CS 3042 Database Systems
CS 3272 Embedded Database Systems

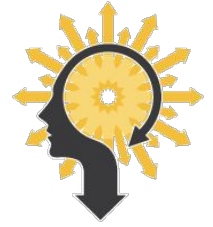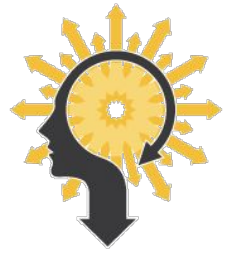# Chapter 1: Introduction

Introduction

- **Learning Outcomes (CS 3042)**
- After completing this module, students should be able to
  - explain the role of database systems in information management.
  - describe the basic fundamental concepts of data modelling and relational databases.
  - apply entity-relationship modelling and normalization for simple database requirements.
  - demonstrate the use of a query language to create, update and query a simple database.
  - construct simple applications that requires manipulating data in a DBMS.
- Recommended Text: Silberschatz, Korth and Sudarshan *Database System Concepts, 7th Edition*, McGraw-Hill 2 McGraw Hill.

Slides are based on the slides provided with this book

Introduction

★ **Learning Outcomes (CS 3272)**

★ After completing this module, students should be able to

- Explain the role of embedded databases in information management

- Describe the use of Embedded Databases for mobile, context-aware very small databases, resident on smart cards, PADs, cellular phone SIM cards or Wireless Sensors.

- Explain the importance of data consistency, concurrency control, query processing, optimization, recovery management in Embedded Databases.

- Demonstrate the use of a query language to create, update and query a simple database.

- Construct simple applications that require manipulating data in an Embedded Database.

Grading Scheme

- **Method of Assessment:**

  - 2 hour closed book examination - 60%

  - Continuous assessments - 50%

- **Breakdown of the Continuous Assessments**

- **Marks:**

  - 2-hour closed book end of semester examination - 60%

  - Mini project (Group) - 20%

  - Mid Term Quiz (Moodle) - 10 %

  - In-class/lab pop quizzes/labs - 10 %

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 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 class rosters

  - 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., account balance > 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 (Cont.)

◆ Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
◆ Concurrent access by multiple users
  - Concurrent access needed for performance
  - 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

Database systems offer solutions to all the above problems

Levels of Abstraction

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

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

    type *instructor* = **record**

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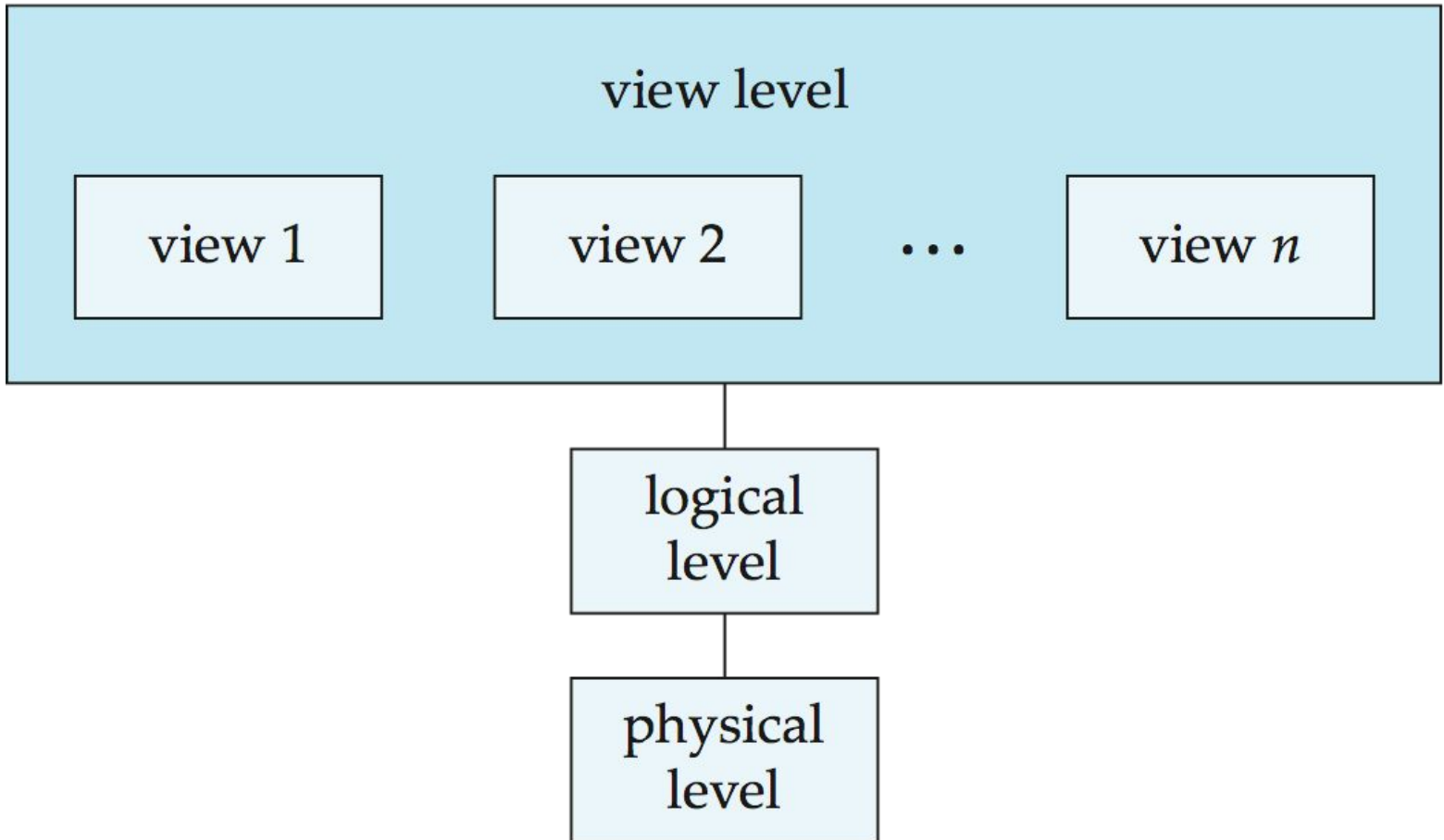   *dept_name* : string;
    salary : integer;

    end;

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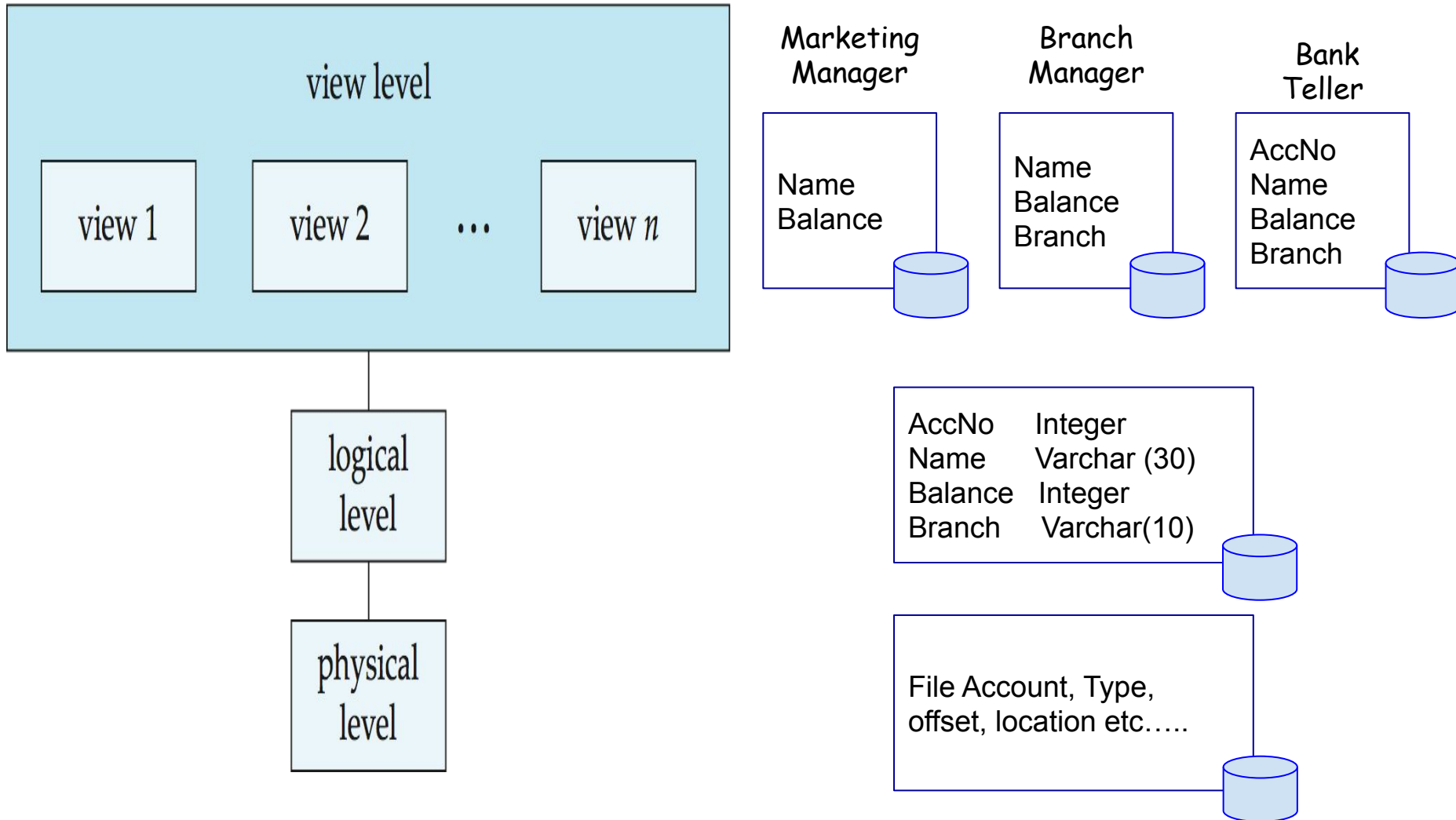
View of Data

An architecture for a database system
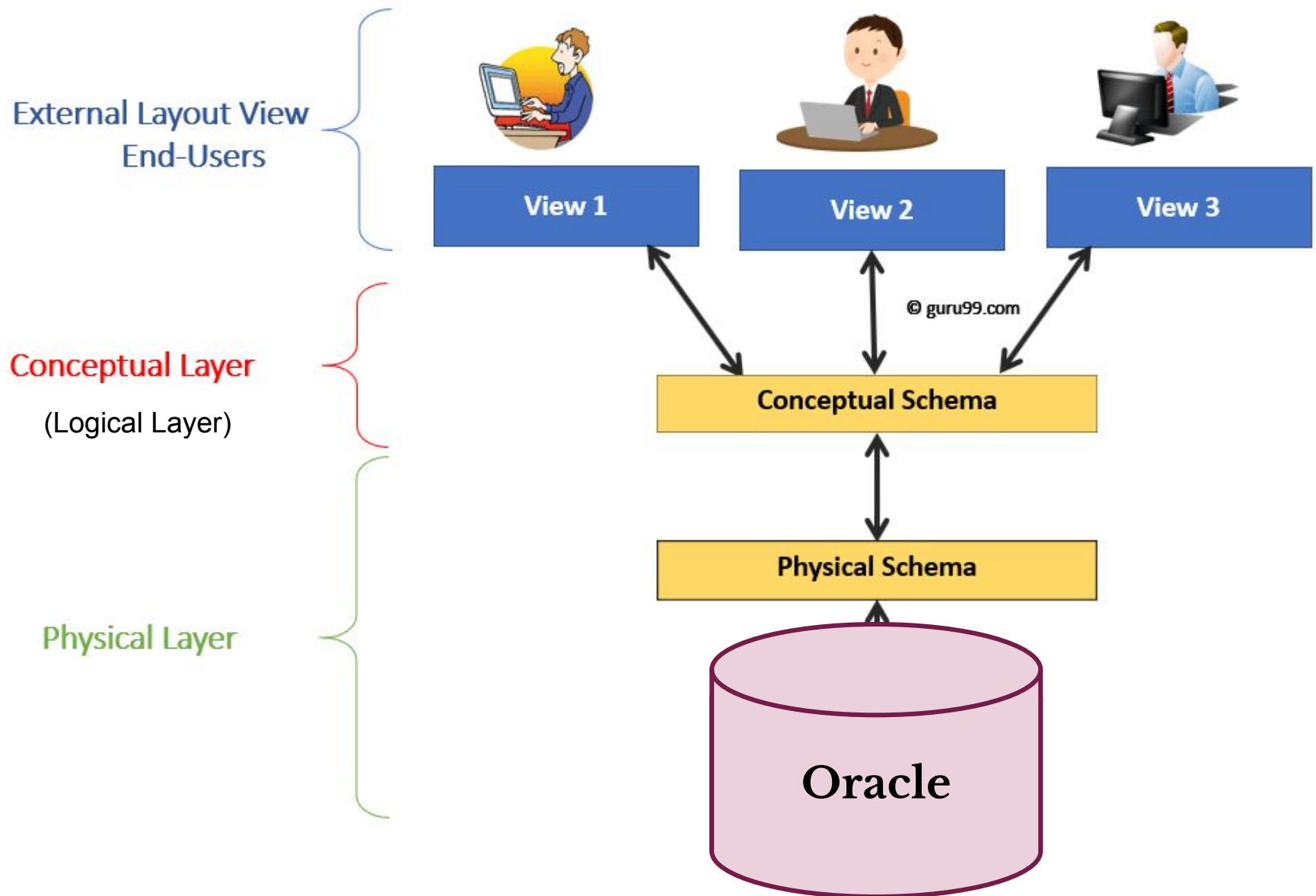
View of Data

An architecture for a database system

Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the 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.

Physical Data Independence

External Layout View End-Users

Conceptual Layer

(Logical Layer)

Physical Layer

| View 1 | View 2 | View 3 |

© guru99.com

Conceptual Schema

Physical Schema

Oracle

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)

- Other older models:
  - Network model
  - Hierarchical model
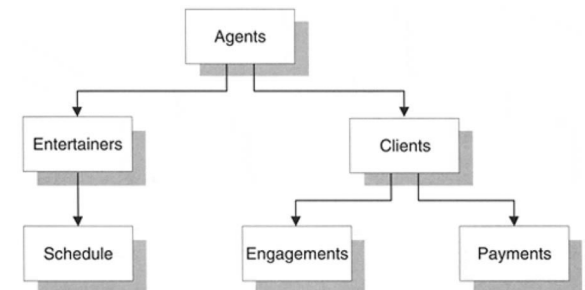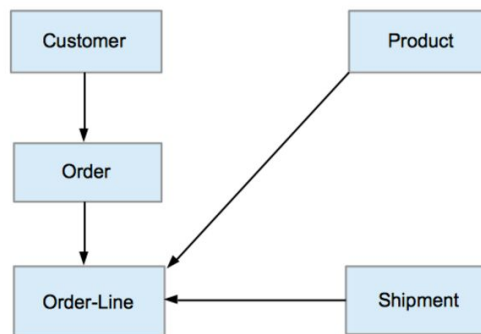
Relational Model

- Relational model (Chapter 2)
- 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

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
    - DML also known as query language
- Two classes of languages
    - **Procedural** – user specifies what data is required and how to get those data
    - **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query 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)
    - Referential integrity (**references** constraint in SQL)
      - e.g. *dept_name* value in any *instructor* tuple must appear in *department* relation
  - Authorization

SQL

- **SQL**: widely used non-procedural language
  - Example: Find the name of the instructor with ID 22222
    select   *name*
    from   *instructor*
    where   *instructor.ID* = '22222'
  - Example: Find the ID and building of instructors in the Physics dept.

    select *instructor.ID*, *department.building*
    from *instructor*, *department*
    where *instructor.dept_name* = *department.dept_name* **and**
            department.dept_name = 'Physics'

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database
- Chapters 3, 4 and 5

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

- Physical Design – Deciding on the physical layout of the database

Database Design?

- Is there any problem with this design?

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

➔ Normalization Theory (Chapter 7)

◆ Formalize what designs are bad, and test for them

➔ Entity Relationship Model (Chapter 6)

◆ Models an enterprise as a collection of *entities* and *relationships*

● Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects

○ Described by a set of *attributes*

● Relationship: an association among several entities

◆ Represented diagrammatically by an *entity-relationship diagram:*

The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
  - Entity: a "thing" or "object" in the enterprise that is distinguishable from other objects
    - Described by a set of *attributes*
  - Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram:*
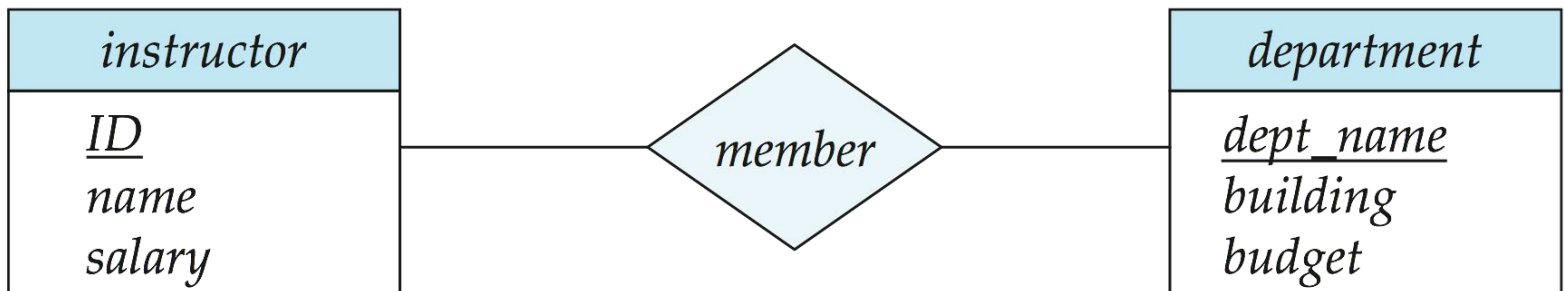
What happened to dept_name of instructor and student?

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

1. Storage manager

2. Query processing

3. Transaction manager
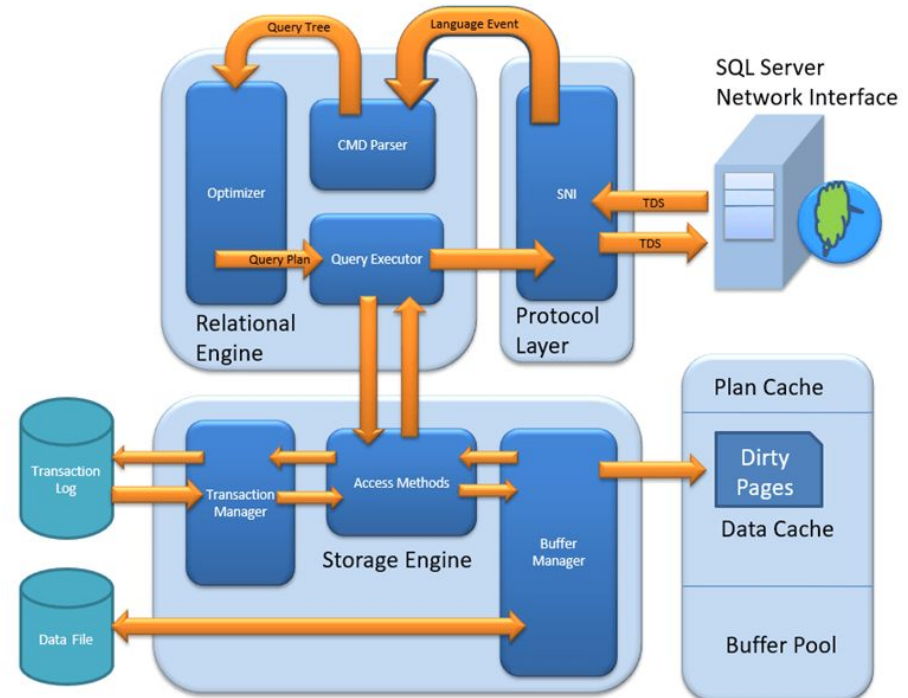
Storage Management

- **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 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
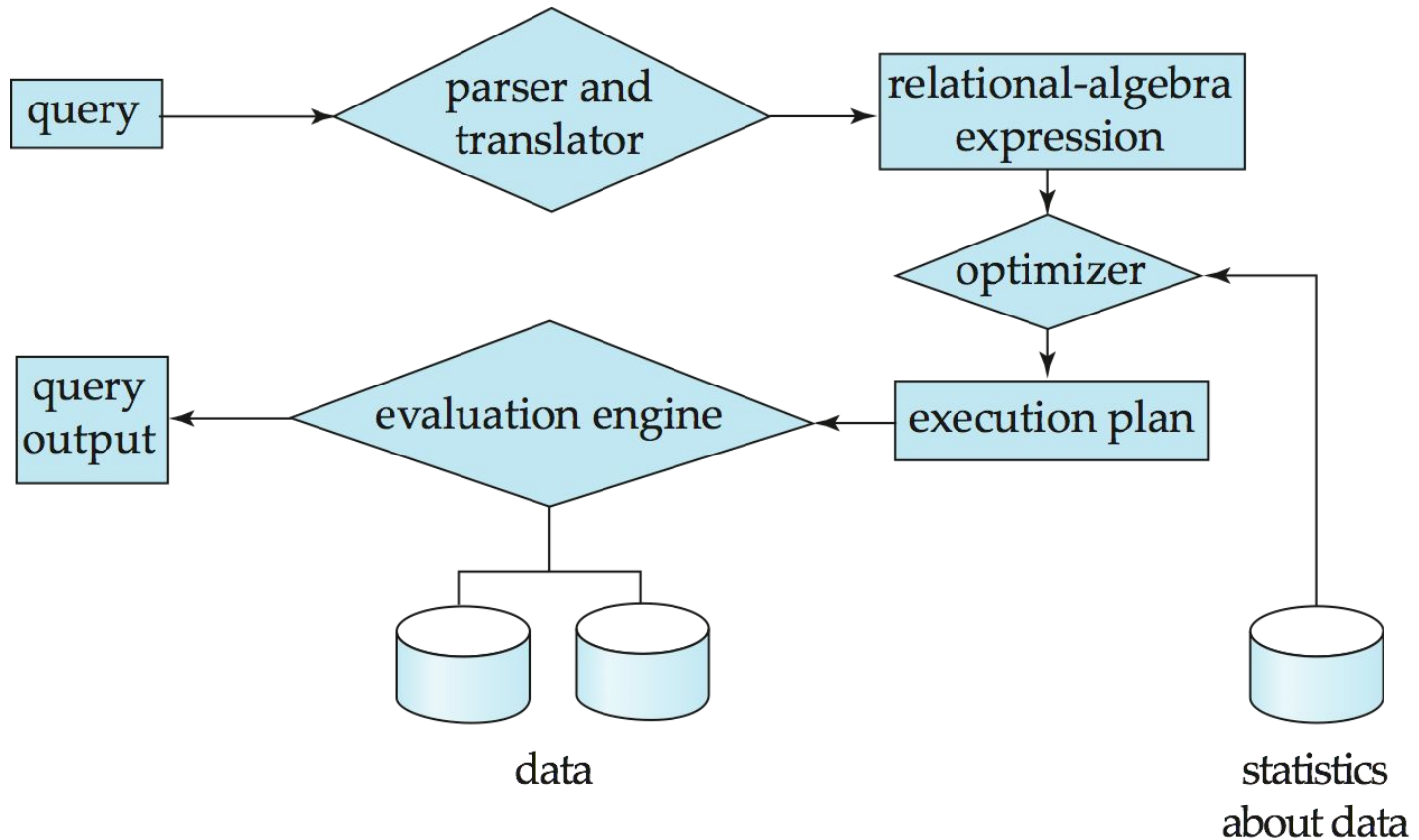
Query Processing (Cont.)

- 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
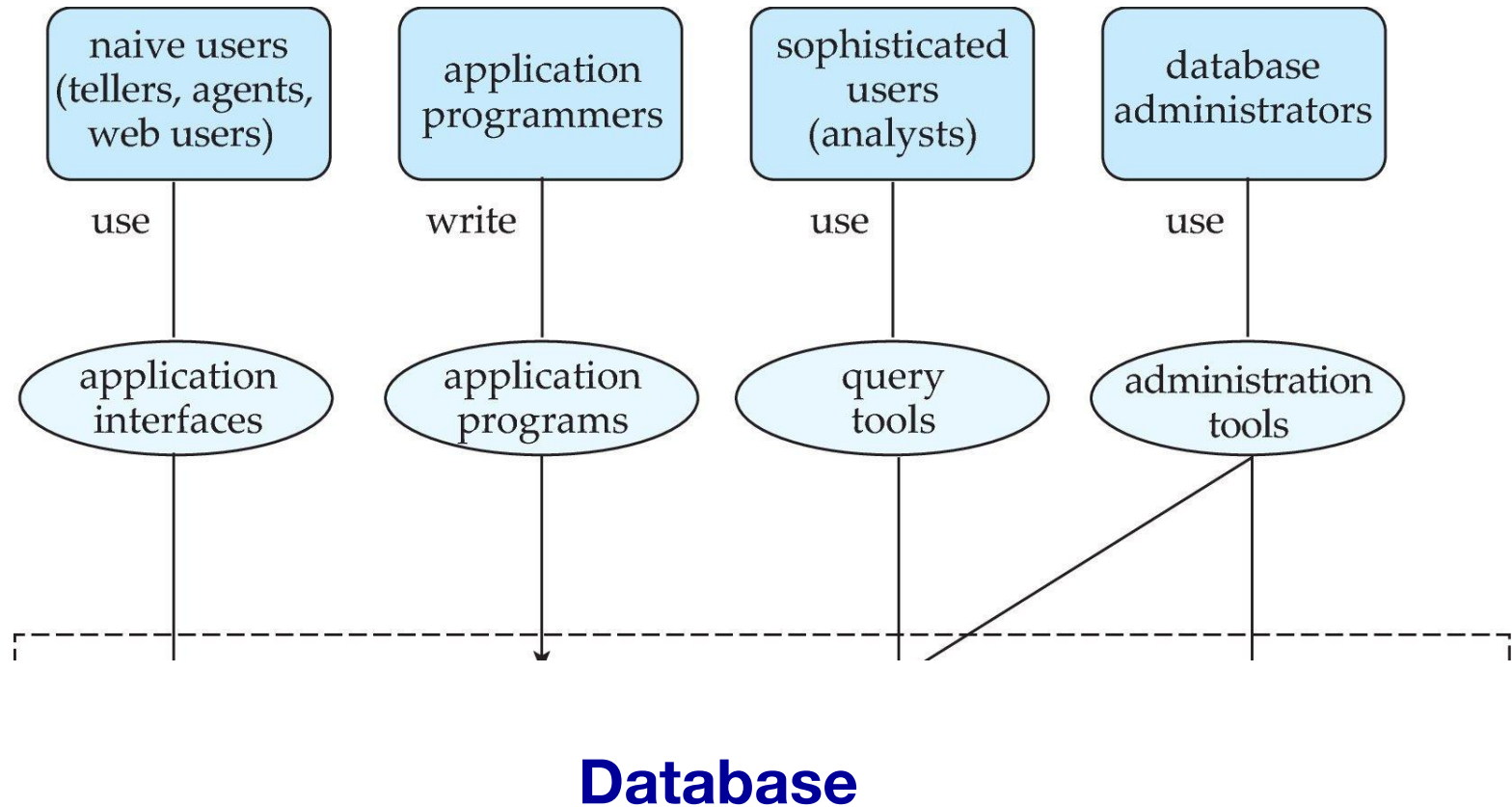
Database

Database System Internals
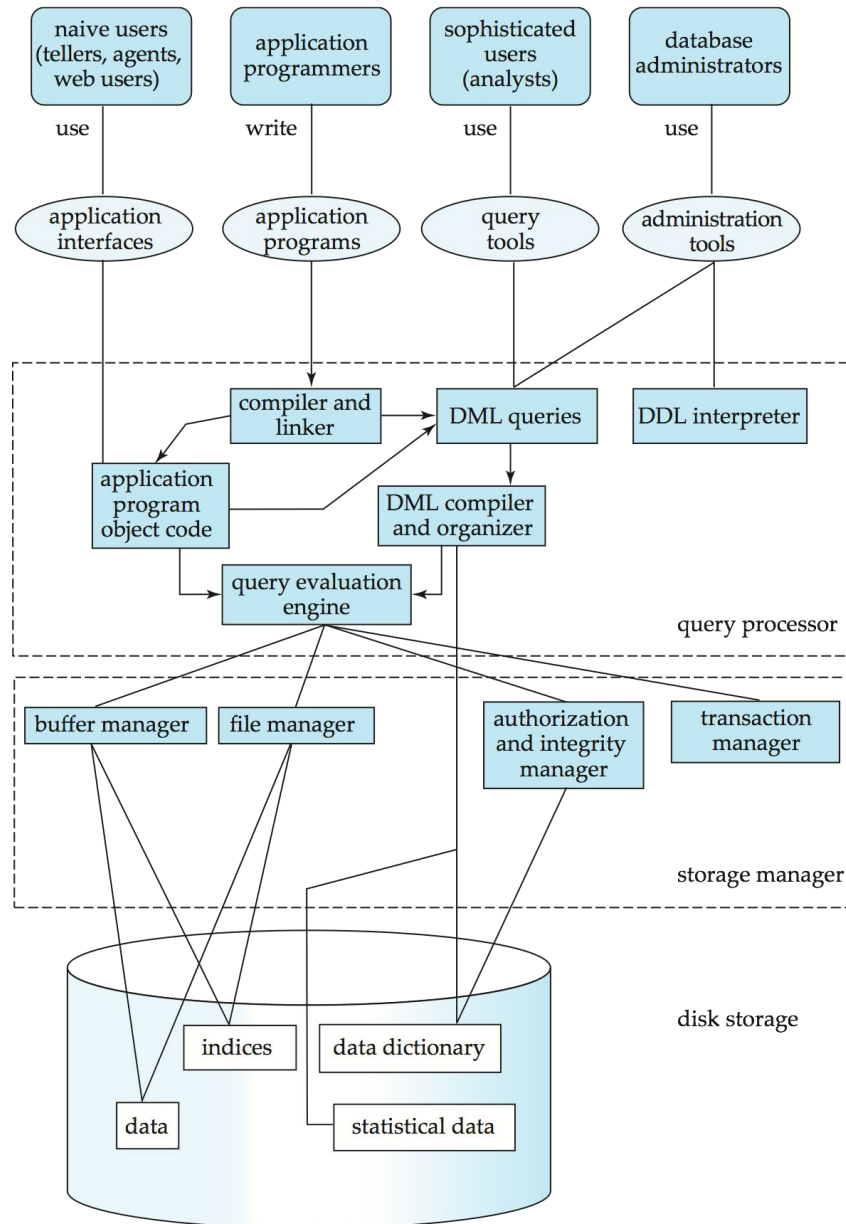
Database Architecture

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

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, ..

History of Database Systems
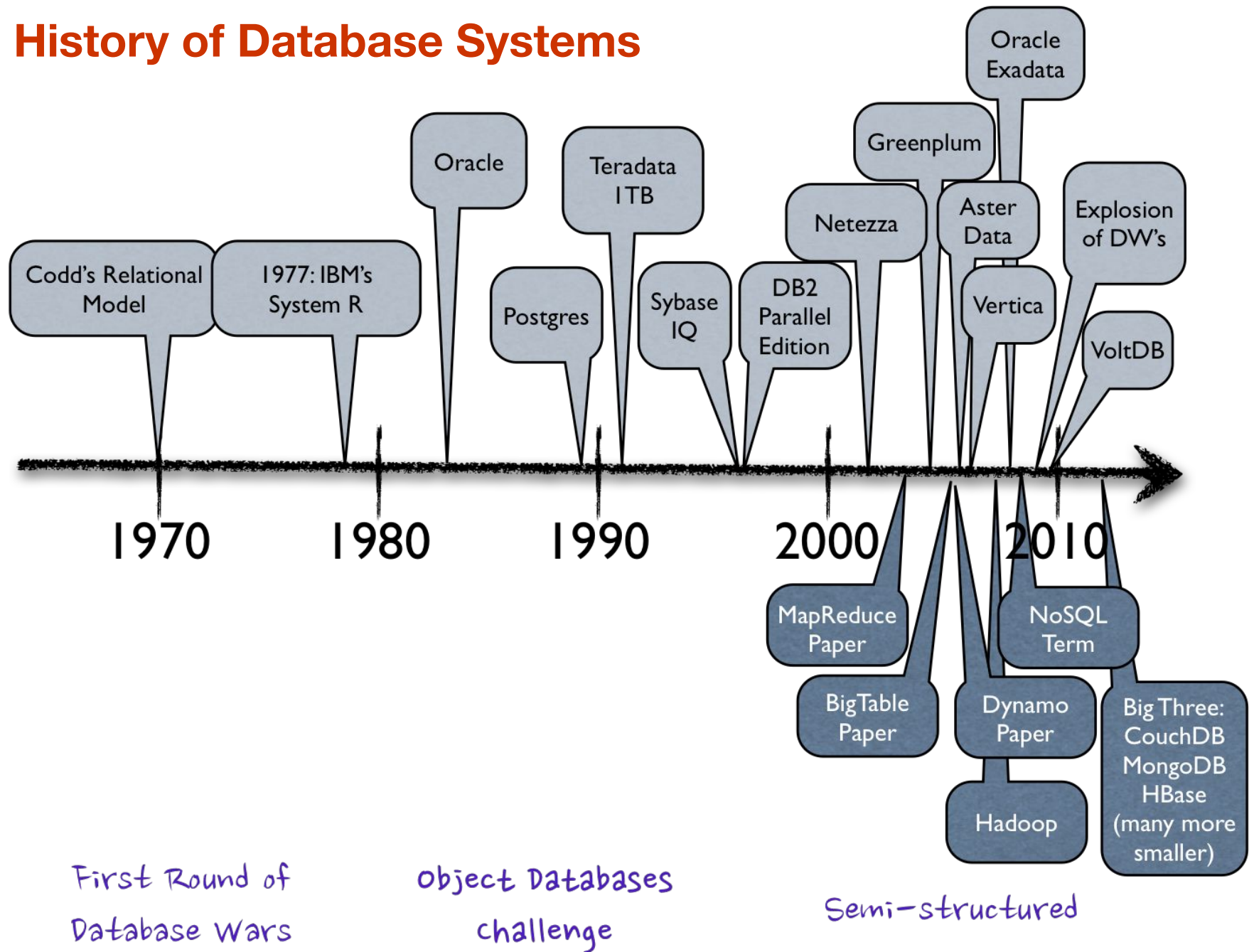
End of Chapter 1

Chapter 2: Intro to Relational Model

Database System Concepts, 6th Ed.

Father of the Relational Database: Edgar F. Codd

A British computer scientist, Codd made important contributions to the theory of relational databases. While working for IBM, he created the relational model for database management.
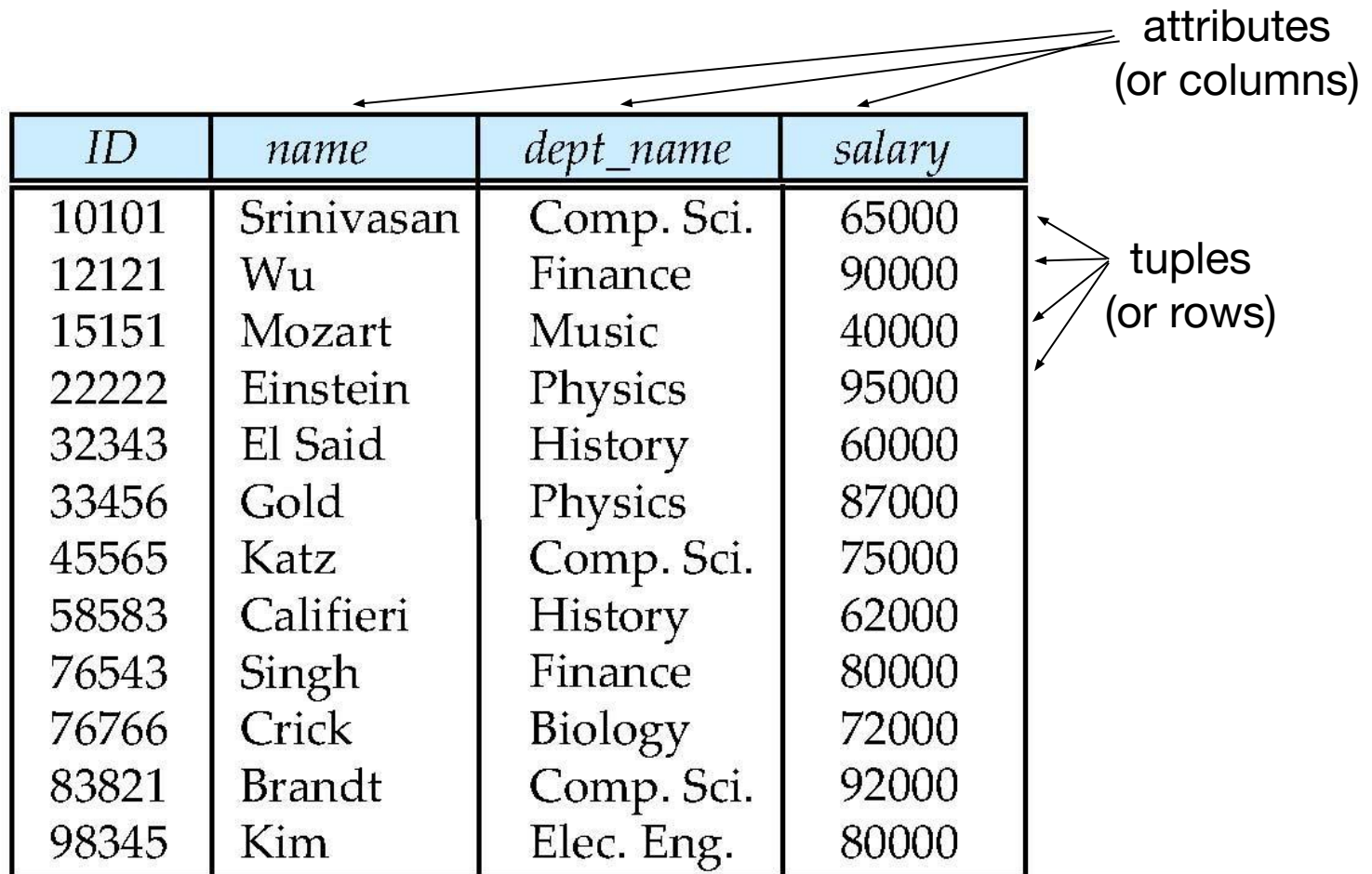
Exercise 1.1

Briefly explain 5 (five) reasons as to why you would use a DBMS for a university student registration system rather than a simple flat file.

Example of a Relation

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

attributes (or columns)

tuples (or rows)

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

- 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

Database

- A database consists of multiple relations

- Information about an enterprise is broken up into parts

  instructor
  student
  advisor

- Bad design:
  univ (instructor -ID, name, dept_name, salary, student_Id, ..)
results in

  - repetition of information (e.g., two students have the same instructor)

  - the need for null values (e.g., represent an student with no advisor)

- Normalization theory (Chapter 7) deals with how to design "good" relational schemas

Keys

- Let K ⊆ 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: Bank DB Schema
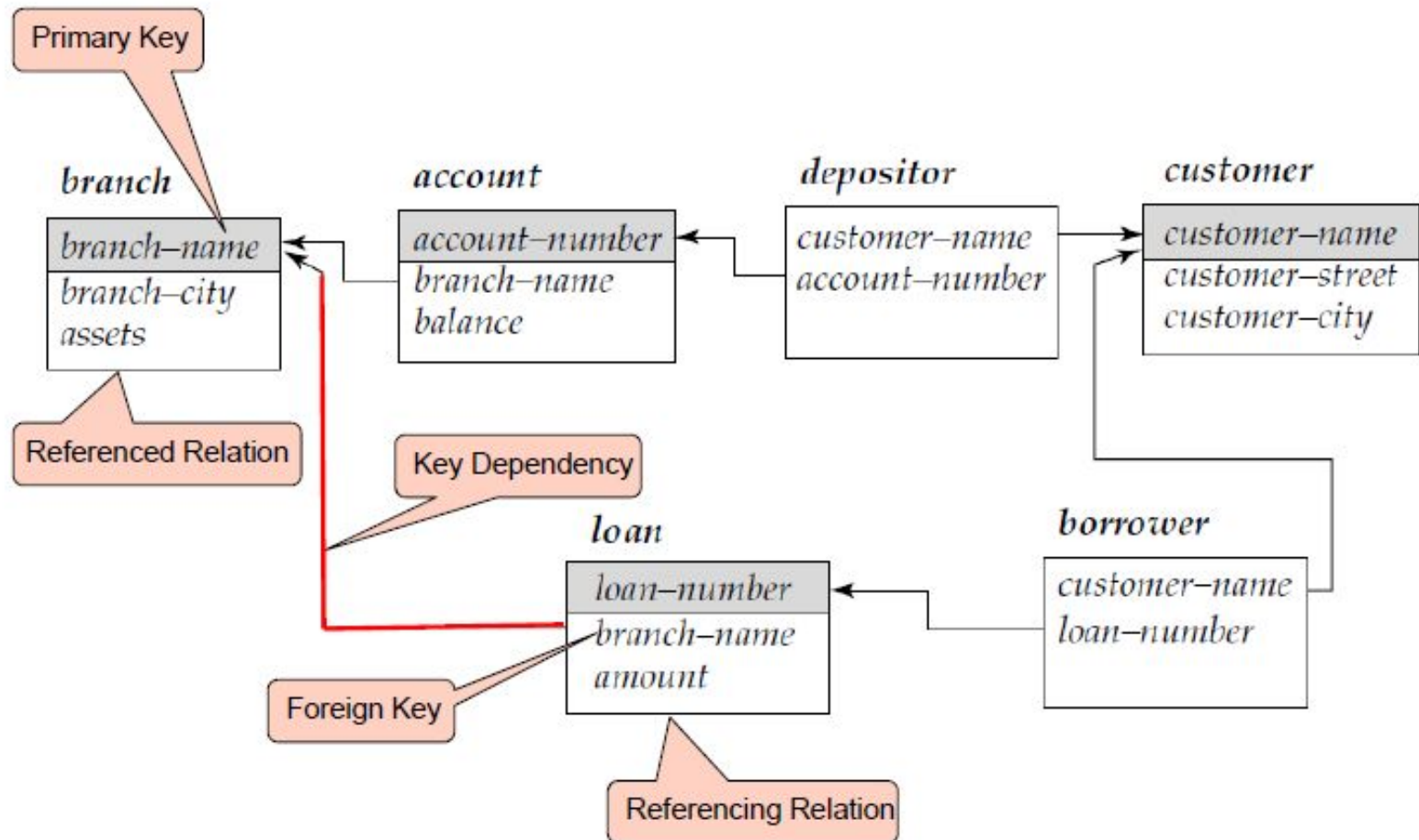
Schema Diagram for University Database

takes
- _ID_
- course_id
- sec_id
- semester
- year
- grade

student
- _ID_
- name
- dept_n
- tot_cre

section
- course_id
- sec_id
- semester
- year
- building
- room_no
- time_slot_id

course
- course_id
- title
- dept_name
- credits

department
- dept_n
- building
- budget

time_slot
- time_slot_id
- day
- start_time
- end_time

classroom
- building
- room_no
- capacity

prereq
- course_id
- prereq_id

instructor
- _ID_
- name
- dept
- sal

teaches
- _ID_
- course_id
- sec_id
- semester
- year

Exercise 1.2 What are appropriate Primary Keys ?

employee (person_name, street, city)
works (person_name, company_name, salary)
company (company_name, city)

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

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

course
course_id
title
dept_name
credits

prereq
course_id
prereq_id

Exercise 1.3 Give an example insert that would violate the foreign key constraint ?

Exercise 1.4 Give an example Delete that would violate the foreign key constraint ?

Exercise 1.5 From the Database Instance of Instructor shown no two instructors have the same name. From this observation can we conclude that name can be used as a superkey (primary key) of 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

Relational Query Languages

- Procedural vs.non-procedural, or declarative
- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- Relational operators
- Relational Algebra operators

  Six basic operators

  Select: σ

  Project: Π

  Union: $\cup$

  set difference: –

  Cartesian product: x

  Rename:

The operators take one or two relations as inputs and produce a new relation as a result.

Selection of tuples

- Relation r

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

Select tuples with A=B and D > 5

$$\sigma_{A=B \text{ and } D > 5}(r)$$

A	B	C	D
α	α	1	7
β	β	23	10

Selection of Columns (Attributes)

Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

Select A and C Projection

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

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Joining two relations – Cartesian Product

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

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*:

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

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r ∩ s*

A	B
α	2

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 ⋈ 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

Figure in-2.1

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.
$\bowtie$ (Natural Join)	*instructor* $\bowtie$ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian Product)	*instructor* $\times$ *department*
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	Π_{name}(*instructor*) $\cup$ Π_{name}(*student*)
	Output the union of tuples from the two input relations.

Exercise 1.6
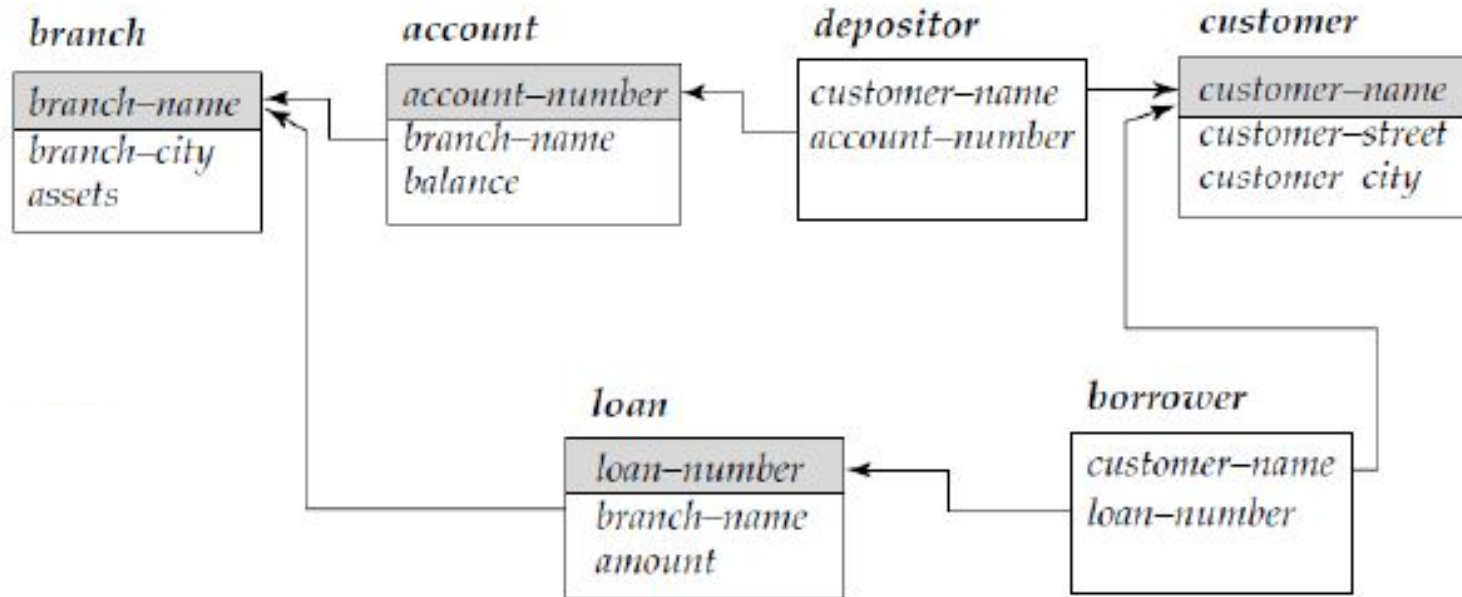
Consider the bank database of Figure ??. Give an expression in the relational algebra for each of the following queries.

a. Find the names of all branches located in "Chicago".

b. Find the names of all borrowers who have a loan in branch "Downtown".

Exercise 1.7

Employee(emp_no, emp_name, emp_city, …….)

Assignment(proj_no, emp_no, hours,……….)

Project(proj_name, budget, proj_no, proj_start_date, proj_end_date, proj_location,……..)

Express each of the following queries in Relational Algebra:

1. List the name(s) and budget(s) of projects started before 1st May 2008.
2. List the name(s) of projects with a budget value above Rs. 1,000,000.
3. Find the name(s) of employees who are from city "Moratuwa".
4. Find the name(s) of employees who are from city "Moratuwa" and work on projects located in "Moratuwa".
5. Find the name(s) of employees who work on projects valued above Rs. 1,000,000.

End of Chapter 2

Figure 2.01

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

Figure 2.02

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure 2.03

course_id	prereq_id
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Figure 2.04

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

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

Figure 2.05

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

Figure 2.06

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.07

ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Figure 2.10

ID	name	dept_name	salary
12121	Wu	Finance	90000
22222	Einstein	Physics	95000
33456	Gold	Physics	87000
83821	Brandt	Comp. Sci.	92000

Figure 2.11

ID	salary
10101	65000
12121	90000
15151	40000
22222	95000
32343	60000
33456	87000
45565	75000
58583	62000
76543	80000
76766	72000
83821	92000
98345	80000

Figure 2.12

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

Figure 2.13

ID	salary
12121	90000
22222	95000
33456	87000
83821	92000

Figure 1.02

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

Figure 1.04

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

Figure 1.06

user

application

client

network

database system

server

(a) Two-tier architecture

user

application client

network

application server

database system
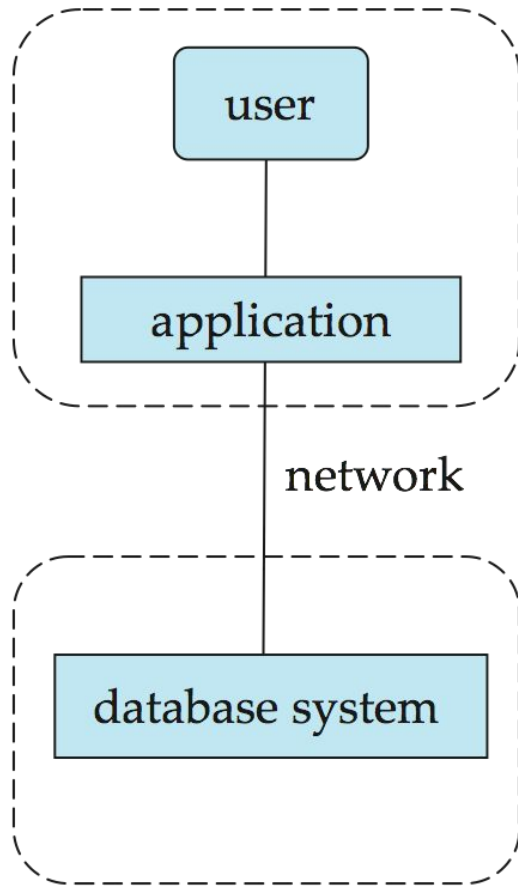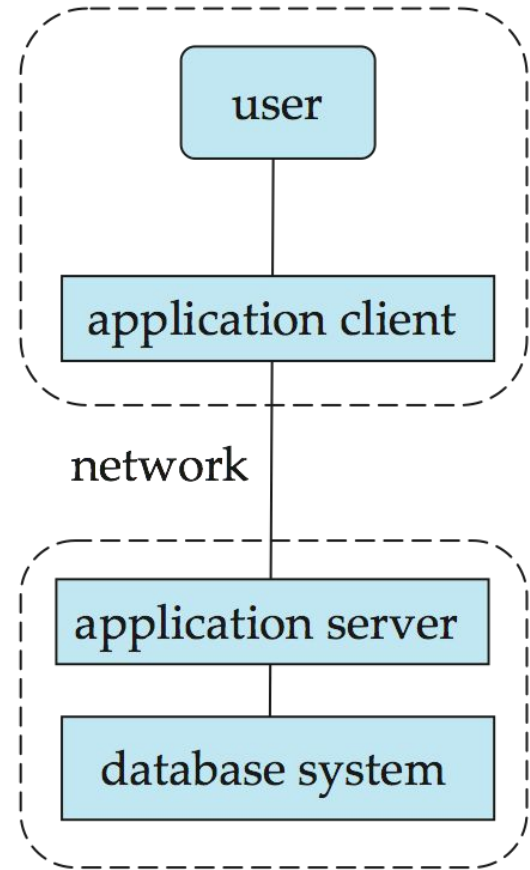
(b) Three-tier architecture