IN 4410 - Big Data Analytics

NoSQL Databases

Week 3

fppt.com

Outline

- Introduction
- NoSQL Storage Types
- NewSQL
- HBase
- **Reference:**
  - Big and Analytics by Seema Acharya
  - Getting Started with NoSQL by Gaurav Vaish

fppt.com

NoSQL

NoSQL stands for "**Not Only SQL**" or "**Not Relational**", is not entirely agreed upon.

- The term NoSQL was first used in 1998 by Carlo Strozzi as the name of his small relational DBMS that did not use SQL for data manipulation.

- Starting in 2009, the term NoSQL is used for the growing number of distributed data management systems that abandoned the support of ACID transactions.

fppt.com

Features of NoSQL

- Non-relational – do not adhere to relational data model
- Distributed – data is distributed across several nodes in a cluster
- No support for ACID properties (Atomicity, Consistency, Isolation, Durability)
- No fixed table schema – has a flexible schema
  - Ability to dynamically add new attributes to data records
- Efficient use of distributed indexes and RAM for data storage

fppt.com

Why NoSQL?

- It has a scale out architecture instead of the monolithic architecture of relational databases
- It can house large volumes of structured, semi structured and unstructured data
- **Dynamic schema**: Allow insertion of data without a predefined schema
- **Auto-sharding**: Automatically spreads data across an arbitrary number of servers
- **Replication**: guarantees high availability, fault tolerance and disaster recovery

fppt.com

CAP Theorem or Brewer's Theorem

- Indicates tradeoffs in NoSQL Systems
- No distributed system can simultaneously ensure the following three properties.
  - **Consistency**: Every read fetches last write. All the nodes contain the consistent data at any time (all the users see the same data at any time.)
  - **Availability**: Reads and writes always succeed. When certain nodes fail, the remaining nodes continue to operate.
  - **Partition-tolerance**: System will continue to function when network partition occurs. If the system splits into disconnected groups of nodes, each group continues to operate.

NoSQL systems generally give up consistency.

fppt.com

Advantages of NoSQL

- Can easily scale-up and down
  - Cluster scale: it allows distribution of database across 100+ nodes
  - Performance scale: it sustains over 100 000+ database reads and writes per second
  - Data scale: it supports housing of 1 billion+ documents in the database
- Does not require a pre-defined schema
- Cheap, easy to implement: at low cost
- Relaxes the data consistency requirement

fppt.com

Advantages of NoSQL

- Data can be replicated to multiple nodes and can be partitioned
  - **Sharding** - each node of the system contains a piece of data called shard and performs operations on that piece.
    - For maintaining horizontal scalability
    - Replication is used as the number of nodes that store data increases, the probability of faults increases
  - **Replication** - an approach under which the same data are stored on several nodes of the network.
    - This improves the system reliability and helps counteract both failures of individual nodes and entire clusters.
    - A significant problem in replication is to maintain the consistency of data copies

Two basic techniques used in distributed data management systems

fppt.com

Data Replication Schemas

- **Master-Slave**
  - Data modification operations are processed only on the master node
  - Updates are synchronously or asynchronously propagated to slave nodes.
  - The data may be read both from the master node (which always contains the latest data version) and slave nodes that may contain out dated data if the replication is performed asynchronously.
- **Master-master or Multi-master**
  - Nodes can process update operations and propagate these updates to other nodes.
  - It is difficult to implement synchronous replication, and delays due to network communications can be significant.
  - If the updates are performed asynchronously, conflicting data versions may occur.

fppt.com

Limitations of NoSQL

- RDBMS focus on ***ACID properties*** while NoSQL systems manly aims at CAP theorem.
- NoSQL does not support ***joins***, ***Group by***.
  - Compensates by embedded documents as in MongoDB
- NoSQL does not have a standard ***SQL*** interface
  - E.g. MongoDB query language, Cassendra query language (CQL)
- The data models supported by NoSQL systems are typically simpler than the relational model, and they usually do not require a fixed database schema and ***integrity constraints***.
- NoSQL systems are still young compared to SQL oriented DBMSs

fppt.com

SQL Vs NoSQL

RDBMS	NoSQL
Relational database	Non-relational, Distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document based, graph based, wide column store or key value pair databases
Vertically scalable (by increasing computer resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage (as in JSON)
Emphasis on ACID properties	Follows Brewer's CAP theorem
Excellent support for vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g. MongoDB)

fppt.com

NewSQL

- New modern RDBMS
  - Supports relational model
- Maintains ACID properties
- Has scalable performance of NoSQL for OLTP
- Uses SQL as the primary interface

fppt.com

	RDBMS	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity	Yes	No	Maybe
Adherence to data model	Relational model		
Data format flexibility	No	Yes	Maybe
Scalabilty	Scale up – vertical scaling	Scale out – Horizontal scaling	Scale out
Distributed computing	Yes	Yes	Yes
Community support	Huge	Growing	Slowly growing

fppt.com

NoSQL Storage Types

- NoSQL databases categorized based on how the data is stored
- Storage Types:
  - Column Oriented
  - Document Store
  - Key Value Store
  - Graph

fppt.com

1. Column Oriented Databases

- Store data as columns as opposed to rows as in RDBMS
- First application in 1969: TAXIR – a biology information retrieval focused application
- Popular Databases: HBase, Apache Cassandra, MS SQL Server 2012 Enterprise Edition, Google BigTable.
- Advantages:
  - Allow adding columns overtime without worrying about filling default values for existing rows for the new columns,
  - Partial data access without touching unrelated columns
  - Faster execution
  - Same and similar adjacent values can be compressed effectively.

fppt.com

1. Column Oriented Databases

- Dataset

EmpId	FirstName	LastName	Age	Salary
SM1	Anuj	Perera	45	100000
MM2	Dinesh		34	50000
T3	Vivek	Silva	39	75000
E4	Kevin	Nugera	32	20000

In **RDBMS**, stored internally as

SM1, Anuj, Perera, 45, 100000

MM2, Dinesh, , 34, 50000

T3, Vivek, Silva, 39, 75000

E4, Kevin, Nugera, 32, 20000

In **Column oriented**, stored internally as

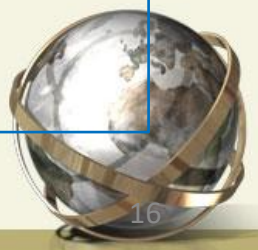
SM1, MM2, T3, E4

Anuj, Dinesh, Vivek, Kevin

Perera, , Silva, Nugera

45, 34, 39, 32

100000, 50000, 75000, 20000



fppt.com

2. Document Store

- Also known as Document oriented database
- Allows insert, retrieve and manipulation of semi structured data
- The unit of storage in document stores is a document
  - this is an object that has an arbitrary set of attributes (fields), which can be represented, e.g., in JSON
- Compared to RDBMS, documents themselves act as records (or rows), may not support specific schema as in RDBMS (as two records may have completely different set of columns)
- However, index can be created and queried

fppt.com

2. Document Store

- Document content using JSON

Document 1

```
{
    "EmpId" : "SM1",
    "FirstName" : "Anuj",
    "LastName" : "Perera",
    "Age" : 45,
    "Salary" :  100000
}
```

Document 2

```
{
    "EmpId" : "MM2",
    "FirstName" : "Dinesh",
     "Age" : 45,
    "Salary" :  100000
    "Address" : {
        "Line1" : "123, 4th Street",
        "City" : "Moratuwa"
        },
    "Projects" :  [
        "nosql-migration",
        "top-secret-007"
        ]
}
```

fppt.com

2. Document Store

Document 3

```
{
    “LocationId” : “Moratuwa-SDC-BTP”,
    “RegisteredName” : “ABC Software Development Ltd.”,
    “RegisteredAddress” :  {
        “Line1” : “123, 4th Street”,
        “City” :  “Moratuwa”,
    },
}
```

No Correlation with Document 1 and Document 2

fppt.com

2. Document Store

- Best practice: Document ID must be embedded in the document in a standard location.
- Modified content

```
{
        “docId” : “SM1”
        …..
}
{
        “docId” : “MM2”
        …..
}
{
        “docId” : “Moratuwa-SDC-BTP“
        ……
}
```

fppt.com

2. Document Store

- Advantages:
  - Content can be schema-less (or loosely defined) to store different types of content
  - Stores based on XML, SML, JSON, BSON and YAML support to retrieve or update a record partially over HTTP protocol using RESTful API
  - Searching through multiple entity types is trivial compared to RDBMS or column-oriented data stores as there is no concept of tables.
  - JSON based data stores are easy to define
- Popular Databases: MongoDB, CouchDB, Apache Cassandra, BaseX

fppt.com

3. Key-value Store

- Closely related to document store
- Difference:
  - document oriented databases provide the capability of querying collections of documents using several constraints on the attributes; they can execute aggregate queries, sort the results, support indexes on document fields, etc.
- Allows storage of values (structured or unstructured data) against one unique key
- Secondary keys and indexes are not supported
- No need to enforce a schema on the value
- This is similar to hash tables or concept of maps or associative maps

fppt.com

Popular Key Value Stores

- Redis - open source data management system written in C. It is used by large scale projects like Twitter, Instagram, Digg, Github, StackOverflow, Flickr.
- Voldemort – open source data store implemented in Java. It is intensively used in LinkedIn.
- Memcached
- Berkley DB

fppt.com

Key-Value Store

- Pros: Fast and horizontal scalability
- Cons: No validation, logic or structure

fppt.com

4. Graph Store

- Relationships are represented as graphs
- There can be multiple links between 2 nodes in a graph to represent the multiple relationships that the 2 nodes have
- For relation heavy data
  - E.g. social relationship between people, transport links between places or network topologies between connected systems
- E.g. Neo4j, FlockDB (from Twitter)

fppt.com

Graph

- Pros: Easy representation, retrieval and manipulation of relationships between the entities of the system

- Cons: Do not use them for complete data store

fppt.com

Multi-storage Type Databases

- Support multiple storage types
- E.g.
  - OrientDB: Supports document store, key- value and graph
  - ArangoDB: Supports document store, key- value and graph
  - Aerospike: Hybrid between RDBMS and NoSQL Store (Supports document store, key- value and graph)

fppt.com

IN 4410 - Big Data Analytics

NOSQL Databases - HBase

fppt.com

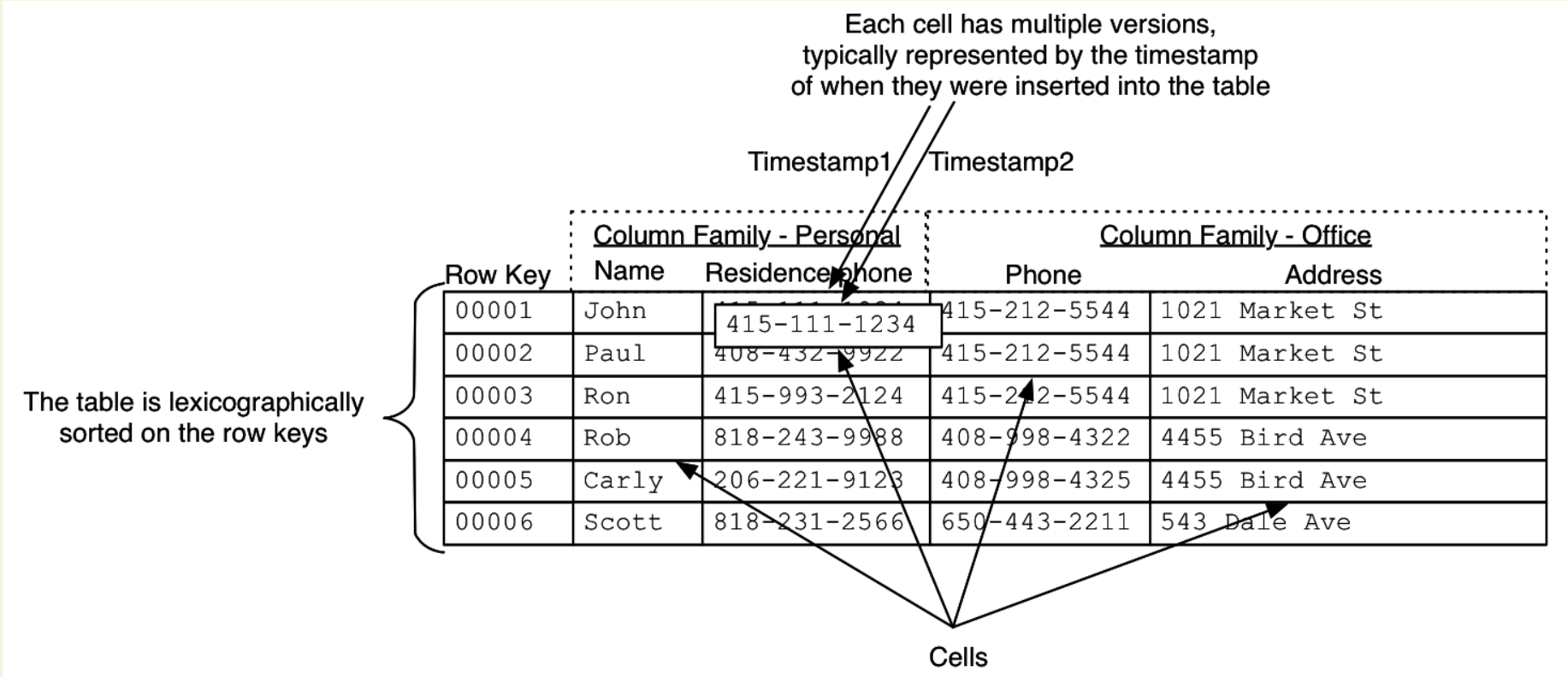
HBase

- An open source project written in Java and developed by Apache Software Foundation.
- HBase adheres to the principles of BigTable.
- It uses the Apache Hadoop framework for distributed computations.
- HBase does not support secondary indexes—records can be queried only by the primary key or by scanning the table.
- One can work with HBase using Java API, REST interface, and using Avro and Thrift.
- HBase is used in big applications and projects with high workload, such as Facebook (for the Facebook Messages service) and Twitter (for supporting MapReduce, people search, and other tasks).

fppt.com

HBase Data Model

- A column-family oriented database



fppt.com

Key Terms

- **Table -** HBase organizes data into tables and has many rows. Table names are Strings.
- **Row** – Rows are identified uniquely by their row key. Row keys do not have a data type and are always treated as a byte[ ] (byte array). Atomic key-value container.
- **Column Qualifier or Column**- A key in the k/v container inside a row. Column qualifiers need not be specified in advance. Column qualifiers need not be consistent between rows (each row can have different keys).

  Like row keys, column qualifiers do not have a data type and are always treated as a byte[ ].

fppt.com

Key Terms

- **Column Family** - Data within a row is grouped by column family.
  - Column families also impact the physical arrangement of data stored in HBase (that is divide columns into physical files). For this reason, they must be defined up front and are not easily modified.
  - Every row in a table has the same column families, although a row need not store data in all its families. Column families are Strings.

- **Cell** - A combination of row key, column family, and column qualifier uniquely identifies a cell. The data stored in a cell is referred to as that cell's value.
- **Value** - a value in the k/v container inside a row. Values also do not have a data type and are always treated as a byte[ ].

fppt.com

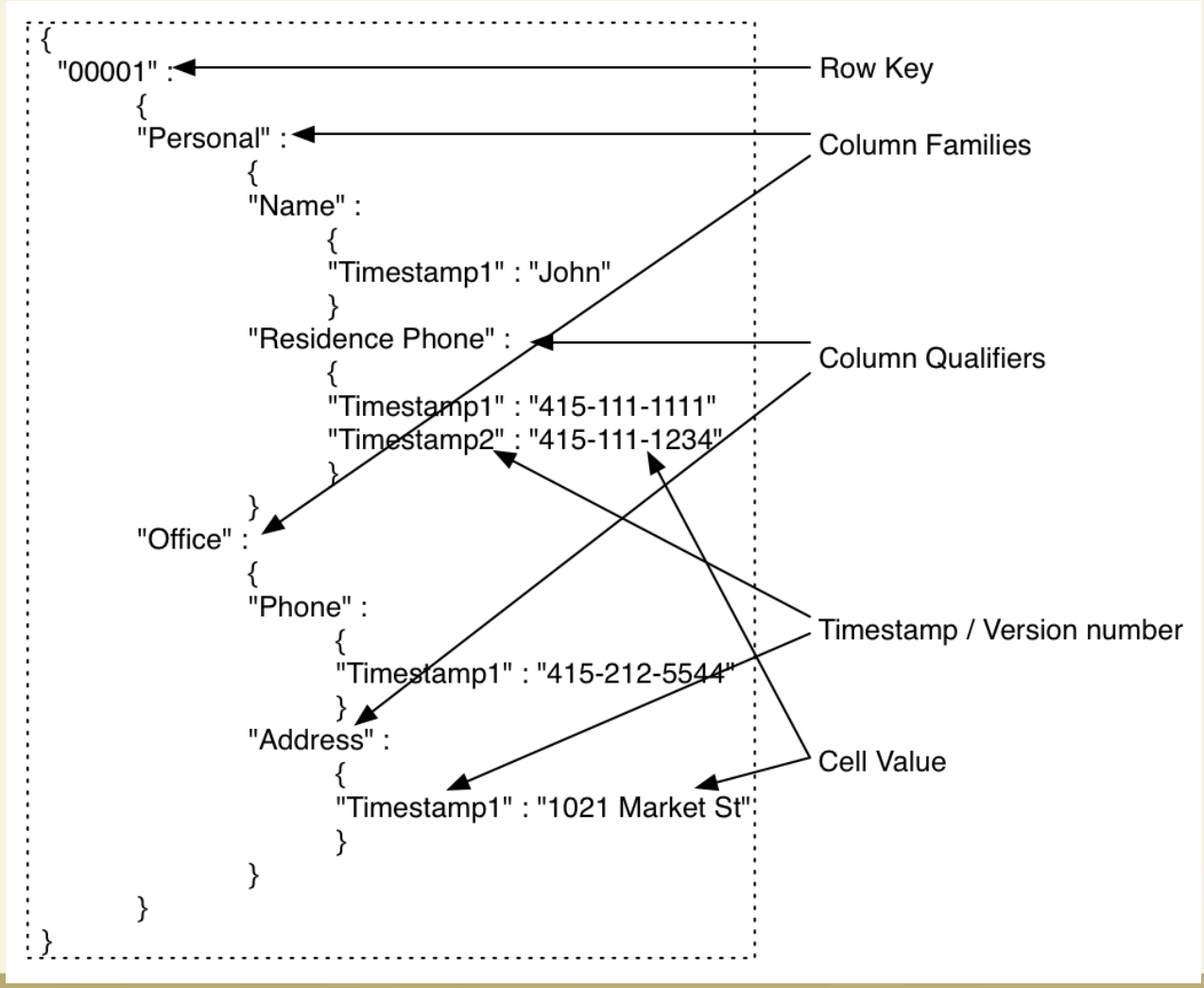
Timestamp

- Values within a cell are versioned.
- Timestamp is in long milliseconds and is sorted in descending
- Timestamp use data type - Long integers
- Versions are identified by their version number, which by default is the timestamp of when the cell was written.
- If a timestamp is not specified during a write, the current timestamp is used.
- If the timestamp is not specified for a read, the latest one is returned.

fppt.com

Multidimensional Map

- One row in an HBase Table



fppt.com

Key-Value Store

00001 → { Personal : { Name : { Timestamp1 : John }, Residence Phone : { Timestamp1 : 415-111-1234 } },
{ Office : { Phone : { Timestamp1 : 415-212-5544 }, Address : { Timestamp1 : 1021 Market St } } }

00001 , Personal → { Name : { Timestamp1 : John }, Residence Phone : { Timestamp1 : 415-111-1234 } }

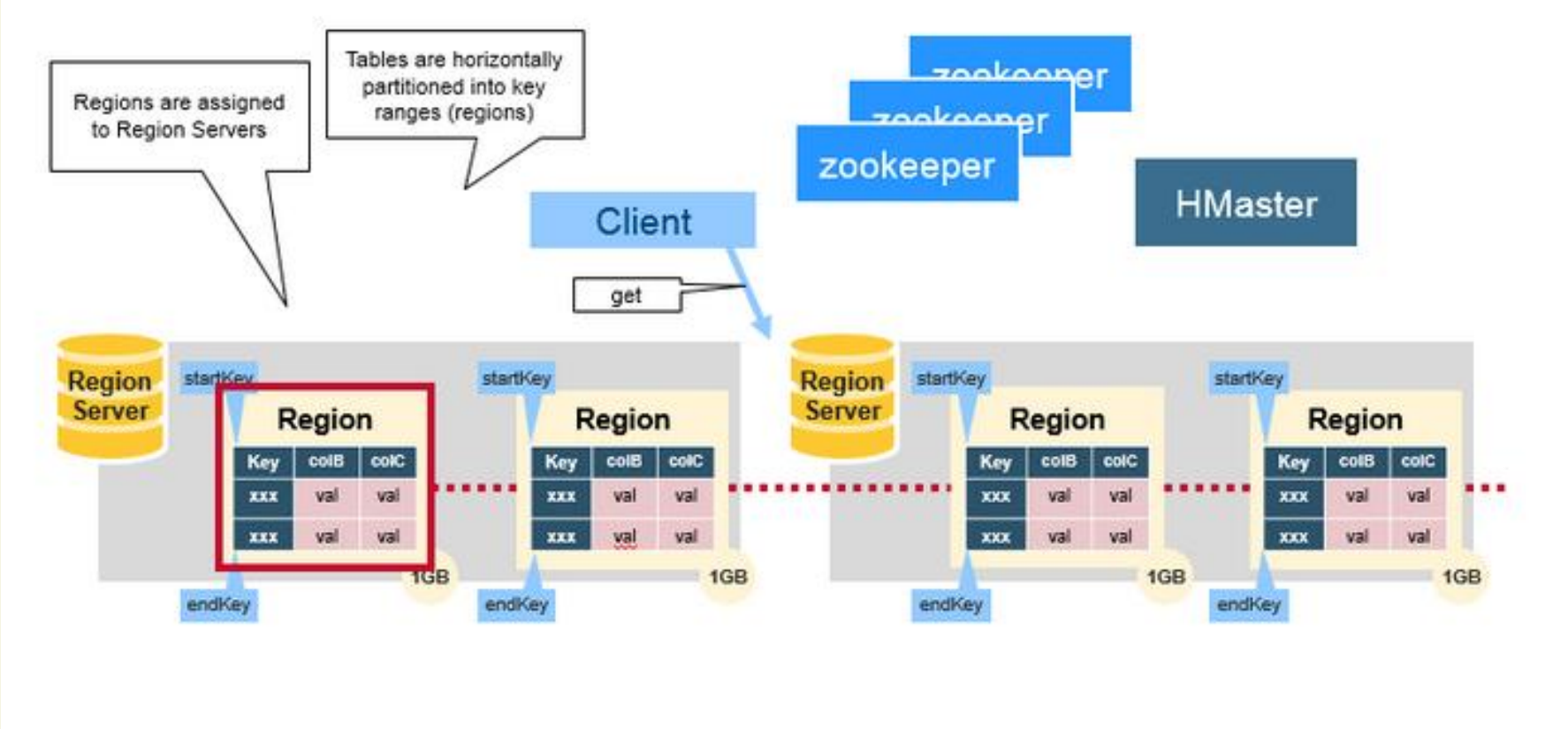
00001, Personal:Residence Phone → { {Timestamp1 : 415-111-1111 } , { Timestamp2 : 415-111-1234))

00001, Personal:Residence Phone , Timestamp1 → { 415-111-1111)

00001, Personal:Residence Phone , Timestamp2 → { 415-111-1234)

fppt.com

Region Server



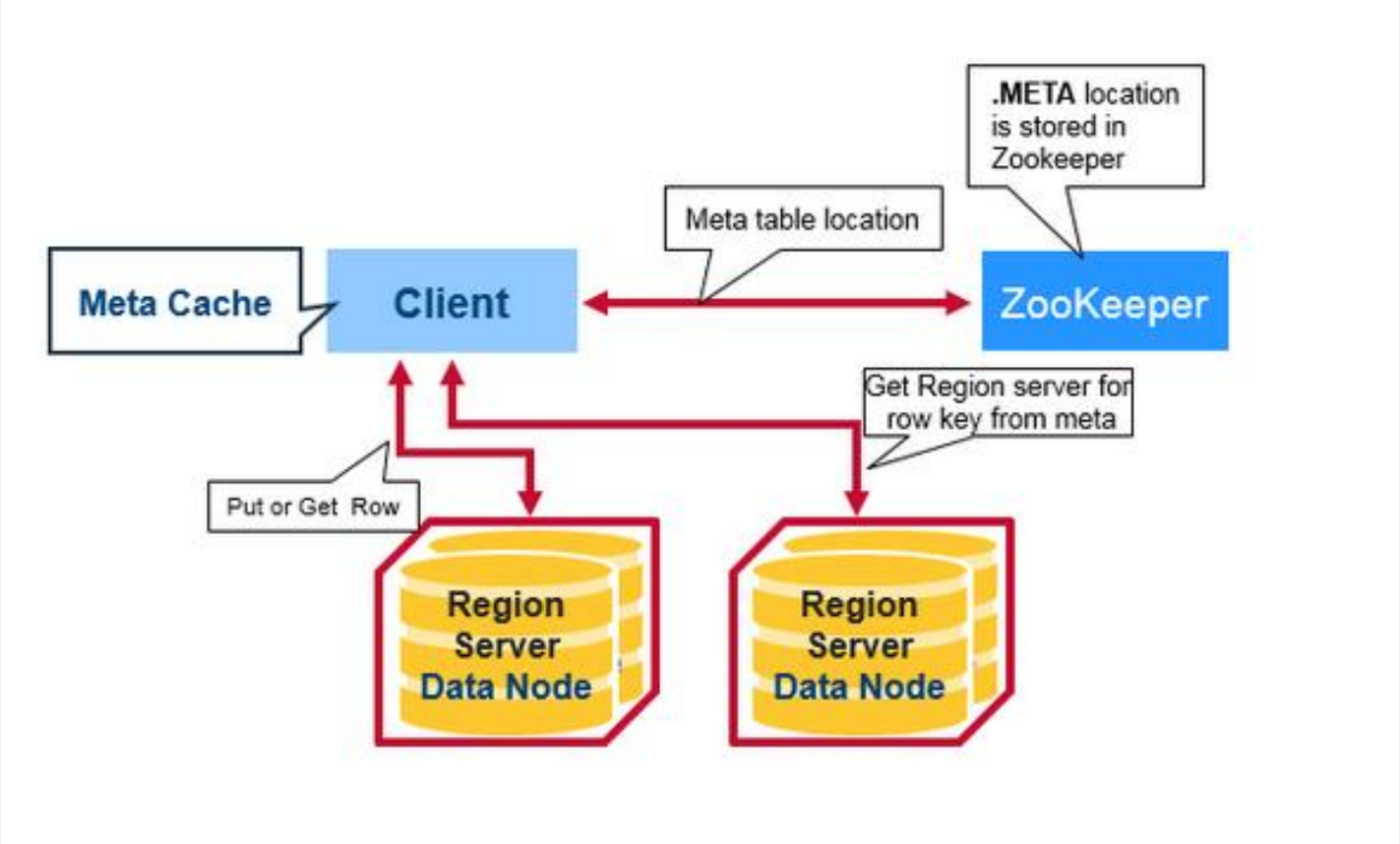
fppt.com

Region Server

- HBase Tables are divided horizontally by row key range into “Regions.”
  - Specify region’s start key and end key.
- Regions are assigned to the nodes in the cluster, called “Region Servers”.
- Region servers serve data for reads and writes.

fppt.com

HBase Read or Write



fppt.com

RegionServer hotspotting in Hbase

- Records in HBase are sorted lexicographically by the row key.
- If sequential row keys are used…
  - all writes hit one Region (uneven load distribution)
  - it limits the write throughput to the capacity of a single server instead of making use of multiple/all nodes in the HBase cluster

fppt.com

Data Manipulation

- Get
- Put

Get / Put: specific to particular rows and need the row key to be provided

- Scan
  - For a range of rows. The range could be defined by a start and stop row key or could be the entire table if no start and stop row keys are defined.

fppt.com

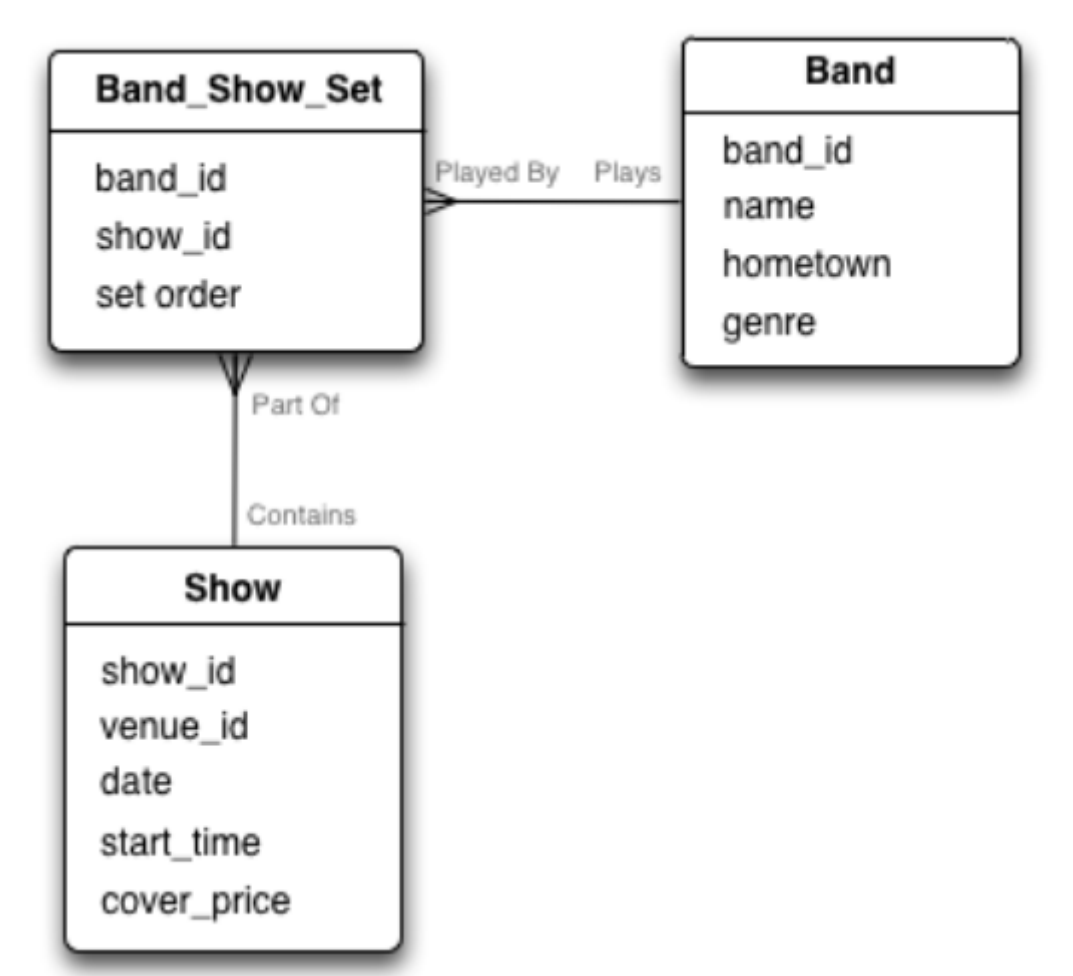
Relationships

- HBase has no foreign keys, or joins, or cross-table transactions.
- If you need relationships between entities – then **Denormalize the data**
  - That is, two logical entities share one physical representation.

fppt.com

Denormalization technique

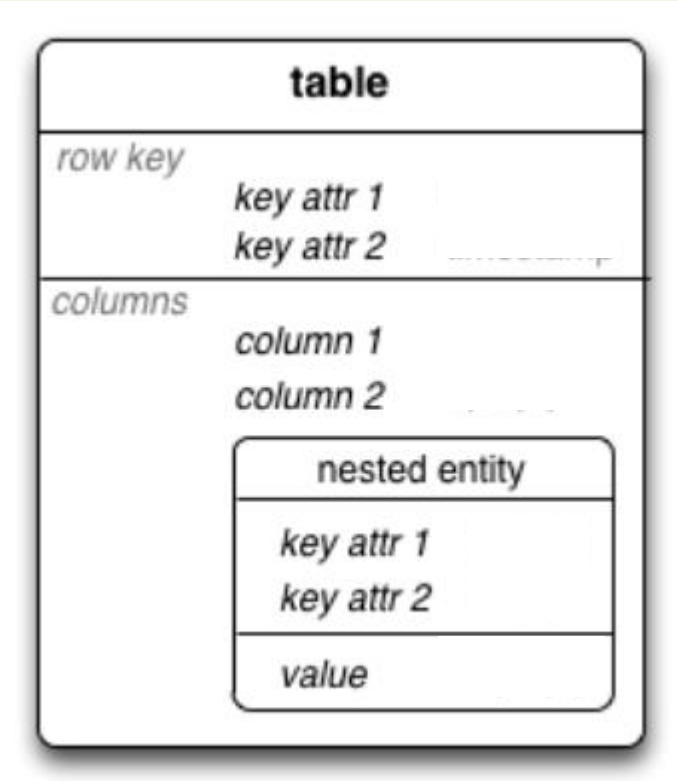
- Normalized



fppt.com

Denormalization technique

- Nested Entity



fppt.com

Graph Data Store

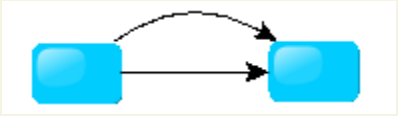
- What is a graph data store?
  - A database with an explicit graph structure
  - Each node knows its adjacent nodes
- Why are graphs important?
  - Modelling chemical and biological data
  - Social networks
  - The web
  - Hierarchical data

fppt.com

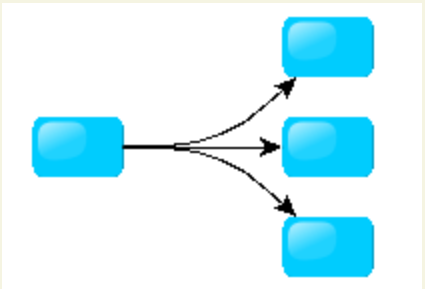
Different Kinds of Graphs

- Undirected Graph
- Directed Graph

- Pseudo Graph
- Multi Graph



- Hyper Graph



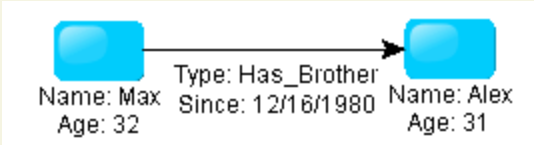
fppt.com

More Kinds of Graphs

- Weighted Graph

- Labeled Graph

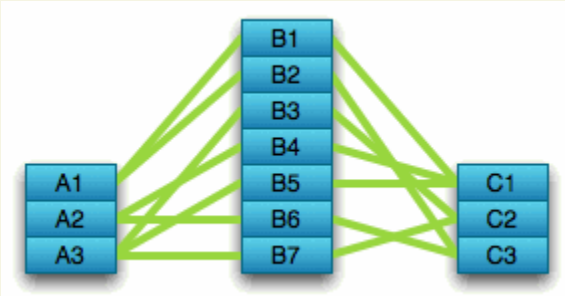
- Property Graph



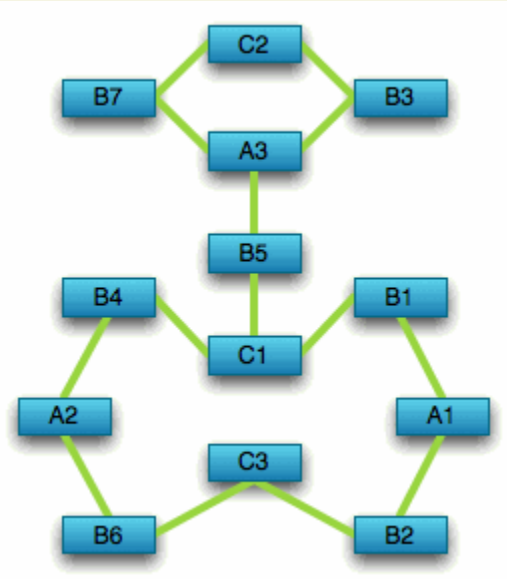
fppt.com

Compared to Relational Databases

Optimized for aggregation



Optimized for connections

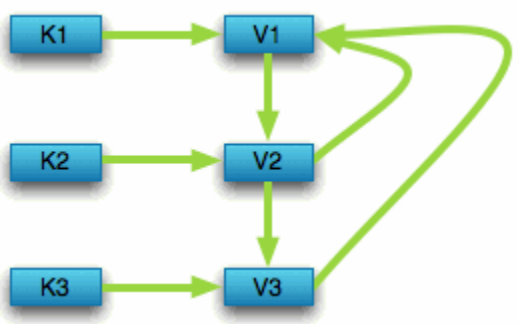


fppt.com

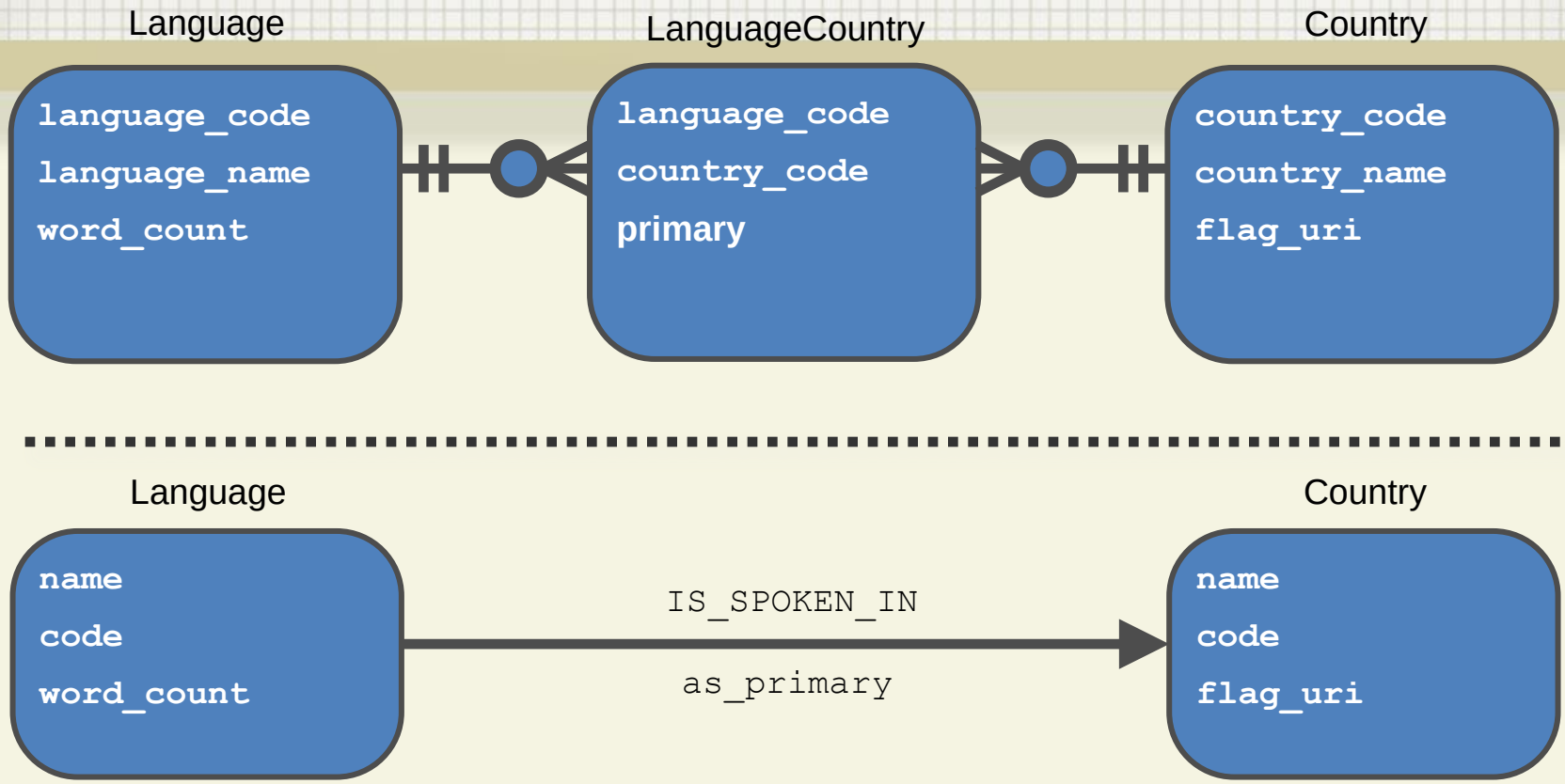
Compared to Key Value Stores

Optimized for simple look-ups

Optimized for traversing connected data

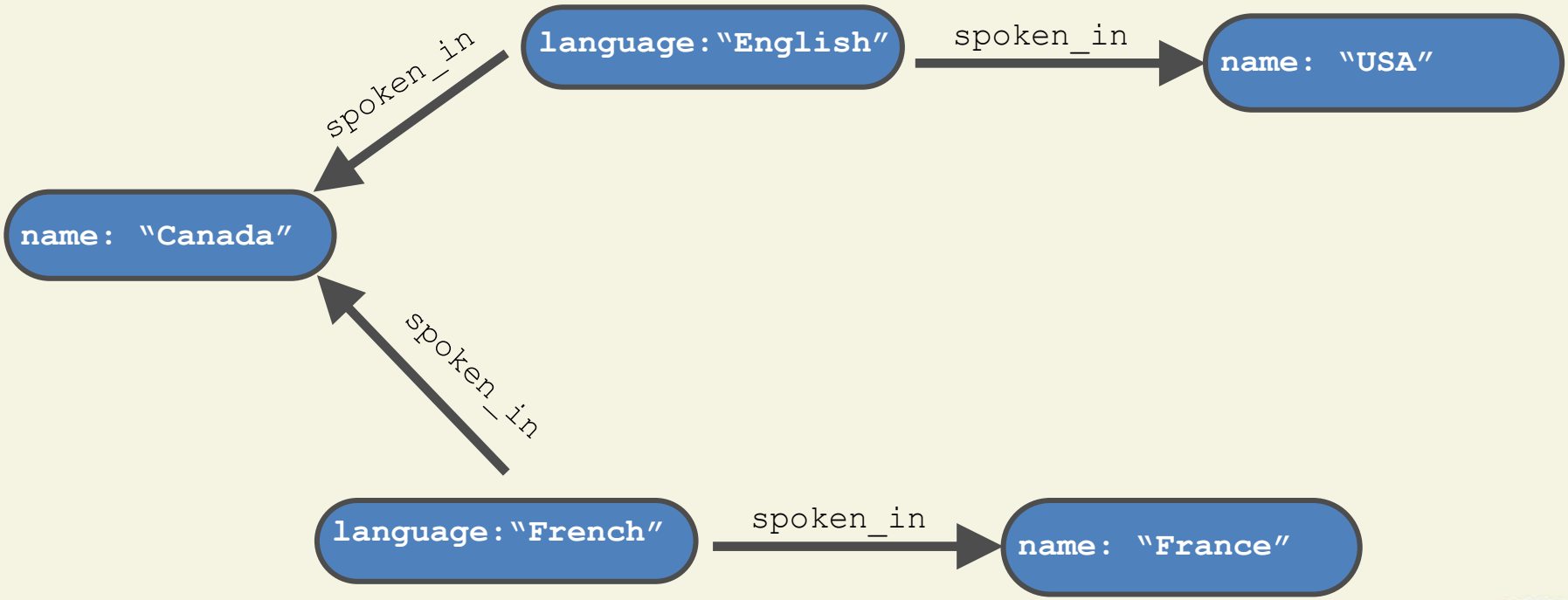


fppt.com


Language
language_code
language_name
word_count
LanguageCountry
language_code
country_code
primary
Country
country_code
country_name
flag_uri
Language
name
code
word_count
IS_SPOKEN_IN
as_primary
Country
name
code
flag_uri

fppt.com

name: "Canada"
languages_spoken: "['English', 'French']"



fppt.com

What is Neo4j?

- A Graph Database + Lucene Index
- Property Graph
  - Each node/edge is uniquely identified
  - Each node has a set of incoming and outgoing edges
  - Each node/edge has a collection of properties
  - Each edge has a label that defines the relationship between its 2 nodes
- Full ACID (atomicity, consistency, isolation, durability)
- High Availability (with Enterprise Edition)
- 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
- Embedded Server
- REST API

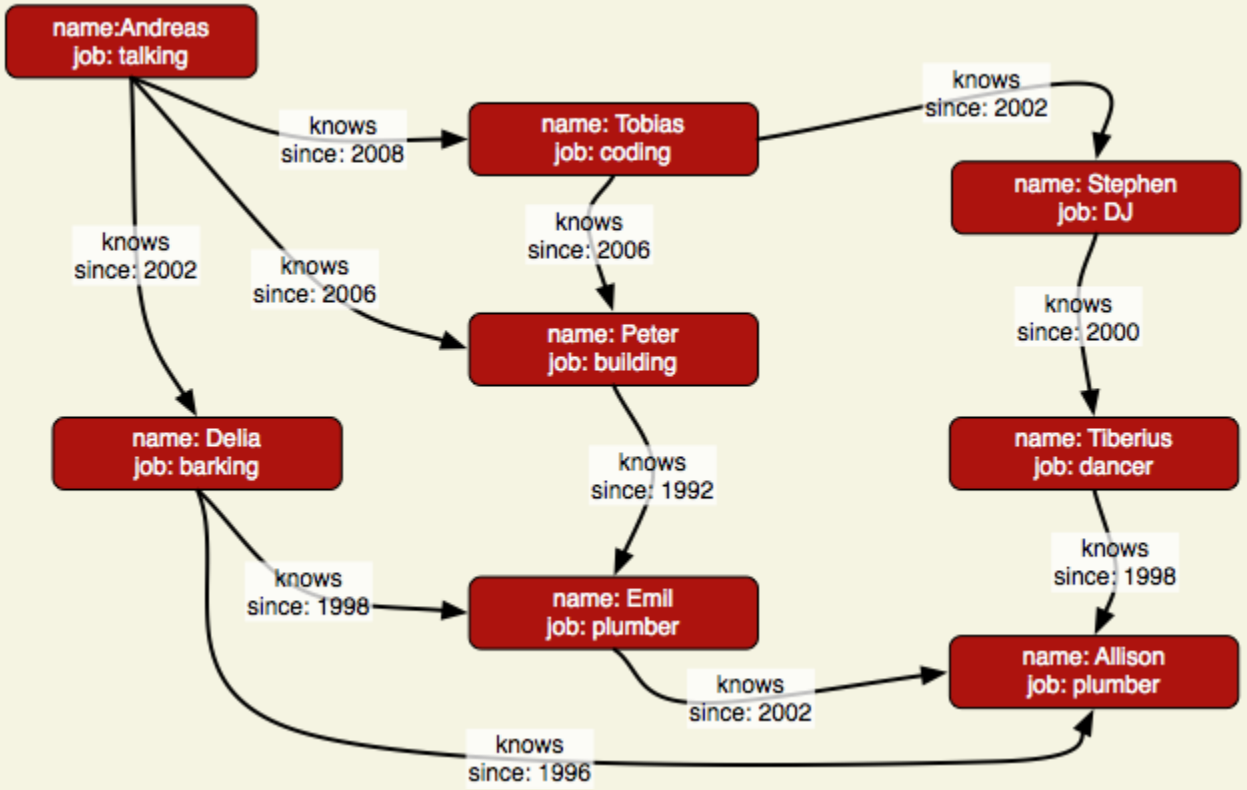
fppt.com

Neo4j - Properties

- *Both nodes and relationships can have properties.*
- Properties are named values where the name is a string.
- The supported property values are:
  - Numeric values
  - String values
  - Boolean values
  - Lists of any other type of value
- NULL is not a valid property value. Instead of storing it in the database NULL can be modelled by the absence of a key.

fppt.com

Property Graph



fppt.com

Neo4j - Labels

- *Labels assign roles or types to nodes.*
- Labels are used to group nodes into sets;
  - all nodes labelled with the same label belongs to the same set.
- A node may be labelled with any number of labels, including none, making labels an optional addition to the graph.

fppt.com

Neo4j - Index

- An index is a redundant copy of information in the database for the purpose of making retrieving said data more efficient.
- This comes at the cost of additional storage space and slower writes.
- Create indexes over a property for all nodes that have a given label.
- Once an index has been created, it will automatically be managed and kept up to date by the database whenever the graph is changed.
- Indexes in Neo4j are *eventually available*.
  - Cannot query until the index is populated. After it is brought online, can query the graph database.

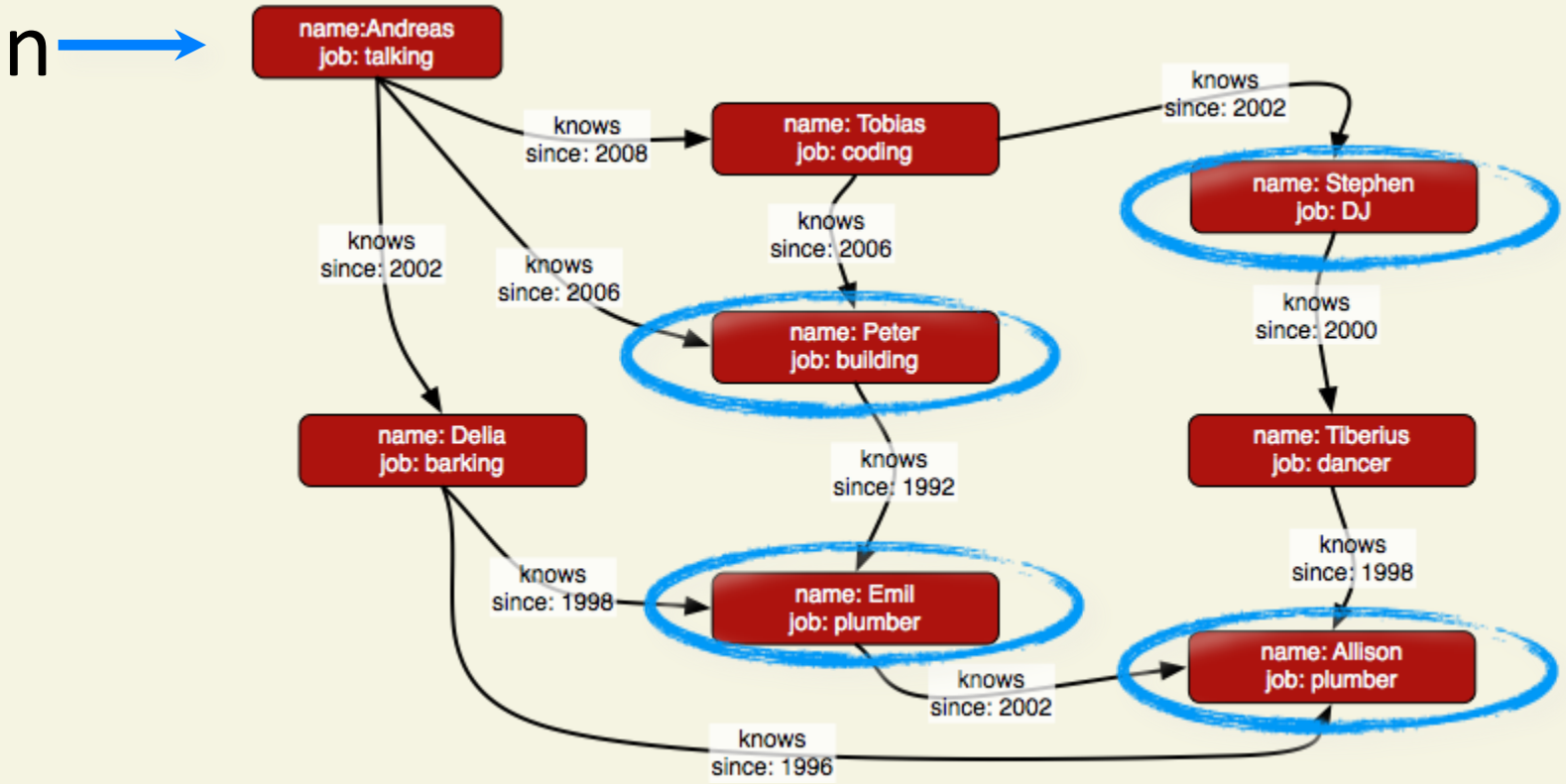
fppt.com

Neo4j -Traversal

- *A traversal navigates through a graph to find paths.*
- Navigates from starting nodes to related nodes, finding answers to questions like "what music do my friends like that I don’t yet own?“

fppt.com

```
// then traverse to find results
start n=(people-index, name, "Andreas")
match (n)--()--(foaf) return foaf
```



fppt.com

Cypher

Pattern Matching Query Language (like SQL for graphs)

```
// get node 0

start a=(0) return a

// traverse from node 1

start a=(1) match (a)-->(b) return b

// return friends of friends

start a=(1) match (a)--()--(c) return c
```

fppt.com

fppt.com