Organization

CS460 Databases for Data Scientists

Hello

- Dr. Sven Hertling
  - Substitute Professor for Data Science
- Research Interests:
  - Knowledge Graph Integration
  - KGs in combination with Large Language Models
  - Information Extraction
- Room: B6 26, B0.21
- eMail: sven.hertling@uni-mannheim.de
- Sven will teach the lectures

Hello

- M.Sc. Franz Krause
  - Graduate Research Associate
- Research Interests:
  - Machine Learning Applications on Linked Data
  - Dynamization of Knowledge Graph Embeddings
  - Knowledge Graph Application and Implementation in Industrial Settings
  - Applied Graph Theory
- Room: B6 26, B 0.02
- eMail: franz.krause@uni-mannheim.de
- Franz will teach the exercise

Course Organization

- Lecture
  - Database concepts
  - Theory of relational algebra, relational modeling, query processing
  - Introduction to SQL
- Exercise
  - Creating example databases
  - Hands-on experience
- Final exam

Course Contents and Schedule

Date	Lecture (Tuesday)	Exercise (Wednesday)
10.02.	no lecture	no exercise
17.02.	Introduction	Introduction
24.02.	SQL Part 1	SQL Part 1
03.03.	SQL Part 2	SQL Part 2
10.03.	ER Models	ER Models
17.03.	Normal Forms	Normal Forms
24.03.	Index and Hashing / Complexity Theory	Index and Hashing + Intro Easter Eggcercise
31.03.	*Holiday*	*Holiday*
07.04.	*Holiday*	*Holiday*
14.04.	DB Architectures	DB Architectures
21.04.	Query Processing	Query Processing
28.04.	Query Optimization	Query Optimization
05.05.	Transactions and Concurrency	Transactions and Concurrency
12.05.	Recovery	Recovery
19.05.	Application Development	Application Development
26.05.	NoSQL + Q&A	NoSQL + Q&A

You are here

you'll get a larger eggxercise assignment here

Day refers to the first day of the week

Course Organization

- Lecture Webpage: Slides, Announcements, Web Links
  - https://www.uni-mannheim.de/dws/teaching/course-details/courses-for-master-candidates/cs-460-databases-for-data-scientists/
  - hint: look at version tags!

- Time and Location
  - Lecture:
    - Tuesday, 13.45 – 15.15
    - B6 A203
  - Exercise:
    - Wednesday, 12.00 – 13.30
    - D 007 Room 2

Exercises are here

You are here

Course Organization

- Additional Material
  - ILIAS eLearning System, https://ilias.uni-mannheim.de/

- Remote teaching
  - For those who can't be there, we provide video recordings in ILIAS (from 2021, but they are widely identical)

- This course (and the majority of the slides) are based on the book
  - Silberschatz et al.: Database System Concepts

- Several copies are available in the library

- Additional material online
  - www.db-book.com
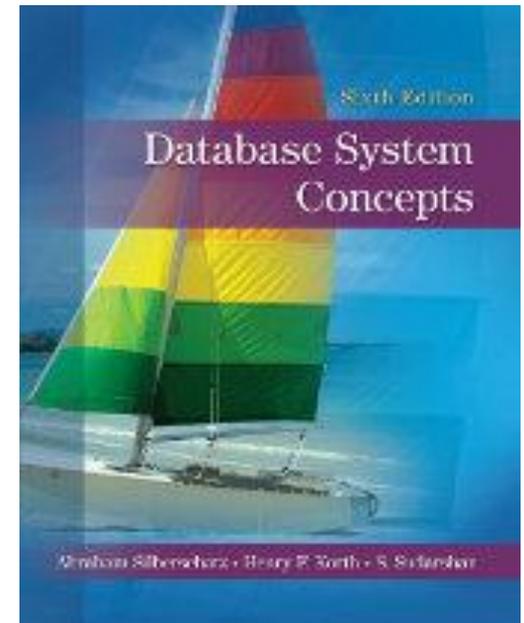
Questions?

Introduction

CS460 Database Technology

Outline

- The Need for Databases

- Data Models

- Database Engine

  - Storage Manager

  - Query Processing

  - Transaction Manager

- Introduction to the Relational Model

Data Base Management Systems (DBMS)

- DBMS contains information about a particular organization
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both convenient and efficient to use

- Database Applications:
  - Banking: 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

- Databases can be very large

- Databases touch all aspects of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate time tables
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

- In the early days, database applications were built directly on top of file systems

Application

Data Files
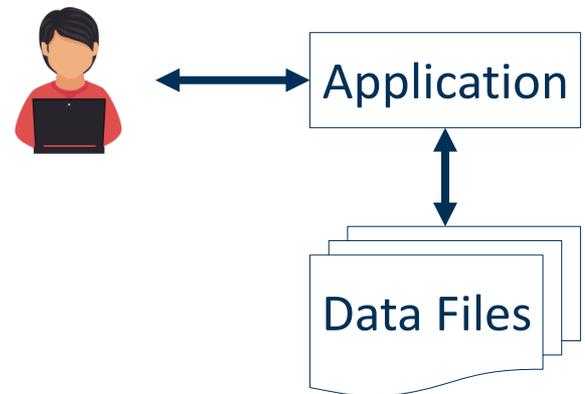
Drawbacks of Using File Systems to Store Data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
  - Example:
    - File system: one file per lecture
    - Change of E-Mail address
      - Needs to be changed in all the files
      - If we forget one, the data becomes inconsistent
    - Problem: E-Mail is stored *redundantly*
      - i.e., once per lecture

Lecture: Databases for Data Scientists
Instructor: Sven Hertling
E-Mail: sven.hertling@uni-mannheim.de
                    ...

Drawbacks of Using File Systems to Store Data

- Integrity problems
  - Integrity constraints (e.g., GPA > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones
  - Example:
    - ZIP code has to be a five digit number
    - As an application developer, would you prefer
      1. Adding a single check in each part of the application where a ZIP code is entered
         - » student applications
         - » contracts with employees
         - » travel reimbursement
         - » …
      2. Adding the check at a single point (i.e., before the data is written into the database)

Drawbacks of Using File Systems to Store Data

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example:
    a. Transfer of funds from one account to another should either complete or not happen at all
    b. retiring a lecturer (pseudo code)

```
Delete from file: active lecturers

Add to file: retired lecturers
```

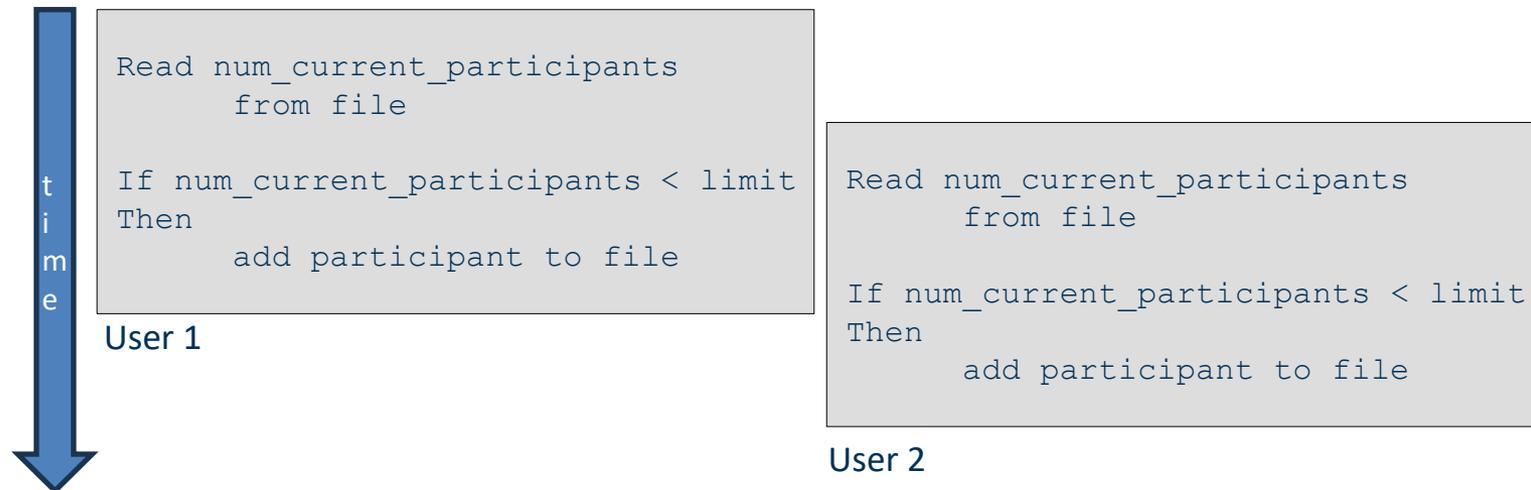Computer crashes here

File:
active lecturers

Prof. Smith
Dr. Stevens
Prof. Miller

File:
retired lecturers

Dr. Hawkins
Prof. Brown
Prof. Wilson

Drawbacks of Using File Systems to Store Data

- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
  - Example:
    a. Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
    b. register for a course if there are places left

```
Read num_current_participants
     from file

If num_current_participants < limit
Then
     add participant to file
```
User 1

```
Read num_current_participants
     from file

If num_current_participants < limit
Then
     add participant to file
```
User 2

time

Drawbacks of Using File Systems to Store Data

- Difficulty in accessing data
  - Need to write a new program to carry out each new task

- Data isolation
  - Multiple files and formats

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

Idea of
Database Management Systems (DBMS)

- Introduce a level of abstraction

- Handle issues of…
    - consistency
    - integrity
    - transaction atomicity
    - concurrency
    - security
    - …

- … in a centralized fashion

Application

DBMS

Data Files
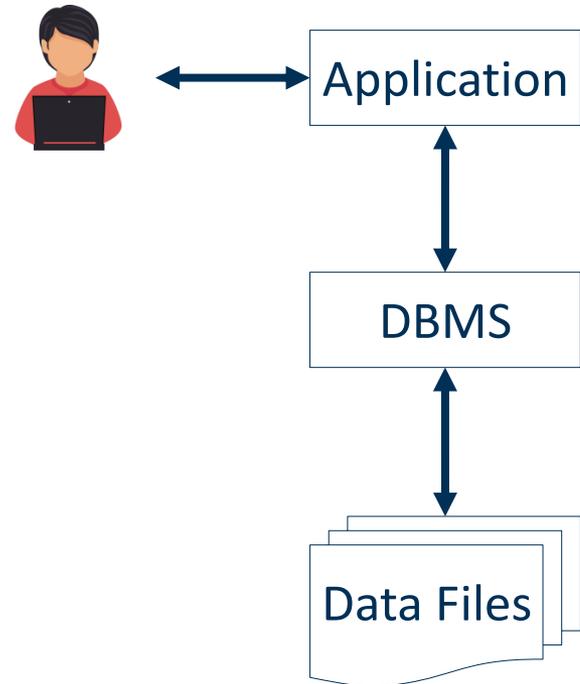
Levels of Abstraction

- **View level:** application programs hide details of data types
  - Views can also hide information (such as an employee's salary) for security purposes

- **Logical level:** describes data stored in database, and the relationships among the data

  type *instructor* = **record**

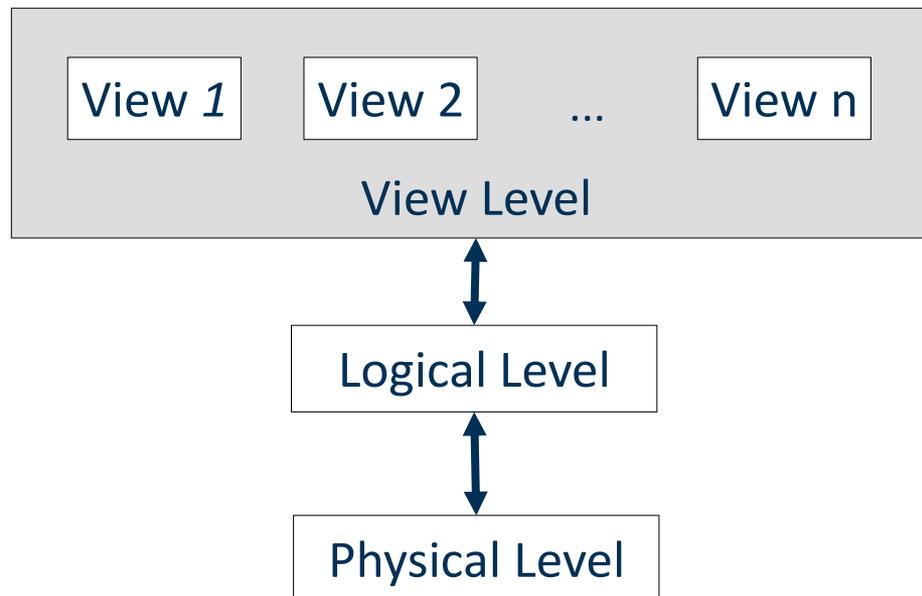 > *ID : string;*
  > *name : string;*
  > *dept_name : string;*
  > *salary : integer;*

  end;

- **Physical level:** describes how a record (e.g., instructor) is stored

Levels of Abstraction

- Architecture of a Database Management System
  - Applications interact with different views

- Decoupling
  - Logical & physical level may be changed without changing the application
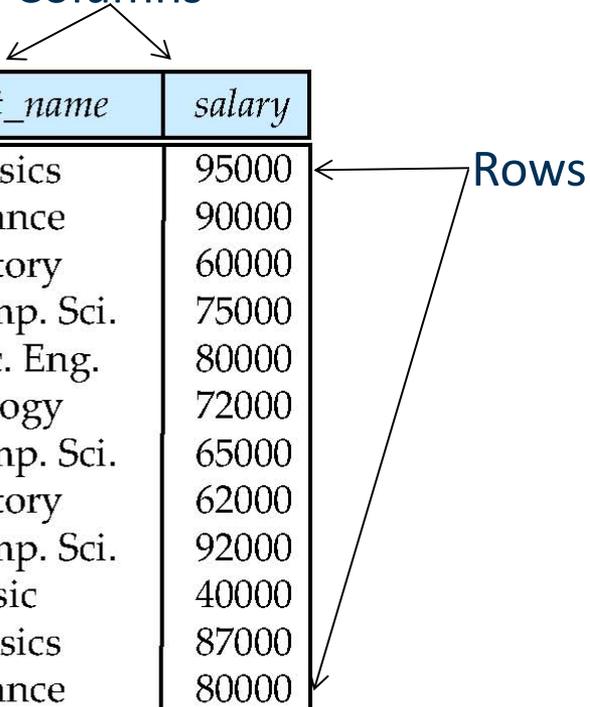
Data Models

- 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 (XML, JSON)

- Other older models:
  - Network model
  - Hierarchical model

The Relational Model

- All data is stored in *tables*

Columns

ID	name	dept_name	salary
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 *database* consists of multiple tables

- A *database* consists of multiple tables

ID	name	dept_name	salary
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

dept_name	building	budget
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

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **CREATE TABLE** *instructor* (
ID	**CHAR**(5),
name	**VARCHAR**(20)**,**
dept_name	**VARCHAR**(20),
salary	**NUMERIC**(8,2))

- DDL compiler generates a set of table templates stored in a *data dictionary*

- Data dictionary contains metadata (i.e., data about data)

  - Database schema

  - Integrity constraints

  - Primary key (ID uniquely identifies instructors)

  - Authorization

- Who can access what

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
  - *Pure* – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - Commercial – used in commercial systems
    - SQL is the most widely used commercial language

Structured Query Language (SQL)

- The most widely used commercial language

- SQL is NOT a Turing machine equivalent language

- To be able to compute complex functions,
  SQL is usually embedded in some higher-level language

- Application programs generally access databases
  through one of

  - Language extensions to allow embedded SQL

  - Application program interfaces (e.g., ODBC/JDBC) which allow SQL
   queries to be sent to a database

Database Design

- Is there any problem with this relation?

| ID	name	salary	dept_name	building	budget
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

Database Engines

- Essential building blocks of database engines
  - Storage manager
    - Transaction manager
  - Query processor

- Various levels of abstraction
  - Users interact with tools
  - Query processor interacts with storage manager
  - Storage manager interacts with disk storage
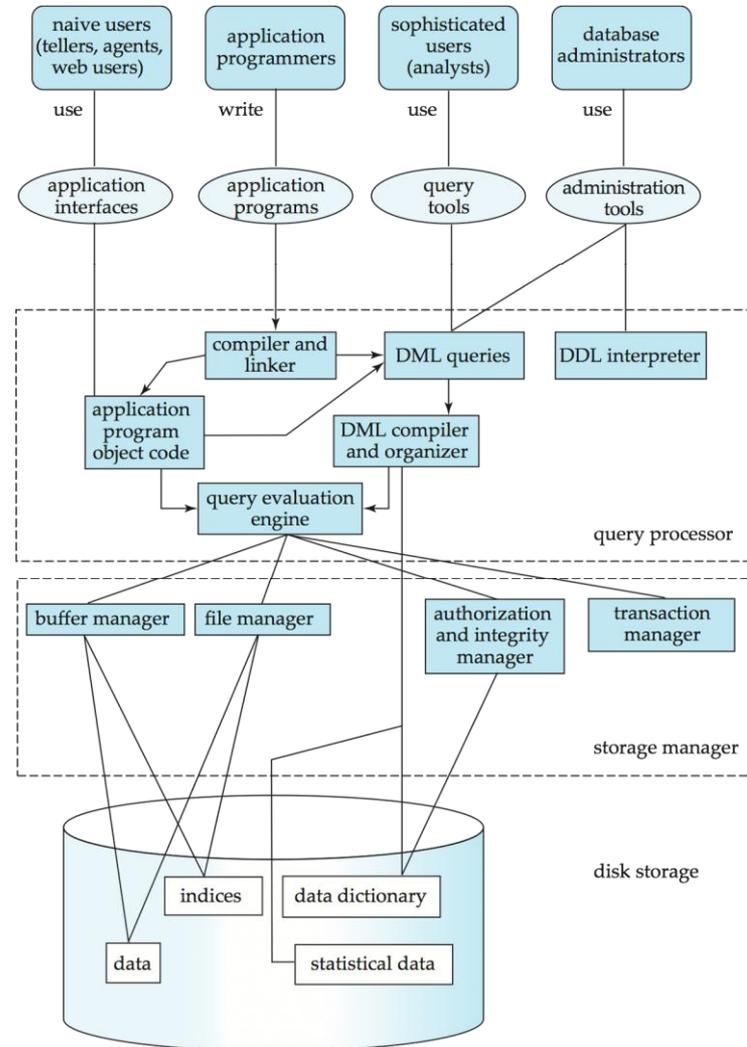
Storage Management

- Provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system

- Tasks
  - Interaction with the OS file manager
  - *Efficient* storing, retrieving, and updating of data

- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Application

DBMS

Storage
Management

Data Files
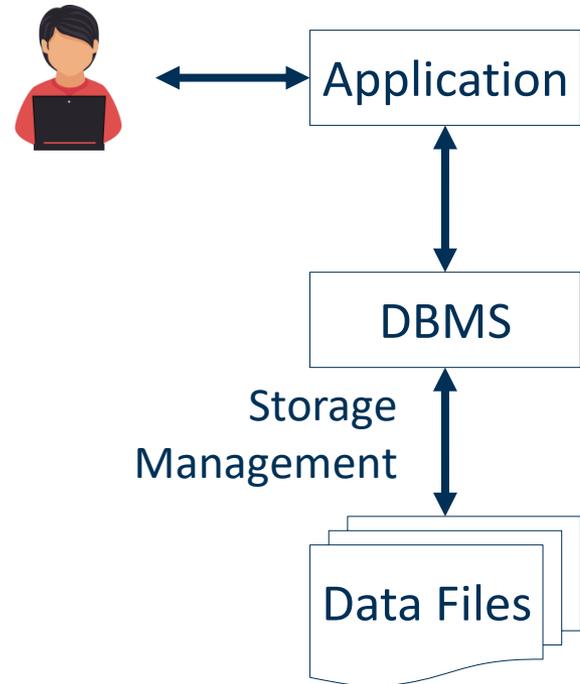
Query Processor

- Tasks
  - Parsing and translation
  - Optimization
  - Evaluation
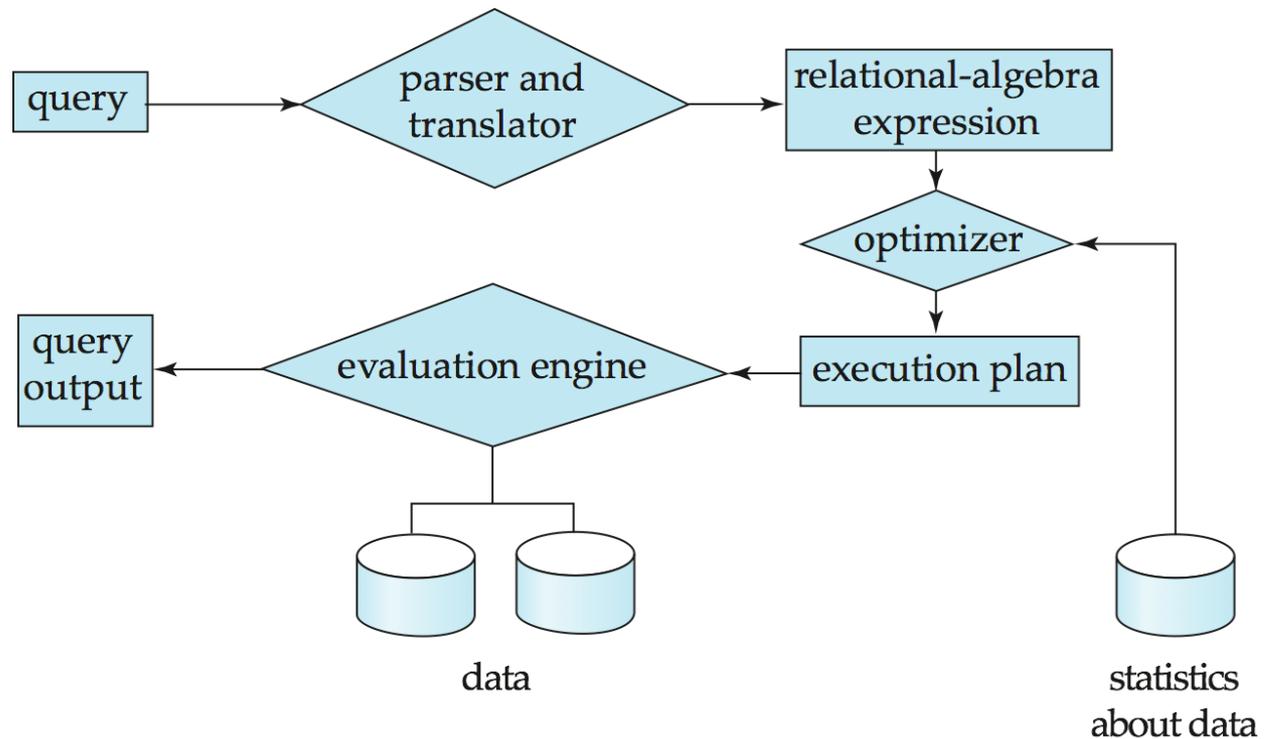
Query Processor

- Alternative ways of evaluating a given query
    - Evaluation order
    - Equivalent expressions
    - Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- 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 if the system fails?

- What if more than one user is concurrently updating the same data?

- Transaction
  - a collection of operations
  - that performs a single logical function in a database application

- *Transaction management component*
  - ensures that the database remains in a consistent (correct) state
  - despite system failures (e.g., power failures and system crashes)
  - transaction failures

- *Concurrency control manager*
  - controls the interaction among the concurrent transactions
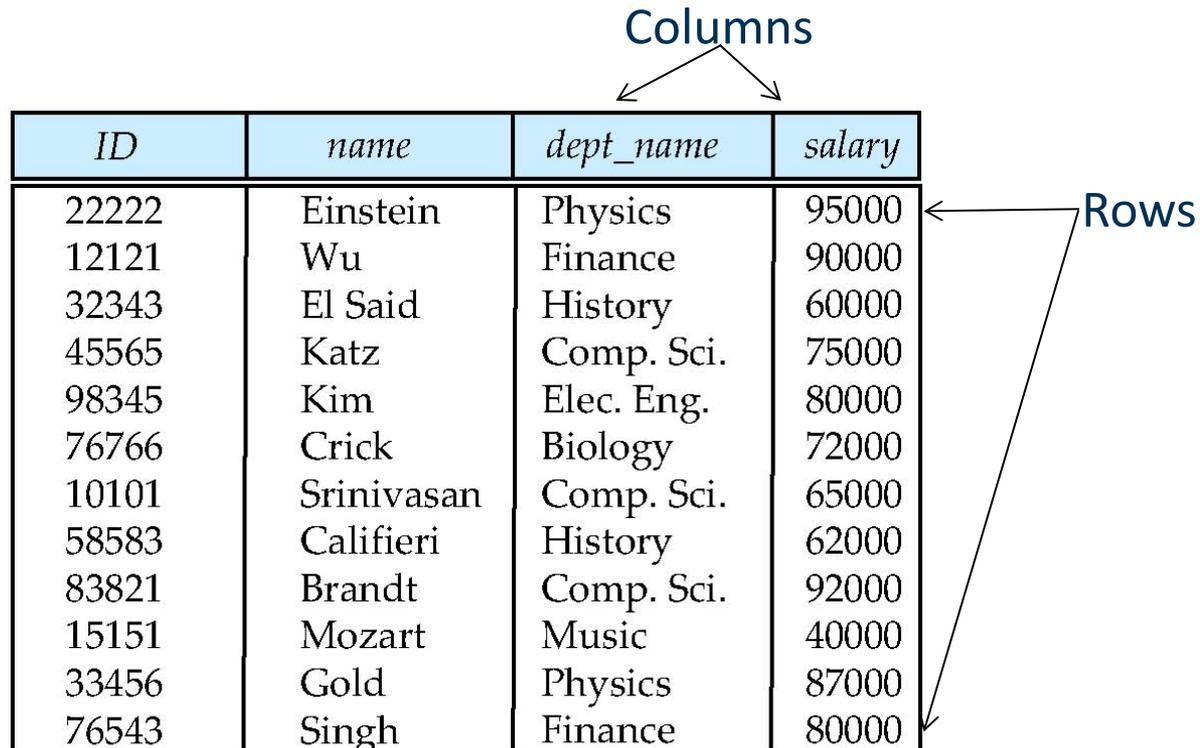 - ensures the consistency of the database

Database Users

- "Naive" users
  - Use program interfaces, e.g., university portal

- Application programmers
  - Write application programs

- Sophisticated users (e.g., analysts)
  - Use query tools
  - Create custom reports

- Database administrators
  - Use administration tools
  - May alter the database structure
  - May grant and revoke rights

The Relational Model

- All data is stored in *tables*

Columns

ID	name	dept_name	salary
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

Attribute Values

- The set of allowed values for each attribute is called the *domain* of the attribute

- Attribute values are (normally) required to be *atomic*
  - i.e., indivisible
  - e.g., break down address into street, number, ZIP code, city, ...

- The special value *null* is a member of every domain
  - indicates that the value is "unknown"
  - The null value causes complications in the definition of many operations

Atomic vs. Non-atomic Values

- Are the following attributes of a person atomic?
  - Address
  - Name
  - Age
  - Birth date
  - Birth place
  - Height
  - Salary
  - E-Mail address

- Typical database design question:
  - Would you rather store the birth date, the age, or both?

Relation Schema and Instance

- $A_1, A_2, ..., A_n$ are *attributes*

- $R = (A_1, A_2, ..., A_n)$ is a *relation schema*

- Example:

  $instructor = (ID, name, dept\_name, salary)$

- Formally, given domains $D_1, D_2, ... D_n$, a **relation r** is a subset of
  D_1 x D_2 x ... x D_n

- Thus, a relation is a set of n-tuples $(a_1, a_2, ..., a_n)$ where each $a_i \in D_i$

- The current values *(relation instance)* of a relation are specified by a table

- An element t of r is a tuple, represented by a *row* in a table

Order of Tuples

- We consider relations as *sets*
  - i.e., order of tuples/rows is irrelevant
    (may be stored in an arbitrary order)

- Example: instructor relation with unordered tuples

ID	name	dept_name	salary
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

Superkeys, Candidate Keys, Primary Keys

- Let $K \subseteq R$

- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*

- Superkey *K* is a **candidate key** if *K* is minimal
  - Example: {*ID*} is a candidate key for *Instructor*

- One of the candidate keys is selected to be the **primary key**.
  - which one?
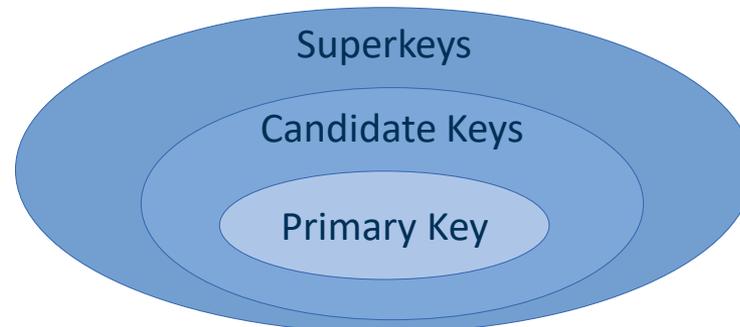
Foreign Key

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

- Foreign keys reduce *redundancy*
  - information about *department* need not be stored with every instructor

ID	name	dept_name	salary
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

dept_name	building	budget
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

Example University Database
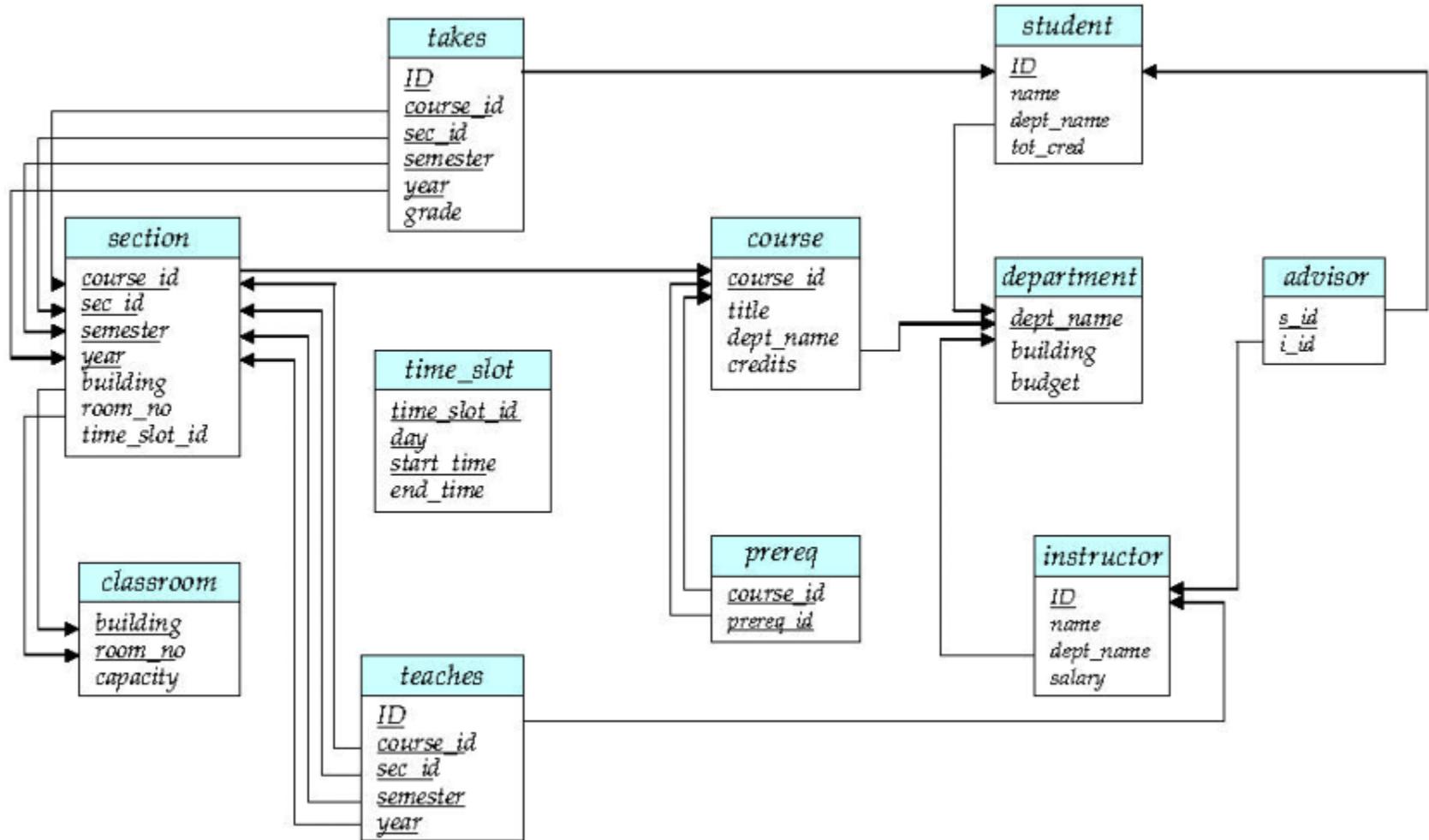
Relational Query Languages

- Procedural vs. non-procedural, or declarative

- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus

- The above three pure languages are equivalent in computing power

- We will concentrate on relational algebra
  - Not Turing machine equivalent
  - Consists of *six basic operations*

Relational Algebra

- Selection of Rows
    - $\sigma_{\text{Building=Taylor} \wedge \text{Budget>80000}}$ (departments)

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Taylor	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Department	Building	Budget
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000

Relational Algebra

- Projection (Selection of Columns)
  - $\prod_{\text{Building,Budget}}$ (departments)

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Taylor	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Building	Budget
Taylor	100000
Watson	90000
Taylor	85000
Taylor	80000
Painter	120000
Painter	50000
Watson	70000

Relational Algebra

- Projection (Selection of Columns) ctd.
  - $\prod_{Building}$ (departments)

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Taylor	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Building
Taylor
Watson
Painter

Remember:
Relations are sets
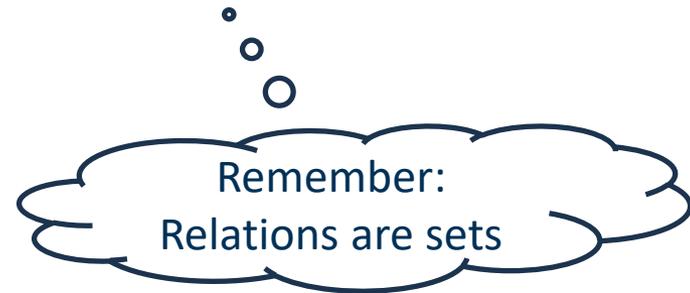
Relational Algebra

- Projection with Renaming
  - $\prod_{Building,Budget \,\text{-> }\,Total}$ (departments)

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Taylor	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Building	Total
Taylor	100000
Watson	90000
Taylor	85000
Taylor	80000
Painter	120000
Painter	50000
Watson	70000

Relational Algebra

- Set Union of Two Relations
  tech_departments and *humanities_departments*
  - tech_departments ∪ humanities_departments

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Physics	Watson	70000

Department	Building	Budget
Literature	Taylor	80000
History	Painter	50000

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Physics	Watson	70000
Literature	Taylor	80000
History	Painter	50000

Relational Algebra

- Set Difference of Two Relations
  departments and *humanities_departments*
  - departments – humanities_departments

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Literature	Taylor	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
Physics	Watson	70000

Department	Building	Budget
Literature	Taylor	80000
History	Painter	50000

Relational Algebra

- Set Intersection of Two Relations
  tech_departments and *science_departments*
  - tech_departments ∩ science_departments

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Physics	Watson	70000

Department	Building	Budget
Biology	Watson	90000
Math	Painter	30000
Astronomy	Taylor	75000
Physics	Watson	70000

Department	Building	Budget
Biology	Watson	90000
Physics	Watson	70000

A Note on Intersection

- Set intersection is *not* considered a basic operation
  - It can be expressed using set difference and union
  - $A \cap B = (A \cup B) - (A - B) - (B - A)$
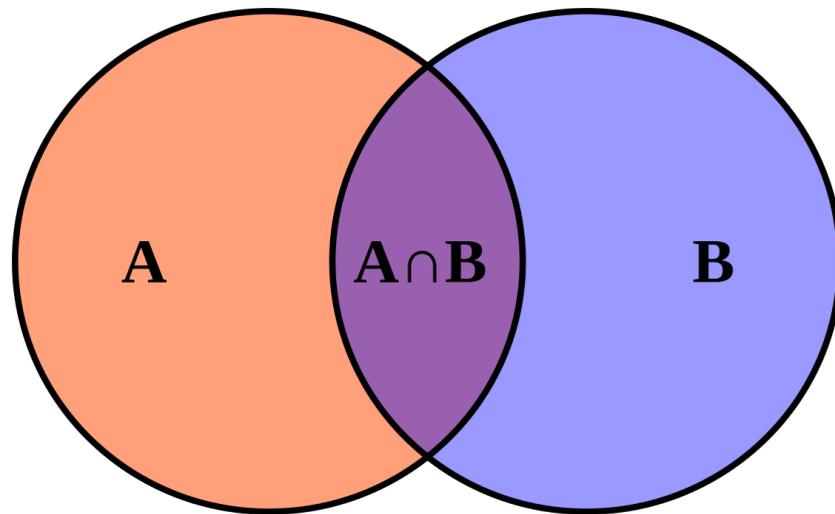
Relational Algebra

- Cartesian Product of Relations *departments* and *deans*
  - departments X deans

Department	Building	Budget
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000

Department	Name
Comp. Sci.	Smith
Biology	Johnson
Elec. Eng.	Miller

Department	Building	Budget	Department	Name
Comp. Sci.	Taylor	100000	Comp. Sci.	Smith
Comp. Sci.	Taylor	100000	Biology	Johnson
Comp. Sci.	Taylor	100000	Elec. Eng.	Miller
Biology	Watson	90000	Comp. Sci.	Smith
Biology	Watson	90000	Biology	Johnson
Biology	Watson	90000	Elec. Eng.	Miller
Elec. Eng.	Taylor	85000	Comp. Sci.	Smith
Elec. Eng.	Taylor	85000	Biology	Johnson
Elec. Eng.	Taylor	85000	Elec. Eng.	Miller

Cartesian Product of Relations: Naming Issues

- departments X deans

- Note: column names must be unique!

departments. Department	Building	Budget	deans. Department	Name
Comp. Sci.	Taylor	100000	Comp. Sci.	Smith
Comp. Sci.	Taylor	100000	Biology	Johnson
Comp. Sci.	Taylor	100000	Elec. Eng.	Miller
Biology	Watson	90000	Comp. Sci.	Smith
Biology	Watson	90000	Biology	Johnson
Biology	Watson	90000	Elec. Eng.	Miller
Elec. Eng.	Taylor	85000	Comp. Sci.	Smith
Elec. Eng.	Taylor	85000	Biology	Johnson
Elec. Eng.	Taylor	85000	Elec. Eng.	Miller

From Cartesian Products to Joins

- A cartesian product alone is not very helpful
    - Typically, we want something different
    - i.e., a combination of a selection and a cartesian product

- $\sigma_{departments.Department=deans.Department}(departments \times deans)$
    - i.e., a list of departments with their *respective* deans

departments. Department	Building	Budget	deans. Department	Name
Comp. Sci.	Taylor	100000	Comp. Sci.	Smith
Biology	Watson	90000	Biology	Johnson
Elec. Eng.	Taylor	85000	Elec. Eng.	Miller

Composing Relational Algebra Operators

- Almost perfect…
  - …but we only need the department name once
  - …and the column name *Name* is a bit uninformative

- $\prod_{\text{departments.Department} \rightarrow \text{Department, Building, Budget, Name} \rightarrow \text{Dean}}$
$$(\sigma_{\text{departments.Department=deans.Department}}(\text{departments X deans}))$$

Department	Building	Budget	Dean
Comp. Sci.	Taylor	100000	Smith
Biology	Watson	90000	Johnson
Elec. Eng.	Taylor	85000	Miller

Natural Join

R	Department	Building	Budget
	Comp. Sci.	Taylor	100000
	Biology	Watson	90000
	Elec. Eng.	Taylor	85000

- Let *r* and *s* be relations on schemas *R* and *S* respectively

- Then, the "natural join" of relations *r* and *s* ($r \bowtie s$) is a relation on schema $R \cup S$ obtained as follows:

  – Consider each pair of tuples t_r from *r* and t_s from *s*.

  i.e., r X s

S	Department	Name
	Comp. Sci.	Smith
	Biology	Johnson
	Elec. Eng.	Miller

  – If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where

  {Department}

    • *t* has the same value as t_r on *r*

    • *t* has the same value as t_s on *s*

Renaming Relations

- Consider this relation *person*:
  - Compile a list of persons, their supervisors and buildings

Person	Building	Supervisor
Smith	Taylor	Johnson
Kim	Watson	Johnson
Johnson	Taylor	Meyer
Meyer	Watson	n/a

- Problem: we need the same relation twice
  - ...but we have to distinguish its roles

- Solution: renaming

- $\sigma_{\text{supervisee.Supervisor=supervisor.Person}}\left(\rho_{supervisee}\,(person) \times \rho_{supervisor}\,(person)\right)$

supervisee. Person	supervisee. Building	Supervisee. Supervisor	Supervisor. Person	Supervisor. Building	Supervisor. Supervisor
Smith			Johnson	Taylor	Meyer
Kim	Watson	Johnson	Johnson	Taylor	Meyer
Johnson	Taylor	Meyer	Meyer	Watson	n/a

Where is Meyer?

Natural Joins

- Natural Joins are frequently used
- e.g., list all instructors with their building
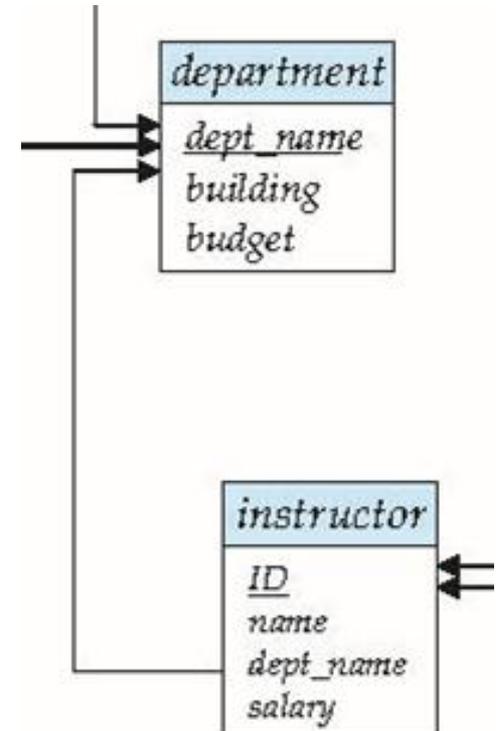
Notes on the Relational Model

- Each Query input is a table (or set of tables)

- Each query output is a table.

- All data in the output table appears
  in one of the input tables

- Relational Algebra is not Turing complete

- e.g., we cannot compute
  - SUM
  - AVG
  - MAX
  - MIN

Summary on Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	σ salary >= 85000 $^{(instructor)}$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	Π *ID, salary* $^{(instructor)}$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output all possible combinations of tuples from both relations.
∪ (Union)	Π *name* $^{(instructor)}$ ∪ Π *name* $^{(student)}$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	Π *name* $^{(instructor)}$ – Π *name* $^{(student)}$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

Intersection is not a basic operator

Summary

- Database Management Systems are an *abstraction layer*

- Applications do not have to interact directly with the file system

- DBMS offer services including

  - Checking consistency

  - Ensuring integrity

  - Security

  - Handing concurrent data access

Summary

- Relational databases are composed of tables (relations)

- Tables can be understood as sets

- Sometimes, we need a combination of values from different tables
  - e.g., all employees with their building
  - e.g., all courses attended by a particular student

- The results of those are tables
  - Not necessarily the tables in the database
  - But: all *values* in the result tables are contained in the database

- With relational algebra, we transform tables into new tables
  - And hopefully get our results…

Questions?