Chapter 2 ER and Relational Models

- E-R Model
- Components of ER
- Types of Key
- Introduction to Relational Model
- Reducing ER diagrams to Relational Schema
- Structure of Relational Databases, Database Schema, Schema Diagrams
- Relational Algebra

E-R Model

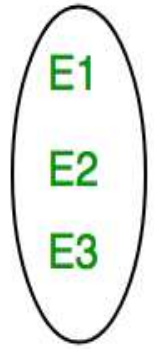
- An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).
- An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.
- An ER diagram shows the relationship among entity sets.
- An entity set is a group of similar entities and these entities can have attributes.

E-R Model

1. **Entity, Entity Type, Entity Set :**

- An Entity may be an object with a physical existence - a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.
- An Entity is an object of Entity Type and set of all entities is called as entity set.
- Example, E1 is an entity having Entity Type Student and set of all students is called Entity Set.
- In E-R diagram, Entity is represented by a rectangle.

Entity Type



Entity Set

E-R Model

2. **Attribute(s):**

- Attributes are the properties which define the entity type.
- For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student.
- In ER diagram, attribute is represented by an oval.

E-R Model

Types of Attribute:

a. Key Attribute –

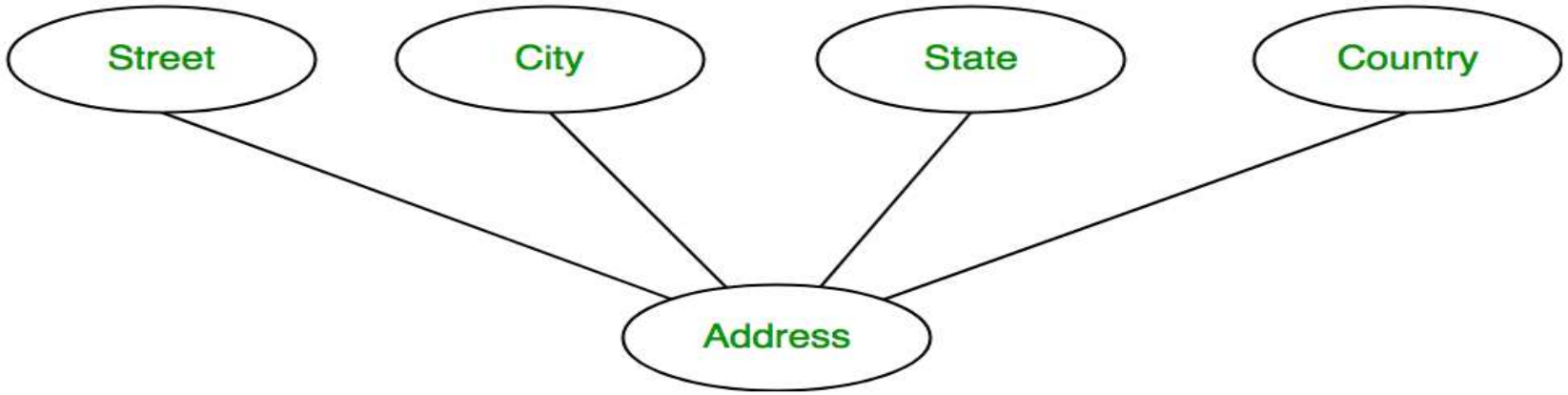
- The attribute which uniquely identifies each entity in the entity set is called key attribute.
- For example, Roll_No will be unique for each student.
- In ER diagram, key attribute is represented by an oval with underlying lines.

E-R Model

Types of Attribute:

b. Composite Attribute –

- An attribute composed of many other attribute is called as composite attribute.
- For example, Address attribute of student Entity type consists of Street, City, State, and Country.
- In ER diagram, composite attribute is represented by an oval comprising of ovals.



E-R Model

Types of Attribute:

c. Multivalued Attribute –

- An attribute consisting more than one value for a given entity.
- For example, Phone_No (can be more than one for a given student).
- In ER diagram, multivalued attribute is represented by double oval.

E-R Model

Types of Attribute:

d. Derived Attribute –

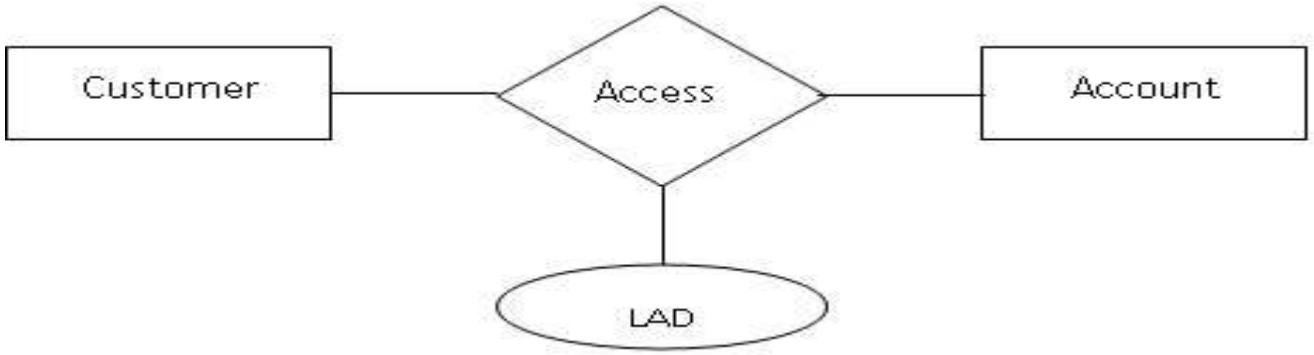
- An attribute which can be derived from other attributes of the entity type is known as derived attribute.
- Example, Age (can be derived from DOB).
- In ER diagram, derived attribute is represented by dashed oval.

E-R Model

Types of attribute

e. Descriptive attribute

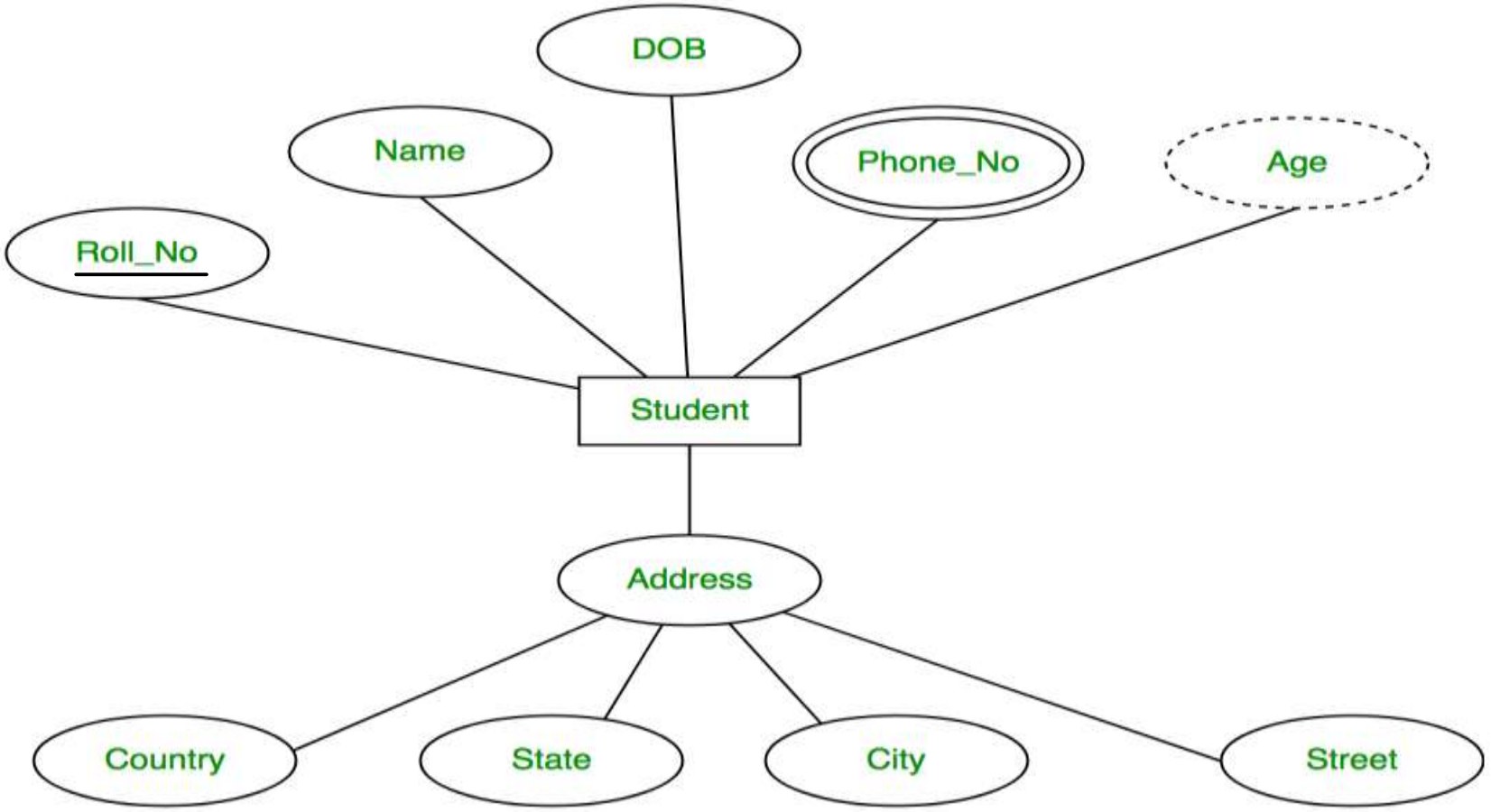
- The attribute(s) used for describing the relationship is called descriptive attributes, also referred as relationship attributes.
- They are actually used for storing information about the relationship.
- A relationship can have zero or more attributes.



-LAD (Last Accessed Date)

E-R Model

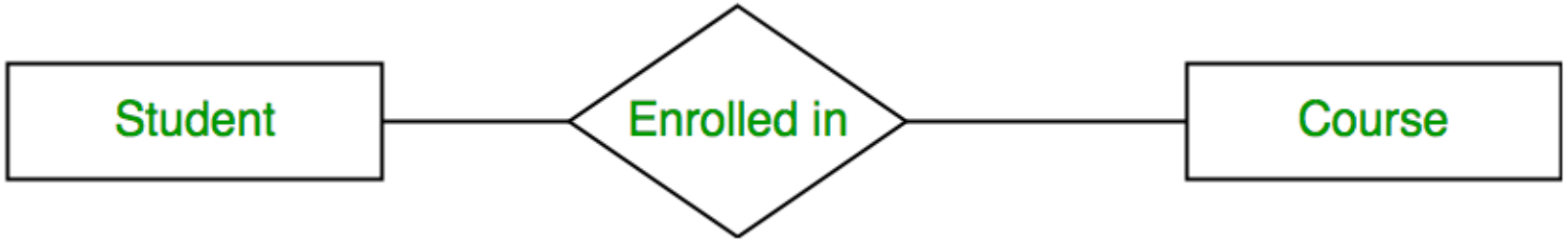
The complete entity type Student with its attributes can be represented as:



E-R Model

3. Relationship Type and Relationship Set:

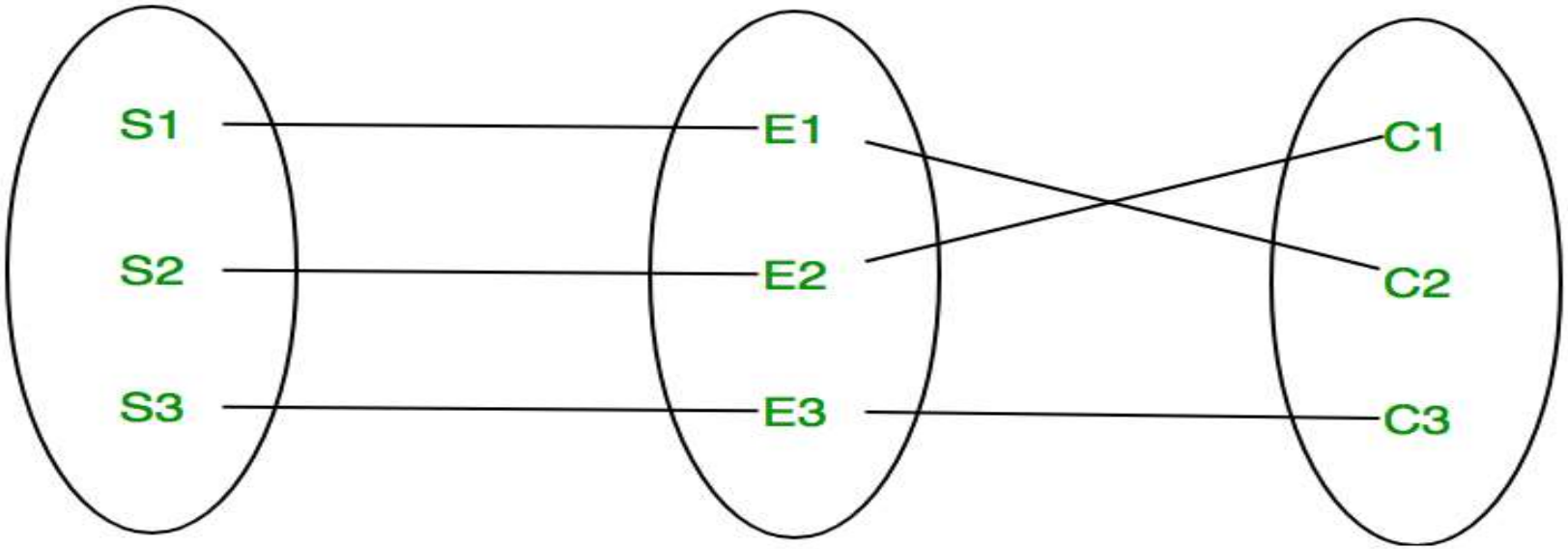
- A relationship type represents the association between entity types.
- For example,‘Enrolled in’ is a relationship type that exists between entity type Student and Course.
- In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.



E-R Model

3. Relationship Type and Relationship Set:

- A set of relationships of same type is known as relationship set.
- The following relationship set depicts S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3.



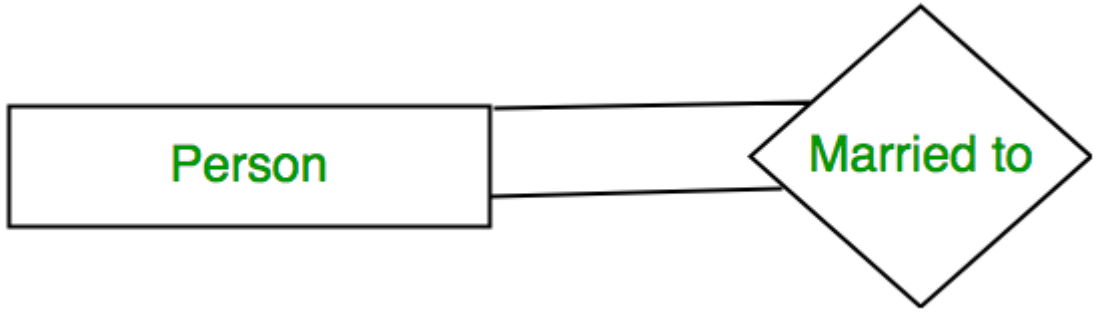
E-R Model

Degree of a relationship Set:

- The number of different entity sets participating in a relationship set is called as degree of a relationship set.

1. Unary Relationship –

- When there is only ONE entity set participating in a relation, the relationship is called as unary relationship.
- For example, one person is married to only one person.



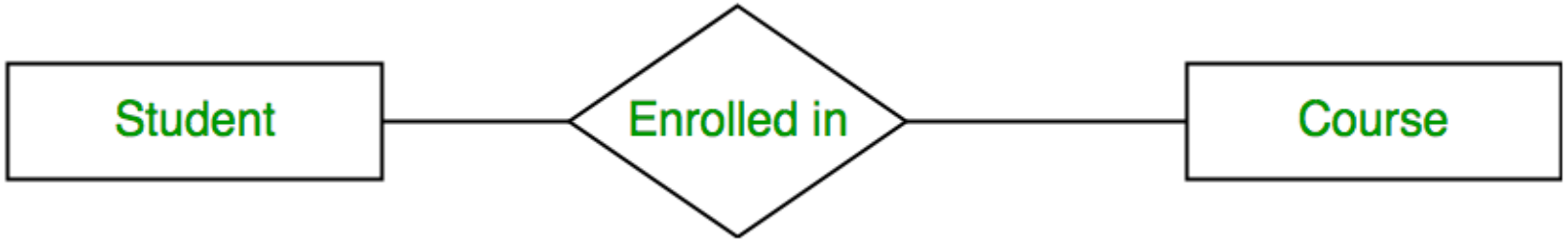
E-R Model

Degree of a relationship Set:

- The number of different entity sets participating in a relationship set is called as degree of a relationship set.

2. Binary Relationship –

- When there are TWO entities set participating in a relation, the relationship is called as binary relationship.
- For example, Student is enrolled in Course.



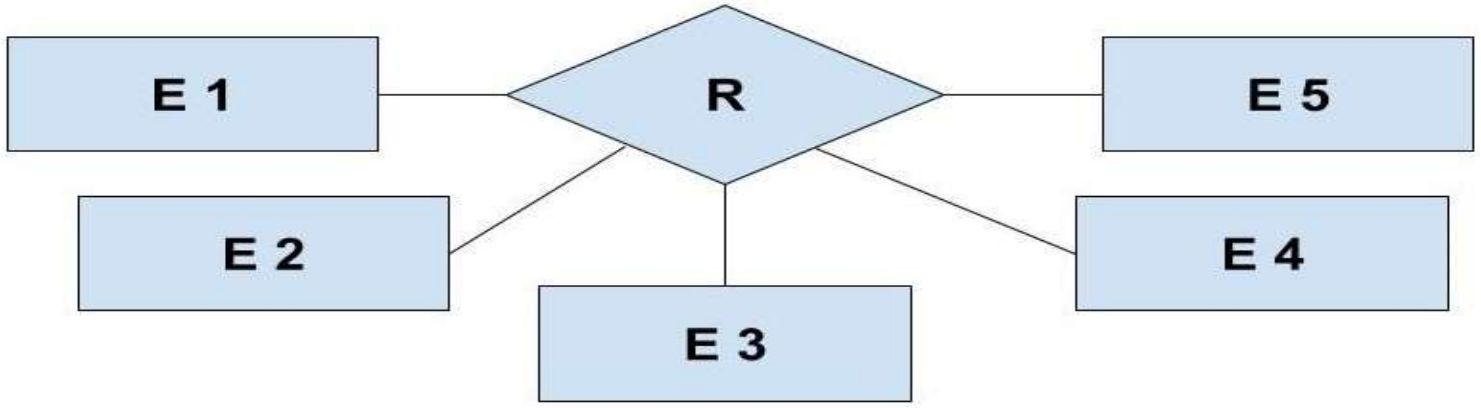
E-R Model

Degree of a relationship Set:

- The number of different entity sets participating in a relationship set is called as degree of a relationship set.

2. n-ary Relationship –

- When there are n entities set participating in a relation, the relationship is called as n-ary relationship.



N-ary Relationship

E-R Model

Cardinality:

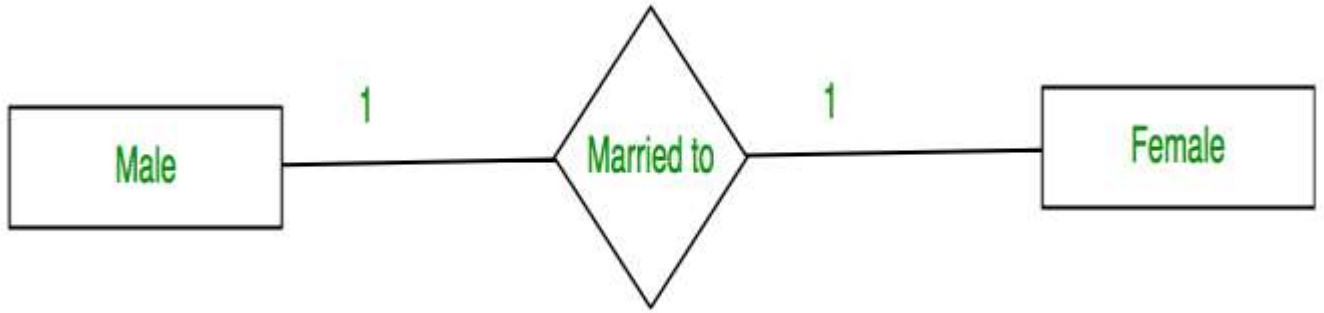
- The number of times an entity of an entity set participates in a relationship set is known as cardinality.
- Cardinality can be of different types:
  - One-to-one
  - Many-to-one
  - Many-to-many

E-R Model

Cardinality:

1. One to one –

- When each entity in each entity set can take part only once in the relationship, the cardinality is one to one.
- Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.

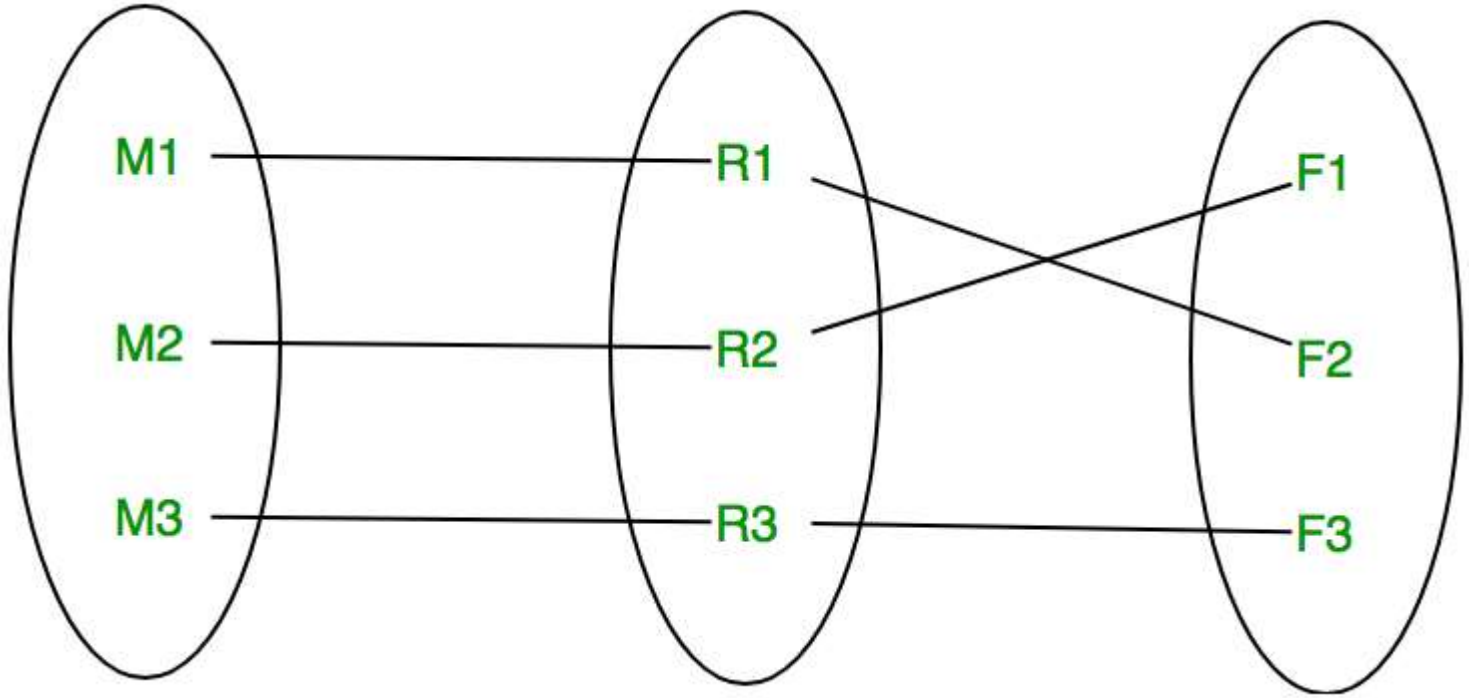


E-R Model

Cardinality:

1. One to one –

- Using Sets, it can be represented as:

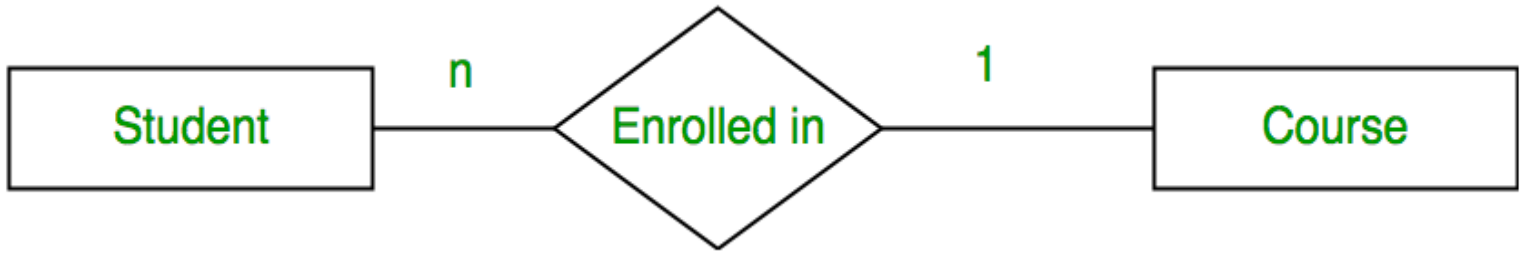


E-R Model

Cardinality:

2. Many to one –

- When entities in one entity set can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set, cardinality is many to one.
- Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1.
- It means that for one course there can be n students but for one student, there will be only one course.

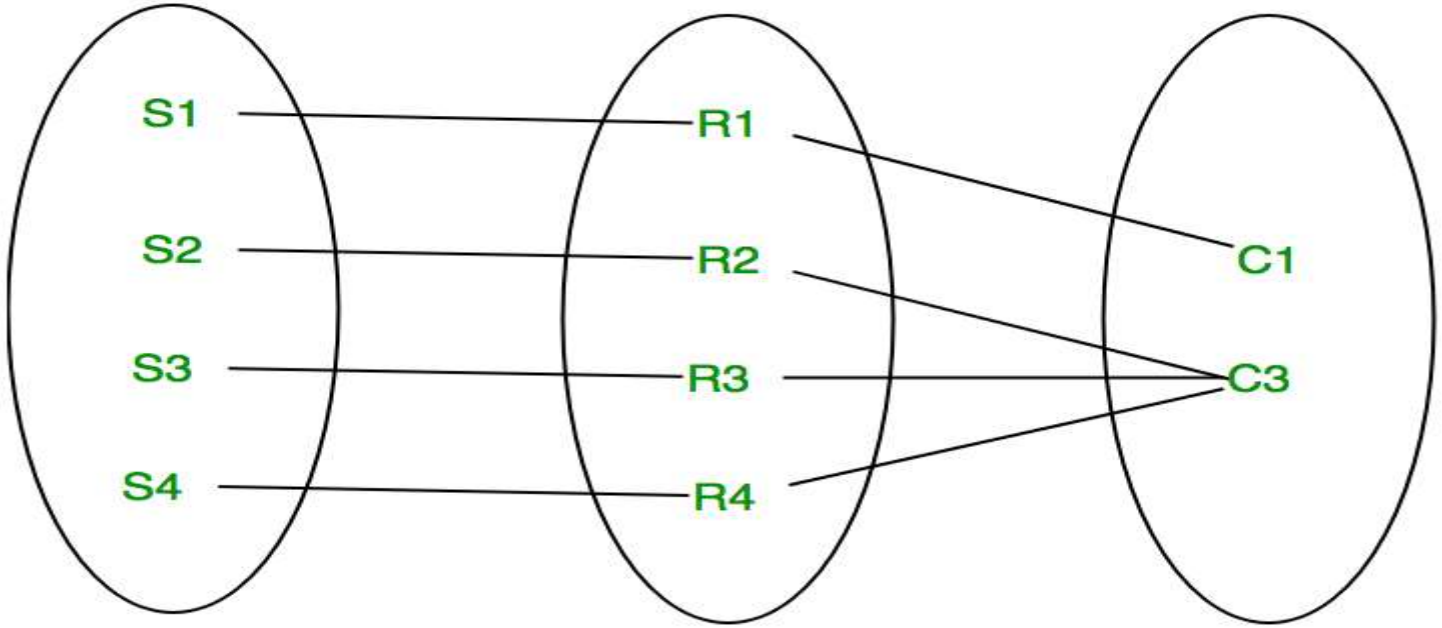


E-R Model

Cardinality:

2. Many to one –

- Using Sets, it can be represented as:

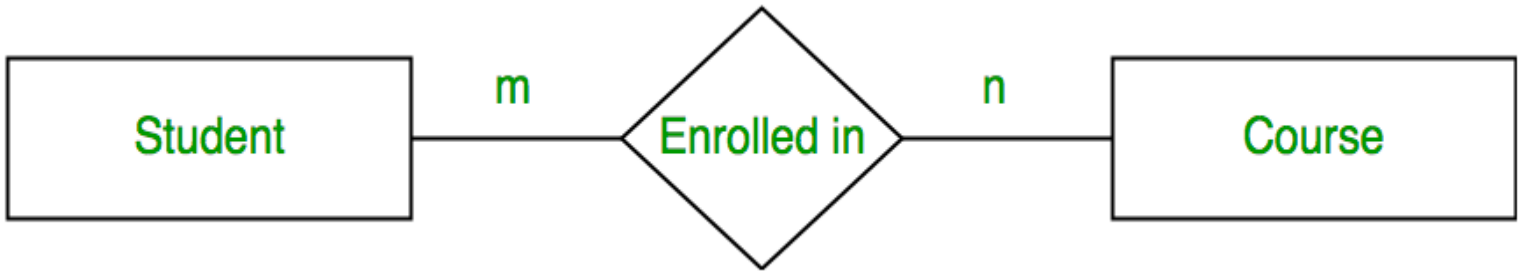


E-R Model

Cardinality:

3. Many to many –

- When entities in all entity sets can take part more than once in the relationship cardinality is many to many.
- Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

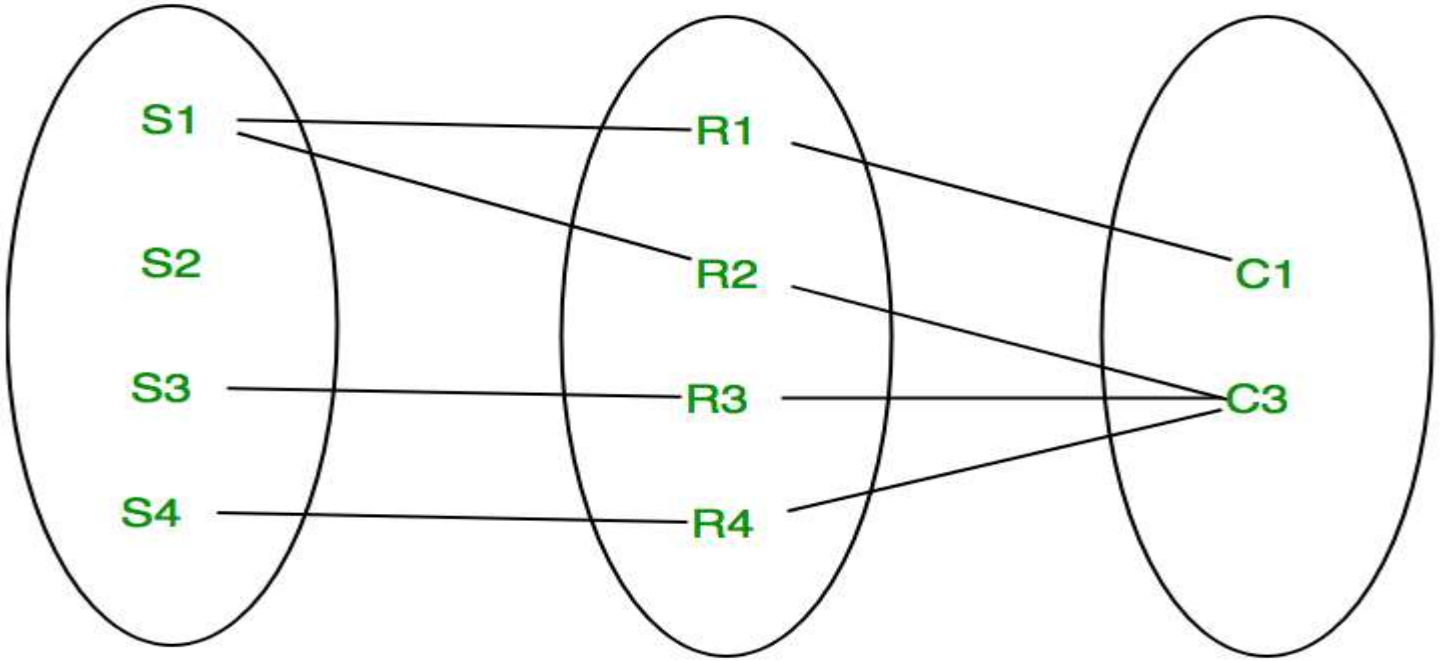


E-R Model

Cardinality:

3. Many to many –

- Using Sets, it can be represented as:



E-R Model

Cardinality:

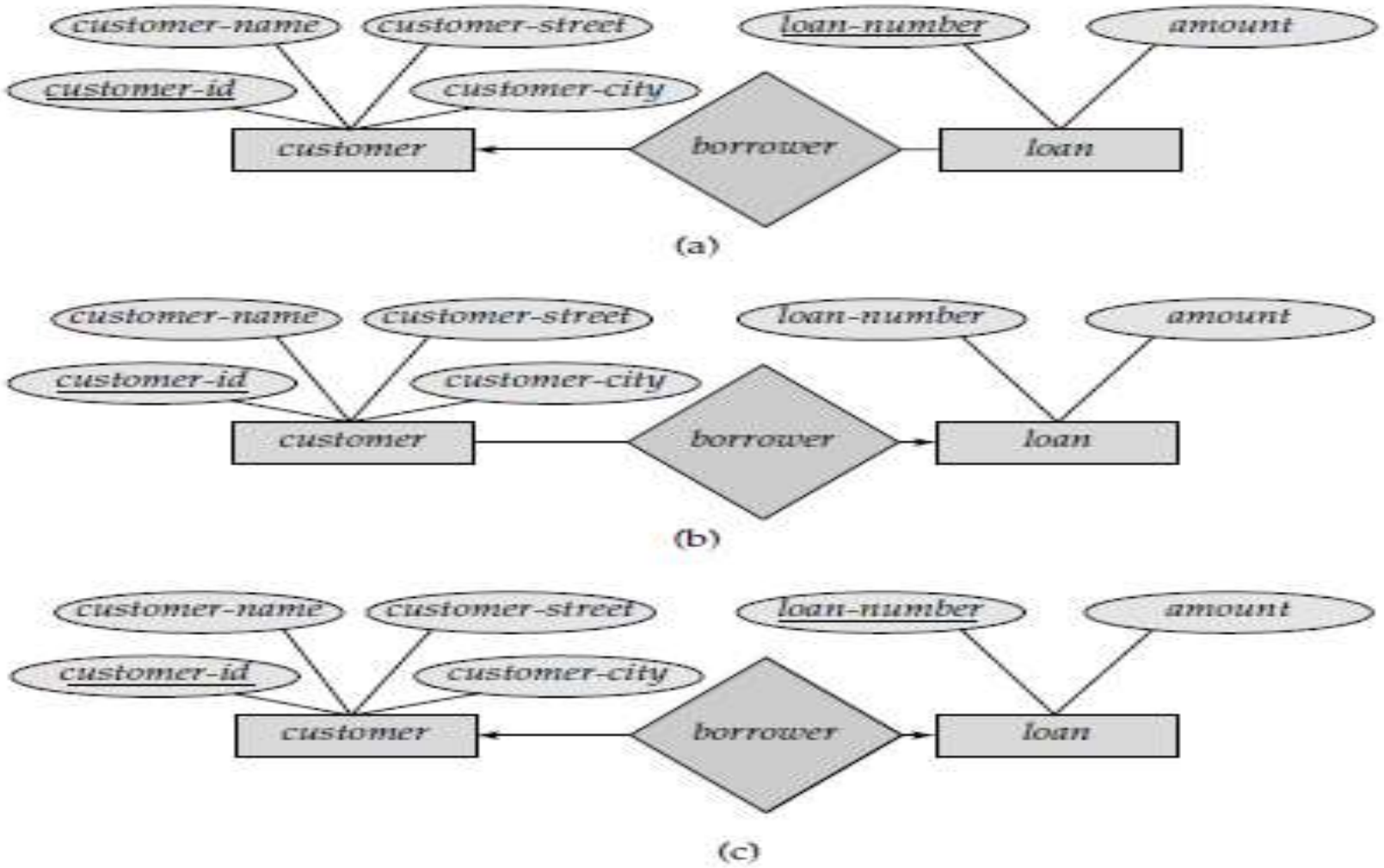


Figure 2.9 Relationships. (a) one to many. (b) many to one. (c) one-to-one.

E-R Model

Participation Constraint:

- Participation Constraint is applied on 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 student will be total. Total participation is shown by double line in ER diagram.

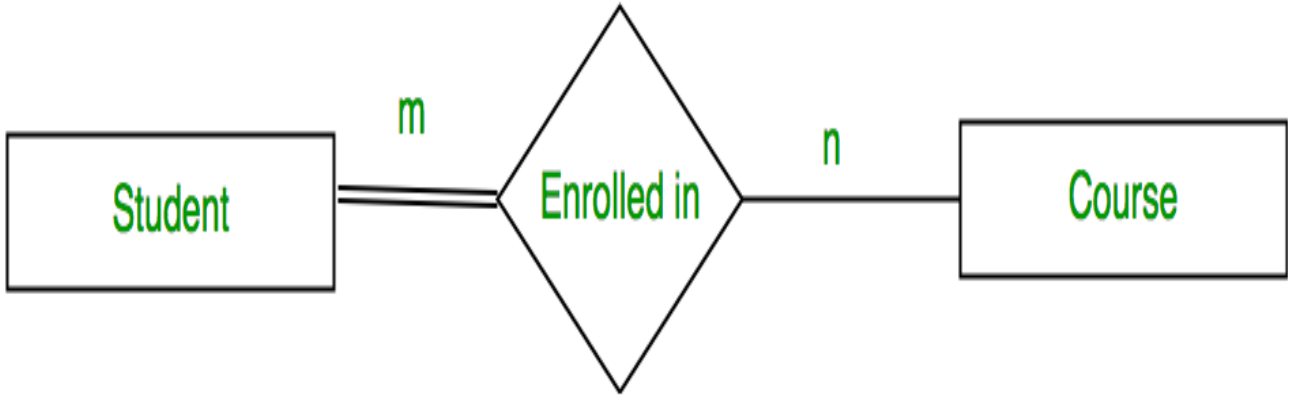
2. Partial Participation –

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

E-R Model

Participation Constraint:

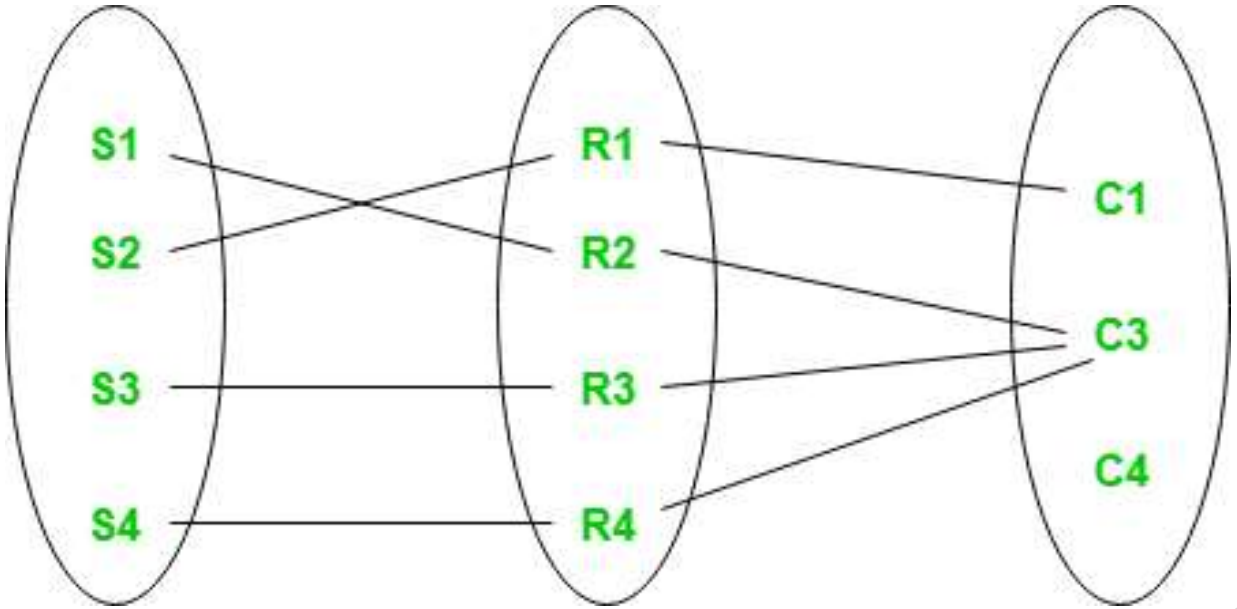
- The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



E-R Model

Participation Constraint:

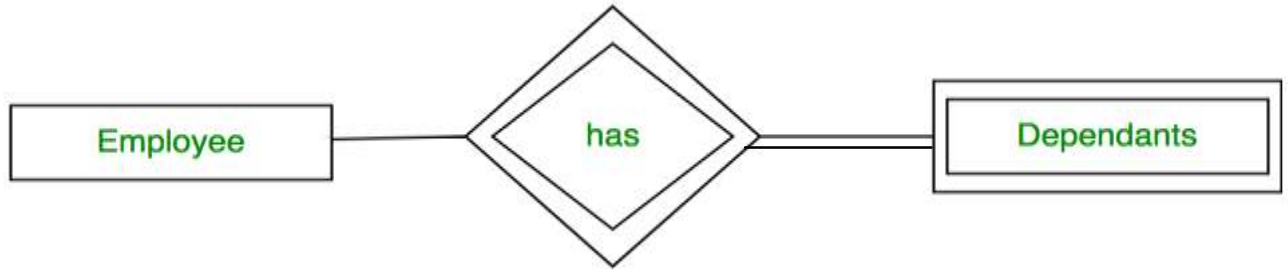
- Using set, it can be represented as
- Every student in Student Entity set is participating in relationship but there exists a course C4 which is not taking part in the relationship.



E-R Model

Weak Entity Type and Identifying Relationship:

- An entity type has a key attribute which uniquely identifies each entity in the entity set.
- But there exists some entity type for which key attribute can’t be defined. These are called Weak Entity type.
- For example, A company may store the information of dependents (Parents, Children, Spouse) of an Employee.
- But the dependents don’t have existence without the employee. So Dependent will be weak entity type and Employee will be Identifying Entity type for Dependent.
- A weak entity type is represented by a double rectangle.
- The participation of weak entity type is always total.
- The relationship between weak entity type and its identifying strong entity type is called identifying relationship and it is represented by double diamond.



E-R Model

Role indicator:

- If an entity set plays more than one role, role indicators describe the different purpose in the relationship.

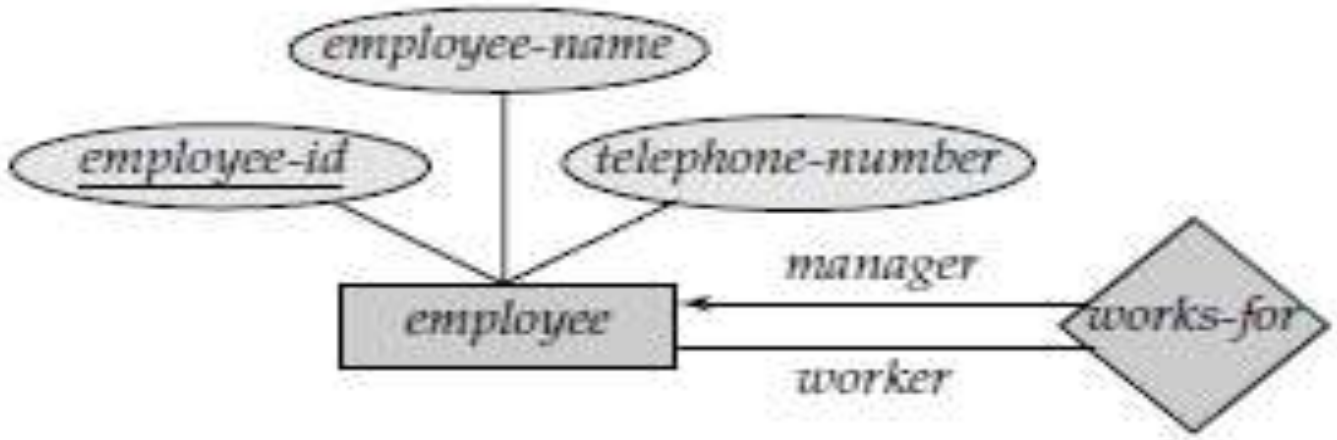


Figure 2.12 E-R diagram with role indicators.

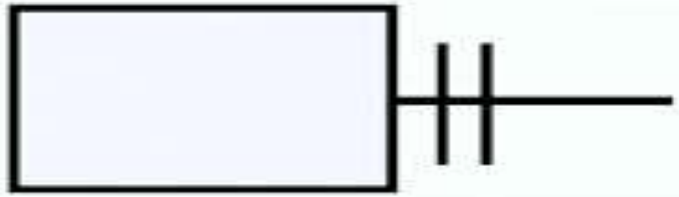
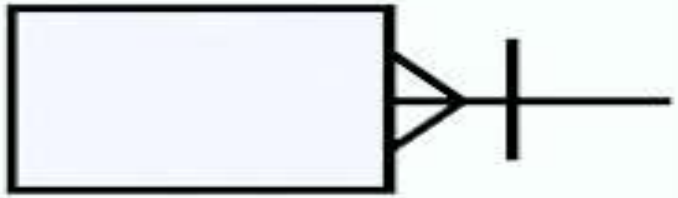
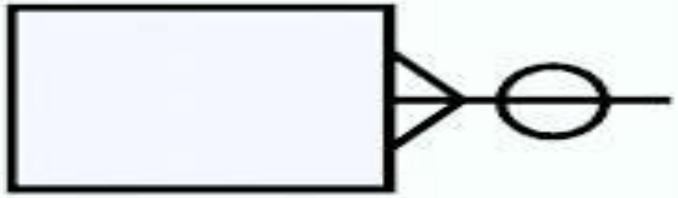
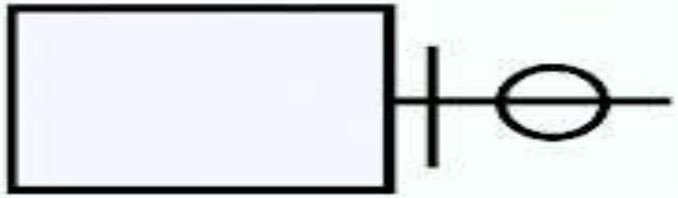
E-R Model

ENTITY-RELATIONSHIP DIAGRAMS

- Crow's foot notation: A type of cardinality notation. It is called crow's foot notation because of the shapes, which include circles, bars, and symbols, that indicate various possibilities.

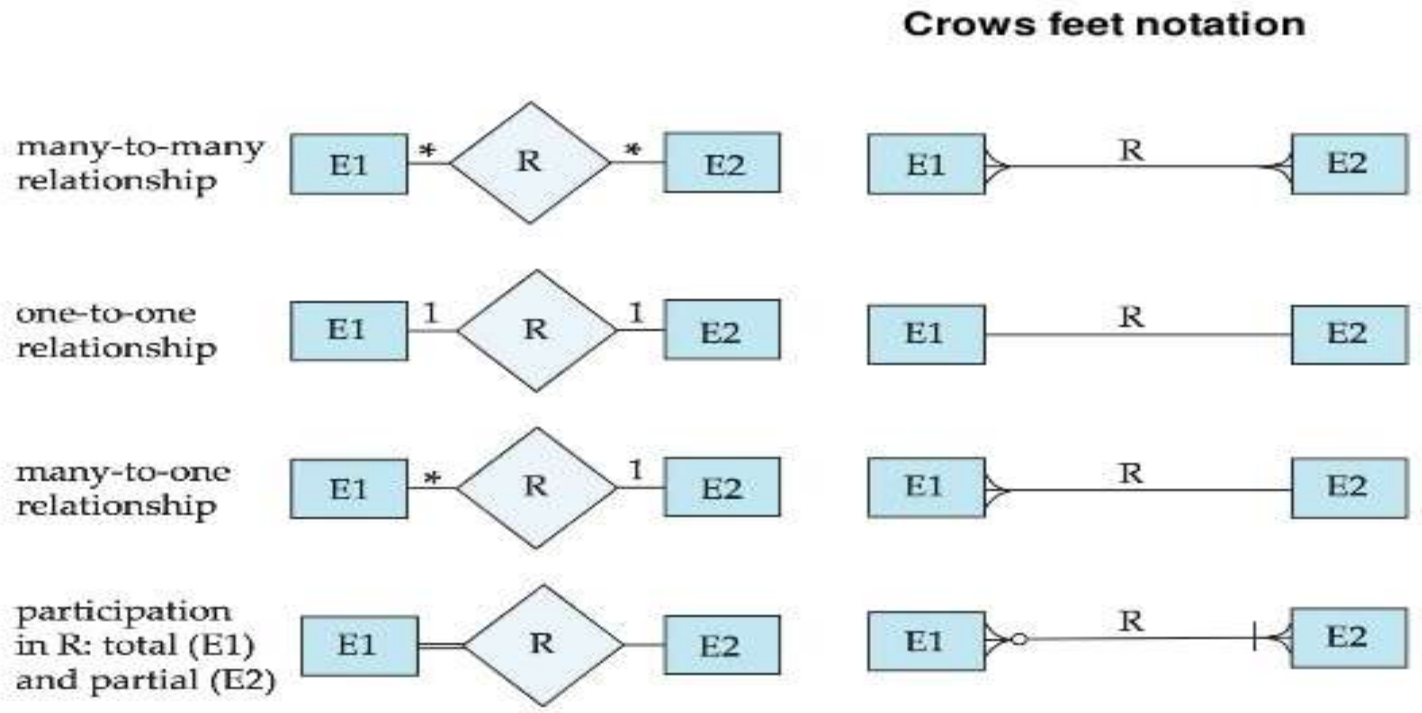
- A single bar indicates one, a double bar indicates one and only one, a circle indicates zero, and a crow's foot indicates many.

E-R Model

SYMBOL	MEANING	UML REPRESENTATION
	One and only one	1
	One or many	1..*
	Zero, or one, or many	0..*
	Zero, or one	0..1

E-R Model

Summary of Symbols Used in E-R Notation



E-R Model

Problem

- A company database needs to store information about employees (identified by *ssn, with salary and phone as attributes*), *departments (identified by dno,* with *dname and budget as attributes), and children of employees (with name and age* as attributes).

Model

Problem

- Employees *work in departments; each department is managed by an* employee; a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.
- Draw an ER diagram that captures this information.

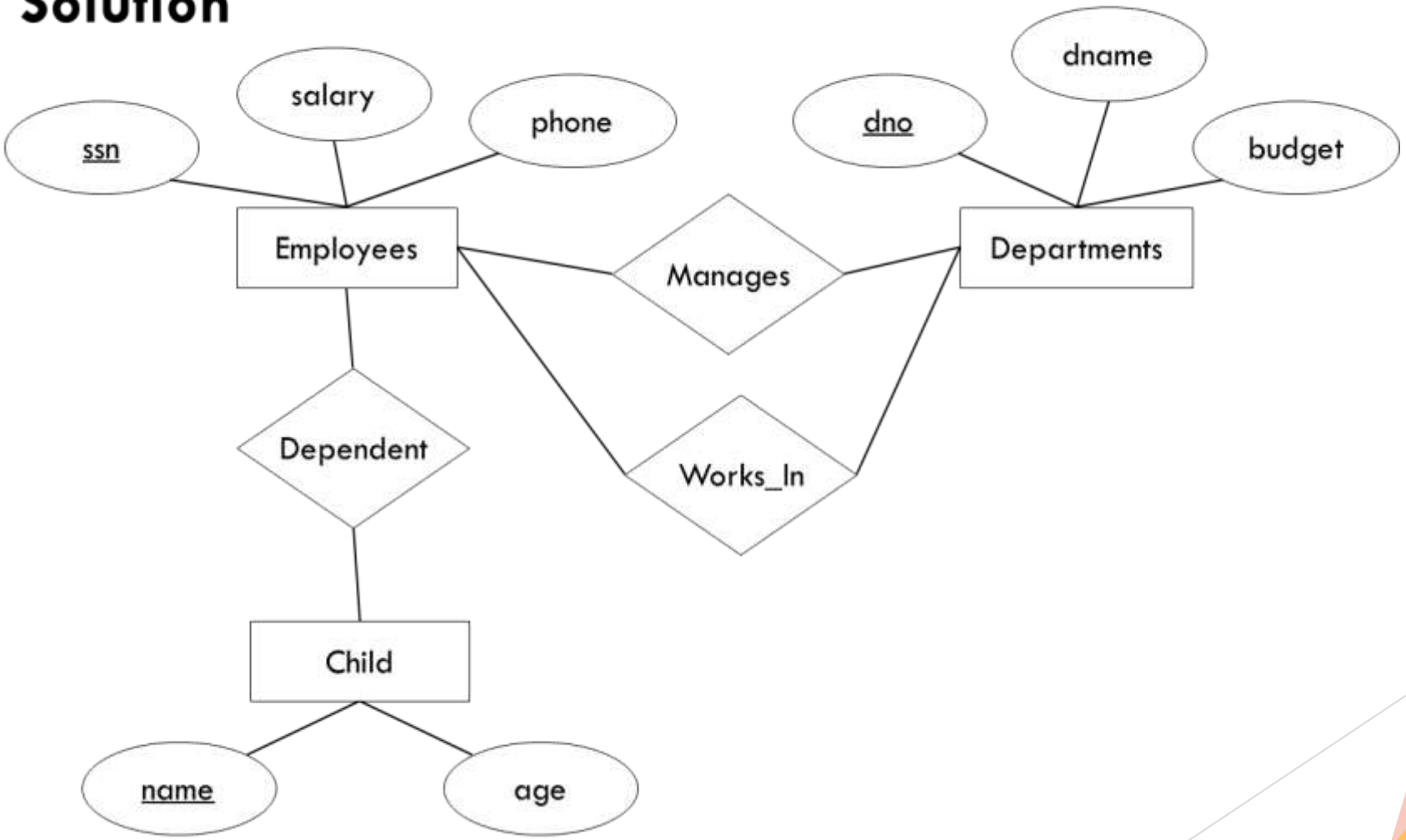
E-R Model

Solution

- First, we shall design the entities and relationships.
  - "Employees *work in departments*…"
  - "…*each department is managed by an* employee…"
  - "…a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known."

E-R Model

Solution

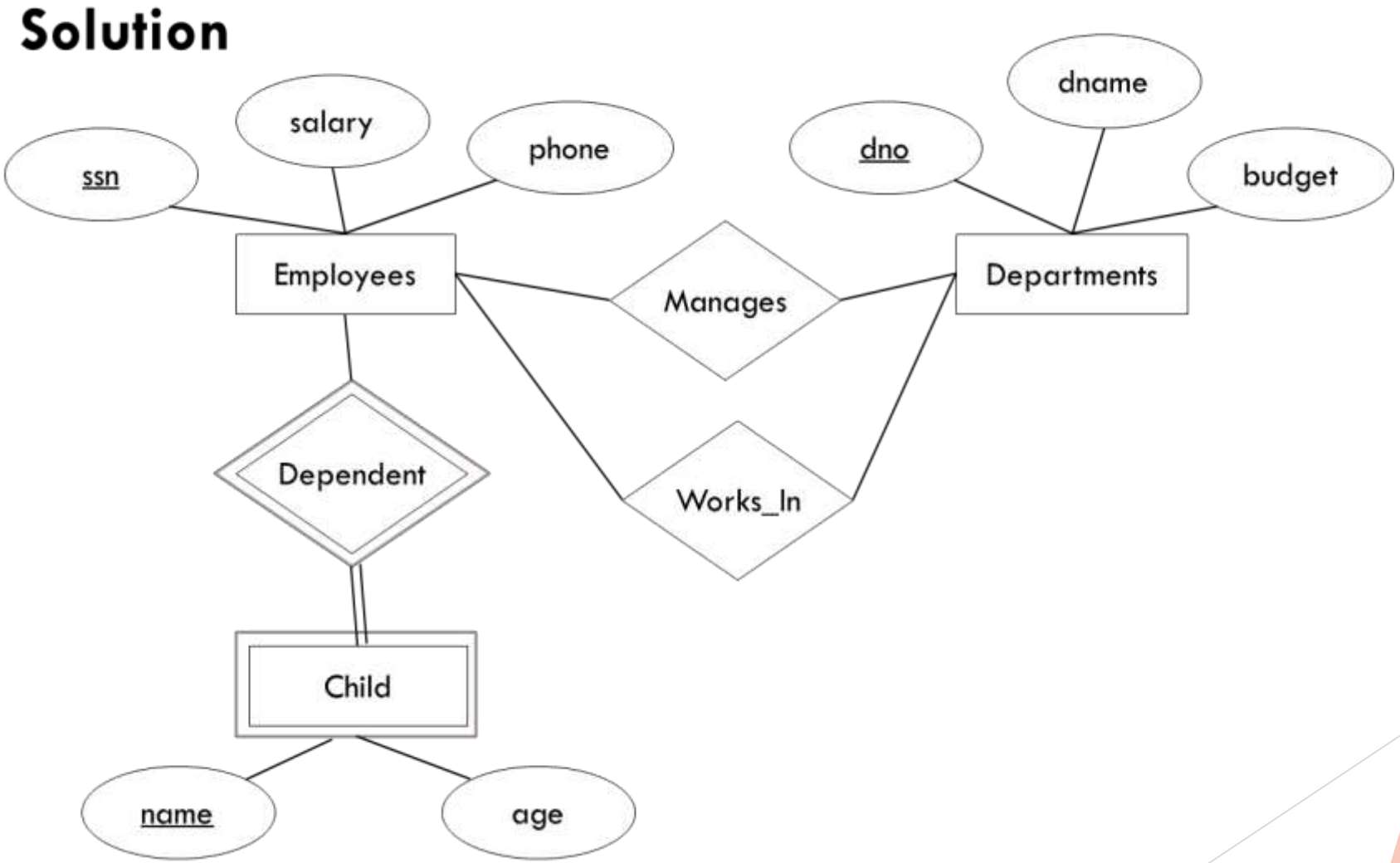


E-R Model

Solution

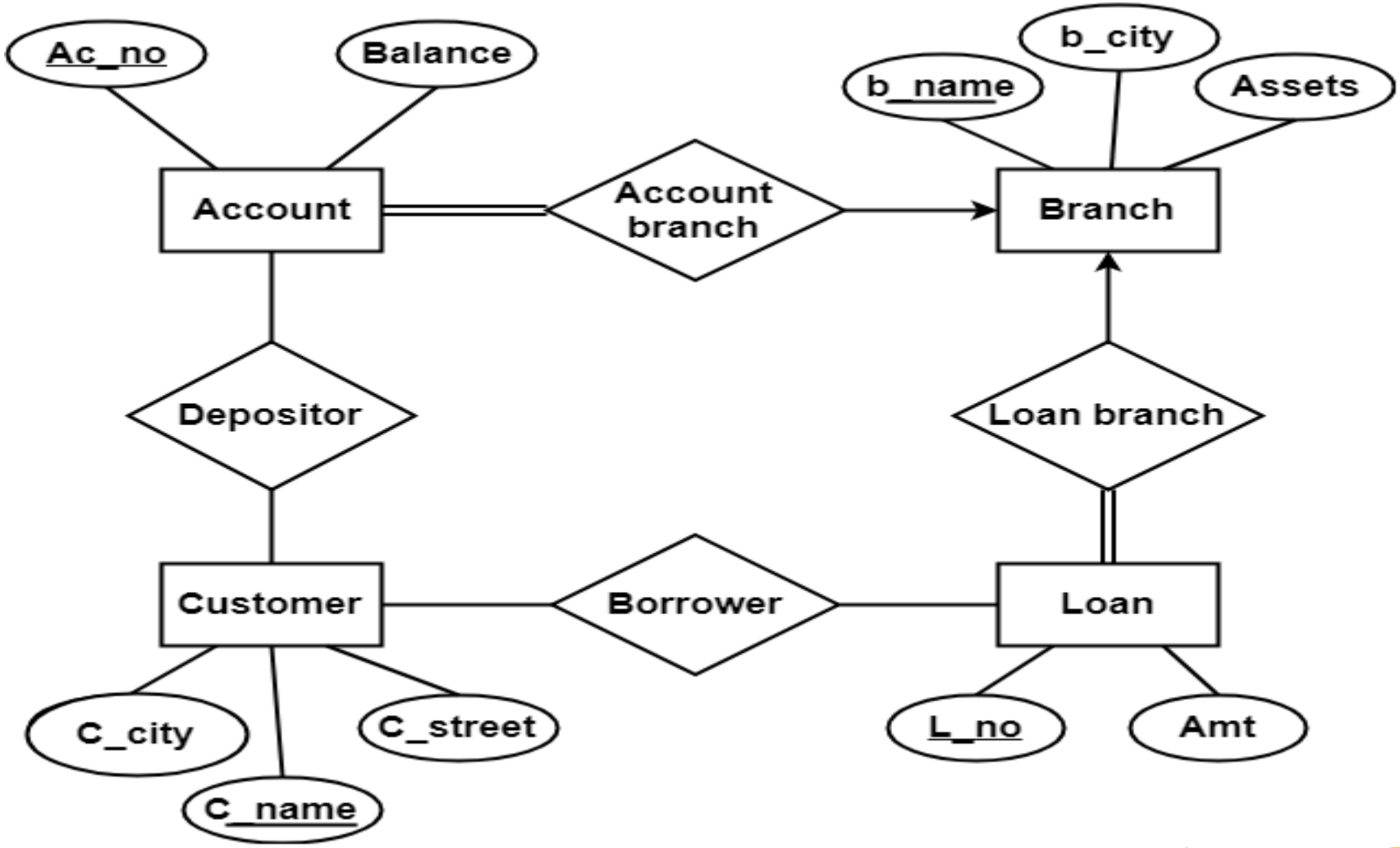
- Now, we will design the constraints.
  - "…*each department is managed by* ***an*** employee…"
  - "…a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. "
  - "We are not interested in information about a child once the parent leaves the company."

E-R Model


Solution
salary
ssn
phone
dname
dno
budget
Employees
Manages
Departments
Dependent
Works_In
Child
name
age

2.3 E-R Model

Example : E-R diagram of Banking Enterprises.



E-R Model

Assignment :

Practice ER Diagram Question – A Sample Solution

Suppose you are given the following requirements for a simple database for the National Cricket League (NCL):

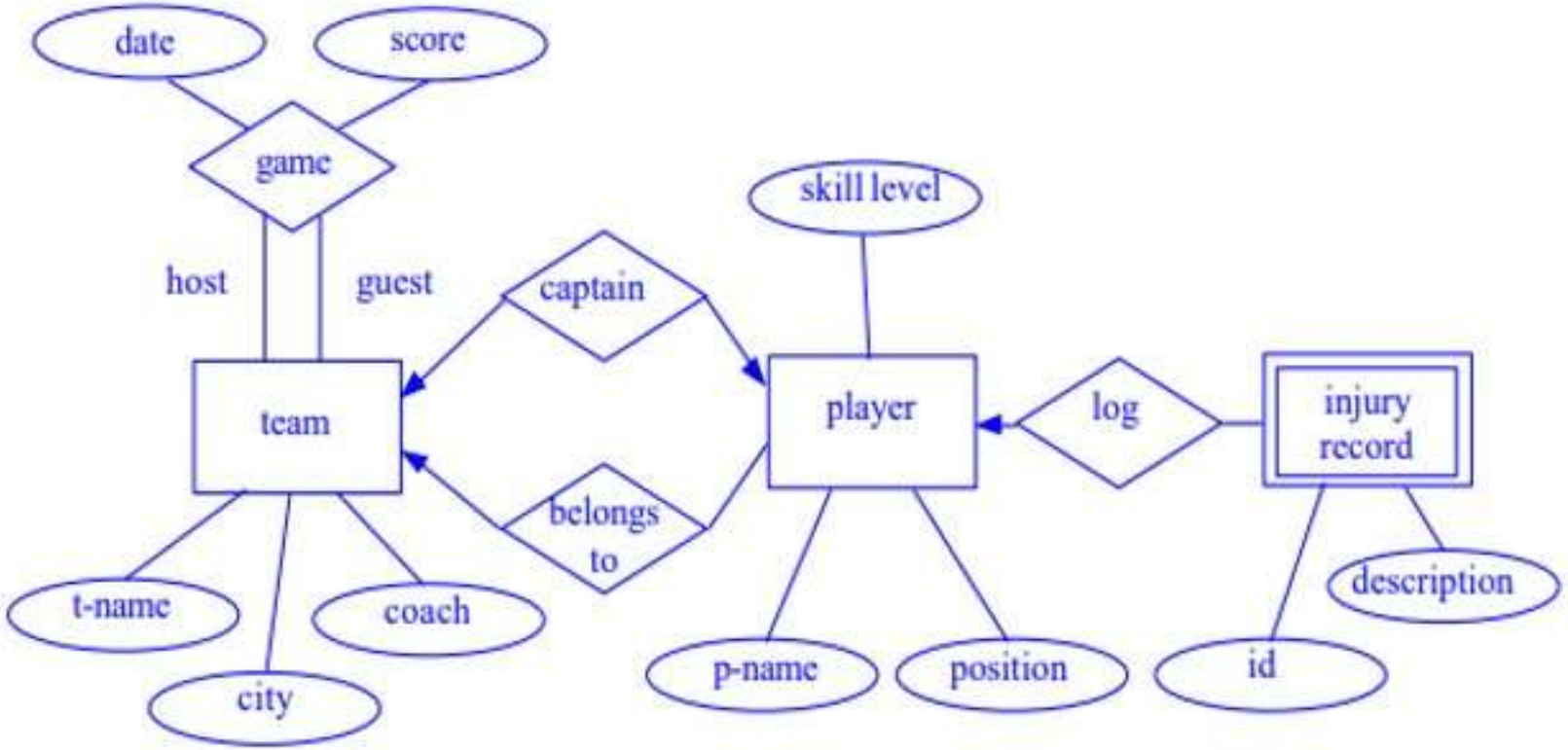
- the NCL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as OCt 3rd, 2020) and a score (such as 4 to 2).

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

E-R Model

Assignment :

Practice ER Diagram Question – A Sample Solution



E-R Model

Extended feature of E-R Model:

- As part of the Enhanced ER Model, along with other improvements, three new concepts were added to the existing ER Model, they were:
  - Generalization
  - Specialization
  - Aggregation
- Let's understand what they are, and why were they added to the existing ER Model

E-R Model

Extended feature of E-R Model:

1. Generalization

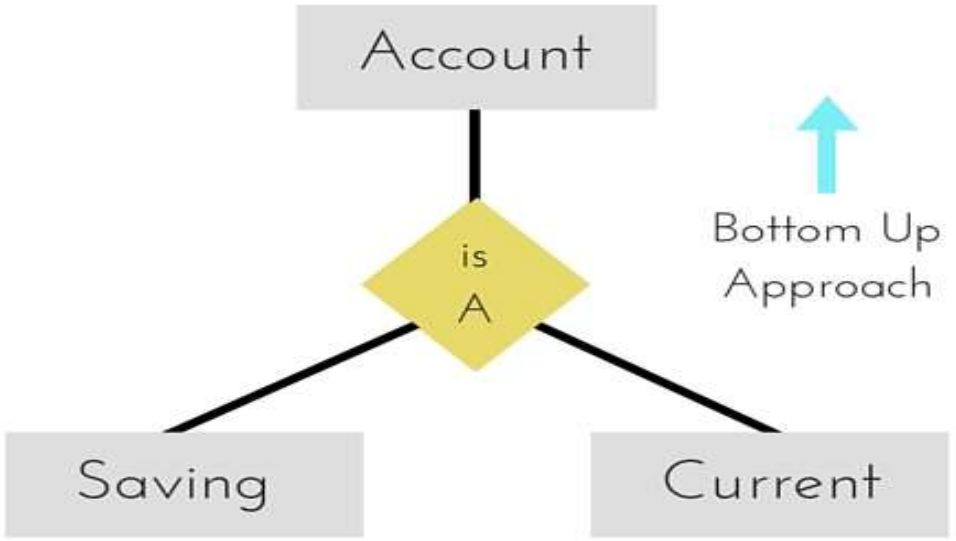
- Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity.
- In generalization, the higher level entity can also combine with other lower level entities to make further higher level entity.
- It's more like Superclass and Subclass system, but the only difference is the approach, which is bottom-up.
- Hence, entities are combined to form a more generalized entity, in other words, sub-classes are combined to form a super-class.

2.3 E-R Model

Extended feature of E-R Model:

1. Generalization

- For example, Saving and Current account types entities can be generalized and an entity with name Account can be created, which covers both.

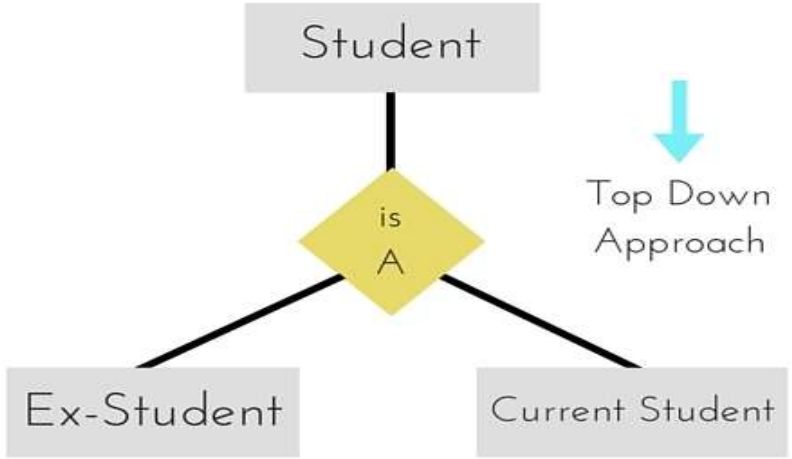


E-R Model

Extended feature of E-R Model:

2. Specialization

- Specialization is opposite to Generalization.
- It is a top-down approach in which one higher level entity can be broken down into two lower level entity.
- In specialization, a higher level entity may not have any lower-level entity sets, it's possible.

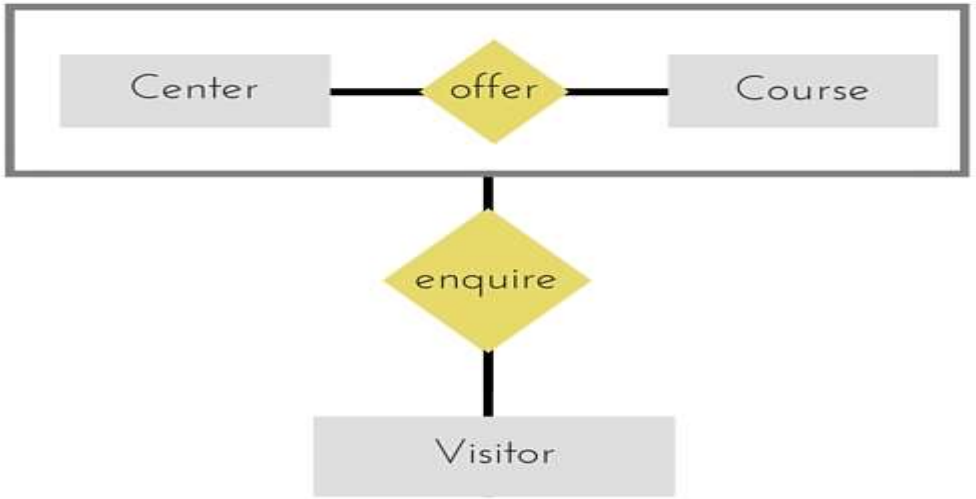


E-R Model

Extended feature of E-R Model:

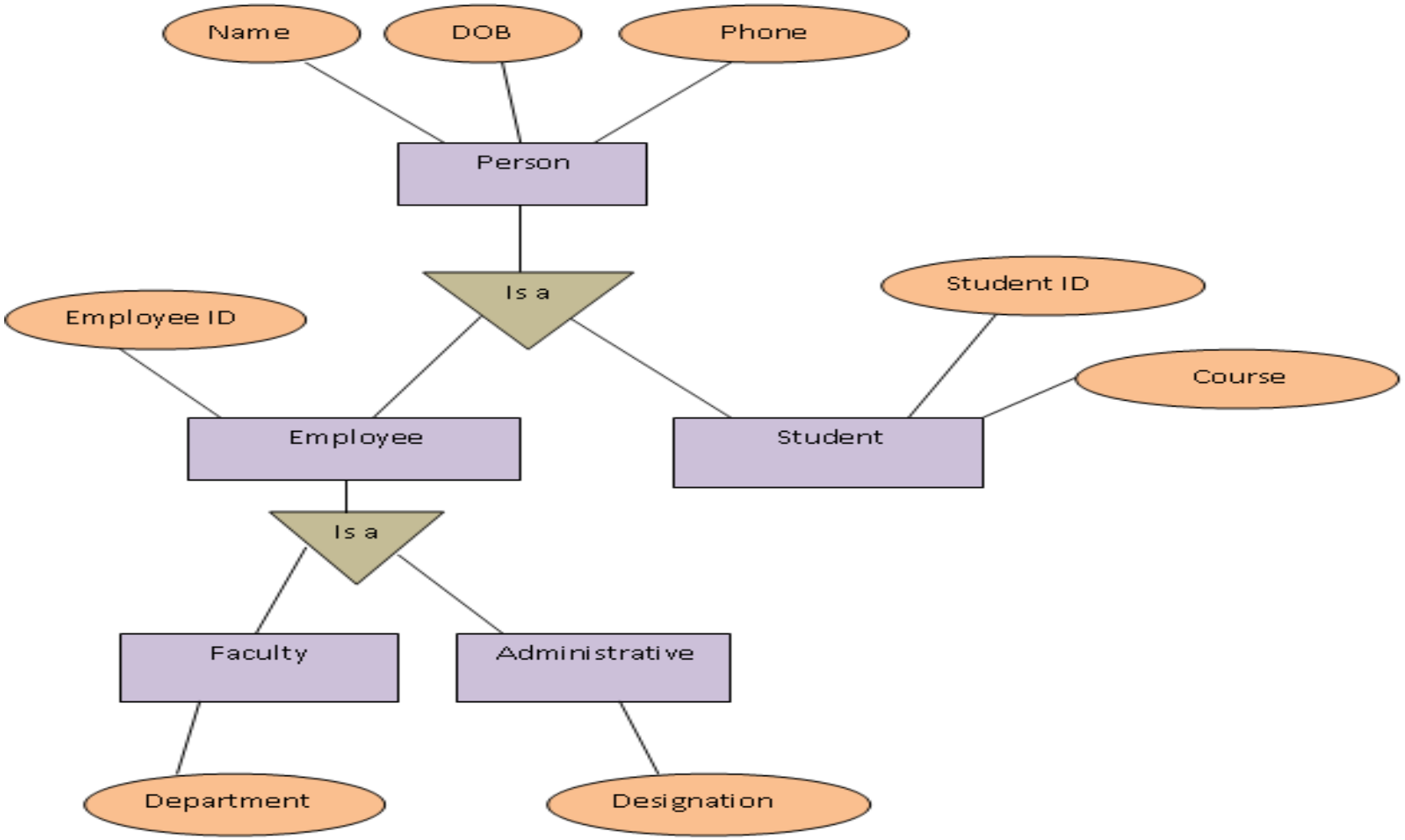
3. Aggregation

- Aggregation is a process when relation between two entities is treated as a single entity.
- In the diagram above, the relationship between Center and Course together, is acting as an Entity, which is in relationship with another entity Visitor.



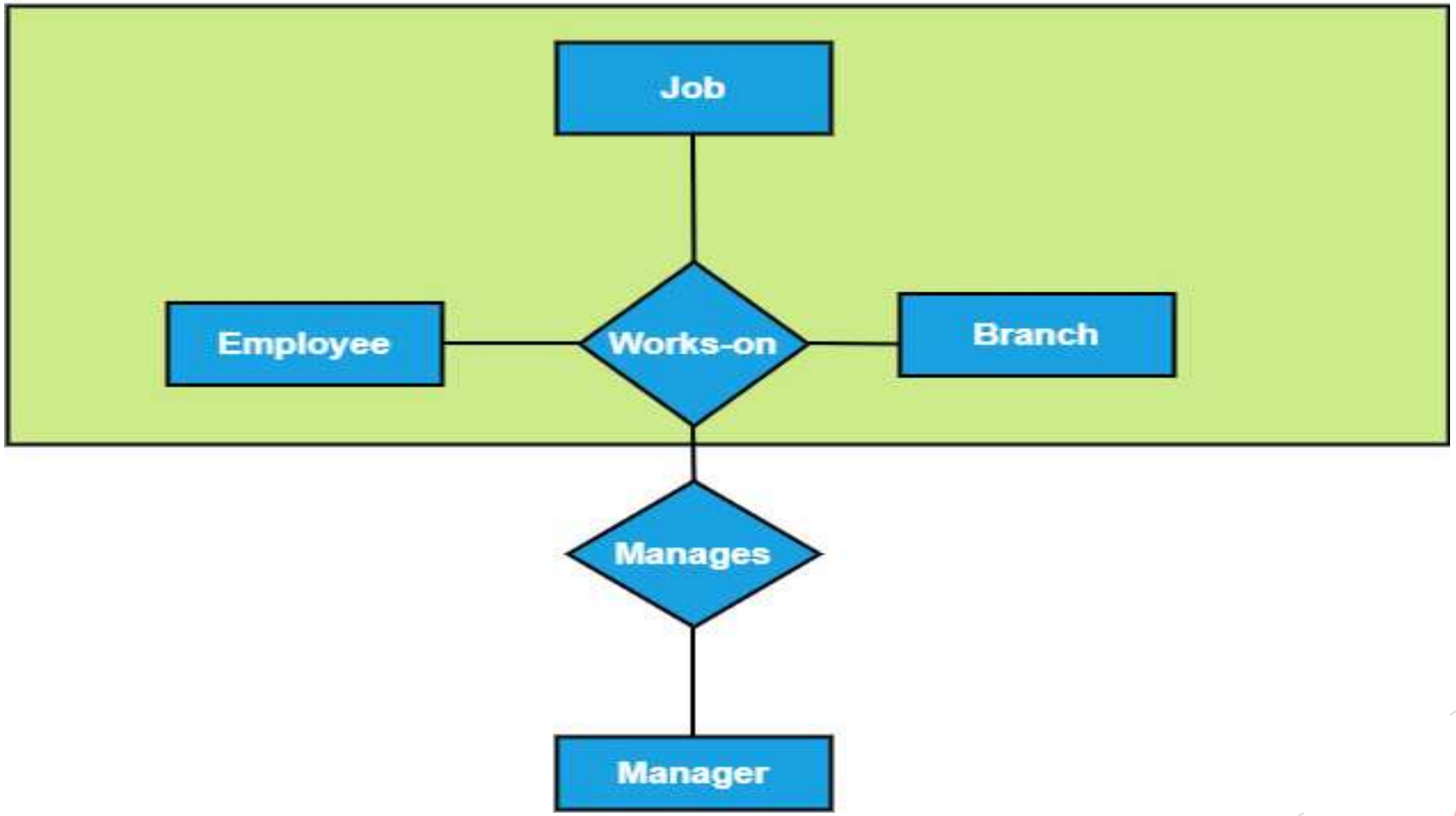
E-R Model

Extended feature of E-R Model: Specialization and Generalization



E-R Model

Extended feature of E-R Model: **Aggregation**



2.3 E-R Model

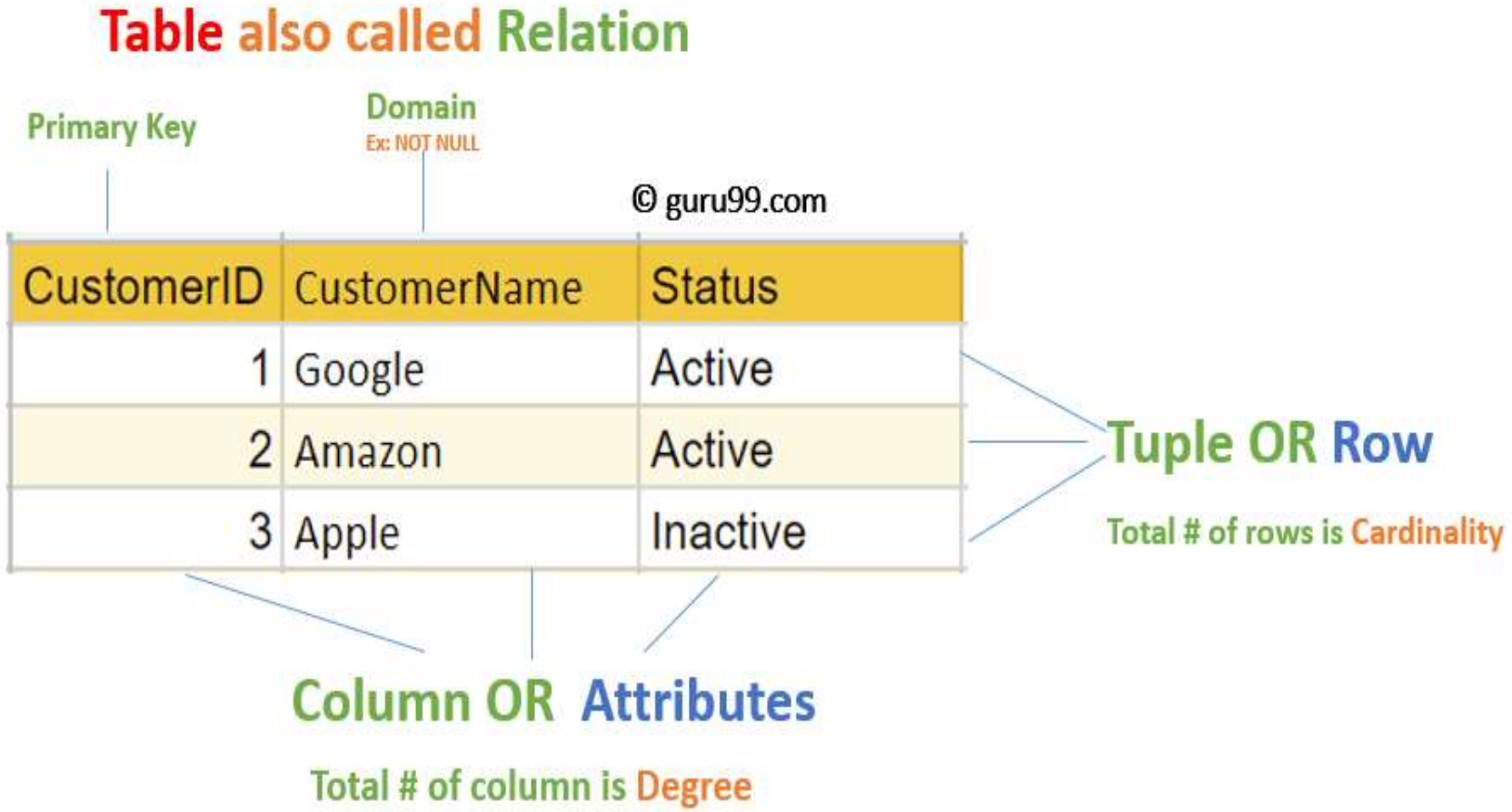
Assignment

Draw an ER diagram of library management system.

Relational Model

- Relational Model was proposed by Dr. Edgar F. Codd to form data in the form of relations or tables.
- Relation model was attempted to specify the database structure in term of matrix i.e. the database should contain tables.
- The table is in form of set of Columns and Rows.
- Tables in the database is known as relation and Columns in the table is called as attributes of an tables and rows in the table is called records or tuples.

Relational Model



Relational Model

Relational Model Terminology

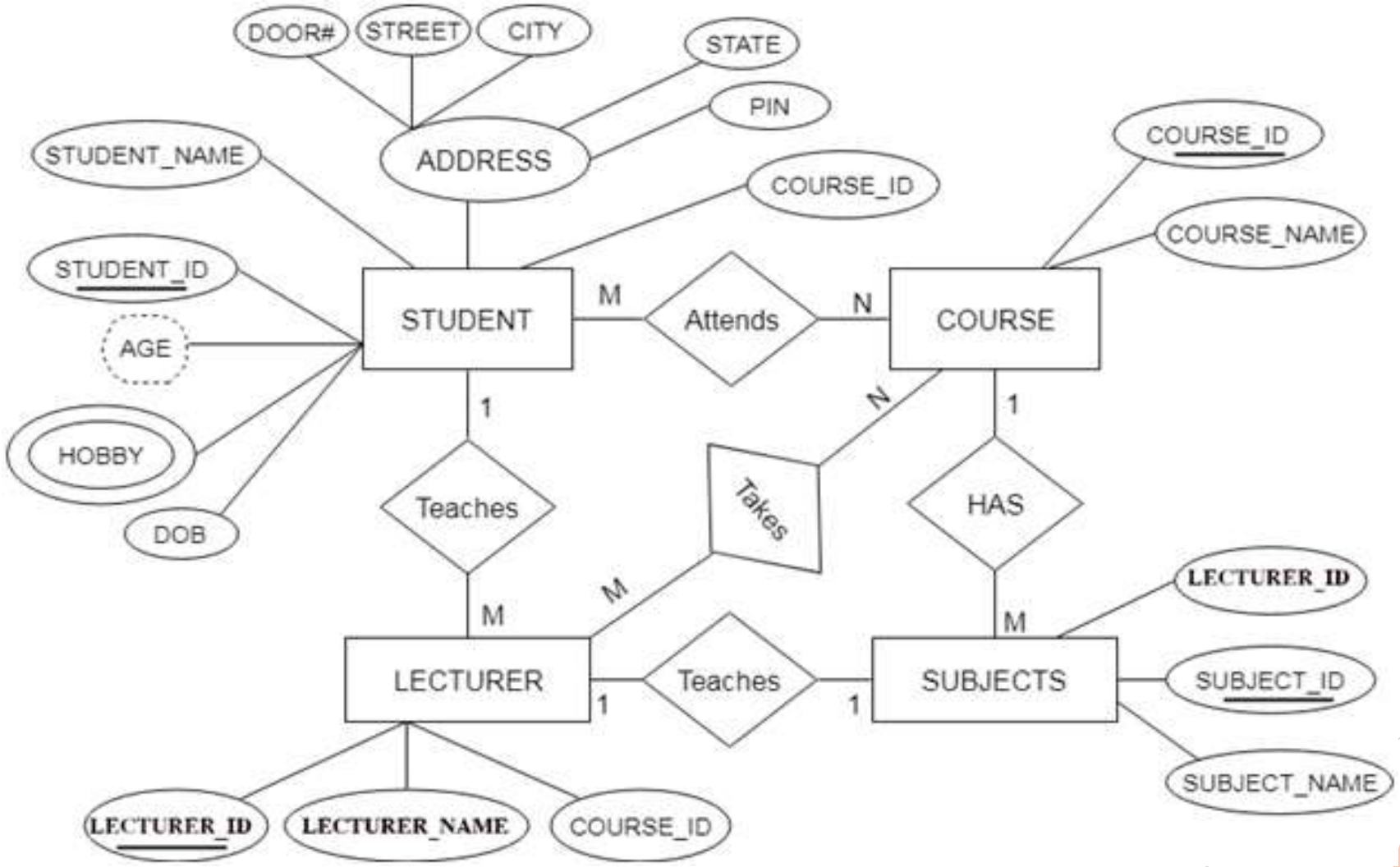
- **Attribute:** Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
- **Tables:** In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
- **Tuple:** It is nothing but a single row of a table, which contains a single record.
- **Relation Schema:** A relation schema represents the name of the relation with its attributes.
- **Degree:** The total number of attributes which in the relation is called the degree of the relation.

Relational Model

Relational Model Terminology

- **Cardinality:** Total number of rows present in the Table.
- **Column:** The column represents the set of values for a specific attribute.
- **Relation instance:** Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
- **Relation key:** Every row has one, two or multiple attributes, which is called relation key.
- **Attribute domain:** Every attribute has some pre-defined value and scope which is known as attribute domain

Reducing E-R diagram to Tables



Reducing E-R diagram to Tables Contd.

Some points to remember:

- **Entity type becomes a table.**
  - In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.
- **All single-valued attribute becomes a column for the table.**
  - In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.
- **A key attribute of the entity type represented by the primary key.**
  - In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

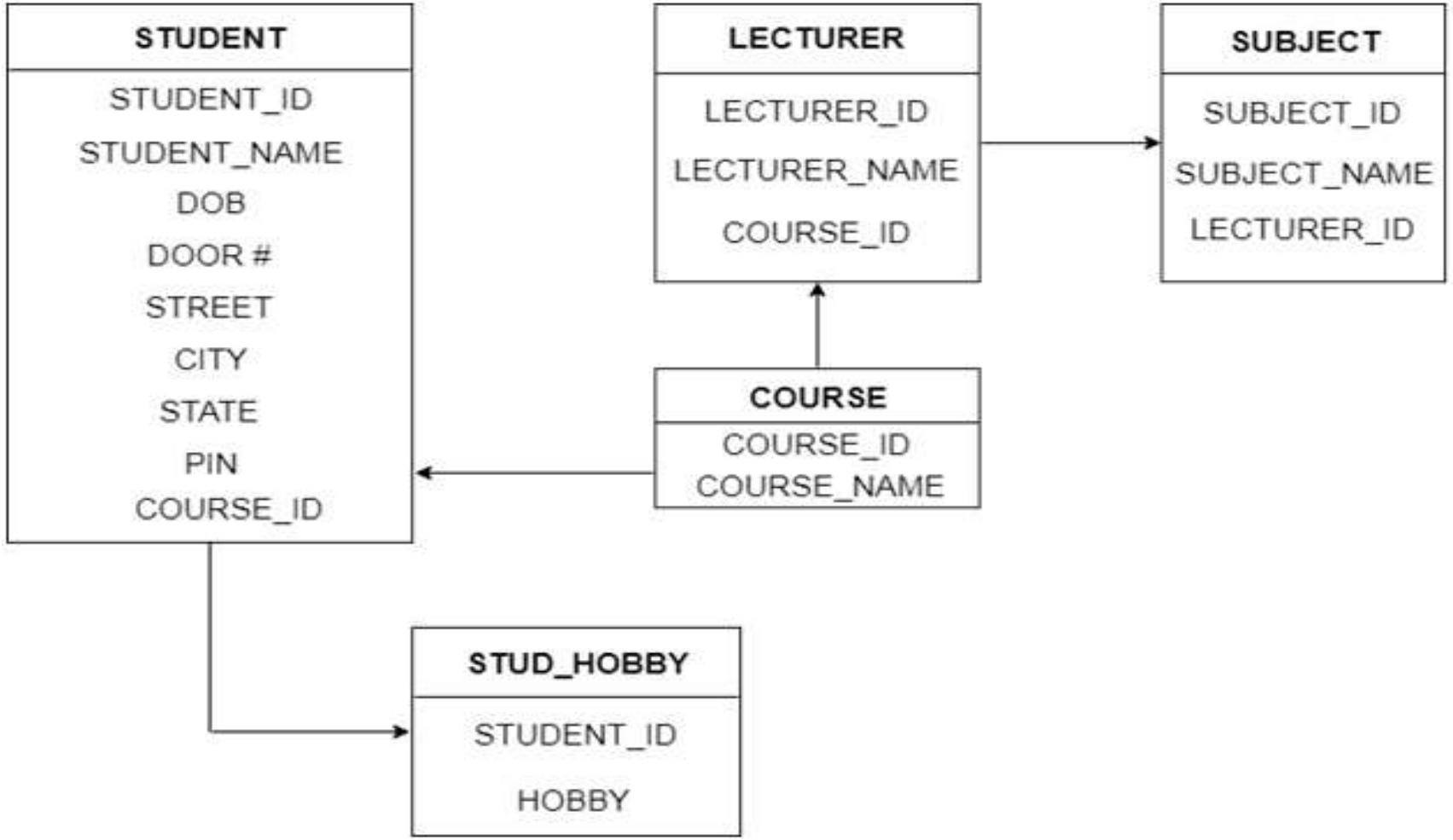
Reducing E-R diagram to Tables Contd.

Some points to remember:

- **The multivalued attribute is represented by a separate table.**
  - In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.
- **Composite attribute represented by components.**
  - In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.
- **Derived attributes are not considered in the table.**
  - In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Reducing E-R diagram to Tables Contd.

Table structure for the given ER diagram is as below:



Reducing E-R diagram to Tables

If you want to know more about how to reduce E-R diagram to tables, just go to the given links.

http://www.exploredatabase.com/2017/07/reduce-er-diagram-to-relation-table-solved-exercise.html

http://www.exploredatabase.com/2016/04/convert-entity-relationship-diagram-to-relation-schemas-exercises.html

Structure of relational database

Basic Structure

- The relational model has following structural characteristics:
  - A relational database contains multiple tables.
  - Each table stores data about one specific entity or object.
  - Fields contain data describing the entity of a table.
  - Records are particular instances of the subject of a table.
  - A special primary key field uniquely identifies each record in a table.

Structure of relational database

Basic Structure

- To illustrate basic structure of database, let us consider a table “account”

account_number	*branch_name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

Structure of relational database

Basic Structure

- Formally, given sets D_1, D_2, ... D_n a **relation** r is a subset of
  $D_1 \times D_2 \times \ldots \times D_n$
  Thus, a relation is a set of n-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- Example: If
  - *customer_name* = {Jones, Smith, Curry, Lindsay, ...}
    /* Set of all customer names */
  - *customer_street* = {Main, North, Park, ...} /* set of all street names*/
  - *customer_city* = {Harrison, Rye, Pittsfield, ...} /* set of all city names */

  Then r = { (Jones, Main, Harrison),
  (Smith, North, Rye),
  (Curry, North, Rye),
  (Lindsay, Park, Pittsfield) }

  is a relation over

  customer_name x *customer_street* x *customer_city*

Structure of relational database

Attribute types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
  - E.g. the value of an attribute can be an account number, but cannot be a set of account numbers
- Domain is said to be atomic if all its members are atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations
  - We shall ignore the effect of null values in our main presentation and consider their effect later

Structure of relational database

Database Schema(Relation Schema)

- A relation schema is a list of attributes and their corresponding domains.
- It is a logical design of the database.
- Database instance (relation instance) is a snapshot of the data in the database at a given instant in time.
- Contents of relation instance (i.e. value) may change with time as relation is updated.
- But schema of a relation does not change generally.

Structure of relational database

Relation Schema

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

```
Customer-schema = (customer_id, customer_name,
                                    customer_city)
```

- We may also specify domains of attributes as

```
Customer-schema = (customer_id: integer, customer_name:
                        string,customer_city:string)
```

- We may state customer is a relation on Customer-schema by

```
customer(Customer-schema)
```

Structure of relational database

Database Schema(Relation Schema)

- Relational database is a collection of relations and relational database schema is a collection of schemas for relations in database.
- Database schemas for banking enterprise

```
Branch-schema = (branch_name,branch_city,assets)
Account_schema = (account_number,branch_name,balance)
Customer-schema =
(customer_id,customer_name,customer_street,customer_cit
y)
Depositor-schema = (customer_id,account_number)
Loan-schema = (loan_number,branch_name,amount)
Borrower-schema = (customer_id,loan_no)
```

Relational Model

Keys

- An attribute or the set of attribute in the table that uniquely identifies each record in the entity set is called a key for that entity set.

Types of keys

- **Simple key –** A key which has single attribute is known as a simple key.
- **Composite Key** - If any single attribute of a table is not capable of being the key i.e. it cannot identify a row uniquely, then we combine two or more attributes to form a key. This is known as a composite key.
- **Super Key** - Super Key is the superset of primary key. The super key contains a set of attributes, including the primary key, which can uniquely identify any data row in the table.

Relational Model

Types of keys

- **Candidate Key** - The candidate keys in a table are defined as the set of keys that is minimal and can uniquely identify any data row in the table.
- **Primary Key** - The primary key is selected from one of the candidate keys and becomes the identifying key of a table. It can uniquely identify any data row of the table.
- **Foreign Key** - A foreign key is an attribute value in a table that acts as the primary key in another. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.
- **Alternate or Secondary Key** - Only one of the candidate keys is selected as the primary key. The rest of them are known as secondary keys.

Relational Model

Keys

- An example to explain the different keys is −

Student

Student_Number	Student_Name	Student_Phone	Subject_Number
1	Andrew	6615927284	10
2	Sara	6583654865	20
3	Harry	4647567463	10

Subject

Subject_Number	Subject_Name	Subject_Instructor
10	DBMS	Korth
20	Algorithms	Cormen
30	Algorithms	Leiserson

Relational Model

Keys

- An example to explain the different keys is −

Enroll

Student_Number	Subject_Number
1	10
2	20
3	10

The Super Keys in **<Subject>** table are −

{Subject_Number}

{Subject_Number,Subject_Name}

{Subject_Number,Subject_Instructor}

{Subject_Number,Subject_Name,Subject_Instructor}

{Subject_Name,Subject_Instructor}

Relational Model

Keys

- An example to explain the different keys is −

The Super Keys in **<Student>** table are −

{Student_Number}
{Student_Phone}
{Student_Number,Student_Name}
{Student_Number,Student_Phone}
{Student_Number,Subject_Number}
{Student_Phone,Student_Name}
{Student_Phone,Subject_Number}
{Student_Number,Student_Name,Student_Phone}
{Student_Number,Student_Phone,Subject_Number}
{Student_Number,Student_Name,Subject_Number}
{Student_Phone,Student_Name,Subject_Number}

The Super Keys in **<Enroll>** table are −

{Student_Number,Subject_Number}

Relational Model

Keys

- The Candidate Key in <Student> table is {Student_Number} or {Student_Phone}
- The Candidate Key in <Subject> table is {Subject_Number} or {Subject_Name,Subject_Instructor}
- The Candidate Key in <Student> table is {Student_Number, Subject_Number}
- The Primary Key in <Student> table is {Student_Number}
- The Primary Key in <Subject> table is {Subject_Number}
- The Primary Key in <Enroll> table is {Student_Number, Subject_Number}
- The Composite Key in <Enroll> table is {Student_Number, Subject_Number}
- The Secondary Key in <Student> table is {Student_Phone}
- The Secondary Key in <Subject> table is {Subject_Name,Subject_Instructor}
- {Subject_Number} is the Foreign Key of <Student> table and Primary key of <Subject> table.

Relational Model

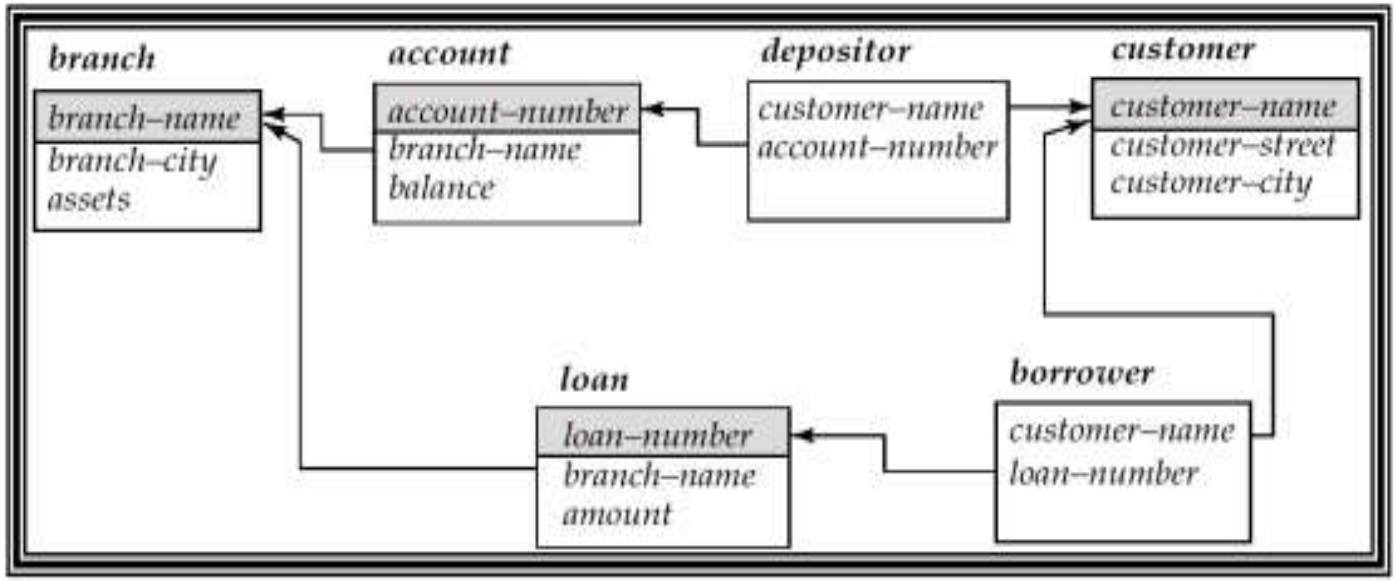
Schema and Schema Diagram

- Relation Schema – list of attributes
- Database Schema – collection of relational schema
- Schema diagram is a graphical representation of database schema along with primary key and foreign key dependencies.
- In schema diagram, each relation is represented by box where attributes are listed inside box and relation name is specified above it.
- Primary key in relation is place above the horizontal line that crosses the box.
- Foreign key in schema diagram appear as arrow from the foreign key attributes of the referencing relation to the primary key of the referenced relation.

Relational Model

Schema Diagram

Now the corresponding schema diagram is given as:



Note: The difference between E-R diagram and schema diagram is, E-R diagram do not shows the foreign key but schema diagram shows it explicitly.

3.3 The Relational Algebra

Query Language

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

** **Assignment** : Difference between Procedural and Non-procedural language.*

3.3 The Relational Algebra

Relational Algebra

- Procedural language
- Basic operators are

Unary operations	Binary operations
▪ Projection (π) ▪ Selection (σ) ▪ Rename (ρ)	▪ JOIN ($\bowtie$) ▪ Division ($\div$) ▪ Union ($\cup$) ▪ Difference (-) ▪ Intersection ($\cap$) ▪ Cartesian Product (x)

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

3.3 The Relational Algebra

Selection operation(σ) : Selects a subset of tuples

- Notation: $\sigma_p(r)$
- p is called the **selection predicate**
- Defined as:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Where p is a formula in propositional calculus consisting of **terms** connected by : $\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
Each **term** is one of:

<attribute> *op* <attribute> or <constant>

where *op* is one of: =, $\neq$, >, $\geq$. <. $\leq$

- Example of selection:

$$\sigma_{branch\_name=\text{“Perryridge”}}(account)$$

3.3 The Relational Algebra

Selection operation Example

- Relation r

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

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

- Resulting relation degree is equal with initial relation degree.
- Resulting relation cardinality is less or equal with initial relation cardinality.

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 1
  1. Selects tuples from books where subject is database.
  2. Select records from books where subject is database and price is Rs 450.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 1

1. $\sigma_{\text{subject=“database”}}$ (Books)
2. $\sigma_{\text{subject=“database” and price = “450”}}$ (Books)

3.3 The Relational Algebra

Projection operation(π): Selects a subset of attributes

- Notation:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

  where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *branch_name* attribute of *account*

$$\Pi_{account\_number,\ balance}(account)$$

3.3 The Relational Algebra

Projection operation Example

- Relation r:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

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

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

- Resulting relation degree is equal with no. of attributes selected.
- Resulting relation cardinality is less or equal with initial relation cardinality.

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 2
  1. Selects or projects columns named as subject and author item relation books.
  2. List all loan numbers and amount of the loan.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 2

1. $\pi_{\text{subject, author}}$ (Books)
2. $\pi_{\text{loan\_number , amount}}$ (Loan)

3.3 The Relational Algebra

Composition of operation

- Can build expressions using multiple operations.
- Example:

  1. $\pi_{\text{Fname,Lname,Salary}}$ ($\sigma_{\text{Dno=5}}$(EMPLOYEE))

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 3
  1. Find those customers name who live in Dhangadhi from Customer relation.
  2. Find the name of subject taught by “Bhaskar” from Course relation.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 3

1. $\pi_{\text{customer\_name}}(\sigma_{\text{customer\_city=“Dhangadhi”}}(\text{Customer}))$
2. $\pi_{\text{subject\_name}}(\sigma_{\text{lecturer=“Bhaskar”}}(\text{Course}))$

3.3 The Relational Algebra

Union operation(∪)

- Notation: $r \cup s$
- Defined as:

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

- For $r \cup s$ to be valid.
  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all customers with either an account or a loan

$$\Pi_{customer\_name}(depositor) \cup \Pi_{customer\_name}(borrower)$$

3.3 The Relational Algebra

Union operation Example

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 3
  1. Find the names of all bank customers who have either an account or a loan or both.
  2. Projects the name of author who have either written a book or an article or both .

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 3

1. First, find the names of all customers with a loan in the bank.

$$\pi_{\text{customer\_name}}(\text{borrower})$$

Also, find the name of all customers with an account in the bank.

$$\pi_{\text{customer\_name}}(\text{depositor})$$

To answer the query, we need the union of these two sets

$$\pi_{\text{customer\_name}}(\text{borrower}) \cup \pi_{\text{customer\_name}}(\text{depositor})$$

2. $\pi_{\text{author}}(\text{Books}) \cup \pi_{\text{author}}(\text{Articles})$

3.3 The Relational Algebra

Set Difference operation(-)

- Notation $r - s$
- Defined as:

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

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible

3.3 The Relational Algebra

Set Difference operation Example

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r - s$:

A	B
α	1
β	1

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 4
  1. Find all customers of the bank who have an account but not a loan.
  2. Results the name of authors who has written books but not articles.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 4

1. $\pi_{customer\_name}(depositor) - \pi_{customer\_name}(borrower)$

2. $\pi_{author}(Books) - \pi_{author}(Articles)$

3.3 The Relational Algebra

Cartesian Product operation(x)

- Notation r x s
- Defined as:

$$r \times s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- Assume that attributes of r(R) and s(S) are disjoint. (That is, $R \cap S = \varnothing$).
- If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.

3.3 The Relational Algebra

Cartesian product operation Example

- Relations *r*, *s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 5
  1. Select all books and articles written by William.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 5

  1. $\sigma_{\text{author = “William”}}$(Books x Articles)

3.3 The Relational Algebra

Rename operation(ρ)

- Allows us to name, and therefore to refer to, the results of relational-algebra expressions.
- Allows us to refer to a relation by more than one name.
- Example:

$$\rho_X(E)$$

returns the expression E under the name X

- If a relational-algebra expression E has arity n, then

$$\rho_{X(A_1,A_2,\ldots,A_n)}(E)$$

returns the result of expression E under the name X, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 6
  1. Rename Name in the relation Student to SName.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 6

  1. $\rho_{\text{Student (Rollno, SName, Address)}}(\text{Student})$

3.3 The Relational Algebra

** Let's do an example.*

- Assume the following relations:

```
BOOKS (DocID, Title, Publisher, Year)
STUDENT(StdID, StName, Major, Age)
AUTHORS(AName, Address)
BORROWS(DocID, StdID, Date)
HAS-WRITTEN(DocID, AName)
DESCRIBES(DocID, Keyword)
```

3.3 The Relational Algebra

** Let's do an example.* *Contd.*

- Write Relational algebra statements for following:

1. List the year and title of each book.
2. List all the information about students whose major is CS.
3. List all students with books they can borrow.
4. List all books published by McGraw-Hill before 1990.
5. List the name of those authors who are living in Davis.
6. List the name of students who are older than 30 and who are not studying CS.
7. Rename AName in the Authors to Name.

3.3 The Relational Algebra

** Assignment*

- Assume the following relations:

```
branch(branch_name,branch_city,assets)
customer(customer_name, customer_street, customer_city)
account(account_number, branch_name, balance)
loan(loan_number, branch_name, amount)
depositor(customer_name, account_number)
borrower(customer_name, loan_number)
```

3.3 The Relational Algebra

** Assignment* *contd.*

- Write Relational algebra statements for following:

1. Find all loans of over $1200.
2. Find the loan number for each loan of an amount greater than $1200.
3. Find the names of all customers who have a loan, an account, or both, from the bank.
4. Find the names of all customers who have a loan at the Perryridge branch.
5. Find the names of all customers who have a loan at the Perryridge branch but do not have an account at any branch of the bank.

3.3 The Relational Algebra

Additional Operations

- Set intersection
- Natural join
- Division
- Assignment

3.3 The Relational Algebra

Set Intersection operation(∩)

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - r, s have the *same arity*
  - attributes of r and s are compatible
- Note: $r \cap s = r - (r - s)$

3.3 The Relational Algebra

Set Intersection operation Example

- Relation *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 7
  1. Find the names of all bank customers who have both an account and a loan.
  2. Projects the name of author who has written a book and an article .

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 7

1. $\pi_{customer\_name}(borrower) \cap \pi_{customer\_name}(depositor)$

2. $\pi_{author}(Books) \cap \pi_{author}(Articles)$

3.3 The Relational Algebra

Natural Join operation(⋈)

- Notation: r ⋈s
- Let *r* and *s* be relations on schemas *R* and *S* respectively.
  Then, r⋈ s is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*
- Example:

  $R = (A, B, C, D)$

  $S = (E, B, D)$
  - Result schema = (*A*, *B*, *C*, *D*, *E*)
  - $r \bowtie s$ is defined as:

$$\Pi_{r.A,\ r.B,\ r.C,\ r.D,\ s.E}\ (\sigma_{r.B = s.B \wedge r.D = s.D}\ (r \times s))$$

3.3 The Relational Algebra

Natural Join operation(⋈) Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	∈

s

- r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

3.3 The Relational Algebra

Division operation(÷)

- Notation: $r \div s$
- Suited to queries that include the phrase “for all”.
- Let *r* and *s* be relations on schemas *R* and *S* respectively where
  - $R = (A_1, \ldots, A_m, B_1, \ldots, B_n)$
  - $S = (B_1, \ldots, B_n)$

  The result of r ÷ s is a relation on schema

  $R - S = (A_1, \ldots, A_m)$

  $$r \div s = \{ t \mid t \in \prod_{R-S}(r) \wedge \forall u \in s (tu \in r) \}$$

  Where *tu* means the concatenation of tuples *t* and *u* to produce a single tuple

3.3 The Relational Algebra

Division operation(÷) Example

- Relations *r*, *s*:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
ϵ	6
ϵ	1
β	2

r

B
1
2

s

- $r \div s$:

A
α
β

3.3 The Relational Algebra

Division operation(÷) Another Example

- Relations *r*, *s*:

A	B	C	D	E
α	a	α	a	1
α	a	γ	a	1
α	a	γ	b	1
β	a	γ	a	1
β	a	γ	b	3
γ	a	γ	a	1
γ	a	γ	b	1
γ	a	β	b	1

r

D	E
a	1
b	1

s

- $r \div s$:

A	B	C
α	a	γ
γ	a	γ

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 8

account(ano,bname,bal)

depositor(cname,ano)

branch(bname,bcity,assets)

1. Find all customers who have account at all branches located at “Dhangadhi”.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 8

  First,

  $R1 \leftarrow \pi_{bname}(\sigma_{bcity=\text{“Dhangadhi”}}(branch))$

  Then,

  $R2 \leftarrow \pi_{cname,bname}(depositor \bowtie account)$

  Final Query

  $R2 \div R1$

3.3 The Relational Algebra

Assignment operation(←)

- The assignment operation (←) provides a convenient way to express complex queries.
  - Write query as a sequential program consisting of
    - a series of assignments
    - followed by an expression whose value is displayed as a result of the query.
  - Assignment must always be made to a temporary relation variable.
  - The result to the right of the ← is assigned to the relation variable on the left of the ←.
  - May use variable in subsequent expressions.

3.3 The Relational Algebra

** Lets solve some problems.*

- Problem 9
  1. Find all customers who have account but no loan.

3.3 The Relational Algebra

** Here is solution of previous problems.*

- Solution 9

  First,

  $R1 \leftarrow \pi_{cname}(depositor)$

  Then,

  $R2 \leftarrow \pi_{cname}(borrower)$

  Final Query

  R1 - R2

3.3 The Relational Algebra

Extended Relational-Algebra-Operations

- Generalized Projection
- Aggregate Functions
- Outer Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
- Inner Join
  - Theta Join
  - Equi Join
  - Natural Join (We have already discussed)

3.3 The Relational Algebra

Generalized Projection(π)

- Extends the projection operation by allowing arithmetic functions to be used in the projection list.

$$\prod_{F_1, F_2, \ldots, F_n}(E)$$

- E is any relational-algebra expression
- Each of F_1, F_2, …, F_n are are arithmetic expressions involving constants and attributes in the schema of E.
- Given relation *credit_info(customer_name, limit, credit_balance)*, find how much more each person can spend:

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

3.3 The Relational Algebra

Aggregate Functions ($\mathcal{G}$)

- **Aggregation function** takes a collection of values and returns a single value as a result.

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

- **Aggregate operation** in relational algebra

$$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2,\ldots,F_n(A_n)}(E)$$

  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

3.3 The Relational Algebra

Aggregate Functions (g) Example

- Relation r:

A	B	C
α	α	7
α	β	7
β	β	3
β	β	10

- $\mathcal{G}_{\text{sum(c)}}(\text{r})$

sum(c)
27

3.3 The Relational Algebra

Aggregate Functions (g) Another Example

- Relation *account* grouped by *branch-name*:

branch_name	*account_number*	*balance*
Perryridge	A-102	400
Perryridge	A-201	900
Brighton	A-217	750
Brighton	A-215	750
Redwood	A-222	700

$$branch\_name\ g\ \mathbf{sum}(balance)\ (account)$$

branch_name	**sum**(*balance*)
Perryridge	1300
Brighton	1500
Redwood	700

3.3 The Relational Algebra

Aggregate Functions ($\mathcal{g}$)

- Result of aggregation does not have a name
  - Can use rename operation to give it a name
  - For convenience, we permit renaming as part of aggregate operation

$$branch\_name\ \mathcal{g}\ \textbf{sum}(balance)\ \textbf{as}\ sum\_balance\ (account)$$

3.3 The Relational Algebra

Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist
  - All comparisons involving *null* are (roughly speaking) **false** by definition.
    - We shall study precise meaning of comparisons with nulls later

3.3 The Relational Algebra

Outer Join Examples

- Relation *loan*

loan_number	*branch_name*	*amount*
L-170	Downtown	3000
L-230	Redwood	4000
L-260	Perryridge	1700

- Relation *borrower*

customer_name	*loan_number*
Jones	L-170
Smith	L-230
Hayes	L-155

3.3 The Relational Algebra

Outer Join Examples

- Join

  $loan \bowtie borrower$

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith

- Left Outer Join

  $loan \mathbin{⟕} borrower$

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*

3.3 The Relational Algebra

Outer Join Examples

- Right Outer Join

  loan ⟖ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-155	*null*	*null*	Hayes

- Full Outer Join

  loan ⟗ *borrower*

loan_number	*branch_name*	*amount*	*customer_name*
L-170	Downtown	3000	Jones
L-230	Redwood	4000	Smith
L-260	Perryridge	1700	*null*
L-155	*null*	*null*	Hayes

3.3 The Relational Algebra

NULL Values

- It is possible for tuples to have a null value, denoted by *null*, for some of their attributes
- *null* signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving *null* is *null.*
- Aggregate functions simply ignore null values (as in SQL)
- For duplicate elimination and grouping, null is treated like any other value, and two nulls are assumed to be the same (as in SQL)

3.3 The Relational Algebra

Inner Join

- In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded.

3.3 The Relational Algebra

Theta Join (θ)

- Theta join combines tuples from different relations provided they satisfy the theta condition.
- Notation : $R1 \bowtie_{\theta} R2$
- R1 and R2 are relations having attributes (A1, A2, .., An) and (B1, B2,.. ,Bn) such that the attributes don't have anything in common, that is $R1 \cap R2 = \Phi$.
- Theta join can use all kinds of comparison operators.

3.3 The Relational Algebra

Theta Join (θ) Example

Student		
SID	Name	Std
101	Alex	10
102	Maria	11

Subjects	
Class	Subject
10	Math
10	English
11	Music
11	Sports

3.3 The Relational Algebra

Theta Join (θ) Example

Student_Detail –

STUDENT $\bowtie_{Student.Std = Subject.Class}$ SUBJECT

Student_detail				
SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

3.3 The Relational Algebra

Equi Join

- When Theta join uses only equality comparison operator, it is said to be equijoin.
- The previous example corresponds to equijoin.

3.3 The Relational Algebra

Modification of the Database

- The content of the database may be modified using the following operations:
  - Deletion
  - Insertion
  - Updating
- All these operations are expressed using the assignment operator

3.3 The Relational Algebra

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.

3.3 The Relational Algebra

Deletion Examples

- Delete all account records in the Perryridge branch.

$$account \leftarrow account - \sigma_{branch\_name = \text{“Perryridge”}}(account)$$

- Delete all loan records with amount in the range of 0 to 50

$$loan \leftarrow loan - \sigma_{amount \geq 0 \text{ and } amount \leq 50}(loan)$$

3.3 The Relational Algebra

Insertion

- To insert data into a relation, we either:
  - specify a tuple to be inserted
  - write a query whose result is a set of tuples to be inserted
- in relational algebra, an insertion is expressed by:

$$r \leftarrow r \cup E$$

  where r is a relation and E is a relational algebra expression.
- The insertion of a single tuple is expressed by letting E be a constant relation containing one tuple.

3.3 The Relational Algebra

Insertion Examples

- Insert information in the database specifying that Smith has $1200 in account A-973 at the Perryridge branch.

$$account \leftarrow account \cup \{(\text{“A-973”}, \text{“Perryridge”}, 1200)\}$$

$$depositor \leftarrow depositor \cup \{(\text{“Smith”}, \text{“A-973”})\}$$

3.3 The Relational Algebra

Updating

- A mechanism to change a value in a tuple without charging *all* values in the tuple
- Use the generalized projection operator to do this task

$$r \leftarrow \prod_{F_1, F_2, \ldots, F_l}(r)$$

- Each F_i is either
  - the I^{th} attribute of r, if the I^{th} attribute is not updated, or,
  - if the attribute is to be updated F_i is an expression, involving only constants and the attributes of r, which gives the new value for the attribute

3.3 The Relational Algebra

Updating Examples

- Make interest payments by increasing all balances by 5 percent.

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

- Pay all accounts with balances over $10,000 6 percent interest and pay all others 5 percent

$$account \leftarrow \prod_{account\_number,\ branch\_name,\ balance\ *\ 1.06} (\sigma_{BAL > 10000} (account)) \cup \prod_{account\_number,\ branch\_name,\ balance\ *\ 1.05} (\sigma_{BAL \leq 10000} (account))$$

Relational Algebra

Views

- A view is a virtual table derived from one or more tables or other views.
- Views allow you to hide data or limit access to a select number of columns; therefore, they can also be used for security purposes.
- A view can be used for the following purposes:
  - Provides user security functions.
  - Simplifies the constructions of complex queries.
  - Summarize data from multiple tables.

Relational Algebra

Views definition

- View is defined by using the create view statement.
- The general structure is

  ```
  Create view <view name> as <query expression>
  ```

  where <query expression> is any legal relational-algebra query expression.
- Once a view is defined, it can refer by its virtual name, called view name.

Relational Algebra

Views Examples

- Create a view “all-customer” consisting branches and their customers.

Create view all-customer as

$$\Pi_{branch\_name,customer\_name}(depositor \bowtie account)$$

$$\cup\ \Pi_{branch\_name,customer\_name}(borrower \bowtie loan)$$

We can query on this view as in other relation.

For example,

Query: find all customer of “B1” branch.

$$\Pi_{customer\_name}(\sigma_{branch\_name=”B1”}(all\text{-}customer))$$

This is the end of the lecture!
I hope you enjoyed it.
Thank You