UNIT -2

Entity-Relationship Model

Introduction to ER Model

ER model is represents real world situations using concepts, which are commonly used by people. It allows defining a representation of the real world at logical level.ER model has no facilities to describe machine-related aspects.

In ER model the logical structure of data is captured by indicating the grouping of data into entities. The ER model also supports a top-down approach by which details can be given in successive stages.

Entity: An entity is something which is described in the database by storing its data, it may be a concrete entity a conceptual entity.

Entity set: An entity set is a collection of similar entities.

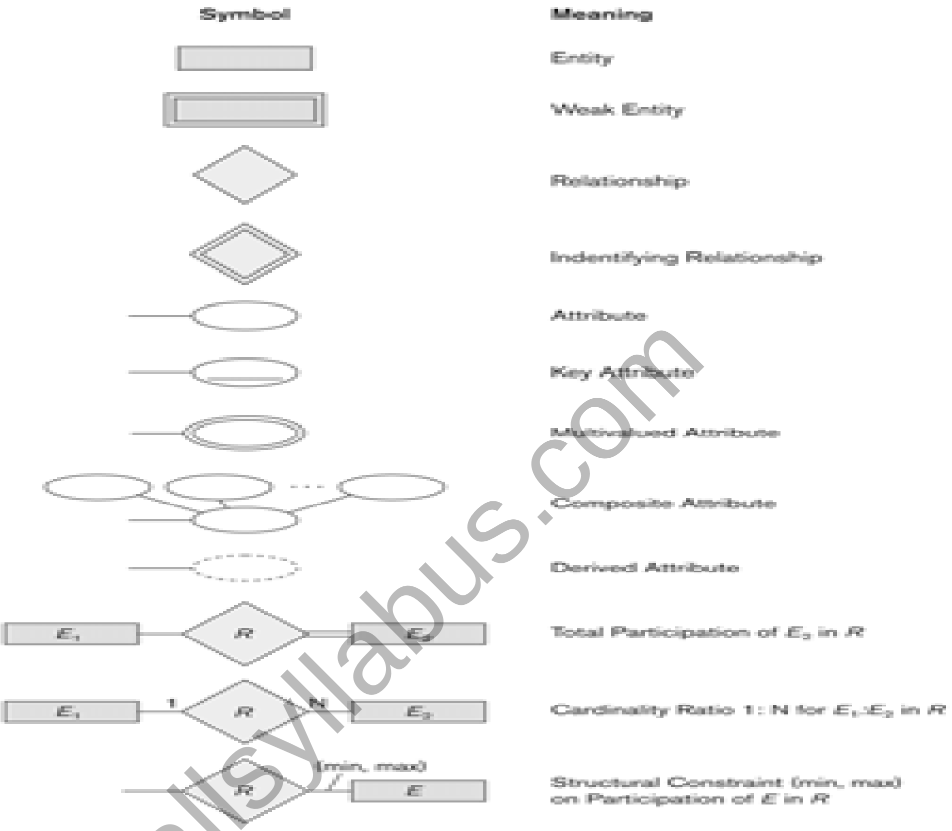
Attribute: An attribute describes a property associated with entities. Attribute will have a name and a value for each entity.

Domain: A domain defines a set of permitted values for a attribute

SYMBOLS IN E-R DIAGRAM

The ER model is represented using different symbols as shown in Fig .a

Figure 3.14
Summary of the notation for ER diagrams.


Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E1
R
E2
Total Participation of E2 in R
E1
1
R
N
E2
Cardinality Ratio 1: N for E1:E2 in R
(min, max)
R
E
Structural Constraint (min, max) on Participation of E in R

Overview of Database Design Process

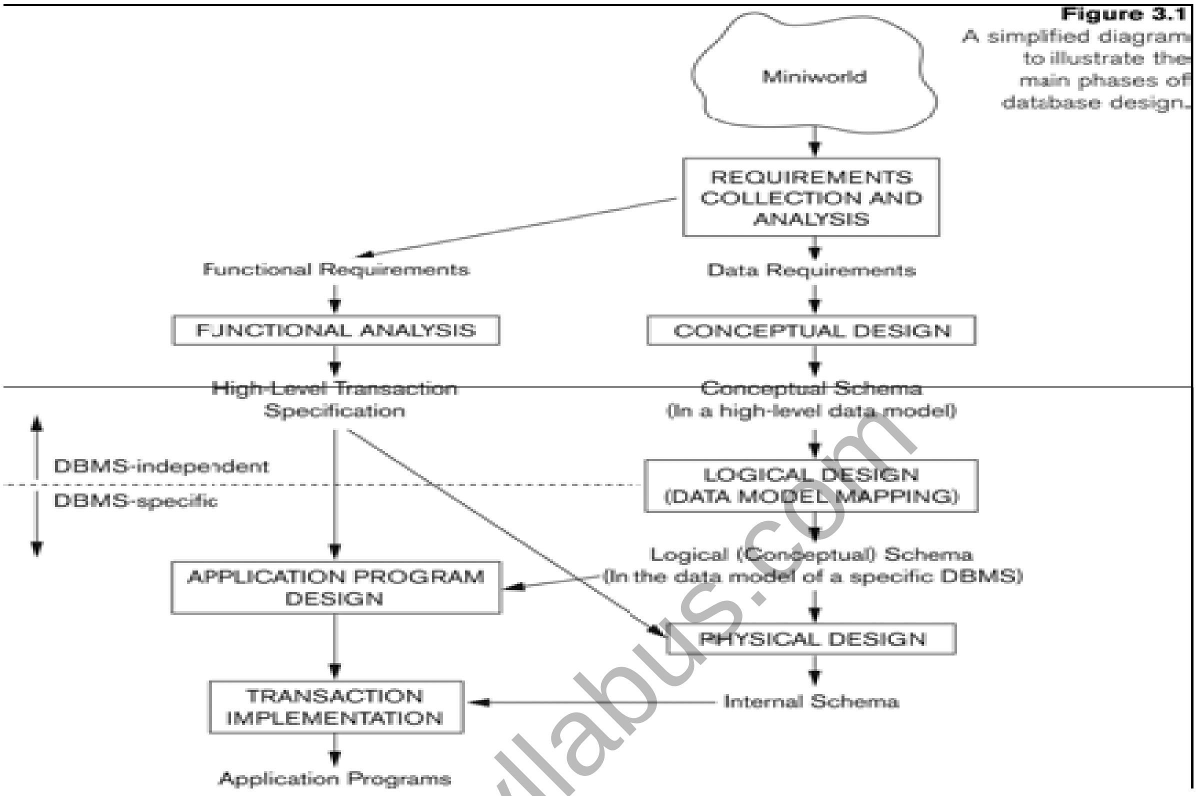


Figure 3.1 A simplified diagram to illustrate the main phases of database design.

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Example COMPANY Database

We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

The company is organized into DEPARTMENTs.

Each department has a name, number and an employee who *manages the department.*

We keep track of the start date of the department manager.

A department may have several locations.

Each department *controls a number of*

PROJECTs. Each project has a unique name, unique number and is located at a single location.

We store each EMPLOYEE's social security number, address, salary, sex, and birth date.

Each employee *works for one department but may work on several projects.*

We keep track of the number of hours per week that an employee currently works on each project.

We also keep track of the *direct supervisor of each* employee.

Each employee may *have a number of* DEPENDENTs.

For each dependent, we keep track of their name, sex, birth date, and relationship to the employee.

ER Model Concepts

Entities and Attributes

Entities are specific objects or things in the mini-world that are represented in the database.

For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT.

Attributes are properties used to describe an entity.

For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate .

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

For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'

Each attribute has a *value set (or data type) associated with* it – e.g. integer, string, subrange, enumerated type,

Types of Attributes

There are two types of Attributes

Simple

Each entity has a single atomic value for the attribute.
For example, SSN or Sex.

Composite

The attribute may be composed of several components. For example:
Address(Apt#, House#, Street, City, State, ZipCode, Country), or Name(FirstName, MiddleName, LastName).
Composition may form a hierarchy where some components are themselves composite.

Multi-valued

An entity may have multiple values for that attribute. For example, Color of a CAR or Previous Degrees of a STUDENT.
Denoted as {Color} or {Previous Degrees}.
In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.
For example, Previous Degrees of a STUDENT is a composite multi-valued attribute denoted by

{Previous Degrees (College, Year, Degree, Field)}

Multiple Previous Degrees values can exist. Each has four subcomponent attributes:

College, Year, Degree, Field

Example of a composite attribute

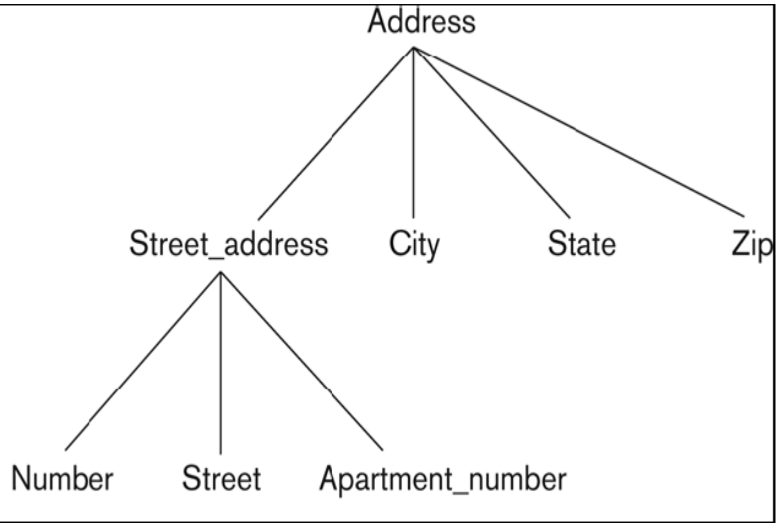


Figure 3.4
A hierarchy of composite attributes.

Entity Types and Key Attributes

Entities with the same basic attributes are grouped or typed into an entity type.
For example, the entity type EMPLOYEE and PROJECT.

An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
For example, SSN of EMPLOYEE.

A key attribute may be composite.
Vehicle Tag Number is a key of the CAR entity type with components (Number, State).

An entity type may have more than one key.
The CAR entity type may have two keys:

VehicleIdentificationNumber (popularly called VIN)
VehicleTagNumber (Number, State), license plate number.

Each key is underlined

Displaying an Entity type

In ER diagrams, an entity type is displayed in a rectangular box
Attributes are displayed in ovals.
Each attribute is connected to its entity type
Components of a composite attribute are connected to the oval representing the composite attribute.
Each key attribute is underlined.
Multivalued attributes displayed in double ovals.

Entity Type CAR with two keys and a corresponding Entity Set

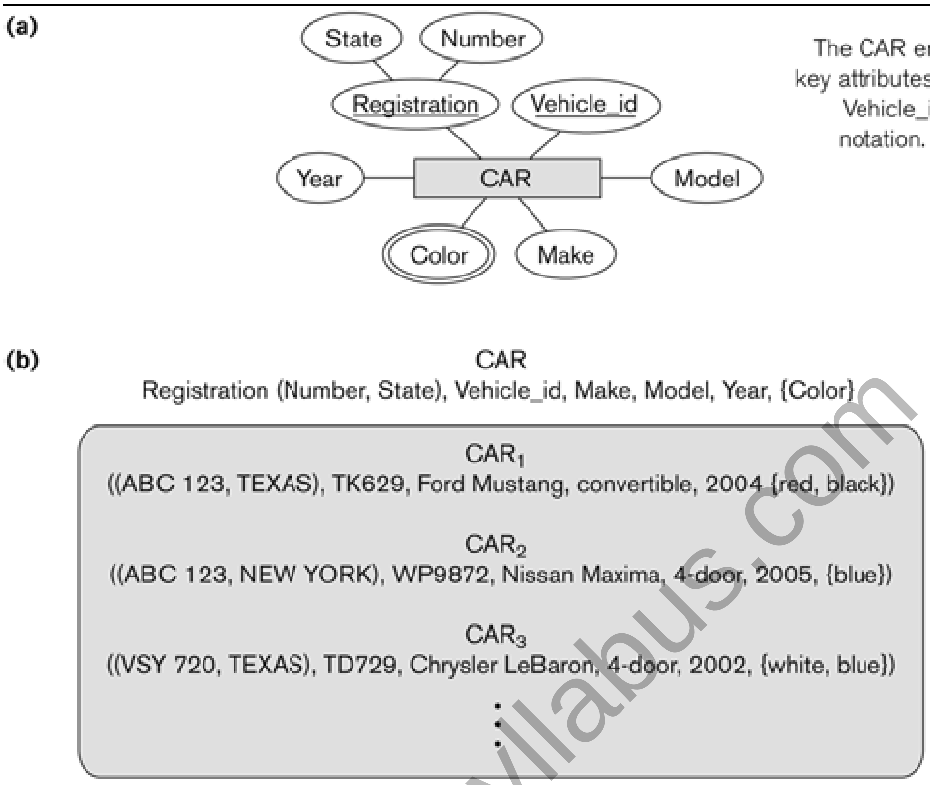

(a)
State
Number
Registration
Vehicle_id
Year
CAR
Model
Color
Make
(b)
CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}
CAR$_1$
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})
CAR$_2$
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})
CAR$_3$
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})
⋮

Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

Entity Set

Each entity type will have a collection of entities stored in the database Called the **entity set.**

The above example shows three CAR entity instances in the entity set for CAR

Same name (CAR) used to refer to both the entity type and the entity set.

Entity set is the current *state of the entities of that*type that are stored in the database.

Initial Design of Entity Types for the COMPANY Database Schema

Based on the requirements, we can identify four initial entity types in the COMPANY database:

DEPARTMENT

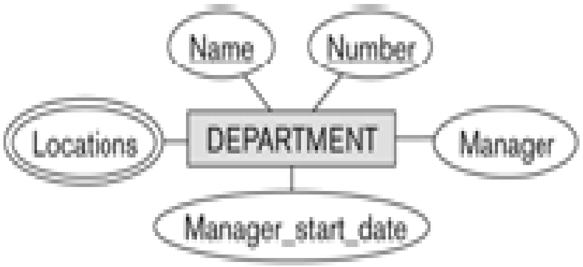
PROJECT

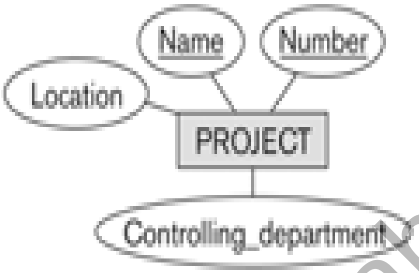
EMPLOYEE

DEPENDENT

Their initial design is shown below.

The initial attributes shown are derived from the requirements description





Initial Design of Entity Types for the COMPANY Database Schema

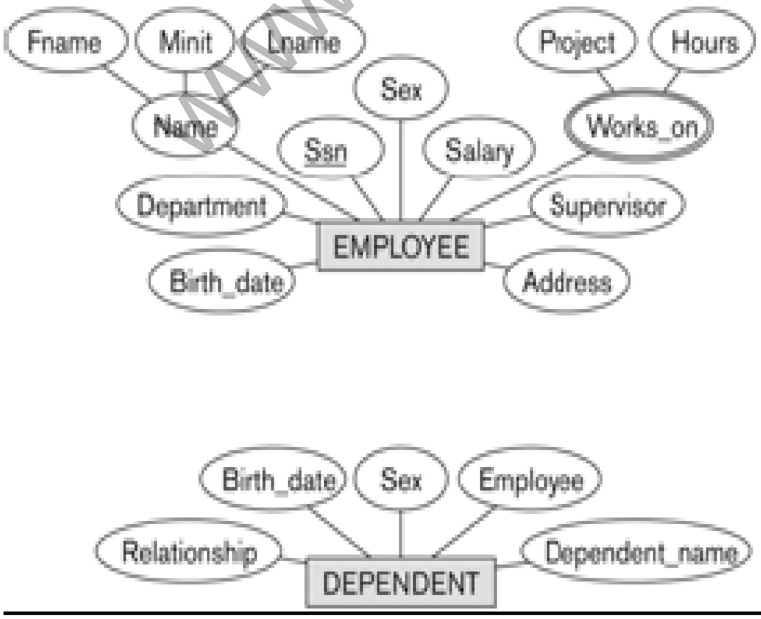


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining the initial design by introducing relationships

The initial design is typically not complete. Some aspects in the requirements will be represented as relationships.

ER model has three main concepts:
Entities (and their entity types and entity sets)
Attributes (simple, composite, multi valued)
Relationships (and their relationship types and relationship sets)

Relationships and Relationship Types

A **relationship relates two or more distinct entities with a** specific meaning.
For example, EMPLOYEE John Smith *works on the ProductX* PROJECT, or EMPLOYEE Franklin Wong *manages the*Research DEPARTMENT.

Relationships of the same type are grouped or typed into a **relationship type.**
For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

The degree of a relationship type is the number of participating entity type.
Both MANAGES and WORKS_ON are *binary relationships.*

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

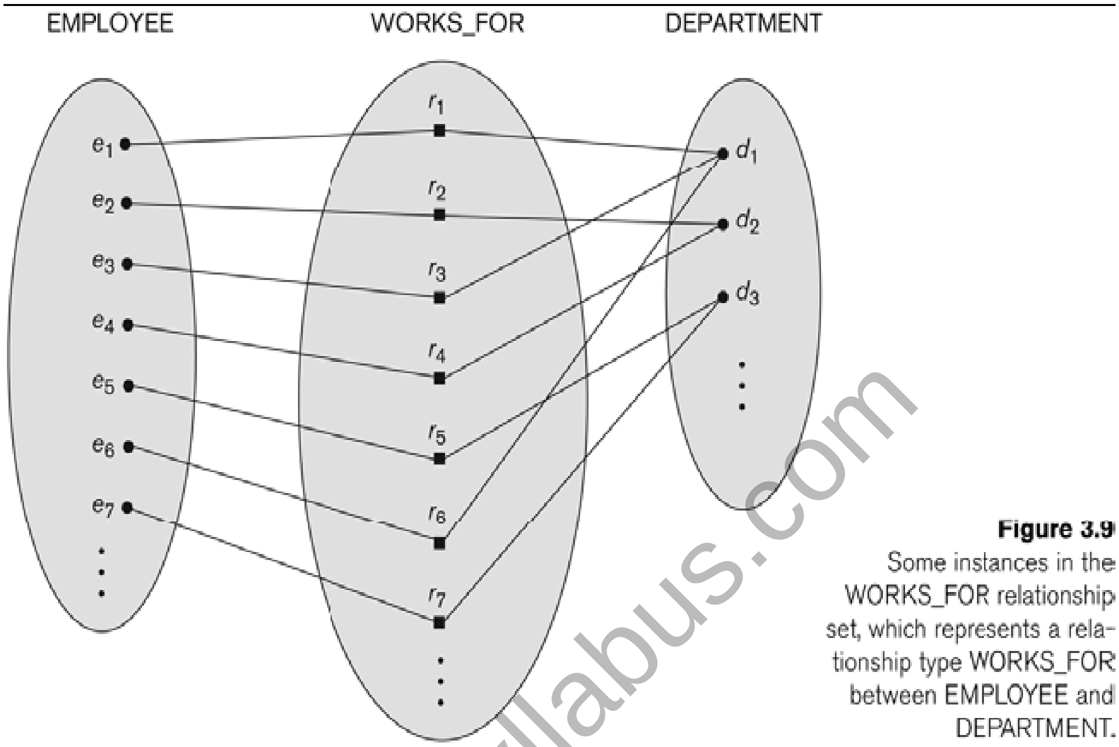


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

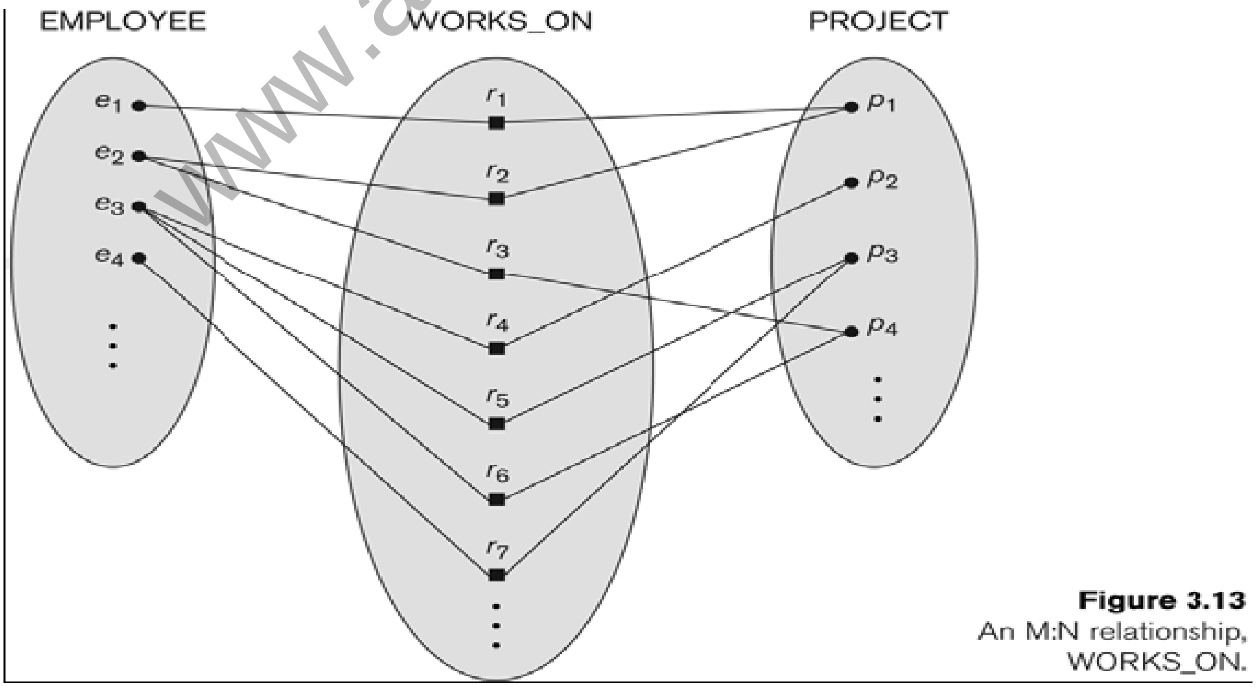


Figure 3.13
An M:N relationship, WORKS_ON.

Relationship type vs. relationship set

Relationship Type:

Is the schema description of a relationship. Identifies the relationship name and the participating entity types. Also identifies certain relationship constraints.

Relationship Set:

The current set of relationship instances represented in the database. The current *state of a relationship type.* Previous figures displayed the relationship sets
Each instance in the set relates individual participating entities – one from each participating entity type.

In ER diagrams, we represent the *relationship type as follows:*
Diamond-shaped box is used to display a relationship type.
Connected to the participating entity types via straight lines.

Refining the COMPANY database schema by introducing relationships

By examining the requirements, six relationship types are identified.

All are *binary relationships(degree 2)*
Listed below with their participating entity types:
WORKS_FOR (between EMPLOYEE, DEPARTMENT)
MANAGES (also between EMPLOYEE, DEPARTMENT)
CONTROLS (between DEPARTMENT, PROJECT)
WORKS_ON (between EMPLOYEE, PROJECT)
SUPERVISION (between EMPLOYEE (as subordinate),
EMPLOYEE (as supervisor))
DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are: WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

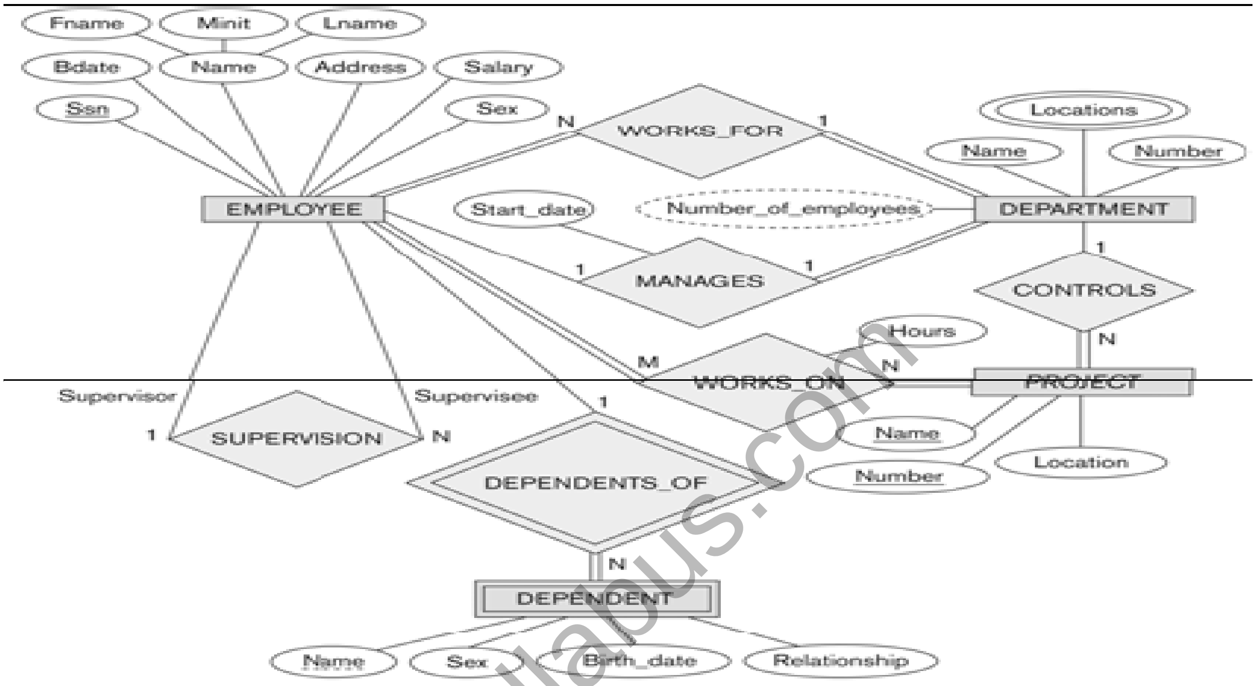


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Relationship Types

In the refined design, some attributes from the initial entity types are refined into relationships:

Manager of DEPARTMENT -> MANAGES
Works_on of EMPLOYEE -> WORKS_ON
Department of EMPLOYEE -> WORKS_FOR etc

In general, more than one relationship type can exist between the same participating entity types MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
Different meanings and different relationship instances.

Recursive Relationship Type

An relationship type whose with the same participating entity type in distinct roles

Example: In the SUPERVISION relationship EMPLOYEE participates twice in two distinct roles:

supervisor (or boss) role

supervisee (or subordinate) role

Each relationship instance relates two distinct EMPLOYEE entities:

One employee in *supervisor role*

One employee in *supervisee role*

Weak Entity Types

An entity that does not have a key attribute. A weak entity must participate in an identifying relationship type with an owner or identifying entity type.

Entities are identified by the combination of:

A partial key of the weak entity type

The particular entity they are related to in the identifying entity type.

Example:

A DEPENDENT entity is identified by the dependent's first name, and the specific EMPLOYEE with whom the dependent is related.

Name of DEPENDENT is the partial key.

DEPENDENT is a weak entity type.

EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Constraints on Relationships

Constraints on Relationship Types

(Also known as ratio constraints)

Cardinality Ratio (specifies *maximum participation)*

One-to-one (1:1)

One-to-many (1:N) or Many-to-one (N:1)

Many-to-many (M:N)

Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)

zero (optional participation, not existence-dependent)

one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship

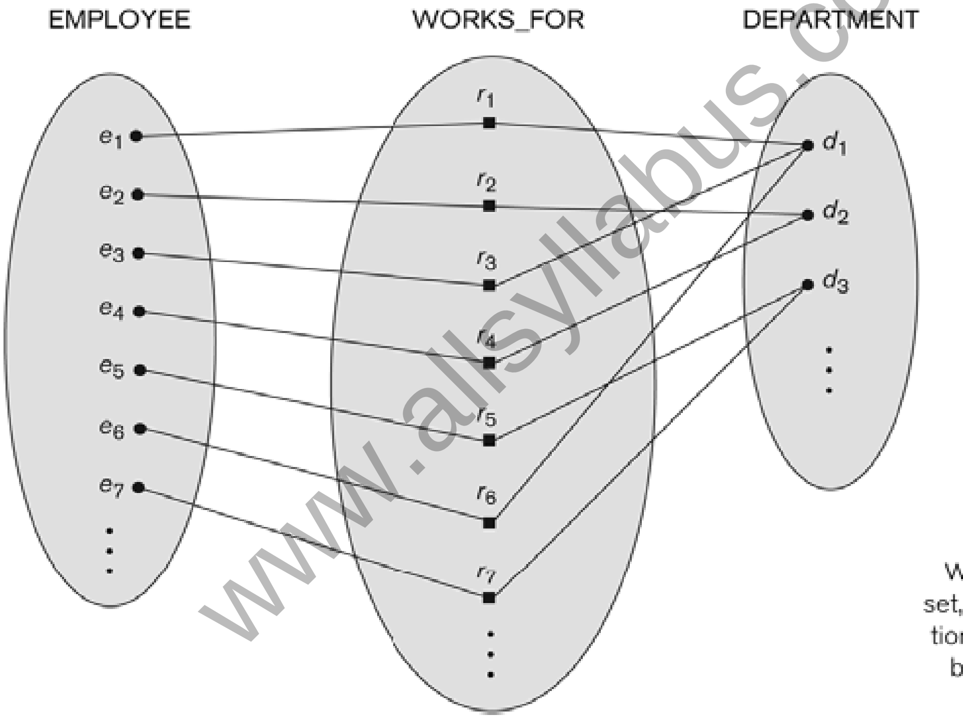


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many-to-many (M:N) Relationship

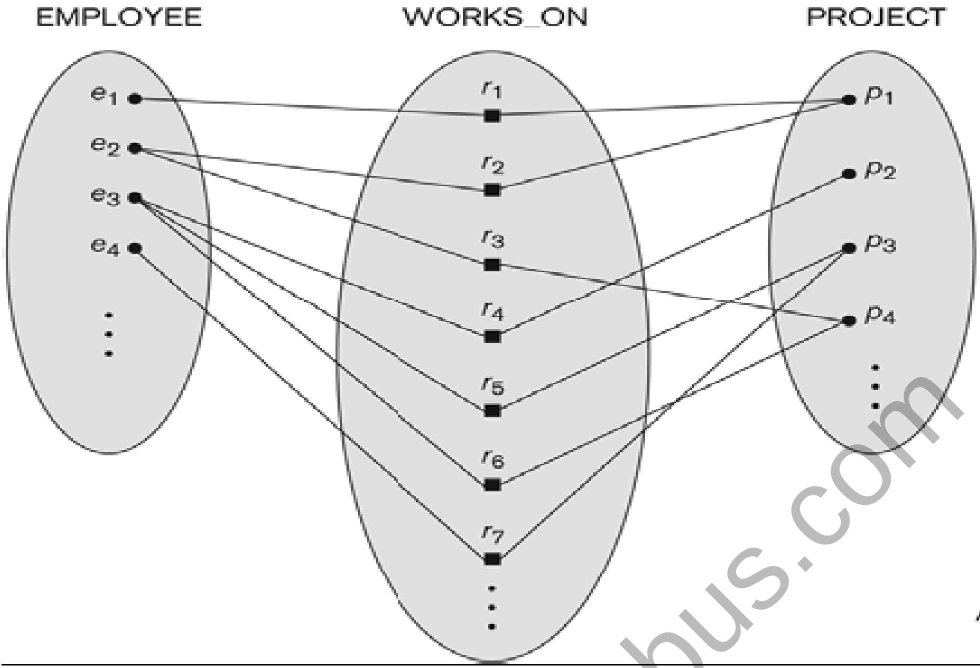


Figure 3.13
An M:N relationship, WORKS_ON.

Displaying a recursive relationship

In a recursive relationship type.

Both participations are same entity type in different roles.

For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).

In following figure, first role participation labeled with 1 and second role participation labeled with 2.

In ER diagram, need to display role names to distinguish participations.

A Recursive Relationship Supervision

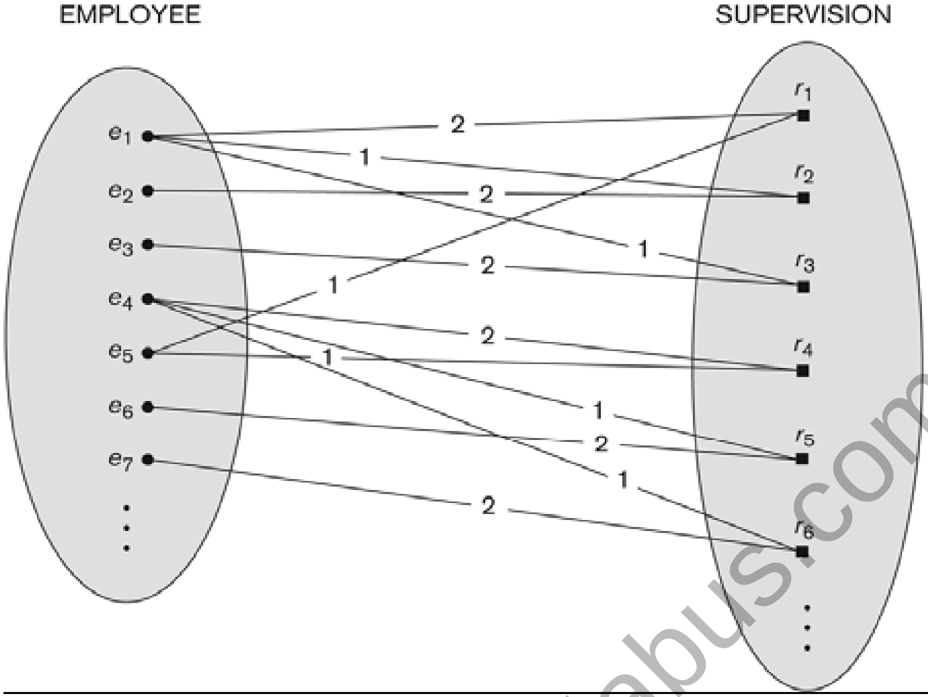


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Recursive Relationship Type is: SUPERVISION (participation role names are shown)

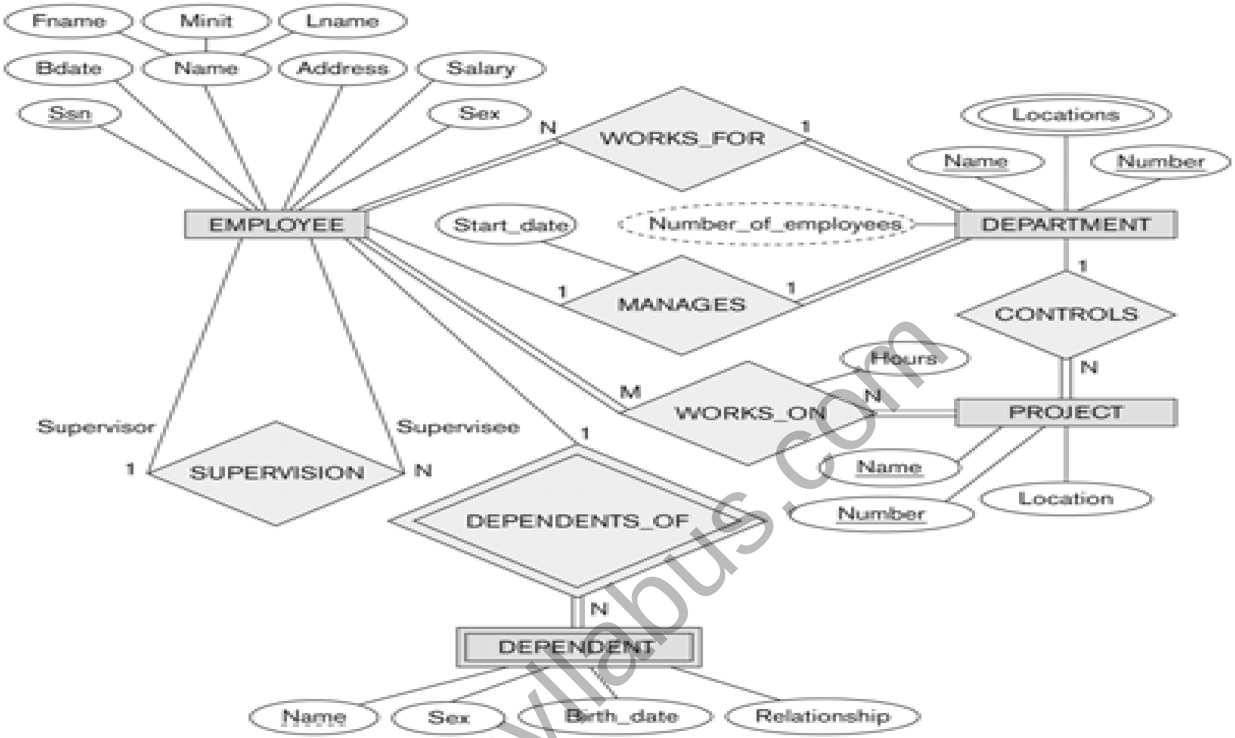


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Attributes of Relationship types

A relationship type can have attributes:

For example, HoursPerWeek of WORKS_ON

Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.

A value of HoursPerWeek depends on a particular (employee, project) combination
Most relationship attributes are used with M:N relationships.
In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship.

Example Attribute of a Relationship Type:
Hours of WORKS_ON

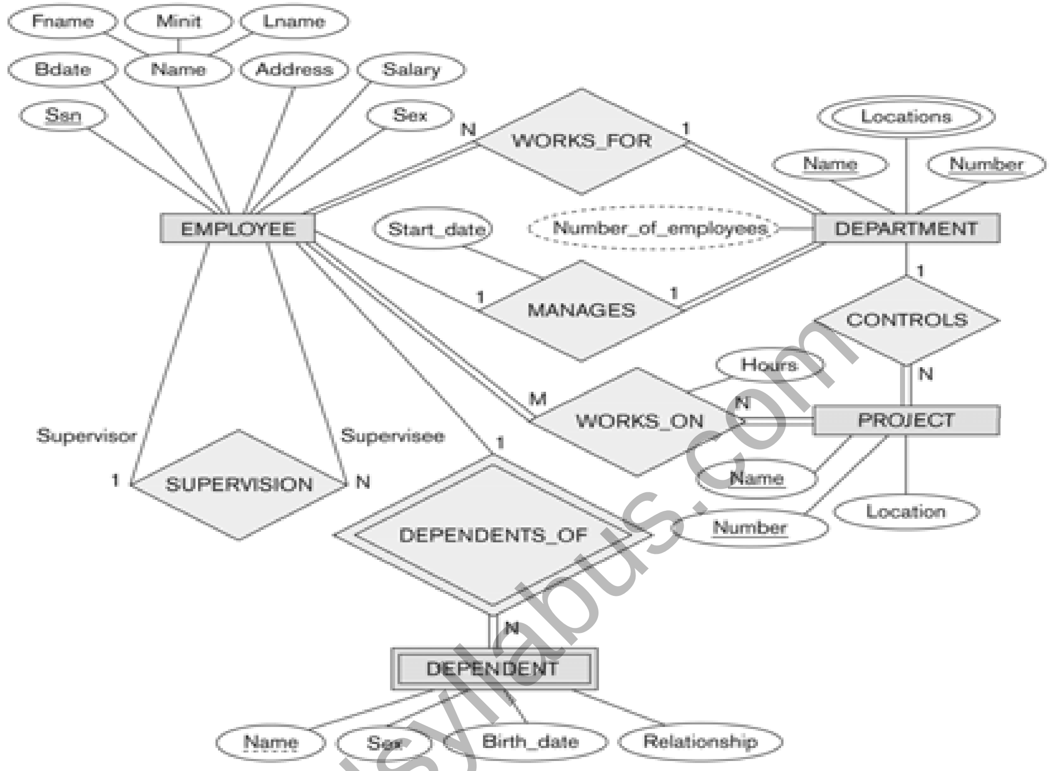


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

Cardinality ratio (of a binary relationship): 1:1,1:N, N:1, or M:N

Shown by placing appropriate numbers on the relationship edges.

Participation constraint (on each participating entity type): total (called existence dependency) or partial.

Total shown by double line, partial by single line

Alternative (min, max) notation for relationship structural constraints:

Specified on each participation of an entity type E in a relationship type R

Specifies that each entity e in E participates in at least *min and at* most *max relationship instances in R*

Default(no constraint): min=0, max=n (signifying no limit)

Must have min≤max, min≥0, max ≥1

Derived from the knowledge of mini-world constraints

Examples:

A department has exactly one manager and an employee can manage at most one department.

Specify (0,1) for participation of EMPLOYEE in MANAGES

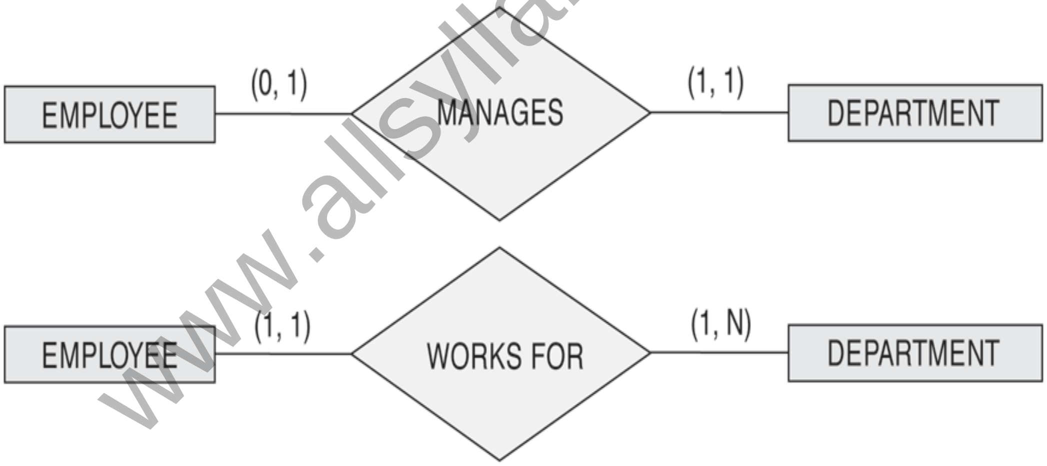
Specify (1,1) for participation of DEPARTMENT in MANAGES

An employee can work for exactly one department but a department can have any number of employees.

Specify (1,1) for participation of EMPLOYEE in WORKS_FOR

Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min , max) notation for relationship constraints



COMPANY ER Schema Diagram using (min , max) notation

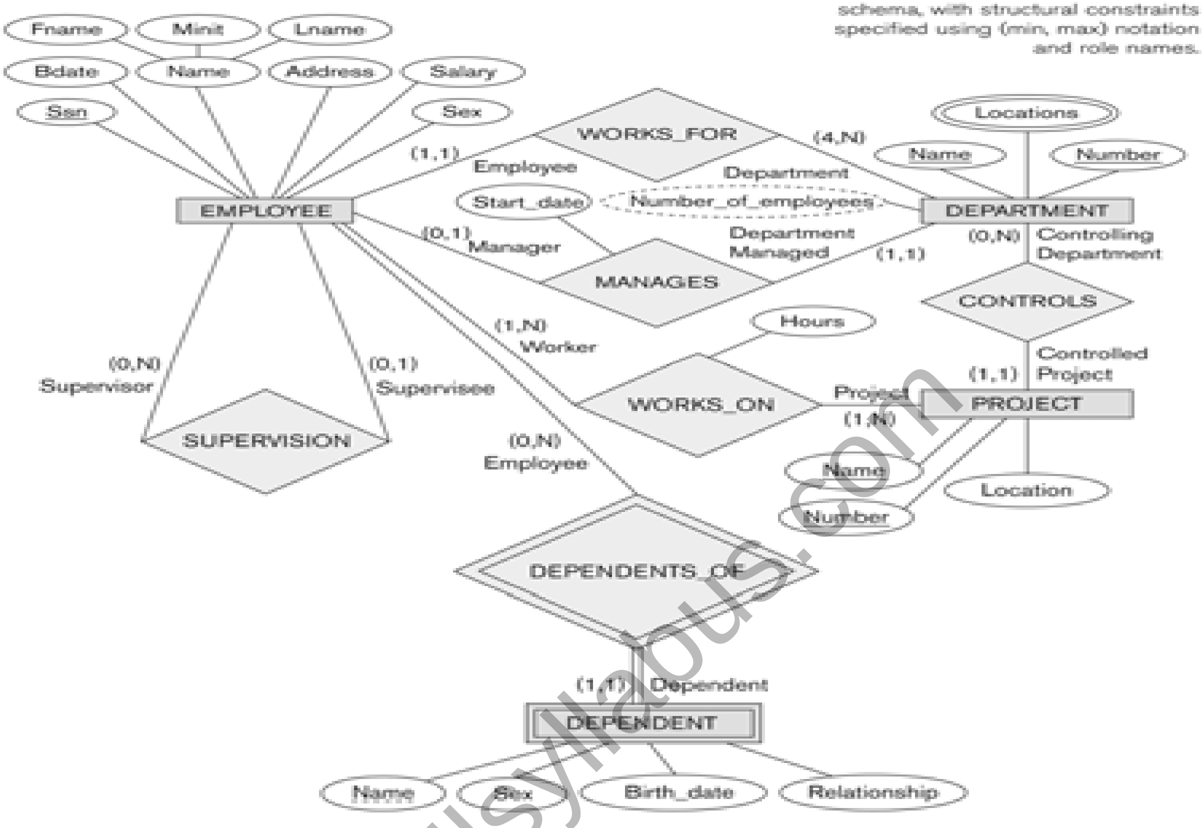


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

n-ary relationships (n > 2)

In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on next slide)

If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)

In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Fig 3.17c)

Example of a ternary relationship

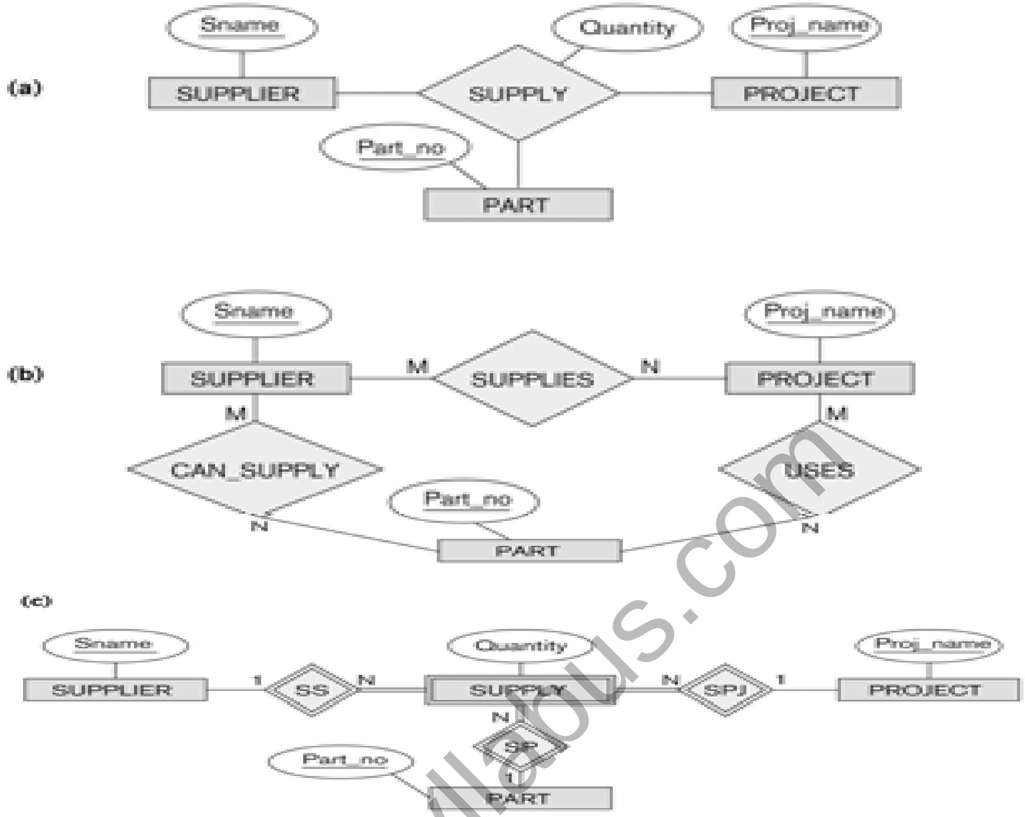


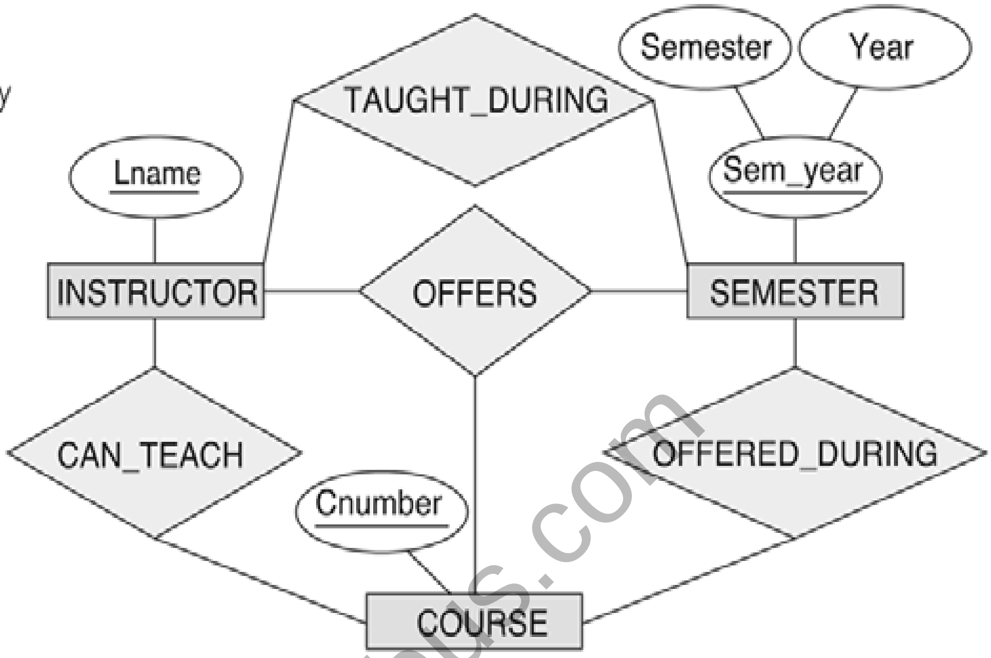
Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant.

For example, the TAUGHT_DURING binary relationship in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)

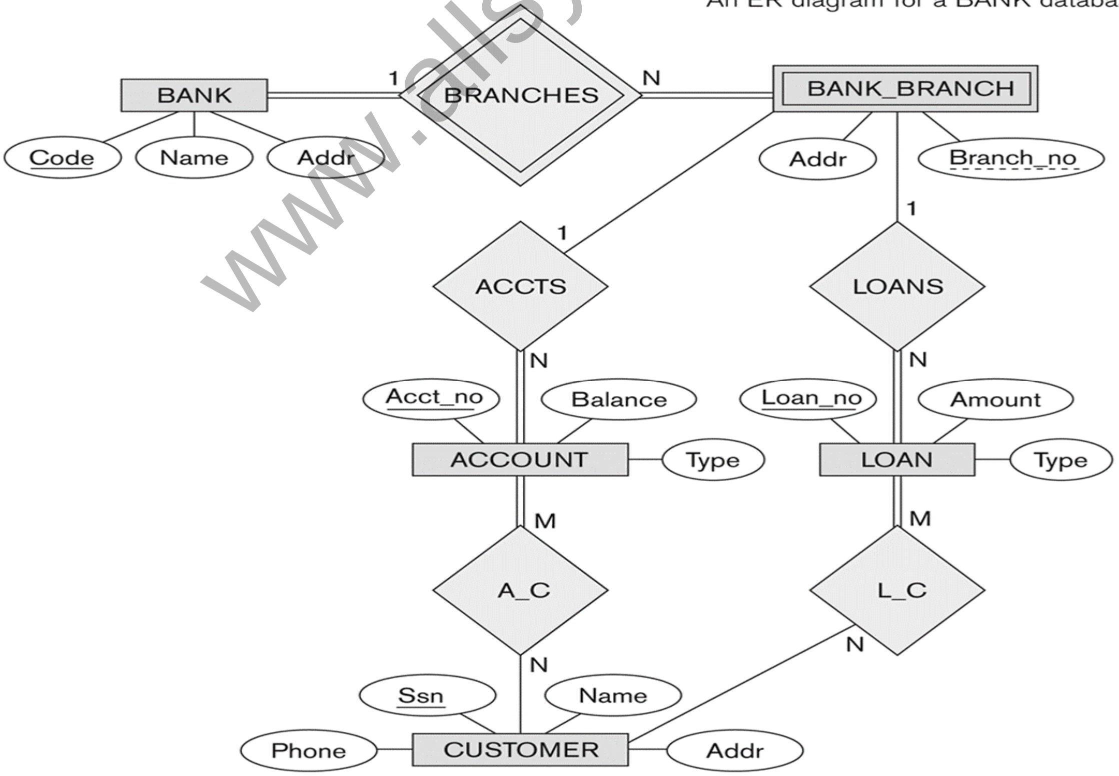
Another example of a ternary relationship

Figure 3.18
Another example of ternary versus binary relationship types.



Bank Database

An ER diagram for a BANK database schema.



There are three basic notations that the E-R Model employs:

1.Entity Sets.

2.Relationship sets.

3.Attributes.

2.2.2 **Entities and Entity sets:** An Entity is any object of interest to and organization or for the representation in the database.They represent objects in the real world which is distinguishable from all other objects.

For eg: Every person in a college is an entity.

Every room in a college is an entity.

Associated with an entity is a set of properties.These properties are used to distinguish to from one entity to another entity.

For Eg:1.The Attributes of the entity of student are

USN,Name,Address.

2.The Attributes of the Entity Of Vehicle are

Vehicle no,Make,Capacity.

For the purpose of accessing and storing information. Only certain attributes are used.Those attributes which uniquely identify every instance of the entity is termed as **primary key.**

An Entity which has a set of attributes.Which can uniquely identify all the entities is termed as **Strong entity**.

An entity whose primary key does not determine all the instance of the entity uniquely termed as **weak entity**.

A collection of similar entities,Which has certain properties which are common forms an entity set for organization such as a college the object of concern include. Student,Teacher,Rooms,Subjects.The collection of similar entities forms entity set.

2.2.3 Attributes.

An Entity is represented by a set of properties called Attributes.The attributes are useful in describing the properties of each entity in the entity set.

Types of attributes:

1.Simple Attributes: The attributes which cannot be further divided into subparts.

Eg; University Seat Number of a student is unique which cannot be further divided.

2. Composite Attributes :The attributes can be further divided into portions.
Eg: The attribute name in the Student Database can be further divided into First name,Middle name,Last name.

Name
Firstname Middle name Last name

3. Single valued attributes : The attribute at any instant contains only a specific value at any instant.

for eg The USN is unique

4.Multivalued Attributes; Certain attributes for example the dependent name in the policy database may have set of values assigned to it.There may be more than one dependent for a single policy holder.

5.Stored Attributes:For a person entity,the value of age can be determined from the current date and the value of that person's birthdate .The Age attribute is hence derived attribute and is said to be derivable from the birthdate attributes,which is called a stored attributes.

6.NULL Attributes: A NULL value attribute is used when an attributes does not have any values.

Data integrity

Data is accepted based on certain rules & there fore data is valid.
Enforcing data integrity ensures that the data in the database is valid and correct. Keys play an important role in maintaining data integrity.

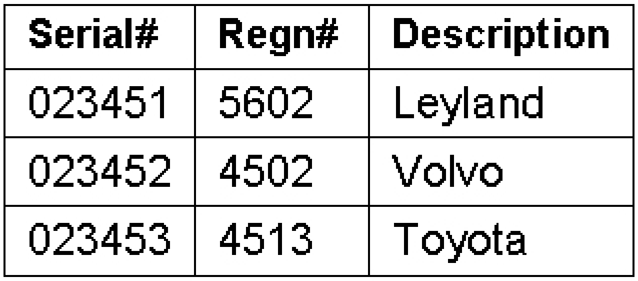
The various types of keys that have been identified are :
Candidate key
Primary key
Alternate key
Composite key
Foreign Key

Candidate key

An attribute or set of attributes that uniquely identifies a row is called a Candidate key.

This attribute has values that are unique

Vehicle



Serial#	Regn#	Description
023451	5602	Leyland
023452	4502	Volvo
023453	4513	Toyota

Primary Key

The Candidate key that you choose to identify each row uniquely is called the Primary key.

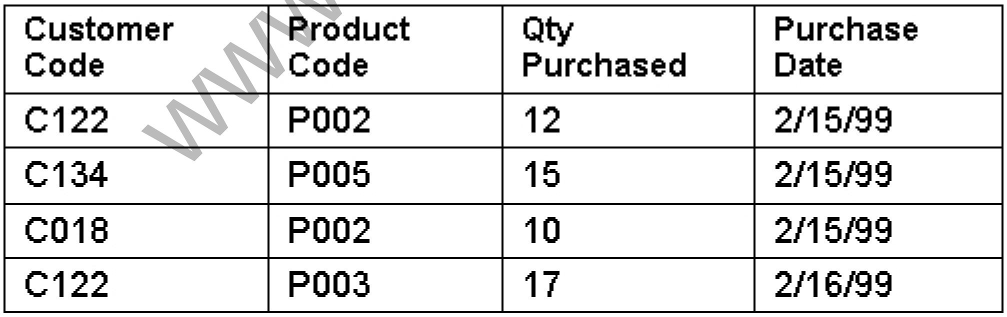
Alternate Key

A Candidate key that is not chosen as a Primary key is an Alternate key.

Composite Key

In certain tables, a single attribute cannot be used to identify rows uniquely and a combination of two or more attributes is used as a Primary key. Such keys are called Composite keys.

Purchase



Customer Code	Product Code	Qty Purchased	Purchase Date
C122	P002	12	2/15/99
C134	P005	15	2/15/99
C018	P002	10	2/15/99
C122	P003	17	2/16/99

Foreign Key

When a primary key of one table appears as an attribute in another table, it is called the Foreign key in the second table

A foreign key is used to relate two tables.

Weak entity:

A weak entity does not have a distinguishing attribute of its own and mostly are dependent entities, which are part of some another entity.

A weak entity will always be related to one or more strong entities.

They can be also understood as multi-valued attributes.

Relationships

A *relationship type* is a meaningful association between entity types

A *relationship* is an association of entities where the association includes one entity from each participating entity type.

Relationship types are represented on the ER diagram by a series of lines.

As always, there are many notations in use today...

In the original Chen notation, the relationship is placed inside a diamond, e.g. managers manage employees:

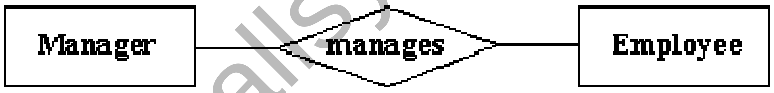


Figure : Chens notation for relationships

For this module, we will use an alternative notation, where the relationship is a label on the line. The meaning is identical

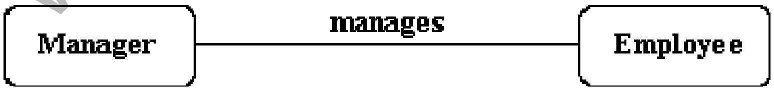


Figure : Relationships used in this document

2.3.1 Degree of a Relationship

The number of participating entities in a relationship is known as the degree of the relationship.

If there are two entity types involved it is a *binary* relationship type

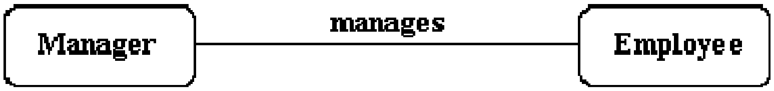


Figure : Binary Relationships

If there are three entity types involved it is a *ternary* relationship type

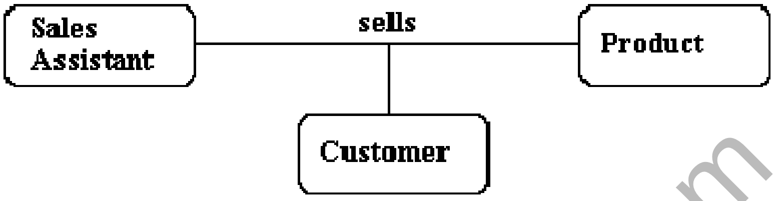


Figure : Ternary relationship

It is possible to have a n-array relationship (e.g. quaternary or unary).

Unary relationships are also known as a *recursive* relationship.

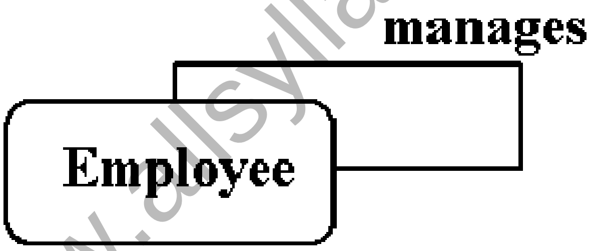


Figure : Recursive relationship

It is a relationship where the same entity participates more than once in different roles.

In the example above we are saying that employees are managed by employees.

If we wanted more information about who manages whom, we could introduce a second entity type called manager.

2.3.2 Replacing ternary relationships

When a ternary relationship occurs in an ER model they should always be removed before finishing the model. Sometimes the relationships can be replaced by a series of binary relationships that link pairs of the original ternary relationship.

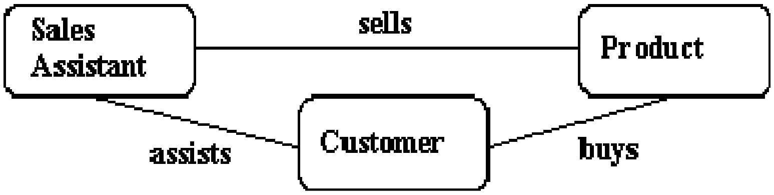


Figure : A ternary relationship example

This can result in the loss of some information - It is no longer clear which sales assistant sold a customer a particular product.

Try replacing the ternary relationship with an entity type and a set of binary relationships.

Relationships are usually verbs, so name the new entity type by the relationship verb rewritten as a noun.

The relationship *sells* can become the entity type *sale*.

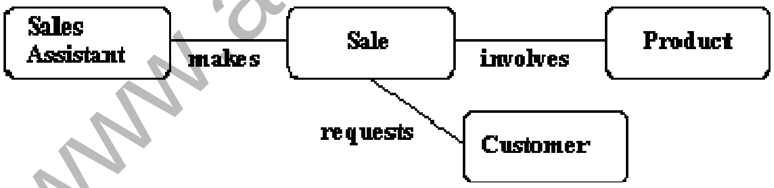


Figure : Replacing a ternary relationship

So a sales assistant can be linked to a specific customer and both of them to the sale of a particular product.

This process also works for higher order relationships.

Cardinality

Relationships are rarely one-to-one

For example, a manager usually manages more than one employee

This is described by the *cardinality* of the relationship, for which there are four possible categories.

One to one (1:1) relationship

One to many (1:m) relationship

Many to one (m:1) relationship

Many to many (m:n) relationship

On an ER diagram, if the end of a relationship is straight, it represents 1, while a "crow's foot" end represents many.

A one to one relationship - a man can only marry one woman, and a woman can only marry one man, so it is a one to one (1:1) relationship

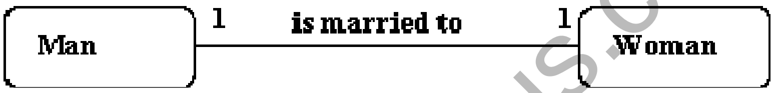


Figure : One to One relationship example

A one to may relationship - one manager manages many employees, but each employee only has one manager, so it is a one to many (1:n) relationship

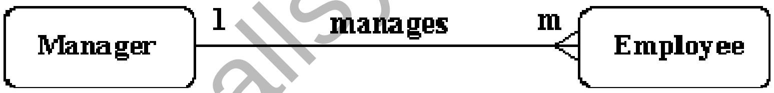


Figure : One to Many relationship example

A many to one relationship - many students study one course. They do not study more than one course, so it is a many to one (m:1) relationship

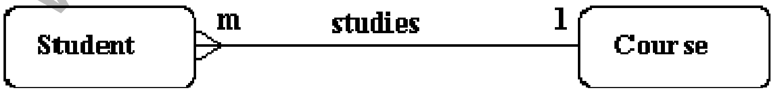


Figure : Many to One relationship example

A many to many relationship - One lecturer teaches many students and a student is taught by many lecturers, so it is a many to many (m:n) relationship

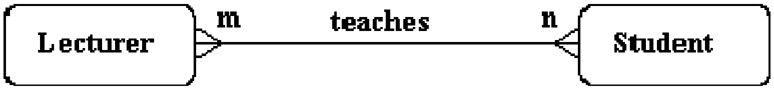


Figure : Many to Many relationship example

2.3.4 Optionality

A relationship can be optional or mandatory.

If the relationship is mandatory

an entity at one end of the relationship must be related to an entity at the other end.

The optionality can be different at each end of the relationship

For example, a student must be on a course. This is mandatory. To the relationship `student studies course' is mandatory.

But a course can exist before any students have enrolled. Thus the relationship `course is_studied_by student' is optional.

To show optionality, put a circle or `0' at the `optional end' of the relationship.

As the optional relationship is `course is_studied_by student', and the optional part of this is the student, then the `O' goes at the student end of the relationship connection.

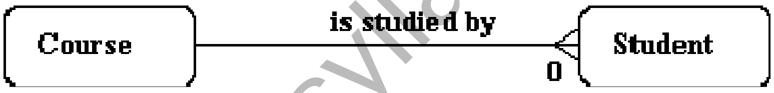


Figure : Optionality example

It is important to know the optionality because you must ensure that whenever you create a new entity it has the required mandatory links.

2.4.1Entities

Bus - Company owns busses and will hold information about them.

Route - Buses travel on routes and will need described.

Town - Buses pass through towns and need to know about them

Driver - Company employs drivers, personnel will hold their data.

Stage - Routes are made up of stages

Garage - Garage houses buses, and need to know where they are.

A bus is allocated to a route and a route may have several buses.

Bus-route (m:1) is serviced by

A route comprises of one or more stages.

route-stage (1:m) comprises

One or more drivers are allocated to each stage.

driver-stage (m:1) is allocated .

A stage passes through some or all of the towns on a route.

stage-town (m:n) passes-through

A route passes through some or all of the towns

route-town (m:n) passes-through

Some of the towns have a garage

garage-town (1:1) is situated

A garage keeps buses and each bus has one `home' garage

garage-bus (m:1) is garaged

E-R Diagram

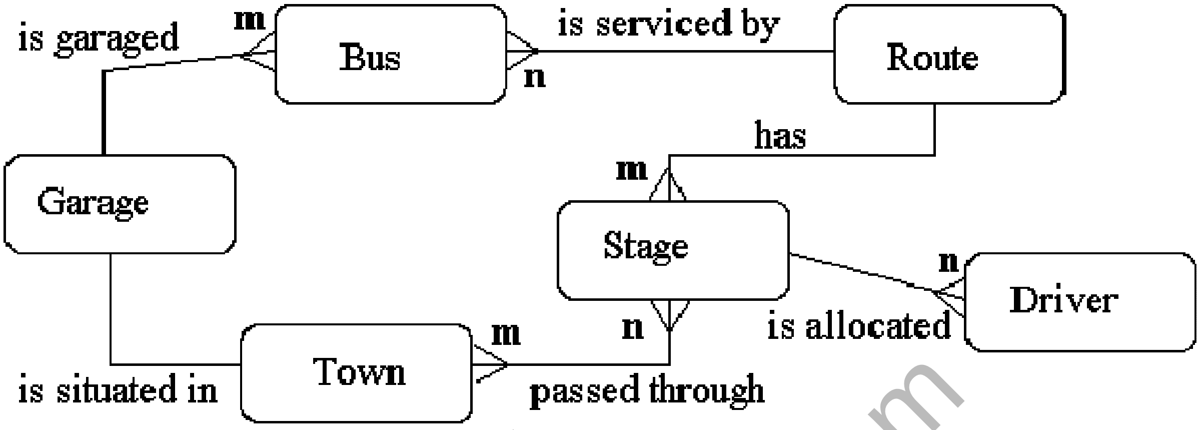


Figure : Bus Company

Attributes

Bus (reg-no,make,size,deck,no-pass)

Route (route-no,avg-pass)

Driver (emp-no,name,address,tel-no)

Town (name)

Stage (stage-no)

Garage (name,address)

Example: Entity and Relationship sets for the hospital called General Hospital, Patients,

Doctors, Beds, Examines, Bed Assigned, Accounts, has Account.

patients, entity set with attributes SSNo, LastName, FirstName, HomePhone, Sex, DateofBirth, Age, Street, City, State, Zip.

doctors, entity set with attributes SSNo, LastName, FirstName, OfficePhone, Pager, Specialty.

examines, relational set with attributes Date, Time, Diagnosis, Fee.

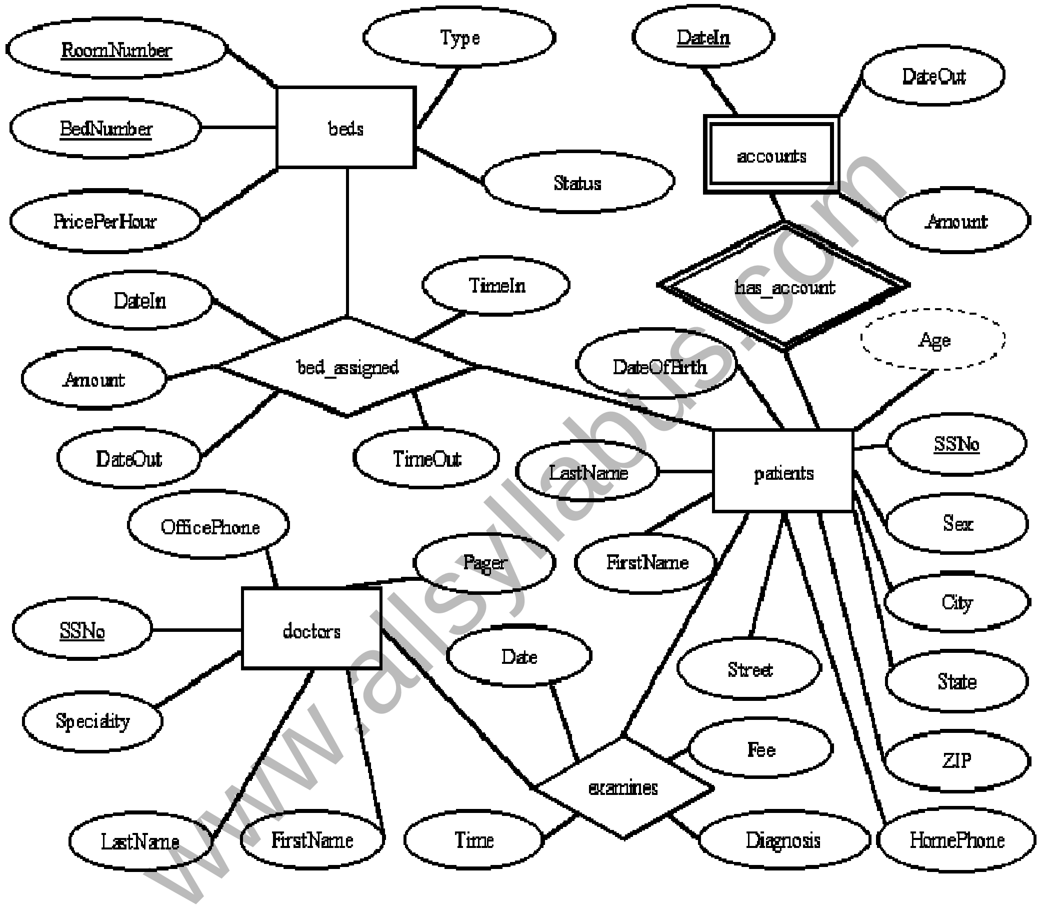
beds, entity set with attributes RoomNumber, BedNumber, Type, Status, PricePerHour.

Bed_assigned, relational set with attributes DateIn, TimeIn, DateOut, TimeOut, Amount.

accounts, weak entity set with attributes DateIn, DateOut, Amount.

has_account, relational set with no Attributes

"Hospital" / Entity-Relationship Diagram



2.5 Constructing an ER model

Before beginning to draw the ER model, read the requirements specification carefully. Document any assumptions you need to make.

1. Identify entities - list all potential entity types. These are the object of interest in the system. It is better to put too many entities in at this stage and them discard them later if necessary.

2. Remove duplicate entities - Ensure that they really separate entity types or just two names for the same thing.
    - Also do not include the system as an entity type
    - e.g. if modelling a library, the entity types might be books, borrowers, etc.
    - The library is the system, thus should not be an entity type.
3. List the attributes of each entity (all properties to describe the entity which are relevant to the application).
    - Ensure that the entity types are really needed.
    - are any of them just attributes of another entity type?
    - if so keep them as attributes and cross them off the entity list.
    - Do not have attributes of one entity as attributes of another entity!
4. Mark the primary keys.
    - Which attributes uniquely identify instances of that entity type?
    - This may not be possible for some weak entities.
5. Define the relationships
    - Examine each entity type to see its relationship to the others.
6. Describe the cardinality and optionality of the relationships
    - Examine the constraints between participating entities.
7. Remove redundant relationships
    - Examine the ER model for redundant relationships.

ER modelling is an iterative process, so draw several versions, refining each one until you are happy with it. Note that there is no one right answer to the problem, but some solutions are better than others!

Overview

- construct an ER model
- understand the problems associated with ER models
- understand the modelling concepts of Enhanced ER modelling

Types of Data Integrity

Data Integrity falls into the following categories

Entity integrity

Entity integrity ensures that each row can be uniquely identified by an attribute called the Primary key. The Primary key cannot have a NULL value.

Domain integrity

Domain integrity refers to the range of valid entries for a given column. It ensures that there are only valid entries in the column**.**

Referential integrity

Referential integrity ensures that for every value of a Foreign key, there is a matching value of the Primary key.

2.7 Relational database

Relations can be represented as two-dimensional data tables with rows and columns

The rows of a relation are called tuples.

The columns of a relation are called attributes.

The attributes draw values from a domain (a legal pool of values).

The number of tuples in a relation is called its cardinality while the number of attributes in a relation is called its degree

A relation also consists of a schema and an instance

Schema defines the structure of a relation which consists of a fixed set of attribute-domain pairs.

An instance of a relation is a time-varying set of tuples where each tuple consists of attribute-value pairs.