UNIVERSITÄT MANNHEIM

Database Technology
Introduction

Heiko Paulheim

Outline

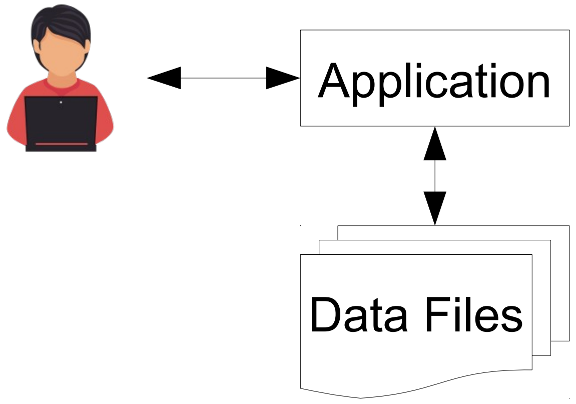
- The Need for Databases
- Data Models
- Relational Databases
- Database Design
- Storage Manager
- Query Processing
- Transaction Manager
- Introduction to the Relational Model

Data Base Management Systems (DBMS)

- DBMS contains information about a particular enterprise
  - 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



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., GPA > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

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: 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
  - Uncontrolled concurrent accesses can lead to inconsistencies
  - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Data Consistency: 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

Data Integrity: Example

- Example: ZIP code has to be a five digit number
- As an application developer, would you prefer
  - Adding a single check in each part of the application where a ZIP code is entered
    - student applications
    - contracts with employees
    - travel reimbursement
    - …
  - Adding the check at a single point (i.e., before the data is written into the database)

Atomicity of Updates: Example

- Example piece of (pseudo) code: retiring a lecturer

```
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

Concurrency: Example

- Example: register for a course if there are places left

```
Read num_current_participants
       from file

If num_current_participants
       < limit
Then
       add participant to fi
```

User 1

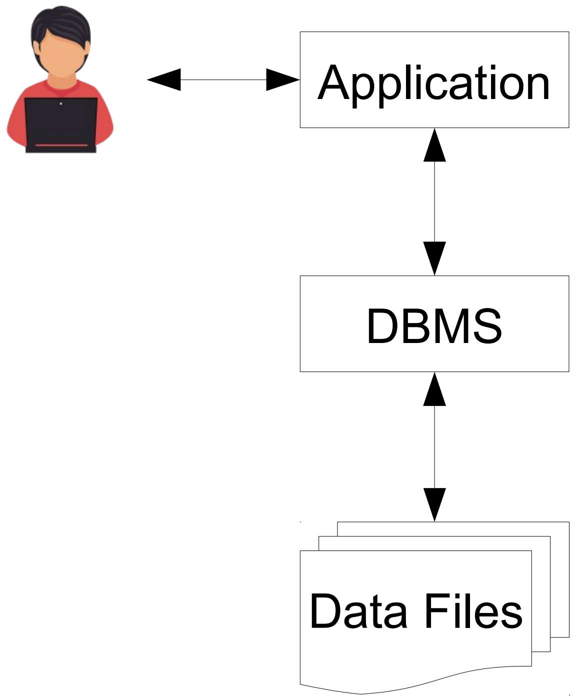
```
Read num_current_participants
       from file

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

User 2

Idea of Database Management Systems

- Introduce a level of abstraction
- Handle issues of…
  - consistency
  - integrity
  - transaction atomicity
  - concurrency
  - security
  - …
- ...in a centralized fashion



Levels of Abstraction

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

  type *instructor* = **record**

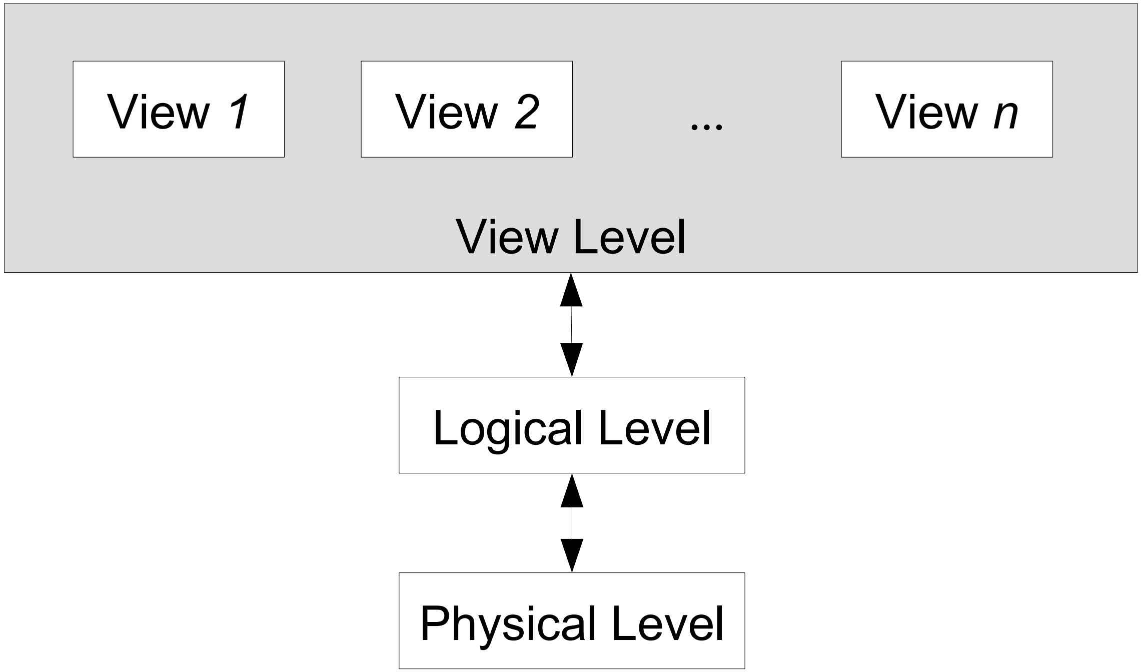
  ID : string;
  name : string;
  dept_name : string;
  salary : integer;

  end;

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

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

The Relational Model

- 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

- 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

Database Design

- Is there any problem with this relation?

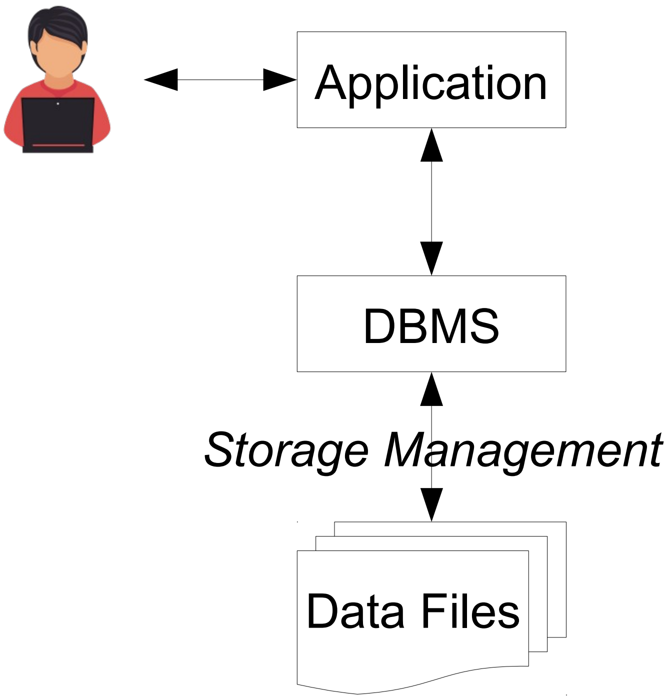
ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	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
  - Query processor
  - Transaction manager

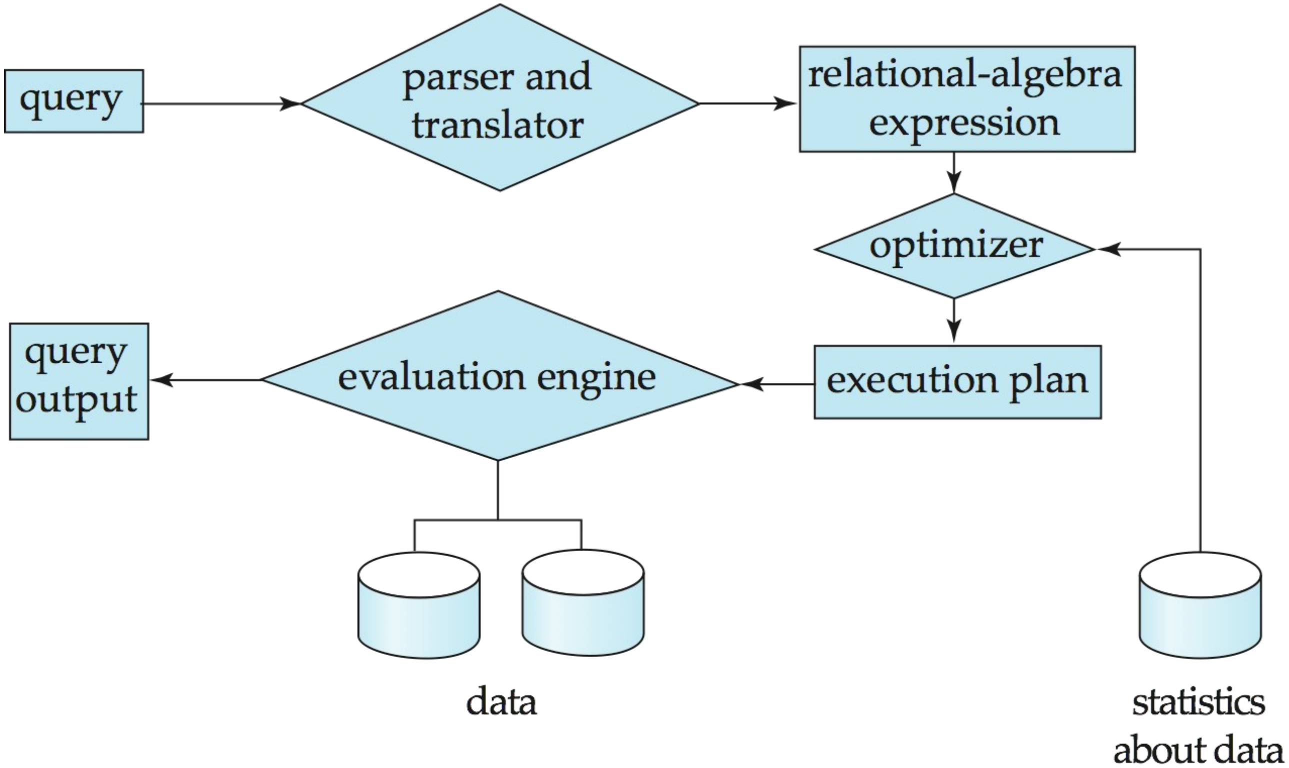
Storage Management

- 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



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

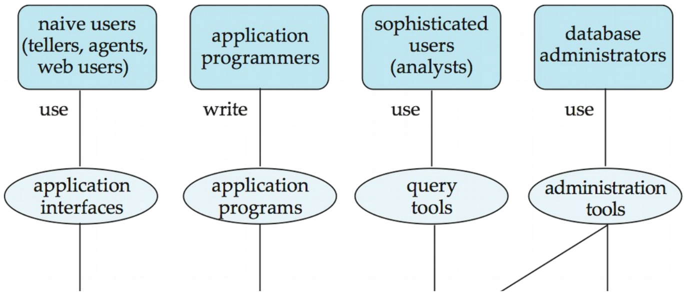
- What if the system fails?
- What if more than one user is concurrently updating the same data?
- Transaction
  - a collection of operations
  - that performs a single logical function in a database application
- *Transaction management component*
  - ensures that the database remains in a consistent (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

Database System Internals

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



Database Architecture

- Design decisions of a database system and application:
  - Centralized
  - Client-server
  - Parallel (multi-processor)
  - Distributed
- Each of those comes with its own requirements
- Needs different solutions, e.g., for security, concurrency handling, etc.

History of Database Systems

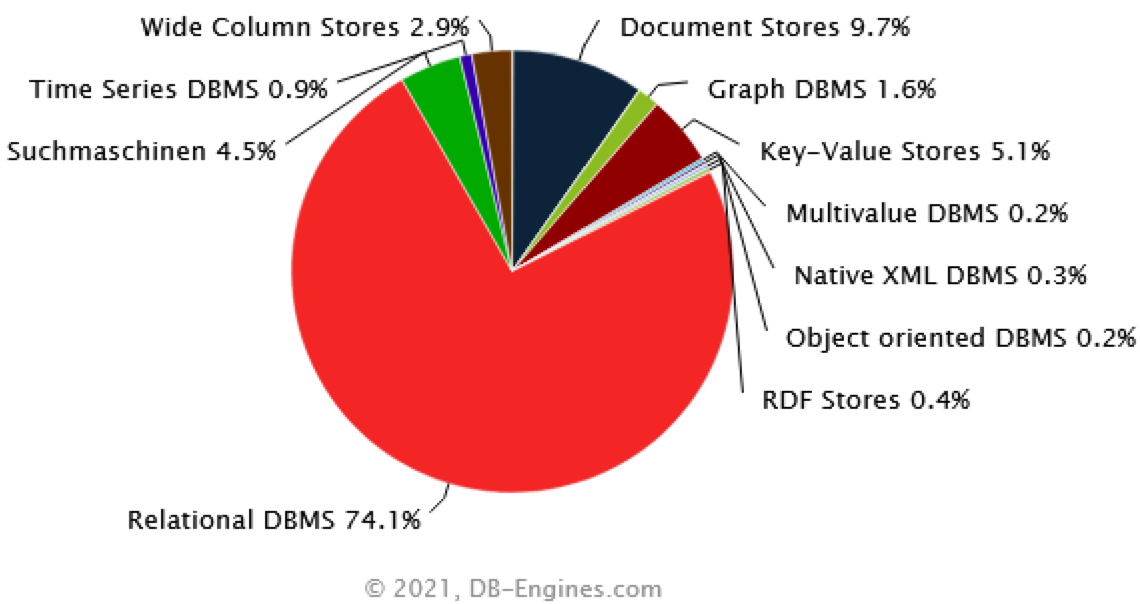
- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
  - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
  - Would win the ACM Turing Award for this work
  - IBM Research begins System R prototype
  - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

History of Database Systems

- 1980s:
  - Research relational prototypes evolve into 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 Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
  - Google BigTable, Yahoo PNuts, Amazon, ..

The Relational Model

- Recent past: much research on novel models
  - graph databases, key value stores (NoSQL), …
  - the relational model is still the most prevalent



The Relational Model

- Recap: all data is stored in *tables*

Columns

Rows

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

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

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

- Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)

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

  $D_1 \times D_2 \times \ldots \times D_n$

- Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ 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

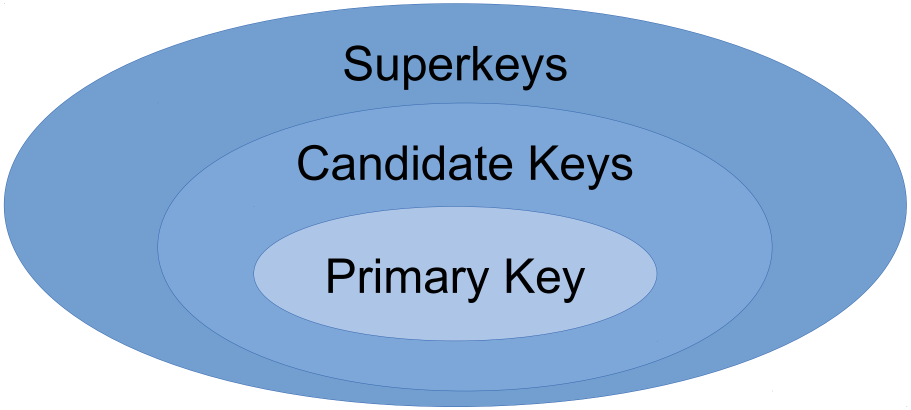
Order of Tuples

- We consider relations as *sets*
  - i.e., order of tuples 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 *instructor* is a foreign key from *instructor* referencing *department*
- Foreign keys reduce *redundancy*
  - information about *department* need not be stored with every instructor

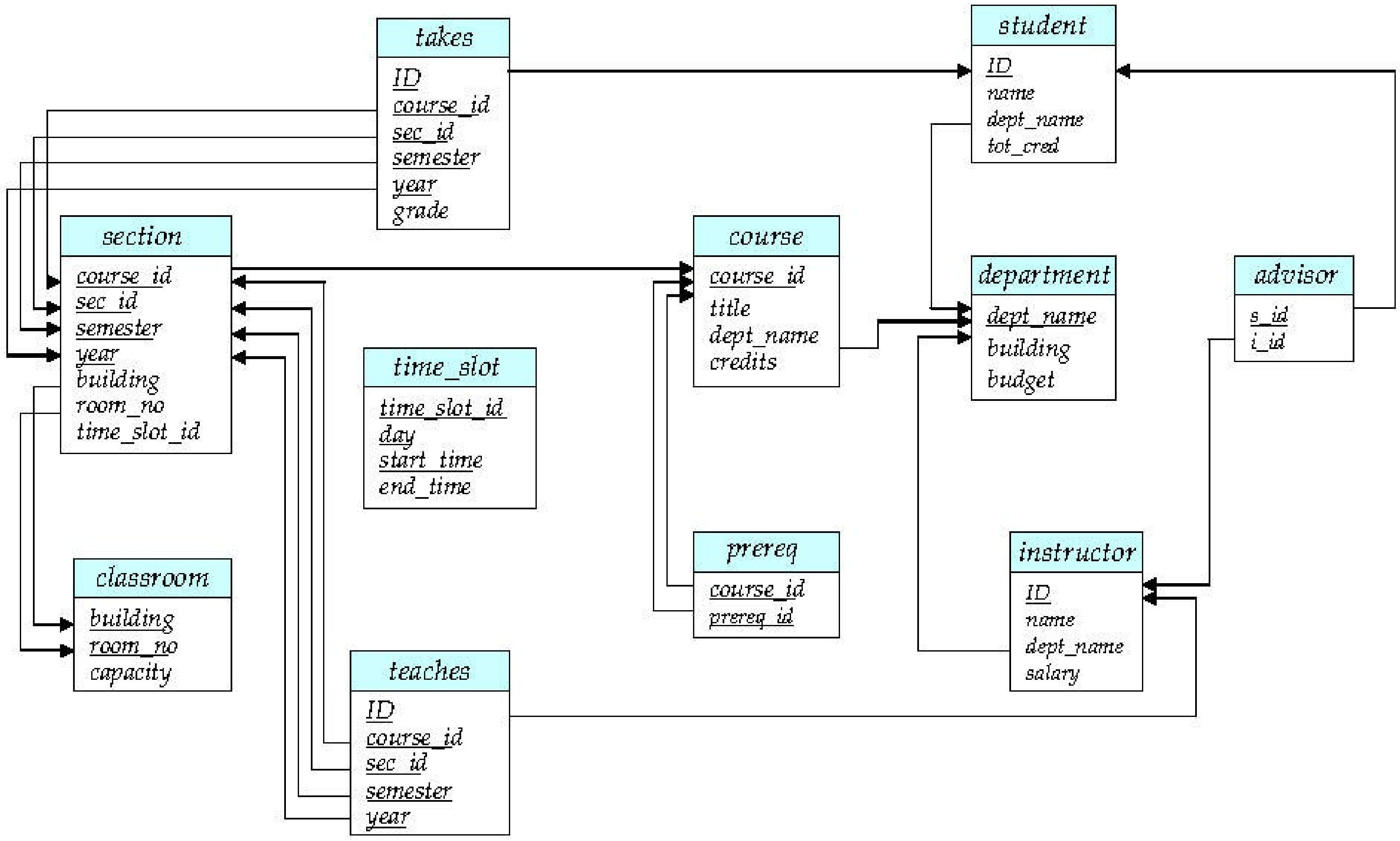
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music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

Selection of Rows (Tuples)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Projection (Selection of Columns)

- Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

Remember:
Relations are sets

- $\Pi_{A,C}$ (*r*)

A	C
α	1
α	1
β	1
β	2

Projection with Renaming

- Relation r

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,D \leftarrow C}\ (r)$

A	D
α	1
α	1
β	1
β	2

D

Union of Two Relations

- Relations r,s

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

Set Difference of Two Relations

- Relations r,s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r – s

A	B
α	1
β	1

Set Intersection of Two Relations

- Relations r,s

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

- *Note:* $r \cap s = r - (r - s) = s - (s - r)$

Cartesian Product (Joining Two Relations)

- Relations r,s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- r x s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian Product: Naming Issue

- Relations r,s

A	B
α	1
β	2

r

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- r x s

A	r.B	s.B	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Renaming Relations

- Allows us to refer to a relation, (say E) by more than one name.
  - $\rho_x(E)$ returns the expression *E* under the name *X*
- Useful, e.g., for joining a table with itself

- Relation r

A	*B*
α	1
β	2

r

- $r \times \rho_s(r)$

r.A	*r.B*	*s.A*	*s.B*
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$
- Relations r,s

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Natural Join

- Let *r* and *s* be relations on schemas *R* and *S* respectively.
- Then, the "natural join" of relations *R* and *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*.
  - 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
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*

Natural Join

- Relations r,s

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

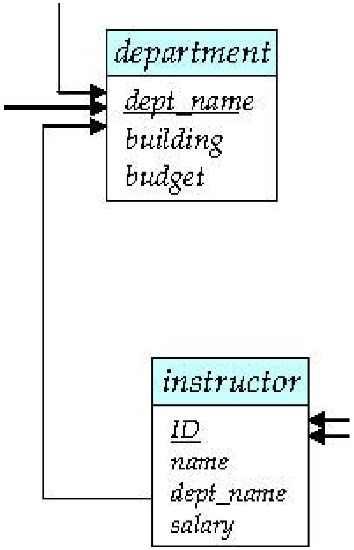
- Natural Join r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$$\Pi_{A,\ r.B,\ C,\ r.D,\ E}(\sigma_{r.B = s.B \wedge r.D = s.D}\ (r \times s)))$$

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)	$\sigma_{\text{salary} >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}(instructor)$
	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)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor) - \Pi_{name}(student)$
	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.

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?