Modern Database Systems

Techniques and technologies for processing Big Data. Introduction to NoSQL databases.

Doc. RNDr. Irena Holubova, Ph.D.

Irena.Holubova@matfyz.cuni.cz

Big Data Tasks

- What do we need to do with Big Data?
  - aggregate
  - manipulate
  - analyze
  - visualize
- A number of techniques and technologies
  - Combination of statistics, computer science, applied mathematics, economics, …
    - Some adapted from techniques for smaller volumes of data
    - Some developed primarily for Big Data
  - New approaches appear rapidly

Big Data Analysis Techniques
Examples

- Association rule learning – discovering interesting relationships, i.e., "association rules," among variables in large databases
  - e.g., market basket analysis
- Classification – to identify the categories in which new data points belong, based on a training set containing data points that have already been categorized
  - Supervised learning
  - e.g., buying decisions
- Cluster analysis – classifying objects that split a diverse group into smaller groups of similar objects
  - Unsupervised learning
- Data fusion and data integration
- Signal processing

https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

Big Data Analysis Techniques
Examples

- Crowdsourcing - collecting data submitted by a large group of people or community
- Data mining - extract patterns from large datasets
  - Involves association rule learning, cluster analysis, classification, regression, …
- Time series analysis and forecasting
  - e.g., hourly value of a stock market index
- Sentiment analysis - identifying the feature/aspect/product about which a sentiment is being expressed,
  - Determining the type (i.e., positive, negative, or neutral)
  - Determining the degree and strength of the sentiment
- Visualization
- …

Big Data Related Technologies

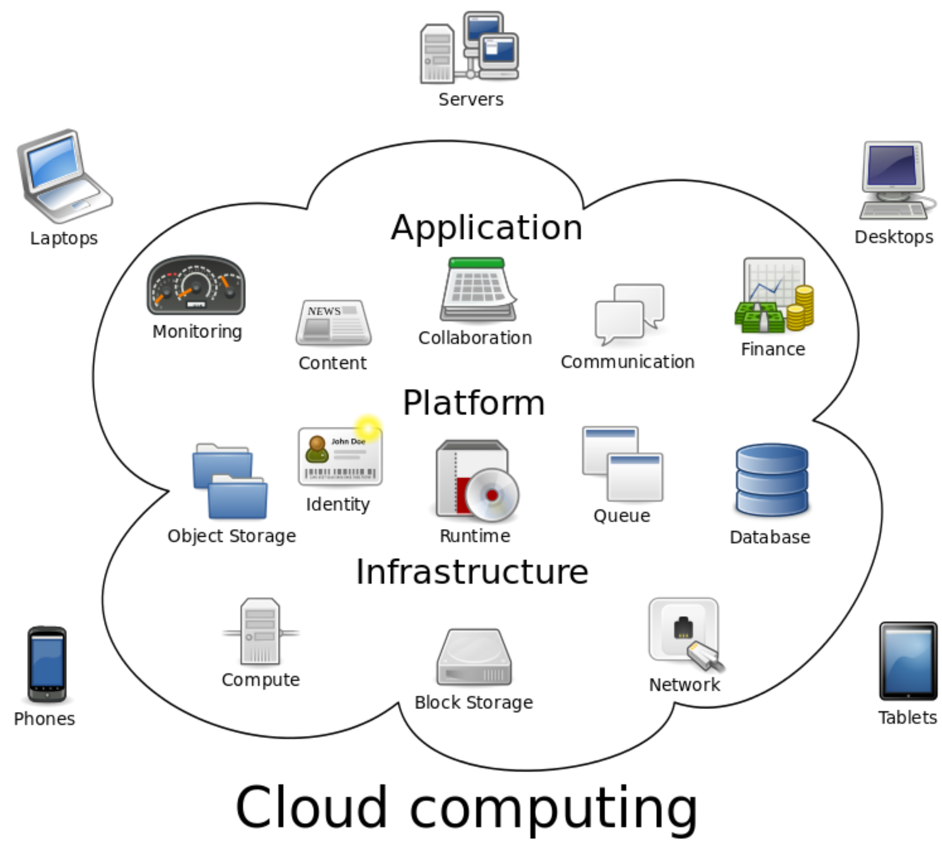
- Distributed file systems
  - e.g., HDFS
- Distributed databases
  - Primarily NoSQL databases
  - And many other types
- Cloud computing
- Data analytics
  - Batch
  - Real-time
  - Stream
- …

Cloud Computing

- Way of creating SW
- Idea: Providing shared IT technologies (HW/SW) and/or data to computers and other devices on demand
  - **Software as a Service** (SaaS)
    - For end-users
  - **Platform as a Service** (PaaS)
    - For developers (tools for SW implementation/deployment)
  - **Infrastructure as a Service** (IaaS)
    - For providing robust expensive and inaccessible HW
- Users pay for the usage (rent)
  - Time of usage, size of the data, …

Cloud Computing

- Services
  - Private – for internal usage of a company
  - Public – for anyone
  - Community – for a selected community
    - Set of customers
  - … and their combinations



Cloud Computing

- Advantages
  - Users do not have to manage the technologies
    - Buy, install, upgrade, maintain, …
  - Thanks to the Internet can be used anywhere
  - Service provider can provide distinct solutions for distinct requirements
    - Within the respective capabilities
  - Data stored at server(s) of the cloud can be easily shared

Cloud Computing

- Disadvantages and challenges
  - We store our private data on a public cloud
    - Theoretically vulnerable (but the protection techniques are still being improved)
  - Vendor lock-in
    - Proprietary technologies and solutions
  - High prices
    - For small companies, universities, …
- Note: Well-known applications have similar features
  - Google Calendar, Dropbox, Gmail

Cloud Computing Platforms

For more details see courses:

Virtualization and Cloud Computing (NSWI150)
Cloud Application Development (NSWI152)

Cloud Computing and Big Data

- We need a cluster of nodes
  - Expensive, demanding installation and maintenance, …

→ Use cloud computing
  - Scalable solutions without the maintenance part
  - For Big Data often cheaper than the HW
    - When the infrastructure is not used, it can be provided to other users
      - E.g. data analysis is done in particular time intervals
  - Easier solutions or even directly particular applications
  - Available "immediately"
- We can focus on the specific functionality
  - E.g. efficient analytical processing of the data
- But: the other disadvantages (safety, vendor lock-in) remain

Types of NoSQL Databases

Core:

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

Non-core:

- Object databases
- XML databases
- …

Further novel extensions:

- Multi-model databases
- Array databases
- NewSQL 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 $\rightarrow$ great performance, easily scaled
- Simple $\rightarrow$ not for complex queries, aggregation needs

Key-value store

Representatives

MemcachedDB

riak

redis

MEMCACHED

ORACLE

BERKELEY DB

HamsterDB
embedded database

amazon web services
Amazon DynamoDB

not
open-source

open-source
version

Project
Voldemort

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

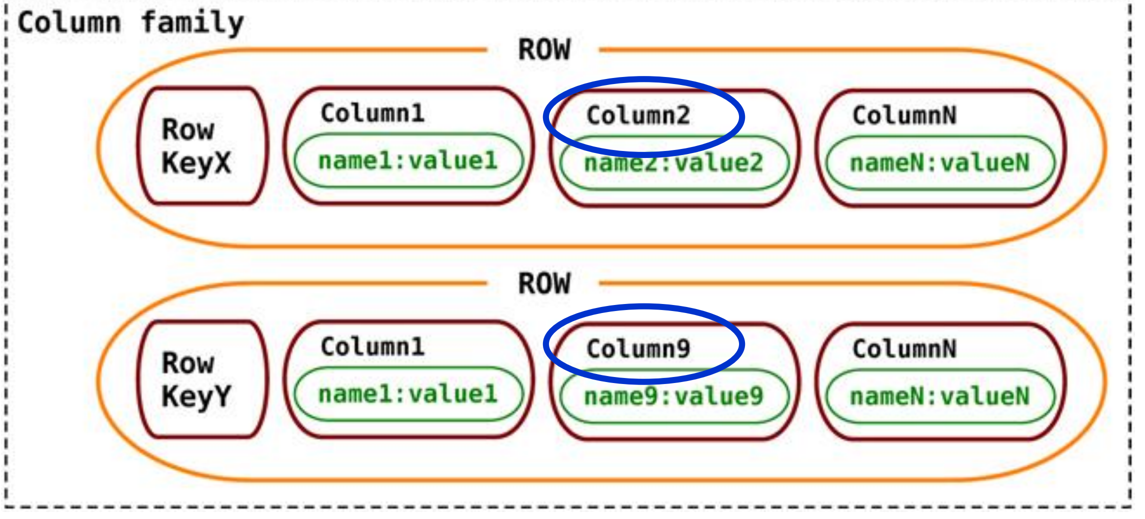
Operations by Sets

- 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 <u>many</u> 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 ✓

HBASE ✓

Cassandra ✓

HYPERTABLE

SimpleDB

Example: Cassandra

RDBMS	Cassandra
database instance	cluster
database	keyspace
table	column family
row	row
column (same for all rows)	column (can be different per row)

- Column = basic unit, consists of a name-value pair
  - Name serves as a key
  - Stored with a timestamp (expired data, resolving conflicts, …)
- Row = a collection of columns attached or linked to a key
  - Columns can be added to any row at any time without having to add it to other rows
- Column family = a collection of <u>similar</u> rows
  - Rows do not have to have the same columns

Example: Cassandra

```
{ name: "firstName",
  value: "Martin",
  timestamp: 12345667890 }
```

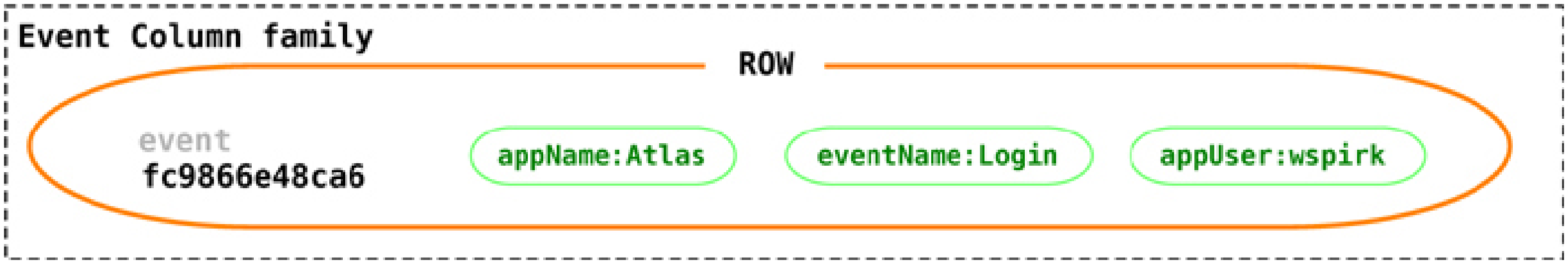
- Column key of firstName and the value of Martin

```
{ "pramod-sadalage" : {
    firstName: "Pramod",
    lastName: "Sadalage",
    lastVisit: "2012/12/12" }
  "martin-fowler" : {
    firstName: "Martin",
    lastName: "Fowler",
    location: "Boston" } }
```

- pramod-sadalage row and the martin-fowler row with different columns; both rows are a part of a column family

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 <u>not</u> 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

Data – Example

```
{ "firstname": "Martin",
  "likes": [ "Biking",
             "Photography" ],
  "lastcity": "Boston",
  "lastVisited": }
```

```
{ "firstname": "Pramod",
  "citiesvisited": [ "Chicago", "London", "Pune", "Bangalore" ],
  "addresses": [
    { "state": "AK",
      "city": "DILLINGHAM",
      "type": "R"    },
    { "state": "MH",
      "city": "PUNE",
      "type": "R" }  ],
  "lastcity": "Chicago“ }
```

Document Databases

Data – Example

- Data are similar, but have differences, e.g., in attribute names
  - Still belong to the same collection
- We can represent
  - A list of cities visited as an array
  - A list of addresses as a list of documents embedded inside the main document

Document Databases

Representatives

Lotus Notes
Storage Facility

Document Databases

Sample Query – MongoDB

- Query language which is expressed via JSON
  - Where clause, sorting, count, sum, showing the execution plan, …

```
SELECT * FROM order
```

```
db.order.find()
```

```
SELECT * FROM order WHERE customerId = "883c2c5b4e5b"
```

```
db.order.find({"customerId":"883c2c5b4e5b"})
```

```
SELECT orderId,orderDate FROM order
WHERE customerId = "883c2c5b4e5b"
```

```
db.order.find({customerId:"883c2c5b4e5b"},
              {orderId:1,orderDate:1})
```

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)

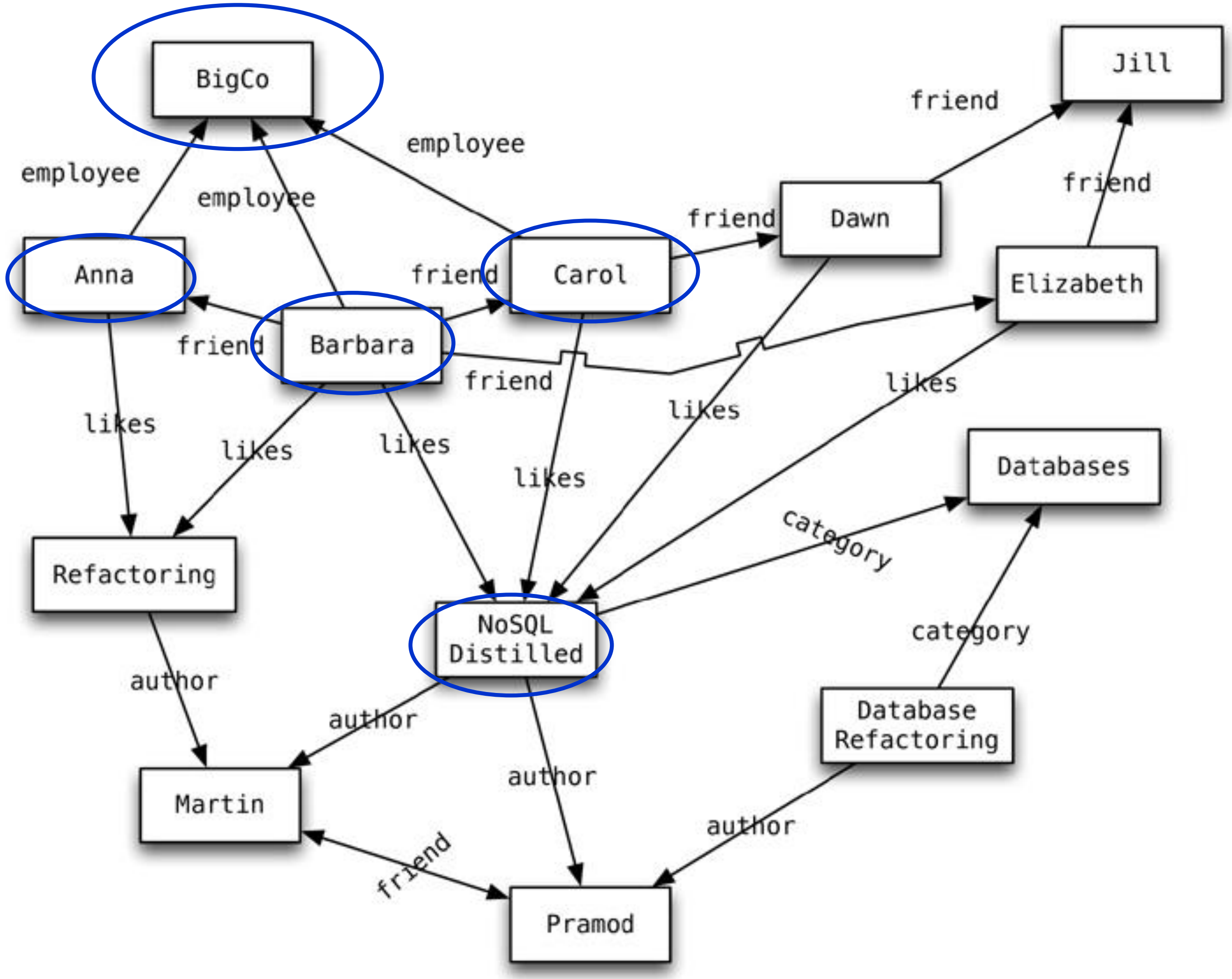
Queries against Varying Aggregate Structure

- 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 people (= nodes in the graph) employed by Big Co that like (book called) NoSQL Distilled"

Example:



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

Neo4j
the graph database

InfiniteGraph
The Distributed Graph Database
for the Cloud and Beyond

OrientDB®

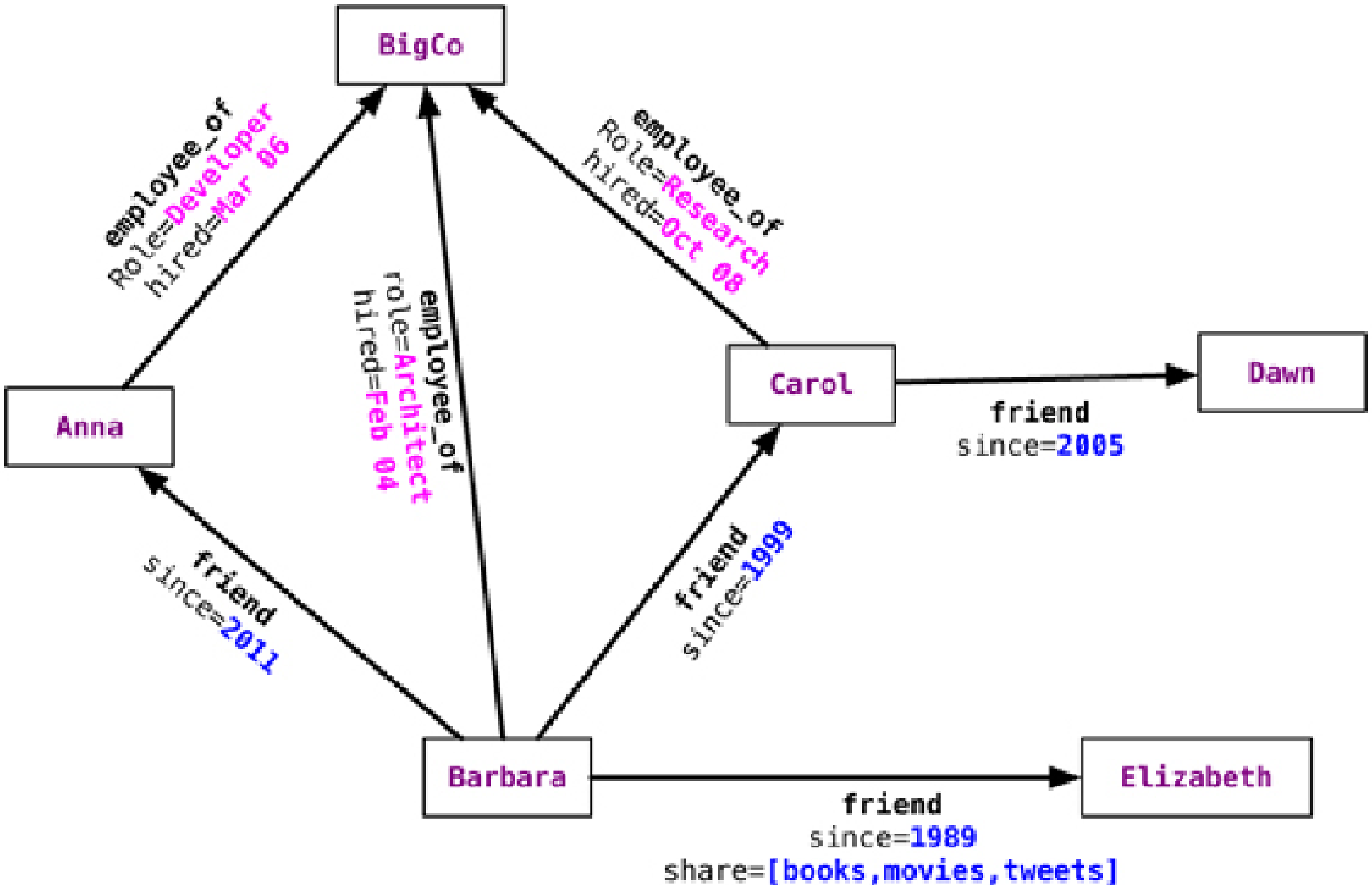
FlockDB

Graph Databases

Basic Characteristics

- Nodes can have different types of relationships between them
  - To represent relationships between the domain entities
  - To have secondary relationships
    - Category, path, time-trees, quad-trees for spatial indexing, linked lists for sorted access, …
- There is no limit to the number and kind of relationships a node can have
  - Except for upper limits of a particular system, if any
- Relationships have type, start node, end node, own properties
  - e.g., since when did they become friends

Example:


BigCo
Anna
Carol
Dawn
Barbara
Elizabeth
employee_of
Role=Developer
hired=Mar 06
employee_of
role=Architect
hired=Feb 04
employee_of
Role=Research
hired=Oct 08
friend
since=2005
friend
since=2011
friend
since=1999
friend
since=1989
share=[books,movies,tweets]

Example: Neo4J

```
Node martin = graphDb.createNode();
martin.setProperty("name", "Martin");
Node pramod = graphDb.createNode();
pramod.setProperty("name", "Pramod");

martin.createRelationshipTo(pramod, FRIEND);
pramod.createRelationshipTo(martin, FRIEND);
```

- We have to create a relationship between the nodes in both directions
  - Nodes know about INCOMING and OUTGOING relationships

Graph Databases
Query

- Properties of a node/edge can be indexed
- Indices are queried to find the starting node to begin a traversal

creating index

adding nodes

```
Transaction transaction = graphDb.beginTx();
try {
    Index<Node> nodeIndex = graphDb.index().forNodes("nodes");
    nodeIndex.add(martin, "name", martin.getProperty("name"));
    nodeIndex.add(pramod, "name", pramod.getProperty("name"));
    transaction.success(); }
finally {
    transaction.finish(); }
```

retrieving a node

```
Node martin = nodeIndex.get("name", "Martin").getSingle();
allRelationships = martin.getRelationships();
```

getting all its relationships

Graph Databases

Query – finding paths

- We are interested in determining if there are multiple paths, finding all of the paths, the shortest path, …

```
Node barbara = nodeIndex.get("name", "Barbara").getSingle();
Node jill    = nodeIndex.get("name", "Jill").getSingle();
PathFinder<Path> finder1 = GraphAlgoFactory.allPaths(
        Traversal.expanderForTypes(FRIEND,Direction.OUTGOING),
                                                  MAX_DEPTH);
Iterable<Path> paths = finder1.findAllPaths(barbara, jill);

PathFinder<Path> finder2 = GraphAlgoFactory.shortestPath(
        Traversal.expanderForTypes(FRIEND,Direction.OUTGOING),
                                                  MAX_DEPTH);
Iterable<Path> paths = finder2.findAllPaths(barbara, jill);
```

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

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 the structure in an 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

Multi-model stores

- Combine various data models, including aggreagate-oriented
- Support references and queries across the models

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**