Non-relational databases for data management

DANIELE APILETTI

POLITECNICO DI TORINO

Technologies for Data management

- Data Warehouses
- NoSQL databases
- MapReduce
  - and other models for distributed programming
- Distributed file systems (GFS, HDFS, etc.)
- Grid computing, cloud computing, HPC computing
- Large-scale machine learning, Big Data

Relational Database Management Systems

- RDBMS are predominant database technologies
  - first defined in 1970 by Edgar Codd of IBM's Research Lab
- Data modeled as relations (tables)
  - object = tuple of attribute values
    - each attribute has a certain domain
  - a table is a set of objects (tuples, rows) of the same type
    - relation is a subset of cartesian product of the attribute domains
  - each tuple identified by a primary key
    - field (or a set of fields) that uniquely identifies a row
  - tables and objects "interconnected" via foreign keys
- SQL query language

RDBMS Example

Students

ID#	Name	Phone	DOB
500	Matt	555-4141	06/03/70
501	Jenny	867-5309	3/15/81
502	Sean	876-9123	10/31/82

ID#	ClassID	Sem
500	1001	Fall02
501	1002	Fall02
501	1002	Spr03
502	1003	S203

Takes_Course

ClassID	Title	ClassNum
1001	Intro to Informatics	I101
1002	Data Mining	I400
1003	Internet and Society	I400

Courses

SELECT Name
FROM Students S, Takes_Course T
WHERE S.ID=T.ID AND ClassID = 1001

source: https://github.com/talhafazal/DataBase/wiki/Home-Work-%23-3-Relational-Data-vs-Non-Relational-Databases

Fundamentals of RDBMS

Relational Database Management Systems (RDMBS)

1. Data structures are broken into the smallest units
   - normalization of database schema
     - because the data structure is known in advance
     - and users/applications query the data in different ways
   - database schema is rigid
2. Queries merge the data from different tables
3. Write operations are simple, search can be slower
4. Strong guarantees for transactional processing

From RDBMS to NoSQL

Efficient implementations of

- **table joins and of**
- **transactional processing**

require centralized system.

- NoSQL Databases:
- Database schema tailored for specific application
  - keep together data pieces that are often accessed together
- Write operations might be slower but read is fast
- Weaker consistency guarantees

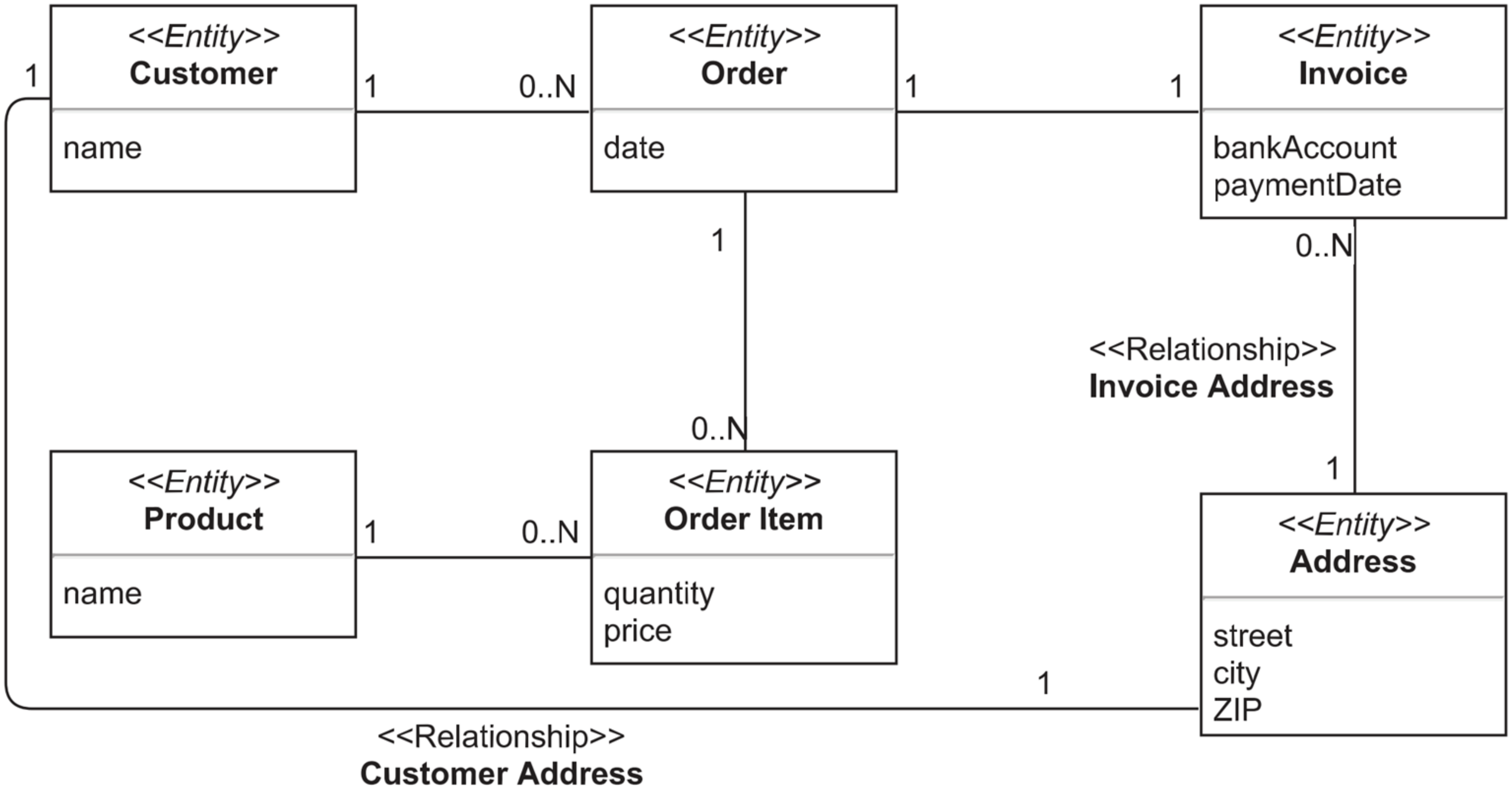
Data model

- The model by which the database **organizes data**

- Each NoSQL DB type has a different data model
  - Key-value, document, column-family, graph
  - The first three are oriented on **aggregates**

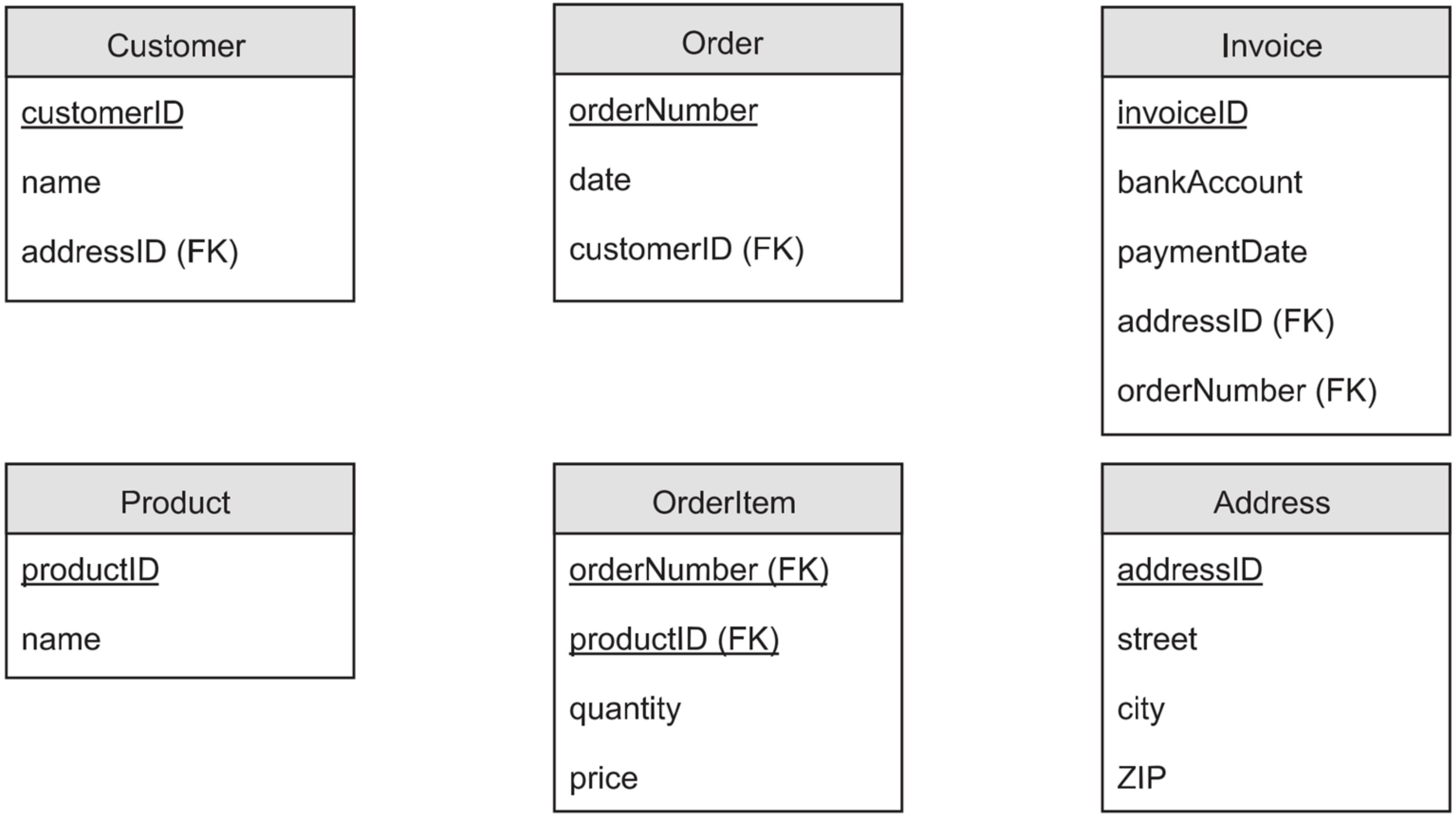
- Let us have a look at the classic relational model

Example: UML Model



source: Holubová, Kosek, Minařík, Novák. Big Data a NoSQL databáze. 2015.

Example: Relational Model



source: Holubová, Kosek, Minařík, Novák. Big Data a NoSQL databáze. 2015.

The Value of Relational Databases

- A (mostly) standard data model
- Many well-developed technologies
    - physical organization of the data, search indexes, query optimization, search operator implementations
- Good concurrency control (ACID)
    - transactions: atomicity, consistency, isolation, durability
- Many reliable integration mechanisms
    - “shared database integration” of applications
- Well-established: familiar, mature, support,…

RDBMS for Data Management

- Relational schema
  - data in tuples
  - a priori known schema
- Schema normalization
  - data split into tables
  - queries merge the data
- Transaction support
  - trans. management with ACID
  - Atomicity, Consistency, Isolation, Durability
  - safety first

- However, real data are
  - naturally flexible and
  - rapidly changing
- You don’t know in advance..

- Inefficient for large data
- Slow in distributed environment

- Full transactions very inefficient in distributed environments

«NoSQL» birth

- In 1998 Carlo Strozzi's lightweight, open-source relational database that did not expose the standard SQL interface
- In 2009 Johan Oskarsson's (Last.fm) organizes an event to discuss recent advances on non-relational databases.
  - A new, unique, short hashtag to promote the event on Twitter was needed: #NoSQL

What is «NoSQL»?

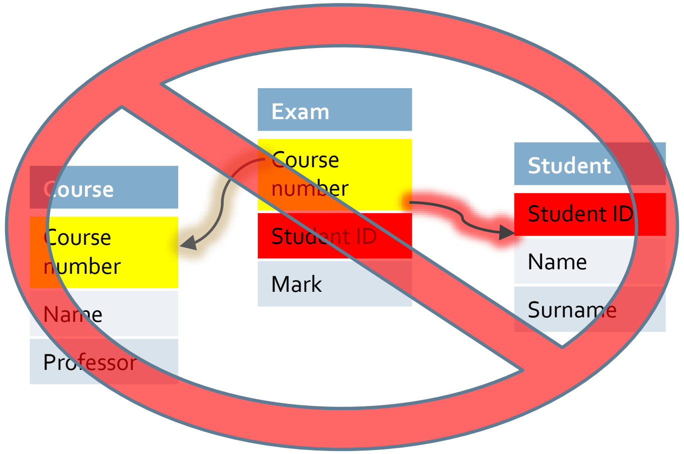
- Term used in late 90s for a different type of technology
  - Carlo Strozzi: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/
- “Not Only SQL”?
  - but many RDBMS are also “not just SQL”

- “NoSQL is an accidental term with no precise definition”
  - first used at an informal meetup in 2009 in San Francisco (presentations from Voldemort, Cassandra, Dynomite, HBase, Hypertable, CouchDB, and MongoDB)

[Sadalage & Fowler: NoSQL Distilled, 2012]

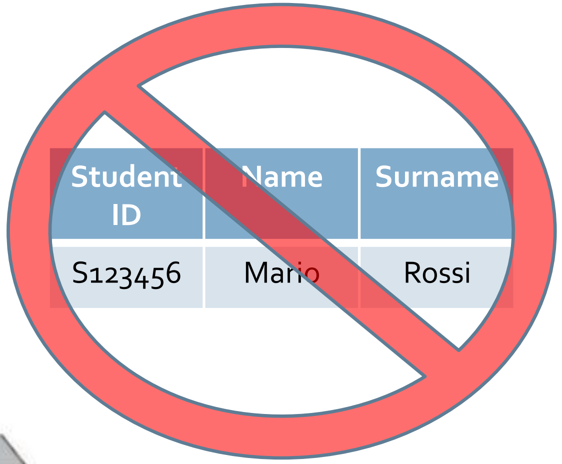
NoSQL main features

no joins

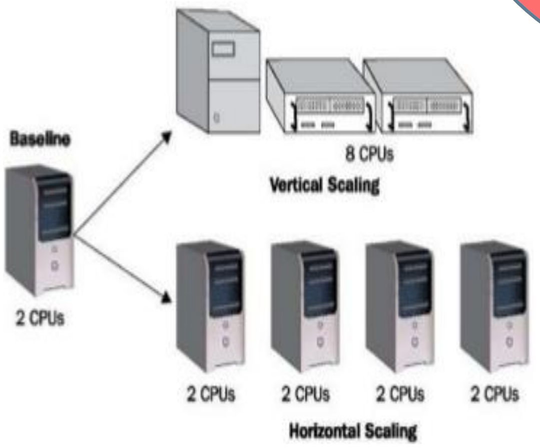


schema-less

(no tables, implicit schema)



horizontal scalability



http://www.slideshare.net/vivekparihar1/mongodb-scalability-and-high-availability-with-replicaset

Comparison

Relational databases	Non-Relational databases
Table-based, each record is a structured row	**Specialized storage solutions**, e.g, document-based, key-value pairs, graph databases, columnar storage
Predefined **schema** for each table, changes allowed but usually blocking (expensive in distributed and live environments)	**Schema-less**, schema-free, schema change is dynamic for each document, suitable for semi-structured or **un-structured data**
Vertically scalable, i.e., typically scaled by increasing the power of the hardware	**Horizontally** scalable, NoSQL databases are scaled by increasing the databases servers in the pool of resources to reduce the load

Comparison

Relational databases	Non-Relational databases
Use **SQL** (Structured Query Language) for defining and manipulating the data, very powerful	**Custom query** languages, focused on collection of documents, graphs, and other specialized data structures
Suitable for **complex queries**, based on data **joins**	**No standard** interfaces to perform complex queries, **no joins**
Suitable for **flat** and structured data storage	Suitable for complex (e.g., **hierarchical**) data, similar to JSON and XML
Examples: MySQL, Oracle, Sqlite, Postgres and Microsoft SQL Server	Examples: MongoDB, BigTable, Redis, Cassandra, HBase and CouchDB

Non-relational/NoSQL DBMSs

- Pros
  - Work with semi-structured data (JSON, XML), typical for web/online applications
  - Scale out (horizontal scaling – parallel query performance, replication)
  - High concurrency, high-volume random reads and writes
  - Massive data stores
  - Schema-free, schema-on-read
  - Support records/documents with different fields
  - High availability by design
  - Speed (join avoidance)

Non-relational/NoSQL DBMSs

- Cons
  - Do not support strict ACID transactional consistency
  - Data is de-normalized
    - requiring mass updates (e.g., product name change)
  - Missing built-in data integrity (do-it-yourself in your code)
  - No relationship enforcement (e.g., foreign keys)
  - Weak SQL
  - Slow mass updates
  - Use more disk space (replicated denormalized records, 10-50x)
  - Difficulty in tracking “schema” (set of attribute) changes over time

Just Another Temporary Trend?

- There have been other trends here before
  - object databases, XML databases, etc.

- But NoSQL databases:
  - are answer to real practical problems big companies have
  - are often developed by the biggest players
  - outside academia but based on solid theoretical results
    - e.g. old results on distributed processing
  - widely used

NoSQL Properties

1. Good scalability
   - horizontal scalability instead of vertical
2. Dynamic schema of data
   - different levels of flexibility for different types of DB
3. Efficient reading
   - spend more time storing the data, but read fast
   - keep relevant information together
4. Cost saving
   - designed to run on commodity hardware
   - typically open-source (with a support from a company)

Challenges of NoSQL Databases

1. Maturity of the technology
   - it's getting better, but RDBMS had a lot of time
2. User support
   - rarely professional support as provided by, e.g. Oracle
3. Administration
   - massive distribution requires advanced administration
4. Standards for data access
   - RDBMS have SQL, but the NoSQL world is more wild
5. Lack of experts
   - not enough DB experts on NoSQL technologies

The End of Relational Databases?

- Relational databases are not going away
  - are ideal for a lot of structured data, reliable, mature, etc.
- RDBMS became one option for data storage

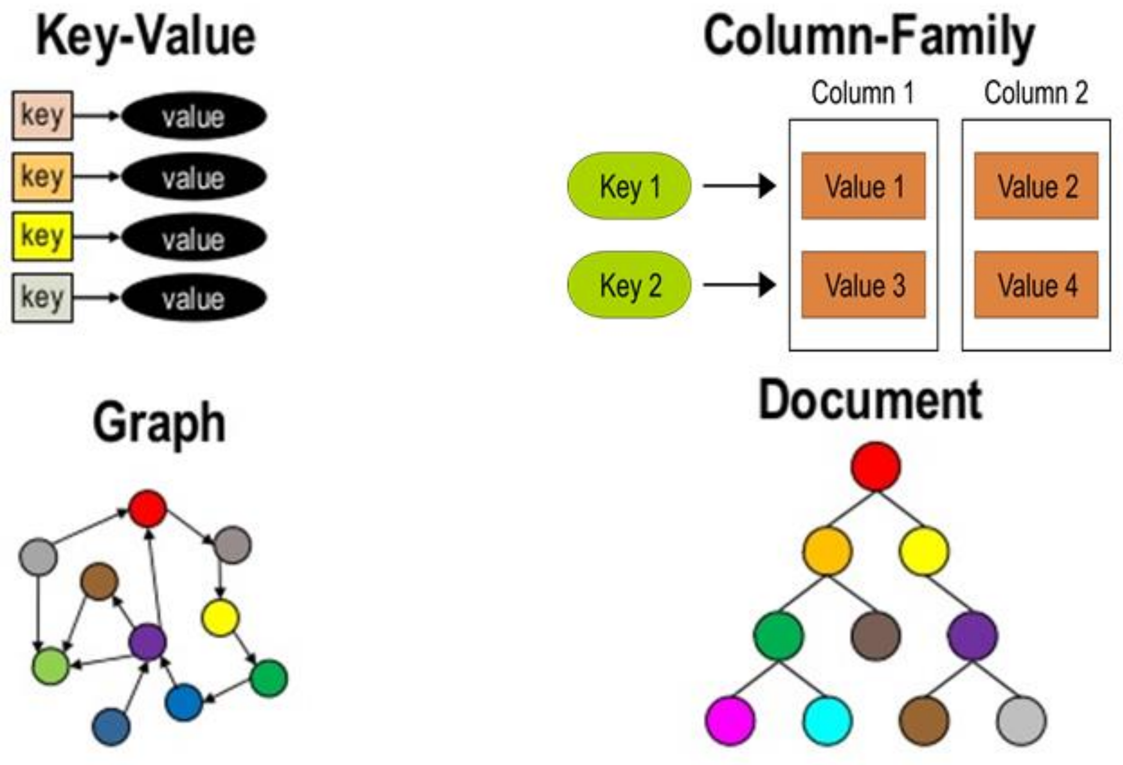
Polyglot persistence – using different data stores in different circumstances

[Sadalage & Fowler: NoSQL Distilled, 2012]

Two trends

1. NoSQL databases implement standard RDBMS features
2. RDBMS are adopting NoSQL principles

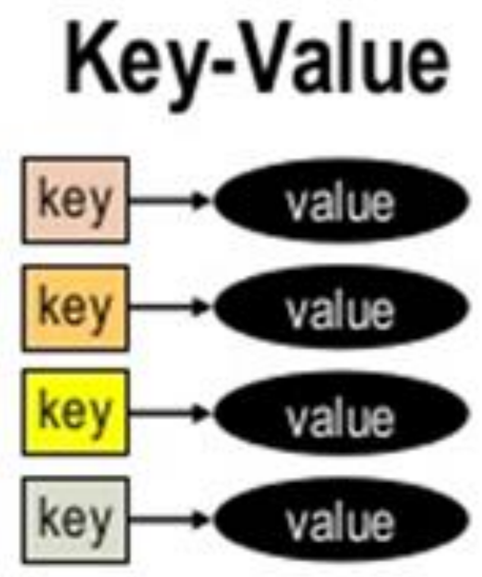
Types of NoSQL databases



http://www.slideshare.net/Couchbase/webinar-making-sense-of-nosql-applying-nonrelational-databases-to-business-needs

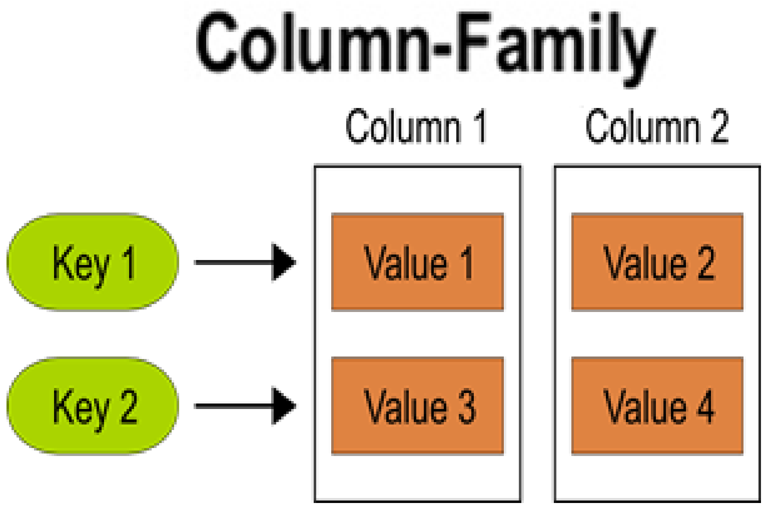
1) Key-values databases

- **Simplest** NoSQL data stores
- Match keys with values
- No structure
- Great **performance**
- Easily scaled
- Very fast
- Examples: Redis, Riak, Memcached



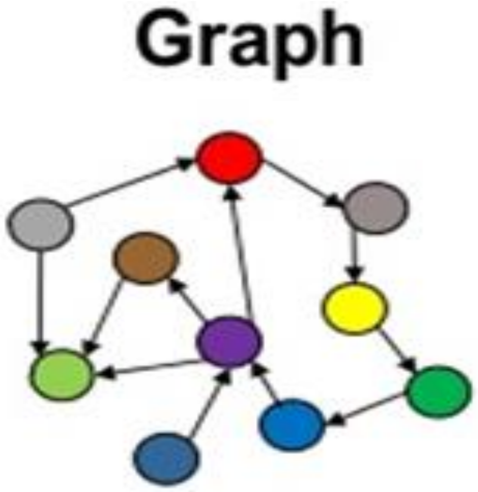
2) Column-oriented databases

- Store data in **columnar** format
  - Name = “Daniele”:row1,row3; “Marco”:row2,row4; ...
  - Surname = “Apiletti”:row1,row5; “Rossi”:row2,row6,row7...
- A column is a (possibly-complex) **attribute**
- Key-value pairs stored and retrieved on key in a paralle system (similar to **indexes**)
- **Rows** can be constructed from column values
- Column stores can produce row output (**tables**)
- Completely transparent to application
- Examples: Cassandra, Hbase, Hypertable, Amazon DynamoDB



3) Graph databases

- Based on graph theory
- Made up by **Vertices** and **unordered** Edges or ordered **Arcs** between each Vertex pair
- Used to store information about **networks**
- Good fit for several real world applications
- Examples: Neo4J, Infinite Graph, OrientDB



4) Document databases

- Database stores and retrieves documents
- Keys are mapped to documents
- Documents are self-describing

  (attribute=value)
- Has hierarchical-tree nested data structures

  (e.g., maps, lists, datetime, …)
- **Heterogeneous** nature of documents
- Examples: **MongoDB**, CouchDB, RavenDB.

Document-based model

- Strongly aggregate-oriented
  - Lots of aggregates
  - Each aggregate has a key
  - Each aggregate is a document
- Data model
  - A **set of <key,value> pairs**
  - Document: an aggregate instance of <key,value> pairs
- Access to an aggregate
  - Queries based on the fields in the aggregate

```
# Customer object
{
"customerId": 1,
"name": "Martin",
"billingAddress": [{"city": "Chicago"}],
"payment": [
  {"type": "debit",
  "ccinfo": "1000-1000-1000-1000"}
  ]
}
```

```
# Order object
{
"orderId": 99,
"customerId": 1,
"orderDate":"Nov-20-2011",
"orderItems":[{"productId":27, "price": 32.45}],
"orderPayment":[{"ccinfo":"1000-1000-1000-1000",
        "txnId":"abelif879rft"}],
"shippingAddress":{"city":"Chicago"}
}
```

Document basics

- Basic concept of data: Document
- Documents are self-describing pieces of data
  - Hierarchical tree data structures
  - Nested associative arrays (maps), collections, scalars
  - XML, JSON (JavaScript Object Notation), BSON, …
- Documents in a collection should be “similar”
  - Their schema can differ
- Documents stored in the value part of key-value
  - Key-value stores where the values are examinable
  - Building search indexes on various keys/fields

Document Example

- key=3 ->
```
{ "personID": 3,
  "firstname": "Martin",
  "likes": [ "Biking","Photography" ],
  "lastcity": "Boston",
  "visited": [ "NYC", "Paris" ] }
```

- key=5 ->
```
{ "personID": 5,
  "firstname": "Pramod",
 "citiesvisited": [ "Chicago", "London","NYC" ],
 "addresses": [
    { "state": "AK",
      "city": "DILLINGHAM" },
    { "state": "MH",
      "city": "PUNE" }  ],
  "lastcity": "Chicago" }
```

source: Sadalage & Fowler: NoSQL Distilled, 2012

Queries on Documents

Example in MongoDB syntax

- Query language expressed via JSON
- clauses: where, sort, count, sum, etc.

SQL: `SELECT * FROM users`

MongoDB: `db.users.find()`

Example 1

```
SELECT *
FROM users
WHERE personID = 3
db.users.find( { "personID": 3 } )
```

Example 2

```
SELECT firstname, lastcity
FROM users
WHERE personID = 5
db.users.find( { "personID": 5},
{firstname:1, lastcity:1} )
```

Document Databases: Representatives

mongoDB

Apache CouchDB relax

OrientDB

RAVENDB

terrastore

PostgreSQL

amazon DynamoDB

MS Azure DocumentDB

Ranked list: http://db-engines.com/en/ranking/document+store

Credits and sources

- P. Atzeni, R. Torlone
  - Dipartimento di Ingegneria, Sezione di Informatica e Automazione, Roma 3
- Martin Svoboda
  - Charles University, Faculty of Mathemacs and Physics Czech Technical
  - Universityin Prague,Faculty of Electrical Engineerin
- David Novak
  - FI, Masaryk University, Brno