Hi!

DS501: Data Storage

Prof. Randy Paffenroth
rcpaffenroth@wpi.edu

Worcester Polytechnic Institute

WPI

Objectives for today

- To discuss “data storage”
- Learn how to store data so that it is easy to access.
- What are the standards for storing data?
- Where are data storage ideas heading?

http://kathleendeery.com/wp-content/uploads/091615_1554_Drinkingfro1.jpg

"Backyardpool" by Vic Brincat from Keswick, Ontario, Canada - 050730_021. Licensed under CC BY 2.0 via Commons - https://commons.wikimedia.org/wiki/File:Backyardpool.jpg#/media/File:Backyardpool.jpg

WPI

Announcements

- Case study 1 is due February 10 (**BEFORE THE START OF CLASS**)
- I **know** you have all started already, so you must have **lots** of question!

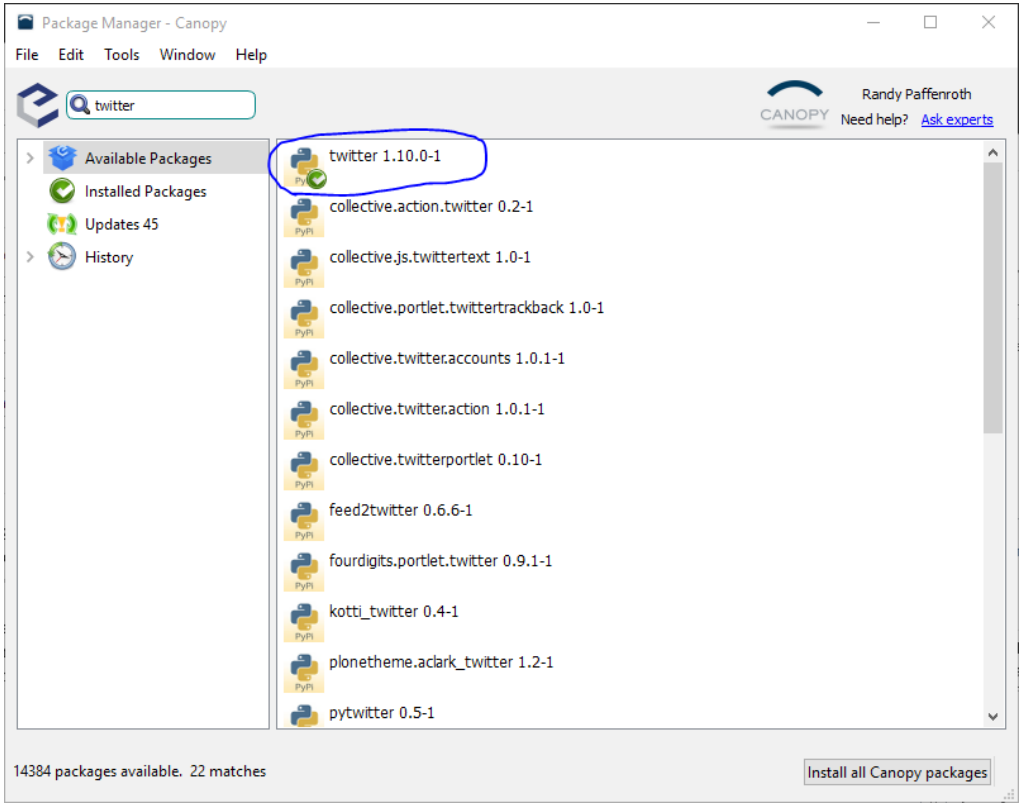
WPI

Announcements

- Which twitter package to use?

1) Be sure you used your ".edu" address when you registered for Canopy and make sure you have the full version of Canopy installed.

2)



WPI

Announcements

- What kinds of words should you count in Case Study 1?
  - Impress me!
  - Think of yourself giving a presentation to the Vice President of your company.
- Hint: look at Example 5-5 in Mining the Social Web
  - http://proquest.safaribooksonline.com.ezproxy.wpi.edu/book/web-applications-and-services/social-media/9781449368180/idot-a-guided-tour-of-the-social-web/ch05_html

Announcements

- We have a grader!
- When you submit your Case Study 1 please send it to both:
  - rcpaffenroth@wpi.edu
  - Liu, Wen <wliu3@wpi.edu>

Announcement

- Peer grading form.

Where do we store data?

Cloud virtual storage

Json, text File, Hard Disks
Log Files

Thumb Drives

excel, access – relational Database

Query's are easy

Hard copy Uniform

DataBases?

Access

mongoDB

oracle postgresql

mysql

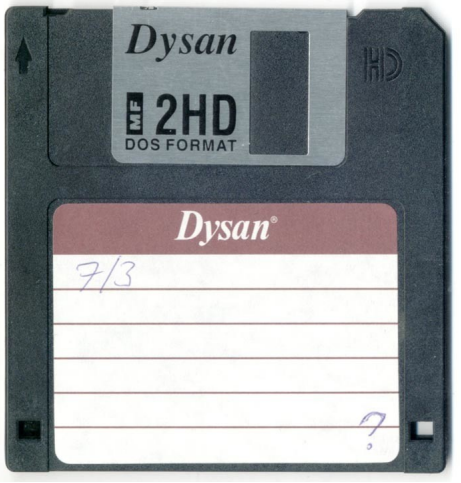
Foreign keys

Rows & Columns

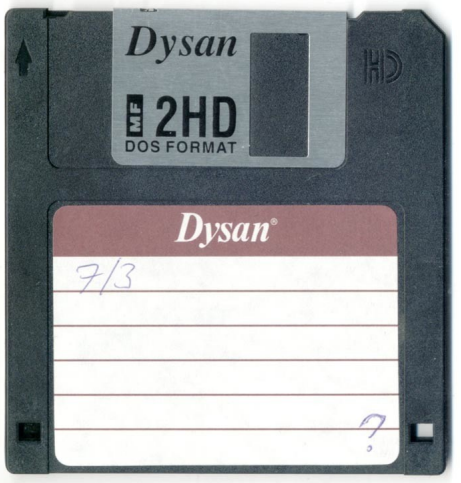
TABLES

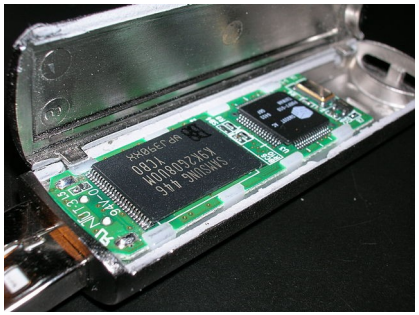
Cassandra

Where do we store data?



Where do we store data?





"USB flash drive". Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:USB_flash_drive.JPG#/media/File:USB_flash_drive.JPG

"Laptop-hard-drive-exposed" by Evan-Amos - Own work. Licensed under CC BY-SA 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Laptop-hard-drive-exposed.jpg#/media/File:Laptop-hard-drive-exposed.jpg

Where we store data in modern computers.

Why not store everything in Main Memory?

Why not use disk only?

Large scale storage

sterlingdatastorage.com

But, how can we all participate? Example...

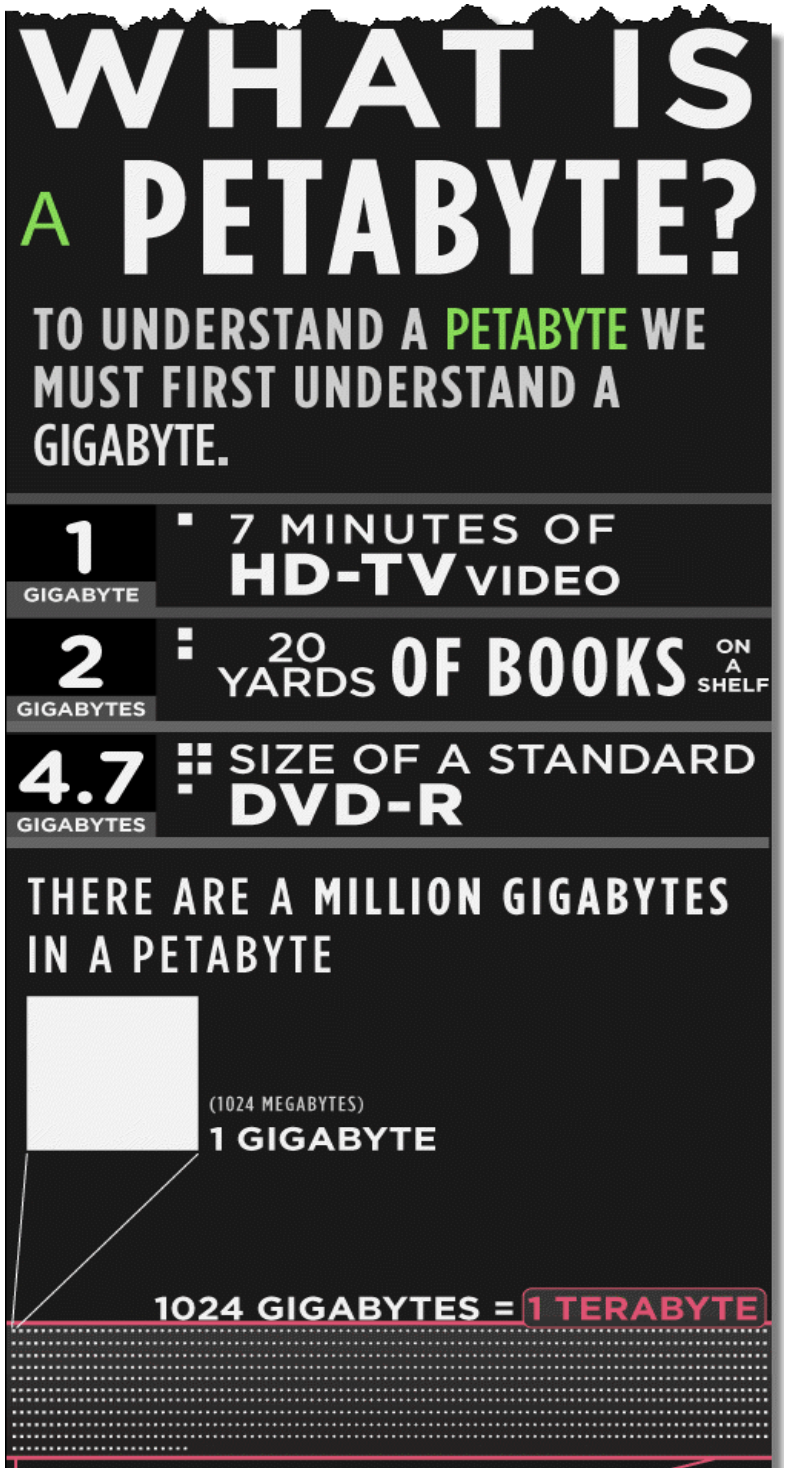
Amazon AWS S3

$5 month/GB

Free month/GB

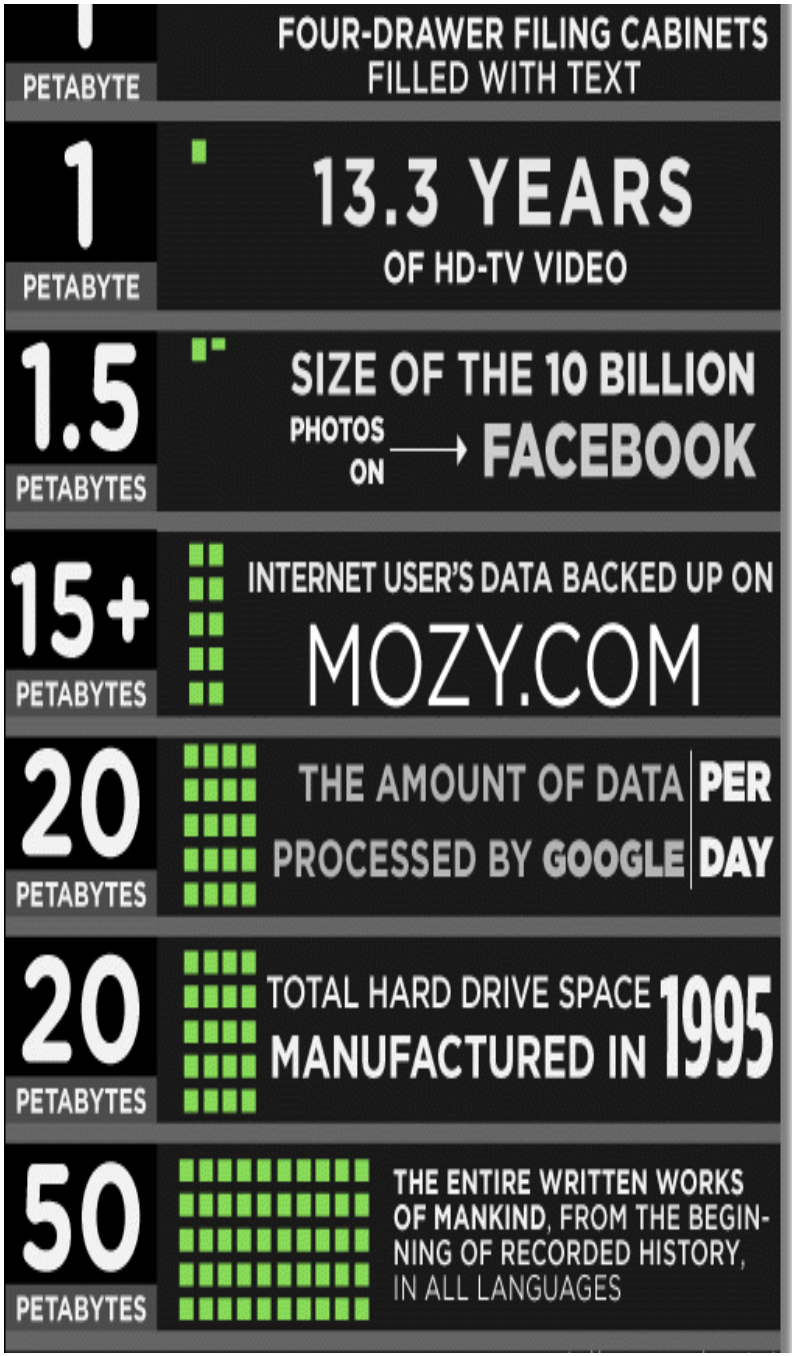
"AWS Simple Icons Storage Amazon S3 Bucket with Objects" by Amazon Web Services LLC - http://aws.typepad.com/aws/2011/12/introducing-aws-simple-icons-for-your-architecture-diagrams.html. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:AWS_Simple_Icons_Storage_Amazon_S3_Bucket_with_Objects.svg#/media/File:AWS_Simple_Icons_Storage_Amazon_S3_Bucket_with_Objects.svg

https://aws.amazon.com/s3/pricing/



1 PetaByte

http://mswhs.files.wordpress.com/2009/07/whatsapetabyte.gif



1 PetaByte

http://mswhs.files.wordpress.com/2009/07/whatsapetabyte.gif

Why relevant to Data Science?

- Beyond just having a place to put the data...

Lots of computing

analysis

Getting the data Backout

Backup

moving it around

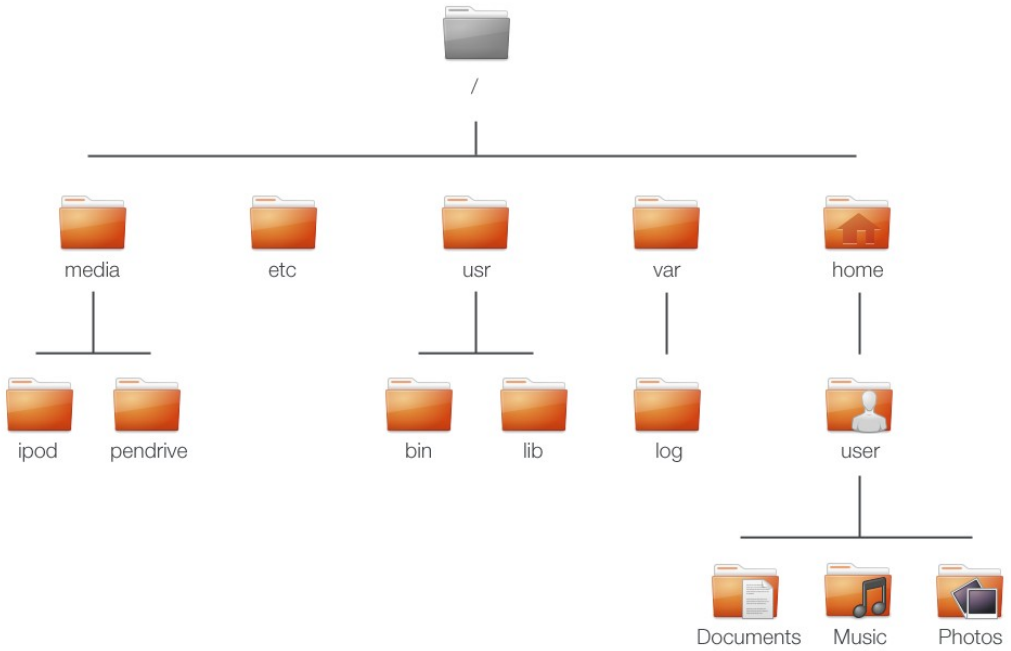
organizing the data

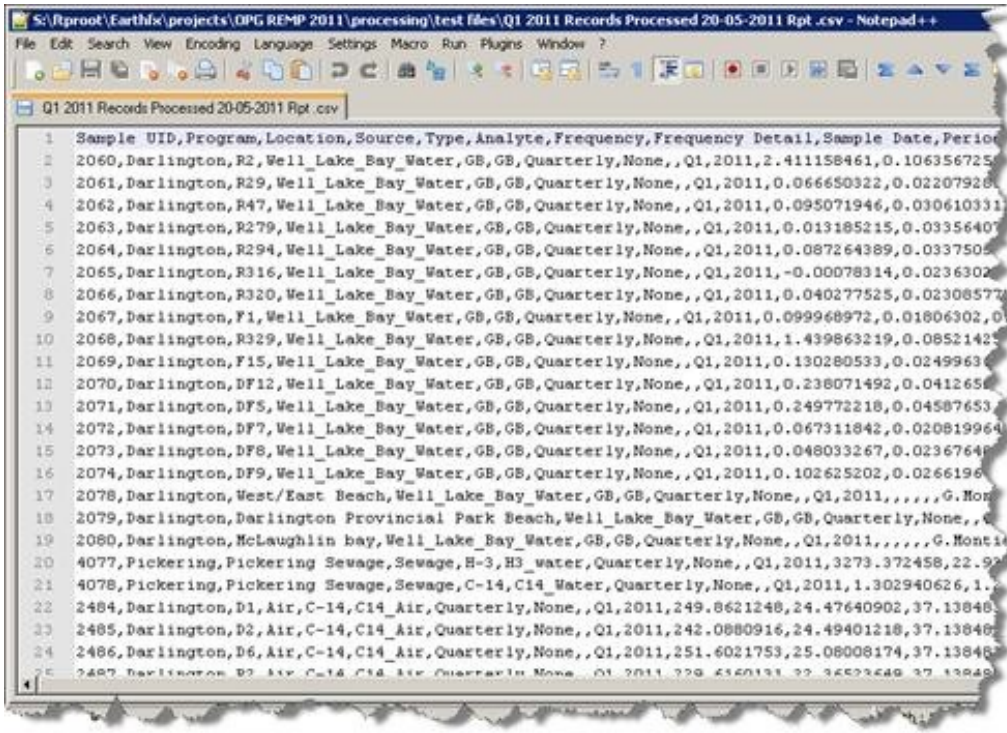
Why relevant to Data Science?

We need to know: How can we ***find*** the data we need in a system?

- How the data is physically organized in the system?
- What kinds of queries are efficiently supported?
- How do we organize new data with the system?

How do we store data?

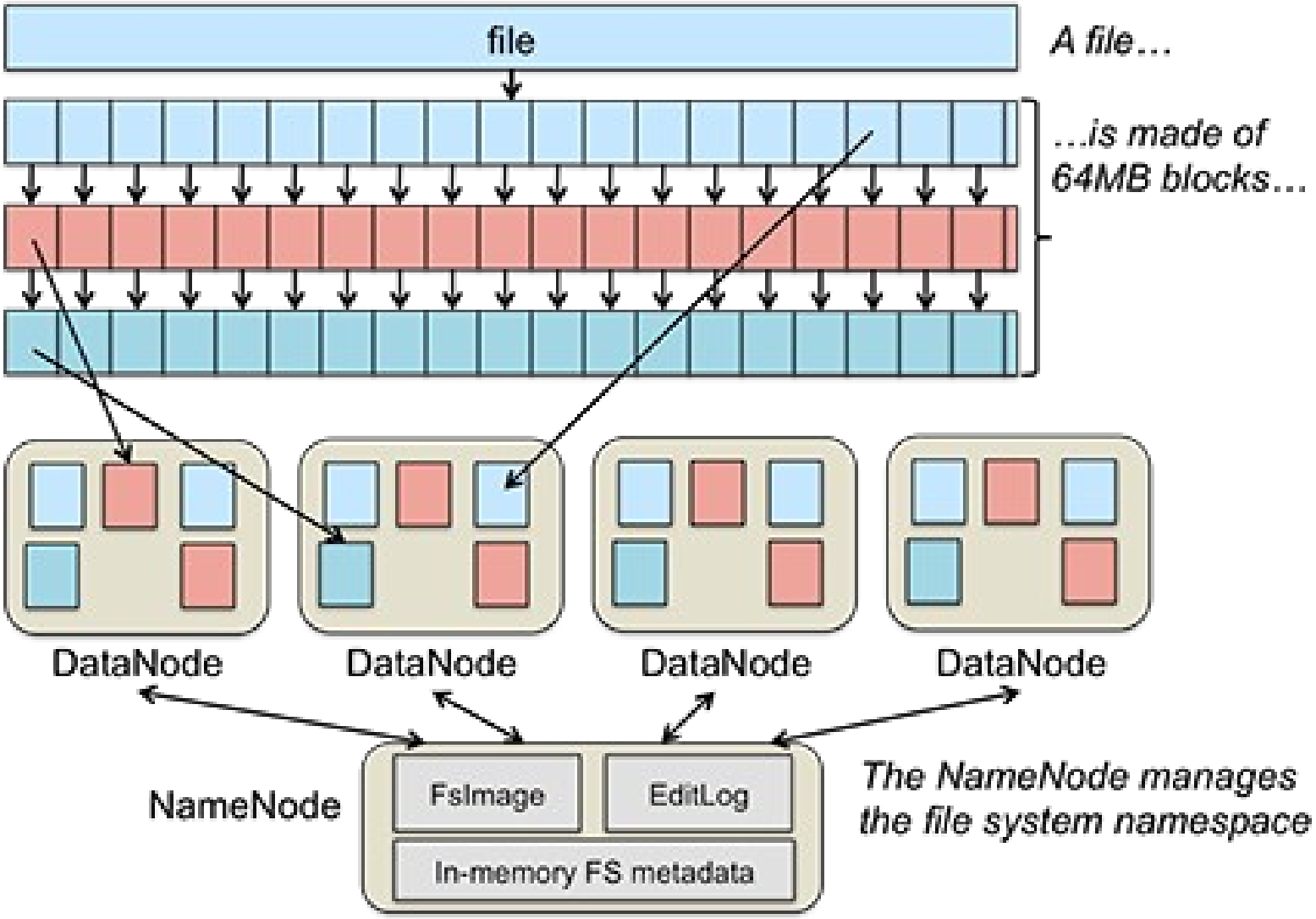




- File System
- Text File

How do we store data?

- Distributed File System
  - HDFS (Hadoop File System)



How do we store data?

- Database System
  - Bank
  - Airline scheduling/ticketing
  - Websites

What is a Database System?

- A **database** is an organized **collection** of data.
  - The focus: efficient data query/ retrieval.
- A "database management system" (DBMS) is a suite of computer software providing the interface between users and a database or databases.

source from: wikipedia

Files vs DBMS

Why not let OS manage all the data?

- Application must move large datasets into memory and make operations. (Reading files)
- Special codes for different queries
- Protect the data from inconsistency when we have multiple users
- More problems: crash recovery, clean data …

Problems DBMS can solve

- **Data Model**: clean and organized data
- **Scale**: too large to fit in memory
- **Sharing**: multiple readers and writers
- Concurrent access, recovery from crashes
- Reduced application development time

Important things in DBMS

- There are ***three*** important things in DBMS:
  - " *Performance*, *Performance*, and *Performance*"

Jennifer Widom
Stanford Univ.

Features of DBMS

- Massive
- Persistent
- Multiple users/applications
- Convenient: high-level query languages
- Efficient
- Reliable

Source: https://class.stanford.edu/courses/Engineering/db/2014_1/

Concepts!

- Data Model
- Schema & Data
- Data Definition Language (DDL)
- Data Manipulation Language (DML)

Data Models

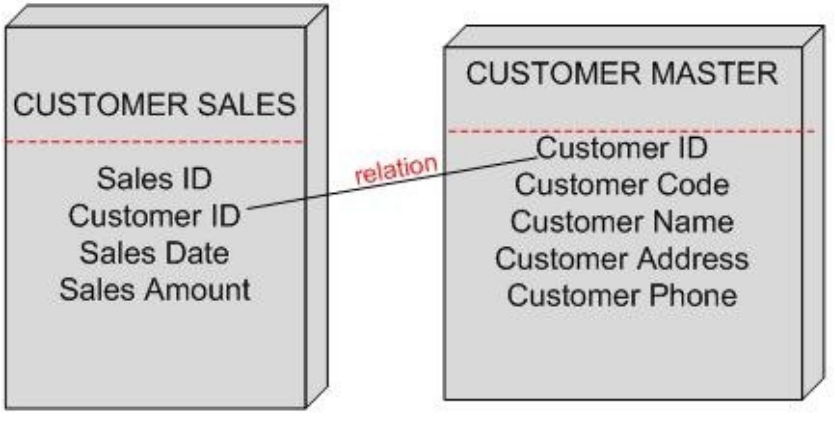
We will describe three, so you get a flavor, but there are many more!

- A **data model** is a collection of **concepts** for describing data.
- A **schema** is a description of a particular collection of data, using the a given data model.
  - **Relational Model (most widely used)**
  - JSON
  - XML
  - …

Relational Model

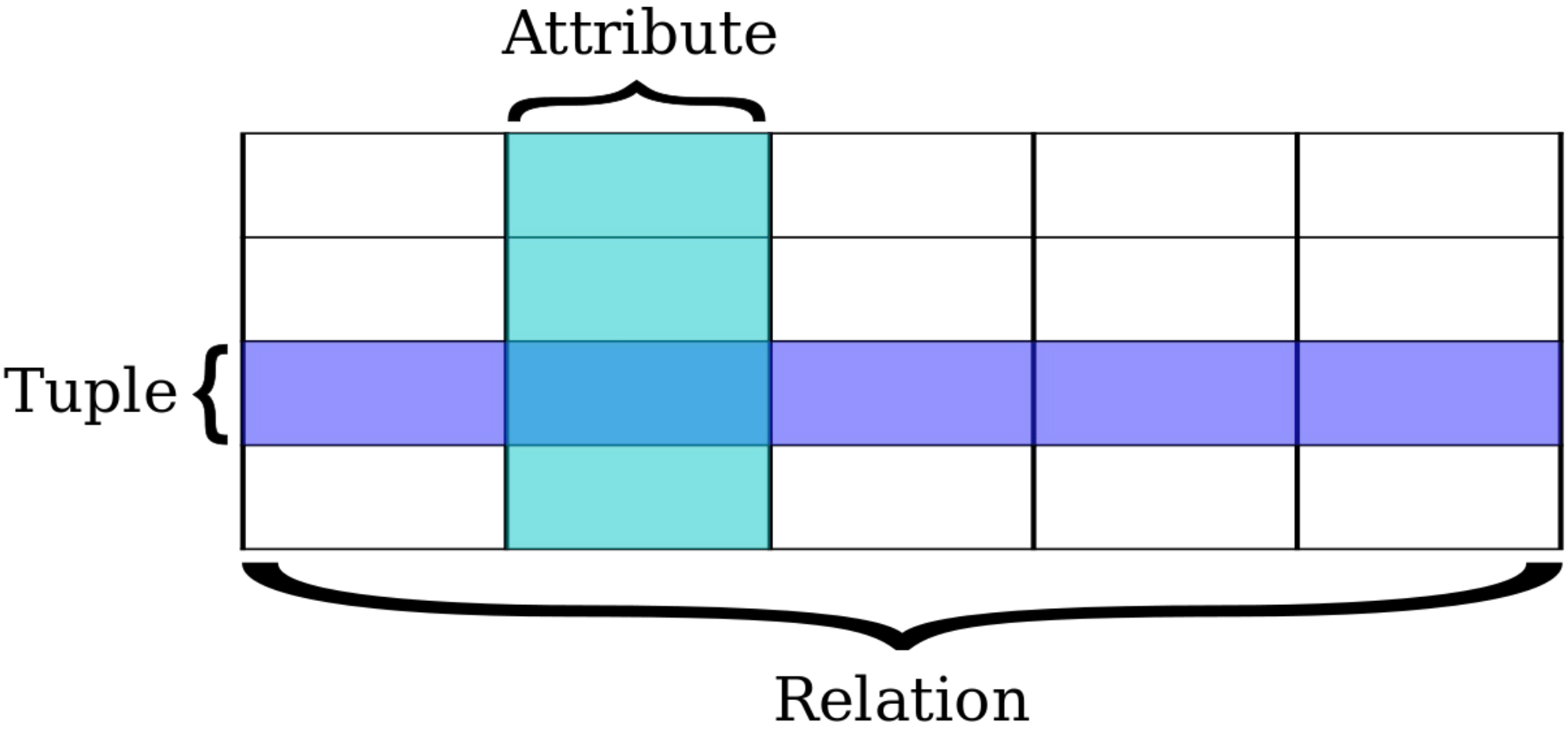
Main concept: **relation**, basically a table with rows and columns.

Every relation has a **schema**, which describes the columns, or fields



Sales ID	Customer ID	Sales Date	Sales Amount
1	101	12/09/2008	10000
2	101	01/09/2008	23789
3	102	02/07/2008	45000
4	103	11/06/2008	25345

Customer ID	Customer Code	Customer Name	Customer Address	Customer Phone
101	C00101	All sec Corp	Houston, Texas	001-325-789-321
102	C00102	John S	Chennai	0091-44-273910
103	C00103	Bridge Inc.	Delhi	0091-11-456801
104	C00104	Symphony Org	Bombay	0091-22-568902


Attribute
Tuple
Relation

Relational Database

- **Relational database**: a set of ***relations***
- **Relation:**
  - Instance: a table with rows and columns
  - Schema: specifies name of relation, name and type of each column.
  - a set of rows (tuples), i.e., all rows are distinct

Example

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- all rows should be distinct
- how about columns?

Example: University DB

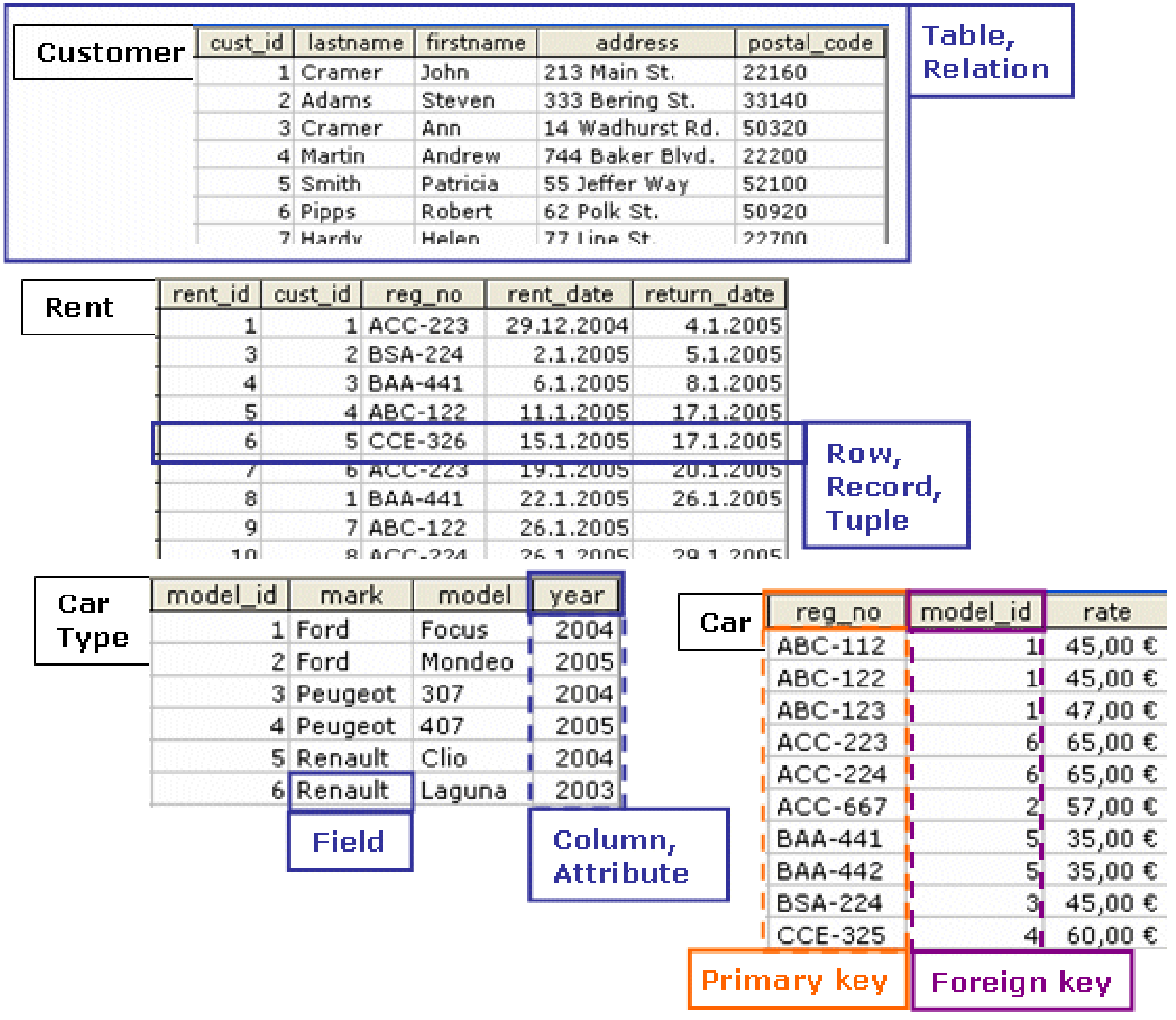
- Schema:
  - Students (sid: string, name: string, login: string, age: integer, gpa: real)
  - Courses (cid:string, cname:string, credits:integer)
  - Enrolled (sid: string, cid: string, grade:string)

Keys

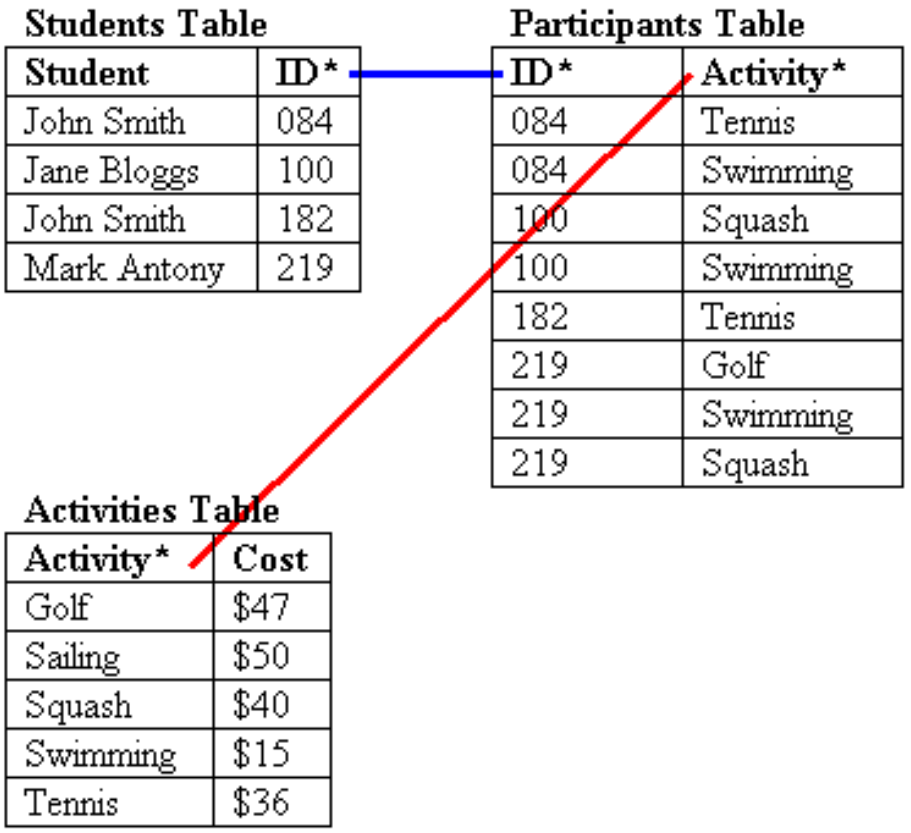
- Key: a field with unique values
- Foreign key: Set of fields in one relation that is used to refer to a tuple in another relation

Example: University DB

- Schema:
  - Students (sid: string, name: string, login: string, age: integer, gpa: real)
  - Courses (cid:string, cname:string, credits:integer)
  - Enrolled (sid: string, cid: string, grade:string)


Customer
cust_id lastname firstname address postal_code
1 Cramer John 213 Main St. 22160
2 Adams Steven 333 Bering St. 33140
3 Cramer Ann 14 Wadhurst Rd. 50320
4 Martin Andrew 744 Baker Blvd. 22200
5 Smith Patricia 55 Jeffer Way 52100
6 Pipps Robert 62 Polk St. 50920
Table, Relation
Rent
rent_id cust_id reg_no rent_date return_date
1 1 ACC-223 29.12.2004 4.1.2005
3 2 BSA-224 2.1.2005 5.1.2005
4 3 BAA-441 6.1.2005 8.1.2005
5 4 ABC-122 11.1.2005 17.1.2005
6 5 CCE-326 15.1.2005 17.1.2005
8 1 BAA-441 22.1.2005 26.1.2005
9 7 ABC-122 26.1.2005
Row, Record, Tuple
Car Type
model_id mark model year
1 Ford Focus 2004
2 Ford Mondeo 2005
3 Peugeot 307 2004
4 Peugeot 407 2005
5 Renault Clio 2004
6 Renault Laguna 2003
Field
Column, Attribute
Car
reg_no model_id rate
ABC-112 1 45,00 €
ABC-122 1 45,00 €
ABC-123 1 47,00 €
ACC-223 6 65,00 €
ACC-224 6 65,00 €
ACC-667 2 57,00 €
BAA-441 5 35,00 €
BAA-442 5 35,00 €
BSA-224 3 45,00 €
CCE-325 4 60,00 €
Primary key
Foreign key

Relational Model



Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Participants Table

ID*	Activity*
084	Tennis
084	Swimming
100	Squash
100	Swimming
182	Tennis
219	Golf
219	Swimming
219	Squash

Activities Table

Activity*	Cost
Golf	$47
Sailing	$50
Squash	$40
Swimming	$15
Tennis	$36

- Why relational Model?

Data Models

- A **data model** is a collection of **concepts** for describing data.
- A **schema** is a description of a particular collection of data, using the a given data model.
    - Relational Model (most widely used)
    - **JSON**
    - XML
    - …

JSON

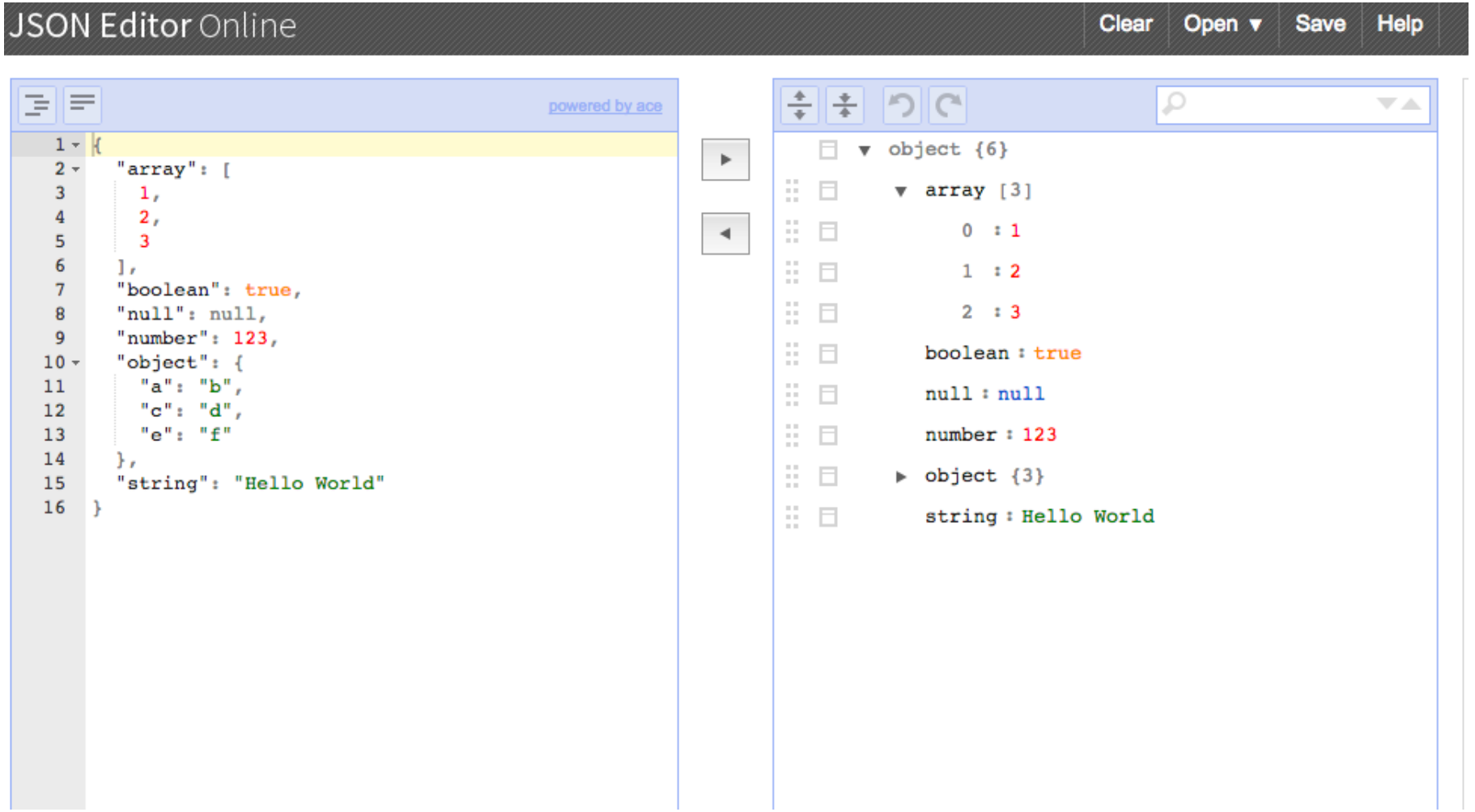
- Data model for semi-structured data

```
{ "users":[
            {
                "firstName":"Ray",
                "lastName":"Villalobos",
                "joined": {
                    "month":"January",
                    "day":12,
                    "year":2012
                }
            },
            {
                "firstName":"John",
                "lastName":"Jones",
                "joined": {
                    "month":"April",
                    "day":28,
                    "year":2010
                }
            }
    ]}
```

- Basic types: number, string, boolean, …
- Objects { }
  - sets of label-value pairs
- Arrays [ ]
  - list of values

JSON Demo

- http://www.jsoneditoronline.org/



Practice

Which is NOT a valid JSON object?

```
{ "name":
"Smiley",
  "age": 20,
  "phone": null,
  "email": null,
  "happy": true
}
```

```
{ "name": "Smiley",
  "age": 20,
  "phone": "888-123-
4567",
  "email":
"smiley@xyz.com",
  "happy": true }
```

```
{ "name": "Smiley",
  "age": 20,
  "phone": "888-123-
4567",
  "email":
smiley@xyz.com,
  "happy": true }
```

```
{ "name":
"Smiley",
  "age": 20,
  "phone": null,
  "email":
"null",
  "happy": true
}
```

Practice

Which is NOT a valid JSON array?

```
[ [1, 2], ["dog", "cat"], [true, false], [1, "dog",
null],
  {"pet":"dog", "fun":true} ]
```

```
[ 1, 2, "dog", "cat", true, false,
[],
  {"pet":"dog", "fun":true} ]
```

```
[ 1, 2, dog, cat, true, false, [1, "dog",
null],
  {"pet":"dog", "fun":true} ]
```

```
[ 1, 2, "dog", "cat", true, false, [1, "dog", null],
{} ]
```

Comparison of Data Models

	Relational	JSON
Structure	Structured	Semi-structured
Schema	Fixed	flexible
Query	simple	not as easy
Ordering	BasedSet	Arrays
Implementation	Native Systems	NoSQL systems

Data Models

- A **data model** is a collection of **concepts** for describing data.
- A **schema** is a description of a particular collection of data, using the a given data model.
    - Relational Model (most widely used)
    - JSON
    - **XML**
    - …

XML

- Extensible Markup Language (XML)

```
<?xml version="1.0" standalone="yes"?>
<BankAccount>
      <Number>1234</Number>
      <Type>Checking</Type>
      <OpenDate>11/04/1974</OpenDate>
      <Balance>25382.20</Balance>
      <AccountHolder>
            <LastName>Singh</LastName>
            <FirstName>Darshan</FirstName>
      </AccountHolder>
</BankAccount>
```

- HTML (format), XML (content)

XML Components

- **Tagged elements (nested)**
- **Attributes**
- **Text**

- XML:tree
- Text: leaf

<Greeting>Hello, world.</Greeting>

<step number="3">Connect A to B.</step>

```
<?xml version="1.0" ?>
<!-- Bookstore with no DTD -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
      - <Author>
          <First_Name>Hector</First_Name>
          <Last_Name>Garcia-Molina</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
```

Practice

- It has a root element "tasklist"
- The root element has 3 "task" subelements
- Each of the "task" subelements has an attribute named "name"
- The values of the "name" attributes for the 3 tasks are "eat", "drink", and "play"

```
<tasklist>
  <task name="eat">
</task>
  <task name="drink">
</task>
  <task name="play">
</task>
</tasklist>
```

```
<tasklist>
  <task
name="eat"/>
  <task
name="drink"/>
  <task
name="play"/>
<tasklist>
```

```
<tasklist>
  <task
name=eat/>
  <task
name=drink/>
  <task
name=play/>
</tasklist>
```

```
<tasklist>
  <task
name="eat">
  <task
name="drink">
  <task
name="play">
</tasklist>
```

Comparison of Data Models

	Relational	JSON	XML
Structure	Structured	Semi-structured	Hierarchical, Tree
Schema	Fixed	flexible	flexible, “self-describing”
Query	simple, easy	not as easy	less so
Ordering	BasedSet	Arrays	Implied ordering
Implementation	Native Systems	NoSQL systems	Various

```
<?xml version="1.0" ?>
<!-- Bookstore with no DTD -->
- <Bookstore>
  - <Book ISBN="ISBN-0-13-713526-2" Price="85" Edition="3rd">
      <Title>A First Course in Database Systems</Title>
    - <Authors>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
          <Last_Name>Widom</Last_Name>
        </Author>
      </Authors>
    </Book>
  - <Book ISBN="ISBN-0-13-815504-6" Price="100">
      <Remark>Buy this book bundled with "A First Course" -- a great deal!</Remark>
      <Title>Database Systems: The Complete Book</Title>
    - <Authors>
      - <Author>
          <First_Name>Hector</First_Name>
          <Last_Name>Garcia-Molina</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jeffrey</First_Name>
          <Last_Name>Ullman</Last_Name>
        </Author>
      - <Author>
          <First_Name>Jennifer</First_Name>
```

Comparison of Data Models

This JSON syntax defines an employees object, with an array of 3 employee records (objects):

JSON Example

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

This XML syntax also defines an employees object with 3 employee records:

XML Example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Comparison of Data Models

Much Like XML

- Both JSON and XML is plain text
- Both JSON and XML is "self-describing" (human readable)
- Both JSON and XML is hierarchical (values within values)
- Both JSON and XML can be fetched with an HttpRequest

Much Unlike XML

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

The biggest difference is:

XML has to be parsed with an XML parser, JSON can be parsed by a standard JavaScript function.

Relational DBMS

- Backend for large websites
- Backend for web services
- Backend for traditional database application

Example

- Building a system to store information about:
  - students
  - courses
  - professors
  - who takes what, who teaches what

Is it possible? without DBMS

- Yes. Just use file systems

Courses.txt

Students.txt

Professors.txt

- Then write a python program to implement specific tasks

without a DBMS

- Enroll “John Smith” into “DS501”

  You need to write a python program:

  Read “courses.txt”

  Read “students.txt”

  Find&update the record “DS501”

  Find&update the record “John Smith”

  Save “courses.txt”

  Save “students.txt”

Issues

- **Large data sets (say 50 GB)**
- **Multiple users**
- **System crash**

Read “courses.txt”

Read “students.txt”

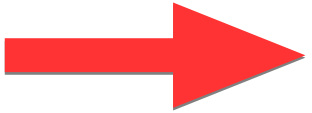
Find&update the record “DS501”

Find&update the record “John Smith”

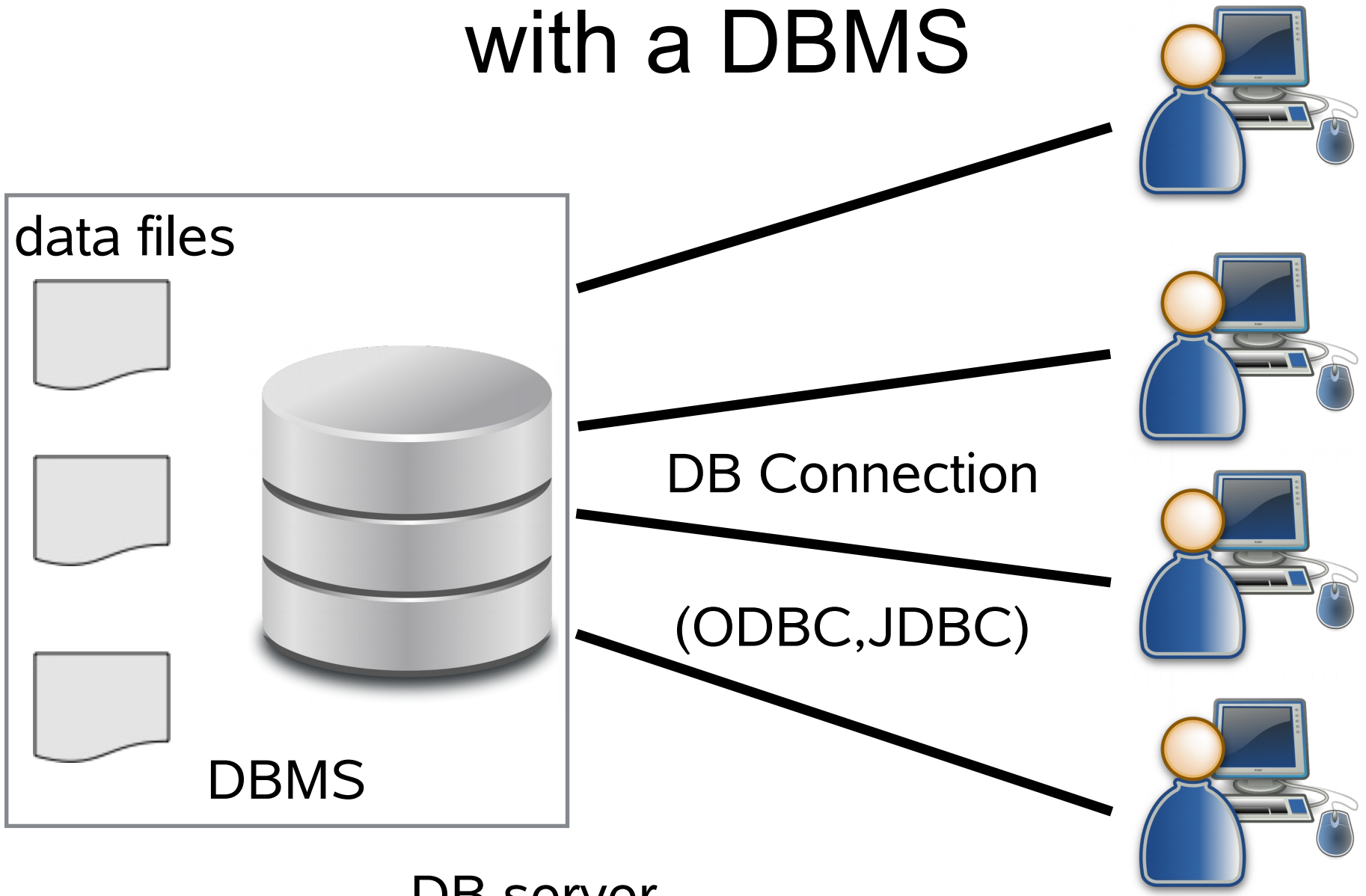
Save “courses.txt”

Save “students.txt”

Ooppsss!



with a DBMS


data files
DBMS
DB Connection
(ODBC,JDBC)
DB server
DB Applications

Evolution of Database Systems

What would you want out of your database?

Integrity

FAST

shared multi user

Reports / queries

up to date - Depends synchronized

organized

safety Reduce anomalies

Goals of Database systems

- All users to ***create*** new databases and specify their schema (using a data-definition language)

- Give users the ability to ***query*** and modify the data (using a query language or data-manipulations language)

- Support storage of very large amounts of data (terabytes or more)

- Enable ***durability***, recover after failures, errors

- Control access to data from many users. Each user should work in *isolation*.

Relational Database Systems

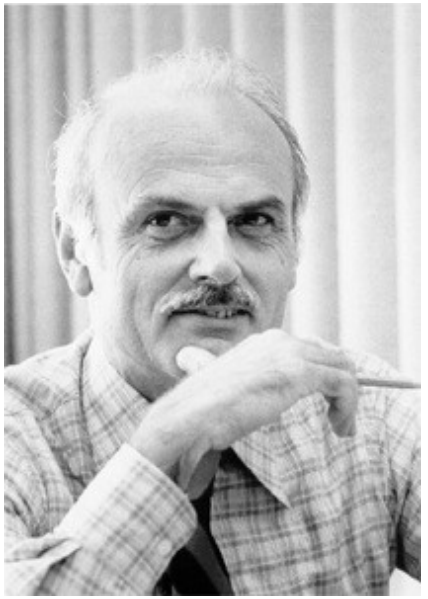
A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed

in Comm. ACM 1970



Edgar F. Codd

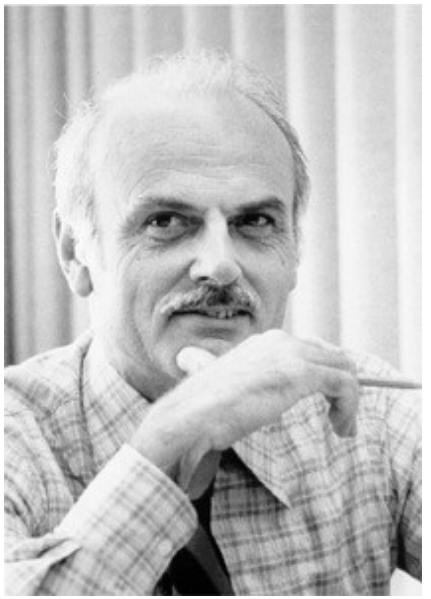
Turing Award
in 1981

ACM SIGMOD

Edgar F. Codd
Innovations Award

Relational Database

- Database systems should present the user with a view of data organized as tables called "*relations*"
- Behind the scenes, there might be complex data structure that allows rapid response to a variety of queries
- the users don't need to know the storage structure
- Queries should be expressed in a very high-level language



Edgar F. Codd

by 1990, relational database became the norm.

Modern DBMS: Smaller

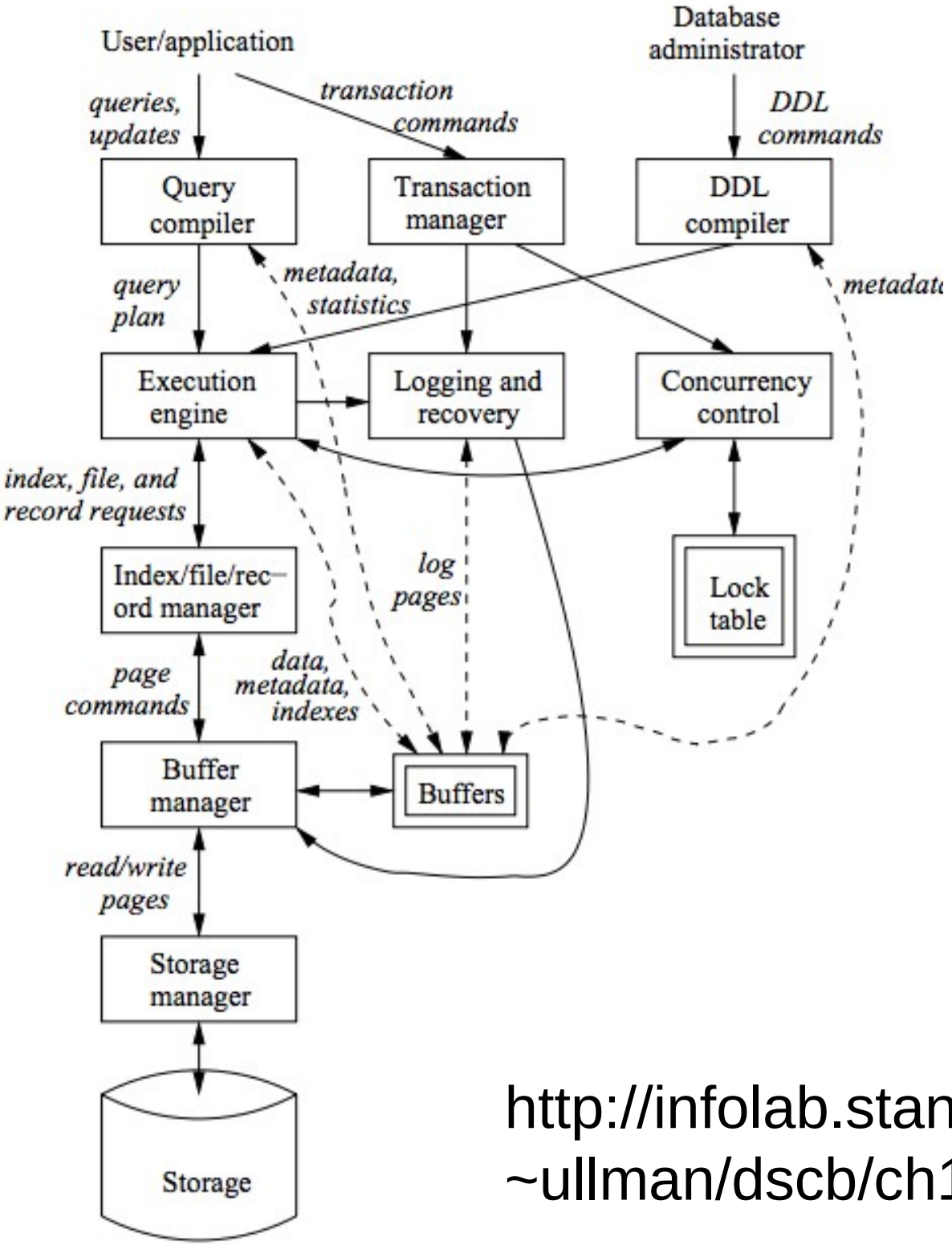
- DBMS were large, expensive systems running on large computers. Because storing 1 gigabyte used to require a large computer system

- Today, terabytes of data can fit on a single disk. DBMS on small computers

Modern DBMS: Bigger

- **gigabytes —> terabytes —> petabytes**
  - Google holds petabytes of data from its crawl of the Web. not in traditional DBMS, but in a specialized structure optimized for search engine
  - Satellites send petabytes of data for storage
  - Pictures. > 1000 words (in space).
    - Flickr store millions of pictures and support search.
  - Videos, 1gigabyte per hours of video. Youtube hold millions of movies and make them available easily
  - P2P file-sharing, distributed data system, small in each node, big together

DBMS Components



http://infolab.stanford.edu/~ullman/dscb/ch1.pdf

Transaction Processing

Group of one or more database operations into a transaction which must be executed atomically and in isolation of other trans.

- ACID properties of Transactions
  - A: “atomicity”, all-or-nothing execution of transactions
  - C: “consistency”, constraints on data, transactions are expected to preserve consistency.
  - I: “isolation”, no other transactions appears to be executed at the same time
  - D: “durability”, the effect of a transaction must never be lost, once the transaction has completed

Transaction Processing

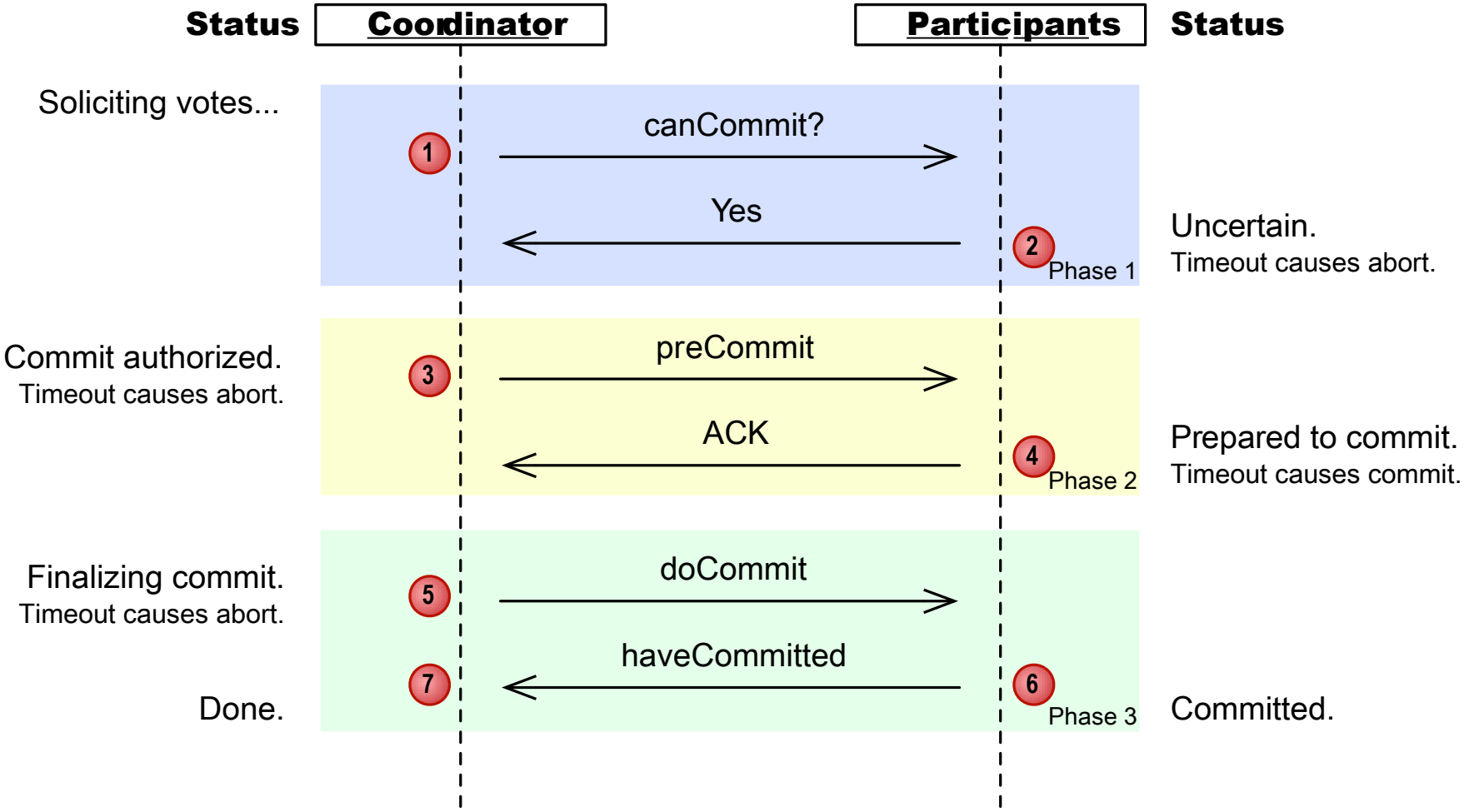
The Transaction processor performs 3 tasks.

1) Logging: Every change in the database is logged on disk (through buffer manager) to enable recovery in case of a crash

2) concurrency control assures
   * atomicity: a transaction is performed either completely or not at all
   * isolation: transactions are executed as if there were no other concurrently executing transactions (uses *locks*)

3) deadlock resolution (“roll back” or “abort” some transaction)

Commit protocol



Query Processing

Query compiler

1) **Parser**: builds a tree structure from the textual form of the query
2) **Preprocessor**: performs semantic checks on the query and translates parse tree into algebraic operators representing the initial query plan
3) **Optimizer**: transforms the initial query plan into the best available sequence of operations on actual data

Execution engine

1) Executes the steps of the chosen query plan
2) Needs to interact with most of the other components of the DBMS

Structured Query Language (SQL)

Operator	Description	Example
`=`	Equal to	`Author = 'Alcott'`
`<>`	Not equal to (many DBMSs accept `!=` in addition to `<>`)	`Dept <> 'Sales'`
`>`	Greater than	`Hire_Date > '2012-01-31'`
`<`	Less than	`Bonus < 50000.00`
`>=`	Greater than or equal	`Dependents >= 2`
`<=`	Less than or equal	`Rate <= 0.05`
`BETWEEN`	Between an inclusive range	`Cost BETWEEN 100.00 AND 500.00`
`LIKE`	Match a character pattern	`First_Name LIKE 'Will%'`
`IN`	Equal to one of multiple possible values	`DeptCode IN (101, 103, 209)`
`IS` *or* `IS NOT`	Compare to null (missing data)	`Address IS NOT NULL`
`IS NOT DISTINCT FROM`	Is equal to value or both are nulls (missing data)	`Debt IS NOT DISTINCT FROM - Receivables`
`AS`	Used to change a field name when viewing results	`SELECT employee AS 'department'`

https://en.wikipedia.org/wiki/SQL

Does it make sense to go beyond tables?

Requirements

simple vs. complicated

Sound

text vs. numeric

Document

email Images

JSON Tweets

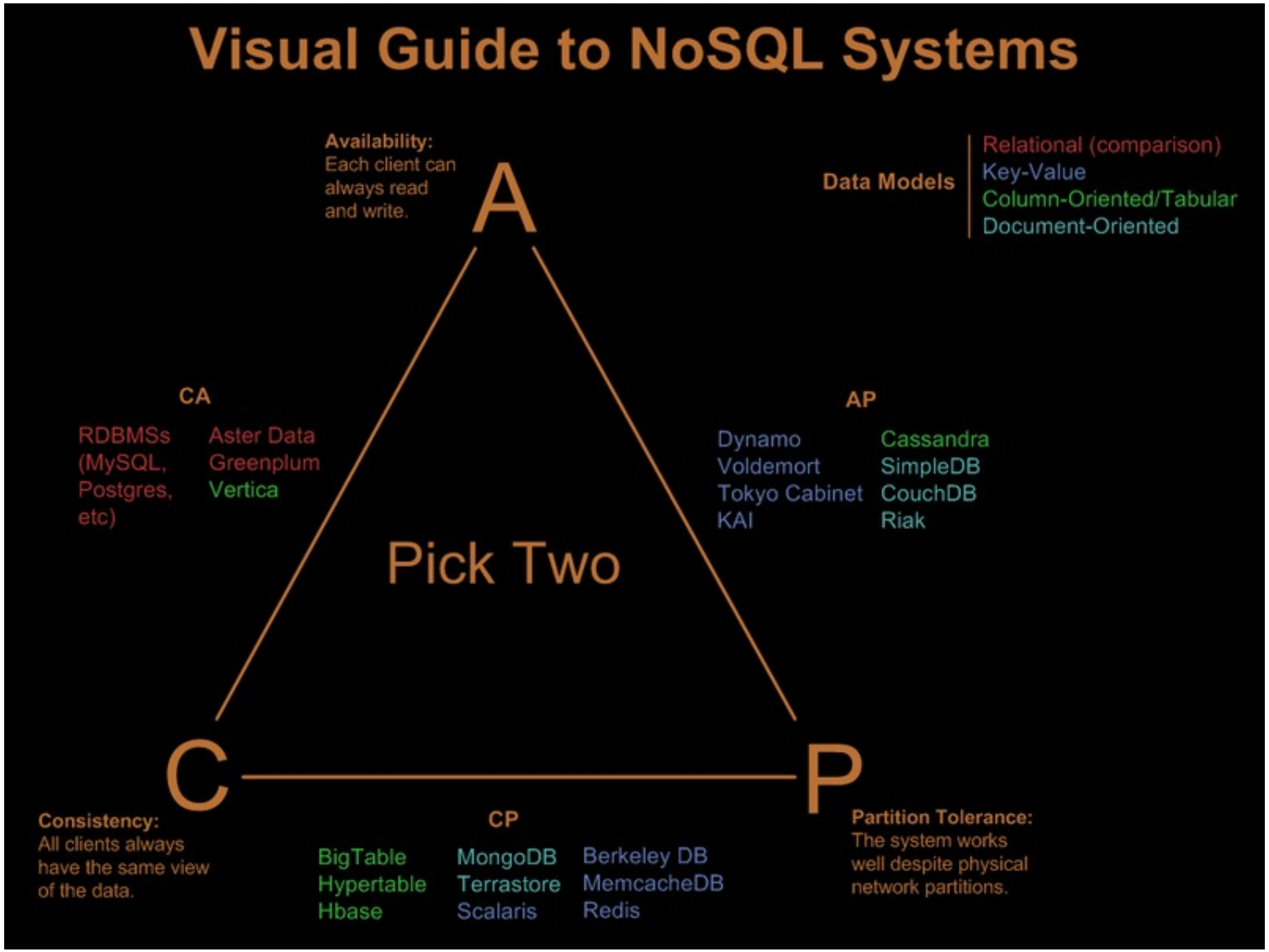
Today NoSQL!

- "Non-SQL"
- "Non-relational"
- "Not only SQL"

Trade-offs

- Cap Theorem or Brewer's Theorem (Eric Brewer)
- It is impossible for a distributed computer systems to simultaneously provide all three of the following guarantees:
  - **Consistency** (all nodes see the same data at the same time)
  - **Availability** (a guarantee that every request receives a response about whether it succeeded or failed)
  - **Partition tolerance** (the system continues to operate despite arbitrary partitioning due to network failures)

NoSQL systems



http://blog.nahurst.com/visual-guide-to-nosql-systems

Optimized for different functions

Data Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key–Value Store	high	high	high	none	variable (none)
Column-Oriented Store	high	high	moderate	low	minimal
Document-Oriented Store	high	variable (high)	high	low	variable (low)
Graph Database	variable	variable	high	high	graph theory
Relational Database	variable	variable	low	moderate	relational algebra

https://en.wikipedia.org/wiki/NoSQL

Most popular DBMS systems

http://db-engines.com/en/ranking

An example...

https://www.mongodb.org/

Pymongo

https://api.mongodb.org/python/current/

Running mongoDB

C:\Program Files\MongoDB\Server\3.2\bin\mongod --dbpath c:\Local\mongodb