Advanced Data Management (CSCI 490/680)

Scalable Databases

Dr. David Koop
Studying Data Availability

• Who mandates data sharing, and what is the impact?
  - Government
  - Funding agencies
  - Institutions
  - Journals

• How does the age of a publication/data item affect availability?
  - If not curated, how to locate?
  - What factors influence this?
Since this is a logistic model, we can readily calculate the effect that the different policy types have on the likelihood that the data will be available. We explore these odds for each type of policy below, using "no policy" as the baseline.

Having a "recommend archiving" policy made it 3.6 times more likely that the data were online compared to having no policy. However, the 95% CI overlapped with 1 (0.96–13.6); hence, this increase in the odds is not significant. Overall, recommending data archiving is only marginally more effective than having no policy at all.

The data were 17 times more likely to be available online for journals that had adopted a mandatory data archiving policy but did not require a data accessibility statement in the manuscript. This odds ratio was significantly higher (95% CI: 3.7–79.6).

For "mandate archiving" journals where a data accessibility statement is required in the manuscript, the odds of finding the data online were 974 times higher compared to having no policy. The 95% CI on these odds is very wide (97.9–9698.8), but nonetheless shows that the combination of a mandatory policy and an accessibility statement is much more effective than any other policy type.

**REQUESTING DATA DIRECTLY FROM AUTHORS**

A number of the "recommend archiving" policies state that the data should also be freely available from the authors by request (see the Journal Policies file at doi: 10.5061/dryad.6bs31); hence, we wanted to evaluate whether obtaining data directly from authors is an effective approach. Part of the dataset collection for our reproducibility study (5) involved e-mailing authors of papers from two of the "recommend archiving" journals (BMC Evolutionary Biology and PLoS One) and requesting their structure input files. Here, we examine how often these requests led to us obtaining the data. We did not e-mail the authors of articles where the data were already available online. A detailed description of our data request process appears on Dryad (doi: 10.5061/dryad.6bs31), but we essentially contacted corresponding and senior authors of each article up to 3 times over a 3-wk period, and recorded if and when the data were received.

We obtained data directly from the authors for 7 of the 12 eligible articles in BMC Evolutionary Biology, and 27 datasets from 45 articles from PLoS One (Table 1). All seven of the BMC Evolutionary Biology datasets arrived between 8 and 14 d after our initial request. Ten of the PLoS One datasets came within 1 wk, 13 came between 8 and 14 d, and 4 arrived between 15 and 21 d. Unlike the online data, which could generally be obtained within a few minutes, the requested datasets took a mean of 7.7 d to arrive, with one author responding that the dataset had been lost in the year since publication. More than one e-mail had to be sent to the corresponding and/or senior author for 53% of papers, and the authors of 29% of the papers did not respond to any of our requests. No data were received 21 d after our initial request. We also note that requesting data via e-mail did upset some authors, particularly when they were reminded of the journal's data archiving policy or when multiple e-mails were sent.

Our average return of 59% in an average of 7.7 d is markedly better than has been reported in similar studies: Wicherts et al. (8) received only 26% of requested datasets after 6 mo of effort with authors of 141 psychology articles, and Savage and Vickers (9) received only 1 of 10 eligible papers with data available online.

**Figure 1.** Percentage of eligible papers published in 2011 that made their data available online, by journal. Number of eligible papers is shown above each column. Within the "mandate archiving" group, "data statement" denotes the journals that require a data accessibility statement in the manuscript, and "no data statement" denotes those that do not.

[T. Vines et al., 2013]
We found a strong effect of article age on the availability of data from these 516 studies. The decline in data availability could arise because the authors of older papers were less likely to respond, but this was not supported by the data. Instead, researchers were equally likely to respond (Figure 1B) and to indicate the status of their data (Figure 1C) across the entire range of article ages. The major cause of the reduced data availability for older papers was the rapid increase in the proportion of data sets reported as either lost or on inaccessible storage media. For papers where authors reported the status of their data, the odds of the data being extant decreased by 17% per year (Figure 1D). There was a continuum of author responses between the data being reported lost and being stored on inaccessible media, and they seemed to vary with the amount of time and effort involved in retrieving the data. Responses included authors being sure that the data were lost (e.g., on a stolen computer) or thinking that they might be stored in some distant location (e.g., their parent's attic) to authors having some degree of certainty that the data are on a Zip or floppy disk in their possession but no longer having the appropriate hardware to access it. In the latter two cases, the authors would have to devote hours or days to retrieving the data. Our reason for needing the data (a reproducibility study) was not especially compelling for authors, and we may have received more of these inaccessible data sets if we had offered authorship on the subsequent paper or said that the data were needed for an important medical or conservation project.

The odds that we were able to find an apparently working e-mail address (either in the paper or by searching online) for any of the contacted authors did decrease by about 7% per year. This decrease was partly driven by a dearth of e-mail addresses in articles published before 2000 (0.38 per paper on average for 1991–1999) compared with those published after 2000.

### Table 1. Breakdown of Data Availability by Year of Publication

<table>
<thead>
<tr>
<th>Year</th>
<th>No Working E-Mail</th>
<th>No Response to E-Mail</th>
<th>Response Did Not Give Status of Data</th>
<th>Data Lost</th>
<th>Data Exist, Unwilling to Share</th>
<th>Data Received</th>
<th>Data Extant (Unwilling to Share + Received)</th>
<th>Number of Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1991</td>
<td>9 (35%)</td>
<td>9 (35%)</td>
<td>2 (8%)</td>
<td>4 (15%)</td>
<td>1 (4%)</td>
<td>1 (4%)</td>
<td>2 (8%)</td>
<td>26</td>
</tr>
<tr>
<td>1993</td>
<td>14 (39%)</td>
<td>11 (31%)</td>
<td>3 (8%)</td>
<td>7 (19%)</td>
<td>0 (0%)</td>
<td>1 (3%)</td>
<td>1 (3%)</td>
<td>36</td>
</tr>
<tr>
<td>1995</td>
<td>11 (31%)</td>
<td>9 (26%)</td>
<td>0 (0%)</td>
<td>7 (20%)</td>
<td>2 (6%)</td>
<td>6 (17%)</td>
<td>8 (23%)</td>
<td>35</td>
</tr>
<tr>
<td>1997</td>
<td>11 (37%)</td>
<td>9 (30%)</td>
<td>1 (3%)</td>
<td>2 (7%)</td>
<td>3 (10%)</td>
<td>4 (13%)</td>
<td>7 (23%)</td>
<td>30</td>
</tr>
<tr>
<td>1999</td>
<td>19 (48%)</td>
<td>13 (32%)</td>
<td>1 (2%)</td>
<td>1 (2%)</td>
<td>0 (0%)</td>
<td>6 (15%)</td>
<td>6 (15%)</td>
<td>40</td>
</tr>
<tr>
<td>2001</td>
<td>13 (30%)</td>
<td>15 (35%)</td>
<td>3 (7%)</td>
<td>4 (9%)</td>
<td>0 (0%)</td>
<td>8 (19%)</td>
<td>8 (19%)</td>
<td>43</td>
</tr>
<tr>
<td>2003</td>
<td>9 (20%)</td>
<td>20 (43%)</td>
<td>4 (9%)</td>
<td>4 (9%)</td>
<td>0 (0%)</td>
<td>11 (24%)</td>
<td>11 (24%)</td>
<td>46</td>
</tr>
<tr>
<td>2005</td>
<td>11 (24%)</td>
<td>14 (31%)</td>
<td>6 (13%)</td>
<td>2 (4%)</td>
<td>0 (0%)</td>
<td>11 (24%)</td>
<td>11 (24%)</td>
<td>45</td>
</tr>
<tr>
<td>2007</td>
<td>12 (18%)</td>
<td>31 (47%)</td>
<td>2 (3%)</td>
<td>4 (6%)</td>
<td>1 (2%)</td>
<td>16 (24%)</td>
<td>17 (26%)</td>
<td>66</td>
</tr>
<tr>
<td>2009</td>
<td>9 (13%)</td>
<td>34 (49%)</td>
<td>3 (4%)</td>
<td>5 (7%)</td>
<td>6 (9%)</td>
<td>12 (17%)</td>
<td>18 (26%)</td>
<td>69</td>
</tr>
<tr>
<td>2011</td>
<td>13 (16%)</td>
<td>29 (36%)</td>
<td>8 (10%)</td>
<td>0 (0%)</td>
<td>7 (9%)</td>
<td>23 (29%)</td>
<td>30 (38%)</td>
<td>80</td>
</tr>
<tr>
<td>Totals</td>
<td>131 (25%)</td>
<td>194 (38%)</td>
<td>33 (6%)</td>
<td>37 (7%)</td>
<td>20 (4%)</td>
<td>101 (19%)</td>
<td>121 (23%)</td>
<td>516</td>
</tr>
</tbody>
</table>

Data are displayed as n (%); the percentages are calculated by rows.
Why Share Data? Increased Citations

![Box plot showing the number of citations for articles with data shared and not shared.]

Articles with Data Not Shared (n=44) vs. Articles with Data Shared (n=41)

Note: log scale

[Why Share Data? Increased Citations](H. Piwowar, 2013)
What Factors Impact Sharing?

- funded by NIH?
- size of grant
- sharing plan req’d?
- funded by non-NIH?
- impact factor
- strength of policy
- open access?
- number of microarray studies published
- years since first paper
- # pubs
- # citations
- previously shared?
- previously reused?
- gender
- sector
- size
- impact rank
- country
- humans?
- mice?
- plants?
- cancer?
- clinical trial?
- number of authors
- year

[H. Piwowar, 2013]
Why not data sharing? (self-reported)

- Sharing is too much effort
- Want student or junior faculty to publish more
- They themselves want to publish more
- Cost
- Industrial sponsor
- Confidentiality
- Commercial value of results

[Campbell et al., 2002 via Piwowar, 2013]
Joint Declaration of Data Citation Principles

- Precursor to FAIR
- Importance: data is legitimate, citations should have importance
- Credit and Attribution: scholarly credit to all contributors
- Evidence: when data is relied on, it should be cited
- Unique Identification: machine-actionable, globally unique, and widely used
- Access: data, metadata, etc. is findable and usable
- Persistence: identifiers, metadata persist regardless of whether data does
- Specificity and Verifiability: provenance, fixity, granularity
- Interoperability and Flexibility: allow for variability across communities
Generic Data Citation

- Author(s), Year, Dataset Title, Global Persistent Identifier, Data Repository or Archive, version or subset
- Authors, repository → Principle 2
- Year and title → not related to principle but consistent with other citations
- Global Persistent Identifier: Principle 4 and 6
Computational Data Citation

• Given a database D and a query Q, generate an appropriate citation.
• Automatic Citation requires the answers to two questions:
  - Does the citation depend on both Q and D or just on the data Q(D) extracted by Q from D?
  - If we have appropriate citations for some queries, can we use them to construct citations for other queries?
• If the data is an image or numbers, cannot expect the citation to live in that data
• If the query returns an empty dataset, we still may wish to cite that
• People know how to cite certain parts of a dataset but not all…

[Buneman et al., 2016]
Views and Citable Units

- Views describe "areas of responsibility" for parts of a database
- Use views to create "citable units"
- Determine which view V answers a particular query Q and generate a citation for the view
- What happens if two different views can answer the same query?
Citable Views and Partial Citations

![Diagram showing a hierarchy with nodes labeled 'root', 'families', 'introduction', 'targets', 'introduction', 'targets', 'tables', and 'tuples'. Each node has associated URIs and contributors.]

[URI: .../family/1234 Collaborators: Harmar, Sharman, Miller]

[URI: .../intro/987 Contributors: Miller, Drucker]

[URI: .../target/1234 Contributors: Miller, Drucker, Salvatori]
Next Class's Reading Response

- Spanner: Google's Globally-Distributed Database
- Reading Response for Monday:
  - Focus on main concepts in the paper
  - Submit to Blackboard
Assignment 4

- World Education Data
- Collected/collated by UNESCO, World Bank, and OECD
- Transform World Bank Data
- Impute missing year data
- Integrate teacher and student numbers
- Fuse three datasets
Scalable Database Systems
Introduction

Fig. 1.1 Main components of a DBMS.

As a well-understood point of reference for new extensions and revolutions in database systems that may arise in the future, we focus on relational database systems throughout this paper.

At heart, a typical RDBMS has five main components, as illustrated in Figure 1.1. As an introduction to each of these components and the way they fit together, we step through the life of a query in a database system. This also serves as an overview of the remaining sections of the paper.

Consider a simple but typical database interaction at an airport, in which a gate agent clicks on a form to request the passenger list for a flight. This button click results in a single-query transaction that works roughly as follows:

1. The personal computer at the airport gate (the “client”) calls an API that in turn communicates over a network to establish a connection with the Client Communications Manager of a DBMS (top of Figure 1.1). In some cases, this connection involves the Catalog Manager, Memory Manager, Administration, Monitoring & Utilities, Replication and Loading Services, Batch Utilities, and Shared Components and Utilities (Section 7).

Relational Database Architecture

[Hellerstein et al., Architecture of a Database System]
How to Scale Relational Databases?
Parallel Architecture: Processes and Memory Coordination

3.3 Shared-Disk

A shared-disk parallel system (Figure 3.3) is one in which all processors can access the disks with about the same performance, but are unable to access each other’s RAM. This architecture is quite common with two prominent examples being Oracle RAC and DB2 for zSeries. Shared-disk has become more common in recent years with the increasing popularity of Storage Area Networks (SAN). A SAN allows one or more logical disks to be mounted by one or more host systems making it easy to create shared disk configurations.

One potential advantage of shared-disk over shared-nothing systems is their lower cost of administration. DBAs of shared-disk systems do not have to consider partitioning tables across machines in order to achieve parallelism. But very large databases still typically do require partitioning so, at this scale, the difference becomes less pronounced.

Another compelling feature of the shared-disk architecture is that the failure of a single DBMS processing node does not affect the other nodes’ ability to access the entire database. This is in contrast to both shared-memory systems that fail as a unit, and shared-nothing systems that lose access to at least some data upon a node failure (unless some alternative data redundancy scheme is used). However, even with these advantages, shared-disk systems are still vulnerable to some single...
Parallel Architecture: Processes and Memory Coordination

3.3 Shared-Disk

A shared-disk parallel system (Figure 3.3) is one in which all processors can access the disks with about the same performance, but are unable to access each other's RAM. This architecture is quite common with two prominent examples being Oracle RAC and DB2 for zSeries SYSPLEX. Shared-disk has become more common in recent years with the increasing popularity of Storage Area Networks (SAN). A SAN allows one or more logical disks to be mounted by one or more host systems making it easy to create shared disk configurations.

One potential advantage of shared-disk over shared-nothing systems is their lower cost of administration. DBAs of shared-disk systems do not have to consider partitioning tables across machines in order to achieve parallelism. But very large databases still typically do require partitioning so, at this scale, the difference becomes less pronounced.

Another compelling feature of the shared-disk architecture is that the failure of a single DBMS processing node does not affect the other nodes' ability to access the entire database. This is in contrast to both shared-memory systems that fail as a unit, and shared-nothing systems that lose access to at least some data upon a node failure (unless some alternative data redundancy scheme is used). However, even with these advantages, shared-disk systems are still vulnerable to some single failure.

[Figure 3.3 Shared-disk architecture.]

[Hellerstein et al., Architecture of a Database System]
Parallel Architecture: Processes and Memory Coordination

Fig. 3.1 Shared-memory architecture.

Buying a smaller number of large, very expensive systems is sometimes viewed to be an acceptable trade-off.

Multi-core processors support multiple processing cores on a single chip and share some infrastructure such as caches and the memory bus. This makes them quite similar to a shared-memory architecture in terms of their programming model. Today, nearly all serious database deployments involve multiple processors, with each processor having more than one CPU. DBMS architectures need to be able to fully exploit this potential parallelism. Fortunately, all three of the DBMS architectures described in Section 2 run well on modern shared-memory hardware architectures.

The process model for shared-memory machines follows quite naturally from the uniprocessor approach. In fact, most database systems evolved from their initial uniprocessor implementations to shared-memory implementations. On shared-memory machines, the OS typically supports the transparent assignment of workers (processes or threads).

1 The dominant cost for DBMS customers is typically paying qualified people to administer high-end systems. This includes Database Administrators (DBAs) who configure and maintain the DBMS, and System Administrators who configure and maintain the hardware and operating systems.

Parallel DB Architecture: Shared Memory

[Petero et al., Architecture of a Database System]
TrafficDB: Shared-Memory Data Store

- Traffic-aware route planning
- Want up-to-date data for all
- Thousands of requests per second
  - High-Frequency Reads
  - Low-Frequency Writes
- "Data must be stored in a region of RAM that can be shared and efficiently accessed by several different application processes"

[R. Fernandes et al., 2016]
3.2 Shared-Nothing

A shared-nothing parallel system (Figure 3.2) is made up of a cluster of independent machines that communicate over a high-speed network interconnect or, increasingly frequently, over commodity networking components. There is no way for a given system to directly access the memory or disk of another system.

Shared-nothing systems provide no hardware sharing abstractions, leaving coordination of the various machines entirely in the hands of the DBMS. The most common technique employed by DBMSs to support these clusters is to run their standard process model on each machine, or node, in the cluster. Each node is capable of accepting client SQL requests, but the data remains independent across nodes.

Fig. 3.2 Shared-nothing architecture.

[Parallel DB Architecture: Shared Nothing]

[Hellerstein et al., Architecture of a Database System]
Sharding
Relational Databases: One size fits all?

- Lots of work goes into relational database development:
  - B-trees
  - Cost-based query optimizers
  - ACID (Atomicity, Consistency, Isolation, Durability)
- Vendors have stuck with this model since the 1980s
- Having different systems leads to business problems:
  - cost problem
  - compatibility problem
  - sales problem
  - marketing problem

[Stonebraker and Çetinetmel, 2005]
ACID Transactions

- Make sure that transactions are processed reliably
- Atomicity: leave the database as is if some part of the transaction fails (e.g. don't add/remove only part of the data) using rollbacks
- Consistency: database moves from one valid state to another
- Isolation: concurrent execution matches serial execution
- Durability: endure hardware failures, make sure changes hit disk
Stonebraker: The End of an Architectural Era

- "RDBMSs were designed for the business data processing market, which is their sweet spot"
- "They can be beaten handily in most any other market of significant enough size to warrant the investment in a specialized engine"
- Changes in markets (science), necessary features (scalability), and technology (amount of memory)
- RDBMS Overhead: Logging, Latching, and Locking
- Relational model is not necessarily the answer
- SQL is not necessarily the answer
### Row Stores

<table>
<thead>
<tr>
<th>id</th>
<th>scientist</th>
<th>death_by</th>
<th>movie_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reinhardt</td>
<td>Crew</td>
<td>The Black Hole</td>
</tr>
<tr>
<td>2</td>
<td>Tyrell</td>
<td>Roy Batty</td>
<td>Blade Runner</td>
</tr>
<tr>
<td>3</td>
<td>Hammond</td>
<td>Dinosaur</td>
<td>Jurassic Park</td>
</tr>
<tr>
<td>4</td>
<td>Soong</td>
<td>Lore</td>
<td>Star Trek: TNG</td>
</tr>
<tr>
<td>5</td>
<td>Morbius</td>
<td>The machine</td>
<td>Forbidden Planet</td>
</tr>
<tr>
<td>6</td>
<td>Dyson</td>
<td>SWAT</td>
<td>Terminator 2: Judgment Day</td>
</tr>
</tbody>
</table>

**Primary Key**

---

J. Swanhart, *Introduction to Column Stores*
OLTP vs. OLAP

• Online Transactional Processing (OLTP) often used in business applications, data entry and retrieval transactions

• OLTP Examples:
  - Add customer's shopping cart to the database of orders
  - Find me all information about John Hammond's death

• OLTP is focused on the day-to-day operations while Online Analytical Processing (OLAP) is focused on analyzing that data for trends, etc.

• OLAP Examples:
  - Find the average amount spent by each customer
  - Find which year had the most movies with scientists dying
Inefficiency in Row Stores for OLAP

```
select sum(metric) as the_sum from fact
```

1. Storage engine gets a whole row from the table

2. SQL interface extracts only requested portion, adds it to “the_sum”

3. IF all rows scanned, send results to client, else GOTO 1

[J. Swanhart, Introduction to Column Stores]
## Column Stores

<table>
<thead>
<tr>
<th>id</th>
<th>Title</th>
<th>Person</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mrs. Doubtfire</td>
<td>Robin Williams</td>
<td>Comedy</td>
</tr>
<tr>
<td>2</td>
<td>Jaws</td>
<td>Roy Scheider</td>
<td>Horror</td>
</tr>
<tr>
<td>3</td>
<td>The Fly</td>
<td>Jeff Goldblum</td>
<td>Horror</td>
</tr>
<tr>
<td>4</td>
<td>Steel Magnolias</td>
<td>Dolly Parton</td>
<td>Drama</td>
</tr>
<tr>
<td>5</td>
<td>The Birdcage</td>
<td>Nathan Lane</td>
<td>Comedy</td>
</tr>
<tr>
<td>6</td>
<td>Erin Brokovich</td>
<td>Julia Roberts</td>
<td>Drama</td>
</tr>
</tbody>
</table>

Each column has a file or segment on disk

[J. Swanhart, *Introduction to Column Stores*]
### Horizontal Partitioning vs. Vertical Partitioning

**Original Table**

<table>
<thead>
<tr>
<th>CUSTOMER ID</th>
<th>FIRST NAME</th>
<th>LAST NAME</th>
<th>FAVORITE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TAEKO</td>
<td>OHNUKI</td>
<td>BLUE</td>
</tr>
<tr>
<td>2</td>
<td>O.V.</td>
<td>WRIGHT</td>
<td>GREEN</td>
</tr>
<tr>
<td>3</td>
<td>SELDA</td>
<td>BAĞCAN</td>
<td>PURPLE</td>
</tr>
<tr>
<td>4</td>
<td>JIM</td>
<td>PEPPER</td>
<td>AUBERGINE</td>
</tr>
</tbody>
</table>
Horizontal Partitioning vs. Vertical Partitioning

### Vertical Partitions

<table>
<thead>
<tr>
<th>VP1</th>
<th>VP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER ID</td>
<td>FIRST NAME</td>
</tr>
<tr>
<td>1</td>
<td>TAEXKO</td>
</tr>
<tr>
<td>2</td>
<td>O.V.</td>
</tr>
<tr>
<td>3</td>
<td>SELDA</td>
</tr>
<tr>
<td>4</td>
<td>JIM</td>
</tr>
</tbody>
</table>

### Horizontal Partitions

<table>
<thead>
<tr>
<th>HP1</th>
<th>HP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER ID</td>
<td>FIRST NAME</td>
</tr>
<tr>
<td>1</td>
<td>TAEXKO</td>
</tr>
<tr>
<td>2</td>
<td>O.V.</td>
</tr>
<tr>
<td>3</td>
<td>SELDA</td>
</tr>
<tr>
<td>4</td>
<td>JIM</td>
</tr>
</tbody>
</table>

Original Table:

<table>
<thead>
<tr>
<th>CUSTOMER ID</th>
<th>FIRST NAME</th>
<th>LAST NAME</th>
<th>FAVORITE COLOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TAEXKO</td>
<td>OHNUKI</td>
<td>BLUE</td>
</tr>
<tr>
<td>2</td>
<td>O.V.</td>
<td>WRIGHT</td>
<td>GREEN</td>
</tr>
<tr>
<td>3</td>
<td>SELDA</td>
<td>BAĞCAN</td>
<td>PURPLE</td>
</tr>
<tr>
<td>4</td>
<td>JIM</td>
<td>PEPPER</td>
<td>AUBERGINE</td>
</tr>
</tbody>
</table>
Problems with Relational Databases

ID: 1001
Customer: Ann

Line Items:
- 0321293533 | 2 | $48 | $96
- 0321601912 | 1 | $39 | $39
- 0131495054 | 1 | $51 | $51

Payment Details:
Card: Amex
CC Number: 12345
Expiry: 04/2001

Orders
Customers
Order Lines
Credit Cards

[P. Sadalage]
NoSQL: Key-Value Databases

- Always use primary-key access
- Operations:
  - Get/put value for key
  - Delete key
- Examples
  - Memcached
  - Amazon DynamoDB
  - Project Voldemort
  - Couchbase
NoSQL: Document Databases

- Documents are the main entity
  - Self-describing
  - Hierarchical
  - Do not have to be the same
- Could be XML, JSON, etc.
- Key-value stores where values are "examinable"
- Can have query language and indices overlaid
- Examples: MongoDB, CouchDB, Terrastore

```json
{  
  "customerid": "fc986e48ca6",  
  "customer": {  
    "firstname": "Pramod",  
    "lastname": "Sadalage",  
    "company": "ThoughtWorks",  
    "likes": [ "Biking", "Photography" ]  
  }  
  "billingaddress": {  
    "state": "AK",  
    "city": "DILLINGHAM",  
    "type": "R"  
  }  
}
```

[P. Sadalage]
NoSQL: Column Stores

• Instead of having rows grouped/sharded, we group columns
• …or families of columns
• Put similar columns together
• Examples: Cassandra, HBase
NoSQL: Graph Databases

- Focus on entities and relationships
- Edges may have properties
- Relational databases required a set traversal
- Traversals in Graph DBs are faster
- Examples:
  - Neo4j
  - Pregel
Distributing Data

• Aggregate-oriented databases

• Sharding (horizontal partitioning): Sharding distributes different data across multiple servers, so each server acts as the single source for a subset of data

• Replication: Replication copies data across multiple servers, so each bit of data can be found in multiple places. Replication comes in two forms,
  - Source-replica replication makes one node the authoritative copy that handles writes, replica synchronizes with the source and may handle reads.
  - Peer-to-peer replication allows writes to any node; the nodes coordinate to synchronize their copies of the data.
CAP Theorem
CAP Theorem

- Consistency: every read would get you the most recent write
- Availability: every node (if not failed) always executes queries
- Partition tolerance: system continues to work even if nodes are down
- Theorem (Brewer): It is impossible for a distributed data store to simultaneously provide more than two of Consistency, Availability, and Partition Tolerance
Think about RDBMS Transactions…
Cassandra:
A Decentralized Structured Storage System

A. Lakshman and P. Malik
What is Cassandra?

- Fast Distributed (Column Family NoSQL) Database
  - High availability
  - Linear Scalability
  - High Performance
- Fault tolerant on Commodity Hardware
- Multi-Data Center Support
- Easy to operate
- Proven: CERN, Netflix, eBay, GitHub, Instagram, Reddit
Cassandra and CAP

RDBMS

Availability

Consistency (ACID)

Partition Tolerance

Atomicity
Consistency
Isolation
Durability
Cassandra: Ring for High Availability

No master, no slave, peer to peer gossip. I'm online! No, no... This isn't gossip. It's the truth.
Slides: Introduction to Cassandra

Robert Stupp
Next Class's Reading Response

• **Spanner: Google's Globally-Distributed Database**
• Reading Response for Monday:
  - Focus on main concepts in the paper
  - Submit to Blackboard