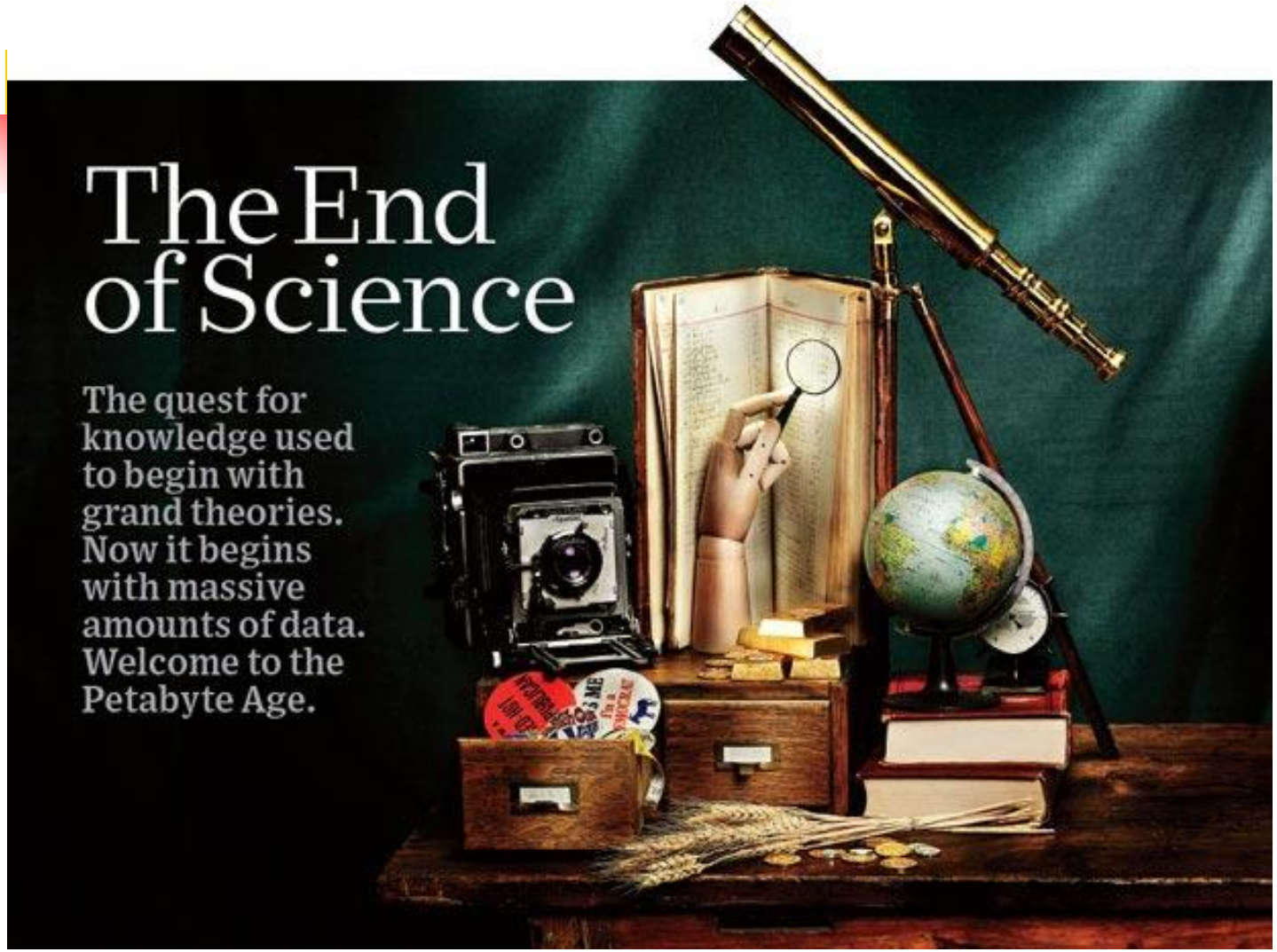
An Introduction to Modern Data Management

Damianos Chatziantoniou (damianos@aueb.gr)
Department of Management Science and Technology
Athens University of Economics and Business




Everyone is Talking about Big Data

The New York Times

The Economist

Harvard Business Review

Gartner

WORLD ECONOMIC FORUM

The President of the United States…

… and Dilbert too! So, it Must be Important


Big Data! But… What Is It? (1)

Big Data

Deep Learning

Data Science

Hadoop

Data Mining

Text Analytics

Visualization

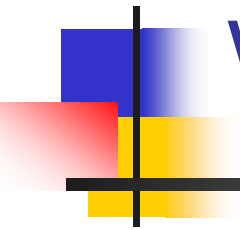

Big Data! But… What Is It? (2)

- Big Data is not about characterizing a data set. It's about a process, a goal, a future economy!

Use Data to Create Value Added Services within an Organization or a Sector


Outline

- What can you do with Big Data?
- What is Big Data / The Big Data Lifecycle
- Data Management: A Retrospective, Fundamentals
- Data Management: Big Data Systems
- Conclusions



What Can you Do with Big Data?

What can you do with Big Data? [1a]

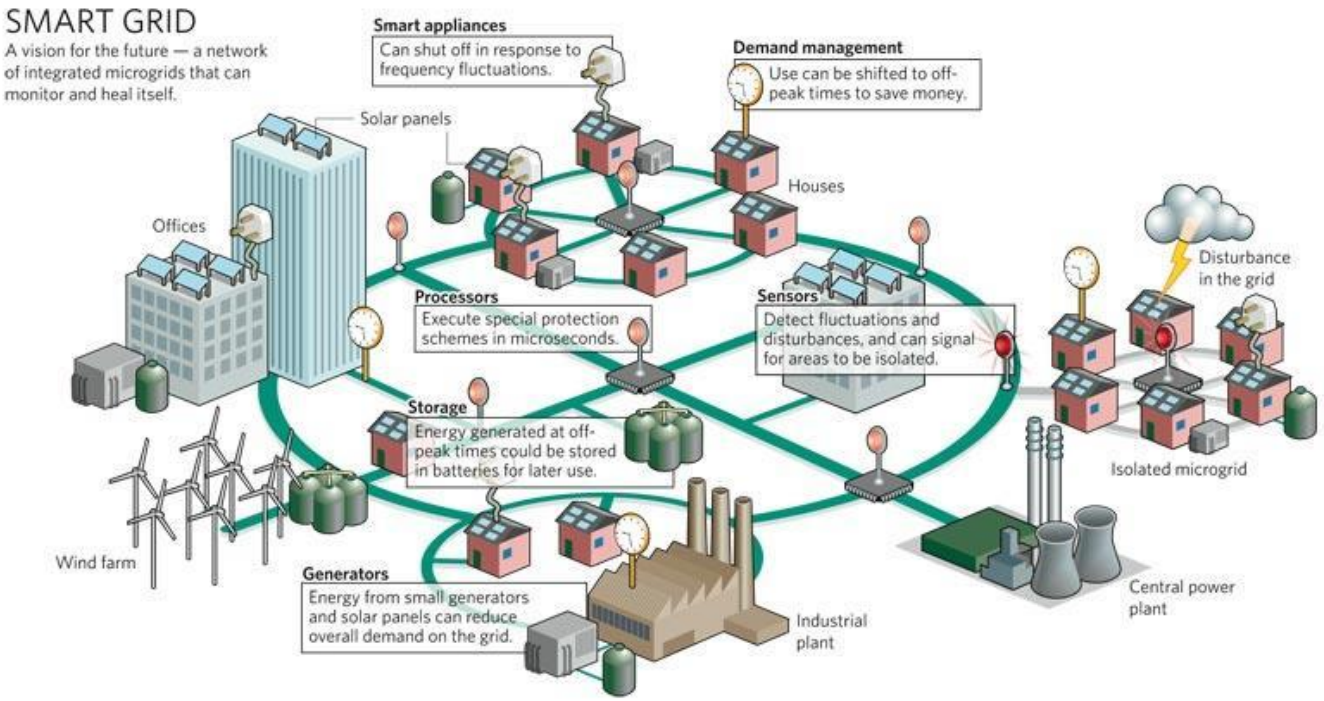
- Become president of USA: “Obama campaign did 66,000 simulations every single night of the presidential campaign”

What can you do with Big Data? [1b]

- Become president of USA (updated): “Use (stolen) facebook data to understand behavior and patterns of some 85M users in order to phrase the same message in a user-specific manner – Cambridge Analytica”
  - https://www.nytimes.com/2018/03/17/us/politics/cambridge-analytica-trump-campaign.html

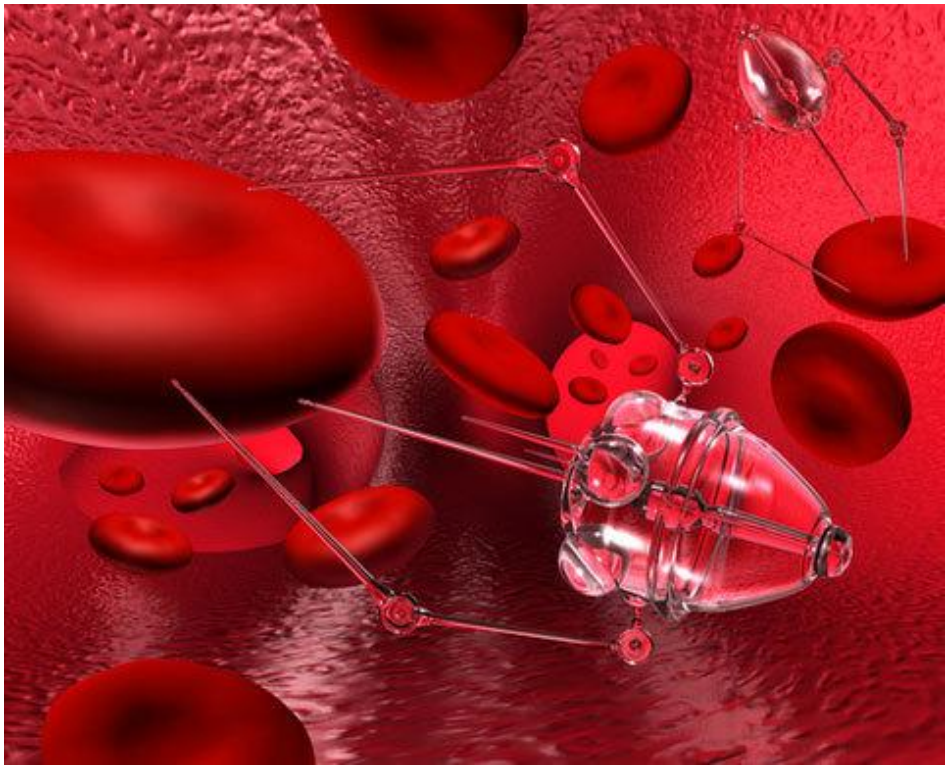
What can you do with Big Data? [2]

- Better Energy Management



What can you do with Big Data? [3]

- Better Health



© Zeitgeist Australia

What can you do with Big Data? [4]

- Better Education

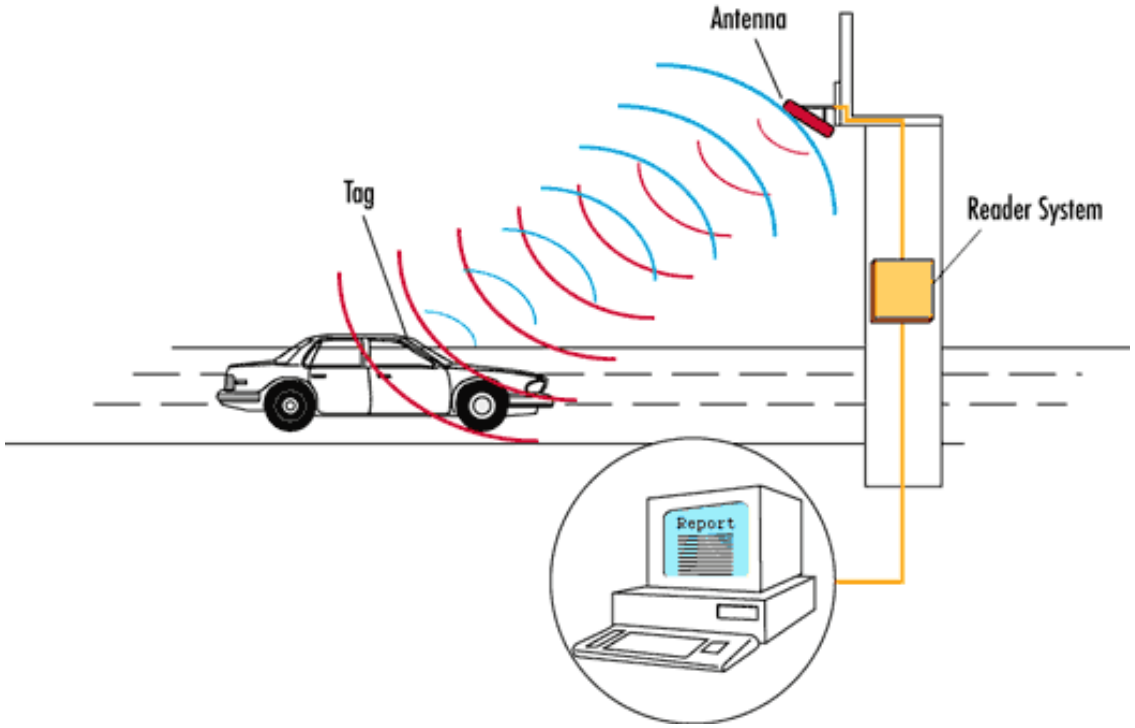
- Daphne Koller, AI Stanford Prof., TED talk: *"What we're learning from online education"*
  - Each keystroke, quiz, peer-to-peer discussion and self-graded assignment builds an unprecedented pool of data on how knowledge is processed.

What can you do with Big Data? [5]

- Better Sport

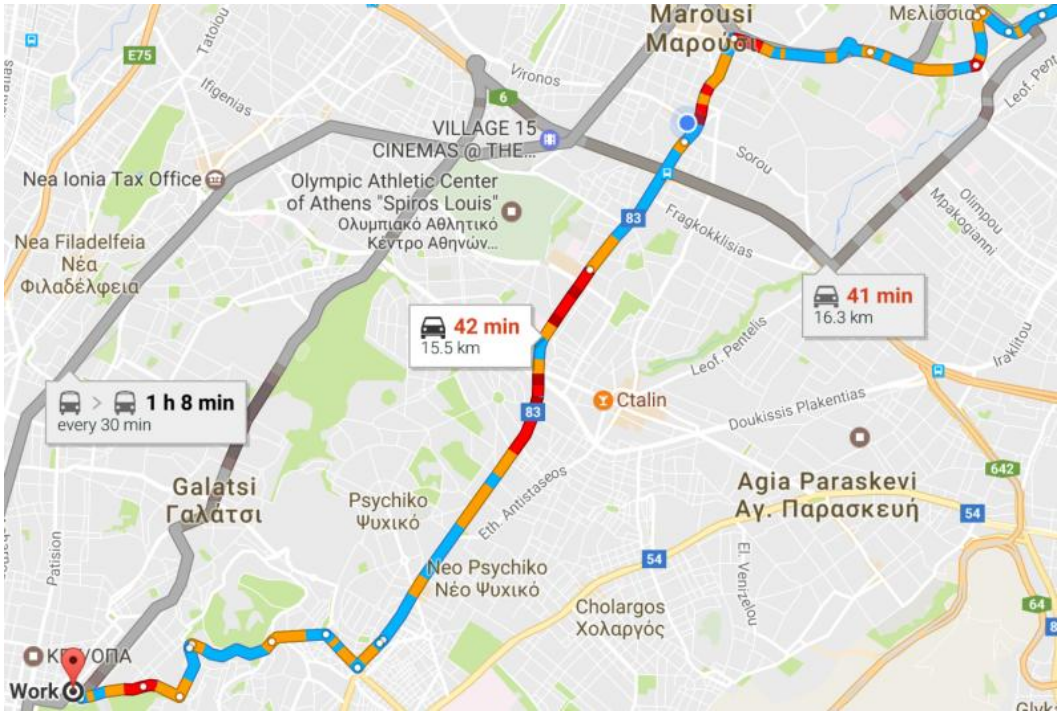
What can you do with Big Data? [6a]

- Better Traffic



What can you do with Big Data? [6b]

- Better Traffic

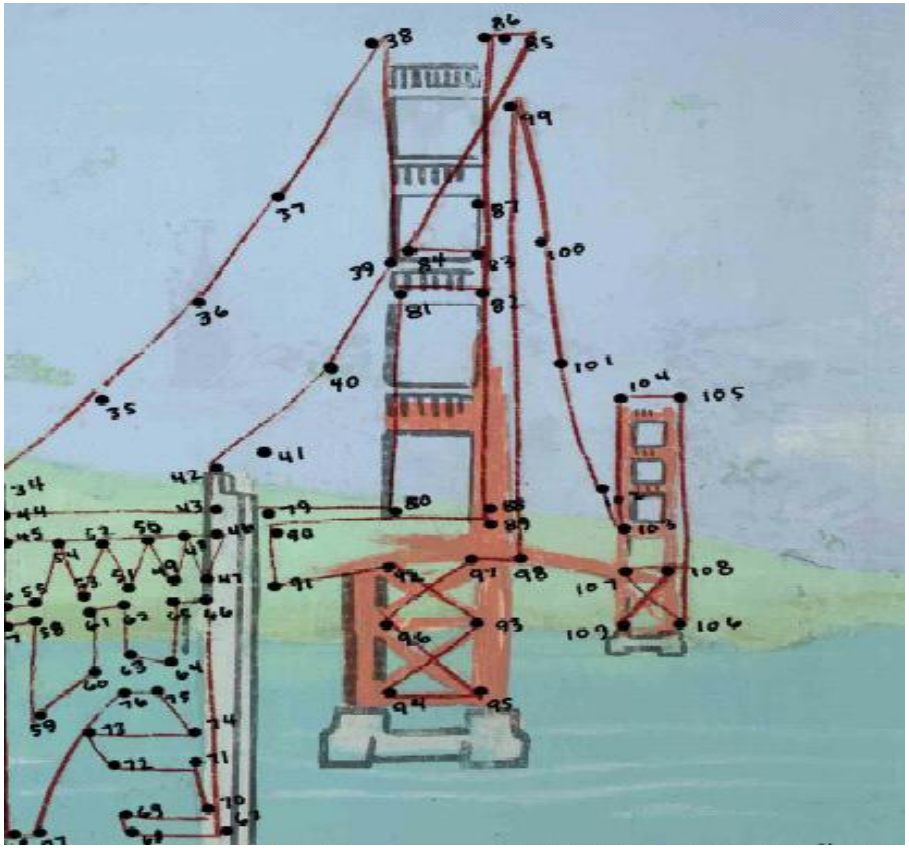


What can you do with Big Data? [7]

- Better Supply Chain, Consumer Behavior

What can you do with Big Data? [8]

- Safer World



© Wired Magazine

What can you do with Big Data? [9]

- Rank Anyone Socially: Klout

© Wired Magazine


What can you do with Big Data? [10]

- Trade stocks algorithmically
  - Financial news in machine-readable format
  - Dow Jones created in 2007 an “elementized” direct newsfeed to customers’ trading platforms. Since then this service has greatly expanded
  - That move spawned an industry of “news analytics” dominated by companies like Raven-Pack, which turns 100,000 news articles a day into trade-ready data
  - Algorithms Take Control of Wall Street (Wired, Jan. 2011)
  - Twitter Mood Predicts the Stock Market

What can you do with Big Data? [11]

- Articles written by computers
  - sport event updates, previews of corporate earnings, summaries of the presidential horse race, reviews
  - **input:** twitter posts, financial news, blogs, apps for iPhone/Android (e.g. Game Changer).
  - **output:** well-written articles – you can even train the system for different writing styles.
- A first step toward a news universe dominated by computer-generated stories.
- Narrative Sciences, Automated Insights

What can you do with Big Data? [12]

- Computer Vision, Image Recognition

Photo CC-BY-NC by stevekc

(a)

Photo CC-BY-NC by edwin.11

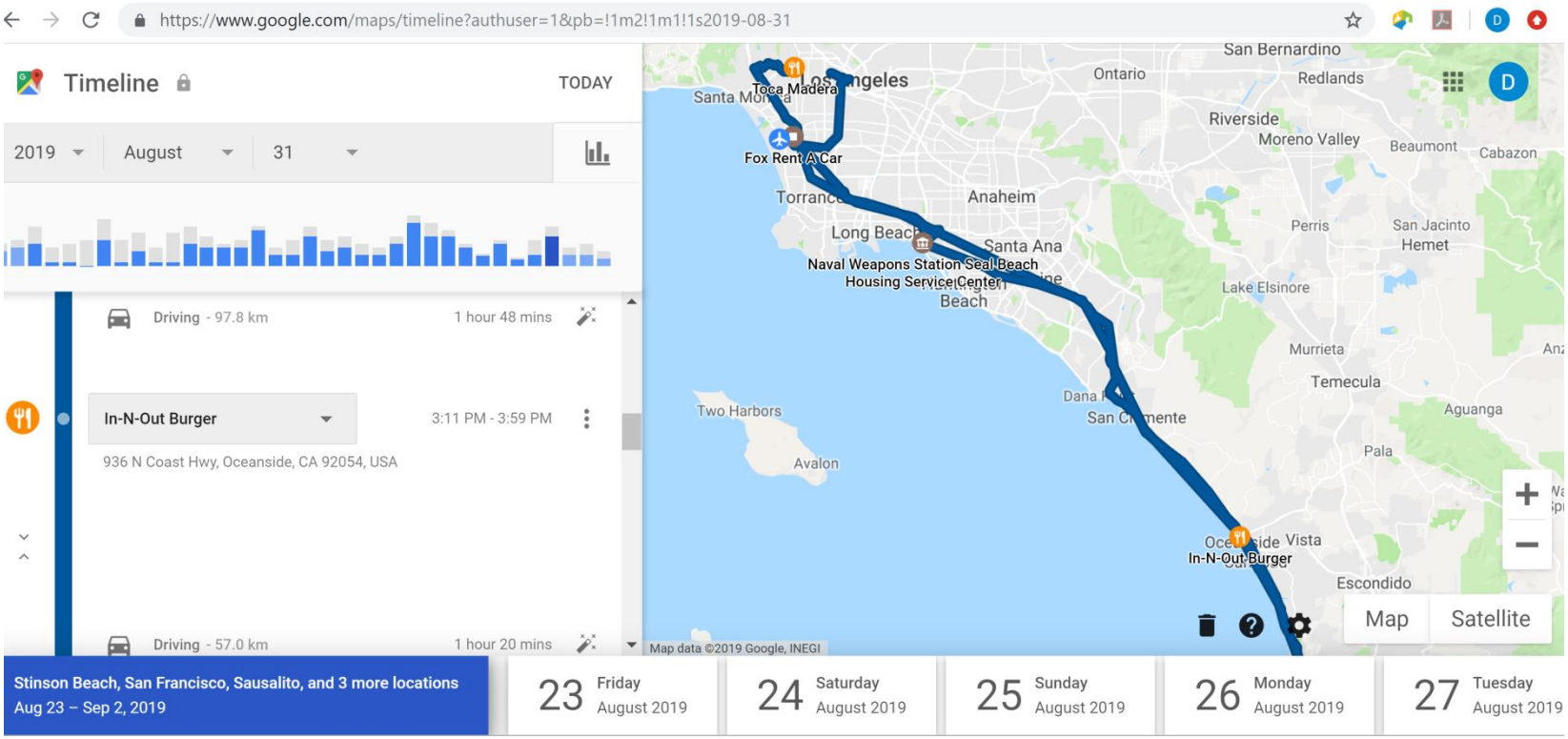
What can you do with Big Data? [13]

- Autonomous Driving

- *Ethical Dillema:*
- A car has to decide whether to run over a group of schoolchildren or plunge off a cliff, killing its own occupants. Who is to decide?
  - https://theconversation.com/the-everyday-ethical-challenges-of-self-driving-cars-92710

What can you do with Big Data? [14]

- Privacy Issues, Google's Timeline



What is Big Data?
The Big Data Lifecycle


BIG data: Size *Matters*

- Facebook
  - 1.5B users
  - 5B likes/daily, 300M photos/daily
  - process over 2.5B pieces of content (500+ TB) daily
  - 100+ PB in data
  - https://zephoria.com/social-media/top-15-valuable-facebook-statistics/
- Google, LinkedIn, Twitter
- Wal-Mart, Citibank, Telcos, etc.


big DATA: Variety, Variety, Variety

- Facebook
  - personal info (database)
  - messages, posts, status updates (text)
  - pictures (images)
  - videos
- Twitter
  - tweets (text)


Big Data is produced Fast!

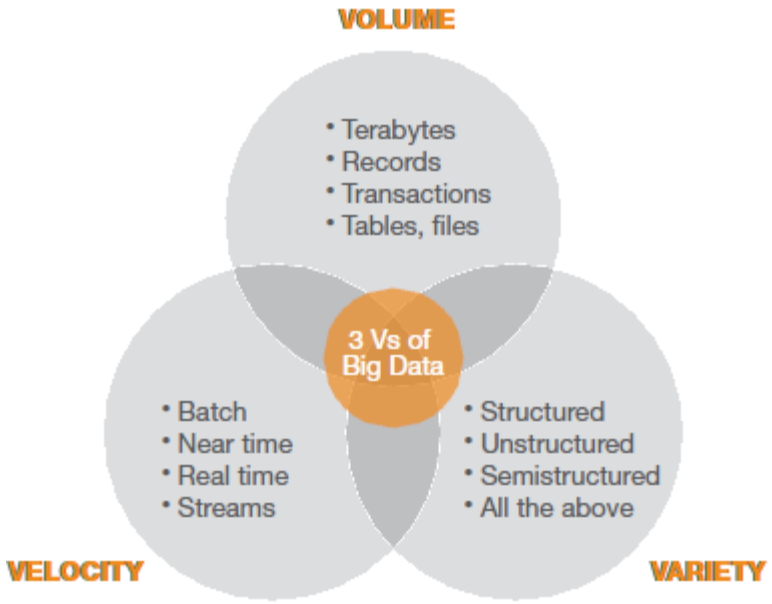
- A jet engine produces 5TB of data / 30 mins
  - 29K aircrafts – at least 2 engines each
- Financial data
  - ticks, buy/sell, news
- Sensors, sensors, sensors
  - smartphones
  - cars
  - manufacturing


So… What is Big Data?

- Manage, analyze and mine data, no matter how ***large*** it is, what ***format*** it has and how ***fast*** is produced.

Big Data – The 3Vs

- Big data is not just large data. It's also diverse data types and streaming data.



Source: The TDWI Best Practices Report, "Big Data Analytics", 4th Quarter 2011


Key Technologies – Data Management

- Aspects of Big Data systems:
  - Volume
    - new architectures, data models, distributed / parallel processing, main memory, cloud
    - availability, fault-tolerance, security, partition
  - Variety, unstructured data
    - have to manage and process this ➔ data management
    - have to bring structure out of this ➔ AI, machine learning
  - Velocity
    - have to manage data streams and deal with stream problems ➔ data management
    - sampling, sketches, mining ➔ statistics


Key Technologies – Data Management

- New RDBMS technologies
  - Main-memory
  - Column-oriented
  - Massive parallel processing (MPPs)
- MapReduce, Hadoop and Related Technologies
- NoSQL Systems
  - Key-value engines, Document stores, Graph databases
- Data Stream Engines


Key Technologies – Statistics

- Regression
- Classification
- Statistical Learning
- Dimension Reduction


Key Technologies – AI/Data Mining

- Supervised Learning
- Unsupervised Learning
- Image/Video/Speech processing
- Deep Learning
- Goal: From unstructured data to structured data


Key Technologies – Text Analytics

- Text mining tasks:
  - text categorization
  - text clustering
  - concept/entity extraction
  - sentiment analysis
  - document summarization
- Examples:
  - Finance, Algorithmic Trading, News Writing, Predicting Movies Success, Market Research


Key Technologies – Graph Theory

- Fundamentals in Graph Theory
- Network Measures
  - Centrality
  - Transitivity and Reciprocity
  - Similarity
  - Balance and Status

Key Technologies – Business/Legal

- Business Process Management
- Data Privacy
- Data Protection
- Innovation and Entrepreneurship
- Story Telling / Presentation Skills

Key Technologies – Ethics

- As machine learning becomes more powerful, the field's researchers increasingly find themselves unable to account for what their algorithms know — or how they know it.
- Can AI Be Taught to Explain Itself?
  - https://www.nytimes.com/2017/11/21/magazine/can-ai-be-taught-to-explain-itself.html
- Two models in AI:
  - Fact-base reasoning
  - Connectionist model


Key Technologies – Visualization

- Theory and techniques to visualize data and patterns
- Hans Rosling video


Key Technologies – Domain Expertise

- Marketing Analytics
- Telecom Analytics
- Financial Analytics
- Healthcare Analytics
- Energy Analytics
- Behavioral Analytics
- …


The Main Phases

- Data Extraction / Data Preparation : 80%
- Analysis : 20%
- Interpretation and Visualization: 80%

Big Data Lifecycle – Main References

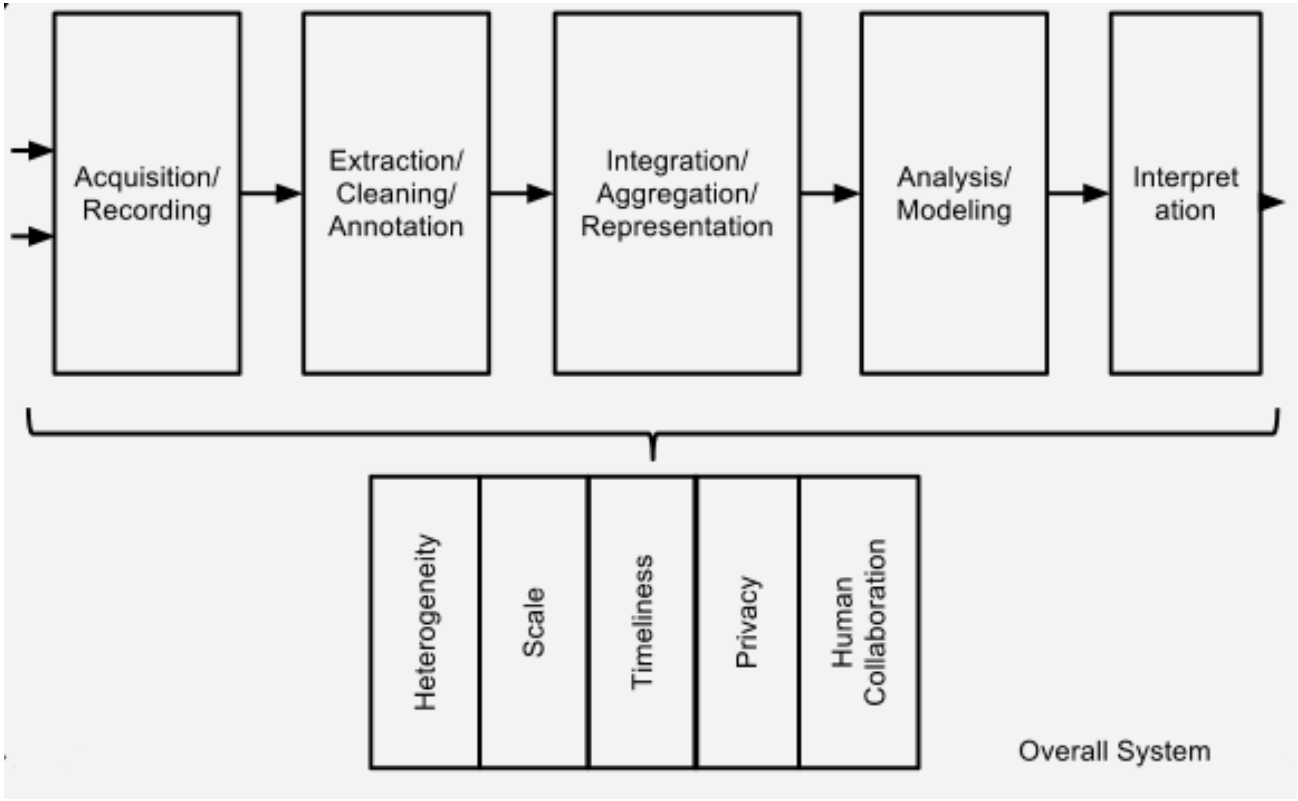
- "Challenges and Opportunities with Big Data"

  Divyakant Agrawal, UC Santa Barbara - Philip Bernstein, Microsoft - Elisa Bertino, Purdue Univ. - Susan Davidson, Univ. of Pennsylvania - Umeshwar Dayal, HP - Michael Franklin, UC Berkeley - Johannes Gehrke, Cornell Univ. - Laura Haas, IBM - Alon Halevy, Google - Jiawei Han, UIUC - H. V. Jagadish, Univ. of Michigan (Coordinator) - Alexandros Labrinidis, Univ. of Pittsburgh - Sam Madden, MIT - Yannis Papakonstantinou, UC San Diego - Jignesh M. Patel, Univ. of Wisconsin - Raghu Ramakrishnan, Yahoo! - Kenneth Ross, Columbia Univ. - Cyrus Shahabi, Univ. of Southern California - Dan Suciu, Univ. of Washington - Shiv Vaithyanathan, IBM - Jennifer Widom, Stanford Univ.

- "The Beckman Report on Database Research"

  Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H.V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jeffrey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Re, Dan Suciu, Michael Stonebraker, Todd Walter, Jennifer Widom

Big Data – Phases and Challenges



Source: Challenges and Opportunities with Big Data


Data Analysis Lifecycle – An Example (1)

- Example – A telecom environment:
  - Define business goals and feasibility
    - e.g. churn prediction
  - What data am I having? Structured? Unstructured? Streams?
    - e.g. call details, billing, traffic, call center, emails
  - Can I create/find additional data sources? How? Where? Can I use them? What business processes do I have to launch?
    - e.g. create an app to offer something and collect data
  - How can I integrate all these data sources to prepare input for analysis? What about data cleaning and transformations?
    - E.g. use Hadoop to collect in one system


Data Analysis Lifecycle – An Example (2)

- Example – A telecom environment (cont.):
  - What kind of analysis?
    - e.g. classification, regression, variable selection
  - What systems and tools will I use?
    - e.g. RDBMS, HDFS/MapReduce, Spark, Java, R, Python
  - How am I going to present findings? What kind of visualizations are appropriate?

Data Management:
A Retrospective, Fundamentals


Databases – History (1)

- 1950s: Cards/Tapes – Batch processing
- Early 1960s:
  - 1st general purpose DBMS: Integrated Data Store
  - Network data model
  - Charles Buchman, 1st Turing Award (1973)
- Late 1960s:
  - Information Management System (IMS), IBM
  - Hierarchical data model

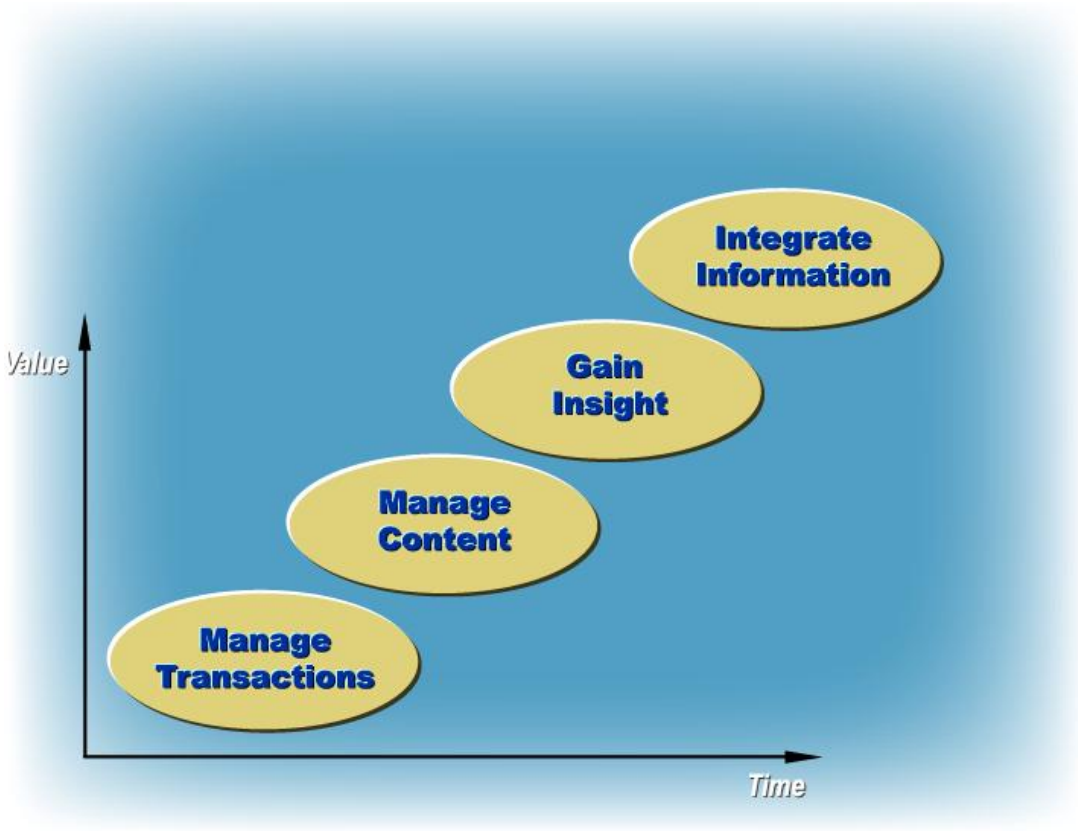

Databases – History (2)

- 1970s: Laying down the fundamentals
  - Relational Modeling, SQL, Optimization
  - System R (IBM), Ingres (Berkeley)
  - Edgar Codd, 2nd Turing Award (1981)
- 1980s: Making systems efficient
  - Transaction processing, Concurrency, Recovery
  - Parallel databases
  - Jim Gray, 3rd Turing Award (1999)


Databases – History (3)

- 1990s: Intelligent use of data
  - Data warehousing / Data analysis
  - Data mining
  - Object-Oriented DBs, Temporal DBs, GIS, etc.
  - The Asilomar Report on Database Research
- 2000s: Web, new applications, data explosion
  - The Lowell Database Research Self-Assessment
  - The Claremont Report on Database Research

Data Management Systems – Evolution

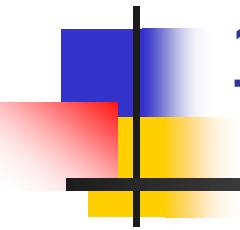



Databases – DBMS (1)

- Database Management Systems (DBMS)
  - Collection of interrelated data
  - Set of programs to access the data
  - DBMS contains information about a particular enterprise
  - DBMS provides an environment that is both convenient and efficient to use.


Databases – DBMS (2)

- DBMS addressed the following problems:
  - Data redundancy and inconsistency
  - Difficulty in accessing data
  - Data isolation (e.g. multiple files and formats)
  - Integrity problems (e.g. primary keys)
  - Atomicity of updates
  - Concurrent access
  - Security



1970 - 1980 : The Fundamentals

Data Modeling,

Relational Algebra,

Query Languages,

Query Processing


Databases – Data Modeling (1)

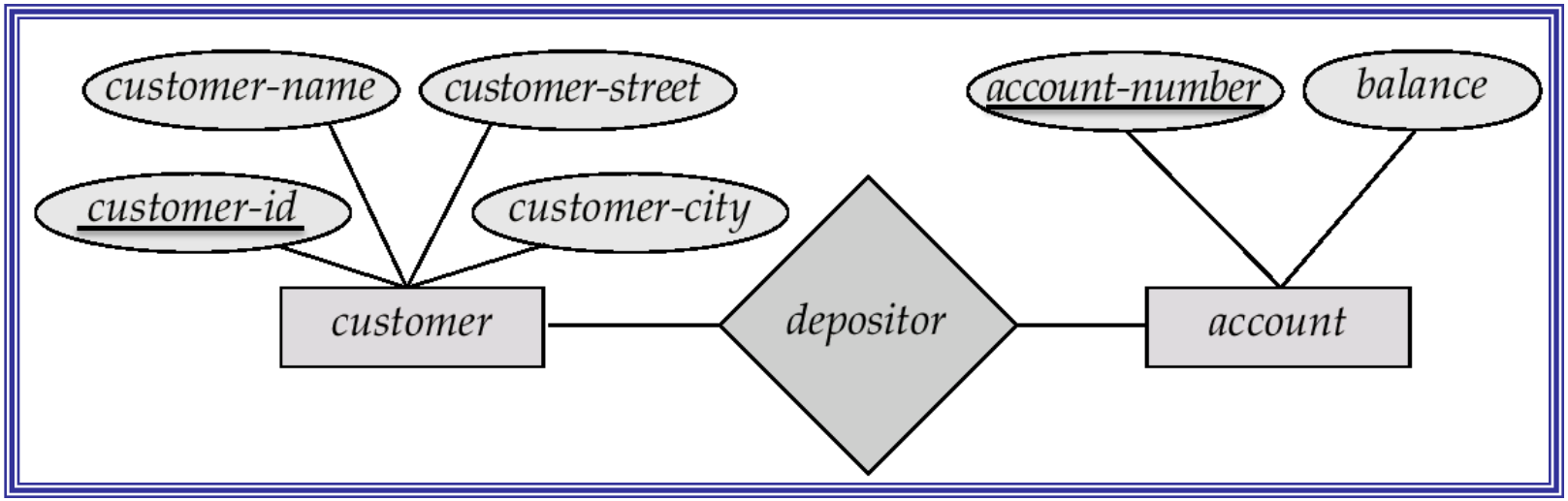
- Model: a set of constructs *and operations* to describe a real-world situation (mathematics, physics, chemistry… even art)
- Data Model: a model to describe
  - data
  - data relationships
  - data semantics
  - data constraints


Databases – Data Modeling (2)

- Data models:
  - Entity-Relationship
  - Object-Oriented
  - Relational
  - Semantic networks
- Data models serve a twofold purpose:
  - team members' interactions ("understand")
  - used as a design document ("contract")

Databases – Data Modeling, E-R Diagrams

- Entity-Relationship model: consists of entities, relationships, attributes and constraints. The final document is called an E-R diagram.




Databases – Data Modeling, Relational (1)

- Relational model: the only modeling construct is a table (relation) having columns.
- The table construct is used to model both entities and relationships.
- The columns of a table correspond to the attributes of the entities/relationships.
- Conceptual (E-R) vs. Logical (Relational)

Databases – Data Modeling, Relational (2)

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table


Databases – Levels of Abstraction

Conceptual/Logical Level:
How data is described using some data model

Physical Level: How data is stored on hardware


Databases – Query Languages

- Query language: a way to express queries over a data set described in some data model.
- Query languages:
  - procedural: specify **how** the answer of the query should be computed (e.g. java, C)
  - declarative: specify **what** the answer of the query is
- SQL is the most widely known query language
- Other: Query by Example (QBE)


Databases – Query Languages, SQL (1)

```
SELECT A1,A2, …,An
FROM T1, T2, …, Tk
WHERE <condition>
```

- Select columns A1, A2, …,An
  from tables T1, T2, …, Tk
  for these rows that satisfy the <condition>


Databases – Query Languages, SQL (2)

- For each bank account, show the name of the customer and the balance of the account.

```
SELECT c.customer-name, a.balance
FROM customer as c, depositor as d, account as a
WHERE c.customer-id = d.customer-id AND
          a.account-number = d.account-number
```


Databases – Query Languages, SQL (3)

- Count the customers for each city.

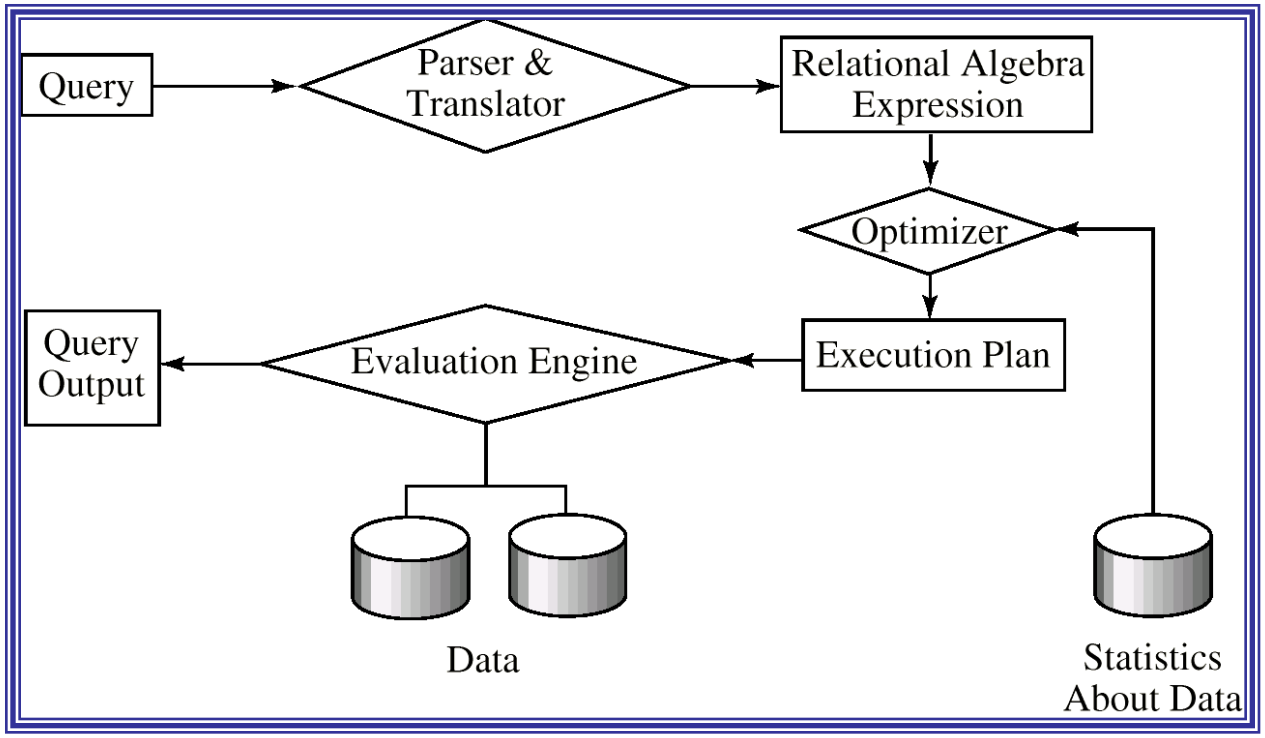
```
SELECT customer-city, count(*)
FROM customer
GROUP BY customer-city
```

- This query is *very* different than the previous one
- Think: show the number of customers per city as percentage of the total number of customers.


Databases – Relational Algebra

- Algebra: A set equipped with operations + closure property.
  - Example: $\mathbb{R}$ (real numbers) with +, *, but not $\sqrt{}$
- Relational algebra: tables with operations:
  - selection (σ): filters rows of a table ➔ table
  - projection (π): keeps columns from a table ➔ table
  - join (⋈): combines two or more tables ➔ table
- SQL translates to a relation algebra expression.

Databases – Query Processing



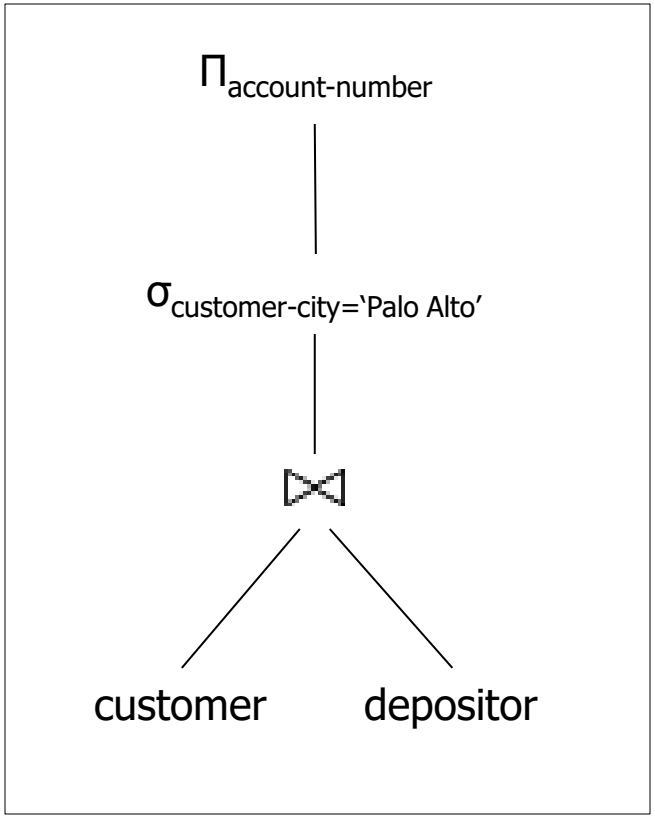

Databases – Query Optimization, Parsing

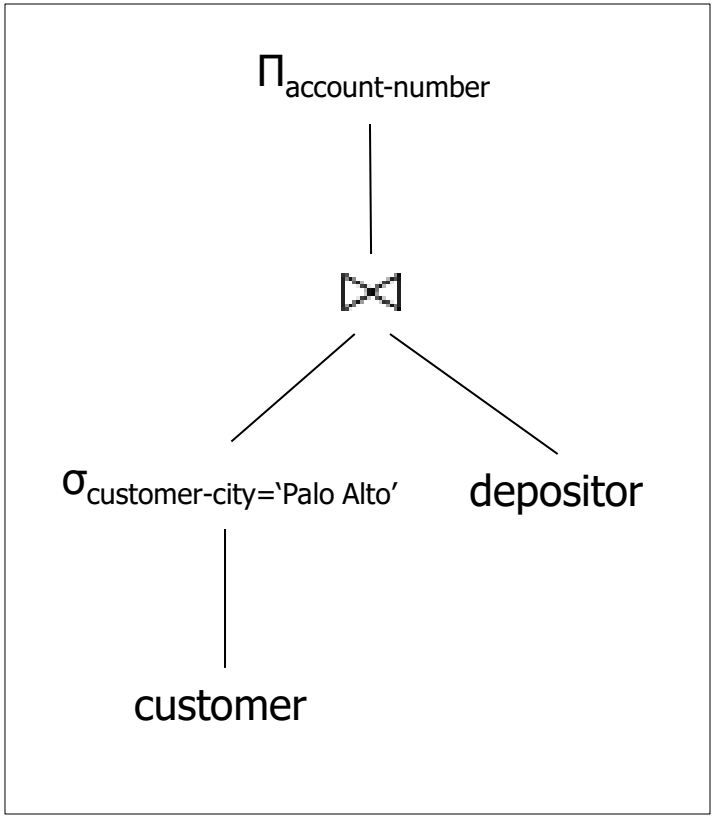
- Show the account number(s) of customers living in Palo Alto.

```
SELECT d.account-number
FROM customer as c, depositor as d
WHERE c.customer-id = d.customer-id AND
        c.customer-city='Palo Alto'
```

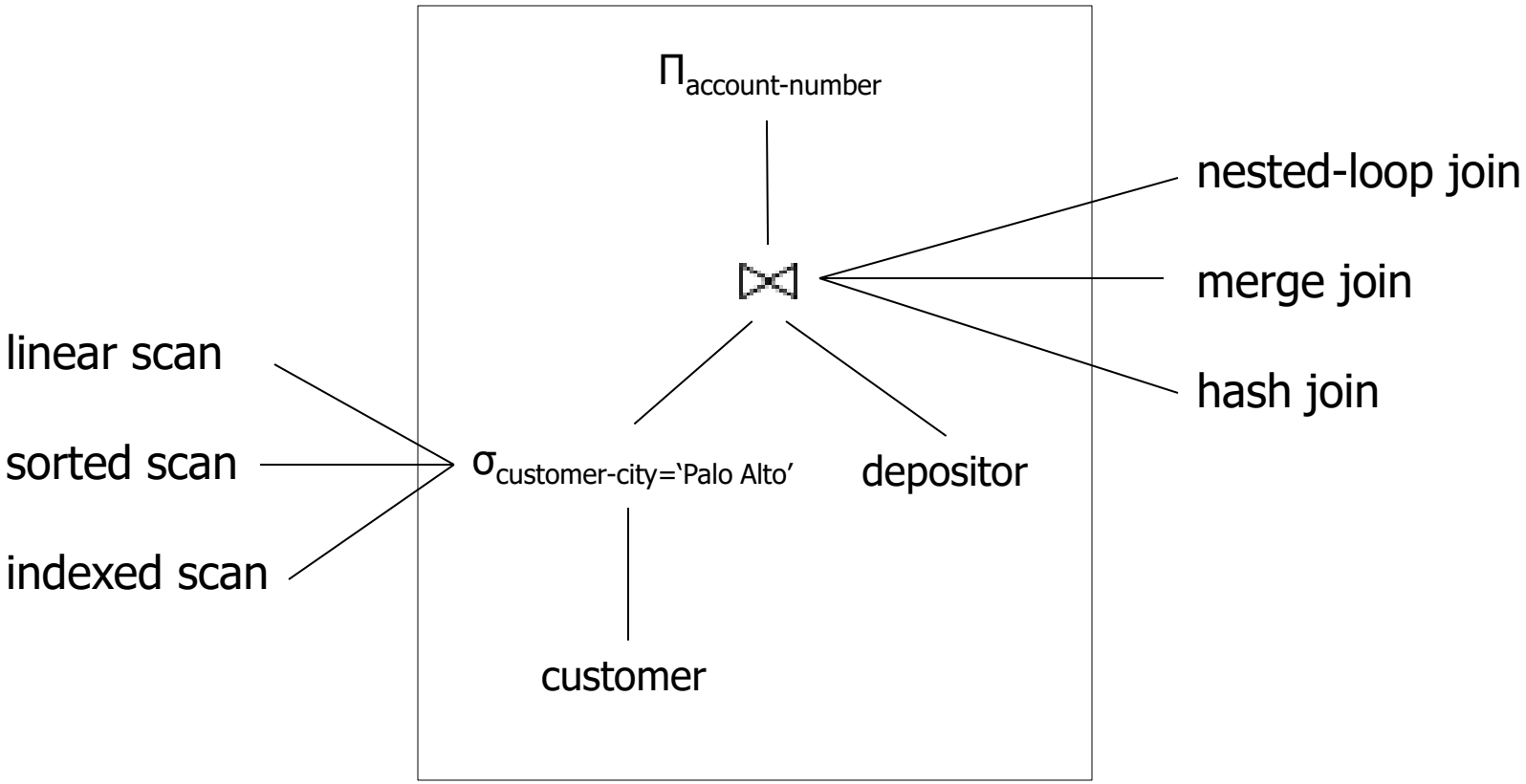
$$\Pi_{\text{account-number}} (\sigma_{\text{customer-city='Palo Alto'}} (\text{customer} \bowtie \text{depositor}))$$

Databases – Query Optimization, Algebra



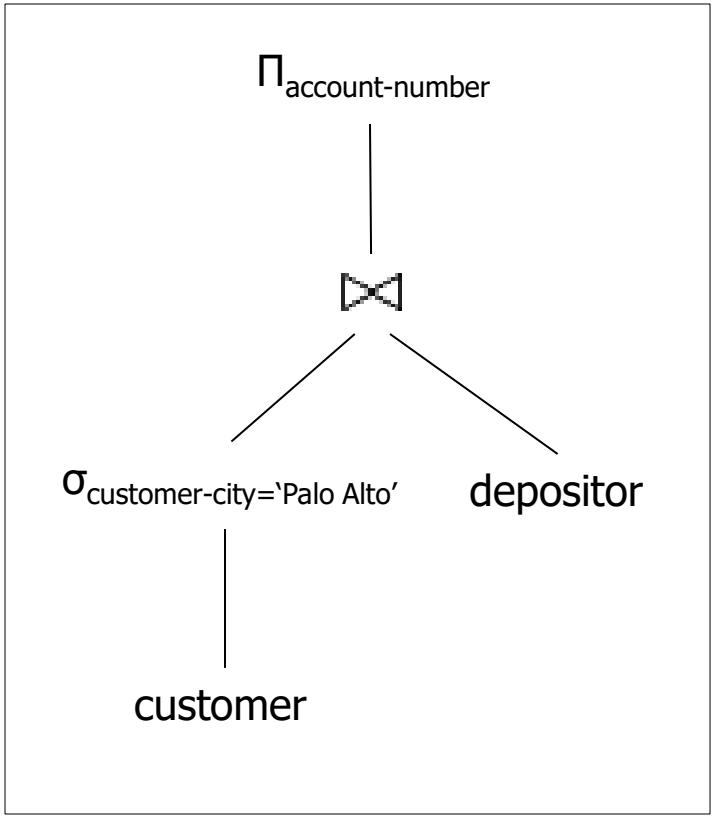


Databases – Query Optimization, Algorithms



Databases – Query Execution, Plans

- Try algebraic transformations
- Try alternative implementation algorithms
- Assign cost to each, choose the cheapest one

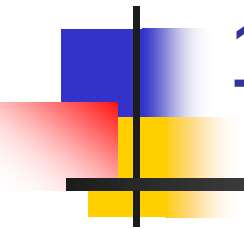



Databases – Performance, Physical Layer

- Question: How do you organize data at the storage layer
- Relational model = tables
- Disk: store in files
  - Heap files, sorted files, hash files
- Performance: Indexing
  - Clustered and non-clustered, primary and secondary
- Partitioning
  - Horizontal partitioning and vertical partitioning
- Main-memory?


Databases – Lessons to Learn

- When it comes to data management, there are always three aspects involved:
  - Data modeling (formal framework, logical modeling)
  - How to express data retrieval (query languages / APIs)
  - Query processing (evaluation, performance, other properties)
- All aspects relate to each other.



1980 - 1990: Efficient Systems

Transaction Processing,

Concurrency & Recovery,

Parallel & Distributed Databases,

Main-Memory Databases


Databases – Transactions, Definition (1)

- A transaction is a unit of program execution – *a single logical function* – that accesses and possibly updates various data items.
- Example: transfer 50€ from account A to B:

1. read(A)
2. A := A – 50
3. write(A)
4. read(B)
5. B := B + 50
6. write(B)


Databases – Transactions, Definition (2)

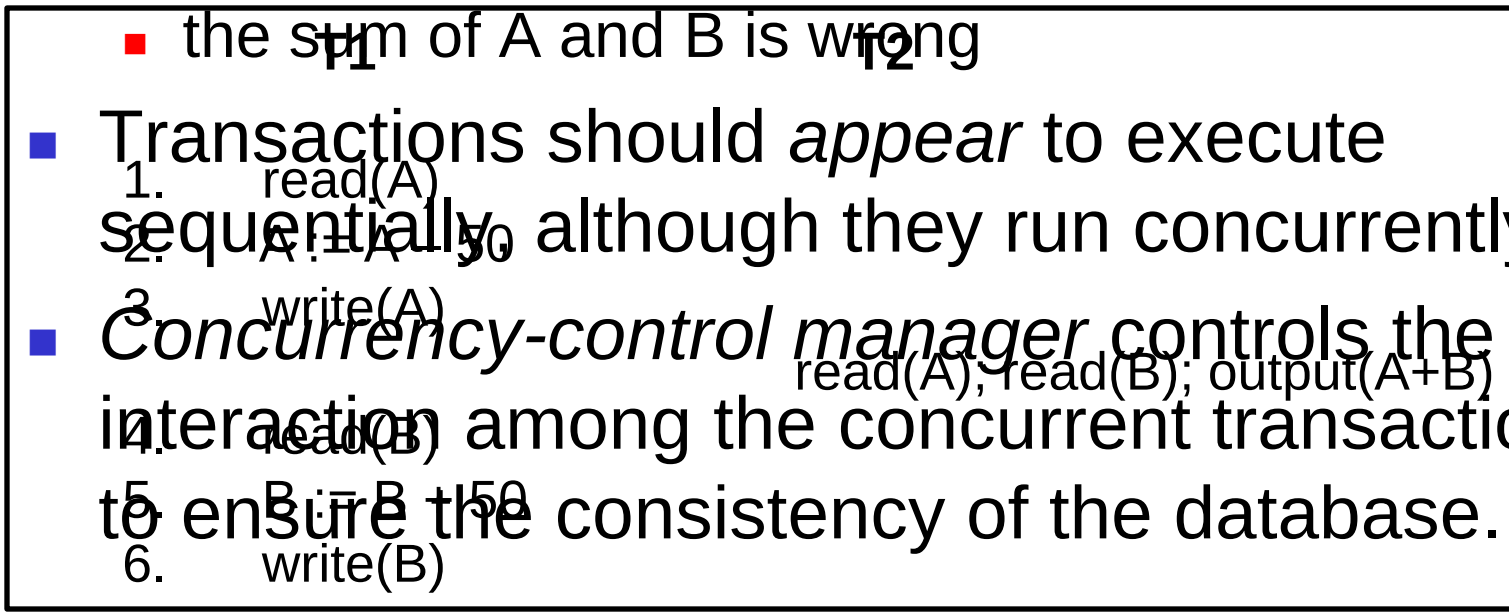
- Hardware and software failures happen.
- Transactions should be allowed to run concurrently for performance reasons.


Databases – Failures, Concept

- What will happen if system fails between steps 3 and 4?
  - Account A will have 50€ less than what it should (database will be in an inconsistent state.)
- Transactions should always either complete (commit) or undo all actions (rollback).
- *Transaction-management component* ensures that the database remains in a consistent state despite system and transaction failures.

Databases – Concurrency, Concept

- What will happen if another transaction reads A and B, between steps 3 and 4?
  - the sum of A and B is wrong
- Transactions should *appear* to execute sequentially, although they run concurrently.
- *Concurrency-control manager* controls the interaction among the concurrent transactions, to ensure the consistency of the database.




Databases – Transaction Management

- Transaction management component implements recovery algorithms.
- Recovery algorithms have two parts:
  - Actions taken during normal transaction processing to ensure enough information exists to recover from failures (logs).
  - Actions taken after a failure to recover the database contents to a state that ensures atomicity, consistency and durability.


Databases – Concurrency Control (Locking)

- A lock is a mechanism to control concurrent access to a data item (record, table, database.)
- A lock can be either a “write lock” or “read lock.”
- Lock requests are made to concurrency-control manager. Transaction can proceed only after request is granted.
- A *locking protocol* is a set of rules followed by all transactions while requesting/releasing locks.


Databases – Transactions and Overhead

- Recovery algorithms maintain a log to ensure consistency and atomicity.
- Transactions have to acquire locks to access data items and follow a protocol.
- Transaction management incurs a great *overhead* to the database system.
- Various levels of consistency.


Databases – Parallel Databases (1)

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
  - A coarse-grain parallel machine consists of a small number of powerful processors; A massively parallel machine utilizes thousands of smaller processors.
- A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries.

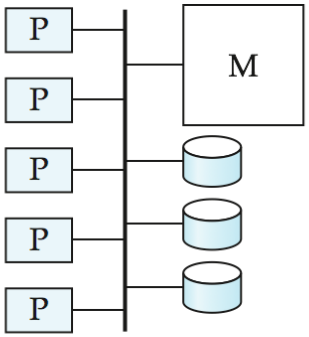

Databases – Parallel Databases (2)

- High-performance through parallelism
  - Low response time with intra-operation parallelism
  - High throughput with inter-query parallelism
- High availability and reliability by exploiting data replication
- Extensibility with the ideal goals
  - Linear speed-up
  - Linear scale-up

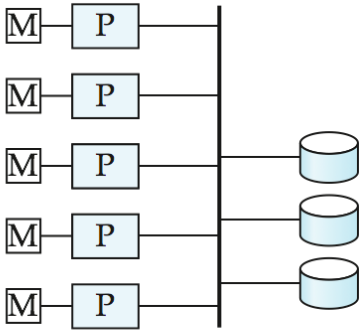

Databases – Parallel Databases (3)

- Architectures:
  - Shared memory architecture
  - Shared disk architecture
  - Shared nothing architecture
  - Hierarchical

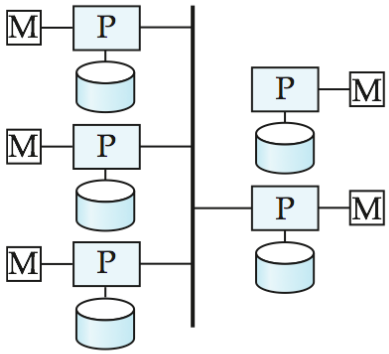
Databases – Parallel Databases (4)



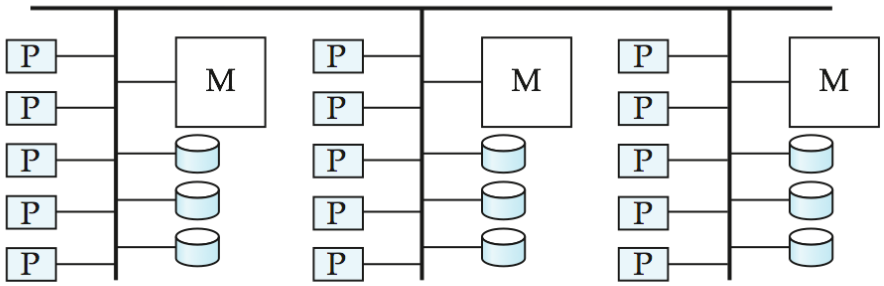
(a) shared memory



(b) shared disk



(c) shared nothing



(d) hierarchical


Databases – Parallel Databases (5)

- I/O parallelism
  - reduce the time required to retrieve relations from disk by partitioning (e.g. round-robin, hash, range)
- Interquery parallelism
  - queries/transactions execute in parallel with one another


Databases – Parallel Databases (6)

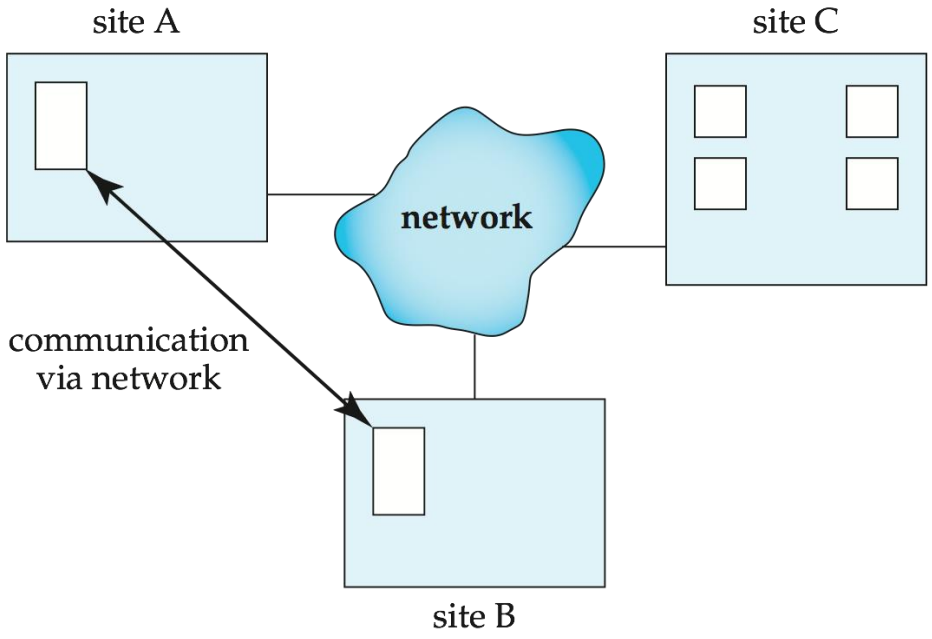
- Intraquery parallelism
  - Execution of a single query in parallel on multiple processors/disks; important for speeding up long-running queries
- Intraoperation parallelism
  - parallelize a single operator in a query plan
- Interoperation parallelism
  - Parallelize execution of operators in a query plan


Databases – Distributed Databases (1)

- Unlike parallel systems, in which the processors are tightly coupled and constitute a *single database system*, a distributed database system consists of loosely coupled sites that share no physical components (shared-nothing architecture.)
- Database systems that run on each site are independent of each other
- Transactions may access data at one or more sites

Databases – Distributed Databases (2)

- Data spread over multiple machines (nodes)
- Network interconnects the machines




Databases – Distributed Databases (3)

- Homogeneous distributed databases
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- Heterogeneous distributed databases
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality


Databases – Distributed Databases (4)

- Advantages:
  - Sharing data – users at one site able to access the data residing at some other sites.
  - Autonomy – each site is able to retain a degree of control over data stored locally.
  - Availability – data can be replicated at remote sites, and system can function even if a site fails.
- Disadvantage:
  - Added complexity required to ensure proper coordination among sites.


Databases – Distributed Databases (5)

- Other issues
  - Distributed Transactions
  - Commit Protocols
  - Concurrency Control in Distributed Databases
  - Availability
  - Distributed Query Processing

Databases – Distributed Databases (6)

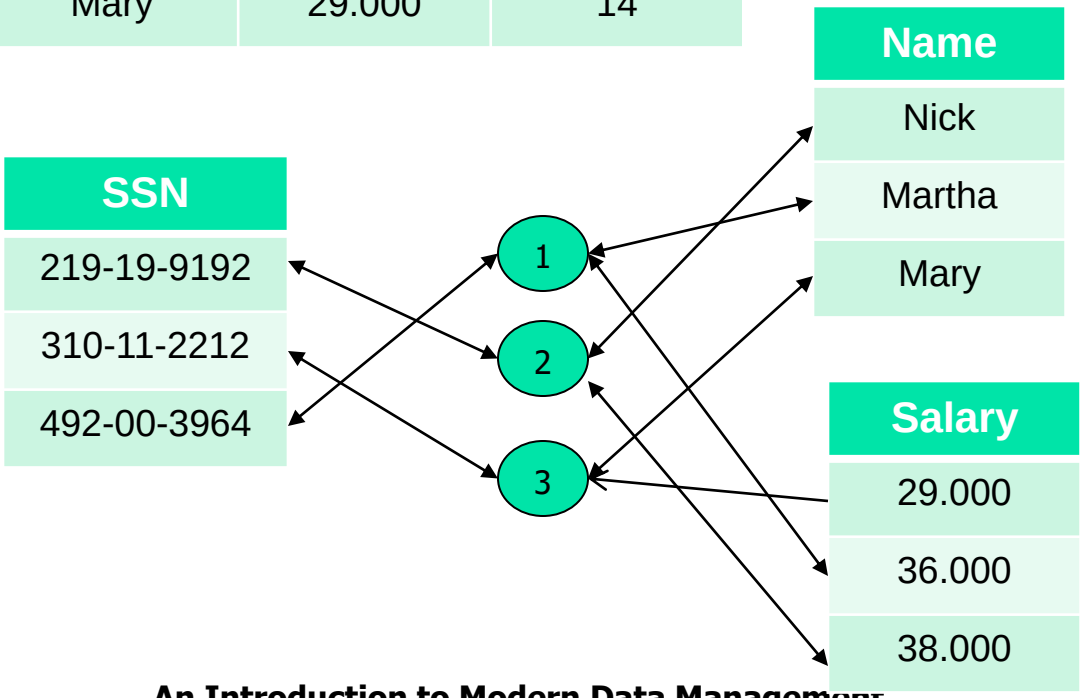
- Distributed query processing:
  - For centralized systems, the primary criterion for measuring the cost of a particular strategy is the number of disk accesses.
  - In a distributed system, other issues must be taken into account:
    - The cost of a data transmission over the network.
    - The potential gain in performance from having several sites process parts of the query in parallel.


Databases – Main Memory DBs (1)

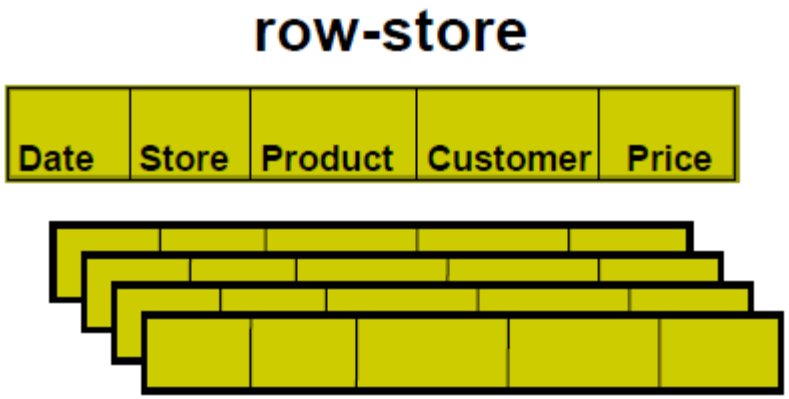
- "Main Memory Database Systems: An Overview", H. Garcia-Molina & K. Salem, IEEE TKDE, Dec. 1992
- Issues:
  - Concurrency Control
  - Commit Protocols
  - Access Methods
  - Data Representation
  - Query Processing
  - Durability / Recovery

Databases – Main Memory DBs (2)

SSN	Name	Salary	DeptCode
492-00-3964	Martha	36.000	11
219-19-9192	Nick	38.000	12
310-11-2212	Mary	29.000	14



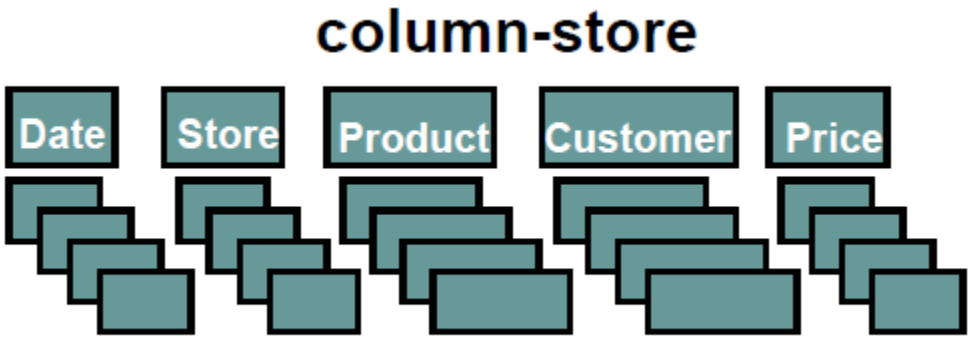
Column-oriented Databases (1)



Source: Column-Oriented Databases Tutorial, Harizopoulos et al., VLDB 2009

- Good for: add/modify records
- Bad for: read unnecessary data during queries

Column-oriented Databases (2)



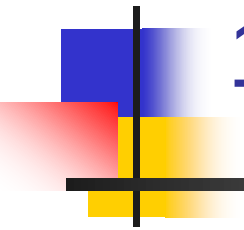
Source: Column-Oriented Databases Tutorial, Harizopoulos et al., VLDB 2009

- Good for: reads in only relevant data (queries)
- Bad for: tuple updates – too many writes

➔ *suitable for read-mostly, read-intensive, large data repositories*


Column-oriented Databases (3)

- Pioneered by Sybase IQ (late 90s)
- All values of a column stored contiguously
- Optimized for “read-mostly” workloads
- Benefits:
  - Better data compression than row-oriented
  - Only relative columns participate in query processing
  - E.g. aggregation on a particular column
- SAP Hana claims that can do both OLTP and OLAP well with a column-oriented storage model



1990 - 2000: From Data to Knowledge

Data Warehousing,

OLAP, Ad Hoc Data Analysis,

Data Mining


Data Warehouses – Definitions

- A single version of the truth [Bill Inmon].
- A single, complete and consistent store of data obtained from different sources, made available to end users in a what they can understand and use in a business context [Barry Devlin].
- A data warehouse is a *subject-oriented*, *integrated*, *time-variant*, and *nonvolatile* collection of data in support of management's decision-making process [Bill Inmon].


Data Warehouses – Examples

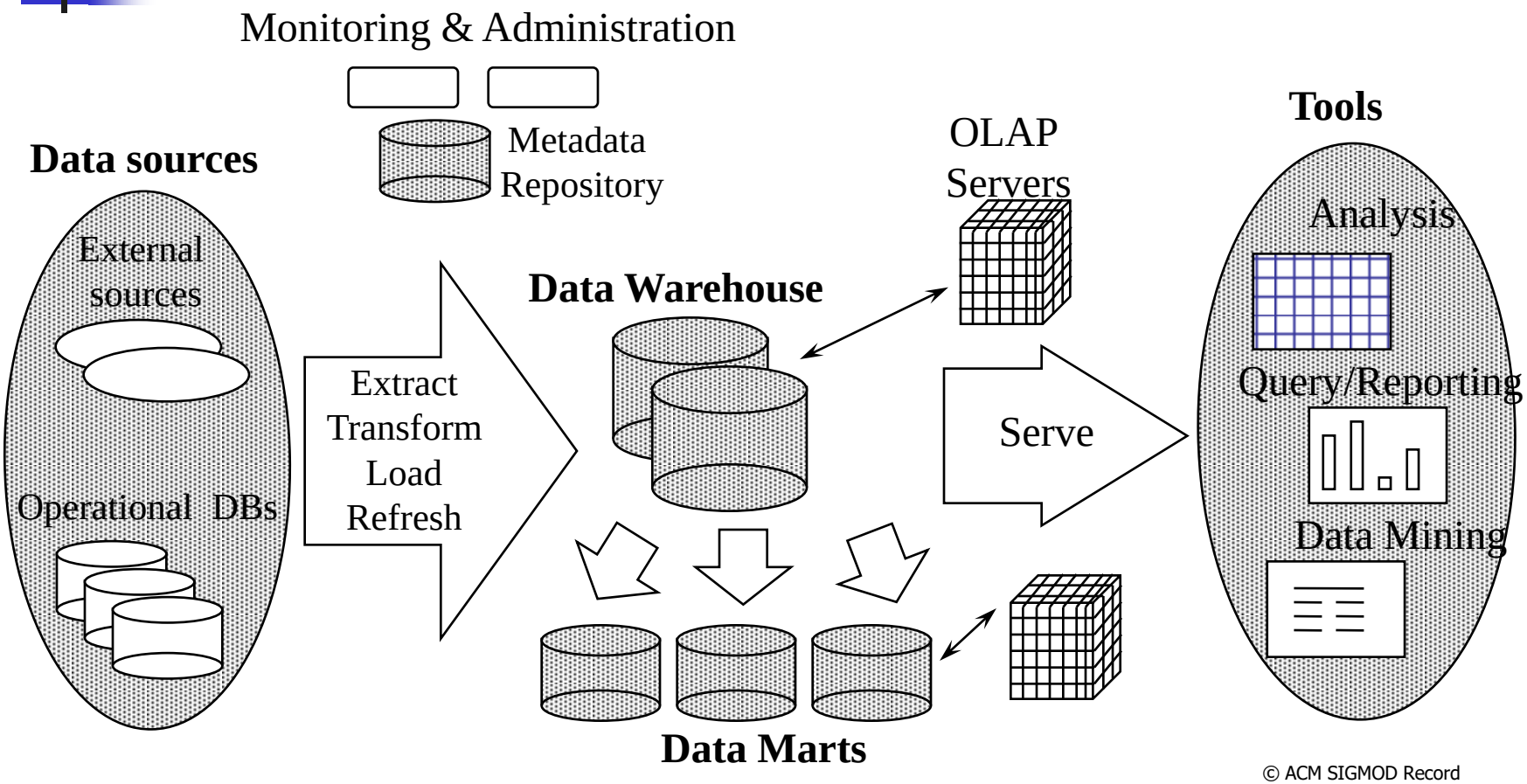
- What impact will new products/services have on revenue and margins?
- Which is the most effective distribution channel?
- How my this month's sales compare to last month's sales on a per category basis?
- What product promotions have the biggest impact on revenue?


Data Warehouses – Challenges

- Data Quality, Integration
  - data must be reliable, clean, complete, integrated.
- Functionality
  - ability to express complex analytics in simple model and language.
- Performance
  - results to queries must be computed *fast.*

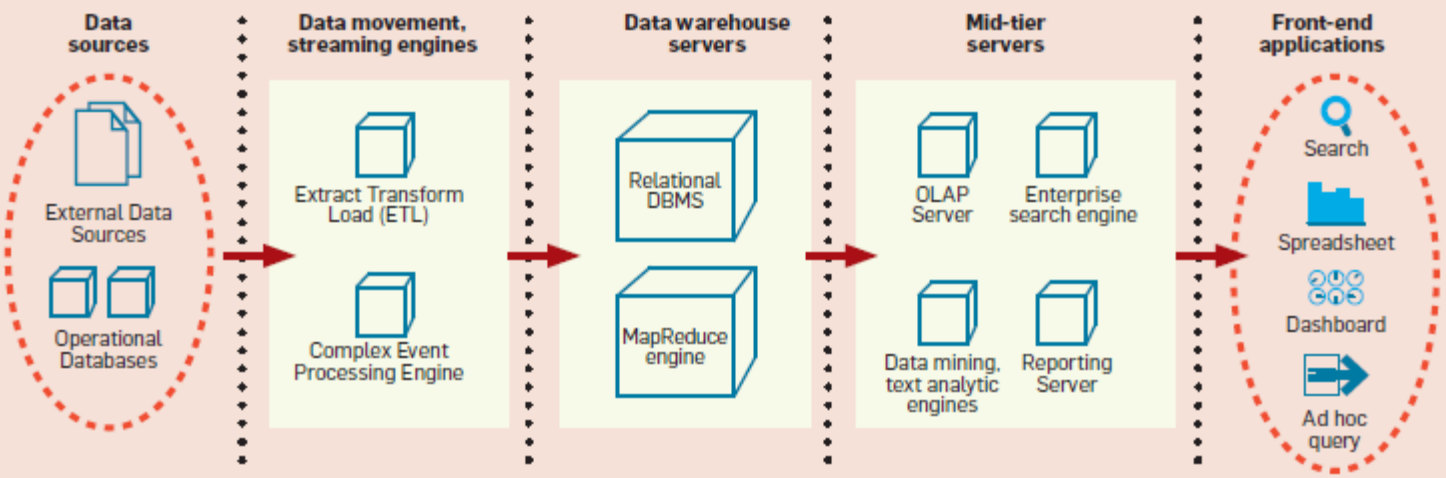
➔ Separate operational DB and data warehouse.

Data Warehouses – Big Picture (2000s)



Data Warehouses – Big Picture (2010s)

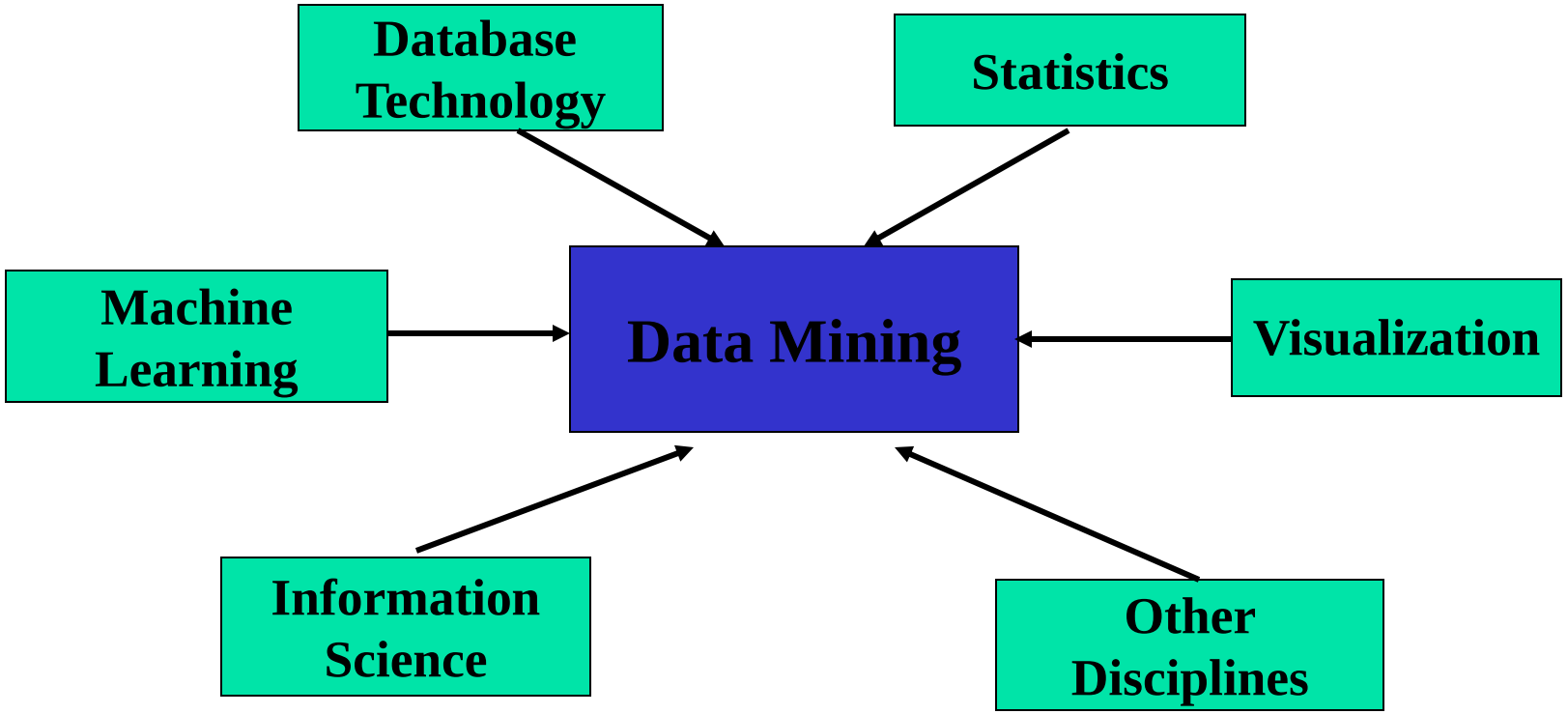
Figure 1. Typical business intelligence architecture.




Data Mining – What is It?

- Data mining (a.k.a. knowledge discovery in databases - KDD): Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from data in large databases
- Data mining # data warehousing, OLAP, SQL, data analysis.

Data Mining – Interdisciplinary

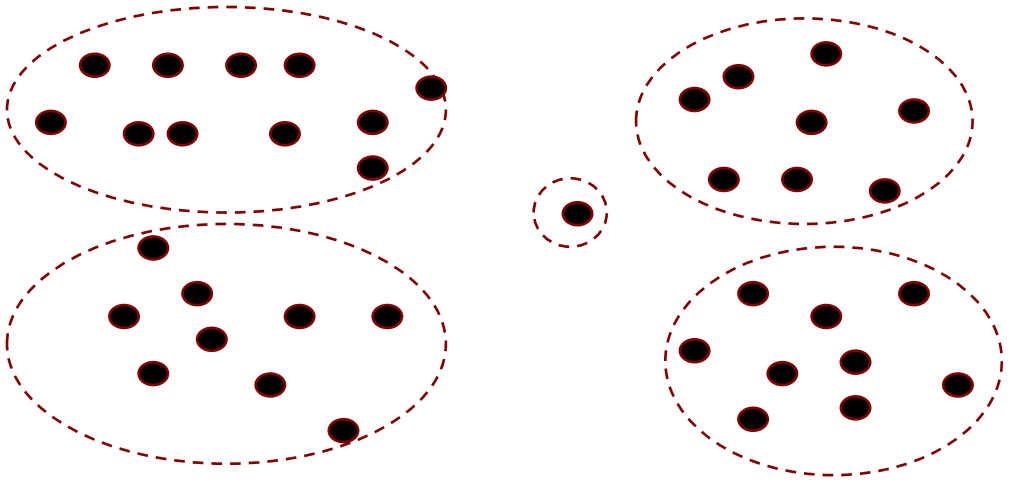



Data Mining – Functions, Classification

- Classification
  - given a set of known classes,
  - given a training set (data already placed in classes)
  - given a classification method (e.g. neural networks)
  - train system to classify new data.
- Examples
  - Banking – Identify individuals with credit risks
  - Telecom – Customer retention
  - Finance – Predicting stock behavior

Data Mining – Functions, Clustering (1)

- Assign members of a data set to clusters, using a distance (or similarity) function. Clusters are not known a-priori.



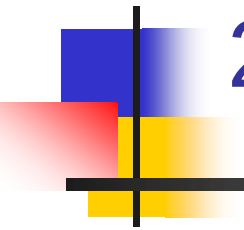
Geographic Distance Based


Data Mining – Functions, Clustering (2)

- Examples
  - Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs.
  - Land use: Identification of areas of similar land use in an earth observation database.
  - Insurance: Identifying groups of motor insurance policy holders with a high average claim cost.


Data Mining – Functions, Association Rules

- Association Rules
  - given a set of items and a set of transactions, identify all association rules, X ➔ Y with a minimum support and confidence, where X and Y are sets of items.
- Examples
  - Wal-Mart: {Beer, Bread} ➔ Diapers, s=5%, c=25%
  - Amazon: books bought together
  - Netflix: suggestions for movie rentals



2000 - 2006: Web & New Applications

Semi-structured Data, XML,

Data Stream Systems


XML – XML specification

- XML 1.0 is the base specification upon which the XML family is built. It describes the syntax that XML documents have to follow, the rules that XML parsers have to follow, and anything else you need to know to read or write XML documents (well-formed XML documents).

XML – XML Documents (Example)

```
<book>
   <title> XML Today </title>
   <prod id="33-657" media="paper"></prod>
   <chapter> Introduction to XML
      <section>What is HTML </section>
      <section>What is XML </section>
      <section>Examples </section>
   </chapter>
   <chapter>XML Syntax
      <section>Basic Rules </section>
      <section>Examples </section>
   </chapter>
   <chapter>XML Queries
      <section>XPath </section>
      <section>Examples </section>
   </chapter>
</book>
```


XML – Basic Rules

- Basic Rules
  - All XML elements must have a closing tag
  - XML tags are case sensitive
  - All XML elements must be properly nested
  - All XML documents must have a root tag
  - Attribute values must always be quoted
  - With XML, white space is preserved
  - With XML, a new line is always stored as LF
  - Comments in XML: <!-- This is a comment -->


XML – DTD and XML Schemas

- XML authors can make up their own structures and element names for their documents. DTDs and Schemas provide ways to define document types. Documents can be checked to make sure that adhere to these templates and other developers can can produce compatible documents.
- Valid tag and attribute names, structures, etc.


XML - XPath

- XPath describes a querying language for addressing parts of an XML document. This allows applications to ask for a specific piece of an XML document, instead of having to always deal with one large chunk of information. For example, XPath could be used to get “all the last names” from a document.
- Example: /Book/Chapter[1]/Section[2]


XML – Other Issues

- XSL and XSLTs
  - A language to transform an XML doc to another.
- XQuery
  - A language to query XML data.
- XML databases
  - A database to store XML data (natively or not)
- Web Services
  - Communication between web servers, request and response encoded in XML.

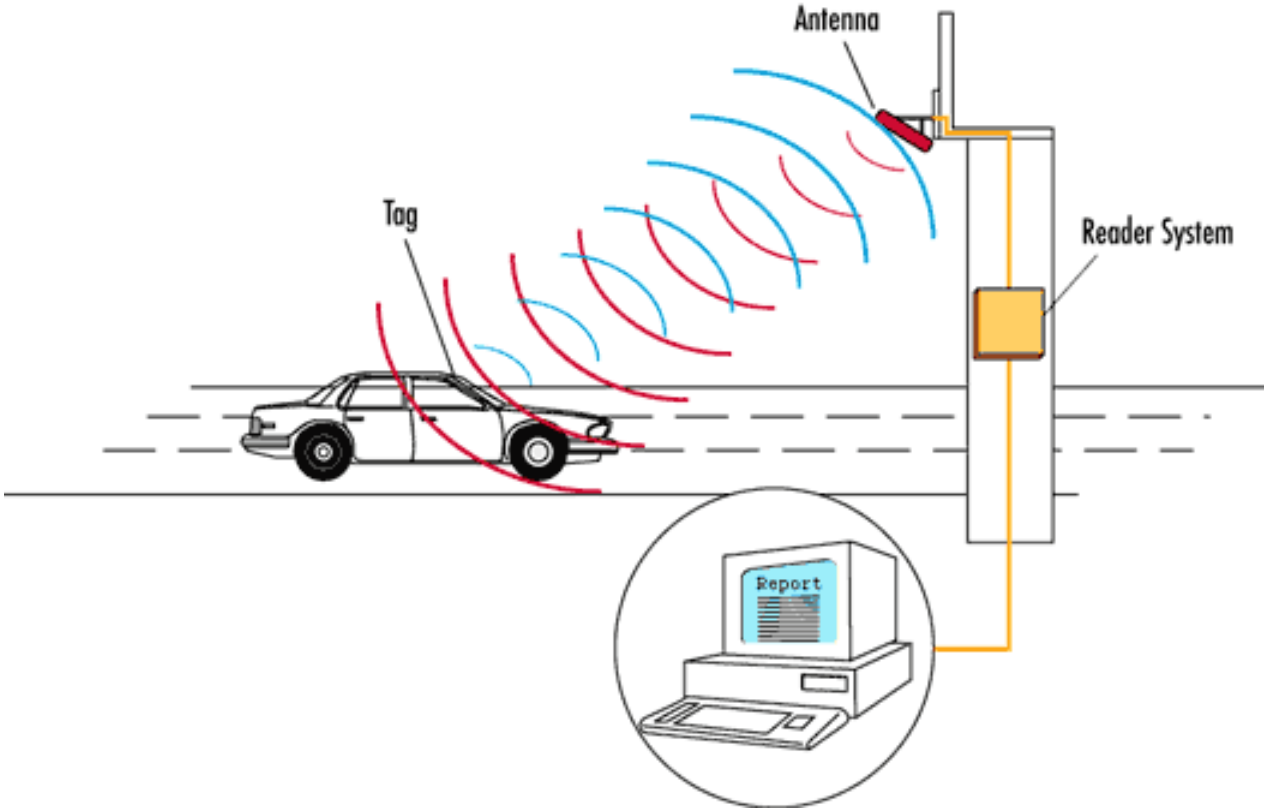

Data Streams - Buzzwords

- Real-Time Enterprise (RTE)
- We are moving toward a sensor world – wireless sensors everywhere
- We are moving toward a stream world – data streams everywhere
- Tactical (act now) vs. strategic (long term) business intelligence
- Real-time Analytics

Data Streams – Sensors, Tags (RFIDs)

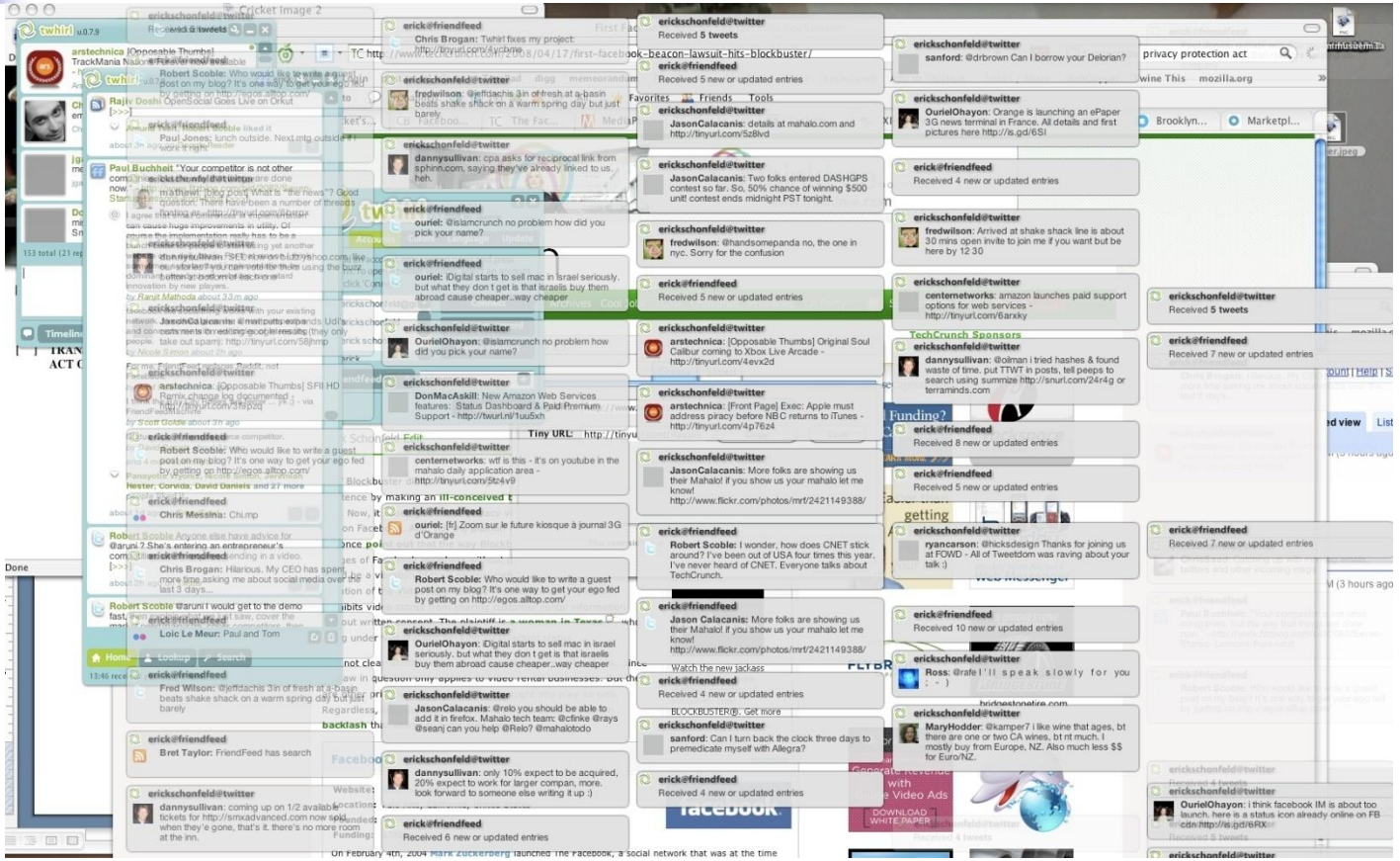
© Wired Magazine

Data Streams – Sensors, Tags (E-stickers)

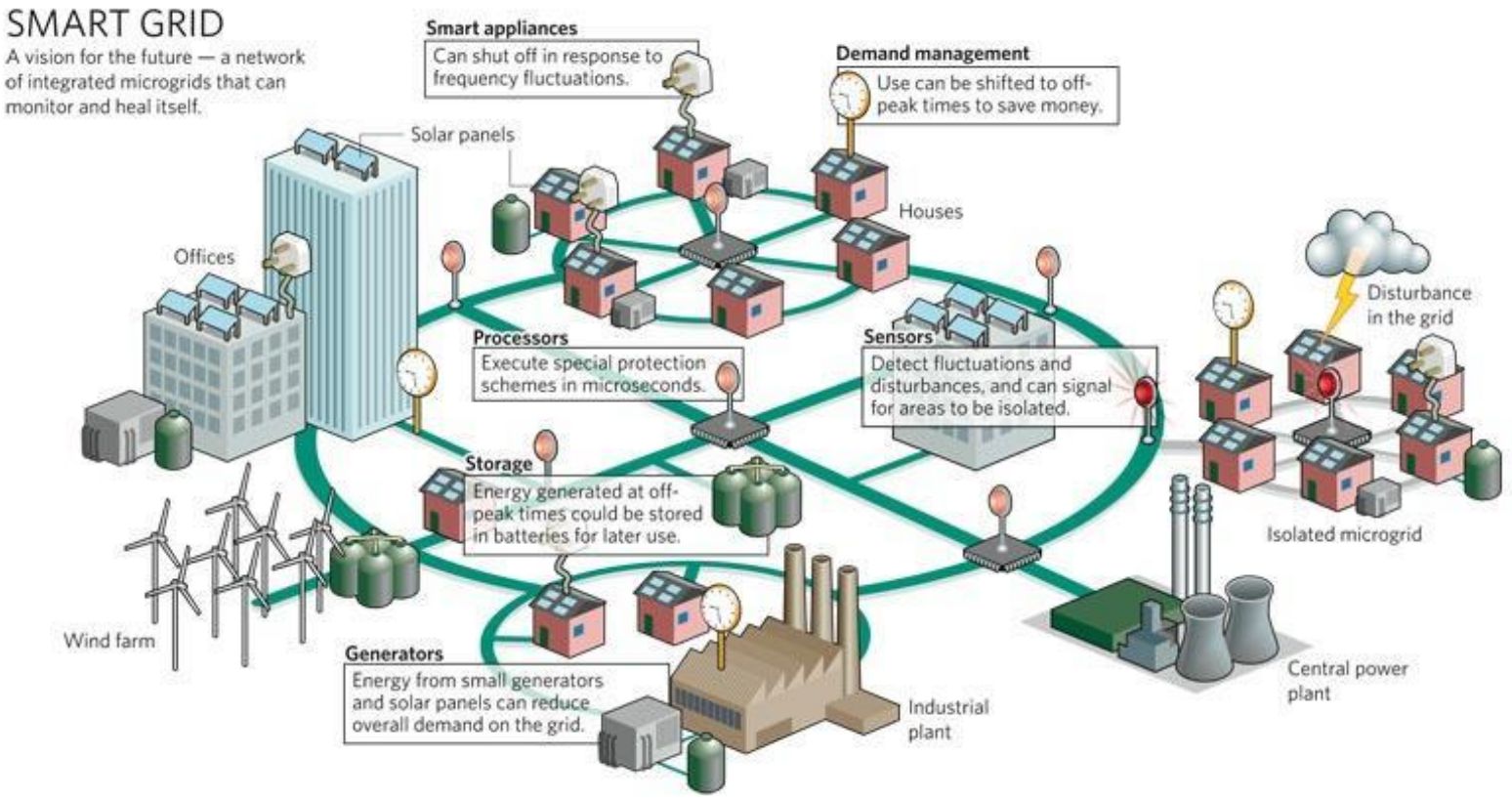


Data Streams – Sensors, Play

Data Streams – Twitter Feeds



Data Streams – Energy Management



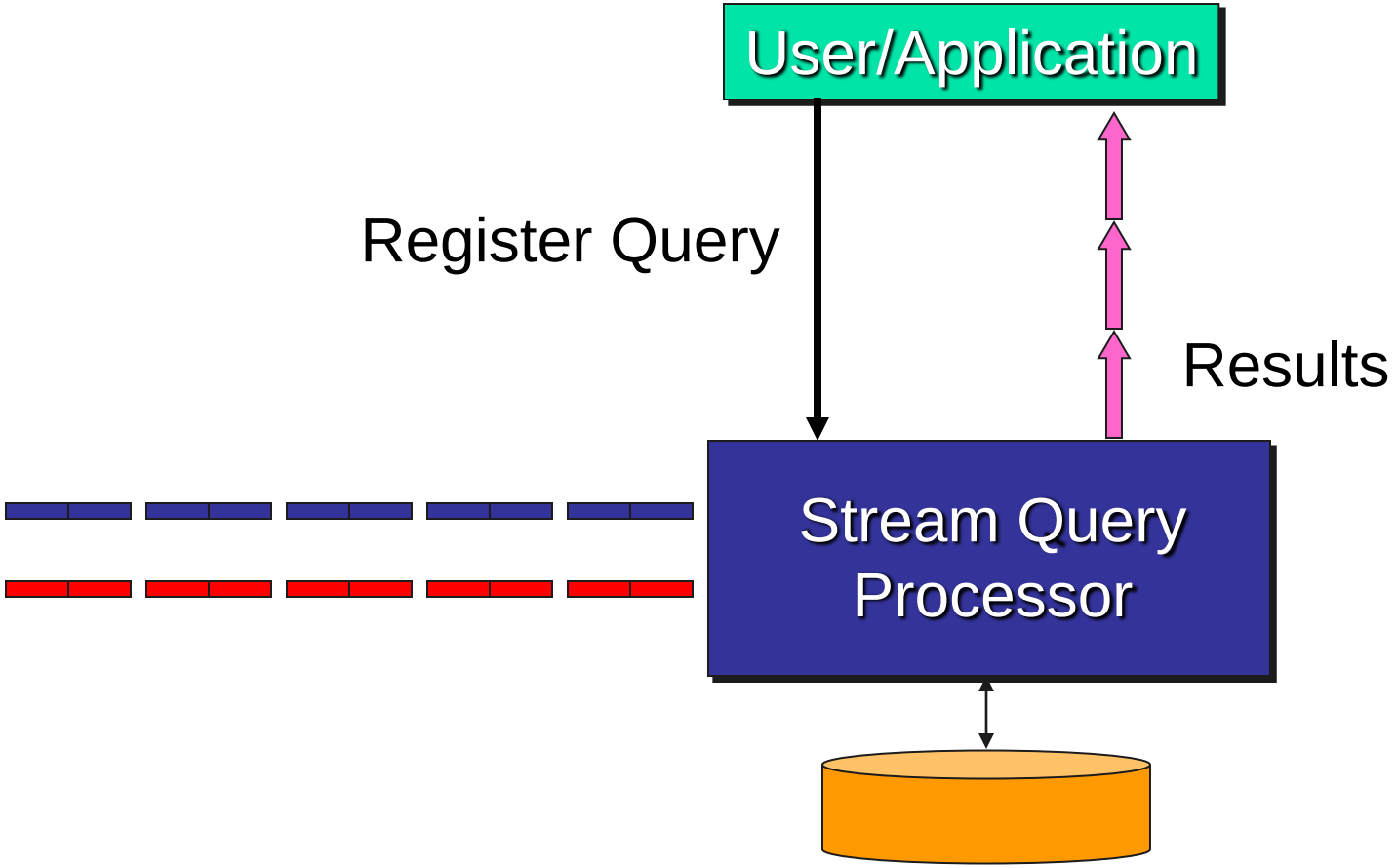

Data Streams – Other Applications

- Network monitoring and traffic engineering
- Telecom call records
- Network security
- Financial applications
- Manufacturing processes
- Web logs and clickstreams
- Nano-robots reporting state continuously


Data Streams – Continuous Queries (1)

- *One-time queries:* a class of queries that includes traditional DBMS queries; evaluated once over a point-in-time snapshot of the data set, with the answer returned to the user.
- *Continuous queries:* evaluated continuously as data streams continue to arrive; the answer is produced over time, always reflecting the stream data seen so far; may be stored and updated as new data arrives, or may produce another stream.

Data Streams – Continuous Queries (2)

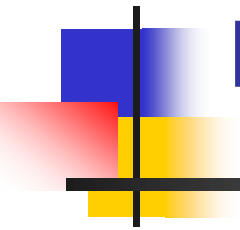



Data Streams – Continuous Queries (3)

- Example: financial data streams come in many different forms, e.g. stock tickers, news feeds, trades, etc.
  - Find all stocks between $20 and $200 where the spread between the high tick and the low tick over the past 30 minutes is greater than 3% of the last price and in the last 5 minutes the average volume has surged by more than 300%.


Data Streams – Continuous Queries (4)

- Queries: correlate, aggregate, localize, alert
- Continuous Query Language (CQL)
  - SQL-based
  - uses windows of streams as tables
- Complex Event Processing (CEP)
  - graph-based
  - each node is an operator
  - network of nodes



Data Management: Big Data Systems


Drivers of Modern Data Systems Features (1)

- Hardware Trends:
  - Multicore, InfiniBand, DRAMs, SSDs, FPGAs, GPUs, SCMs
- Big Data:
  - Compression, Column Handling, Scalability (Sharding), Avoiding Data Loading, Linkage to Analytics Tools
- Wider Adoption of Database Technology:
  - Ease of Use, Appliances
- Monetary Constraints:
  - Open Source, Commodity Processors/Disks, Local Disks, Economic Factors in General


Drivers of Modern Data Systems Features (2)

- Skills Shortage:
  - DBAs, Admins in General, SQL Knowledge
- Social Media:
  - Relationship Graphs, Ultra Low Response Times, Data Heterogeneity/Volume, Mobile Users
- Internet of Things (IOT):
  - Smarter Planet, Sensors, Low Power
- Cloud Platforms:
  - Elasticity, Multi-tenancy, Security, Transborder Data Flows


MapReduce & Hadoop - History

- 2002: Doug Cutting and Mike Cafarella, Nutch
- 2003: Google's paper on GFS, Cutting starts working on Nutch Distributed FS
- 2004: Google's paper on MapReduce
- 2005: MapReduce is included into Nutch
- 2006: Hadoop kicks off as a separate project. Yahoo develops Pig.


MapReduce – Motivation (1)

- Scalability to large data volumes:
  - Scan 100 TB on 1 node @ 50 MB/s = 24 days
  - Scan on a 1000-node cluster = 35 minutes
- Cost-efficiency:
  - Commodity nodes (cheap, but unreliable)
  - Commodity network
  - Automatic fault-tolerance (fewer admins)
  - Easy to use (fewer programmers)


MapReduce – Motivation (2)

- Cheap nodes fail, especially if you have many
  - Mean time between failures for 1 node = 3 years
  - MTBF for 1000 nodes = 1 day
  - Solution: Build fault-tolerance into system
- Commodity network = low bandwidth
  - Solution: Push computation to the data
- Programming distributed systems is hard
  - Solution: Users write data-parallel “map” and “reduce” functions, system handles work distribution and faults


MapReduce – Introduction

- MapReduce is a parallel programming paradigm for efficient, distributed, fault-tolerant computation of per-key aggregates.
- Pioneered by Google
  - Processes 20 PB of data per day
- Popularized by open-source Hadoop project
  - Used by Yahoo!, Facebook, Amazon, and others…
- Main Reference:
  - "MapReduce: Simplified Data Processing on Large Clusters", Dean & Chemawat, OSDI '04


MapReduce – Idea

- Simple Idea:
  - A set of keys K = {k_1, k_2, …, k_n}
  - A set of values V
  - You assign each v in V to a value k in K **[mapping]**
  - Now you have a list of values L_k for each k in K
  - Now, iterate over the elements of L_k and compute something (e.g. the sum of L_k's elements) **[reduce]**
- Key idea: Values to be read in are distributed to a number of nodes and the whole computation takes place in a distributed fashion


MapReduce – A First Example (1)

- Classic example: for a large set of web pages, count the occurrences (how many) of distinct words.
- Map phase:

```
for each page {
   for each word w in page {
      create a (w, 1) pair;
   }
}
```

- Reduce phase: for each word w, sum 1s

MapReduce – A First Example (2)

- Map phase: (“google”, 1), (“greece”,1), (“january”,1), …

keys		**list of mapped values**
google	→	**{1, 1, …, 1}**
greece	→	**{1, 1, 1, 1, …, 1}**
january	→	**{1, 1, 1, …, 1}**

- Reduce phase: for each key, iterate over all mapped values and perform some computation.


MapReduce – Map and Reduce Functions

```
map(String input_key, String value):
   // input_key: document name
   // value: document contents
   for each word w in value:
   EmitIntermediate(w, "1");
```

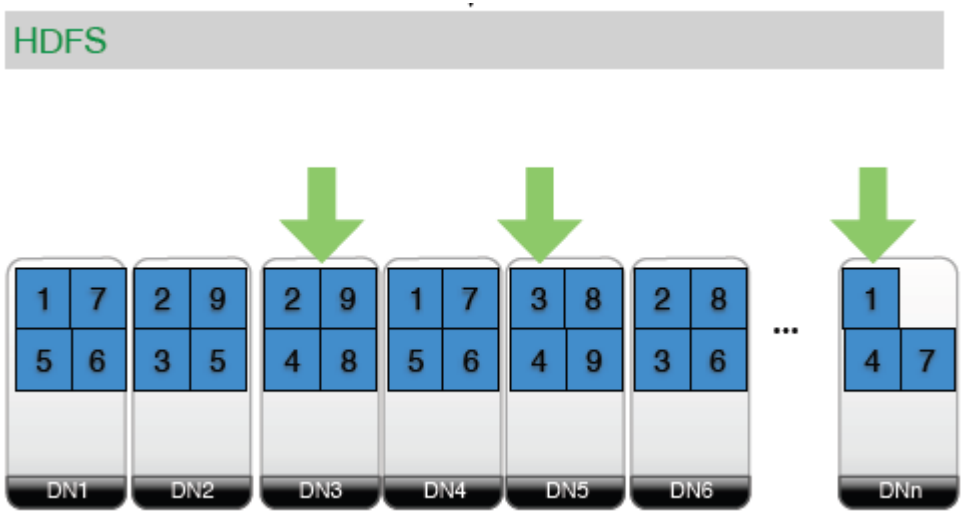
Users only write the **map** and **reduce** functions of a MapReduce job – they don't care for other details (distribution/crashes)

```
reduce(String output_key, Iterator values):
   // output_key: a word
   // values: a list of counts
   int result = 0;
   for each v in values:
   result += ParseInt(v);
   Emit(AsString(result));
```


Hadoop – Introduction

- Hadoop is an open source project that enables distributed parallel processing on huge amounts of data across inexpensive, commodity servers. Equipped with MapReduce (java-based) capabilities.
- Yahoo! has contributed a great deal to it.
- Key concept: Hadoop Distributed File System (HDFS) – is a virtual file system; when a file is moved on HDFS, this file is split into many small files and each of those files is replicated and stored on 3 servers for fault tolerance.

Hadoop – HDFS



Replication

Failover

Load Balancing

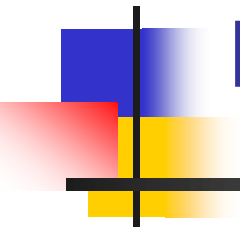
Source: Efficient Big Data Processing in Hadoop MapReduce, Tutorial, VLDB 2012, by Dittrich et al.


Retrieving Data – From MR to SQL

- MapReduce framework is Java
  - Difficult to write, optimize, understand
- Need for declarative query languages
  - PigLatin (declarative, not SQL, translates to MapReduce)
  - Hive (SQL like, translates to MapReduce)
  - Impala (SQL, just uses HDFS)
  - Several SQL versions of well-known vendors (IBM, EMC, etc.)


Summary of MapReduce and Hadoop

- Developed with **data analysis** tasks in mind
  - over *very large* datasets. This means
    - data distributed in a network of nodes. This requires:
      - parallel processing
      - fault-tolerance
  - over *different* datasets: structured, unstructured (e.g. text)
  - requiring *procedural flexibility*, i.e. processing varies significantly from task to task



NoSQL Systems


NoSQL Systems – What are they?

- A new generation of data management systems. Do not support the relational data model and generally do not provide SQL and ACID properties
- Accommodating Web 2.0 applications, which:
  - deal with humongous data sets
  - perform thousands of updates per second (e.g. Facebook)
  - have specific data retrieval/storage requirements
  - require fault-tolerance
  - must be highly available
  - must scale (horizontally) fast
- Goal: transaction processing, not data analysis


NoSQL Systems – Key Features

- NoSQL systems generally have six features
  - horizontally scale “simple operations” throughput over many servers
  - replicate and distribute (partition) data over many servers
  - a simple call level interface or protocol (in contrast to SQL)
  - offer a weaker concurrency model than the ACID transactions of RDBMS
  - efficient use of distributed indexes and RAM for data storage
  - schema-flexible: dynamically add new attributes to data records


NoSQL Systems - History

- NoSQL systems didn't even exist 8 years ago (2007).
- Early systems that "paved the way":
  - Memcached: in-memory indexes can be highly scalable, distributing objects over multiple nodes
  - Dynamo (Amazon): pioneered the idea of eventual consistency:
    - reading data may not be up-to-date
    - updates eventually propagates to all nodes
  - BigTable (Google): persistent record storage can be scaled to thousands of nodes


NoSQL Systems – Categorization

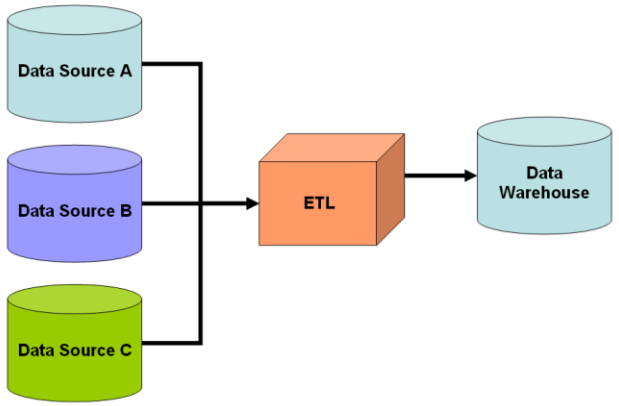
- Key-value Stores (e.g. Redis)
- Document Stores (e.g. Mongo)
- Extensible Record (Columnar) Stores (e.g. Cassandra)
- Graph Databases (e.g. Neo4j)


Data Integration (1)

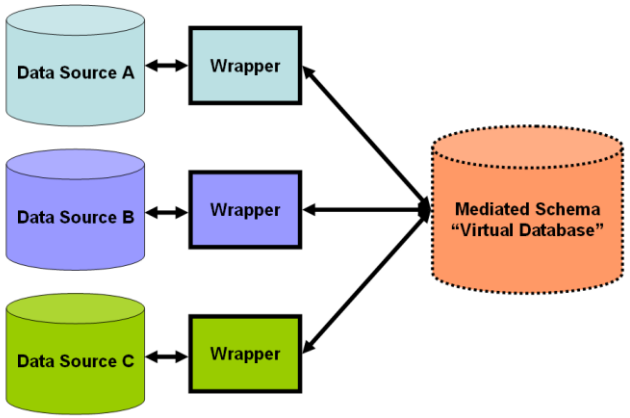
- Data:
  - Persistent (databases)
  - Streams/transient/at motion
- Data Management Systems:
  - Relational DBs
  - Hadoop/NoSQL (Semi-structured, Graph, Key-Value, etc.)
  - Stream Engines
- ***Data integration*** involves combining data residing in different sources and providing users with a unified view of them

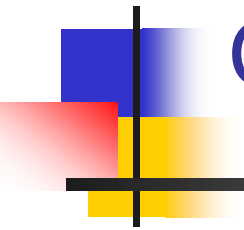
Data Integration (2)

- Data Warehousing (source: Wikipedia)



- Mediators, Wrappers (source: Wikipedia)





Conclusions


Conclusions (1)

- Use data to understand people, behaviors, processes
  - data is exploding
  - data comes to different formats
  - data comes fast
  - 3Vs: Volume + Variety + Velocity
- A wide variety of skills required
  - data management
  - business
  - statistics, machine learning, optimization


Conclusions (2)

- We are living at very interesting times!
- The information world is changing *very fast*.
- Data is widely available at finest granularity.
- Sensors will change everything ➔ Internet of things
- Ability to analyze and mine very large datasets.
- Market Economy in the 60s ➔ Data Economy in the 20s


Conclusions (3)

"Sensors everywhere. Infinite storage. Clouds of processors. Our ability to capture, warehouse, and understand massive amounts of data is changing science, medicine, business, and technology. As our collection of facts and figures grows, so will the opportunity to find answers to fundamental questions. Because in the era of big data, more isn't just more. More is different."

(http://www.wired.com/science/discoveries/magazine/16-07/pb_theory)


Conclusions (4)

- To analyze data, you have to manage it first! From creation, to storing, to manipulation, to integration.
- So… let's learn data management concepts!