Chapter 1.1
Enhanced Entity Relationship Model and Relational Model

- Entity Relationship Model Revised,
- Subclasses, Super classes, and Inheritance: Specialization and Generalization.
- Union Types; Aggregation.
- Constraints on specialization and Generalization.
- Relational Model Revised.
- Converting ER and EER Model to Relational Model.
- SQL and Advanced Features.
- Concepts of File Structures, Hashing, and Indexing

Introduction

- A data models is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
- Every database and database management system are based on particular database model.
- Data model consists of set of rules and standards that define how the data is organized in a database.
- A data model also provides a set of operation for manipulation the data stored in the database.
- There are three different data models.
  1. Physical data model
     - The physical data model describes the way how data is stored in the computer by representing records format, record ordering and access path(indexing).

  2. Logical data model
     - These models are used to specify the overall logical structures of database.
     - Three most widely used logical data model are hierarchical data model, network data model and relational data model.

  3. Conceptual data model

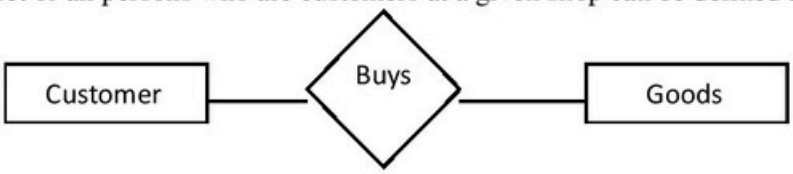
Conceptual model / Entity-Relationship (ER) Model

- These models are used for describing data and data relationship by using the concept such as entities, attributes, relationships etc. The most widely used conceptual model is entity relationship (E-R) model.
- The E-R model is a conceptual model which is based on the perception of the world consisting of collection of basic objects (entities) and relationship among these objects.
- To develop a database model based on E-R technique, entities and their relationship should be identified first.
- The E-R model is very useful in mapping the meaning and interactions of real word enterprise onto a conceptual schema.
- E-R diagram can be used to graphically represent the overall structure of database.

Components of E-R Model

1. **Entity and Entity set:**
   - An entity is an object or thing that has its existence in the real world and is distinguishable from other objects.
   - It includes all those things about which data is collected.
   - An entity may be person, place or thing which can be distinctly identified.
   - For example, each person in a university is an entity. An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.

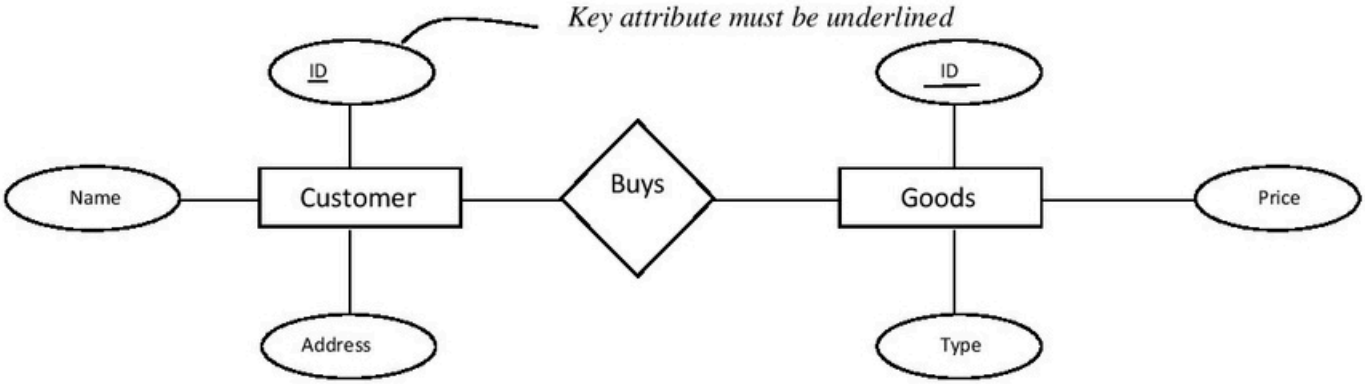
- For instance, a person may have a person id property whose value uniquely identifies that person.
- An entity set is a set of entities of same type that share same properties or attributes.
- Example: The set of all persons who are customers at a given shop can be defined as the entity set costumer.



- In the above figure, if we say that a customer buys goods, it means customers and goods are entity set.
- The notation of **Entity set in** E-R diagram is **Rectangle**.
- **Diamond** with a name inside is used to represent their **relationship.**

2. Attributes

- Attributes are units that are used to describe the characteristics or properties of entities set.
- Attributes are descriptive properties possessed by all member of entity set.
- The customer entity set might have attributes like ID, Name, Address etc. similarly goods entity set might have attributes like ID, type, price, cost etc.
- For each attributes, there is a set of permitted values called the **domain** of that attributes. The domain of attribute "Name" for "Customer entity set" might be a set of all text string of a certain length.
- The notation of attributes in E-R diagram is **elliptical** shape.



Attribute Types:

i. **Simple and Composite Attributes:** Simple attributes are those attributes which can't be divided into subparts i.e. other attributes. E.g. attribute ID. Composite attributes are those attributes which can be divided into subparts. E.g. an attribute "Name" can be considered as composite attribute consisting of First Name, Middle Name and Last Name.

ii. **Single-Valued and Multi-Valued Attributes:** The attributes which has a single value for a particular entity are called single-valued attributes. E.g. "Social Number" attribute because each person have only on social number. `The attributes which has a set of values for a particular entity are called multi-Valued attributes. The notation for Multi-valued Attributes in ERD is double elliptical shape. E.g. "Phone Number" attribute because each person may have zero, one or many phone numbers.

iii. **Derived Attributes:** The value for this type of attributes can be derived from the values of other related attributes. E.g. the attribute "age" is a derived attribute because we can calculate age from another attribute "DOB" and "Current Date". The value of derived attribute is not stored but is computed when required. The notation for Derived attribute in ERD is Dashed Elliptical shape.

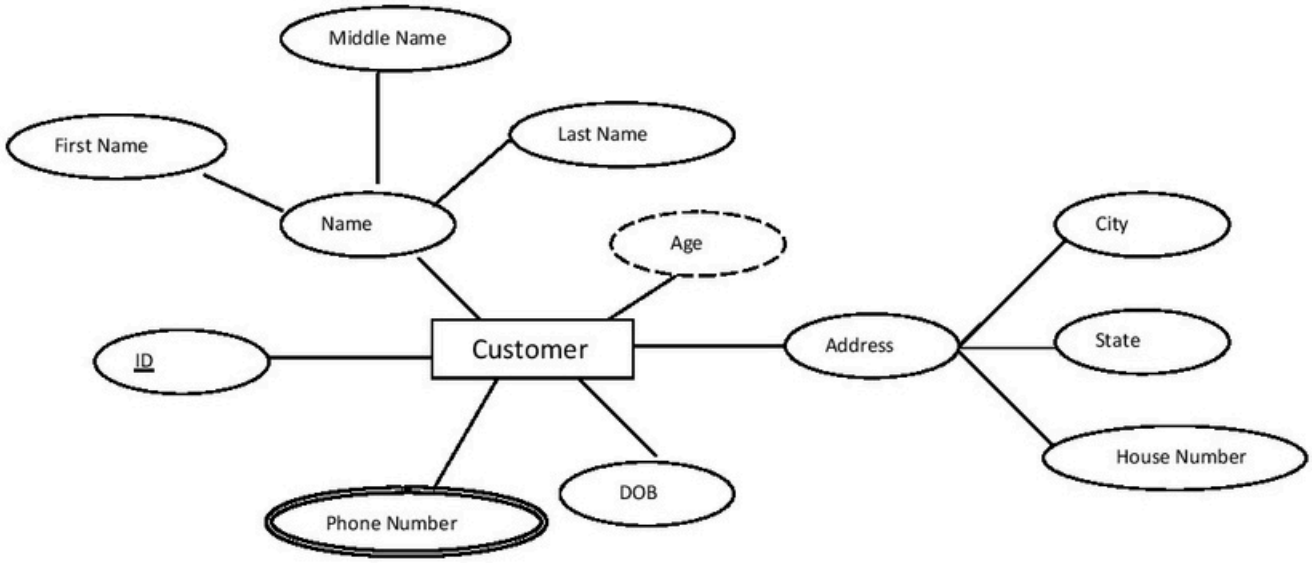
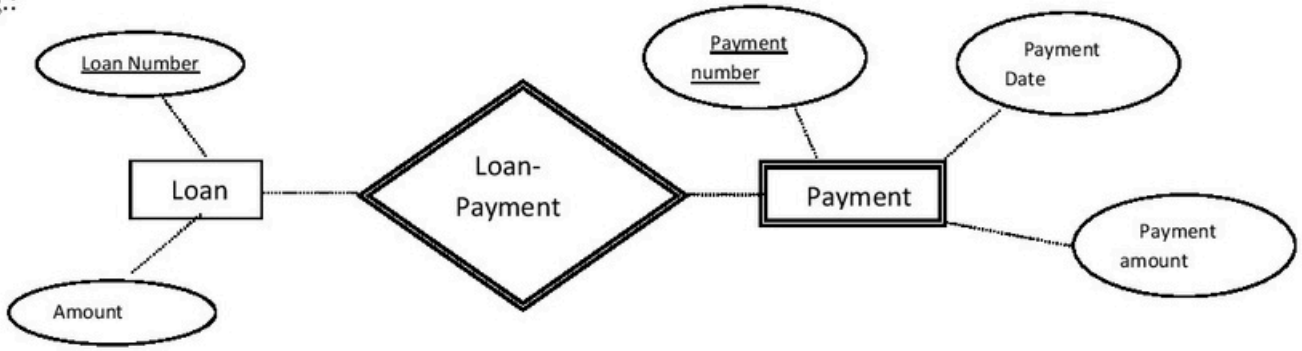


Fig: ERD showing composite, multi-valued and derived attributes

3. Weak-Strong Entity Set:

- An entity set that doesn't have enough keys to define a primary key is referred to as a weak entity set.
- The existence of a weak entity set depends on the existence of a strong entity set.
- The notation for weak entity (Dependent entity) is double lined rectangle.
- And strong/weak relationship is represented by Double Diamond.
- If an entity set has primary key, then it is a strong entity (independent entity) set.

E.g.:



- In the above E-R diagram, Loan is strong entity set while Payment is weak entity set.
- For Loan we can use loan number as primary key as it can distinguish each entity in Loan entity set.
- But in case of weak entity set, Payment, we can't only define payment number as primary key because it can't uniquely identify the entity set. For this reason, an attribute of weak entity set must be used in combination with primary key of strong entity set. So the primary key for payment is (loan number + payment number).

4. **Relationship Set**

- A relationship set is set of relationship of same type.
- Consider the two entity sets instructor and student in below figure.
- In ERD, relationship set is represented by a diamond with a name inside.
- We define the relationship set advisor to denote the association between two entities set instructors and students.
- Figure below depicts this association.

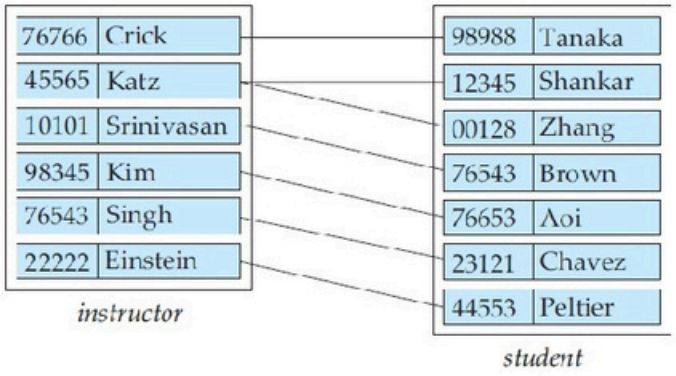


Figure 7.2 Relationship set *advisor*.

5. Relationship

- A relationship is an association that exists between entities and may be looked as a mapping between two or more entity set.
- Example: In the above figure, we can define a relationship advisor that associates instructor Katz with student Shankar.
- This relationship specifies that Katz is an advisor to student Shankar.
- If a relationship set has some attributes associates with it, then we enclose the attributes in a rectangle and link the rectangle with the dashed line to the diamond representing the relationship set.
- For example, in below figure, the date attribute attached to the relationship set advisor to specify the date on which an instructor became the advisor.

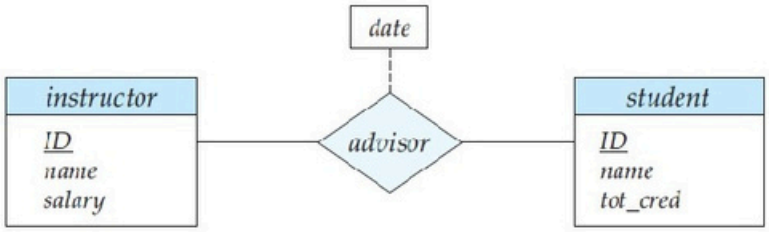


Figure 7.8 E-R diagram with an attribute attached to a relationship set.

- Relationship Instance is an association of entities, where the association includes exactly one entity from each participating entity type.

6. Degree of Relationship

- The degree of relationship is the number of entities associated in the relationship.
- Unary, Binary, and ternary relationship have degree 1, 2 and 3 respectively.

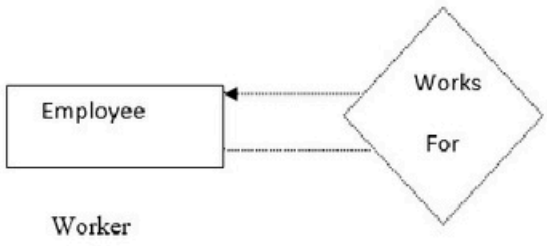


Fig 1: Unary/Recursive Relationship type

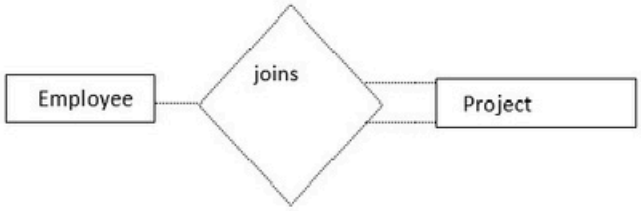


fig 2: Binary relationship type

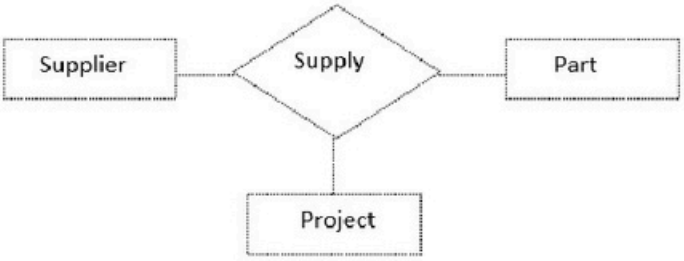


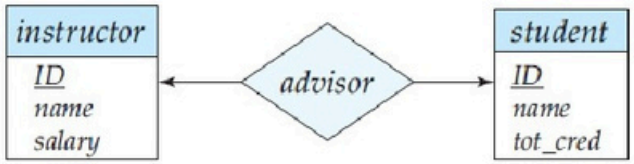
Fig 3: Ternary Relationship type

7. Constraints on ER Diagram

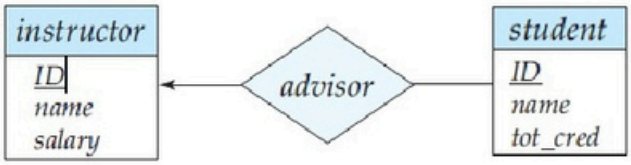
- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.
- Some of the constraints that can be enforced during ER modelling are mapping cardinality, participation and keys.

i. Mapping Cardinality

- It describes the maximum number of entities that a given entity can be associated by a relationship.
- The relationship set advisor between the instructor and student entity sets may be one-to-one, one-to-many, or many-to-many. To distinguish among these types, we draw either a directed line (→) or and undirected line (−) between the relationship set and the entity set as follows.



(a)



(b)

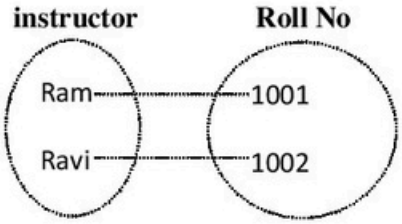
(c)

- Relationships. (a) One-to-one. (b) One-to-many. (c) Many-to-many.

The cardinality constraints for binary relationship are:

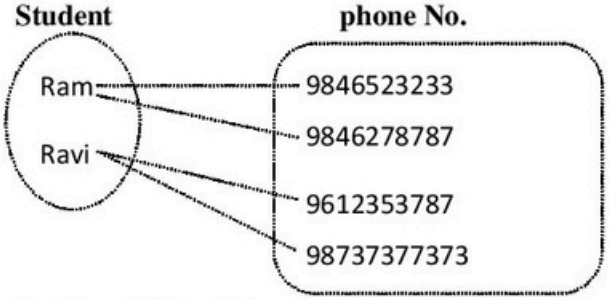
a. One to One (1-1):

- An entity in entity set A is associated with almost one entity in entity set B, and an entity in B is associated with almost one entity in A.
- E.g., driver and car



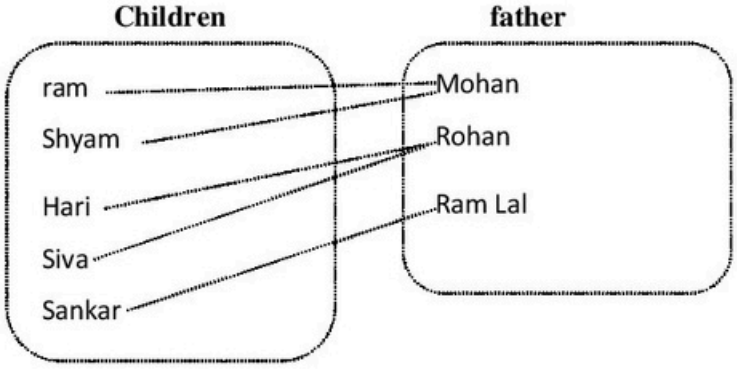
b. One to many(1-m)

- An entity in entity set A is associated with single or multiple entities in entity set B, and an entity in B is associated with at most one entity in A.
- E.g., Teacher and Students



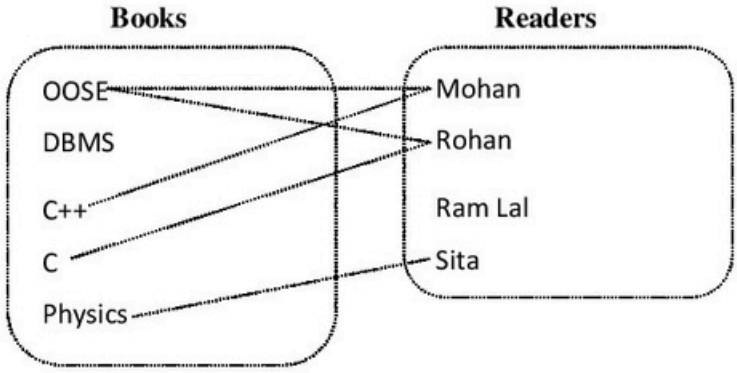
c. Many to One(m-1)

- An entity in entity set A is associated with at most one entity in entity set B, and an entity in B can be associated with single or multiple entities in A.
- E.g., Children and Father



d. Many to Many(m-m)

- An entity in entity set A is associated with single or multiple entities in entity set B, and an entity in B is associated with single or multiple entities in A.
- E.g., Books and Readers



- E-R diagrams also provide a way to indicate more complex constraints on the number of times each entity participates in relationships in a relationship set.
- A line may have an associated minimum and maximum cardinality which can be shown in the form of l..h, where l is the minimum and h is the maximum cardinality. A minimum value of 1 indicates that each entity in the entity set occurs at least one relationship in that relationship set. A maximum value of 1 indicates that the entity participates in at most one relationship. If * is given as maximum value, then it indicates no limit. For example

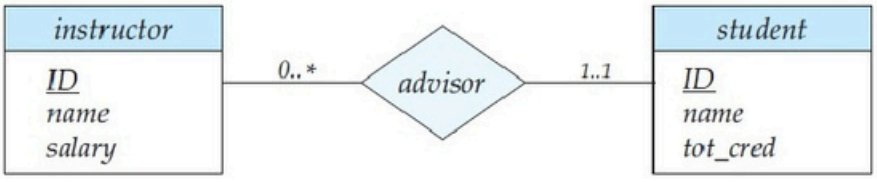


Figure 7.10 Cardinality limits on relationship sets.

- The line between advisor and student has a cardinality constraints of 1..1, this means, each student must have exactly one advisor. Hence the participation of student in advisor is **total.**
- The line between instructor and advisor has a cardinality constraints of 0..*, this mean, an instructor can have zero or any number of students. So, the relationship advisor is one to many from instructor to student.

ii. Participation

- The participation constraints specify whether existence of an entity depends on its being related to another entity via the relationship type.
- This constraint determines whether all or only some entity occurrence participates in a relationship.
- Participation may be.
  a. Total: Every entity in the entity set participates in at least one relationship in the relationship set.
  b. Partial: Some entity may not participate in any relationship in the relationship set.
- The notation for **total** participation in ER model is **double line.**
- The notation for **partial** participation in ER model is s**ingle line.**

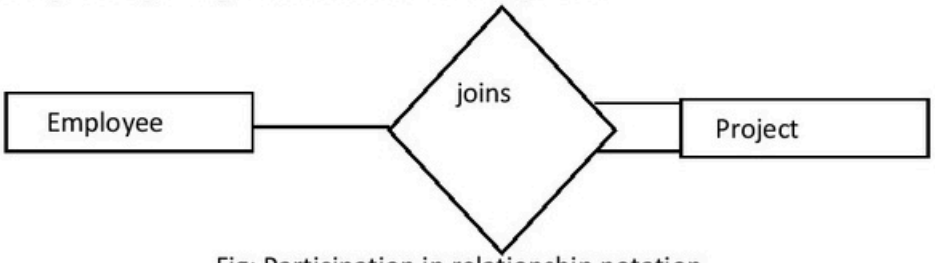


Fig: Participation in relationship notation

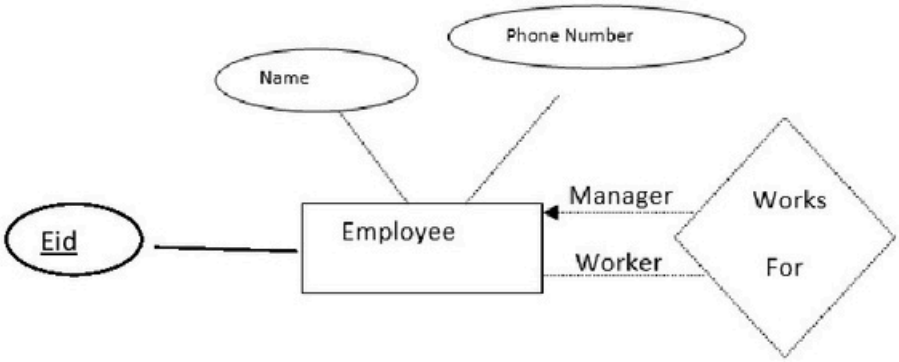
- In the above ERD, every project has at least one employee joined in it. So, Participation of entities in entity set "project "in "joins" relationship set is **total.**
- It also shows that, not every employee in the company joins in a project, so participation of entities in entity set "Employee" in "joins" relationship set is **partial.**

iii. Keys

- Keys are the constraints that uniquely distinguish each entity in within a given entity set.
- A key is an important constraint which is a group of one or more attributes that uniquely identify an entity in the entity set.
- The main integrity of information stored in database is controlled by keys.
- In ER diagram keys attributes are **underlined.**
- A key is a column value in a table that is used to uniquely identify a row of data in a table or establish a relationship with another table. There are six types of keys.
  1. **Super Key:** a set of one or more attributes which taken collectively allows to uniquely identify entity in entity set.
     E.g. for entity set "Student"=**{student_id, batch, faculty, sem, roll no, name, address}**
     Above entity set has many super keys. Some of them are {student_id, batch, faculty, sem, roll no, name, address}, {batch, faculty, sem, roll no}, {student_id}etc.
  2. **Candidate Key:** The smallest possible super key that uniquely identifies record in a table is called candidate key for primary key. **Candidate key is a minimal super key that doesn't have any proper subset**. Candidate keys are also Super key but all super key are not candidate key. Some of the candidate key for above entity set are {batch, sem, roll no}, {student_id} etc because these are the minimal super key.
  3. Primary Key: Primary key is a key which uniquely identify records in a table. The primary key field doesn't accept redundant data. Any of the candidate key can be used as primary key. Example: student_id
  4. Alternate Key: Any attribute that is a candidate for primary key but is not used as the primary key is called the alternate key. Among two candidate key if sudent_id is chosen as primary key then candidate key{batch, sem,roll} is now alternate key.
  5. Composite Key: When a key that uniquely identifies the records of the table is made up of more than one attributes, it is called composite key. Example {batch, sem, roll}
  6. Foreign Key: A foreign key is a linking pin between two tables. The key connects to another table when a relationship is being established. A foreign key is a copy of a primary key in another table.

8. Roles:

- The functions that an entity plays in a relationship is called role.
- Roles are indicated in E-R diagram by labeling the lines that connect diamonds to rectangles.
- The positions i.e., manager and worker are called the roles.
- In the below ERD, it specifies how "employee" entities interact via "works for" relationship set.
- Role labels are optional and are used to clarify the structure of relationship.



Advantage of ER model

- Easy and simple to understand with minimal training.
- It is possible to find connection from one node to all other nodes.
- It has explicit linkages of entities.

Disadvantages of ER model

- Limited relationship representation.
- Limited constraint representation.
- No representation of data manipulation.

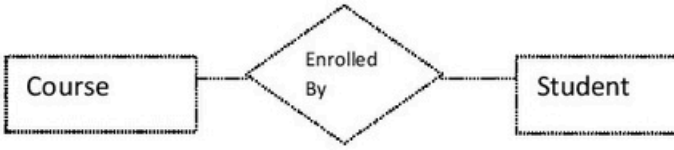
Example:

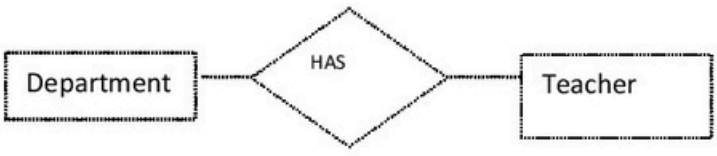
Draw ERD for your college database with as set of department, courses, teachers and students.

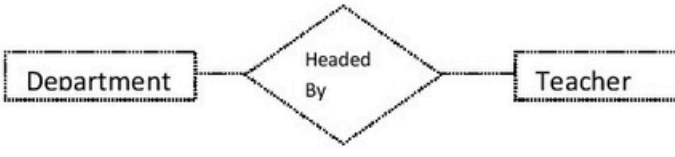
Step1: Identify the entities.

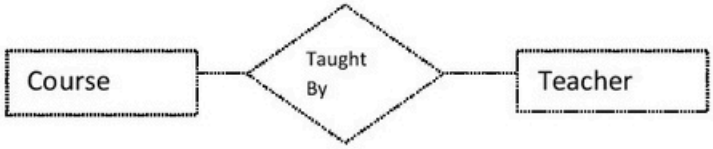
- Here there are four entities named department, courses, teacher, and student.

Step2: Find the relationship between these entities.









Step3: Identify the attributes for each entity.

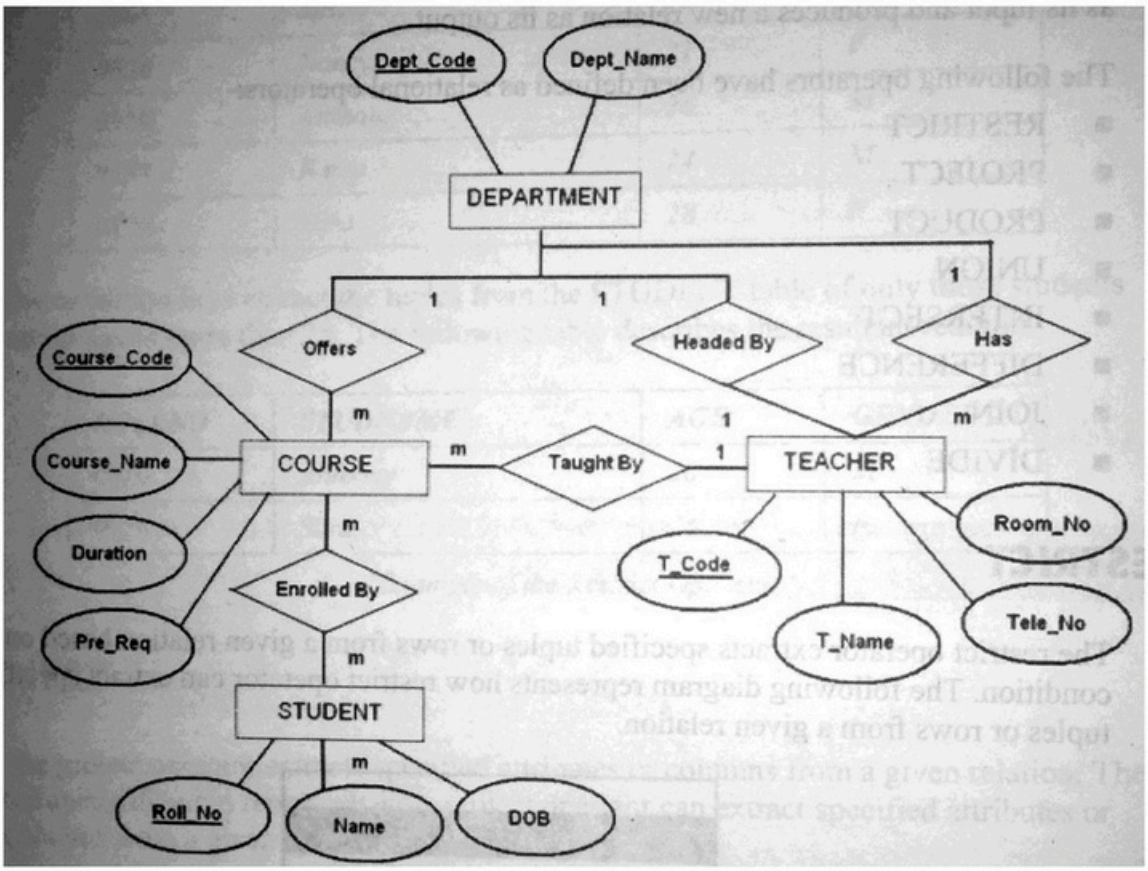
For Department:Department code(Dept_code), Department Name (Dept_Name)

For Course: Course_Code, Course_Name, Duration, Prerequisite (Pre-Req)

For Student: RollNo., Student Name(Name), Date of birth (DOB)

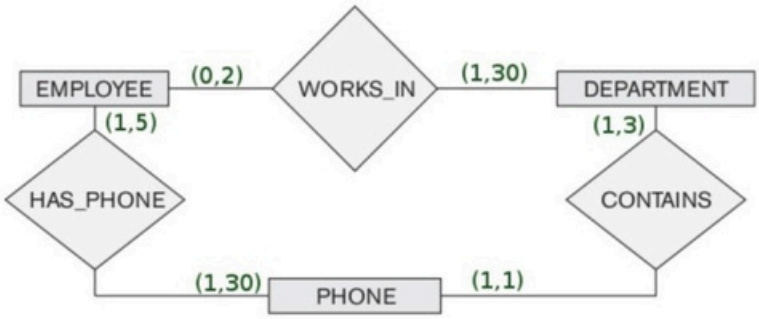
For Teacher: Teachers code (T_Code), Teacher Name (T_name), Telephone Number (Tele_No)

Step4: Now draw ERD with the above information and mapping cardinality.



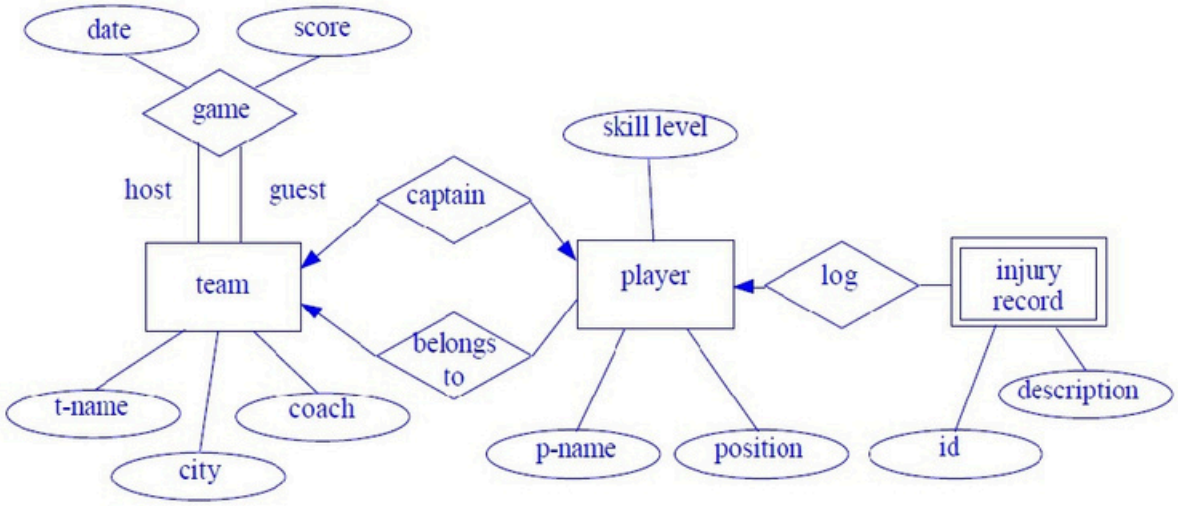
1. Given the following requirements.
   - i. An employee may work in up to two departments or may not be assigned to any department.
   - ii. Each department must have one and may have up to three phone numbers.
   - iii. Each department can have anywhere between 1 and 30 employees.
   - iv. Each phone is used by one, and only one, department.
   - v. Each phone is assigned to at least one and may be assigned to up to 30 employees.
   - vi. Each employee is assigned at least one, but no more than 5 phones.

Construct a clean and concise ER diagram for the database. Clearly indicate the cardinality mappings.



2. Suppose you are given the following requirements for a simple database for the National Cricket League (NCL):
    - the NCL has many teams,
    - each team has a name, a city, a coach, a captain, and a set of players,
    - each player belongs to only one team,
    - each player has a name, a position (such as *left bowler* or *batsman*), a skill level, and a set of injury records,
    - a team captain is also a player,
    - a game is played between two teams (referred to as host_team and guest_team) and has a date (such as *May 11th, 1999*) and a score (such as *4 or 2*).

   Construct a clean and concise ER diagram for the NHL database using the Chen notation as in your textbook. List your assumptions and clearly indicate the cardinality mappings as well as any role indicators in your ER diagram.



Extender ER features: Specialization , Generalization, and Aggregation

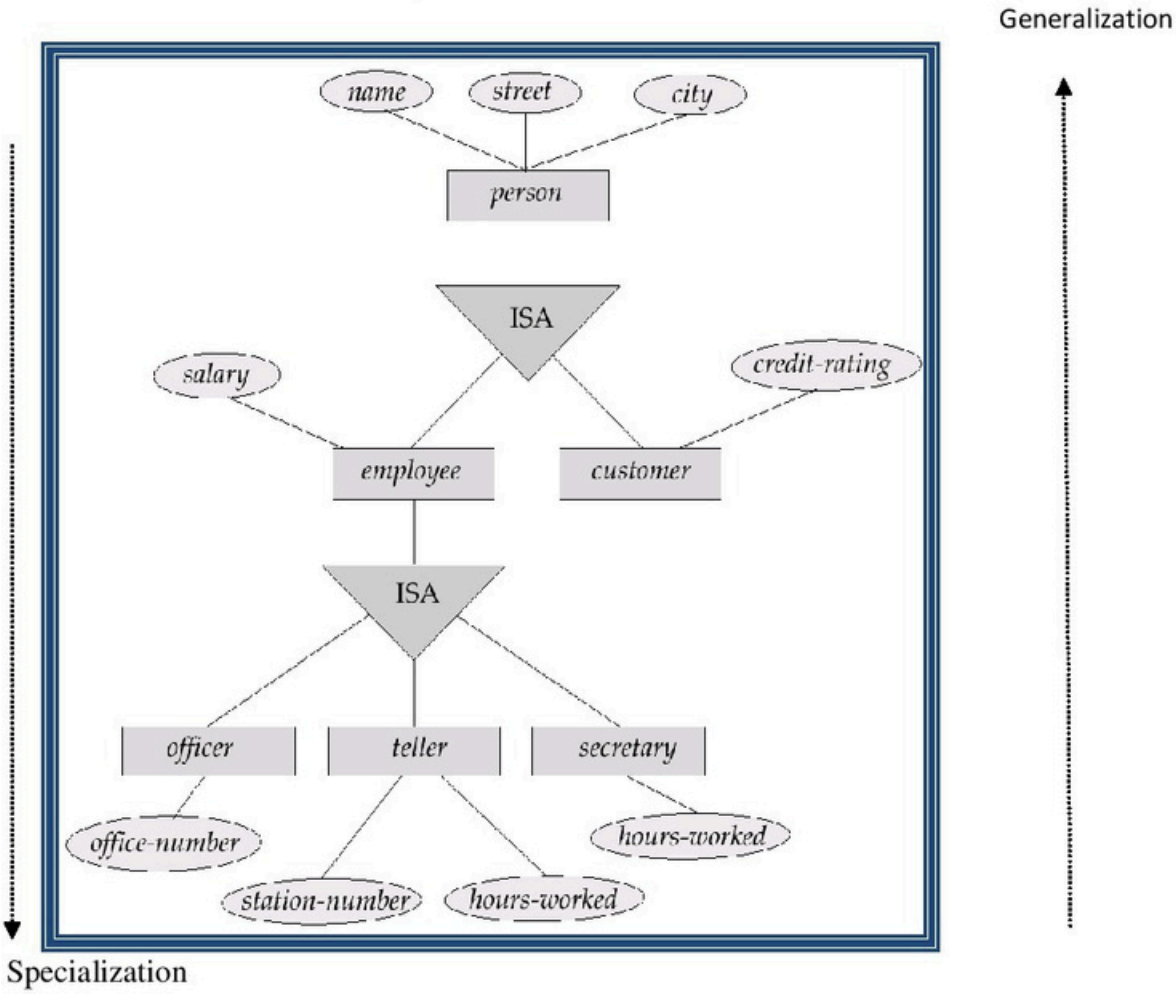
- Although the basic E-R concepts can model most database features, some aspects of a database may be more suitably expressed by certain extensions to the basic E-R model.
- For this, we discuss the extended E-R features of specialization, generalization, higher- and lower-level entity sets, attribute inheritance, and aggregation.

Superclass, Subclass, and Inheritance: Specialization and Generalization

- The process of subgrouping a higher-level entity set into multiple lower-level entity set is called **specialization**. The specialization relationship may also be referred to as superclass-subclass relationship.
- It is a top-down design process.
- These sub groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to higher-level entity set.
- The entity set that has number of sub grouping is called **super class** while the subgroups are called **sub classes** of super classes.

- A lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked. This is called **attribute inheritance.**
- In ERD specialization is represented by triangle symbol labeled by ISA.
- The ISA relationship is also referred to as a superclass –subclass relationship.

- **Generalization** is the process of taking union of two or more lower-level entity set to produce a higher-level entity set.
- It is a bottom-up design process because we combine all the entities sets that share the same features into higher level entity.
- Specialization and Generalization are simple inversion of each other.
- They are represented in ERD in the same way.



Another example:

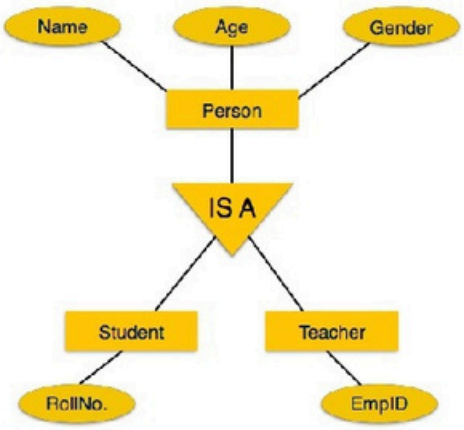


Fig: Specialization and Generalization

Union Type: Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships. Aggregation is a technique to express relationship among relationship.
- Aggregation is an **abstraction through which relationships are treated as higher level entities**. Aggregation is the concept that represents the relationship between a whole object and its components units.
- Aggregations shows a has-a or is-part-of relationship between entities where one represent the whole and other the part. Simply, aggregation represents a relationship where one entity (the whole) includes or is composed of other entities (the parts)
- Consider the following ERD (Fig I) with ternary relationship.

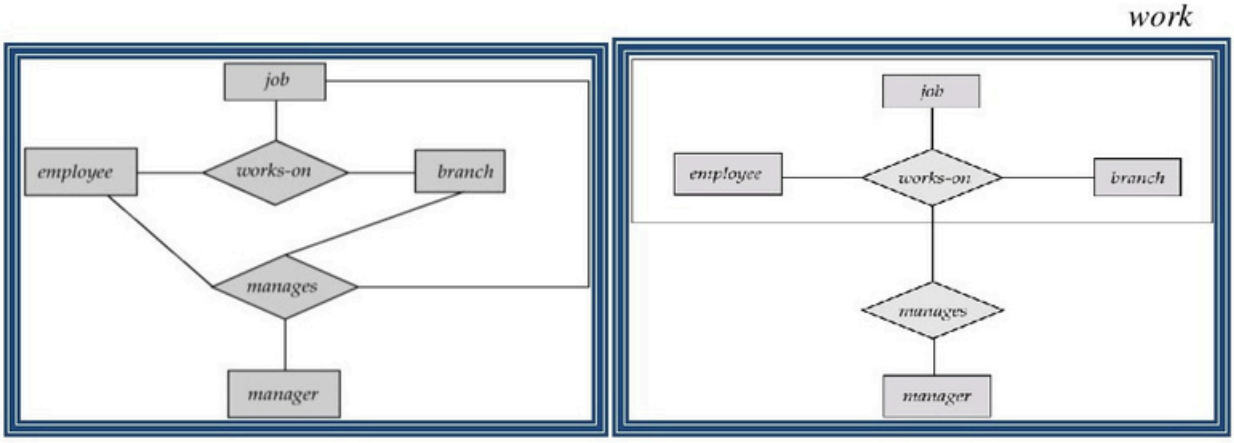


Fig I Fig II

- In the above ERD (Fig I) the relationship set "manages" and "works-on" represent overlapping information.
- Every "manages" relationship corresponds to "works-on" relationship but some "works-on" relationship may not correspond to any "manages" relationship so we can't discard "works-on" relationship.
- This problem of redundancy can be eliminated via aggregation i.e. treat relationship as higher level entity.
- So, using aggregation, we treat the relationship set "works-on" and the entity set "employee" "branch" and job as higher-level entity set called "work". i.e. higher level entity set work is composed of other entities like employee, job, & branch and their relationship.
- Fig II shows ERD with aggregation and without redundancy. Here, an employee works on particular job at particular branch. While an employee, branch job combination may have an associated manager.

Another examples:

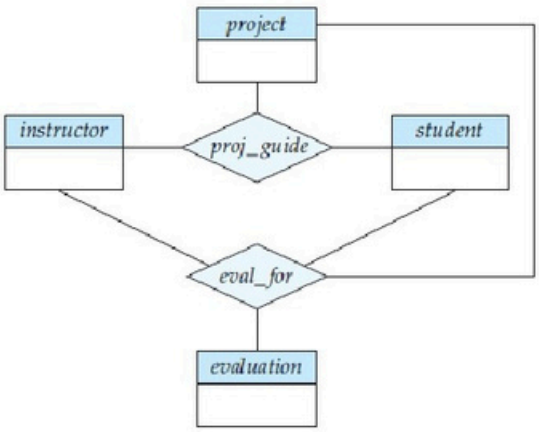


Figure 7.22 E-R diagram with redundant relationships.

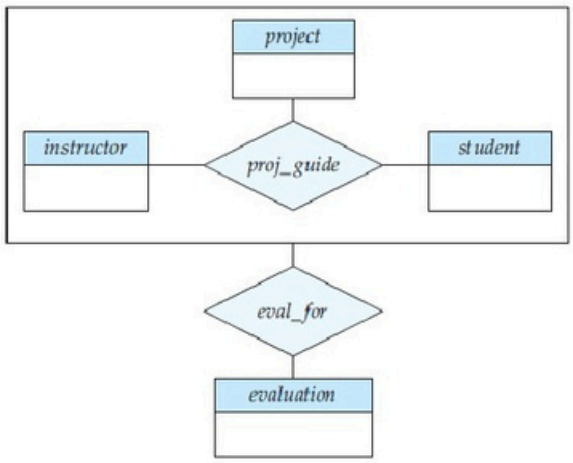


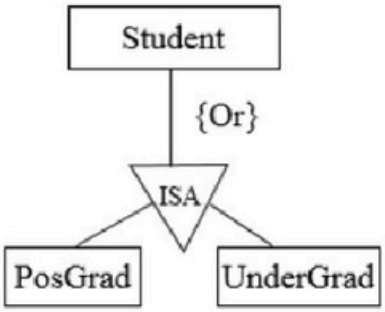
Figure 7.23 E-R diagram with aggregation.

Constraints on specialization and Generalization.

- Specialization and generalization are two important concepts in entity-relationship modeling, used to represent hierarchical relationships between entities.
- Some of the constraints on Specialization and Generalization includes.

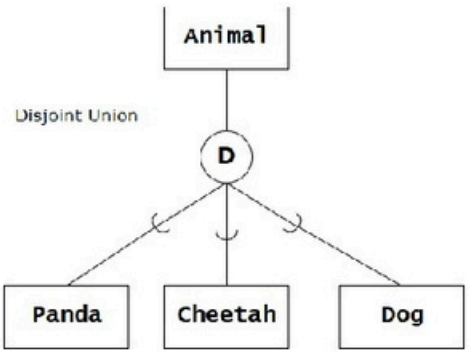
1. Disjointness Constraints

➢ The disjoint constraint only applies when a superclass has more than one subclass. If the subclasses are disjoint, then an entity occurrence can be a member of only one of the subclasses, e.g., postgrads or undergrads i.e., you cannot be both. To represent a disjoint superclass/subclass relationship, "Or" is used.



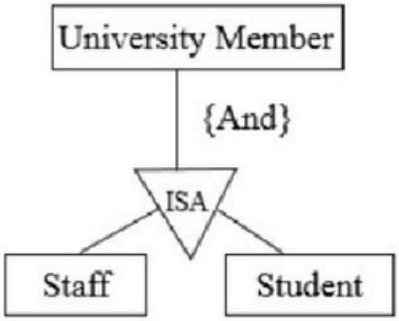
Another Example

Animal can be Panda, or Cheetah or Dog but not both of all. In this example, disjoint constraints is specified with another equivalent representation with D in circle.



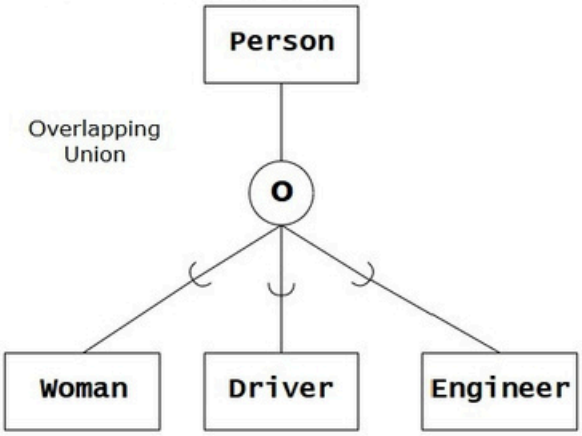
2. Overlapping constraints

- This applies when an entity occurrence may be a member of more than one subclass, e.g. student and staff i.e. "some people are both. "And" is used to represent the overlapping specialization/generalization relationship in the ER diagram.



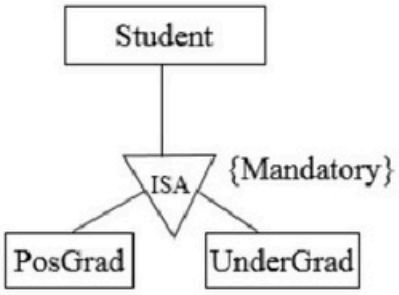
Another Example:

Some Persons are both women or Driver or Engineer. In this example, overlapping constraints is specified with another equivalent representation with O letter in circle.



3. Completeness/Totalness Constraint

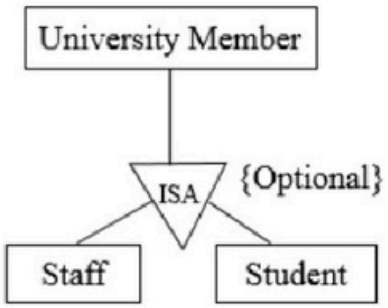
- The completeness constraint on a generalization or specialization specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following:
  a. **Total generalization or specialization**: Each higher-level entity must belong to a lower-level entity set. e.g., A student must be postgrad or undergrad. To represent total in the specialization/generalization relationship, the keyword ˜Mandatory" is used.



Note:

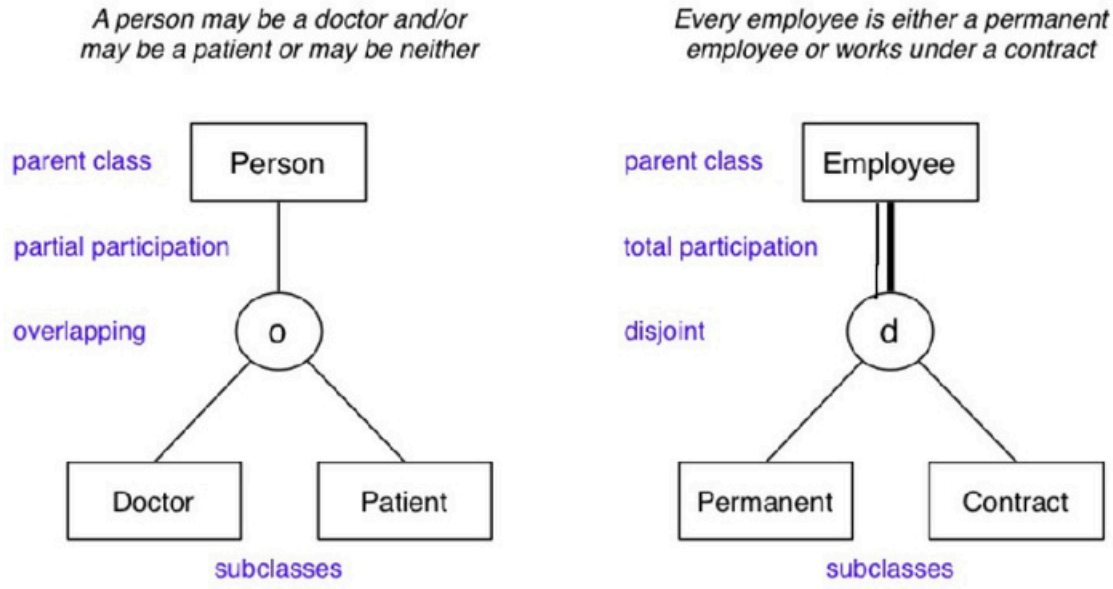
Also we can use,

  b. **Partial generalization or specialization:** Some higher-level entities may not belong to any lower-level entity set. e.g., Some people at Tribhuvan University are neither student nor staff. The keyword "Optional" is used to represent a partial specialization/generalization relationship.



Another Example:

The below figure shows another technique to represent total and partial participation in EER Diagram. For total participation we use double line and for partial participation we use single line as shown in figure below.

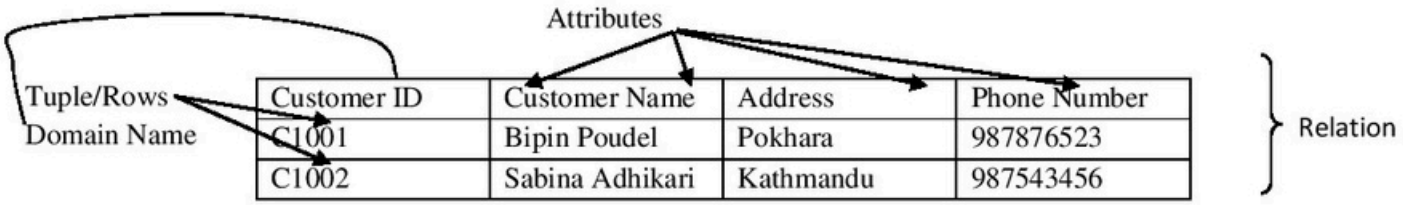


Relational Model

- In the relational model, data is organized into tables, which are also known as relations. Each table consists of rows and columns, with each row representing a unique record or entity, and each column representing a specific attribute or characteristic of that record. The columns are defined by their data types, such as integers, strings, dates, or Booleans.
- The relational model has established as the primary data model for commercial data processing application.
- Users of the database can create tables, insert new records in tables or modify the existing tables records using **query language**.
- A relational database is a database structured in relational model.
- A RDBMS is a collection of programs that can be used to create, maintain, modify, and manipulate the relational database. Some of the examples of RDBMS are Oracle, MS-SQL Server, MS-Access etc.

Structure of Relational Model

- A relational database consists of collections of tables, each of which is assigned a unique name.
- A row in a table represents a relationship among a set of values.
- A table is a collection of such relationships.



Customer ID	Customer Name	Address	Phone Number
C1001	Bipin Poudel	Pokhara	987876523
C1002	Sabina Adhikari	Kathmandu	987543456

The above table consists of below listed components according to relational model.

1. Domain: A set of permitted values for an attribute.
2. Tuple: A record/row in a table is called tuple. Every relation is made up of many tuples.
3. Relation: The term relation in this model refers to the two-dimensional table of data.
4. Relation Schema: A relation schema consists of a list of attributes and their corresponding domains.
5. Relation instance: The relation instance is the collection of data at any instance of time within a relation.
6. The degree (arity) of a relation: The degree of any relation is the number of attributes in the relation schema.
7. The Cardinality of a relation: The total number of tuples in the relation is called the cardinality.

Converting ER Model to Relational Model

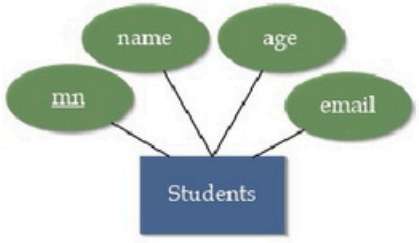
- We can represent a database that conforms to an E-R database schema by a collection of relation schemas.
- For each entity set and for each relationship set in the database design, there is a unique relation schema to which we assign the name of the corresponding entity set or relationship set.

Representation of Strong Entity Sets with Simple Attributes

- Let E be a strong entity set with only simple descriptive attributes a1, a2, . . . , a_n.
- We represent this entity by a schema called E with n distinct attributes.
- Each tuple in a relation on this schema corresponds to one entity of the entity set E.

Example:

- Consider the entity set students of the E-R diagram in Figure below. This entity set has four attributes: mn, name, age and email.
- We represent this entity set by a schema called students with four attributes:



Students (mn, name, age email)

Representation of Strong Entity Sets with Complex Attributes

- When the entity set consists of complex attributes such as composite, derived, and multivalued, things are a bit more complex.
- For the composite attribute *name*, the schema generated for customer contains the attributes first name, middle name, and last name; there is no separate attribute or schema for name.
- Multivalued attributes generally map directly into attributes for the appropriate relation schemas.
- Derived attributes are not represented explicitly in relational model.

Example:

- Consider the entity set customer of the E-R diagram in figure below.
- Here we see composite attributes "name" and address, multivalued attribute "phone number" and derived attribute "Age"

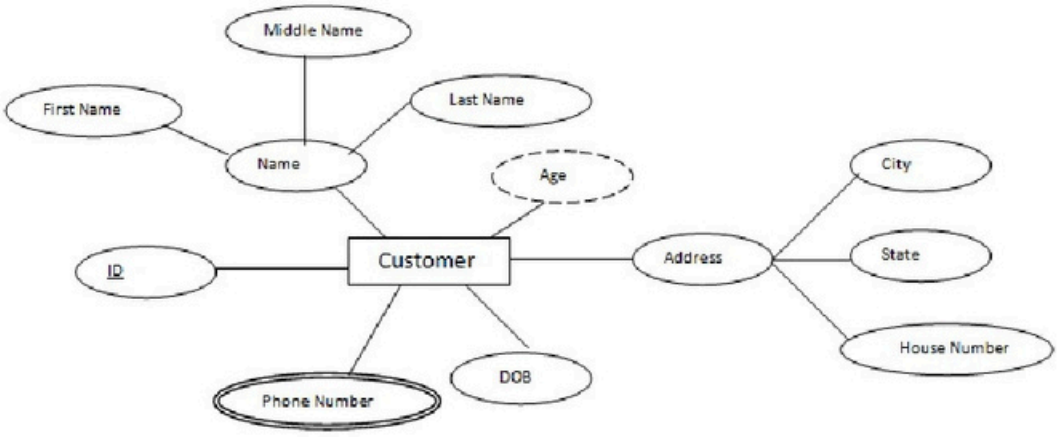


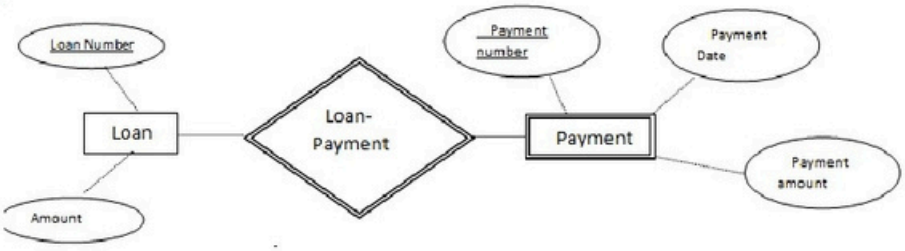
Fig: ERD showing composite, multi-valued and derived attributes

- We represent this entity set by a two schema as below.

Customer (ID, First Name, Middle Name, Last Name, City, State, House Number, DOB)
Customer_Phone(ID, Phone Number)

Representation of Weak Entity Sets

- Weak entity types are converted into a table of their own, with the primary key of the strong entity acting as a foreign key in the table.
- This foreign key along with the key of the weak entity form the composite primary key of this table.
- The entities in the weak entity set must be automatically deleted (cascade deleted) for which there are no owners in strong entity set.



We represent this entity set by a two schema as below.

Loan(Loan Number, Amount)
Payment(Loan Number, Payment Number, Payment Date, Payment Amount)

Here foreign key "Loan Number" references primary key " Loan number" in Loan

Representation of Relationship Set

i. Binary one to one Relationship
   - The primary key of either of the participants can become a foreign key in the other

We represent these entities set by a two schema as below.

Way1:

Instructor(instructor.ID, name,salary)
Student(student.ID,name, tot_cred)
Advisor(Instructor.ID,Student.ID)

Here in relational schema Advisor, foreign key instructo.ID references primary key ID in instructor, foreign key student.ID references primary key ID in student and combination of these two acts as primary key in schema Advisor. Further in relational schema Advisor, Instructo.ID and Student.ID both should have Unique constraints.

Way2:

Student(Student.ID, name, tot_cred, instructor.ID)
Instructor(Instructor.ID, name, salary)

Here in relational schema Student, instructor.ID_is foreign key that references primary key instructor.ID in instructor. Also instructor.id must have unique constraints.

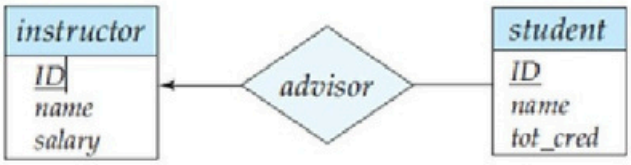
Way3:

Instructor(instructor.ID,name,salary, student.ID)
Student(student.ID, name, tot_cred)

Here in relational schema Instructor, student.ID is a foreign key that references primary key student.ID in student. Also, student.id must have unique constraints.

ii. Binary one to many or many to one relationship

- The primary key of the relation on the "one" side of the relationship becomes a foreign key in the relation on the "many" side



- We represent these entity set by a two schema as below.

  Way1:

  Instructor(instructor.ID, name,salary)
  Student(student.ID,name, tot_cred)
  Advisor(Instructor.ID,Student.ID)

  Here in relational schema Advisor, foreign key instructo.ID references primary key ID in instructor, foreign key student.ID references primary key ID in student and combination of these two acts as primary key in schema Advisor. Further in relational schema Advisor, Student.ID should have Unique constraints.

  Way2:

Instructor(ID, name, salary)

Student(Student.ID, name, tot_cred, instructor.ID)

Here -in relational schema Student, the foreign key instructo.ID references primary key ID in instructor

iii. Binary Many to many relationships

- A new table is created to represent the relationship set.
- Contains two foreign keys - one from each of the participants in the relationship
- The primary key of the new table is the combination of the two foreign keys

Example:

- We represent these entity set by a three schema as below.

Instructor(ID, Name, salary)

Student(ID, name,tot_cred)

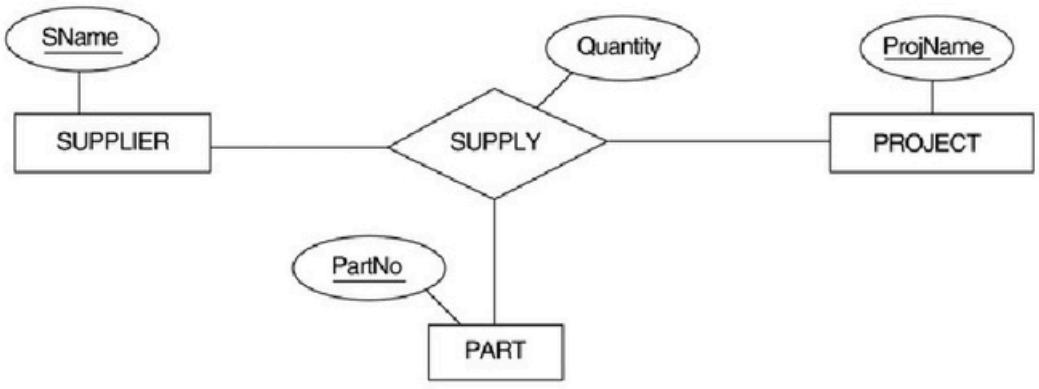
Advisor(instructor.ID , student.ID)

Here in relational schema Advisor, foreign key instructo.ID references primary key ID in instructor, foreign key student.ID references primry key ID in student and combination of these two acts as primary key in schema Advisor.

Mapping of N-ary relationship type

- The general algorithm for mapping N ary relationship is
  - i. For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
  - ii. Include as foreign key attributes in S, the primary keys of the relations that represent the participating entity types.
  - iii. Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S
  - iv. The primary key of the new table is the combination of all n foreign keys

- Consider the following ERD with ternary relationship i.e. 3-ary relationship



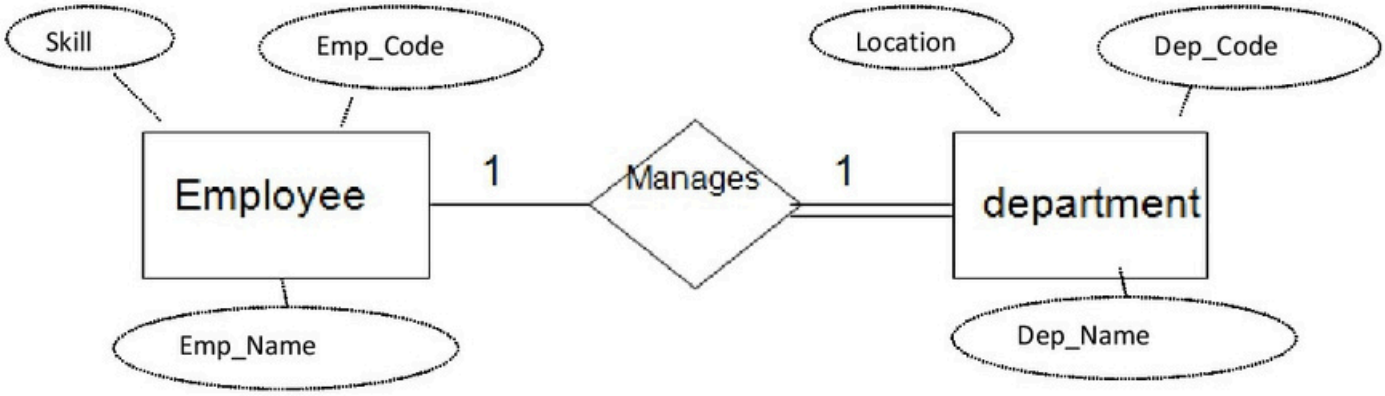
We represent these entities set in Relational model by four relational schema as below.

Supplier (sname,……)
Project(projname, ……)
Part(PartNo, ……)
Supply(sname, projname, PartNo, quantity)

Here, sname, projname and partno in supply are foreign keys that references primary key sname, projname and partno in Supplier, project and part respectively. Also sname+projname+quantity is the primary key for new relation supply.

Representation of total and partial participation

Consider the following ERD



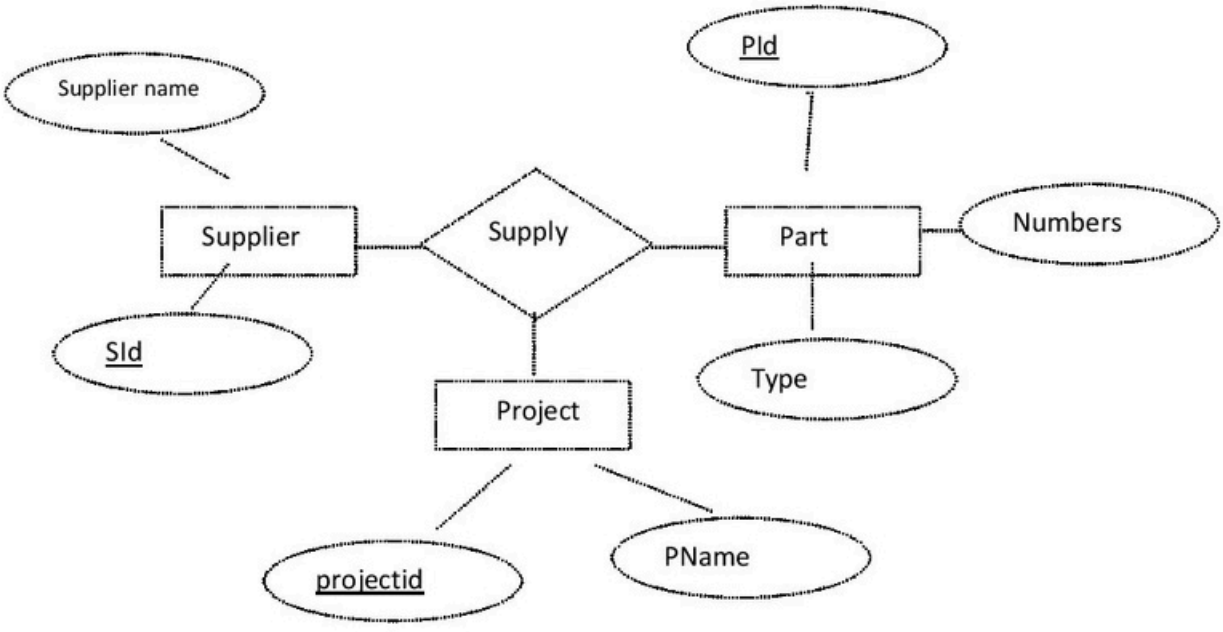
- Here the participation of the Department in Manages relationship is total. This means, every department value in department row must appear in rows of Employee. Hence, the primary key of the partial participant will become the foreign key of the total participant. So the equivalent relational schema are given below

Employee(Emp_code, Skill, Emp_Name)
Department(Depcode, Dep_Name, Location, Emp_code)

Here the foreign key emp_code in department references primary key Emp_code in Employee. Also Emp_code in Department must have **Not Null** constraints.

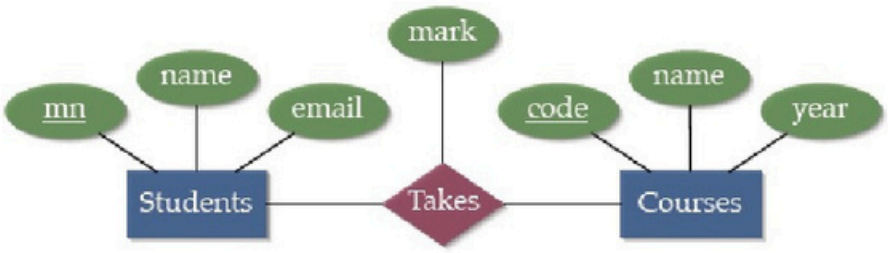
Convert the following E-R diagram to equivalent Relational schema.

1.



Supplier(sid, Supplier name)
Project(projectid, Pname)
Part(Pid,Type,Number)
Supply(sid, projected, Pid)

2.



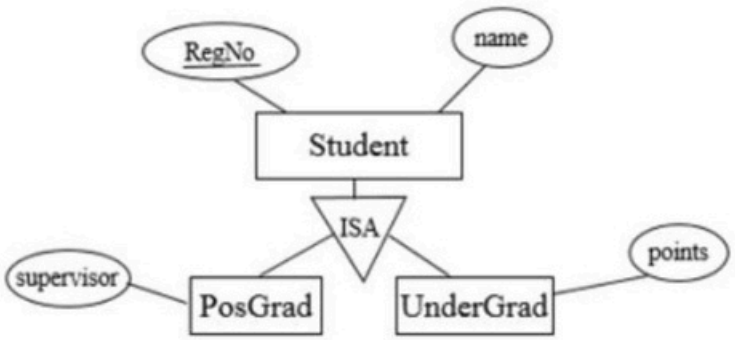
Student(mn , name , email)
Courses(code , name , year)
Takes(mn,code,mark)

Note:In Take schema, foreign key mn references primary key mn in student while foreign key code references primary key code in courses

Converting EER Model to Relational Model

Mapping specialization/generalization to relational tables

- Create table for higher-level entity set, and treat specialized entity subsets like weak entity sets
- Specialization/generalization relationship can be mapped to relational tables in three methods.
- To demonstrate the methods, we will take the student, postgraduate and undergraduate relationship. A student in the university has a registration number and a name. Only postgraduate students have supervisors. Undergraduates accumulate points through their coursework.



Method 1

All the entities in the relationship are mapped to individual tables.

Student (*Regno*, name)
PosGrad (*Regno*, supervisor)
UnderGrad (*Regno*, points)

- This method is preferred when inheritance with completeness(partial) constraints i.e. Some students may not be any of PostGrad or UnderGrad, is to be enforced.
- Similarly, we can enforce disjoint constraints, e.g. A student can be PosGrad or UnderGrad but not both. For this, we have to create trigger on insert operations that verify both schema PosGrad and UnderGrad Regno value should not contain duplicate value.

- However, in absence of this trigger, the subclasses PosGrad and UnderGrad will observe overlapping constraints i.e. a student can be both PosGrad and UnderGrad, which is not normally desirable.

Method 2

Only subclasses are mapped to tables. The attributes in the superclass are duplicated in all subclasses.

PosGrad (*Regno*, name, supervisor)

UnderGrad (*Regno*, name, points)

This method is most preferred when inheritance is complete(total), e.g. every student is either PosGrad or UnderGrad. Similarly, we can enforce disjoint constraints as in method 1 if necessary.

Method 3

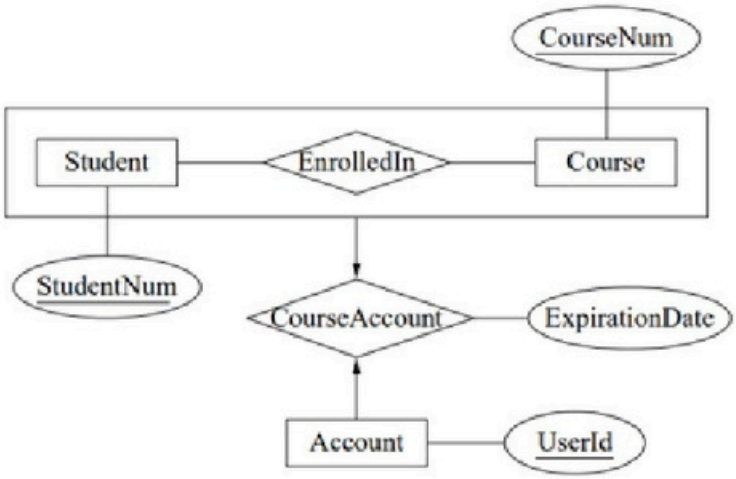
Only the superclass is mapped to a table. The attributes in the subclasses are taken to the superclass.

Student (*Regno*, name, supervisor, points)

This method will introduce null values. When we insert an undergraduate record in the table, the supervisor column value will be null. In the same way, when we insert a postgraduate record in the table, the points value will be null.

Mapping aggregation to relational tables

- In this case, there is no distinction between entity sets and relationship sets in relations model.
- So an EER diagram containing aggregation expresses relationship between a relationship set and an entity set. In relational model, separate relation is created for this relationship and the relation contains the primary key of associated entity set and the relationship set and its own attributes.



The relational schema for the above EER diagram are

Student(StudentNum)
Course(CourseNum)
Account(UserID)
EnrolledIn(StudentNum,CourseNum)
CourseAccount(UserId,StudentNum,CourseNum,ExpirationDate)

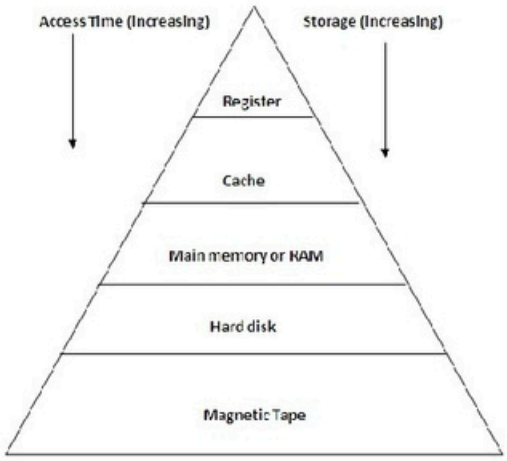
Draw an EER model for library management system having generalization and specialization hierarchies. The EER model should have disjoint and overlapping constraints with at least one of the entities having total participation. Use your own assumptions for other concept. Now convert so designed EER model to relational model.

File and File System

- A collection of data or information that has a name, called the *filename*.
- Almost all information stored in a computer must be in a file.
- There are many different types of files: *data files,* text files ,program *files,* directory *files,* and so on..
- The file consists of records and records may consist of several fields.
- The basic file operations that can be performed on file are.
  1. Creation of file
     Before creating a file, we have to collect data, process the data to give it a record like structure. Then we choose a name for the file and open the file on secondary storage device and store the collected data on it.
  2. Reading a file
     After creating a file, it is needed at any time that we have to read the file. We may have to read a particular record or the whole file.
  3. Update of a file
     Updating a file means that we have to perform some changes on the content of file.
  4. Insertion in file
     It is required to insert a record or set of records at some specific location.
  5. Delete
     When some record is not required in the file, then we delete that record to free up the memory.

Overview of storage Device

A typical hierarchy is illustrated below.



As we go down the hierarchy, the following occurs.

1. Decreasing cost per bit.
2. Increasing capacity.
3. Increasing access time.
4. Decreasing frequency of access of the memory by the processor.

Thus, smaller, more expensive and faster memories are supplemented by larger, cheaper and slower memories.

Computer memory can be classified in the below given hierarchy:

1) Internal register: Internal register in a CPU is used for holding variables and temporary results. Internal registers have a very small storage; however, they can be accessed instantly. Accessing data from the internal register is the fastest way to access memory.

2) Cache: Cache is used by the CPU for memory which is being accessed over and over again. Instead of pulling it every time from the main memory, it is put in cache for fast access. It is also a smaller memory, however, larger than the internal register.

Cache is further classified to L1, L2 and L3:

a) L1 cache: It is accessed without any delay.
b) L2 cache: It takes more clock cycles to access than L1 cache.
c) L3 cache: It takes more clock cycles to access than L2 cache.

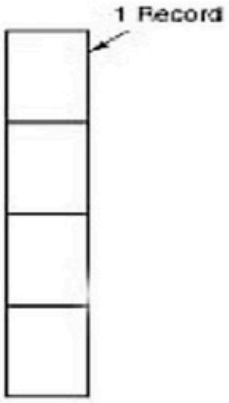
3) Main memory or RAM (Random Access Memory): It is a type of the computer memory and is a hardware component. It can be increased provided the operating system can handle it. It is accessed slowly as compared to cache. They are volatile i.e. content of memory are usually loss in case of power failure or crash.

4) Hard disk: A hard disk is a hardware component in a computer. Data is kept permanently in this memory. Memory from hard disk is not directly accessed by the CPU, hence it is slower. As compared with RAM, hard disk is cheaper per bit. Typically, the entire database is stored on disk. Data must be moved from disk to main memory in order for data to be processed.

5) Magnetic tape: Magnetic tape memory is usually used for backing up large data. When the system needs to access a tape, it is first mounted to access the data. When the data is accessed, it is then unmounted. The memory access time is slower in magnetic tape and it usually takes few minutes to access a tape.

File Organization: Organization of Records into Block

- A database is mapped into a number of different files that are maintained by the underlying operating system.
- These files reside permanently on disks. A file is organized logically as a sequence of records. These records are mapped onto disk blocks.
- A block is the unit for storage allocation and data transfer i.e. database files are organized into blocks.
- A database is stored as a collection of files.
- A file is organized logically as a sequence of records.
- A record is a sequence of fields.



- Blocks are of a fixed size determined by the operating system, but Record sizes may vary. In relational database, tuples of distinct relations may be of different sizes.
- **There are two approaches of organization of records into block.**
  1. **Fixed length records**
     - Here each record contains the same number of bytes or length.
     - Consider a file of deposit records of the form:

```
type deposit = record
        bname : char(20);
        account# : integer;
        cname : char(20);
        balance : real;
    end
```

- If we assume that each character occupies one byte, an integer occupies 4 bytes, and a real 8 bytes, our deposit record is 52 bytes long for each record.
- The simplest approach is to use the first 52 bytes for the first record, the next 52 bytes for the second, and so on.

Advantage

- Since the computer knows where each record starts, it is very fast to access and also easy to implement.
- Each record can be accessed as (N-1) * size of record.

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

Fig: Fixed length record (Instructor Record)

- However, there are two problems with this approach.
  - Unless the block size happens to be a multiple of 52 (which is unlikely), some records will cross block boundaries. It would then require two blocks of access to read or write such a record. To avoid the first problem, we allocate only as many records to a block as would fit entirely in the block (this number can be computed easily by dividing the block size by the record size and discarding the fractional part).
  - It is difficult to delete a record from this structure. Space occupied by the records to be deleted must be filled with some other records of the file. When a record is deleted, we could move all successive records up one, which may require moving a lot of records as follows.

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

Fig: Record state after deleting record 3.

2. Variable length records.

- A variable length record file is one in which records can be of different length.
- **Variable**-length records arise in a database in several ways:
  - Storage of multiple record types in a file

- Record types allowing variable field size.
- Record types allowing repeating fields such as arrays

- **One example with a variable-length record:**

```
type instructor = record
                    ID varchar (5);
                    name varchar(20);
                    dept_name varchar (20);
                    salary numeric (8,2);
                  end
```

- So, the record length will be of some variable length because it consists of several variable length fields like id, name, and dept-name.
- The space available for the records must be managed carefully.
- To manage free space, there is a pointer that indicates the start of the free space area.

Advantage:

1. Need less storage space.

Disadvantage:

1. The computer doesn't know where each record starts so processing the record is slower than the previous one.

Variable length record Representation

- The representation of a record with variable-length attributes typically has two parts.
  1. An initial part with fixed length attributes,
  2. Followed by data for variable length attributes.
- Fixed-length attributes, such as numeric values, dates, or fixed length character strings are allocated as many bytes as required to store their value.
- Variable-length attributes, such as varchar types, are represented in the initial part of the record by a pair (offset, length), where offset denotes where the data for that attribute begins within the record, and length is the length in bytes of the variable-sized attribute. The values for these attributes are stored consecutively, after the initial fixed-length part of the record.
- Thus, the initial part of the record stores a fixed size of information about each attribute, whether it is fixed-length or variable-length.

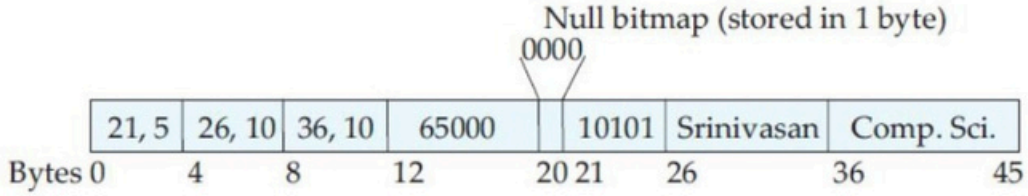


Fig: Representation of Variable length record

- The figure shows an instructor record, whose first three attributes ID, name, and dept name are variable-length strings, and whose fourth attribute salary is a fixed-sized number.
- We assume that the offset and length values are stored in two bytes each, for a total of 4 bytes per attribute.
- The salary attribute is assumed to be stored in 8 bytes, and each string takes as many bytes as it has characters.
- The figure also illustrates the use of a null bitmap, which indicates which attributes of the record have a null value. In this record, if the salary were null, the fourth bit of the bitmap would be set to 1, and the salary value stored in bytes 12 through 19 would be ignored. Since the record has four attributes, the null bitmap for this record fits in 1 byte, although more bytes may be required with more attributes. This representation is

particularly useful for certain applications where records have a large number of fields, most of which are null.

File Organization

- A file is a sequence of records stored in binary format.
- A disk drive is formatted into several blocks that can store records.
- File records are mapped onto those disk blocks.
- File Organization defines how file records are mapped onto disk blocks.
- We have four types of File Organization to organize file records.
  1. Heap (Pile) file organization
  2. Sequential file organization
  3. Indexed Sequential file organization.
  4. Hash file organization

1. Heap file organization

- In a heap or pile file, records are collected in the order they arrive.
- The blocks used in heap are linked by pointers.
- Any records can be placed anywhere in the file where there is space for the records.
- There is no ordering/ sorting of the records.
- When a new record is to be inserted it is placed in the last block if there is space.
- If the last block cannot accommodate those records, a new block is allocated and the records to be inserted is placed.
- The time required to locate a record in a heap is time consuming if the file is spread over more than few blocks.
- The heap file organization is generally used for small files.

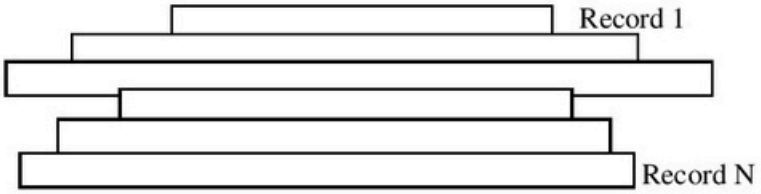


Fig: A heap file Structure

Advantage:

- Very good method of file organization for bulk insertion
- It is suitable for very small files as the fetching of records is faster in them.

Disadvantage

- This method is inefficient for larger databases as it takes time to search/modify/delete the record.
- Proper memory management is required to boost performance. Otherwise, there would be lots of unused memory blocks lying and memory size will simply be growing.

2. Sequential file organization

- A sequential file is designed for efficient processing of records in sorted order based on **some search key**.
- In sequential file organization records are placed sequentially onto the storage media.
- A search key is an attribute or set of attributes, but it need not to be a primary key.
- In addition, the records in the file are usually ordered according to the vales of key attributes.
- To permit fast retrieval of records in search key order, we chain together records by pointers.
- The pointer in each record points to the next record in search key order.
- In the figure below, all the records are stored in search key order using branch name as search key.

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

Advantage:

- The design is very simple compared to other file organization.
- Searching and deletion operation is faster as compared to heap file organization.
- This method is good in case of report generation or statistical calculations.
- These files can be stored in magnetic tapes which are comparatively cheap.

Disadvantage

- This method always involves the effort for sorting the record. Each time any insert/update/ delete transaction is performed, the file must be sorted.

3. Indexing

- An index is any data structure that take as input a property of records, typically the values of one or more fields and finds the records with that property quickly.
- The field in which the index is based is called the search key.
- An index is a collection of entries where each index corresponds to a data records.
- An index takes as input a search key value and returns the address of the records hold that values.

Search Key	Pointer

Fig: Structure of Index

- The search key value stored in the index are sorted. There are two basic kind of indices.
  - i. Ordered Indices
  - ii. Hash indices

i. Ordered indices

- Ordered indices are based on a sorted ordering of values.
- There are two types of ordered indices.
  a. Primary index
  b. Secondary index

`` **Primary index**

- We assume all files are ordered sequentially on some search key, such file is also called as **index sequential file.**
- A primary index is associated with **dense and sparse index.**

Dense Index

- An index record appears for every search key value in a file.
- This record contains search key value and a pointer to the actual record.
- It requires more memory space for index.

- Records are accessed fast.

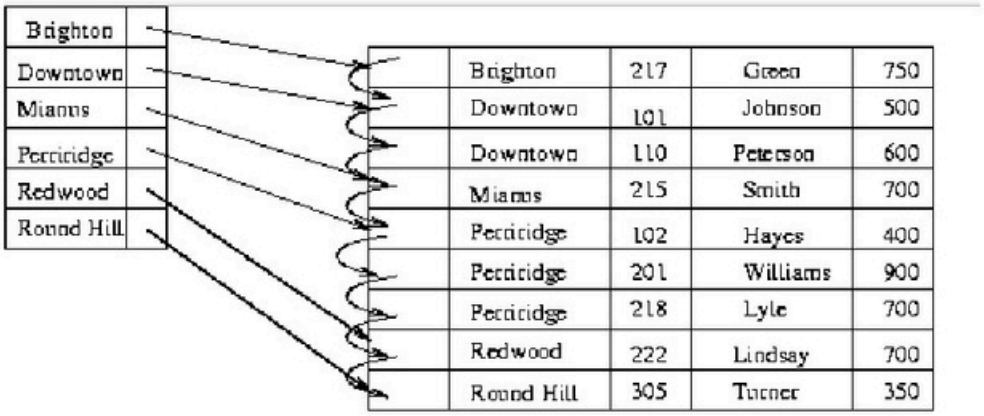


Fig: Dense index

Sparse Index

- In this indexing technique, index records are created only for some of the records.
- To locate a record, we find the index with the largest search key value less than or equal to the search key value we are looking for.
- We start at that record pointed to by the index record, and proceed along the pointer in file (i.e. sequentially) until we find the desired record.

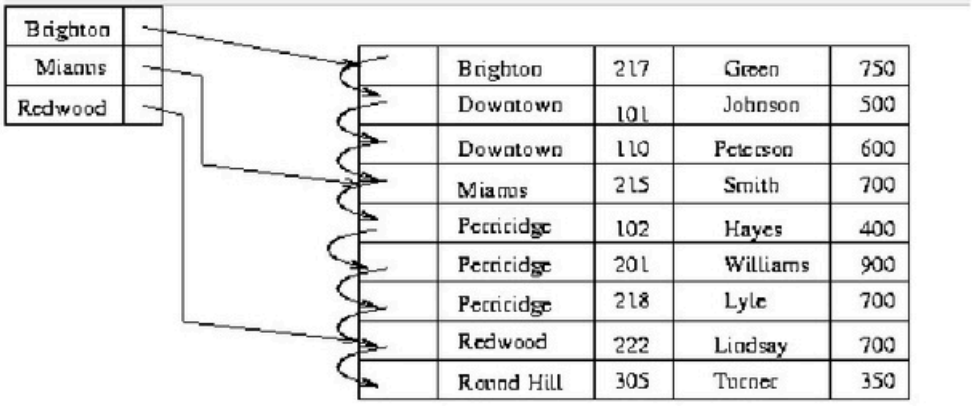


Fig: Sparse Index

Secondary index

- We can use extra level of indirection to implement secondary indices on each search key that are not candidate key.
- The pointer in such secondary index do not point directly to the file but to a bucket that contains pointer to the file.
- Secondary index must be dense with and index entry for each search key values and the pointer to every record in the file.
- Secondary indices improve the performance of queries that uses keys other than the search key of the primary index

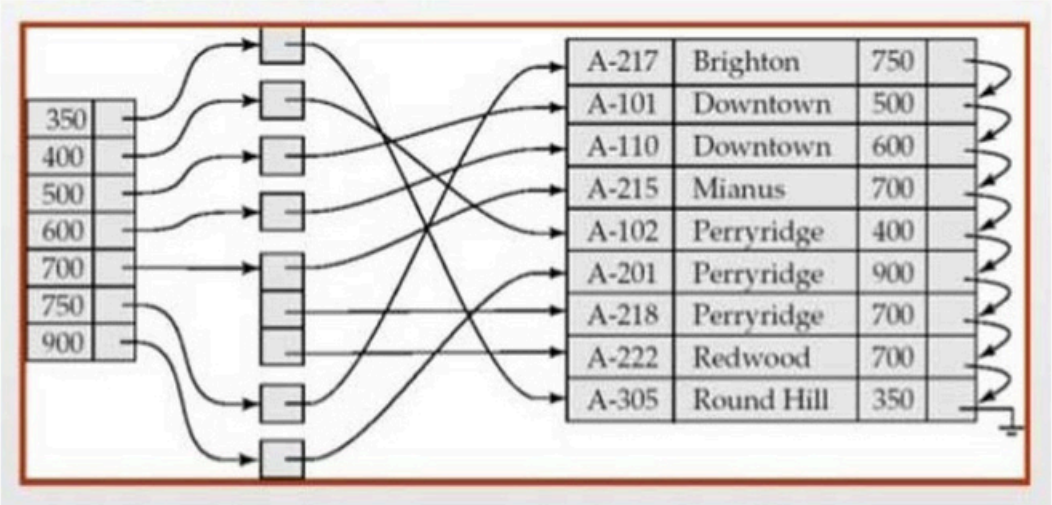


Fig: Secondary Index on balance field of account

SQL to create Index

- We use SQL DDL statements "Create Index Statement" to create index on a given relation.
- A simple index is an index on a single column, while a composite index is an index on two or more columns.
- The syntax for creating an index is.

  CREATE INDEX "index_name" ON "table_name" (column_name);

  Example:

  Consider a relational schema Employee(id, name, address).

  1. Create index on column name address.

     CREATE INDEX IDX_address ON Employee (address);

  2. Create index on column name name and address.

     CREATE INDEX IDX_address_name ON Employee (name, address);

B+ tree index file

- A B+ tree is a balanced binary search tree that follows a multi-level index format.
- The main disadvantage of index sequential file organization is that performance degrades as the file size grow.
- Although this degradation can be reduced by reorganization of files, it requires periodic reorganization of entire file.
- B+ tree file structure maintains the efficient tree as it automatically reorganizes itself after every insertion and deletion.

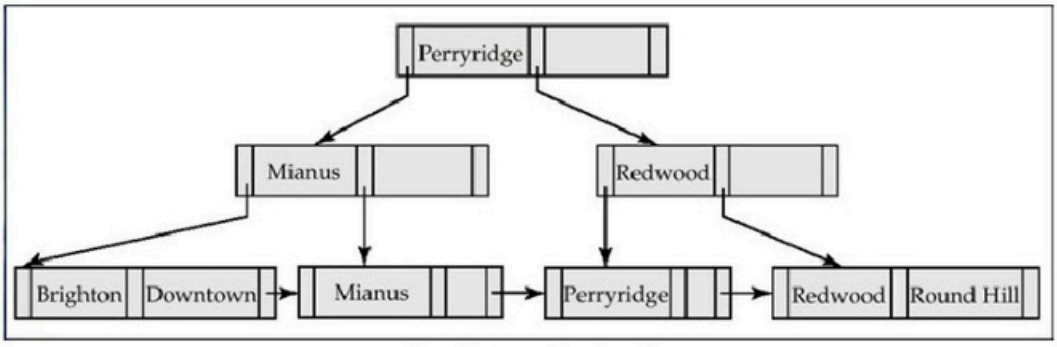


Fig: B-tree of order 3

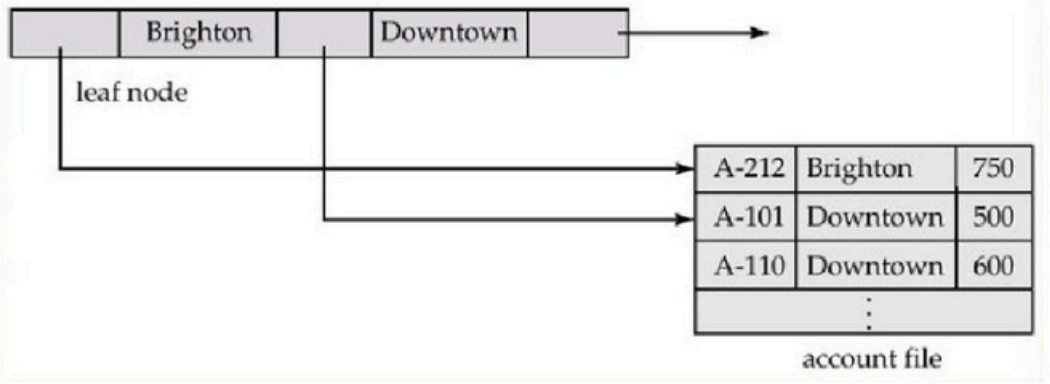


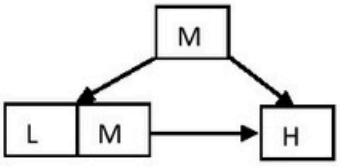
Fig: Leaf node mapping to Record

Properties

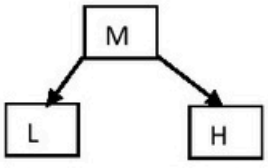
- It is also called as balanced tree as all the leaf node are at same level.
- All the non-leaf node has one fewer key than the number of its children.
- Each internal node in the N order B tree, except the root, must have at least $\lceil n/2 \rceil$ children and at most n children. Example: if the order is 7, then minimum number of children each internal node must be 4 and maximum is 7 children's.
- All the node except root have at least (n/2)-1 keys i.e. half and at most (m-1) keys. Example: if the order is 7, then minimum number of keys in each node must be 3 keys and maximum is 6 keys.
- All non-leaf node has one fewer keys than the number of its children.

Insertion in B+ Tree

- Traverse B-tree until we find leaf node suitable for insertion.
- We have two cases for inserting a key.
  - i. Node is not full
    - Insertion is done in the node in increasing order of key
  - ii. Node is full
    - a. If a node is full and is a leaf, classify the keys L(Lowest), M(Middle) and H(Highest) and split the node as below.



    - b. If a node is full and is non leaf, classify the keys L(Lowest), M(Middle) and H(Highest) and split the node as below.

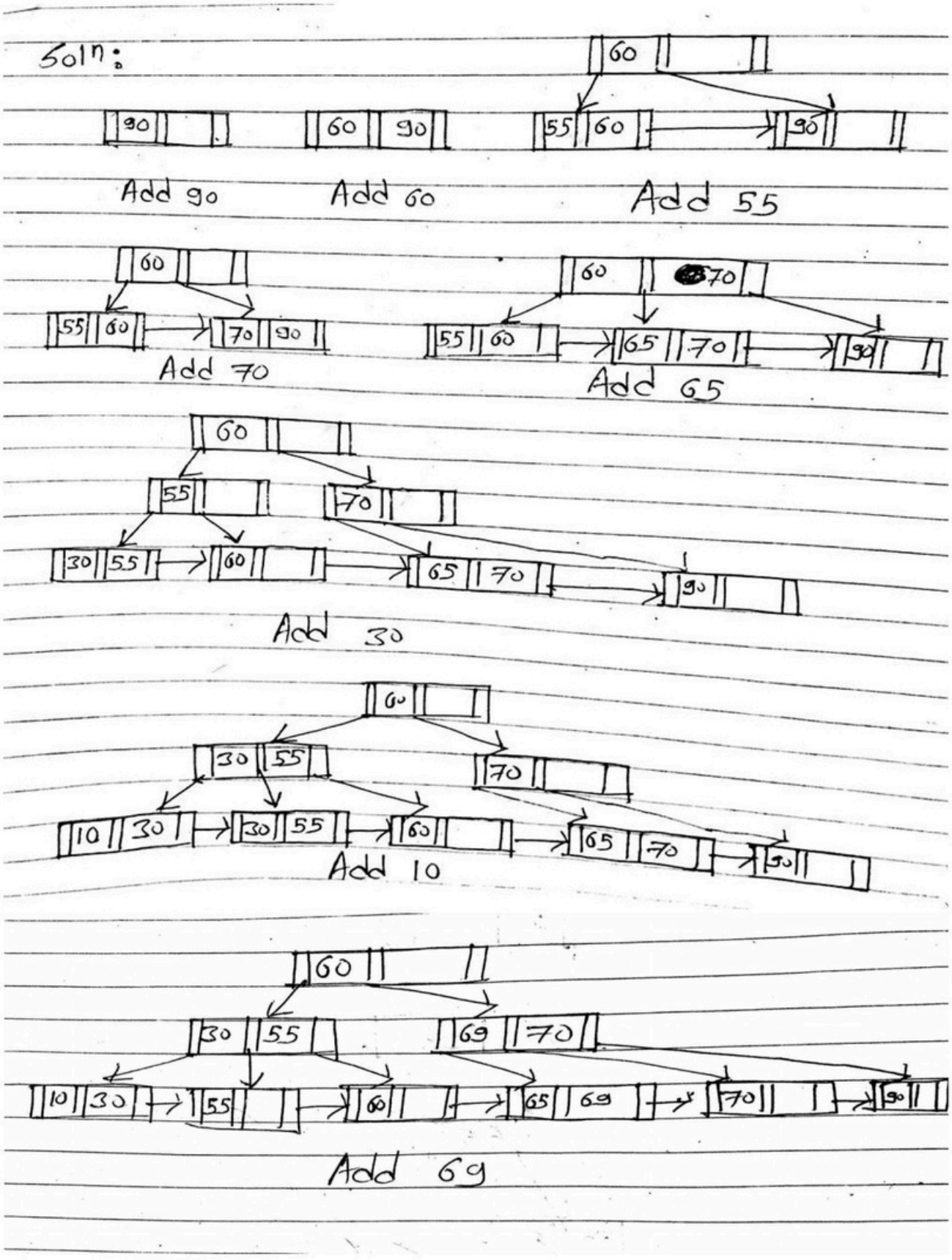


- The median value (M) is pushed one level up in the parent node. If the parent node is not full then accommodate the median value otherwise repeat same procedure in case II.

Example:

Construct B+ tree of order 3 for the following set of keys.

90, 60, 55, 70, 65, 30, 10, 69



This is the final B+ tree.

Construct B+ tree of order 4 for the following set of keys.

2, 3, 5, 7, 11, 17, 19, 23, 29, 31

Note: if the order is odd, then M = (N+1)/2 the item but if the order is even, then M = **(N/2) th item (preferred** howerver (N+1)/2 th item also can be used as middle number))

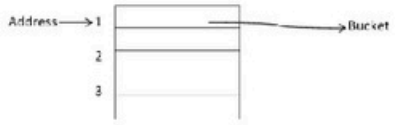
See class note for solution... ... ...

Static Hashing

- Static hashing uses hash functions in which the set of bucket addresses is fixed. Such hash functions cannot easily accommodate databases that grow significantly larger over time.
- Formally, let K denote the set of all search-key values, and let B denote the set of all bucket addresses. A hash function h is a function from K to B.
- Let h denote a hash function. To insert a record with search key Ki, we compute h(Ki), which gives the address of the bucket for that record. Assume for now that there is space in the bucket to store the record. Then, the record is stored in that bucket.
- To perform a lookup on a search-key value Ki , we simply compute h(Ki), then search the bucket with that address. Suppose that two search keys, K5 and K7, have the same hash value; that is, h(K5) = h(K7). If we perform a lookup
  on K5, the bucket h(K5) contains records with search-key values K5 and records with search-key values K7. Thus, we have to check the search-key value of every record in the bucket to verify that the record is one that we want.
- Deletion is equally straightforward. If the search-key value of the record to be deleted is Ki, we compute h(Ki), then search the corresponding bucket for that record, and delete the record from the bucket.
- Hashing can be used for two different purposes.
  1. In **a hash file organization,** we obtain the address of the disk block containing a desired record directly by computing a function on the search-key value of the record.
  2. In a **hash index organization,** we organize the search keys, with their associated pointers, into a hash file structure.

Hash Indices

- A hash index organizes the search keys with their associated pointers into a hash file structure.
- A hash file structure is simply a hash table which is an array addressed via hash function.



- A hash function can be defined as a function that is used to compute the address of a record in hash table.
- **In, hash index organization,** we apply a hash function on a search key to identify a bucket (a unit of storage that can be used to store search key), and store the key and its associated pointers in the bucket (or in overflow buckets).

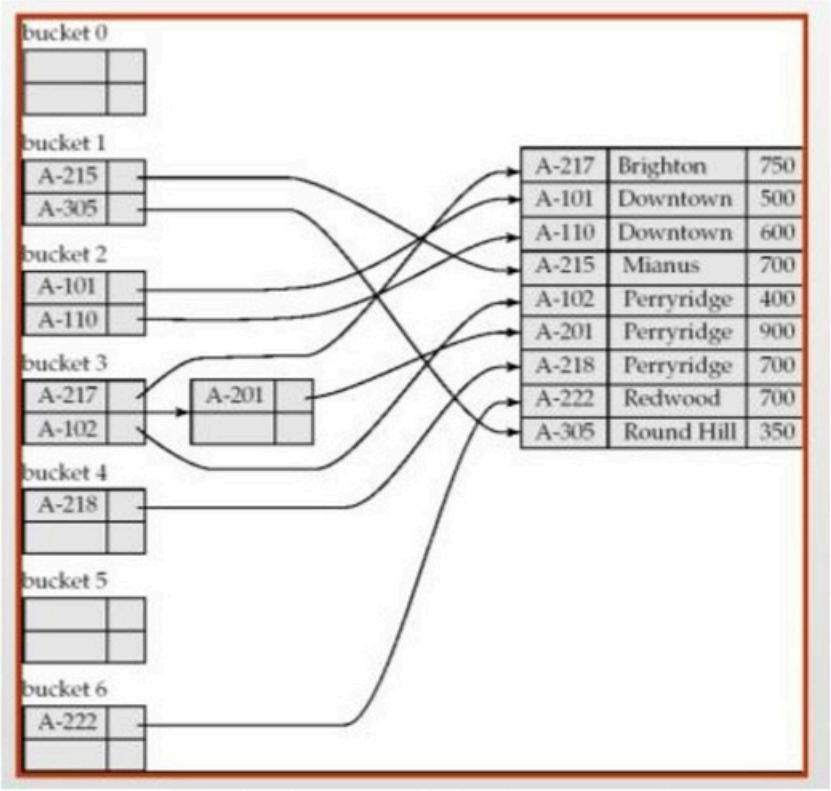


Fig: Hash index on search key Account number

- A hash function, h(K), is a mapping function that maps all the set of search-keys K to the bucket where keys and its associated pointers are placed.
- The hash function in the above function is

  H(K)= Sum of the digits of account number % 7
- The hash index has 7 buckets each of size 2.
- A situation called hash overflow /collision may occur multiple keys are mapped to same location and the number of mapped keys for particular location is greater than the size of bucket. Example: in the above figure, one of the bucket has 3 keys mapped so overflow occurs and for handling it has an **overflow bucket.**

- Formally let K denote the set of all search keys value, B denote the set of all bucket address and h denote the hash function. To insert a record with search key k_i, we compute $h(K_i)$ which gives the address of bucket for that record. That record is then stored in that bucket
- Deletion is also simple. If the search key value of the record to be deleted is Ki. We compute h(ki) to find the corresponding bucket for that record and delete the record from the bucket.

Handling bucket overflow

- If the bucket doesn't have enough space to store the currently hashed key value, a bucket overflow is said to occur. Bucket overflow can occur for the following reasons.

i. Insufficient Buckets

ii. Skew: Some buckets are assigned more records than others so bucket may overflow even when other bucket have space. This situation is called bucket skew.

- The worst possible hash function maps al search key values to same bucket. Such a hash function is undesirable because all the records have to be kept in same bucket. **An ideal hash function** distributes the stored key uniformly across the buckets so that every bucket has the same number of records. We handle bucket overflow by using overflow buckets. If a record must be inserted in bucket b and bucket b is already full, the system provides overflow bucket for b and insert the record into the overflow bucket as shown in above figure. If the overflow bucket is also full then the system provides another overflow bucket and so on. All the overflow buckets of the given bucket are chained together in a linked list. This kind of overflow

handling technique is called as **overflow chaining.** Also bucket overflow can be reduced by choosing the bucket size big enough to store the index for all the records but should not waste too much waste space.

- The form of hash structure that we have just described is sometimes referred to as **closed hashing.**
- Under an alternative approach, called **open hashing**, the set of buckets is fixed, and there are no overflow chains.
- Instead, if a bucket is full, the system inserts records in some other bucket in the initial set of buckets B. One policy is to use the next bucket (in cyclic order) that has space; this policy is called linear probing. Other policies, such as computing further hash functions, are also used.
- Closed hashing is preferable for database systems. The reason is that deletion under open hashing is troublesome.
- However, in a database system, it is important to be able to handle deletion as well as insertion. Thus, open hashing is of only minor importance in database implementation.

Assignment:

1. **Explain how deletion operation is performed in B+ tree? Also give necessary example.**
2. **Write short notes on**
   i. **Dynamic Hashing**

`37