NoSQL

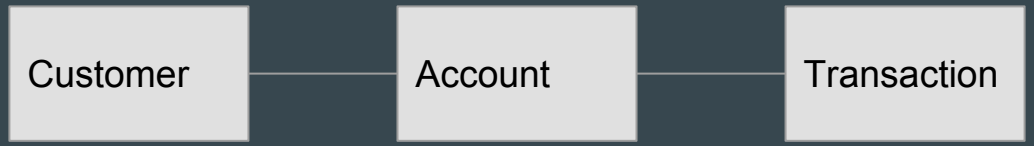
Ronald Peterson ~ Quintor

Agenda

1. Agenda
2. The bank case
3. NoSQL databases
4. The bank case revisited
5. Ronald's database wis'dom's
6. Conclusion

The bank case

The 'bank' case




```
SELECT
  SUM(t.Amount)
FROM
  Transaction t
  INNER JOIN Account a ON (a.Id = t.AccountId)
  INNER JOIN Customer c ON (c.Id = a.CustomerId)
WHERE
  c.Id = "13435432"
```

Assumptions

- 6.000.000 customers
- 2 accounts per customer
- 400 transaction each year per account
- 5 year history

24.000.000.000 rows

NoSQL databases

NoSQL databases

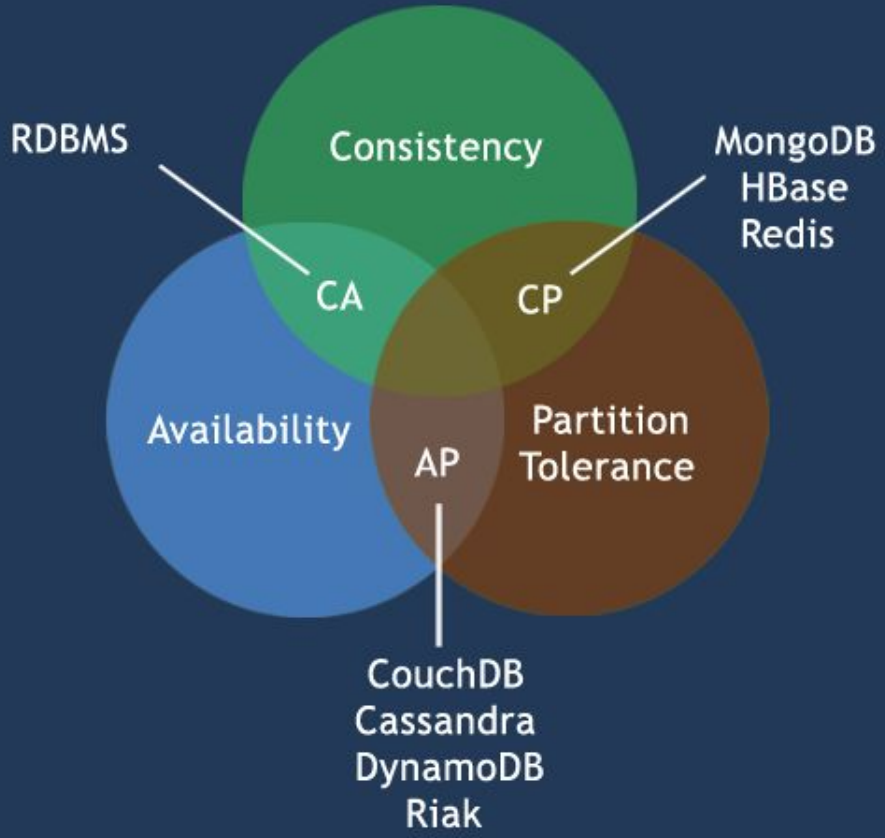
- Not only SQL
- Relaxes on some of the relational properties:
  - Denormalization is the 'norm'
  - Conforming to all the ACID properties not (always) necessary
- Designed to run on many servers

BASE

- Basic Availability ~ The database appears to work most of the time
- Soft-state ~ The state of the system may change over time
- Eventual consistency ~ Stores exhibit consistency at some later point

- Flexible trade-off between consistency and latency, for example using the write/read quorum

CAP Theorem


RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
Partition
Tolerance
AP
CouchDB
Cassandra
DynamoDB
Riak

Kinds of NoSQL databases

- Key-Value pair
- Document
- Column family
- Graph

Key-Value pair

- Essential features: Simplicity, Speed & Scalability
- Determine a unique key, for example Customer:13435432:firstname
- Use hashes to determine server
- No to limited (range) searching
- No Standard query language

Redis/Amazon DynamoDB

Document

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber": [
        { "type": "home", "number": "212 555-1234" },
        { "type": "fax", "number": "646 555-4567" }
    ]
}
```

Document

- XML/JSON documents
- Embedded documents
- Use the application queries as a guideline for the design of the documents
- Transactions on a document are atomic
- Range searches supported, the database interprets the xml/json

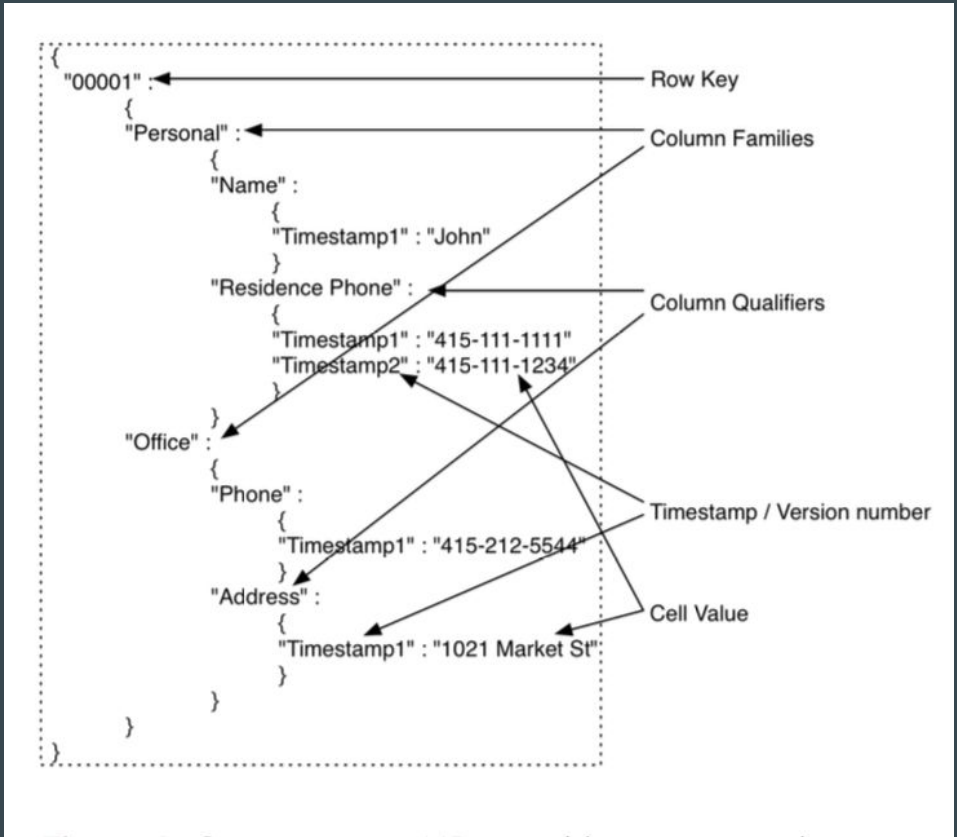
MongoDB query: Db.transaction.find({amount : { "$gte" : 0, "$lt" : 100} })

Document

- Schemaless/Polymorphic schema
  - More flexibility
  - More responsibility for the application
- Related data is stored together in a document:
  - Horizontal partitioning/sharding data is easier
  - Important to determine the right sharding key
  - Developer/Application responsible for avoiding data anomalies

MongoDB/CouchDB

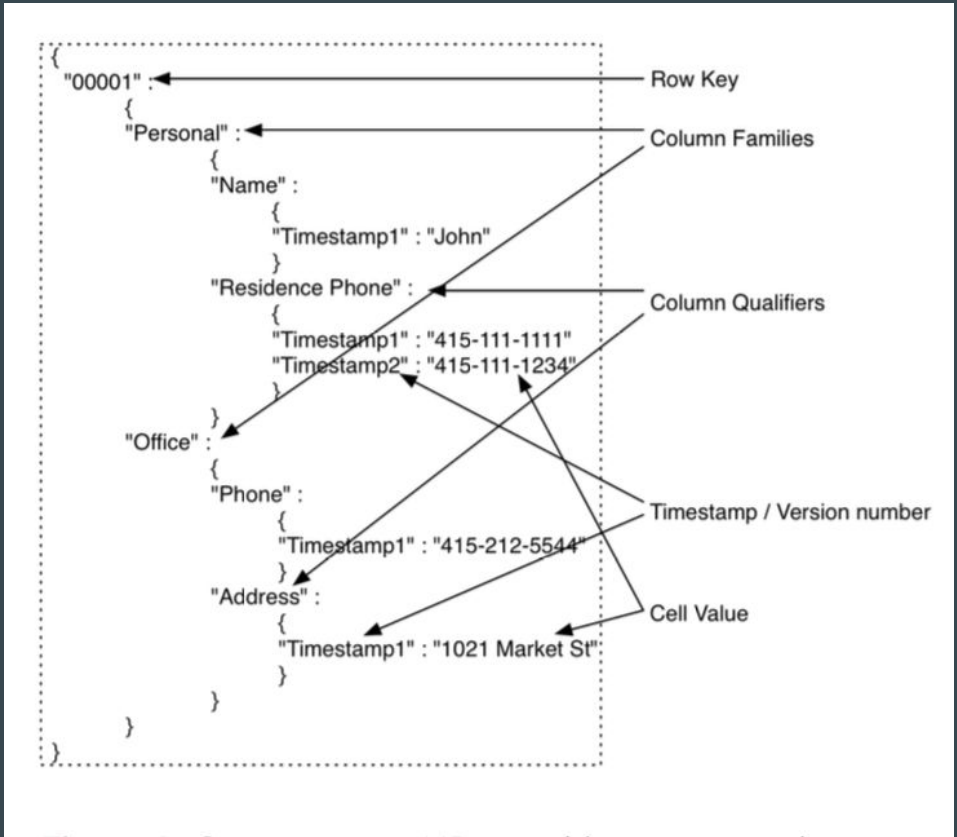
Column family



Column family

- Row key determines the server
- Column family determines the file on the disk
- Data in a column family is stored on disk ordered by row

Column family



Column family

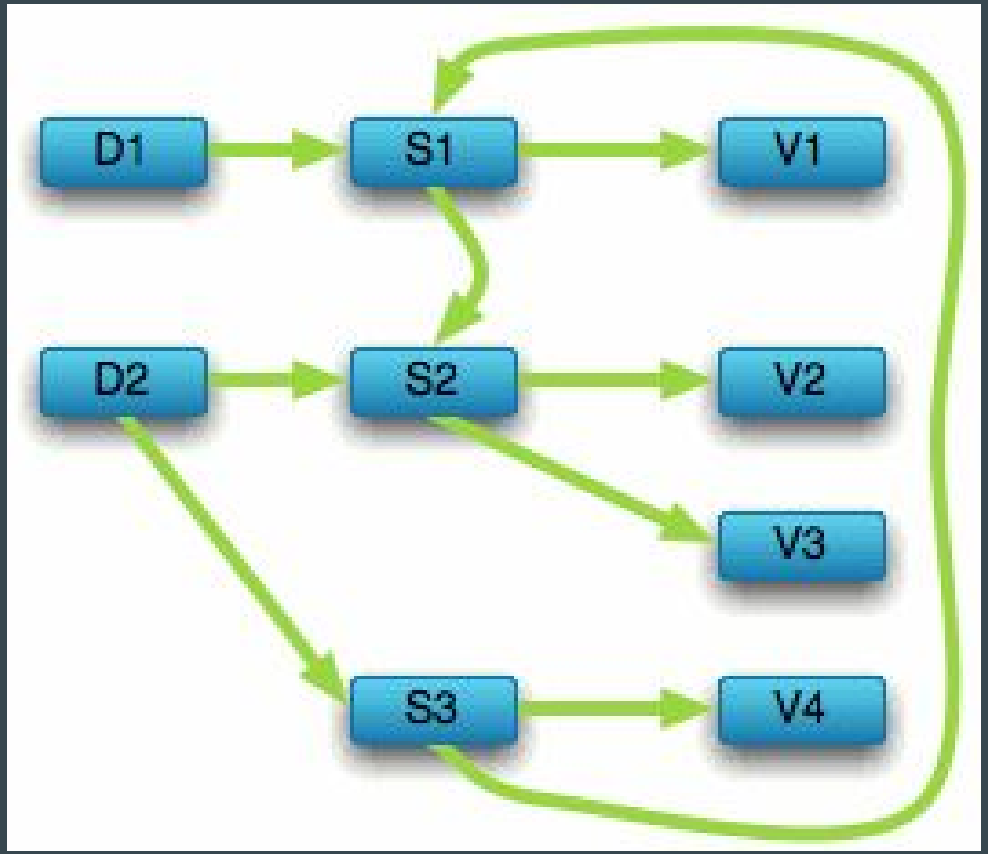
- A column belongs to a column family
- A column name can contain 'data'
- Columns can vary from row to row
- Columns are grouped in column families
- When a value is stored usually also the timestamp of the insert/update is stored

Column family

- Generally read-writes are atomic on the row level
- Peer to peer replication is used
- To reduce the number of internal messages a gossip protocol is used
- Anti-entropy algorithm is used to find and eliminate inconsistencies

- Cassandra/Hadoop/Google BigTable

Graph



Graph

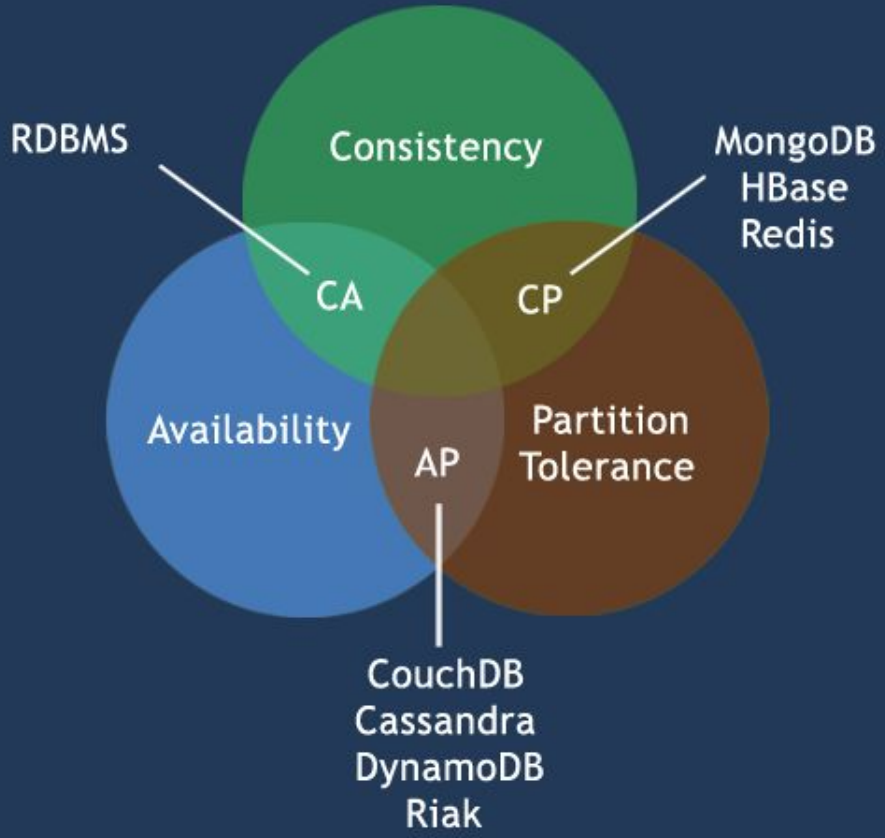
- Not unusual to follow the ACID rules
- Suitable for 'specific' problems:
  - Routing
  - User click path
  - Social links

Neo4J

The bank case revisited

Choose the database(s) to use for the 'Bank'

CAP Theorem


RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
Partition
Tolerance
AP
CouchDB
Cassandra
DynamoDB
Riak

Ronald's database wis'dom's

A million rows is not a lot

Start with writing the queries

Inspect the query plans

Use indexes

But… an index for the indexes shouldn't be needed

Do not datamine the application database

Conclusion

Conclusion

- RDBMS
  - Complex joins
  - Complex multi entity transactions
  - ACID
  - 'Small' datasets (several million rows is considered small)
- NoSQL
  - Horizontal scaling
  - Schema flexibility
  - High performance
  - Indexes of the RDBMS do not fit in memory
- Both
  - If the needs depend on the data use both databases!

Questions?