NDBI040

Big Data Management and NoSQL Databases

Lecture 1. Introduction

Doc. RNDr. Irena Holubova, Ph.D.
holubova@ksi.mff.cuni.cz

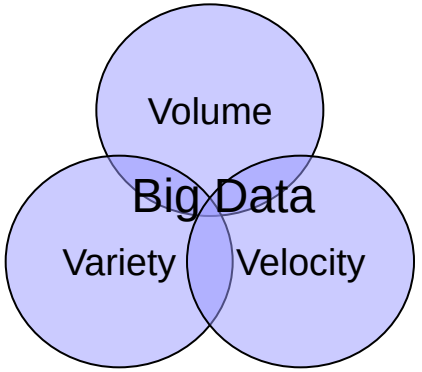
http://www.ksi.mff.cuni.cz/~holubova/NDBI040/

What is Big Data?

- buzzword?
- bubble?
- gold rush?
- revolution?

"Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."
Dan Ariely

What is Big Data?



- No standard definition
- First occurrence of the term: High Performance Computing (HPC)

Gartner: *"**Big Data**" is high **v**olume, high **v**elocity, and/or high **v**ariety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.*

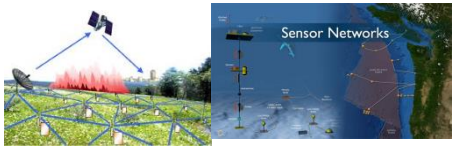
3 (4, 5) Vs

What is Big Data?

Mobile devices
(tracking all objects all the time)

Social media and networks
(all of us are generating data)

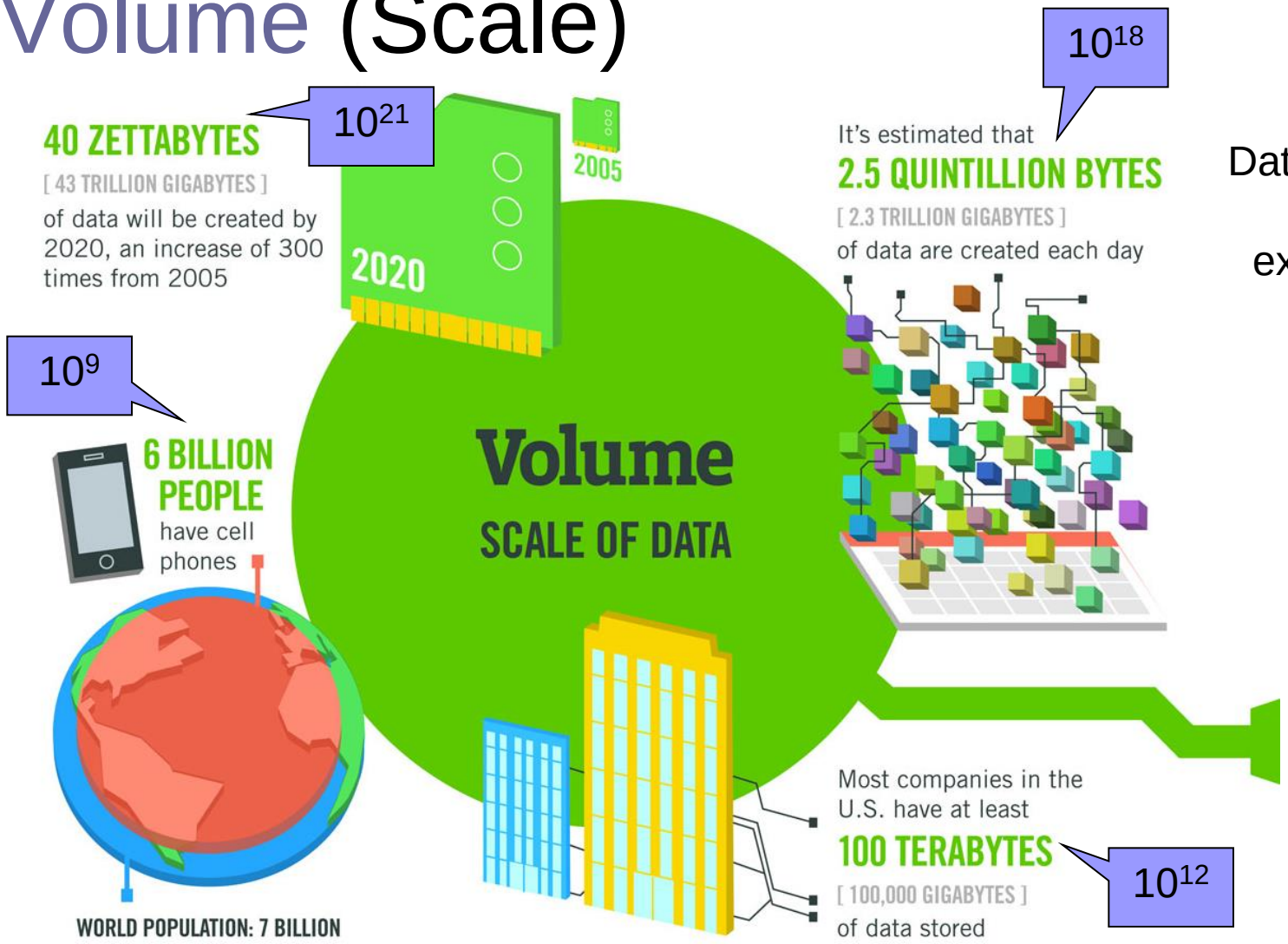
Scientific instruments
(collecting all sorts of data)



Sensor technology and networks
(measuring all kinds of data)

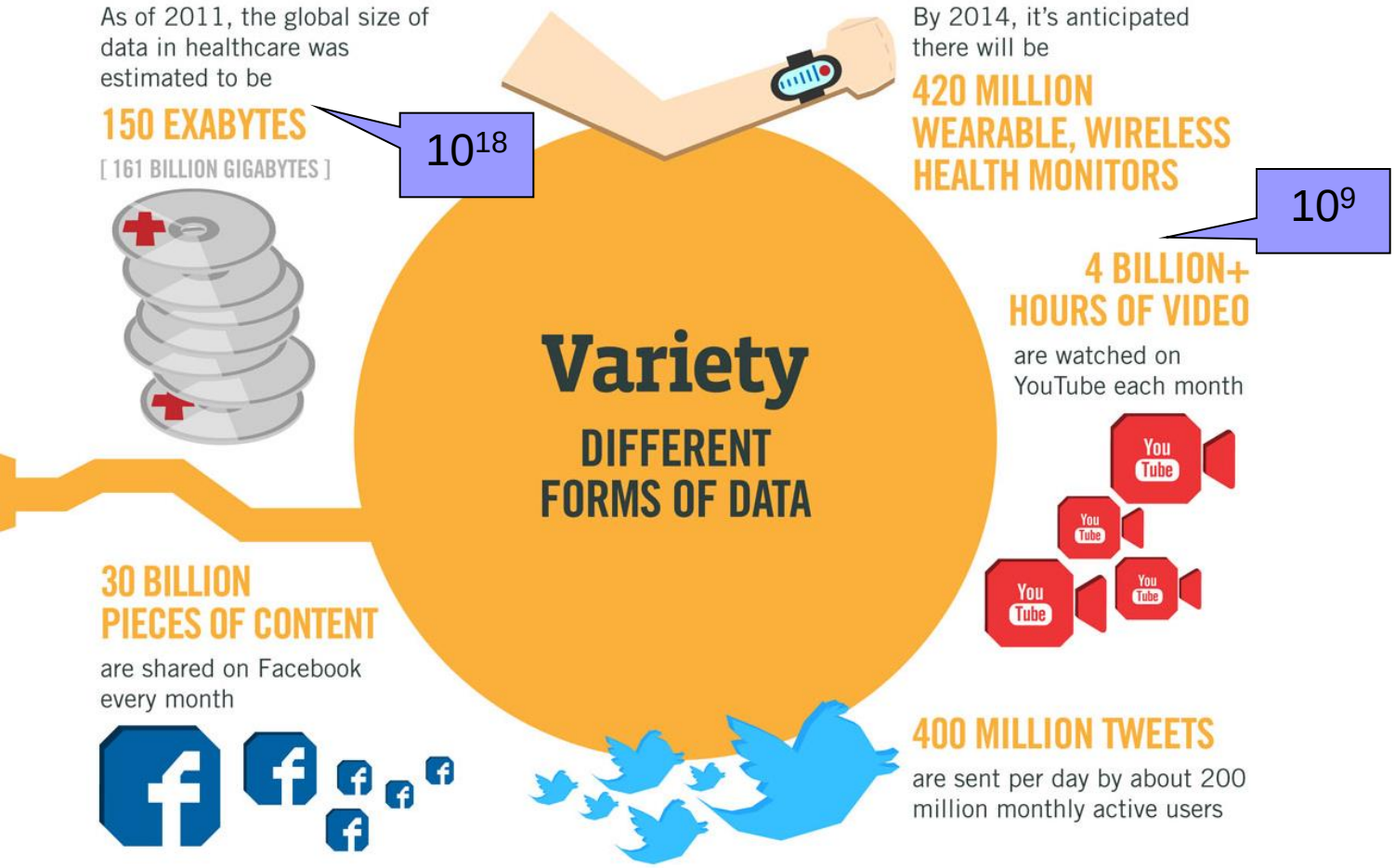
IBM: *Depending on the industry and organization,* ***Big Data*** *encompasses information from internal and external sources such as transactions, social media, enterprise content, sensors, and mobile devices.*
Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.

Big Data Characteristics: Volume (Scale)



Data volume is increasing exponentially, not linearly

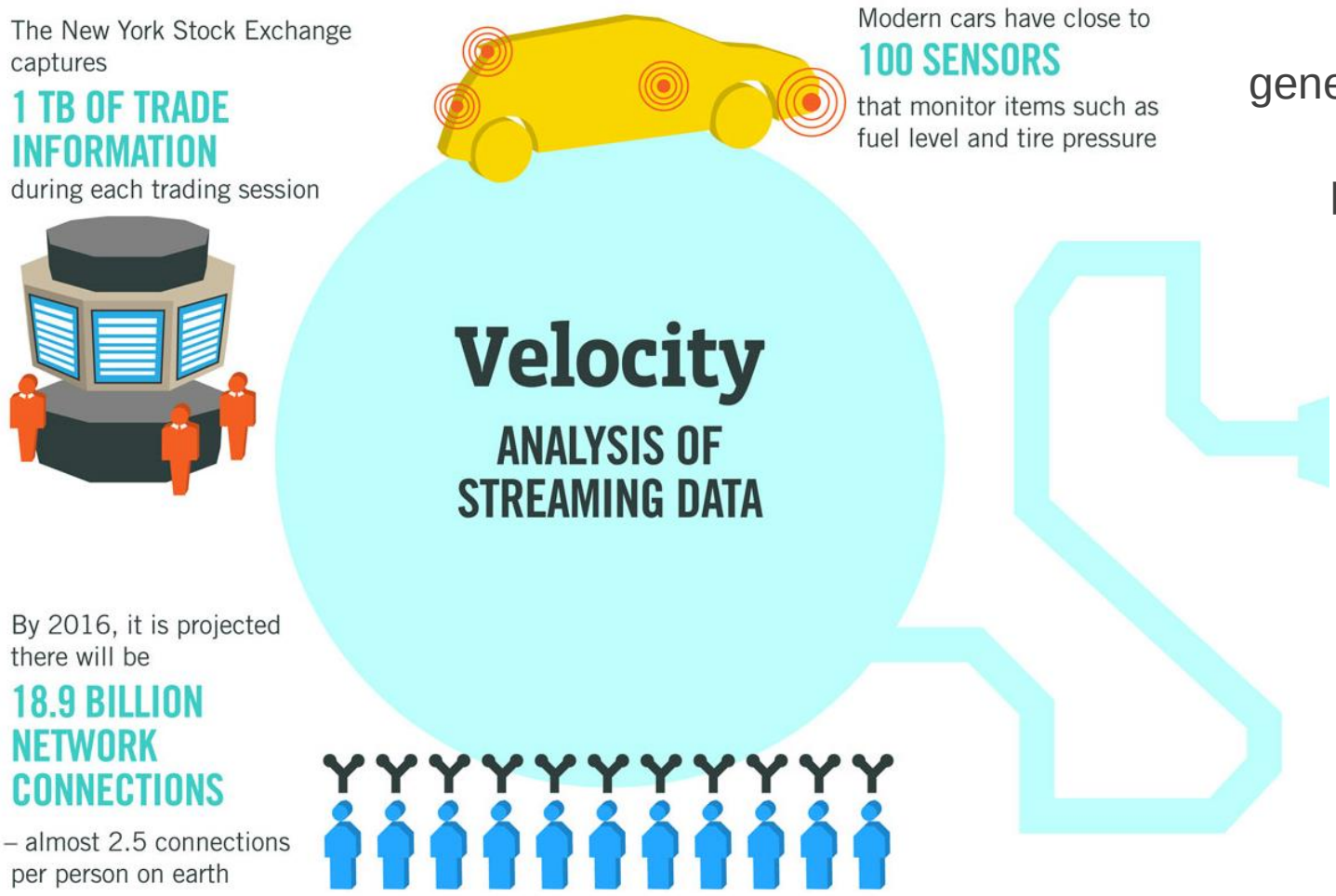
Big Data Characteristics: Variety (Complexity)



Various formats, types, and structures (from semi-structured XML to unstructured multimedia)

Static data vs. streaming data

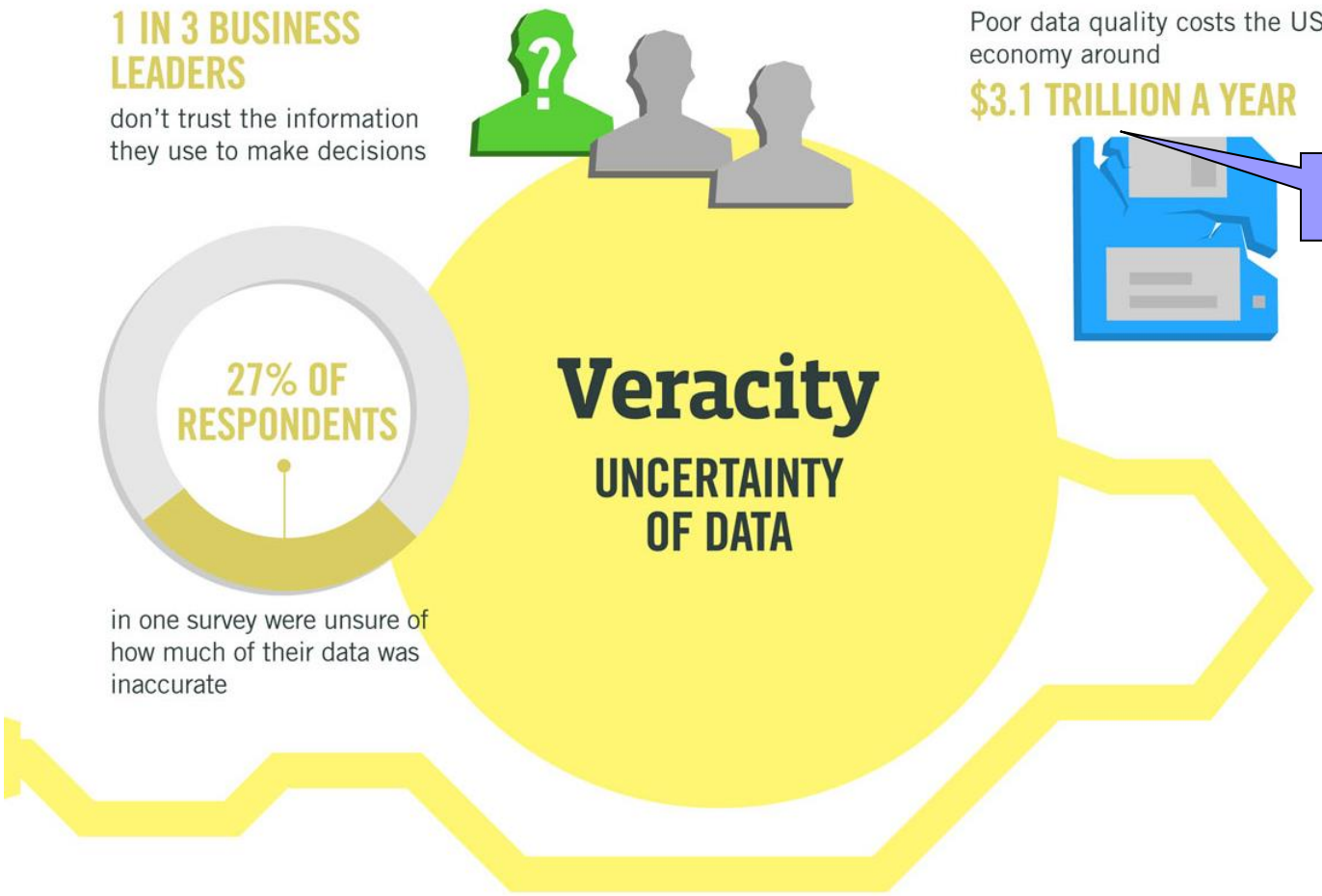
Big Data Characteristics: Velocity (Speed)



Data is being generated fast and need to be processed fast

Online Data Analytics

Big Data Characteristics: Veracity (Uncertainty)



Uncertainty due to inconsistency, incompleteness, latency, ambiguities, or approximations.

Processing Big Data

- OLTP: Online Transaction Processing (DBMSs)
  - Database applications
  - Storing, querying, multiuser access
- OLAP: Online Analytical Processing (Data Warehousing)
  - Answer multi-dimensional analytical queries
  - Financial/marketing reporting, budgeting, forecasting, …
- RTAP: Real-Time Analytic Processing (Big Data Architecture & Technology)
  - Data gathered & processed in a real-time
    - Streaming fashion
  - Real-time data queried and presented in an online fashion
  - Real-time and history data combined and mined interactively

Key Big Data-Related Technologies

- Distributed file systems
- NoSQL databases
- Grid computing, cloud computing
- MapReduce and other new paradigms
- Large scale machine learning

Relational Database Management Systems (RDMBSs)

- Predominant technology for storing structured data
  - Web and business applications
- Relational calculus, SQL
- Often thought of as the only alternative for data storage
  - Persistence, concurrency control, integration mechanism, …
- Alternatives: Object databases or XML stores
  - Never gained the same adoption and market share

„NoSQL“

- 1998 first used for a relational database that omitted the use of SQL
  - Carlo Strozzi
- 2009 used for conferences of advocates of non-relational databases
  - Eric Evans
    - Blogger, developer at Rackspace

NoSQL movement = “the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for”

„NoSQL“

- Not „no to SQL“
  - Another option, not the only one
- Not „not only SQL“
  - Oracle DB or PostgreSQL would fit the definition
- „Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable. The original intention has been modern web-scale databases. Often more characteristics apply as: schema-free, easy replication support, simple API, eventually consistent (BASE, not ACID), a huge data amount, and more“

The End of Relational Databases?

- Relational databases are not going away
- Compelling arguments for most projects
  - Familiarity, stability, feature set, and available support
- We should see relational databases as one option for data storage
  - Polyglot persistence – using different data stores in different circumstances
  - Search for optimal storage for a particular application

Motivation for NoSQL Databases

- Huge amounts of data are now handled in real-time
- Both data and use cases are getting more and more dynamic
- Social networks (relying on graph data) have gained impressive momentum
  - Special type of NoSQL databases: graph databases
- Full-texts have always been treated shabbily by RDBMS

Example: FaceBook

Statistics from 2010

- 500 million users
- 570 billion page views per month
- 3 billion photos uploaded per month
- 1.2 million photos served per second
- 25 billion pieces of content (updates, comments) shared every month
- 50 million server-side operations per second

2008: 10,000 servers
2009: 30,000 servers
…

→ One RDBMS may not be enough to keep this going on!

And even bewer numbers:
https://research.facebook.com/blog/1522692927972019/facebook-s-top-open-data-problems/

Example: FaceBook

Architecture from 2010

Cassandra

- NoSQL distributed storage system with no single point of failure
- For inbox searching

Hadoop/Hive

- An open source MapReduce implementation
- Enables to perform calculations on massive amounts of data
- Hive enables to use SQL queries against Hadoop

Example: FaceBook

Architecture from 2010 and later

Memcached

- Distributed memory caching system
- Caching layer between the web servers and MySQL servers
  - Since database access is relatively slow

HBase

- Hadoop database, used for e-mails, instant messaging and SMS
- Has recently replaced MySQL, Cassandra and few others
- Built on Google's BigTable model

NoSQL Databases

Five Advantages

1. Elastic scaling

- "Classical" database administrators scale up – buy bigger servers as database load increases
- Scaling out – distributing the database across multiple hosts as load increases

2. Big Data

- Volumes of data that are being stored have increased massively
- Opens new dimensions that cannot be handled with RDBMS

http://www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772

NoSQL Databases

Five Advantages

3. Goodbye DBAs (see you later?)

- Automatic repair, distribution, tuning, … vs. expensive, highly trained DBAs of RDBMS

4. Economics

- Based on cheap commodity servers → less costs per transaction/second

5. Flexible Data Models

- Non-existing/relaxed data schema → structural changes cause no overhead

NoSQL Databases

Five Challenges

1. Maturity

- Still in pre-production phase
- Key features yet to be implemented

2. Support

- Mostly open source, result from start-ups
  - Enables fact development
- Limited resources or credibility

3. Administration

- Require lot of skill to install and effort to maintain

NoSQL Databases

Five Challenges

4. Analytics and Business Intelligence

- Focused on web apps scenarios
  - Modern Web 2.0 applications
  - Insert-read-update-delete
- Limited ad-hoc querying
  - Even a simple query requires significant programming expertise

5. Expertise

- Few number of NoSQL experts available in the market

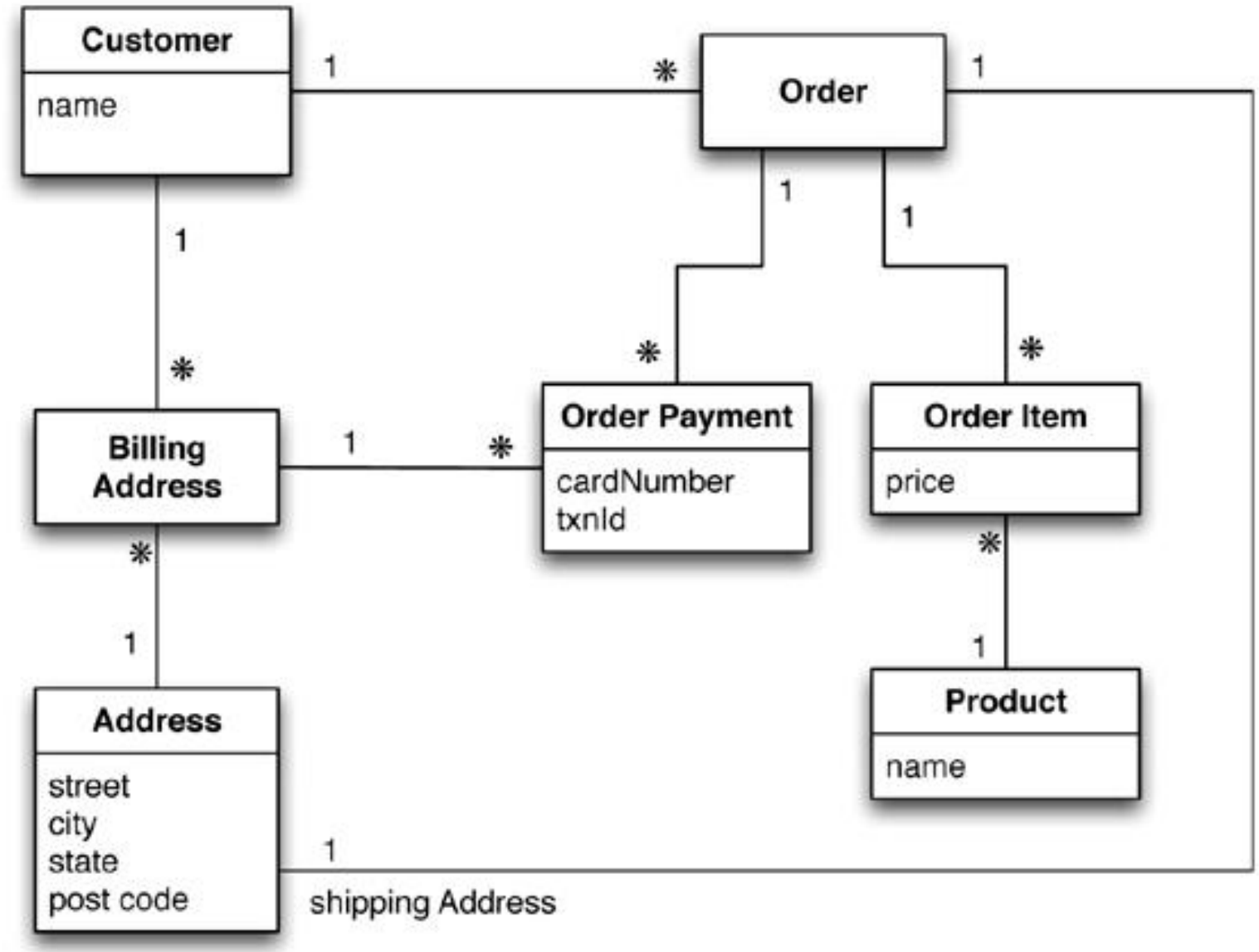
Data Assumptions

RDBMS	NoSQL
integrity is mission-critical	OK as long as most data is correct
data format consistent, well-defined	data format unknown or inconsistent
data is of long-term value	data are expected to be replaced
data updates are frequent	write-once, read multiple (no updates, or at least not often)
predictable, linear growth	unpredictable growth (exponential)
non-programmers writing queries	only programmers writing queries
regular backup	replication
access through master server	sharding across multiple nodes

NoSQL Data Model

Aggregates

- Data model = the model by which the database organizes data
- Each NoSQL solution has a different model
  - Key-value, document, column-family, graph
  - First three orient on aggregates
- Aggregate
  - A data unit with a complex structure
    - Not just a set of tuples like in RDBMS
  - Domain-Driven Design: “an aggregate is a collection of related objects that we wish to treat as a unit”
    - A unit for data manipulation and management of consistency


Customer
name
Order
Billing Address
Order Payment
cardNumber
txnId
Order Item
price
Address
street
city
state
post code
Product
name
shipping Address
1
*

Customer

Id	Name
1	Martin

Orders

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

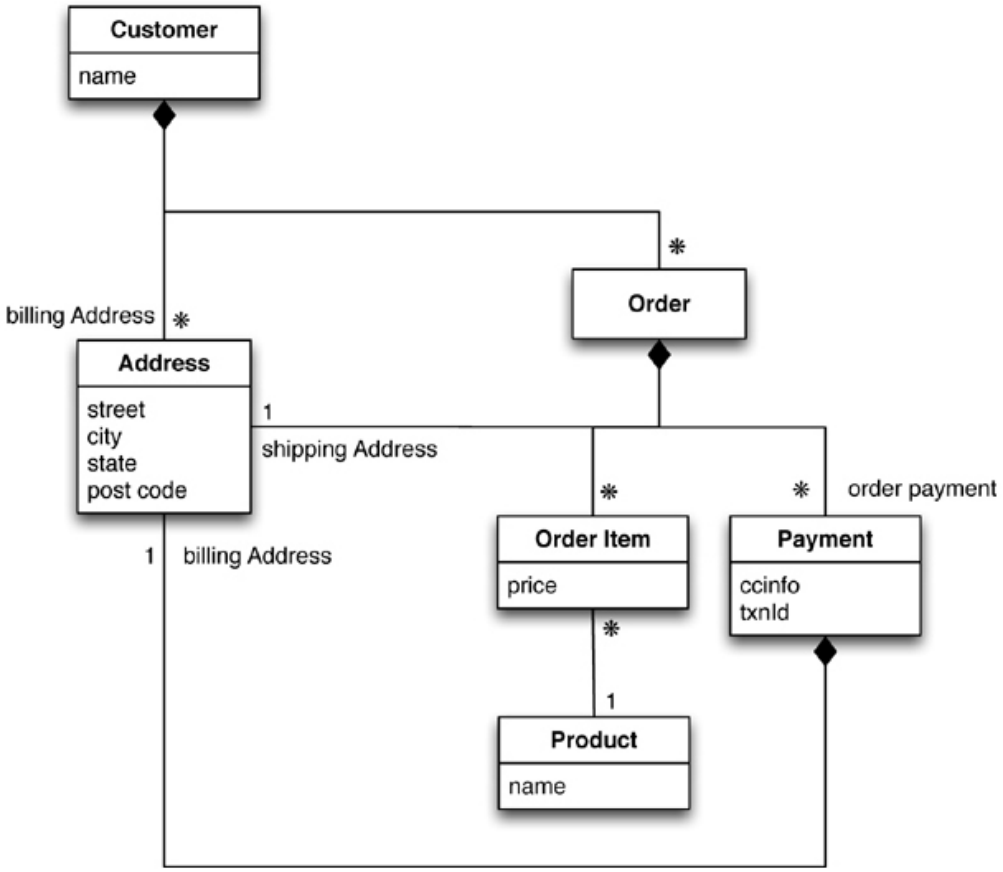
Id	OrderId	ProductId	Price
100	99	27	32.45

Address

Id	City
77	Chicago

OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

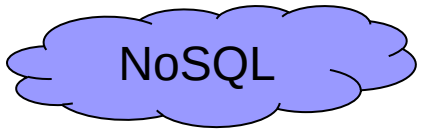



```
// in customers
{
"customer": {
"id": 1,
"name": "Martin",
"billingAddress": [{"city": "Chicago"}],
"orders": [
  {
    "id":99,
    "customerId":1,
    "orderItems":[
    {
    "productId":27,
    "price": 32.45,
    "productName": "NoSQL Distilled"
    }
  ],
  "shippingAddress":[{"city":"Chicago"}]
  "orderPayment":[
    {
    "ccinfo":"1000-1000-1000-1000",
    "txnId":"abelif879rft",
    "billingAddress": {"city": "Chicago"}
    }],
  }]
}
}
```

NoSQL Data Model

Aggregates – aggregate-ignorant

- There is no universal strategy how to draw aggregate boundaries
  - Depends on how we manipulate the data
- RDBMS and graph databases are aggregate-ignorant
  - It is not a bad thing, it is a feature
  - Allows to easily look at the data in different ways
  - Better choice when we do not have a primary structure for manipulating data



NoSQL Data Model

Aggregates – aggregate-oriented

- Aggregate orientation
  - Aggregates give the database information about which bits of data will be manipulated together
    - Which should live on the same node
  - Helps greatly with running on a cluster
    - We need to minimize the number of nodes we need to query when we are gathering data
- Consequence for transactions
  - NoSQL databases support atomic manipulation of a single aggregate at a time

NoSQL Databases

Materialized Views

- Disadvantage: the aggregated structure is given, other types of aggregations cannot be done easily
  - RDBMSs lack of aggregate structure → support for accessing data in different ways (using views)
- Solution: materialized views
  - Pre-computed and cached queries
- Strategies:
  - Update materialized view when we update the base data
    - For more frequent reads of the view than writes
  - Run batch jobs to update the materialized views at regular intervals

NoSQL Databases

Schemalessness

- When we want to store data in a RDBMS, we need to define a schema
- Advocates of schemalessness rejoice in freedom and flexibility
  - Allows to easily change your data storage as we learn more about the project
  - Easier to deal with non-uniform data
- Fact: there is usually an implicit schema present
  - The program working with the data must know its structure

Types of NoSQL Databases

Core:

- Key-value databases
- Document databases
- Column-family (column-oriented/columnar) stores
- Graph databases

Non-core:

- Object databases
- XML databases
- …

http://nosql-database.org/

Key-value store

Basic characteristics

- The simplest NoSQL data stores
- A simple hash table (map), primarily used when all access to the database is via primary key
- A table in RDBMS with two columns, such as ID and NAME
  - ID column being the key
  - NAME column storing the value
    - A BLOB that the data store just stores
- Basic operations:
  - Get the value for the key
  - Put a value for a key
  - Delete a key from the data store
- Simple → great performance, easily scaled
- Simple → not for complex queries, aggregation needs

Key-value store

Representatives

riak ✓

redis ✓

MemcachedDB

MEMCACHED

ORACLE

BERKELEY DB

HAMSTER DB
embedded database

amazon web services
Amazon DynamoDB

not
open-source

Project Voldemort

open-source
version

Key-value store

Suitable Use Cases

Storing Session Information

- Every web session is assigned a unique session_id value
- Everything about the session can be stored by a single PUT request or retrieved using a single GET
- Fast, everything is stored in a single object

User Profiles, Preferences

- Every user has a unique user_id, user_name + preferences such as language, colour, time zone, which products the user has access to, …
- As in the previous case:
  - Fast, single object, single GET/PUT

Shopping Cart Data

- Similar to the previous cases

Key-value store

When Not to Use

Relationships among Data

- Relationships between different sets of data
- Some key-value stores provide link-walking features
  - Not usual

Multioperation Transactions

- Saving multiple keys
  - Failure to save any one of them → revert or roll back the rest of the operations

Query by Data

- Search the keys based on something found in the value part

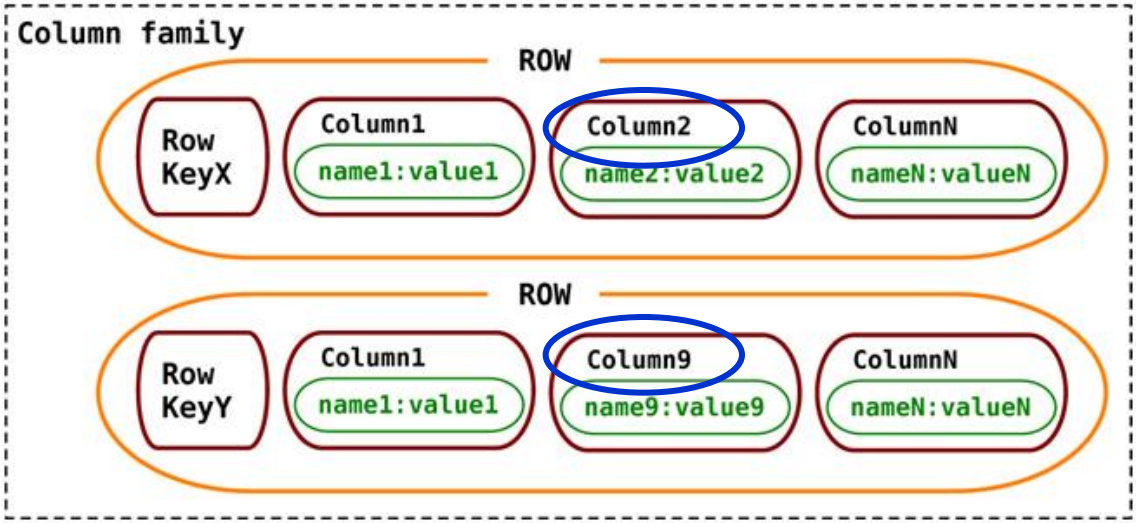
Operations by Sets

- Operations are limited to one key at a time
- No way to operate upon multiple keys at the same time

Column-Family Stores

Basic Characteristics

- Also “columnar” or “column-oriented”
- Column families = rows that have many columns associated with a row key
- Column families are groups of related data that is often accessed together
  - e.g., for a customer we access all profile information at the same time, but not orders



Column-Family Stores

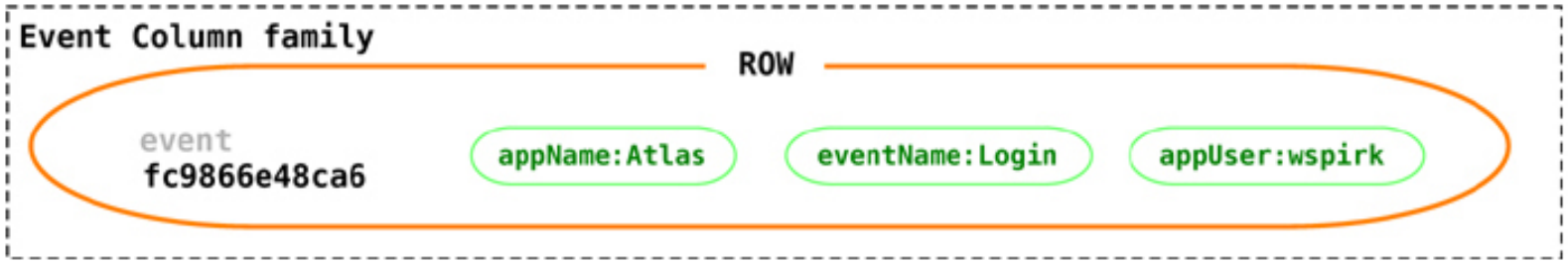
Representatives

Google's BigTable

HYPERTABLE

Column-Family Stores

Suitable Use Cases



Event Logging

- Ability to store any data structures → good choice to store event information

Content Management Systems, Blogging Platforms

- We can store blog entries with tags, categories, links, and trackbacks in different columns
- Comments can be either stored in the same row or moved to a different keyspace
- Blog users and the actual blogs can be put into different column families

Column-Family Stores

When Not to Use

Systems that Require ACID Transactions

- Column-family stores are not just a special kind of RDBMSs with variable set of columns!

Aggregation of the Data Using Queries

- (such as SUM or AVG)
- Have to be done on the client side

For Early Prototypes

- We are not sure how the query patterns may change
- As the query patterns change, we have to change the column family design

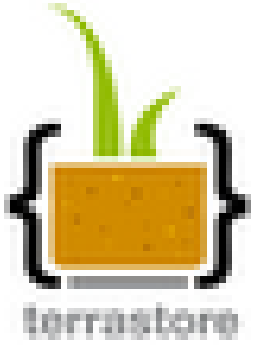
Document Databases

Basic Characteristics

- Documents are the main concept
  - Stored and retrieved
  - XML, JSON, …
- Documents are
  - Self-describing
  - Hierarchical tree data structures
  - Can consist of maps, collections (lists, sets, …), scalar values, nested documents, …
- Documents in a collection are expected to be similar
  - Their schema can differ
- Document databases store documents in the value part of the key-value store
  - Key-value stores where the value is examinable

Document Databases

Representatives



Lotus Notes
Storage Facility

Document Databases

Suitable Use Cases

Event Logging

- Many different applications want to log events
  - Type of data being captured keeps changing
- Events can be sharded (i.e. divided) by the name of the application or type of event

Content Management Systems, Blogging Platforms

- Managing user comments, user registrations, profiles, web-facing documents, …

Web Analytics or Real-Time Analytics

- Parts of the document can be updated
- New metrics can be easily added without schema changes
  - E.g. adding a member of a list, set,…

E-Commerce Applications

- Flexible schema for products and orders
- Evolving data models without expensive data migration

Document Databases

When Not to Use

Complex Transactions Spanning Different Operations

- Atomic cross-document operations
  - Some document databases do support (e.g., RavenDB)

Queries against Varying Aggregate Structure

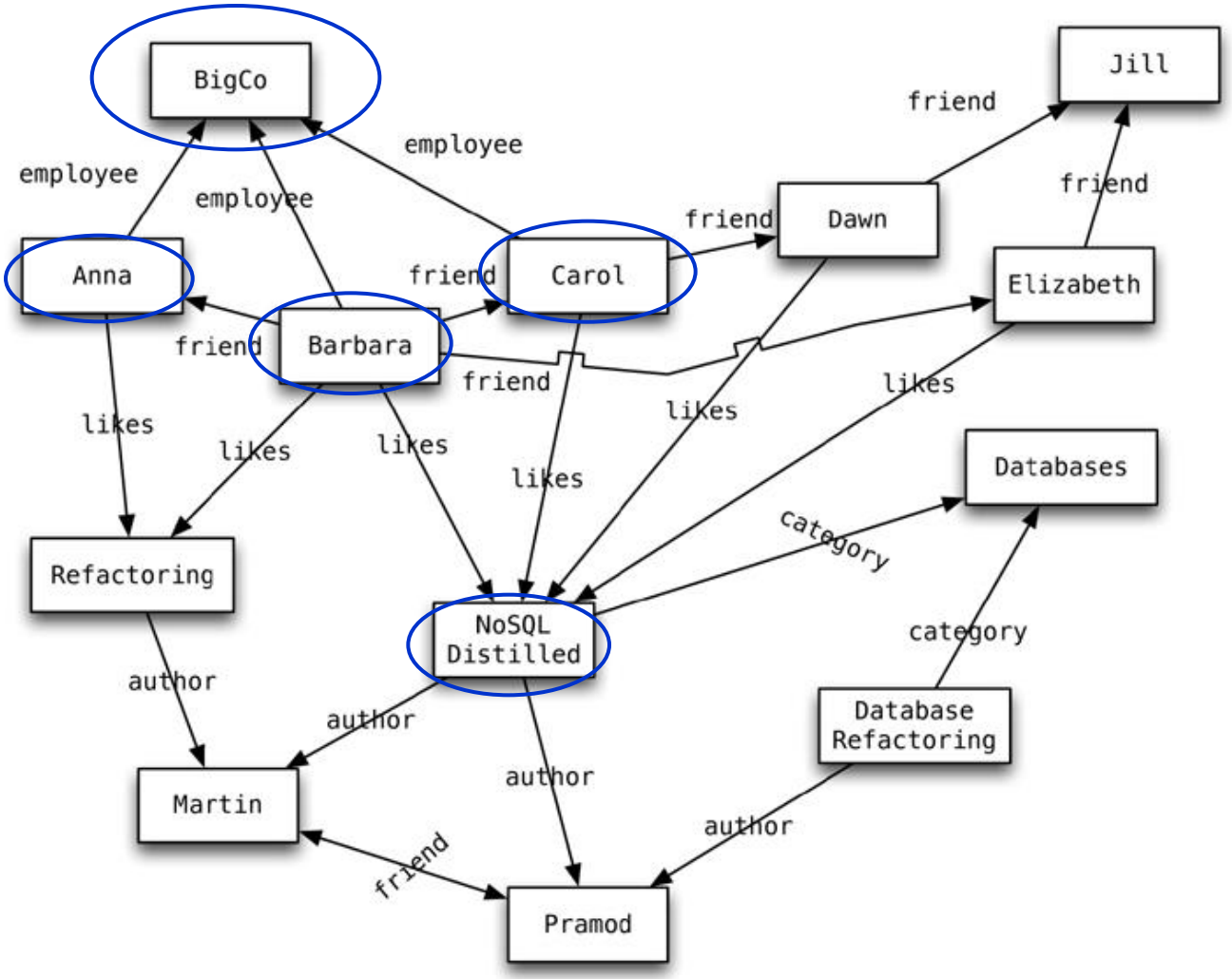
- Design of aggregate is constantly changing → we need to save the aggregates at the lowest level of granularity
  - i.e. to normalize the data

Graph Databases

Basic Characteristics

- To store entities and relationships between these entities
  - Node is an instance of an object
  - Nodes have properties
    - e.g., name
  - Edges have directional significance
  - Edges have types
    - e.g., likes, friend, …
- Nodes are organized by relationships
  - Allow to find interesting patterns
  - e.g., “Get all nodes employed by Big Co that like NoSQL Distilled”

Example:


BigCo
Jill
friend
employee
employee
employee
friend
friend
Dawn
Anna
friend
Carol
Elizabeth
friend
Barbara
friend
likes
likes
likes
likes
likes
likes
Databases
category
Refactoring
NoSQL
Distilled
category
author
author
Database
Refactoring
author
author
Martin
friend
Pramod

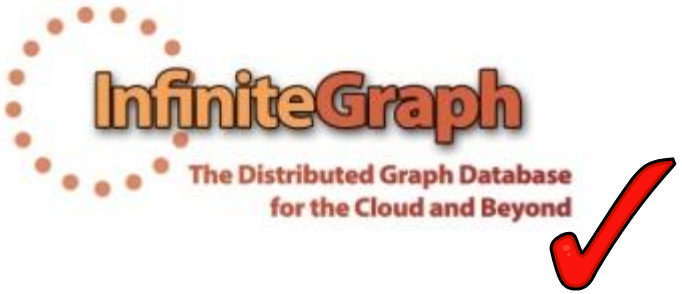
Graph Databases

RDBMS vs. Graph Databases

- When we store a graph-like structure in RDBMS, it is for a single type of relationship
  - “Who is my manager”
- Adding another relationship usually means a lot of schema changes
- In RDBMS we model the graph beforehand based on the Traversal we want
  - If the Traversal changes, the data will have to change
  - In graph databases the relationship is not calculated at query time but persisted

Graph Databases

Representatives



FlockDB

Graph Databases

Suitable Use Cases

Connected Data

- Social networks
- Any link-rich domain is well suited for graph databases

Routing, Dispatch, and Location-Based Services

- Node = location or address that has a delivery
- Graph = nodes where a delivery has to be made
- Relationships = distance

Recommendation Engines

- “your friends also bought this product”
- “when invoicing this item, these other items are usually invoiced”

Graph Databases

When Not to Use

- When we want to update all or a subset of entities
  - Changing a property on all the nodes is not a straightforward operation
  - e.g., analytics solution where all entities may need to be updated with a changed property
- Some graph databases may be unable to handle lots of data
  - Distribution of a graph is difficult or impossible

NoSQL Data Model

Aggregates and NoSQL databases

Key-value database

- Aggregate = some big blob of mostly meaningless bits
  - But we can store anything
- We can only access an aggregate by lookup based on its key

Document database

- Enables to see a structure in the aggregate
  - But we are limited by the structure when storing (similarity)
- We can submit queries to the database based on the fields in the aggregate

NoSQL Data Model

Aggregates and NoSQL databases

Column-family stores

- A two-level aggregate structure
  - The first key is a row identifier, picking up the aggregate of interest
  - The second-level values are referred to as columns
- Ways to think about how the data is structured:
  - Row-oriented: each row is an aggregate with column families representing useful chunks of data (profile, order history)
  - Column-oriented: each column family defines a record type (e.g., customer profiles) with rows for each of the records; a row is the join of records in all column families

References

- http://nosql-database.org/
- Pramod J. Sadalage – Martin Fowler: **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**
- Eric Redmond – Jim R. Wilson: **Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement**
- Sherif Sakr – Eric Pardede: **Graph Data Management: Techniques and Applications**
- Shashank Tiwari: **Professional NoSQL**