CSC 443
Database Management Systems

Winter 2017
Professor: *Marina Barsky*
http://www.cdf.toronto.edu/~csc443h/winter/

Recap: what is a *database*?

A collection of data that exists over a long period of time, organized to afford efficient retrieval.

Two characteristics:

- **Non-volatile** **reliable** storage
- Organized for **efficient** operations

Useful definitions

- A *data model* is a collection of concepts for describing data
- A *schema* is a description of a particular collection of data, using a given data model
- A *view* – result of a stored query

  Same data – multiple views

Example: University Database

- *Logical* model:
  Relational: tables
- Schema:
  Students (sid: string, name: string, age: integer, gpa:real)
  Courses (cid: string, cname:string, credits:integer)
  Enrolled (sid:string, cid:string, grade:string)
- *Physical* model:
  Relations stored as unordered files.
  Index on first column of Students.
- View:
  Course_info (cid:string, enrollment:integer)

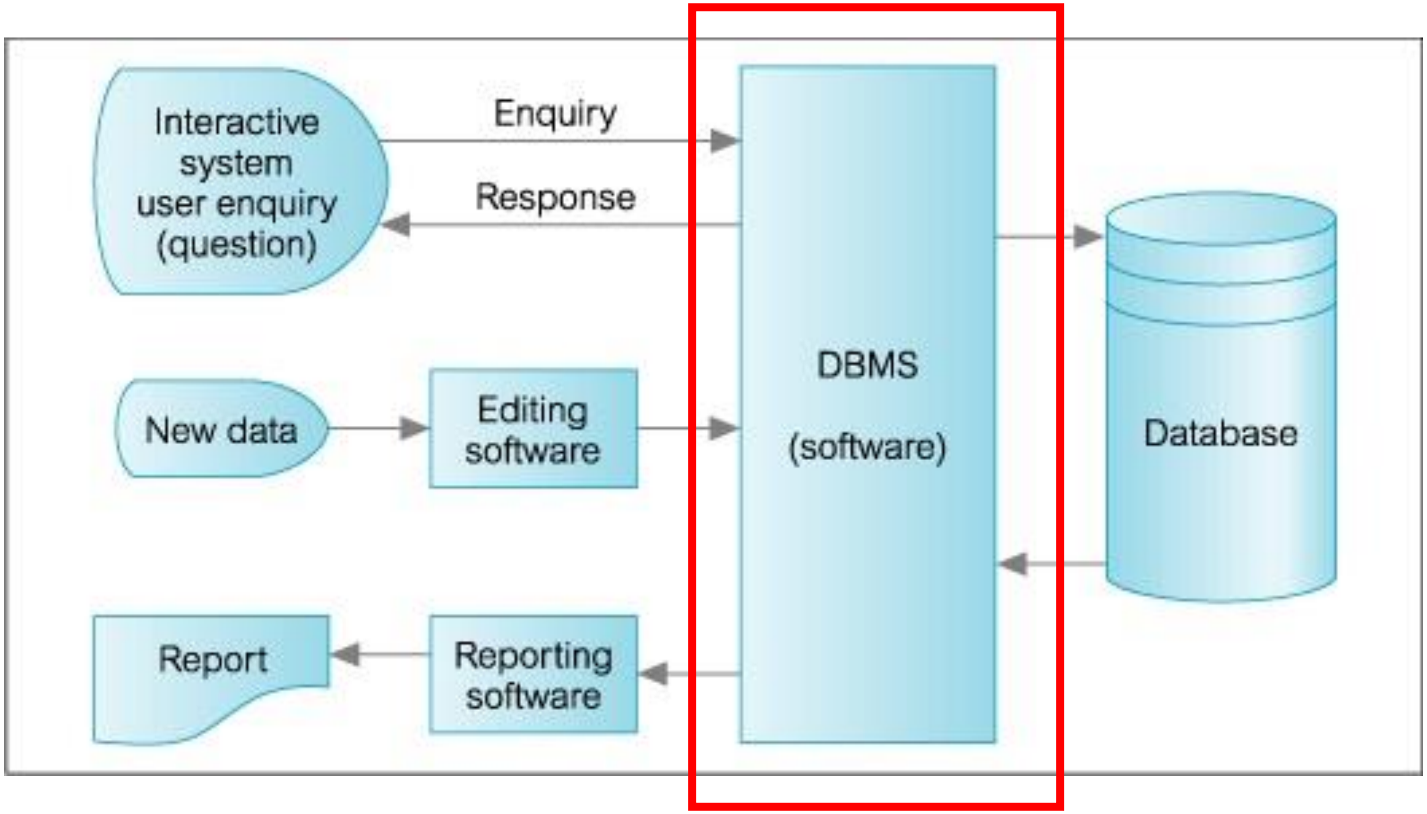
What is a *Database Management System (DBMS)*

A complex *software* for storing and managing databases.

Solves problems of:

- **Scale**: data exceeds main memory, specialized (quite complex) EM algorithms, efficiently implemented
- **Sharing**: using the same data by multiple user programs simultaneously
- **Fault-tolerance**: avoiding data loss
- **Consistency**: clean consistent snapshots of data, reinforcing data constraints

Database management system

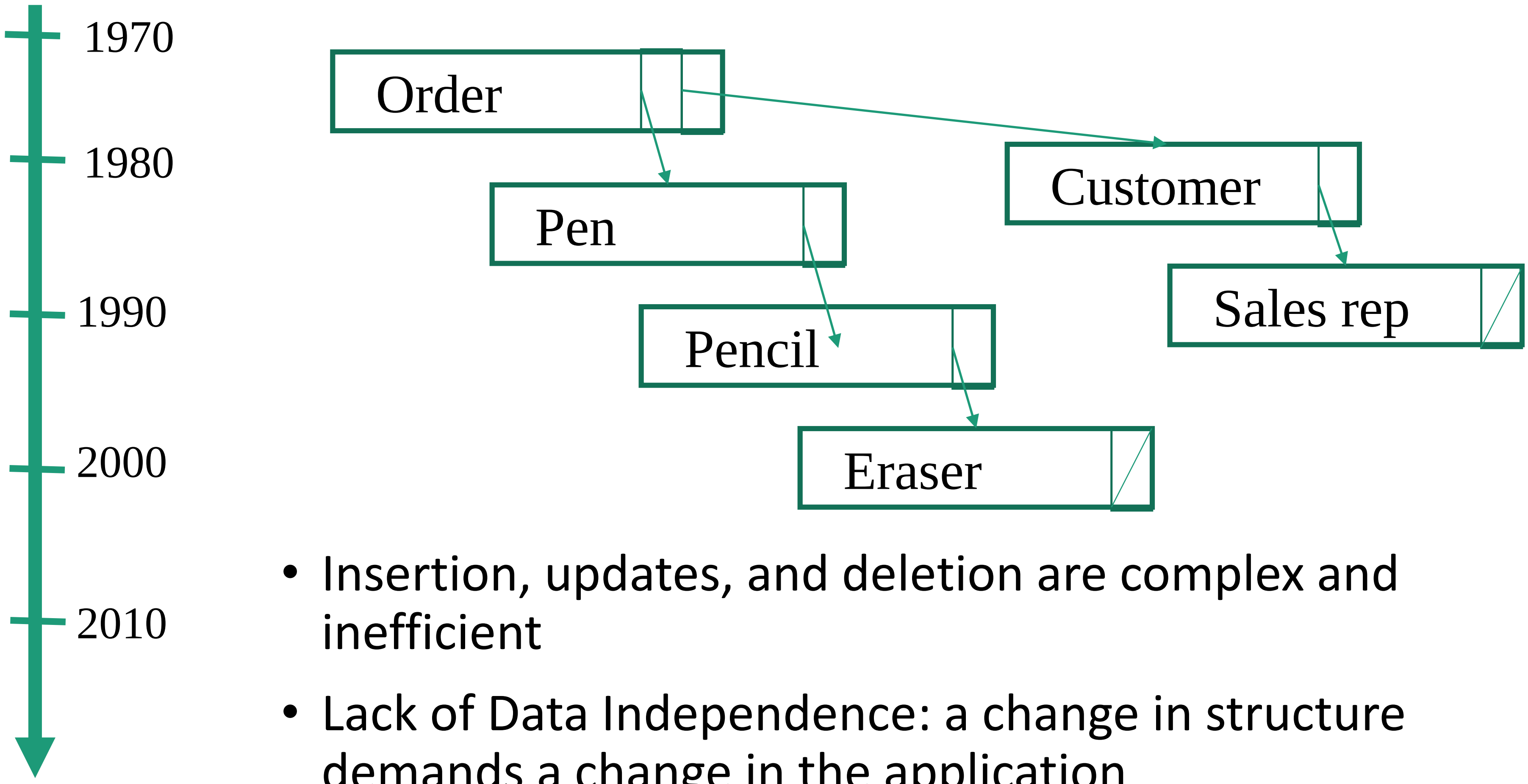


Data models - logical abstractions of data

- Files
- Network databases
- Hierarchical databases
- Relational databases
- Object-oriented databases
- NoSQL databases
- …

History

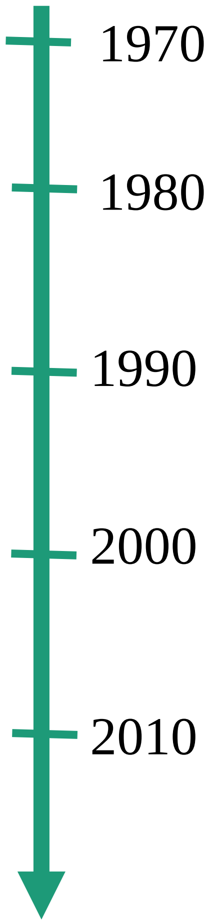
Network databases

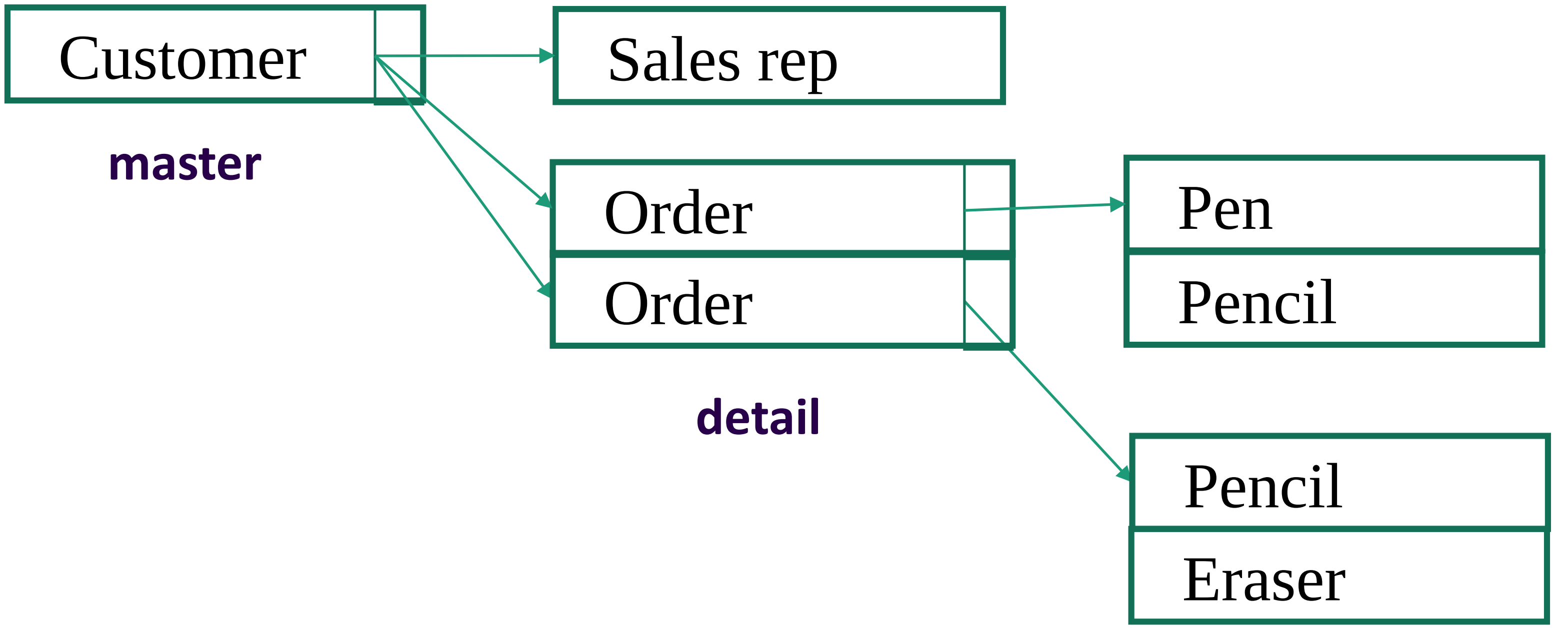


- Insertion, updates, and deletion are complex and inefficient
- Lack of Data Independence: a change in structure demands a change in the application
- Unanticipated queries cannot be performed efficiently

History

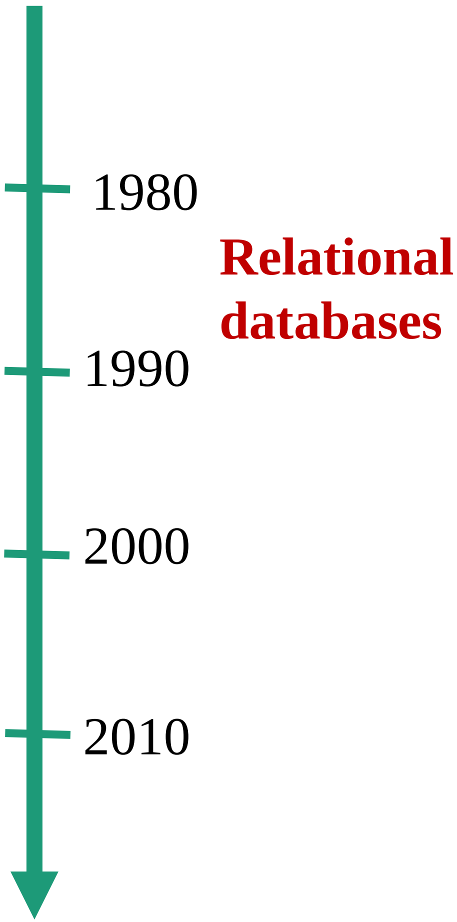
Hierarchical databases





- Data is repetitively stored in many different entities.
- Slow search – scan entire model from top to bottom
- One-to-many relationships only

History



God made the integers;

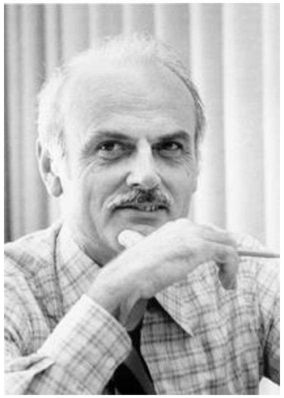
all else is the work of man.

L. Kronecker, 19-th century mathematician

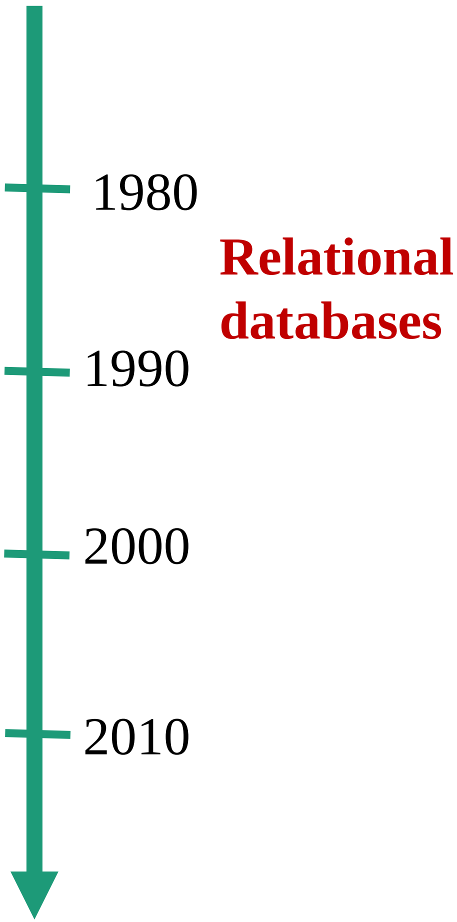
Codd made relations;

all else is the work of man.

R. Ramakrishnan



History



Think in terms of tables, not bits on disk.

"Activities of users at terminals ***should remain unaffected when the internal representation of data is changed***."

- Pre-relational: if your data changed, your application broke
- Early RDBMSs were buggy and slow, but required only 5% of the application code

Relational databases: key idea

Programs that manipulate tabular data exhibit an ***algebraic structure*** allowing reasoning and manipulation independently of physical data representation

Algebraic optimization: symbolic reasoning on integers

N = ((z*2) + ((z*3) + 0))/1

Algebraic laws:

1. Identity: x+0 = x
2. Identity: x/1 = x
3. Distributive: ax + ay = a*(x+y)
4. Commutative: x*y = y*x

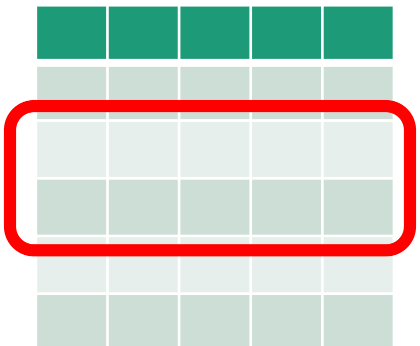
Apply rules 1,3,4,2:

N = (2+3)*z

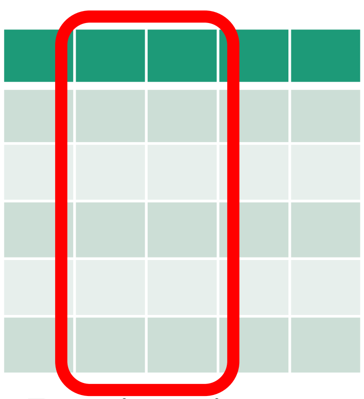
One operation instead of five, no division.
Closure: each operation returns the value of the same type, so operations can be chained

Same idea works with relational algebra!

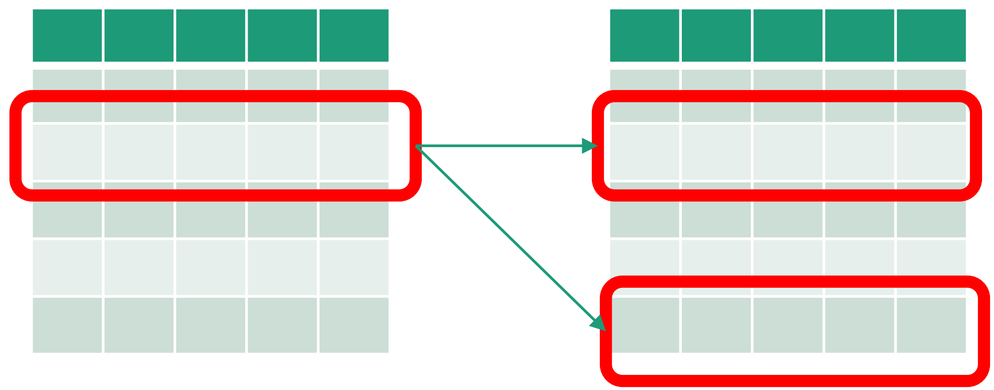
Recap: algebra of tables



Selection σ



Projection π



Join ⋈

Cross-product x

Union ∪

Difference –

Intersection ∩

What is the meaning of the following relational algebra query?

Product (<u>productID</u>, name, price)

Customer (<u>customerID</u>, name, city)

Order (<u>productID</u>, <u>customerID</u>, store)

$\pi_{\text{name, store}} \sigma_{\text{city='Seattle'}}(\text{Orders} \bowtie \text{Customers})$

A. Produce list of stores where each customer from Seattle made orders

B. Produce all combinations of customers and stores in Seattle

Example: SQL query

Product (<u>productID</u>, name, price)

Customer (<u>customerID</u>, name, city)

Order (<u>productID</u>, <u>customerID</u>, store)

```
SELECT DISTINCT p.name, c.name
FROM Product p, Order o, Customer c
WHERE p.productID = o.productID
and c.customerID = o.customerID
and p.price > 100
and c.city = 'Seattle'
```

One SQL - many equivalent RA expressions

```
SELECT DISTINCT p.name, c.name
FROM Product p, Order o, Customer c
WHERE p.productID = o.productID and c.customerID = o.customerID
and p.price > 100 and c.city = 'Seattle'
```

$\pi_{\text{p.name, c.name}}\ \sigma_{\text{p.price >100 and c.city = 'Seattle' and p.productid = o.productid and c.customerID = o.customerID}}(P \times O \times C)$

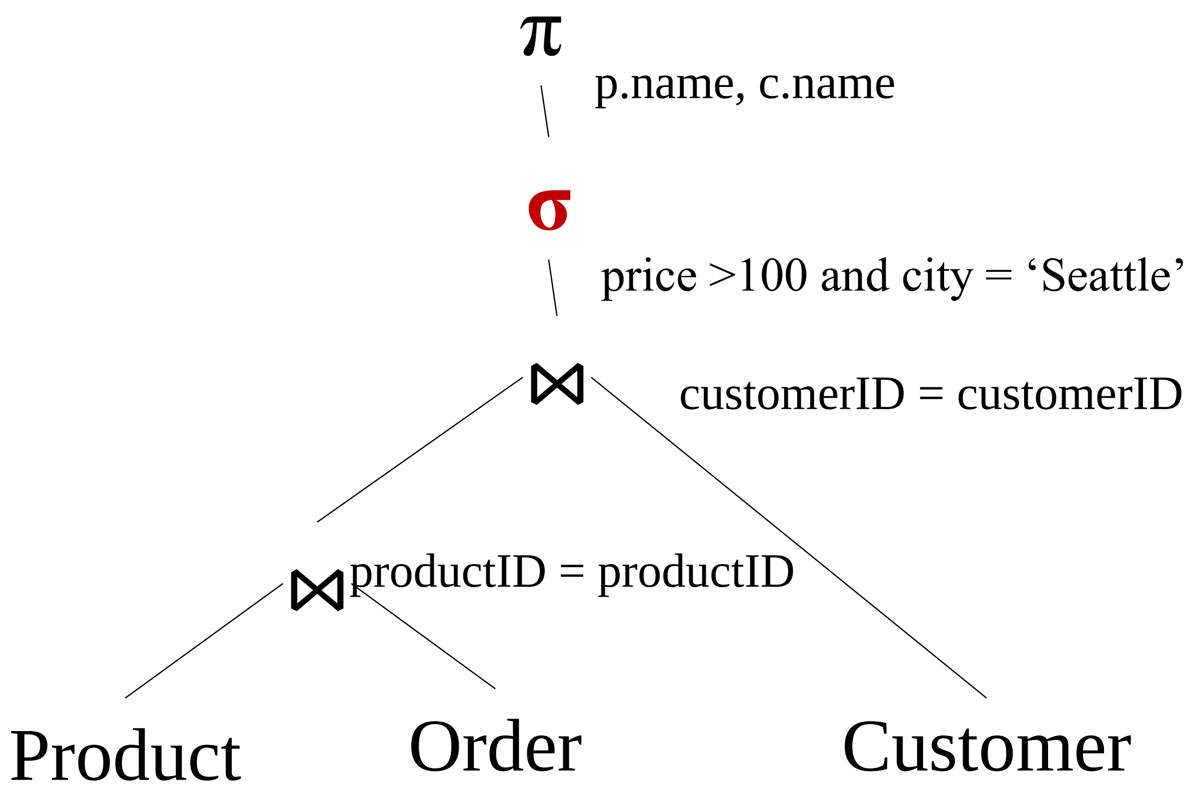
$\pi_{\text{p.name, c.name}}\ \sigma_{\text{p.price >100 and c.city = 'Seattle'}}((P \bowtie O) \bowtie C)$

$\pi_{\text{p.name, c.name}}\ \sigma_{\text{p.price >100 and c.city = 'Seattle'}}((C \bowtie O) \bowtie P)$

$\pi_{\text{p.name, c.name}}(\sigma_{\text{price >100}}(P) \bowtie \sigma_{\text{c.city = 'Seattle'}}(C)) \bowtie O)$

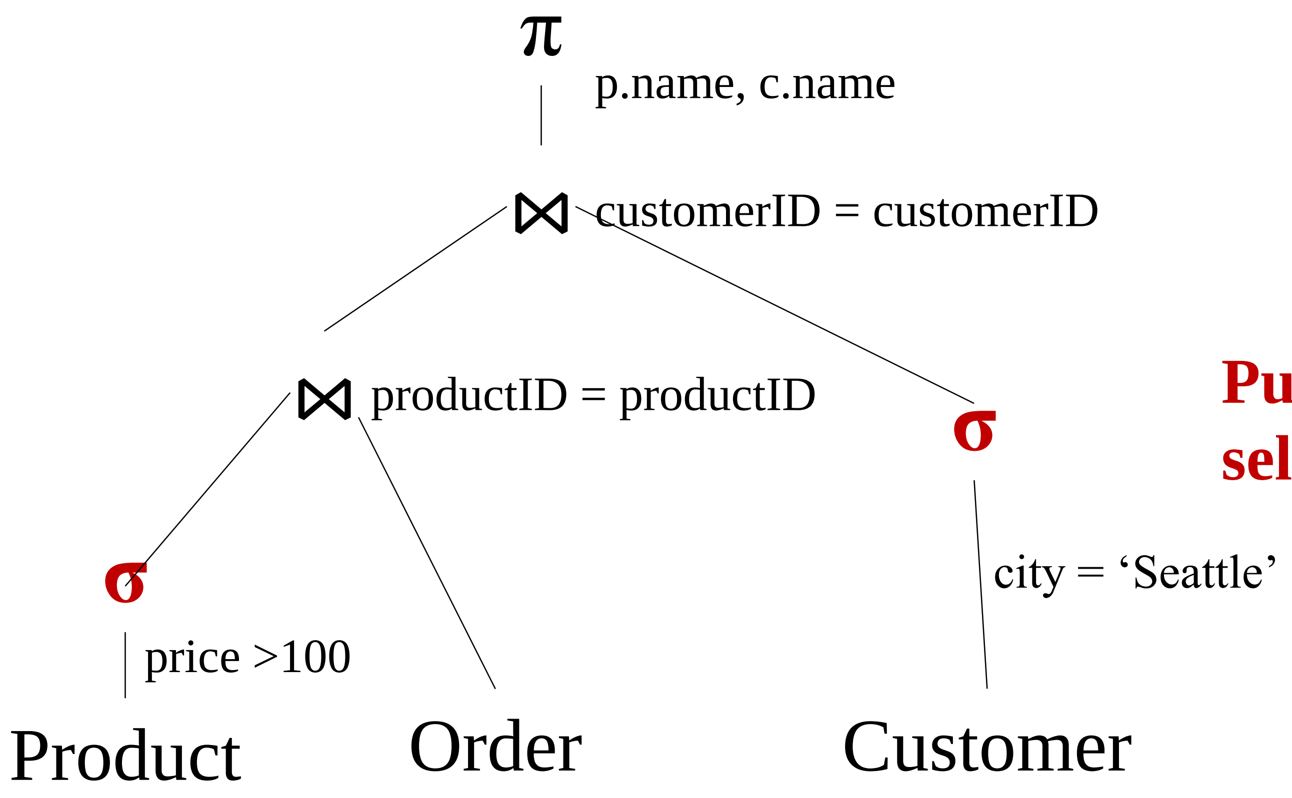
Symbolic reasoning on big tables: query plan 1

$\pi_{\text{p.name, c.name}}\ \sigma_{\text{p.price >100 and c.city = 'Seattle'}}\ ((P \bowtie O) \bowtie C)$



Symbolic reasoning on big tables: query plan 2

$\pi_{\text{p.name, c.name}} (\sigma_{\text{price} > 100} (P) \bowtie O) \bowtie (\sigma_{\text{c.city = 'Seattle'}} (C))$



Pushing selections !

In what sense is "Algebraic Optimization" "optimizing" a user query?

A. The process uses faster algorithms to perform each step.
B. The expression is executed multiple times until the optimal result is determined.
C. The process finds an equivalent expression to the original, but one that is less expensive to compute - the expression has been "optimized“.

Case in favor of Relational Database Management Systems

RDBMS provides:

- Physical and logical data independence
- Automatic indexing
- Efficient implementation of RA operators
- Query optimization
- Support and guarantees of atomic transactions

Imagine adding all these features yourself for your next data product!

What do we mean by “Big data”?

- Basic demographic information—age, sex, income, ethnicity, language, religion, housing status, and location—of every living human being on the planet can be stored in 100GB
- This would create a table of 6.75 billion rows and 10 columns.
- Should that be considered “big data”?

From “Pathologies of Big Data” Article by Adam Jacobs in the ACM Communications, August 2009.

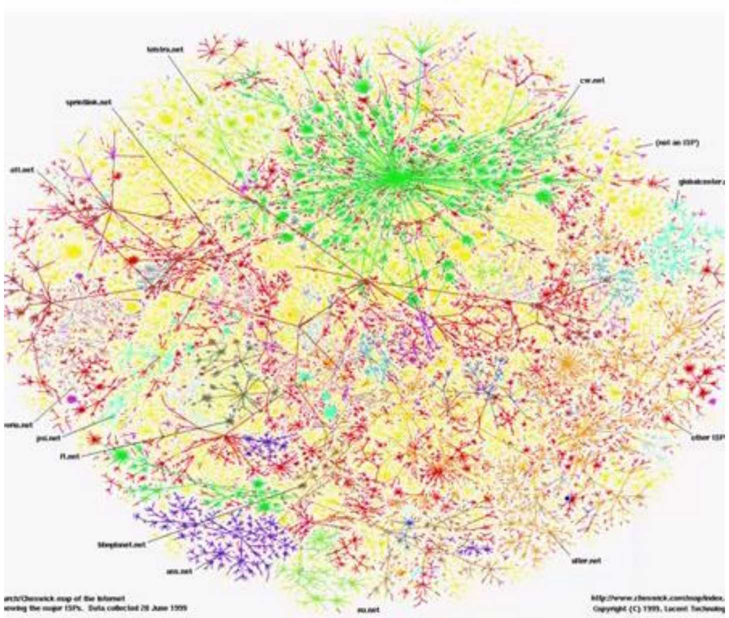
Data Units

			Roughly:
K	Kilo	2^{10}	10^{3}
M	Mega	2^{20}	10^{6}
G	Giga	2^{30}	10^{9}
T	Tera	2^{40}	10^{12}
P	Peta	2^{50}	10^{15}

Example: Volume

- The web
  - 20+ billion web pages x 20KB = 400+ TB
  - One computer can read 30-35 MB/sec from one disk – 4 months just to read the web

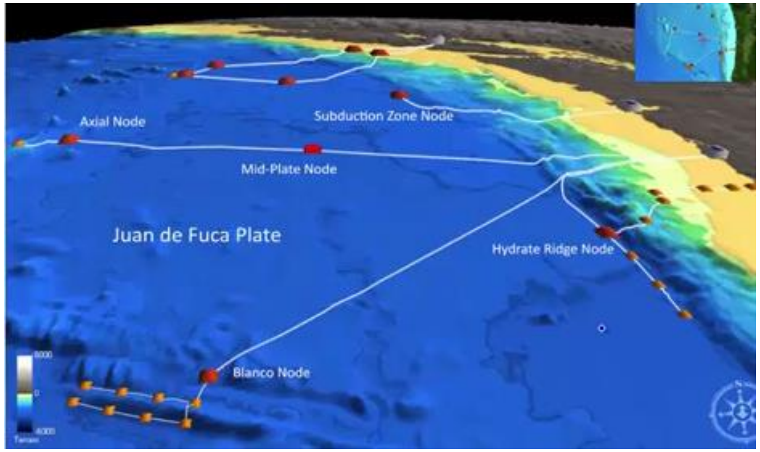
The web



Example: Variety

- NSF Ocean Observatories Initiative
  - Data is collected from satellites, vessels, censors
  - 1000 km of optic cable on the seafloor with thousands of chemical, physical, biological sensors
  - 50 TB/year of different data types

Ocean Sciences



Example: Velocity

- Large Synoptic Survey Telescope (LSST)
  - 40 TB/day
  - 40+ PB in its 10 year lifetime
  - 400 mbps sustained data exchange rate between Chile and NSCA
- Largest database in the world: World Data Centre for Climate (WDCC):
  - 100 TB of sensor data/year
  - 110 TB of simulation data/year
  - 6PB of additional information stored on tapes

Astronomy

Big Data: 4V

- **V**olume
- **V**ariety
- **V**elocity

- **V**eracity: can we trust this data?

Evolution of Science

- Empirical Science – collect and systematize facts
- Theoretical Science – formulate theories and empirically test them
- Computational Science – run automatic proofs, simulations
- **e-Science** (Data Science) – collect data without clear goal - and test theories, find patterns **in the data itself**

SLOAN DIGITAL SKY SURVEY

Science is about asking questions

Traditionally: "Query the world"

Data acquisition for a specific hypotheses

Data science: "Download the world"

Data acquired en masse in support of future hypotheses

Computational challenge

The cost of data acquisition has dropped

The cost of **processing**, **integrating** and **analyzing** data is the new bottleneck

"…the necessity of grappling with Big Data, and the desirability of unlocking the information hidden within it, is now a key theme in all the sciences – arguably the key scientific theme of our times"

F. Diebold

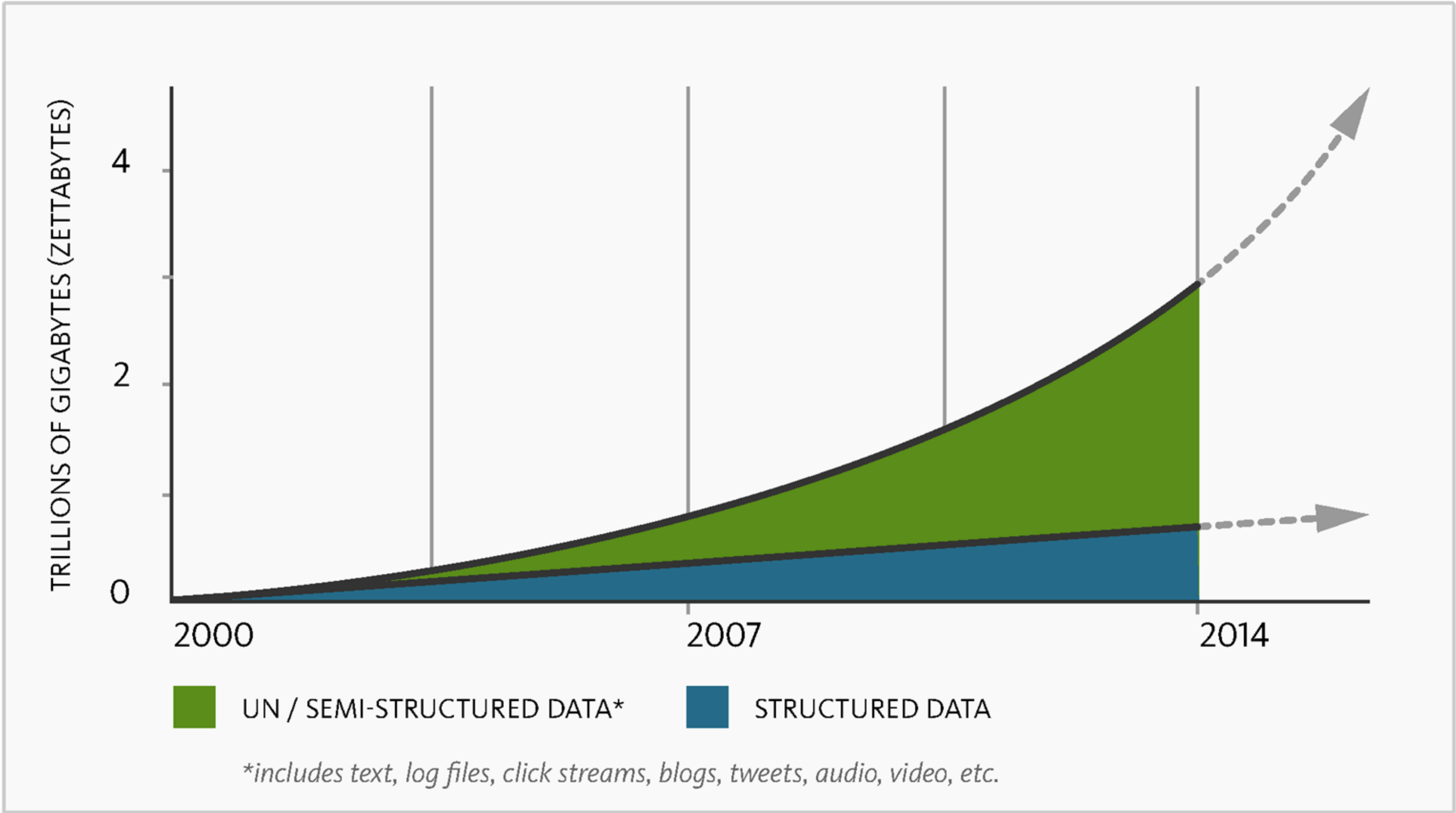
Efficient data manipulation

Poll: How much time modern scientists spend "handling data" as opposed to "doing science"?

Mode answer: 90%

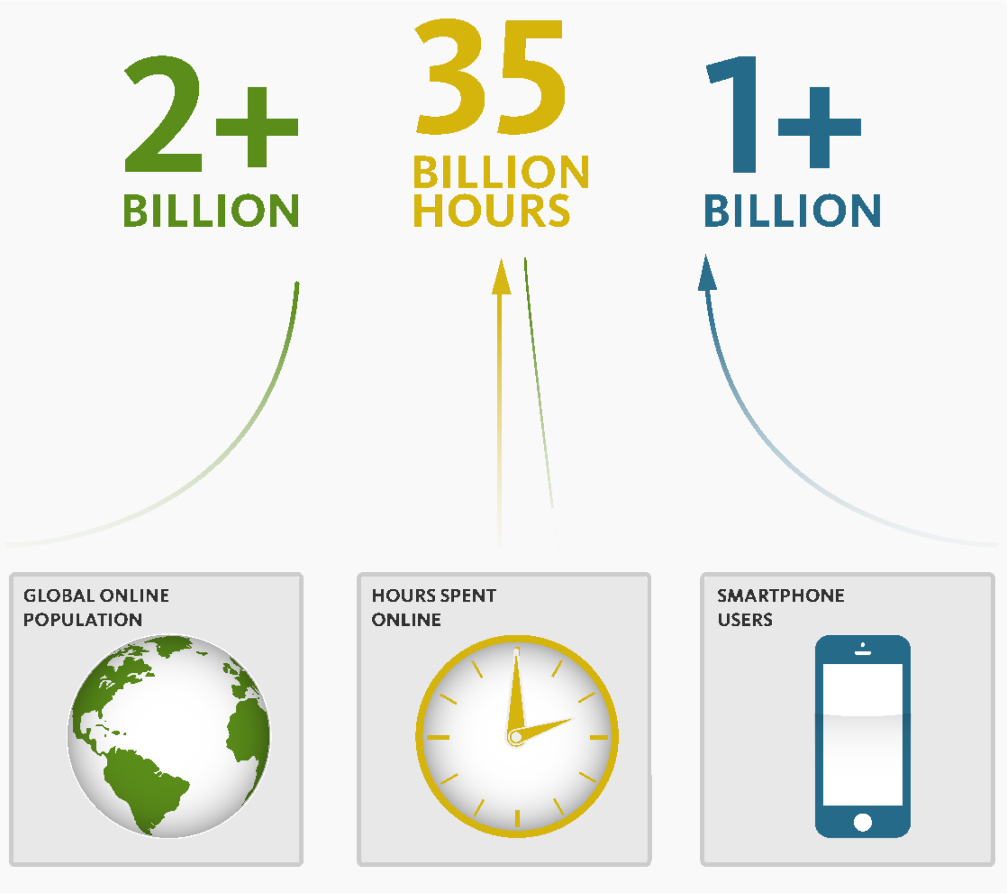
"the Next Wave of InfraSress" ***(***J. Mashey)

Current Trends: Big Data



source: http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQL-Whitepaper.pdf

Current Trends: Lots of traffic



source: http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/NoSQL-Whitepaper.pdf

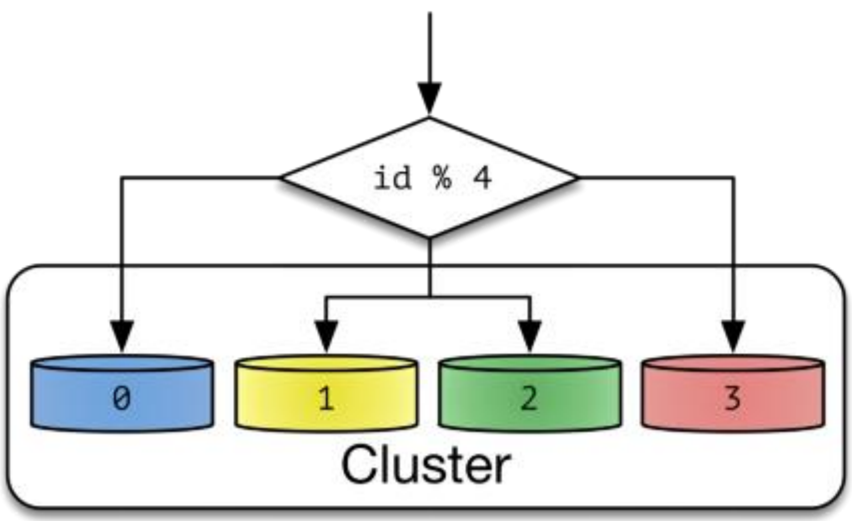
Current Trends: Cloud Computing

source: http://www.profitbricks.com/what-is-iaas

Scaling up

Two alternatives:

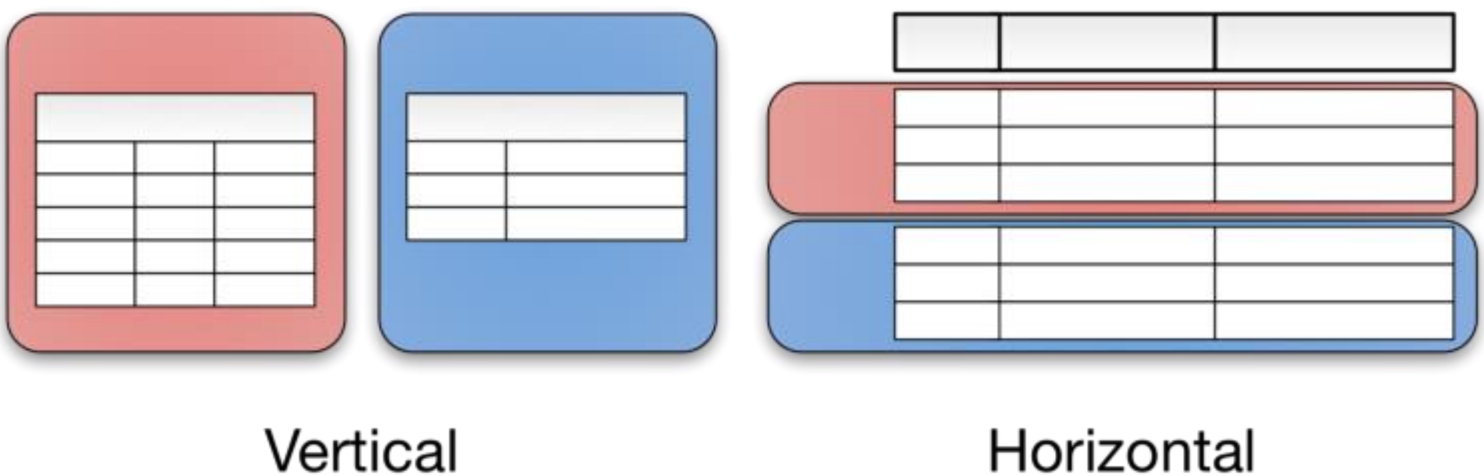
- Bigger servers
- Lots of little boxes in massive grids



Parallelism is not natural for relational databases

- Vertical: normalization, splitting into smaller tables
- Horizontal: splitting single table into multiple sets of rows

- SQL designed to run as a single node
- Both vertical partitioning and horizontal partitioning introduce performance bottlenecks



History

1980

Relational databases

1990

2000

NoSQL databases

2010

Aggregate databases:

Key-value
Document
Wide-column

Graph databases

String databases

Future?

1980

Relational databases

1990

2000

NoSQL databases

2010

Polyglot persistence

When to use RDBMS

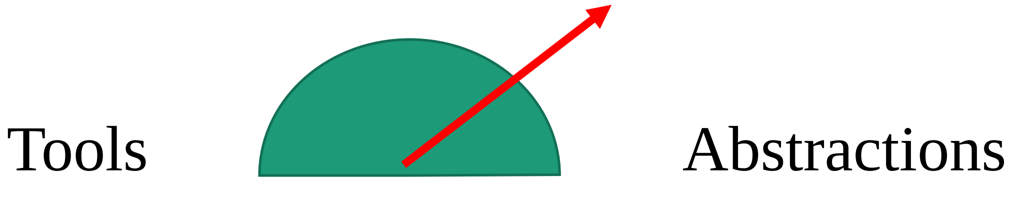
- Fast application development
- Data integrity and security is important
- Loss of data is unacceptable
- Concurrent data modification: by multiple users
- Data can be easily modeled as relations

When to consider alternative data stores

- String databases
- Audio, video databases
- Document databases
- Graph databases

This course objectives

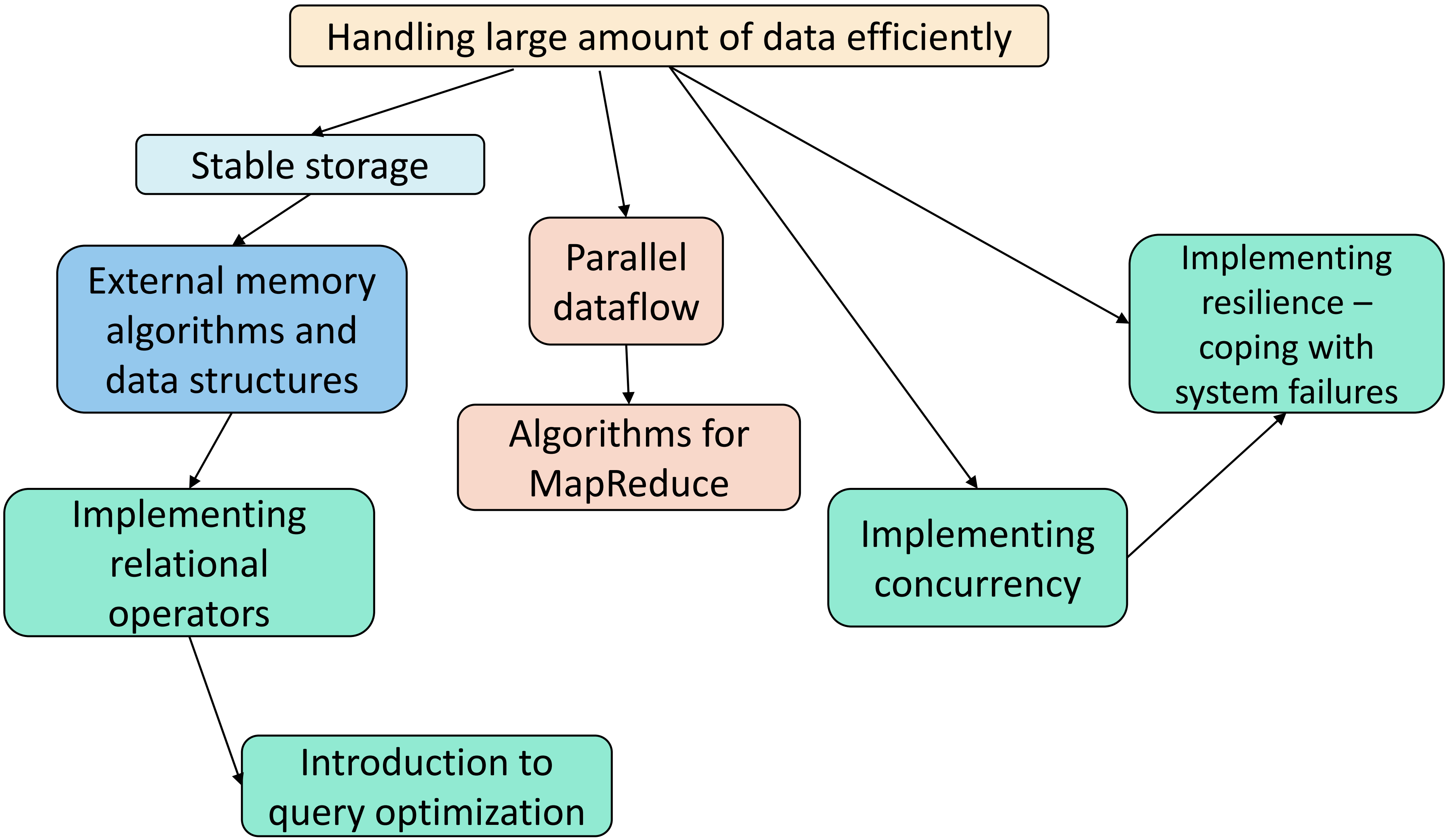
- Understand a Big-picture of different aspects of DBMS
- Experience challenges of database system implementation through programming assignments
- Learn techniques for working with big inputs
- Be able to solve system problems without reinventing the wheel – using what studied and understood



Many facets of Database studies

- Logical design
  - What kinds of information to store?
  - How to *model* data?
  - How are data items connected?
- Database programming
  - How does one express queries on the database?
  - How is database programming combined with conventional programming?
- Database system implementation
  - How does one build a DBMS

Roadmap


Handling large amount of data efficiently
Stable storage
External memory algorithms and data structures
Parallel dataflow
Algorithms for MapReduce
Implementing resilience – coping with system failures
Implementing relational operators
Implementing concurrency
Introduction to query optimization

Textbook

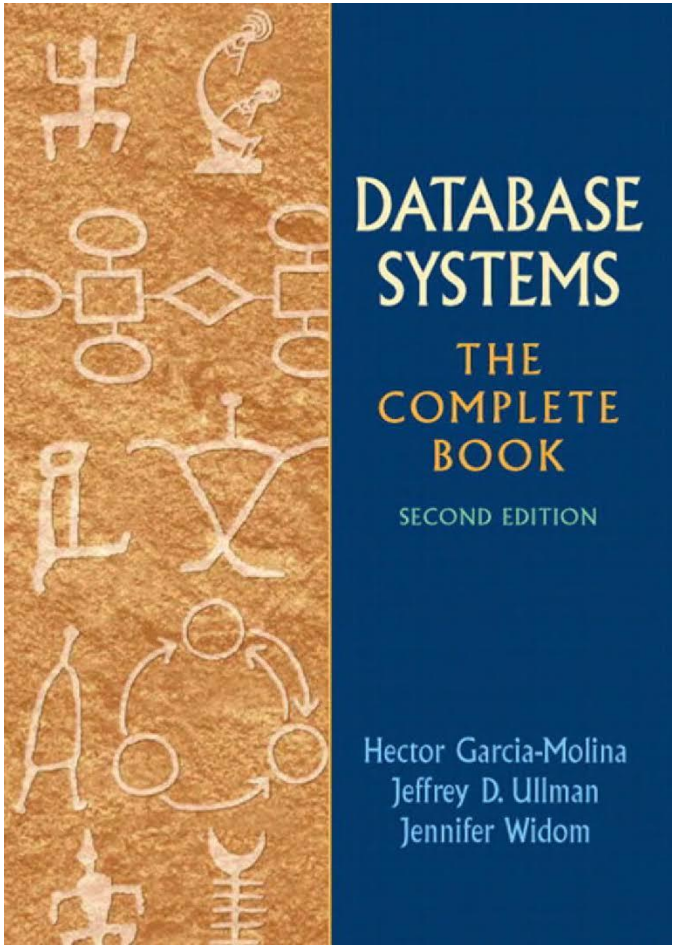
"Database Systems: The Complete Book"

by *H. Garcia-Molina,*

J. D. Ullman,

and J. Widom,

2nd Edition.



Deliverables

- 2 programming assignments: 40%
- 10 weekly tests (during tutorials): 20%
- Final exam: 40% *

*You need to score at least 50% on the final exam in order to pass the course

Bonus – for inspired

- http://worrydream.com/ExplorableExplanations/

- http://setosa.io/ev/principal-component-analysis/
- http://setosa.io/ev/eigenvectors-and-eigenvalues/
- http://setosa.io/ev/markov-chains/

- My explorable: Knapsack 01

- Plenty of algorithms to make an explorable