Big Data Systems

Djellel Difallah

Spring 2023

Lecture 1 - Class Introduction and Review of Relational DBMSs

Outline

- Introduction
- Syllabus Quick Tour
- Review of relational DBMSs
  - (Relation Algebra, SQL, Data Warehousing)

General Course Description

- Study new principles of data management
  - Data properties, data independence, scale-out, etc.
- Study key DBMS design issues
  - Transactions, parallel data processing, etc.
  - Consistency, Availability, network Partition
- Hands on some Big Data systems
  - Get exposed to established solutions to scale-out
- More broadly, ensure that
  - You are comfortable using a DBMS
  - You have an idea about large (distributed) data management
  - You know a bit about current research topics in data management
  - You know a bit about the current market players

Exascale Data Deluge

- Web companies
  - Google
  - Ebay
  - Yahoo
- Science
  - Biology
  - Astronomy
  - Remote Sensing
- Financial services, retail companies, governments, etc.

New data formats

New machines

Peta & exa-scale datasets

Obsolescence of traditional information infrastructures

Big Data Buzz

Between now and 2015, the firm expects big data to create some **4.4 million IT jobs** globally; of those, 1.9 million will be in the U.S. Applying an economic multiplier to that estimate, Gartner expects each new big-data-related IT job to create work for three more people outside the tech industry, for a total of almost 6 million more U.S. jobs.

Growth in the Asia Pacific Big Data market is expected to accelerate rapidly in two to three years time, from a mere US$258.5 million last year to in excess of **$1.76 billion in 2016**, with highest growth in the storage segment.

Office of Science and Technolog
Executive Office of the President
New Executive Office Building
Washington, DC 20502

FOR IMMEDIATE RELEASE
March 29, 2012

Contact: Rick Weiss 202 456-6037 rweiss@ostp.
Lisa-Joy Zgorski 703 292-8311 lisajoy@

OBAMA ADMINISTRATION UNVEILS "BIG DATA" INITIATIVE: ANNOUNCES $200 MILLION IN NEW R&D INVESTMENTS

Aiming to make the most of the fast-growing volume of digital data, the Obama Administration today announced a "Big Data Research and Development Initiative. improving our ability to extract knowledge and insights from large and complex collections of digital data, the initiative promises to help solve some the Nation's m pressing challenges.

To launch the initiative, six Federal departments and agencies today announced m than $200 million in new commitments that, together, promise to greatly improve th tools and techniques needed to access, organize, and glean discoveries from huge

Big Data Everywhere!

- The Age of Big Data (NYTimes Feb. 11, 2012) http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html

  "Welcome to the Age of Big Data. The new mega rich of Silicon Valley, first at Google and now Facebook, are masters at harnessing the data of the Web — online searches, posts and messages — with Internet advertising. At the World Economic Forum last month in Davos, Switzerland, Big Data was a marquee topic. A report by the forum, "Big Data, Big Impact," declared data a new class of economic asset, like currency or gold."

Big data can generate significant financial value across sectors

US health care

- $300 billion value per year
- ~0.7 percent annual productivity growth

Europe public sector administration

- €250 billion value per year
- ~0.5 percent annual productivity growth

Global personal location data

- $100 billion+ revenue for service providers
- Up to $700 billion value to end users

US retail

- 60+% increase in net margin possible
- 0.5–1.0 percent annual productivity growth

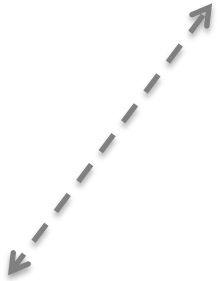
Manufacturing

- Up to 50 percent decrease in product development, assembly costs
- Up to 7 percent reduction in working capital

SOURCE: McKinsey Global Institute analysis

Big Data Central Theorem

Data + Technology ➔ Actionable Insight ➔ $$

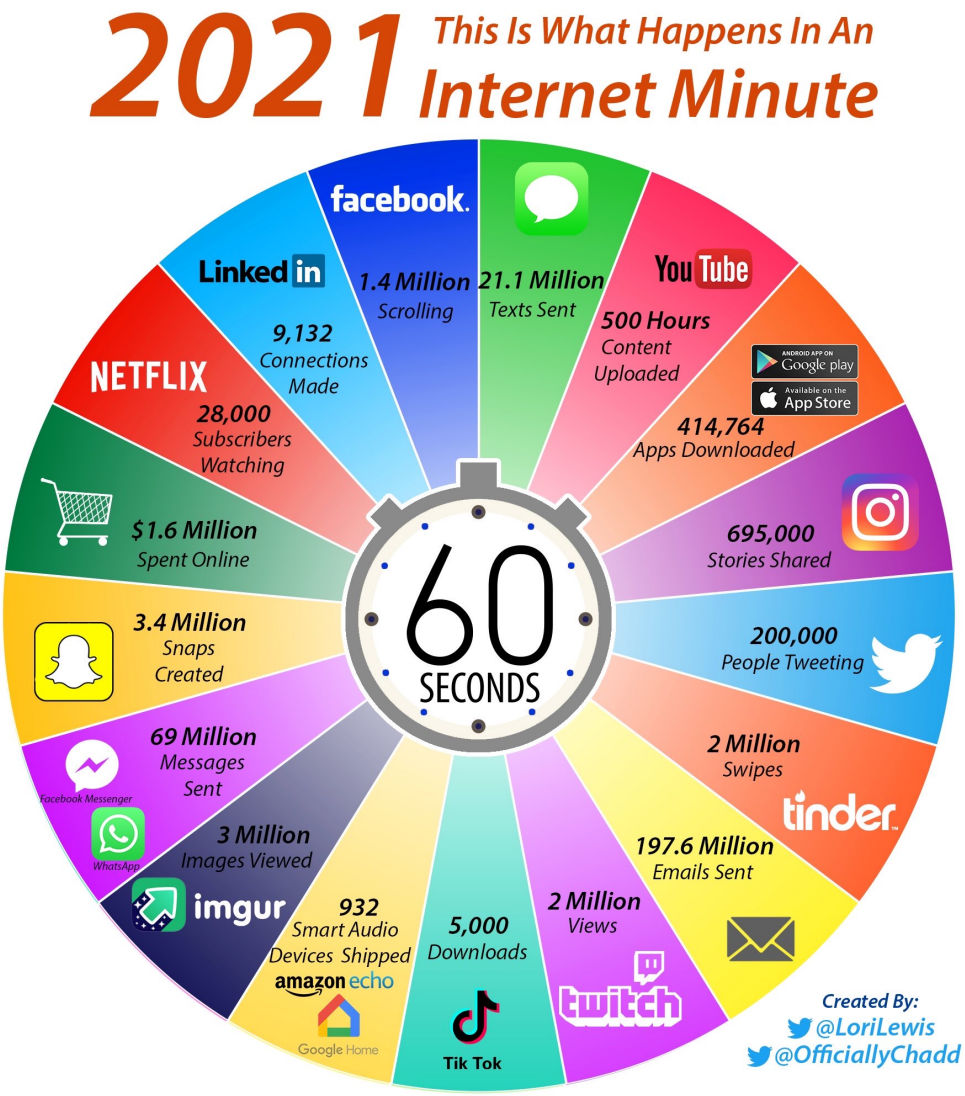


Reporting, Monitoring, Root Cause Analysis, (User) Modeling, Prediction

What can you do with the data?

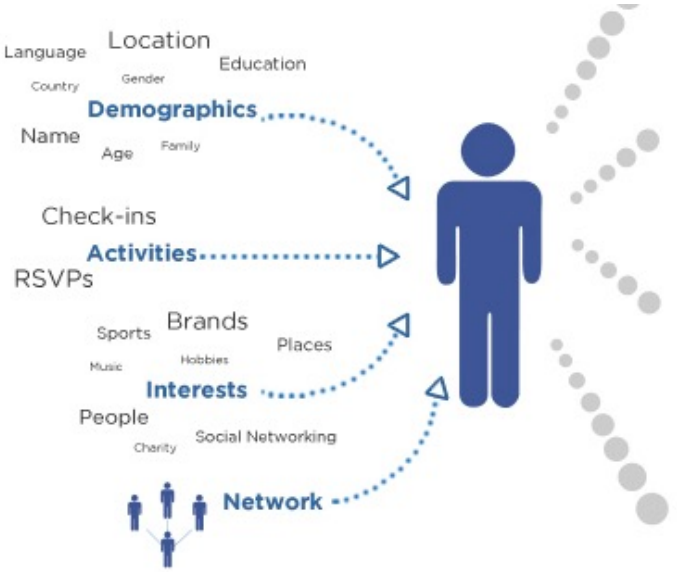
- Reporting
  - Post Hoc
  - Real time
- Monitoring (fine-grained)
- Exploration
- Finding Patterns
- Root Cause Analysis
- Closed-loop Control
- Model construction
- Prediction
- …

User Generated Bigdata



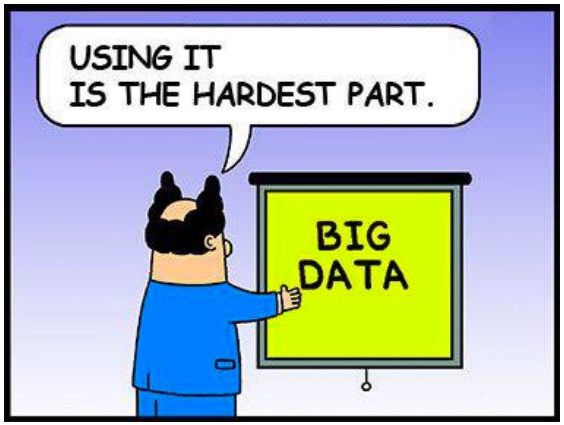
Typical Big Data Success Story

- Modeling users through Big Data
  - Online ads sale / placement [e.g., Facebook]
  - Personalized Coupons [e.g., Target]
  - Product Placement [Walmart]
  - Content Generation [e.g., Netflix]
  - Personalized learning [e.g., Duolingo]
  - HR Recruiting [e.g., Gild]



Big Data

- 3 V's of Big Data
  - Volume, Velocity, Speed
- What's new?
  - Powerful and/cheaper hardware
  - Distributed/fast algorithms



- Data processing has become orders of magnitudes more efficient
- Hadoop and co.
  - Embarrassingly parallel problem

The 3-Vs of Big Data

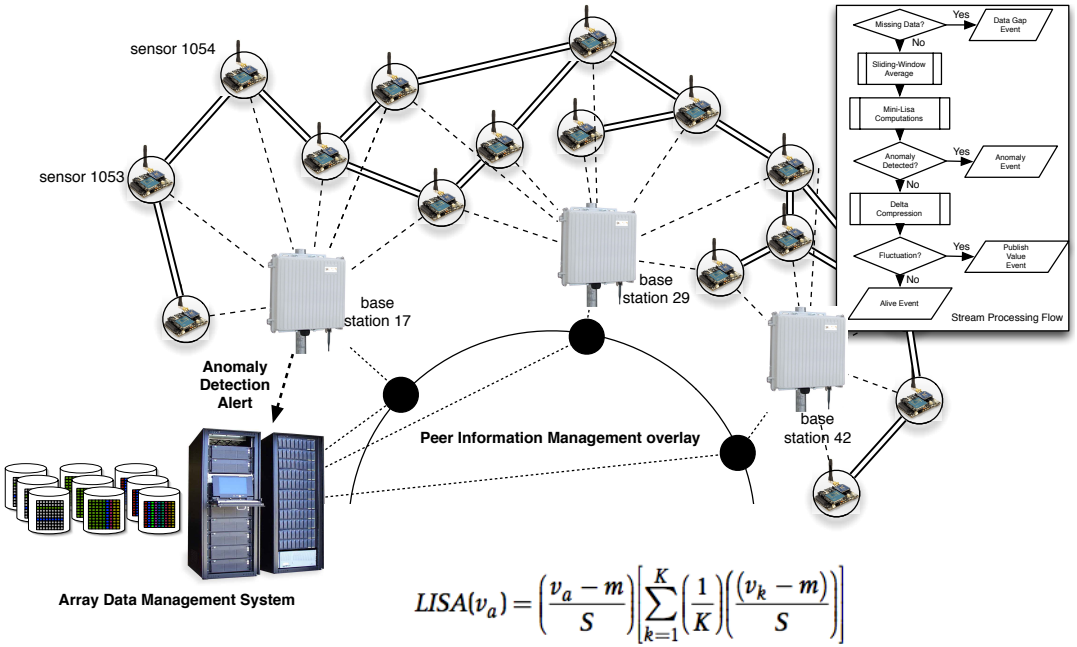
- Volume
  - Amount of data
- Velocity
  - speed of data in and out
- Variety
  - range of data types and sources
- [Gartner 2012] *"Big Data are high-volume, high-velocity, and/or high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization"*

3 Examples from *COLAB*

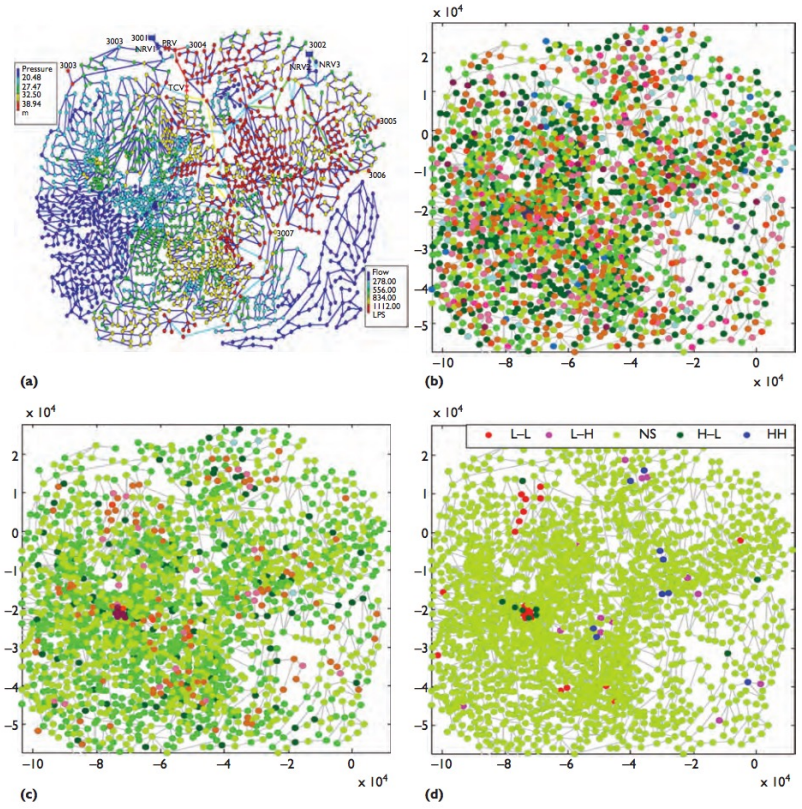
- Three examples from my lab:
  - Volume: Using Hadoop to store/explore Wikipedia and Web content.
  - Velocity: Detecting anomalies in water distribution networks and telecoms.
  - Variety: Integrating text and semi-structured data. We use natural language processing and semantic web technologies to extract entities and their relationships from text.

Sensors installed in the water pipes!

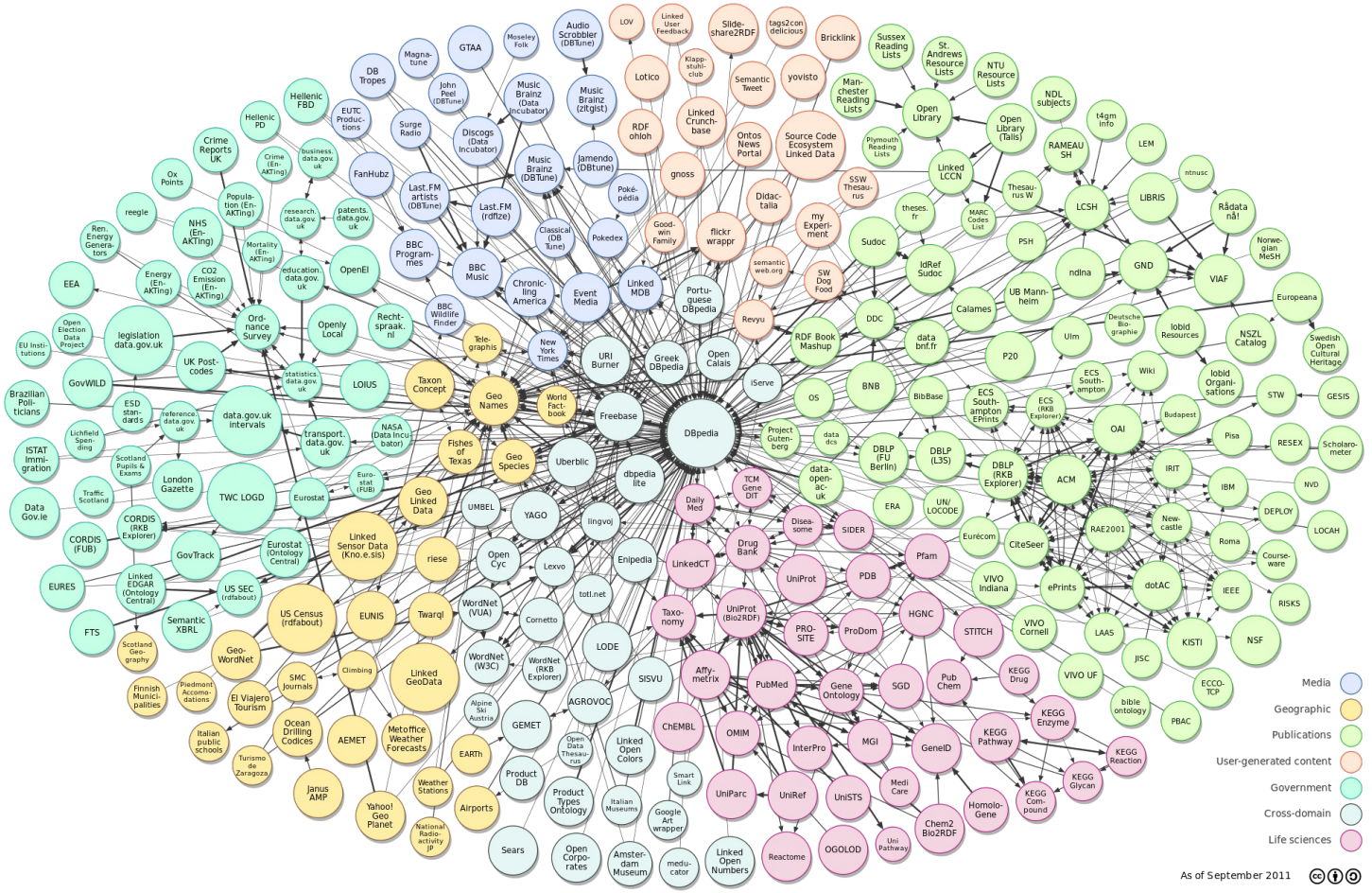
- Spatial + temporal statistical processing
- Stream processing (Storm) + Array processing (SciDB)



$$LISA(v_a) = \left(\frac{v_a - m}{S}\right)\left[\sum_{k=1}^{K}\left(\frac{1}{K}\right)\left(\frac{(v_k - m)}{S}\right)\right]$$



Variety: Integrating Textual & Semi-Structured Data



Linked Open Data Cloud

Quick tour

SYLLABUS

Who/When/Where?

Faculty Details	**Professor**	**Teaching Assistant**
Name	Djellel Difallah	Sara Mumtaz
Email	djellel@nyu.edu	sm11206@nyu.edu
Telephone	+917 2 628 7380	TBD
Workspace	A1-185	A2-186F
Office Hours	By Appointment	Wed 2.30 PM -3.30 PM

Course Details	Day/Time	Location
Lecture	Tue/Thu 3:35 PM - 4:50 PM	Lecture room
Mid Term Exam	Last session before Spring Break	Lecture room
Final Exam	Finals week	TBD

Class Format

- Weekly lecture + demo/hands-on + discussions
  - Don’t be afraid to express yourselves and ask questions
  - Some lectures will be based on papers
- Assignment
  - Bi-weekly
  - Deliverables are individual, but can help each other ***debug*** issues
- Guest lectures
  - Invited industry experts (labs and lectures)

Grading

Activity Detail	Grade Percentage	Submission Date/Week
Assignments (6)	60% (10% each)	Biweekly (or as per the schedule)
Midterm Exam	20%	Week 7
Final Exam	20%	Finals' week

A	A-	B+	B	B-	C+	C	C-	D+	D	F
[95-100]	[90-95)	[87-90)	[83-87)	[80-83)	[77-80)	[73-77)	[70-73)	[67-70)	[63-67)	[0-63)

Readings and Notes

- Some readings are based on papers
  - Mix of old seminal papers and new papers
  - Papers will be available online on website
  - Three types of readings
    - Mandatory, and optional
- Background readings from online books
  - Mining of Massive Datasets. (Leskovec, Rajaraman, Ullman) 3rd edition (http://www.mmds.org/)
  - Intro to databases (Stanford online course) http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=IntroToDatabases
  - Reading from "system" books available online.
- Lecture notes (the slides)
  - Posted on the class website before each lecture

Schedule: First half

Week	Session	Topic	Reading	Other
1	Session 1	*Introduction: Big Data Challenges + Relational Databases*	[MMD] ch.1	*Course Survey and SQL primer*
	Session 2	Continued lecture + lab (Infrastructure Setup)		
2	Session 1	*Review of Database Concepts: Storage, Indexing, Transactions, Parallel DBMSs, CAP*	[SQL] Sections 1-5 [CAP]	*Assignments 1 Published*
	Session 2	Continued lecture + lab (DB Install and SQL)		
3	Session 1	*Overview on Peer-to-Peer Systems*	[CHR] Sections 1-4, 6-7	
	Session 2	**Continued lecture + lab**		*Assignments 1 Due*
4	Session 1	*Data Analytics with Column Stores*	[CST]	*Assignments 2 Published*
	Session 2	Continued lecture + lab (MonetDB)		
5	Session 1	*NOSQL: Document Stores, Graph Databases, Array Data Management*	[DYN]	
	Session 2	Continued lecture + lab (MongoDB, Neo4J)		*Assignments 2 Due*
6	Session 1	*Key-Value Stores: Memcached, Cassandra*	[CAS]	*Assignments 3 Published*
	Session 2	Continued lecture + lab (Memcahed, Redis)		
7	Session 1	*Distributed Graph Processing: Pregel, Apache Giraph*	[DGP] Sections 8.1-3 [PRG]	
	Session 2	**Midterm Exam**		*Assignments 3 Due*

Schedule: Second half

8	Session 1	*Distributed File Systems: Google File System + HDFS*	[DTP] Ch.2.5 [MMD] Ch.2 [HDG] Ch.3	*Assignments 4 Published*
	Session 2	Continued lecture + lab (HDFS / Parquet)		
9	Session 1	*Introduction to MapReduce and its Design Patterns*	[DTP] Ch.1, Ch.2, Ch.6 [MAP]	
	Session 2	Continued lecture + lab (Word count, Terasort)		
10	Session 1	*Apache Hadoop + Config (Zookeeper) + Scheduling (YARN)*	[HDG] Ch.4	
	Session 2	Continued lecture + lab (Setting up Queues)		*Assignments 4 Due*
11	Session 1	*Apache Spark*	[SPK] Ch.1, Ch.3	*Assignments 5 Published*
	Session 2	Continued lecture + lab (Spark)		
12	Session 1	*Stream Processing (Spark Streaming) and Event-streaming (Kafka)*	[MML] Ch.4 [SPK] Ch.8 [KFK] Ch.1	
	Session 2	Continued lecture + lab (Kafka)		*Assignments 5 Due*
13	Session 1	*Big Data Integration and Cleaning*	Lecture Notes or Guest Lecture (TBD)	*Assignments 6 Published*
	Session 2	Continued lecture + lab		
14	Session 1	*Big Data and Machine Learning: Snorkel*	[SNO]	
	Session 2	Course Review		*Assignments 6 Due*

Review

RDBMS

The DBMS Ice Breaker

- **What is a database?**
- **Do you know any database system?**
- **How do you pronounce SQL?**
- **Foreign-Key?**
- **Normal Forms?**
- **ACID?**
- **Two-Phase Commit?**
- **Hadoop?**

Let’s get started

- What is a database?
  - A collection of files storing related data
- Examples of databases
  - Accounts database; payroll database; NYUAD’s students database; Amazon’s products database; airline reservation database

Data Management

- What *operations* do we want to perform on this data?
- What *functionality* do we need to manage this data?

Required Functionality

1. Describe real-world entities in terms of stored data
2. Create & persistently store large datasets
3. Efficiently query & update
    1. Must handle complex questions about data (***OLAP***)
    2. Must handle sophisticated updates (***OLTP***)
4. Change structure (e.g., add attributes)
5. Concurrency control: enable simultaneous updates
6. Crash recovery
7. Access control, security, integrity
8. Distribution, *cloudiness*, etc.

Difficult and costly to implement all these features

Managing Data in the 21 Century

- Creating data management infrastructures is perhaps the most challenging part to scale a company
  - Why? Because distributed data management is, strictly speaking, impossible (CAP theorem)

- Some back stories:
  - Facebook VS Friendster
  - Amazon.com VS the rest of the world
  - Google VS the rest of the world

- *Essential* to follow this course to avoid such mistakes in start-ups / large enterprises / administrations / etc.

Database Management System

- A DBMS is a software system designed to provide data management services

- Examples of DBMS
  - Oracle, DB2 (IBM), SQL Server (Microsoft),
  - PostgreSQL, MySQL, SciDB
  - Teradata, Vertica
  - Google BigTable, Yahoo! PNUTS, (Facebook) Cassandra

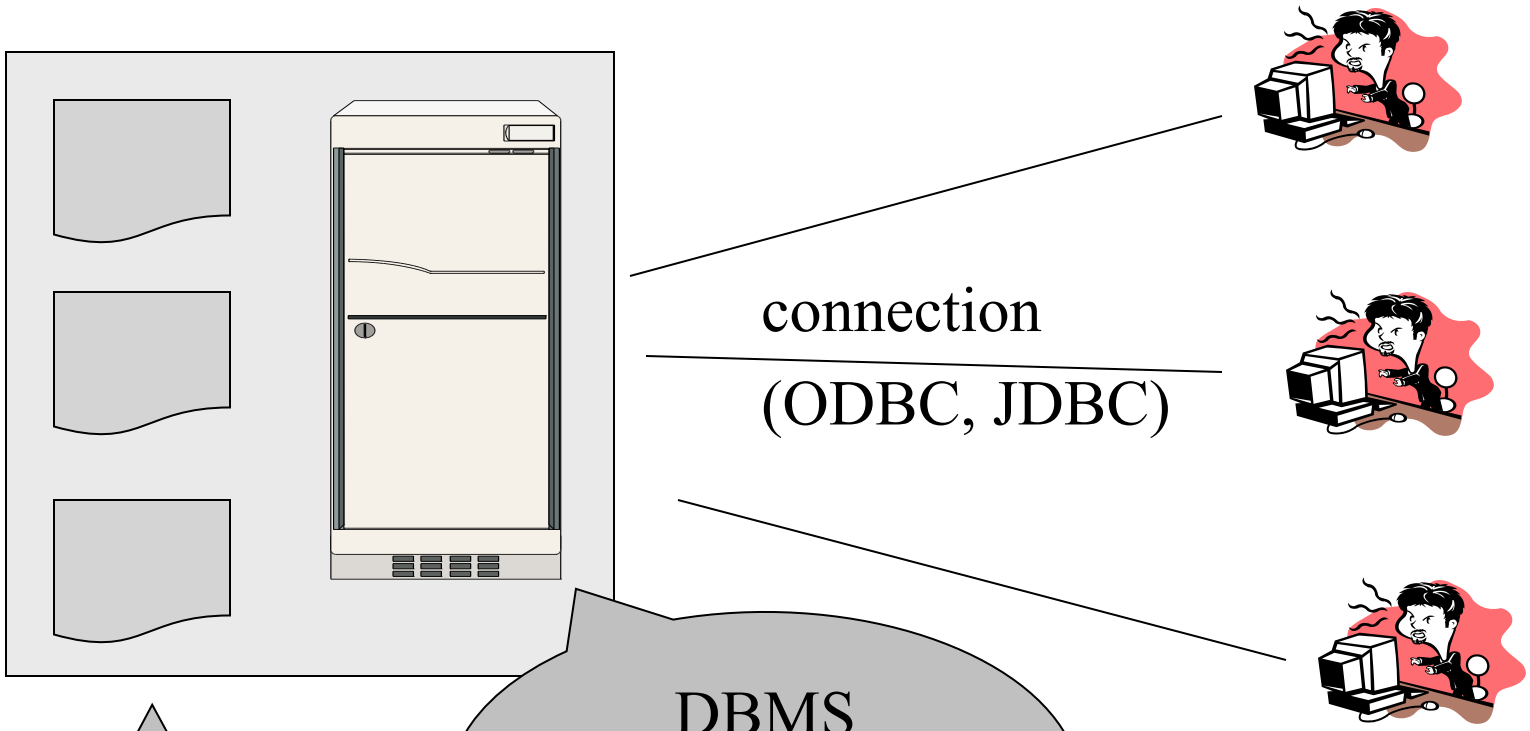
Market Shares

- In 2004 (from www.computerworld.com)
  - IBM, 35% market with $2.5 billion in sales
  - Oracle, 33% market with $2.3 billion in sales
  - Microsoft, 19% market with $1.3 billion in sales

- Quite different today with new players and start-ups
  - no-SQL systems
  - Vertical systems
  - High-end analytics
  - Stream databases
  - Cloud databases
  - Main-memory transactional systems
  - Etc.

Typical System Architecture

"Two tier system" or "client-server"



Main DBMS Features

- Data independence
  - Data model
  - Data definition language
  - Data manipulation language
- Efficient data access
- Data integrity and security
- Data administration
- Concurrency control
- Crash recovery
- Reduced application development time

How to decide what features should go into the DBMS?

When not to use a DBMS?

- A DBMS is optimized for a certain workload
  - No one-size-fits-all solution?
- Some applications may need
  - A completely different data model
  - Completely different operations
  - A few time-critical operations
- Examples
  - Text processing
  - Scientific analysis

Key Concepts in Databases

RELATIONS

Relation Definition

- **Database is collection of relations**

- **Relation R is subset of S_1 x S_2 x … x S_n**
  - Where **S_i** is the domain of attribute **i**
  - **n** is the number of attributes of the relation

- A Relation is basically a table with rows & n columns
  - SQL uses word *table* to refer to relations

Properties of a Relation

- Each row represents an n-tuple of R
- Ordering of rows is immaterial
- All rows are distinct *(**really?**)*

- Often, ordering of columns is significant
  - Because two columns can have same domain
  - But columns are labeled with names
  - Applications need not worry about order, if they simply use the names

- Domain of each column is a primitive type

- Relation consists of a **relation schema** and **instances**

Schema

- **Relation schema**: describes column heads
  - Relation name
  - Name of each field (or column, or attribute)
  - Domain of each field
- **Degree (or arity) of relation**: num. attributes
- **Database schema**: set of all relation schemas

Instance

- **Relation instance**: concrete table content
  - Set of tuples (also called records) matching the schema
- **Cardinality of relation instance**: num. tuples
- **Database instance**: set of all relation instances

Example

- Relation schema

  Supplier(sno: integer, sname: string, scity: string, scountry: string)

  Degree?

- Relation instance

sno	sname	scity	scountry
1	s1	city 1	FR
2	s2	city 1	FR
3	s3	city 2	FR
4	s4	city 2	FR

Cardinality?

Key Concepts in Databases

CONSTRAINTS

Integrity Constraints

- **Integrity constraint**
  - Condition specified on a database schema
  - Restricts data that can be stored in db instance
- DBMS enforces integrity constraints
  - Ensures only legal database instances exist
- Simplest form of constraint is domain constraint
  - Attribute values must come from attribute domain

Key Constraints

- **Key constraint:** "certain minimal subset of fields is a **unique identifier** for a tuple"

- **Candidate key**
  - Minimal set of fields
  - Uniquely identify each tuple in a relation

- **Primary key**
  - **One** candidate key can be selected as primary key

Foreign Key Constraints

- A relation can refer to a tuple in another relation

- **Foreign key**
  - Field/Attribute that refers to tuples in another relation
  - Typically, this field refers to the primary key of other relation

Key Constraint SQL Examples

```
CREATE TABLE Part (
   pno integer,              //part number
   pname varchar(20),        //part name
   psize integer,            // part size
   pcolor varchar(20),       // part color
   PRIMARY KEY (pno)
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,     //supply number
   pno integer,     //part number
   qty integer,     // quantity
   price integer    // price
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno)
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno),
   FOREIGN KEY (sno) REFERENCES Supplier,
   FOREIGN KEY (pno) REFERENCES Part
);
```

Key Constraint SQL Examples

```
CREATE TABLE Supply(
   sno integer,
   pno integer,
   qty integer,
   price integer,
   PRIMARY KEY (sno,pno),
   FOREIGN KEY (sno) REFERENCES Supplier
                          ON DELETE NO ACTION,
   FOREIGN KEY (pno) REFERENCES Part
                          ON DELETE CASCADE
);
```

General Constraints

- Table constraints serve to express complex constraints over a single table

```
CREATE TABLE Part (
  pno integer,
  pname varchar(20),
  psize integer,
  pcolor varchar(20),
  PRIMARY KEY (pno),
  CHECK ( psize > 0 )
);
```

Key Concepts in Databases

RELATIONAL ALGEBRA AND OPERATORS

Relational Queries and Algebra

- **Query language** associated with relational model
- **Query inputs and outputs are relations**
- **Queries specified in an operational manner**
  - A query gives a list of operations on data
- **Relational operators**
  - Take one or more relation instances as argument
  - Return one relation instance as result
  - Easy to **compose** into **relational algebra expressions**

Relational Operators

- Let S and R bet a set of n-tuples from some relations.
- Selection: $\sigma_{condition}(S)$
  - Condition is Boolean combination ($\wedge$,$\vee$) of terms (predicates)
    - attribute operator constant.
    - attribute operator attribute.
  - Operator is: <, <=, =, ≠, >=, or >
- Projection: $\pi_{list\text{-}of\text{-}attributes}(S)$
- Union ($\cup$), Intersection ($\cap$), Set difference (–),
- Cross-product or Cartesian product ($\times$)
- Join: $R \bowtie_\theta S = \sigma_\theta(R \times S)$

Selection & Projection Examples

Patient

no	name	zip	disease
1	p1	98125	flu
2	p2	98125	heart
3	p3	98120	lung
4	p4	98120	heart

$\pi_{zip,disease}$(Patient)

zip	disease
98125	flu
98125	heart
98120	lung
98120	heart

$\sigma_{disease='heart'}$(Patient)

no	name	zip	disease
2	p2	98125	heart
4	p4	98120	heart

π_{zip} ($\sigma_{disease='heart'}$(Patient))

zip
98120
98125

Cross-Product Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters

name	age	zip
p1	54	98125
p2	20	98120

P x V

P.age	P.zip	disease	name	V.age	V.zip
54	98125	heart	p1	54	98125
54	98125	heart	p2	20	98120
20	98120	flu	p1	54	98125
20	98120	flu	p2	20	98120

Different Types of Join

- **Theta-join**: $R \bowtie_\theta S = \sigma_\theta(R \times S)$
  - Join of R and S with a join condition θ
  - Cross-product followed by selection θ

- **Equijoin**: $R \bowtie_\theta S = \pi_A (\sigma_\theta(R \times S))$
  - Join condition θ consists only of equalities
  - Projection π_A drops some redundant attributes

- **Natural join**: $R \bowtie S = \pi_A (\sigma_\theta(R \times S))$
  - Equijoin
  - Equality on **all** fields with same name in R and in S

Theta-Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age \wedge P.zip=A.zip \wedge P.age < 50} V$

P.age	P.zip	disease	name	V.age	V.zip
20	98120	flu	p2	20	98120

Equijoin Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \bowtie_{P.age=V.age} V$

age	P.zip	disease	name	V.zip
54	98125	heart	p1	98125
20	98120	flu	p2	98120

Natural Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu

Voters V

name	age	zip
p1	54	98125
p2	20	98120

P ⋈ V

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2

More Joins

- **Outer join**
  - Natural Join including tuples with no matches in the output
  - Use NULL values for missing attributes

- Variants
  - Left outer join
  - Right outer join
  - Full outer join

Outer Join Example

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

Voters V

name	age	zip
p1	54	98125
p2	20	98120

$P \overset{o}{\bowtie} V$

age	zip	disease	name
54	98125	heart	p1
20	98120	flu	p2
33	98120	lung	null

Example of Algebra Queries

Q1: Names of patients who have heart disease

$\pi_{name}(Voter \bowtie(\sigma_{disease='heart'} (AnonPatient)))$

AnonPatient P

age	zip	disease
54	98125	heart
20	98120	flu
33	98120	lung

Voters V

name	age	zip
p1	54	98125
p2	20	98120

More Examples

Relations

```
Supplier(sno,sname,scity,sstate)
Part(pno,pname,psize,pcolor)
Supply(sno,pno,qty,price)
```

Q2: Name of supplier of parts with size greater than 10

π_{sname}(Supplier $\bowtie$ Supply $\bowtie$($\sigma_{psize>10}$ (Part))

Q3: Name of supplier of red parts or parts with size greater than 10

π_{sname}(Supplier $\bowtie$ Supply $\bowtie$($\sigma_{psize>10}$ (Part) $\cup$ $\sigma_{pcolor='red'}$ (Part)))

(Many more examples online and in books)

Extended Operators of Relational Algebra

- Duplicate elimination (δ)
  - Since commercial DBMSs operate on multisets not sets
- Aggregate operators (γ)
  - Min, max, sum, average, count
- Sort operator (τ)

Key Concepts in Databases

SQL

Structured Query Language: SQL

- Declarative query language
  - Logic over how to (the latter is the DBMS job)
- Multiple aspects of the language
  - Data definition language
    - Statements to create, modify tables and views
  - Data manipulation language
    - Statements to issue queries, insert, delete data
  - More

SQL Query

Basic form:

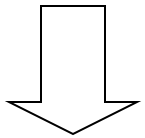
```
SELECT  <attributes>
FROM    <one or more relations>
WHERE   <conditions>
```

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  *
FROM    Product
WHERE   category='Gadgets'
```



PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

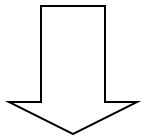
"selection"

Simple SQL Query

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT  PName, Price, Manufacturer
FROM    Product
WHERE   Price > 100
```

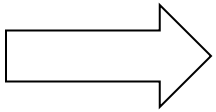


"selection" and "projection"

PName	Price	Manufacturer
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

Eliminating Duplicates

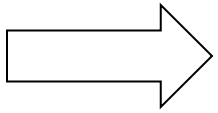
```
SELECT  DISTINCT category
FROM    Product
```



Category
Gadgets
Photography
Household

Compare to:

```
SELECT  category
FROM    Product
```



Category
Gadgets
Gadgets
Photography
Household

Ordering the Results

```
SELECT   pname, price, manufacturer
FROM     Product
WHERE    category='gizmo' AND price > 50
ORDER BY  price, pname
```

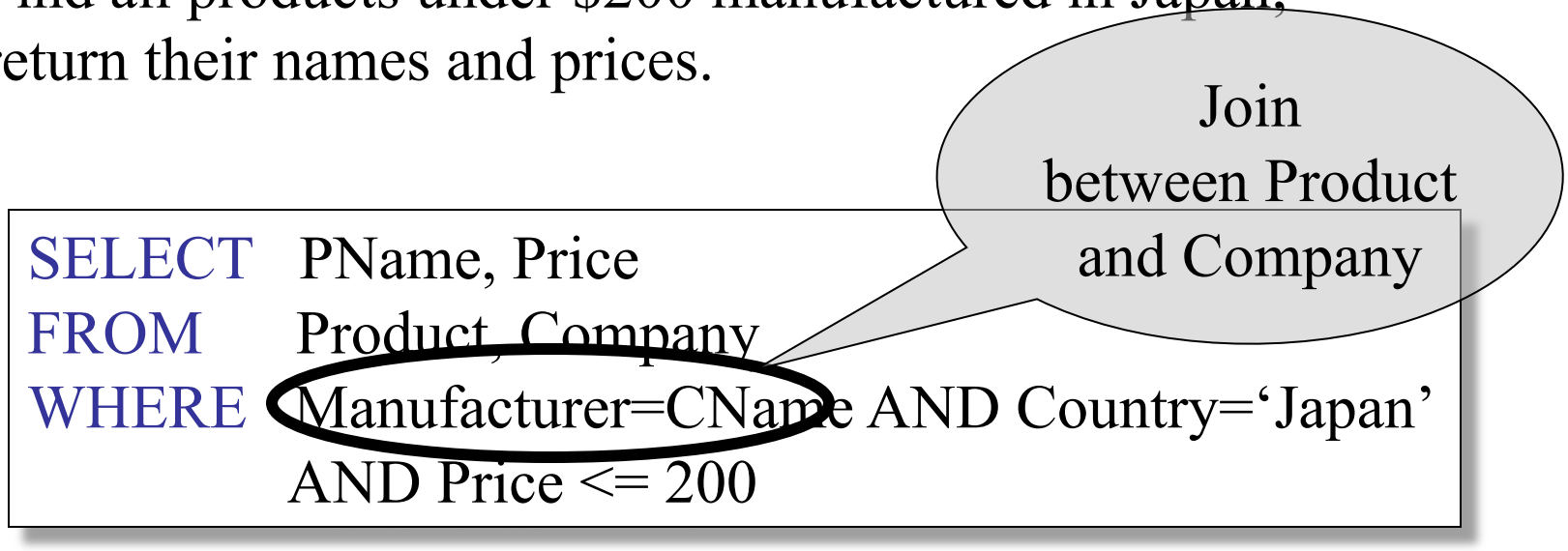
Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

Joins

Product (pname, price, category, manufacturer)
Company (cname, stockPrice, country)

Find all products under $200 manufactured in Japan; return their names and prices.

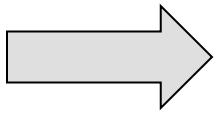


Tuple Variables

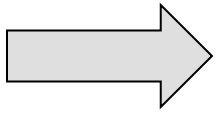
Person(pname, address, worksfor)
Company(cname, address)

```
SELECT   DISTINCT pname, address
FROM     Person, Company
WHERE    worksfor = cname
```

Which address ?



```
SELECT   DISTINCT Person.pname, Company.address
FROM     Person, Company
WHERE    Person.worksfor = Company.cname
```



```
SELECT   DISTINCT x.pname, y.address
FROM     Person AS x, Company AS y
WHERE    x.worksfor = y.cname
```

Nested Queries

- **Nested query**
  - Query that has another query embedded within it
  - The embedded query is called a **subquery**

- Why do we need them?
  - Enables us to refer to a table that must itself be computed

- Subqueries can appear in
  - WHERE clause (common)
  - FROM clause (less common)
  - HAVING clause (less common)

Subqueries Returning Relations

Company(name, city)
Product(pname, maker)
Purchase(id, product, buyer)

Return cities where one can find companies that manufacture products bought by Joe Blow

```
SELECT  Company.city
FROM    Company
WHERE  Company.name  IN
                (SELECT Product.maker
                  FROM   Purchase , Product
                  WHERE Product.pname=Purchase.product
                      AND Purchase .buyer = 'Joe Blow');
```

Subqueries Returning Relations

You can also use: s > ALL R
s > ANY R
EXISTS R

Product (pname, price, category, maker)

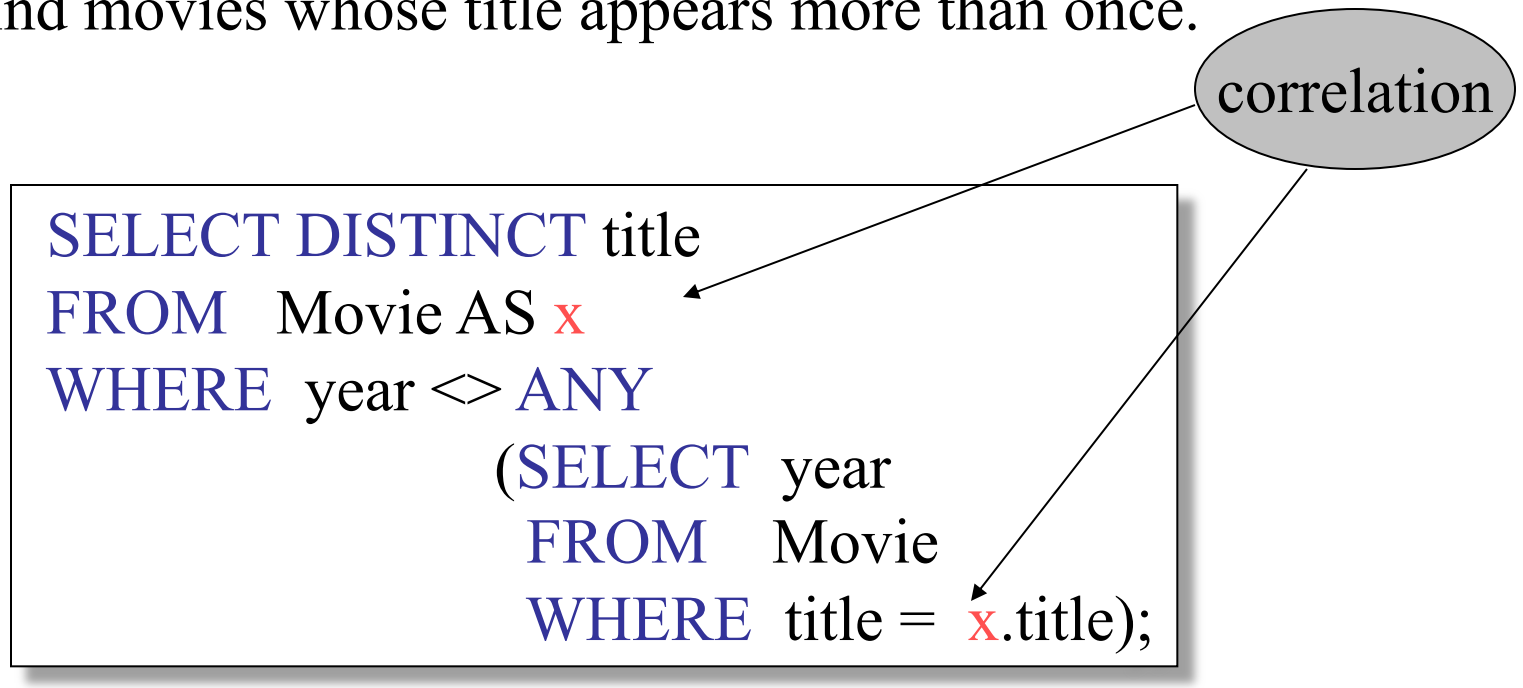
Find products that are more expensive than all those produced By "Gizmo-Works"

```
SELECT  name
FROM    Product
WHERE  price >  ALL (SELECT price
                     FROM    Purchase
                     WHERE  maker='Gizmo-Works')
```

Correlated Queries

Movie (title, year, director, length)

Find movies whose title appears more than once.



Complex Correlated Query

Product (pname, price, category, maker, year)

- Find products (and their manufacturers) that are more expensive than all products made by the same manufacturer before 1972

```
SELECT DISTINCT  pname, maker
FROM     Product AS x
WHERE  price > ALL  (SELECT  price
                     FROM    Product AS y
                     WHERE  x.maker = y.maker AND y.year < 1972);
```

Aggregation

```
SELECT  avg(price)
FROM    Product
WHERE   maker="Toyota"
```

```
SELECT  count(*)
FROM    Product
WHERE   year > 1995
```

SQL supports several aggregation operations:

sum, count, min, max, avg

Except count, all aggregations apply to a single attribute

Grouping and Aggregation

```
SELECT    S
FROM      R1,…,Rn
WHERE     C1
GROUP BY  a1,…,ak
HAVING    C2
```

Conceptual evaluation steps:

1. Evaluate FROM-WHERE, apply condition C1
2. Group by the attributes $a_1,\ldots,a_k$
3. Apply condition C2 to each group (may have aggregates)
4. Compute aggregates in S and return the result

Group By Example

Product(pname, pprice, maker, category, quantityOrdered)

```
SELECT category, SUM(quantityOrdered) FROM Product
WHERE category='Laptop' OR category='Desktop' OR category='Tablet'
GROUP BY category
HAVING SUM(quantityOrdered)>1500
```

Results:

Category	Sum(quantityOrdered)
Laptop	20000
Desktop	2000

Many more examples in book / online…

Key Concepts in Databases

DATA WAREHOUSING

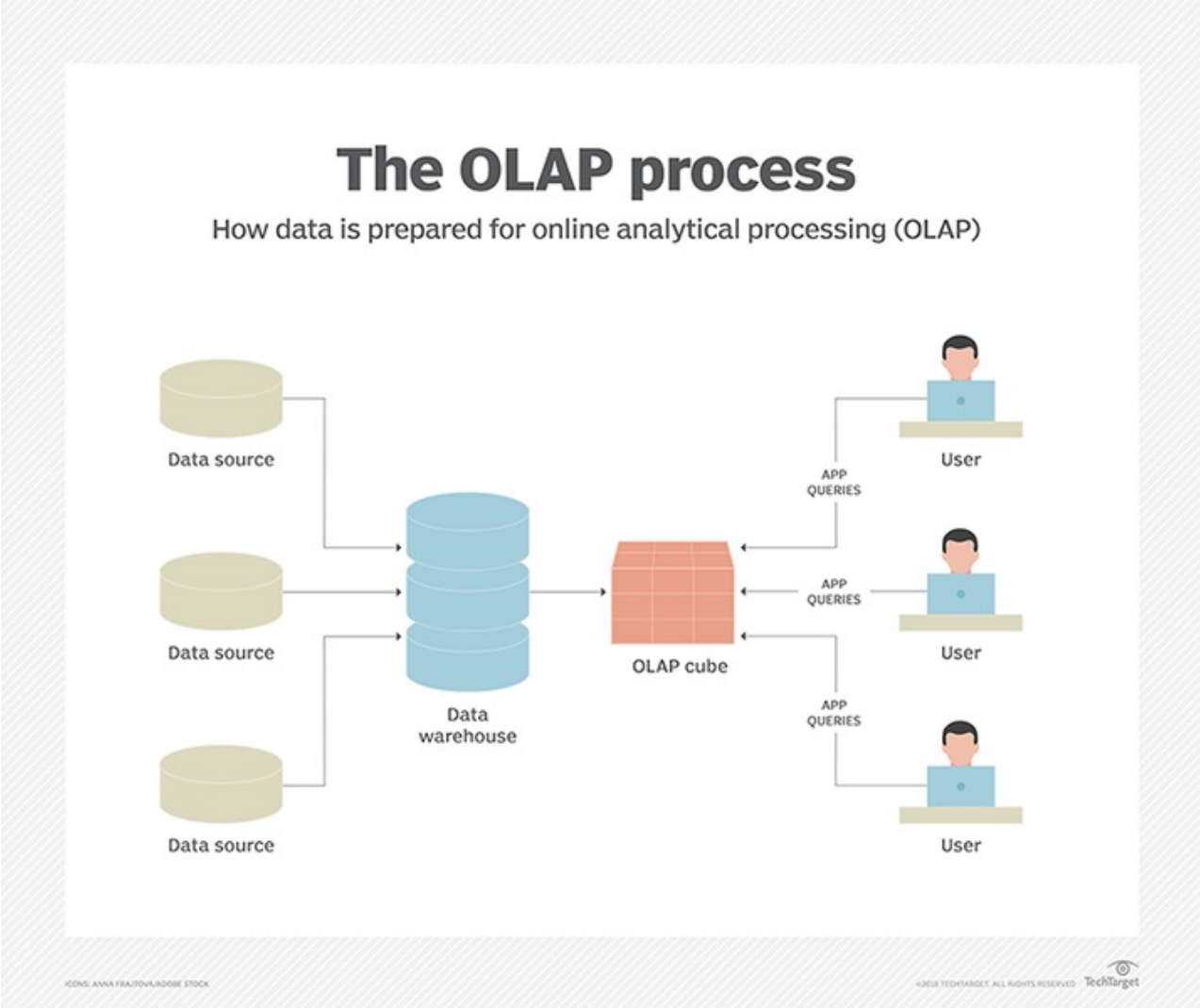
Data Warehouses

- DBMSs designed to manage operational data
  - Goal: support every day activities
  - Online transaction processing (OLTP)
  - Ex: Tracking sales and inventory of each Walmart store

- Enterprises also need to analyze and explore their data
  - Goal: summarize and discover trends to support decision making
  - Online analytical processing (OLAP)
  - Ex: Analyzing sales of all Walmart stores by quarter and region

- To support OLAP consolidate all data into a **warehouse**
  - Lazy data replication

From OLTP to OLAP



Data Warehouse Overview

- Consolidated data from many sources
  - Must create a single unified schema
  - The warehouse is like a materialized view
- Very large size: terabytes of data are common
- Complex read-only queries (no updates)
- Fast response time is important

Creating a Data Warehouse

- **Extract** data from distributed operational databases
- **Clean** to minimize errors and fill in missing information
- **Transform** to reconcile semantic mismatches
  - Performed by defining views over the data sources
- **Load** to materialize the above defined views
  - Build indexes and additional materialized views
- **Refresh** to propagate updates to warehouse periodically

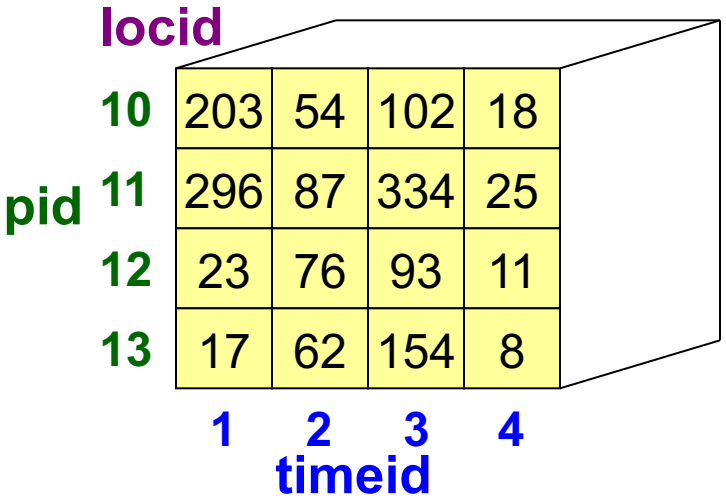
Data Analysis Cycle

- Formulate query that extracts data from the database
  - Typically ad-hoc complex query with group by and aggregate
- Visualize the data (e.g., spreadsheet)
  - Dataset is an N-dimensional space
- Analyze the data
  - Identify “interesting” subspace by aggregating other dimensions
  - Categorize the data an compare categories with each other
  - Roll-up and drill-down on the data

Multidimensional Data Model

- Focus of the analysis is a collection of measures
  - Example: Ikea sales

- Each measure depends on a set of dimensions
  - Example: product (pid), location (lid), and time of the sale (timeid)

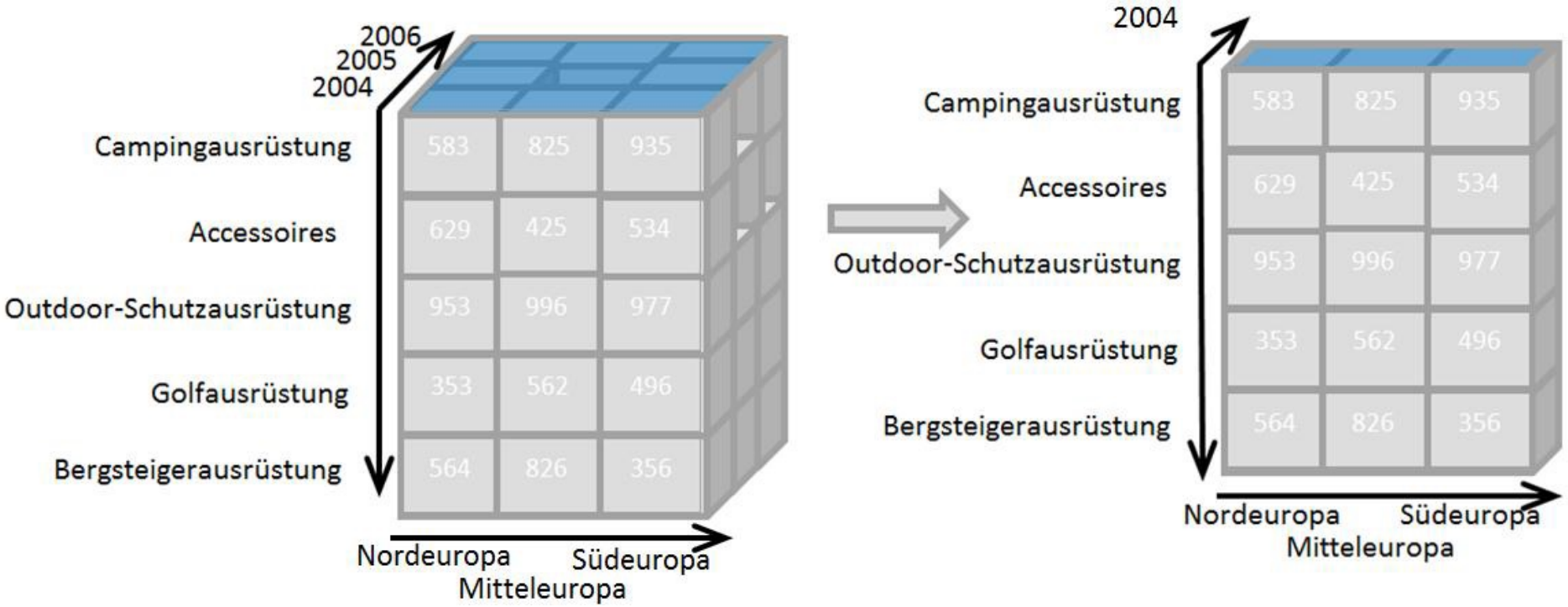


Slicing: equality selection on one or more dimensions

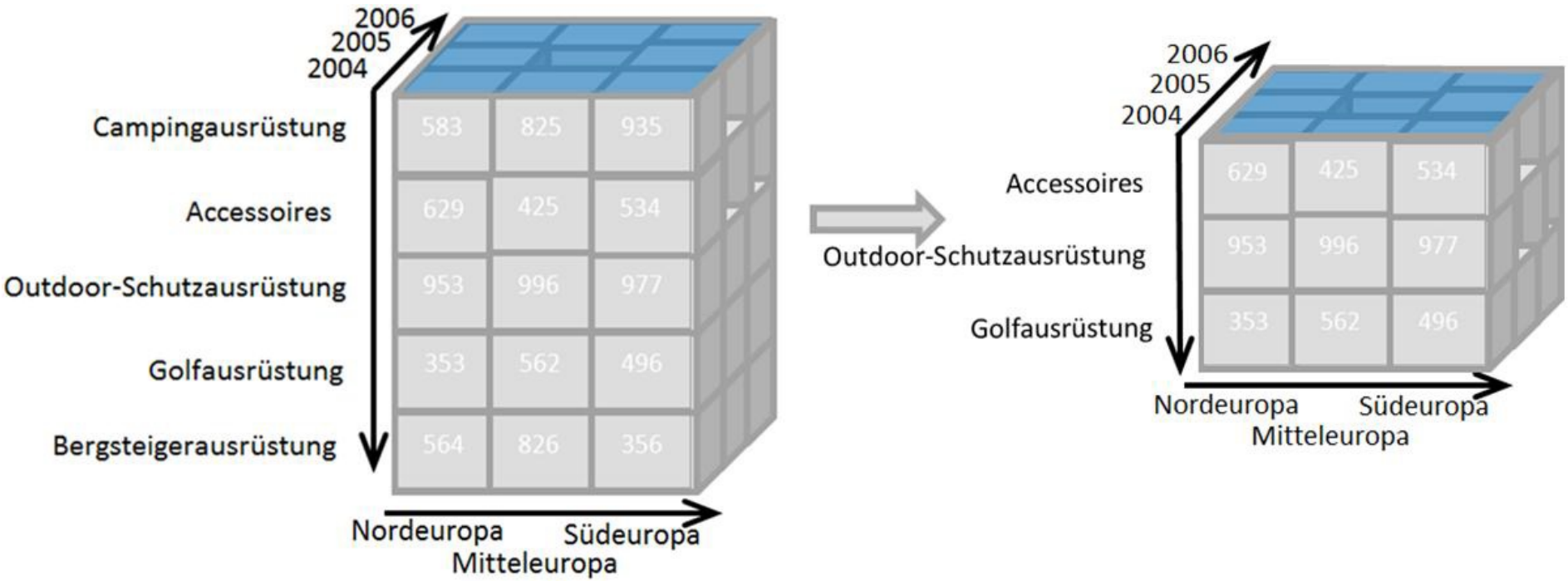
Dicing: range selection

Slicing

- [from Wikipedia]



Dicing


2006
2005
2004
Campingausrüstung
Accessoires
Outdoor-Schutzausrüstung
Golfausrüstung
Bergsteigerausrüstung
583 825 935
629 425 534
953 996 977
353 562 496
564 826 356
Nordeuropa
Mitteleuropa
Südeuropa
Outdoor-Schutzausrüstung
Accessoires
Golfausrüstung
629 425 534
953 996 977
353 562 496
Nordeuropa
Mitteleuropa
Südeuropa

Data Cube

- CUBE is the N-dimensional generalization of aggregate
- Cube in SQL-1999

```
SELECT T.year, L.state, SUM(S.sales)
FROM Sales S, Times T, Locations L
WHERE S.timeid=T.timeid and S.locid=L.locid
GROUP BY CUBE (T.year,L.state)
```

- Creating a data cube requires generating the power set of the aggregation columns