DATABASE SYSTEM
SUB.CODE:18K5CSELCS1:A

Mrs. S.Ramya M.Sc., M.Phil

Semester – V
MSE –I

Hours - 6
Credit – 5

Databases Systems (18KSCSELGS1:A)

Objective: To impart knowledge on Database concepts with an overview on SQL and Oracle.

UNIT I : Introduction : Database System Applications – Purpose of Database Systems – View of Data – Database Languages – Relational Databases – Database Design – Data Storage And Querying – Transaction Management – Database Architecture – Data Mining and Information Retrieval – Specialty Databases – Database User and Administrators History of Database Systems.

UNIT II : Relational Databases : Introduction to the Relational Model : Structure of Relational Databases – Database Schema – Keys – Schema Diagrams – Relational Query Languages – Relational Operations.

UNIT III : Introduction to SQL : Overview of the SQL Query Language – SQL Data Definition Basic Structure of SQL Queries – Additional Basic Operations – Set Operations – Null Values – Aggregate Functions – Nested Subqueries – Modification of the Database. Intermediate SQL : Join Expressions – Views - Transactions – Integrity Constraints – SQL Data Types and Schemas – Authorization.

UNIT IV : Database Design and E-R Model – Overview of the Design Process – The Entity-Relationship Model Constraints – Removing Redundant Attributes in Entity Sets – Entity-Relationship Diagrams. Relational Database Design: Features of Good Relational Design – Atomic Domains And First Normal Form – Decomposition using Functional dependencies.

UNIT V : Critical Database Concept : Oracle Sharing Knowledge and Success: The Cooperative approach – Familiar Language of Oracle. The dangers in a Relational DataBase – Codes, Abbreviation and Naming Standards – Normalizing Names – Understanding the Data Basic Parts of Speech in SQL : Style – Logic and Value – Combining Logic.

TEXT: Unit I - IV

"Database System Concepts" – Abraham Silberschatz, Henry F.Korth, S.Sudarshan– McGraw Hill Education (India) Pvt. Ltd., – Sixth Edition - Second Reprint -2013.

Chapters : 1, 2, 3, 4, 7.1 - 7.5, 8.1 - 8.3.

Unit – V

"Oracle 9i – The Complete Reference" – Kevin Loney, George Koch And the Experts at TUSC – Tata McGraw-Hill Edition – 2002.

Chapters: 1 – 3 (Relevent Topics Only).

Reference:

1. "Database Management System" – Ramakrishnam – McGraw – Hill Higher Education - Third Edition - 2002.
2. "Database Management System the Complete Book"- Jennifer widom,Jeffrey D. Ullman,Hector Garcia –molina Pearson - First Edition - 2003.
3. "Database Management System" - G.K.Gupta – Tata Mcgraw Hill Education Private Limited - First Edition - 2011.

Head, Dept of Computer Science
Kunthavai Naachiaar Govt Arts College (W)
THANJAVUR- 7.

Database System
Unit :I Chapter :1 Introduction

A database-management system (DBMS)

- It is a collection of interrelated data and a set of programs to access those data.
- The collection of data, usually referred to as the database, contains information relevant to an enterprise.
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

Database systems are designed to manage large bodies of information.

- Mangagment of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- In addition, the database system 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.
- Information is so important in most organizations,
- computer scientists have developed a large body of concepts and techniques for managing data.

1.1 Database-System Applications

- Databases are widely used. Here are some representative applications:

Enterprise Information

- *Sales*: For customer, product, and purchase information.
- *Accounting*: For payments, receipts, account balances, assets and other accounting information.
- *Human resources*: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- *Manufacturing*: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- *Online retailers*: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Banking and Finance

- *Banking*: For customer information, accounts, loans, and banking transactions.
- *Credit card transactions*: For purchases on credit cards and generation of monthly statements.
- *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.

- *Universities*: For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- *Airlines*: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- Databases form an essential part of every enterprise today, storing not only types of information that are common to most enterprises.
- In the early days, very few people interacted directly with database systems, although without realizing it, they interacted with databases indirectly— through printed reports
- such as

1. credit card statements or
2. Through agents such as bank tellers
3. Airline reservation agents

- Then auto- mated teller machines came along and let users interact directly with databases.
- Phone interfaces to computers (interactive voice-response systems) also allowed users to deal directly with databases— a caller could dial a number, and press phone keys to enter information or to select alternative options, to find flight arrival The Internet revolution of the late 1990s sharply increased direct user access to databases.
- Organizations converted many of their phone interfaces to databases into Web interfaces, and made a variety of services and information available online.
- For instance, when you access an online bookstore and browse a book or music collection, you are accessing data stored in a database al/departure times, for example, or to register for courses in a university.

- When you access a bank Web site and retrieve your bank balance and transaction information, the information is retrieved from the bank's database system
- The importance of database systems can be judged in another way— today, database system vendors like Oracle are among the largest software companies in the world, and database systems form an important part of the product line of Microsoft and IBM.

Purpose of Database Systems

- Database systems arose in response to early methods of computerized management of commercial data.
- As an example of such methods, typical of the 1960s, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings.
- One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

1. Add new students, instructors, and courses
2. Register students for courses and generate class rosters
3. Assign grades to students, compute grade point averages (GPA), and generate transcripts

- System programmers wrote these application programs to meet the needs of the university.
- New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major (say, computer science).
- As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, etc.

- New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.
- This typical file-processing system is supported by a conventional operating system.
- The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.
- Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.
- Keeping organizational information in a file-processing system has a number of major disadvantages:

File-processing system has a number of major disadvantages:

1.Data redundancy and inconsistency:

- Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages.
- Moreover, the same information may be duplicated in several places (files).
- For example, if a student has a double major (say, music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department.

Difficulty in accessing data.

- Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area.
- The clerk asks the data-processing department to generate such a list.
- Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it.
- There is, however, an application program to generate the list of *all* students.
- The university clerk has now two choices:

either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program.

Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours.

As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory.

- The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

1. Data isolation. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
2. Integrity problems. The data values stored in the database must satisfy certain types of consistency constraints. Suppose the university maintains an account for each department, and records the balance amount in each ac- count.

Atomicity problems.

- A computer system, like any other device, is subject to failure.
- In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.
- Consider a program to transfer $500 from the account balance of department *A* to the account balance of department *B*.
- A($500) ----- B ($500)

- If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of department *A*
- but was not credited to the balance of department *B*,
- A($0) ----- B ($0)

- resulting in an inconsistent database state.
- Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur.
- That is, the funds transfer must be *atomic* — it must happen in its entirety or not at all.
- It is difficult to ensure atomicity in a conventional file-processing system.

Concurrent-access anomalies.

- For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.
- Indeed, today, the largest Internet retailers may have millions of accesses per day to their data by shoppers.
- In such an environment, interaction of concurrent updates is possible and may result in inconsistent data.
- Consider department *A*, with an account balance of $10,000.
- If two department clerks debit the account balance (by say $500 and $100, respectively) of department *A* at almost exactly the same time
- , the result of the concurrent executions may leave the budget in an incorrect (or inconsistent) state.

- Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back.
- If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively.
- Depending on which one writes the value last, the account balance of department *A* may contain either $9500 or $9900, rather than the correct value of $9400
- . To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

- Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively.
- Depending on which one writes the value last, the account balance of department *A* may contain either $9500 or $9900, rather than the correct value of $9400. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

Security problems.

- Not every user of the database system should be able to access all the data.
- For example, in a university, payroll personnel need to see only that part of the database that has financial information.
- They do not need access to information about academic records.
- But, since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.
- These difficulties, among others, prompted the development of database systems.
- In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems.
- In most of this book, we use a university organization as a running example of a typical data-processing application.

1.4.1 Data-Manipulation Language

- A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. 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:
    - Procedural DMLs require a user to specify *what* data are needed and *how* to get those data.
    - Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify *what* data are needed *without* specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.
- A query is a statement requesting the retrieval of information.
- The portion of a DML that involves information retrieval is called a query language.
- Although technically incorrect, it is common practice to use the terms *query language* and *data-manipulation language* synonymously.

- There are a number of database query languages in use, either commercially or experimentally
- The levels of abstraction that we discussed in Section 1.3 apply not only to defining or structuring data,
- but also to manipulating data.
- At the physical level, we must define algorithms that allow efficient access to data.
- At higher levels of abstraction, we emphasize ease of use. The goal is to allow humans to interact efficiently with the system.

1.4.2 Data-Definition Language

- We specify a database schema by a set of definitions expressed by a special language called a data-definition language (DDL). The DDL is also used to specify additional properties of the data.
- We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language.
- These statements define the implementation details of the database schemas, which are usually hidden from the users.
- The data values stored in the database must satisfy certain consistency constraints.
- For example, suppose the university requires that the account balance of a department must never be negative.
- The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated.
- In general, a constraint can be an arbitrary predicate pertaining to the database.
- However, arbitrary predicates may be costly to test. Thus, database systems implement integrity constraints that can be tested with minimal overhead:

1.Domain Constraints.

- A domain of possible values must be associated with every attribute
- (for example, integer types, character types, date/time types).
- Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take.
- Domain constraints are the most elementary form of integrity constraint.
- They are tested easily by the system whenever a new data item is entered into the database.

2. Referential Integrity

- There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in
- For example, the department listed for each course must be one that actually exists.
- More precisely, the *dept name* value in a *course* record must appear in the *dept name* attribute of some record of the *department* relation.
- Database modifications can cause violations of referential integrity. When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

3.Assertions.

- An assertion is any condition that the database must always satisfy.
- Domain constraints and referential-integrity constraints are special forms of assertions.
- However, there are many constraints that we cannot express by using only these special forms.
- For example, “Every department must have at least five courses offered every semester” must be expressed as an assertion.
- When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not cause that assertion to be violated.

4.Authorization.

- We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database.
- These differentiations are expressed in terms of authorization
- The most common being:

a. read authorization: which allows reading, but not modification, of data;

b. insert authorization: which allows insertion of new data, but not modification of existing data;

c. update authorization: which allows modification, but not deletion, of data;

d. delete authorization: which allows deletion of data. We may assign the user all, none, or a combination of these types of authorization.

- The DDL, just like any other programming language, gets as input some instructions (statements) and generates some output.
- The output of the DDL is placed in the data dictionary, which contains metadata— that is, data about data. The data dictionary is considered to be a special type of table that can only be accessed and updated by the database system itself (not a regular user).
- The database system consults the data dictionary before reading or modifying actual data.

1.5 Relational Databases

- A relational database is based on the relational model and uses a collection of tables to represent both data and the relationships among those data.
- It also includes a DML and DDL. Most commercial relational database systems employ the SQL language.

1.5.1 Tables

- Each table has multiple columns and each column has a unique name. Figure 1.2 presents a sample relational database comprising two tables: one shows details of university instructors and the other shows details of the various university departments.
- The first table, the *instructor* table, shows, for example, that an instructor named Einstein with *ID* 22222 is a member of the Physics department and has an annual salary of $95,000.
- The second table, *department*, shows, for example, that the Biology department is located in the Watson building and has a budget of
- $90,000. Of course, a real-world university would have many more departments and instructors. We use small tables in the text to illustrate concepts. A larger example for the same schema is available online.

Instructor Table

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

Department 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

- The relational model is an example of a record-based model.
- Record-based models are so named because the database is structured in fixed-format records of several types.
- Each table contains records of a particular type.
- Each record type defines a fixed number of fields, or attributes.
- The columns of the table correspond to the attributes of the record type.
- It is not hard to see how tables may be stored in files.
- For instance, a special character (such as a comma) may be used to delimit the different attributes of a record,
- another special character (such as a new-line character) may be used to delimit records.
- The relational model hides such low-level implementation details from database developers and users.
- We also note that it is possible to create schemas in the relational model that have problems such as unnecessarily duplicated information.
- For example, suppose we store the department *budget* as an attribute of the *instructor* record.
- Then, whenever the value of a particular budget (say that one for the Physics department) changes, that change must to be reflected in the records of all instructors

- SQL provides a rich DDL that allows one to define tables, integrity constraints, assertions, etc.

1.5.3Data-Definition Language

- For instance, the following SQL DDL statement defines the *department* table:

- Execution of the above DDL statement creates the *department* table with three columns: *dept name*, *building*, and *budget*, each of which has a specific data type associated with it

1.5.2 Data-Manipulation Language

- The SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table. Here is an example of an SQL query that finds the names of all
- instructors in the History department:
- The query specifies that those rows from the table *instructor* where the *dept name* is History must be retrieved, and the *name* attribute of these rows must be displayed labled name, and a set of rows, each of which contains the name of an instructor whose *dept name*, is History. If the query is run on the table in Figure 1.2, the result will consist of two rows, one with the name El Said and the other with the name Califieri.

- Queries may involve information from more than one table. For instance, the following query finds the instructor ID and department name of all instructors associated with a department with budget of greater than $95,000. If the above query were run on the tables in Figure 1.2,

```
select instructor.ID, department.dept name
from instructor, department
where instructor.dept name= department.dept name and department.budget > 95000;
```

ID	Depttname
45565	Comp.Sci
10101	Comp.Sci
83821	Comp.Sci
12121	Finance

- ,

```
select instructor. name, department.dept name
from instructor, department
where instructor.dept name= department.dept name and department.budget <50000;
```

-

name	Depttname
katz	Comp.Sci
Srinivasan	Comp.Sci
Brandt	Comp.Sci
Wu	Finance
Crick	Biology
Kim	Electrical Eng
Mozart	Music
Einstein	Physics
Gold	physics

- The system would find that there are two departments with budget of greater than $95,000 — Computer Science and Finance; there are five instructors in these departments.
- Thus, the result will consist of a table with two columns (*ID*, *dept name*) and five rows:
- (12121, Finance), (45565, Computer Science), (10101, Computer Science), (83821, Computer Science), and (76543, Finance).

1.5.4 Database Access from Application Programs

- SQL is not as powerful as a universal Turing machine;
- that is, there are some computations that are possible using a general-purpose programming language but are not possible using SQL.
- SQL also does not support actions such as input from users,
- output to displays, or communication over the network. Such computations and actions must be written in a *host* language, such as C, C++, or Java, with embedded SQL queries that access the data in the database.
- Application programs are programs that are used to interact with the database in this fashion.
- By providing an application program interface (set of procedures) that can be used to send DML and DDL statements to the database and retrieve the results.
- The Open Database Connectivity (ODBC) standard for use with the C language is a commonly used application program interface standard.

1.6 Database Design

- Database systems are designed to manage large bodies of information.
- These large bodies of information do not exist in isolation.
- They are part of the operation of some enterprise whose end product may be information from the database or may be some device or service for which the database plays only a supporting role.
- Database design mainly involves the design of the database schema.
- The design of a complete database application environment that meets the needs of the enterprise being modeled requires attention to a broader set of issues
- 1.6.1 Design Process
- A high-level data model provides the database designer with a conceptual framework in which to specify the data requirements of the database users, and how the database will be structured to fulfill these requirements.
- The initial phase of database design, then, is to characterize fully the data needs of the prospective database users.
- The database designer needs to interact extensively with domain experts and users to carry out this task.
- The outcome of this phase is a specification of user requirements

- Next, the designer chooses a data model, and by applying the concepts of the chosen data model,
- translates these requirements into a conceptual schema of the database.
- The schema developed at this conceptual-design phase provides a detailed overview of the enterprise.
- The designer reviews the schema to confirm that all data requirements are indeed satisfied and are not in conflict with one another. The designer can also examine the design to remove any redundant features.
- The focus at this point is on describing the data and their relationships, rather than on specifying physical storage details.
- In terms of the relational model, the conceptual-design process involves decisions on *what* attributes we want to capture in the database

  and *how to group* these attributes to form the various tables.
- The "what" part is basically a business decision,
- The "how" part is mainly a computer-science problem. There are principally two ways to tackle the problem.
- The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.
- In the logical-design phase, the designer maps the high-level conceptual schema onto the implementation data model of the database system that will be used.
- The designer uses the resulting system-specific database schema in the subsequent physical-design phase, in which the physical features of the database are specified.

1.6.2 Database Design for a University Organization

- To illustrate the design process, let us examine how a database for a university could be designed.
- The initial specification of user requirements may be based on interviews with the database users, and on the designer's own analysis of the organization.
- The description that arises from this design phase serves as the basis for specifying the conceptual structure of the database.
- Here are the major characteristics of the university.
- The university is organized into departments.
- Each department is identified by a unique name (*dept name*), is located in a particular *building*, and has a *budget*

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

- The university maintains a list of all classes (sections) taught. Each section is identified by a *course id*, *sec id*, *year*, and *semester*, and has associated with it a *semester*, *year*, *building*, *room number*, and *time slot id* (the time slot when the class meets.
- The department has a list of teaching assignments specifying, for each instructor, the sections the instructor is teaching.
- The university has a list of all student course registrations, specifying, for each student, the courses and the associated sections that the student has taken (registered for).

1.6.3The Entity-Relationship Model

- The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects.
- An entity is a “thing” or “object” in the real world that is distinguishable from other objects.
- For example, each person is an entity, and bank accounts can be considered as entities
- Entities are described in a database by a set of attributes. For example, the attributes *dept name*, *building*, and *budget* may describe one particular department in a university, and they form attributes of the *department* entity set. Similarly, attributes *ID*, *name*, and *salary* may describe an *instructor* entity
- The overall logical structure (schema) of a database can be expressed graphically by an *entity-relationship (E-R) diagram*. There are several ways in which to draw these diagrams. One of the most popular is to use the Unified Modeling Language (UML). In the notation we use, which is based on UML, an E-R diagram is represented as follows:

1.6.4 Normalization

- Another method for designing a relational database is to use a process commonly known as normalization.
- The goal is to generate a set of relation schemas that allows us to store information without unnecessary redundancy,
- yet also allows us to retrieve information easily. The approach is to design schemas that are in an appropriate *normal form*.
- To determine whether a relation schema is in one of the desirable normal forms, we need additional information about the real-world enterprise that we are modeling with the database.

1.7 Data Storage and Querying

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the

❑ storage manager and

❑ The query processor components.

- The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases,
- Terabytes of data.

- A gigabyte is approximately 1000 megabytes (actually 1024) (1 billion bytes),
- and a terabyte is 1 million megabytes (1 trillion bytes).
- Since the main memory of computers cannot store this much information, the information is stored on disks.
- Data are moved between disk storage and main memory as needed.
- Since the movement of data to and from disk is slow relative to the speed of the central processing unit,
- it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory.
- The query processor is important because it helps the database system to simplify and facilitate access to data.
- The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level.
- It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level,
- into an efficient sequence of operations at the physical level.

1.7.1 Storage Manager

- The *storage manager* is the component of a database system 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 for the interaction with the file manager.
- The raw data are stored on the disk using the file system provided by the operating system.
- The storage manager translates the various DML statements into low-level file-system commands.
- Thus, the storage manager is responsible for

❑ storing,

❑ retrieving, and

❑ updating data in the database.

- The storage manager components include:
    - Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
    - Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
    - File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

- The storage manager implements several data structures as part of the physical system implementation:
    - Data files, which store the database itself.
    - Data dictionary, which stores metadata about the structure of the database, in particular the schema of the database.
    - Indices, which can provide fast access to data items.

      Like the index in this textbook, a database index provides pointers to those data items that hold a particular value.

      For example, we could use an index to find the *instructor* record with a particular *ID*, or all *instructor* records with a particular *name*. Hashing is an alternative to indexing that is faster in some but not all cases.
- 1.7.2 The Query Processor

- The query processor components include:
    - DDL interpreter, which interprets DDL statements and records the definitions in the data dictionary.
    - DML compiler, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization; that is, it picks the lowest cost evaluation plan from among the alternatives.
- Query evaluation engine, which executes low-level instructions generated by the DML compiler.

1.8 Transaction Management

Often, several operations on the database form a single logical unit of work. An example is a funds transfer, as in Section 1.2, in which one department account (say *A*) is debited and another department account (say *B*) is credited.

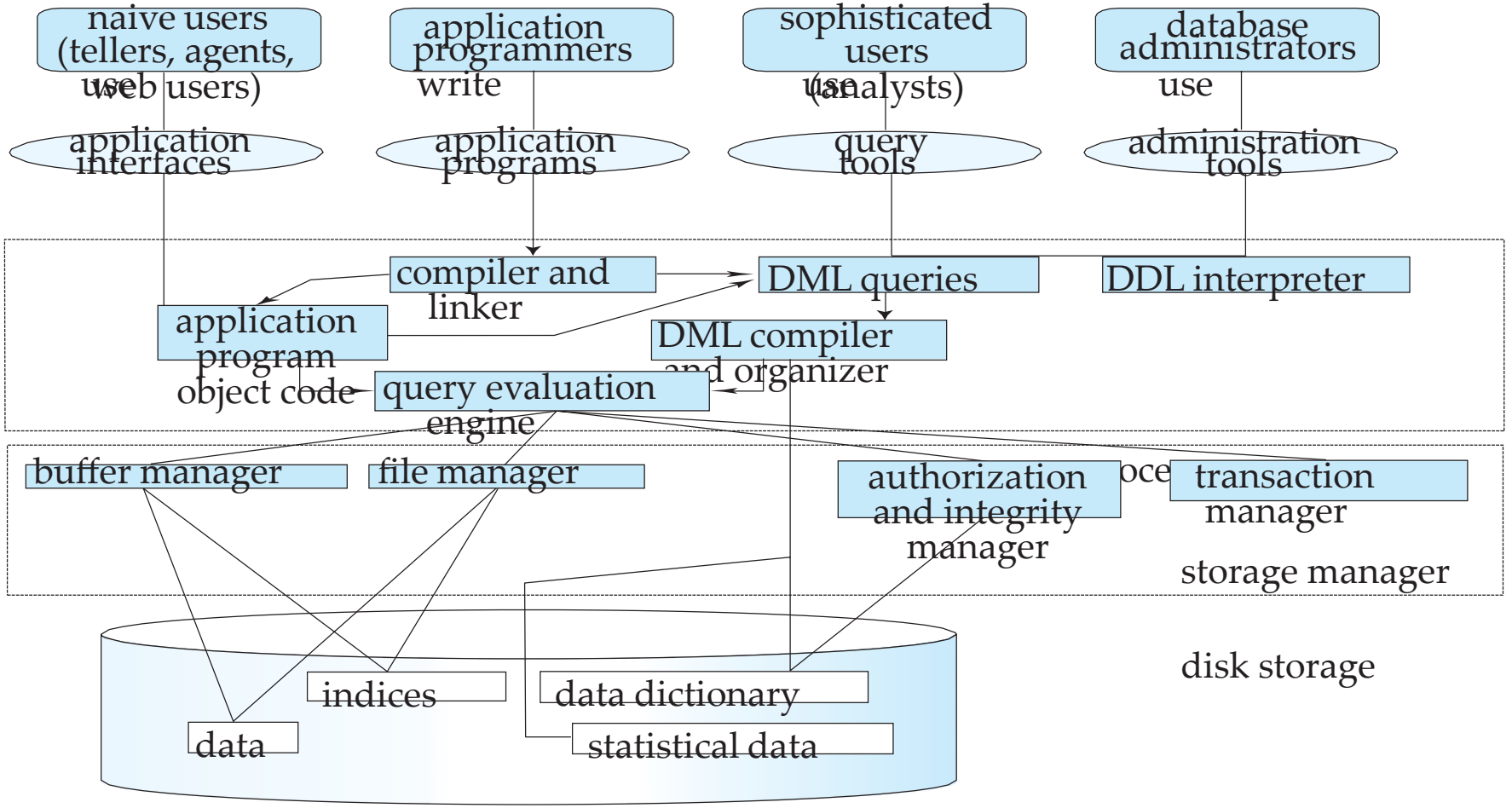
- Clearly, it is essential that either both the credit and debit occur, or that neither occur. That is, the funds transfer must happen in its entirety or not at all.
- This all-or-none requirement is called atomicity. In addition, it is essential that the execution of the funds transfer preserve the consistency of the database.
- That is, the value of the sum of the balances of *A* and *B* must be preserved. This correctness requirement is called consistency.
- Finally, after the successful execution of a funds transfer, the new values of the balances of accounts *A* and *B* must persist, despite the possibility of system failure. This persistence requirement is called durability.

- A transaction is a collection of operations that performs a single logical function in a database application.
- Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database- consistency constraints.
- That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.
- However, during the execution of a transaction, it may be necessary temporarily to allow inconsistency, since either the debit of *A* or the credit of *B* must be done before the other.
- This temporary inconsistency, although necessary, may lead to difficulty if a failure occurs.

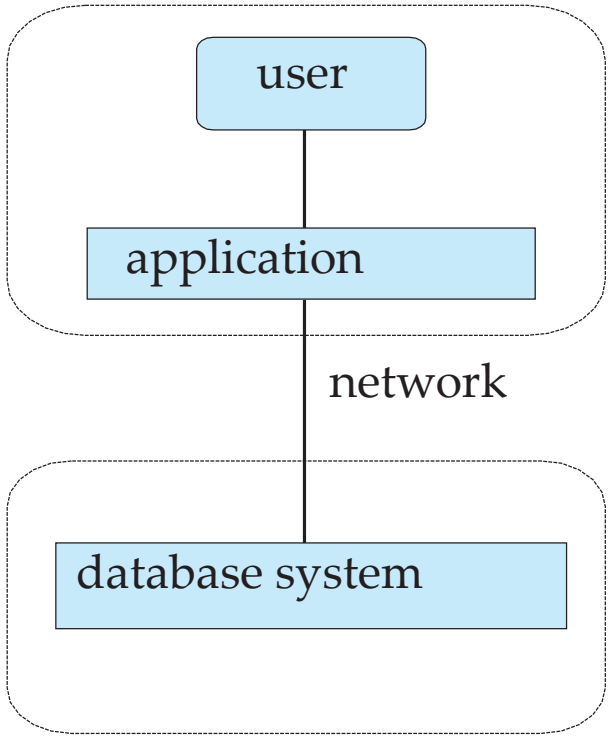
- It is the programmer's responsibility to define properly the various transactions, so that each preserves the consistency of the database.
- For example, the transaction to transfer funds from the account of department *A* to the account of department *B* could be defined to be composed of two separate programs: one that debits account *A*, and another that credits account *B*.
- The execution of these two programs one after the other will indeed preserve consistency.
- However, each program by itself does not transform the database from a consistent state to a new consistent state. Thus, those programs are not transactions.
- Ensuring the atomicity and durability properties is the responsibility of the database system itself— specifically, of the recovery manager. In the absence of failures, all transactions complete successfully, and atomicity is achieved easily.
- However, because of various types of failure, a transaction may not always complete its execution successfully.
- If we are to ensure the atomicity property, a failed transaction must have no effect on the state of the database. Thus, the database must be restored to the state in which it was before the transaction in question started executing.
- The database system must therefore perform failure recovery, that is, detect system failures and restore the database to the state that existed prior to the occurrence of the failure.
- Finally, when several transactions update the database concurrently, the consistency of data may no longer be preserved, even though each individual transaction is correct.
- It is the responsibility of the concurrency-control manager to control the interaction among the concurrent transactions, to ensure the consistency of the database. The transaction manager consists of the concurrency-control manager and the recovery manager.

1.9 Database Architecture

- We are now in a position to provide a single picture (Figure 1.5) of the various components of a database system and the connections among them.
- The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs.
- Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines.
- Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.
- The issues include how to store data, how to ensure atomicity of transactions that execute at multiple sites, how to perform concurrency control, and how to provide high availability in the presence of failures. Distributed query processing and directory systems are also described in this chapter.
- Most users of a database system today are not present at the site of the database system, but connect to it through a network.
- We can therefore differentiate between client machines, on which remote database users work, and server machines, on which the database system runs.


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
indices
data dictionary
data
statistical data
disk storage

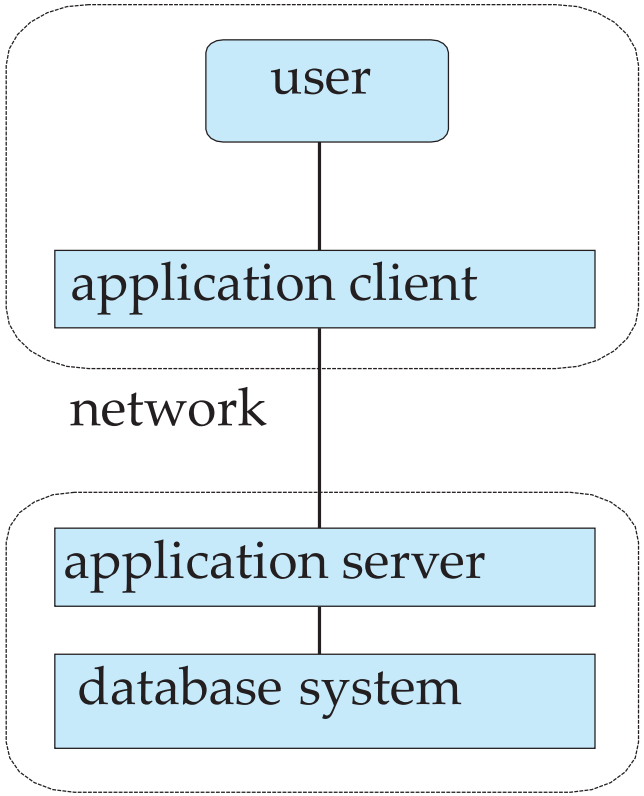
Two tier Architecture



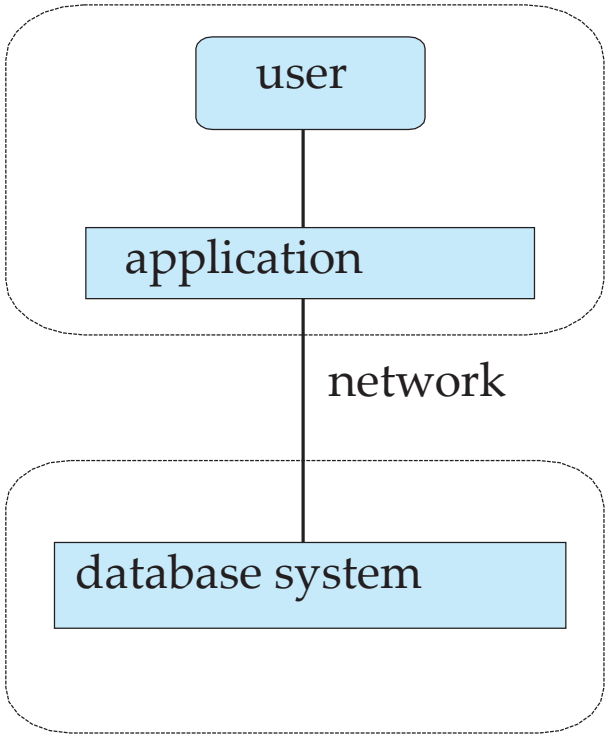
client

server

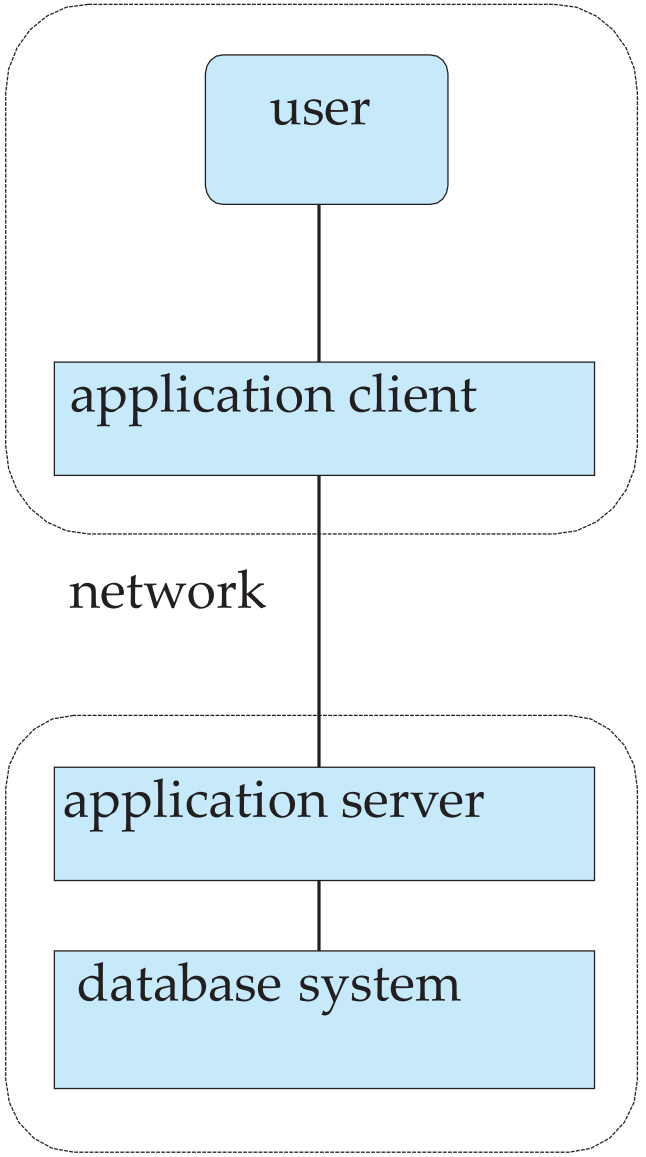
Three tier Architecture



- Database applications are usually partitioned into two or three parts, In a two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC
- and JDBC are used for interaction between the client and the server.



- In contrast, in a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls.
- Instead, the client end communicates with an application server, usually through a forms interface.
- The application server in turn communicates with a database system to access data.
- The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients.
- Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web



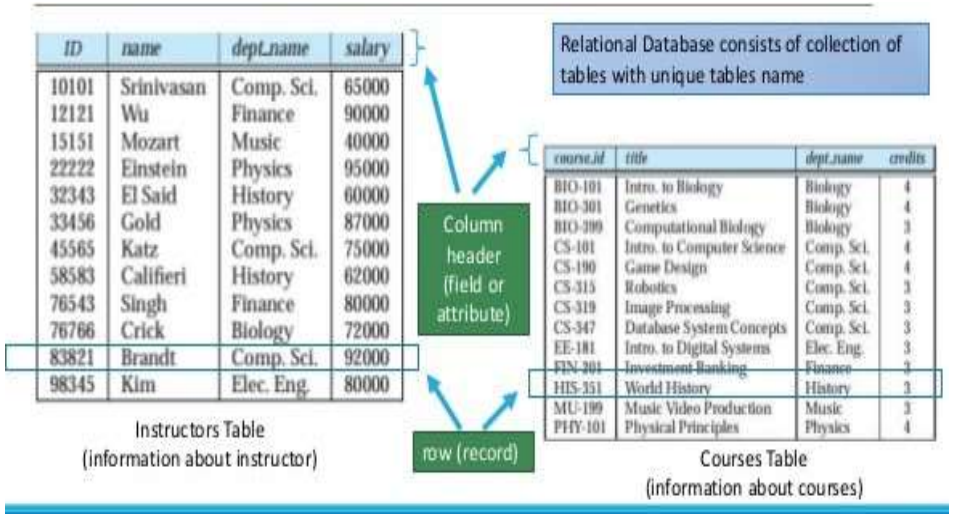
1.10 Data Mining and Information Retrieval

- The term data mining refers loosely to the process of semi automatically analyzing large databases to find useful patterns.
- Like knowledge discovery in artificial intelligence (also called machine learning) or statistical analysis, data mining attempts to discover rules and patterns from data.
- However, data mining differs from machine learning and statistics in that it deals with large volumes of data, stored primarily on disk.
- That is, data mining deals with “knowledge discovery in database.
- Some types of knowledge discovered from a database can be represented by a set of rules.
- The following is an example of a rule, stated informally: “Young women with annual incomes greater than $50,000 are the most likely people to buy small sports cars.”
- Of course such rules are not universally true, but rather have degrees of “support” and “confidence.” Other types of knowledge are represented by equations relating different variables to each other, or by other mechanisms for predicting outcomes when the values of some variables are known.
- There are a variety of possible types of patterns that may be useful, and different techniques are used to find different types of patterns.
- Usually there is a manual component to data mining, consisting of
- preprocessing data to a form acceptable to the algorithms,
- post processing of discovered patterns to find novel ones that could be useful.
- There may also be more than one type of pattern that can be discovered from a given database, and manual interaction may be needed to pick useful types of patterns.
- For this reason, data mining is really a semiautomatic process in real life. However, in our description we concentrate on the automatic aspect of mining.

UNIT II

Structure of Relational Databases

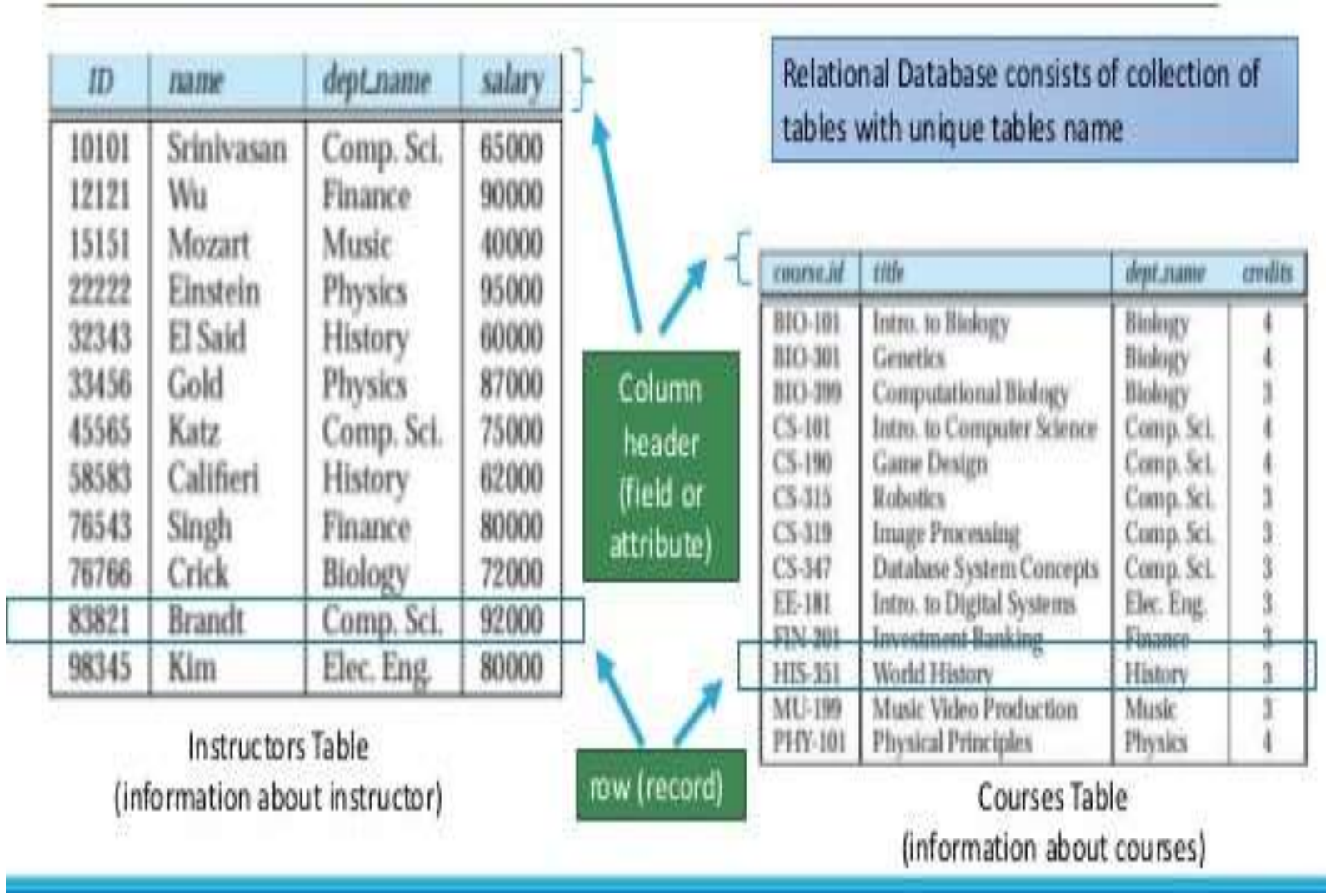
- A relational database is nothing but a collection of tables, in which each table assigned with a unique name.
- For example, instructor, course, department



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

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

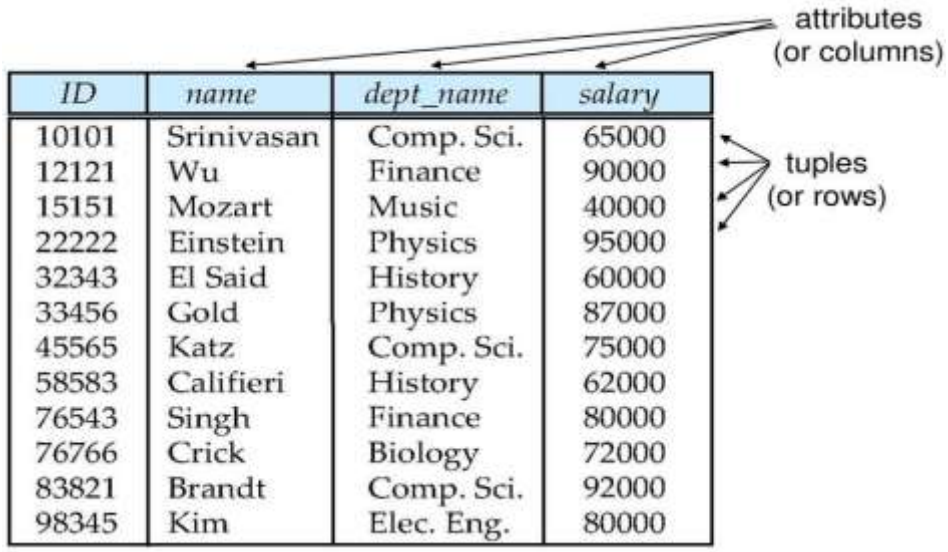
- consider the instructor_table, which holds the information about instructors.
- Instructor table has four attributes namely ID, name, dept_name, and salary.
- Each row of this table stores information about an instructor, such as instructor’s ID, name, dept_name, and salary.
- Similarly, the course table stores information about courses, such as course_id, title, dept name, and credits, for each course.



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

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

- A tuple is known as a sequence or list of values stored in a relation.
- A relationship between n values is represented by an n-tuple of values,
- i.e., a tuple with n values, which means a row in a table which is shown below.



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

- Thus, in the relational model the term relation is used to refer to a table.
- The term tuple is used to refer to a row.
- Similarly, the term attribute is used to refer to a column of a table.
- The term relation instance is used to refer to a specific instance of a relation,
- i.e., containing a specific set of rows.

- For each attribute of a relation, there is a set of permitted values, called the domain of that attribute.
- A domain is referred to as atomic only if the elements of the domain are indivisible units.
- The null value is a special value that signifies when the value is unknown or does not exist.

Database Schema

- The database schema, is nothing but the logical design of the database.
- The database instance, is nothing but a snapshot of the data in the database at a given instant in time.
- *Let,* A_1, A_2, …, A_n are referred as *attributes then*
- *Relation schema is referred as* $R = (A_1, A_2, \ldots, A_n)$

  For example:

  department(dept_*name, building, budget*)
- *r*(*R*) denotes a *relation r* on the *relation schema R.*

- The concept of a relation instance corresponds to the programming-language notion of a value of a variable.
- The value of a given variable may change with time.
- Similarly when the contents of a relation instance may change with time, the relation is updated.
- But the schema of a relation does not generally change.

- The schema for the relation department

  (ref: 1st chap) is

    - department(dept _name, building, budget)
- It is to be noted that the attribute dept_name appears in both the instructor schema and the department schema.
- This duplication is not a coincidence.
- Rather, using common attributes in relation schemas is one way of relating tuples of distinct relations.

Keys

- **Primary Key** – A primary key is a column or set of columns in a table that uniquely identifies tuples or rows in that table.
- **Super Key** – A super **key** is a set of one of more columns or attributes to uniquely identify rows in a table.
- **Candidate Key** – A super **key** with no redundant(repitition) attribute is known as candidate **key**.

- **Alternate key** – Out of all candidate keys, only one gets selected as primary key, all other keys are known as alternate or secondary keys.
- **Composite key** – A key that consists of more than one attribute to uniquely identify rows in a table is called composite key.
- **Foreign key** – Foreign keys are the columns of a table that points to the primary key of another table.
- They act as a cross-reference between tables.

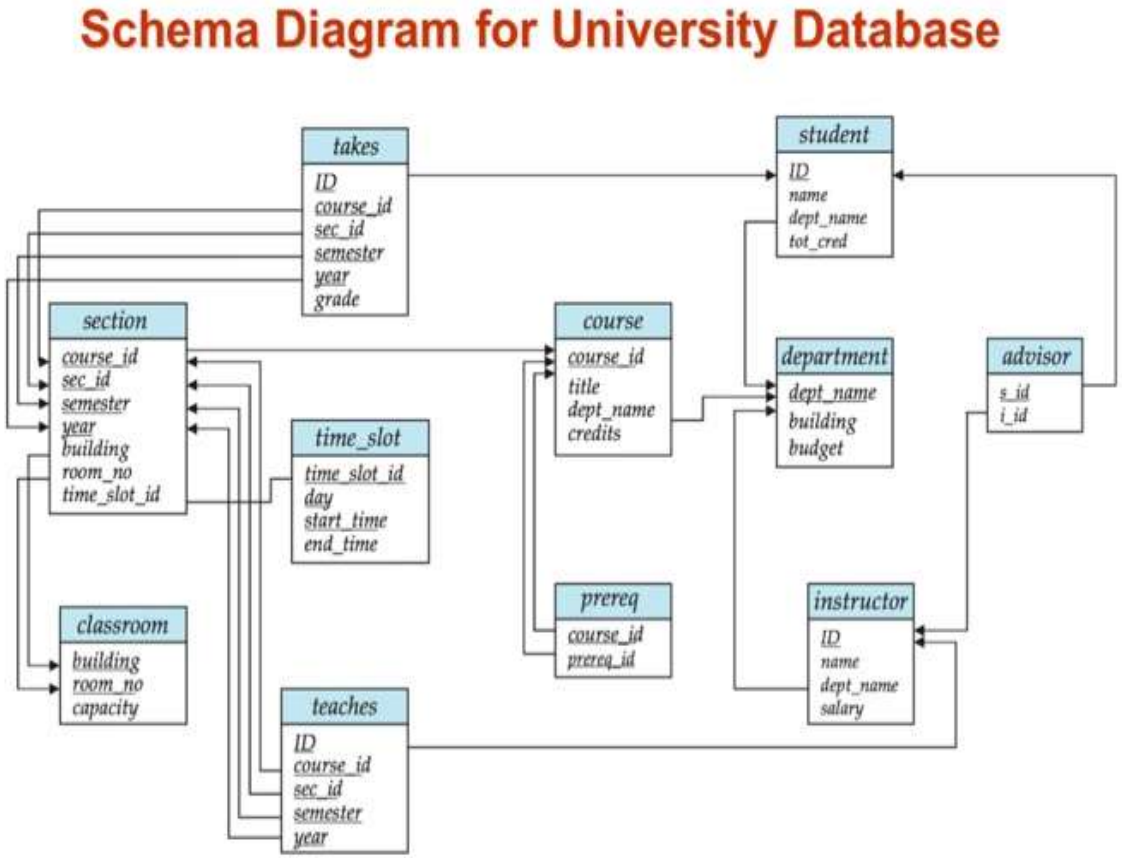
- suppose K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - by “possible *r* ” it mean a relation *r* that could exist in the enterprise.
  - Example: {staff_*name, department*} and

    {*staff_name*}

    are both superkeys of department, if no two staff of department can possibly have the same name.
    - In real life, an attribute such as *staff_id* would be used instead of *staff_name* to uniquely identify customers.

- *K* is a **candidate key** if *K* is minimal
  Example: {*staff_name*} is a candidate key for *department*, since it is a superkey and no subset of it is a superkey.
- **Primary key:** a candidate key chosen as the principal means of identifying tuples within a relation should choose an attribute whose value never, or very rarely, changes.
  - E.g. email address is unique, but may change

Schema Diagrams



- A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams.
- Each relation are drawn as a box, with the relation name at the top in blue, and the attributes are listed inside the box.
- Primary key attributes are shown underlined. Foreign key dependencies are noted with arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

Schema of the university database

- classroom(building, room_number, capacity)
- department(dept_name, building, budget)
- course(course_id, title, dept_name, credits)
- instructor(ID, name, dept_name, salary)
- section(course_id, sec_id, semester, year, building, room number, time slot_id)
- teaches(ID, course_id, sec_id, semester, year)
- student(ID, name, dept_name, tot_cred)
- takes(ID, course_id, sec_id, semester, year, grade)
- advisor(s ID, i_ID)
- time slot(time slot_id, day, start_time, end_ time)
- prereq(course_id, prereq_id)

Relational Query Language

- A query language is a language in which a user needs some information from the database.
- These query languages are usually higher than the standard programming language.
- Query languages can be categorized as either procedural language or nonprocedural language.
- In a procedural language, the user insists the system to perform a sequence of operations on the database to compute the desired result.
- In a nonprocedural language, the user describes the desired information without giving a specific procedure for getting that information.

- Language in which user requests information from the database.
- Categories of languages are as follows
  - Procedural
  - Non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- Pure languages form underlying basis of query languages that people use.

Relational Operations

- All procedural relational query languages provide a set of operations that can be applied to either a single relation or a pair of relations.
- This property allows one to combine several relational operations in a modular way.
- Since the result of a relational query will also be a relation, relational operations can be applied to the results of queries as well as to the given set of relations.

- Another frequent operation is to select certain attributes ie., columns from a relation.
- The result is a new relation having only those selected attributes.
- The relational operations include join operation, natural join and so on…
- Not only those but also finding Cartesian products, set operations such as union, intersection, set difference.

- The join operation combines any two relations by merging pairs of tuples, one from each relation into a single tuple.
- In general, the natural join operation on two relations which matches tuples and those values are the same on all attribute names that should be common to both relations.

- The Cartesian product operation combines tuples from two relations, but unlike the join operation, its result contains all pairs of tuples from the two relations, regardless of whether their attribute values match.
- The union operation performs a set union of two similarly structured tables.
- Other set operations are union, intersection and set difference.

Relational Algebra

- There are six basic operators they are
  - select: σ
  - project: Π
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

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)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.