Information Engineering 2

NoSQL Systems

Prof. Dr. Kurt Stockinger

Semesterplan

SW	Datum	Vorlesungsthema	Praktikum
1	23.02.2022	Data Warehousing Einführung	Praktikum 1: KNIME Tutorial
2	02.03.2022	Dimensionale Datenmodellierung 1	Praktikum 1: KNIME Tutorial (Vertiefung)
3	09.03.2022	Dimensionale Datenmodellierung 2	Praktikum 2: Datenmodellierung
4	16.03.2022	Datenqualität und Data Matching	Praktikum 3: Star-Schema, Bonus: Praktikum 4: Slowly Changing Dimensions
5	23.03.2022	Big Data Einführung	DWH Projekt - Teil 1
6	30.03.2022	Spark - Data Frames	DWH Projekt - Teil 2 (Abgabe: 4.4.2022 23:59:59)
7	06.04.2022	Data Storage: Hadoop Distributed File System & Parquet	Praktikum 1: Data Frames
8	13.04.2022	Query Optimization	Praktikum 2: Data Storage
9	20.04.2022	Spark Best Practices & Applications	Praktikum 3: Query Optimization & Performance Analysis
10	27.04.2022	Machine Learning mit Spark 1	Praktikum 3: Query Optimization & Performance Analysis (Vertiefung)
11	04.05.2022	Machine Learning mit Spark 2 + Q&A	Praktikum 4: Machine Learning (Regression)
12	11.05.2022	NoSQL Systems	Big Data Projekt - Teil 1
13	18.05.2022	Keine Vorlesung (Arbeit am Projekt)	Big Data Projekt - Teil 2
14	25.05.2022	Keine Vorlesung (Arbeit am Projekt)	Big Data Projekt - Teil 3 (Abgabe: 30.5.2022 23:59:59)

Educational Objectives for Today

- Key-Value Store
- Document Store
- Column-Family Store
- GraphDB

Benefits of Relational Databases

- Very powerful for structured data where schema is stable
- Query optimization with indexes
- Widely used in industry
- Typically designed for transaction processing

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Physical Storage of Relational Databases: Row Store

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

Row 1	India
	Chocolate
	1000
Row 2	India
	Ice-cream
	2000
Row 3	Germany
	Chocolate
	4000
Row 4	US
	Noodle
	500

Optimized for transaction processing:

- insert, update, delete

How well is it suited for analytical processing?

Physical Storage of Relational Databases: Column Store

Table

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500

Row Store

	India
Row 1	Chocolate
	1000
	India
Row 2	Ice-cream
	2000
	Germany
Row 3	Chocolate
	4000
	US
Row 4	Noodle
	500

Column Store

	India
Country	India
	Germany
	US
	Chocolate
Product	Ice-cream
	Chocolate
	Noodle
	1000
Sales	2000
	4000
	500

Optimized for analytical processing:

- SELECT Product
  FROM X
  WHERE Sales > 300

What is Wrong with Relational Databases?

Limits of Relational Databases

- Cannot easily scale to “really big” data
- Cannot just plug in new server and achieve scalable processing
- Schema updates are typically very expensive

Wish List for NoSQL Storage System

- Flexible, semi-structured data model:
  - Load first, model afterwards

- Support for different data types:
  - Text
  - Temporal data
  - Images
  - Video

- Full query language like SQL

- Efficient parallel query runtime

CAP Theorem

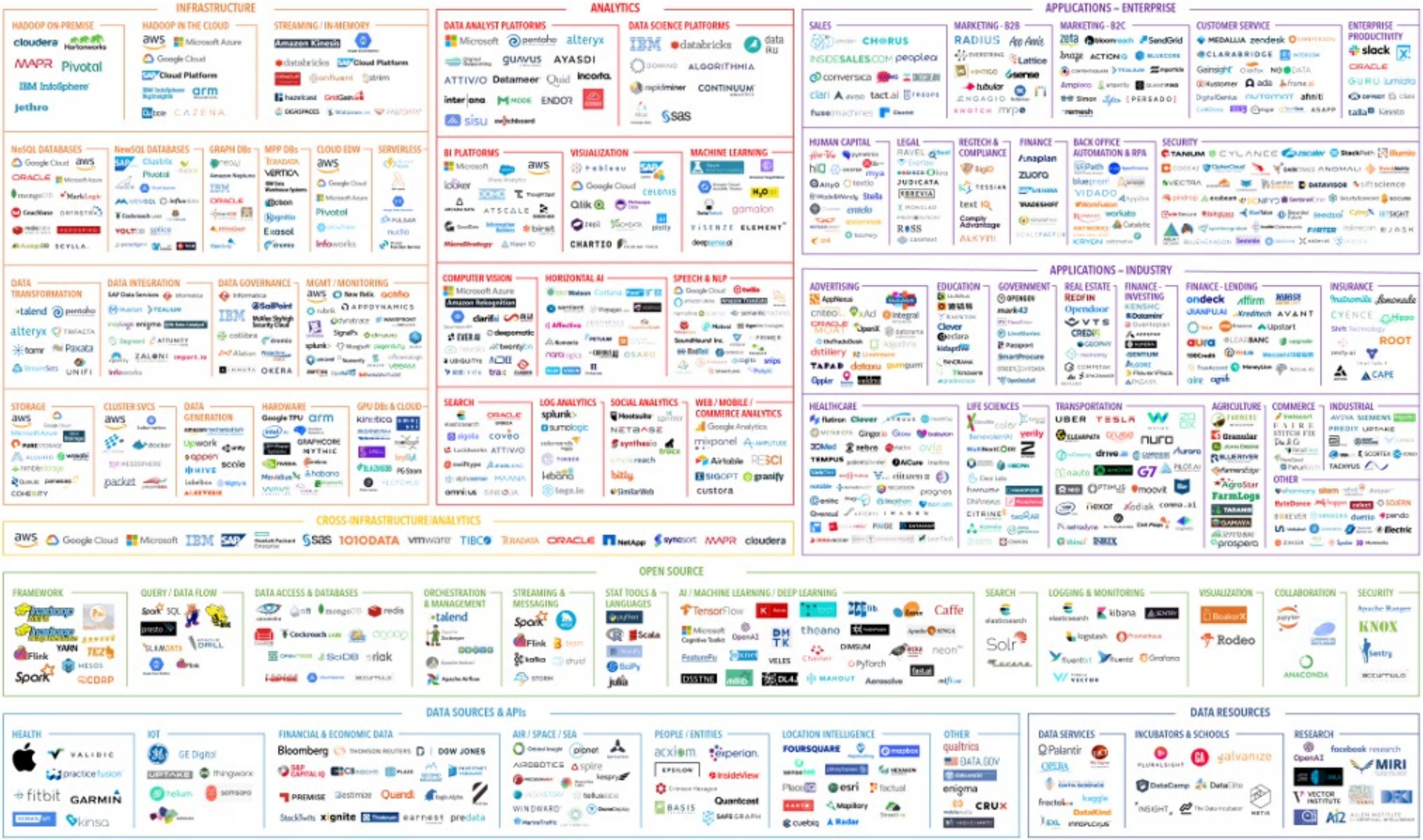
- Hard to achieve ACID properties in a distributed computing environment

- NoSQL systems support CAP properties:

  - Consistency:
    - All nodes see the same data at any time
  - Availability:
    - Every request gets feedback if OK or failed
  - Partition tolerance:
    - System operates even if some partitions fail due to network outages

- NoSQL systems only support two of them

DATA & AI LANDSCAPE 2019



Jun 27, 2019

© Matt Turck (@mattturck), Lisa Xu (@lisaxu92), & FirstMark (@firstmarkcap) mattturck.com/bigdata2019

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

Data & AI Landscape – "Zoom In"

Redis: Key-Value Store

MongoDB: Document Store

Cassandra: Colum-Family Store

Neo4J: Graph DB

Educational Objectives for Today

- Key-Value Store
- Document Store
- Column-Family Store
- GraphDB

Key-Value Stores

- Simplest type of NoSQL database
- Very simple storage concept: key-value-pair (similar to dictionary or hash-table)
  - E.g. <0041 76 245 8005, “Kurt”>
    <0041 78 285 3453, “Luana”>
- Key has to be unique
- Typically keys and values are simple data types
- Typically no indexes are required

Use Case for Key-Value Store

- Manage log files or session information of web applications:
  - Need to look up by phone number or devices

- Manage documents:
  - Need to look up by document name

Examples of Key Value Stores

- Berkeley DB
- MemcacheDB
- Redis
- Riak

Example: Redis - Strings

- In-memory data structure store
- Simple key-value data type:

```
> set mykey somevalue
OK
> get mykey
"somevalue"
```

- Set and get multiple keys:

```
> mset a 10 b 20 c 30
OK
> mget a b c
1) "10"
2) "20"
3) "30"
```

Example: Redis - Hashes

- Stored as key-value pair
- Keys are not directly manipulated but via values (fields)

```
> hmset user:1000 username antirez birthyear 1977 verified 1
OK
> hget user:1000 username
"antirez"
> hget user:1000 birthyear
"1977"
> hgetall user:1000
1) "username"
2) "antirez"
3) "birthyear"
4) "1977"
5) "verified"
6) "1"
```

Architecture for Distributed Data

- Data can be partitioned by key across cluster nodes

- Partitioning enables taking advantage of main memory over n machines:
  - Without partitioning, the main memory of a single computer is the limit

- Different types of partitions:
  - Range partitioning
  - Hash partitioning

Partitioning Strategies: Range Partitioning

- Map the data range to different machines
  - E.g. Users
  - Users with ID 1 to 1,000,000 go to machine 1
  - Users with ID 1,000,001 to 2,000,000 go to machine 2 etc.

- Advantage:
  - Easy to use

- Disadvantage:
  - Requires a table that maps ranges to partitions
  - Table needs to be managed
  - Partitions might be uneven

Partitioning Strategies: Hash Partitioning

- Use a hash function to distribute data to different machines

- Advantage:
  - Even distribution of data

- Disadvantage:
  - Can’t be influenced

Key-Value Stores:
Advantages and Disadvantages

- Advantages:
    - Key-value stores are basically main memory data structures (persistent hash tables)
    - Good for intermediate or permanent storage of simple data structures
    - Fast read and write operations
    - No predefined schema required

- Disadvantages:
    - No complete database system (“one trick pony”)
    - No support for complex queries (a > 5 AND b < 3 OR c > 7)
    - Only limited operations: get, put, …
    - No standardized query language

Educational Objectives for Today

- Key-Value Store
- Document Store
- Column-Family Store
- GraphDB

Document Stores

- Extension of key-value store
- Value can be complex data structure – a document
- Documents are independent of each other
- Also values (attributes) can be indexed – not only keys:
    - Queries on keys and values
- Can be considered as full-fledged database systems

Use Case for Document Store

- Managing unstructured data or data with heterogonous schema:
  - Manage documents
  - Manage social media information, e.g. chat sessions, tweets, blogs
  - New information can be added easily without requiring a schema change
  - IoT (Internet of Things) and device sensor data

Example: MongoDB - Sample Document

- Data is stored as JSON-object
- Can be of complex structure

```
db.users.insertMany(
  [
    {
      _id: 1,
      name: "sue",
      age: 19,
      type: 1,
      status: "P",
      favorites: { artist: "Picasso", food: "pizza" },
      finished: [ 17, 3 ],
      badges: [ "blue", "black" ],
      points: [
         { points: 85, bonus: 20 },
         { points: 85, bonus: 10 }
      ]
    },
    {
      _id: 2,
      name: "bob",
      age: 42,
      type: 1,
      status: "A",
      favorites: { artist: "Miro", food: "meringue" },
      finished: [ 11, 25 ],
      badges: [ "green" ],
      points: [
         { points: 85, bonus: 20 },
         { points: 64, bonus: 12 }
      ]
    },
```

Example: MongoDB - Query

```
db.users.insertMany(
  [
     {
       _id: 1,
       name: "sue",
       age: 19,
       type: 1,
       status: "P",
       favorites: { artist: "Picasso", food: "pizza" },
       finished: [ 17, 3 ],
       badges: [ "blue", "black" ],
       points: [
          { points: 85, bonus: 20 },
          { points: 85, bonus: 10 }
       ]
     },
     {
       _id: 2,
       name: "bob",
       age: 42,
       type: 1,
       status: "A",
       favorites: { artist: "Miro", food: "meringue" },
       finished: [ 11, 25 ],
       badges: [ "green" ],
       points: [
          { points: 85, bonus: 20 },
          { points: 64, bonus: 12 }
       ]
     },
```

- Find users with status “A”

```
db.users.find( { status: "A" } )
```

- Find users with status “P” OR “D”

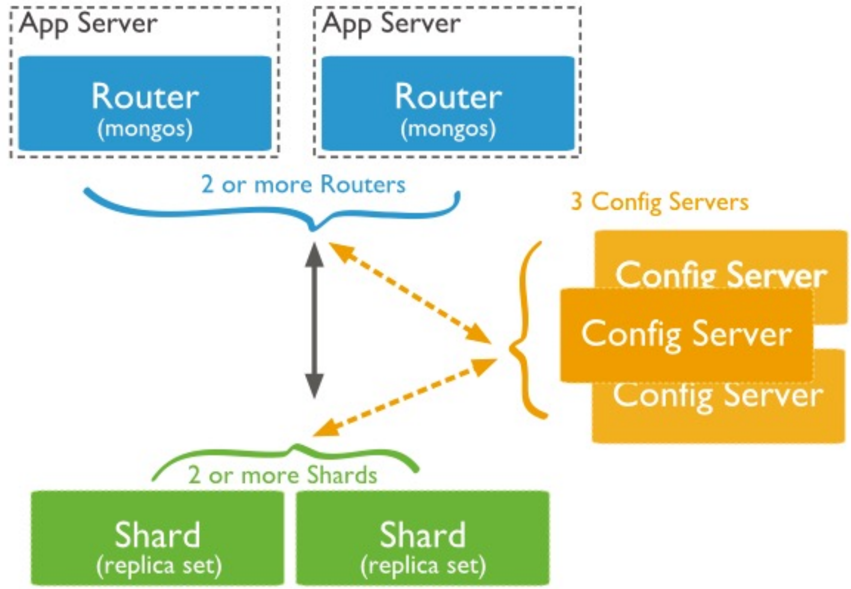
```
db.users.find( { status: { $in: [ "P", "D" ] } } )
```

- Find users with status “A” AND age < 30

```
db.users.find( { status: "A", age: { $lt: 30 } } )
```

Architecture for Distributed Data

- Data sharding: store data across cluster machines
- Solves the problem of horizontal scaling
- Primary-secondary replication
- MongoDB architecture:
  - Shards:
    - Data storage
    - Provide high availability and consistency
  - Config server:
    - Store cluster's metadata
    - Mapping of data to shards
    - Query routing to shards
  - Query router:
    - Send query to shards
    - Receive results



Document Stores: Advantages and Disadvantages

- Advantages:
  - Database for objects (documents) with different schemas
    - E.g. Storing of address with different numbers of attributes

- Disadvantages:
  - Not efficient for databases with complex relationships

Educational Objectives for Today

- Key-Value Store
- Document Store
- Column-Family Store
- GraphDB

Column-Family Stores

- Column family (wide column stores):
  - Stored as multi-dimensional maps
  - Maps consist of key-value pairs

- Similar to traditional relational databases:
  - Data is stored row-wise
  - Table corresponds to a collection of columns

- Also similar to document stores:
  - Key-value pairs where values can be complex data types

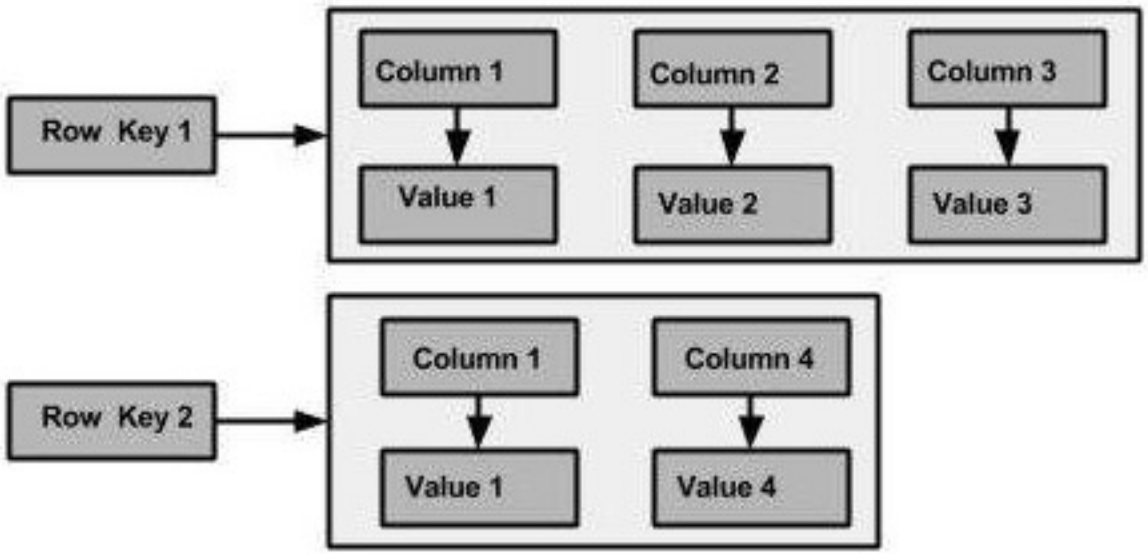
Use Case for Column-Family Store

- Content Management:
  - Store document information with heterogeneous or no schema
  - Document name = key
  - Document information = column family

Examples of Column-Family Stores

- Apache HBase
- Cassandra
- Amazon SimpleDB
- Hypertable

Example: Cassandra – Column Family



Example: Cassandra – CREATE TABLE

```
CREATE TABLE example (
    key1 text PRIMARY KEY,
    map1 map<text,text>,
    list1 list<text>,
    set1 set<text>
    );
```

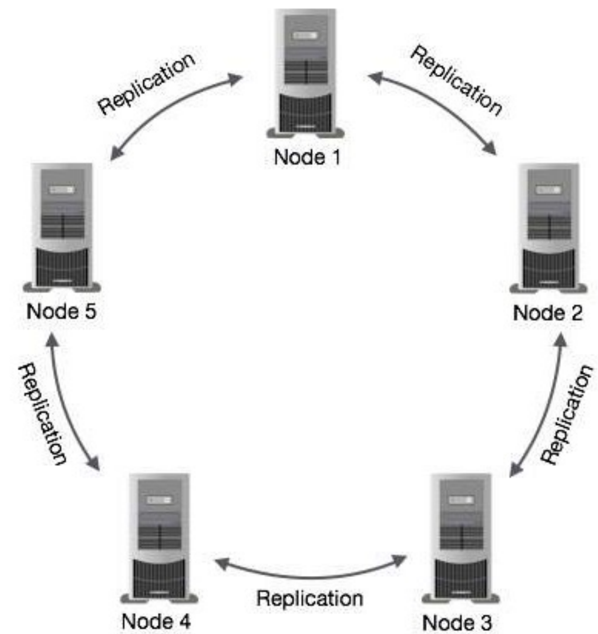
```
INSERT INTO example (
    key1,
    map1,
    list1,
    set1
    ) VALUES (
    'john',
    {'patricia':'555-4326','doug':'555-1579'},
    ['doug','scott'],
    {'patricia','scott'}
    )
```

Architecture for Distributed Data

- Handles big data workloads across multiple cluster nodes without single point of failure

- Peer-to-peer replication, i.e. no master node

- Data is distributed across all cluster nodes:
  - All nodes play the same role
  - Each node is independent
  - Each node can accept read and write requests (local and remote)
  - When a node goes down, read/write requests can be served from other nodes in the network



Column-Family Stores: Advantages and Disadvantages

- Advantages:
  - Flexible schema definition
  - Fast read and write operations
  - Better scalability over multiple machines than relational database
- Disadvantages:
  - No joins
  - Queries only by primary key
  - Not efficient for aggregations over columns, i.e. they need to be implemented by the client application

Educational Objectives for Today

- Key-Value Store
- Document Store
- Column-Family Store
- GraphDB

GraphDB

- Relational databases are not efficient for queries over complex relationships
  - E.g. friends of friends of friends
  - Requires 3-way join
- GraphDBs designed for querying complex networks
- Many graph operations supported:
  - Shortest paths between nodes X and Z
  - In-degree
  - Out-degree

Friends of Friends Problem: Bob's Friends?

Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Friends of Friends Problem: Bob’s Friends

Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

Easy

```sql
SELECT p1.Person
FROM Person p1 JOIN PersonFriend
  ON PersonFriend.FriendID = p1.ID
JOIN Person p2
  ON PersonFriend.PersonID = p2.ID
WHERE p2.Person = 'Bob'
```

Zurich University
of Applied Sciences

Friends of Friends Problem:
Alice’s Friends of Friends?

Person

ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend

PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

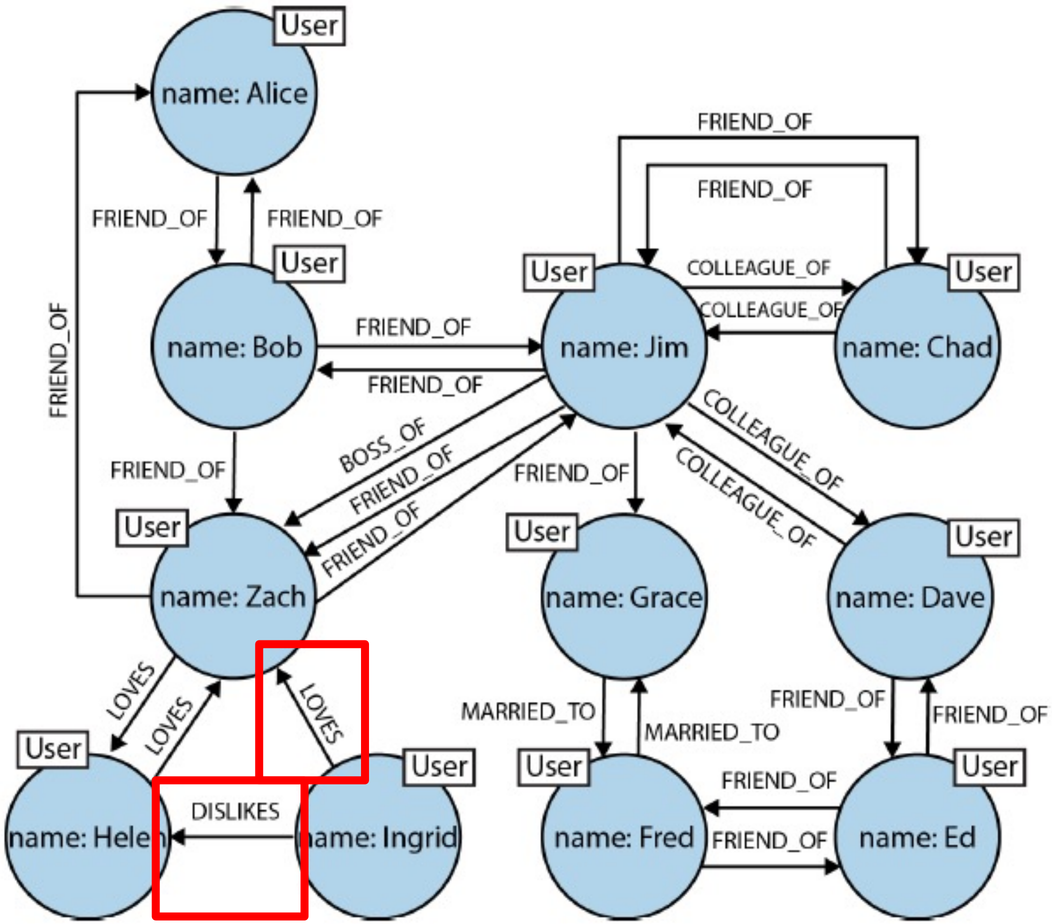
Friends of Friends Problem: Alice's Friends of Friends?

Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

```sql
SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
  ON pf1.PersonID = p1.ID
JOIN PersonFriend pf2
  ON pf2.PersonID = pf1.FriendID
JOIN Person p2
  ON pf2.FriendID = p2.ID
WHERE p1.Person = 'Alice' AND pf2.FriendID <> p1.ID
```

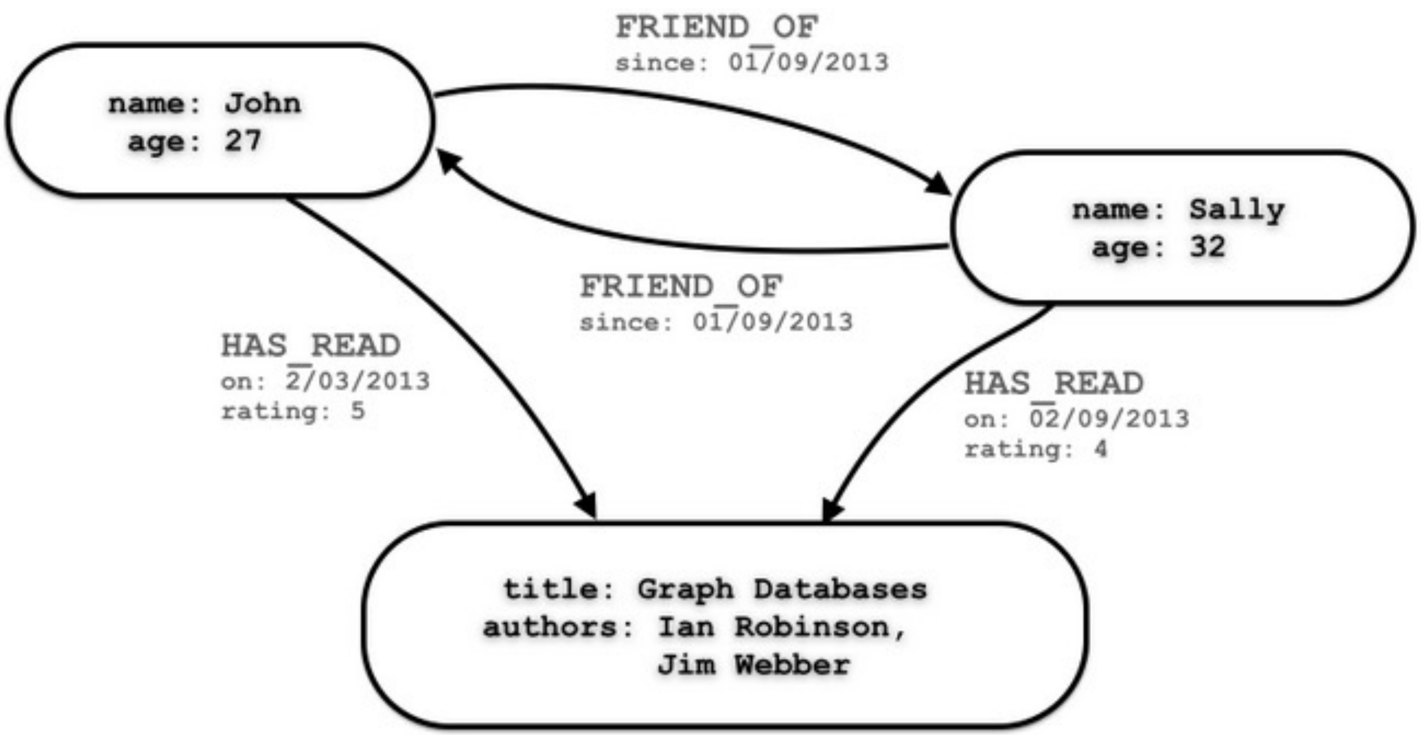
How to Model Friends with a GraphDB?



Examples of GraphDBs

- AllegroGraph
- GraphX (Apache Spark)
- Apache Giraph
- Neo4J

Example: Neo4J



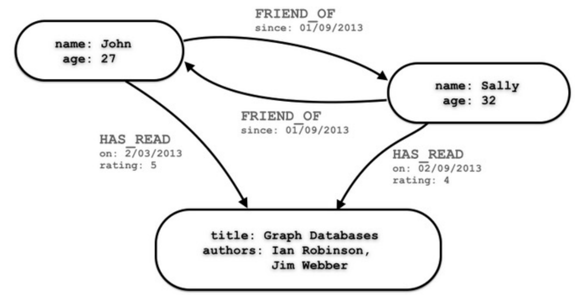
Example: Neo4J

Nodes

```
// Create Sally
CREATE (sally:Person { name: 'Sally', age: 32 })

// Create John
CREATE (john:Person { name: 'John', age: 27 })

// Create Graph Databases book
CREATE (gdb:Book { title: 'Graph Databases',
                   authors: ['Ian Robinson', 'Jim Webber'] })
```

Edges

```
// Connect Sally and John as friends
CREATE (sally)-[:FRIEND_OF { since: 1357718400 }]->(john)

// Connect Sally to Graph Databases book
CREATE (sally)-[:HAS_READ { rating: 4, on: 1360396800 }]->(gdb)

// Connect John to Graph Databases book
CREATE (john)-[:HAS_READ { rating: 5, on: 1359878400 }]->(gdb)
```

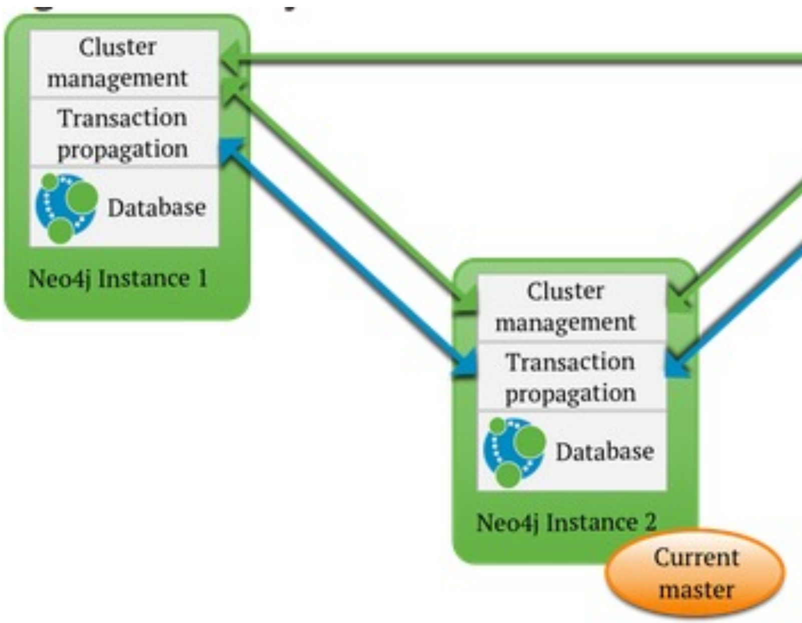


When Did John and Sally become Friends?

```
MATCH (sally:Person { name: 'Sally' })
MATCH (john:Person { name: 'John' })
MATCH (sally)-[r:FRIEND_OF]-(john)
RETURN r.since AS friends_since
```

Architecture for Distributed Data

- One master and n worker instances
- All instances have full copy of the data (data is replicated)



GraphDB:
Advantages and Disadvantages

- Advantages:
  - Good for queries over complex relationships (“friends of friends of friends”)

- Disadvantages:
  - No standardized query language

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

ZNS - Efficient query processing with ZurichNoSQL

Kurt Stockinger*, Richard Bödi, Jonas Heitz, Thomas Weinmann

Zurich University of Applied Sciences, Switzerland

ARTICLE INFO

Keywords:
NoSQL
Main memory database
Query processing

ABSTRACT

NoSQL data stores have recently gained popularity as an alternative to relational database management systems since they typically do not require a fixed schema and scale well for large data sets. These systems have often been tuned to a number of very specific operations such as writing or reading of large data sets. However, none of these novel systems has been demonstrated to efficiently perform multi-dimensional range queries incorporating many boolean operators, a task which is commonly used in scientific data exploration, data warehousing and business analytics.

In this paper we introduce ZurichNoSQL (ZNS) - a novel NoSQL main memory store that supports efficient processing of multi-dimensional point queries and range queries. The key idea of ZNS is to store the data in a column format (compressed column storage) similar to systems used in high performance computing. Moreover, the ZNS architecture is based on a set of low-level main memory techniques ensuring that CPU caches are being used efficiently. Our experimental results comparing to popular NoSQL stores such as FastBit, MongoDB and Spark SQL demonstrate that ZNS significantly outperforms these systems in most cases.

Source: https://www.sciencedirect.com/science/article/abs/pii/S0169023X17304457?via%3Dihub

Some ZNS Results

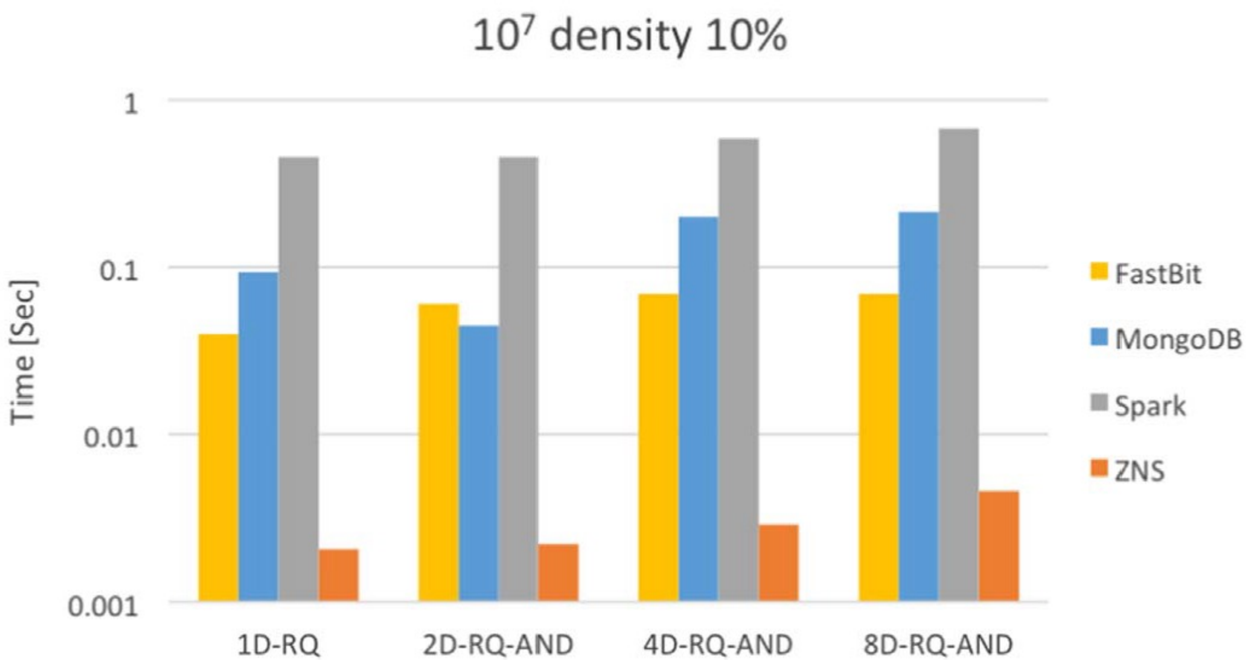


Fig. 12. Response times for range queries with AND-operator. Data set size: 10^7 rows with attribute density of 10%.

Conclusions

- NoSQL stores are special-purpose databases without ACID properties

- Advantages:

  - No ACID
  - Optimized for scalability
  - Use for simple query (key-value lookups)

- Disadvantages:

  - No general purpose database system
  - Not suitable for complex analytical queries

Unterrichtsevaluation

Die Unterrichtsevaluation findet vom 02.05. – 14.05.2022 statt.

Bitte noch ausfüllen, falls noch nicht gemacht.