CSEN2061 – DATABASE MANAGEMENT SYSTEMS

Semester – 5

Syllabus

CSEN2061: DATABASE MANAGEMENT SYSTEMS

UNIT I - Introduction to DBMS and Database Design

Introduction to DBMS: File system vs DBMS, advantages of DBMS, storage data, queries, DBMS structure, Types of Databases – Hierarchical, Network, Relational, Key-Value, Object Oriented, XML DB Overview of File Structures in database.

Data base Design: data models, the importance of data models.

E-R model: Entities, attributes and entity sets, relationship and relationship set, mapping cardinalities, keys, features of ER model, conceptual database design with ER model.

UNIT 2 RELATIONAL MODEL AND BASIC SQL

Relational model: Integrity constraints over relations and enforcement, querying relation data, logical database design, views, destroying/altering tables and views.

Basic SQL: Introduction to SQL, Basic SQL Queries: DML, DDL, DCL, TCL

UNIT 3 ADVANCES SQL AND PL/SQL

Structured Query Language (SQL): Select Commands, Union, Intersection, Except, Nested Queries, Aggregate Operators, Null values, Relational set operators, SQL join operators

Relational Algebra(RA): Selection, Projection, Set operations, Joins

Relational Calculus (TRC, DRC): Tuple Relational Calculus, Domain Relational Calculus PL/SQL, Assertions, Triggers

UNIT 4 SCHEMA REFINEMENT AND NORMAL FORMS

Schema Refinement and Normal Forms: Introduction to Schema Refinement, Functional Dependencies, Reasoning about Functional Dependencies. Normal Forms, Properties of Decomposition, Normalization, different types of dependencies.

UNIT 5 Introduction to Transaction Management, Concurrency Control and Crash Recovery

Introduction to Transaction Management: ACID properties, Transactions and Schedules, Concurrent Execution of Transactions, Lock-Based Concurrency Control.

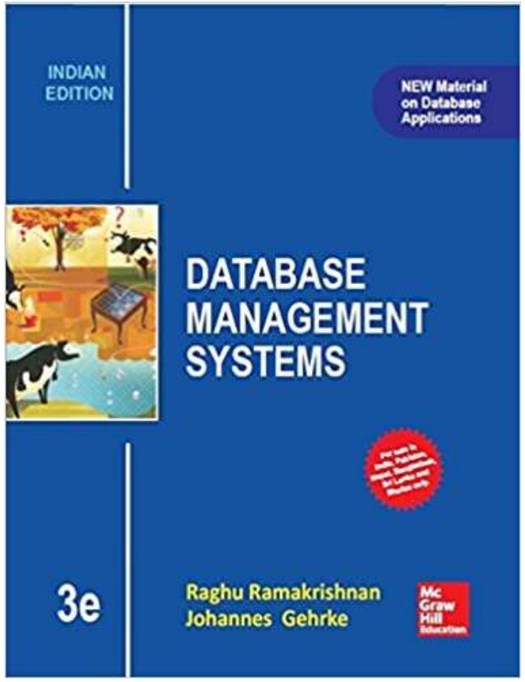
Concurrency Control: 2PL, Serializability and Recoverability, Introduction to Lock Management, Lock Conversions, Dealing with Deadlocks, Concurrency control without locking.

Crash Recovery: Aries, Recovering from a System Crash.

BOOKS

Text Book(s):

Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, McGraw-Hill, 3e, 2014.



References

1. H.F.Korth and A.silberschatz, Database System Concepts, McGraw-Hill, 6e, 2011.
2. RamezElmasri, Shamkant B. Navathe, Fundamentals of Database Systems, Pearson Education,
7e, 2016.

Module 1 - Introduction to DBMS and Database Design

Introduction to DBMS

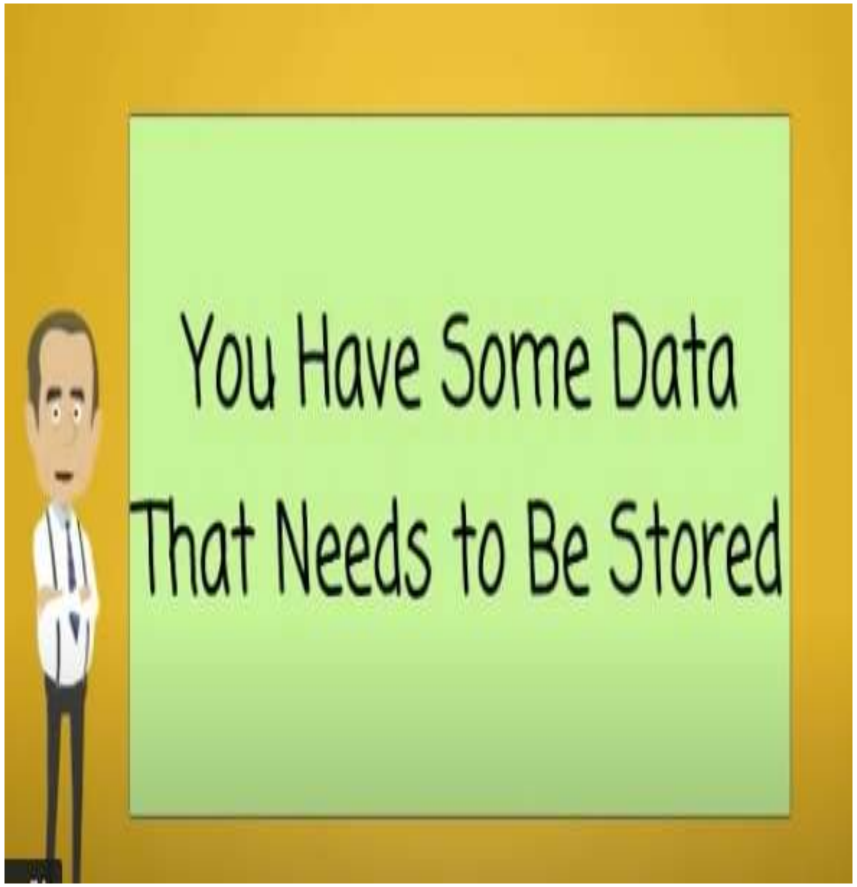
- The study of DBMS is an essential part of every Computer Science student.
- The evolution of database can be tracked backed from ancient to modern(like libraries, medical record, organizations etc.,)
- There is a long history of information storage, indexing and retrieval of data.
- There is something to learn from these histories and their success and failures

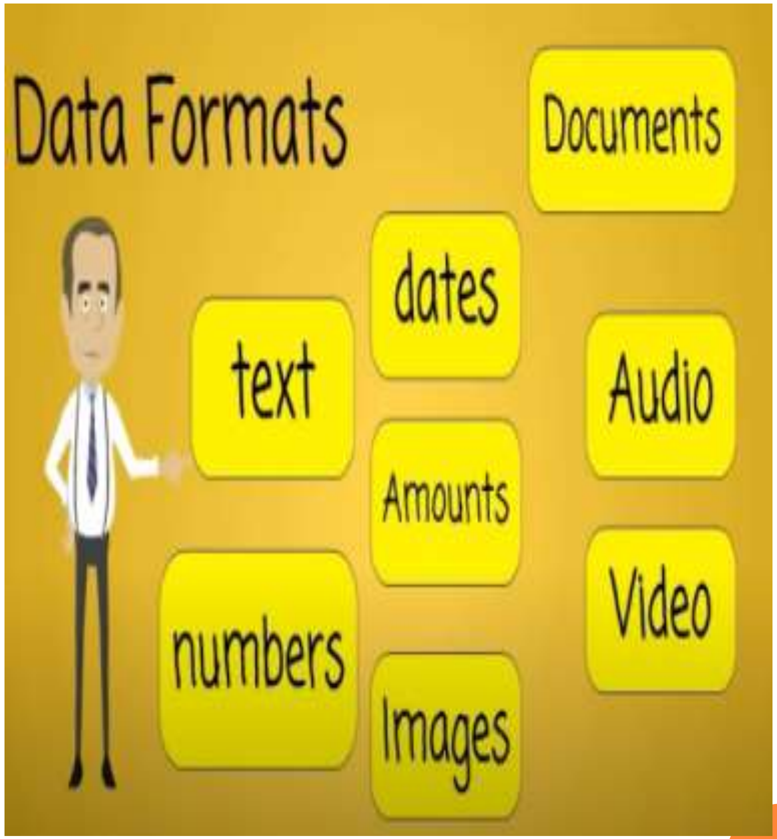
What is database?
Why do we need it?

INTRODUCTION

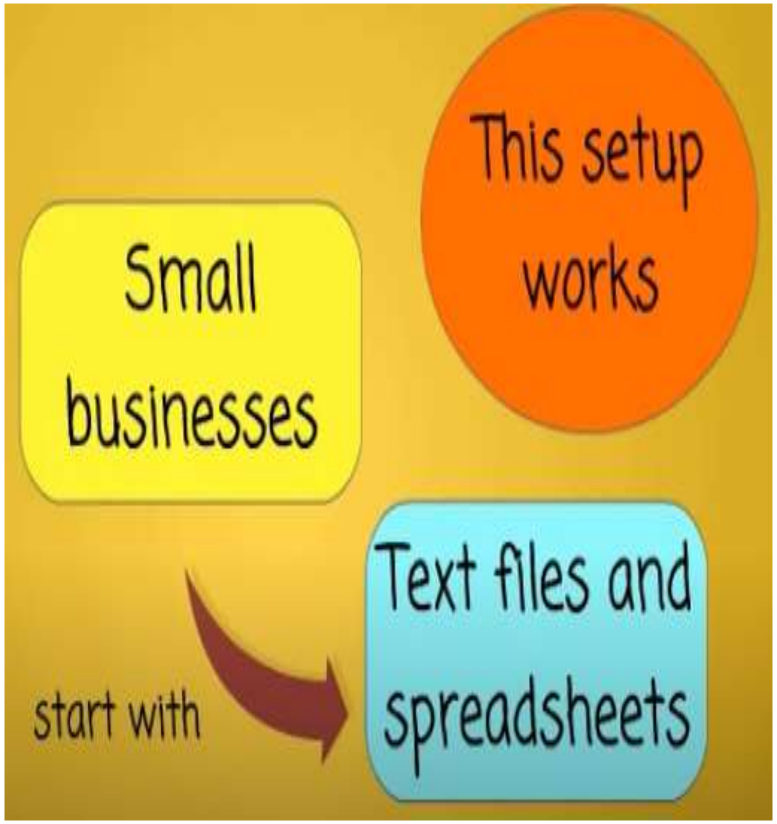
- **Data :** means known fact.
- A database is a collection of related data which are known facts
- Why we need database to store data?

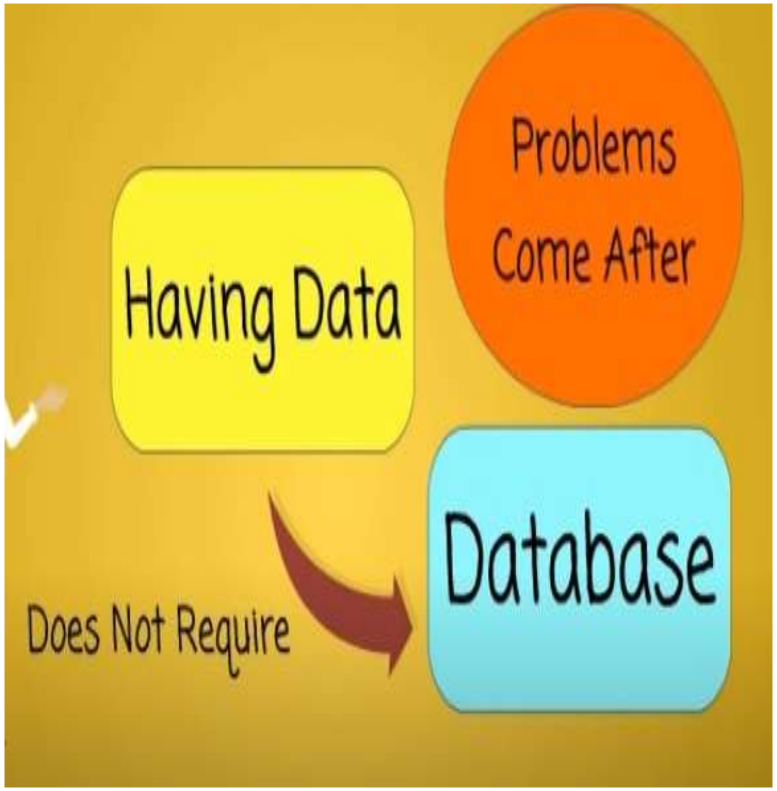
Contd..


You Have Some Data
That Needs to Be Stored


Data Formats
Documents
dates
text
Audio
Amounts
numbers
Video
Images

Contd..

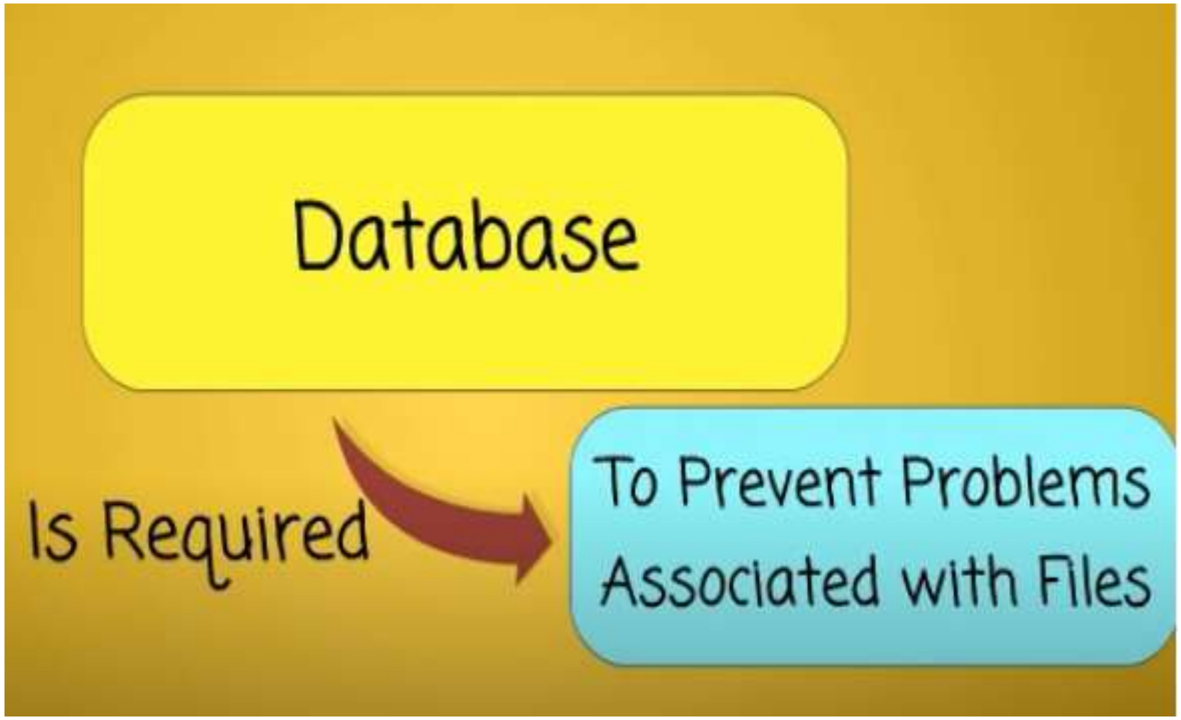

Small businesses
This setup works
start with
Text files and spreadsheets


Having Data
Problems Come After
Does Not Require
Database

LIMITATIONS OF FILE SYSTEM

1. Data redundancy and inconsistency can’t be avoided in File System
2. Limited Data Sharing is seen in File System
3. Concurrency is not possible in File system
4. Difficulty in Accessing data is seen in File System
5. Data integrity problem cant be resolved in File System(FS)
6. Atomicity Problems cant be resolved in FS
7. Security Issues are seen in FS
8. Data Isolation

CONTD..


Database
Is Required
To Prevent Problems Associated with Files

Definition of Database

- A database is an organised collection of data user can query and generate report based on user request.

- Example: Library, student records.

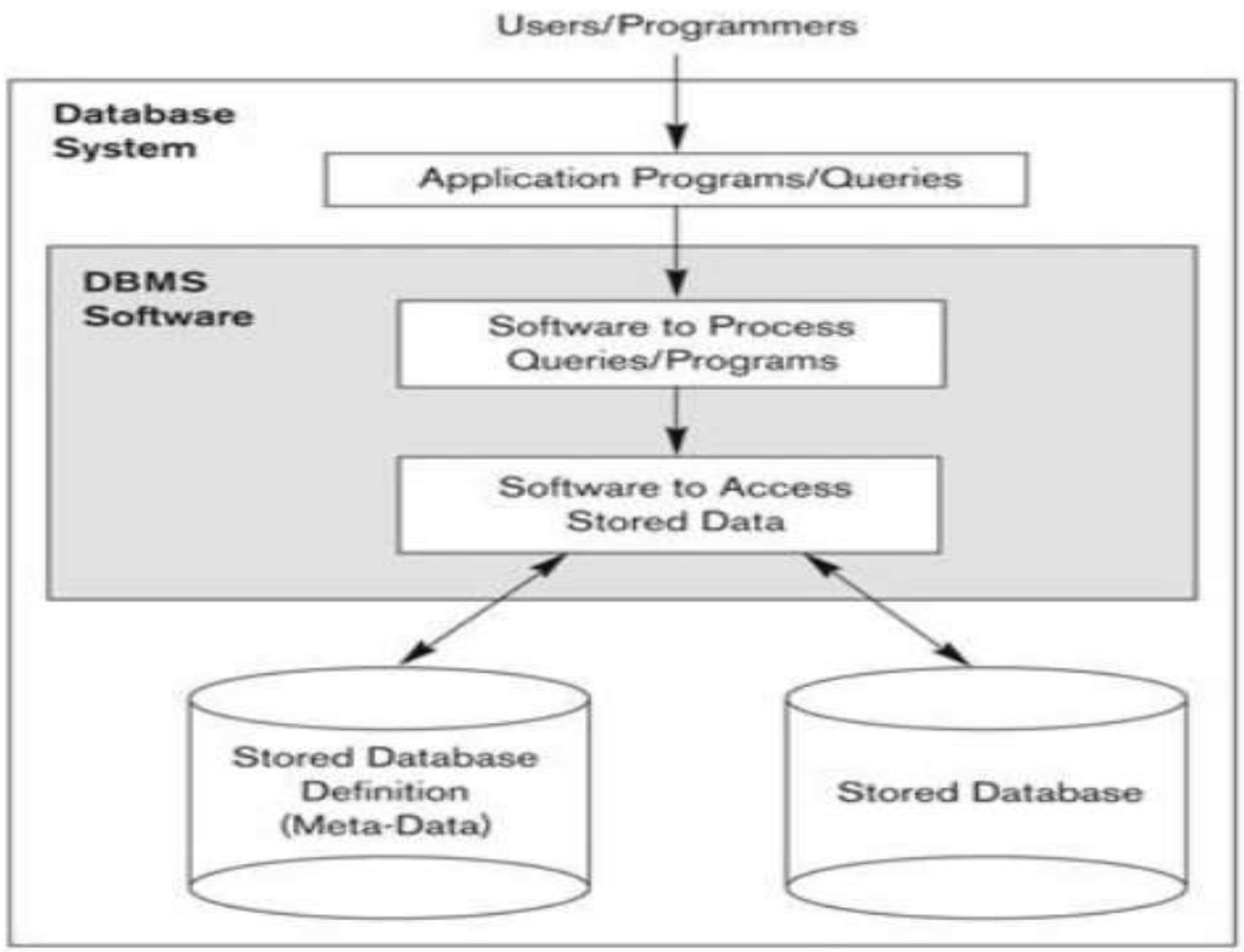
- Well-known databases include: MySql, Microsoft sql server, oracle etc.

What is Database Management System (DBMS)

- DBMS is a software that is meant for storing records in system.

- This software allows accessing of data such as delete, store, retrieve etc that are available in database.

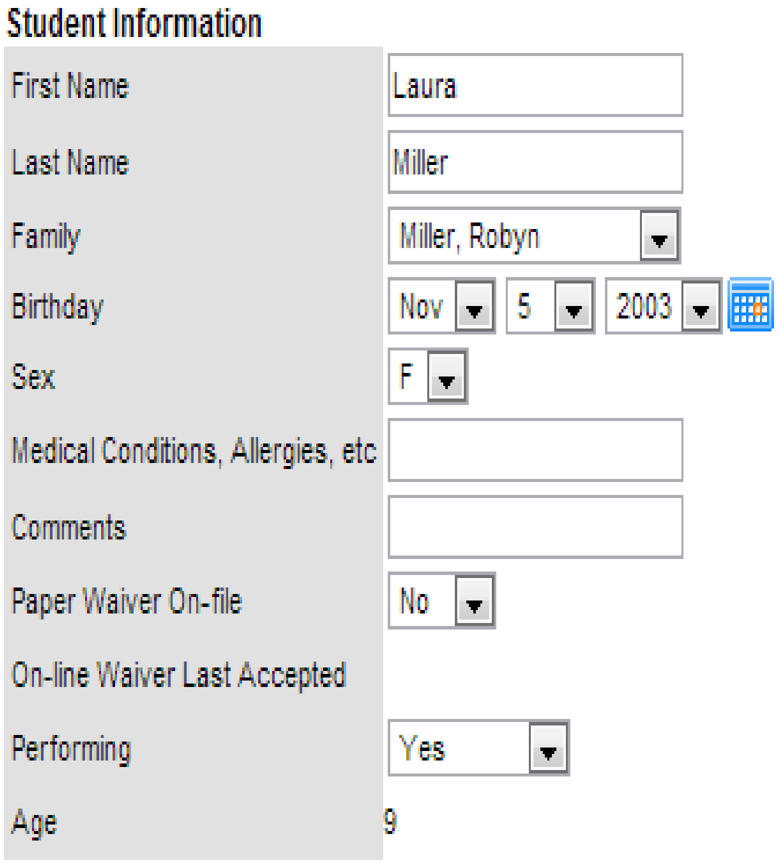
OVERVIEW OF DATABASE SYSTEM


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Example

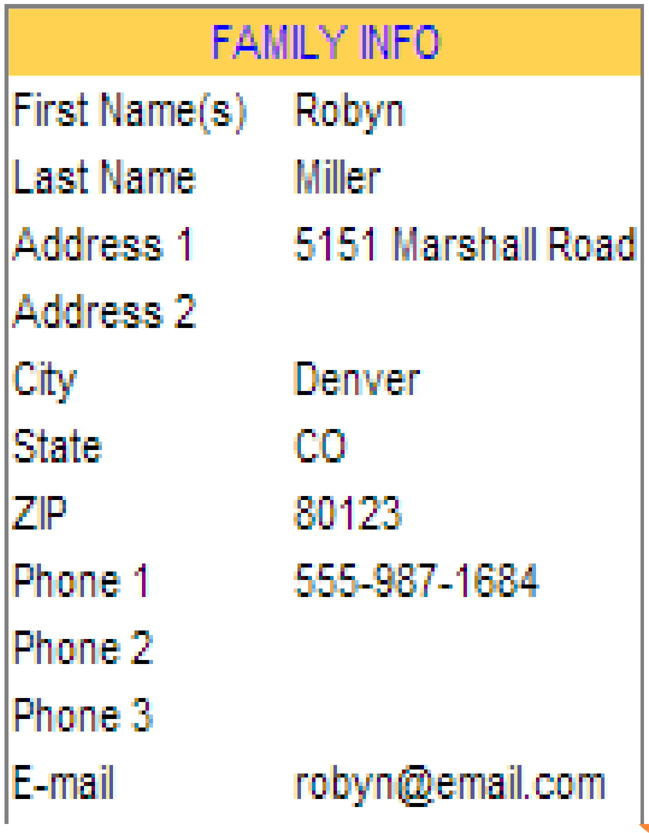
Storing a new information

Viewing the stored data



Student Information

First Name	Laura
Last Name	Miller
Family	Miller, Robyn
Birthday	Nov 5 2003
Sex	F
Medical Conditions, Allergies, etc	
Comments	
Paper Waiver On-file	No
On-line Waiver Last Accepted	
Performing	Yes
Age	9



FAMILY INFO

First Name(s)	Robyn
Last Name	Miller
Address 1	5151 Marshall Road
Address 2	
City	Denver
State	CO
ZIP	80123
Phone 1	555-987-1684
Phone 2	
Phone 3	
E-mail	robyn@email.com

Advantages of DBMS

1. Minimize Data Redundancy
2. Sharing Of Data
3. Data Consistency
4. Search capability
5. Security
6. Simplicity
7. Backup and Recovery
8. Integrity Constraints
9. Data atomicity
10. Concurrency Control
11. Maintaining Cost is lower

Storage data

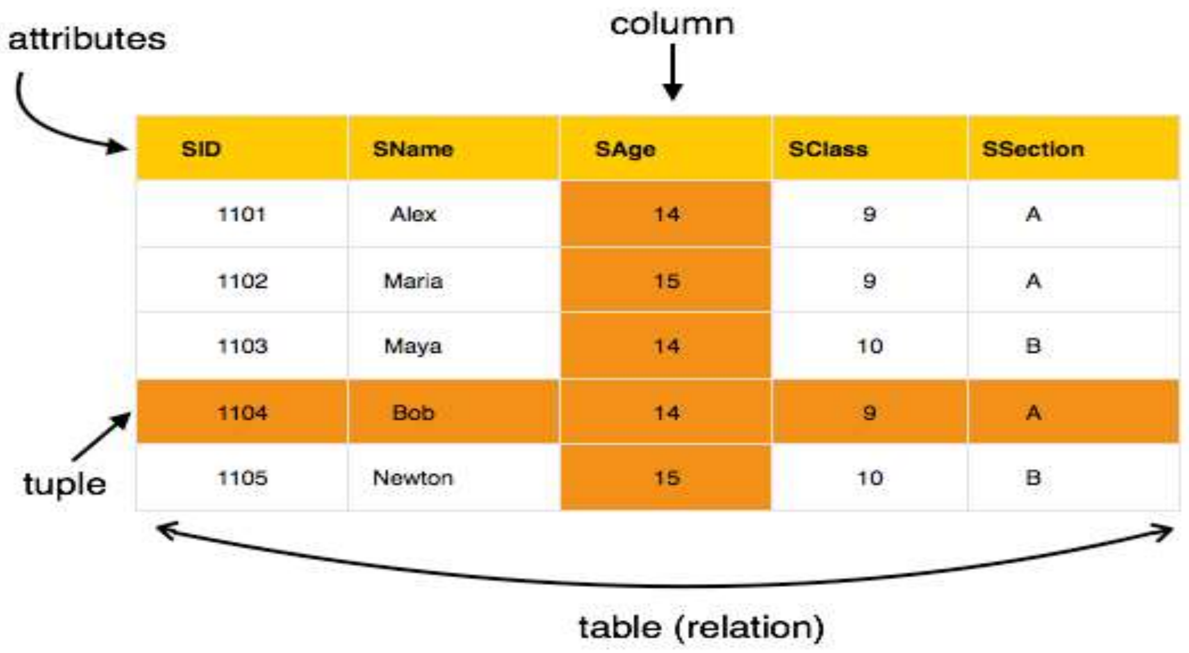
The user of a DBMS is ultimately concerned with some real world enterprise, and the data to be stored describes various aspects of this enterprise that is Data Model

- A semantic data model
- Relational model

- Relational Model - A description of data in terms of a data model is called a schema.

- The following schema shows that:

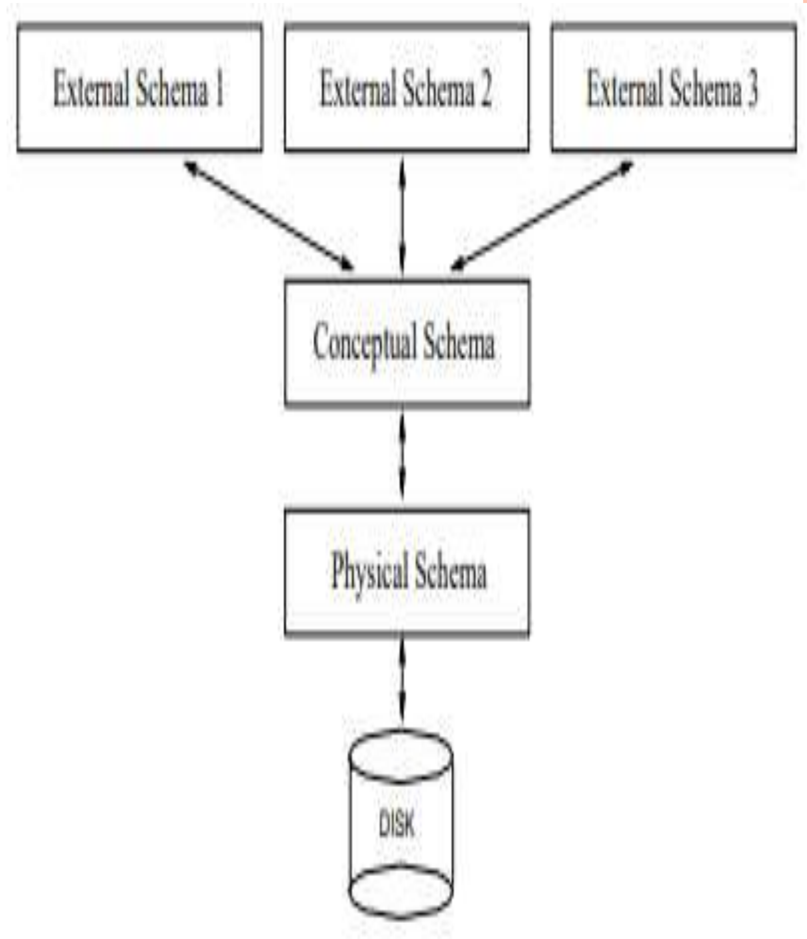
Student (sid:integer, sname: string, sage: integer, sclass: integer, ssection: string)



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

Levels of Abstraction

- Views describe how users see the data
- Conceptual schema defines logical structure.
- Physical schema describes the files and indexes used.



CONTD..

- Physical level: describes how a record is stored (eg: student record)

- Conceptual level: describes data stored in database and its relationship among data.

  Example: **student record (database)**
  student-name: varchar ;
  student-id: integer
  student-age: integer

- View level: application program hides the details of data types. It can also hide information for security purpose.

Data Independence

- Data Independence means- Data independence helps you to keep data separated from all programs that make use of it.

- The following are the types of data independence:

  1. Logical data independence
  2. Physical data independence

Logical data independence

- It is the capacity to change the conceptual schema without having to change the external schemas or application programs.
- Conceptual schema may be changed to expand the database, to change constraints, or to reduce the database by removing a record type or data item.

Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.

- Logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs.

Physical data independence

- It is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.
- Changes to the internal schema may be needed because some physical files may be reorganized, for example, by creating additional access structures to improve the performance of retrieval or update.
- If the same data as before remains in the database, there is no need to change the conceptual schema.
- Physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk,and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications remain unaware of these details.

What is Query?

- A query is a request for data or information from a database table or combination of tables.
- Example : **How many students are enrolled in section A?**
- A DBMS provides a specialized language, called the **query language,** in which queries can be posed.
- **Relational calculus** is a formal query language based on mathematical logic, and queries in this language have an intuitive, precise meaning.
- **Relational algebra** is another formal query language, based on a collection of operators for manipulating relations, which is equivalent in power to the calculus.

DBMS STRUCTURE

- The DBMS accepts SQL commands generated from a variety of user interfaces, produces query evaluation plans, executes these plans against the database, and returns the answers.
- When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query.
- An execution plan is a blueprint for evaluating a query, usually represented as a tree of relational operators.

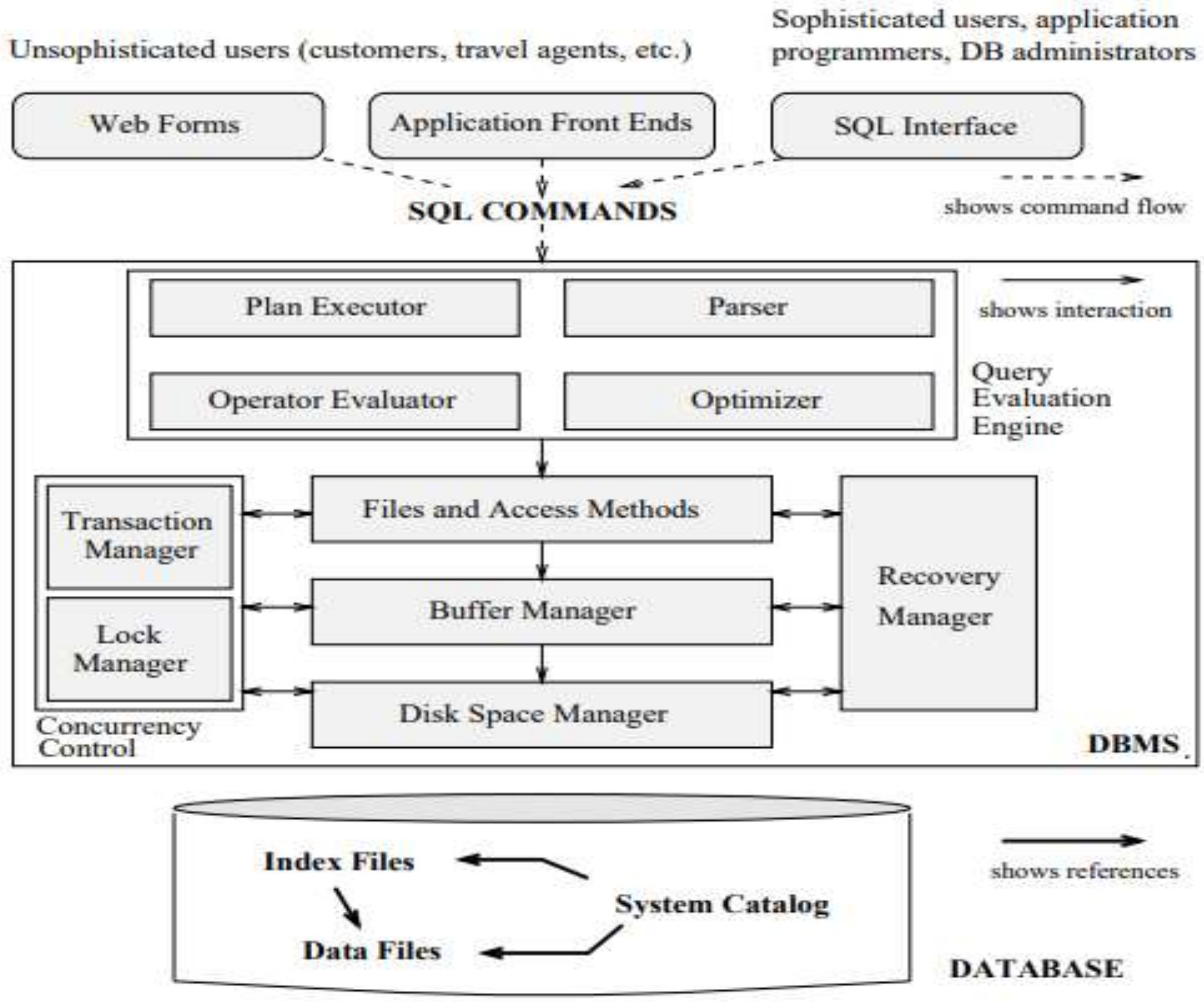


Figure 1.3 Architecture of a DBMS

- The code that implements relational operators sits on top of the file and access methods layer. This layer supports the concept of a file, which, in a DBMS, is a collection of pages or a collection of records.
- The files and access methods layer code sits on top of the buffer manager, which brings pages in from disk to main memory ct." needed in response to read requests.
- The lowest layer of the DBMS software deals with management of space on disk, where the data is stored. Higher layers allocate, deallocate, read, and write pages through (routines provided by) this layer, called the disk space manager.

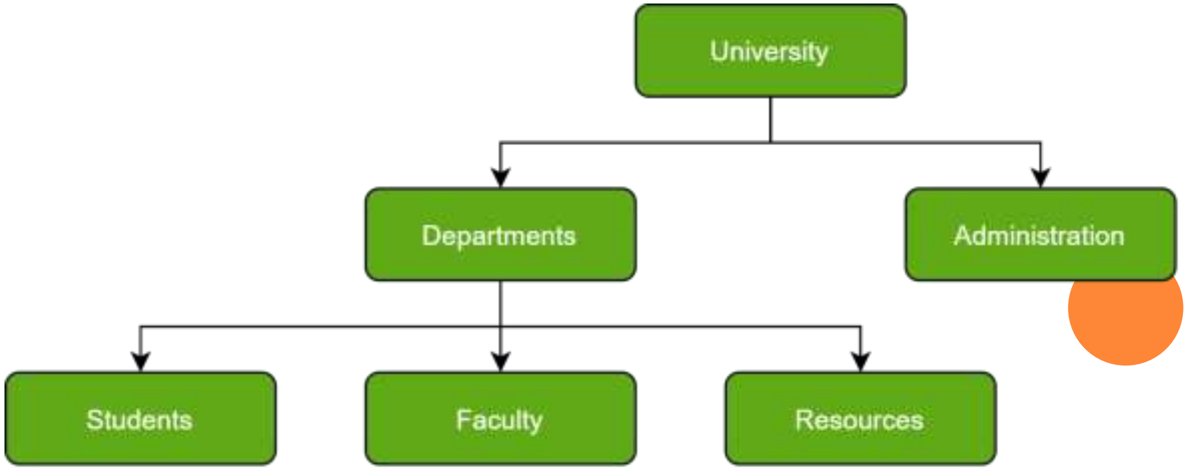
- The DBMS supports concurrency and crash recovery by carefully scheduling user requests and maintaining a log of all changes to the database.
- DBMS components associated with concurrency control and recovery include the transaction manager, which ensures that transactions request and release locks according to a suitable locking protocol and schedules the execution transactions; the lock manager, which keeps track of requests for locks and grants locks on database objects when they become available; and the recovery manager, which is responsible for maintaining a log and restoring the system to a consistent state after a crash.
- The disk space manager, buffer manager, and file and access method layers must interact with these components.

Types of Databases

- Hierarchical
- Network
- Relational
- Key-Value
- Object Oriented
- XML DB

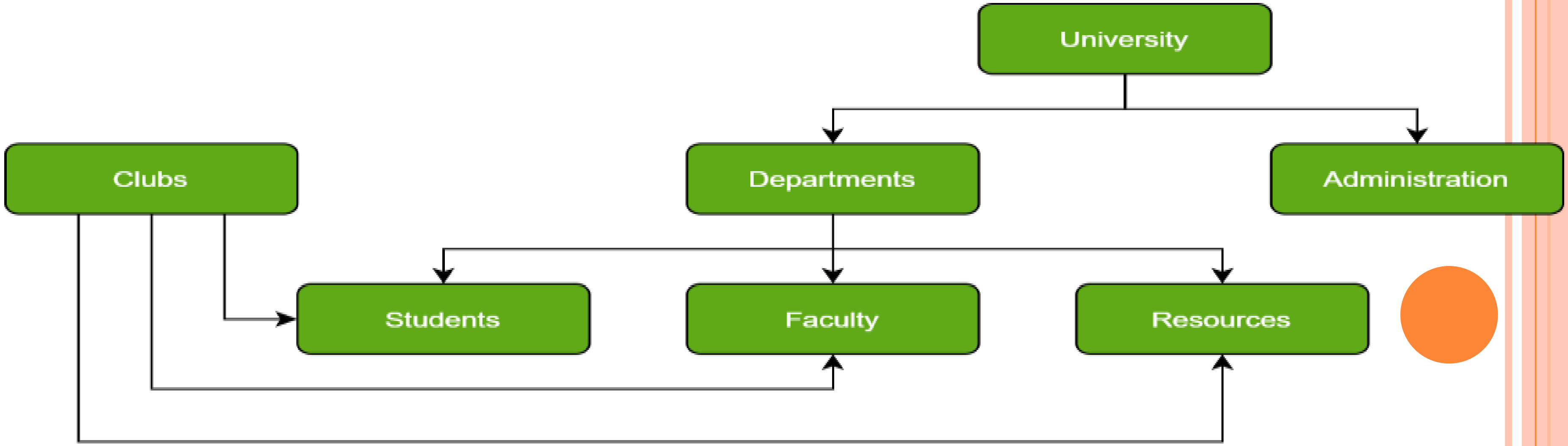
HIERARCHICAL DATABASES

- A Hierarchical Database Management System (HDBMS) is a type of DBMS that organizes data in a hierarchical tree-like structure.
- In an HDBMS, data is represented as a series of records, with each record having one parent record and one or more child records. This creates a parent-child relationship between records, with the parent record being at the top of the hierarchy and child records being at the bottom.



NETWORK DATABASES

- This is looks like a Hierarchical database model due to which many time it is called as modified version of Hierarchical database.
- Network database model organised data more like a graph and can have more than one parent node. The network model is a database model conceived as a flexible way of representing objects and their relationships.



Relational Database

- A relational database is developed by E. F. Codd in 1970. The various software systems used to maintain relational databases are known as a relational database management system (RDBMS).
- In this model, data is organised in rows and column structure i.e., two-dimensional tables and the relationship is maintained by storing a common field.
- It consists of three major components.
- In relational model, three key terms are heavily used such as relations, attributes, and domains.
- A relation nothing but is a table with rows and columns.
- The named columns of the relation are called as attributes, and finally the domain is nothing but the set of values the attributes can take.

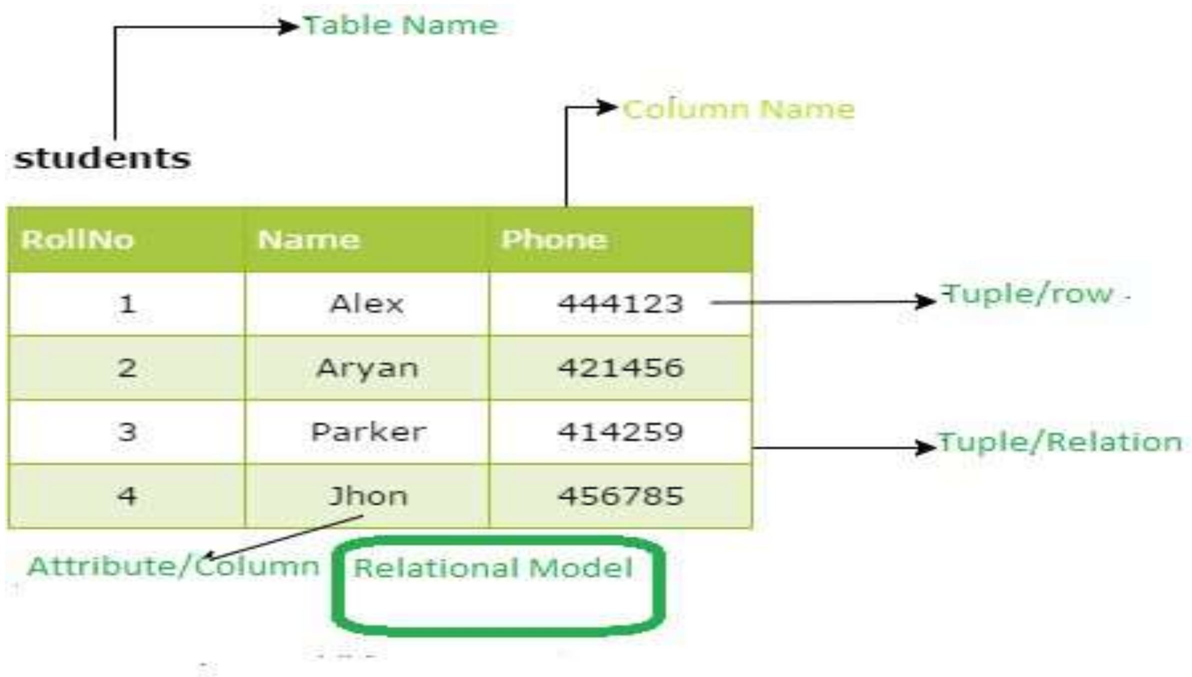
Relational Database

Terminology used in Relational Model

- Tuple: Each row in a table is known as tuple.
- Cardinality of a relation: The number of tuples in a relation determines its cardinality.

In this case, the relation has a cardinality of 4.

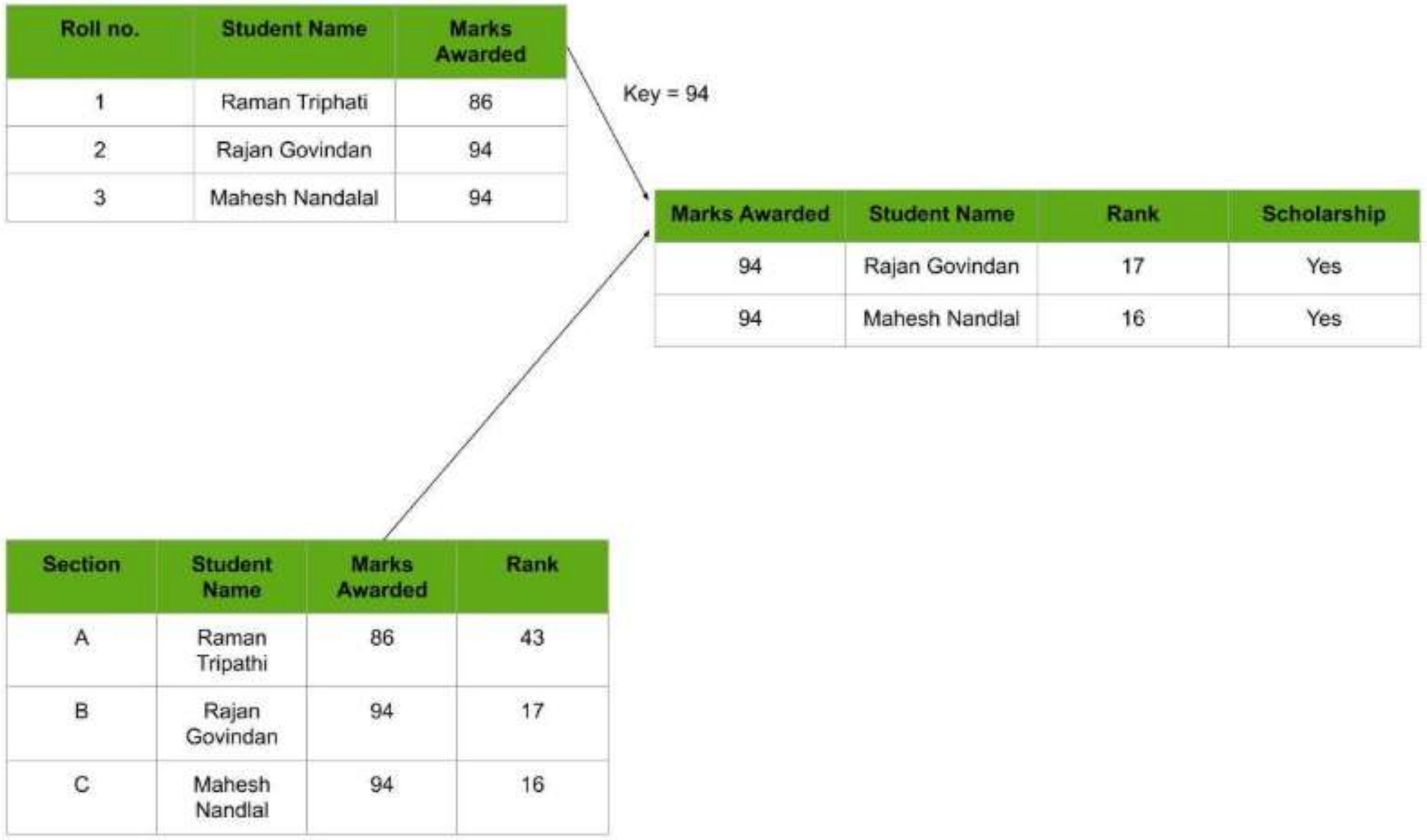
- Degree of a relation: Each column in the tuple is called an attribute. The number of attributes in a relation determines its degree. The relation in figure has a degree of 3.



RollNo	Name	Phone
1	Alex	444123
2	Aryan	421456
3	Parker	414259
4	Jhon	456785

Relational Database

- Refer to the diagram below and notice how the concept of ‘Keys’ is used to link two tables.



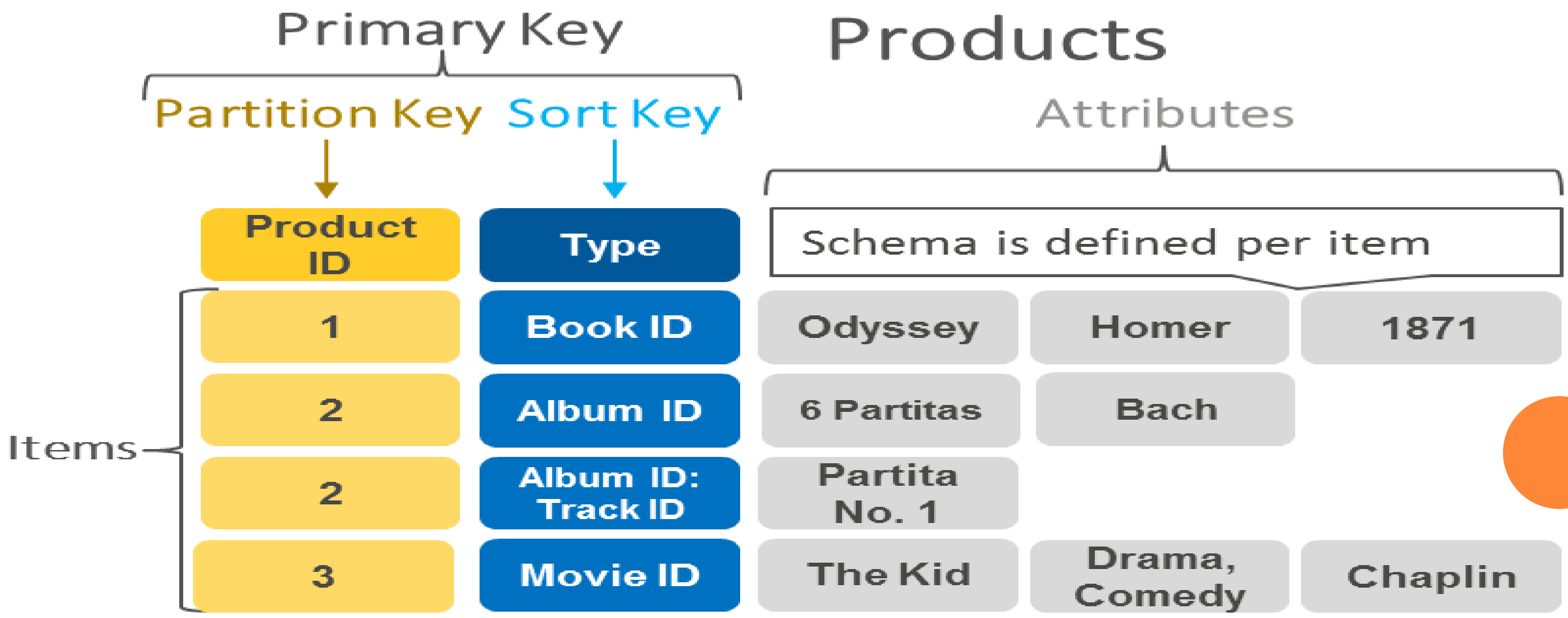
Roll no.	Student Name	Marks Awarded
1	Raman Triphati	86
2	Rajan Govindan	94
3	Mahesh Nandalal	94

Marks Awarded	Student Name	Rank	Scholarship
94	Rajan Govindan	17	Yes
94	Mahesh Nandlal	16	Yes

Section	Student Name	Marks Awarded	Rank
A	Raman Tripathi	86	43
B	Rajan Govindan	94	17
C	Mahesh Nandlal	94	16

KEY-VALUE DATABASES

- A key-value database is a type of nonrelational database that uses a simple key-value method to store data.
- A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier.
- Both keys and values can be anything, ranging from simple objects to complex compound objects.
- Key-value databases are highly partitionable and allow horizontal scaling at scales that other types of databases cannot achieve.
- A type of NoSQL DBMS that store data as a mapping of keys to values and are optimized for high-speed data retrieval.

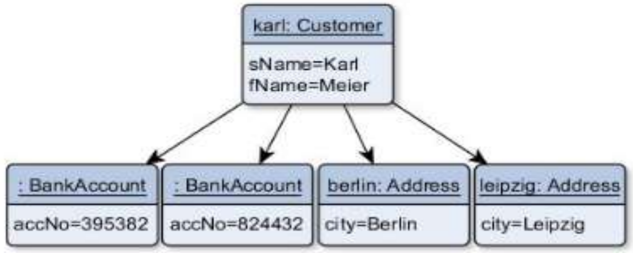


Object Oriented Database

- An Object-oriented Database Management System (OODBMS) is a type of DBMS that organizes data into objects and allows for the creation of classes and inheritance.
- In an OODBMS, data is stored in a format that is similar to objects in object-oriented programming languages, such as Java or C++. Each object has its own properties, methods, and behaviors, and can be part of a class or hierarchy of classes.
- An object database is a system in which information is represented in the form of objects as used in object-oriented programming.
- Object oriented databases are different from relational databases which are table-oriented.

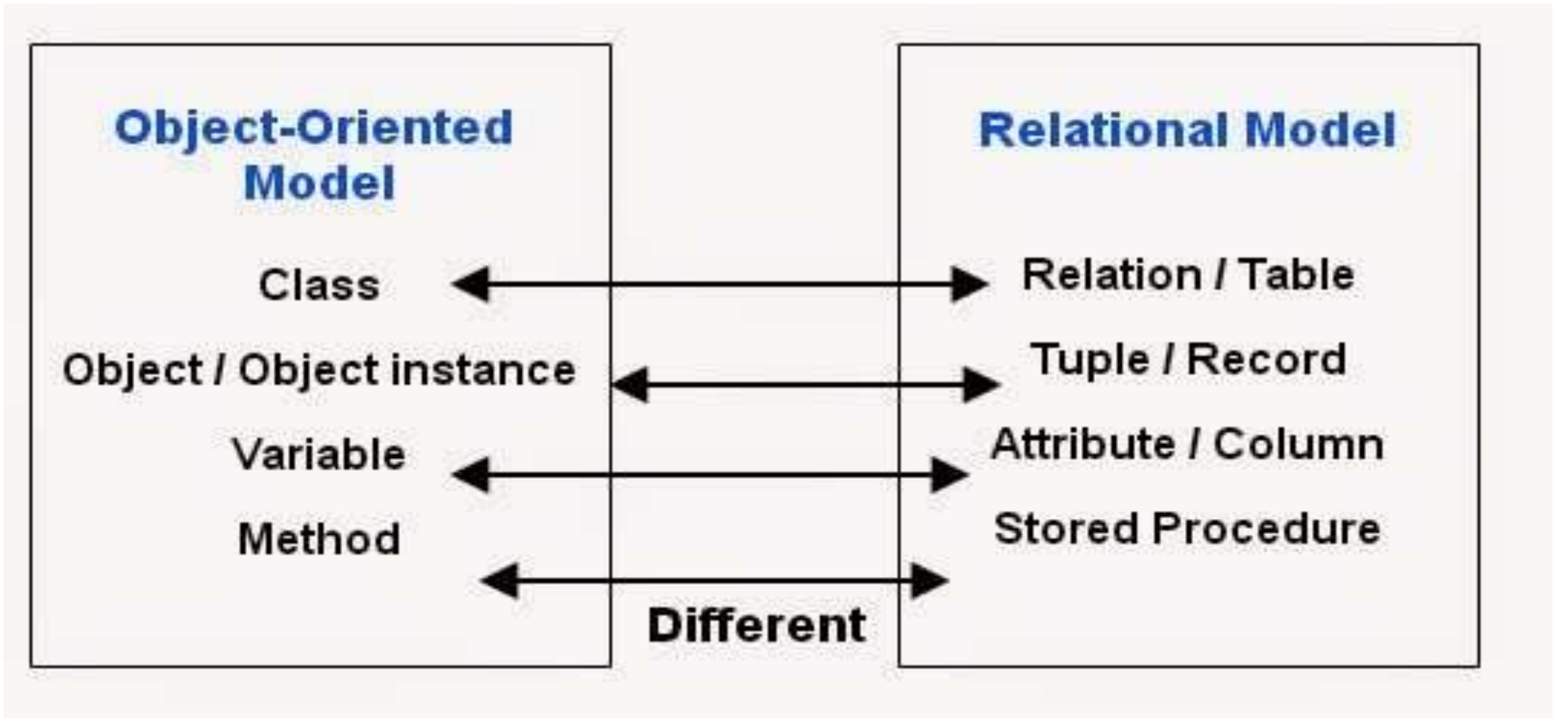
OBJECT ORIENTED DATABASE

- The object-oriented data model is based on the object-oriented- programming language concept, which is now in wide use. Inheritance, polymorphism, overloading. object-identity, encapsulation and information hiding with methods to provide an interface to objects, are among the key concepts of object-oriented programming that have found applications in data modelling. The object-oriented data model also supports a rich type system, including structured and collection types.
- In object-oriented programming, an **Object Database** is a system in which data is represented as objects.
- Relational Databases, which are table-oriented, are not the same as object-oriented Databases.
- The Object-Oriented Data Model is one of the types of database models that is based on the widely used concept of object-oriented programming languages.



OBJECT ORIENTED DATABASE

- The following figure shows the difference between relation and object-oriented database model.

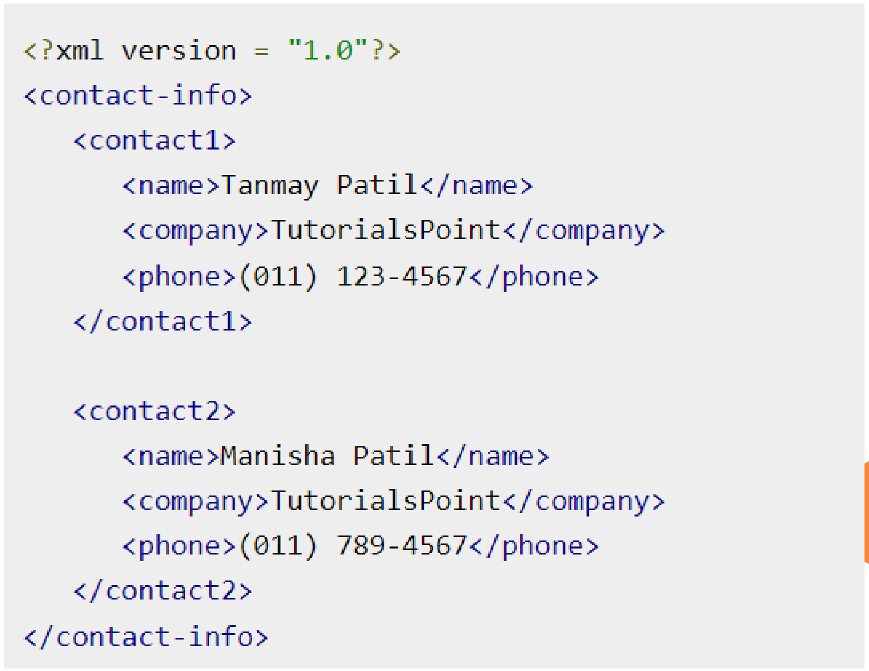


XML Database

- The Extensible Markup Language (XML) was not designed for database applications.
- In fact, like the Hyper-Text Markup Language (HTML) on which the World Wide Web is based, XML has its roots in document management, and is derived from a language for structuring large documents known as the Standard Generalized Markup Language (SGML).
- However, unlike SGML and HTML, XML is designed to represent data. It is particularly useful as a data format when an application must communicate with another application, or integrate information from several other applications.

XML Database

- XML database is a data persistence software system used for storing the huge amount of information in XML format. It provides a secure place to store XML documents.
- You can query your stored data by using XQuery, export and serialize into desired format. XML databases are usually associated with document-oriented databases.



```xml
<?xml version = "1.0"?>
<contact-info>
   <contact1>
      <name>Tanmay Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 123-4567</phone>
   </contact1>

   <contact2>
      <name>Manisha Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 789-4567</phone>
   </contact2>
</contact-info>
```

Overview of File Structures in database

- Relative data and information is stored collectively in file formats.
- A file is sequence of records stored in binary format.
- A disk drive is formatted into several blocks, which are capable for storing records.
- File records are mapped onto those disk blocks.

File Organization

- The database is stored as a collection of *files*. Each file is a sequence of *records*. A record is a sequence of fields.
- One approach:
  - assume record size is fixed
  - each file has records of one particular type only
  - different files are used for different relations

  This case is easiest to implement; will consider variable length records later.

Fixed-Length Records

- Simple approach:
  - Store record i starting from byte $n * (i - 1)$, where n is the size of each record.
  - Record access is simple but records may cross blocks
    - Modification: do not allow records to cross block boundaries

- Deletion of record i: alternatives:
  - move records $i + 1, \ldots, n$ to $i, \ldots, n - 1$
  - move record n to i
  - do not move records, but link all free records on a *free list*

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Deleting record 3 and compacting

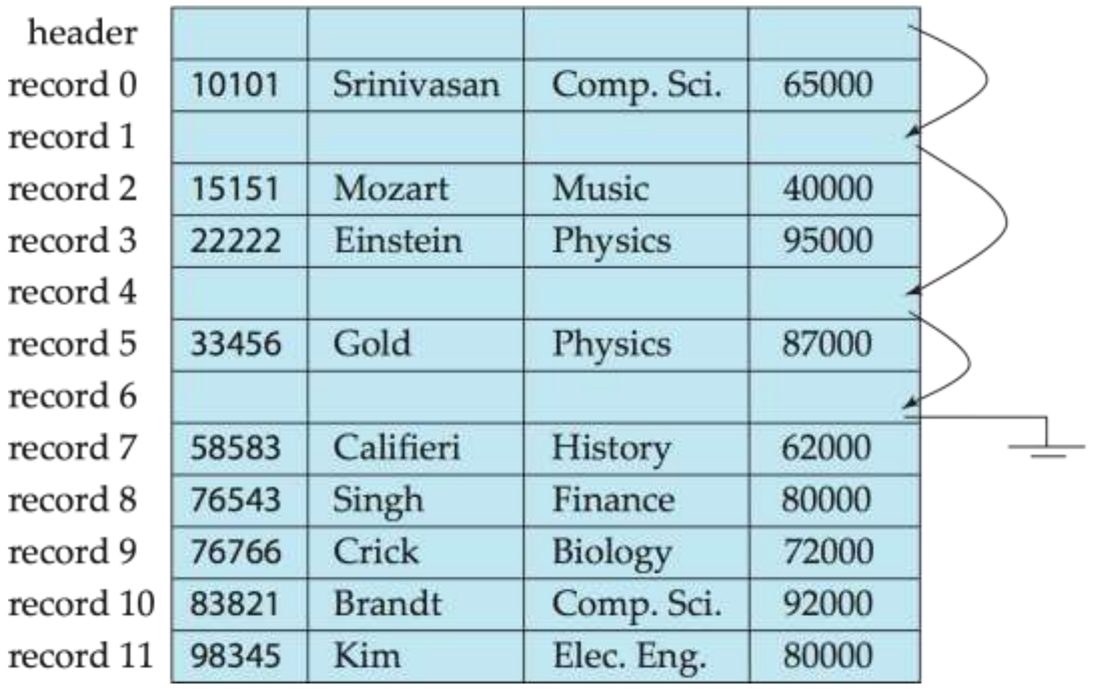
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Deleting record 3 and moving last record

record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

FREE LISTS

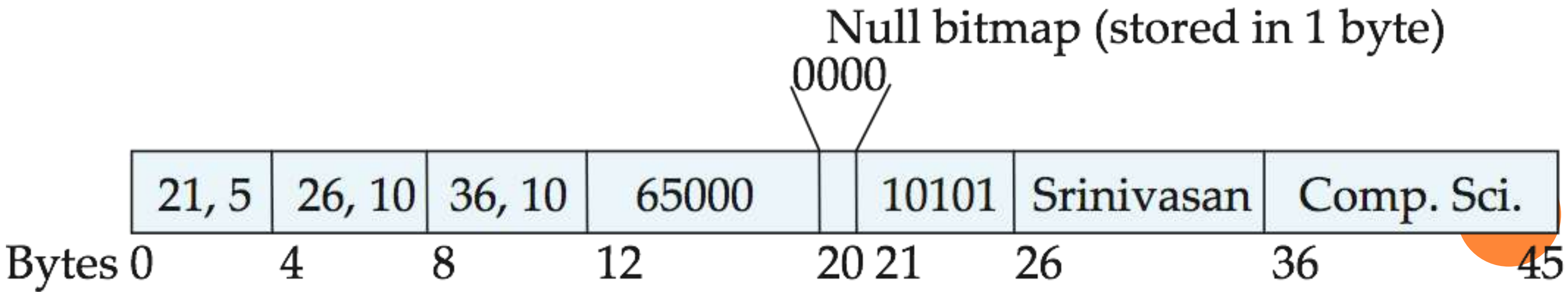
- Store the address of the first deleted record in the file header.
- Use this first record to store the address of the second deleted record, and so on
- Can think of these stored addresses as pointers since they "point" to the location of a record.
- More space efficient representation: reuse space for normal attributes of free records to store pointers. (No pointers stored in in-use records.)



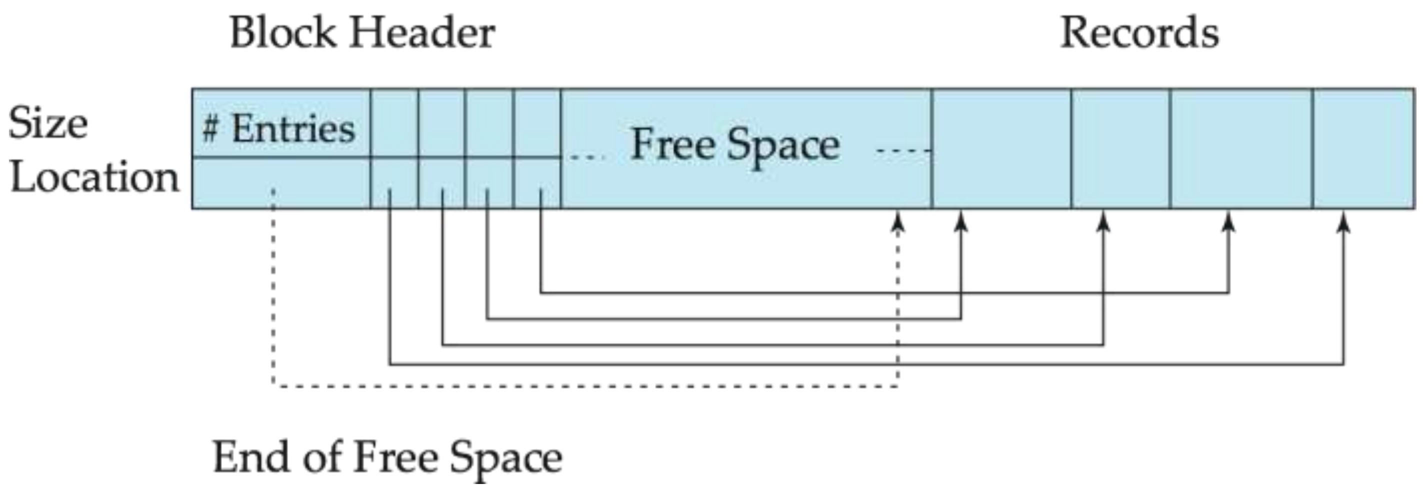
header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

Variable-Length Records

- Variable-length records arise in database systems in several ways:
  - Storage of multiple record types in a file.
  - Record types that allow variable lengths for one or more fields such as strings (**varchar**)
  - Record types that allow repeating fields (used in some older data models).
- Attributes are stored in order
- Variable length attributes represented by fixed size (offset, length), with actual data stored after all fixed length attributes
- Null values represented by null-value bitmap



Variable-Length Records: Slotted Page Structure



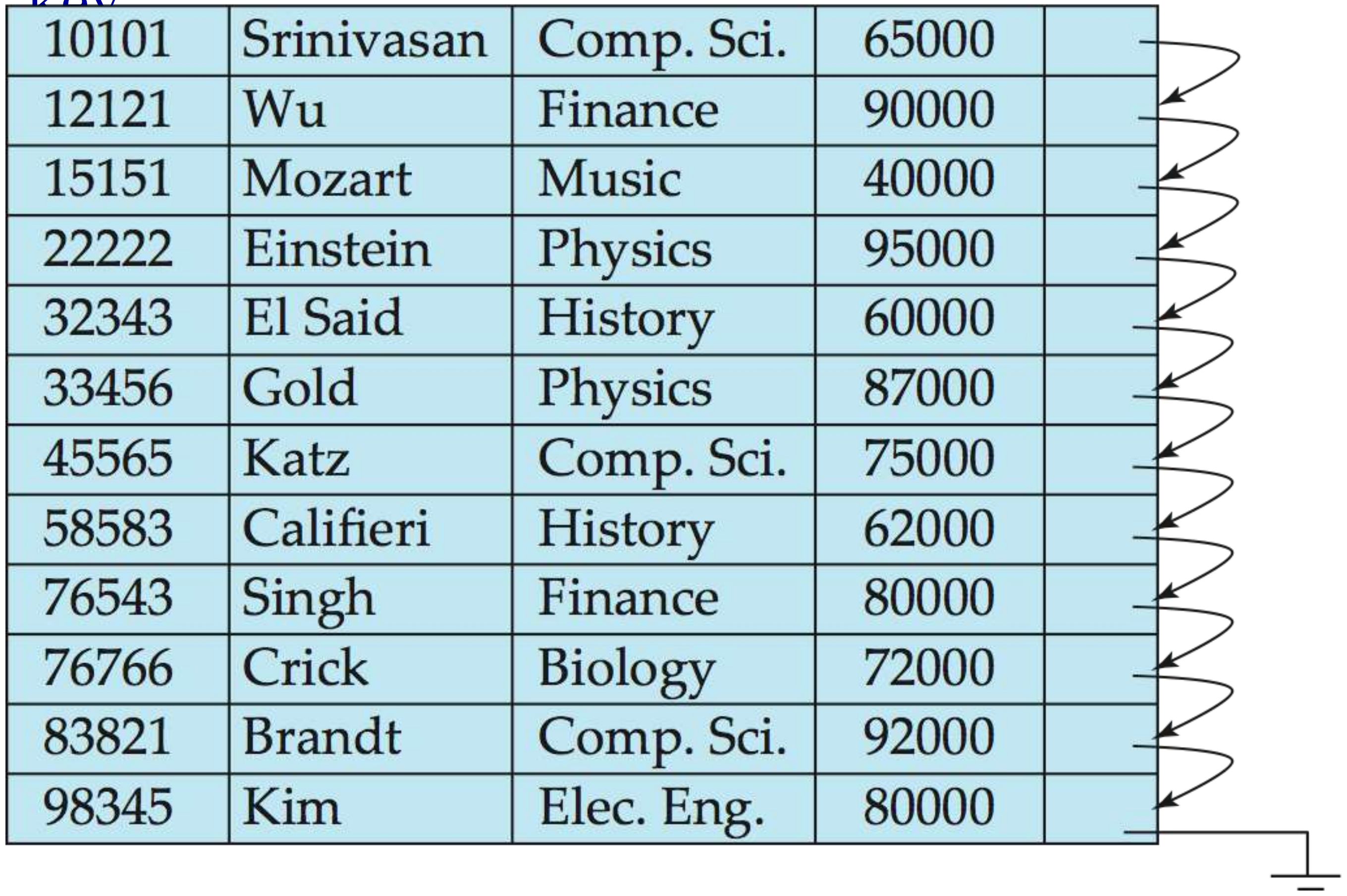
- **Slotted page** header contains:
  - number of record entries
  - end of free space in the block
  - location and size of each record
- Records can be moved around within a page to keep them contiguous with no empty space between them; entry in the header must be updated.
- Pointers should not point directly to record — instead they should point to the entry for the record in header.

Organization of Records in Files

- **Heap** – a record can be placed anywhere in the file where there is space
- **Sequential** – store records in sequential order, based on the value of the search key of each record
- **Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed
- Records of each relation may be stored in a separate file. In a **multitable clustering file organization** records of several different relations can be stored in the same file
  - Motivation: store related records on the same block to minimize I/O

Sequential File Organization

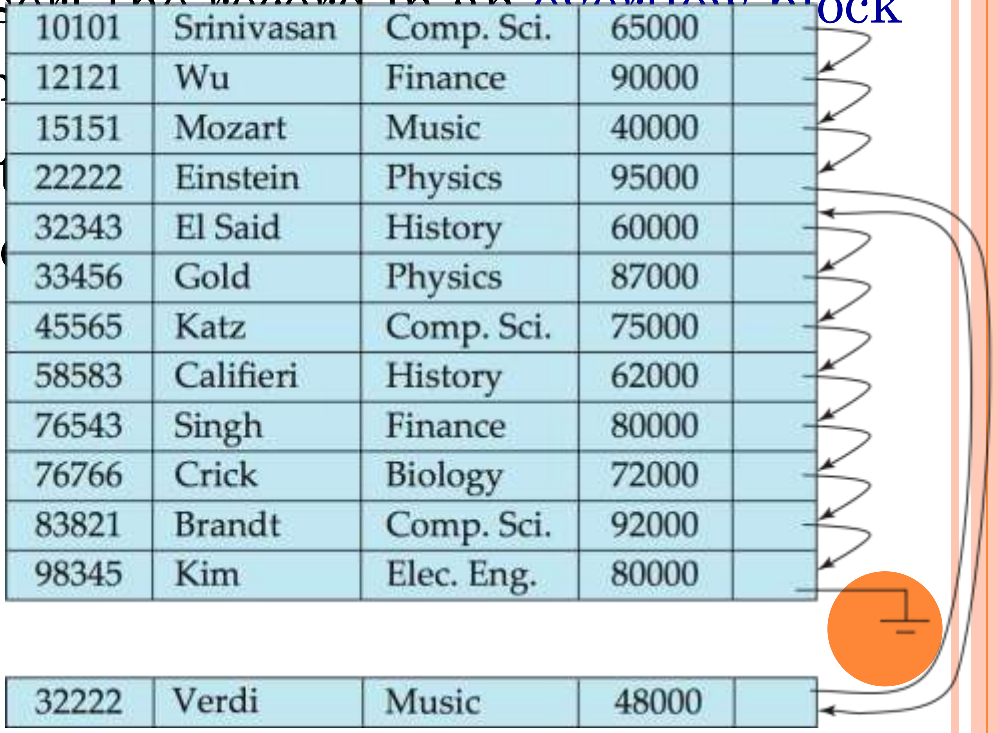
- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key



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	

Sequential File Organization (Cont.)

- Deletion – use pointer chains
- Insertion –locate the position where the record is to be inserted
  - if there is free space insert there
  - if no free space, insert the record in an overflow block
  - In either case, poin
- Need to reorganize t from time to time t sequential order



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

32222	Verdi	Music	48000

Multitable Clustering File Organization

Store several relations in one file using a **multitable clustering** file organization

department

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Physics	Watson	70000

instructor

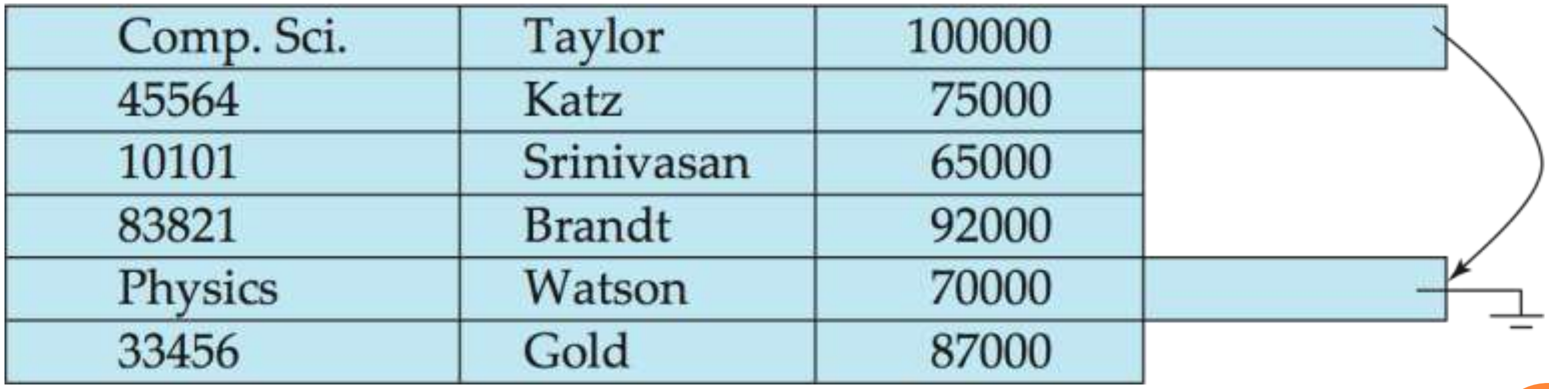
ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000

multitable clustering of *department* and *instructor*

Comp. Sci.	Taylor	100000
45564	Katz	75000
10101	Srinivasan	65000
83821	Brandt	92000
Physics	Watson	70000
33456	Gold	87000

Multitable Clustering File Organization (cont.)

- good for queries involving *department* $\bowtie$ *instructor*, and for queries involving one single department and its instructors
- bad for queries involving only *department*
- results in variable size records
- Can add pointer chains to link records of a particular relation



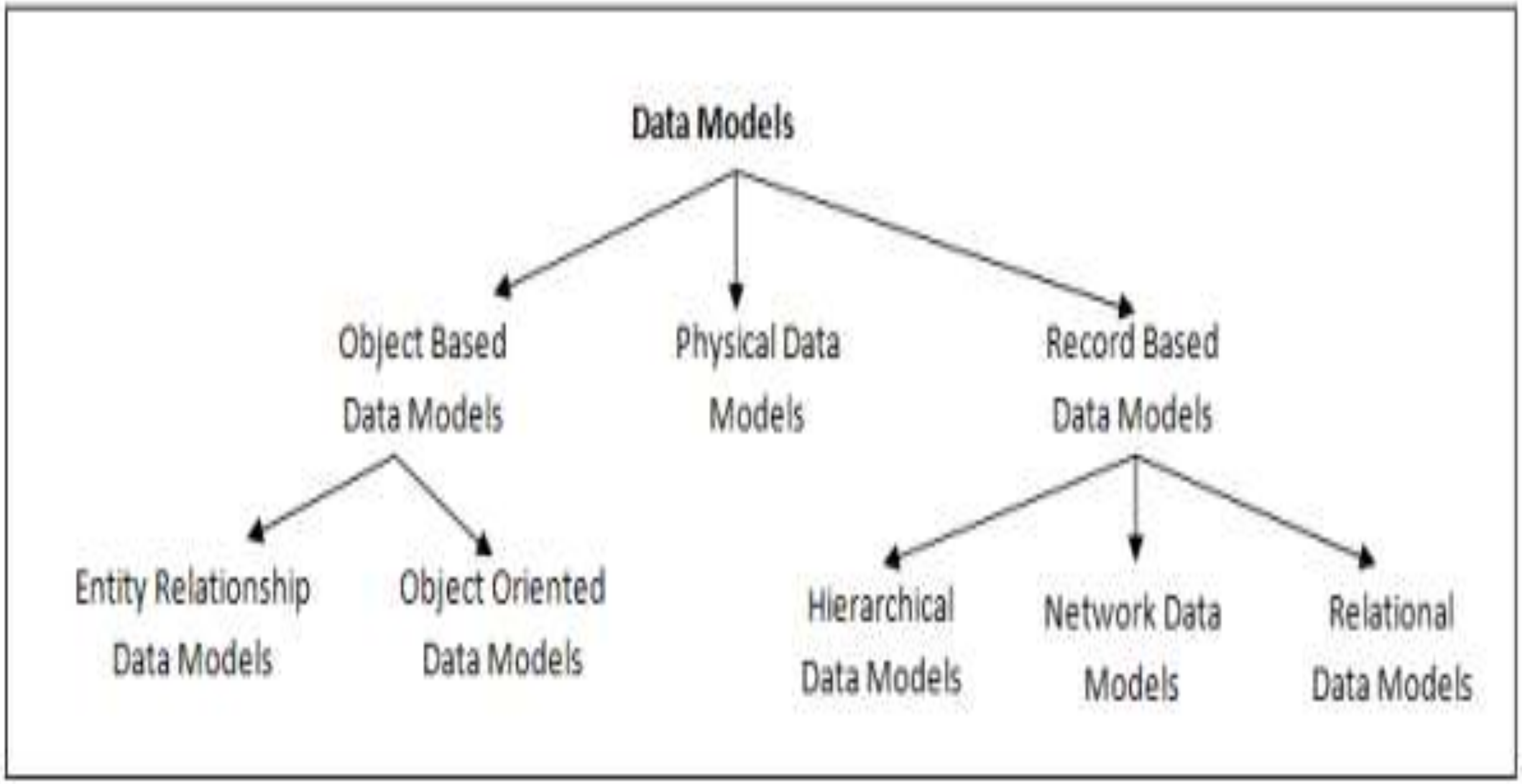
Comp. Sci.	Taylor	100000	
45564	Katz	75000	
10101	Srinivasan	65000	
83821	Brandt	92000	
Physics	Watson	70000	
33456	Gold	87000	

Data base Design: data models, the importance of data models

Data Model

- A **Data Model** is a logical structure of Database.
- It describes the design of database to reflect entities, attributes, relationship among data, constraints etc.. that determine how data can be stored and accessed.

Data model is divided into 3 categories



Object based Data Models

It is based on real world objects. It is designed by using the entities, attributes and their relationship..

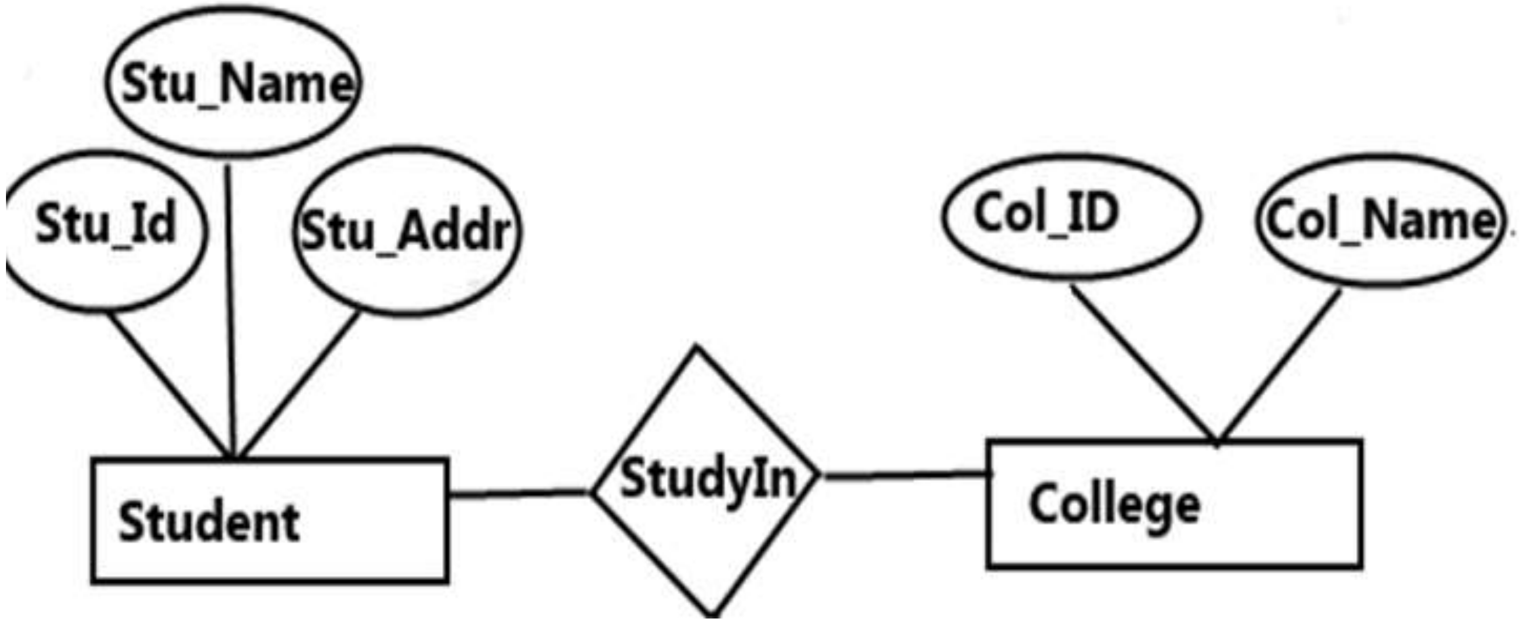
There are two types of object based data Models

a. Entity Relationship Model and

b. Object oriented data model.

Entity Relationship Model

- Graphical representation of entities and their relationships in a database structure.



Advantages:

- It makes the requirement simple and easily understandable .
- One can convert ER diagrams into record based data model easily .

Disadvantages:

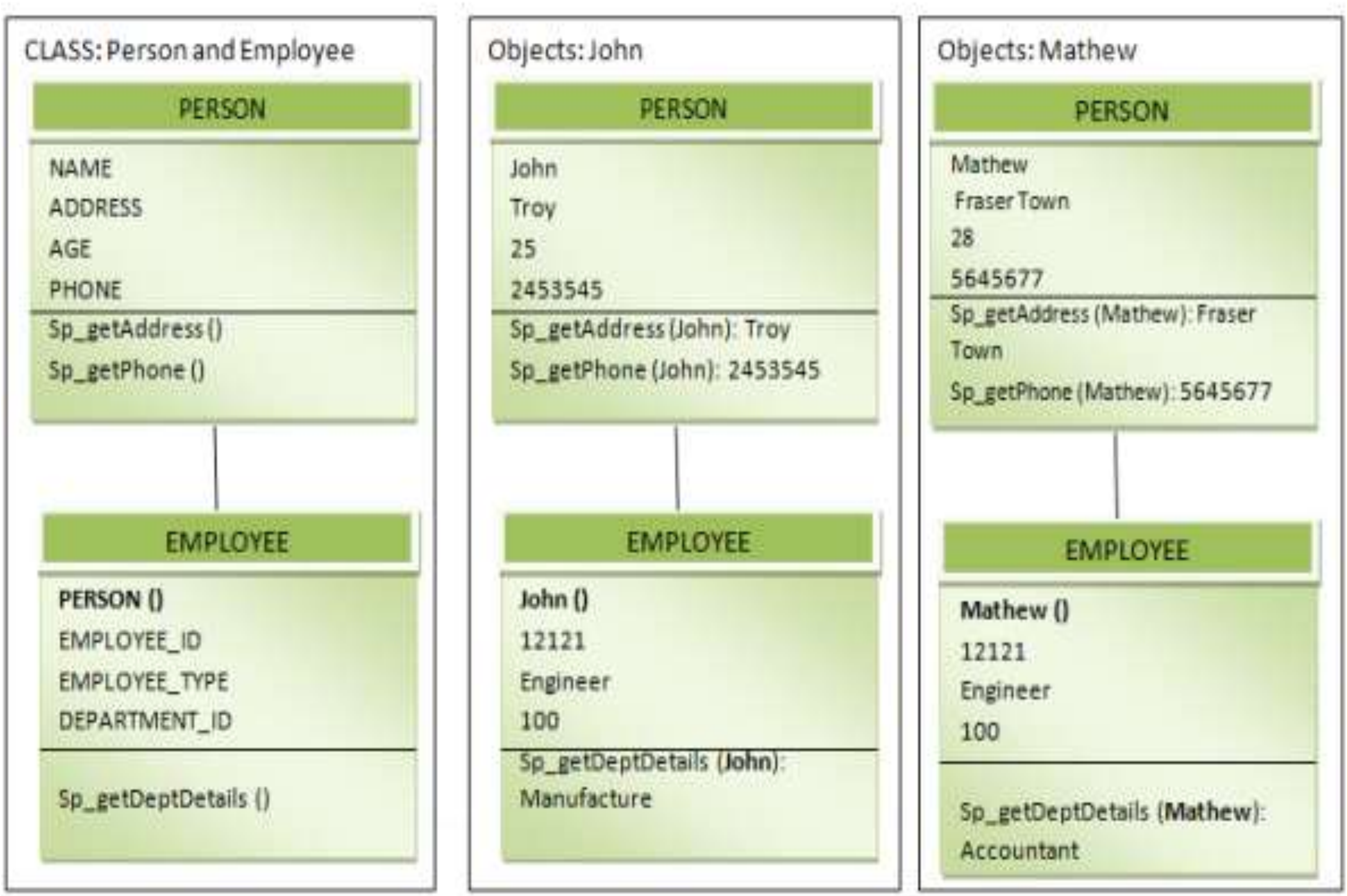
- No standard notations are available for ER diagram.
- It is meant for high level designs .

Object oriented data model

- Along with the mapping between the entities, describes the state of each entity and the tasks performed by them.

- It considers each object in the world as objects and isolates it from each other. It groups the related functionalities together and allows inheriting its functionality to other related sub-groups.

CONTD..


CLASS: Person and Employee
PERSON
NAME
ADDRESS
AGE
PHONE
Sp_getAddress ()
Sp_getPhone ()
EMPLOYEE
PERSON ()
EMPLOYEE_ID
EMPLOYEE_TYPE
DEPARTMENT_ID
Sp_getDeptDetails ()
Objects: John
PERSON
John
Troy
25
2453545
Sp_getAddress (John): Troy
Sp_getPhone (John): 2453545
EMPLOYEE
John ()
12121
Engineer
100
Sp_getDeptDetails (John): Manufacture
Objects: Mathew
PERSON
Mathew
Fraser Town
28
5645677
Sp_getAddress (Mathew): Fraser Town
Sp_getPhone (Mathew): 5645677
EMPLOYEE
Mathew ()
12121
Engineer
100
Sp_getDeptDetails (Mathew): Accountant

Advantages:

- Because of its inheritance property, we can re-use the attributes and functionalities.
- If we need any new feature we can easily add new class inherited from parent class.

Disadvantages:

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.

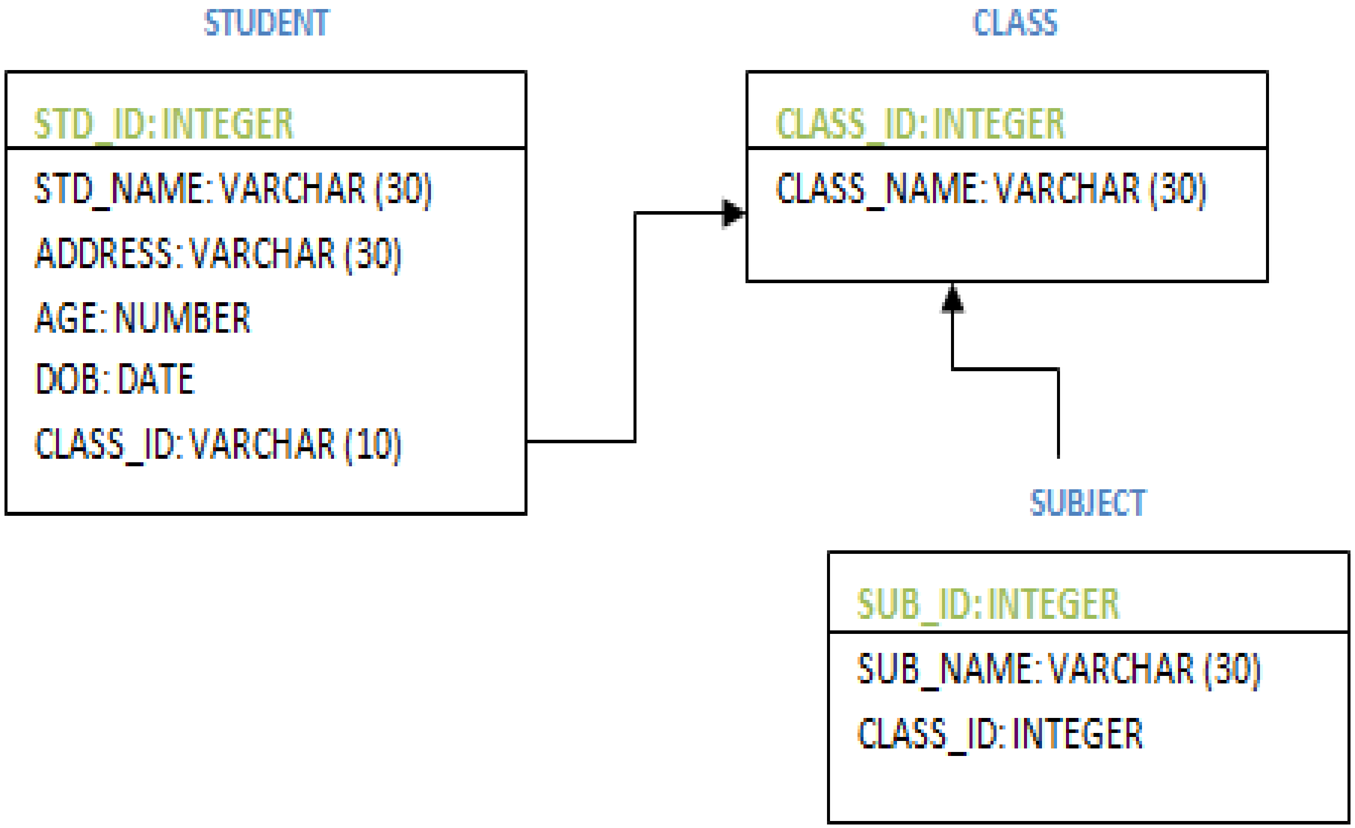
Physical Data Models

- It describes how data are stored in computer memory, how they are scattered and ordered in the memory and how they would be retrieved from memory.

- It represents each table, their columns and specifications, constraints like primary key, foreign key etc.

- It is represented as UML diagram along with table and its columns. Primary key is represented at the top.

Contd..


STUDENT
STD_ID:INTEGER
STD_NAME:VARCHAR (30)
ADDRESS:VARCHAR (30)
AGE:NUMBER
DOB:DATE
CLASS_ID:VARCHAR (10)
CLASS
CLASS_ID:INTEGER
CLASS_NAME:VARCHAR (30)
SUBJECT
SUB_ID:INTEGER
SUB_NAME:VARCHAR (30)
CLASS_ID:INTEGER

Record based data model

- These data models are based on application and user levels of data. This data models defines the actual relationship between the data in the entities.

- There are 3 types of record based data models

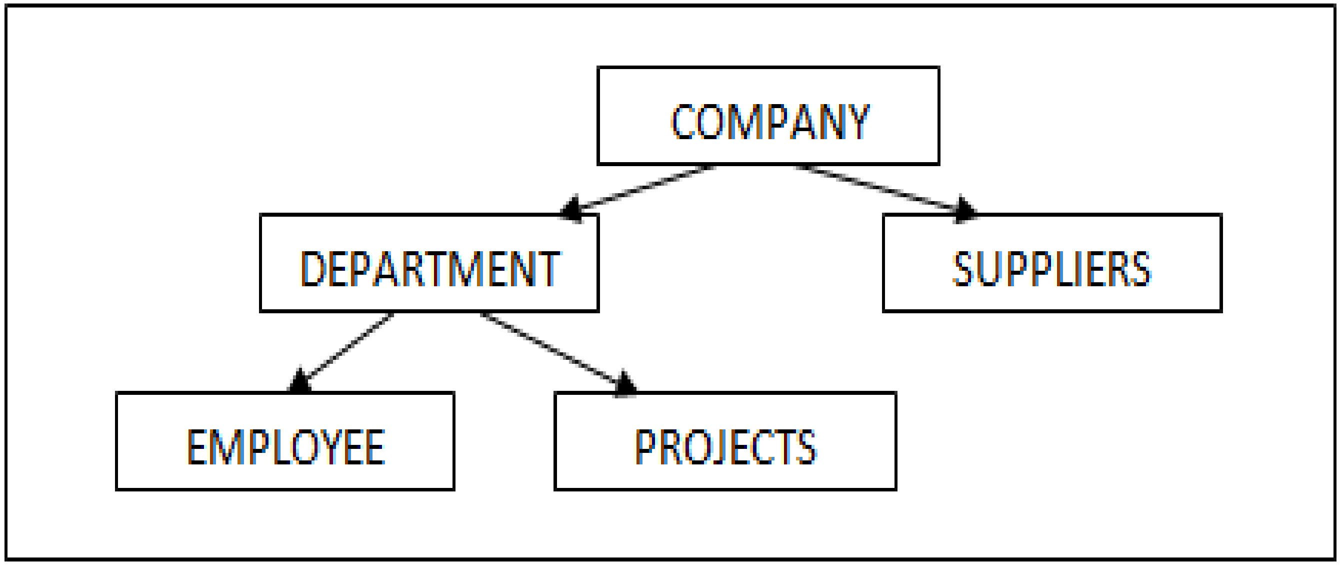
  a. Hierarchical data model

  b. Network data model

  c. Relational data models.

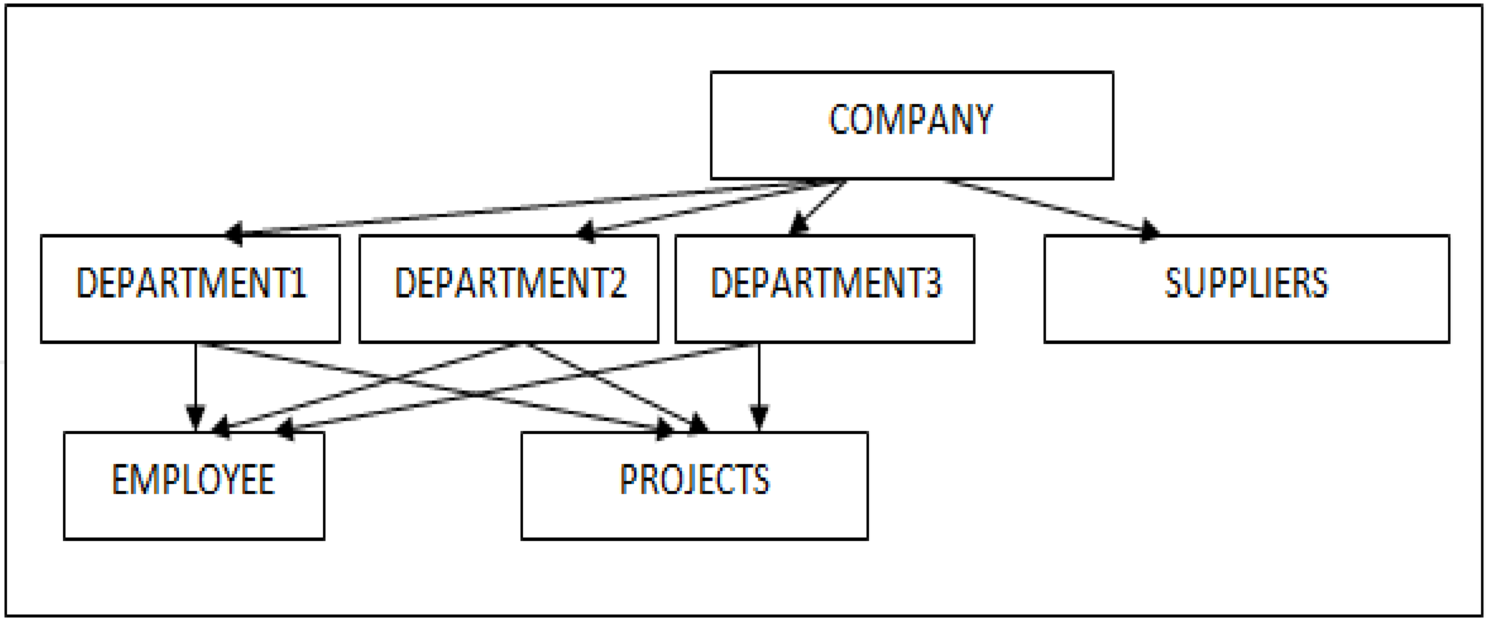
Hierarchical Data Models

- In this data model, the entities are represented in a hierarchical fashion (tree like structure). Here we identify a parent entity and its child entity.



CONTD..

Example: One company has multiple departments (1:N), one company has multiple suppliers (1:N),one department has multiple employees (1:N), each department has multiple projects(1:N).

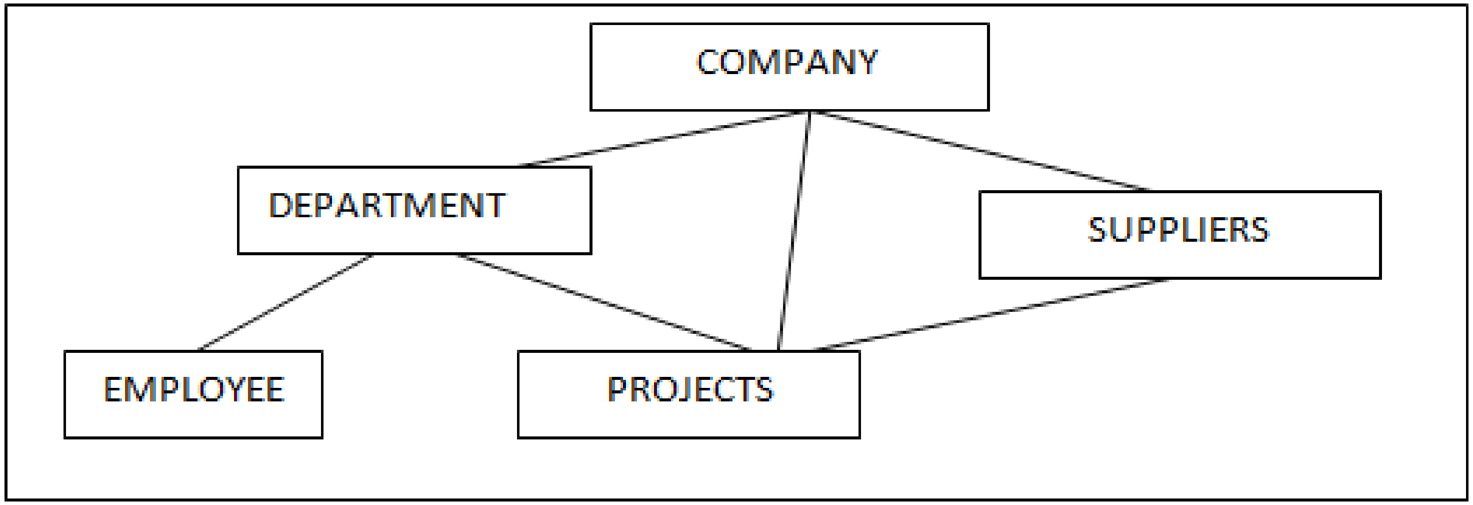


Disadvantages:

- Redundancy: In such case, we have to store same project information for more than one department. This is duplication of data.
- It fails to handle many to many relationships efficiently.
- If we need to fetch any, we have to start from the root and traverse through its child till we get the result.

NETWORK DATA MODEL

- It is designed to address the drawbacks of the hierarchical model. It helps to address M:N relationship.
- This model will not have single parent concept. Any child in the tree can have multiple parents.



Advantages:

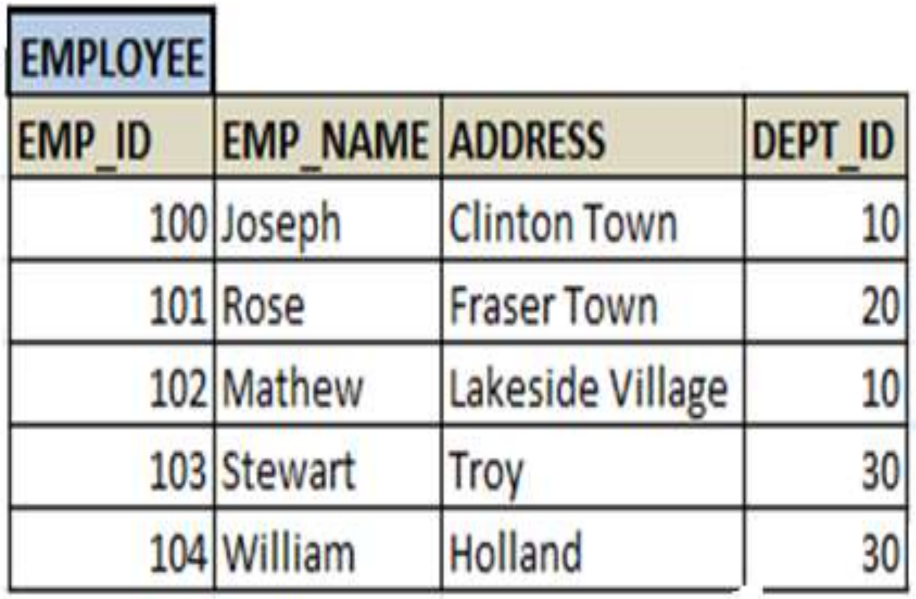
- Addresses many to many relationships.
- One can easily navigate among the tables and get any data.

Disadvantages:

- There is no independence between any objects .

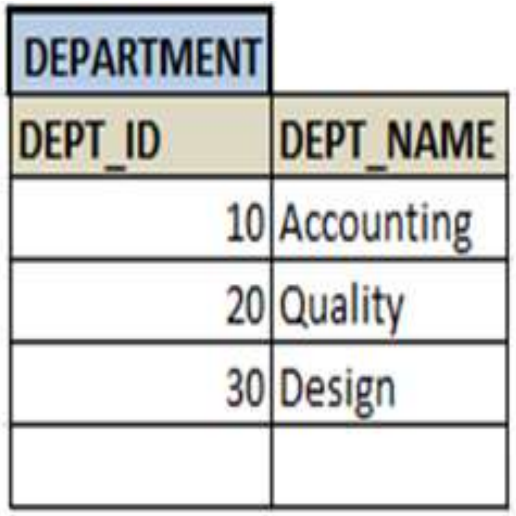
Relational data model

- It overcome the drawbacks of hierarchical and network models.
- This models define how they are structured in the database physically and how they are interrelated.



EMPLOYEE

EMP_ID	EMP_NAME	ADDRESS	DEPT_ID
100	Joseph	Clinton Town	10
101	Rose	Fraser Town	20
102	Mathew	Lakeside Village	10
103	Stewart	Troy	30
104	William	Holland	30



DEPARTMENT

DEPT_ID	DEPT_NAME
10	Accounting
20	Quality
30	Design

- A relational data model revolves around 5 important rules.

1. Order of rows / records in the table is not important. Example, displaying the records for Joseph is independent of displaying the records for Rose or Mathew in Employee table.

- It does not change the meaning or level of them. Each record in the table is independent of other.
- Similarly, order of columns in the table is not important. That means, the value in each column for a record is independent of other.

For example, representing DEPT_ID at the end or at the beginning in the employee table does not have any affect.

2. Each record in the table can be maintained unique. That is there is no duplicate record exists in the table.

This is achieved by the use of primary key or unique constraint.

3. Each column/attribute will have single value in a row. For example, in Department table, DEPT_NAME column cannot have ‘Accounting’ and ‘Quality’ together in a single cell. Both has to be in two different rows as shown above.

- 4. All attribute values should be from same domain. That means each column should have meaningful value. For example, Age column cannot have dates in it. It should contain only valid numbers to represent individual's age. Similarly, name columns should have valid names, Date columns should have proper dates.
- 5. Table names in the database should be unique. In the database, same schema cannot contain two or more tables with same name. But two tables with different names can have same column names. But same column name is not allowed in the same table.

Advantages

- Structural independence: Any changes to the database structure, does not affect the way we are accessing the data.
- Simplicity

Disadvantages

- Design will be designed till the minute level, which will lead to complexity in the database.

Importance of data model

1. **Higher quality:** A data model helps define the problem, enabling us to consider different Approaches and choose the best one.

2. **Reduced cost:** You can build applications at lower cost via data models. Data modeling typically consumes less than 10 percent of a project budget, and can reduce the 70 percent of budget that is typically devoted to programming.

- The models promote clarity of thought and provide the basis for generating much of the needed database and programming code.

3. **Quicker time to market**. You can also build software faster by catching errors early. In addition, a data model can automate some tasks – design tools can take a model as an input and generate the initial database structure, as well as some data access code.

4. **Clearer scope**. A data model provides a focus for determining scope. It provides something tangible to help business sponsors and developers agree over precisely what is included with the software and what is omitted.

5. **Faster performance.** A sound model simplifies database tuning. A well-constructed database typically runs fast, often quicker than expected.

6. **Better documentation**. Models document important concepts and jargon, proving a basis for long-term maintenance.

7. **Fewer application errors.** A data model causes participants to crisply define concepts and resolve confusion. As a result, application development starts with a clear vision. Developers can still make detailed errors as they write application code, but they are less likely to make deep errors that are difficult to resolve.

8. **Managed risk.** You can use a data model to estimate the complexity of software, and gain insight into the level of development effort and project risk. We should consider the size of a model, as well as the intensity of inter-table connections.

9. **A good start for data mining.** The documentation inherent in a model serves as a starting point for analytical data mining.

Data Model Basic Building Blocks

The basic building blocks of all data models are

1. Entities
2. Attributes
3. Relationships and constraints.

CONTD..

An **Entity** is real world thing, such as a person, place, thing, or event, about which data are to be collected and stored.

Example: CUSTOMER, STUDENT, EMPLOYEE

An **Attribute** is a characteristic of an entity.

Example: a CUSTOMER - customer last name, customer first name, customer phone.

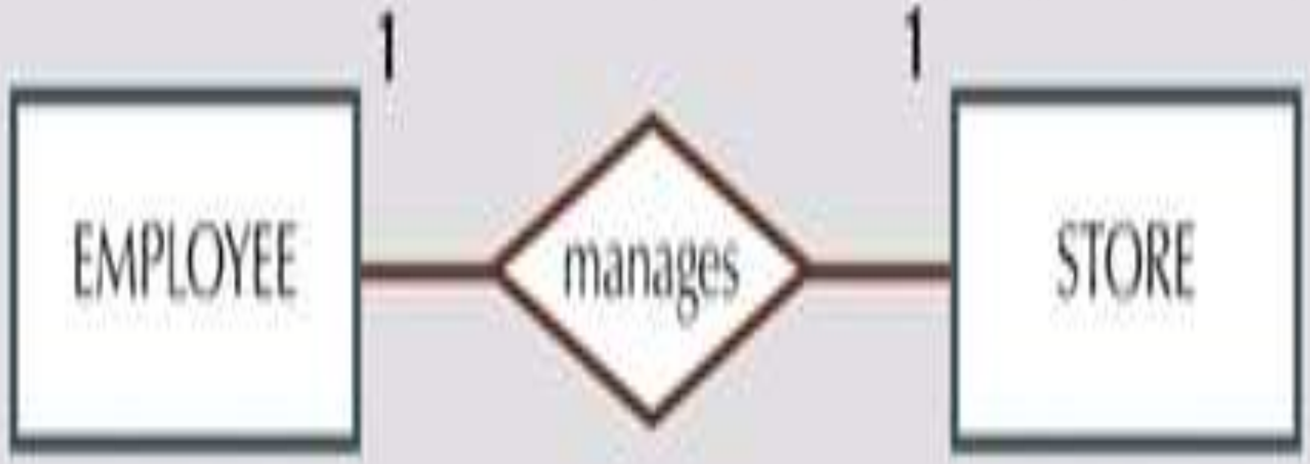
Contd..

- A **Relationship** describes an association among (two or more) entities

- There are three types of relationships:
  1. **one-to-one**
  2. **one-to-many**
  3. **many-to-many**

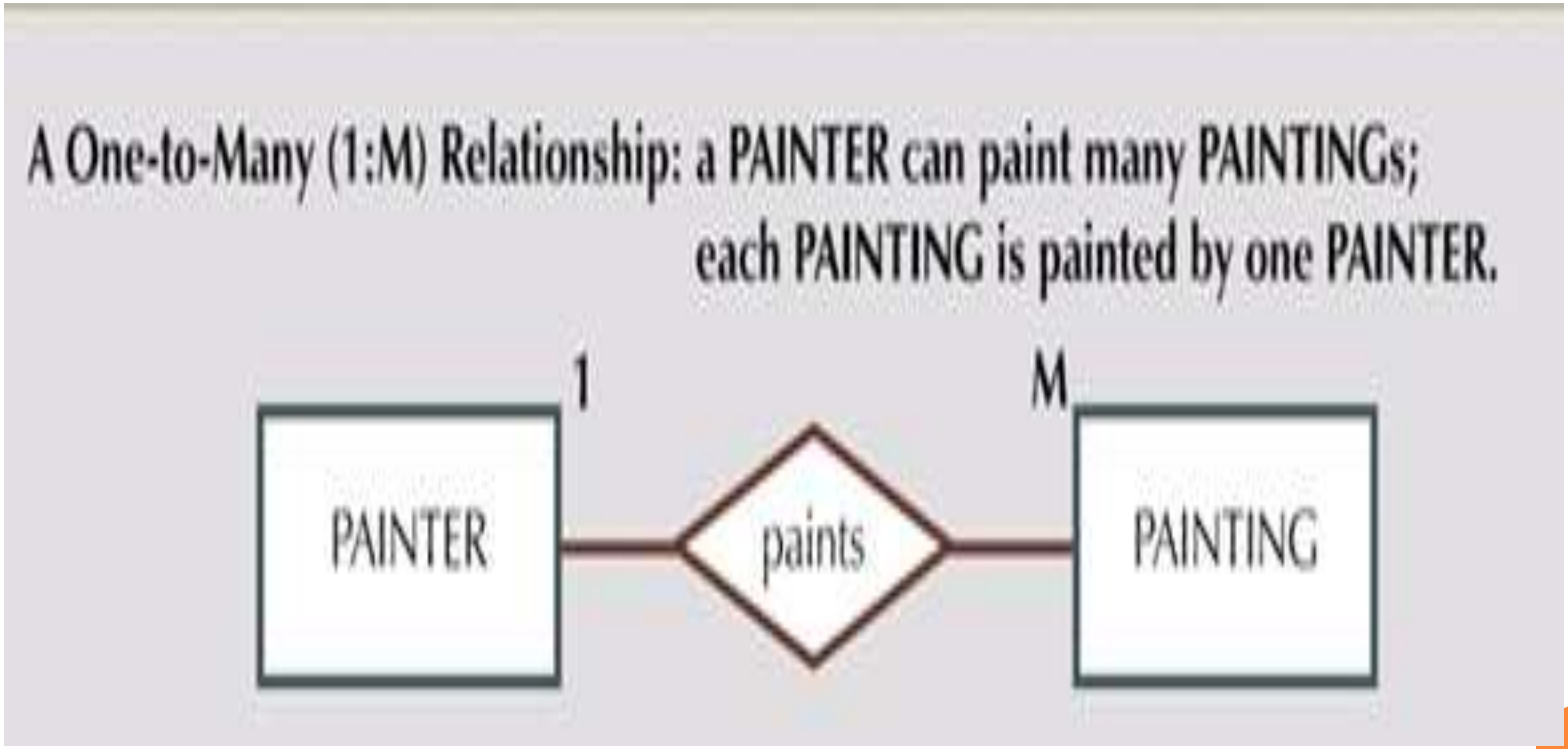
One-to-one relationship (1:1)

A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.



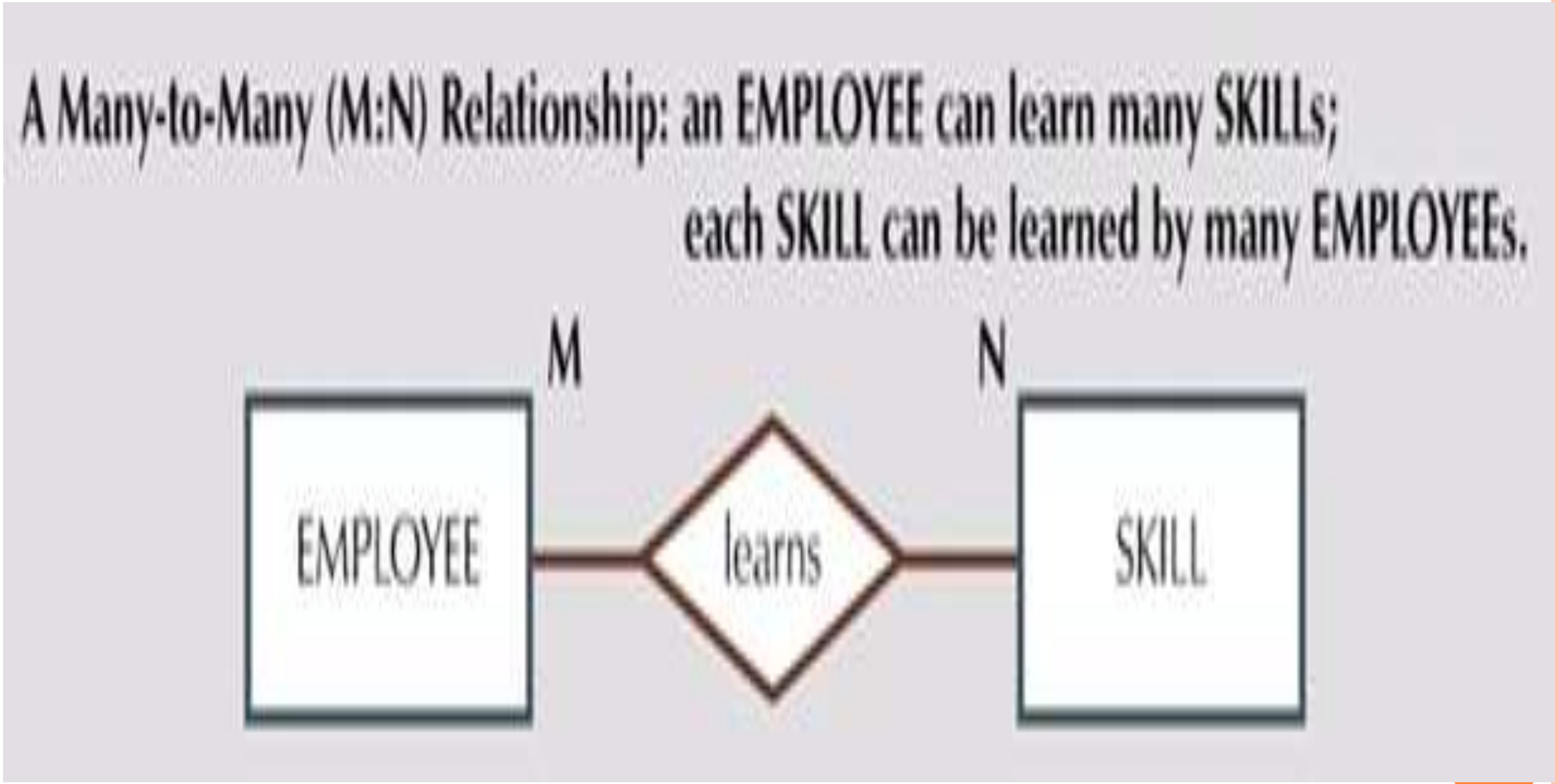
Contd..

One-to-many relationship (1:M)


A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.

Contd..

Many-to-many relationship (M:N)


A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.

Contd..

- **Constraint**: is a restriction placed on the data. Constraints are important because they help to ensure data integrity.

  Example: An employee's salary must have values that are between 6000 and 35000.

ENTITY-RELATIONSHIP MODEL (ER Model)

ER MODEL

- A Data Model is a logical structure of Database.
- It describes the design of database to reflect entities, attributes, relationship among data, constraints etc.
- A data model is a conceptual representation of the data that are required by a database.
- The Entity Relationship (ER) Model allows us to describe the data involved in a real-world in terms of objects and their relationships and is widely used to develop an initial database design.

ER-Model

- ER-Models are used in the design of conceptual schemas for database applications. The diagrammatic notation associated with the ER model, known as **ER diagrams.**

- The ER model describes data as entities, relationships and attributes.

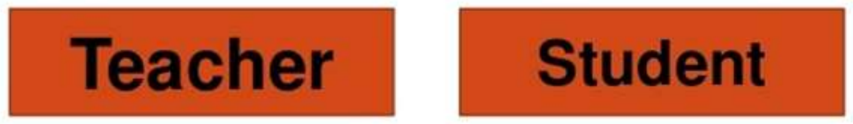
ENTITY , ATTRIBUTES

Entity:

Which is a thing in the real world with an independent existence.

Examples: STUDENT, EMPLOYEE, TEACHER etc.

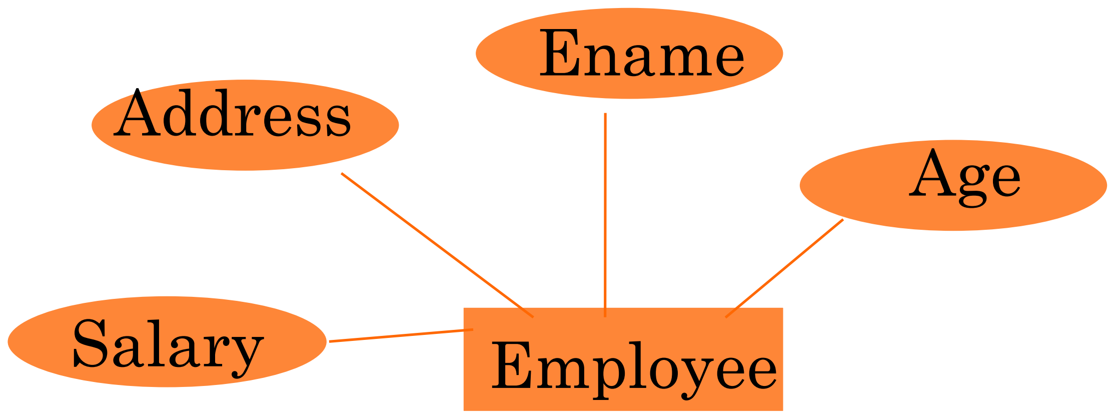
In ER diagram entity is represented with rectangle box.



Attributes :

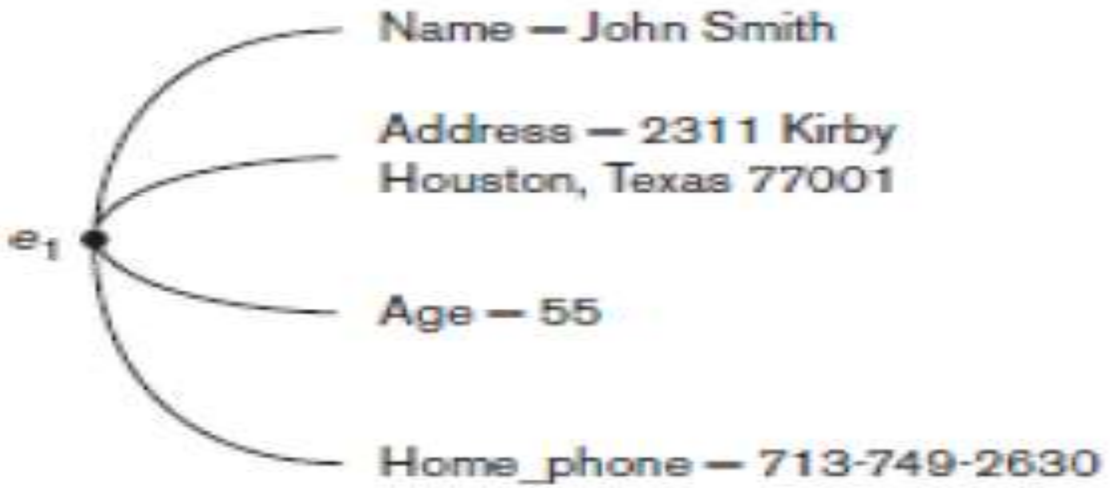
Each entity has attributes, which are the particular properties that describe it.

Example: EMPLOYEE - employee’s name, age, address, salary. Is represented as oval.



Contd..

A particular entity will have a **value for each of its attributes.**



e_1
Name = John Smith
Address = 2311 Kirby Houston, Texas 77001
Age = 55
Home_phone = 713-749-2630

TYPES OF ATTRIBUTES

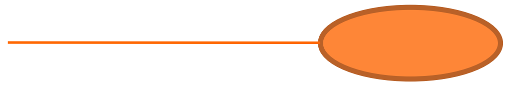
- Simple Attribute
- Composite Attribute
- Single value Attribute
- Multi-value Attribute
- Stored Attribute
- Derived Attribute
- Null-value Attribute
- Key Attribute

Contd..

Simple Attribute: are atomic values, which cannot be divided further.

Example: Age attribute of student entity

Representation:

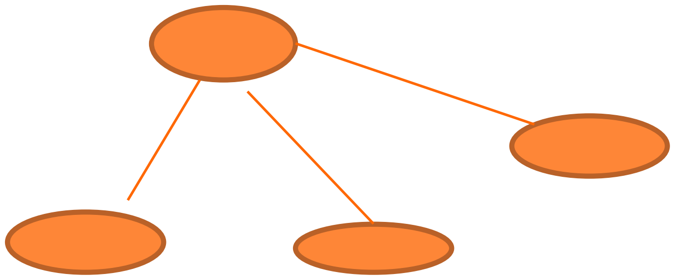


Composite attribute:

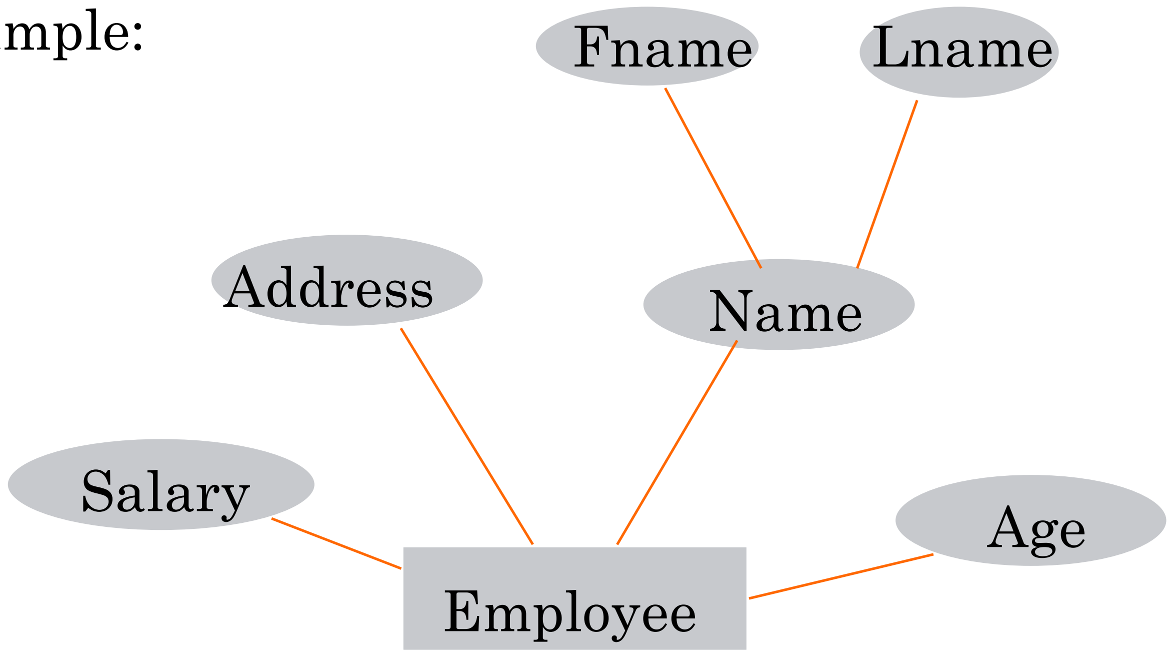
An attribute that can be divided into smaller independent attribute.

Example: a student's name may divided into first name and lastname.

Representation:



Composite attribute
Example:



Single valued attribute :

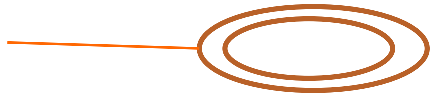
An attribute that has only single value for an entity.

Example: Any manufactured product can have only one serial no.

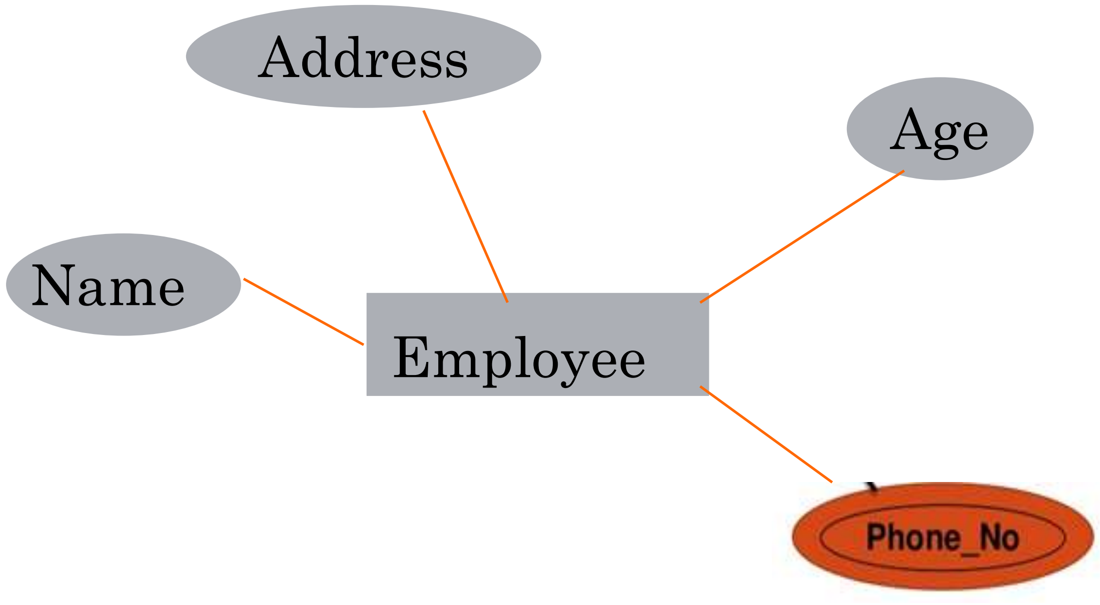
Multi valued attribute :

An attribute that can have multiple values for an entity .

Example: Phone no, email address



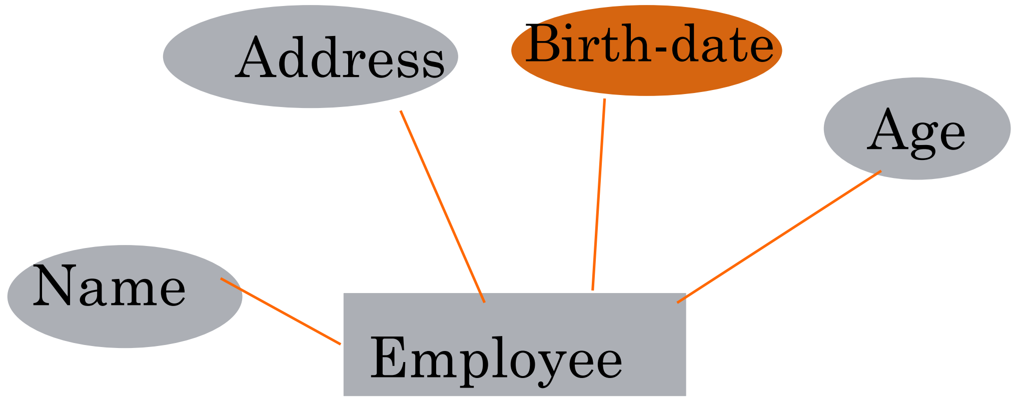
Multi-valued attribute:



Contd..

Stored attribute :

An attribute that cannot be derived from another attribute. Example: birth date cannot derive from age of student.

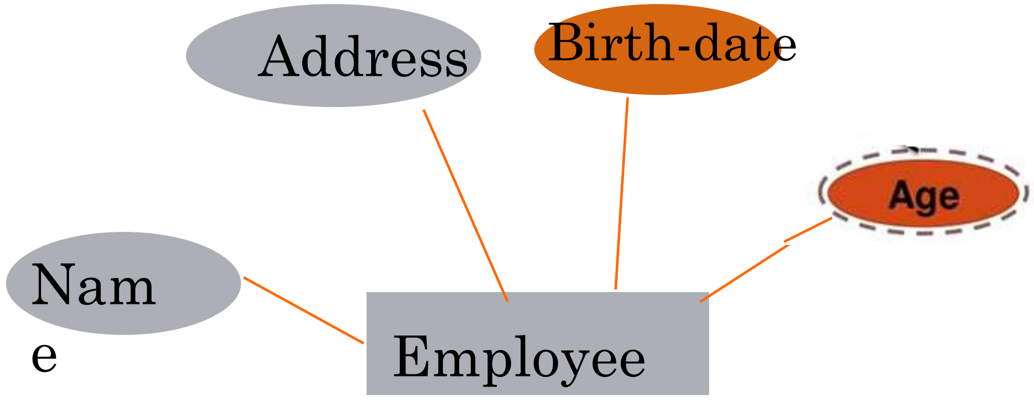


CONTD..

Derived attribute:

An attribute that can be derived from another attribute is known as derived attribute.

Example: age attribute of Employee entity



Null valued attribute:

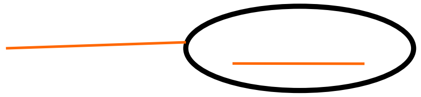
An attribute, which has not any value for an entity.

Example: Passport attribute of student may be null.

Key attribute:

An attribute that has unique value of each entity is known as **key attribute.**

Example:every student has unique roll no. Here roll no is **key attribute**.



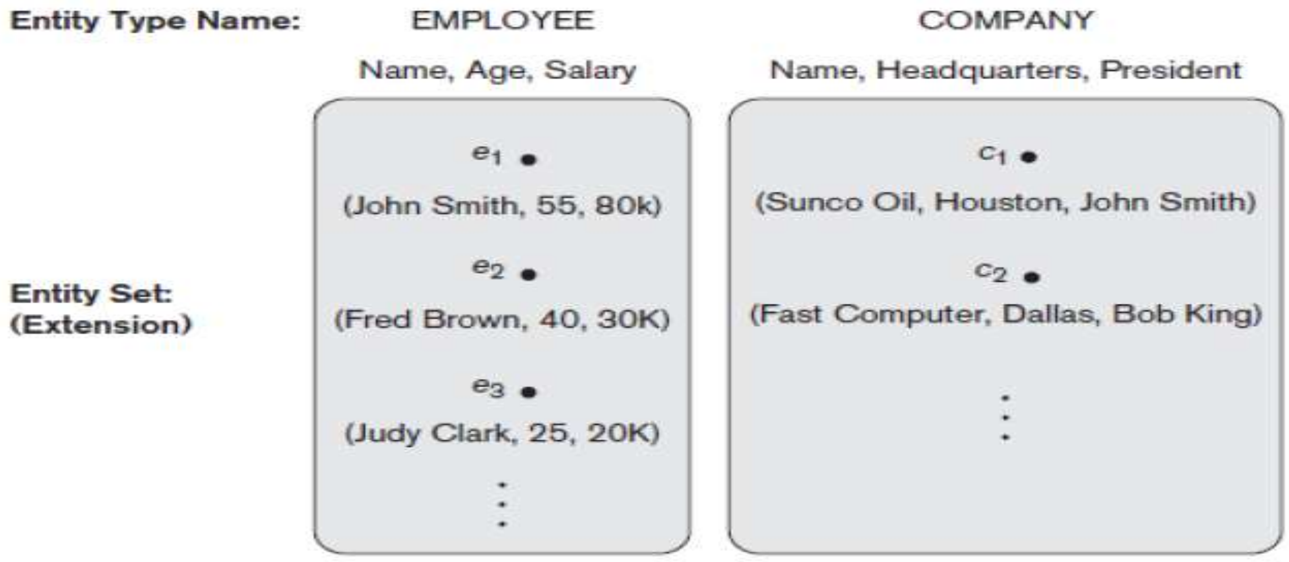
Complex Attributes:

- Composite and Multivalued attributes can be nested arbitrarily.Such attributes are called as complex attributes.
- Composite attributes can be represented by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multivalued attributes between braces { }.

Example:If a person can have more than one residence and each residence can have a single address and multiplephones. Both Phone and Address are themselves composite attributes.

Entity Types and Entity Sets

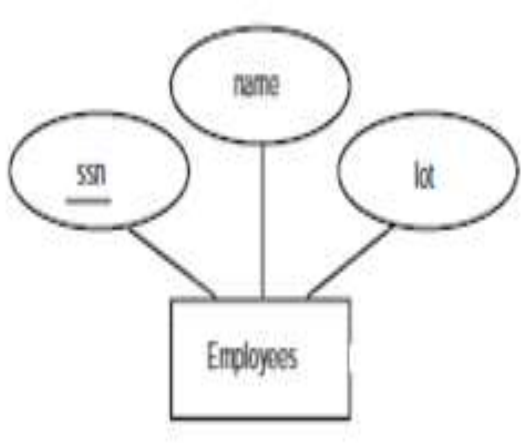
- An entity type defines a collection of entities that have the same attributes.
- The collection of all entities of a particular entity type in the database is called an **entitv set.**



Key Attributes of an Entity Type

Key plays an important role in database; it is used for identifying unique rows from entity.

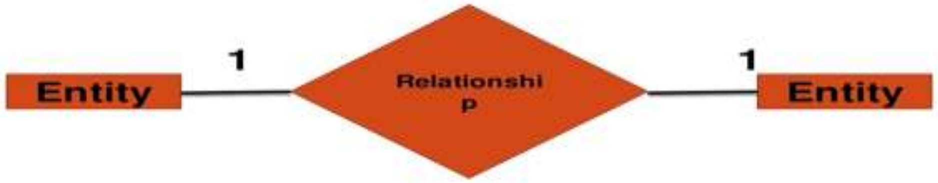
Example: ssn attribute of Employee entity.



ssn	name	age
1	Amith	30
2	Ajay	30
3	Rohith	25

Relationship Types, Sets

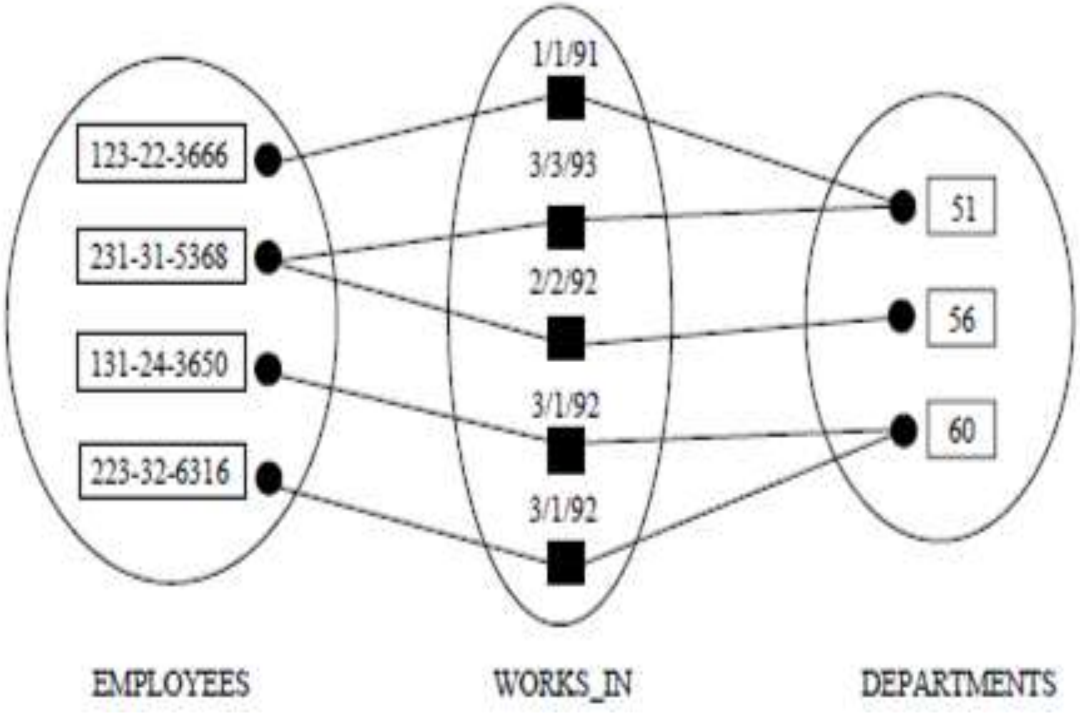
A Relationship is a association among two or more entity sets.
Example: DEPARTMENT refers to an EMPLOYEE who manages the department.



A Relationship set is a set of relationships of the same type
Descriptive attributes are used to record information about the relationship

Example: consider a relationship type WORKS_IN between the two entity types EMPLOYEE and DEPARTMENT

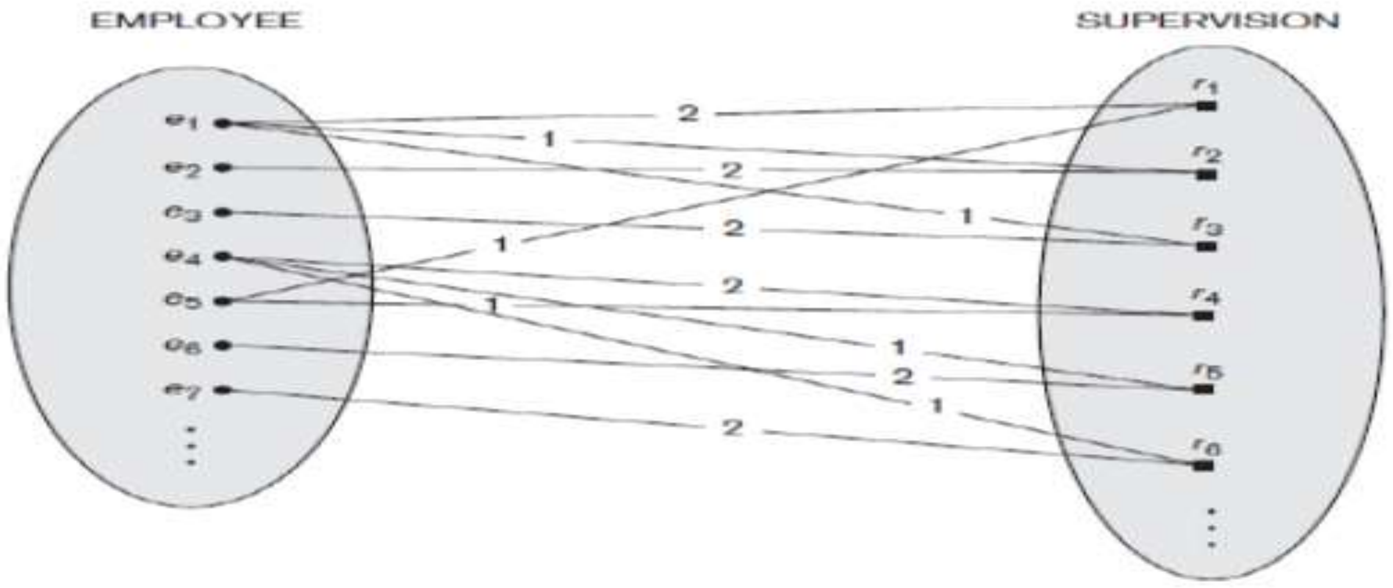
Contd..

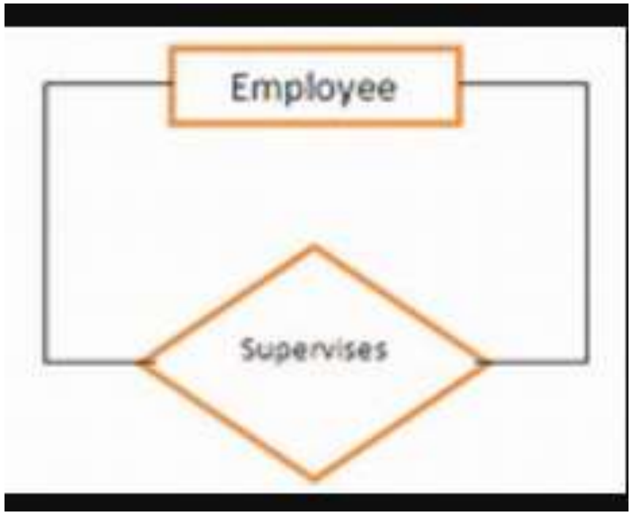


123-22-3666
231-31-5368
131-24-3650
223-32-6316
1/1/91
3/3/93
2/2/92
3/1/92
3/1/92
51
56
60
EMPLOYEES
WORKS_IN
DEPARTMENTS

Relationship Degree

Unary Relationship : if number of participating entity type is only one then its degree is one. Also called recursive relationship.





Consider both emp1and emp2 are entities in Employees.

However, they play different roles: emp1 reports to the managing employee emp2, which is reflected in the role indicators supervisor and subordinate

CONTD..

Binary Relationship : if number of participating entity type is two then its degree is two.

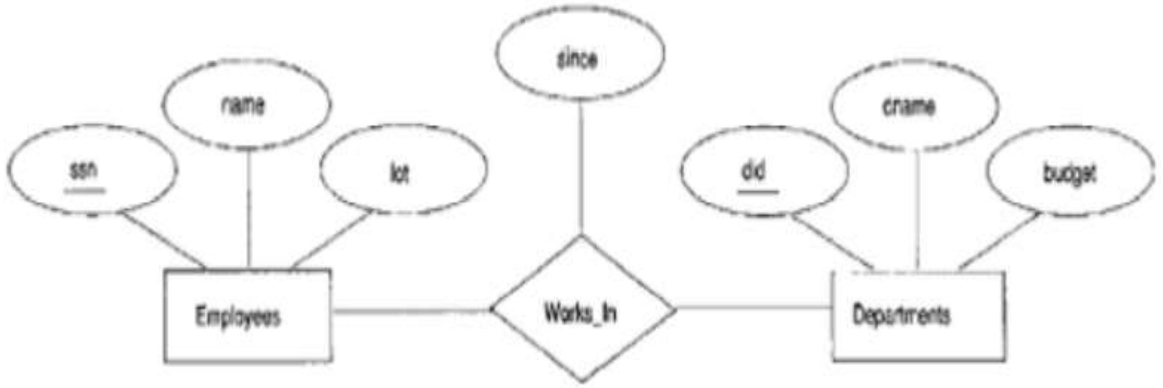
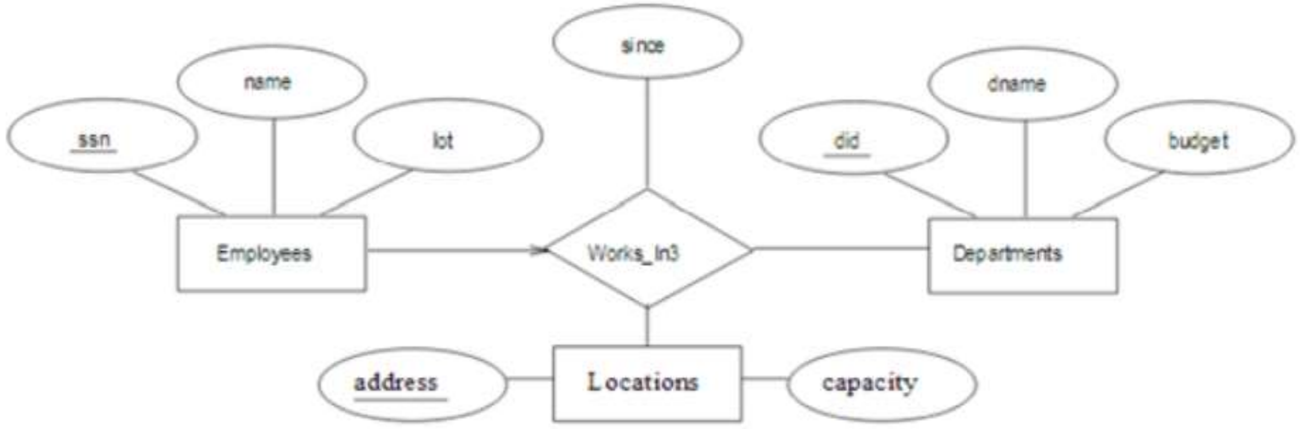


Figure Binary Relationship

Ternary relationship : if number of participating entity type is three then its degree is three.

From the below figure the WORKS_IN relationship type is of degree three since three entity types EMPLOYEE ,LOCATION and DEPARTMENT participate in the relationship.

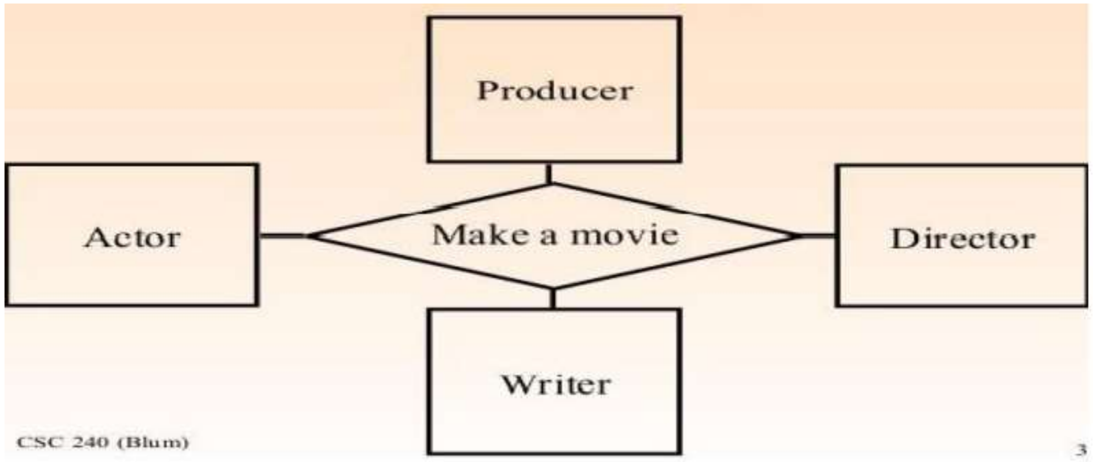


ER DIAGRAM TERNARY RELATIONSHIP

QUARTERNARY RELATIONSHIP

- A relationship that has association with four entities is known as Quarternary Relationship.

Example: The Make a movie relationship type is of degree four since four entity types Actor, Producer, Writer, Director participate in the relationship.



Constraints on Binary Relationship Types

Relationship types usually have certain constraints that limit the possible combinations of entities that may participate.

There are two main types of binary relationship constraints:

1. Cardinality ratio and
2. Participation Constraints

Mapping Cardinalities:

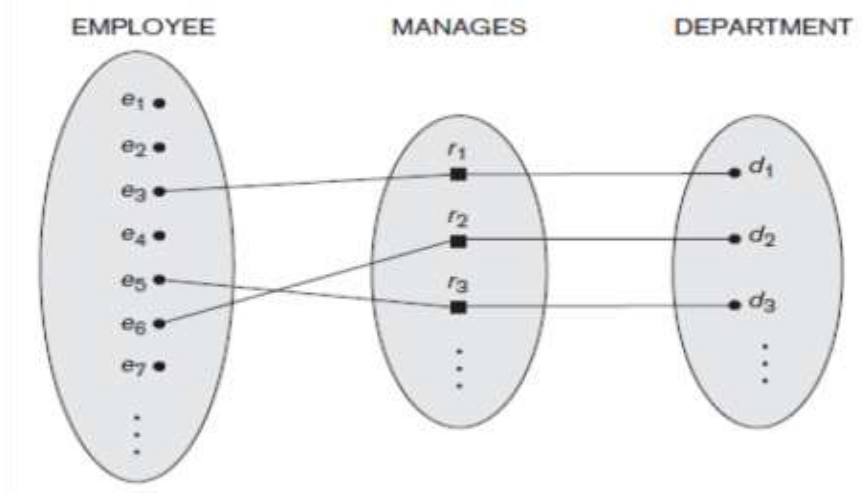
- It expresses the number of entities to which another entity can be associated via a relationship set.

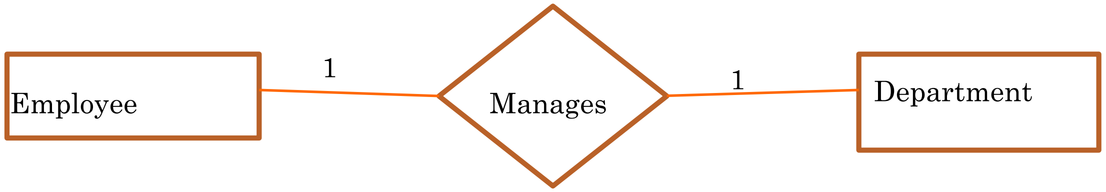
- Most useful in describing binary relationship sets.

- For a binary relationship set the mapping cardinality must be one of the following types: i.e., The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

Contd..

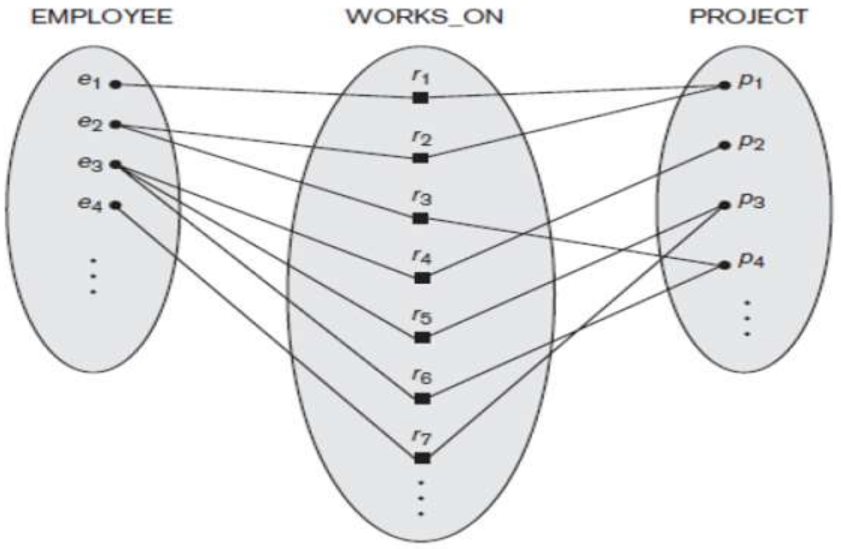
Example of 1: 1 relationship

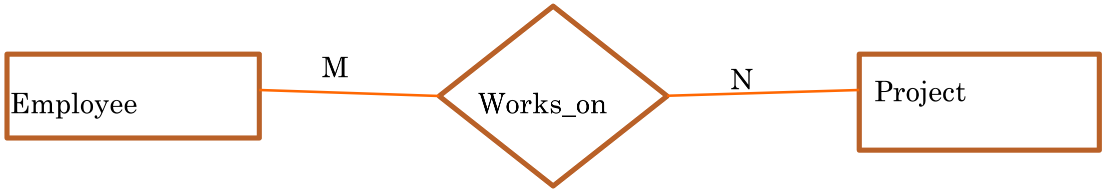




Contd..

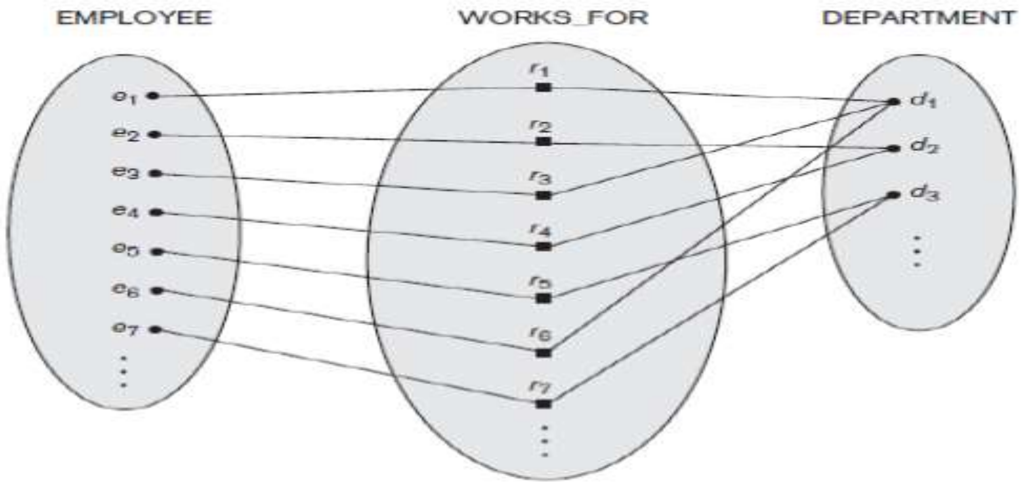
Example of M:N relationship

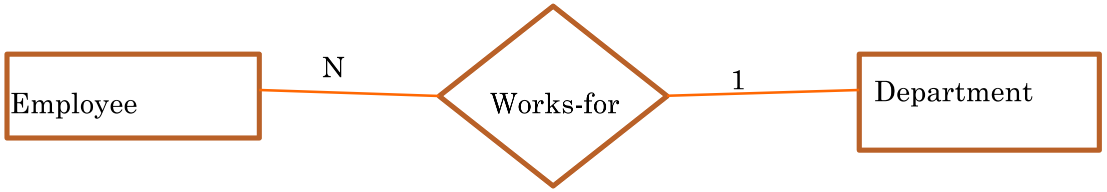




Contd..

Example of N : 1 relationship





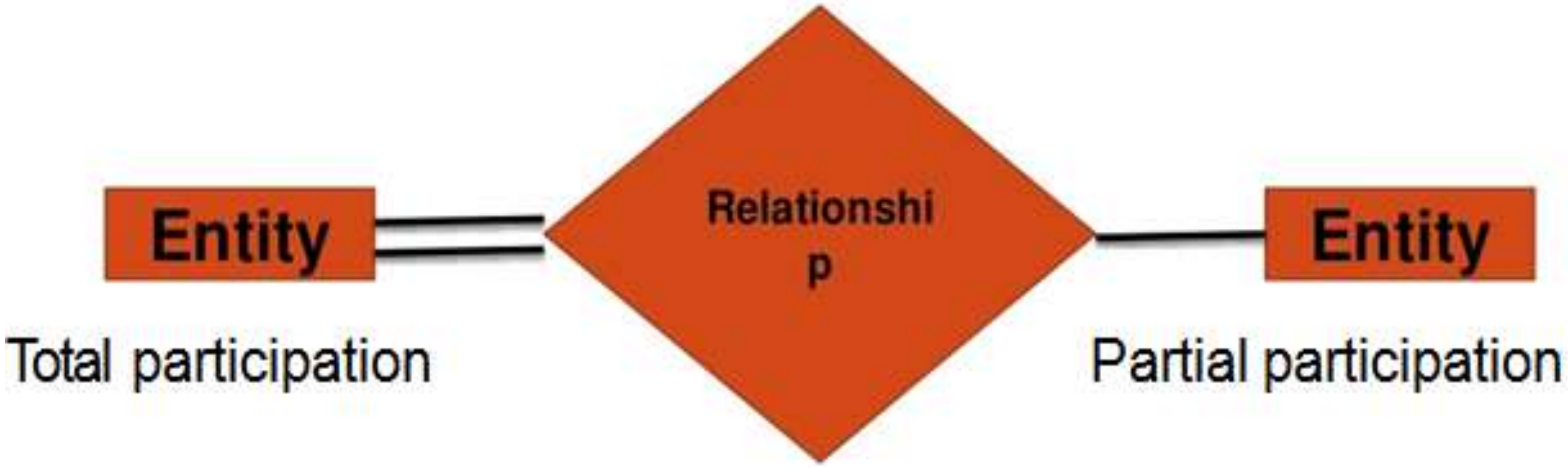
Participation Constraints

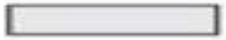
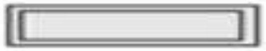
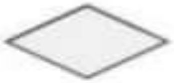
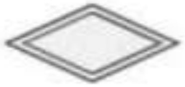
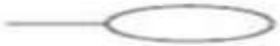
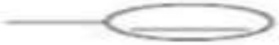
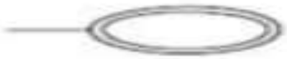
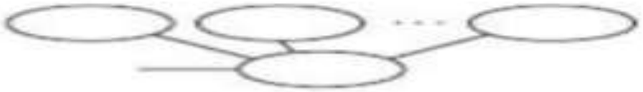
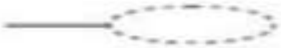
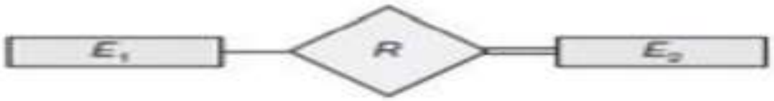
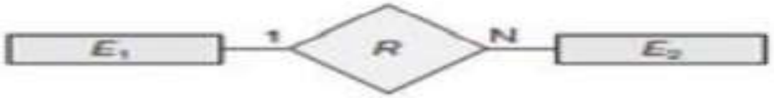
This constraint specifies the minimum number of relationship instances that each entity can participate in.

There are two types of participation constraints

1. Total participation constraints
2. Partial participation constraints

Representation:



Symbols	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1 : N for E_1 : E_2 in R

KEYS

KEYS: Any attribute in the table which uniquely identifies each record in the table is called key. It can be a single attribute or a combination of attributes.

- For example, in STUDENT table, STUDENT_ID is a key, since it is unique for each student.
- In PERSON table, his passport number, driving license number, phone number, SSN, email address is keys since they are unique for each person.

Why we need a Key?

- In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.
- Also, tables store a lot of data in them. Tables generally extends to thousands of records stored in them, unsorted and unorganized.
- Now to fetch any particular record from such dataset, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the wrong data. To avoid all this, Keys are defined to easily identify any row of data in a table SUPER KEY

Super Key:

Super Key is defined as a set of attributes within a table that can uniquely identify each record within a table. Super Key is a superset of Candidate key.

- In the table super key would include student_id, (student_id, name), phone etc.

The student_id is unique for every row of data, hence it can be used to identity each row uniquely.

- Next comes, (student_id, name), now name of two students can be same, but their student_id can't be same hence this combination can also be a key.

Candidate Key

- The minimal set of attribute which can uniquely identify a tuple is known as candidate key.

• In the example, an employee is identified by his ID in his office. Apart from his ID, passport number, PAN number, driving license number, email address etc also identifies a person uniquely.

But we can choose any one of these unique attribute as primary key in the table. Rest of the attributes, which holds as strong as primary key are considered as secondary key.

• In our example of employee table, EMPLOYEE_ID is best suited for primary key as its from his own employer.

PRIMARY KEY:

- It is the first and foremost key which is used to uniquely identify a record.

• It can be a single attribute or a combination of attributes. For an entity, there could be multiple keys as we saw in PERSON table.

• Most suitable key from those lists of candidate key becomes a primary key.

• In the Person table above, we can select Employee id as primary key, since it is unique for each person.

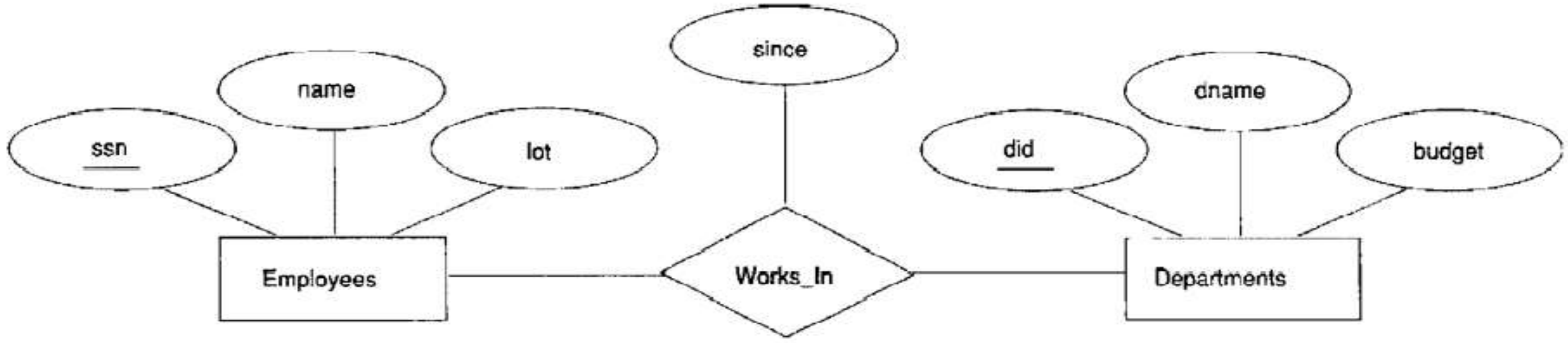
FOREIGN KEY

- In a company there would be different departments - Accounting, Human Resource (HR), development, Quality, etc.
- An employee, who works for that company, works in specific department. But we know that employee and department are two different entities.
- So we cannot store his department information in employee table. Instead what we do is we link these two tables by means of primary key of one of the table i.e.; In this case, we pick the primary key of department table - DEPARTMENT_ID and add it as a new attribute/column in the Employee table.
- Now ***DEPARTMENT_ID is a foreign key for Employee table***, and both the tables are related.

ADDITIONAL FEATURES OF ER-MODEL

1. KEY CONSTRAINTS

Consider the Works-.In relationship shown in Figure below. An employee can work in scvcral dcpartmcnts, and a dcpartmcnt can havc scvcral cmployccs.



- Consider another relationship set called Manages between the Employees and Departments entity sets such that each department have at most one manager, although a single employee is allowed to manage more than one department.

Each department has at most one manager is the restriction specified for department is an example of a **key constraint.**

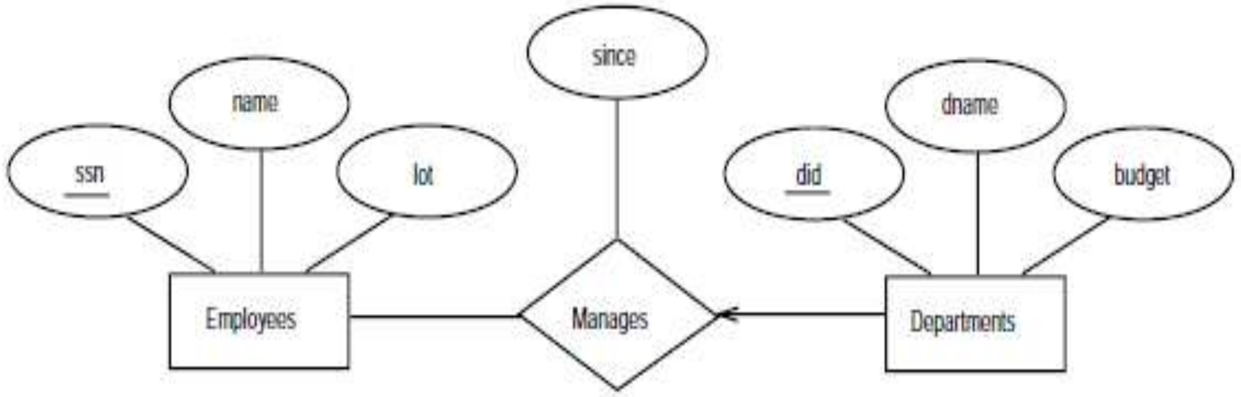
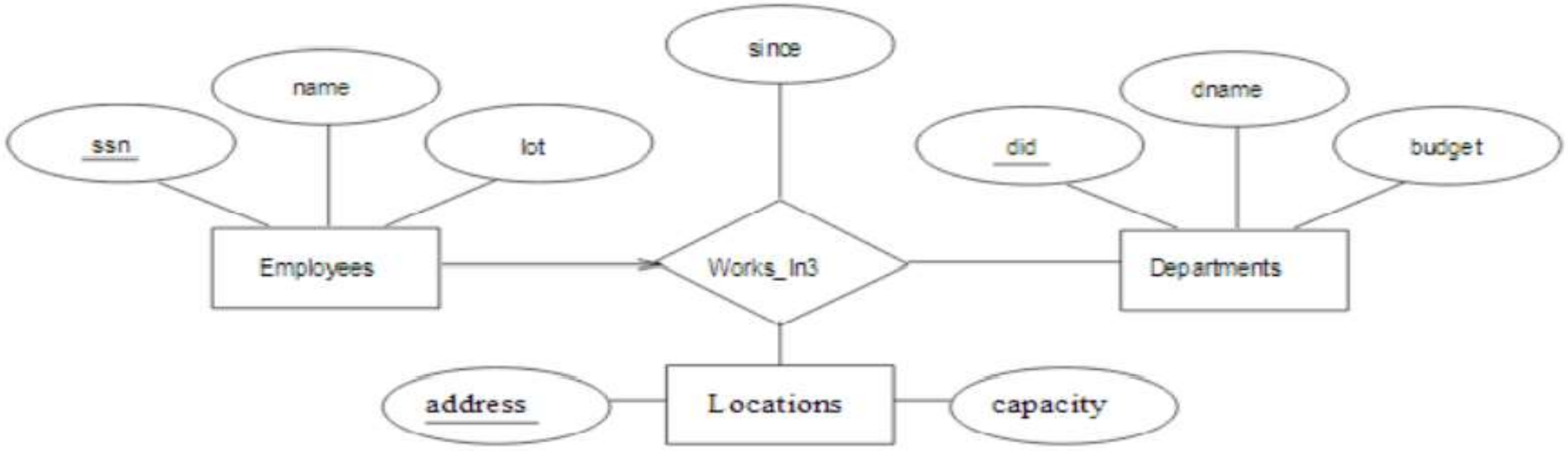


Figure 2.6 Key Constraint on Manages

- A relationship set like Manages is sometimes said to be one-to-many, to indicate that one employee can be associated with many departments, whereas each department can be associated with at most one employee as its manager.
- In contrast, the Works-.In relationship set, in which an employee is allowed to work in several departments and a department is allowed to have several employees, is said to be many-to-many.
- If we add the restriction that each employee can manage at most one department to the Manages relationship set, which would be indicated by adding an arrow from Employees to Manages in Figure we have a one-to-one relationship set.

2. KEY CONSTRAINTS FOR TERNARY RELATIONSHIPS:

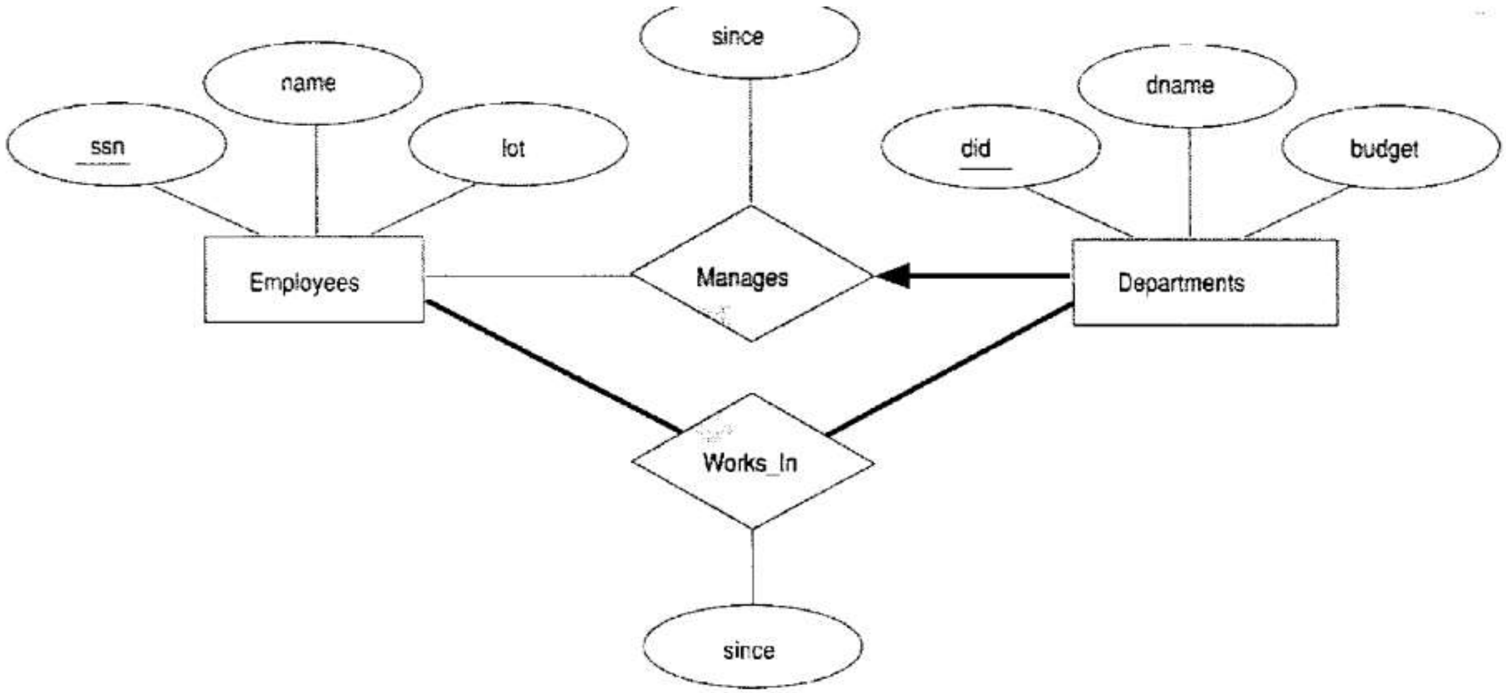
- If an entity set E has a key constraint in a relationship set R, each entity in an instance of E appears in at most one relationship in (a corresponding instance of) R.
- To indicate a key constraint on entity set E in relationship set R, we draw an arrow from E to R.
- In Figure, we show a ternary relationship with key constraints.
- Each employee works in at most one department and at a single location.



ER DIAGRAM TERNARY RELATIONS

3.PARTICIPATION CONSTRAINTS

- Entities can participate in a relationship either totally or partially.
- If every entity in an entity set participates in a relationship of relationshipset,then the participation is said to be total else partial.
- The key constraint on Manages tells us that a department has at most one manager.
- Let us say that every department is required to have a manager.The participation of the entity set Departments in the relationship set Manages is said to be total.
- A participation that is not total is said to be partial. As an example, the participation of the entity set Employees in Manages is partial, since not every employee gets to manage a department.



If the participation of an entity set in a relationship set is total, the two are connected by a thick line; the presence of an arrow indicates a key constraint.

4.WEAK ENTITIES:

- Entities are divided into two types: Strong and Weak entities.
- Entity which has primary key is known as strong entity.
- Weak entity is one that depends on strong entity for existence.
- A weak entity cannot be identified uniquely as it does not have sufficient entities to form a primary key.
- To make it uniquely identifiable weak entity set is associated with another entity set called identifying or owner entity set. The relationship among two entities is called identifying relationship.

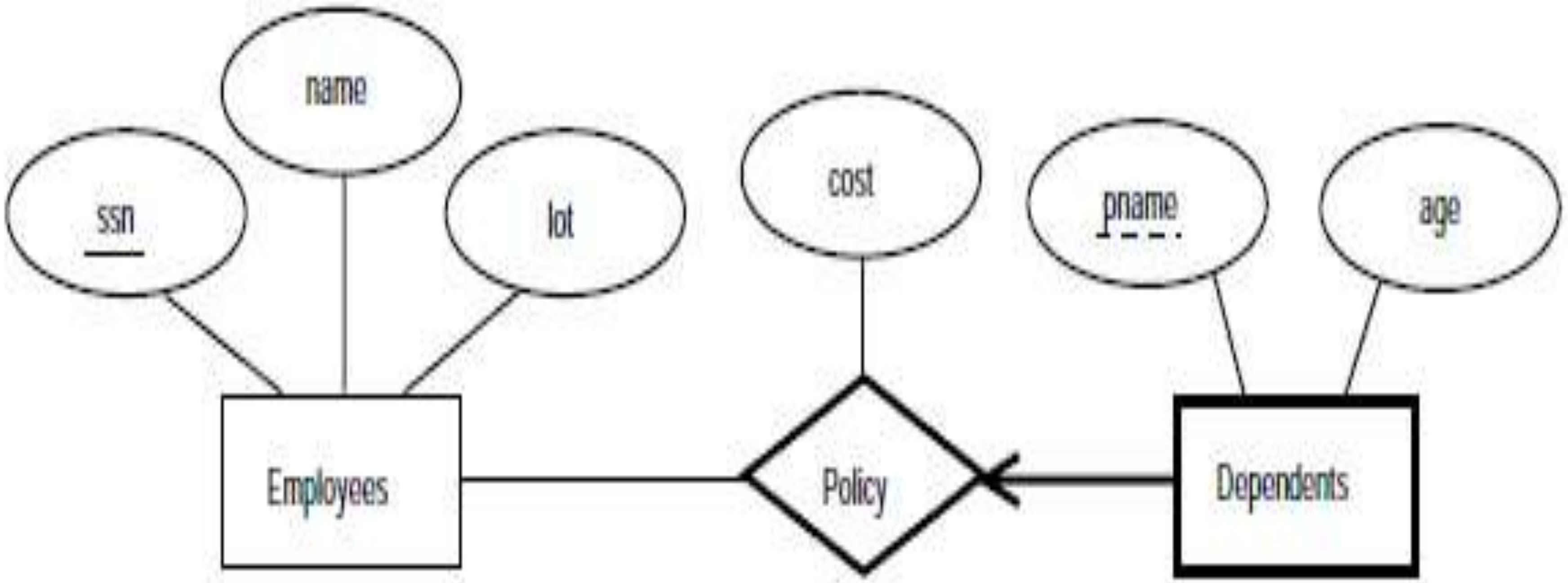


Figure 2.11 A Weak Entity Set

- If an employee quits, any policy owned by the employee is terminated and we want to delete all the relevant policy and dependent information from the database.
- We might choose to identify a dependent by name alone in this situation, since it is reasonable to expect that the dependents of a given employee have different names.
- Thus the attributes of the Dependents entity set might be pname and age. The attribute pname does not identify a dependent uniquely. Dependents is an example of a weak entity set.
- **A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, which is called the identifying owner**.

- The following restrictions must hold:

i) The owner entity set and the weak entity set must participate in a one to- many relationship set (one owner entity is associated with one or more weak entities, but each weak entity has a single owner). This relationship set is called the identifying relationship set of the weak entity set.

ii) The weak entity set must have total participation in the identifying relationship set. For example, a Dependents entity can be identified uniquely only if we take the key of the owning Employees entity and the pname of the Dependents entity. The set of attributes of a weak entity set that uniquely identify a weak entity for a given owner entity is called a partial key of the weak entity set. In our example, pname is a partial key for Dependents.

- The total participation of Dependents in Policy is indicated by linking them with a dark line.
- The arrow from Dependents to Policy indicates that each Dependents entity appears in at most one (indeed, exactly one, because of the participation constraint) Policy relationship.
- To underscore the fact that Dependents is a weak entity and Policy is its identifying relationship, we draw both with dark lines/double lines.
- To indicate that pname is a partial key for Dependents, we underline it using a broken line. This means that there may well be two dependents with the same pname value.

5.CLASS HIERARCHIES/ISA (`IS A') HIERARCHIES:

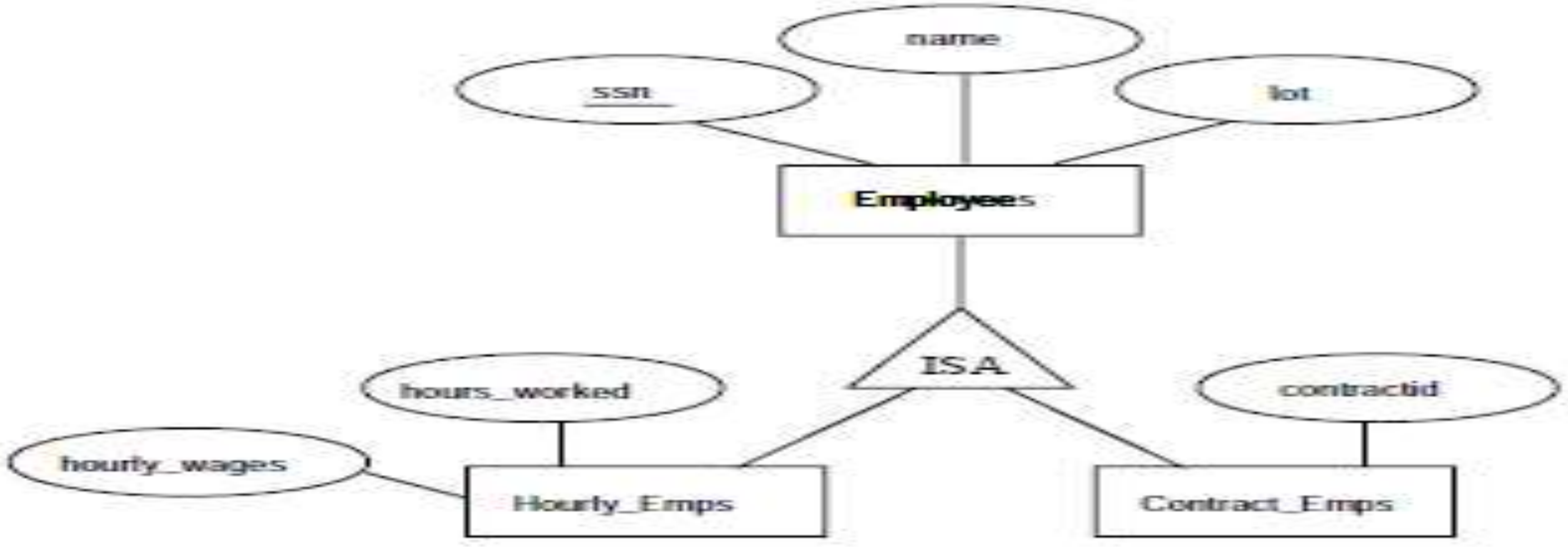


Figure 2.12 Class Hierarchy

- A class hierarchy can be viewed in one of two ways:

i) Specialization is the process of identifying subsets of an entity set (the superclass) that share some distinguishing characteristic. Typically, the superclass is defined first, the subclasses are defined next, and subclass specific attributes and relationship sets are then added.

ii) Hourly-Emps and ContracLEmps are generalized by Employees. As another example, two entity sets Motorboats and Cars may be generalized into an entity set MotorVehicles.

Generalization consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics.Typically, the subclasses are defined first, the superclass is defined next, and any relationship sets that involve the superclass are then defined.

Overlap constraints determine whether two subclasses are allowed to contain the same entity. For example, can Attishoo be both an Hourly_Emps entity and a ContracLEmps entity. Intuitively,no.So no overlapping exists.

- Can he be both a ContractEmps entity and a Senior-Emps entity? Intuitively, yes. We denote this by writing ContractEmps OVERLAPS Senior-Emps.In the absence of such a statement, we assume by default that entity sets are constrained to have no overlap.

Covering constraints determine whether the entities in the subclasses collectively include all entities in the superclass. A characteristic property of generalization hierarchies is that every instance of a superclass is an instance of a subclass.

AGGREGATION:

- A relationship set is an association between entity sets. To model a relationship among relationship set aggregation is used.

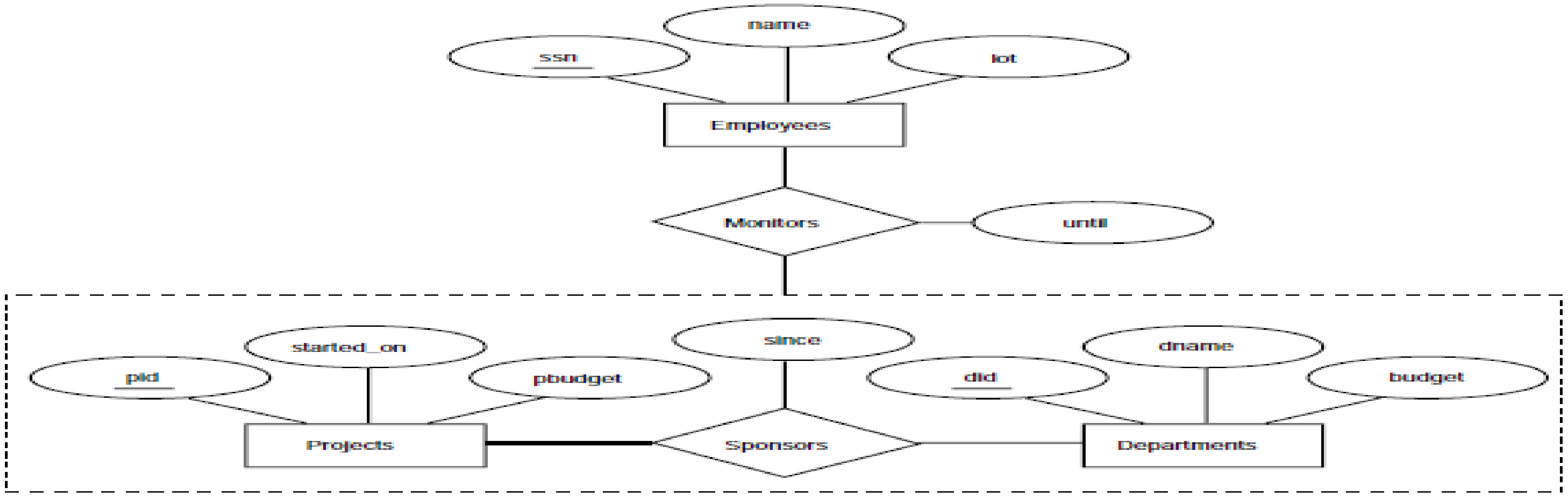


Figure 2.13 Aggregation

- Suppose that we have an entity set called Projects and that each Projects entity is sponsored by one or more departments. The Sponsors relationship set captures this information.
- A department that sponsors a project might assign employees to monitor the sponsorship. Intuitively, Monitors should be a relationship set that associates a Sponsors relationship (rather than a Projects or Departments entity) with an Employees entity. To define a relationship set such as Monitors, we introduce a new feature of the ER model, called aggregation. Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.
- This effectively allows us to treat Sponsors as an entity set for purposes of defining the Monitors relationship set

CONCEPTUAL DATABASE DESIGN WITH THE ER MODEL

1. Entity versus Attribute

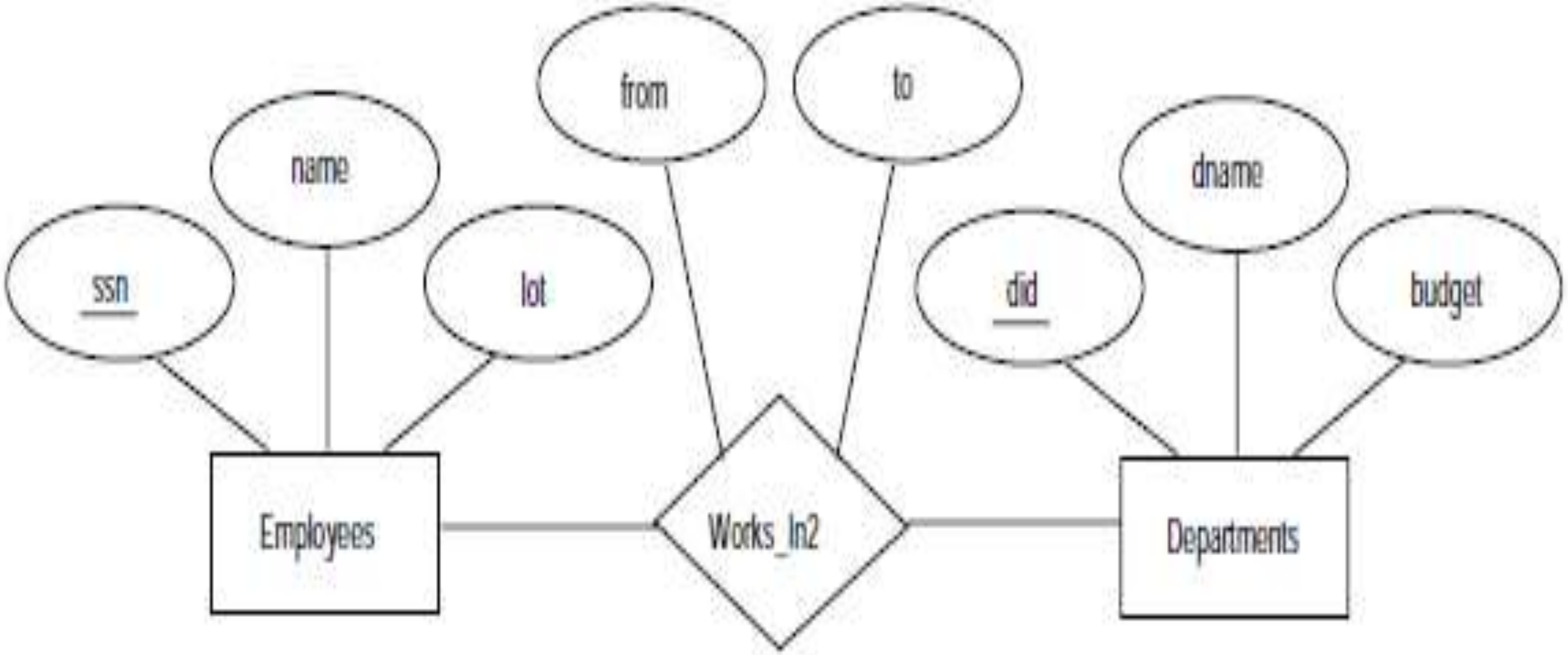
While identifying the attributes of an entity set, it is sometimes not clear whether a property should be modeled as an attribute or as an entity set.

For example, consider adding address information to the Employees entity set.

One option is to use an attribute address. This option is appropriate if we need to record only one address per employee.

An alternative is to create an entity set called Addresses ,which is complex alternative necessary in two situations:

- To **record more than one address for an employee**.
- To capture the structure of an address in our ER diagram. For example, we might break down an address into city, state, country, and Zip code, in addition to a string for street information.


from
to
name
dname
ssn
lot
did
budget
Employees
Works_In2
Departments

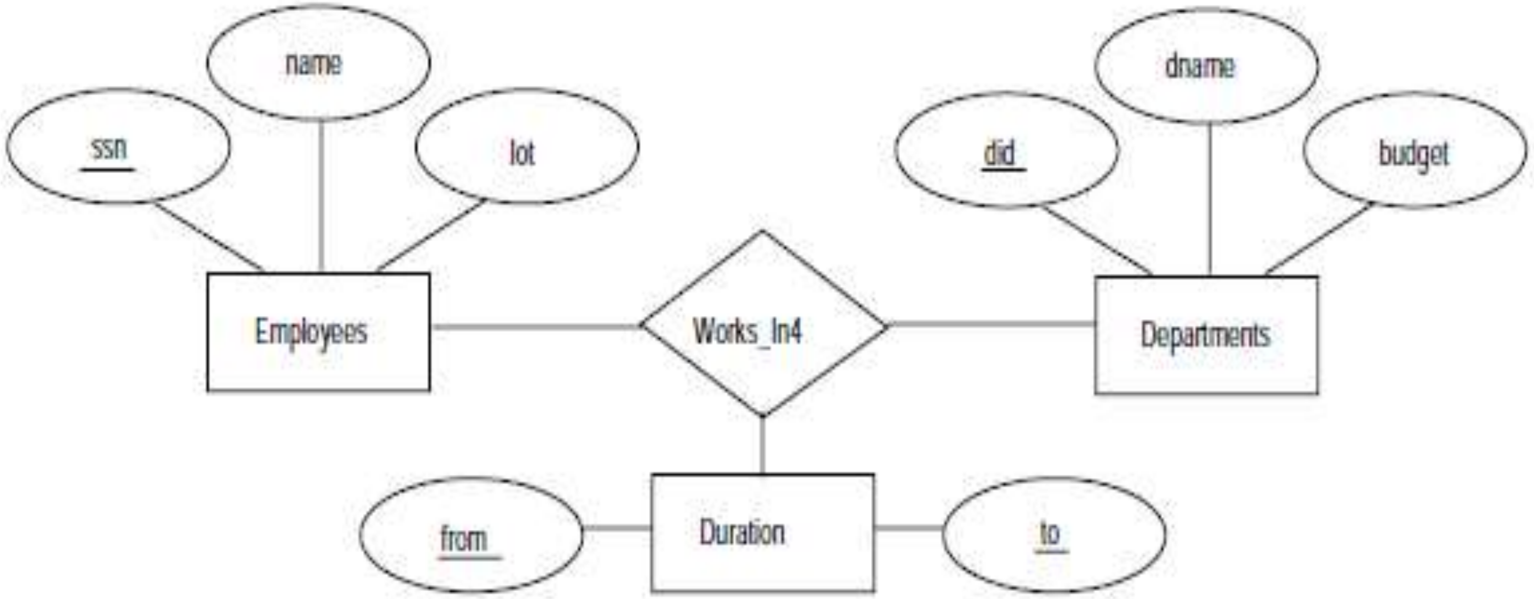


Figure 2.15 The Works_In4 Relationship Set

2. Entity versus Relationship

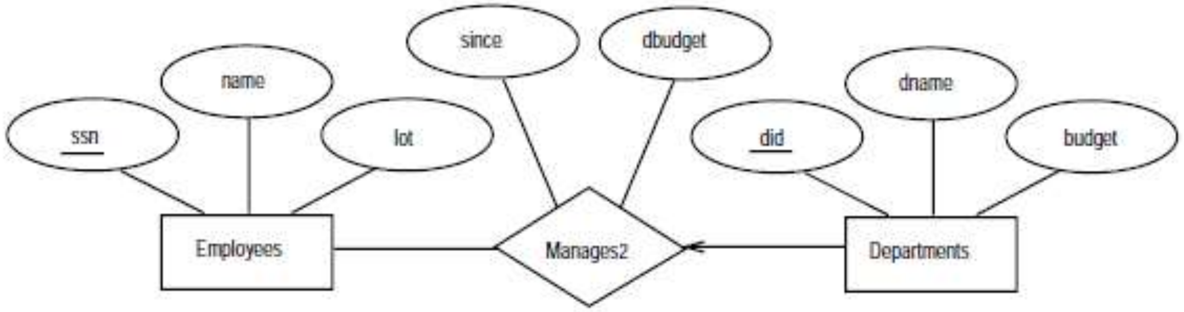
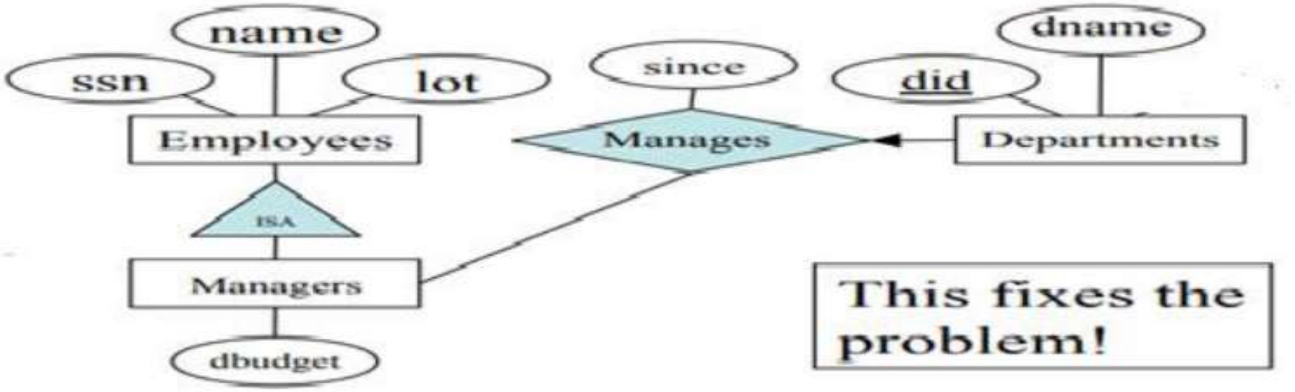


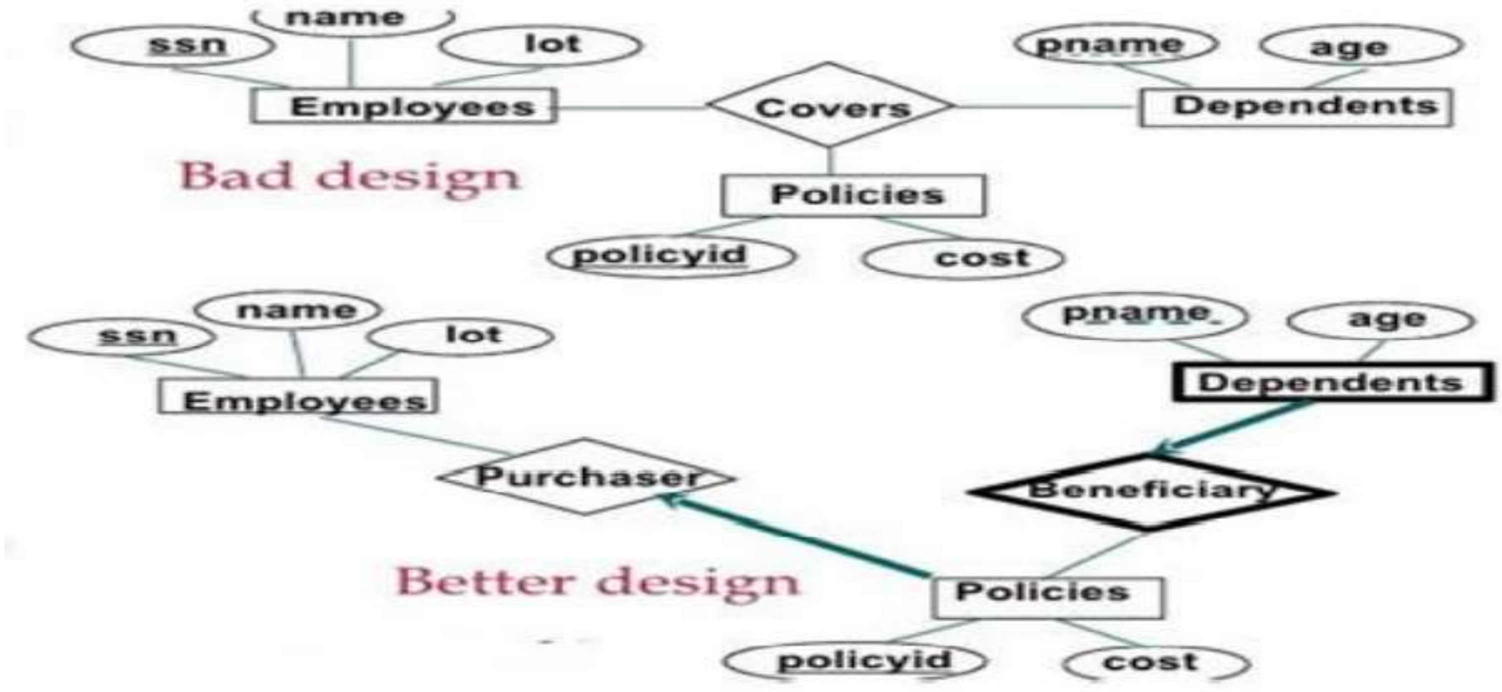
Figure 2.16 Entity versus Relationship



This fixes the problem!

- Given a department, we know the manager, as well as the manager's starting date and budget for that department.
- But if the budget is a sum that covers all departments managed by that employee, In this case, each Manages relationship that involves a given employee will have the same value in the dbudget field, leading to redundant storage of the same information.
- We can address this problems by introducing a new entity set called Managers (which can be placed below Employees in an ISA hierarchy, to show that every manager is also an employee).
- The attributes since and dbudget now describe a manager entity, as intended. Normalization is a technique used to eliminate redundancies from tables.

3.Binary vs Ternary



- It models a situation in which an employee can own several policies, each policy can be owned by several employees, and each dependent can be covered by several policies.

Suppose that we have the following additional requirements:

1.A policy cannot be owned jointly by two or more employees.

2.Every policy must be owned by some employee.

3. Dependents is a weak entity set, and each dependent entity is uniquely identified by taking pname in conjunction with the policyid of a policy entity (which, intuitively, covers the given dependent).

i)The first requirement suggests that we impose a key constraint on Policies with respect to Covers, but this constraint has the unintended side effect that a policy can cover only one dependent.

ii)The second requirement suggests that we impose a total participation constraint on Policies. This solution is acceptable if each policy covers at least one dependent.

iii)The third requirement forces us to introduce an identifying relationship that is binary. The best way to model this situation is to use two binary relationships, as shown in Figure(better design)

4. Aggregation versus Ternary Relationships

The choice between using aggregation or a ternary relationship is mainly determined by the existence of a relationship and also by constraints.

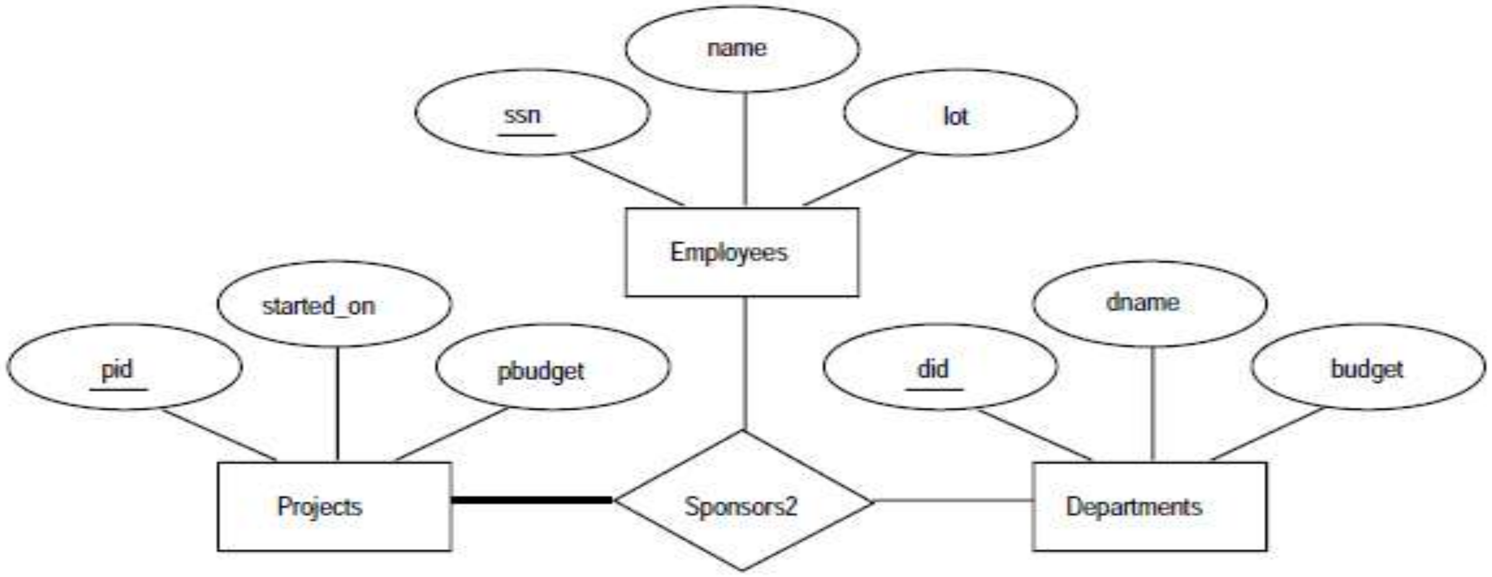


Figure 2.20 Using a Ternary Relationship instead of Aggregation

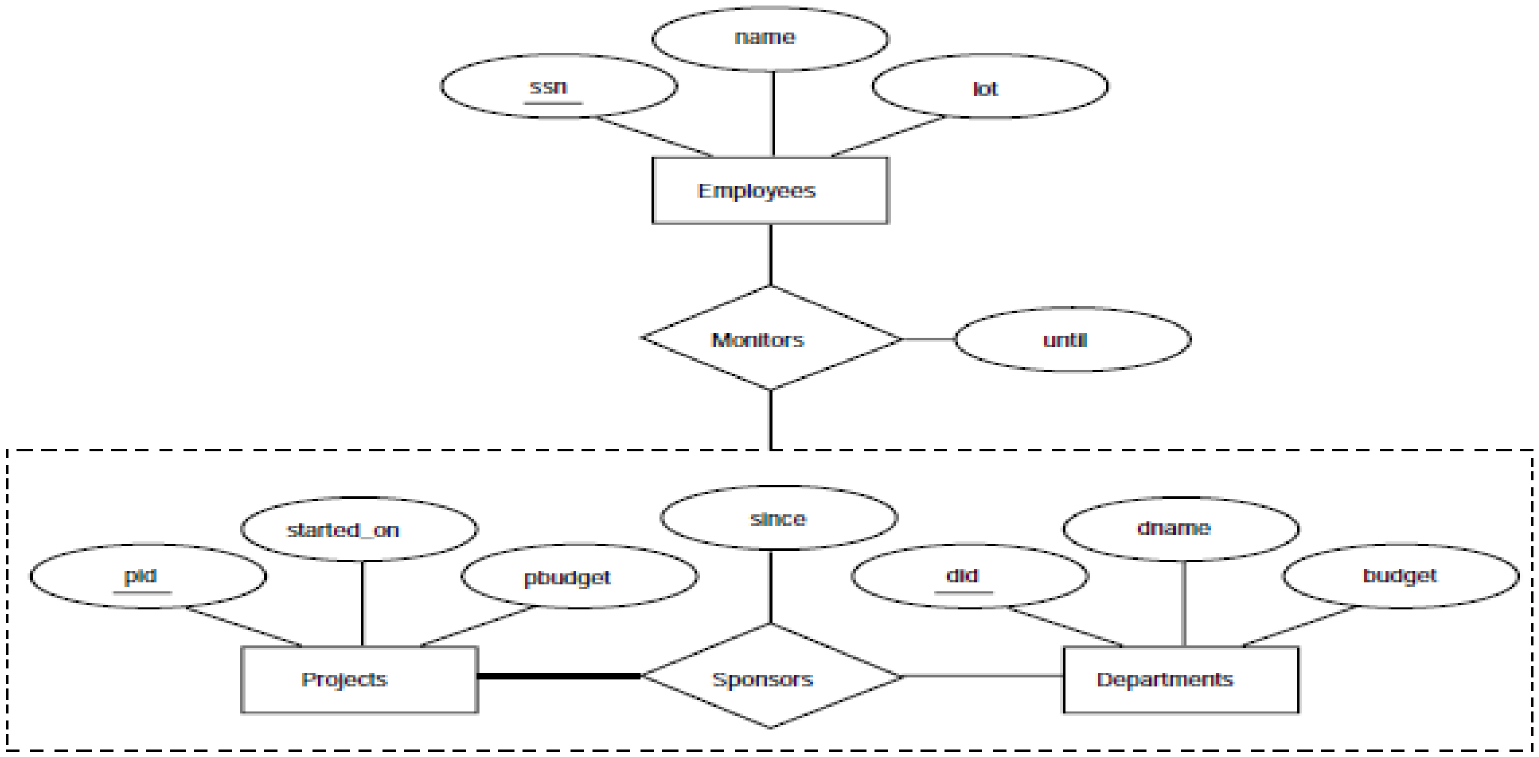


Figure 2.13 Aggregation

- If we don't need to record the until attribute of Monitors, then we might reasonably use a ternary relationship, say, Sponsors2
- Consider the **constraint that each sponsorship (of a project by a department) be monitored by at most one employee. This constraint cannot be expressed in terms of the Sponsors2 relationship set.**

Thank You