

# Data Storage

Philip J. Cwynar

University of Pittsburgh  
School of Information Sciences  
[pcwynar@pitt.edu](mailto:pcwynar@pitt.edu)

DSL

Data Storage

## Outline

- Collecting data
- Relational Databases
- NoSQL Databases
  - Key-value databases
  - Document databases
  - Column-family stores
  - Graph databases
- Beyond NoSQL Databases

DSL

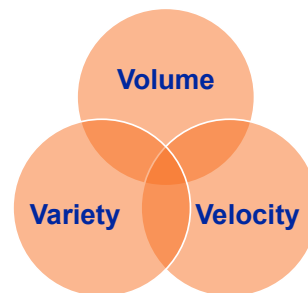
Data Storage

## Collecting Data

DSL

Data Storage

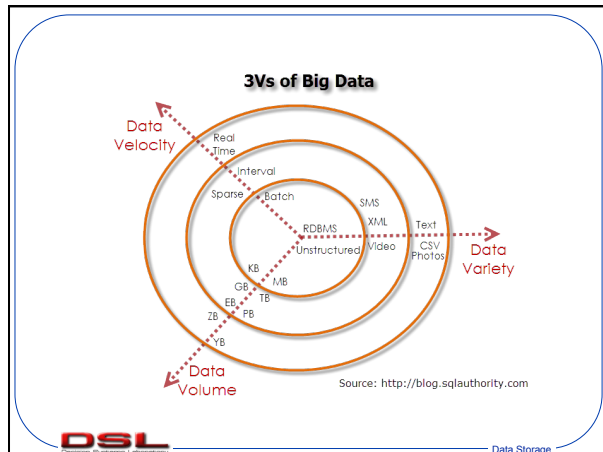
## Big Data's 3V



- Collecting data
- Relational Databases
- NoSQL Databases
- Beyond NoSQL Databases

DSL

Data Storage



### The quantity of data collected

- The New York Stock Exchange generates about one terabyte of new trade data per day.
- Facebook hosts approximately 10 billion photos, taking up one petabyte of storage.
- Ancestry.com, the genealogy site, stores around 2.5 petabytes ( $10^{15}$ ) of data.
- The Internet Archive stores around 2 petabytes ( $10^{15}$ ) of data, and is growing at a rate of 20 terabytes ( $10^{12}$ ) per month.
- The Large Hadron Collider near Geneva, Switzerland, will produce about 15 petabytes ( $10^{15}$ ) of data per year.

Reference: Tom White, Hadoop: The Definitive Guide, Third Edition, 2012

**DSL**  
Decision Systems Laboratory

Data Storage

## Structured Data

I suspect the term SD came from the name for a common language used to access DBs, called Structured Query Language, or [SQL](#).

SQL provides a well-defined way for applications to manage data in a DB.

The little snippet of SQL code here illustrates how an application could retrieve all rows from a table called Book where the Price is greater than 100 and request that the result be sorted in ascending order by title.

```
SELECT *
FROM Book
WHERE price > 100
ORDER BY title;
```

**DSL**  
Decision Systems Laboratory

Data Storage

## Storing Data: Relational Databases

**DSL**  
Decision Systems Laboratory

Data Storage

## Relational databases: Definition

- **Relational database:** A set of relations
- **Relation:** Made up of 2 parts:
  - **Schema:** specifies name of relation plus name and type of each column  
Customer(id:int, name:string, gender:string, email:string)
  - **Instance:** a table, with rows and columns

id	name	gender	email
1	Philip	Male	pcwynar@pitt.edu
2	Chirayu	Male	chw@pitt.edu

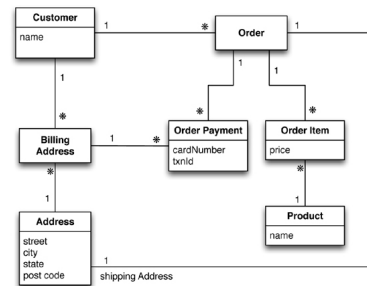
- Can think of a relation as a set of rows (tuples)

Reference for this section: NoSQL distilled: A brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL

Data Storage

## Relational databases: Data Model Entity Relationship Diagram



Data model oriented around a relational database (using UML notation)

DSL

Data Storage

## Relational databases: Data Model

Customer	Orders
Id Name	Id CustomerId ShippingAddressId
1 Martin	99 1 77

Product	BillingAddress
Id Name	Id CustomerId AddressId
27 NoSQL Distilled	55 1 77

OrderItem	Address
Id OrderId ProductId Price	Id City
100 99 27 32.45	77 Chicago

OrderPayment
Id OrderId CardNumber BillingAddressId txnId
33 99 1000-1000 55 abeli979rft

Typical data using RDBMS data model

DSL

Data Storage

## Relational databases: Integrity constraints (ICs)

- **IC:** condition that must be true for any instance of the database, e.g., domain constraints
  - ICs are specified when schema is defined
  - ICs are checked when relations are modified
- A legal instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances

DSL

Data Storage

## Relational databases: Primary key constraints

- A set of fields is a key for a relation if:
  1. No two distinct tuples can have same value in all key fields, and
  2. This is not true for any subset of the key
    - » If part 2 is false, then it is super key.
- K is a candidate key if K is minimal.
- Among all candidate keys we must select one that becomes the Primary Key

## Relational databases: Foreign key constraints

- Foreign key: Set of fields in one relation that is used to refer to a tuple in another relation (must correspond to a primary key of the second relation)
- If all foreign key constraints are enforced, referential integrity is achieved

Customer		Orders		
Id	Name	Id	CustomerId	ShippingAddressId
1	Martin	99	1	77

## Relational databases: Normalization

- **1<sup>st</sup> Normal Form**: make tables flat (all elements atomic)
- **2<sup>nd</sup> Normal Form**: Every non-prime attribute depends on a candidate key or another non-prime attribute
- **3<sup>rd</sup> Normal Form**: some redundancy but dependency preserving
- **BCNF** (Boyce Codd Normal Form): no redundancy but not dependency preserving
- ...
- Redundancy might lead to update anomalies
- Note: we are going to break normalization (denormalize) later

## Relational databases: Query language

- A major strength of the relational model:
  - Supports simple, powerful querying of data (SQL)
    - » **DDL** (Data Description Language): create, drop, alter
    - » **DML** (Data Manipulation Language): insert, update, delete, select
    - » **DCL** (Data Control Language): grant, revoke
    - » **TCL** (Transaction Control Language): commit, rollback
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation
  - The key: precise semantics for relational queries
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.

## Relational databases: Example

Genres		movie	genre
ID	Name	13519	1
12	Action	12056	1
1	Adult	12429	2
16	Adventure	6745	3
10	Animation	6745	4
22	Biography	12540	5
6	Comedy	12540	4
13	Crime	15399	6
7	Documentary	5183	7
2	Drama	15760	3
11	Famly	8769	7
14	Fantasy	8769	7
24	Film-Noir	8769	8
25	Game-Show	8769	9
8	History	2945	6
		2945	4

[illegible]

- Collecting data
- Relational Databases
- NoSQL Databases
- Beyond NoSQL Databases

- Data Storage

## Relational databases: Query Example

Year	Title	0.00741
1993	North by Northwest	0.00741
1994	Star Trek: The Next Generation: Season 1	0.00741
1995	Seven Sevens	0.00633
1996	Road to Damascus	0.02770
1997	Allex: Season 1	0.04811
1998	Samurai Champloo	0.08241
1999	Star Trek: The Next Generation: Season 5	0.08241
2000	Care Closed: Season 2	0.08241
2001	Ten Ties: Season 2	0.08373
2002	The Last Great Generation: Season 4	0.08373
2003	Star Trek: The Next Generation: Season 6	0.08373
2004	Star Trek: Season 2	0.08373
2005	Batman Begins	0.02350
2006	Fanfiction: The Postscripter Writs	0.02350
2007	Fanfiction: The Postscripter Writs	0.02350
2008	Allex: Season 2	0.08373
2009	Star Trek: The Next Generation: Season 4	0.08373
2010	Allex: Season 3	0.04901
2011	Smulfin' Season 2	0.03986
2012	The Pinocchio Bride	0.03986
2013	Batfield Gals: Season 1	0.03986
2014	Smulfin' Season 1	0.03986
2015	Readers of the Lost Ark	0.03986
2016	Smulfin' Season 3	0.01962
2017	The Incubator	0.04773
2018	The Great Escape	0.01621
2019	Happy Potter: The Prince and the Prisoner of Azkaban	0.00737
2020	Smulfin' Season 4	0.04422
2021	Toy Story	0.00611
2022	Gleadow	0.00602
2023	Toy Story 2	0.00611
2024	Star Trek: The Next Generation: Season 5	0.02409
2025	Smulfin' Season 5	0.07419
2026	Indiana Jones and the Last Crusade	0.02409

- Collecting data
  - Relational Databases
  - NoSQL Databases
  - Beyond NoSQL Databases

- Data Storage

## Relational databases: ACID transactions

- **Transaction** is a unit of work performed within a database management system against a database
- **Atomicity** – all or nothing
- **Consistency** – bring database from one valid state to another
- **Isolation** - the intermediate state of a transaction is invisible to other transactions
- **Durability** - after a transaction successfully completes, changes to data persist and are not undone, even in the event of a system failure.

- Collecting data
- Relational Databases
- NoSQL Databases
- Beyond NoSQL Databases

- Data Storage

## Structured Data Warehousing Approaches

## Bill Inmon – “Father of Data Warehousing”

----Enterprise Data Warehouse

## Ralph Kimball – “Father of Business Intelligence”

---Data Mart / Star Schema



- Data Storage



### Star Schema – Data Mart Design Ralph Kimball

The star schema architecture is the simplest [data](#) warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from a center.

The center of the star consists of fact [table](#) and the points of the star are the dimension tables.

Usually the fact tables in a star schema are in third normal form(3NF) whereas dimensional tables are de-normalized.

Despite the fact that the star schema is the simplest architecture, it is most commonly used nowadays and is recommended by [Oracle](#).



Data Storage

### Pros and cons of both the approaches

	Inmon	Kimball
Building Data warehouse	Time Consuming	Takes lesser time
Maintenance	Easy	Difficult, often redundant and subject to revisions
Cost	High initial cost. Subsequent project development costs will be much lower	Low initial cost. Each subsequent phase will cost almost the same
Time	Longer start-up time	Shorter time for initial set-up
Skill Requirement	Specialist team	Generalist team
Data Integration requirements	Enterprise-wide	Individual business areas



Data Storage

### Storing Data: NoSQL Databases



Data Storage

### Unstructured Data Types

Here is a limited list of types of unstructured data:

- Emails
- Word Processing Files
- PDF files
- Spreadsheets
- Digital Images
- Video
- Audio
- Social Media Posts



Data Storage

## NoSQL databases

### NoSQL databases:

- Not using the relational model
- Schemaless
- Running well on clusters
- Tend to be open-source
- List of NoSQL databases (more than 150!):  
<http://nosql-database.org/>

## Why are NoSQL databases interesting?

### Two primary reasons:

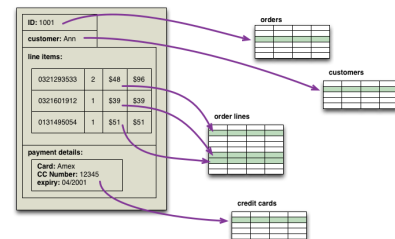
- Application development productivity
- Large-scale data

## NoSQL databases: Data models

- **Key-value** databases:
  - BerkeleyDB, LevelDB, Memcached, Project Voldemort, Redis, Riak, ...
- **Document** databases:
  - CouchDB, MongoDB, OrientDB, RavenDB, Terrastore, ...
- **Column-family** stores:
  - Amazon SimpleDB, Cassandra, Hbase, Hypertable, ...
- **Graph** databases:
  - FlockDB, HyperGraphDB, Infinite Graph, Neo4J, OrientDB, ...
- Other

[http://en.wikipedia.org/wiki/Data\\_warehouse](http://en.wikipedia.org/wiki/Data_warehouse)

## Aggregate data model: Example



- DDD: Domain-Driven Design
- Apparently easier to program
- Definitely, makes it easier to store and process data on multiple computer clusters.

<http://martinfowler.com/bliki/AggregateOrientedDatabase.html>



Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

### Aggregate data model

```

// in customers
{
  "id": 1,
  "name": "Martin",
  "billingAddress": [{"city": "Chicago"}]
}
// in orders
{
  "id": 99,
  "customerid": 1,
  "orderItems": [
    {
      "productid": 27,
      "price": 32.45,
      "productName": "NoSQL Distilled"
    }
  ],
  "shippingAddress": [{"city": "Chicago"}],
  "orderPayment": [
    {
      "ccinfo": "1000-1000-1000-1000",
      "txnId": "label#79rt",
      "billingAddress": [{"city": "Chicago"}]
    }
  ]
}
  
```

```

graph TD
    Customer -- "billing Address" --> Address
    Customer -- "1" --> Order
    Address -- "shipping Address" --> Order
    Address -- "1" --> Address
    Order -- "1" --> OrderItem
    Order -- "1" --> Payment
    OrderItem -- "1" --> Product
    Payment -- "1" --> Product
  
```

**An aggregate data model**

<http://martinfowler.com/bliki/AggregateOrientedDatabase.html>  
 NoSQL distilled : a brief guide to the emerging world of  
 polyglot persistence / Pramod J Sadalage, Martin Fowler.

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

### Aggregate data model

```

// in customers
{
  "customer": {
    "id": 1,
    "name": "Martin",
    "billingAddress": [{"city": "Chicago"}],
    "orders": [
      {
        "id": 99,
        "customerid": 1,
        "orderItems": [
          {
            "productid": 27,
            "price": 32.45,
            "productName": "NoSQL Distilled"
          }
        ],
        "shippingAddress": [{"city": "Chicago"}],
        "orderPayment": [
          {
            "ccinfo": "1000-1000-1000-1000",
            "txnId": "label#79rt",
            "billingAddress": [{"city": "Chicago"}]
          }
        ]
      }
    ]
  }
}
  
```

```

graph TD
    Customer -- "billing Address" --> Address
    Customer -- "1" --> Order
    Address -- "shipping Address" --> Order
    Address -- "1" --> Address
    Order -- "1" --> OrderItem
    Order -- "1" --> Payment
    OrderItem -- "1" --> Product
    Payment -- "1" --> Product
  
```

**An aggregate data model**

<http://martinfowler.com/bliki/AggregateOrientedDatabase.html>  
 NoSQL distilled : a brief guide to the emerging world of  
 polyglot persistence / Pramod J Sadalage, Martin Fowler.

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

### Aggregate data model: Pros and cons

- Cons:
  - It is often difficult to draw aggregate boundaries well
  - Does not support ACID transactions (thus sacrifices consistency)
  - Some queries are easier (to the point of being practical) but others may be really hard (e.g., to get to product sales history, you'll have to dig into every aggregate in the database).
- Pros:
  - Helps greatly with running on a cluster: This is the main argument for NoSQL databases.

<http://blog.dynatrace.com/2011/10/05/noSQL-or-rdbms-are-we-asking-the-right-questions/>

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

### Aggregate data model: Key points

- An aggregate is a collection of data that we interact with as a unit.
- Key-value, document, and column-family databases can all be seen as forms of aggregate-oriented database.
- Aggregates make it easier for the database to manage data storage over clusters.

NoSQL distilled : a brief guide to the emerging world of  
 polyglot persistence / Pramod J Sadalage, Martin Fowler.

Data Storage

## Storing Data: NoSQL Database: Key-Value Databases

### Key-value databases

- A key-value store is a simple hash table
  - Get the value for the key
  - Put a value for a key
  - Delete a key from the data store
- The value is a blob
- You can think of such databases as databases with only one table, which has two columns: ID and Value.

### Key-value databases: Riak



#### Terminology comparison:

RDBMS	Riak
Database	Riak cluster
Table	Bucket
Row	Key-value
RowID (at least in Oracle)	key

1,000s of startups, enterprises, and organizations have deployed Riak for their production systems.



References: <http://wiki.basho.com/Riak.html>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

### Key-value databases: Riak



- Writing to the Riak bucket using the *store* API:  
 Bucket bucket = getBucket(bucketName);  
 IRiakObject riakObject = bucket.store(key, value).execute();
- Getting value for the key using *fetch* API:  
 Bucket bucket = getBucket(bucketName);  
 IRiakObject riakObject = bucket.fetch(key).execute();  
 byte[] bytes = riakObject.getValue();  
 String value = new String(bytes);

## Key-value databases: Riak



Riak provides an HTTP-based interface (this allows all operations to be performed from a web browser or command line):

```
- curl -X PUT HTTP://127.0.0.1:8098/riak/images/1.jpg -H "Content-type: image/jpeg" --data-binary @images.jpg
- curl -i HTTP://127.0.0.1:8098/riak/images/1.jpg
- curl -v -X POST -d '{
  "lastVisit":1324669989288,
  "user":{"customerId":"1",
  "name":"buyer",
  "countryCode":"US",
  "tzOffset":0}
}' -H "Content-Type:application/json" http://127.0.0.1:8098/riak/test/1
- curl -i HTTP://127.0.0.1:8098/riak/test/1
```

References: <http://wiki.basho.com/Riak.html>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL

Data Storage

## Key-value databases: Usage

- When to use:
  - Storing session information
  - User profiles, preferences
  - Shopping cart data
- When not to use
  - Relationships among data
  - Multi-operation transactions
  - Query by data
  - Operations by sets

References: <http://wiki.basho.com/Riak.html>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL

Data Storage

## Storing Data: NoSQL Database: Document Databases

DSL

Data Storage

## Document databases

- Documents are the main concept here
  - DB stores and retrieves documents
- Documents stored are similar to each other but do not have to be exactly the same
  - Schemaless
- Documents are stored in the value part of the key-value store

NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL

Data Storage


Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Document databases: What is document?

- One document:
  - { "firstname": "Martin",  
"likes": [ "Biking", "Photography" ],  
"lastcity": "Boston",  
"lastVisited":  
}
- Another document:
  - { "firstname": "Pramod",  
"citiesvisited": [ "Chicago",  
"London", "Pune", "Bangalore" ],  
"addresses": [ { "state": "AK",  
"city": "DILLINGHAM",  
"type": "R" },  
{ "state": "MH",  
"city": "PUNE",  
"type": "R" } ],  
"lastcity": "Chicago" }


Schema can differ significantly among documents in the same database.

This was not possible in RDBMS databases.


NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage





Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Document databases: MongoDB




### Terminology comparison:

RDBMS	MongoDB
Database	MongoDB
Table	Collection
Row	Document
Rowid (at least Oracle)	_id
Join	DBRef







References: <http://www.mongodb.org/>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.


Data Storage


Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Document databases: MongoDB




- [DEMO] - <http://www.mongodb.org/#>
- Save document to MongoDB:
  - db.collectionName.insert({firstname: 'Martin', likes: ['Biking', 'Photography'], lastcity: 'Boston', lastVisited: ''});
- Save another document to MongoDB:
  - db.collectionName.insert({firstname: 'Pramod', citiesvisited: ['Chicago', 'London', 'Pune', 'Bangalore'], addresses: [{ state: 'AK', city: 'DILLINGHAM', type: 'R' }, { state: 'MH', city: 'PUNE', type: 'R' }], lastcity: 'Chicago'});

References: <http://www.mongodb.org/>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.


Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Document databases: MongoDB




[DEMO] - <http://www.mongodb.org/#>

### Querying:


- All document in docName colleciton:
  - db.collectionName.find();
  - SQL: select \* from docName
- Documents which satisfy a condition:
  - db.collectionName.find({firstname:"Martin"});
  - Equivalent to SQL query:  
select \* from collectionName where firstname = "Martin"

References: <http://www.mongodb.org/>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.


Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Document databases: MongoDB



- SQL:
  - SELECT \* FROM customerOrder, orderItem, product WHERE customerOrder.orderId = orderItem.customerOrderId AND orderItem.productId = product.productId AND product.name LIKE '%Big Data%'
- MongoDB:
  - db.collectionName.find({"orders.productName": "/Big Data/});

References: <http://www.mongodb.org/>  
NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL  
Decision Systems Laboratory

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Document databases: Usage

- When to Use:
  - Event Logging
  - Content Management Systems, Blogging Platforms
  - Web Analytics or Real-Time Analytics
  - E-Commerce Applications
- When Not to Use
  - Complex Transactions Spanning Different Operations
  - Queries against Varying Aggregate Structure

NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL  
Decision Systems Laboratory

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Key-value vs. document data models

	Key-Value Model	Document Model
Aggregate	Opaque	Transparent
Access	Only lookup based on key (return whole aggregation)	Queries based on fields in the aggregate (can retrieve parts)
Index	-	Can create indexes based on the contents of the aggregate

<http://blog.dynatrace.com/2011/10/05/noSQL-or-columns-are-we-asking-the-right-questions/>

DSL  
Decision Systems Laboratory

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Storing Data: NoSQL Database: Column-Family Stores

DSL  
Decision Systems Laboratory

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Column-oriented data model

Id	First Name	Last Name	Salary
1	Joe	Smith	40000
2	Mary	Jones	50000
3	Mike	Johnson	45000

1, Joe, Smith, 40000  
2, Mary, Jones, 50000  
3, Mike, Johnson, 45000

1, 2, 3  
Joe, Mary, Mike  
40000,50000,45000

DSL  
Decision Systems Laboratory

[http://en.wikipedia.org/wiki/Column-oriented\\_DBMS](http://en.wikipedia.org/wiki/Column-oriented_DBMS)

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Column-oriented data model

- Column-oriented organizations are more efficient when an aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data.
- Column-oriented organizations are more efficient when new values of a column are supplied for all rows at once, because that column data can be written efficiently and replace old column data without touching any other columns for the rows.
- Row-oriented organizations are more efficient when many columns of a single row are required at the same time, and when row-size is relatively small, as the entire row can be retrieved with a single disk seek.
- Row-oriented organizations are more efficient when writing a new row if all of the column data are supplied at the same time, as the entire row can be written with a single disk seek.

DSL  
Decision Systems Laboratory

[http://en.wikipedia.org/wiki/Column-oriented\\_DBMS](http://en.wikipedia.org/wiki/Column-oriented_DBMS)

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Column-oriented data model

- Data indexed by:
  - (row:string, column:string, time:int64) → string
- #of distinct column families - small (in hundreds)
- unbounded number of columns
- Data processing is pushed to the application

DSL  
Decision Systems Laboratory

NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

Column-family databases

Conceptual View:

Row Key	Time Stamp	Column Family contents	Column Family anchor
"com.cnn.www"	19		anchor.cnnsi.com="CNN"
"com.cnn.www"	18		anchor.my.look.ca="CNN.com"
"com.cnn.www"	16	contents.html="<html>..."	
"com.cnn.www"	15	contents.html="<html>..."	
"com.cnn.www"	13	contents.html="<html>..."	

Physical View

Column Family anchor

Row Key	Time Stamp	Column Family anchor
"com.cnn.www"	19	anchor.cnnsi.com="CNN"
"com.cnn.www"	18	anchor.my.look.ca="CNN.com"

Column Family contents

Row Key	Time Stamp	Column Family contents
"com.cnn.www"	16	contents.html="<html>..."
"com.cnn.www"	15	contents.html="<html>..."
"com.cnn.www"	13	contents.html="<html>..."

DSL  
Decision Systems Laboratory




<http://hbase.apache.org/book.html#datamodel>

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

## Column-family databases: HBase

- HBase is the Hadoop database
  - Distributed, scalable, big data store
- Use HBase when you need random, real-time read/write access to your Big Data
- Goal is to host very large tables:
  - billions of rows X millions of columns
- HBase is an open-source, distributed, versioned, column-oriented store modeled after Google's Bigtable






<http://hbase.apache.org/>  
DSL Decision Systems Laboratory Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

## Column-family databases: HBase

- No SQL-like query language
  - Java API
  - HBase Shell
    - » create 'test', 'cf'
    - » put 'test', 'row1', 'cf:a', 'value1'
    - » put 'test', 'row2', 'cf:b', 'value2'
    - » get 'test', 'row1'
      - COLUMN CELL
      - cf:a timestamp=1288380727188, value=value1
- HBase – separate project to simplify the usage of Hbase (hbase.com)



[http://hbase.apache.org/book/quickstart.html#shell\\_exercises](http://hbase.apache.org/book/quickstart.html#shell_exercises)  
DSL Decision Systems Laboratory Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

## Column-family databases: Usage

- When to use:
  - Event logging
  - Content management systems, blogging platforms
  - Expiring usage
  - Need aggregate using, e.g., SUM or AVG
- When not to use
  - Frequent changes to the database (inserts and deletes maybe expensive)

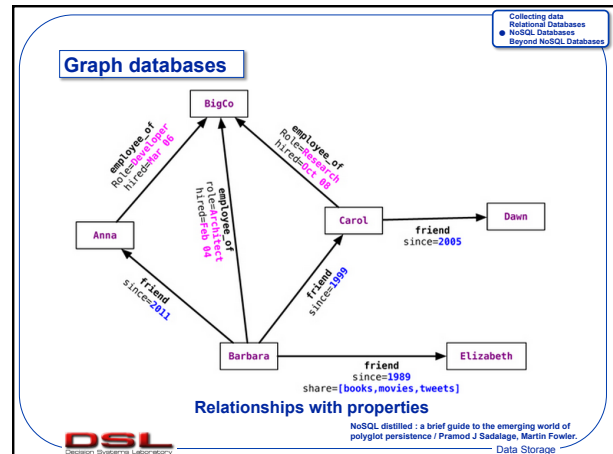
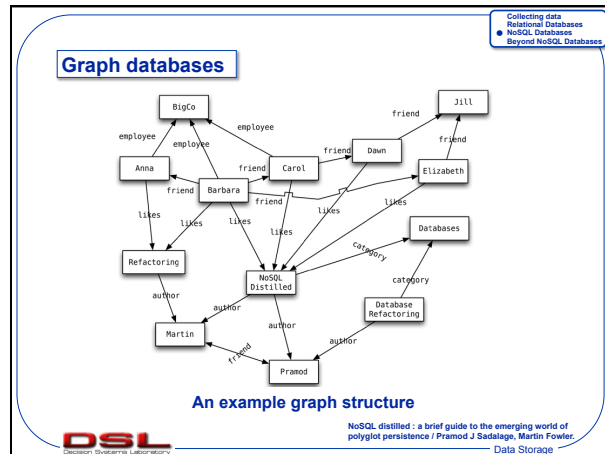
References: <http://wiki.hbase.com/Riak.html>;  
 NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

DSL Decision Systems Laboratory Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

## Storing Data: NoSQL Database: Graph Databases

DSL Decision Systems Laboratory Data Storage



Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

### Graph Databases: Neo4j

- Node creation:
  - Node martin = graphDb.createNode();
  - martin.setProperty("name", "Martin");
  - Node pramod = graphDb.createNode();
  - pramod.setProperty("name", "Pramod");
- Relationship creation:
  - martin.createRelationshipTo(pramod, FRIEND);
  - pramod.createRelationshipTo(martin, FRIEND);

Neo4j  
the graph database

Adobe

CISCO

NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

DSL  
Decision Systems Laboratory

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

### Graph Databases: Neo4j, Querying

Create index:

```
Transaction transaction = graphDb.beginTx();
try {
    Index<Node> nodeIndex = graphDb.index().forNodes("nodes");
    nodeIndex.add(martin, "name", martin.getProperty("name"));
    nodeIndex.add(pramod, "name", pramod.getProperty("name"));
    transaction.success();
} finally {
    transaction.finish();
}
```

Relationship creation:

```
Node martin = nodeIndex.get("name", "Martin").getSingle();
allRelationships = martin.getRelationships();
incomingRelations = martin.getRelationships(Direction.INCOMING);
```

Neo4j  
the graph database


NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

DSL  
Decision Systems Laboratory




Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Graph Databases: Neo4j, Querying




- Traverse the graphs at any depth:
  - Node barbara = nodeIndex.get("name", "Barbara").getSingle();
  - Traverser friendsTraverser = barbara.traverse(Order.BREADTH\_FIRST, StopEvaluator.END\_OF\_GRAPH, ReturnableEvaluator.ALL\_BUT\_START\_NODE, EdgeType.FRIEND, Direction.OUTGOING);
- Finding paths between two nodes:
  - Node barbara = nodeIndex.get("name", "Barbara").getSingle();
  - Node jill = nodeIndex.get("name", "Jill").getSingle();
  - PathFinder<Path> finder = GraphAlgoFactory.allPaths(Traversal.expanderForTypes(FRIEND, Direction.OUTGOING), MAX\_DEPTH);
  - Iterable<Path> paths = finder.findAllPaths(barbara, jill);



NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Graph Databases: Neo4j, Cypher QL



Cypher query language:

START beginningNode = (beginning node specification)

MATCH (relationship, pattern matches)


WHERE (filtering condition: on data in nodes and relationships)

RETURN (What to return: nodes, relationships, properties)

ORDER BY (properties to order by)

SKIP (nodes to skip from top)


LIMIT (limit results)




NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Graph Databases: Neo4j, Cypher QL




- Find all nodes connected to Barbara, either incoming or outgoing:
  - START barbara = node:nodeIndex(name = "Barbara")
  - MATCH (barbara)--(connected\_node)
  - RETURN connected\_node
- When we are interested in directional significance:
  - For incoming relationship
  - MATCH (barbara) <-- (connected\_node)
  - For outgoing relationship
  - MATCH (barbara) --> (connected\_node)




NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Graph Databases: Neo4j, Cypher QL



- Match can also be done on specific relationships using the :RELATIONSHIP\_TYPE convention and returning the required fields or nodes:
  - START barbara = node:nodeIndex(name = "Barbara")
  - MATCH (barbara)-[:FRIEND]->(friend\_node)
  - RETURN friend\_node.name, friend\_node.location
- Query for relationships where a particular relationship property exists:
  - START barbara = node:nodeIndex(name = "Barbara")
  - MATCH (barbara)-[relation]->(related\_node)
  - WHERE type(relation) = 'FRIEND'
  - RETURN related\_node.name, relation.since



NoSQL distilled : a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.  
Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

**Graph databases: Usage**

- When to Use:
  - Connected Data
  - Routing, Dispatch, and Location-Based Services
  - Recommendation Engines
- When Not to Use
  - When you want to update all or a subset of entities
  - Not “graph” data model

**DSL**  
 Decision Systems Laboratory

NoSQL distilled : a brief guide to the emerging world of  
 polyglot persistence / Pramod J Sadalage, Martin Fowler.  
 Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

**NoSQL databases: Summary**

- Key-value databases
- Document databases
- Column-family stores
- Graph databases

**DSL**  
 Decision Systems Laboratory

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

**NoSQL Databases: Goals**

- Not using the relational model
- Schemaless
- Running well on clusters
- Aggregates – nested data stored together

**DSL**  
 Decision Systems Laboratory

Data Storage

Collecting data  
 Relational Databases  
 • NoSQL Databases  
 Beyond NoSQL Databases

**NoSQL: Schemaless**

- Relational DB:
  - You first have to define a schema for your database
- NoSQL:
  - Storing data is more casual:
    - » Key-value store – allow any data under a key
    - » Document DB – no restrictions on the structure of the document
    - » Column-family DB – allow rows have different columns, any data under any column
    - » Graph DB – allow freely adding new edges and properties to nodes and edges

**DSL**  
 Decision Systems Laboratory

Data Storage

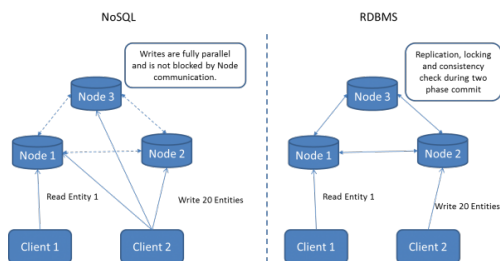
## NoSQL: Schemaless

- **Advantages:**
  - Easy to handle changes
  - Easy to deal with non-uniform data
    - » where each record has different set of fields
- **Disadvantages:**
  - Database remains ignorant of the schema
    - » Can't validate data types
  - Implicit schema in the application code
    - » Bad and/or complicated code
    - » Multiple application access the same database

## NoSQL Databases – BASE properties

- **Basically available:** Use replication to reduce the likelihood of data unavailability and use “sharding” (partitioning the data among many different storage servers) to make any remaining failures partial. The result is a system that is always available, even if subsets of the data become unavailable for short periods of time.
- **Soft state:** While ACID systems assume that data consistency is a hard requirement, NoSQL systems allow data to be inconsistent and relegate designing around such inconsistencies to application developers.
- **Eventually consistent:** Although applications must deal with instantaneous consistency, NoSQL systems ensure that at some future point in time the data assumes a consistent state. In contrast to ACID systems that enforce consistency at transaction commit, NoSQL guarantees consistency only at some undefined future time.

## Why an RDBMS does not scale and many NoSQL solutions do?



NoSQL differs to RDBMS in the way entities get distributed and that no consistency is enforced across those entities

<http://blog.dynatrace.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/>

## NoSQL databases

- NoSQL databases:**
- Not using the relational model ✓
  - Schemaless ✓
  - Running well on clusters ✓

## Storing Data: Beyond NoSQL

DSL

Data Storage

## Beyond NoSQL

- File systems
- XML Databases
- Object Databases
- Others ...

DSL

Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
• Beyond NoSQL Databases

## File systems

- Simple and widely implemented
  - Most of the devices have one or another file system
- More like key-value stores with hierarchic key
  - No support for queries
- Little control over concurrency
  - Simple locking
- Cope with very large entities
  - Video, audio
- Very good for sequence access
- Works best for relatively small number of large files that can be processed in big chunks

DSL

Data Storage

## XML databases

- Document-like databases
  - Documents are compatible with XML and various XML technologies are used to manipulate the document
- Allow to define schema
  - DTD, XML Schema
- Allow to perform transformation
  - XSLT
- Allow to query documents:
  - XPath, XQuery
  - SQL/XML

DSL


Data Storage

Collecting data  
Relational Databases  
NoSQL Databases  
• Beyond NoSQL Databases

Collecting data  
 Relational Databases  
 NoSQL Databases  
 Beyond NoSQL Databases

## Object databases

- Mapping from in-memory data structures to relational tables
- Close integration with the application



Data Storage

Collecting data  
 Relational Databases  
 NoSQL Databases  
 Beyond NoSQL Databases

## Storage: Summary When and why you (should) choose an RDBMS?

- Table based
- Relations between distinct Table Entities and Rows
- Referential Integrity
- ACID Transactions
- Arbitrary Queries and Joins

<http://blog.dynatrace.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/>



Data Storage

Collecting data  
 Relational Databases  
 NoSQL Databases  
 Beyond NoSQL Databases

## Storage: Summary Why an RDBMS might not be right for you?

- If you just want to store your application entities in a persistent and consistent way
- If you have hierarchical application objects and need some query capability into them
- If you ever tried to store large trees or networks you will know that an RDBMS is not the best solution here
- If you are running in the Cloud and need to run a distributed database for durability and availability.
- You might already use a data warehouse for your analytics. If your data grows to large to be processed on a single machine, you might look into hadoop or any other solution that supports distributed Map/Reduce.

<http://blog.dynatrace.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/>


Data Storage

Collecting data  
 Relational Databases  
 NoSQL Databases  
 Beyond NoSQL Databases

## Storage: Summary Hybrid Systems? (examples)


**F1 - The Fault-Tolerant Distributed RDBMS Supporting Google's Ad Business**

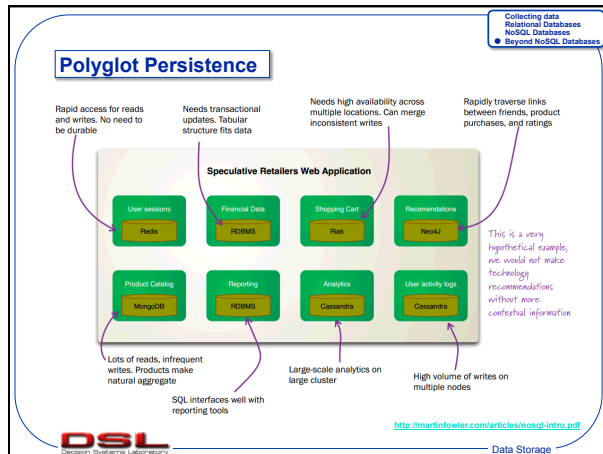
Jeff Shute, Mircea Oancea, Stephan Ellner, Ben Handy, Eric Rollins, Bart Samwel, Radek Vingralek, Chad Whipkey, Xin Chen, Beat Jegerlehner, Kyle Littlefield, and Phoenix Tong. 2012. F1: the fault-tolerant distributed RDBMS supporting google's ad business. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12). ACM, New York, NY, USA, 777-778. DOI=10.1145/2213836.2213954  
<http://doi.acm.org/10.1145/2213836.2213954>

**Oracle In-Database Hadoop: When MapReduce Meets RDBMS**

Xueyuan Su and Garret Swart. 2012. Oracle in-database hadoop: when mapreduce meets RDBMS. In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD '12). ACM, New York, NY, USA, 779-790. DOI=10.1145/2213836.2213955  
<http://doi.acm.org/10.1145/2213836.2213955>

<http://blog.dynatrace.com/2011/10/05/nosql-or-rdbms-are-we-asking-the-right-questions/>


Data Storage



Collecting data  
Relational Databases  
NoSQL Databases  
Beyond NoSQL Databases

## Suggested readings

Raghu Ramakrishnan, Johannes Gehrke. Database Management Systems. 3d Edition. WCB/McGraw-Hill. 2005

Michael Stonebraker and Joseph M. Hellerstein , What Goes Around Comes Around ( <http://mitpress.mit.edu/books/chapters/0262693143chapm1.pdf>)

NoSQL distilled: a brief guide to the emerging world of polyglot persistence / Pramod J Sadalage, Martin Fowler.

Tom White, Hadoop: The Definitive Guide, 3rd Edition, 2012

Fay Chang et. al, Bigtable: A Distributed Storage System for Structured Data

Jeffrey Dean and Sanjay Ghemaw, MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008)

<http://hadoop.apache.org/>

[http://hadoop.apache.org/docs/r0.20.2/hdfs\\_design.html](http://hadoop.apache.org/docs/r0.20.2/hdfs_design.html)

<http://hive.apache.org/docs/r0.9.0/>

<http://hbase.apache.org/>

<http://pig.apache.org/docs/r0.8.1/>

[http://static.googleusercontent.com/external\\_content/untrusted\\_dlcp/research.google.com/en/us/pubs/archive/38125.pdf](http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/pubs/archive/38125.pdf)

DSL  
Decision Systems Laboratory

Data Storage

