Designing Data-Intensive Applications

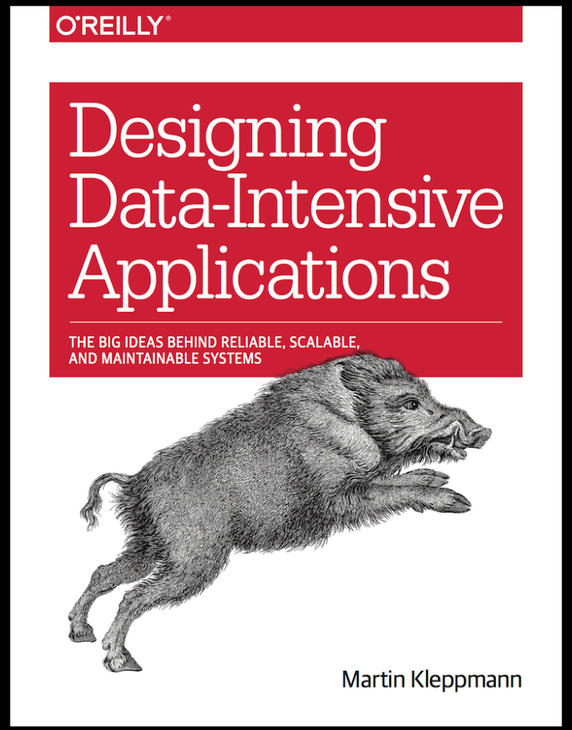
Part 1

Mark Stetzer - Proofpoint

@stetzer

Quick Disclaimer

— I didn't write this book, just really admire it
— You should get a copy https://dataintensive.net/
— Seriously, you should get a copy (everyone on my team has one)



Outline

- Data Models and Query Languages
  - Relational vs Document Models
    - Representing Many-to-Many Relationships
  - Querying Your Data
    - SQL vs M/R
  - Graph Data Models

Outline

- Storage and Retrieval
  - Data Structures Behind Your Favorite Databases
  - Optimized for Transactions or Analytics?
  - Row- or Column-Oriented?

Relational Models

— Probably what most people are familiar with
— Been around since 1970s; an *eternity* in CS!
— Generalizes very well (breaking large things into their smaller relationships)
— Powers most of what you see on the web, meatspace today

Document Models

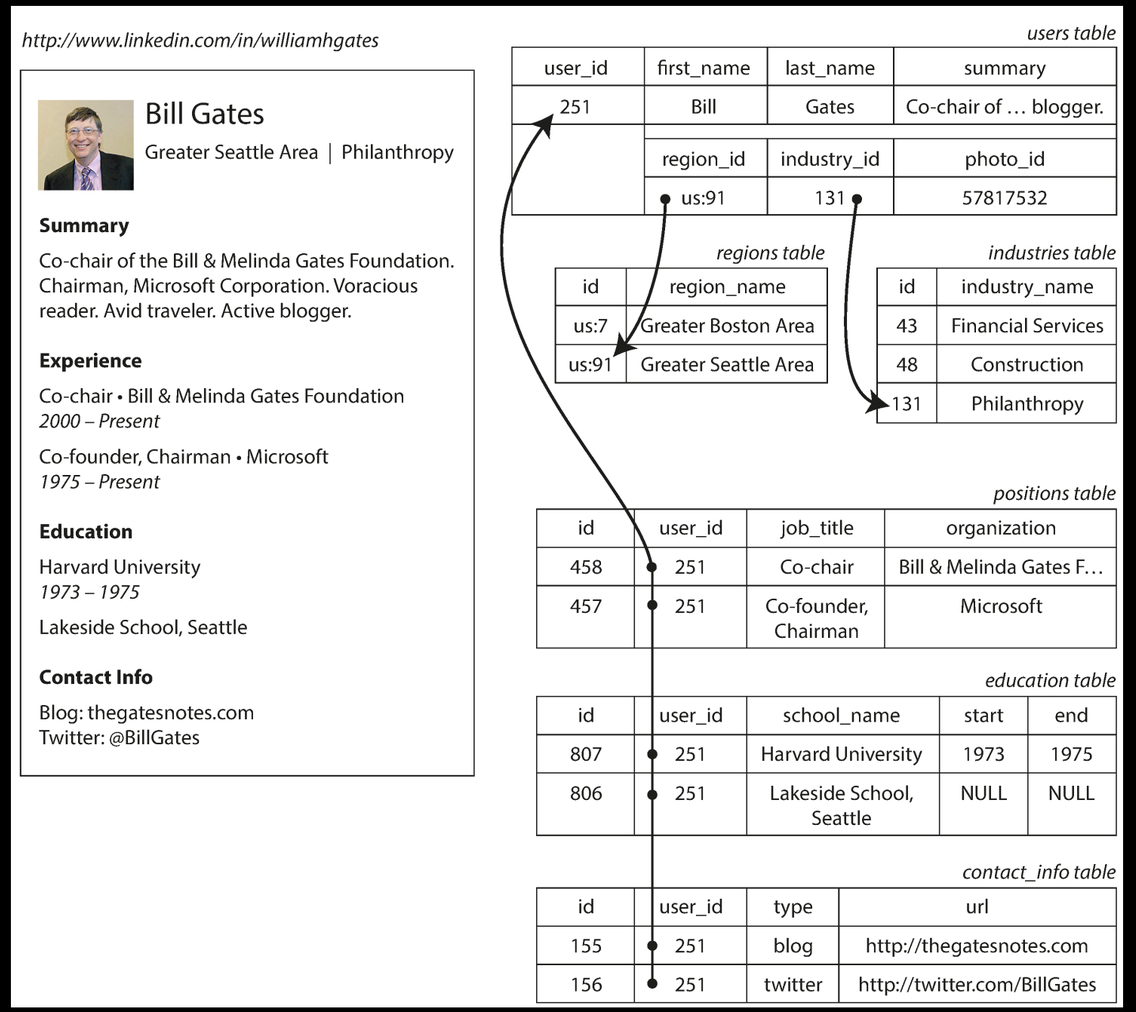
— Object databases showed up in '80s & '90s; XML in 2000s

— For object-oriented codebases, relational models can feel in an *impedance mismatch*

— Object Relational Mappers invented to deal with this mismatch, but are imperfect

— RDBMSes can store/query XML, JSON, etc. to varying degrees

An Example


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 reader. Avid traveler. Active blogger.
Experience
Co-chair • Bill & Melinda Gates Foundation
2000 – Present
Co-founder, Chairman • Microsoft
1975 – Present
Education
Harvard University
1973 – 1975
Lakeside School, Seattle
Contact Info
Blog: thegatesnotes.com
Twitter: @BillGates
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

Pros & Cons

Item	Relational Model	Document Model
Simplicity	Must model data	No barrier to entry
Locality	Multiple queries	One spot
Duplication of data	Normalized relations	Lots of duplication
Search	Modeled relationships (e.g. geo)	String search across docs
Reporting	Robust reporting capabilities, known tools	Map/Reduce for reports?

To Schema, or not to Schema...

— Common claim is that document databases don't enforce schemas

— *Technically* true, but there is at least an implicit schema in code (schema-on-read)

— Schema-on-read can be advantageous if you have many types of objects (and don't want to have a table for each)

Querying Your Data: SQL

— Most programming languages are *imperative*: tell me what to do, in specified order

— SQL is *declarative*: tell me what to do, I'll figure out how

— SQL seems to provide similar level of simplicity to querying that document model did for writing data

Querying Your Data: Map/Reduce

— M/R somewhere between imperative & declarative: you provide code for specific map & reduce functions; framework takes care of rest

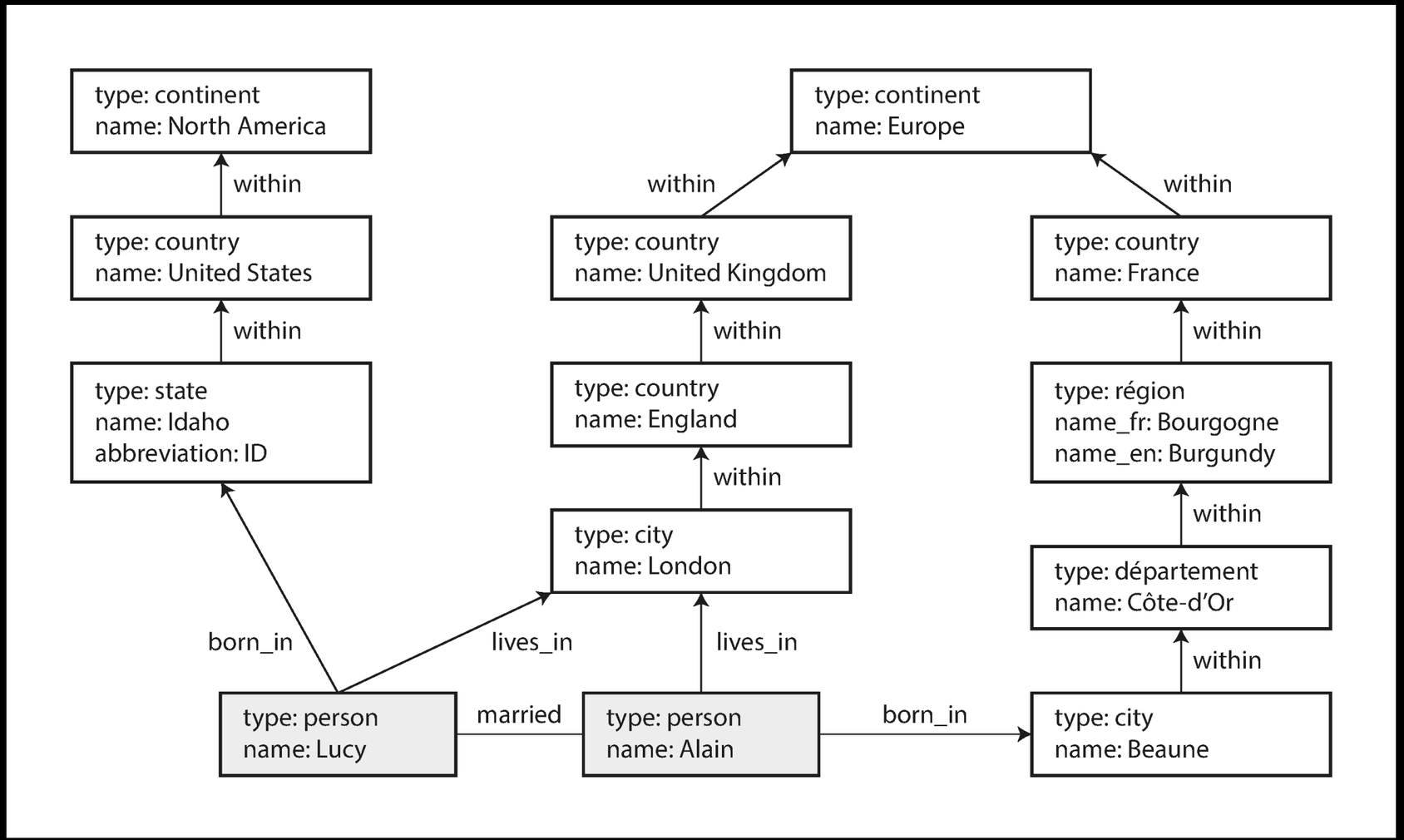
— Typically restrictions on what function code you can provide for *map* & *reduce*; pure functional (no additional db queries, etc.)

Graph Models

— Can be useful to model data where many-to-many relationships are common

— Optionally can provide direction between relationships (directional edges)

An Example


type: continent
name: North America
within
type: country
name: United States
within
type: state
name: Idaho
abbreviation: ID
type: continent
name: Europe
within
within
type: country
name: United Kingdom
type: country
name: France
within
within
type: country
name: England
type: région
name_fr: Bourgogne
name_en: Burgundy
within
within
type: city
name: London
type: département
name: Côte-d'Or
within
born_in
lives_in
lives_in
type: person
name: Lucy
married
type: person
name: Alain
born_in
type: city
name: Beaune

Property Graphs

- Vertex
  - ID
  - Outgoing Edges
  - Incoming Edges
  - Collection of Properties
- Edge
  - ID
  - Edge Start (tail vertex)
  - Edge End (head vertex)
  - Label
  - Collection of Properties

Querying Property Graphs

— One example of how to query is *Cypher*, the language created for Neo4j

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

Triple-Stores

— Mostly equivalent to property graph model
— e.g. *Mark, works for, Proofpoint*
— Specifically mentioned in book due to semantic web-type context (e.g. RDF data, SPARQL)

Models mentioned, not covered in this section

— Sequence-similarity stores like *GenBank*
— Custom stores for extremely large data volumes (CERN's ROOT)
— Full-text search indexes (covered later)

What's Next?

— Storage & Retrieval deep dive
— Encoding & Evolution
— Data Replication & Partitioning
— etc.

Futher Reading

— Why You Should Never Use MongoDB
— Schemaless Data Structures
— The Trouble with Types
— Spanner: Google's Globally-Distributed Database
— ROOT for Big Data Analysis
— BLAST Your Way through Malware (Malware Analysis Assisted by Bioinformatics Tools)