Big Data: Introduction to Databases

Juliana Freire

Today

- Why study databases?
- Why use databases?
- Introduction to Relational Databases
- Representing structured data with the Relational Model
- Accessing and querying data using SQL

Why study databases?

- Databases used to be *specialized applications,* now they are a *central component* in computing environments
- Knowledge of database concepts is *essential* for computer scientists and for anyone who needs to *manipulate* data

Why study databases?

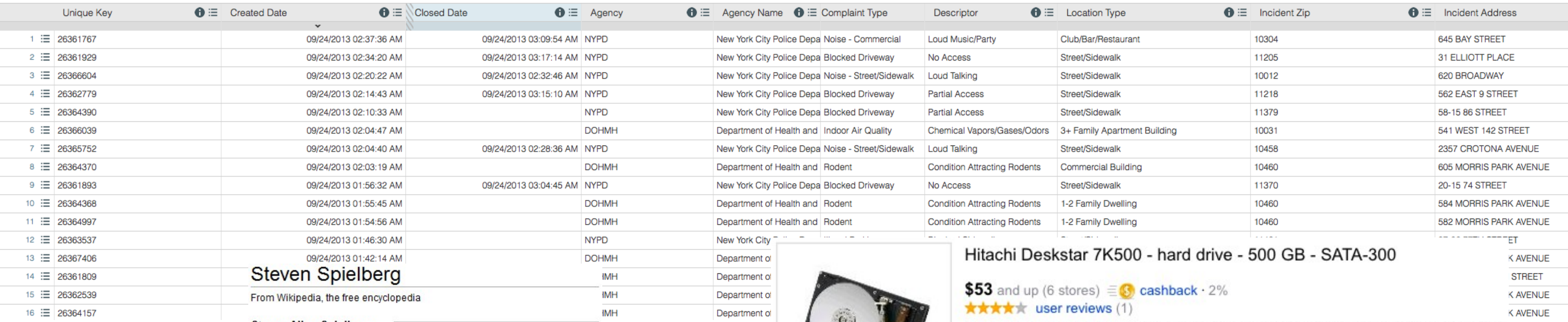
- Databases are everywhere, even when you don't see them: most activities involve data
  - Banking + credit cards: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Telecommunications/networks
  - Sales: customers, products, purchases
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
  - **Web sites**: front end to information stored in databases; e.g., Google, YouTube, Flickr, Amazon…
  - Scientific research, e.g., studying the environment, cities, …
- Data needs to be *managed*

Why study databases?

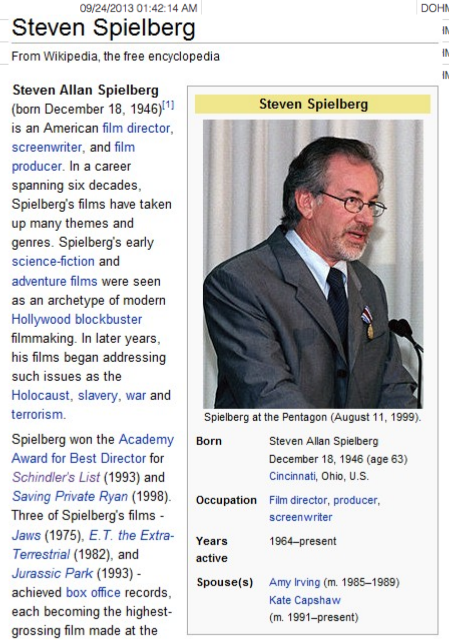
- Data is valuable:
  - E.g., bank account records, tax records, student records, your videos and photos…
  - It must be protected - no matter what happens whether we have machine crashes, disk crashes, hurricanes/floods;
  - It also needs to be protected from **people**

Why study databases?

- Data is often structured



	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address
1	26361767	09/24/2013 02:37:36 AM	09/24/2013 03:09:54 AM	NYPD	New York City Police Depa	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	10304	645 BAY STREET
2	26361929	09/24/2013 02:34:20 AM	09/24/2013 03:17:14 AM	NYPD	New York City Police Depa	Blocked Driveway	No Access	Street/Sidewalk	11205	31 ELLIOTT PLACE
3	26366604	09/24/2013 02:20:22 AM	09/24/2013 02:32:46 AM	NYPD	New York City Police Depa	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	10012	620 BROADWAY
4	26362779	09/24/2013 02:14:43 AM	09/24/2013 03:15:10 AM	NYPD	New York City Police Depa	Blocked Driveway	Partial Access	Street/Sidewalk	11218	562 EAST 9 STREET
5	26364390	09/24/2013 02:10:33 AM		NYPD	New York City Police Depa	Blocked Driveway	Partial Access	Street/Sidewalk	11379	58-15 86 STREET
6	26366039	09/24/2013 02:04:47 AM		DOHMH	Department of Health and	Indoor Air Quality	Chemical Vapors/Gases/Odors	3+ Family Apartment Building	10031	541 WEST 142 STREET
7	26365752	09/24/2013 02:04:40 AM	09/24/2013 02:28:36 AM	NYPD	New York City Police Depa	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	10458	2357 CROTONA AVENUE
8	26364370	09/24/2013 02:03:19 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	Commercial Building	10460	605 MORRIS PARK AVENUE
9	26361893	09/24/2013 01:56:32 AM	09/24/2013 03:04:45 AM	NYPD	New York City Police Depa	Blocked Driveway	No Access	Street/Sidewalk	11370	20-15 74 STREET
10	26364368	09/24/2013 01:55:45 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	1-2 Family Dwelling	10460	584 MORRIS PARK AVENUE
11	26364997	09/24/2013 01:54:56 AM		DOHMH	Department of Health and	Rodent	Condition Attracting Rodents	1-2 Family Dwelling	10460	582 MORRIS PARK AVENUE
12	26363537	09/24/2013 01:46:30 AM		NYPD	New York City					ET
13	26367406	09/24/2013 01:42:14 AM		DOHMH	Department o					K AVENUE
14	26361809			IMH	Department o					STREET
15	26362539			IMH	Department o					K AVENUE
16	26364157			IMH	Department o					K AVENUE



Steven Spielberg

From Wikipedia, the free encyclopedia

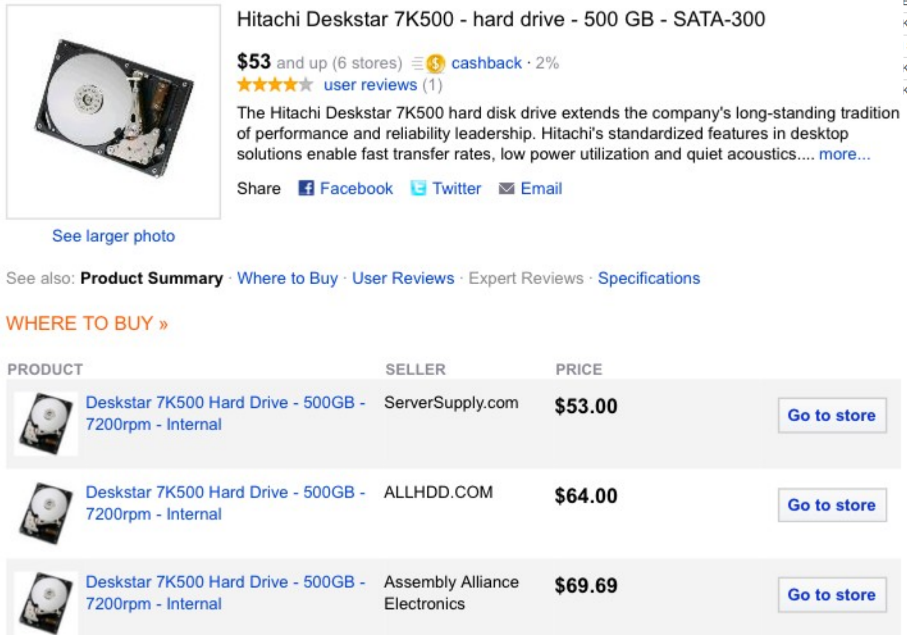
Steven Allan Spielberg (born December 18, 1946)[1] is an American film director, screenwriter, and film producer. In a career spanning six decades, Spielberg's films have taken up many themes and genres. Spielberg's early science-fiction and adventure films were seen as an archetype of modern Hollywood blockbuster filmmaking. In later years, his films began addressing such issues as the Holocaust, slavery, war and terrorism.

Spielberg won the Academy Award for Best Director for *Schindler's List* (1993) and *Saving Private Ryan* (1998). Three of Spielberg's films - *Jaws* (1975), *E.T. the Extra-Terrestrial* (1982), and *Jurassic Park* (1993) - achieved box office records, each becoming the highest-grossing film made at the

Steven Spielberg

Spielberg at the Pentagon (August 11, 1999).

Born	Steven Allan Spielberg December 18, 1946 (age 63) Cincinnati, Ohio, U.S.
Occupation	Film director, producer, screenwriter
Years active	1964–present
Spouse(s)	Amy Irving (m. 1985–1989) Kate Capshaw (m. 1991–present)



Hitachi Deskstar 7K500 - hard drive - 500 GB - SATA-300

$53 and up (6 stores) cashback · 2%

★★★★☆ user reviews (1)

The Hitachi Deskstar 7K500 hard disk drive extends the company's long-standing tradition of performance and reliability leadership. Hitachi's standardized features in desktop solutions enable fast transfer rates, low power utilization and quiet acoustics.... more...

Share Facebook Twitter Email

See larger photo

See also: **Product Summary** · Where to Buy · User Reviews · Expert Reviews · Specifications

WHERE TO BUY »

PRODUCT	SELLER	PRICE	
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	ServerSupply.com	$53.00	Go to store
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	ALLHDD.COM	$64.00	Go to store
Deskstar 7K500 Hard Drive - 500GB - 7200rpm - Internal	Assembly Alliance Electronics	$69.69	Go to store

Why study databases?

- Data is often structured
- We can exploit this regular structure
  - To retrieve data in useful ways (that is, we can use a *query* language)
  - To store data efficiently

Why study Databases?

- Because the database field has made a number of contributions to basic computer science
  - Databases are behind many of important contributions and impact that CS has had
  - Find, gather, analyze and understand data, e.g., Banks, human genome, ecommerce, Web:
- *Understand concepts and apply to different problems and different areas, e.g., Big Data*
- Because DBMS software is highly successful as a commercial technology (Oracle, DB2, MS SQL Server…)
- Because DB research is highly active and **very** interesting!
  - Lots of opportunities to have practical impact

Database Systems: The Basics

A Simple Data Management Problem: Address Book

- Solution
  - Create a text file
- Advantages
  - Easy to add and modify
  - Easily copied (e.g., for backup, or paper dump)
  - Shareable (as a unit)
  - Substring searchable
  - Powerful, programmable tools
- But there can be complications…

Complication 1: File Gets **Very Large**

- Problem:
  - Searching gets slow and imprecise
  - Search for “Elm Street” yields “Wilhelm Streeter”
- Solution
  - Add indexes over fields commonly searched upon
  - Structure data into fields
    - Search for street=“Elm Street”

Database Concepts:

- ***Record organization***
- ***Indexes***

Complication 2: Data Redundancy

- Why?
  - Large families, frequent moves
  - Might forget to update addresses of some family members
  - Want space economy, single point of update
  - Importance of residence as separate entity: 1 Xmas card each
- Solution:
  - Separate residences from names: 2 files, one for persons, one for residence
  - But how do we associate a residence with a person?
  - How many residences can a person have? 0? 1? Several?

Database Concepts:

- ***Consistency***
- ***Normalization***
- ***Foreign keys***

Complication 3: Multiple Associations Of Persons and Residences

- Meaning:
  - People can own, rent, manage, visit residences
  - May want constraints on numbers of residences per person
- Examples:
  - many-one (single family), many-many (rich folks), one-many (builder)

Database Concepts:

- ***Relationships***
- ***Cardinality constraints***
- ***Consistency***

Complication 4: Need To Add Information For New Purposes

- Examples:
  - Xmas cards sent and received
  - Post office gives big discount for using Zip+4 addressing
- Requirements:
  - Adding fields and/or new tables

Database Concept:

•***Schema evolution***

Complication 5: Doing Ad Hoc Analysis and Retrieval

- Example:
  - "Who have we sent cards to each of the past 5 years, but received 2 or fewer cards in return?"
- Requires:
  - Language for expressing analysis and retrieval
  - Implementation that performs analysis and retrieval correctly and efficiently

Database Concepts:

- ***Query languages***
- ***Query optimization and execution***

Complication 6: Want To Organize The Data Differently For Some Users

- Examples:
  - Other family members want to see names and residences together
  - You don’t want your kids to see your business entries
- Solution:
  - Use stored queries as “windows” onto the database
  - Data not selected by query is “not there”

Database Concepts:

- ***Joins***
- ***Views***
- ***Security***

Complication 7: Required Existence Of Associated Data

Database Concept:

- ***Referential integrity***

- Examples:
  - Can’t send a Xmas card to someone without an address
  - Names are not unique unless qualified by residence: the John Jones living at 123 Elm Street
- Solutions:
  - Refuse to insert a name unless it is associated with an address
  - Refuse to delete an address if it is associated with a name
  - Or, tolerate multiple non-unique names

Complication 8: Want Programmed Access To Data

- Meaning:
  - Want to write a Java program to search, display, update entries
- Solution:
  - Use data organization to define corresponding datatypes
  - Use access library to open, retrieve, update data

Database Concepts:

- ***Database schemas***
- ***API***
- ***Embedded querying***

Complication 9: Multiple Updates On All Or None Basis

- Examples:
  - Two households merge
  - Requires changing residences of several persons
  - What if your computer crashes during updates?
- Solution:
  - Present illusion that all updates are done simultaneously
  - Implemented by commit or rollback of entire piece of work

Database Concept:

• ***Transactions***

• ***Atomicity***

Complication 10: Your Computer Crashes (Again)

- Will your data still be present
  - Uncorrupted?
  - In what state, given that a transaction was in progress?
- Solution:
  - Make sure old data are safely accessible until latest commit

Database Concept:

- ***Data durability***
- ***Recovery***

Complication 11: Two Computers In Your Household

- How can data be shared?
  - USB key? Ugh, multiple version headaches
  - Dropbox – changes can be overwritten
  - Let's assume the database is shared somehow
  - What if one user is merging households, another is splitting one up?
  - What are meaningful results?
- A common policy:
  - Transactions are atomic
  - They appear to run one after the other, in some order

Database Concepts:

- ***Transaction isolation***
- ***Concurrency control***
- ***Transaction serializability***

Complication 12: A Home Computer And A Business Computer

- Is there one database or two?
  - Want speed, reliability of local data at each site
  - But logically, one database for maintenance and querying
  - Data communication between them (most of the time …)
  - Want some capability for independent operation (robustness)
- Solutions:
  - Personal data on the home computer
  - Business data on the business computer
  - Common logical view

Database Concepts:

- ***Distributed databases***
- ***Data partitioning***
- ***Data replication***

Complication 13: Your Uncle Louie Gets The Genealogy Bug

- His grand vision:
  - All family members pool their databases over the Internet
  - Together, all genealogy relationships can be recorded
- But:
  - Aunt Sarah is paranoid: will not reveal birthdates
  - You are too: you don't want your business associates in the genealogy database
  - Everyone wants complete control over safety of their own data
  - People use different formats for records, and different name abbreviations for entries

Database Concepts:

• *Federated databases*

•*Data integration*

Complication 14: You Become President

- Of USA, of University, of a large organization
  - Your address list grows to hundreds of thousands or more
  - You realize it contains useful information *in the large*
- Examples
  - Which are top 10 zip codes on the list?
  - Which zip codes have addresses that are most likely to send cards to you when you send cards to them?
  - Which of those zip codes are in states that had less than 5% difference in Republican / Democratic presidential votes in 2004?

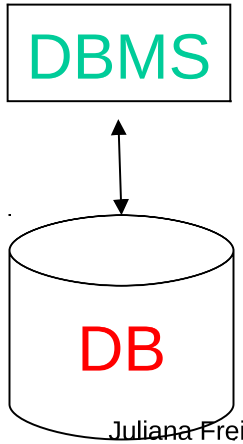
Database Concepts:

• ***Data mining***

•***Online analytical processing***

Databases and Database Management Systems

- Database (DB) is an integrated collection of data
  - Models real-world objects
    - Entities (e.g., people, residence, Christmas cards)
    - Relationships (e.g., John Doe lives on 123 Elm St)
  - Captures *structure* – allows data to be queried
- A Database Management System (DBMS) is a software suite designed to store and manage databases
  - Provides environment that is both *convenient* and *efficient* to use.
  - Address all *complications* discussed



Storing Data: Database vs File System

- Once upon a time database applications were built on top of file systems…
- But this has many drawbacks:
  - Data redundancy, inconsistency and isolation
    - 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, e.g., search people by zip code or last name; update telephone number
  - Integrity problems
    - Integrity constraints (e.g., num_residence = 1) become part of program code -- hard to add new constraints or change existing ones
  - Failures may leave database in an inconsistent state with partial updates carried out, e.g., John and Mary get married, add new residence, update John's entry, and database crashes while Mary's entry is being updated…

Storing Data: Database vs File System (cont.)

- Concurrent access by multiple users
  - Needed for performance: can you imagine if only 1 person at a time could buy a ticket from Delta?
  - Uncontrolled concurrent access can lead to inconsistencies, e.g., the same seat could be sold multiple times…
    - There are 3 seats left; I buy 2 seats; John buys 3 seats at the same time
    - If I hit enter 1st there will be 1 seat left; if John is faster there will be 0; but 5 seats have been sold and we will fight at the airport!

Database systems offer solutions to all the above problems

Why use Database Systems?

- Data independence and efficient access
  - Easy + efficient access through declarative query languages and optimization
- Data integrity and security
  - Preventing inconsistencies, safeguarding data from failures and malicious access
- Concurrent access
- Reduced application development time
- Uniform data administration

When not to use Database Systems?

What's in a DBMS?

	Data model	*Query language*	*Transactions and crash recovery*
"above the water"	Logical DB design Relational Model XML data model	SQL, QBE, views XPath, XQuery	Transactions
"below the water"	Map data to files Clustering Indexes	Query optimization Query evaluation	Locking Concurrency control Recovery Logs

Designing a database: The Conceptual Model

- What are the *entities* and *relationships* among these entities in the application?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hold?
- Different applications have different needs, and different perspectives – even to model the *same* object

billing department: patient(id, name, insurance, address)
visit(patientId, procedure, date, charge)

inpatient: patient(id,name,age,address)
alergies(id,alergies)
prescription(patientId,date,medicine)

Designing a database: The Conceptual Design

- What are the *entities* and *relationships* among these entities in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hol
- Dif fferent perspectives – even to model the *same* object

Requires a good understanding of the *semantics* of the application

billing department: patient(id, name, insurance, address)
visit(patientId, procedure, date, charge)

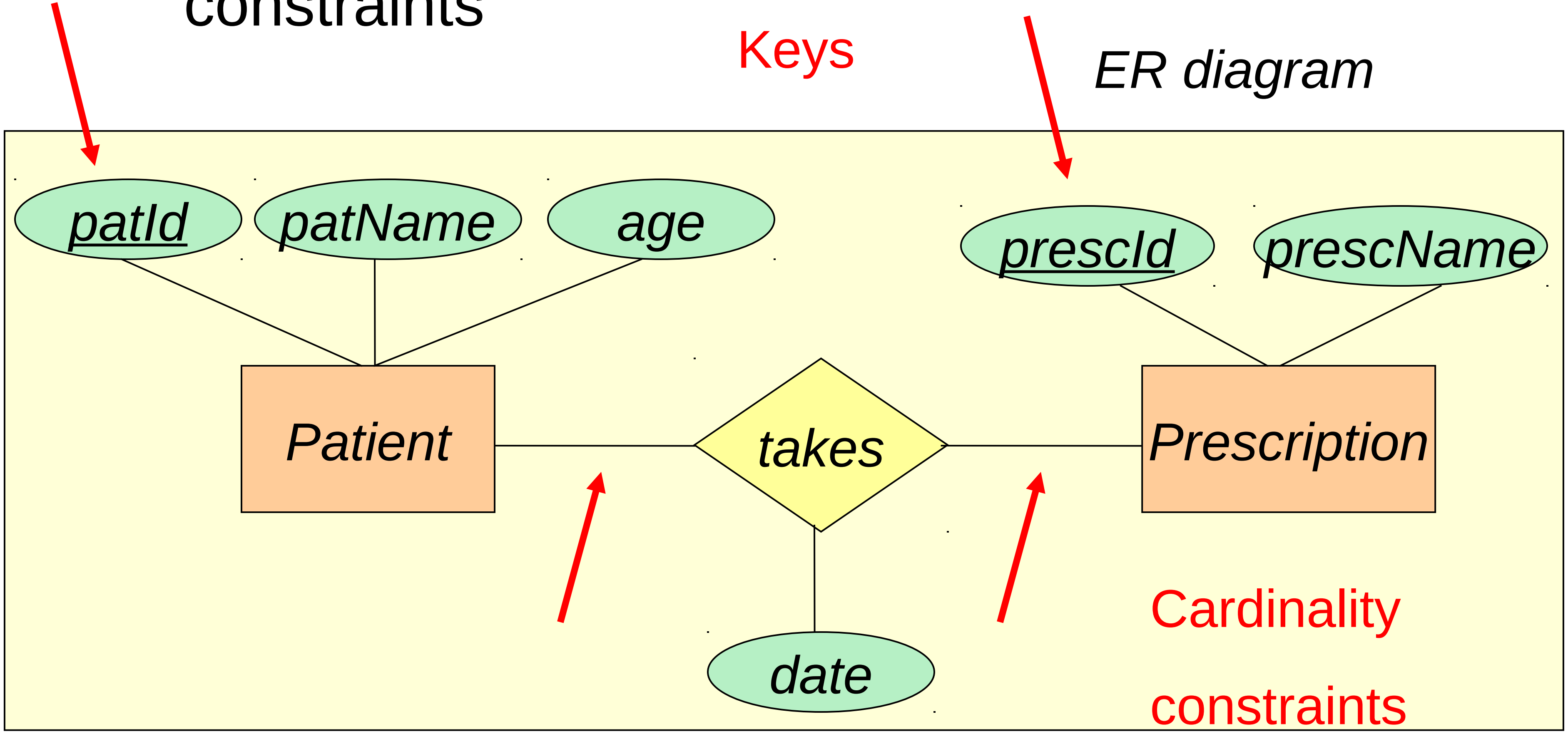
inpatient: patient(id,name,age,address)

alergies(id,alergies)

prescription(patientId,date,medicine)

The Entity Relationship (ER) Data Model

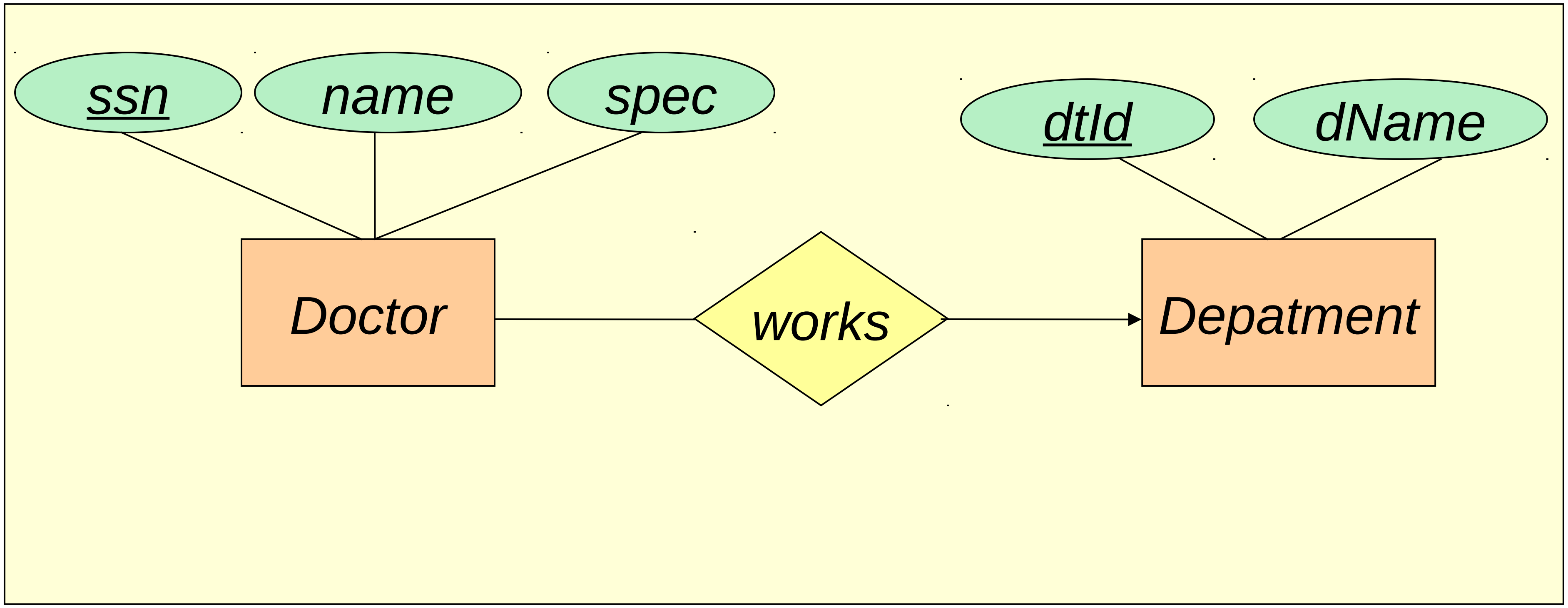
- A *data model* is a collection of concepts for describing data, relationships, semantics and constraints



ER: Another Example

- A department has many doctors, but a doctor can only work in one department

ER diagram



Relational Data Model

- ER used for conceptual design is then mapped into the relational model
- The *relational model of data* is the most widely used model today
  - Main concept: *relation*, basically a table with rows and columns
  - Every relation has a *schema*, which describes the columns, or fields
- A *schema* is a description of a particular collection of data, using a given data model

Patient(patientId:int, patientName:str, age: int)

Takes(patientId:int,prescId:inte,prescDate:date)

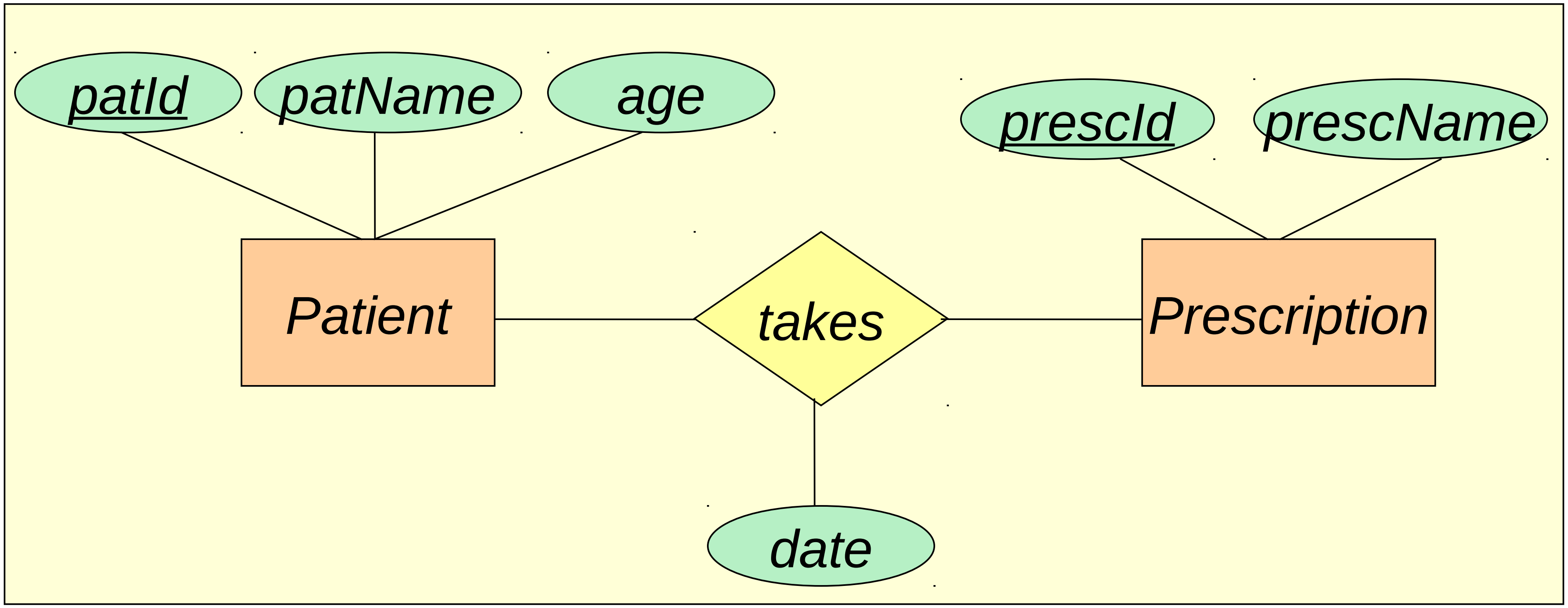
Prescription(prescId:int, presName:str)

ER to Relational

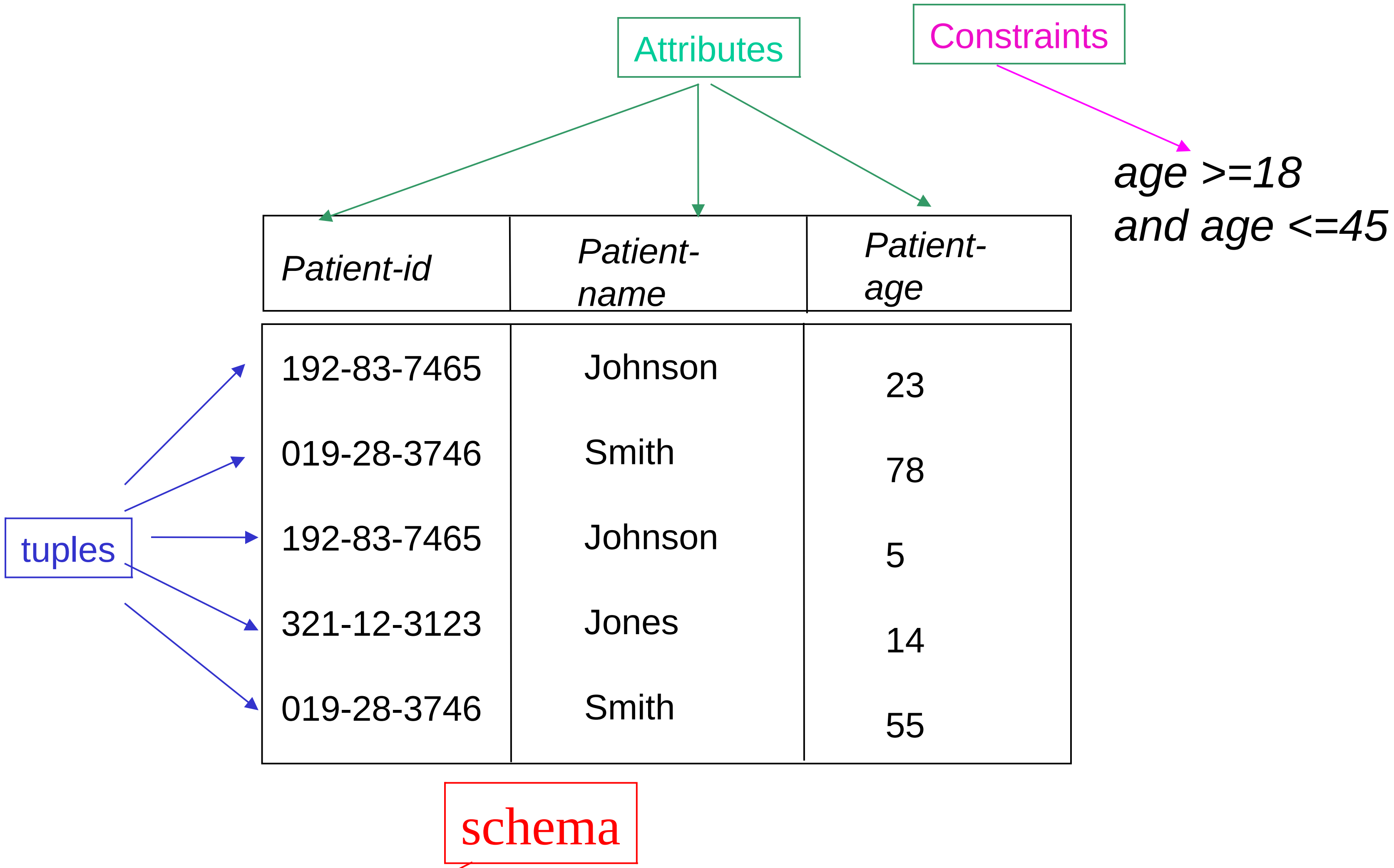
Patient($\underline{\text{patientId:int}}$, patientName:str, age: int)

Takes($\underline{\text{patientId:int,prescId:int}}$,prescDate:date)

Prescription($\underline{\text{prescId:int}}$, presName:str)



Relational Model: Terminology



Patient-id	*Patient-name*	*Patient-age*
192-83-7465	Johnson	23
019-28-3746	Smith	78
192-83-7465	Johnson	5
321-12-3123	Jones	14
019-28-3746	Smith	55

Patient(<u>patientId:int</u>, patientName:str, age: int)

Pitfalls in Relational Database Design

- Find a "good" collection of relation schemas
- Bad design may lead to
  - Repetition of information → inconsistencies!
    - E.g., keeping people and addresses in a single file
  - Inability to represent certain information
    - E.g., a doctor that is both a cardiologist and a pediatrician
- Design Goals:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Ensure constraints are properly modeled: updates check for violation of database integrity constraints

Query Languages

- *Query languages*: Allow *manipulation* and *retrieval* of data from a database
- Queries are posed wrt data model
  - Operations over objects defined in data model
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for optimization
- Query Languages **!=** programming languages
  - QLs support easy, efficient access to large data sets
  - QLs not expected to be "Turing complete"
  - QLs not intended to be used for complex calculations

Query Languages

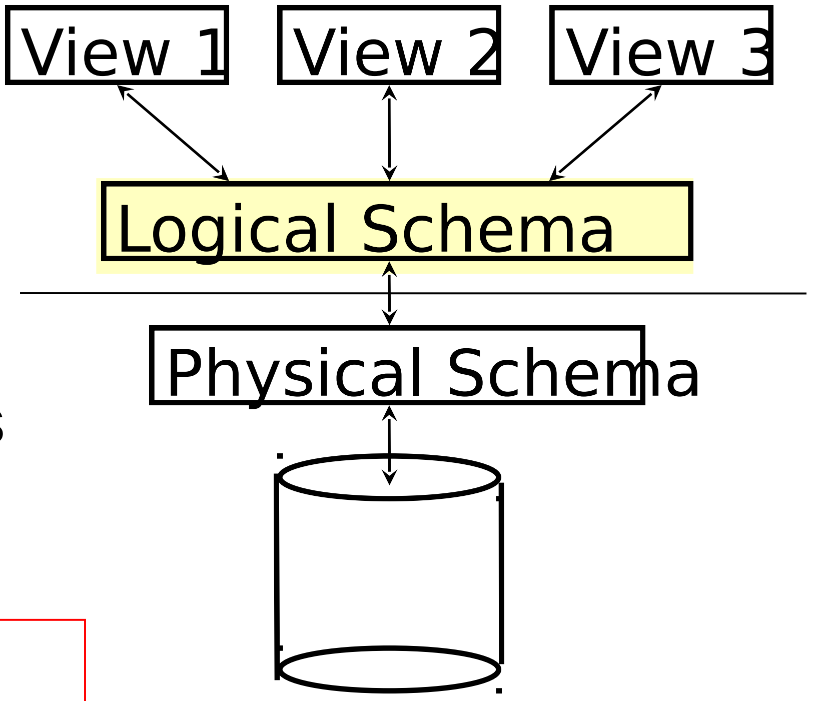
- *Query languages:* Allow *manipulation* and *retrieval* of data from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on l
  - Allows for much optimization.
- Query Languages **!=** programmin
  - QLs not expected to be "Turing complete".
  - QLs support easy, efficient access to large data sets.
  - QLs not intended to be used for complex calculations.

a language that can compute anything that can be computed

Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*
  - Views describe how users see the data

  - Logical schema defines logical structure
  - Physical schema describes the files and indexes used



Key to good performance

Example: University Database

- Physical schema:
  - Students stored in id order
  - Index on last name
- Logical schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence

- Applications insulated from how data is structured and stored
- *Logical data independence*: Protection from changes in *logical* structure of data
  - Changes in the logical schema do not affect users as long as their views are still available
- *Physical data independence:* Protection from changes in *physical* structure of data
  - Changes in the physical layout of the data or in the indexes used do not affect the logical relations

One of the most important benefits of using a DBMS!

"above the water"

"below the water"

	Data model	*Query language*	*Transactions and crash recovery*
"above the water"	Logical DB design Relational Model XML data model	SQL, QBE, views XPath, XQuery	Transactions
"below the water"	Map data to files Clustering Indexes	Query optimization Query evaluation	Locking Concurrency control Recovery Logs

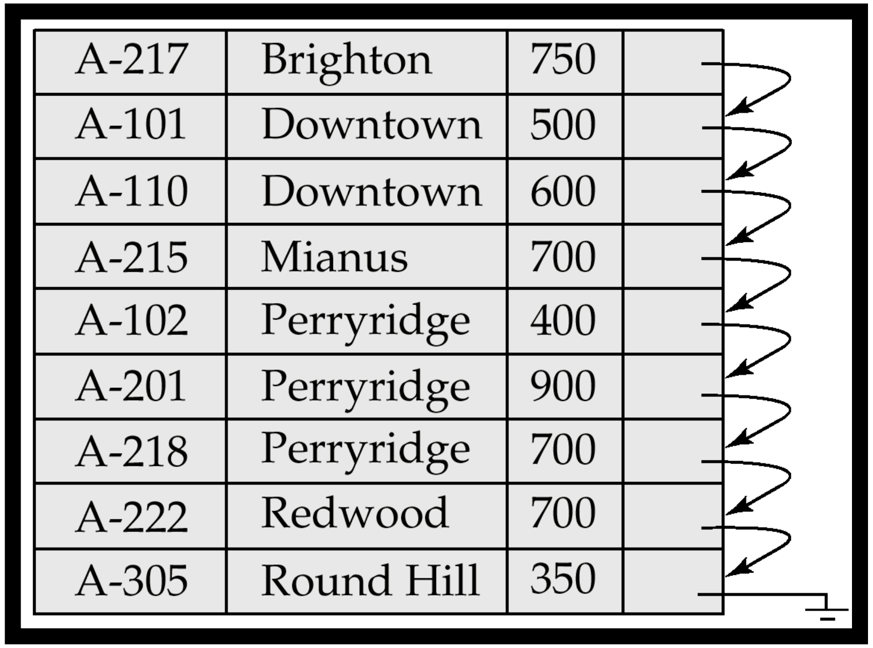
Let's dive now…

Storage and Indexing

- The *DB administrator* designs the physical structures
- Nowadays, database systems can do (some of) this automatically: autoadmin, index advisors
- File structures: sequential, hashing, clustering, single or multiple disks, etc.
- Example – Bank accounts
  - Good for:

    List all accounts in the Downtown branch
  - What about:

    List all accounts with balance = 350



A-217	Brighton	750
A-101	Downtown	500
A-110	Downtown	600
A-215	Mianus	700
A-102	Perryridge	400
A-201	Perryridge	900
A-218	Perryridge	700
A-222	Redwood	700
A-305	Round Hill	350

Storage and Indexing

- Indexes:
  - Select attributes to index
  - Select the type of index
- Storage manager is a module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

Query Optimization and Evaluation

- DBMS must provide efficient access to data
  - In an emergency, can't wait 10 minutes to find patient allergies
- Declarative queries are translated into imperative query plans
  - Declarative queries → logical data model
  - Imperative plans → physical structure
- Relational optimizers aim to find the best imperative plans (i.e., shortest execution time)
  - In practice they avoid the worst plans…

Example: Query Optimization

select number

from accounts

where balance = 350

$$\begin{tikzcd}
\Pi_{\text{number}} \arrow[d, dash] \\
\sigma_{\text{balance=350}} \text{, use index(balance)} \arrow[d, dash] \\
\text{accounts}
\end{tikzcd}$$

Transaction: An Execution of a DB Program

- Key concept is *transaction*, which is an *atomic* sequence of database actions (reads/writes)
- Each transaction, executed completely, must leave the DB in a *consistent state* if DB is consistent when the transaction begins
  - Ensuring that a transaction (run alone) preserves consistency is ultimately the programmer's responsibility!
- 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
  - DBMS ensures *atomicity* (all-or-nothing property)

Concurrency Control

- Concurrent execution of user programs is essential for good DBMS performance
- But interleaving actions of different user programs can lead to inconsistency
  - e.g., nurse and doctor can simultaneously edit a patient record
- DBMS ensures such problems don't arise: users can pretend they are using a single-user system

Ensuring Atomicity

- DBMS ensures *atomicity* (all-or-nothing property) even if system crashes in the middle of a transaction
  - If there is power outage, will the patient database become inconsistent?
- Idea: Keep a *log* (history) of all actions carried out by the DBMS while executing a set of transactions
  - Before a change is made to the database, the corresponding log entry is forced to a safe location.
  - After a crash, the effects of partially executed transactions are *undone* using the log; and if log entry wasn't saved before the crash, corresponding change was not applied to database!

Databases make these folks happy ...

- End users
- DBMS vendors: $20B+ industry
- DB application programmers
- Database administrator (DBA)
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

Summary

- DBMS used to maintain and query (*large*) structured datasets
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security
- Levels of abstraction give data independence