

Using High-Level Conceptual Data Models for Database Design:-

once the requirements have been collected and analyzed, the next step is to create a Conceptual Schema for the database, using a high-level conceptual data model. This step is called Conceptual design. The Conceptual Schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships and constraints.

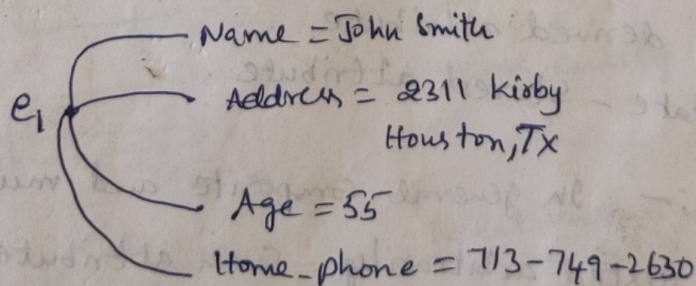
During or after the Conceptual Schema design, the basic data model operations can be used to specify the high level user queries and operations identified during functional analysis.

Entity Types, Entity Sets, Attributes and Keys:-

An entity is a thing in the real world with an independent existence. It may be an object with a physical existence or it may be an object with a conceptual existence.
 ↳ person, car
 ↳ company, course
 Each entity has attributes - the particular properties that describe it.

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

Eg:-



Several types of attributes occur in the ER Model

Simple versus Composite

Single valued versus multivalued

Stored versus derived.

Composite Versus Simple attributes:-

Composite attributes can be divided into smaller subparts.

For eg, the Address attribute of the Employee entity shown above can be subdivided into Street-address, City, State and Zip.

Attributes that are not divisible are called simple or atomic attributes.

The value of a composite attribute is the concatenation of the values of its component simple attributes.

Single valued versus multivalued attributes:-

The attributes that have a single value for a particular entity are known as Single-valued attributes. Eg:- Age.

The attributes that have a set of values for the same entity are known as multi-valued attributes. A multivalued attribute may have lower and upper bounds to constrain the number of values allowed for each individual entity.

eg:- Colors attribute of a Car - lower bound - 1 upper bound - 3.
degrees attribute of a person.

Stored versus Derived attributes:-

A derived attribute is an attribute whose values can be derived from another attribute called stored attribute.

eg:- age - derived attribute
birthdate - stored attribute.

Complex Attributes:- In general composite and multivalued attributes can be nested arbitrarily. Such attributes are called

Complex attributes.

Eg:- Address-Phone.

{ Address-Phone ({ Phone (Area-Code, Phone-no),
Address (street-address (no, street, Apt-no),
city, state, zip)) }) }.

Entity Types and Entity sets:-

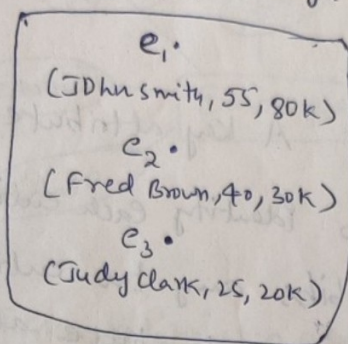
4-3

An entity type defines a collection of entities that have the same attributes. Each entity type in the database is described by its name and attributes.

Eg:-

Employee

Name, Age, Salary



The collection of all entities of a particular entity type in the database at any point in time is called an entity set.

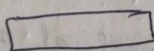
An entity type describes the schema or intension for a set of entities that share the same structure.

An entity set is also called an extension of the entity type.

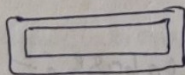
Symbols of ER diagram:-

Symbol

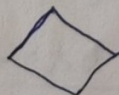
Meaning



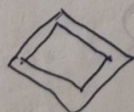
Entity



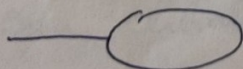
Weak entity



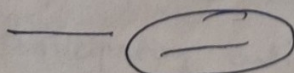
Relationship



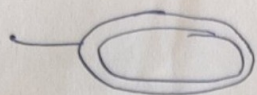
Identifying relationship



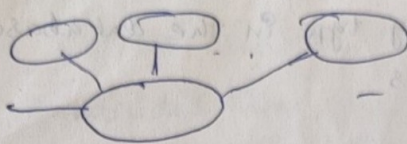
attribute



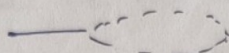
Key attribute



- Multivalued attribute



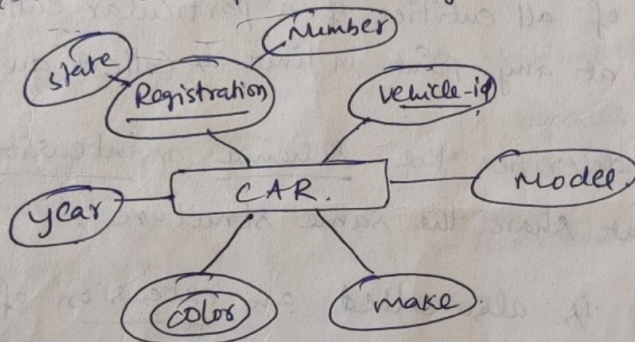
- Composite attribute



- Derived attribute

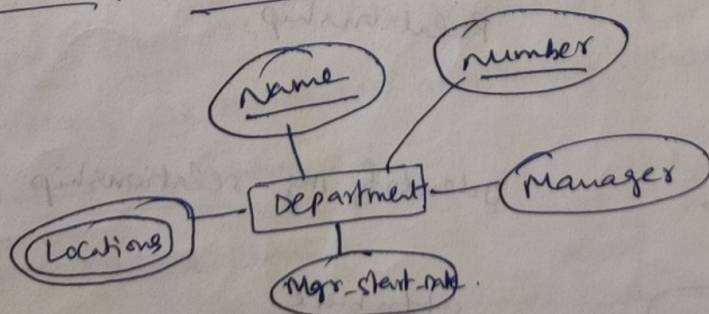
Key attributes of an Entity type:- A key attribute is an attribute which can be used to identify each entity uniquely. It is a constraint that prohibits any two entities from having the same value for the key attribute at the same time.

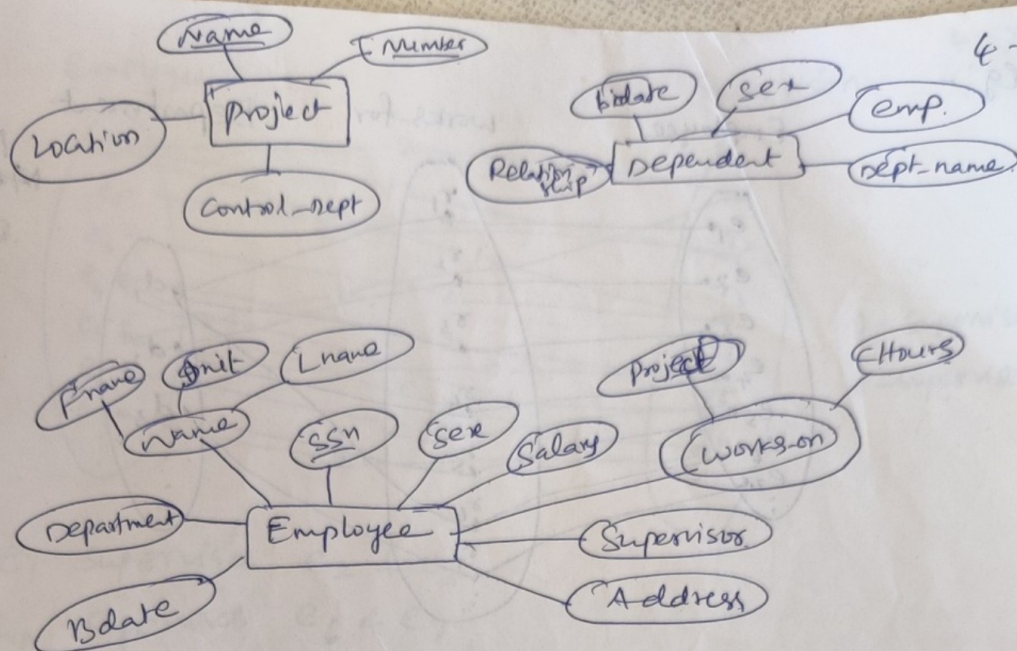
Eg:-



Value sets of Attributes:- Each simple attribute of an entity type is associated with a value set or domain of values, which specifies the set of values that may be assigned to that attribute for each individual entity.

Initial Conceptual Design of the Company Database:-





Relationship Types, Relationship sets, Roles and structural Constraints:-

In the above dig's there are several implicit relationships among the various entity types. whenever an attribute of one entity type refers to another entity type, some relationship exists.

eg:- The attribute Manager of Department refers to an employee who manages the department.

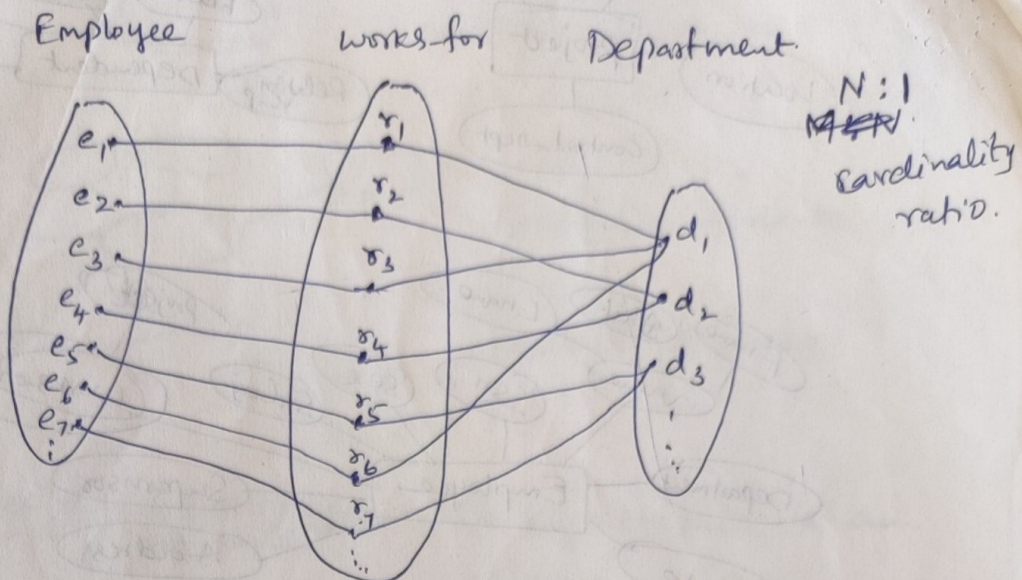
In ER model these references should be represented as relationships.

A relationship type R among n entity types $E_1, E_2, \dots, E_n$ defines a set of associations - or a relationship set - among entities from these entity types.

A relationship set R is a set of relationship instances r_i , where each r_i associates n individual entities $(e_1, e_2, \dots, e_n)$ and each entity e_j in r_i is a member of entity set E_j , $1 \leq j \leq n$.

Informally, each relationship instance r_i in R is an association of entities, where the association includes exactly one entity from each participating entity type.

Eg:-



The degree of a relationship type is the number of participating entity types.

For the above eg work_for relationship degree is 2

A relationship type of degree two is called binary and one of degree three is called ternary.

Role names :- Each entity type that participates in a relationship type plays a particular role in the relationship.

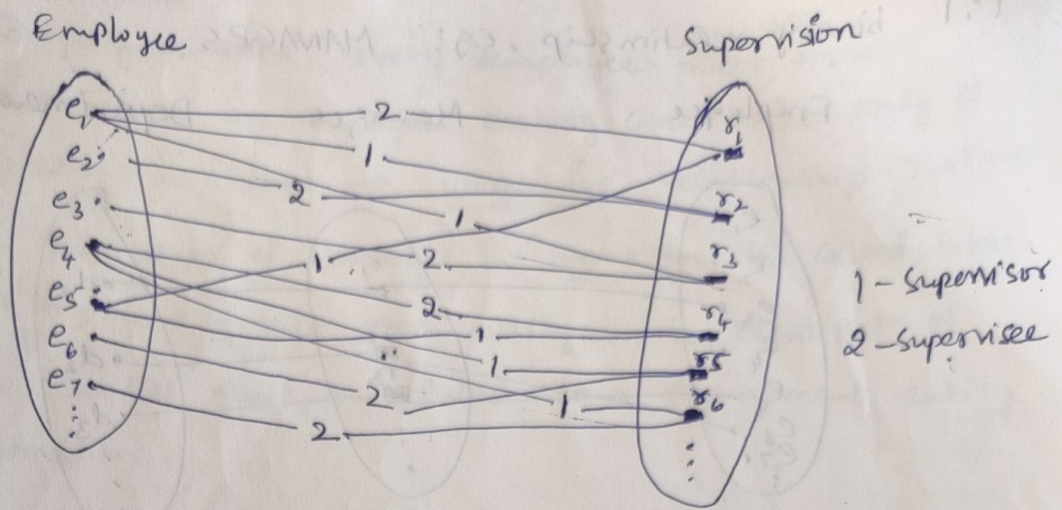
The role name signifies the role that a participating entity from the entity type plays in each relationship instance and helps to explain what the relationship means.

Eg:- Employee - employee is works Dept - Department or employee
In some cases the same entity type participates more than

once in a relationship type in different roles. In such cases the rolename becomes essential for distinguishing the meaning of the role that each participating entity plays. Such relationship types are called recursive relationships.

Eg:- The Supervision relationship type relates an employee to a supervisor, where both employee and supervisor entities are members of the same Employee entity set.

Hence, Employee entity type participates twice in Supervision
once as supervisor (boss), once as supervisee (Subordinate)



e_1 Supervises e_2 & e_3 .
 e_4 Supervises e_6 & e_7 .
 e_5 " e_1 & e_4 .

Constraints on Binary Relationship Types: - Relationship types have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set. The two main types of binary relationship constraints are: Cardinality ratio and Participation.

The Cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.

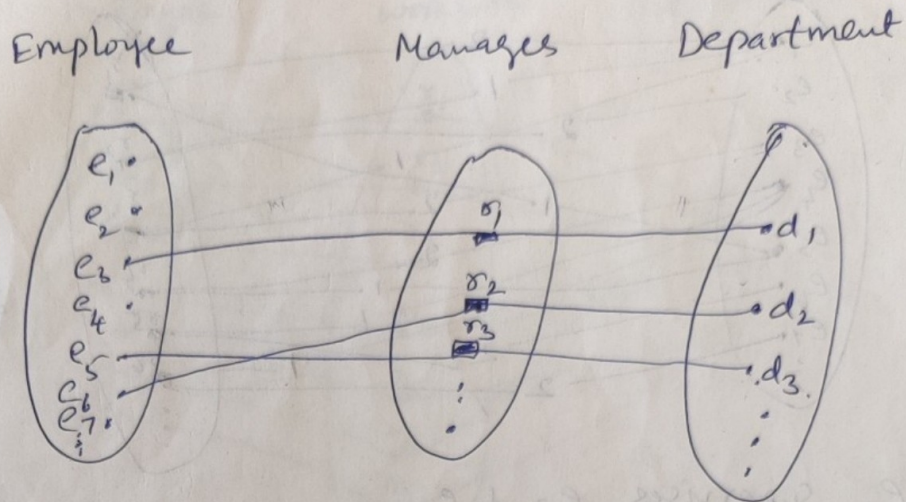
For eg, in the works-for binary relationship type, the Cardinality ratio is $N:1$, meaning that each department can be related to any no. of employees, but an employee can be related to only one department.

The other Cardinality ratios for binary relationship types are: $1:1$, $1:N$, $N:1$ and $M:N$.

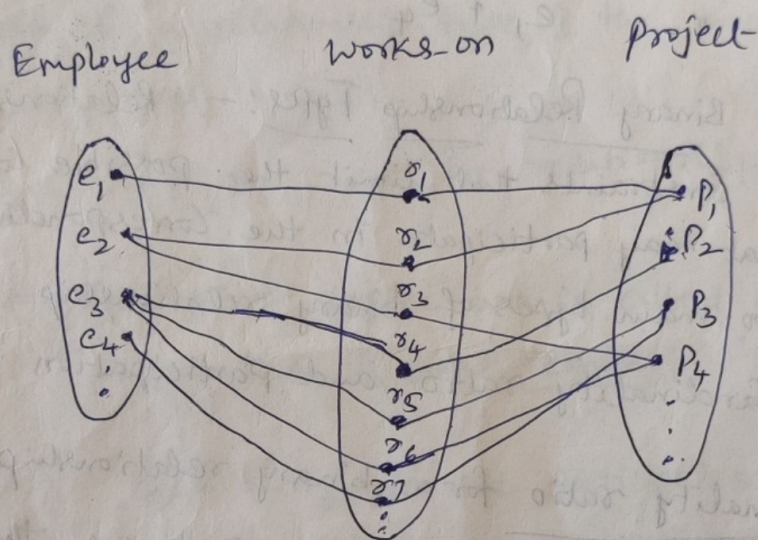
For eg, in the works-for binary relationship type,

Department : Employee is of Cardinality ratio $1:N$, meaning that each dept can be related to any no. of employees, but an employee can be related to only one department.

1.1 binary relationship, eg:- MANAGES



M:N binary relationship eg:- WORKS-ON



An employee can work on several projects and a project can have several employees.

Participation Constraints and Existence Dependencies:- The Participation Constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This Constraint specifies the minimum number of relationship instances that each entity can participate in, and is some times called the minimum Cardinality Constraint.

Two types of participation Constraints: total and partial.

Total Participation is also called existence dependency.

total Participation:-

eg. If for Company database, every ^{work} employee must ^{for} a department, then an employee entity can exist only if it participates in at least one works-for relationship instance.

Thus the participation of Employee in works-for is called total Participation, meaning that every entity in the total set of employee entities must be related to a department entity via works-for.

eg:- partial participation:-

2. If for Company database, if it is not must for every employee to manage a department, then the participation of Employee

in the manages relationship type is partial, meaning that some part of the set of employee entities are related to some department via manages, but not necessarily all.

The Cardinality ratio and participation constraints together can be referred as structural constraints of a relationship type.

Attributes on relationship types:- Relationship types can also have attributes.

eg:- Hours attribute for works-on relationship type
start-date attribute for manages relationship type.

Attributes of 1:1 or 1:N relationship types can be migrated to one of the participating entity types.

eg:- start-date ~~can be~~ of manages relationship can be migrated ~~as~~ either employee or department entity attribute, because it is a 1:1 relationship type.

For 1:N relationship types, a relationship attribute can be migrated only to the entity type on the N-side of the relationship. eg:- works-for relationship b/w Employee & Dept. start-date attribute for works-for can be migrated to employee entity.

5-1 For M:N relationship types, some attributes may be determined by the combination of participation entities in a relationship instance, not by any single entity. Such attributes must be specified as relationship attributes.

Eg:- Hours attributes of works-on

Weak entity types :- Entity types that do not have key attributes of their own are called weak entity types.

Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.

Such another entity type is called the identifying or owner entity type, and the relationship type that relates a weak entity^{type} to its owner is called the identifying relationship of the weak entity type.

A weak entity type always has a total participation constraint w.r.t its identifying relationship.

Eg:- Dependent entity of the Company database.

Each employee entity is said to own the dependent entities related to it.

Dependent entity attributes are - name, salary, sex, ship

A weak entity type has a partial key, which is the attribute that can uniquely identify weak entities that are related to the same owner entity.

Eg:- name is the partial key of dependent.

In the worst case, a composite attribute of all the weak entity's attributes will be the partial key.

Weak entity types can sometimes be represented as complex attributes.

Eg:- dependent attribute for employee entity.

Refining the ER design for the Company Database :-

4-11

Fig: 7.2
Pg no: 202.

The database design for Company database can be refined by changing the attributes that represent the relationships into relationship types.

The relationship types are as follows:

- Manages, a 1:1 relationship type between Employee and Department. start-date is the attribute assigned to this relationship type. Employee participation is partial.
- works-for, a 1:N relationship type between Department and Employee.

Both entities participations are total.

- Controls, a 1:N relationship type between Department and project.

Project participation is total

Department participation is partial

- Supervision, a 1:N relationship type between Employee (supervisor) and Employee (supervisee)

Both entities participations are partial

- works-on, a M:N relationship type with attribute Hours, between Employee and project.

Both entities participations are total.

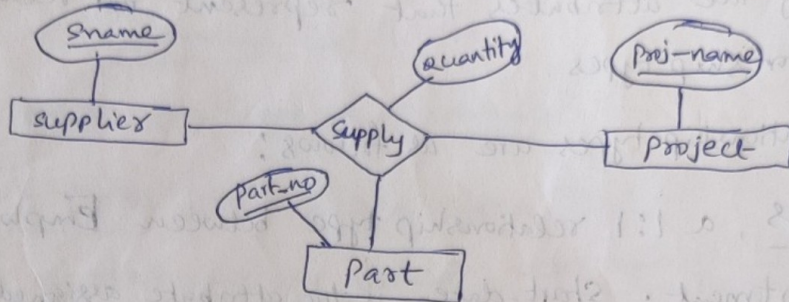
- Dependents-of, a 1:N relationship type between Employee and Dependent, which is also the identifying relationship for the weak entity type Dependent.

Employee participation is partial.

Dependent participation is total.

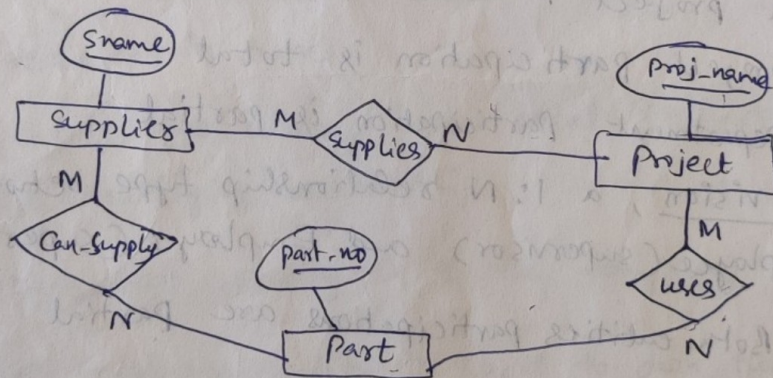
Relationship types of Degree higher than two :-

Ternary relationship example



The relationship set of Supply is a set of relationship instances (S, J, P) , where S is a Suppliers who is currently supplying a part P to project J .

Three binary relationship types example.



Can-supply — instance (S, P) — S^{Can} Supplier, P^{Can} Part P
 uses — instance (J, P) — Project J uses Part P .
 supplies — instance (S, J) — Supplier S supplies some part to Project J .

It is often difficult to decide whether a particular relationship should be represented as a relationship type of degree n or should be broken down into several relationship types of smaller degrees.

Some database design tools permit only binary relationships. In this case a ternary relationship such as Supply must be

Relational Database design using ER-to-Relational Mapping :-

Step 1: Mapping of Regular Entity types :- For each regular (strong) entity type E in the ER schema, Create a relation R that includes all the simple attributes of E and simple Component attributes of a Composite attribute. Choose one of the key attributes of E as the primary key for R .

Eg:- Employee.

fname	lname	<u>ssn</u>	bdate	address	sex	salary
-------	-------	------------	-------	---------	-----	--------

Department

dname	<u>dnumber</u>
-------	----------------

project

pname	<u>pnumber</u>	plocation
-------	----------------	-----------

Step 2: Mapping of weak entity types :- For each weak entity type w in the ER schema with owner entity type E , Create a relation R and include all simple attributes of w as attributes of R . In addition, include as foreign key attributes of R , the primary key attributes of the relation that correspond to the owner entity type. The PK of R is the combination of the PK of the owner and the partial key of the weak entity type w , if any.

Eg:- Dependent

<u>Essn</u>	<u>Dependent-name</u>	sex	bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Step 3: Mapping of Binary 1:1 Relationship Types :- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R .

Three possible approaches:

4-15

1. Foreign key approach
2. Merged relationship approach
3. Cross reference or relationship approach

Foreign key approach: - Choose one of the relations-S, say and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes of the 1:1 relationship type R as attributes of S.

Eg: - Manages - relationship

Department entity - S - total participation

The primary key of the Employee relation as foreign key in the Department relation. Rename it as mgr-ssn.

The start-date attribute of the Manages relationship is included in the Department relation and rename it as mgr-start-date.

Department

Dname	Dnumber	mgr-ssn	mgr-start-date
-------	---------	---------	----------------

Merged relation approach: - This approach is to merge the two entity types and the relationship into a single relation. This is possible when both participations are total.

Cross-reference or relationship relation approach: - This approach is to set up a third relation R for the purpose of cross referencing the primary keys of the two relations S and T representing the entity types. The relation R is called a relationship relation.

Step 4: Mapping of Binary 1:N Relationship types: - For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the 1-side of the relationship type

4-16
Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R .

Eg:- 1:N relationship types - works-for, controls & supervision.

For works-for - PK dno of the dept relation - fk in employee

For supervision - PK ssn of the emp relation - fk in emp itself

For controls - PK dnumber of the dept relation - fk in project.

Step 2: Mapping of Binary M:N relationship types :- For each binary M:N relationship type R , create a new relation S to represent R . Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Include the simple attributes of the M:N relationship type as attributes of S .

Eg:- M:N relationship type - works-on between - project & employee

works-on

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

Step 6: Mapping of Multivalued attributes:- For each multivalued attribute A , create a new relation R which includes an attribute corresponding to A plus the PK attribute K - as $\{K\}$ in R . The PK of R is the combination of A and K .

Eg:- relation - Dept-locations

Dept-locations

<u>Dnumber</u>	<u>Dlocations</u>
----------------	-------------------

PK of Dept-locations is

the combination of {Dnumber, Dlocation}

Step 7: Mapping of N-ary Relationship Types :- For each n-ary

relationship type R , where $n > 2$, create a new relation

S to represent R . Include as fk attributes in S the PKs

of the relations that represent the participating entity types.

Also include any simple attributes of the n-ary relationship type as attributes of S .

Eg:- Supply - PK (sname, partno, projname)