COMP9311:
Database Systems

Term 3 2022

Week 10 (Non-Relational Database Systems)

By Xiaoyang Wang, CSE UNSW

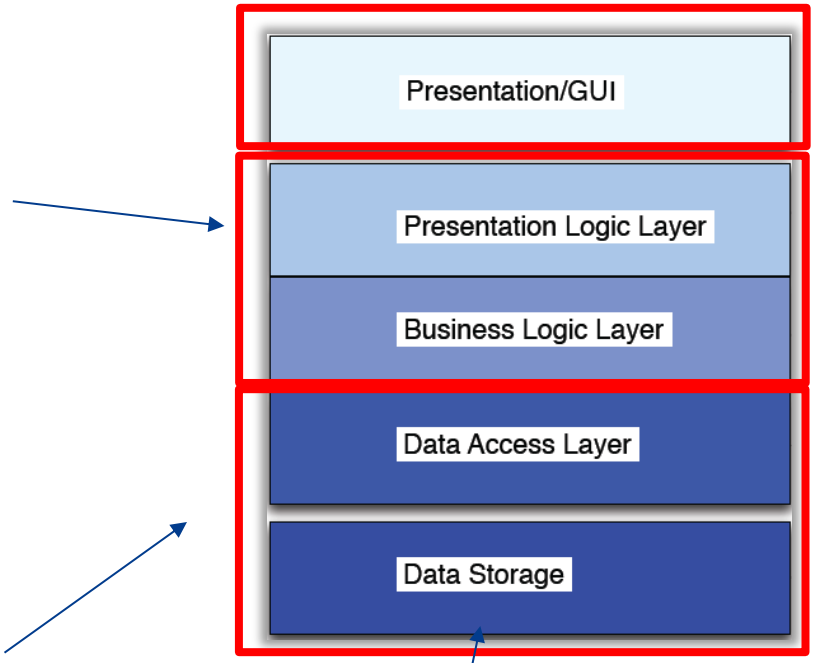
Textbook: Chapters 24 and 25

Disclaimer: the course materials are sourced from

- previous offerings of COMP9311 and COMP3311
- Prof. Werner Nutt on Introduction to Database Systems (http://www.inf.unibz.it/~nutt/Teaching/IDBs1011/)

What is in a data model …

An application developer "thinks" in terms of the real world "things" (people, organisations, actions, goods, etc.) … and model it as objects/data structures



When you want to store the objects, you express them in generic-purpose data model such as relational model …

Data access layer = ***translations*** *between the application model to the data model of the database …. Only with this translation* the objects to be queried, searched or manipulated (e.g., think of Python objects of students that are retrieved from a database)

The need to connect the application objects to your chosen database's data model

Database management system … like PostgreSQL - (based on relational model). This layer decides how the data models are physically stored in memory and disk

What models other than relational models, and why ??

We are Generating Vast Amount of Data …

Air Bus A380:

- Each engine generate 10 TB every 30 min

Twitter:

http://www.internetlivestats.com/twitter-statistics/

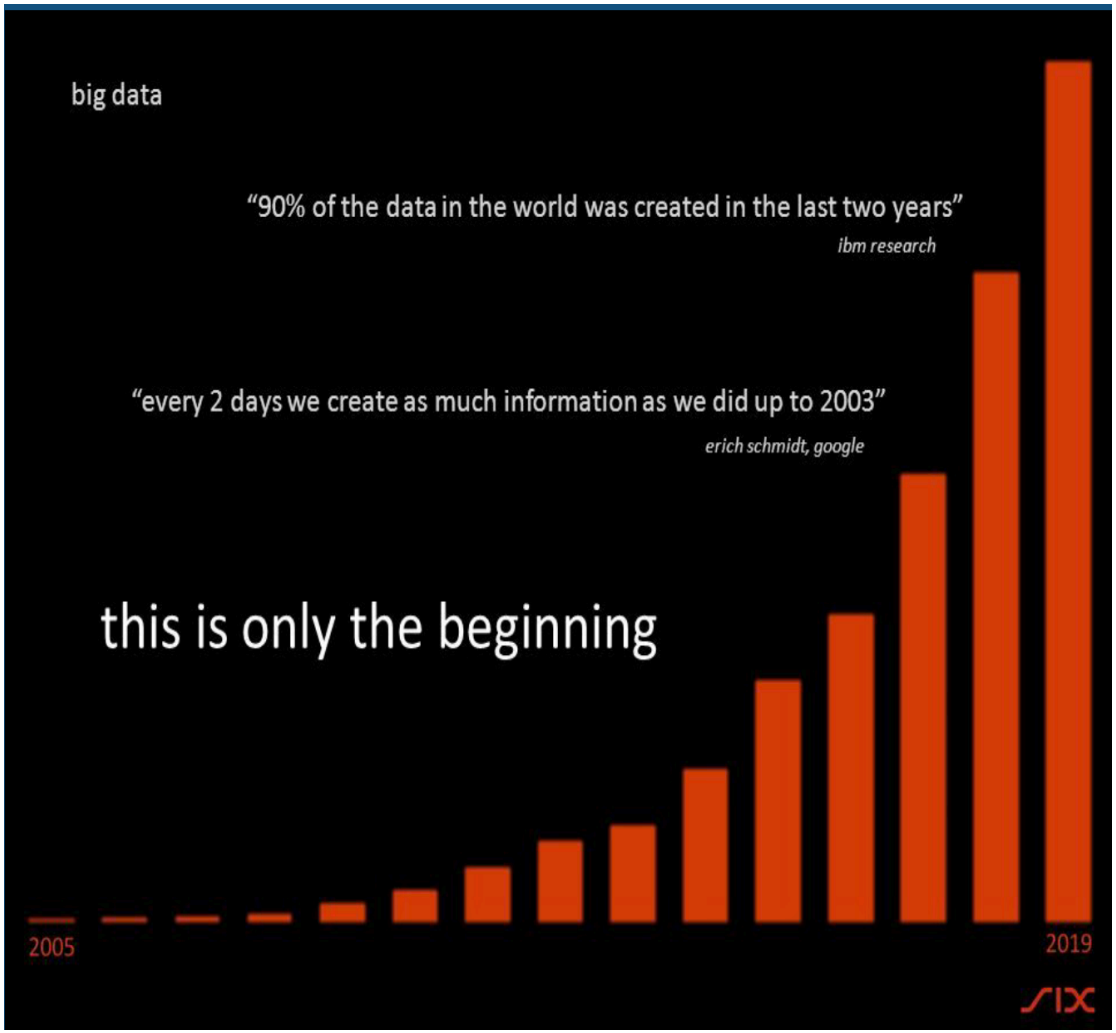
- Generate approximately 12 TB of data per day.

Facebook:

- Facebook data grows by over 500 TB daily.

New York Stock:

- Exchange 1TB of data everyday.

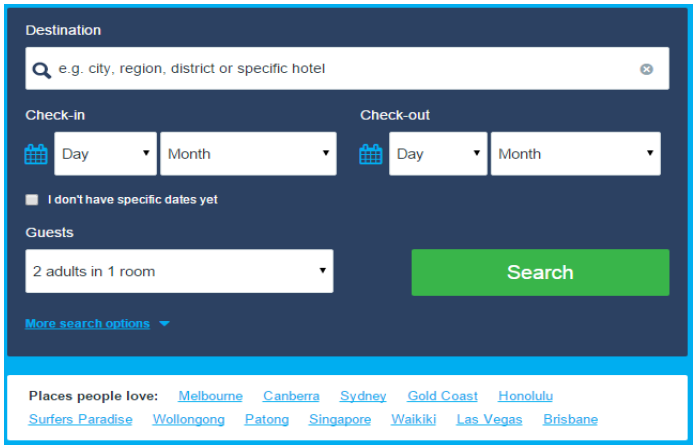


https://www.brandwatch.com/2016/03/96-amazing-social-media-statistics-and-facts-for-2016/

Different applications, different data requirements

E-Commerce website

- Data operations are mainly transactions (buying, paying, etc.)
  - Read/write
- Response time should be quick and important to maintain **reliability/integrity** of the *transactions – database wide.*
- i.e., ACID properties are important



Different applications, different data requirements

Image serving website (or many social network types sites in general)

- Data operations are mainly fetching information (posts)
  - Ready heavy.
  - High bandwidth requirement (fast loading)
- Getting up-to-date read or consistent data at all time is less critical …
  - ACID requirements to achieve *strict* serialisability can be relaxed in favour of allowing more transactions to access data

Different applications, different data requirements

Fan-out effect

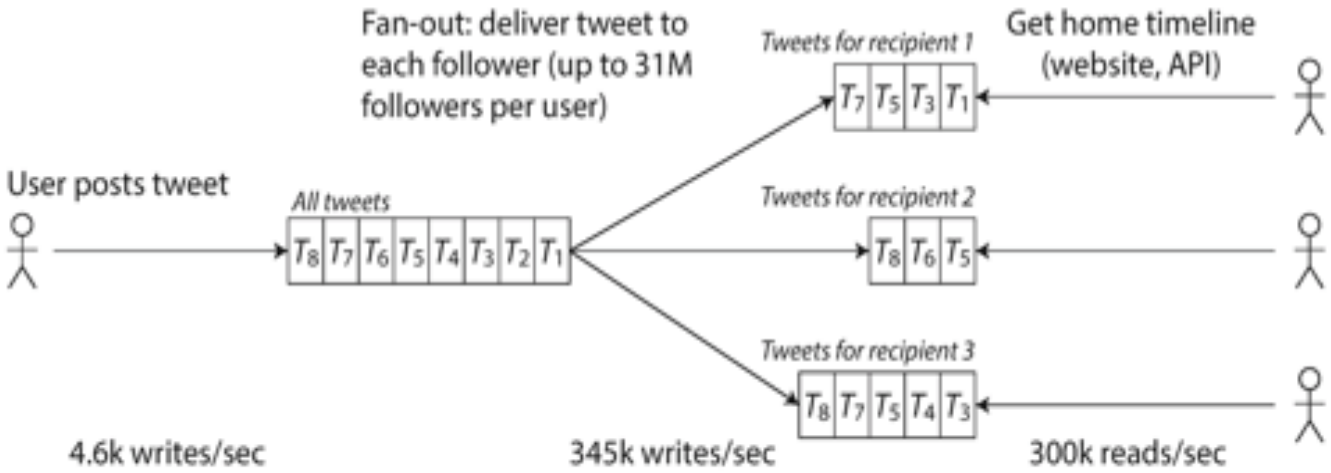


Figure 1-3. Twitter's data pipeline for delivering tweets to followers, with load parameters as of November 2012 [16].

A user can see tweets posted by the people they follow …

- also creates a lot of 'writing' work … On average 75 followers, but can vary widely (some users have 30 million followers), a lot of 'reading' work generated from a proportionally smaller number of 'posts'
- A new post –> look up the followers and 'write' to each follower's timeline ahead of time -> makes reading easy (note: write/read do not have to be "synchronised")

Relational Model vs. “NoSQL” Models

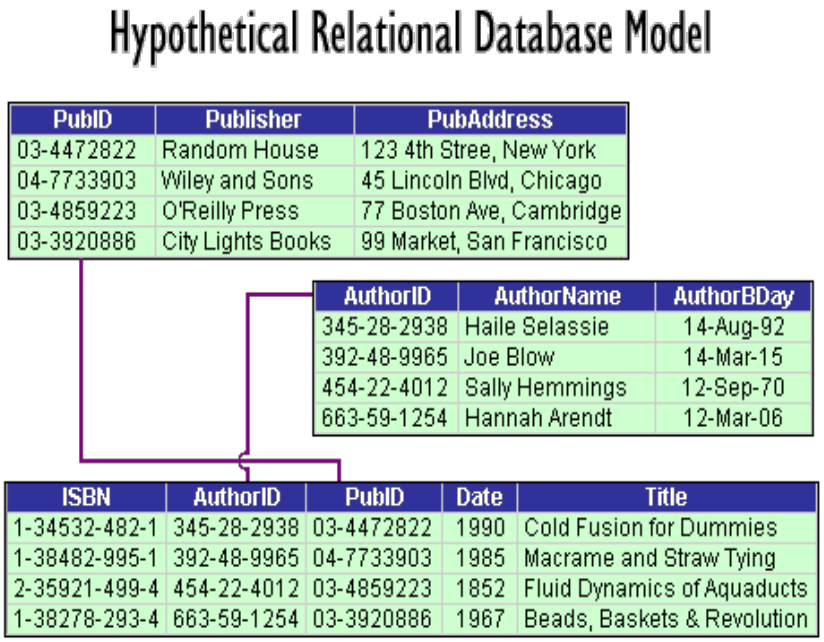
Relational Model and RDBMS (more or so synonymous with SQL)

- The best known, probably the most successful data model which has proven itself in many aspects to be the data model of choice in many applications
- Data is organised into relations (table) and relationships + constraints

Based on solid theory and well engineered implementation -> many competing models have been proposed, but never managed to take over SQL

Built for business data processing

- Typical business transactions (airline reservations, stock keeping, etc.)
- Also generically effective in many modern Web applications as well



But new types of applications → different data requirements → new types of data model and new database systems.

Image: http://gnosis.cx/publish/programming/xml_matters_8.html

Relational Model vs. “NoSQL” Models

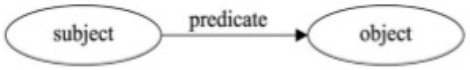
The rise of NoSQL … (since 2010 or so)

- NoSQL = Not Only SQL models
- Refers to a host of technologies that implement distributed, “non-relational” databases

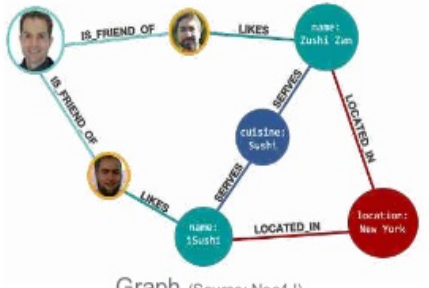
Why NoSQL?

- A need for greater scalability – very large datasets or very high read and write throughput
- A need for more expressive and flexible data model (→ *less formal*)
  - Usually do not require a fixed table schema nor do they use the concept of joins
  - All NoSQL offerings relax one or more of the ACID properties

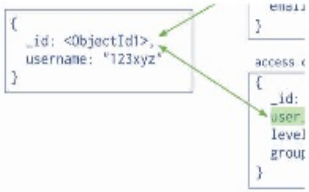
NoSQL Data Models



Source: W3C (2004)
RDF/Triple Store



Graph (Source: Neo4J)



Document Store (Source: MongoDB)

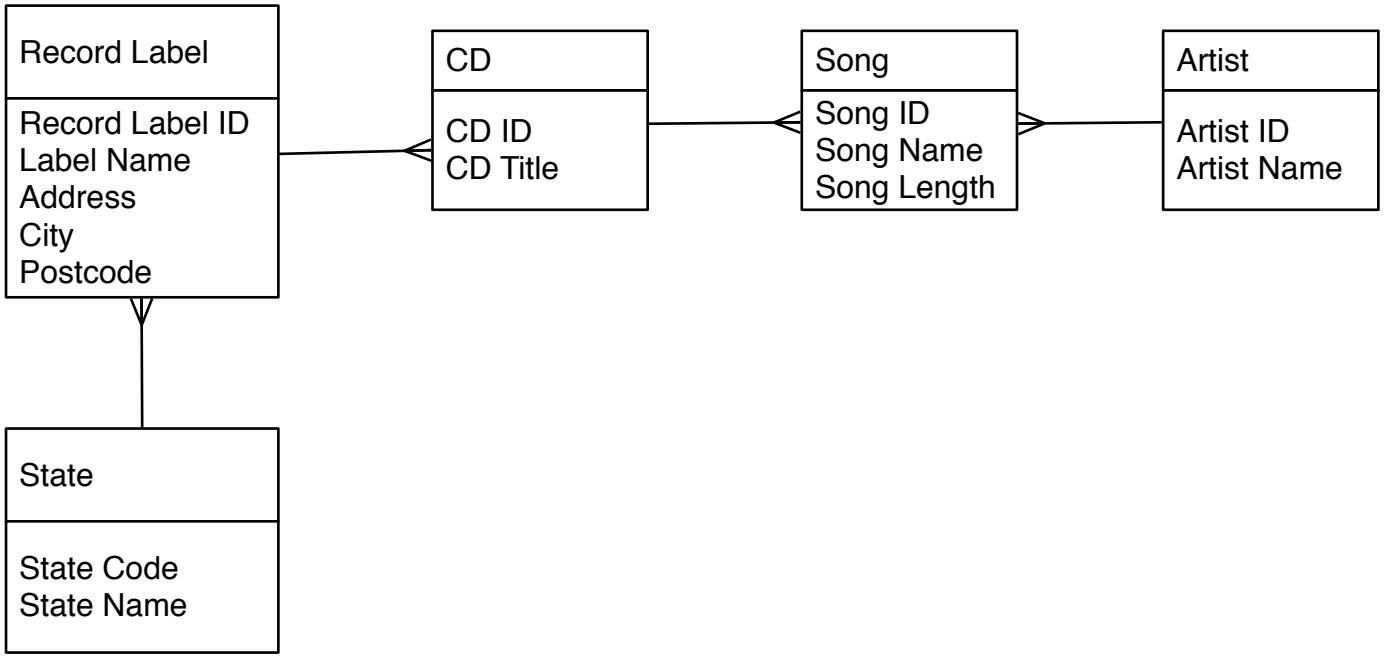
Column Store (Source: Toadworld)

data blueprint UNLOCKING BUSINESS VALUE

Designing Data-Intensive Applications, by Martin Kleppmann

Image: https://www.slideshare.net/Dataversity/trends-in-data-modeling

One of the problems with Relational Models



Normalisation … 3NF

* many fragments -> leading to many joins -> scalability for BIG data applications?

e.g., Key Value Store (a NoSQL data model)

Friendship of Facebook

as a Relational Database:

- relations and FKs
- `People (Pid, Name, ...)`
- `Friend (Pid1, Pid2, Date)`

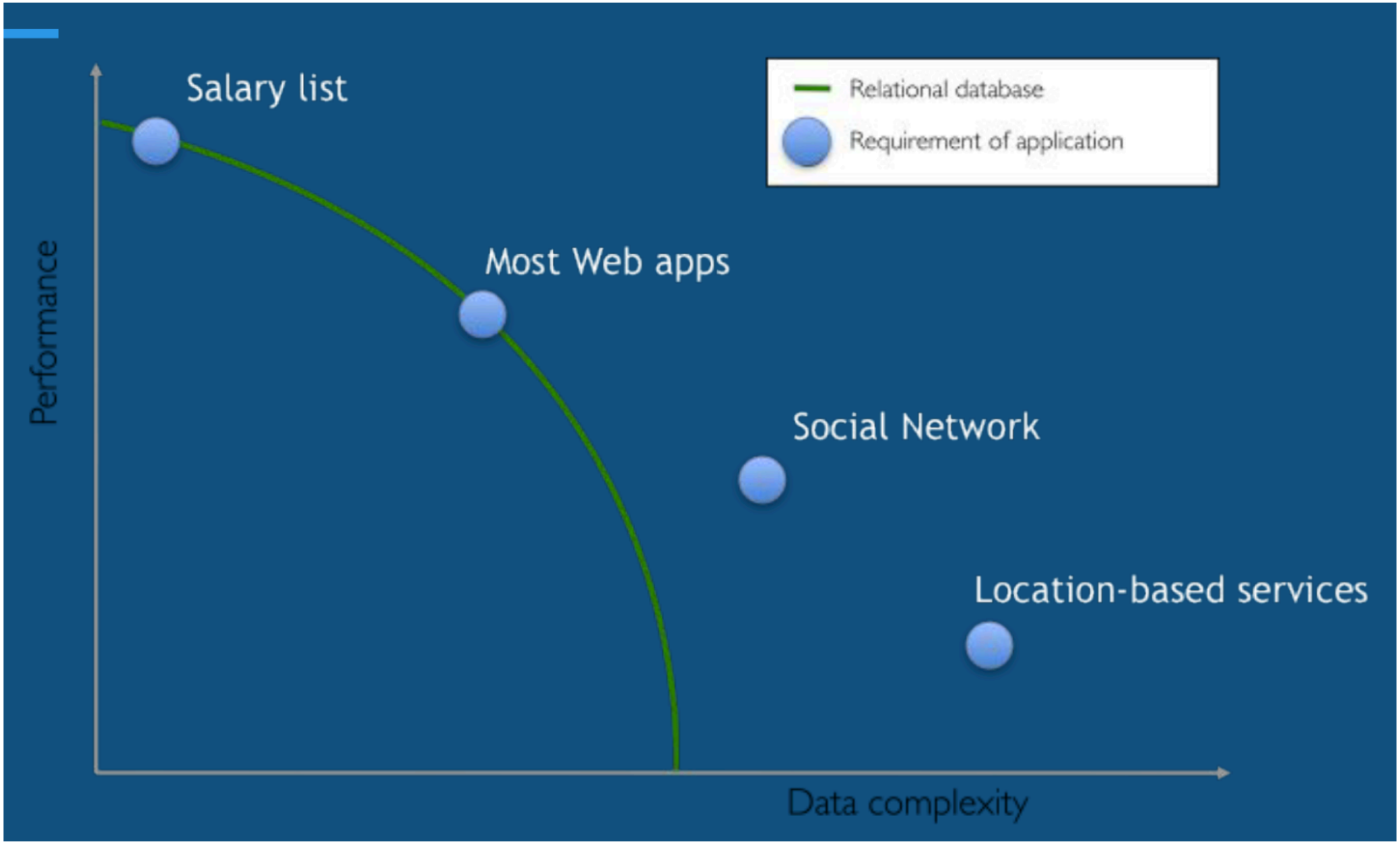
as a Key-Value Store:

- Key-Value pairs
- `A Person KV: <Id; Name>`
- `Friendship KV: <Id; All_Friends>`
  - `E.g., <0001; {(Id1, Date1), (Id2, Date2), ...}>`

When we want to get all friends

- Relational Database: Join People with Friend (costly)
- Key-Value Store: Get directly from the Friendship KV of a given `Person Id` (i.e., one read).

RDBMS Performance



Designing Data-Intensive Applications, by Martin Kleppmann

Image: https://www.slideshare.net/Dataversity/trends-in-data-modeling

Alternative Data Models?

Relational Modelling
of a resume (e.g., LinkedIn Profile)

Typical normalised form
would put multi-values in
separate tables with user_id
as foreign key … also uses look-up
tables (e.g., regions, industry)

Fragmented tables -> join

users table

user_id	first_name	last_name	summary
251	Bill	Gates	Co-chair of … blogger.

region_id	industry_id	photo_id
us:91	131	57817532

regions table

id	region_name
us:7	Greater Boston Area
us:91	Greater Seattle Area

industries table

id	industry_name
43	Financial Services
48	Construction
131	Philanthropy

positions table

id	user_id	job_title	organization
458	251	Co-chair	Bill & Melinda Gates F…
457	251	Co-founder, Chairman	Microsoft

education table

id	user_id	school_name	start	end
807	251	Harvard University	1973	1975
806	251	Lakeside School, Seattle	NULL	NULL

contact_info table

id	user_id	type	url
155	251	blog	http://thegatesnotes.com
156	251	twitter	http://twitter.com/BillGates

Alternative Data Models?

Example 2-1. Representing a LinkedIn profile as a JSON document

```
{
  "user_id":     251,
  "first_name":  "Bill",
  "last_name":   "Gates",
  "summary":     "Co-chair of the Bill & Melinda Gates...
Active blogger.",
  "region_id":   "us:91",
  "industry_id": 131,
  "photo_url":   "/p/7/000/253/05b/308dd6e.jpg",
  "positions": [
    {"job_title": "Co-chair", "organization": "Bill &
Melinda Gates Foundation"},
    {"job_title": "Co-founder, Chairman", "organization":
"Microsoft"}
  ],
  "education": [
    {"school_name": "Harvard University",       "start":
1973, "end": 1975},
    {"school_name": "Lakeside School, Seattle", "start":
null, "end": null}
  ],
  "contact_info": {
    "blog":     "http://thegatesnotes.com",
    "twitter": "http://twitter.com/BillGates"
  }
}
```

Document-based option:

- Encodes jobs, education, contact info as a document (expressed using JSON or XML syntax)

Document-based databases

MongoDB (the most well-known example)

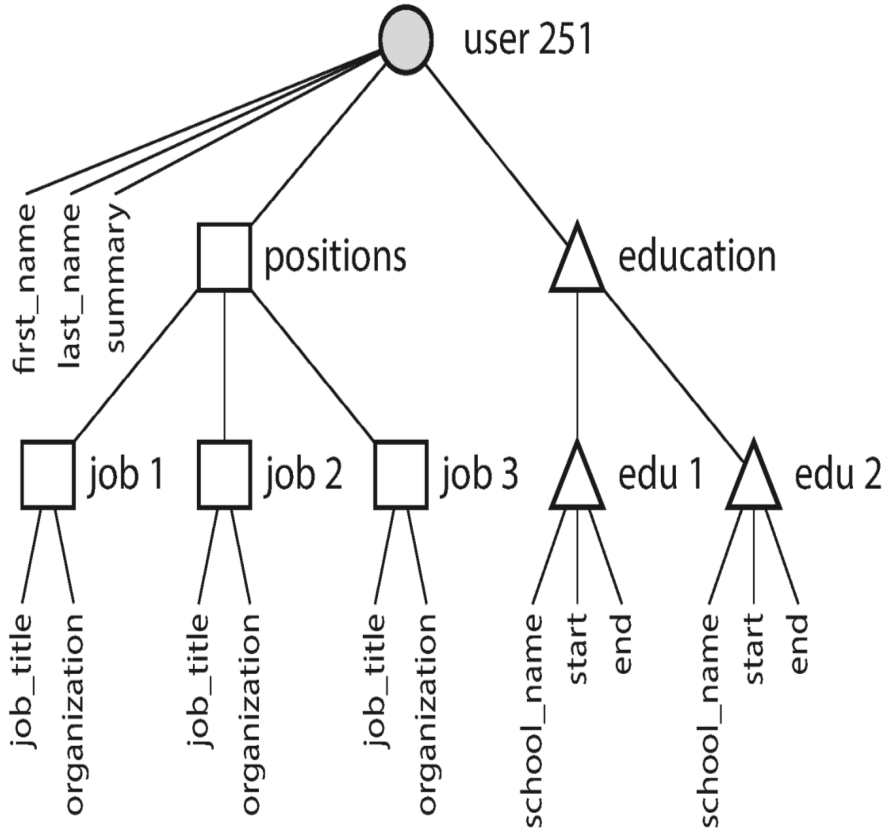
RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)

Notable points:

- Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose
- No joins (everything embedded in a single object)

https://www.tutorialspoint.com/mongodb/mongodb_overview.htm

Document-based databases



Embedded objects normally are the result of One-to-Many relationships

Improved “locality”

- a single retrieval request is enough to get all necessary into on “User”

Document model is not good with …

http://www.linkedin.com/in/williamhgates

Bill Gates

Greater Seattle Area | Philanthropy

Summary

Co-chair of the Bill & Melinda Gates Foundation.
Chairman, Microsoft Corporation, Voracious

Lookup situations?

users table

user_id	first_name	last_name	summary
251	Bill	Gates	Co-chair of … blogger.

region_id	industry_id	photo_id
us:91	131	57817532

regions table

id	region_name
us:7	Greater Boston Area
us:91	Greater Seattle Area

industries table

id	industry_name
43	Financial Services
48	Construction
131	Philanthropy

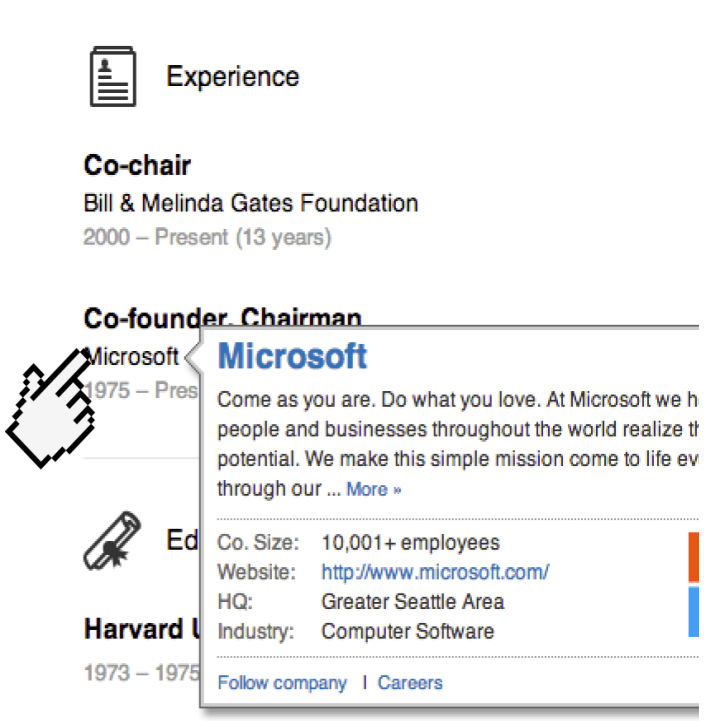
The relational model based solution of these “look-up tables” are useful:

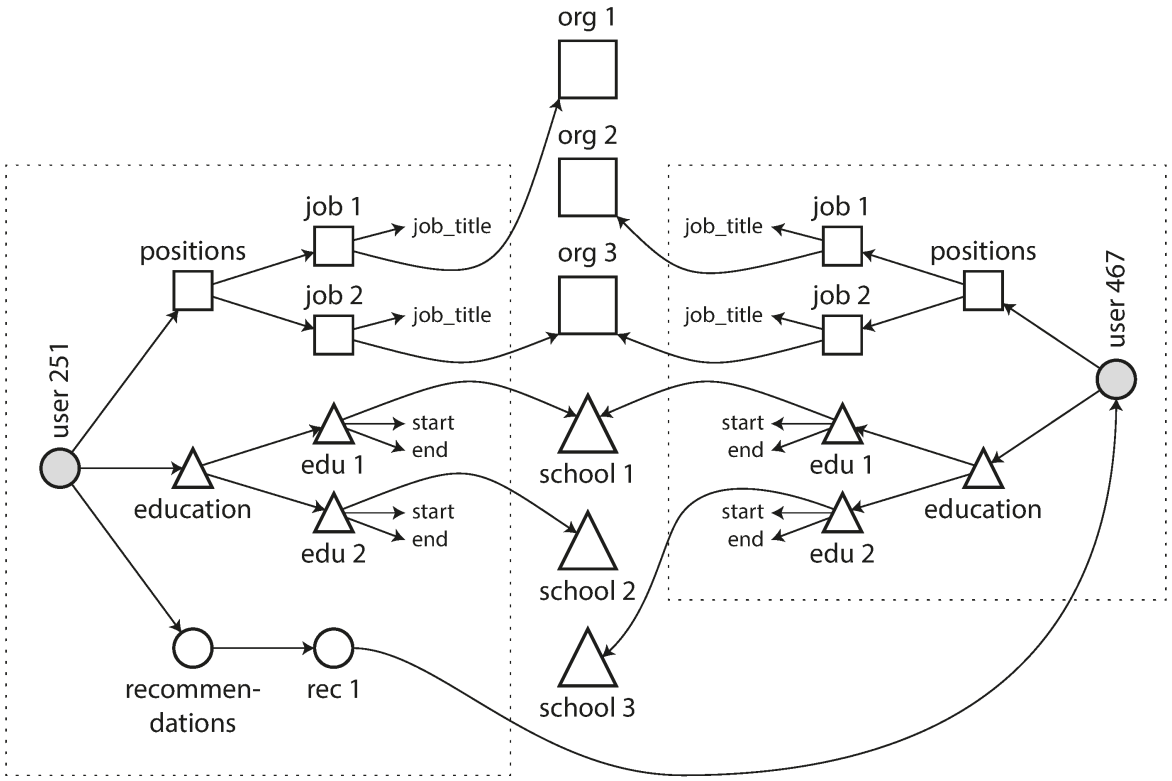
- Consistent style and spelling across Users
- Avoiding ambiguity (e.g., if several cities with the same name)
- Ease of updating (the name is stored in only one place)

Storing ID vs Text -> NOT duplicating text is more flexible and keeps data consistent – reason for normalising in RDB

Document model is not good with …

The single “documents” tend to become more interconnected as more features are added





The company – linking it as a full entity by itself (i.e., another document)

The recommendations – linking it to other Users **(Many-to-Many Relationships)**

In Document-based model, join support could be weak, the application code might have to resolve these relationships as needed (i.e., more hand-coding by the application developer)

Relational vs. Document

Which data model leads to simpler application code?

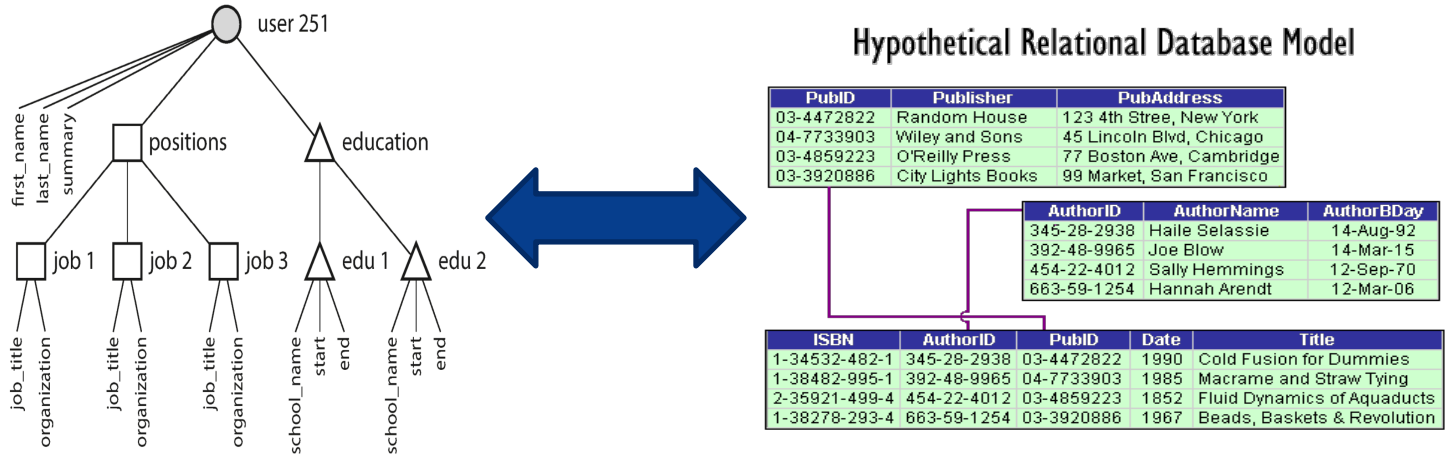
- If the application data objects looks more like a tree (i.e., document-like) → it can be loaded at once using the document-based model
- If M-M relationships are central to the application data → since the relational model is efficient in joins, relational DB may be advantageous. If document model is used, some of the 'join' logic will have to done by the application developers themselves (e.g., via for-loops)

Consider the kinds of relationships between data items. If they are highly interconnected data (e.g., social network)

- document model is not so good,
- relational model is OK …
- graph models would be natural (to be seen later)

Relational vs. Document

Convergence of document and relational databases



- PostgreSQL (since v.9.3), MySQL (since v.5.7). IBM DB2 (since v.10) support JSON documents.
- RethinkDB, MongoDB (document-based) support relational-like joins in its query language
- The two models can complement each other -> A hybrid model seems like a trend in these two systems

Designing Data-Intensive Applications, by Martin Kleppmann
https://www.zdnet.com/article/the-emergence-of-nosql-and-convergence-with-relational-databases/

PostgreSQL and JSON document type

https://www.postgresqltutorial.com/postgresql-json/

```
CREATE TABLE orders (

    id serial NOT NULL PRIMARY KEY,

    info json NOT NULL

);

INSERT INTO orders (info)

VALUES('{ "customer": "Lily Bush", "items": {"product": "Diaper","qty": 24}}'),

      ('{ "customer": "Josh William", "items": {"product": "Toy Car","qty": 1}}'),

      ('{ "customer": "Mary Clark", "items": {"product": "Toy Train","qty": 2},
                                             {“product”: “A Kitty Doll”, “qty”:1}}’);



SELECT info ->> 'customer' AS customer

FROM orders

WHERE info -> 'items' ->> 'product' = 'Diaper’;

Customer
--------
Lily Bush
```

UNSW SYDNEY

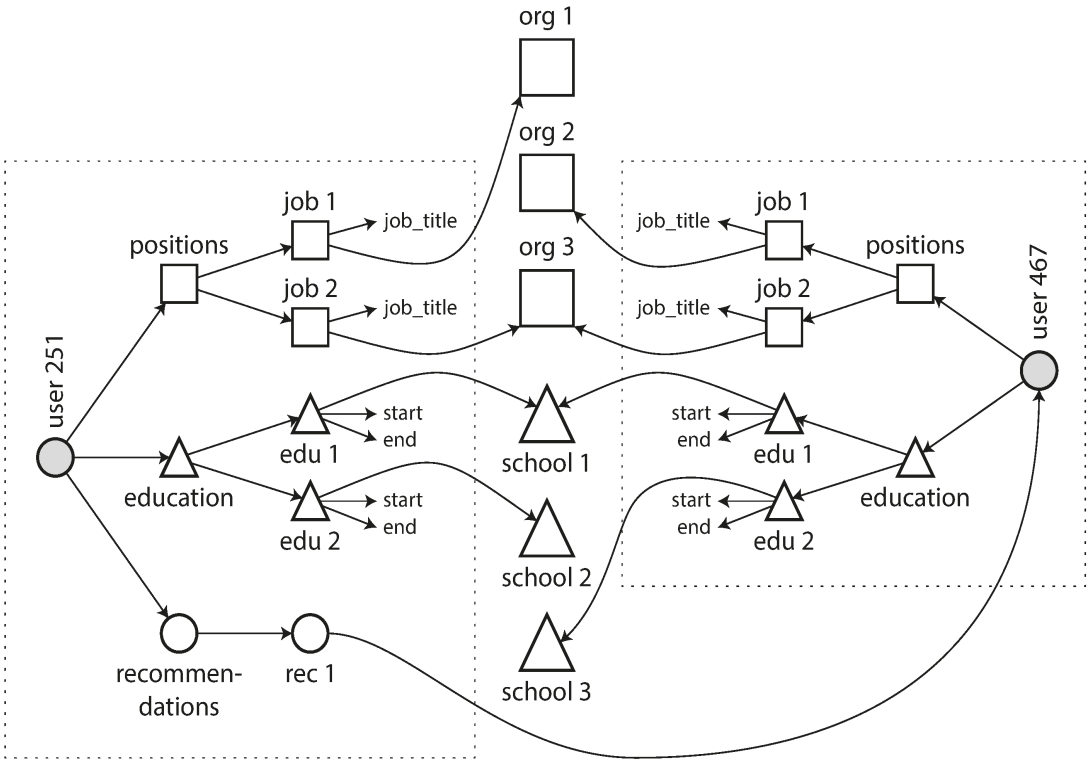
Graph-like Models

M-M relationships are an important factor in deciding which data model to go with

1-M (tree/doc), self-contained -> Document model

M-M -> either relational or graph

Highly M-M, complicated connections -> graph …



Graph:

- Vertices/nodes: represent entities
- Edges/arcs: represent relationships

The recommendations – linking it to other Users **(Many-to-Many Relationships)**

Graph-like Models

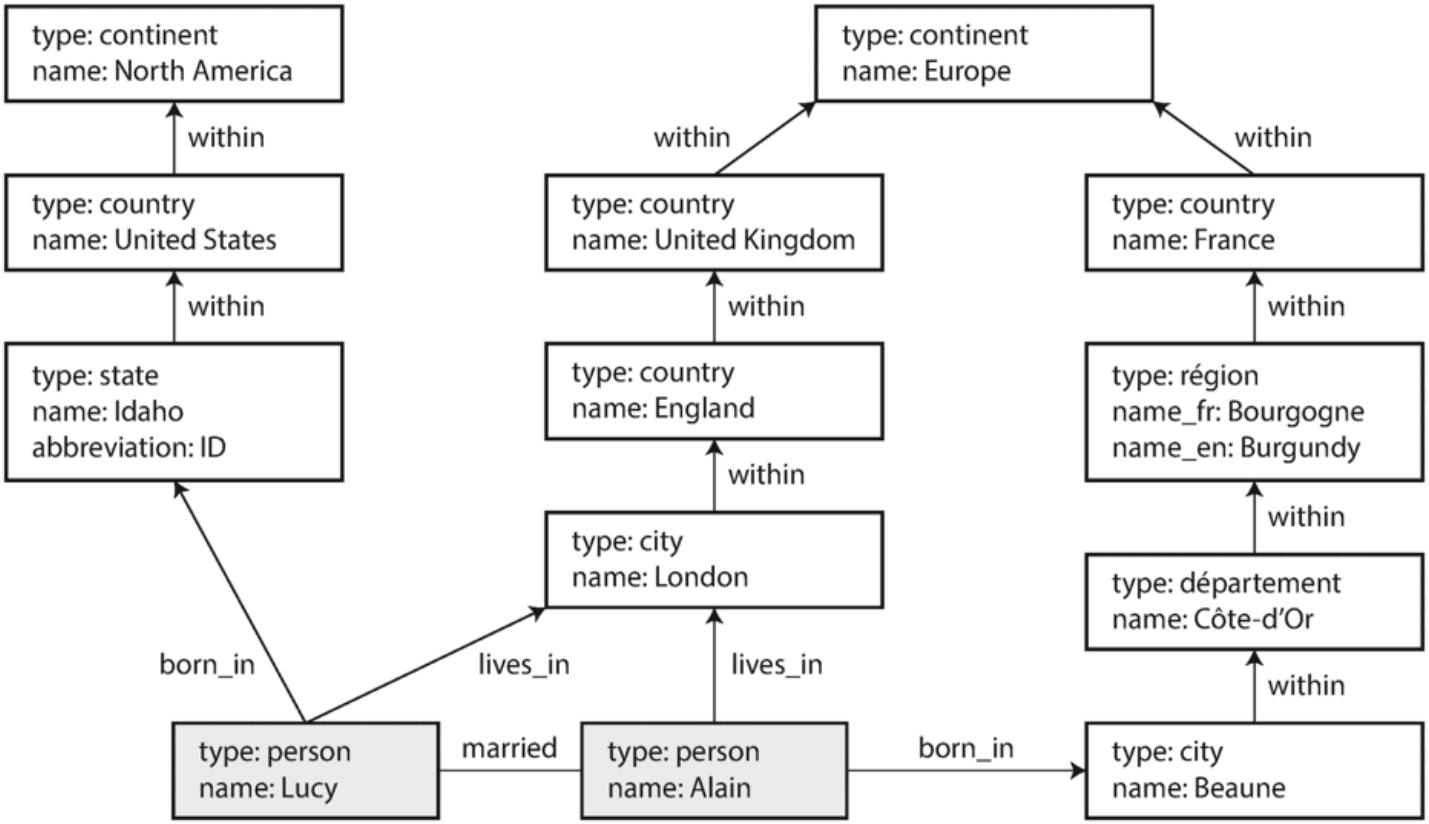
Many kinds of data can be modelled as a graph

- Social Graph – vertices are people, edges indicate which people know each other
- The Web Graph – vertices are web pages and edges indicate HTML links to other pages
- Road or Rail networks – vertices are junctions and edges represent the roads/railways between them

Well-known algorithms on the model

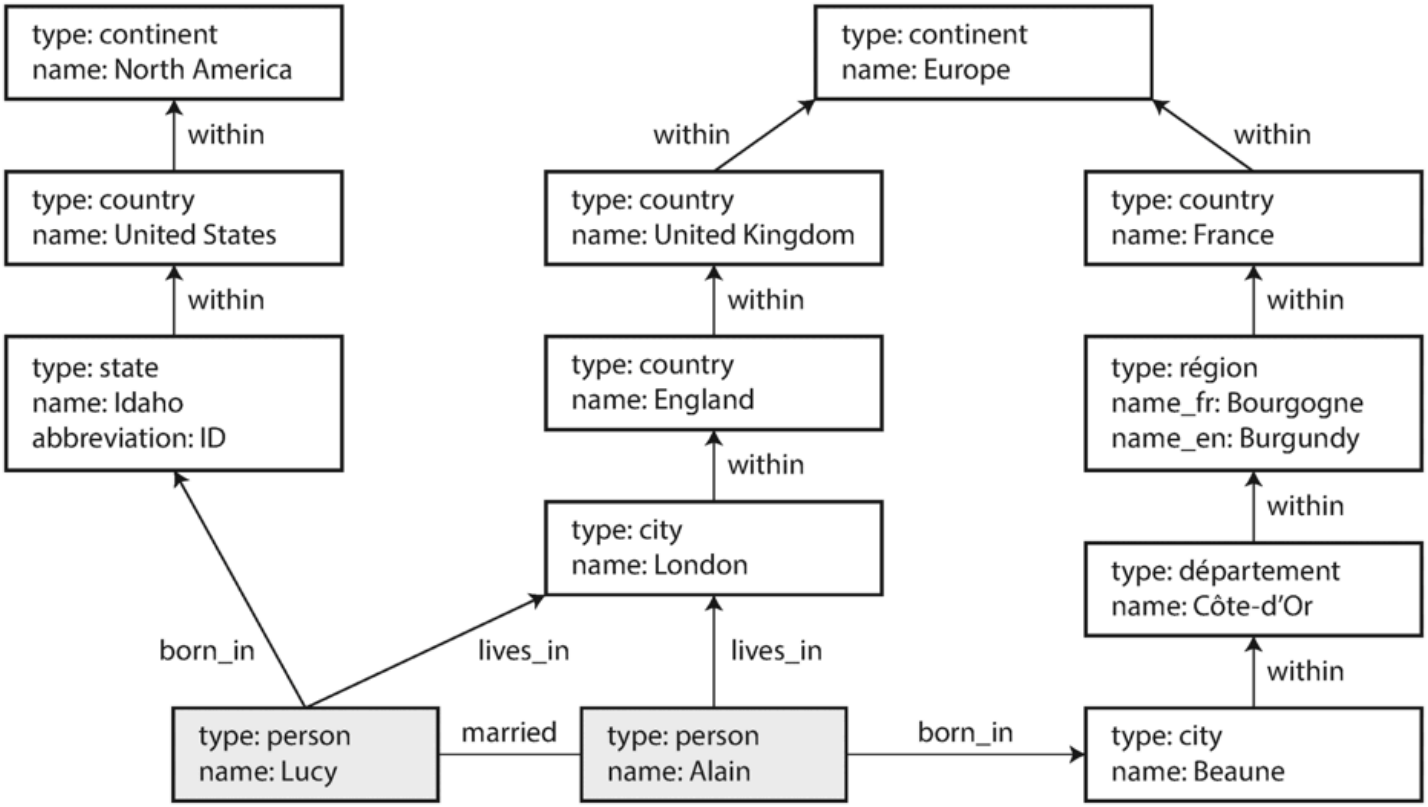
http://www.supplychain247.com/article/why_supply_chains_should_be_more_socially_engaged
http://canacopegdl.com/single.php?id=http://www-inst.eecs.berkeley.edu/~cs61bl/r//cur/graphs/web.graph.png
https://visualign.wordpress.com/2012/07/11/london-tube-map-and-graph-visualizations/

Graph-like Models



Vertices are not limited to the same type of data.

So graphs are “very” flexible … (cf. RDB)



- Different kinds of regional structures in different countries
- Type country “within” a type country
- Varying granularity (e.g., born_in “type:city”, born_in “type:state”)

Graph-like Models

Facebook, TAO system (2013)

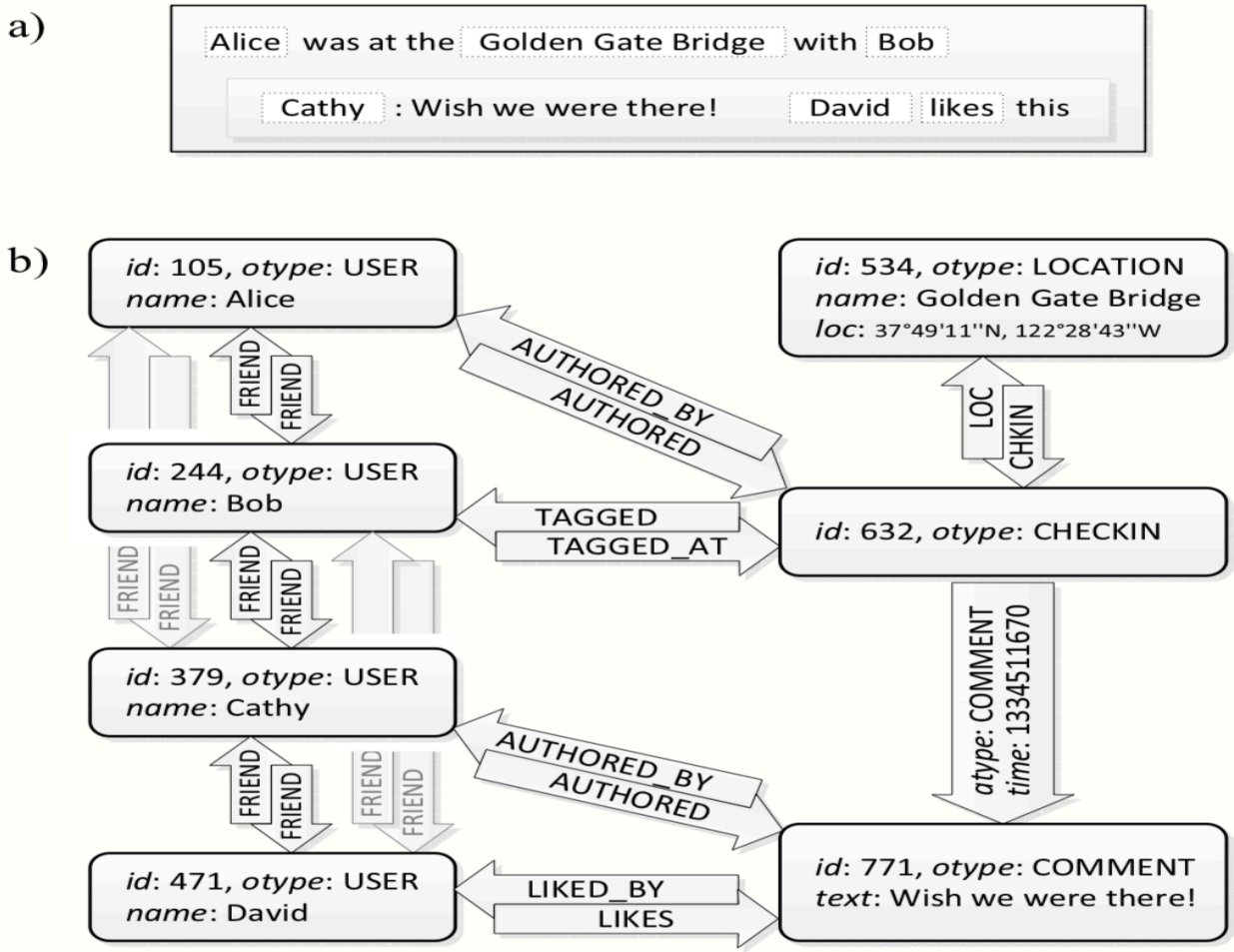


Figure 1: A running example of how a user's checkin might be mapped to objects and associations.

Designing Data-Intensive Applications, by Martin Kleppmann

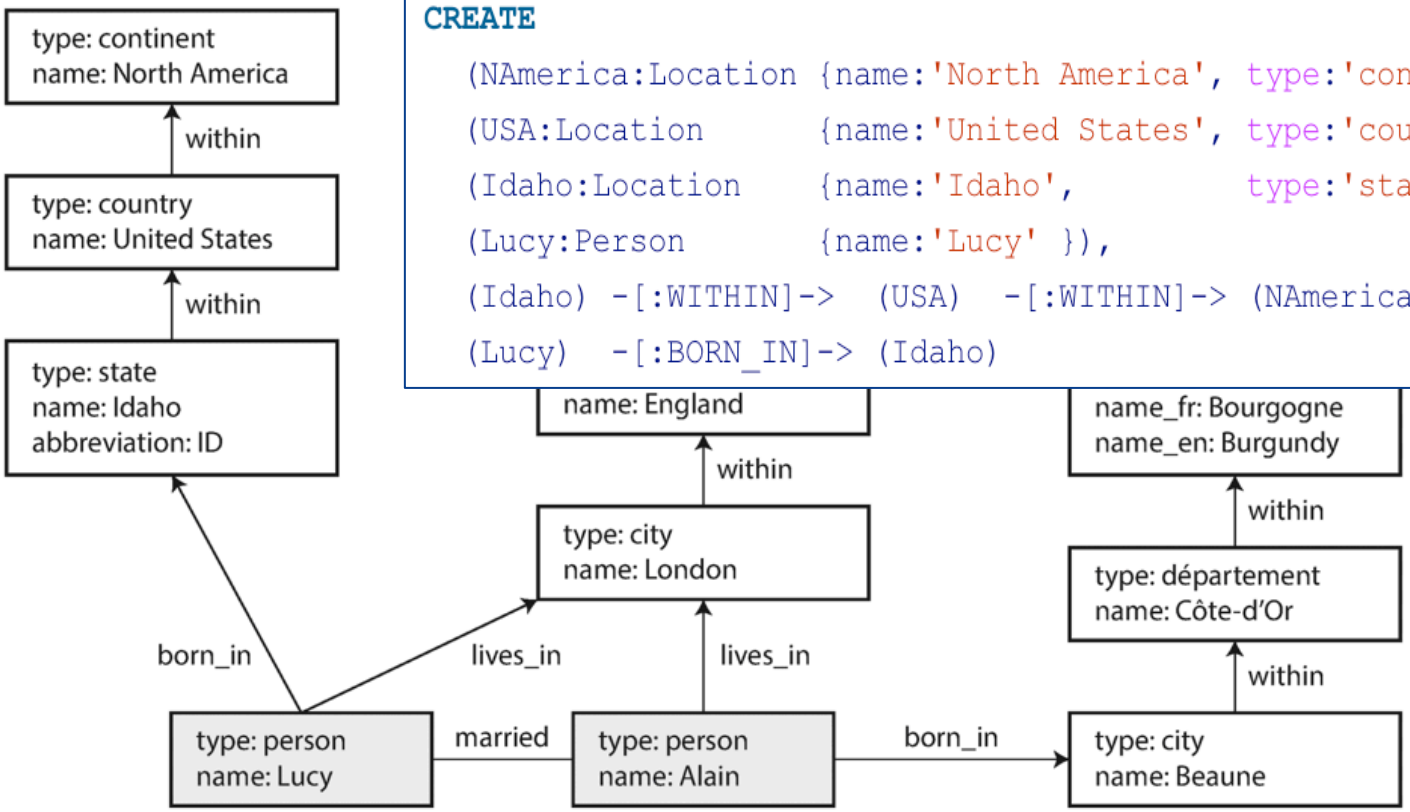
(https://www.usenix.org/system/files/conference/atc13/atc13-bronson.pdf)

Storing and Querying Graph-like Models

Cypher Query - declarative query language for graphs

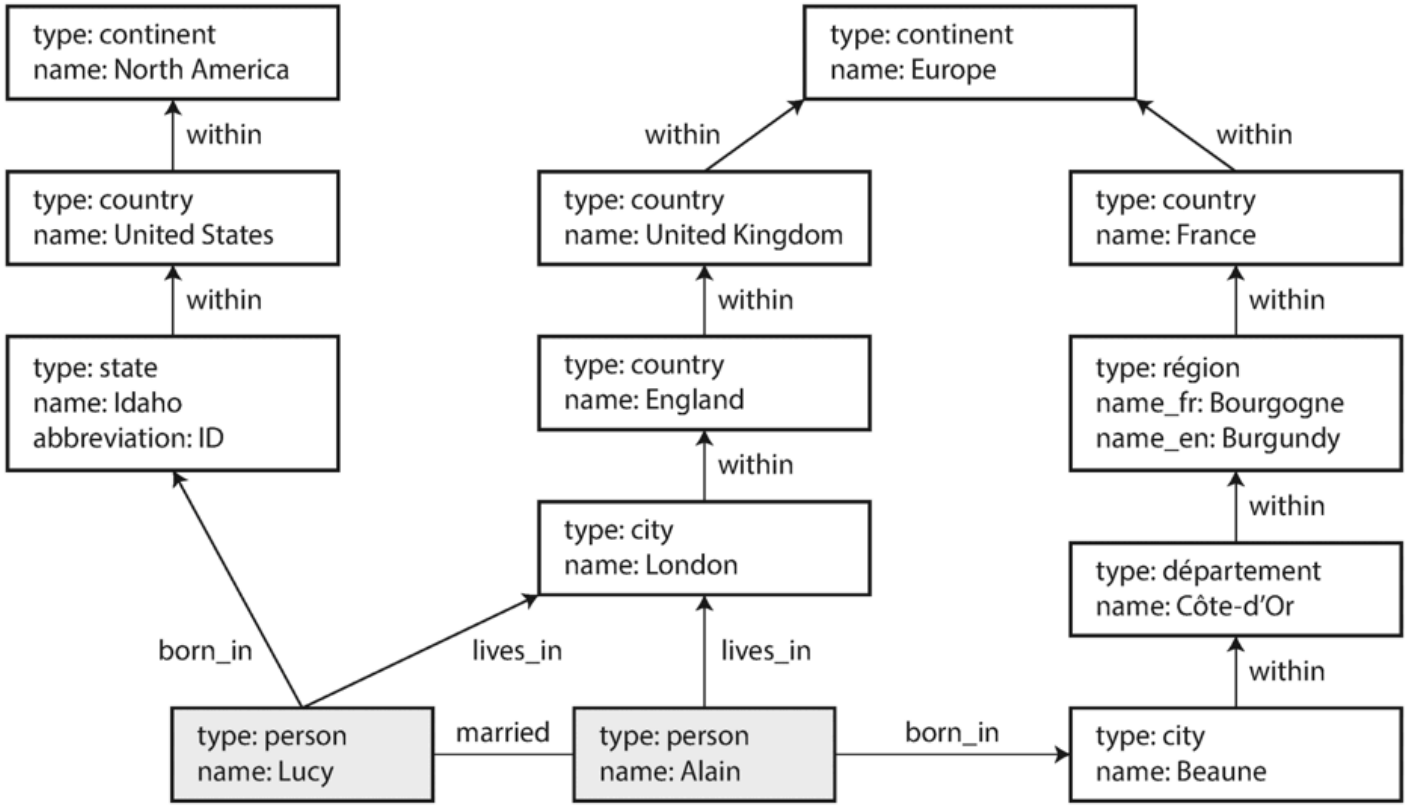
Example 2-3. A subset of the data in Figure 2-5, represented as a Cypher query

```
CREATE
  (NAmerica:Location {name:'North America', type:'continent'}),
  (USA:Location      {name:'United States', type:'country'  }),
  (Idaho:Location    {name:'Idaho',         type:'state'    }),
  (Lucy:Person       {name:'Lucy' }),
  (Idaho) -[:WITHIN]->  (USA)  -[:WITHIN]-> (NAmerica),
  (Lucy)  -[:BORN_IN]-> (Idaho)
```



Example 2-4. Cypher query to find people who emigrated from the US to Europe

```
MATCH
  (person) -[:BORN_IN]->  () -[:WITHIN*0..]-> (us:Location {name:'United
States'}),
  (person) -[:LIVES_IN]-> () -[:WITHIN*0..]-> (eu:Location
{name:'Europe'})
RETURN person.name
```



"Find the names of all the people who emigrated from the United States to Europe"

"Declarative" language -> the execution details hidden

XML data model and query language

in HTML ...

```
<html>
<h1>Bibliography</h1>
<ol>
<li><i>Foundation of Databases</i>,
  <b>Abiteboul, Hull</b>, 1995</li>
<li><i>Database Systems</i>
  <b>Elmasri, Navathe</b>, 1994</li>
</ol>
</html>
```

in XML ...

```
<bibliography>
<book>
 <title>Foundation of Databases</titl
 <author>Abiteboul</author>
 <author>Hull</author>
 <year>1995</year>
</book>
<book> <!-- continues --> </book>
</bibliography>
```

- A simple, very flexible and extensible text data format
- “extensible” because the markup format is not fixed like HTML
  - It lets you design your own customised markup
- XML is a language that describes data
  - It separates presentation issues from the actual data

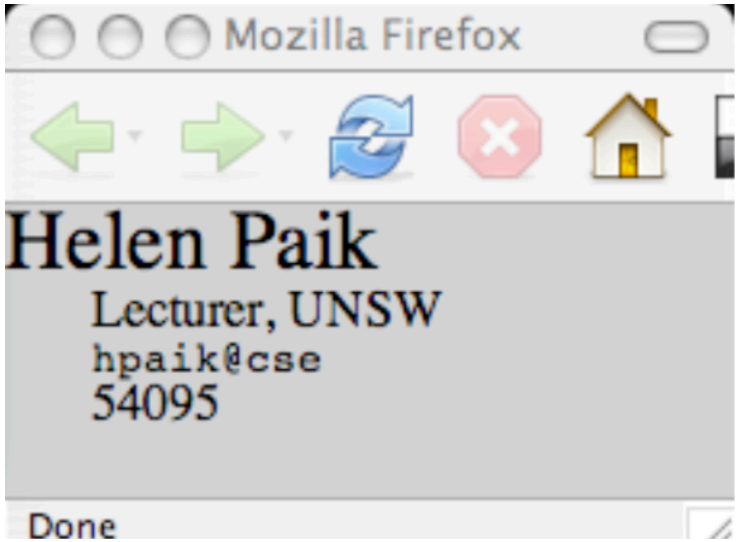
XML – separating content/presentation

XML

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="staffcard.css" ?>
<staff>
   <name>Helen Paik</name>
   <title>Lecturer, UNSW</title>
   <email>hpaik@cse</email>
   <extension>54095</extension>
   <photo src="me.gif" />
</staff>
```

CSS

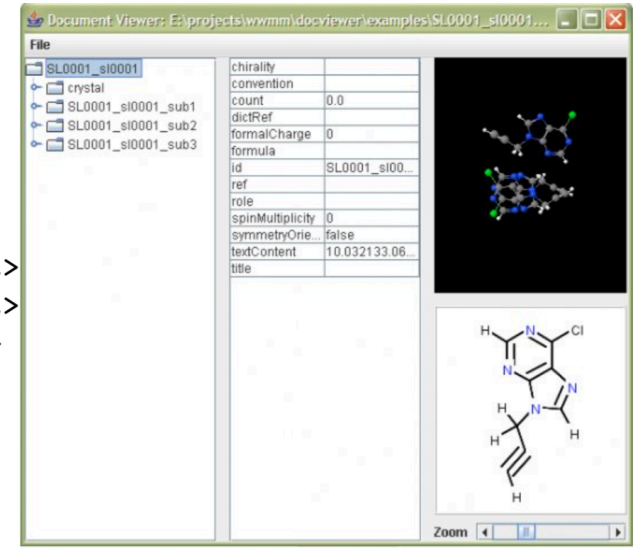
```
staff{background-color: #cccccc; ...}
name{display: block; font-size: 20pt; ... }
title{display: block; margin-left: 20pt;}
email{display: block; font-family: monospace;
extension{display: block; margin-left: 20pt;}
```



XML – many applications …

Chemical Markup Language (CML)

```
<atom id="caffeine_karne_a_1">
 <float builtin="x3" units="A">-2.8709</float>
 <float builtin="y3" units="A">-1.0499</float>
 <float builtin="z3" units="A">0.1718</float>
 <string builtin="elementType">C</string>
</atom>
```



Math Markup Language (MathML)

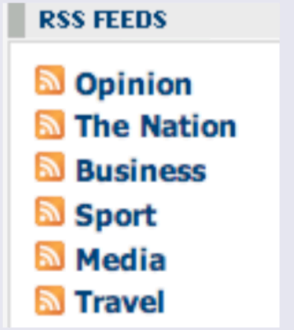
```
<mrow>
<apply><eq/>
 <ci>A</ci>
 <matrix>
   <matrixrow><ci>x</ci><ci>y</ci></matrixrow>
   <matrixrow><ci>z</ci><ci>w</ci></matrixrow>
 </matrix>
</apply>
</mrow>
```

$$A = \begin{pmatrix} X & Y \\ Z & W \end{pmatrix}$$

XML – many applications …

Data Feeds (RSS and ATOM)

```
<rss version="0.91">
<channel>
<title>CNN.com</title>
<item>
<title>July ends with 76 ... killed</title>
<link>http://www.cnn.com/.../story.html</link>
<description>Three U.S. soldiers were ...</description>
</item>
```



SVG

- https://pixabay.com/en/photos/svg/

Many more … (e.g., system configuration files, XML-based APIs)

Accessing/Querying XML files

XQuery is a declarative language in which a query is represented as an expression

Sample XQuery

```
<bib>
{
  for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
     { $b/author}
    </book>
}
</bib>
```

Can you "read" it?

file: select-v2.xq

Accessing/Querying XML files

```
for $book in doc("books.xml")//BOOKS/ITEM,
    $quote in doc("quotes.xml")//listing/ITEM
where $book/ISBN = $quote/ISBN
return
 <book>
   {$book/TITLE, $quote/PRICE}
 </book>
```

Join two documents

```
<results> {
for $b in doc("bib.xml")//book
return
<holding>
  {
    $b/title,
    if ($b/@type = "Journal")
       then $b/editor
       else $b/author
  }
</holding>
}
</results>
```

If/else conditions

XML data model and query language

Benefits of using XML in document

- Self-describing, modular and portable data
- A common, widely accepted data representation language for the Web
- Standard support for checking validity of data (XML can have a schema)
- Efficient search and query language
  - Standard support for querying XML docs
  - Quick and simple search (XPath)
  - More comprehensive keyword + structure based search possible as well (XQuery)

Advantages/Disadvantages of NoSQL

Which available data model to use should be decided on your data requirements.

Not all NoSQL solutions are equal – i.e., document-based model and graph-based model serve different data requirements

Generally, NoSQL solutions are considered lightweight and easy to implement (e.g., no schema required) – and could have high read/write throughput due to the relaxation in data consistency requirement

However, NoSQL technologies are relatively new still – not as well optimised/developed as RDBMS

Schema-less data storage could lead to less manageable database overtime.