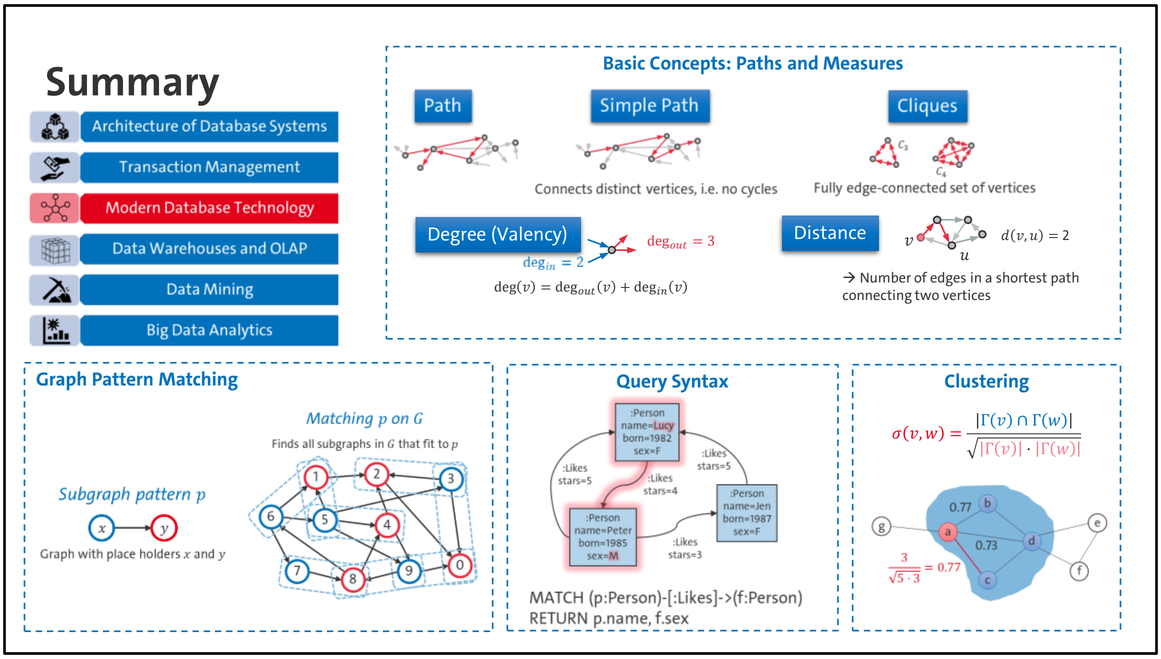

Summary
Architecture of Database Systems
Transaction Management
Modern Database Technology
Data Warehouses and OLAP
Data Mining
Big Data Analytics
Basic Concepts: Paths and Measures
Path
Simple Path
Connects distinct vertices, i.e. no cycles
Cliques
Fully edge-connected set of vertices
Degree (Valency)
$\deg_{out} = 3$
$\deg_{in} = 2$
$\deg(v) = \deg_{out}(v) + \deg_{in}(v)$
Distance
$d(v,u) = 2$
→ Number of edges in a shortest path connecting two vertices
Graph Pattern Matching
Subgraph pattern p
Graph with place holders x and y
Matching p on G
Finds all subgraphs in G that fit to p
Query Syntax
:Person name=Lucy born=1982 sex=F
:Person name=Peter born=1985 sex=M
:Person name=Jen born=1987 sex=F
:Likes stars=5
:Likes stars=4
:Likes stars=3
MATCH (p:Person)-[:Likes]->(f:Person)
RETURN p.name, f.sex
Clustering
$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)| \cdot |\Gamma(w)|}}$
$\frac{3}{\sqrt{5 \cdot 3}} = 0.77$
0.77
0.73

Research News – This month @ ICDE

- **Yet another B-Tree paper**: *Improving the Relationship between B+-Tree and Memory Allocator for Persistent Memory*, Wei Yan (Xi'an Jiaotong University); Xingjun Zhang (Xi'an Jiaotong University)
- **…and another one:** *Bwe-tree: an Evolution of Bw-tree on Fast Storage,* Rui Wang (Alibaba Group); xinjun Yang (Alibaba Group); Feifei Li (Alibaba Group); David B Lomet (retired); Xin Liu (Alibaba Inc); panfeng zhou (Alibaba); Yongxiang Chen (Alibaba Group); David Zhang (Alibaba Group); Jingren Zhou (Alibaba Group); Jiesheng Wu (Alibaba)
- **5 papers about LSM trees** (which we will cover when we talk about KV stores)
- **Multiple papers about clustering**, e.g. *McCatch: Scalable Microcluster Detection in Dimensional and Nondimensional Datasets*, Braulio Sánchez (University of São Paulo); Robson Cordeiro (University of São Paulo); Christos Faloutsos (CMU)
- An **invited talk about Duckdb's approach to benchmarks** with a reference to a paper describing a **PGQ implementation for DuckDB:** *DuckPGQ: Efficient Property Graph Queries in an analytical RDBMS*, Wolde et al., CIDR'23
- Multiple papers on **transaction processing and/or recovery**, e.g. *Fast Parallel Recovery for Transactional Stream Processing on Multicores*, Jianjun Zhao (Huazhong University of Science and Technology); Haikun Liu (Huazhong University of Science and Technology); Shuhao Zhang (Nanyang Technological University); Zhuohui Duan (Huazhong University of Science and Technology); Xiaofei Liao (HUST); Hai Jin (Huazhong University of Science and Technology); Yu Zhang (Huazhong University of Science and Technology)

The Bw tree is controversial within the database community with opinions ranging from "The 'Influential Paper award' at ICDE23 was totally justified." to "'Influential' does not necessarily mean that it's a good influence."

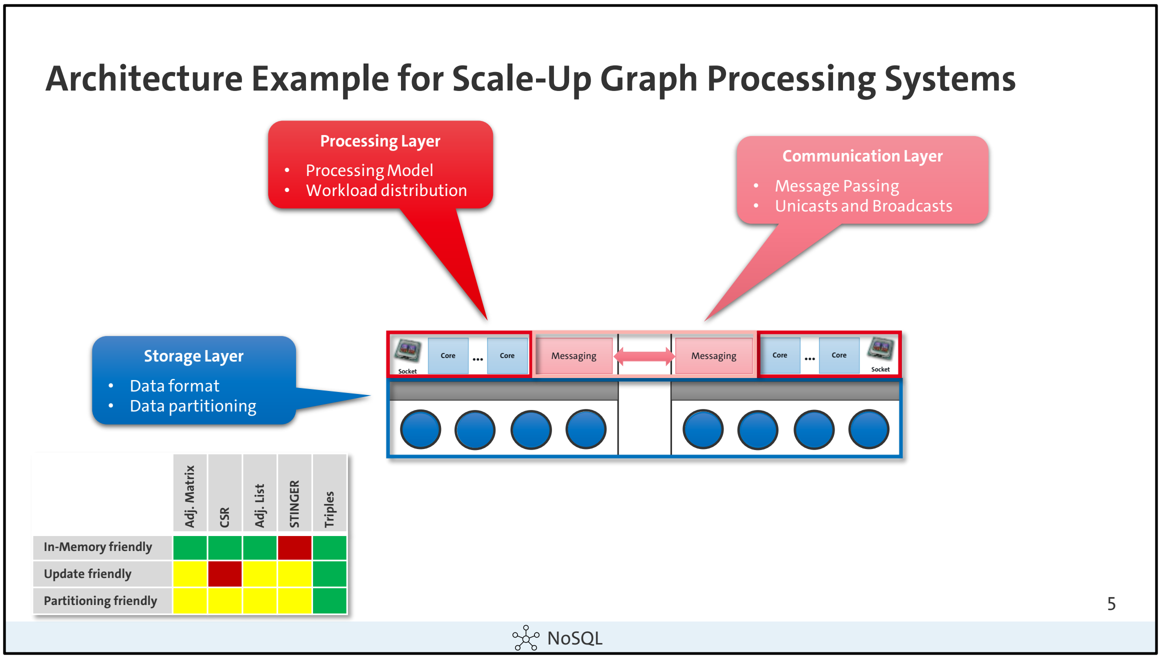
Course Outline

- Architecture of Database Systems
- Transaction Management
- Modern Database Technology
  - Distributed Systems
  - Optimizations in RDBMS
  - **NoSQL**
- Data Warehouses and OLAP
- Data Mining
- Big Data Analytics

Graph Processing

- Models ✓
- PGM ✓
- Representations ✓
- Queries ✓
- Use Cases ✓
- Challenges ⬅

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG



- Challenges grow with the system → We will have a look at graph query processing in scale-up systems
- Each layer introduces different challenges:
  - Communication Layer: Scalability (communication must not impact scaling), NUMA friendly (optimizable for concurrency and remote communication)
  - Processing Layer: Flexibility (adapt to workload changes and parallel queries), Parallelism (leverage hardware)
  - Storage Layer: In-Memory friendly, update friendly (easy and fast updates, from a system perspective), partitioning friendly (especially if data model is altered)

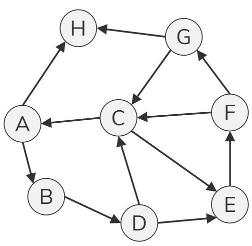
Further reading

Another storage format for graphs: Bader, David A., et al. "Stinger: Spatio-temporal interaction networks and graphs (sting) extensible representation." *Georgia Institute of Technology, Tech. Rep* (2009).
Graph pattern matching on scale-up systems: Krause, Alexander, et al. "Asynchronous graph pattern matching on multiprocessor systems." *New Trends in Databases and Information Systems: ADBIS 2017 Short Papers and Workshops, AMSD, BigNovelTI, DAS, SW4CH, DC, Nicosia, Cyprus, September 24–27, 2017, Proceedings 21*. Springer International Publishing, 2017.

Graph Partitioning Classification

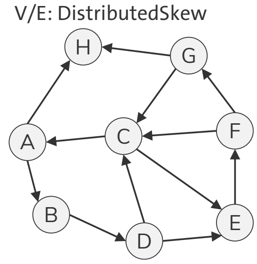
Edges

- Partition based solely on edges
- Algorithm
  - RoundRobin



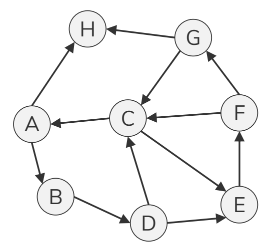
Vertices

- Partition by Vertices, i.e. distribute a vertex and its adjacency to a partition
- Algorithms
  - V/V: RoundRobinVertices
  - V/E: BalancedEdges
  - V/E: DistributedSkew



Components

- Partition by components, distribute a vertex and its adjacency to a partition
- Algorithm
  - C/V: k-Way



Alexander Krause, et al.: **Partitioning Strategy Selection for In-Memory Graph Pattern Matching on Multiprocessor Systems**. Euro-Par 2017: 149-163

6

- There is no single best algorithm
- The most suited algorithm depends on the workload and the data

Further Reading

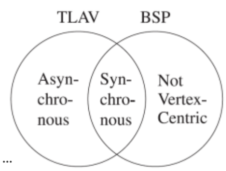
George Karypis and Vipin Kumar: A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm, PPSC. 1997

Alexander Krause, et al.: Partitioning Strategy Selection for In-Memory Graph Pattern Matching on Multiprocessor Systems. Euro-Par 2017: 149-163

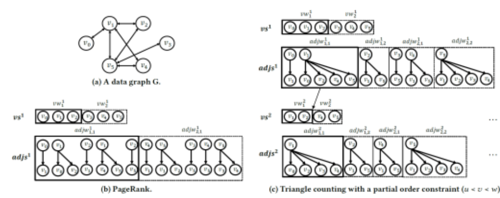
Graph Processing

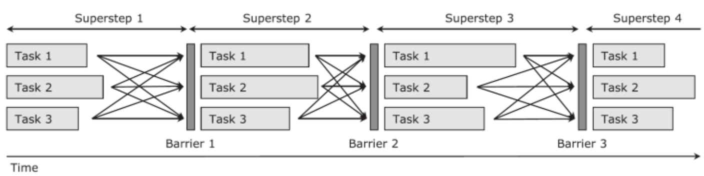
Bulk Synchronous Processing / Think Like A Vertex

- Multistep Process
- MPI for communication
- Local computation
- Exchange intermediates
- Giraph, GraphX, Pregel, Powergraph, …



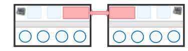
Nested Windowed Streaming Model





	TLAV	NWSM
Flexibility		
Parallelism		

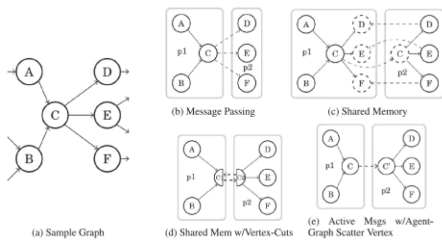
Robert Ryan McCune, et al.: **Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing**. ACM Comput. Surv. 48(2): 25:1-25:39 (2015),
Seongyun Ko, et al.: **Turbograph++: A Scalable and Fast Graph Analytics System**, SIGMOD, 2018



Communication Models

Selected Available Protocols

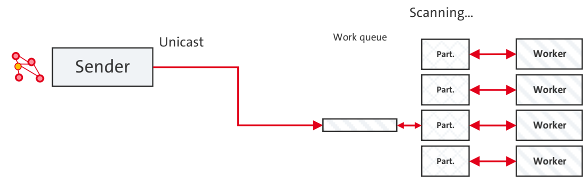
- Shared Memory
- Message Passing / Delegation



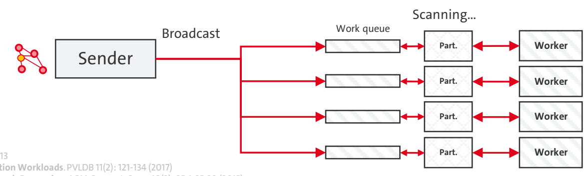
	Shared Memory	Delegation
Scalability		
NUMA friendly		

Messaging

- Unicasts for direct lookups

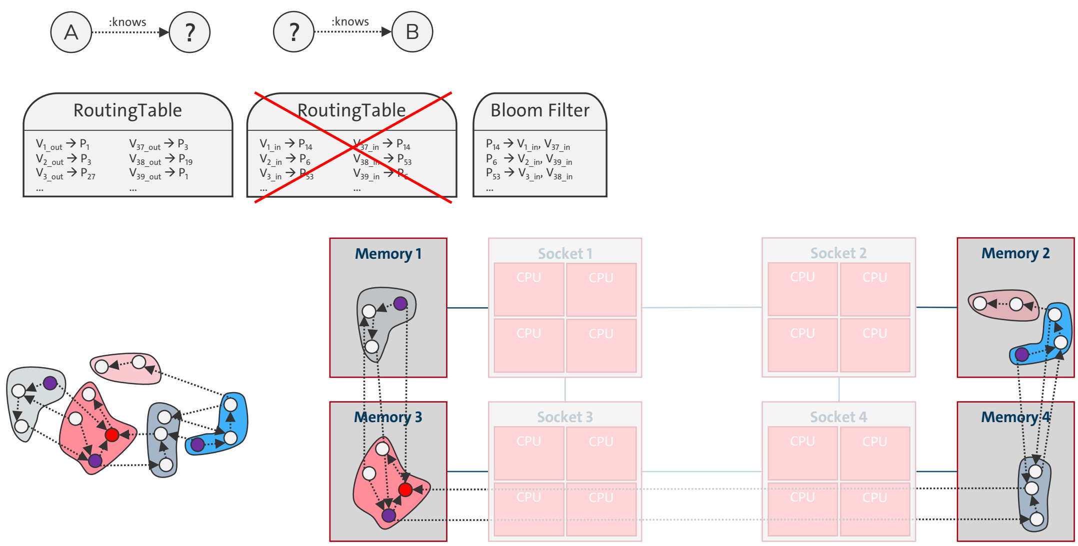


- Broadcasts for low selectivity or missing locality information



Irina Calciu, et al.: **Message Passing or Shared Memory: Evaluating the Delegation Abstraction for Multicores**, OPODIS, 2013
Raja Appuswamy, et al.: **Analyzing the Impact of System Architecture on the Scalability of OLTP Engines for High-Contention Workloads**. PVLDB 11(2): 121-134 (2017)
Robert Ryan McCune, et al.: **Thinking Like a Vertex: A Survey of Vertex-Centric Frameworks for Large-Scale Distributed Graph Processing**. ACM Comput. Surv. 48(2): 25:1-25:39 (2015)

Messaging for Graph Pattern Matching



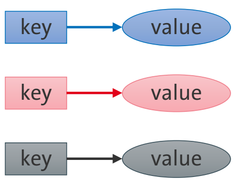
Course Outline

- Architecture of Database Systems
- Transaction Management
- Modern Database Technology
  - Distributed Systems
  - Optimizations in RDBMS
  - **NoSQL**
    - **Graph Processing**
      - Models ✓
      - PGM ✓
      - Representations ✓
      - Queries ✓
      - Use Cases ✓
      - Challenges ✓
    - **Key-Value Stores**
- Data Warehouses and OLAP
- Data Mining
- Big Data Analytics

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

NoSQL Databases: Key-Value Stores

Key-Value Stores



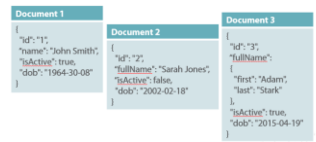
Get … a value by a given key

Put … a new key-value pair into the database

Delete … a key and the associated value

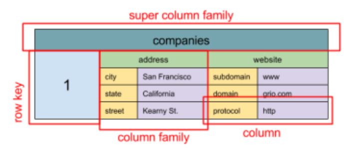
Value is a (semi) structured document

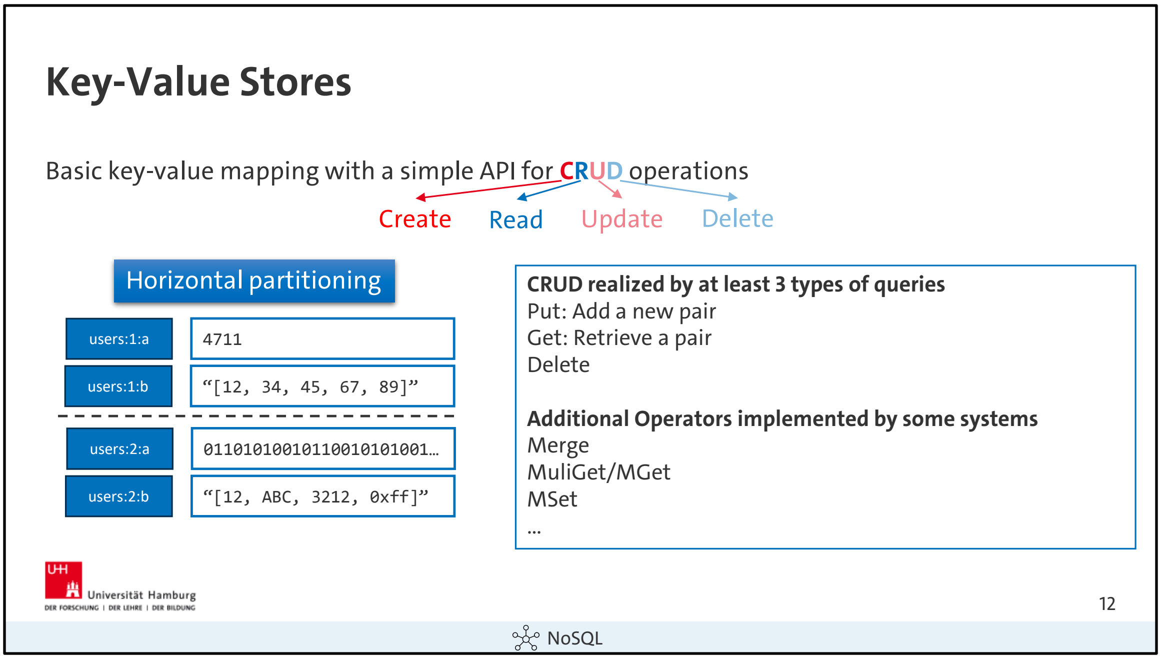
Document Stores



Value is a row or a table

Wide Column Stores





Motivation

- Basic key-value mapping via simple API (more complex data models can be mapped to key-value representations)
- Reliability at massive scale on commodity HW (cloud computing)

System Architecture

- Database is a collection of key/value pairs
- Key-value maps, where values can be of a variety of data types
- The key for each pair is unique
- APIs for CRUD operations (create, read, update, delete)
- Scalability via sharding (horizontal partitioning)

Example Systems

- Redis (2009, CP/AP) → not a plain kv-store, but “data structure server” with persistent log
- LevelDB
- RocksDB
- Memcached (started as a simple caching tool)

Futher Reading

Giuseppe DeCandia et al: Dynamo: amazon's highly available key-value store. SOSP 2007

Examples

Memcached via telnet

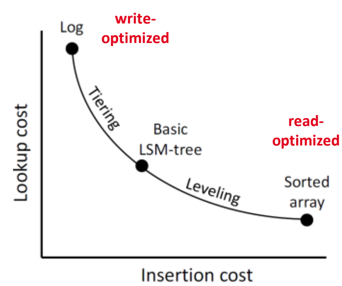
Add a new KV-pair
set AgeAlice 0 120 1 [Press Enter]
26 [Press Enter]
Retrieve a KV-pair
get AgeAlice
Delete a KV-pair
delete AgeAlice

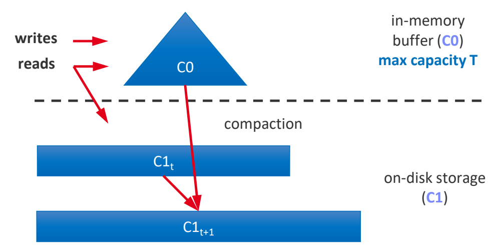
RocksDB with C++

Add a new KV-pair
database->Put(WriteOptions(), AgeAlice, “26”);
Retrieve a KV-pair
std::string content;
database->Get(ReadOptions(), AgeAlice, &content);
Delete a KV-pair
database->Delete(WriteOptions(), AgeAlice)

Syntax of Set in Memcached via telnet
set KEY META_DATA EXPIRATION_TIME_IN_S LENGTH_IN_BYTES
[Press Enter]
VALUE
[Press Enter]

Log-Structured Merge Trees (LSM Trees)





LSM Overview

- Many KV-stores rely on LSM-trees as their storage engine (e.g., BigTable, DynamoDB, LevelDB, Riak, RocksDB, Cassandra, HBase)
- Approach: Buffers writes in memory, flushes data as sorted runs to storage, merges runs into larger runs of next level (compaction)

System Architecture

- Writes in C0
- Reads against C0 and C1 (w/ buffer for C1)
- Compaction (rolling merge): sort, merge, includingdeduplication

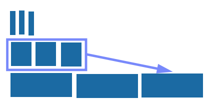
Further Reading

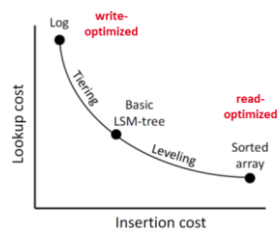
Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, Elizabeth J. O'Neil: The Log-Structured Merge-Tree (LSM-Tree). Acta Inf. 1996

LSM Trees: Merging

LSM Tiering

- Keep up to T-1 (sorted) runs per level L
- Merge all runs of L_i into 1 run of L_{i+1}
  - L1
  - L2
  - L3





LSM Leveling

- Keep 1 (sorted) run per level L
- Sort-Merge run of L_i with L_i+1
  - L1
  - L2
  - L3

Storing key-value pairs: Sorted String Tables (sst)

All keys are sorted within an sst file
→ New sst file if new key doesn't keep order

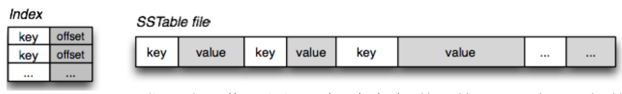
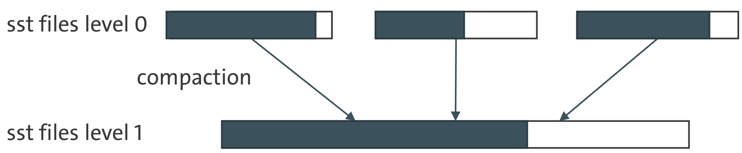


**image: https://www.igvita.com/2012/02/06/sstable-and-log-structured-storage-leveldb/*



→ Up to 7 levels (originally developed for LevelDB)

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Sorted String Tables

- Popular format for key-value stores
- Stores data in a compact format
- Basically implements an lsm tree
- For large SSTables, index can be kept separately in-memory

SSTables example

K,V-Pairs are inserted in the following order:

(2,'an')
(3, 'example.')
(0,'This')
(1,'is')

Assume each key is stored as a character and each character is stored using 1 byte and memory is byte-addressable

Index 1

2	0
3	3

Index 2

0	0
1	5

SSTable 1

2	an	3	an example.

SSTable 2

0	This	1	is

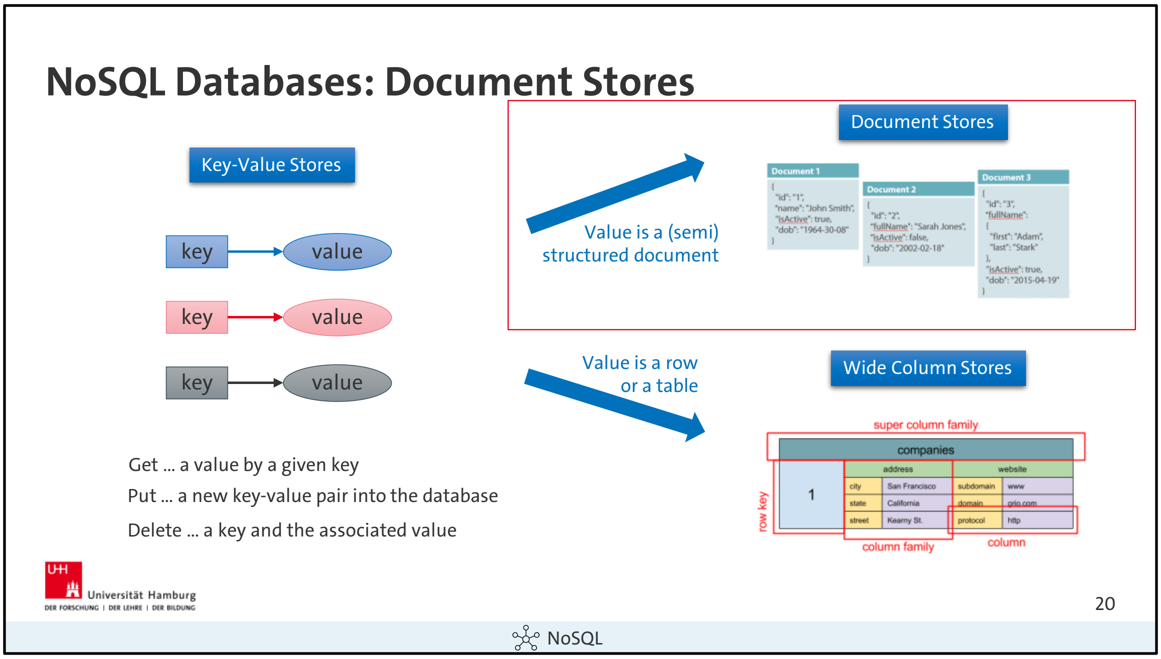
In practice, the length of a key is no necessarily constant
→ Delimiters or a fixed maximum key size are used to separate the key from the offset

Exercise: SSTables

- Insert the following (key,value)-pairs into an empty sst:

  (5,’!’), (0,’H’), (1,’ey, h’),(3,’l’),(4,’o’), (2,’el’)

- Assume each character takes up 1 Byte, each key is a character, and memory is byte-addressable
  - How many SSTables are created?
  - What do they look like?
  - What does the index look like?

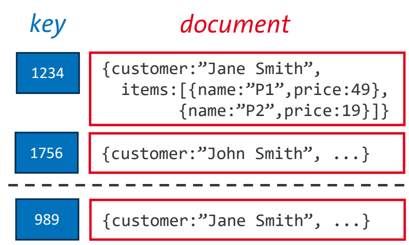


Basic Data Model

- The general notion of a document – words, phrases, sentences, paragraphs, sections, subsections, footnotes, etc.
- Flexible schema: subcomponent structure may be nested, and vary from document-to-document
- Essentially, they support the embedding of documents and arrays within other documents and arrays
- Document structures do not have to be predefined, so are schema-free (XML, mostly JSON)

Document Stores

- Application-oriented management of structured, semi-structured, and unstructured information
- Collections of (key, document)-pairs



Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Motivation
- Application-oriented management of structured, semi-structured, and unstructured information
- Scalability via parallelization on commodity HW (cloud computing)

System Architecture
- Collections of (key, document)
- Scalability via sharding (horizontal partitioning)
- Custom SQL-like or functional query languages

Example Systems
- MongoDB (C++, 2007, CP) → RethinkDB, Espresso, Amazon DocumentDB (Jan 2019)
- CouchDB (Erlang, 2005, AP) → CouchBase

Recap: JSON (JavaScript Object Notation)

JSON Data Model

- Data exchange format for semi-structured data
- Not as verbose as XML (especially for arrays)
- Popular format

```
{"students:"[
  {"id": 1, "courses":[
    {"id":"INF.01017UF", "name":"DM"},
    {"id":"706.550", "name":"AMLS"}]},
 {"id": 5, "courses":[
    {"id":"706.520", "name":"DIA"}]},
]}
```

Query Languages

- Most common: libraries for tree traversal and data extraction
- JSONig: XQuery-like query language
- JSONPath: XPath-like query language

JSONiq Example:

```
declare option jsoniq-version "…";
for $x in collection("students")
  where $x.id lt 10
  let $c := count($x.courses)
  return {"sid":$x.id, "count":$c}
```

[http://www.jsoniq.org/docs/JSONiq/html-single/index.html]

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Example MongoDB

[**Credit:** https://api.mongodb.com/python/current]

Creating a Collection

```
import pymongo as m
conn = m.MongoClient(“mongodb://localhost:123/”)
db = conn[“dbs19”]      # database dbs19
cust = db[“customers”]  # collection customers
```

Inserting into a Collection

```
mdict = {
  “name“: “Jane Smith”,
  “address”: “Inffeldgasse 13, Graz”
}
id = cust.insert_one(mdict).inserted_id
# ids = cust.insert_many(mlist).inserted_ids
```

Querying a Collection

```
print(cust.find_one({"_id": id}))

ret = cust.find({"name": "Jane Smith"})
for x in ret:
  print(x)
```