



CS6083: Principles of Databases Systems

Section A (Prof. Suel)

- **Tuesdays from 3:22pm to 5:50pm in room 202 in 370 Jay St.**
- **Instructor: Torsten Suel torsten.suel@nyu.edu**
- **Course page: on NYU Courses**
- **Office hours: Tuesdays 2-3pm in 10.046 (2 MetroTech)**
- **Also, graders will have their own office hours, as needed**

- **Textbook: Silberschatz, Korth, Sudarshan: Database Systems Concepts, 6th Edition, McGraw-Hill 2009. (The 4th and 5th editions are also OK to use.)**
- **Will use slides from book as well as extra material**
- **Usually posted on NYU Courses ahead of class**



CS6083: Principles of Database Systems

- **Prerequisites: Some algorithms and data structures**
 - **Binary search trees, hash tables, heaps, sorting**
 - **Basic OS structure, concurrency, file systems, caching**
 - **Also useful: some basic logic and discrete math
(Boolean AND/OR, sets and relations, Exists/All notation)**
 - **Plus general programming experience**
-
- **If you have deficiencies, make sure to review basic data structures, algorithms, and OS by late September.**



CS6083: Principles of Database Systems

- No previous DB required – we start from beginning!
- Know too much already? → maybe you don't ...
- WE MOVE FAST (cover chapters 1-16 of book)
- This is a Computer-Science oriented course
- How to use a database system (1-8)
- But also how database systems work internally (9-16)
- Large project in second half of semester
- Project usually combines DB, web, text, social networks



Homeworks and Grading

- **5 Homeworks: 4 graded, 1 ungraded**
- **Homeworks are mix of theory and hands-on**
- **Submission via online system – wait for instructions**
- **Homeworks are 20% of total grade (4 x 5%)**
- **Do not copy!**
- **Project in 2 parts, 20% (7% + 13%)**
- **Second part with a live demo to the graders**
- **Midterms or Tests 20%, Final Exam 40% of total grade**
- **Graders: TBD**



Course Discussions

- **Discussions on piazza**
- **You will receive email on how to access**
- **Ask questions, see answers**
- **Instructors, TAs, and other students can answer**
- **3 ways to post: public, private, anonymous**
- **Only use private for direct questions about your work**
- **Anonymous means everyone can see question and answer, but not who you are**
- **Or use public**
- **Please do not use messaging inside NYU classes to contact the instructor or TAs – use email**



Chapter 1: Introduction

- **Purpose of Database Systems**
- **View of Data**
- **Database Languages**
- **Relational Databases**
- **Database Design**
- **Object-based and semistructured databases**
- **Data Storage and Querying**
- **Transaction Management**
- **Database Architecture**
- **Database Users and Administrators**
- **Overall Structure**
- **History of Database Systems**
- **Some Popular DB Systems**
- **Perspective: Databases vs. Information Retrieval**



Database Management System (DBMS)

- Basically, a system for managing data
- DBMS contains information about a particular enterprise (application)
 - Collection of interrelated data
 - Set of programs to access the data
 - An environment that is both *convenient* and *efficient* to use
- Database Applications:
 - Banking: all transactions
 - Airlines: reservations, schedules
 - Universities: registration, grades
 - Sales: customers, products, purchases
 - Online retailers: order tracking, customized recommendations
 - Manufacturing: production, inventory, orders, supply chain
 - Human resources: employee records, salaries, tax deductions
 - Telecommunications: accounts, phone cards, 1-800
- Note: integration of different databases is a huge issue



University Database Example

- **Application and application program example:**
 - **University application**
 - **Add new students, instructors, and courses**
 - **Register students for courses, and generate class rosters**
 - **Assign grades to students, compute grade point averages (GPA) and generate transcripts**
- **Another one: customers, products, purchases**
- **In the early days, database applications were built directly on top of file systems**



Purpose of Database Systems

- **Drawbacks of using file systems to store data:**
 - **Data redundancy and inconsistency**
 - ▶ Multiple file formats, duplication of information in different files
 - **Difficulty in accessing data**
 - ▶ Need to write a new program to carry out each new task
 - **Data isolation — multiple files and formats**
 - **Integrity problems**
 - ▶ Integrity constraints (e.g. account balance > 0) become “buried” in program code rather than being stated explicitly
 - ▶ Hard to add new constraints or change existing ones



Purpose of Database Systems (Cont.)

■ Drawbacks of using file systems (cont.)

● Atomicity of updates

- ▶ Failures may leave database in an inconsistent state with partial updates carried out
- ▶ Example: Transfer of funds from one account to another should either complete or not happen at all

● Concurrent access by multiple users

- ▶ Concurrent access needed for performance and usability
- ▶ Uncontrolled concurrent accesses can lead to inconsistencies
 - Example: Two people reading a balance and updating it at the same time

● Security problems

- ▶ Hard to provide user access to some, but not all, data

■ Modern database systems offer solutions to all the above problems



Levels of Abstraction

- **Physical level:** describes how data records (e.g., a customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

```
type customer = record
```

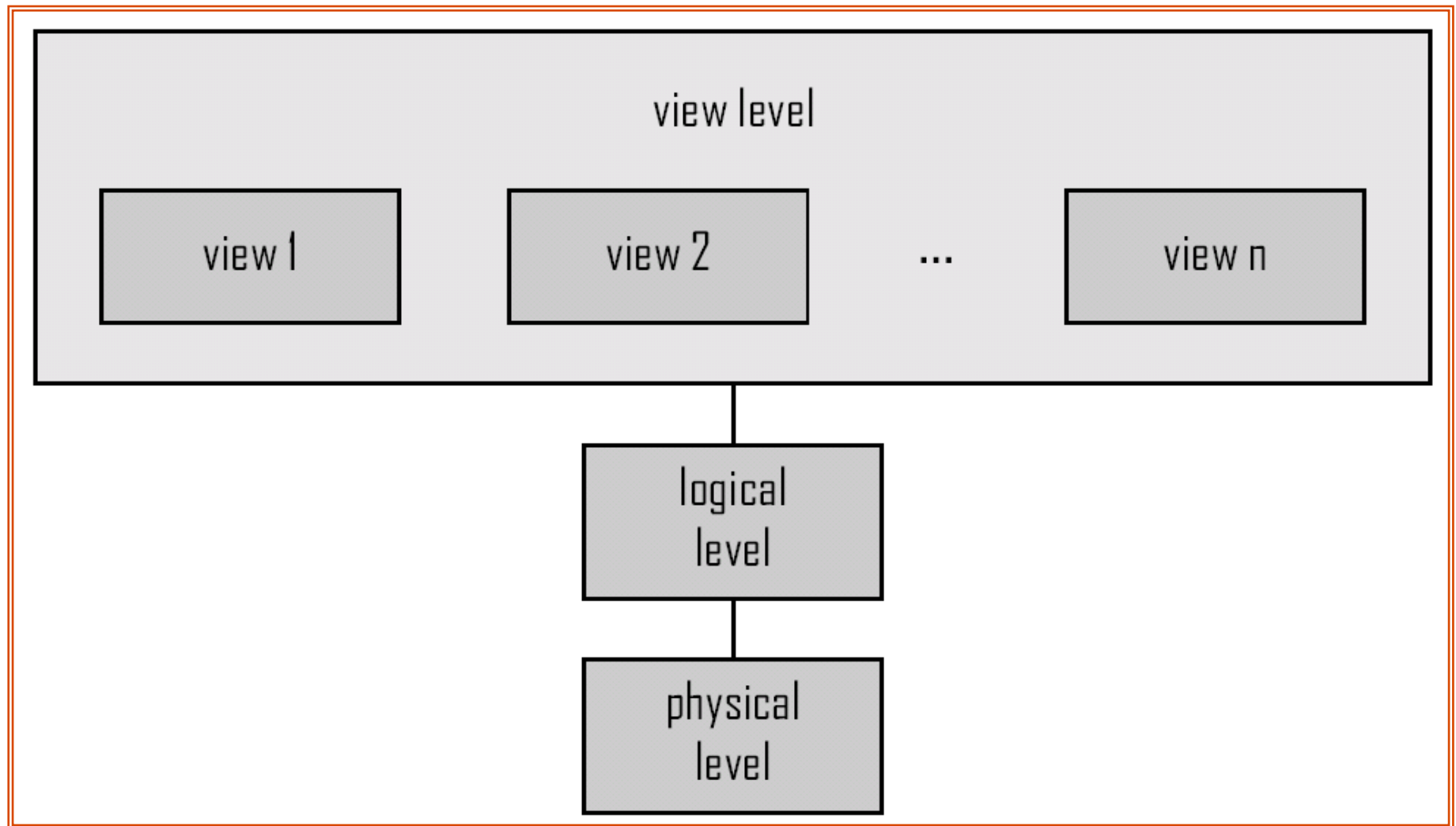
```
    customer_id : string;  
    customer_name : string;  
    customer_street : string;  
    customer_city : integer;
```

```
end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes – in this case details are hidden *from* the application program



View of Data





Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
 - Example: The database consists of information about a set of customers and accounts and the relationship between them)
 - Analogous to type information of a variable in a program
 - Physical schema: database design at the physical level
 - Logical schema: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
 - Analogous to the current values of a set of variables
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
 - Applications depend on the logical schema (or the view level schema)
 - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.



Data Models

- A collection of tools for describing
 - Data
 - Data relationships
 - Data semantics
 - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-Oriented and Object-Relational)
- Semistructured data model (e.g., XML)
- Other older models (1960s):
 - Network model
 - Hierarchical model



Relational Model

- Relational model (Chapter 2)
- Example of tabular data in the relational model

Columns

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Rows

(a) The *instructor* table



A Sample Relational Database

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



Another Sample Relational Database

<i>customer_id</i>	<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
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

<i>account_number</i>	<i>balance</i>
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

<i>customer_id</i>	<i>account_number</i>
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



Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
 - DML also known as query language
- Two classes of languages
 - **Procedural** – user specifies what data is desired and how to compute the data
 - **Declarative (nonprocedural)** – user specifies what data is desired without specifying how to compute the data
- SQL is the most widely used query language
- SQL is nonprocedural ... basically, sort of



Data Definition Language (DDL)

- Specification notation for defining the database schema

Example: **create table *account* (**
 account-number **char(10),**
 balance **integer)**

- DDL compiler generates a set of tables stored in a *data dictionary*
- Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Storage structure and supported access methods
 - ▶ Specified using a *data storage and definition language*
 - Integrity constraints
 - ▶ Domain constraints
 - ▶ Referential integrity (references constraint in SQL)
 - ▶ Assertions
 - Authorization



SQL

- **SQL**: widely used non-procedural language (DDL+DML)
 - Example: Find the name of the instructor with ID 22222

```
select name  
from instructor  
where instructor.ID = '22222'
```
 - **select** *instructor.ID, department.dept name*
from *instructor, department*
where *instructor.dept name*= *department.dept name* **and**
department.budget > 95000
- Application programs generally access databases through one of
 - Language extensions to allow embedded SQL
 - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database
- Chapters 3, 4 and 5



Database Design

The process of designing the general structure of the database:

- **Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.**
 - **Business decision – What attributes should we record in the database?**
 - **Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?**
- **Physical Design – Deciding on the physical layout of the database**
- **Design at the application level (views, stored procedures, interfaces)**
- **Logical design based on experience *and* based on sound theory**
- **... and based on knowledge of the application domain**



Database Design: Example

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

<i>dept_name</i>	<i>building</i>	<i>budget</i>
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table



Database Design?

■ Is there any problem with this design?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000



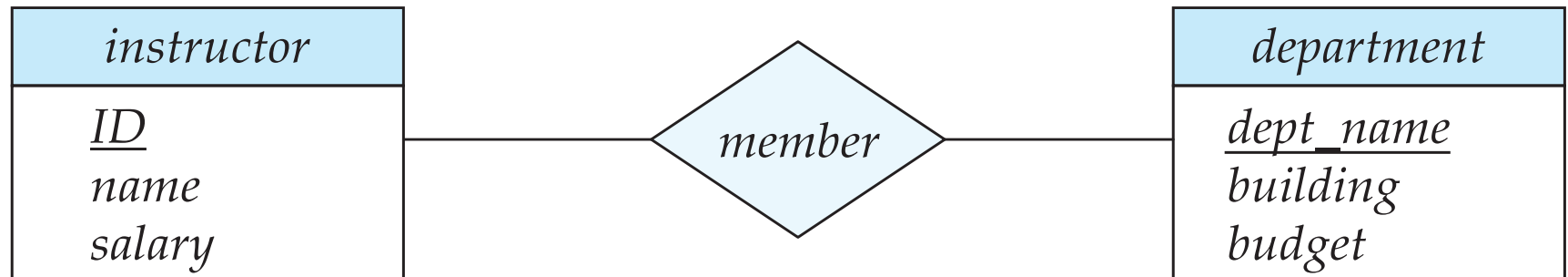
Design Approaches

- Normalization Theory (Chapter 8)
 - Formalize what designs are bad, and test for them
- Entity Relationship Model (Chapter 7)
 - Models scenario as a collection of *entities* and *relationships*
 - ▶ Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - Described by a set of *attributes*
 - ▶ Relationship: an association among several entities
 - Represented diagrammatically by an *entity-relationship diagram*:
- DB Design: Science + Craft



The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
 - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - ▶ Described by a set of *attributes*
 - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram*:



What happened to dept_name of instructor?



Object-Relational Data Models

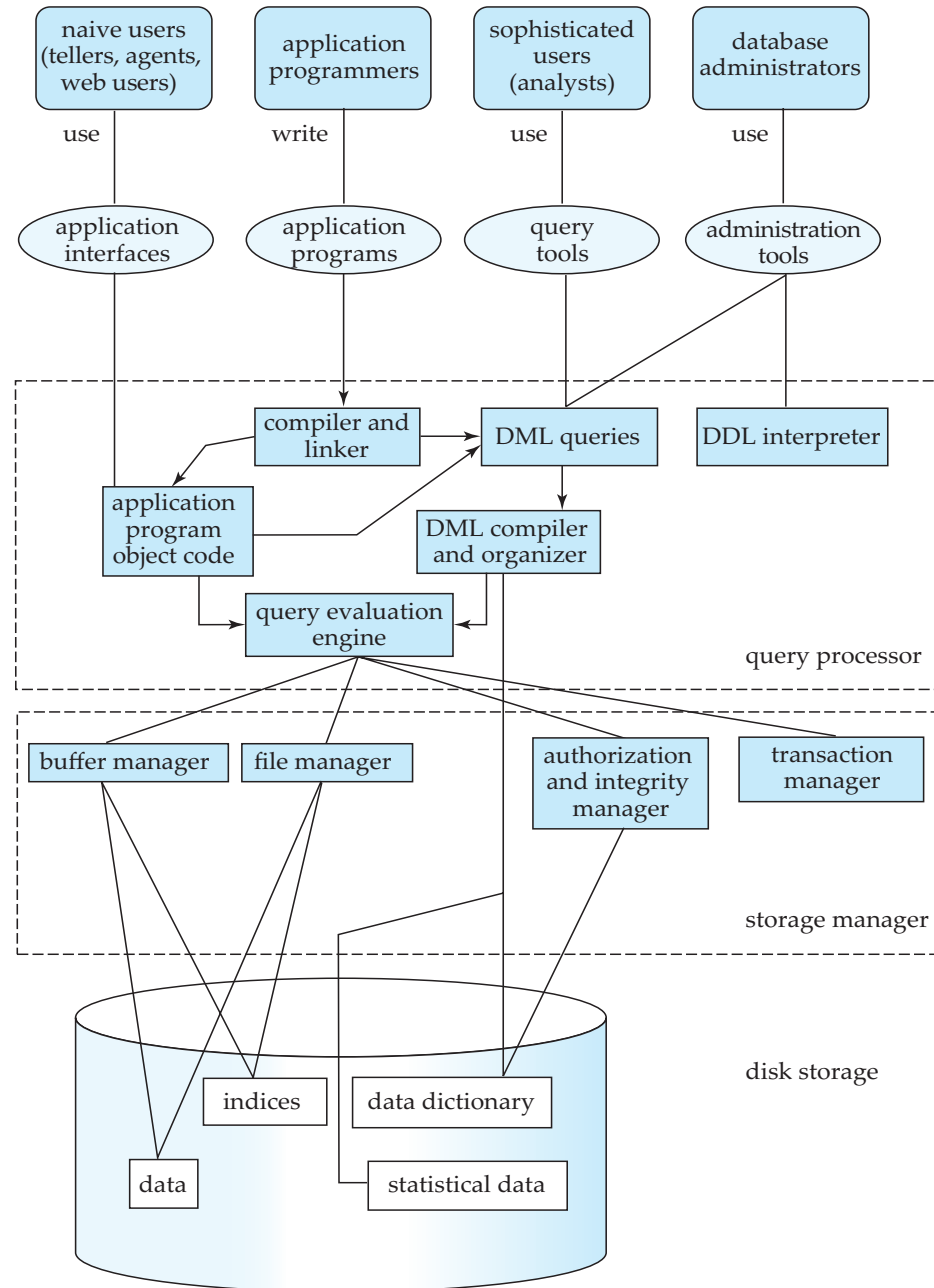
- Extend the relational data model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
- Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language, not a DB language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange data, not just documents
- XML has become basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data,

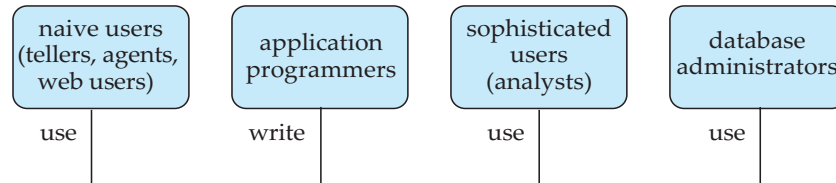


Database System Internals



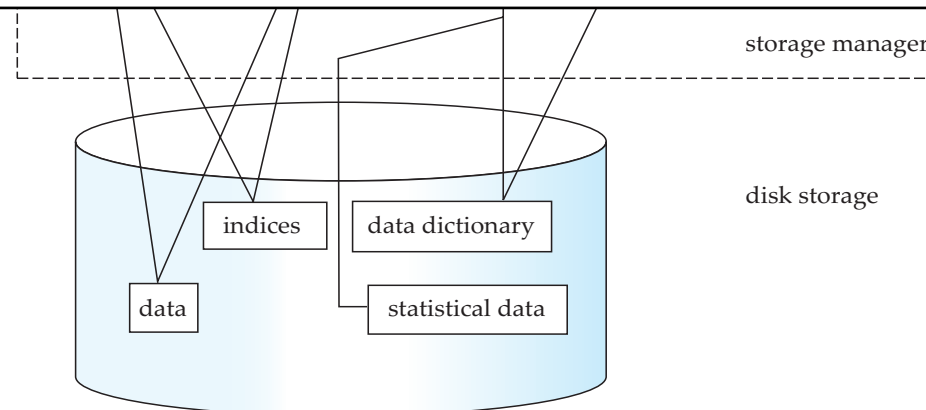


Database System Internals



What is the point here?

- Database systems are complicated pieces of software
- Millions of lines of code, 1000s of years of programming
- Many components, but some common structure
- Lower levels for data access, higher for queries
- We will learn a lot more about this during the semester





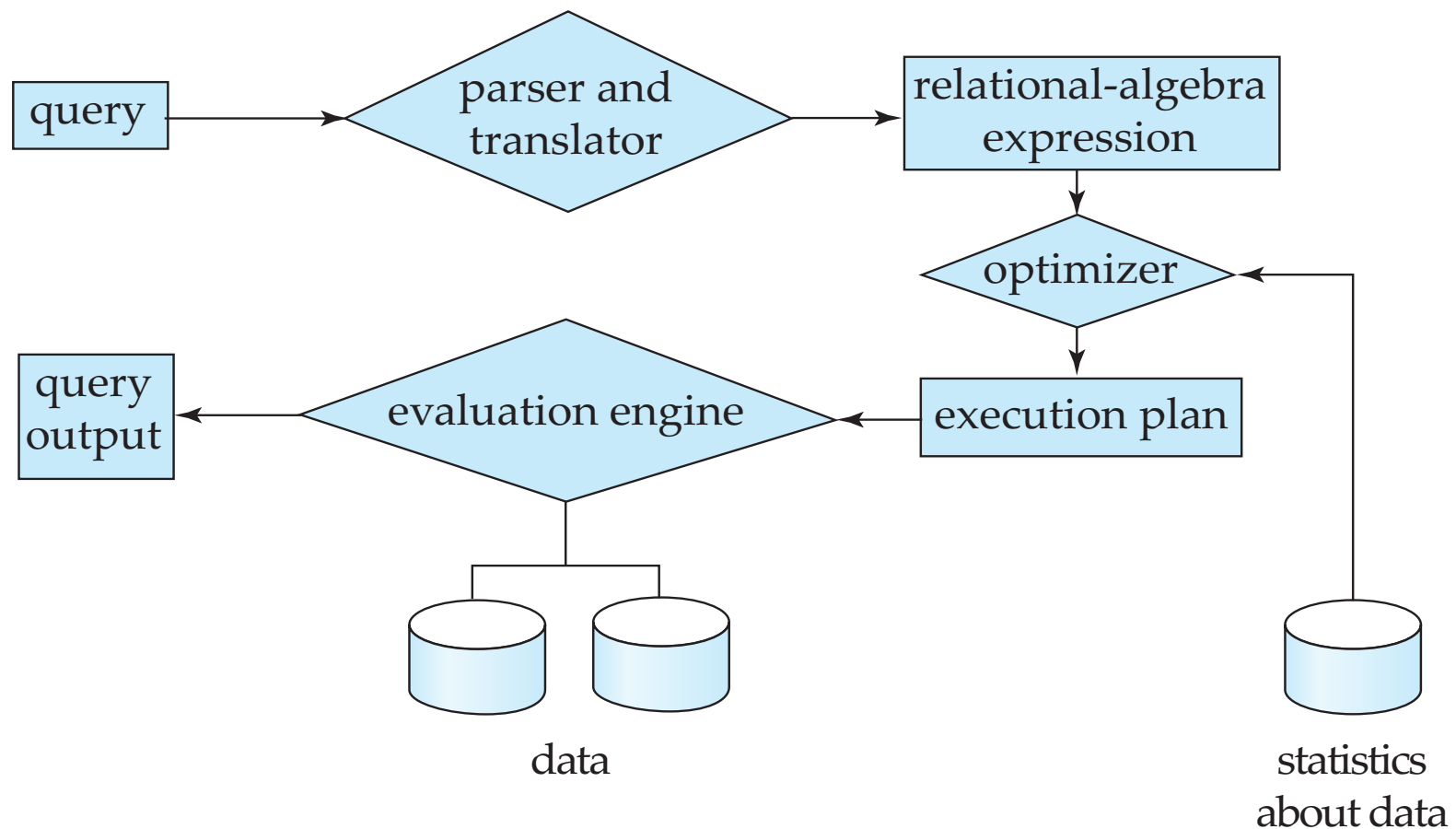
Storage Management

- **Storage manager:** program module that provides interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- **The storage manager is responsible to the following tasks:**
 - interaction with the file manager (OS)
 - efficient storage, retrieval and updating of data
- **Components:**
 - authorization/integrity manager
 - transaction manager
 - file manager
 - buffer manager
- **The storage manager stores:**
 - data (relations)
 - data dictionary (sometimes called catalog)
 - Statistical data (samples, histograms)
 - indexes



Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation





Query Processing (Cont.)

- **Alternative ways of evaluating a given query**
 - **Equivalent expressions**
 - **Different algorithms for each operation**
- **Cost difference between a good and a bad way of evaluating a query can be enormous**
- **Cost-based and rule-based optimization**
- **Cost-based: need to estimate the cost of operations**
 - **Depends critically on statistical information about relations which the database must maintain**
 - **Need to estimate statistics for intermediate results to compute cost of complex expressions**



Transaction Management

- What is the system fails?
- What if more than one user is concurrently accessing the same data?
- A *transaction* is a collection of operations that performs a single logical function in a database application
 - e.g., deposit, withdrawal, transfer between accounts
- Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures
 - e.g., system crash cannot wipe out “committed” transactions
- Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database
 - e.g., two users accessing the same bank account cannot “corrupt” the system or withdraw more than allowed



Example: E-Commerce Database

- Let's try to model a database for a simple e-commerce store
- Needs to keep track of:
 - Customers and their information (name, address, etc.)
 - Products and their information (name, type, description)
 - Which customer bought which product at what price



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr

■ **Many simplifying assumptions:**

- **Only one credit card and one address per user. Address simplified**
- **Only one item per order**
- **No shipping costs, no taxes, no reviews, no tracking, etc.**



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr

■ **Query: all customers who live in Chicago**



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr

■ Query: all customers named “John Smith”



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr

■ **Query: all customers who have bought an item for >\$10**



Example: E-Commerce Database

Customer

cid	cname	cadress	ccn

Purchase

cid	pid	price	timedate

Product

pid	pname	ptype	pdescr

■ **Query: all customers who bought item “Iphone 9”**



Database Users

Users are differentiated by the way they expect to interact with system

- **Application programmers** – interact with system through DML calls
- **Naïve users** – invoke one of the permanent application programs that have been written previously
 - Examples, people accessing database over the web, bank tellers, clerical staff
 - Have no clue about database technology
- **Sophisticated users** – form requests in a database query language
 - Or some other tool such as a spreadsheet, stats package, or specialized GUI
 - Knows about data and stats, but not much about DB technology

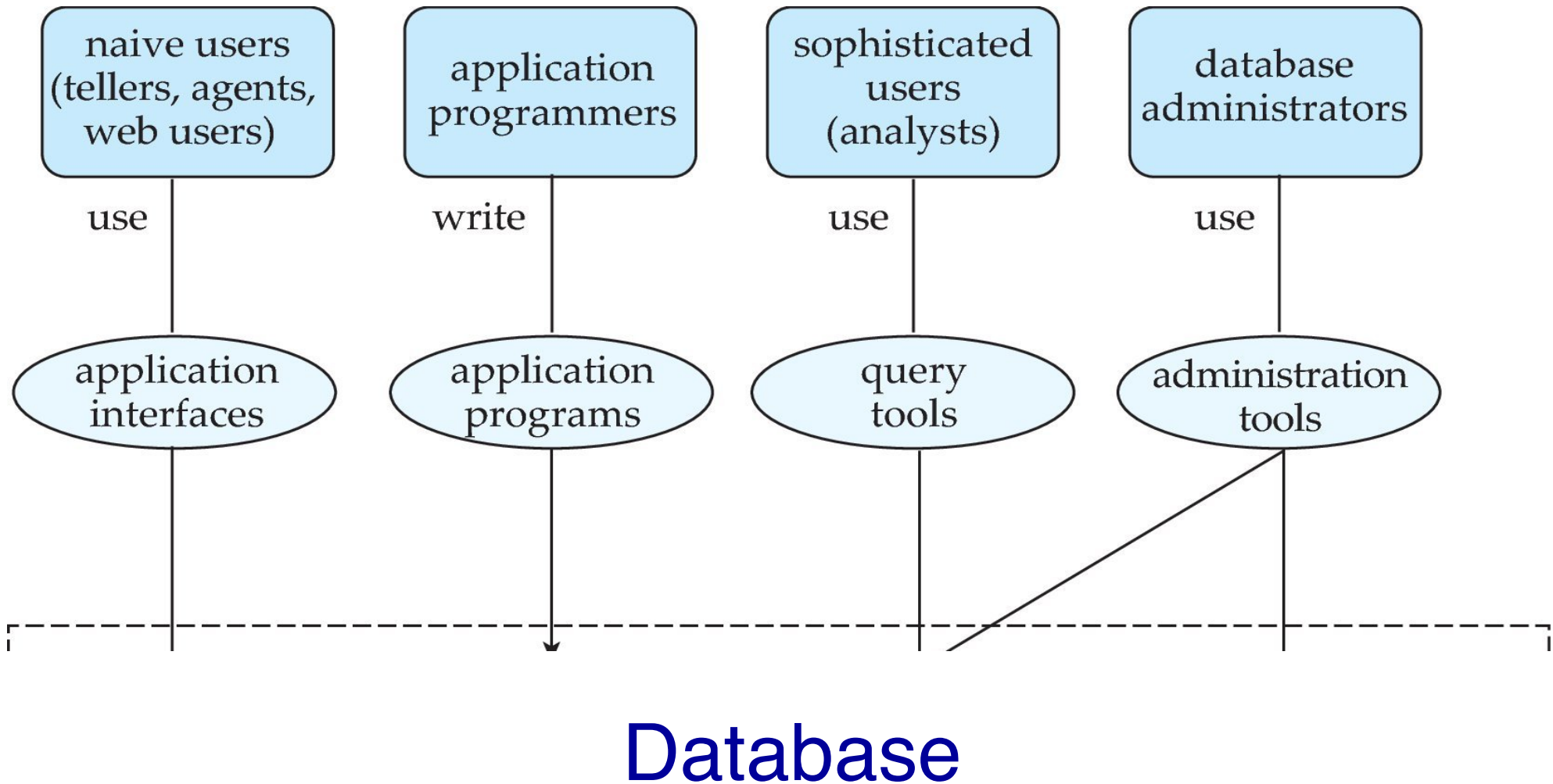


Database Administrator

- **Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise's information resources and needs.**
- **Database administrator's duties include:**
 - **Schema definition**
 - **Storage structure and access method definition**
 - **Schema and physical organization modification**
 - **Granting user authority to access the database**
 - **Specifying integrity constraints**
 - **Acting as liaison with users**
 - **Monitoring performance and responding to changes in requirements**
- **Similar to Unix system administrator, but for the database (DBMS)**



Database Users and Administrators



Database



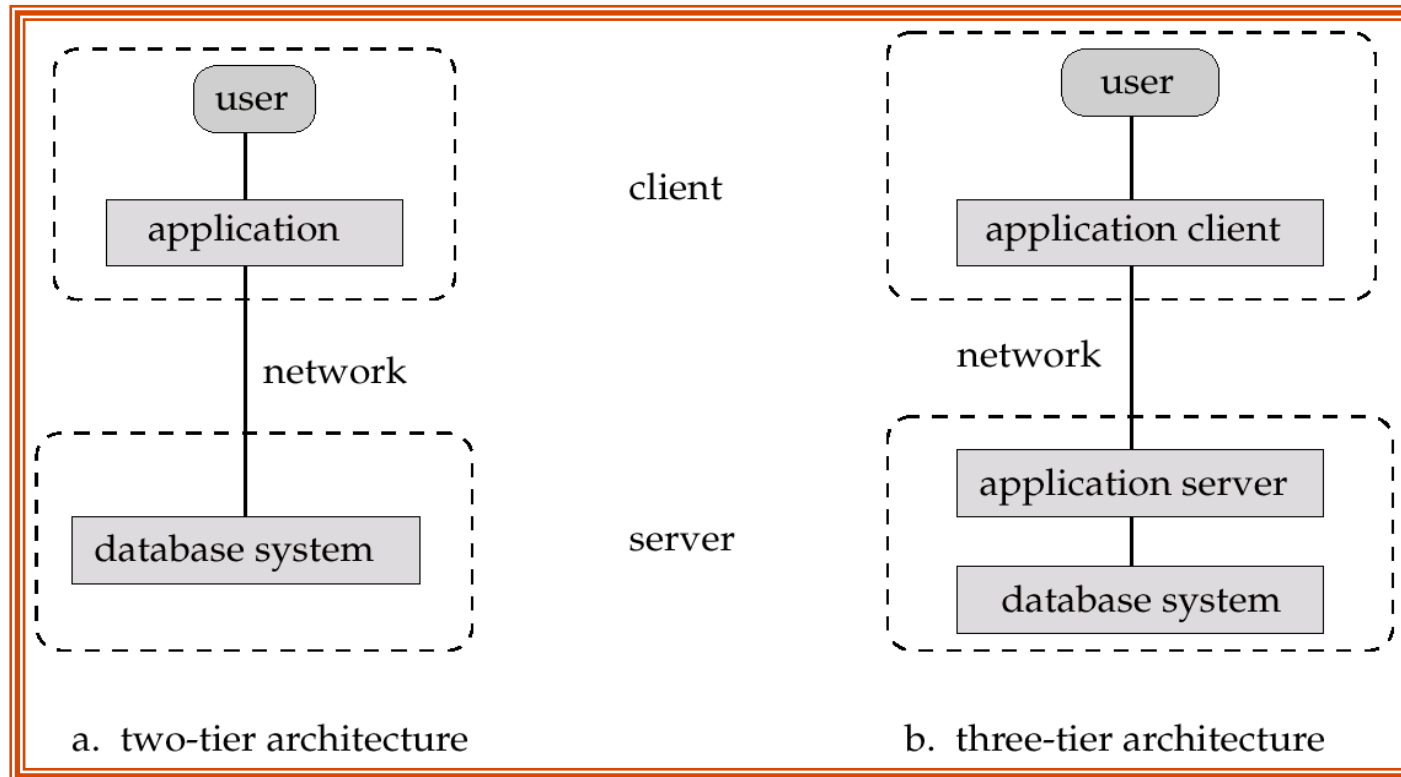
Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- **Centralized**
- **Client-server (two-tier and three-tier)**
- **Parallel (multi-processor)**
- **Distributed**



Client-Server Application Architectures



- **Two-tier architecture:** E.g. client programs using ODBC/JDBC to communicate with a database (or embedded SQL)
- **Three-tier architecture:** E.g. web-based applications, and applications built using “middleware”
- **Web server + CGI vs. application server**
- **CAREFUL:** different definitions of two- vs. three-tier !!!



History of Database Systems

- **1950s and early 1960s:**
 - **Data processing using magnetic tapes for storage**
 - ▶ **Tapes provide only sequential access**
 - **Punched cards for input**
- **Late 1960s and 1970s:**
 - **Hard disks allow direct access to data**
 - **Network and hierarchical data models in widespread use**
 - **Edward Codd defines the relational data model**
 - ▶ **Later wins the ACM Turing Award for this work**
 - ▶ **IBM Research begins work on System R prototype**
 - ▶ **UC Berkeley begins Ingres prototype**
 - **Query optimization technology (IBM System R)**
 - **Transaction processing technology (Jim Gray)**
 - **Recovery mechanisms (C. Mohan, ARIES protocol)**



History (cont.)

- **1980s:**
 - Research relational prototypes lead to commercial systems
 - ▶ SQL becomes industrial standard
 - Parallel and distributed database systems
 - Object-oriented database systems
- **1990s:**
 - Large decision support and data-mining applications
 - Large multi-terabyte data warehouses
 - Emergence of the web and of e-commerce and its impact on databases
- **Since early 2000s:**
 - XML and XQuery standards
 - Automated database administration
 - Column stores and main memory DBs
 - Streaming databases, graph databases
 - Giant data storage systems: BigTable (Google), Hbase (Apache), PNuts (Yahoo!), Dynamo (Amazon), Cassandra (Facebook), Voldemort (LinkedIn)
 - MapReduce (Hadoop), Pig (Yahoo!), Dryad (MSFT), etc.



Some Popular Database Systems

■ Big, expensive, “enterprise”

- Oracle, IBM DB2 Universal Database, Sybase
- Lots of features, tens of thousands of \$
- Multiple platforms, scalable to multiple machines
- Many extension packages for text, images
- Applications: SAP, Peoplesoft, IBM CICS

■ Almost (?):

- MS SQL Server
- Many features, pretty good performance, cheaper
- Windows platform only

■ Not: MS Access

- Easy to use, simple interface
- Limited SQL, limited functionality
- Do not use in this course ...



Some Popular Database Systems

■ Open Source

● **mySQL:**

- ▶ still limited features, but getting better all the time
- ▶ not great for OLAP, and optimizer not that good yet
- ▶ extremely widely used, e.g., web servers (LAMP)

● **postgreSQL:**

- ▶ rich SQL features, decent query optimizer
- ▶ Not as widely used but popular in academia

● **mapReduce/Hadoop, Pig, Pregel, Apache Spark, etc.**

- ▶ systems for data and text analysis (search engines etc)

● **Hbase, Cassandra, Dynamo, CouchDB, MongoDB, etc.**

- ▶ To store large amounts of data for web services
- ▶ No SQL, limited transaction/recovery support, mostly store&fetch

● **Many other specialized systems for warehousing, TP, continuous queries, geographic data, graph data ...**



OLTP, OLAP, and Data Mining

- **Online Transaction Processing:**
 - Many simple queries that update data
 - E.g., bank account transactions
 - Focus on data consistency, data protection, throughput
- **Online Analytical Processing:**
 - Complex query to explore data
 - E.g., find total sales listed by product or region
 - E.g., find customers that often fall behind in payments
 - Focus on complex queries and performance on such queries
 - Sometimes read-only
- **Data Mining:**
 - Find interesting patterns in data
 - Find rules or exceptions to rules
 - “tell me something interesting”
- **Often all of these needed by different people**
- **Use the same system (DBMS) for all three???**



Information Retrieval

“IR is concerned with the representation, storage, organization of, and access to information items”

- **focus on automatic processing (indexing, clustering, search) of unstructured data (text, images, audio, ...)**
- **subfield of Computer Science, but with roots in Library Science, Information Science, and Linguistics**
- **main focus on text data, but also images, audio, video**
- **applications:**
 - **searching in a library catalog**
 - **categorizing a collection of medical articles by area**
 - **web search engines**



Historical Perspective

- **WW2 era computers built for number crunching:
ballistics computations, code breaking**
- **since earliest days, also used to “organize information”**
 - Memex (Vannevar Bush, 1945)
- **today, this is the main application!**
 - store and organize data and documents
 - model organizations and work processes
- **~~Computer~~ Organizer** (also: communications/media)
- **... however, no clean separation**



Structured vs. Unstructured Data

- **IR: lesser known cousin of field of *Databases***
- **Databases: focus on structured data**

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
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

<i>account-number</i>	<i>balance</i>
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

- **IR: unstructured data: “documents”**
 - scientific articles, novels, poems, jokes, web pages, ...
- **Information retrieval vs. data retrieval**
- **IR focused on human user, “fuzzy” results**
- **DBs used for building software on top, precise semantics**



Structured vs. Unstructured Data

<i>customer-id</i>	<i>customer-name</i>	<i>customer-street</i>	<i>customer-city</i>
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

<i>account-number</i>	<i>balance</i>
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

- **Structured data: actually means *simple* structure**
- **Unstructured data means a lot of structure!**
- **... but too complicated to model in structured way**
- **Hence, treat as unstructured**
- **Semi-structured: more complicated than structured data, but we still try to model the structure (?)**



Text IR Applications

- **Digital Libraries and Library Catalog Systems**
 - full-text indexing and search of electronic documents
 - keyword search for other documents
 - categorization of medical or news articles
 - browsing by categories and relationships
- **Web Search Engines**
 - very large amounts of data, data acquisition problem
 - many short documents, very informal, manipulation
- **National Security and Competitive Intelligence:**
 - web surveillance (extremist sites, competitors)
 - filtering and querying intercepted communications
 - searching internal documents (files from investigations)
 - analyzing phone call graphs, server and search engine logs



Related Course Offerings at NYU-Poly

- **CS6913: Web Search Engines**

- **Course Objectives**

- web and search engine architecture *how does it all work?*
- working with massive data sets *storing and analyzing terabytes*
- introduction to Information Retrieval *unstructured data, text*
- system building skills *building distributed systems*
- the Web as a social network *adversarial behavior, spam, communities*

- **Offered in Fall 2018**



Other Related Courses at NYU

- **Machine Learning**
- **Big Data Management & Analysis**
- **Natural Language Processing**
- **Programming for Big Data Analytics**
- **Visual Analytics/Information Visualization**
- **Cloud Computing**

- **Advanced Databases? (rarely offered)**

Big Data is about more than just relational DBs!