Non-relational databases for data management
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

```
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-%233-Relational-Data-vs-Non-Relational-Databases
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
Efficient implementations of
- table joins and of
- transactional processing
require centralized system.

• NoSQL Databases:

• Database schema tailored for specific application
  o 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
  o Key-value, document, column-family, graph
  o 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

Customer
- customerID
- name
- addressID (FK)

Order
- orderNumber
- date
- customerID (FK)

Invoice
- invoiceID
- bankAccount
- paymentDate
- addressID (FK)
- orderNumber (FK)

Product
- productID
- name

OrderItem
- orderNumber (FK)
- productID (FK)
- quantity
- price

Address
- addressID
- street
- city
- ZIP

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
  o data in tuples
  o a priori known schema

• Schema normalization
  o data split into tables
  o queries merge the data

• Transaction support
  o trans. management with ACID
  o Atomicity, Consistency, Isolation, Durability
  o 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.
  
  o 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
  o Carlo Strozzi: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/

• “Not Only SQL”?
  o but many RDBMS are also “not just SQL”

• “NoSQL is an accidental term with no precise definition”
  o 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**
(no tables, implicit schema)

**horizontal scalability**

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

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

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

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

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

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

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

Key-Value

Column-Family

Graph

Document

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
  o Name = “Daniele”:row1,row3; “Marco”:row2,row4; ...
  o 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 parallel 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
  o Lots of aggregates
  o Each aggregate has a key
  o Each aggregate is a document

• Data model
  o A set of <key, value> pairs
  o Document: an aggregate instance of <key, value> pairs

• Access to an aggregate
  o 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": "abe1f879rft"},
                   "shippingAddress": {"city": "Chicago"}
  ]
```
Document basics

• Basic concept of data: Document
• Documents are self-describing pieces of data
  o Hierarchical tree data structures
  o Nested associative arrays (maps), collections, scalars
  o XML, JSON (JavaScript Object Notation), BSON, ...
• Documents in a collection should be “similar”
  o Their schema can differ
• Documents stored in the value part of key-value
  o Key-value stores where the values are examinable
  o 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
Examples 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**

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

**Example 2**

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

Credits and sources

• P. Atzeni, R. Torlone
  o Dipartimento di Ingegneria, Sezione di Informatica e Automazione, Roma 3

• Martin Svoboda
  o Charles University, Faculty of Mathematics and Physics Czech Technical
  o University in Prague, Faculty of Electrical Engineering

• David Novak
  o FI, Masaryk University, Brno