:
  - Hauptspeicher ist über Virtual Memory erweiterbar (zur Auslagerung von seltener benutzten Datensätzen)
  - Konfigurierbare Regeln für Verfallsdaten, so dass die Gesamtanzahl der Schlüssel nicht ungehindert anwächst
  - Beständigkeitsebenen und Replikationsoptionen
- Logging:
  - Zyklisches Schreiben der Daten auf Festplatte (Standardvariante)
  - Append-Only-Mode, bei dem Befehle, die zur Rekonstruktion des Datenbestands nach einem Systemcrash erforderlich sind, in ein Logfile geschrieben werden.
- Recovery:
  - (Fast) vollständige Wiederherstellung des Datenbestandes aus dem Log möglich.
  - "Append Only"-Modus benötigt einen höheren Zeitaufwand als die Standardvariante „zyklisches Schreiben“



<sup>1</sup>Redis ist sehr schnell, daher ist ein sehr (!) langes Passwort notwendig um brute force Angriffe abzuwehren.



# Agenda

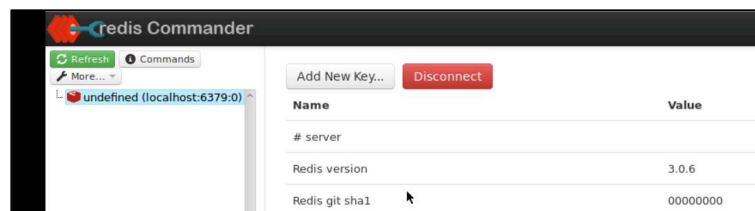
|       |                                      |    |
|-------|--------------------------------------|----|
| 1     | Beispiel: Datenanalyse               | 2  |
| 2     | Marktrecherche Key-Value Datenbanken | 14 |
| 3     | Key-Value Datenbank Redis            | 19 |
| <hr/> |                                      |    |
|       | 3.1 Redis - Befehle (Auswahl)        | 26 |
| <hr/> |                                      |    |
|       | 3.2 Javaschnittstelle JRedis         | 37 |
|       | 3.3 Beispiel: Evaluation             | 41 |

# Schnittstellen und Management-Werkzeuge

Kommandozeile:

```
C:\WINDOWS\system32\cmd.exe - redis-cli  
  
C:\Programme\redis>redis-cli ping  
PONG
```

Management Werkzeug: \$ redis-commander -p 8090



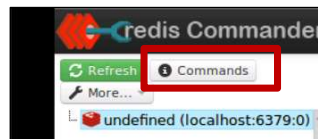
Programmierschnittstellen:

|              |         |           |       |            |             |
|--------------|---------|-----------|-------|------------|-------------|
| ActionScript | Bash    | C         | C#    | C++        | Clojure     |
| Common Lisp  | Crystal | D         | Dart  | Delphi     | Elixir      |
| emacs lisp   | Erlang  | Fancy     | gawk  | GNU Prolog | Go          |
| Haskell      | Haxe    | Io        | Java  | Julia      | Lasso       |
| Lua          | Matlab  | mruby     | Nim   | Node.js    | Objective-C |
| OCaml        | Pascal  | Perl      | PHP   | PL/SQL     | Pure Data   |
| Python       | R       | Racket    | Rebol | Ruby       | Rust        |
| Scala        | Scheme  | Smalltalk | Swift | Tcl        | VB          |
| VCL          | Xojo    |           |       |            |             |

- Schlüssel-Wert-Speicher (Key-Value Datenbank)
  - Key: Zeichenkette
  - Values: Datentypen Zeichenkette, **Listen, Mengen und Hashmaps**
  - Kapazität der Collection-Datentypen: maximal  $2^{32}$  (~ 4 Milliarden) Elemente pro Schlüssel
- Konvention für die Bezeichnung der Befehle:

| Befehl | Bezug                               |
|--------|-------------------------------------|
| S...   | SET-Befehl                          |
| H...   | Hash-Befehl                         |
| Z...   | Sortierte Liste                     |
| L...   | Listenbefehl (Operation von links)  |
| R...   | Listenbefehl (Operation von rechts) |

- Befehlsübersicht: <http://redis.io/commands>



## ■ Grundlegende Befehle:

- Schreiben eines Wertpaares: `set <key> <value>`
- Lesen eines Wertpaares: `get <key>`

## ● Beispiel – “Hello world”

- Schlüssel-Wertpaar anlegen

```
redis> SET „hello“ „world“  
OK  
  
redis> GET „hello“  
„world“
```

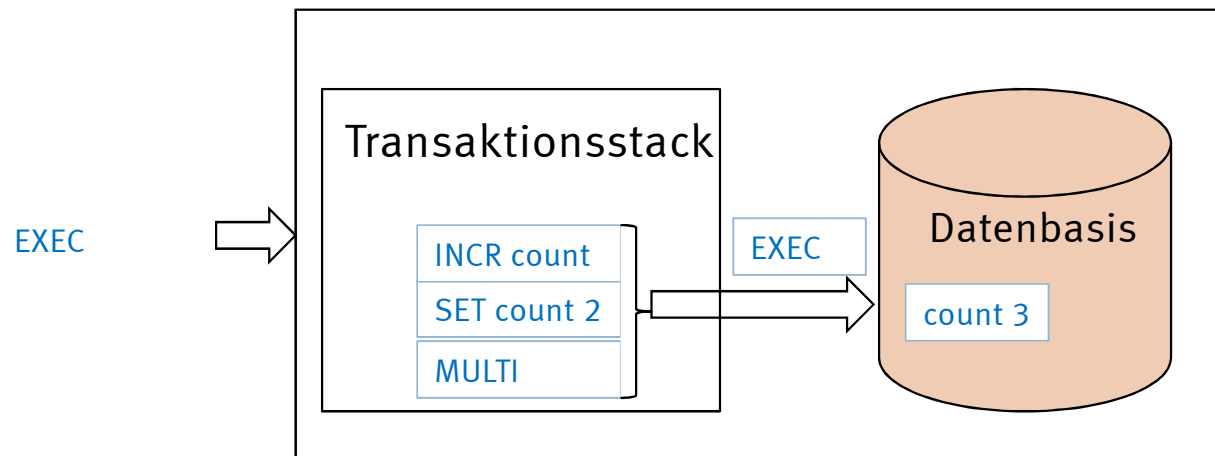
- Liefere alle Schlüssel:

```
127.0.0.1:6379> keys *  
1) "hello"  
2) "mykey"  
3) "foo"  
4) "bild"
```

# Redis - Grundlegende Befehle

| Befehl          | Bedeutung                                                                       | Beispiel                                   |
|-----------------|---------------------------------------------------------------------------------|--------------------------------------------|
| SET<br>GET      | Einfügen bzw. Auslesen eines Schlüssel-Wert-Paares                              | SET key1 v1<br>GET key1 key2               |
| DEL             | Schlüssel-Wert Paare löschen                                                    | DEL key 1 [key 2 ...]                      |
| MSET<br>MGET    | Einfügen bzw. auslesen von mehreren Schlüssel-Wert-Paaren                       | MSET key1 v1 key2 v2<br>MGET key1 key2 ... |
| INCR,<br>INCRBY | Inkrementieren von Ganzzahlen um 1 bzw. um beliebigen Wert.                     | SET count 2<br>INCR count<br>DECR count    |
| DECR,<br>DECRBY | Dekrementieren von Ganzzahlen um 1 bzw. um beliebigen Wert.                     |                                            |
| EXPIRE<br>SETEX | Relative Verfallszeit in Sekunden festlegen<br>Einfügen und Verfallszeit setzen | EXPIRE count 10<br>SETEX key value 10      |
| EXISTS          | Existenzabfrage                                                                 | EXISTS count                               |
| PERSIST         | Aufheben der Verfallszeit (solange Schlüssel-Wert-Paar noch existiert)          | PERSIST count                              |

| Befehl  | Bedeutung                                                                                              | Beispiel                                   |
|---------|--------------------------------------------------------------------------------------------------------|--------------------------------------------|
| MULTI   | Befehle werden auf den Transaktionsstack gelegt, aber noch nicht ausgeführt.                           | MULTI<br>SET count 2<br>INCR count<br>EXEC |
| EXEC    | Befehle auf dem Transaktionsstack werden ausgeführt.                                                   |                                            |
| DISCARD | Befehle auf dem Transaktionsstack werden gelöscht, d.h. die Ausführung der Transaktion wird storniert. |                                            |



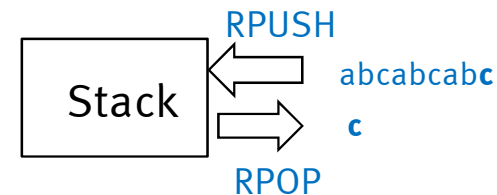
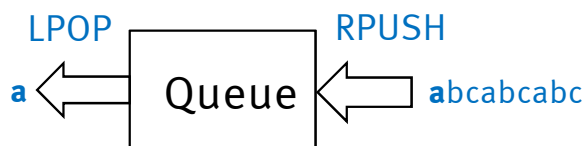
HashMaps können genutzt werden zur Zusammenfassung von einzelnen Schlüssel-Wert-Paaren. Eine Verschachtelung von HashMaps ist nicht möglich. Es können in HashMaps nur Zeichenketten gespeichert werden.

| Befehl           | Bedeutung                               | Beispiel                                     |
|------------------|-----------------------------------------|----------------------------------------------|
| HMSET<br>HGETALL | Einfügen bzw. auslesen eines Hash-Werts | HMSET hmkey key1 v1 key2 v2<br>HGETALL hmkey |

Einfügen von Personendaten:

|                                                              |                                                                                                                                                                                                            |
|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Key:<br>pid<br>Values:<br>Map{ pid,<br>nachname,<br>vorname} | <pre>127.0.0.1:6379&gt; hmset 2310 pid 2310 nachname Meitner vorname Lise OK 127.0.0.1:6379&gt; hgetall 2310 1) "pid" 2) "2310" 3) "nachname" 4) "Meitner" 5) "vorname" 6) "Lise" 127.0.0.1:6379&gt;</pre> |
|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Listen enthalten mehrere geordnete Werte und können sowohl als Queues als auch Stacks genutzt werden.



| Befehl         | Bedeutung                                                                                             | Beispiel                                                                                  |
|----------------|-------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| RPUSH<br>LPUSH | Einfügen in die Liste von Rechts bzw. Links                                                           | <pre>127.0.0.1:6379&gt; RPush liste a b c a b c a b c (integer) 9</pre>                   |
| RPOP<br>LPOP   | Auslesen der Liste von Rechts bzw. Links                                                              | <pre>127.0.0.1:6379&gt; LPop liste "a" 127.0.0.1:6379&gt; LLen liste (integer) 8</pre>    |
| LLEN           | Länge der Liste                                                                                       |                                                                                           |
| LREM           | Löschen von n-Werten der Liste zu einem Schlüssel. Ist n negativ, so wird vom Ende der Liste gezählt. | <pre>LREM liste -1 b</pre> <p><b>bcabcabc</b> <math>\Rightarrow</math> <b>bcabcac</b></p> |
| BRPOP          | Gepuffertes Auslesen aus einer Liste mit einem Timeout von n Sekunden.                                | <pre>n=100 127.0.0.1:6379&gt; BRPOP liste 100 1) "liste" 2) "c"</pre>                     |



Sets sind ungeordnete Mengen ohne Duplikate, mit denen Mengenoperationen auf mehreren Schlüsselwerten ausgeführt werden können.

| Befehl   | Bedeutung                               | Beispiel                                 |
|----------|-----------------------------------------|------------------------------------------|
| SADD     | Einfügen von Werten in eine Menge       | SADD gerade 0 2 4<br>SADD ungerade 0 1 3 |
| SMEMBERS | Auslesen der Menge                      | SMEMBERS ungerade                        |
| SINTER   | Schnittmenge bilden                     | SINTER gerade ungerade                   |
| SDIFF    | Differenzmenge bilden                   | SDIFF gerade ungerade                    |
| SUNION   | Vereinigungsmenge bilden ohne Duplikate | SUNION gerade ungerade                   |
| SREM     | Löschen von Elementen                   | SREM ungerade 0                          |

In einer sortierten Menge wird jedem Wert zusätzlich ein numerischer Wert (Gewicht) zugeordnet, anhand dessen die gesamte Menge sortiert wird.

| Befehl                                                                | Bedeutung                                                                                                          | Beispiel                                                                                |
|-----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| ZADD                                                                  | Einfügen von Werten in eine Menge                                                                                  | ZADD menge 5 E 3 C 1 A<br>ZADD menge2 2 A                                               |
| ZINCR<br>ZINCRBY                                                      | Inkrementieren eines Wertes                                                                                        | ZINCRBY menge 5 C<br>→ E5 C8 A1                                                         |
| ZRANGE<br>ZREVRANGE                                                   | Abfrage der Position von .. bis<br>- WITHSCORES inkl. Gewicht<br>- REV umgekehrte Reihenfolge                      | ZRANGE menge 0 1 WITHSCORES<br>→ A1 E5                                                  |
| ZRANGEBYSCORE                                                         | Abfrage der Werte von .. bis<br>(inkl. Grenzen)<br>inf: „unendlich“ (infinity)                                     | ZRANGEBYSCORE menge 3 inf<br>→ E C                                                      |
| ZUNIONSTORE<br>ziel n<br>schlüsselliste<br>[AGGREGATE<br>SUM MIN MAX] | Vereinigungsmenge <ziel><br>bilden aus einer Anzahl von n<br>sortierten Mengen ggf. unter<br>Aggregation der Werte | ZUNIONSTORE vereinigung 2<br>menge menge2<br>AGGREGATE SUM<br>↓<br>vereinigung A3 E5 C8 |

# Agenda

|     |                                      |    |
|-----|--------------------------------------|----|
| 1   | Beispiel: Datenanalyse               | 2  |
| 2   | Marktrecherche Key-Value Datenbanken | 14 |
| 3   | Key-Value Datenbank Redis            | 19 |
| 3.1 | Redis - Befehle (Auswahl)            | 26 |
| 3.2 | Javaschnittstelle JRedis             | 37 |
| 3.3 | Beispiel: Evaluation                 | 41 |

Um aus Java heraus mit der Datenbank zu arbeiten, können verschiedene Libraries genutzt werden.

Hier: Jedis-Library

Link: <https://github.com/xetorthio/jedis>

Verbindung zur Redis-Datenbank herstellen mit Jedis:

```
public class RedisZugriff {  
    private static final String HOST = "172.22.114.8";  
    private static final int PORT = 6379;  
    private Jedis jedis;  
    private static String password = "1f984967df71201576ce3d44f956ef9f";  
  
    RedisZugriff(){  
        if (jedis == null ) {  
            jedis = new Jedis(HOST, PORT);  
            String a = jedis.auth(password);  
        }  
    }  
}
```

Jedis-2.1.0.jar im Verzeichnis libs einfügen und in Buildpath einbinden

- Interaktion aus einem Java-Programm heraus:
  - Hashwerte speichern und auslesen:

```
JRedis jredis = new JRedisClient();  
// Default-Werte fuer Host (localhost) und Port (6379)  
  
jredis.set("erster_schluessel", "Hello World");  
  
System.out.println(jredis.get("erster_schluessel"));
```

- Auslesen aller vorhandenen Schlüssel (Befehl KEYS \* ):

```
Set<String> keys = jedis.keys("*");
```

Einfügen einer Map mit dem Key pid:

```
jedis.connect();
if (jedis.isConnected()) {
    Map <String, String> map = new HashMap<String, String>();
    if (pid != null){
        map.put("pid", pid);
        if (nachname != null)
            map.put("nachname", nachname);
        if (vorname != null)
            map.put("vorname", vorname);
        jedis.hmset(pid, map);
        System.out.println("Person geschrieben");
    }else{
        System.out.println("Pid ist null");
    }
    jedis.disconnect();
}
```

Key:  
pid  
Values:  
Map{ pid,  
nachname,  
vorname}

Auslesen der Map zum Key pid:

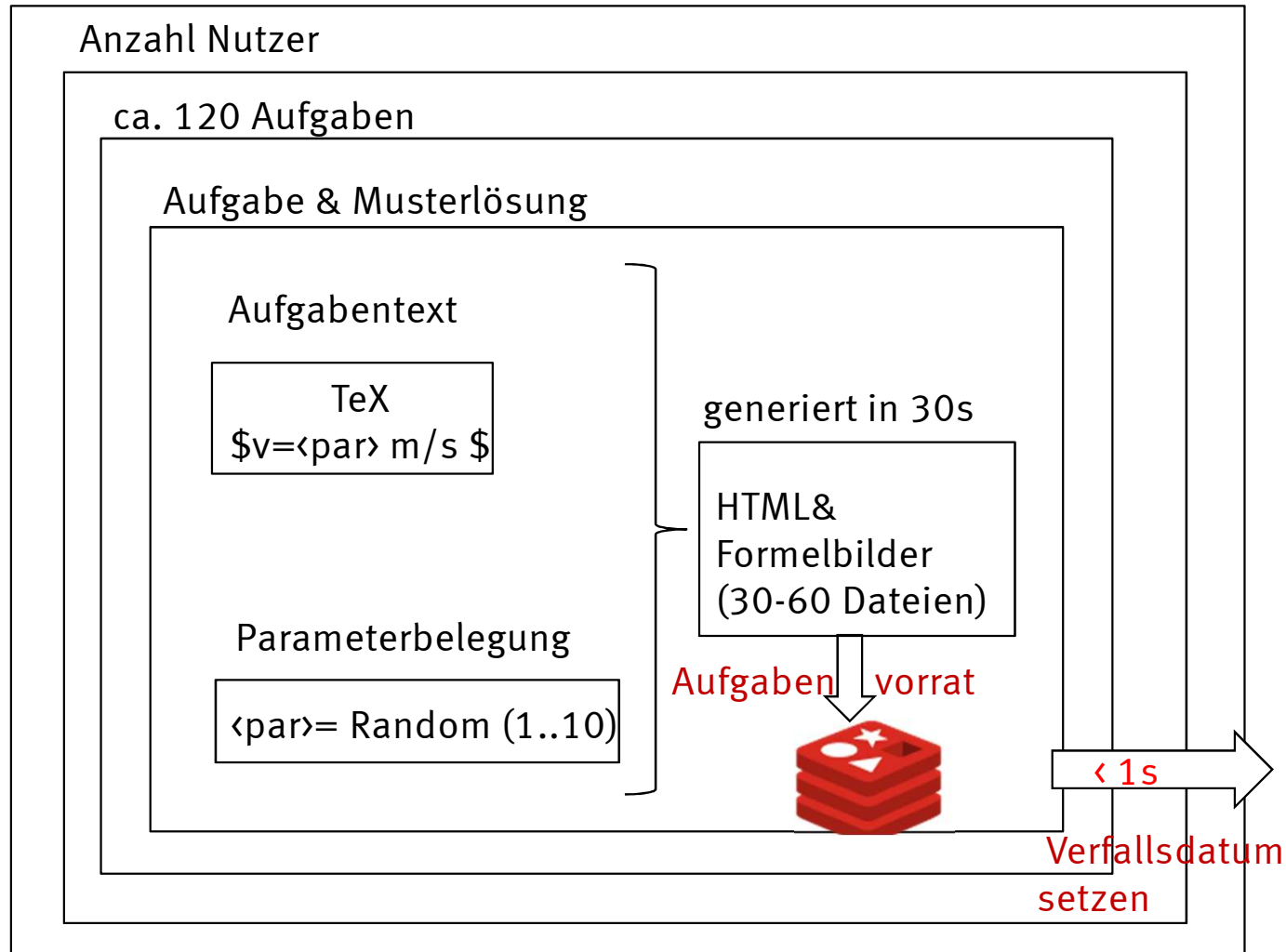
```
Map<String, String> person = jedis.hgetAll(pid);
```

# Agenda

|     |                                      |    |
|-----|--------------------------------------|----|
| 1   | Beispiel: Datenanalyse               | 2  |
| 2   | Marktrecherche Key-Value Datenbanken | 14 |
| 3   | Key-Value Datenbank Redis            | 19 |
| 3.1 | Redis - Befehle (Auswahl)            | 26 |
| 3.2 | Javaschnittstelle JRedis             | 37 |
| 3.3 | Beispiel: Evaluation                 | 41 |

# Problemstellung: Physik-Aufgabentrainer

Semester





```
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <link href="default.css" rel="stylesheet" type="text/css" />
8 <link href="cssLayout.css" rel="stylesheet" type="text/css" />
9 <script language="javascript" type="text/javascript" src="script.js"></script>
10 <script src="http://code.jquery.com/jquery-latest.js"></script>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
12 <title>Webdis</title>
13 </head>
14 <body>
15 <?php
16 require "predis-1.1/Autoload.php";
17 Predis\Autoloader::register();
18 $redis = new Predis\Client();
19 $redis = new Predis\Client('tcp://127.0.0.1:6379');
20 if($redis)
21 {
22     echo "Redis connected succesfully <p>";
23     $redis->set('bild', file_get_contents("testbild.png"));
24     $value = $redis->get('bild');
25     $handle = fopen ("gelesenesBild.png", "w");
26     fwrite ($handle, $value);
27     fclose ($handle);
28     echo "Datei geschrieben";
29 }
30 else
31 {
32     echo "Redis Not connected";
33 }
34 ?>
35 <!DOCTYPE html>
36
37 </body>
38
39 </html>
```

Einbindung des  
PHP-Package PRedis

Verbindungsaufbau

Speicherung und Laden  
eines Bildes

30-50 Bilder



- Lernmodul „InMemory und Key-Value Datenbanken“
- Praktikumsaufgabe zu Redis (pdf im Redis-Ordner)
- Abgabe Projektidee (in ILIAS)

## Reflexion Schlüssel-Wert Speicher

- 1) Welche Datenformate können in Schlüssel-Wert Speichern abgelegt werden?
- 2) Was sind die Eigenschaften und Grenzen einer InMemory-Datenbank?
- 3) Welche Möglichkeiten bietet Redis, um Daten zu speichern, zu ändern und auszuwählen?
- 4) Wie werden Transaktionen verwaltet?
- 5) Welche Vor- und Nachteile bieten der Einsatz einer von Schlüssel-Wert Speichern?
- 6) Was gibt es für sinnvolle Anwendungsszenarien?

# Vielen Dank für Ihre Aufmerksamkeit

- [EFHB 2010]  
S. Edlich, A. Friedland, J. Hampe, B. Brauer  
NoSQL – Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken  
Hanser Verlag 2010
- [RW 2012] E. Redmond, J. Wilson, Sieben Wochen, sieben Datenbanken,  
o'Reilly-Verlag
- [SF 2013] P. Sadalage, M. Fowler, NoSQL Distilled, Addison-Wesley