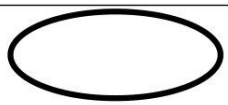
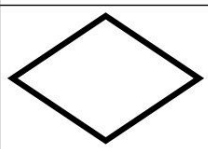
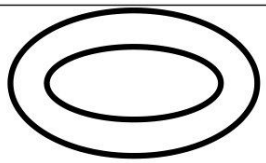
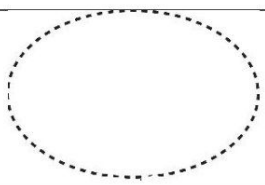
Unit 2

ER and Relational Model

Introduction to ER Model

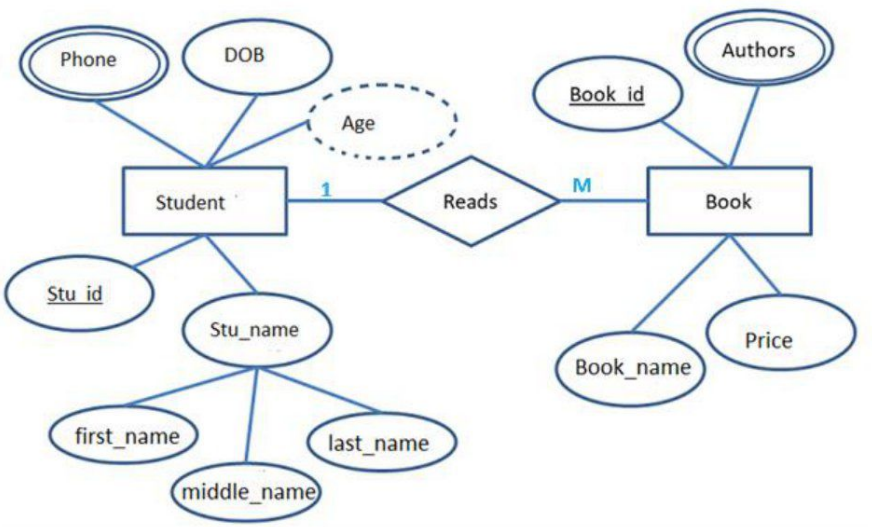
- In 1976, Peter Chen developed the Entity-Relationship (ER) model.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- An Entity-Relationship Model represents the structure of the database with the help of a diagram containing entities, relationships among them, and attributes of the entities.
- Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.
- Creation of an ER diagram, which is one of the first steps in designing a database, helps the designer(s) to understand and to specify the desired components of the database and the relationships among those components.

Symbols used in ER-diagram

Rectangle:-		to represent entity set
Ellipse:-		to represent attributes
Diamonds :-		to represent Relationship among entity set
Lines:-		to Link attributes to entity sets and entity sets to relationship.
Double ellipse:-		To represent multivalued attributes
Dashed Ellipse:-		To represent derived attribute

Double Rectangle:		To represent weak entity set

Example



- ✓ Here ER-diagram shows the relationship named reads in between two entities Student and Book.
- ✓ This is a one-to-many relationship.
- ✓ Student entity have attributes such as key attribute of stu_id, a composite attribute of stud_name with the components such as first_name,middle_name and last_name multivalued attribute of phone and derived attribute of age with its base attribute of DOB.
- ✓ Similarly, the book entity has a key attribute of book _id, a multivalued attribute of authors and other two attributes such as book_name and price.

Basic concept

This model is based on three basic concepts:

1. Entity and Entity sets
2. Relationships and Relationship sets
3. Attribute

1. Entity and Entity set

Entity

- An entity is a "thing" or "object" in the real world that is distinguished from all other objects. An Entity may be
  - ✓ **concrete** such as person, book, driver, product, employee etc.
  - ✓ it may be **abstract** such as course

- An entity has a set of properties, and the values for some set of properties may uniquely identify an entity.
- For example, employee may have employee_id property whose value uniquely identifies that employee. Thus, the value 677-89-9011 for employee_id would uniquely identify one particular employee in the organization.

Entity Set

- An entity set is a set of entities of the same type that share the same properties.
- **Example:**
  - ✓ The set of all employees of an organization can be defined as the entity set employees.
  - ✓ Similarly, the entity set projects represents the set of all projects undertaken by the organization.
- An entity set is represented by a set of attributes. Possible attributes of the employee's entity set are emp_id, emp_name, address, sex, date_of_birth. Possible attributes of the project entity set are project_id, Project_name.
- For each attribute there is a set of permitted values, called the domain of that attribute, which can be assigned to the attribute.

  One instance of the entity set **employee** and **project** are shown as:

Employee

Eid	Name	Sex	Address
E1	Ram	Male	Pokhara
E2	Shyam	Male	Butwal
E3	Sita	Female	Chitwan
E4	Krishna	Male	Kathmandu

Project

Project_id	Project _Name
P1	MIS
P2	Image processing
P3	Linux
P4	OS

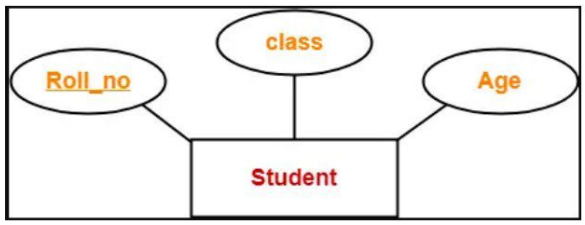
2. Attributes

- Attributes are descriptive properties possessed by each member of entity set.
- There exists a specific domain or set of values for each attribute from where the attribute can take its values.
- Eg. The student entity might have attributes like id, name, age program, semester etc.
- In ER diagram attributes are represented by an ellipse.

Attribute types

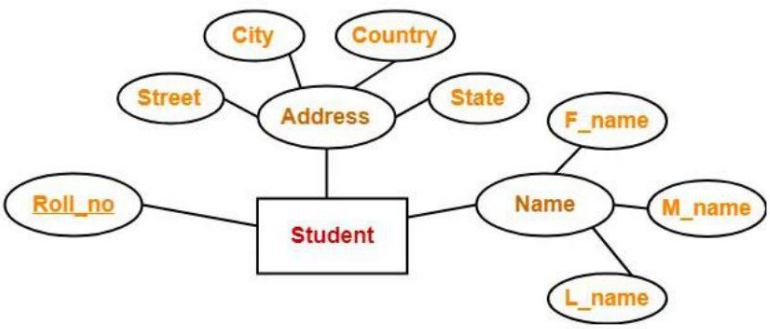
❖ Simple Attribute:

- ✓ Simple attributes are those attributes which cannot be divided further.
- ✓ Here, all the attributes are simple attributes as they cannot be divided further.



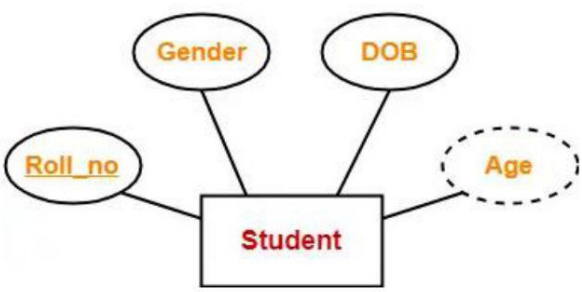
❖ Composite Attribute:

- ✓ Composite attributes are those attributes which are composed of many other simple attributes.
- ✓ Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.
- ✓ In ER diagram, composite attribute is represented by an ellipse comprising of ellipse.



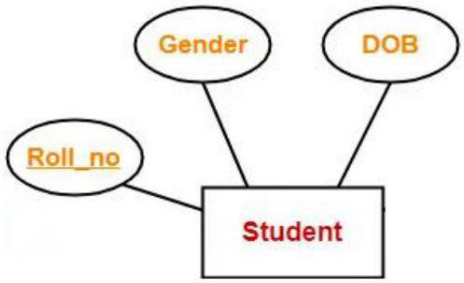
❖ Derived Attribute:

- ✓ Derived attributes are those attributes which can be derived from other attribute(s).
- ✓ Here, the attribute "Age" is a derived attribute as it can be derived from the attribute "DOB".
- ✓ In ER diagram, the derived attribute is represented by dashed ellipse.



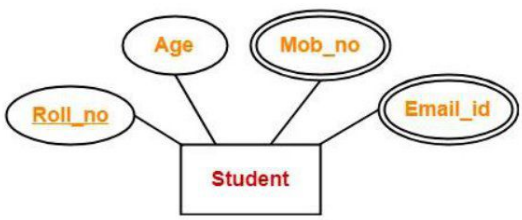
❖ Single valued Attribute:

- ✓ Single valued attributes are those attributes which can take only one value for a given entity from an entity set.
- ✓ Here, all the attributes are single valued attributes as they can take only one specific value for each entity.



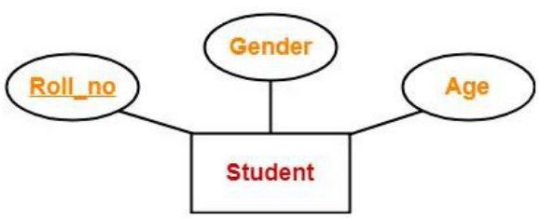
❖ Multivalued attribute

- ✓ Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.
- ✓ Here, the attributes "Mob_no" and "Email_id" are multi valued attributes as they can take more than one values for a given entity.
- ✓ In ER diagram, multivalued attribute is represented by double ellipse.



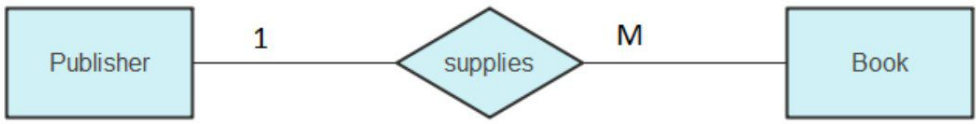
❖ Key attributes

- ✓ Key attributes are those attributes which can identify an entity uniquely in an entity set.
- ✓ Here, the attribute "Roll_no" is a key attribute as it can identify any student uniquely.
- ✓ In ER diagram, key attribute is represented by an ellipse with underlying lines.



Relationship and relationship sets

- A relationship is an association among several entities.
- It is represented using a diamond shape in the entity-relationship diagram. Example: Publisher supplies a book.



Here In the above example, the publisher and the book are entities whereas the word supplies is a relationship between those entities. Moreover, the attributes of the publisher entity can be that is a pub_id,Pub_name,address and the attributes of book entity can be book_id,book_name.

- A relationship set is a set of relationships of same type. It is a mathematical relation of n>=2 (possibly non distinct) entity sets.
- If E1, E2, E3,...., En are entity sets, then a relationship set R is a subset of {(e1, e2, e3, ..., en) | e1 ∈ E1, e2 ∈ E2, ..., en ∈ En} Where (e1, e2, e3, ..., en) is a relationship.
- The association between entity sets is referred to as participation; that is the entity sets E1,E2,....En participate in a relationship set R.
- A relationship instance in an E-R schema represents an association between entities in the real world that is being modeled.

Consider the two-entity set publisher and book. The relationship set supplies to denote the association between publisher and book that publisher supplies.

Publisher entity set

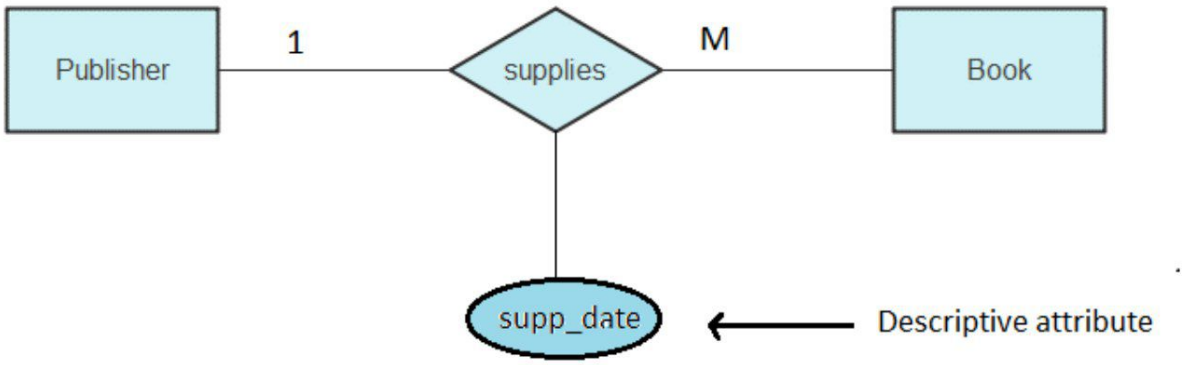
Pub_id	Pub_name	Address
P1	Asmita book suppliers	Newroad
P2	Goodwill	Kalanki
P3	Pariabi prakasan	Biratnagar

Book entity set

Book_ id	Book name
B1	Dbms
B2	Operating system
B3	C++
B4	Math

As an illustration, an individual Publisher entity Asmita book suppliers who has pub_id and the Book entity who has Book_id B1 and B2, participate in a relationship instance of supplies. This relationship instance represents that Asmita book suppliers supplies two books i.e DBMS and Operating System.

A relationship may also have attributes called descriptive attributes. Consider a relationship set supplies with entity sets Publisher and Book. We could associate the attribute date with that relationship to specify date when supplier supplies book.



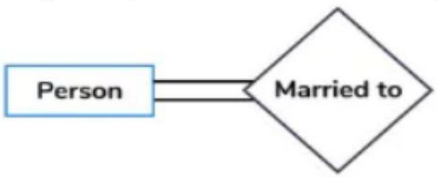
Degree of Relationship Set

- It refers to the number of entities sets that participate in a relationship.
- Unary relationship is of degree 1.
- Binary relationship is of degree 2.
- Ternary relationship is of degree 3.
- Relationships are often binary.

On the basis of degree of a relationship set, a relationship set can be classified into the following types:

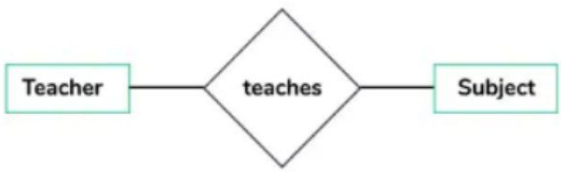
Unary Relationship set

- Unary relationship set is a relationship set where only one entity set participates in a relationship set.
- Eg. One person is married to only one person.



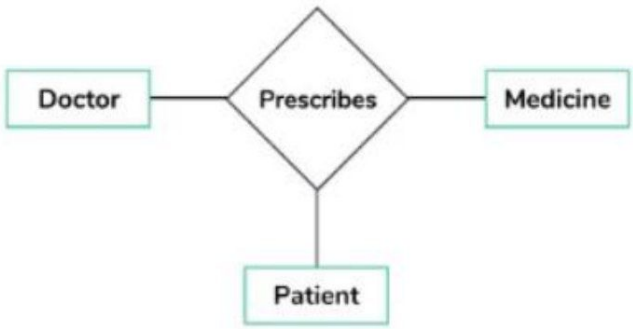
Binary Relationship set

- Binary relationship set is a relationship set where two entity sets participate in a relationship set.
- Eg. Teacher teaches a subject here 2 entities are teacher and subject for the relationship teacher teaches subject



Ternary Relationship set

- Ternary relationship set is a relationship set where three entity sets participate in a relationship set.
- Eg. In the real world, a patient goes to a doctor and doctor prescribes the medicine to the patient, three entities Doctor, patient and medicine are involved in the relationship "prescribes".



N-ary relationship set

- Ternary relationship set is a relationship set where n entity sets participate in a relationship set.

Constraints in E-R models

An ER enterprise schema may define certain constraints to which the contents of a database must conform. We have

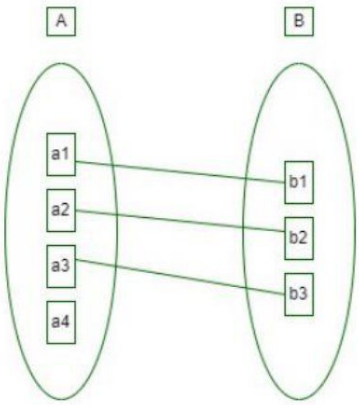
i. Mapping cardinalities
ii. Participation constraints

Mapping cardinalities

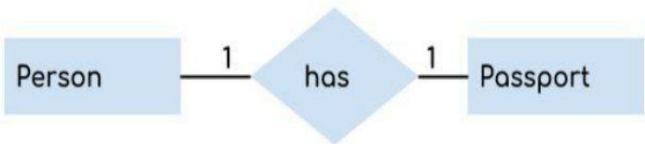
- Mapping cardinalities, or cardinality ratios, express the number of entities with which another entity can be associated via a relationship set.
- Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set R between entity sets A and B, the mapping cardinality must be one of the following:

One to One: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

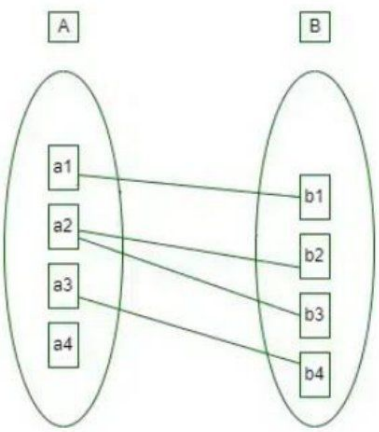


For example, a person has only one passport and a passport is given to one person.



One to many:

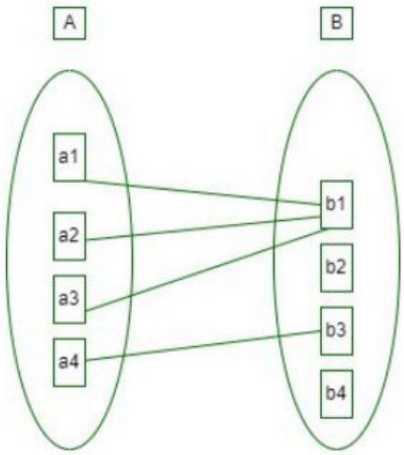
An entity in A is associated with any number (zero or more) of entities in B, and an entity in B, however, B can be associated with at most one entity in A.



For example, a customer can place many orders but a order cannot be placed by many customers.

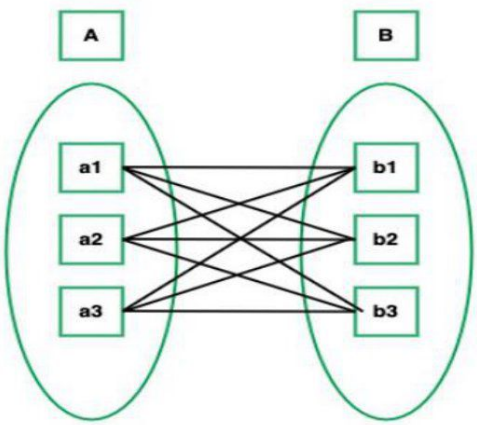
Many to One:

An entity in A is associated with at most one entity in B, and an entity in B, however, can be associated with any number (zero or more) of entities in A.



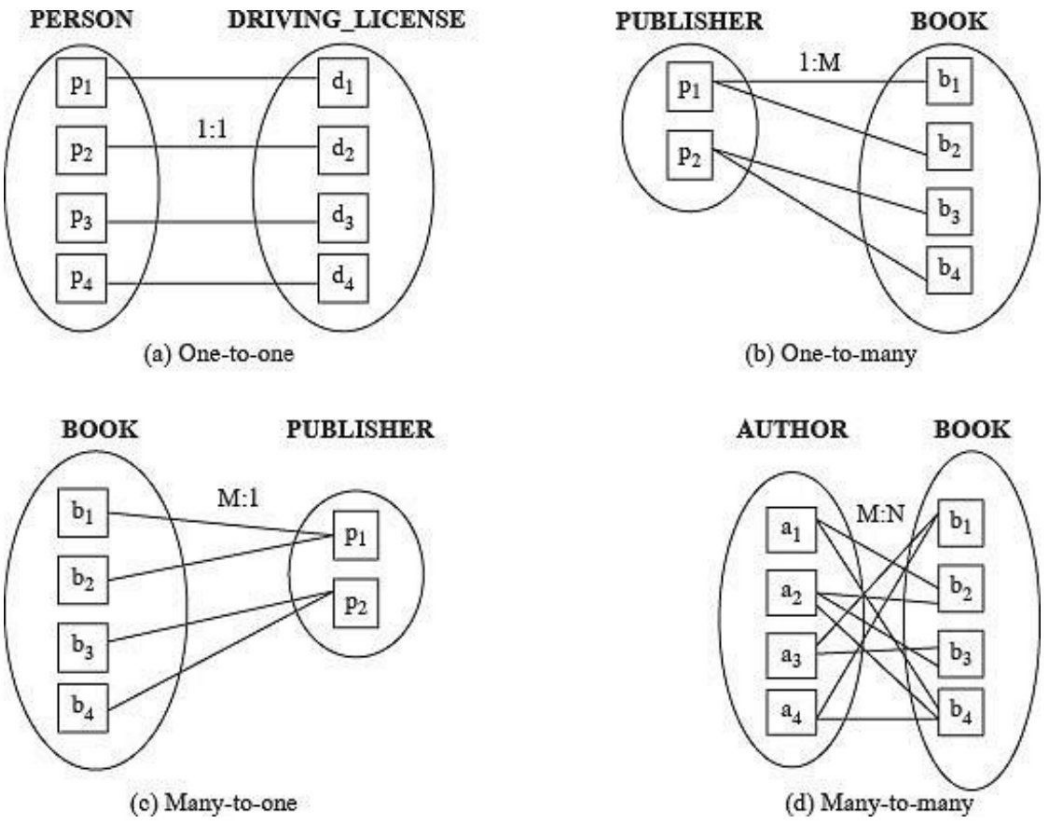
For example, many students can study in a single college, but a student cannot study in many colleges at the same time.

Many to Many: An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



Eg. Student can be assigned to many projects and a project can be assigned to many students.

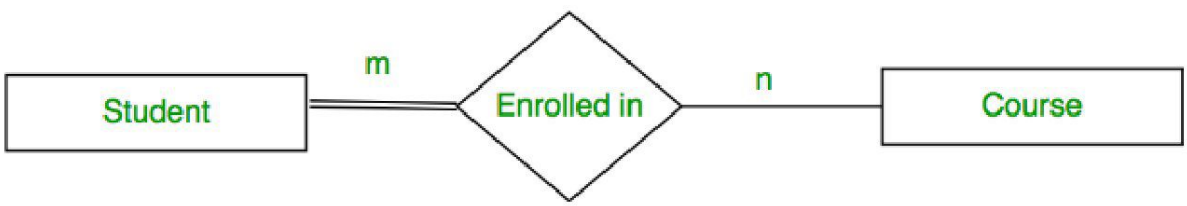
Alternative Example



Participation Constraints

Participation Constraint is applied to the entity participating in the relationship set.
1. Total Participation – Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

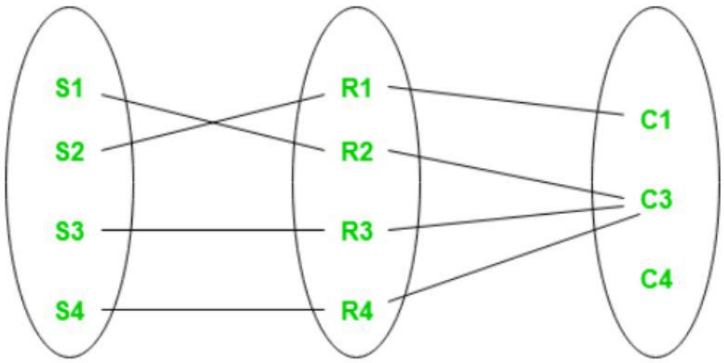
2. Partial Participation – The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.



Total Participation and Partial Participation

The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.

Using Set, it can be represented as



Set representation of Total Participation and Partial Participation

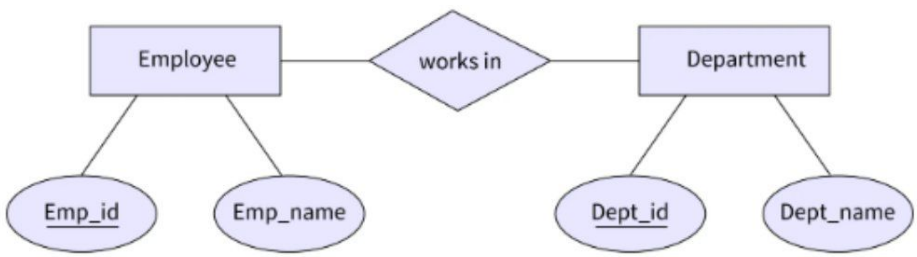
Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

Strong and weak entity

Strong Entity

- A strong entity is not dependent on any other entity in the schema.
- A strong entity will always have a primary key.
- Strong entities are represented by a single rectangle.
- The relationship of two strong entities is represented by a single diamond.
- A strong entity set is a set that is made up of many strong entities.
- It may or may not participate in a relationship between entities.

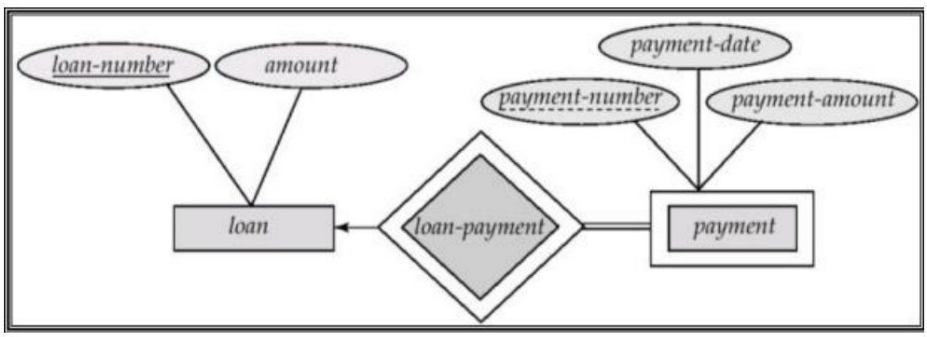
Example:



- In the given image, we have two strong entities namely Employee and Department hence they are represented using a single rectangle.
- The relationship between them is works in i.e it gives information about an employee working in a particular department hence it is represented using a single diamond.
- In the above image, if we remove the relationship between the two entities then also the two entities will exist i.e Employee as well as Department will exist since they both are independent of each other, this explains **the independent nature of strong entities.**

Weak entity

- A weak entity is dependent on a strong entity to ensure its existence.
- Unlike a strong entity, a weak entity does not have any primary key. It instead has a partial discriminator key.
- A weak entity is represented by a double rectangle.
- The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as identifying relationship.
- It always participates in the relationship between entities since its existence is dependent on other strong entities and it always has total participation.



- We underline the discriminator of a weak entity set with a dashed line.
- payment-number – discriminator (partial key)of the payment entity set
- Primary key for payment – (loan-number, payment-number)

Keys

- ✓ Key is a way to specify how entities within a given entity set are distinguished.
- ✓ Conceptually, individual entities are distinct; from a database perspective, however, the difference among them must be expressed in terms of their attributes.
- ✓ Therefore, the values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.
- ✓ A key allows us to identify a set of attributes that is enough to distinguish entities from each other.
- ✓ Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

Role of keys in relational database

- KEYS in DBMS is an attribute or set of attributes which helps us to identify a row(tuple) in a relation(table).
- In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys in RDBMS ensure that you can uniquely identify a table record despite these challenges.
- They allow us to find and establish the relation between two tables.
- Help us to enforce identity and integrity in the relationship**.**

Let us consider the table named **Employee_info**

Employee_id	Name	Address	Pan_no	Age
1	Pradip	Kathmandu	55821	43
2	Hari	Pokhara	55837	32
3	Nikita	Chitwan	55237	24
4	Pradip	Butwal	55345	43
5	Sita	Lalitpur	55432	48
6	Ramesh	Chitwan	55755	25
7	Hari	Pokhara	55896	32

- 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. How many trials before you get the right data?
- To avoid all this, Keys are defined to easily identify any row of data in a table.

Various types of keys are:

1. Super key
2. Candidate key
3. Primary key
4. Alternate key
5. Foreign key
6. Composite key

1. 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 above example following are the examples of super keys

{Employee_id}
{Pan_no}
{Employee_id,Name}
{Pan_no,Name}
{Employee_id,Address} etc.

2. Candidate key

- ✓ A set of minimal attributes that can identify each tuple uniquely in the given relation is called a candidate key.
- ✓ The value of candidate key must always be unique.
- ✓ It is possible to have multiple candidate keys in a relation.
- ✓ In above example following are the examples of candidate keys

  {Employee_id}

  {Pan_no}

3. Primary key

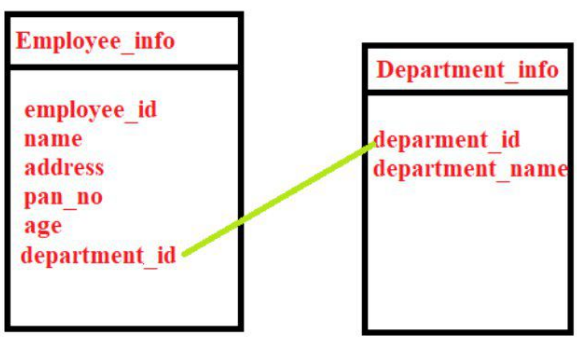
- ✓ The primary key is a candidate key that is most appropriate to become the main key for any table. It is a key that can uniquely identify each record in a table.
- ✓ In the above example {Employee_id} can be chosen are primary key.

4.Alternate key

- ✓ Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys
- ✓ In this example {Pan_no} act as alternate keys.

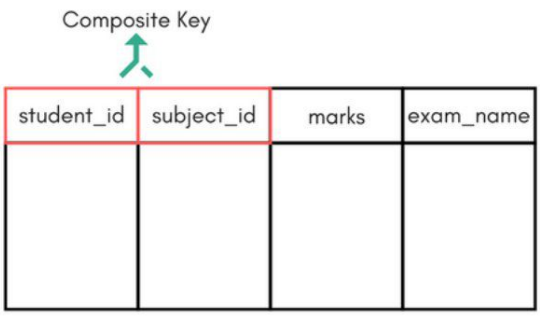
5. Foreign key

- ✓ Foreign keys are the column of the table used to point to the primary key of another table.
- ✓ Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee_info table. That's why we link these two tables through the primary key of one table.
- ✓ We add the primary key of the DEPARTMENT table, Department_id, as a new attribute in the employee_info table.
- ✓ In the EMPLOYEE table, Department_id is the foreign key, and both the tables are related.



6.Composite key

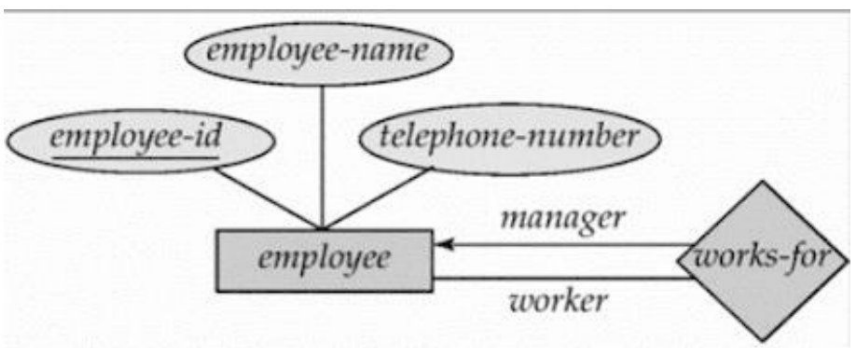
- ✓ Key that consists of two or more attributes that uniquely identify any record in a table is called Composite key. But the attributes which together form the Composite key are not a key independently or individually.
- ✓ In the above picture we have a Score table which stores the marks scored by a student in a particular subject.
- ✓ In this table student_id and subject_id together will form the composite key



Score Table - To save scores of the student for various subjects.

Roles in E-R diagram

- It is a function that plays(relationship) in an entity called its role.
- Roles are normally explicit and not specified.
- Useful when the relationship meaning needs to be clarified.
- Example, the relationship works-for define in employees as manager or worker. The labels “manager” and “worker” are called roles; they specify how employee entities interact via the works-for relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional and are used to clarify the semantics of the relationship.

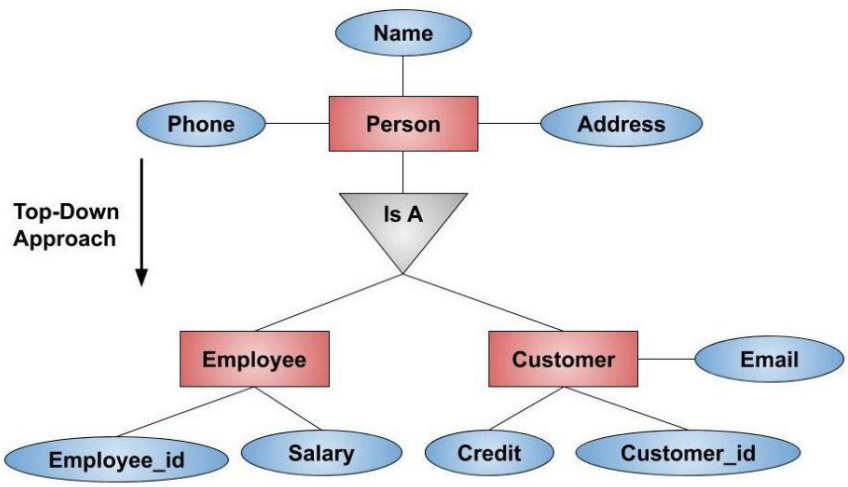


EER(Extended / Enhanced ER model)

- Today the complexity of the data is increasing so it becomes more and more difficult to use the traditional ER model for database modeling.
- To reduce this complexity of modeling we must make improvements or enhancements to the existing ER model to make it able to handle the complex application in a better way.
- Enhanced entity-relationship diagrams are advanced database diagrams very similar to regular ER diagrams which represent the requirements and complexities of complex databases.
- It is a diagrammatic technique for displaying the Sub Class and Super Class; Specialization and Generalization; Aggregation etc.

Specialization

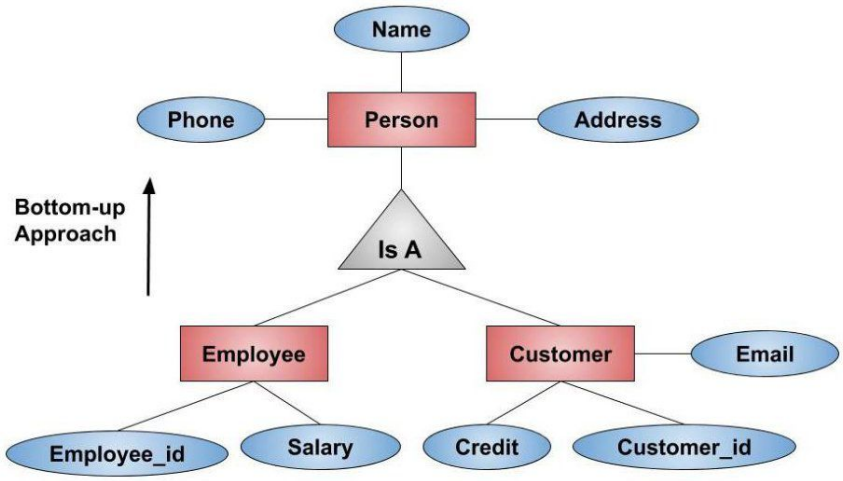
- The process of designating sub grouping within an entity set is called specialization.
- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA (E.g. customer "is a" person).
- Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.



Specialization

Generalization

- A bottom-up design process – combine a number of entities sets that share the same features into a higher-level entity set.
- Generalization is a containment relationship that exists between a higher level entity set and one or more lower level entity sets.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.



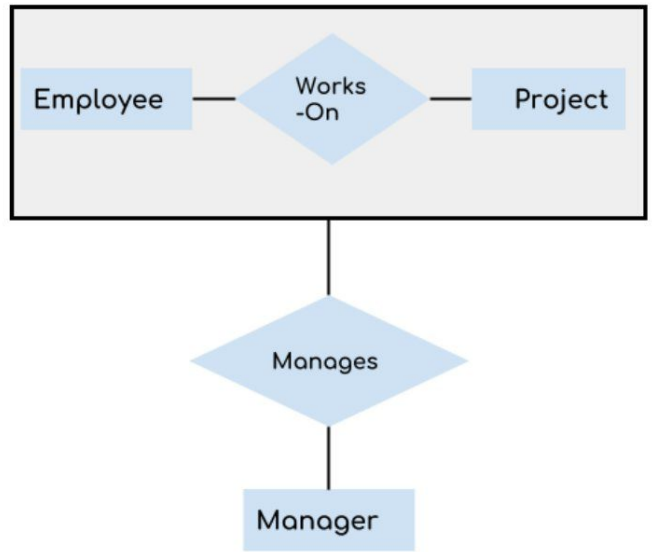
Generalization

Generalization vs specialization

Generalization	Specialization
Generalization works in Bottom-Up approach.	Specialization works in top-down approach.
Generalization can be defined as a process of creating groupings from various entity sets	Specialization can be defined as process of creating subgrouping within an entity set
In Generalization process, what actually happens is that it takes the union of two or more lower-level entity sets to produce a higher-level entity sets.	Specialization is the reverse of Generalization. Specialization is a process of taking a subset of a higher-level entity set to form a lower-level entity set
Generalization process starts with the number of entity sets and it creates high-level entity with the help of some common features	Specialization process starts from a single entity set and it creates a different entity set by using some different features.
Generalization is normally applied to group of entities.	We can apply Specialization to a single entity.
In Generalization, size of schema gets reduced.	In Specialization, size of schema gets increased.

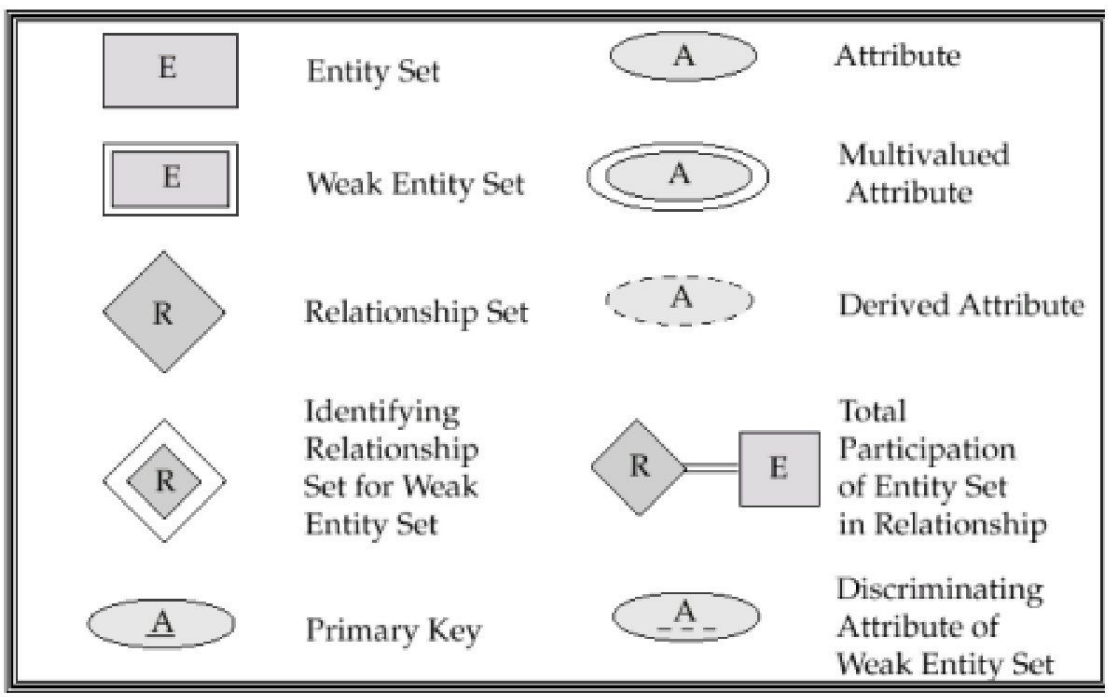
Aggregation

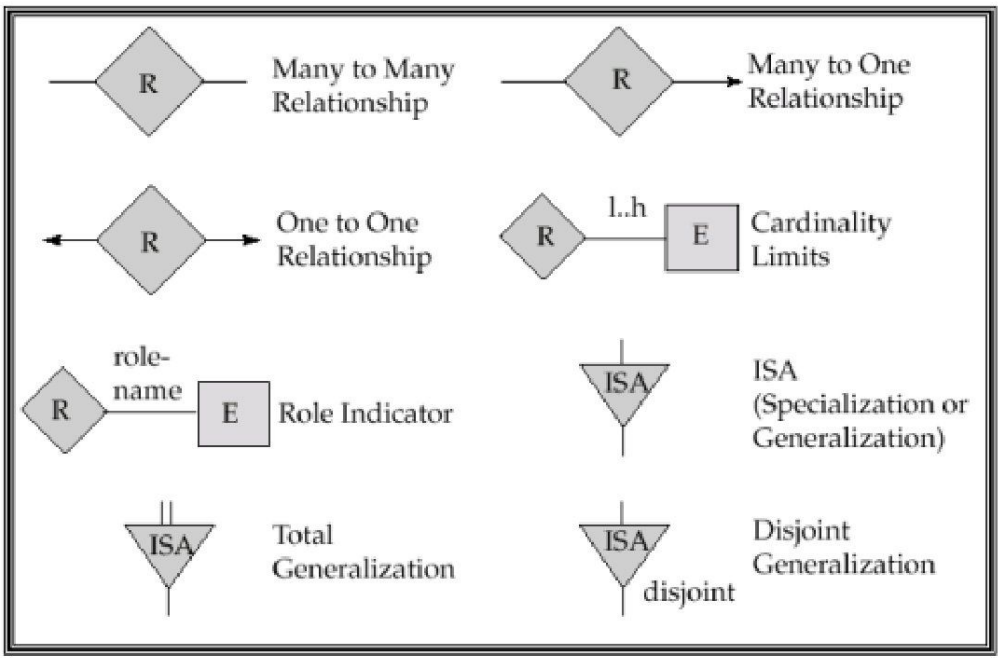
Aggregation in DBMS (Database Management System) is a process of combining one or more entities into a single one that is the more meaningful entity. Also, it is a process in which a single entity does not make sense in a relationship, and one cannot infer results from that relationship so the relationship of one or more entities acts as one entity.



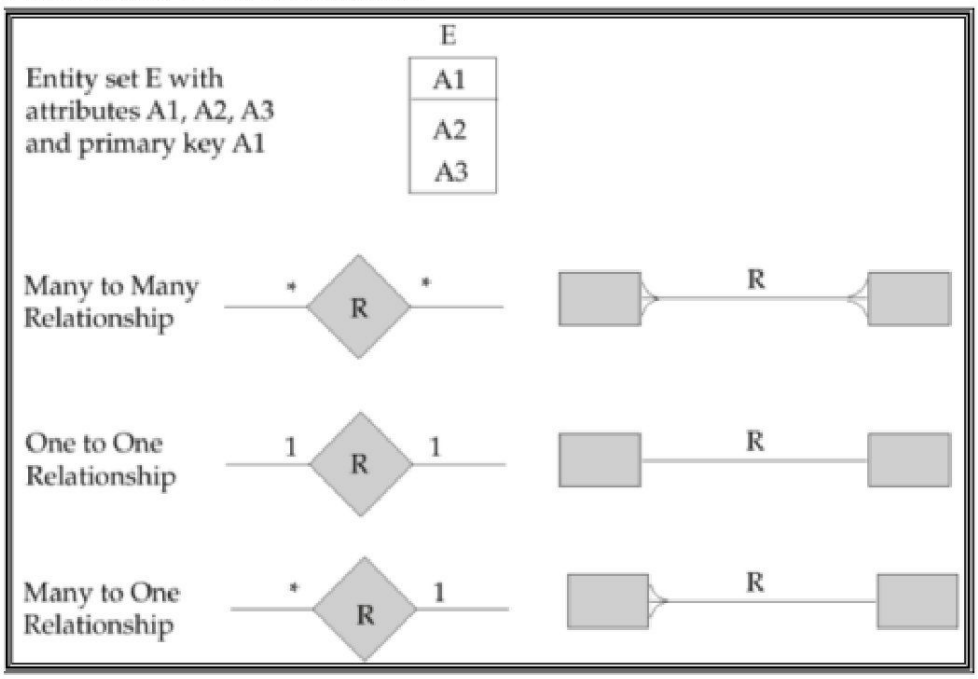
In the real world, we know that a manager not only manages the employee working under them, but he has to manage the project as well. In such scenario if entity "Manager" makes a "manages" relationship with either "Employee" or "Project" entity alone then it will not make any sense because he has to manage both. In these cases, the relationship of two entities acts as one entity. In our example, the relationship "Works-On" between "Employee" & "Project" acts as one entity that has a relationship "Manages" with the entity "Manager".

Summary of Symbols Used in E-R Notation

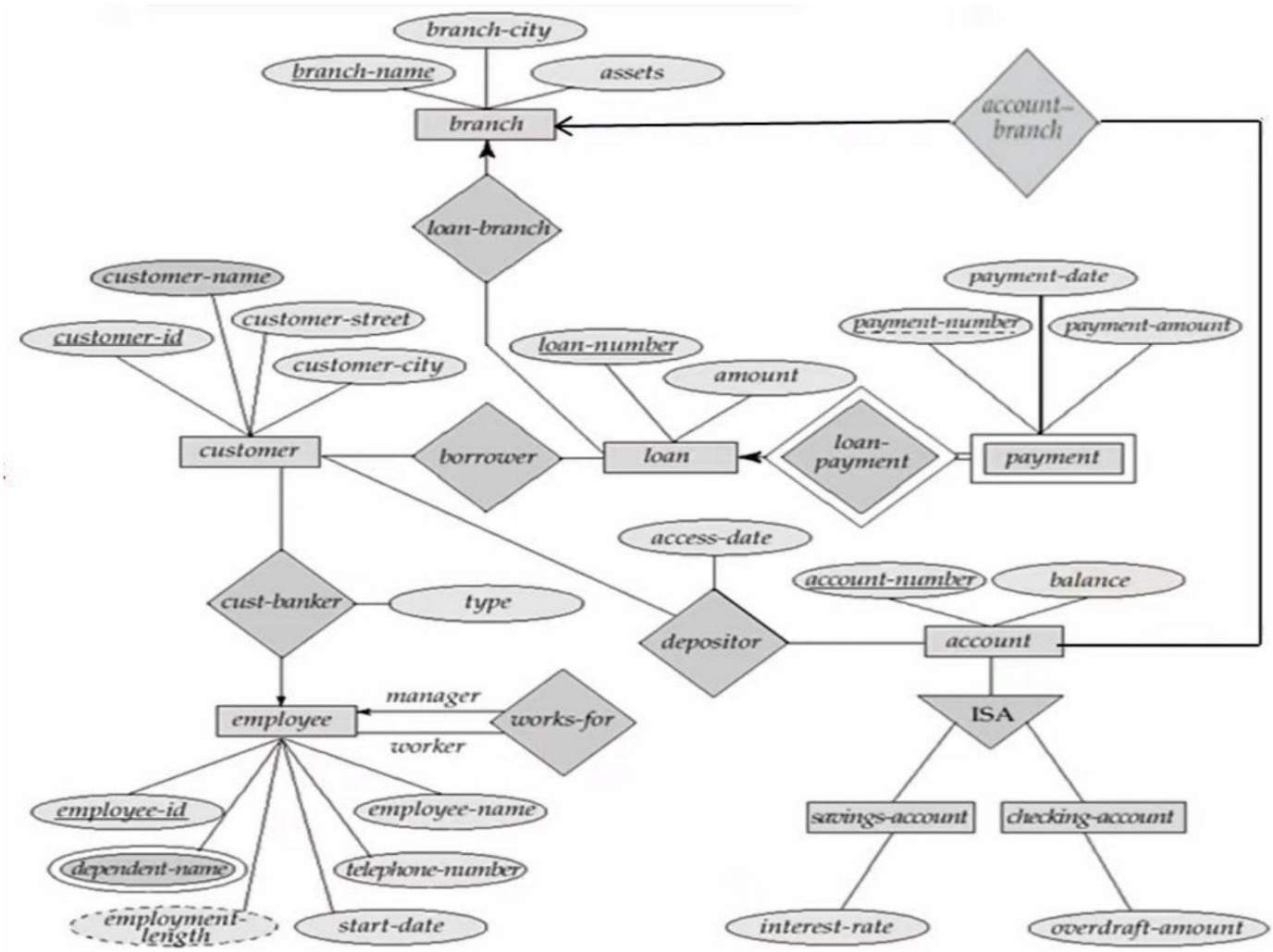




Alternative ER notations



Draw an ER-diagram for Banking enterprise that keeps the information about the employee, customer, loan, account and payment.



Introduction to Relational Model

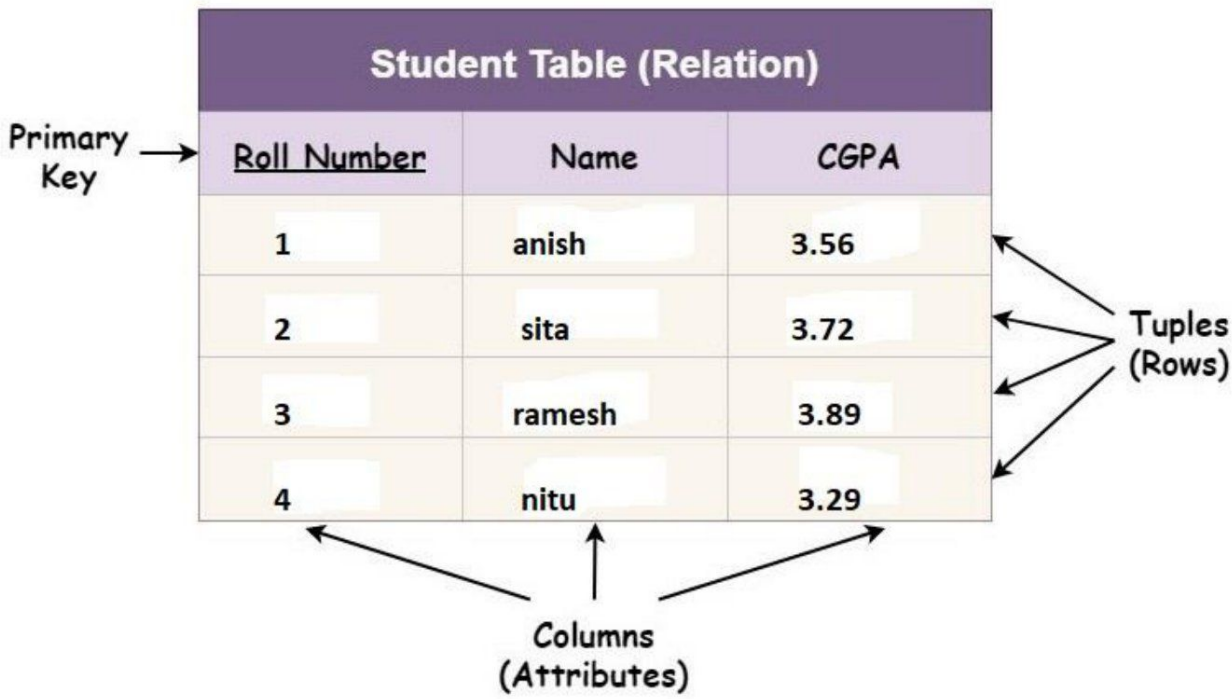
Definitions and Terminology

- Relational model is today a primary data model in which database is represented as a collection of Relation where each relation is represented by a two dimensional table.
- The relational model was first proposed by E.F. Codd in 1970.
- Because of its simplicity, the relational model is now the dominant model for commercial data processing operation.

Structure of relational model

- In this model, the data is organized into a collection of two-dimensional inter-related tables, also known as relations.
- Each relation is a collection of columns and rows.
    - ✓ column represents the attributes of an entity
    - ✓ the rows (or tuples) represents the records.

Consider a case we wish to store the name, the CGPA attained, and the roll number of all the students of a particular class. This structured data can be easily stored in a table as described below.



As we can notice from the above relation:

- Any given row of the relation indicates a student i.e. the row of the table describes a real-world entity.
- The columns of the table indicate the attributes related to the entity. In this case, the roll number, name and CGPA of student.

- ✓ Relational database is a collection of organized set of tables related to each other, and from which data can be accessed easily.
- ✓ **A Relational Database management System (RDBMS) is a database management system based on the relational model .It is used to manage Relational database.**
- ✓ Examples of RDBMS are Oracle, MySQL, Microsoft SQL Server, PostgreSQL etc.

As discussed earlier, a relational database is based on the relational model. This database consists of various components based on the relational model. These include:

Relation: Two-dimensional table used to store a collection of data elements.

Tuple: Each row of a table is known as record. It is also known as tuple. For example, the following row is a record that we have taken from the above table.

1	anish	3.56

Attribute: Column of the relation, depicting properties that define the relation.
Eg. Roll_Number, Name, CGPA

Domain: A domain is a set of permitted values for an attribute in table. For example, a domain of CGPA must be in the range of 0 to 4.

Relation Schema: A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g. student (Roll_Number, Name, CGPA) is the relation schema for **student**. If a schema has more than 1 relation, it is called Relational Schema.

Relational Instance: It is the collection of records present in the relation at a given time. Above table shows the relation instance of **student** at a particular time. It can change whenever there is an insertion, deletion, or update in the database.

Degree: It is the total number of attributes present in the relation.eg. The **Student** relation defined above has degree 3.

Cardinality: It specifies the number of entities involved in the relation i.e., it is the total number of rows present in the relation. The **student** relation defined above has cardinality 4.

Properties of Relational model

- Each relation (or table) in a database has a unique name
- An entry at the intersection of each row and column is atomic (Each relation cell contains exactly one atomic (single) value)
- Each row is unique; no two rows in a relation are identical
- Each attribute (or column) within a table has a unique name
- Tuples in a relation do not have to follow a significant order as the relation is not order-sensitive.
- Similarly, the attributes of a relation also do not have to follow certain ordering, it's up to the developer to decide the ordering of attributes.

Advantages of Relational model

- ✓ A relational database model is much simpler compared to other data models because data is stored in the form of rows and columns.
- ✓ Since there are several tables in a relational database, certain tables can be made to be confidential. These tables are protected with username and password such that only authorized users will be able to access them. The users are only allowed to work on that specific table.
- ✓ Relational database uses primary keys and foreign keys to make the tables interrelated to each other. Thus, all the data which is stored is non-repetitive. Which means that the data does not duplicate. Therefore, the data stored can be guaranteed to be accurate.
- ✓ It is flexible, so one can get the data in the form which he/she wants. He/she can extract the information very easily and information can also be manipulated by using various operators such as project, join, etc.
- ✓ The Structure of Relational database can be changed without having to change any application.

Disadvantages of relational model

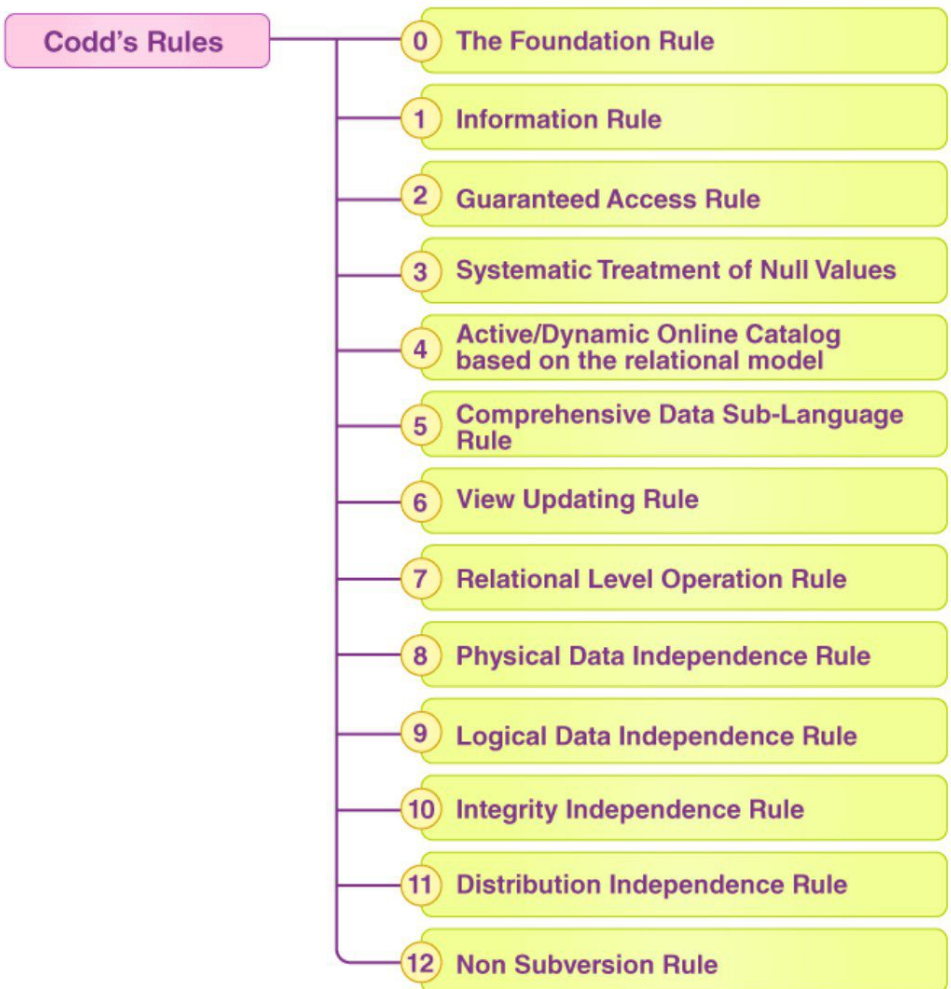
- ✓ The performance of the relational model depends upon the number of relations present in the database.
- ✓ Hence, as the number of tables increases, the requirement of physical memory increases.
- ✓ The structure becomes complex and there is a decrease in the response time for the queries.
- ✓ Because of all these factors, the cost of implementing a relational database increase.

Difference between DBMS and RDBMS

DBMS	RDBMS
DBMS applications store data as file.	RDBMS applications store data in a tabular form.
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other.
Normalization is not present in DBMS.	Normalization is present in RDBMS
DBMS does not support distributed database.	RDBMS supports distributed database.
DBMS is meant to be for small organization and deal with small data.	RDBMS is designed to handle large amount of data.
The software and hardware requirements are low.	The software and hardware requirements are higher.
The data in a DBMS is subject to low security levels with regards to data manipulation.	It features multiple layers of security while handling data.
Examples: Window Registry, Forxpro, dbase III plus etc.	Examples: MySQL, PostgreSQL, SQL Server, Oracle, Microsoft Access etc.

E.F. Codd's 12 Rules for RDBMS

- ✓ E.F Codd was a Computer Scientist who invented the Relational model for Database management. Based on relational model, the Relational database was created.
- ✓ Codd proposed 13 rules popularly known as Codd's 12 rules to test DBMS's concept against his relational model.
- ✓ Codd's rule actualy define what quality a DBMS requires in order to become a Relational Database Management System(RDBMS).



Rule 0: The Foundation Rule

The database must be in relational form. So that the system can manage the database through its relational capabilities.

Rule 1: The Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: The Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: The Systematic Treatment of Null Values

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.

Rule 4: The Dynamic/Active Online Catalog on the basis of the Relational Model

Database dictionary(catalog) is the structure description of the complete Database and it must be stored online. The Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

Rule 5: The Comprehensive Data SubLanguage Rule

The relational database supports a variety of languages, and in order to access the database, the language has to be linear, explicit, or a well-defined syntax, character strings. It must support the following operations: view definition, integrity constraints, data manipulation, data definition, as well as limit transaction management. It is considered a DB violation if the DB permits access to the data and information without the use of any language.

Rule 6: The View Updating Rule

A view table can theoretically be updated, and DB systems must update them in practice.

Rule 7: The Relational Level Operation (or High-Level Insert, Delete, and Update) Rule

A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Rule 8: The Physical Data Independence Rule

The working of a database system should be independent of the physical storage of its data. If a file is modified (renamed or moved to another location), it should not interfere with the working of the system.

Rule 9: The Logical Data Independence Rule

It indicates that any modifications made at the logical level (or the table structures) should not have an impact on the user's experience (application). For example, if a table is split into two separate tables or into two table joins in order to produce a single table, the application at the user view should not be affected.

Rule 10: The Integrity Independence Rule

A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application

Rule 11: The Distribution Independence Rule

This rule denotes that a database must function properly even if it's stored in multiple locations and used by various end-users. Let's say a person uses an application to access the database. In such a case, they must not be aware that another user is using the same data, and thus, the data they always obtain is only available on one site. The database can be accessed by end-users, and each user's access data must be independent in order for them to run SQL queries.

Rule 12: The Non-Subversion Rule

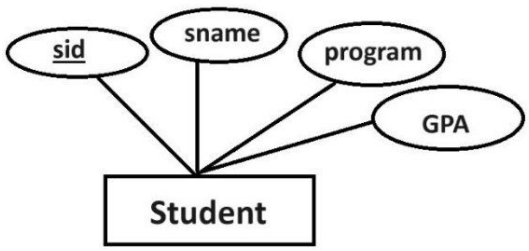
The non-subversion rule defines RDBMS as a SQL language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data**.**

Reducing ER diagram to relational schema

Rule 1: Strong Entity Set with only simple attributes

- ✓ Attributes of the schema will be attributes of the entity set.
- ✓ The primary key of the schema will be the key attribute of the entity set.

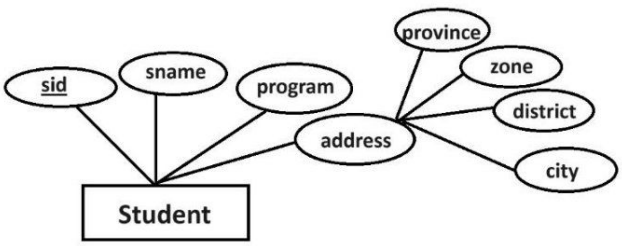
Example:



Schema:

Student(sid, sname, program, GPA)

Rule 2: Strong Entity Set with composite attributes



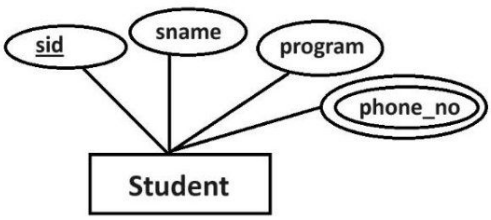
- ✓ A strong entity set with any number of composite attributes requires only one schema in relational model.
- ✓ While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

Schema:

Student(sid, sname, program, province, zone, district, city)

Rule 3: Strong Entity set with multivalued attributes.

- ✓ A strong Entity set with any number of multivalued attributes will require two schemas in relational model.
- ➢ One schema will contain all the simple attributes with the primary key.
- ➢ Other schema will contain the primary key and all the multivalued attributes.



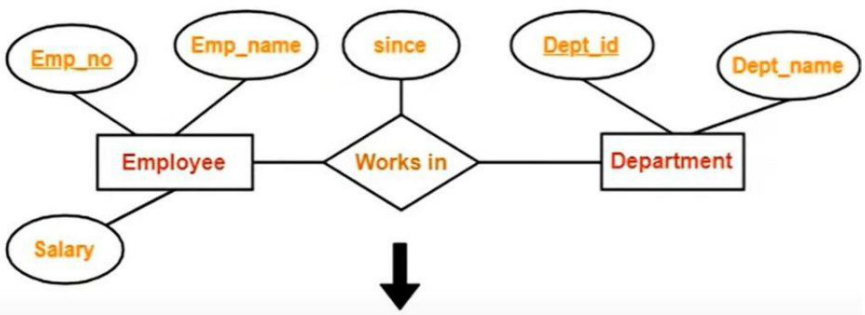
Schema:

Student(sid, sname, program)

Student_phone_no(sid, phone_no)

Rule 4: Translating Relationship set into a schema.

- ➢ A relationship set will require one schema in relational model.
- ➢ Attributes of a schemas are:
  - ✓ Primary key attributes of the participating entity set.
  - ✓ Its own descriptive attribute if any.
- ➢ A set of non-descriptive attributes will be the primary key.



Schemas:

Employee(Emp_no, Emp_name, Salary)

Workin(Emp_no, Dept_id, since)

Department(Dept_id, Dept_name)

Rule 5: For Binary relationships with cardinality ratios

the following four cases are possible:

Case-1: Binary relationship with cardinality ratio m:n

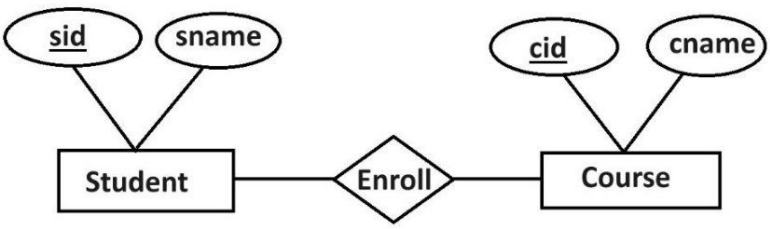
Case-2: Binary relationship with cardinality ratio 1:n

Case-3: Binary relationship with cardinality ratio m:1

Case-4: Binary relationship with cardinality ratio 1:1

Case-1: Binary relationship with cardinality ratio m:n

Union of the primary key attributes from the participating entity sets become the primary key of the relationship.



Schemas:

Student(sid, sname)

Enroll(sid, cid)

Course(cid, cname)

If enroll date is mentioned as descriptive attribute, then

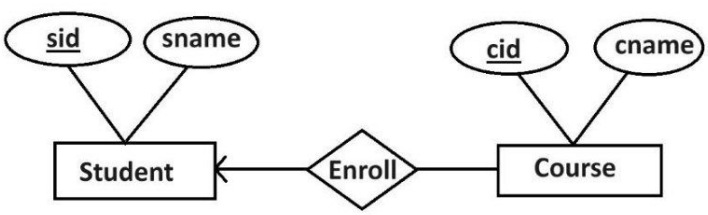
Enroll(sid, cid, enroll_date)

Case-2: Binary relationship with cardinality ratio m:1/1:m

- ✓ Construct two schemas, one for entity set for 1 side and another for entity set at M side.
- ✓ Add the descriptive attributes and a reference of the primary key of 1 side to the entity set at M side.

For Binary relationship with cardinality ratio 1:m

Example:



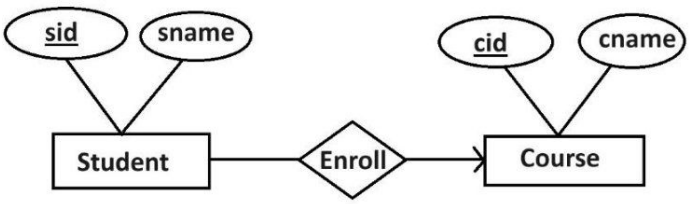
Schemas:

Student(sid, sname)

Course_enroll(sid, cid, cname)

For Binary relationship with cardinality ratio m:1

Example:



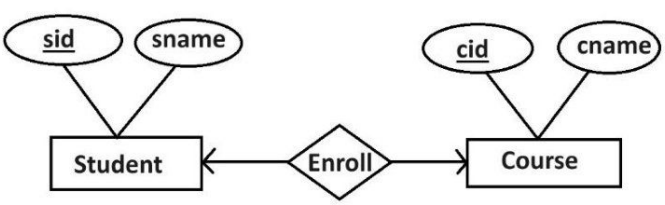
Schemas:

Student_enroll(sid, sname, cid)

Course(cid, cname)

Case-3: Binary relationship with cardinality ratio 1:1

Construct two schemas. In this case, either side can be chosen to act as the many side. That is, extra attributes can be added to either side of table corresponding to the two entity sets, but not at the same time.



- ✓ **If student entity set is considered many side**

  Schemas:

  Student_enroll(sid, sname, cid)
  Course(cid, cname)

- ✓ **If course entity set is considered many side**

  Schemas:

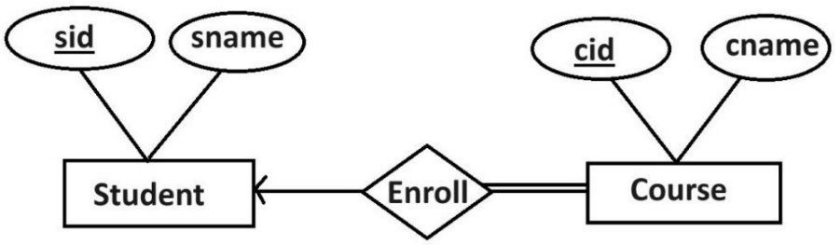
  Student(sid, sname)
  Course_enroll(sid, cid, cname)

Rule 6: For Binary Relationship With Both Cardinality Constraints and Participation Constraints

- ✓ Cardinality constraints will be implemented as discussed in Rule-05.
- ✓ Because of the total participation constraint, foreign key acquires NOT NULL constraint i.e. now foreign key can not be null.

Case-01: For Binary Relationship with Cardinality Constraint and Total Participation Constraint From One Side

Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.



Then, two schemas will be required-

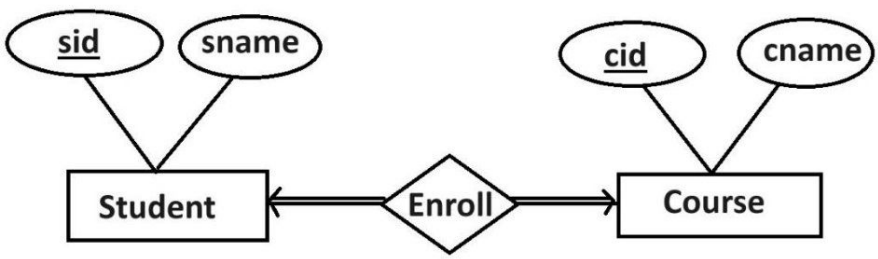
Student(sid, sname)

Student_Enroll(sid, cid, cname)

Because of total participation, foreign key sid has acquired NOT NULL constraint, so it can't be null now.

Case-02: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single schema.

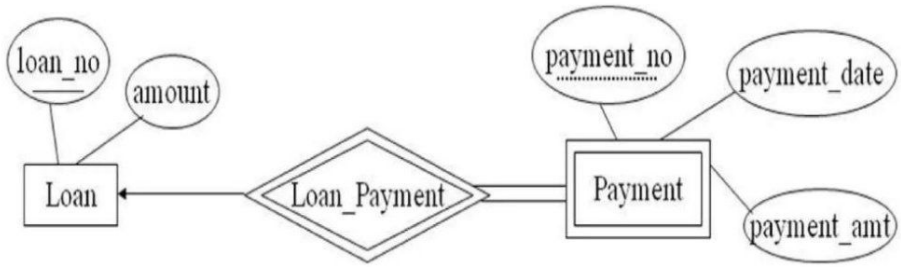


Here only one schema is required,

Student_enroll_course(sid, sname, cid, cname)

Rule 7: Representation of weak entity sets

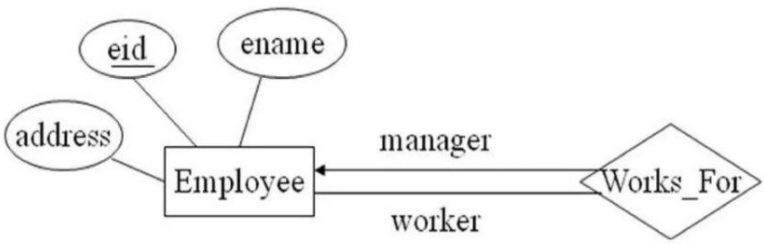
- ✓ A weak entity set becomes a schema that includes a column for the primary key of the identifying strong entity set.
- ✓ The primary key is constructed by the collection of foreign key and partial key.



Loan(loan_no, amount)

Payment(loan_no, payment_no, payment_date, payment_amt)

Rule 8: Representation of Recursive relationship sets



Two schemas will be constructed; one for entity set and one for relationship set.

Employee(eid, ename, address)

Works_for(mgrid,workerid)

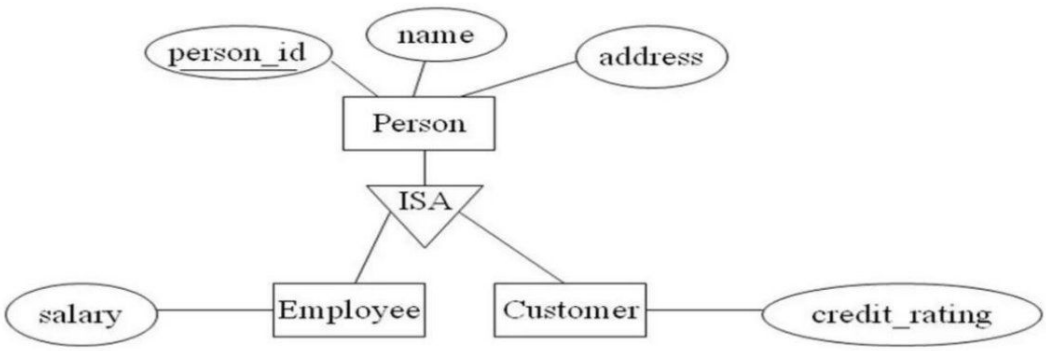
This ER diagram can also be represented by using a single relation schema. In such cases, the schema contains a foreign key for each tuple in the original entity set.

Employee(eid,ename,address,manager_id)

Rule 9:

Representation of Generalization/Specialization.

In case of generalization/specialization related ER diagram, one schema will be constructed for the generalized entity set and the schemas for each of the specialized entity sets.



Person(person_id,name,address)
Employee(person_id,salary)
Customer(person_id,credit_rating)

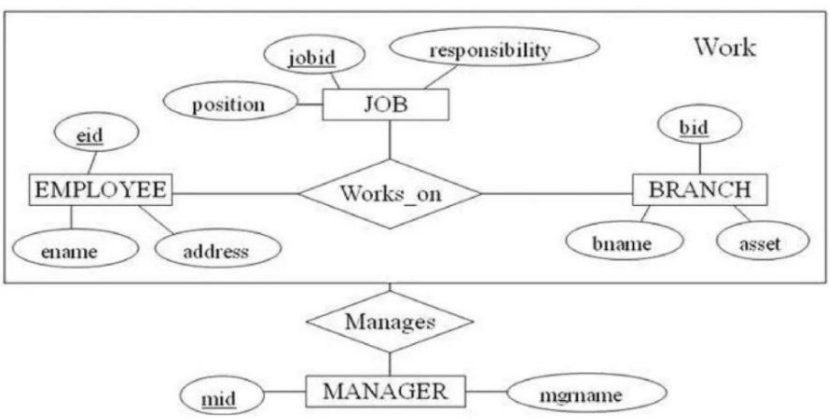
When the generalization/specialization is a disjointness case, the schemas are constructed only for the specialized entity sets.

Employee(Employee_id, name, address,salary)
Customer(customer_id, name, address, credit_rating)

Rule 10:

Representation of Aggregation

To represent aggregation, create a schema containing the primary key of the aggregated relationship ,primary key of the associated entity set and descriptive attributes(if any).



Employee (eid, name, address)

Branch(bid, bname, asset)

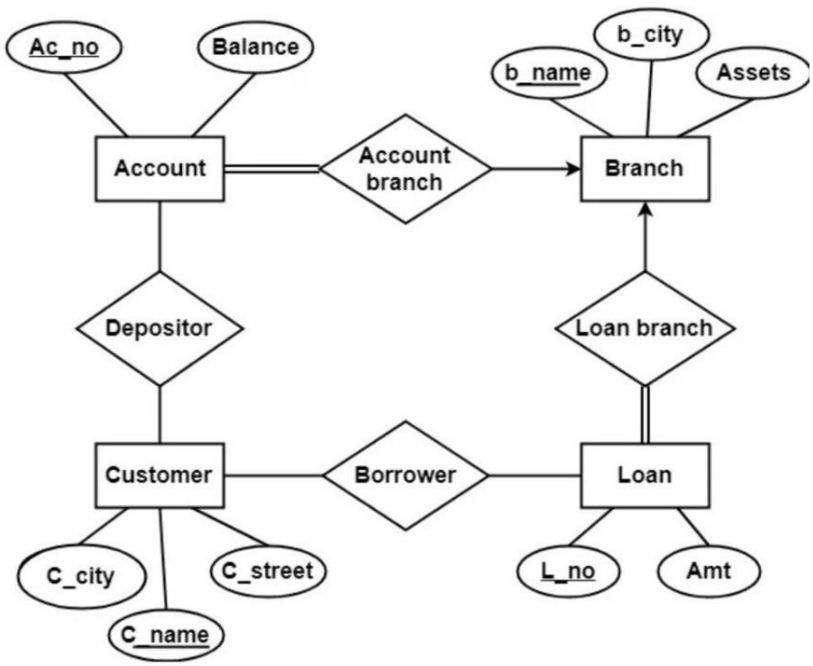
Job(jobid, position, responsibility)

Works_on(eid, bid, jobid)

Manager(mid, mgrname)

Manages(eid, bid, job_id, mid)

Example:



Applying the rules that we have learnt, minimum 6 tables will be required-

Account (Ac_no , Balance , b_name)

Branch (b_name , b_city , Assets)

Loan (L_no , Amt , b_name)

Borrower (C_name , L_no)

Customer (C_name , C_street , C_city)

Depositor (C_name , Ac_no)

Things to be understand

Many to Many cardinality

For example, a student can be enrolled many courses and a course can be enrolled by many students.

Student(sid,sname)

Enroll(sid,cid)

Course(cid,cname)

Table Student

sid	sname
1	Ram
2	Sita
3	Anish
4	Gita

Table Course

cid	cname
C1	C++
C2	DBMS
C3	Microprocessor

Table Enroll

sid	cid
1	C1
1	C2
2	C1
2	C2
3	C3
4	C2

Many to one cardinality

For example, a student can be enrolled only in one course, but a course can be enrolled by many students.

Student_enroll(sid,sname,cid)

Course(cid,cname)

Student_enroll

sid	sname	cid
1	Ram	c1
2	Sita	c1
3	Anish	c2
4	Gita	c3

Course

cid	cname
c1	C++
c2	DBMS
c3	Microprocessor

One to many cardinality

For example, a student can be enrolled many course, but a course can be enrolled by only one student.

Student_enroll(sid,sname)

Course(sid,cid,cname)

Student_enroll

sid	sname
1	Ram
2	Sita
3	Anish
4	Gita

Course

sid	cid	cname
1	c1	C++
2	c2	DBMS
1	c3	Microprocessor
2	c4	Instrumentation

Relational Algebra

- ✓ The relational algebra is a procedural query language.
- ✓ It consist set of operation that takes one or more relations as inputs and produce a new relation as output.
- ✓ The fundamental operations in relational algebra are selection, projection, union, set difference, Cartesian product, and rename.
- ✓ Set intersection, natural join, division and assignments other operations of relational algebra which can be define in terms of fundamental operations.

1. Fundamental Operations

- ✓ The fundamental operations **selection, projection and rename** on one relation so they called unary operations.
- ✓ Others operations **union, set difference and Cartesian product** operates on pairs of relations and so called binary operations.

1.1 Selection Operation

- ✓ The Select Operation selects tuples that satisfy a given predicate.
- ✓ Select is denoted by a lowercase Greek letter sigma (σ), with the predicate appearing as a subscript.
- ✓ The relation is specifying within parentheses after σ. That is, general structure of selection is $\sigma_{p(r)}$ where p is selection predicate.
- ✓ Formally, selection operation define as
  $\sigma_{p(r)} = \{t | t \in r \text{ and } p(t)\}$
  where p is formula in propositional calculus consisting terms connected by connectives: ^(and), ∨ (or), ¬ (not).
- ✓ Each term is in the format <attribute>op<attribute>or <constant>
  where op is one of the comparison operators: =,≠ ,<,≤,>,≥

Let us consider the following employee relation

emp_id	name	department	salary
1	ramesh	civil	68000
2	krishna	computer	75000
3	rita	software	65000
4	sita	mechanical	32000
5	juna	mechanical	85000
6	nikita	computer	60000
7	anish	civil	55000

Find all the employees working on computer department

$\sigma_{department="computer"}(employee)$

output

emp_id	name	department	Salary
2	krishna	computer	75000
6	nikita	computer	60000

Find all the employees whose salary is greater than 70000

$\sigma_{salary>70000}(employee)$

output

emp_id	name	department	salary
2	krishna	computer	75000
5	juna	mechanical	85000

Find all the employees with name sita and department is mechanical

$\sigma_{name= "sita" \wedge department="mechanical"}(employee)$

emp_id	name	department	salary
4	sita	mechanical	32000

Find all the employees whose department is either civil or computer

$\sigma_{department= "civil" \vee department="computer"}(employee)$

emp_id	name	department	salary
1	ramesh	civil	68000
2	krishna	computer	75000
6	nikita	computer	60000
7	anish	civil	55000

Find all the employees whose department is either civil or computer and name is not nikita

$\sigma_{((department="civil" \vee department="computer") \wedge name \neq nikita)}(employee)$

emp_id	name	department	salary
1	ramesh	civil	68000
2	krishna	computer	75000
7	anish	civil	55000

1.2 Projection operation

- ✓ The projection operation retrieves tuples for specified attributes of relation.
- ✓ It eliminates duplicate tuples in relation.
- ✓ The projection is denoted by uppercase Greek letter pi (∏).
- ✓ We need to specify attributes that we wish to appear in the result as a subscript to ∏.
- ✓ The general structure of projection is

$\prod_{A1,A2,\ldots,Ak}(r)$
where A1, A2, . .Ak are attributes of relation r.

Example:

Find name and their salary from employee relation

$\prod_{name,salary}(employee)$

output

name	salary
Ramesh	68000
krishna	75000
rita	65000
sita	32000
juna	85000
nikita	60000
anish	55000

Composition of relational operations

Relational algebra operations can be composed together into relational-algebra expression. This required for complicated query.

Example:

Find name and salary of employees of civil department

$\prod_{name,\ salary}(\sigma_{department="civil"}(employee))$

output

name	salary
ramesh	68000
anish	55000

Find name and department of employees whose salary is less than or equals to 60000

$\prod_{name, department}(\sigma_{salary \leq 60000}(employee))$

output

name	department
sita	mechanical
nikita	computer
anish	civil

1.3 Union operation

Suppose r and s are two relations, then union operation contains all tuples that appear in r, s, or both.

The union of two relations r and s are defines as

$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

For r U s to be valid, it must hold

- ✓ r,s must have same arity (same number of attributes)
- ✓ The attribute domain must be compatible (e.g. domain of ith column of r must deals with same type of domain of ith column of s)
- ✓ Duplicate rows are eliminated by this operations

Let us consider two relations

depositer

customer_name	account_no
ram	A-101
sita	A-105
hari	A-205
nishant	A-405
gita	A-505

borrower

customer_name	loan_no
Aditya	L-224
nishant	L-185
mahesh	L-150
gita	L-174
ronish	L-145

Example:

Find name of all customers who have either account or loan

$\prod_{customer\_name}$(depositor) U $\prod_{customer\_name}$(borrower)

Output

customer_name
ram
sita
hari
nishant
gita
Aditya
Mahesh
ronish

1.4 Set difference Operation

- ✓ The set difference allows us to find tuples that are in one relation but not in another relation. The expression r-s produces a relation containing those tuples in r but not in s.
- ✓ Formally, let r and s are two relations then their difference r-s define as

  $r-s= \{t \mid t \in r \text{ and } t \notin s\}$

- ✓ The set difference must be taken between compatible relations. For r-s to be valid, it must hold

  - r and s must have the same arity (same number of attributes)
  - attribute domains of r and s must be compatible

Example:

Find name of all customer of the bank who have account but not loan

$\prod_{customer\_name}$(depositor) - $\prod_{customer\_name}$(borrower)

customer_name
ram
sita
hari

1.5 Cartesian product

- ✓ It allows us to combine information from any two relations.
- ✓ The Cartesian product operation denoted by cross (X).
- ✓ Cartesian product of two relations r and s, denoted by r X s returns a relation instance whose schema contains all the fields of r (in same order as they appear in r(followed all field of s (in the same order as they appear in s).
- ✓ The result of r×s contains one tuples <r,s> (concatenation of tuples of r and s) for each pair tuples t∈r, q∈s.

Formally, r X s={<t,q>|t∈r and q∈s}

Note:

If r and s are two relation having n and m number of attributes and p and q number of records respectively, then the Cartesian product of these two relation denoted by r X s results a new relation having (n+m) numberof attributes and (p*q) number of records

Example:

Relation Employee

emp_id	name
1	anish
2	sita
3	nitesh

Relation Department

dept_id	dept_name
1	computer
2	civil
3	electrical

Employee X Department

emp_id	name	dept_id	dept_name
1	anish	1	computer
1	anish	2	civil
1	anish	3	electrical
2	nitesh	1	computer
2	nitesh	2	civil
2	nitesh	3	electrical
3	sita	1	computer
3	sita	2	civil
3	sita	3	electrical

Example 2:

Relation borrower

customer_name	loan_number
X	L01
Y	L02

Relation loan

loan_number	branch_name	amount
L01	B1	5000
L02	B2	6000

Query: Find all customer who taken loan from branch "B1".

$\prod_{customer\_name}(\sigma_{borrower.loan\_number=loan.loan\_number}(\sigma_{branch\_name="B1"}(borrower \times loan)))$

Process:

borrower×loan

customer_name	borrower.loan_number	loan.loan_number	branch_name	amount
X	L01	L01	B1	5000
X	L01	L02	B2	6000
Y	L02	L01	B1	5000
Y	L02	L02	B2	6000

$\sigma_{branch\_name="B1"}(borrower \times loan)$

customer_name	borrower.loan_number	loan.loan_number	branch_name	amount
X	L01	L01	B1	5000
Y	L02	L01	B1	5000

$\sigma_{borrower.loan\_number=loan.loan\_number}(\sigma_{branch\_name="B1"}(borrower \times loan))$

customer_name	borrower.loan_number	loan.loan_number	branch_name	amount
X	L01	L01	B1	5000

$\prod_{customer\_name}(\sigma_{borrower.loan\_number=loan.loan\_number}(\sigma_{branch\_name="B1"}(borrower \times loan)))$

customer_name
X

1.6 Rename operation

- ✓ The result of relational-algebra expression does not have a name to refer it. It is better to give name to result relation.
- ✓ The rename operator is denoted by lower case Greek letter rho (ρ).
- ✓ Rename operation in relation-algebra expressed as $\rho_x(E)$ where E is a relational algebra expression and x is name for result relation. It returns the result of expression E under the name x.
- ✓ Since a relation r is itself a relational-algebra expression thus, the rename operation can also apply to rename the relation r (i.e. to get same relation under a new name).
- ✓ Rename operation can also used to rename attributes of relation. Assume a relational algebra expression E has arity n. Then expression $\rho_{x(A1,A2,...An)}(E)$ returns the result of expression E under the name x and it renames attributes to A1,A2,..,An.

Let us consider the relation

Customer(customer_id,customer_name,customer_city)

Example1:

To find all the customer_name from this relation algebra expression can be written as

$\prod_{customer\_name}(customer)$

This result of the expression can be renamed as

$\rho_{cname}(\prod_{customer\_name}(customer))$

Example 2:

Rename the relation named customer

$\rho_{newcustomer}(customer)$

Here relation named customer can be renamed as newcustomer.

Example 3:

Relation named can also be renamed as well their attributes also

$\rho_{c(cid,cname,ccity)}$ (Customer)

Her relation name customer is renamed as c and their attributes customer_id,customer_name,customer_city are renamed as cid,cname and ccity respectively.

2. Additional relational operations

2.1 Set intersection operation

- ✓ Suppose r and s are two relations, then set intersection operation contains all tuples that are in both r and s.
- ✓ Let r and s are two relation having same arity and attributes of r and s are compatible then their intersection r∩ s define as r∩ s={t|t∈r and t∈s}

Example

Find all customer who have both loan and account

$\prod_{customer\_name}$(borrower) ∩ $\prod_{customer\_name}$ (depositor)

customer_name
nishant
gita

2.2 Division

- ✓ The division operator takes two relations and builds another relation consisting of values of an attribute of one relation that matches all the values in another relation
- ✓ Division operator r÷s or r/s can be applied if and only if, Attributes of s is proper subset of Attributes of r.
- ✓ The relation returned by division operator will have attributes = (All attributes of r – All Attributes of s)

Example 1:

K	X	Y
1	A	2
1	B	4
2	A	2
3	B	4
4	B	4
3	A	2

Relation r

X	Y
A	2
B	4

Relation s

r÷ s:

K
1
3

Example 2:

Consider the following relation

subject

subject_name	course_name
DBMS	CMP
C++	ELX
C++	CMP
OS	CMP

course

course_name
CMP
ELX

Find the name of subject taught in all course

subject÷ course

subject_name
C++

2.3 Assignment Operation

- ✓ The assignment operation provides convenient way to express complex query.
- ✓ The assignment operation denoted by ←, works like assignment in programming language.
- ✓ The evaluation of an assignment does not result any relation being displayed to the user. But the result of the expression to the right of the ← is assigned to the relation variable.
- ✓ This relation variable may used in subsequent expressions.
- ✓ With the assignment expression, a query can be written as a sequential program consisting a series of assignments followed by an expression whose value is displayed as the result of the query.

Example: Find all customer who taken loan from bank as well as has bank account.

temp1$\leftarrow \prod_{customer_name}$(borrower)

temp2$\leftarrow \prod_{customer_name}$(depositor)

result$\leftarrow$ temp1 $\cap$ temp2

2.4 Join Operation

- ✓ A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied.
- ✓ It is denoted by ⋈.
- ✓ Join operation is essentially a Cartesian product followed by a selection criterion.
- ✓ In its simplest form, the join operation is just the cross product of two relations which produce large result size but using this operation, one record from relation **R** and on record from Relation **S** can be combined together to form the result if the combination satisfies the join condition.
- ✓ The join condition can be =, , ≤, ≥, ≠

Various forms of join operation are:

1. Equi Join
2. Natural Join
3. Outer Join
    i. Left Outer Join
    ii. Right Outer Join
    iii. Full Outer Join

Before discussing various joins let us consider the following relations

Student

stu_id	sname	address
1	Anish	Kathmandu
2	Rita	Lalitpur
3	Krishna	Bhaktapur
4	Nitu	Butwal

Course

stu_id	cname	fee
1	Java	25000
3	Python	22000
4	PHP	18000
5	MERN stack	30000

1. Equi join

An equijoin is an operation that combines two relations based on the equality of values in specified attributes.

It is denoted by the symbol ⋈ and is defined as follows:

R ⋈ <condition> S

Where R and S are the relations to be joined, and <condition> represents the equality condition on the common attribute(**S**).

student ⋈ Student.stu_id = Course.stu_id course

stu_id	sname	address	cname	fee
1	Anish	Kathmandu	Java	25000
3	Krishna	Bhaktapur	Python	22000
4	Nitu	Butwal	PHP	18000

The resulting relation includes only the tuples that have matching "stu_id" values in both the "Student" and "Course" relations.

2. Natural Join

- ✓ Natural join automatically matches and combines tuples from two relations based on the common attribute(s) with the same name.
- ✓ In example given below, the common attribute is "stu_id".
- ✓ Natural join does not use any comparison operator
- ✓ The name and type of the attribute must be same.
- ✓ It eliminates duplicate attributes in the result.
- ✓ It is denoted by ⋈

Student ⋈ course

stu_id	sname	address	cname	fee
1	Anish	Kathmandu	Java	25000
3	Krishna	Bhaktapur	Python	22000
4	Nitu	Butwal	PHP	18000

The resulting relation includes only the tuples that have matching attribute values with the same name ("stu_id" in this case).

3. Outer join

- ✓ It is an extension of natural join to deal with missing values of relation.
- ✓ It is used to retrieve all records from relation, even for those tuples with no matching value in the other relation based on the join condition.
- ✓ In such cases, it returns NULL as the value for the missing attributes.

It is further classified as:

i. Left Outer Join
ii. Right Outer Join
iii. Full Outer Join

i) Left outer join (⟕)

- ✓ Left outer join returns all tuples from the left relation and the matching tuples from the right relation.
- ✓ If there is no match in the right relation, NULL values are used for the attributes of the right relation in the resulting relation

Student ⟕ course

stu_id	sname	address	cname	fee
1	Anish	Kathmandu	Java	25000
2	Rita	Lalitpur	NULL	NULL
3	Krishna	Bhaktapur	python	22000
4	Nitu	Butwal	PHP	18000

The resulting relation includes all tuples from the left relation ("Student") and the matching tuples from the right relation ("Course"). The NULL values in the "cname" and "fee" columns indicate that there is no match for the corresponding tuples in the "Course" relation.

ii) Right Outer Join(⟖)

- ✓ Right outer join returns all tuples from the right relation and the matching tuples from the left relation.
- ✓ If there is no match in the left relation, NULL values are used for the attributes of the left relation in the resulting relation.

Student ⟖ Course

stu_id	sname	address	cname	fee
1	Anish	Kathmandu	Java	25000
3	Krishna	Bhaktapur	Python	22000
4	Nitu	Butwal	PHP	18000
5	NULL	NULL	MERN stack	30000

The resulting relation includes all tuples from the right relation ("Course") and the matching tuples from the left relation ("Student"). The NULL values in the "sname" and "address" columns indicate that there is no match for the corresponding tuples in the "Student" relation.

iii) Full outer join(⟗)

- ✓ Full outer join returns all tuples from both relations.
- ✓ If there is no match for a tuple in either relation, NULL values are used for the attributes of the relation that does not have a match.

Student ⟗ Course

stu_id	sname	address	cname	fee
1	Anish	Kathmandu	Java	25000
2	Rita	Lalitpur	NULL	NULL
3	Krishna	Bhaktapur	python	22000
4	Nitu	Butwal	PHP	18000
5	NULL	NULL	MERN stack	30000

The resulting relation includes all tuples from both the left relation ("Student") and the right relation ("Course"). The NULL values indicate that there is no match for the corresponding tuples in either relation.

3. Extended Relational- Algebra Operations

3.1 Generalized Projection

Generalized projection operation allows arithmetic and string functions in the projection list.

The generalized projection has the form

$\prod_{F1,F2,\ldots Fn}(E)$

where E is any relational algebra expression. Each F1,F2, . .Fn are arithmetic expression involving constants and attributes in the schema of E.

Example 1:

Suppose a relation

credit_info(customer_name, credit_limit,credit_balance)

Find how much more each person can spend.

$\prod_{customer\_name,\ credit\_limit - credit\_balance}(credit\_info)$

Example 2:

Suppose relation

employee(employee_id,ename,salary)

Find employee and their corresponding bonus, assume that bonus for each employee is 10% of his/her salary.

$\prod_{ename,salary*1.10}(employee)$

3.2 Aggregation

- ✓ The aggregate operation permits the use of aggregate functions such as min ,average etc. on set of values.
- ✓ Aggregation function takes a collection of values and returns a single value as a result.

Some aggregate functions are

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of value

- ✓ Aggregate operation in relational algebra denoted by the symbol g (i.e. $\mathcal{G}$ is the letter G in calligraphic font)

$$_{G1, G2, \ldots, Gn}\mathcal{G}_{F1(A1), F2(A2), \ldots, Fn(An)}(E)$$

- E is any relational-algebra expression
- $G_1, G_2 \ldots, G_n$ is a list of attributes on which to group (can be empty)
- Each F_i is an aggregate function
- Each A_i is an attribute name

Let us consider the following relation named Employee

emp_id	name	department	salary
1	Ramesh	Civil	55000
2	Prizma	Computer	65000
3	Riya	IT	52000
4	Narayan	Civil	25000
5	Nimesh	computer	5000

Find the average salary of employee

$$\mathcal{G}_{avg(salary)}(\text{Employee})$$

Find the minimum salary of employee

$$\mathcal{G}_{min(salary)}(\text{Employee})$$

Find the total salary paid by employee in each department

$$_{department}\mathcal{G}_{sum(salary)}(\text{Employee})$$

department	salary
Civil	80000
Computer	70000
IT	52000

Modification of database

Insertion, deletion and updating operations are responsible for database modification

Deletion

- ✓ A delete request is expressed similarly to a query, except instead of displaying tuples to the user, the selected tuples are removed from the database.
- ✓ Can delete only whole tuples; cannot delete values on only particular attributes
- ✓ A deletion is expressed in relational algebra by:

$$r \leftarrow r - E$$

where r is a relation and E is a relational algebra query.

Let us consider the following relation

Employee(employee_id,name,department,salary)

Example 1: Delete information of all employee from civil department

Employee ← Employee- $\sigma_{department="civil"}$ (Employee)

Example 2:

Delete all records of employee with salary in the range 25000 to 60000

$Employee \leftarrow Employee - \sigma_{(salary \geq 25000) \wedge (salary \leq 60000)}$ (Employee)

Insertion

To insert data into relation we can either specify a tuple to be inserted or write a query whose result is a set of tuples to be inserted.

In relational-algebra, an insertion is express by $r \leftarrow r \cup E$
where r is a relation and E is a relational algebra expression

Example:

Insert the information of employee whose employee id is 10, named "rikesh" working in civil department and having salary 50000

Employee ← Employee $\cup$ {(10,"rikesh", "civil",50000)}

Updating

Updating allow to change a value in a tuple without changing all values in tuple. In relational algebra, updating express by
$r \leftarrow \prod_{F1,F2,\ldots,Fn}(r)$

where each F_i is either

- ✓ the *i*th attribute of *r*, if the *i*th attribute is not updated, or,
- ✓ expression involving only constant and attributes of r, if the attribute is to be updated. It gives the new value for the attribute.

Example:

Increase salary of all employees by 5 %

$Employee \leftarrow \prod_{employee\_id,name,deparment,salary*1.05}(Employee)$

Increase the salary of employee by 20% if salary is less than 50000 and increase salary of remaining employees by 10%

$$Employee \leftarrow \prod_{employee\_id,\ name,,department,\ salary\ *\ 1.2}(\sigma_{\ salary\ <50000}\ (Employee)) \cup \prod_{employee\_id,name,department,salary*1.1}(\sigma_{salary \geq 50000}\ (Employee))$$

Increase salary of employees of civil department by 15%

$Employee \leftarrow \prod_{employee\_id,name,department,salary*1.15}(\sigma_{department=\ "civil}\ (Employee))$

$U\ (Employee - \sigma_{department=\ "civil}\ (Employee))$

Update an employee so that ram now shifted to computer department

$Employee \leftarrow \prod_{employee\_id,name,"computer",salary}(\sigma_{name=\ "ram"}(Employee))\ U$

$(Employee - \sigma_{name="ram"}(Employee))$

Note:

update operation will be in another form as well

$r \leftarrow \prod_{F1,F2,F3,\ldots Fn}(\ \sigma_p\ (r)\ )\ U\ (r - \sigma_p(r)\ )$

Database schema

Database schema is a logical design of a database and the database instance is a snapshot of the data in the database at a given instant in time.

The concept of a relation corresponds to the programming language notation of a variable While the concept of a relation schema corresponds to the programming language notation of type definition. In general a relation schema consists of list of attributes and their corresponding domains.

The concept of a **relation instance** corresponds to the programming –language notation of a value of a variable. The value of a given variable may change with time; similarly the contents of a relation instance may change with time as the relation is updated. In contrast, the schema of a relation does not generally change.

For example, the relation schema for relation customer is express as

Customer (customer_id, customer_name, customer_city)

We may also specify domains of attributes as

Customer (customer_id: integer, customer_name: string,customer_city:string)

The below figure shows the instance of relation

customer_id	customer_name	customer_city
1	ronit	Kathmandu
2	sita	Pokhara
3	nitu	Lalitpur

Let us consider university database example

Each course in a university may be offered multiple times, across different semesters, or even within a semester. We need a relation to describe each individual offering, or section, of the class.

The schema is section (course id, sec id, semester, year, building, room number, time slot id)

We need a relation to describe the association between instructors and the class sections that they teach. The relation schema to describe this association is

teaches (ID, course id, sec id, semester, year)

As we can imagine, there are many more relations maintained in a real university database.

Now, all the database schemas for university database can be listed as

instructor(id,name,dept_name,salary)
course(course_id,title,dept_name,credits)
department (dept_name, building, budget)
section (course_id, sec_id, semester, year, building, room_number, time_slot_id)
teaches (ID, course_id, sec_id, semester, year)
student (ID, name, dept_name, tot_cred)
advisor (s_id, i_id)
prereq(course_id,prereq_id)
takes (ID, course_id, sec_id, semester, year, grade)
classroom (building, room_number, capacity)
time_slot (time_slot_id, day, start_time, end_time)

Schema diagrams

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams. Figure given below shows the schema diagram for university organization. Each relation appears as a box, with the relation name at the top in blue, and the attributes listed inside the box. Primary key attributes are shown underlined. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

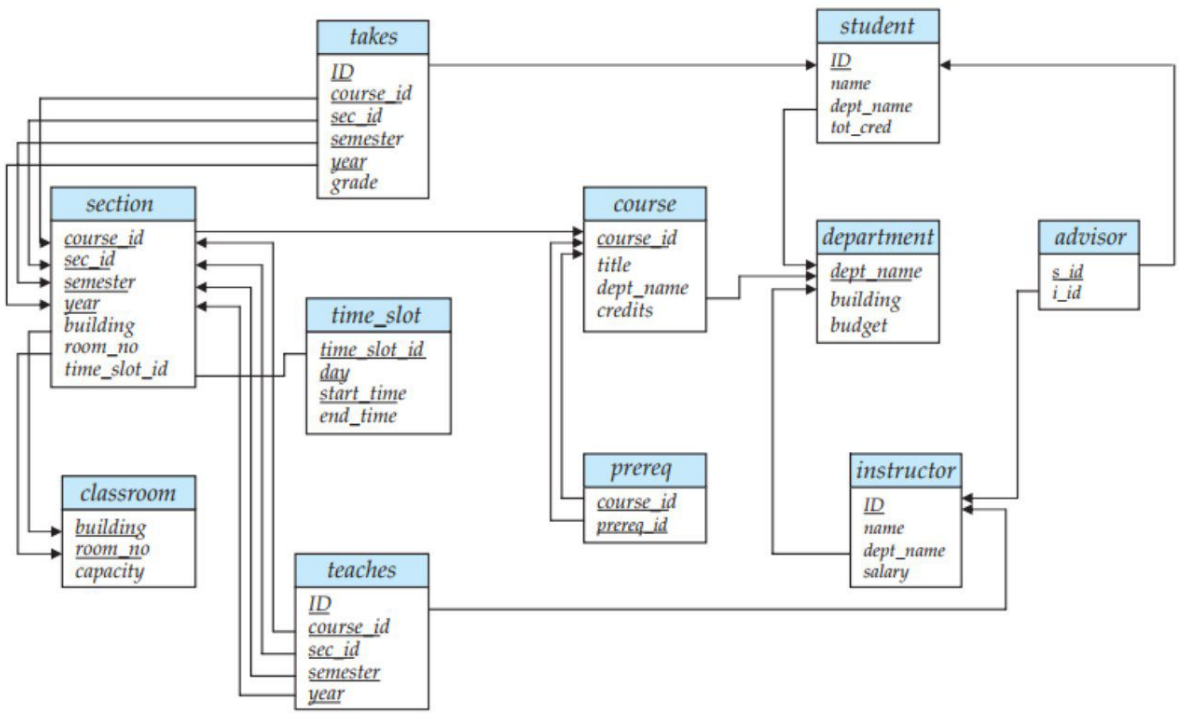


Figure:Schema diagram for the university database.

Assignment

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course: **[PU:2017 spring]**

STUDENT(SSN,Name,Major,Bdate)
COURSE(Course#,Cname,Dept)
ENROLL(SSN,Course#,Quarter,Grade)
BOOK_ADOPTION(Course#,Quarter,Book_ISBN)
TEXT(Book_ISBN,Book_Title,Publisher,Author)

Draw a relational schema diagram specifying the foreign keys for this schema.

Data Dictionary storage

A relational database system needs to maintain data about the relations, such as the schema of the relations. In general, such "data about data" is referred to as metadata. Relational schemas and other metadata about relations are stored in a structure called the **data dictionary** or system catalog.

Among the types of information that the system must store are these:

- ✓ Names of the relations.
- ✓ Names of the attributes of each relation.
- ✓ Domains and lengths of attributes.
- ✓ Names of views defined on the database, and definitions of those views.
- ✓ Integrity constraints (for example, key constraints).

In addition, many systems keep the following data on users of the system:

- ✓ Names of authorized users.
- ✓ Authorization and accounting information about users.
- ✓ Passwords or other information used to authenticate users.

Further, the database may store statistical and descriptive data about the relations, such as:

- ✓ Number of tuples in each relation.
- ✓ Method of storage for each relation

The data dictionary may also note the storage organization (sequential, hash, or heap) of relations, and the location where each relation is stored:

- ✓ If relations are stored in operating system files, the dictionary would note the names of the file (or files) containing each relation.
- ✓ If the database stores all relations in a single file, the dictionary may note the blocks containing records of each relation in a data structure such as a linked list.

Assignment:

How relational algebra is different from relational calculus? Define Tuple Relational Calculus and Domain Relational Calculus.

(Refer: Text Book "Database system concept" by Abraham Silberschatz)

RELATIONAL ALGREBRA OLD QUESTIONS SOLUTION

1. Consider the following relational database, where primary keys are underlined.

   employee(person_name,street,city)

   works(person_name,company_name,salary)

   company(company_name,city)

   manages(person_name,manager_name)

   Given an expression in the relational algebra to express each of the following queries.

   i)Find the name of all employees who work for first bank corporation.

   $\prod_{person\_name} (\sigma_{company\_name="first\ bank\ corporation"}(works))$

   ii)Find the name and cities of residence of all employees who work for first bank corporation.

   $\prod_{person\_name,city} (\sigma_{company\_name="first\ bank\ corporation"}(employee \bowtie works))$

   iii)Find the name, street address and city of residence of all employees who work for first bank corporation and earn more than $10000 per annum.

   $\prod_{person\_name,street,city} (\sigma_{company\_name="first\ bank\ corporation" \wedge salary>10000}(employee \bowtie works))$

   (Here we assume attribute named salary represent Annual Salary)

   iv)Find the name of all employees in this database who lives in same city as the company for which they work.

   $\prod_{person\_name}(employee \bowtie works \bowtie company)$

   v)Find the name of all employees who live in the same city and on the same street as do their managers.

   $\prod_{person\_name}((employee \bowtie manages) \bowtie_{(manages.manager\_name=employee2.person\_name \wedge employee.street=employee2.street \wedge employee.city=employee2.city)} (\rho_{employee2}(employee)))$

   vi)Find the name of all employees in this database who do not work for first bank corporation

   $\prod_{person\_name} (\sigma_{company\_name \neq "first\ bank\ corporation"}(works))$

   vii) Assume that companies may located in several cities. Find all companies located in every city in which small bank corporation is located.

   $\prod_{company\_name,city}(company) \div \prod_{city} (\sigma_{company\_name ="small\ bank\ corporation"}(company))$

2. Consider the following relational database.

Students(RollNo, StudentName, Address, Semester)

Teachers(TeacherID, TeacherName,CourseID,Salary,Department)

Courses(CourseID,RollNo,CourseTitle,Semester)

Write relational Algebra Expressions for the following requests.

i) Find the name of students of 4th semester and studying "Operating System"

$\prod_{StudentName}(\sigma_{Semester= "4th" \wedge CourseTitle="Operating System"}(Students \bowtie Courses))$

ii) Find the name of teacher who teaches subject "DBMS" to "Arten Khadka"

$\prod_{TeacherName}(\sigma_{CourseTitle="DBMS" \wedge StudentName="Arten Khadka"}((Students \bowtie Courses) \bowtie Teachers))$

iii) Delete Record of 2nd semester students of Account Department

$Students \leftarrow \prod_{RollNo, StudentName, Address, Semester}(Students)$

$- \prod_{RollNo, StudentName, Address, Semester}(\sigma_{Department="account" \wedge Semester=2nd}((Students \bowtie Courses) \bowtie Teachers))$

iv) Increase salary of "Bhaskar Bhatta" by 6%

$Teachers \leftarrow \prod_{TeacherID, TeacherName,CourseID,Salary*1.06,Department}(\sigma_{TeacherName= "Bhaskar Bhatta"}(Teachers)) \cup (Teachers - \sigma_{TeacherName= "Bhaskar Bhatta"}(Teachers))$

3. Consider the following schema

SUPPLIER(Sid,S_name,S_addr)

PARTS(Pid,p_name,color)

CATALOG(sid,pid,cost)

Now answer the following queries in Relational Algebra.

i) Find the name of all supplier who supply yellow parts

$\prod_{S\_name}(\sigma_{color='yellow'}(SUPPLIER \bowtie CATALOG \bowtie PARTS))$

ii) Find the name of suppliers who supply Both blue and black parts.

$\prod_{S\_name,color}(SUPPLIER \bowtie (CATALOG \bowtie PARTS)) \div \prod_{color}(\sigma_{color='blue' \vee color='black'}(PARTS))$

iii) Find the name of suppliers who supply all parts.

$\prod_{S\_name,Pid}(SUPPLIER \bowtie CATALOG) \div \prod_{Pid}(PARTS)$

4. **Consider the relational database** **[PU:2010 spring]**

Employee(Empname,street,city)

Works(Empname,post,cmpname,salary)

Company(cmpname,location)

Write relational algebraic expression for

i) An employee named John is promoted from Assistant manager to manager.

$Works \leftarrow \prod_{Empname,"Manager",cmpname,salary}(\sigma_{Empname="John" \wedge post="Assistant\ Manager"}(Works))$
$\cup\ (Works - \sigma_{Empname="John" \wedge post="Assistant\ Manager"}(Works))$

ii) Update the relation company so that all companies located in Biratnagar is shifted to Kathmandu.

$Company \leftarrow \prod_{cmpname,"Kathmandu"}(\sigma_{location="Biratnagar"}(Company))$
$\cup\ (Company - \sigma_{location="Biratnagar"}(Company))$

iii) Remove all the records of employee who lives in pokhara

$Employee \leftarrow Employee - \sigma_{city="Pokhara"}(Employee)$

5. Consider the following schema

customer(cus_id,cus_name,cus_phno)
employee(cus_id,emp_id,emp_name,emp_add)
works(branch_id,salary,cus_id)
branch(branch_id,branch_name)
Write relational algebra notations for the following queries for the given schema.
[PU:2011 spring]

i) select name of all employees.

$\prod_{emp\_name}(employee)$

ii) Give salary rise to 5% to all the employee

$works \leftarrow \prod_{branch\_id,salary*1.05,cust\_id}(works)$

iii) List all branch names

$\prod_{branch\_name}(branch)$

iv) select name of all employees working for "manang" branch

$\prod_{emp\_name}(\sigma_{branch\_name="manang"}(branch \bowtie (works \bowtie employee)))$

v) Delete any record from work table

$works \leftarrow works - \sigma_{branch\_id=101}(works)$

This operation can be customized according to requirement.

vi) List the name and phno of all customers

$\prod_{cust\_name,cus\_phno}(customer)$

vii) select name of all employees deal with customer having id "201"

$\prod_{emp\_name}(\sigma_{cus\_id=201}(employee))$

viii) Delete all records from works table whose salary is less than 10000

$works \leftarrow works - \sigma_{salary<10000}(works)$

ix) Delete all records from works table

$works \leftarrow works - \prod_{branch\_id,salary,cus\_id}(works)$

6. By using the following schemas write relational algebraic expression and SQL statements. (underlined attributes represent primary key attributes)

EMPLOYEE(EMPNO,NAME,ADDRESS)
PROJECT(PNO,PNAME)
WORKON(EMPNO,PNO)
PART(PARTNO,PARTNAME,QTY_ON_HAND)
USE(EMP_NO,PNO,PARTNO,NUMBER) **[PU:2011 fall]**

i)Listing all employees details who are not working yet

$\prod_{EMPNO,NAME,ADDRESS}(Employee) - \prod_{EMPNO,NAME,ADDRESS}(Employee \bowtie WORKON)$

ii) Listing Part Name and Quantity on hand those were used in DBMS project

$\prod_{PARTNAME,QTY\_ON\_HAND}(PART \bowtie (USE \bowtie \sigma_{PNAME="DBMS"}(PROJECT)))$

OR

$\prod_{PARTNAME,QTY\_ON\_HAND}(\sigma_{PNAME="DBMS"}(PART \bowtie (USE \bowtie PROJECT)))$

iii) List the name of projects that are used by employee from Kathmandu

$\prod_{PNAME}(\sigma_{ADDRESS="Kathmandu"}((EMPLOYEE \bowtie WORKON) \bowtie PROJECT))$

7. Consider the following relations for order processing database application in a company.

CUSTOMER(Cust#,Cname,City)

ORDER(Order#,Odate,Cust#,ord_Amt)

ORDER_ITEM(Order#,Item#,Qty)

ITEM(Item#,Unit_price)

SHIPMENT(Order#,Warehouse#,Ship_date)

WAREHOUSE(Warehouse#,City)

Answer the following queries in relational algebra. **[PU:2012 spring]**

i) List the order# and ship_date for all orders shipped from Warehouse number "W2".

$\prod_{Order\#,Ship\_date}(\sigma_{warehouse\#="w2"}(SHIPMENT))$

ii) List the warehouse information for which the customer named 'JOSE Copez' was supplied his orders.

$\prod_{warehouse\#,city}(\sigma_{Cname="JOSE\ Copez"}(CUSTOMER \bowtie (ORDER \bowtie (SHIPMENT \bowtie WAREHOUSE))))$

iii) List the orders that were not shipped within 30 days of ordering.

$TIMELY\_SHIPPED \leftarrow \sigma_{Ship\_date \le Odate+30}(ORDER \bowtie SHIPMENT)$

$RESULT \leftarrow \prod_{Order\#}(ORDER) - \prod_{Order\#}(TIMELY\_SHIPPED)$

iv) List the order# for orders that were shipped from all warehouses in the network.

$\prod_{Order\#,\ Warehouse\#}(Shipment) \div \prod_{Warehouse\#}(\sigma_{City = "New\ York"}(Warehouse))$

8. consider the following database: [**PU:2012 fall]**

Student(sid,name,age)
Has(sid,cid)
College(cid,cname)
Write relational algebra expression to perform the following.

i) find the average age of student.

$\mathcal{G}_{avg(age)}$ (Student)

ii) Display the name of student who studies in "QWERT" college.

$\prod_{name}(\sigma_{cname="QWERT"} (Student \bowtie (Has \bowtie College)))$

iii) Insert a new student.

Student ←Student U {104,"Roshan",29}

iv) Delete record of "ASDFG" college from college relation

College←College-($\sigma_{cname="ASDFG"}$ (College))

v) Display name of students whose name begin from 'S.'

$\prod_{name}(\sigma_{name\ LIKE\ 'S\%'}$ (Student)

9. Consider the following relation R and S **[PU:2013 spring]**

R

Sid	SName	Marks(%)
S001	Hari	85
S002	Sita	78
S003	Bidur	85
S005	Vinod	68

S

Sid	SName	Marks(%)
S004	Sarita	76
S003	Bidur	85
S006	Shyam	75
S005	Vinod	68

i) Show the id and name of those students whose marks is less than 80 from relation schema R.(Write only relational schema)

$\prod_{Sid,SName}(\sigma_{Marks<80} (R))$

ii) Write the results.

RUS

Sid	SName	Marks(%)
S001	Hari	85
S002	Sita	78
S003	Bidur	85
S005	Vinod	68
S004	Sarita	76
S006	Shyam	75

R-S

Sid	SName	Marks(%)
S001	Hari	85
S002	Sita	78

$\Pi_{SName}(\sigma_{Marks=85}(S))$

SName
Bidur

10. Consider the relational database of figure below, where primary keys are underlined. Given an expression in the relational algebra to express each of the following queries. [PU:2014 spring]

Employee(person_name,street,city)
Works(person_name,bank_name,salary)
Bank(bank_name,city)
Manages(person_name,manager_name)

i) Find the total salary sum of all the banks.

$\mathcal{G}_{sum(salary)}$ (works)

ii) Modify the database so that Ram now lives in Kathmandu.

$Employee \leftarrow \Pi_{person\_name,street,"Kathmandu"}(\sigma_{person\_name="Ram"}(Employee))$
$U\ (Employee - \sigma_{person\_name="Ram"}(Employee))$

iii) Find the name, street address and cities of residence of all employees who work for Nepal world Bank corporation and earn more than $10,000 per annum.

$\prod_{person\_name,street,city}(\sigma_{bank\_name="Nepal\ World\ Bank\ Corporation"\wedge salary>10000}(Employee \bowtie Works))$

Here, we assume attribute named salary represent annual salary.

iv) Delete all tuples in work relation for employee of small bank corporation.

$Works \leftarrow Works - \sigma_{bank\_name="Small\ Bank\ Corporation"}(Works))$

11. Consider the following relational database of figure below, where the primary keys are underlined. Give an expression in the relational algebra to express the following queries

employee(person_name,street,city)

works(person_name,bank_name,salary)

company(bank_name,city)

manages(person_name,manager_name)

Given an expression in the relational algebra to express each of the following queries.

i) Find the name of all employees in this database who lives in same city as the company for which they work.

$\prod_{person\_name}(employee \bowtie works \bowtie company)$

ii) Give all employees of First Bank Corporation a 10 percent salary rise

$Works \leftarrow \prod_{person\_name,bank\_name,salary*1.1}(\sigma_{bank\_name="First\ Bank\ corporation"}(Works))$
$\cup\ (Works - \sigma_{bank\_name="First\ Bank\ Corporation"}(Works))$

iii) Modify the database so that Hari now lives in Biratnagar

$Employee \leftarrow \prod_{person\_name,street,"Biratnagar"}(\sigma_{person\_name="Hari"}(Employee))$
$\cup\ (Employee - \sigma_{person\_name="Hari"}(Employee))$

iv) Delete all tuples in work relation for employee of First Bank Corporation.

$Works \leftarrow Works - (\sigma_{bank\_name="First\ Bank\ Corporation"}(Works))$

12. Consider the following relational database of figure below, Where primary keys are underlined. Given an expression in the relational algebra to express each of the following queries. [PU:2015 fall]

employee(person_name,street,city)
works(person_name,bank_name,salary)
bank(bank_name,city)
manages(person_name,manager_name)

i) Find the name of all employees who work for Nepal Rastra Bank and Salary greater than $10000.

$\Pi_{person\_name}(\sigma_{bank\_name="Nepal\ Rastra\ Bank" \wedge salary > 10000}(employee \bowtie works))$

ii) Find the name and cities of residence of all employees who work for Nepal Rastra Bank

$\Pi_{person\_name,city}(\sigma_{bank\_name = "Nepal\ Rastra\ Bank"}(employee \bowtie works))$

iii) Find name, street address, and cities of residence of all employees who work for Nepal Rastra Bank Corporation and earn more than $10000 per annum.

$\Pi_{person\_name,street,city}(\sigma_{bank\_name = "Nepal\ Rastra\ Bank\ corporation" \wedge salary > 10000*12}(employee \bowtie works))$

(Here we assume that attributed named salary represents monthly salary)

iv) Delete all tuples in work relation for employee of Nepal Rastra Bank

$works \leftarrow works - \sigma_{bank\_name = "Nepal\ Rastra\ Bank"}(works))$

13. Consider the student registration database comprising of the schema below. **[PU:2016 fall]**

Student(CRN,Name,Gender,Address,Telephone)
Course(CourseID,CourseName,Hour,TeacherID)
Teacher(TeacherID,TeacherName,Office)
Registration(CRN,CourseID,Date)

i) Count the number of student registered subject in year 2015 gender wise.

$_{Gender}\mathcal{G}_{count(CRN)}(\sigma_{EXTRACT(YEAR\ FROM\ Date)=2015}(Student))$

ii) Show student details taught by teacher Ronit Shreshta.

$\prod_{CRN,Name,Gender,Address,Telephone}(\sigma_{TeacherName="Ronit\ Shreshta)}(((Student \bowtie Registration) \bowtie course) \bowtie Teacher))$

iii) Delete student information taught by teacher N.Mathema

$Student \leftarrow Student - \prod_{CRN,Name,Gender,Address,Telephone}(\sigma_{TeacherName="N.Mathema"}(((Student \bowtie Registration) \bowtie course) \bowtie Teacher))$

14. Consider the following schema of a relational database.
Branch(branch_name, branch_city, assests)
Account(account_number,branch_name,balance)
Customer(customer_id,customer_name,customer_street,customer_city)
Depositor(customer_id,account_number)
Loan(loan_number,branch_name,amount)
Borrower(customer_id,loan_number)

Write relational algebra for the following queries: **[PU:2016 spring]**

i) Find all customer either account or loan

$\prod_{customer\_name}(Customer \bowtie Depostior) \cup \prod_{customer\_name}(Customer \bowtie Borrower)$

ii) List the name and city of customer who have their account at the branch location "Butwal"

$\prod_{customer\_name,customer\_city}(\sigma_{branch\_city="Butwal"}(((Customer \bowtie Depositor) \bowtie Account) \bowtie Branch))$

iii) Delete all account in the branch "B1"

$Account \leftarrow Account - \sigma_{branch\text{-}name = "B1"}(Account)$

iv) Increase balance by 5% to all branches

$Account \leftarrow \prod_{account\_number,\ branch\_name,\ balance * 1.05}(Account)$

15. Consider the following schema: **[PU:2017 fall]**

employee(person_name,street,city)

works(person_name,company_name,salary)

company(company_name,city)

manages(person_name,manager_name)

Given an expression in relational algebra to express each of the following queries.

i) Find the name of all employees who earn more than their managers.

$\prod_{person\_name}$ ((works ⋈ manages) $\bowtie_{manages.manager\_name=e2.person\_name \wedge works.salary>e2.salary}$ (ρ_{e2} (works)))

In SQL this will works

```
SELECT works.person_name
FROM works NATURAL JOIN manages
JOIN works AS e2 ON manages.manager_name = e2.person_name
AND works.salary > e2.salary;
```

ii) Find the name of all employees who live in the same city and on the same street as their managers.

$\prod_{person\_name}$ ((employee ⋈ manages) $\bowtie_{(manages.manager\_name=employee2.person\_name \wedge employee.street=employee2.street \wedge employee.city=employee2.city)}$ ($\rho_{employee2}$ (employee)))

iii) Find the name of all employees with database that do not work for "NBL company".

$\prod_{person\_name}$($\sigma_{company\_name\neq"NBL\ company"}$ (works))

iv) Find the name of all employees in the database who earn more than top earner at "NBL company in the database".

topearner← $\mathcal{G}_{max(salary)}$ ($\sigma_{company\_name=" NBL\ company"}$(works))
result← $\prod_{personname}$($\sigma_{salary>topearner}$ (works))

16. Consider the following relational schema **[PU:2018 Fall]**

Department(**DepartmentID**,DepartmentName)
Designation(**DesignationID,**DesignationName,Salary)
Employee(**EmpID**,EmpName,Gender,DesignationID,DepartmentID)
Allowance(**AllowanceID**,AllowanceName)
Allowance_Details(DetailID,EmpID,AllowanceID,Amount)

Write the relational algebraic expression for the following task:

i) Find the number of employees department-wise.

$_{\text{DepartmentName}}\mathcal{G}_{\text{count(eid)}}$ (Employee⋈ Department)

ii) List the employee details whose salary is above 50000.

$\prod_{\text{EmpID,EmpName,Gender,DesignationID,DepartmentID}}(\sigma_{\text{(Salary+amount)>50000}}((\text{Employee}\bowtie \text{Designation}) \bowtie \text{Allowance}))$

iii) List the employee those who are getting house allowance.

$\prod_{\text{EmpName}}(\sigma_{\text{AllowanceName="houseallowance"}}((\text{Employee}\bowtie \text{Allowance\_Details}) \bowtie \text{Allowance}))$

17. Consider the relational database model
Users(<u>uid</u>,cname,city)
Items(<u>itemid</u>,itemname,city,quantity,price)
Manager(<u>mid</u>,aname,city)
Query(<u>queryno</u>,uid,mid,itemid,query_details,hitratio)
[PU:2018 spring]
Write the relational algebraic expressions for the following tasks.

i) Find all (queryno,uid) pairs for query with a hit ratio value greater than 500.

$\prod_{\text{queryno,uid}}(\sigma_{\text{hitratio>500}}(\text{Query}))$

ii) Find all item names of items in Pokhara ordered with query details as Pokhara details.

$\prod_{\text{itemname}}(\sigma_{\text{querydetails="Pokhara details"^city="Pokhara"}}(\text{Items}\bowtie \text{Query}))$

iii) Find item ids of items ordered through manager 35 but not through manager 27.

$\prod_{\text{itemid}}(\sigma_{\text{mid=35}}(\text{Query})) - \prod_{\text{itemid}}(\sigma_{\text{mid=27}}(\text{Query}))$

18. Using the following schema represent the following queries using Relational algebra.

PROJECT(Projectnum,ProjectName,ProjectType,ProjectManager)
EMPLOYEE(Empnum,Empname)
ASSIGNED_TO(Projectnum,Empnum)

[PU:2019 spring]

i) Find employee details working on project name starts with 'L'.

$\prod_{Empnum,Empname}(\sigma_{ProjectName\ Like\ 'L\%'}(\ (EMPLOYEE \bowtie ASSIGNED\_TO) \bowtie PROJECT)\ )$

ii) List all the employee details who are working under project manager "Roshan".

$\prod_{Empnum,Empname}(\sigma_{ProjectManager="Rohan"}((EMPLOYEE \bowtie ASSIGNED\_TO) \bowtie PROJECT)\ )$

iii) List the employes who are still not assigned with any project.

$\prod_{Empnum,Empname}(EMPLOYEE) - \prod_{Empnum,Empname}(EMPLOYEE \bowtie ASSIGNED\_TO)$

iv) List the employees who are working in more than one project.

$temp \leftarrow {}_{Empname}\mathcal{G}_{count\ (Projectnum)}(Employee \bowtie ASSIGNED\_TO)$

$result \leftarrow \prod_{Empname}((\sigma_{Projectnum>1}(temp))$

19. Write relational algebra for the following schemas. (Underlined indicates Primary) [PU:2020 spring]

Employee(**Emp_No**,Name,Address)
Project(**PNO**,Pname)
Workon(**Emp_No**,PNo)
Part(**Partno**,Part_name,Qty_on_hand)
Use(**Emp_No**,**PNO**,**Partno**,Number)

i)Listing all employees details who are not working yet.

$\prod_{Emp\_No,Name,Address}$ (Employee) - $\prod_{Emp\_No,Name,Address}$ (Employee⋈Workon)

ii) Listing Part Name and Quantity on hand those were used in DBMS project

$\prod_{Part\_name,Qty\_On\_hand}$ (Part⋈ (Use⋈$\sigma_{Pname="DBMS"}$(Project)))

iii) List the name of projects that are used by employee from London

$\prod_{Pname}$($\sigma_{Address="London"}$((Employee⋈Workon)⋈ Project))

iv) Modify the database so that Jones now live in USA.

Employee← $\prod_{Emp\_No,Name,"USA"}$($\sigma_{Name="Jones"}$(Employee)) U (Employee-$\sigma_{Name="Jones"}$(Employee))

v) Update address of an employee 'Japan' to 'USA'

Employee← $\prod_{Emp\_No,Name,"USA"}$($\sigma_{Address="Japan"}$(Employee)) U (Employee-$\sigma_{Address="Japan"}$(Employee))

20. Consider the following schemas: **[PU:2021 spring]**

Sailors(sid,sname,rating,age)

Boats(bid,bname,color)

Reserves(sid,bid,day)

Write relational algebra expression for the following queries:

i) Find record of sailors who have reserved boat number 103(bid=103)

$\prod_{sid,sname,rating,age}$ (Sailors⋈ (Reserves⋈($\sigma_{bid=103}$(Boat)))

OR

$\prod_{sid,sname,rating,age}$ ($\sigma_{bid=103}$ (Sailors⋈ (Reserves ⋈ Boat)))

ii) Update the color of the boat, where bid is 104,into green.

Boat→$\prod_{bid,bname,"green"}$ ($\sigma_{bid=104}$(Boat)) U (Boat-$\sigma_{bid=104}$(Boat))

iii) Find the name of sailors who have reserved a red or green boat.

$\prod_{sname}$ (Sailors⋈ (Reserves⋈($\sigma_{color="red" \vee color="green"}$(Boat)))

iv) Find the name of sailors who have reserved boat number 103 on day 5.

$\prod_{sname}$ ($\sigma_{bid=103 \wedge day=5}$ (Sailors⋈ Reserves))

v) Find the name of sailors whose name is not 'Ram'.

$\prod_{sname}$ ($\sigma_{sname \neq "ram"}$(Sailors))

vi) Find the name of all boats.

$\prod_{bname}$ (Boats)

21. Suppose we have the following relation. **[PU:2022 fall]**

Employee(person_name,street,city)
Works(person_name,company_name,salary)
Company(company_name,city)

Write relational algebra for the following queries.

i) Find the name of all employees who live in 'Butwal' and whose salary is less than Rs.50,000

$\prod_{person\_name}$($\sigma_{city="Butwal" \wedge salary<50000}$ (Employee ⋈ Works))

ii) Find the name of all employees who work for "Nepal Bank Limited".

$\prod_{person\_name}$ ($\sigma_{company\_name="Nepal\ Bank\ Limited"}$(works))

iii) Find the name and cities of residence of all employees who work for "Global Bank"

$\prod_{person\_name,city}$ ($\sigma_{company\_name="Global\ Bank"}$(employee⋈works))

iv) Update the salary of all employees by 10%

Works←$\prod_{person\_name,company\_name,salary*1.1}$(Works)

22. Suppose we have the following relation
Employee(person_name,street,city)
Works(person_name,company_name,salary)
Company(company_name,city)
Write relational algebraic expressions for the following queries:

i)List the name and city of employee who work in "Pokhara" and have salary greater than Rs.50,000.

$\prod_{Employee.person\_name,Employee.city}$ ($\sigma_{Company.city="Pokhara"\wedge salary>50000}$ (Employee $\bowtie_{Employee.person\_name=works.person\_name}$ (works$\bowtie$company)))

ii)Find the names of all employees who work for "ABC bank".

$\prod_{person\_name}$ ($\sigma_{company\_name="ABC\ bank"}$(works))

iii)Delete all employee who come from "Chitwan".

Employee$\leftarrow$Employee-$\sigma_{city="Chitwan"}$(Employee)

iv)Increase salary of all employee by 15%

Works$\leftarrow\prod_{person_name,company_name,salary*1.15}$(Works)