

NoSQL Databases

Davood Rafiei

Outline

- What?
- Why?
- NoSQL databases
 - Key-value stores
 - Wide column stores
 - Map reduce
 - more...

What?

- NoSQL != No SQL
- NoSQL an umbrella term for data stores that don't follow RDBMS principles
 - Data may not be relational
 - SQL may or may not be used
 - Schema may not be available
 - BASE consistency model (much loser than ACID)
- NoSQL ~ “not only SQL”

Why?

- We want our access to data to be
 - Convenient
 - Reliable
 - Scalable
 - Efficient

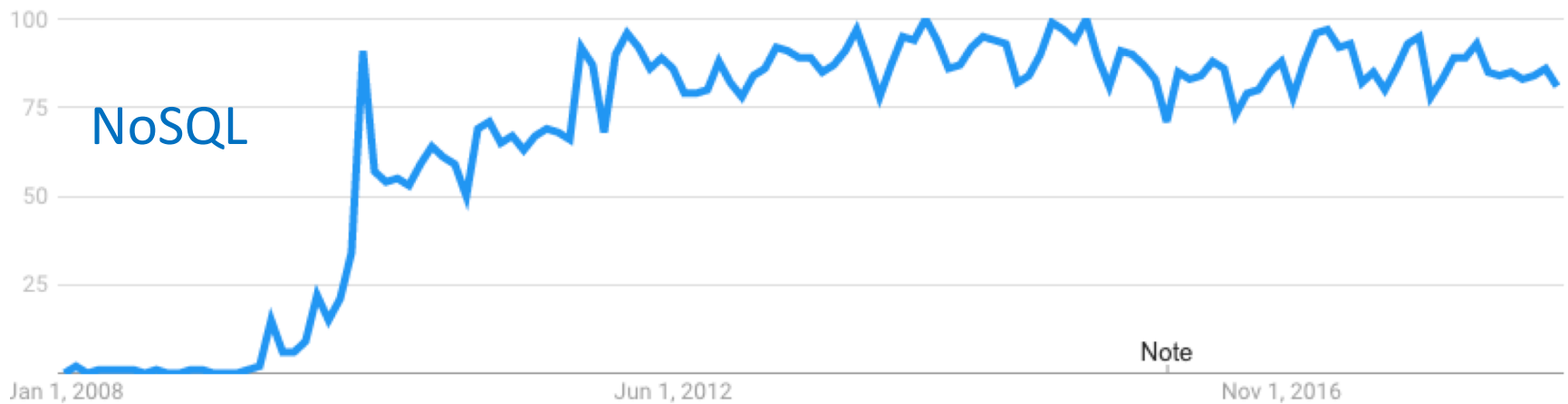
Why?

- We want our access to data to be
 - Convenient
 - Reliable
 - Scalable
 - Efficient
- Willing to sacrifice convenience and reliability for scalability and efficiency

Gaining in popularity ...

Google

Trends



... but still behind SQL



NoSQL Systems

- As an alternative to relational DBMS
 - More flexible schema
 - Cheaper to setup
 - Better scalability in some cases
 - Higher performance and availability (with a relaxed consistency)
- More programming (less declarative querying)
- Less consistency

Positive

Negative

ACID

- **A**tomicity
 - Transactions are atomic
- **C**onsistency
 - Database is always moved from one consistent state to another (through transactions)
 - Applications are insulated from dirty reads and writes
- **I**solation
 - Transactions can run concurrently
- **D**urability
 - Committed transactions cannot be lost (e.g. due to power failure)

BASE Consistency Model

- **B**asic availability
 - Database available most of the time
- **S**oft state
 - Stores and replicas may not be consistent
- **E**ventual consistency
 - Stores may not have the latest updates
 - common in distributed systems

Trade consistency for high availability

NoSQL Databases

- Key-value stores
- Wide column stores
- Document stores
- Graph data stores
- Map-reduce
- More...

Key-Value

- Data storage for a binary table (key, value)
 - Aka dictionary, hash or hashmap
- Examples
 - (url, content), (term, docids)
 - (stuid, courses_enrolled), (stuid, contact_Info)
- Relational to key-value mapping
 - (sin, <name, phone>)
 - (sin, name), (sin, phone)

Key-Value Stores

- Simple interface
 - Data model: (key, value) pairs
 - Operations: insert(key, value), update(key, value), fetch(key), delete(key)
- Efficient/scalable implementation
 - Records distributed to machines based on key
 - Fault tolerance is supported by replication
 - Replicas: eventually consistent
 - Single record transactions

Key-Value Stores

- Redis
 - A networked in-memory store
 - Supports lists, sets, hashes, ...
 - A set of operations to operate over the supported types
 - Client-server model
 - Open source and quite popular
 - Optional durability with a snapshot stored on disk

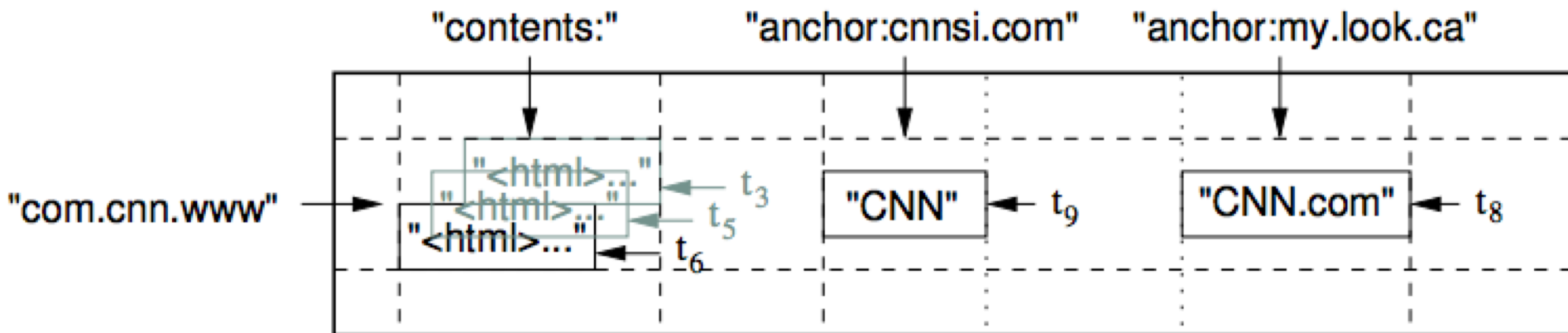
Key-Value Stores (Cont.)

- Berkeley DB
 - A disk-based store
 - Supports btree, hash, queue, recno (heap)
 - No network access (similar to SQLite)
 - Supports transactions
 - Covered in the lab (and used in Mini-project 2)

Wide Column Stores

- Data can have too many columns and those columns may change over time
 - E.g. Time series, streaming events
- Columns can change from one row to next
- A limited set of column families (e.g. hobby, jobs) but an infinite number of values in each family
- A timestmap may be assigned to each cell (at a row and a column)

A Use Case for Google Bigtable



Another Use Case

"follows" column family

	Follows			
Row Key	gwasington	jadams	tjefferson	wmckinley
gwasington		1		
jadams	1		1	
tjefferson	1	1		1
wmckinley			1	

Multiple versions

Content under CC by 3.0 license

Wide Column Stores

- Data stored as records
- Number and names of columns are not fixed
 - A record can have millions of columns
 - Columns change dynamically
- Seen as two-dimensional key-value stores
- Examples of column stores
 - Google's BigTable
 - Cassandra
 - Hbase
 - Microsoft Azure Cosmos DB

Document Store

- Consider storing the following json document in an RDBMS

```
{
  {title: 'This is a sample post',
    desc: 'document store sample',
    by: 'Joe221',
    url: 'www.a.com/p200.html',
    tags: ['davood', '291', 'database'],
    likes: 40}
,
  {title: 'CMPUT 291',
    desc: 'a course in databases',
    by: 'davood',
    url: 'www.a.com/p202.html',
    tags: ['nosql', 'database', '291', 'example'],
    likes: 38,
    comments: [ {user: 'bob10',
                  msg: 'great post',
                  post_date: 'Oct 19, 2017',
                  like: 2}]]}
}
```

RDBMS Tables

- Posts(post_id, title, desc, by, url, likes)
- Tags1(tid, post_id, tag)
- Comments(cid, post_id, by, msg, post_date, likes)

Multiple insert statements per document

A document Store

- Insert into store *db* under collection *posts*
 - `Db.posts.insert([{"title:"}])`
- Find and print it
 - `Db.posts.find({"by": "davood"}).pretty()`
 - `Db.posts.find({"likes": {$gt:20}}).pretty()`
- Document encoding: json, xml, etc.
- Each document has a (implicit/explicit) key
 - Unlike key-value stores, content can be searched

Document Stores

- ElasticSearch
 - An infrastructure on top of Apache's Lucene
 - Stores json in a distributed fashion
 - Scales well across machines and data centers
 - Support searches over documents
 - Each document is stored under a url
 - <https://localhost:9200/index-name/type-name/id>

ElasticSearch

CRUD operations

```
curl -XPUT "http://localhost:9200/movies/movie/1" -d'
```

```
{  
  "title": "It is a wonderful life",  
  "director": "Frank Capra",  
  "year": 1946,  
  "genre": ["drama","family"]  
}
```

Store in index movies

```
curl -XGET "http://localhost:9200/movies/movie/1" -d'
```

```
curl -XPOST "http://localhost:9200/_search" -d'
```

```
{  
  "query": {  
    "query_string":{"query":"life"}  
  }  
}
```

Access by id

Free text search

```
curl -XPOST "http://localhost:9200/_search" -d'  
{
```

```
  "query": {  
    "query_string":{"query":"life", "fields":["title"]}  
  }  
}
```

Search in title

```
curl -CDELETE "http://localhost:9200/movies/movie/1" -d''
```

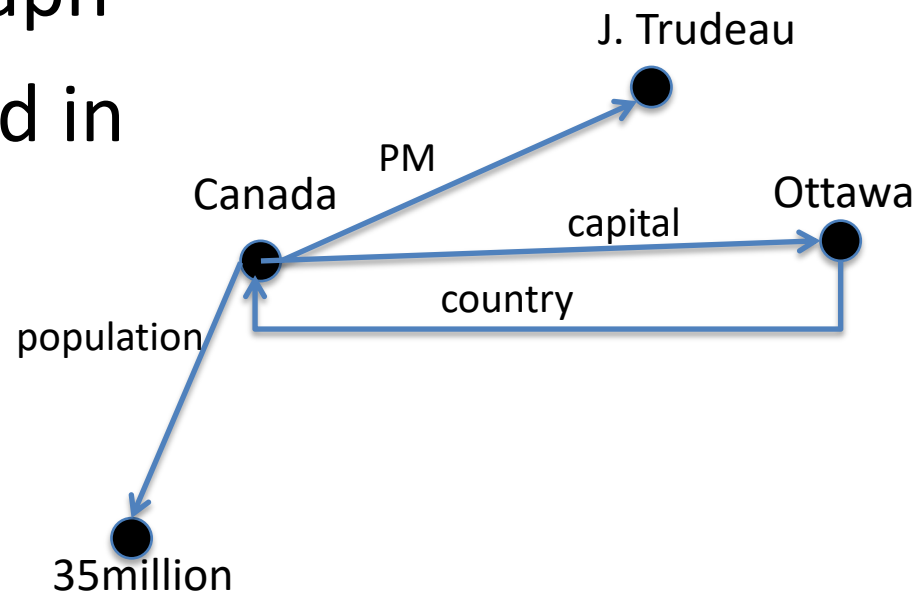
Delete it

Other Document Stores

- MongoDB
- CouchDB
- Azure DocumentDB
- Many more...

Graph Databases

- Let nodes represent entities and edges describe relationships.
- e.g. knowledge graph
- Data may be stored in a Rel. DB



Triple Stores

- Set of triples (subject, predicate, object)
 - E.g. (Canada, capital, Ottawa),
(Canada, prime_minister, J_Trudeau)
- Queried using graph patterns

SPARQL

- Datasets
 - Dbpedia

```
<http://dbpedia.org/resource/Camrose,_Alberta> <http://xmlns.com/foaf/0.1/name> "City of Camrose" .  
<http://dbpedia.org/resource/Banff,_Alberta> <http://xmlns.com/foaf/0.1/name> "Town of Banff" .  
<http://dbpedia.org/resource/Camrose,_Alberta> <http://dbpedia.org/ontology/populationTotal> "19742" .  
<http://dbpedia.org/resource/James_Gosling> <http://dbpedia.org/ontology/birthPlace> <http://dbpedia.org/resource/Alberta> .  
<http://dbpedia.org/resource/Java_programming_language> <http://dbpedia.org/ontology/developer> <http://dbpedia.org/resource/James_Gosling> .  
<http://dbpedia.org/resource/James_Gosling> <http://dbpedia.org/ontology/award> <http://dbpedia.org/resource/Order_of_Canada> .  
<http://dbpedia.org/resource/James_Gosling> <http://dbpedia.org/ontology/employer> <http://dbpedia.org/resource/Google> .
```

– Our own data

@prefix foaf: <http://xmlns.com/foaf/0.1> .

@prefix uofa: <http://www.ualberta.ca/ontology> .

```
<http://www.ualberta.ca/people/davood> foaf:name "Davood Rafiei" .
```

```
<http://www.ualberta.ca/courses/291> foaf:name "File and Data Management" .
```

```
<http://www.ualberta.ca/people/davood> uofa:teach <http://www.ualberta.ca/courses/291> .
```

SPARQL

- Queries

```
@prefix foaf: <http://xmlns.com/foaf/0.1> .  
@prefix uofa: <http://www.ualberta.ca/ontology> .  
@prefix dbr: <http://dbpedia.org/resource> .  
@prefix dbo: <http://dbpedia.org/ontology> .
```

```
SELECT ?name  
WHERE {  
    ?person foaf:name ?name .  
}
```

```
SELECT ?person  
WHERE {  
    dbr:Java_programming_language dbo:developer ?person .  
}
```

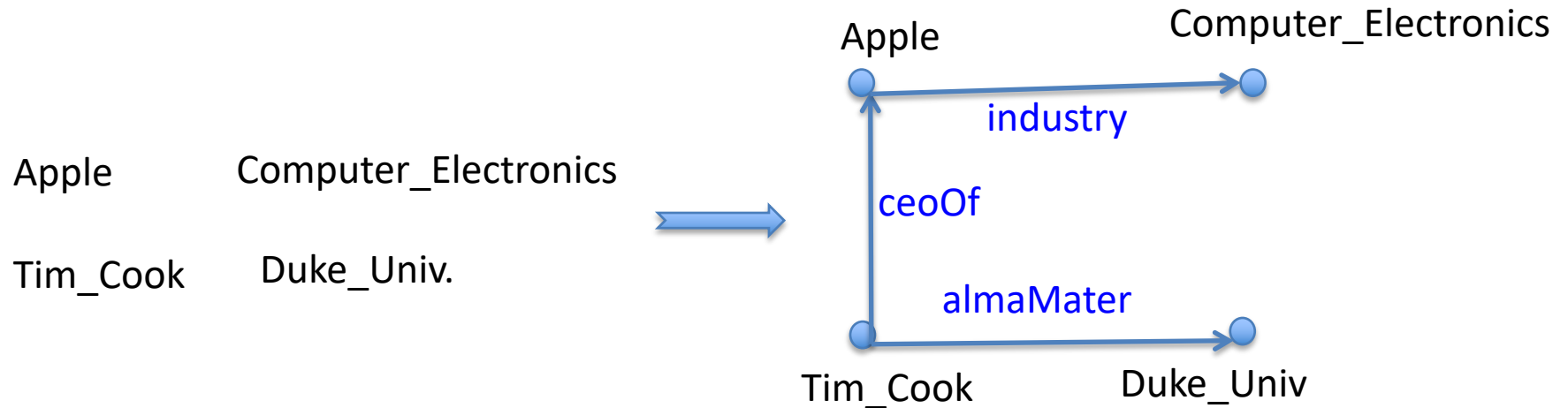
SPARQL

- Queries

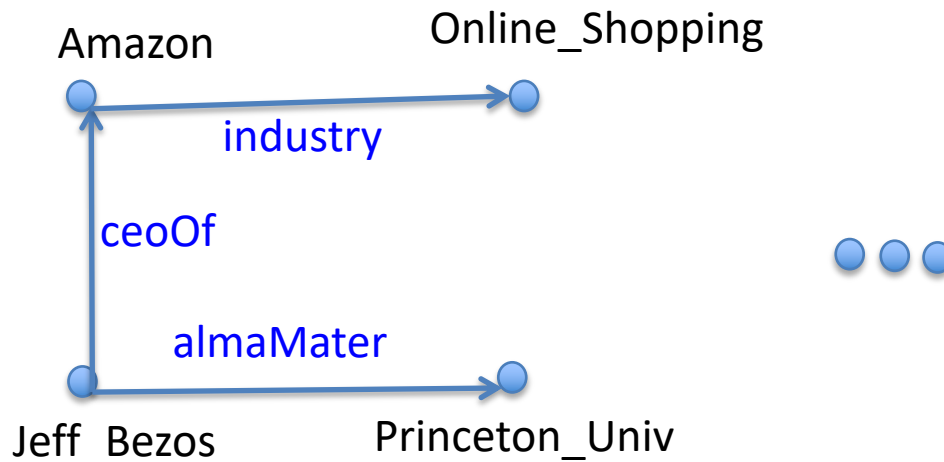
```
@prefix foaf: <http://xmlns.com/foaf/0.1> .  
@prefix uofa: <http://www.ualberta.ca/ontology> .  
@prefix dbr: <http://dbpedia.org/resource> .  
@prefix dbo: <http://dbpedia.org/ontology> .
```

```
SELECT ?invention  
WHERE {  
    ?invention dbo:developer ?person .  
    ?person dbo:birthPlace <http://dbpedia.org/resource/Alberta> .  
}
```

Query By Example



Output:



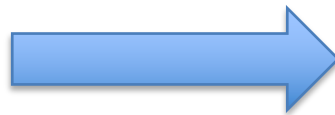
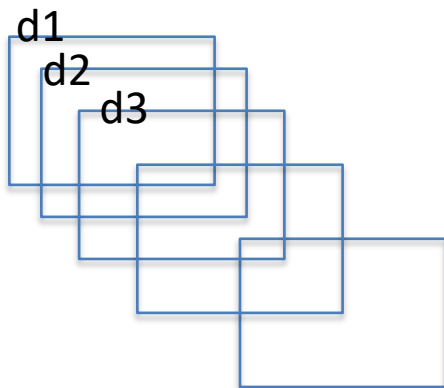
Map Reduce

- Developed at Google in 2004
- Problem:
 - Data files are spread over 1000's of machines
 - How to collect information quickly?
- Parallelism
 - Take advantage of the data spread

Hadoop: open source implementation of MapReduce

Map Reduce

- Large scale data processing over a network of machines
 - More machines leads to more parallelism and less response time
- Data stored in files
- E.g. computing word counts

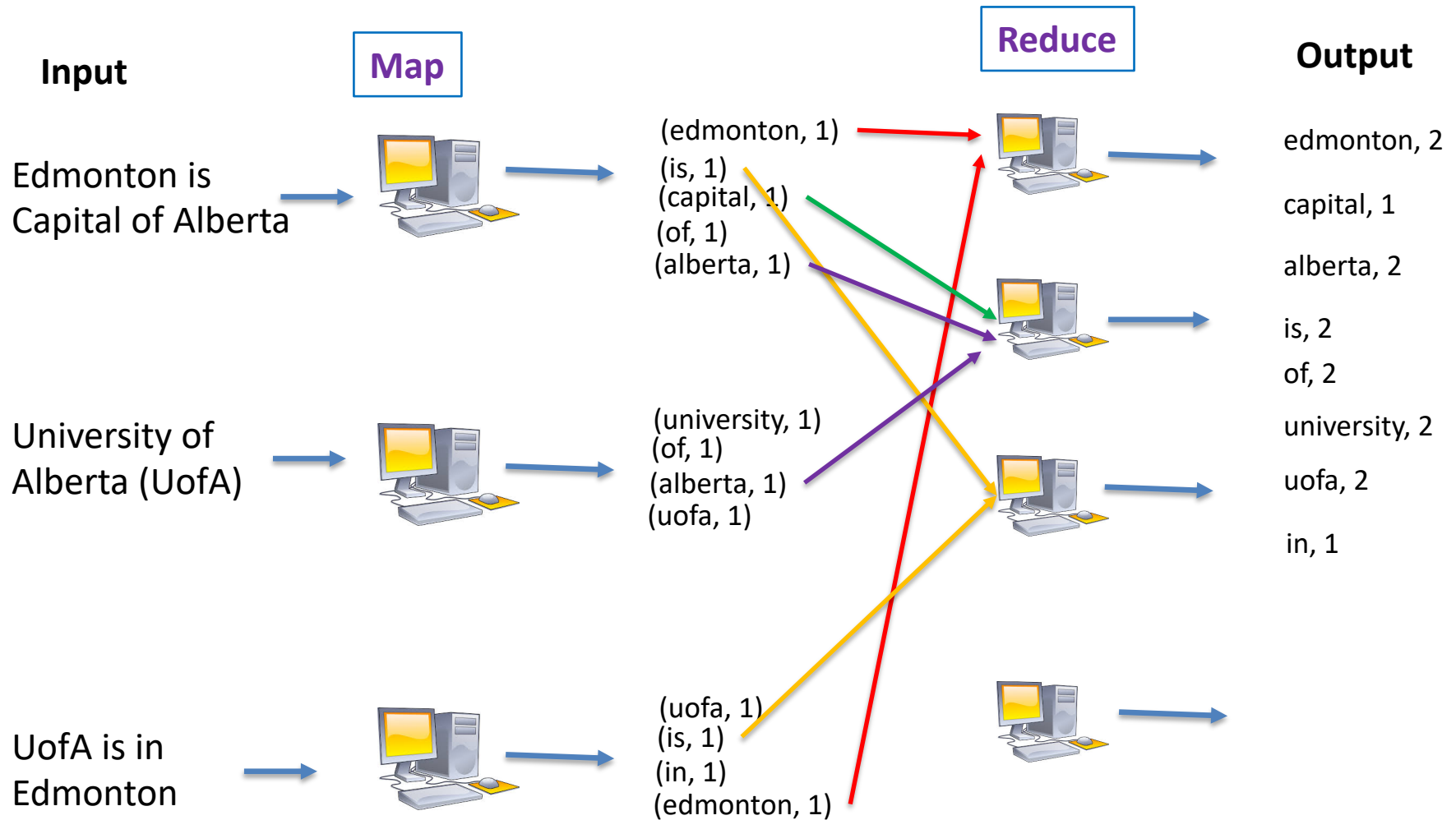


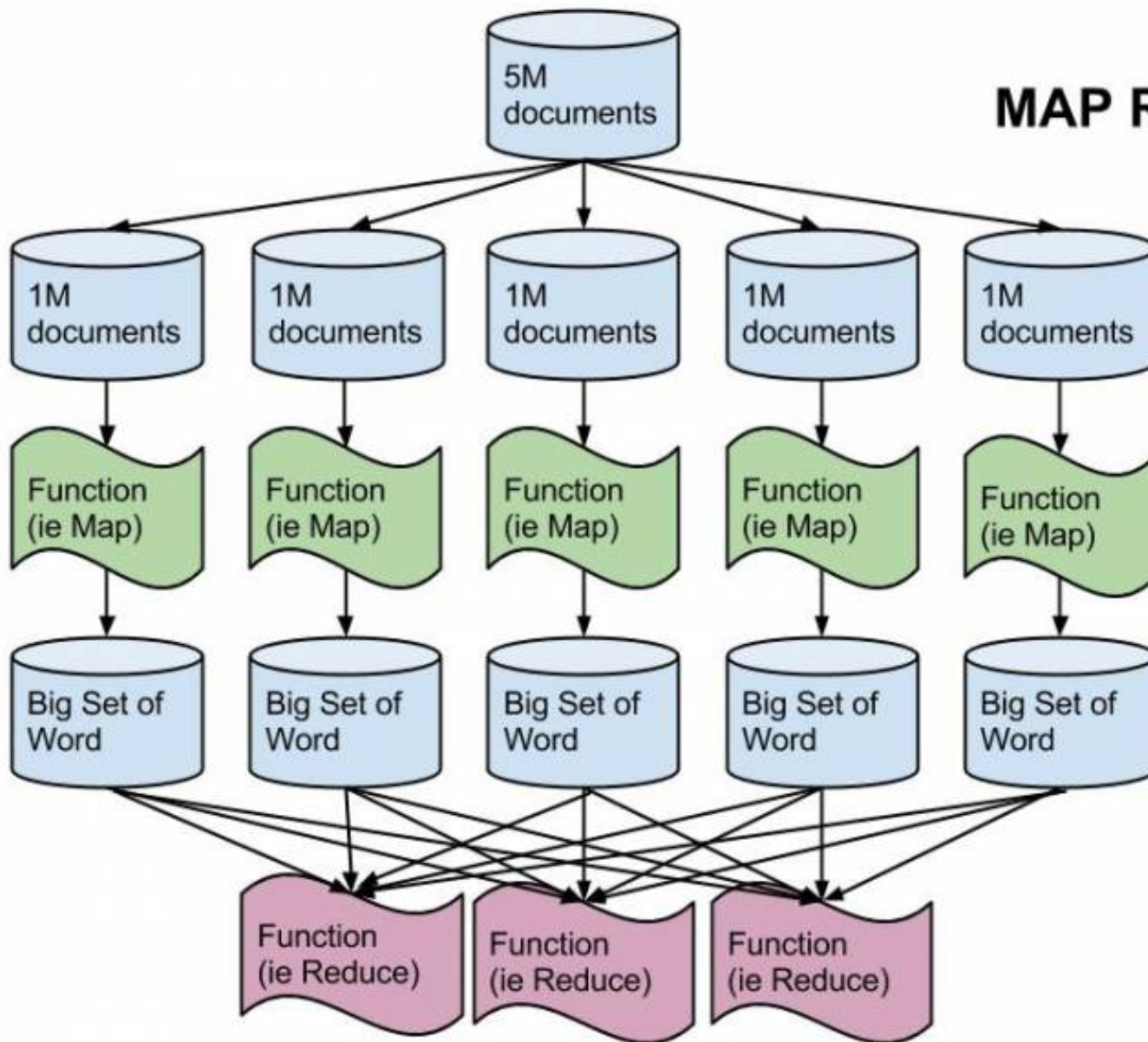
(term, number_of_docs)
(database, 1250)
(relational, 600)
...

Two Functions

- Map: divide the problem into smaller problems
 - $\text{Map}(d1) \rightarrow (\text{key}, \text{value})$ pairs
- Reduce: work on sub-problems and combine the results
 - $\text{redce}(\text{key}, \text{set of values}) \rightarrow (\text{key}, \text{summary})$

Word Counts Example





MAP REDUCE

Distribute the documents across N computers

For each document, return a set of (word, frequency) pair

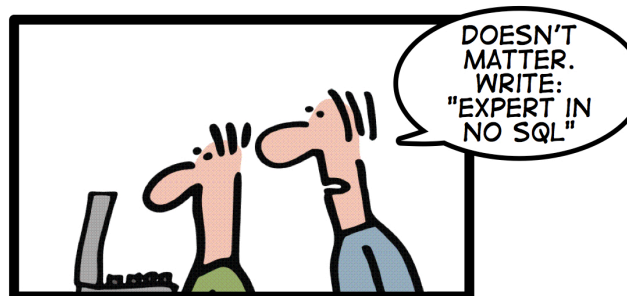
We get a big distributed list of sets of words frequency

Each Reduce function count the occurrences of one word.

Summary

- *Three 291 students walked into a noSQL bar. They quickly walked out because they could not find a table.*
- The term noSQL started being used in 2009
- Desirable properties
 - Network based databases,
 - Schema free access
 - Batch (and scalable) processing

HOW TO WRITE A CV



Leverage the NoSQL boom