Introduction

Outline

- *The Need for Databases*
- *Data Models*
- *Relational Databases*
- *Database Design*
- *Storage Manager*
- *Query Processing*
- *Transaction Manager*

Database Management System (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 systems*
  - designed to manage large bodies of information.
  - Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
  - must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.
  - If data are to be shared among several users, the system must avoid possible anomalous results.
  - Because information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data.
- *Databases can be very large.*
- *Databases touch all aspects of our lives*

Database Management System (DBMS)

- *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

Purpose of Database Systems

- *Computerizing the management of commercial data*
- *In the early days, database applications were built directly on top of file systems*
  - Disadvantages
    - 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.
    - Atomicity problems - 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
- *Application programs*

University Database Example

- *Application program examples*
  - 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

Database systems offer solutions to all the above problems

View of data

- *A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.*
- *A major purpose of a database system is to provide users with an abstract view of the data.*
- *That is, the system hides certain details of how the data are stored and maintained.*

Levels of Abstraction

- ***Physical level:***
  - lowest level of abstraction
  - describes how a record (e.g., instructor) is stored.
- ***Logical level:*** *describes data stored in database, and the relationships among the data.*
  - physical data independence. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

type *instructor* = ***record***

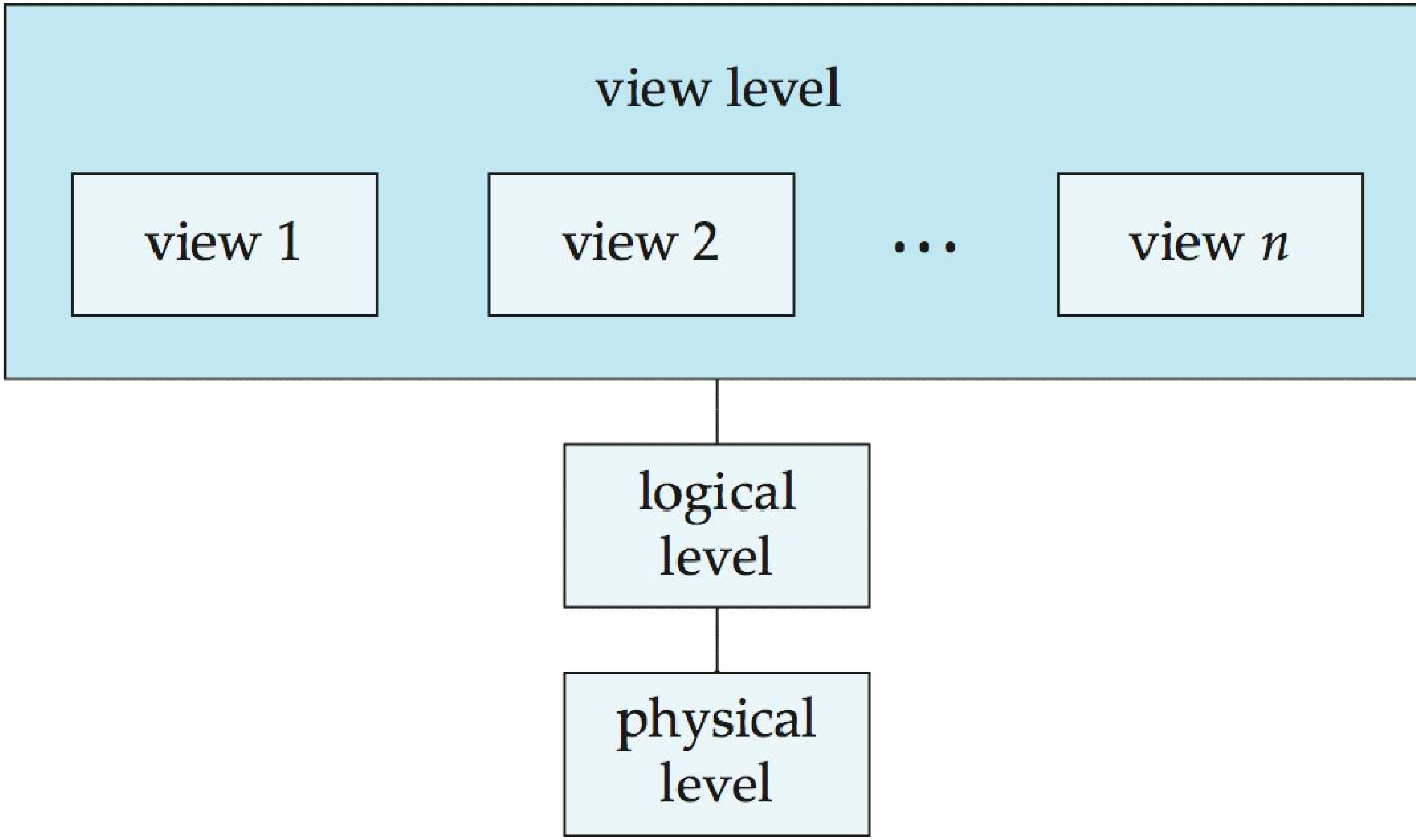
ID : string;
name : string;
dept_name : string;
salary : integer;

end*;*

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

View of Data

An architecture for a database system



Instances and Schemas

- *Similar to types and variables in programming languages*
- *The overall design of the database is called the database schema.*
- ***Logical Schema*** *– the overall logical structure of the database*
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- ***Physical schema****– the overall physical structure of the database*
- ***Instance*** *– the actual content of the database at a particular point in time*
  - Analogous to the value of a variable
- ***Physical Data Independence*** *– the ability to modify the physical schema without changing the logical schema*
  - Applications depend on the logical 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

- *the structure of a database is the data model:*
- *a collection of conceptual 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)*
- *Other older models:*
  - Network model
  - Hierarchical model

Relational Model

- *All the data is stored in various tables.*
- *Example of tabular data in the relational model*

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

A Sample Relational Database

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

Database Languages

- *A database system provides*
  - data-definition language to specify the database schema
  - data-manipulation language to express database queries and updates.
- *In practice, the data-definition and data-manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.*

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
- *The types of access are:*
  - Retrieval of information stored in the database
  - Insertion of new information into the database
  - Deletion of information from the database
  - Modification of information stored in the database
- *There are basically two types:two types:*
  - Procedural DML s require a user to specify what data are needed and how to get those data.
  - Declarative DML s (also referred to as nonprocedural DML s) require a user to specify what data are needed without specifying how to get those data.

Data Manipulation Language (DML)

- *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
  - A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language.

SQL

- *The most widely used commercial 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 interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

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*

Database Design (Cont.)

- *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

Design Approaches

- *Need to come up with a methodology to ensure that each of the relations in the database is "good"*
- *Two ways of doing so:*
  - Entity Relationship Model (Chapter 7)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 8)
    - Formalize what designs are bad, and test for them

Object-Relational Data Models

- *Relational model: flat, “atomic” values*
- *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 database 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 the basis for all new generation data interchange formats.*
- *A wide variety of tools is available for parsing, browsing and querying XML documents/data*

Database Engine

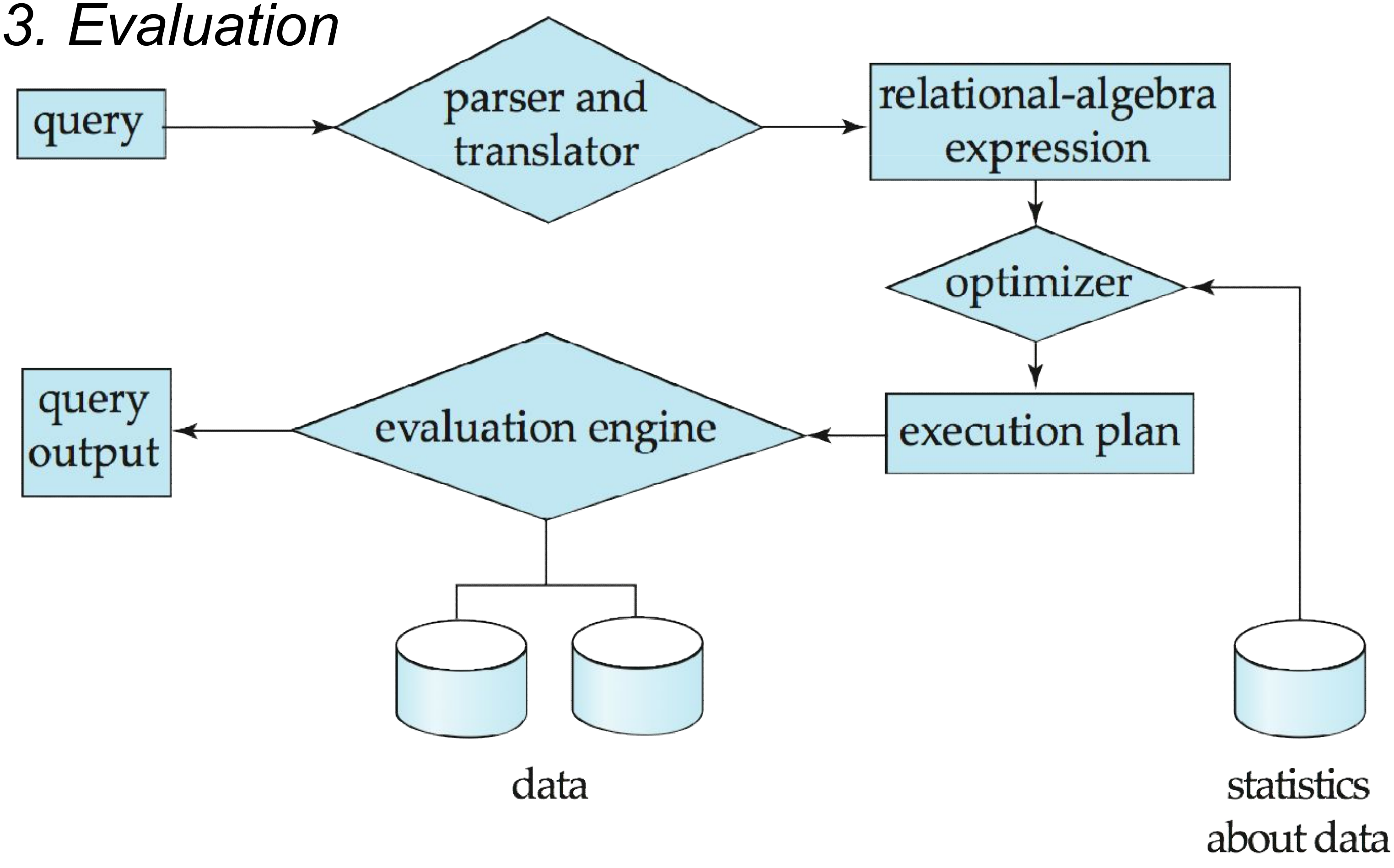
- *Storage manager*
- *Query processing*
- *Transaction manager*

Storage Management

- ***Storage manager*** *is a program module that provides the 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 OS file manager
  - Efficient storing, retrieving and updating of data
- *Issues:*
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

1. *Parsing and translation*
2. *Optimization*
3. *Evaluation*



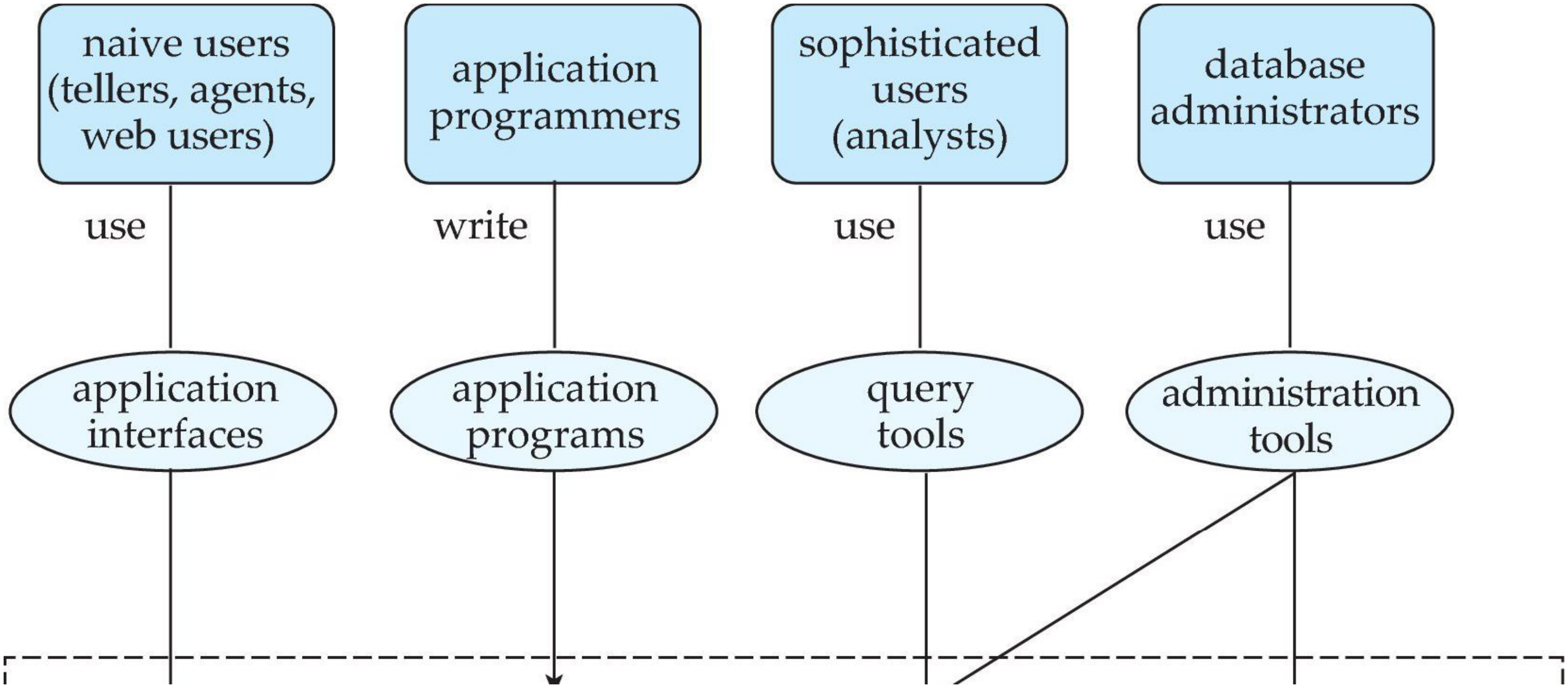
Query Processing (Cont.)

- *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*
- *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?*
- *A **transaction** is 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 operating system crashes) and transaction failures.*
- ***Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.*

Database Users and Administrators



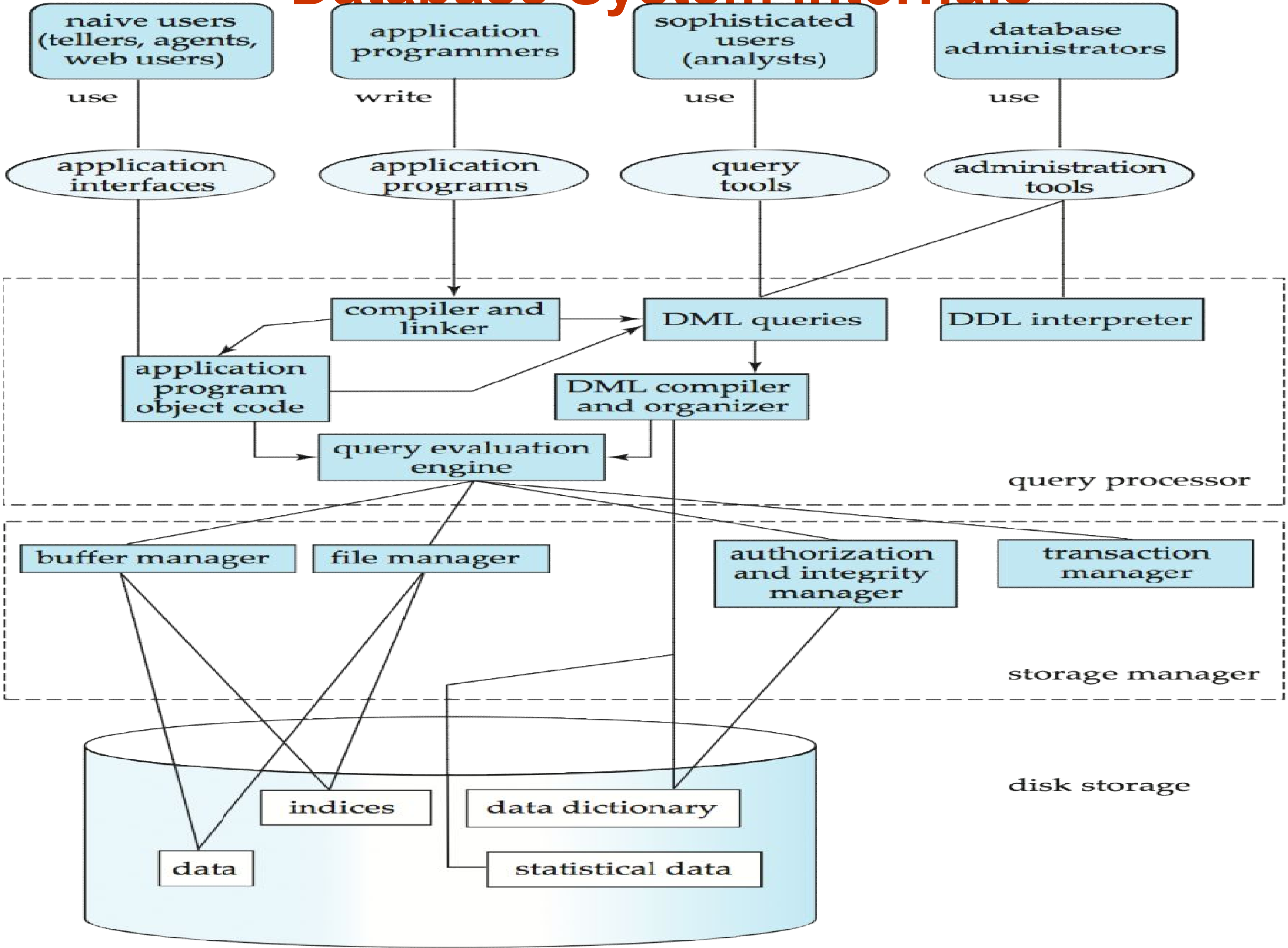
Database

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*
- *Parallel (multi-processor)*
- *Distributed*

Database System Internals



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 (cont.)

- *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, ..

Introduction to Relational Model

Example of a Relation

attributes
(or columns)

tuples
(or rows)

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

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

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 sets $D_1, D_2, \ldots. D_n$ 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

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

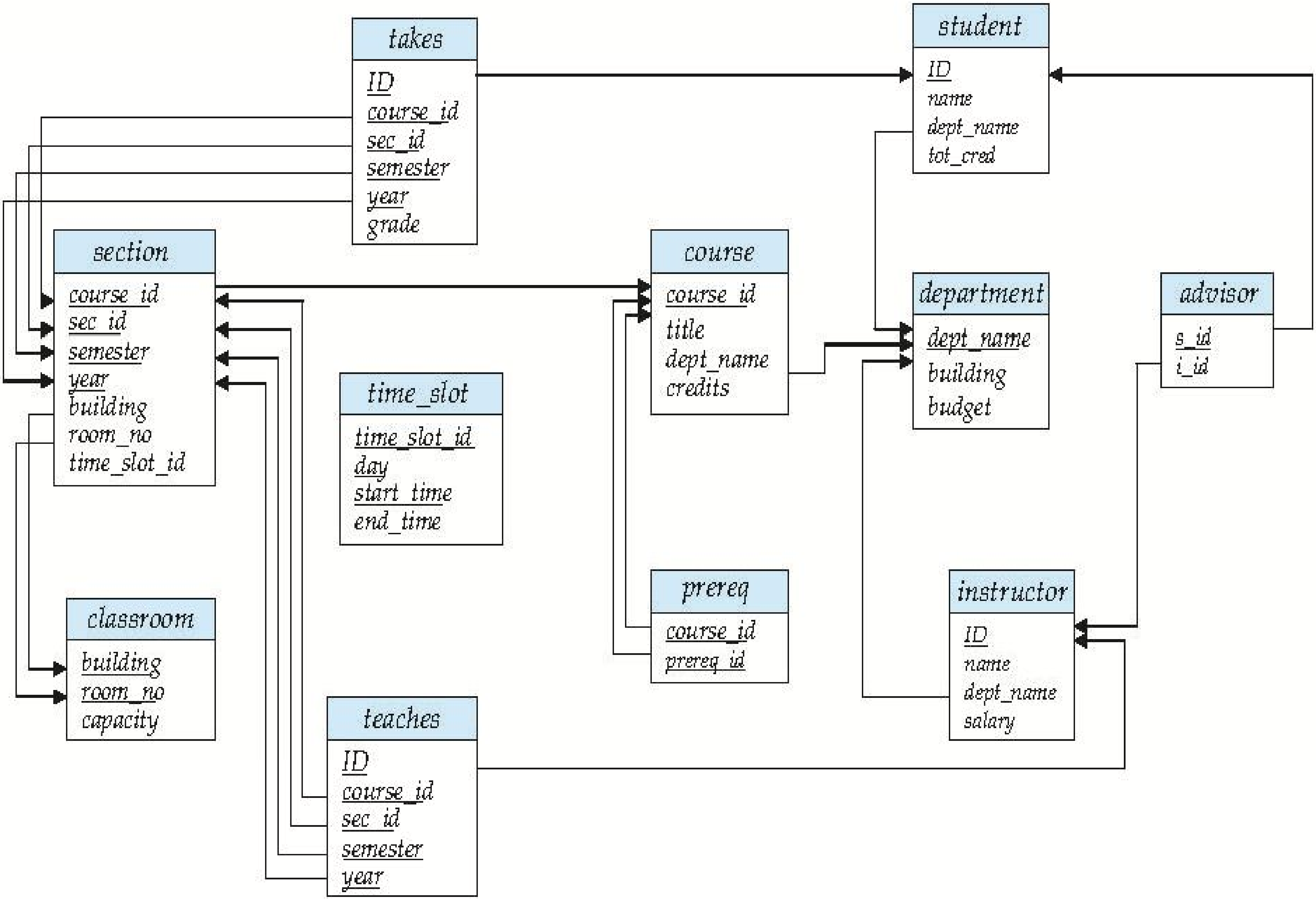
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music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

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** 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*

Schema Diagram for University Database



Relational Query Languages

- Procedural vs .non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not turning-machine equivalent
  - consists of 6 basic operations

Select Operation – 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

Project Operation – selection of columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

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

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Union 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
α	2
β	1
β	3

Set difference 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
β	1

Set intersection of two relations

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

joining two relations -- Cartesian-product

- Relations *r, s*:

r

A	B
α	1
β	2

s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

- *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 a Table

- 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*

- Relations *r*

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- $r \times \rho_s (r)$

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

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

Composition of Operations

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

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

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

Joining two relations – 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 Example

- 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
  - $r \bowtie s$

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

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

Notes about Relational Languages

- 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 Turning complete
- Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

Summary of 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 pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (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.