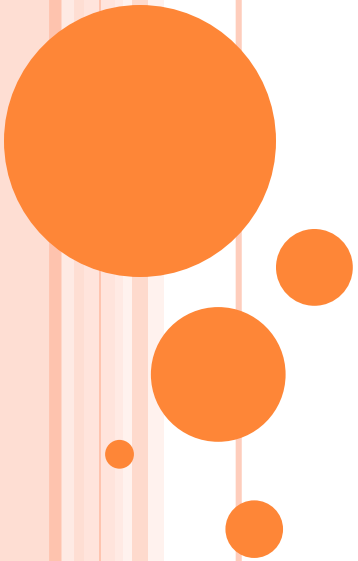
DATABASE DESIGN



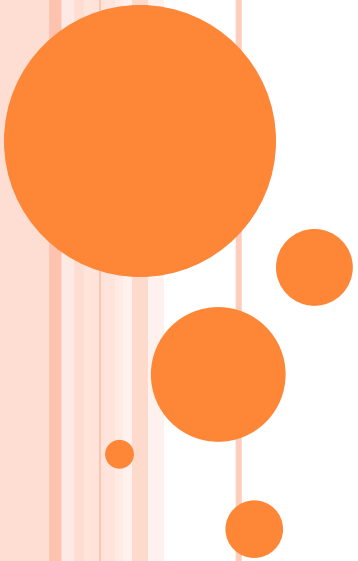
CONTENT

- Conceptual Database Design
- Logical Database Design
- Physical Database Design

Database Design

- Steps in building a database for an application:
  1. Understand the real-world domain being captured.
  2. Specify it using a database conceptual model (E/R,OO).
  3. Translate specification to model of DBMS (relational).
  4. Create schema using DBMS commands (DDL).
  5. Load data (DML).

Entity-Relationship Model



Entity-Relationship Model (E/R)

- The **Entity-Relationship model** (ER) is a high-level description of the structure of the DB.
- The **Entity-Relationship Diagram** (ERD) is a graphical model for representing the conceptual model for the data.
- A E/R models the DB using three element types:
  - Entities
  - Attributes
  - Relationships

Entities & Entity Type

- **Entity** is an object that exists and is distinguishable from other objects.
  - e.g. person, company, course, university
- **Entity Type** is a set of entities of the same type that share the same properties.
  - e.g. set of all persons, companies, trees, courses

Relationships & Relationship Types

- A **relationship** associates 2 or more entities.
- A **relationship type** is a set of associations between entity types.

Degree of Relationship Type

- **Degree of relationship** refers to number of participating entity types in a relationship.
  - A relationship of degree two (2 entity types) are **binary.**
  - A relationship of degree three (3 entity types) are **ternary.**

Recursive Relationship

- **Recursive relationship** is a relationship type where the same entity type participates more than once in a different role. It is a **unary relationship.**

ROLES

- **Role** indicates the purpose that each participating entity type plays in a relationship. (e.g. manger, branch office)

- **Role** can be used when two entities are associated through more than one relationship to classify the purpose of each relationship

ATTRIBUTES

- **Attributes** are descriptive properties for an entity type **or** a relationship type.

- All entities in one entity type have the same attributes.

Attributes of Relationships

- All relationships in one relationship type have the same attributes.
- Relationships can be distinguished not only by their attributes but also by their participating entities.

Simple & Composite Attributes

- **Simple attribute** is an attribute that have a single value with an independent existence.
  - e.g. salary, age, gender,...
- **Composite attribute** is an attribute composed of multiple distinct components, each with an independent existence.
  - e.g. address (street, area, city, post code)

    name (First name, initial, Last name)

    phone no. (area code, number, exchange no)

Single-valued & Multi-valued Attributes

- **Single-valued attribute** is an attribute that holds a single value for a single entity. It is not necessarily a simple attribute.
  - e.g. student_no, age, gender,...
- **Multi-valued attribute** is an attribute that may hold multiple values, of the same type, for a single entity.
  - e.g. tel_no, degrees,…

Derived Attributes

Derived attribute is an attribute that represents a value that is derived from the value of a related attribute,not necessarily in the same entity type.

e.g. **age** is derived from **date_of_birth**

total_cost is derived from **quantity*unit_price**

KEYS

Candidate key (CK) is the minimal set of attributes that *uniquely* identifies an entity. It cannot contain null.

e.g. student_no, social_security_no, branch_no…

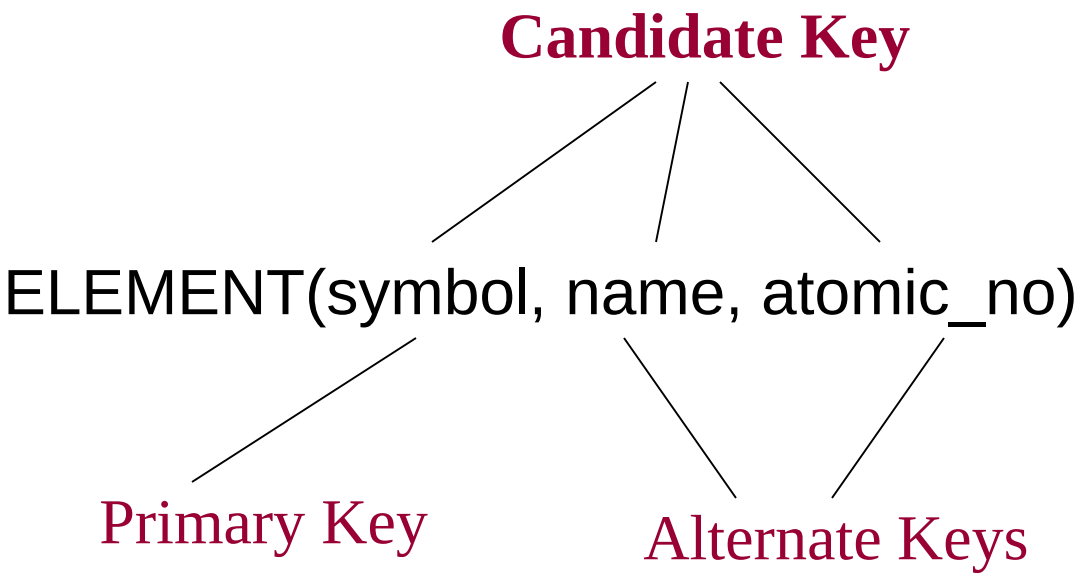
Primary Key (PK) is a candidate key that is selected to uniquely identify each entity.

Alternate Key (AK) is a candidate key that is NOT selected to be the primary key.

KEYS EXAMPLE

ELEMENT(symbol, name, atomic_no)

KEYS EXAMPLE



Choice of PK

- Choice of Primary Key (PK) is based on:
  - Attribute length
  - Number of attributes required
  - Certainty of uniqueness

KEYS

A key can be:

- **simple key** is a candidate key of one attribute.
  - e.g. student_no, branch_no…
- **composite key** is a candidate key that consists of two or more attributes.
  - e.g. enrolment(student_no, course_no, grade)
    - Key: student_no, course_no together

Strong & Weak Entity Types

- A **strong entity type** is NOT existence-dependent on some other entity type. It has a PK.

- A **weak entity type** is an entity type that is existence-dependent on some other entity type. It does not have a PK.
- The existence of a weak entity type depends on the existence of a strong entity set; it must relate to the strong entity type via a relationship type called **identifying relationship**.

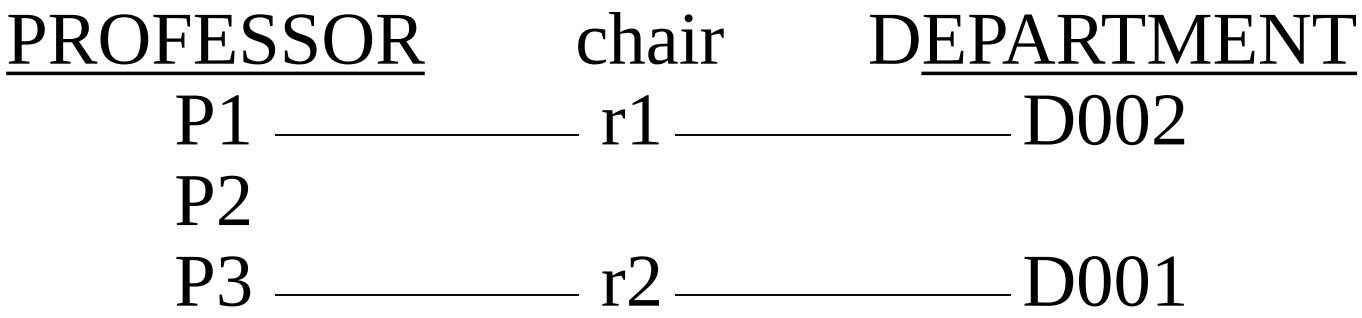
- The PK of a weak entity set is formed by the PK of its strong entity type, plus a weak entity type discriminator attribute.

CARDINALITIES

- **Cardinality ratio** expresses the number of relationships an entity can participate in.

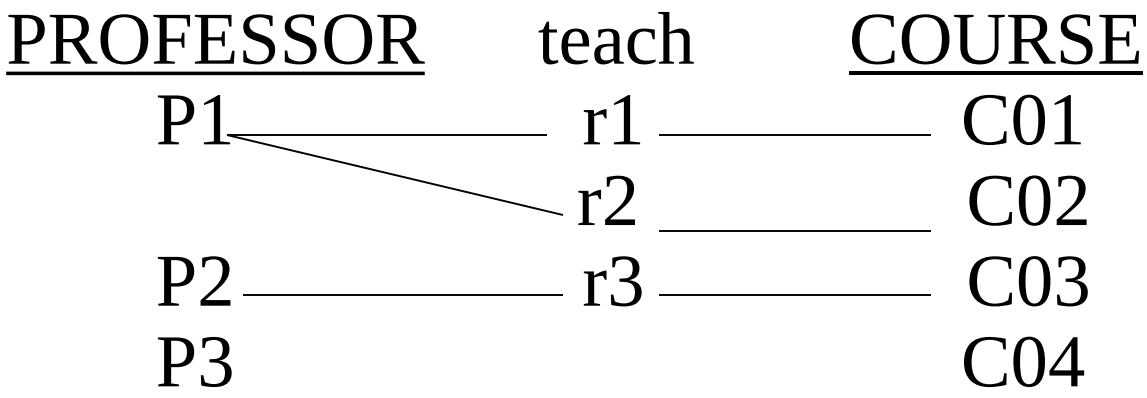
- Most useful in describing binary relationship types.
- For a binary relationship type the mapping cardinality must be one of the following types:
  - One to one (1:1)
  - One to many(1:M)
  - Many to one (M:1)
  - Many to many (M:N)

ONE-TO-ONE RELATIONSHIP



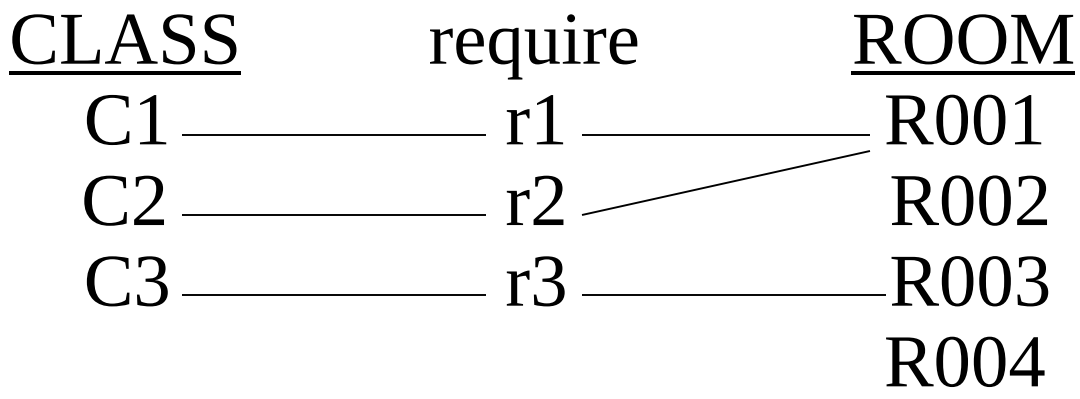
A professor chairs at most one department; and a department is chaired by only one professor.

ONE-TO-MANY RELATIONSHIP



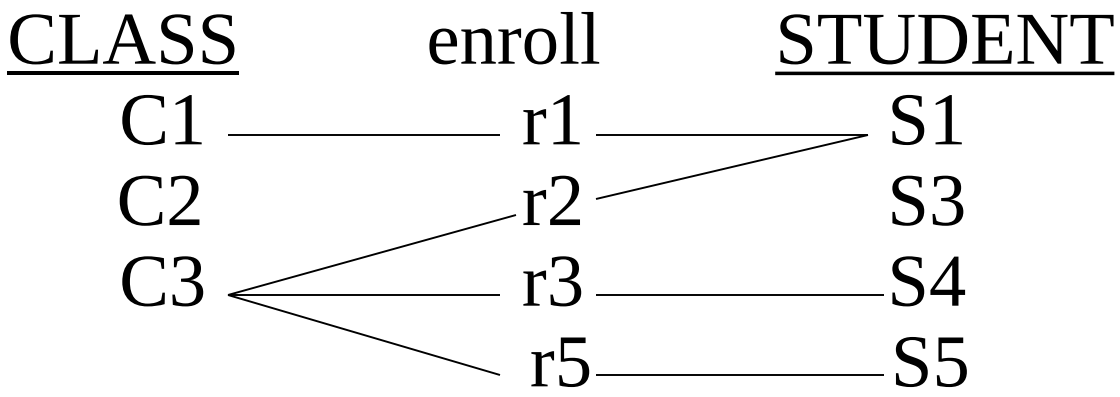
A course is taught by at most one professor; a professor teaches many courses.

MANY-TO-ONE RELATIONSHIP



A class requires one room; while a room can be scheduled for many classes.

MANY-TO-MANY RELATIONSHIP



A class enrolls many students; and each student is enrolled in many classes.

Multiplicity

- **Multiplicity** is the number (range) of possible entities that may relate to a single association through a particular relationship.
- It is best determined using sample data.
- Takes the form (min#, max#)

Participation Constraints

Participation constraints determine whether all or only some entities participate in a relationship.

Two types of participation:

- **Mandatory** (total)
- **Optional** (partial)

Participation Constraints…

- **Mandatory** (total) (1:*): if an entity's existence requires the existence of an associated entity in a particular relationship (existence-dependent).

  e.g. CLASS taught-by PROFESSOR

  i.e. CLASS is a *total participator* in the relation.

A weak entity always has a mandatory participation constraints but the opposite not always true.

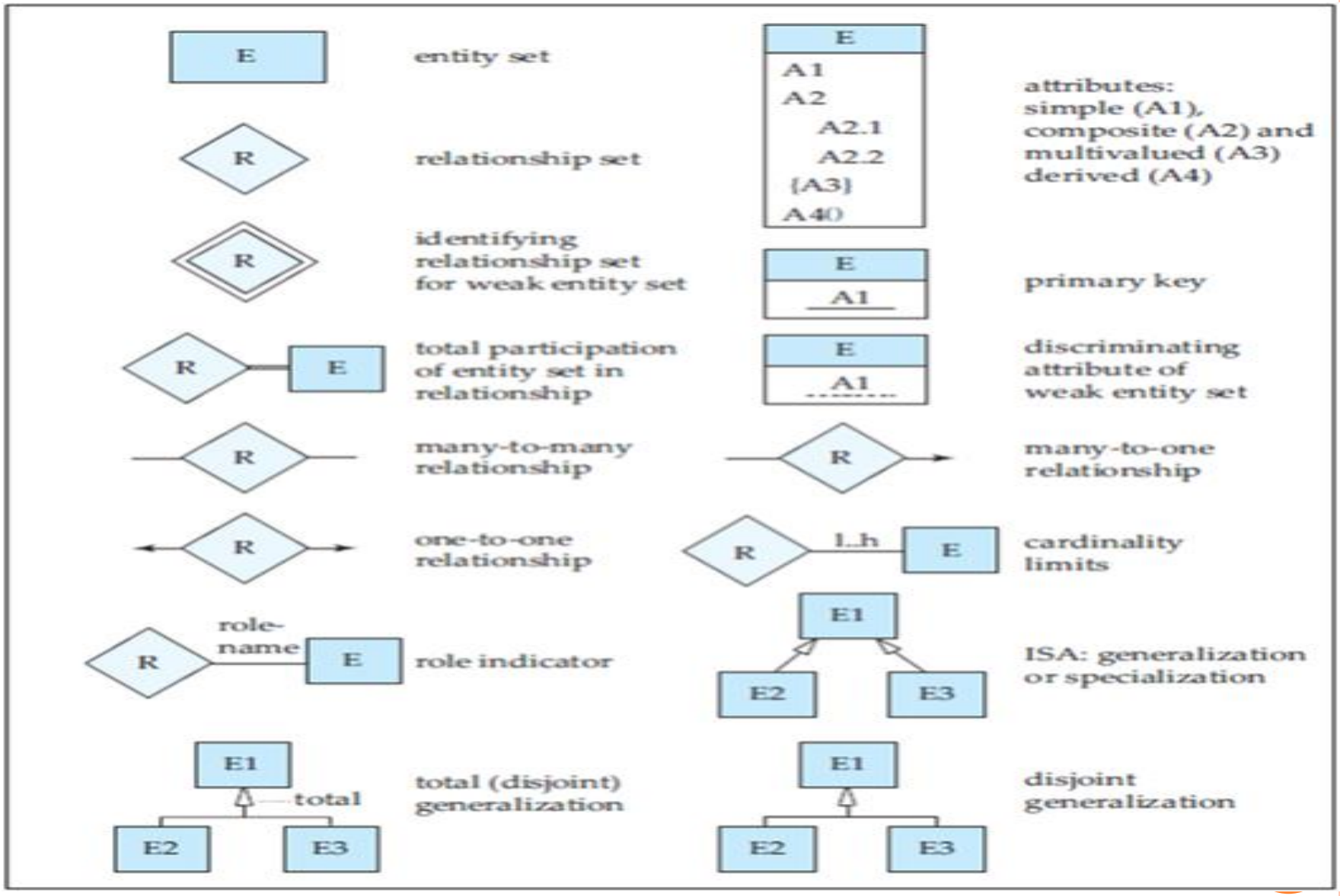
Participation Constraints…

- **Optional** (partial) (0:*): if an entity's existence does not require a corresponding entity in a particular relationship. (Not existence-dependent).

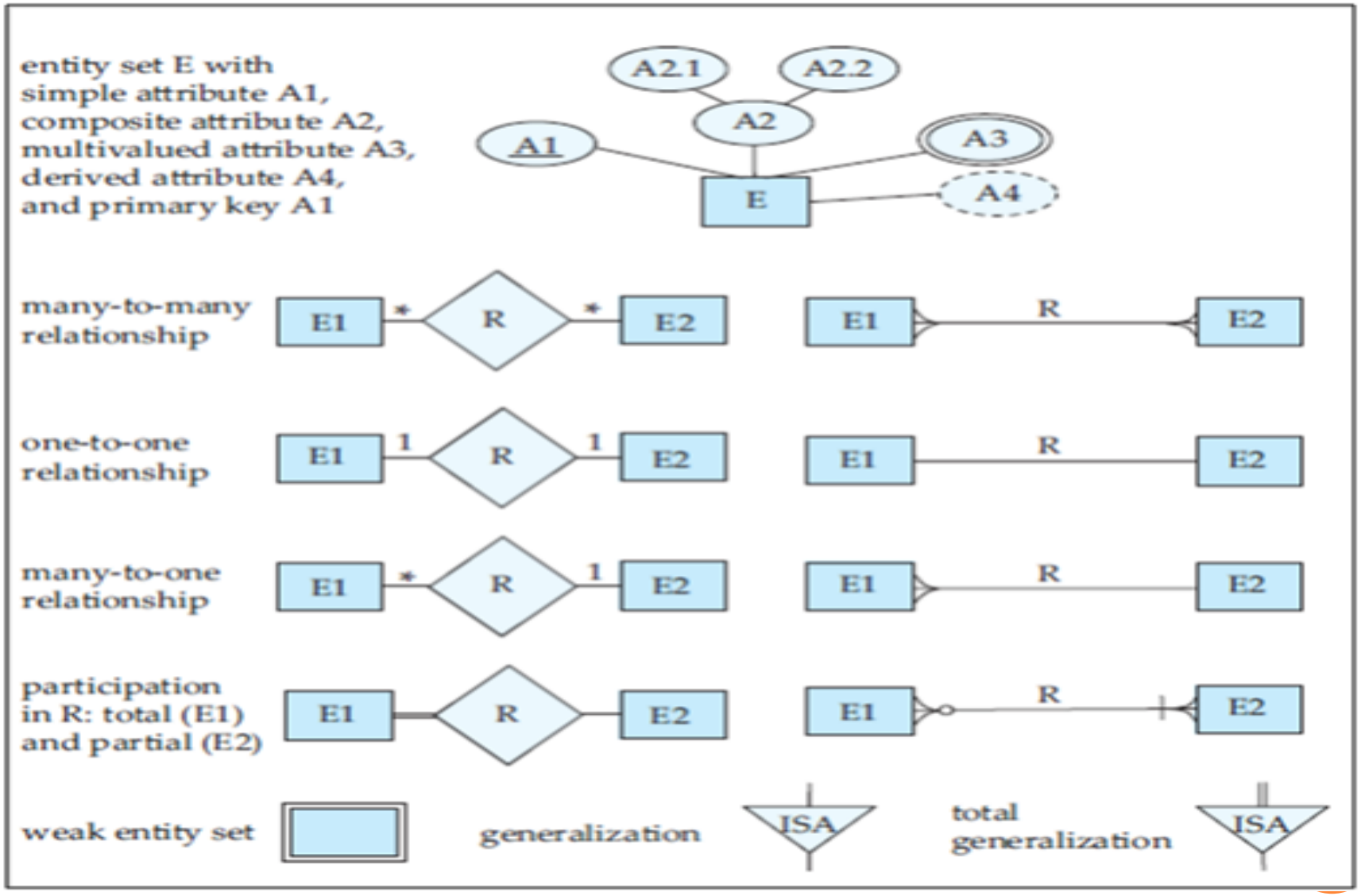
  e.g. PROFESSOR teach CLASS

  i.e. PROFESSOR is a *partial participator* in the relation.

SYMBOLS USED TO DRAW ERD



Symbols Used to draw ERD



ERD Example

- Example Database Application (COMPANY)
  - ER Model Concepts
    - Entities and Attributes
    - Entity Types, Value Sets, and Key Attributes
    - Relationships and Relationship Types
    - Weak Entity Types
    - Roles and Attributes in Relationship Types
  - ER Diagrams – Notation(different notation)
  - ER Diagram for COMPANY Schema
  - Alternative Notations – UML class diagrams, others

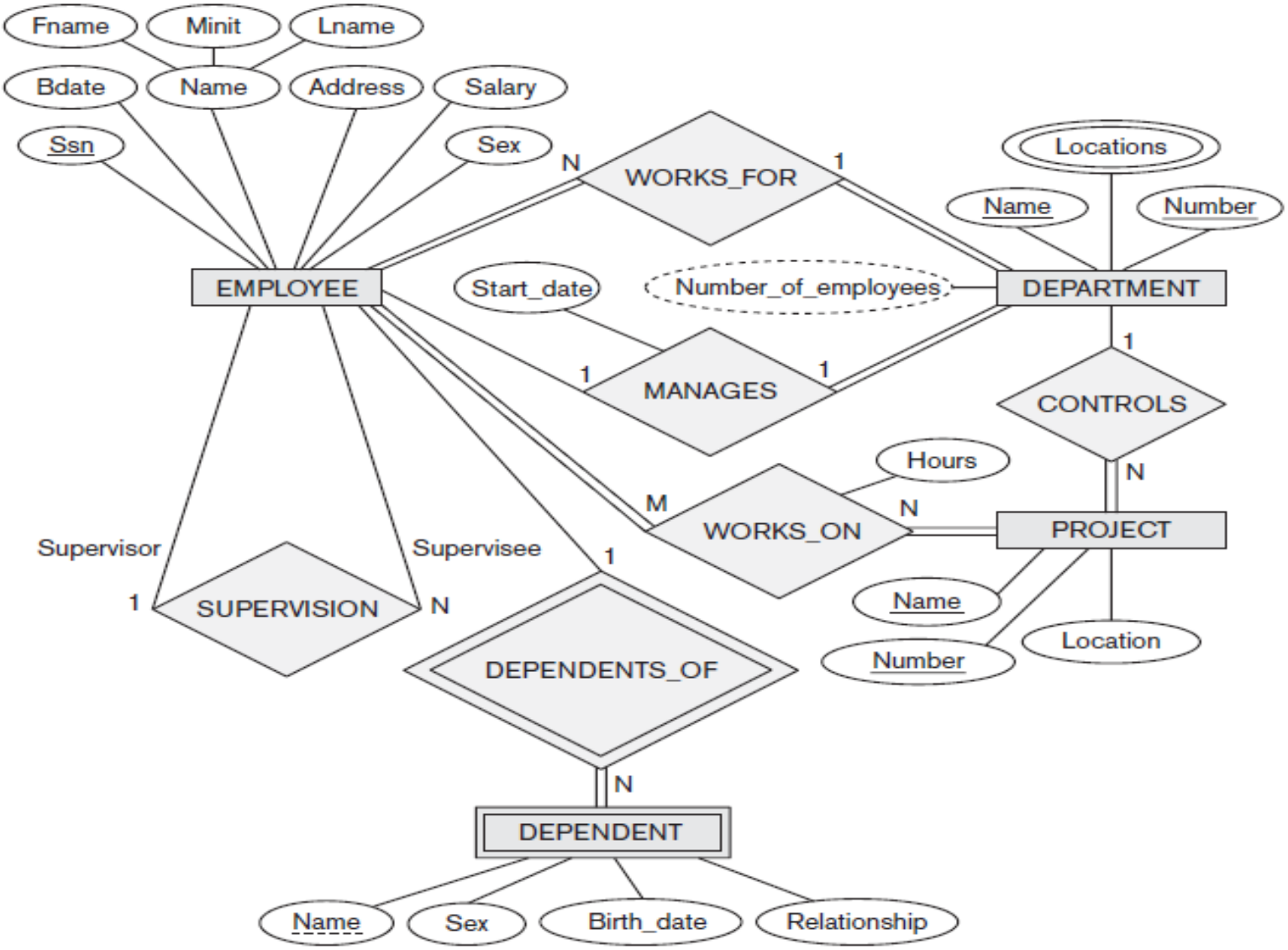
EXAMPLE COMPANY DATABASE

- Requirements of the Company (oversimplified for illustrative purposes)
  - 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.
  - Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.

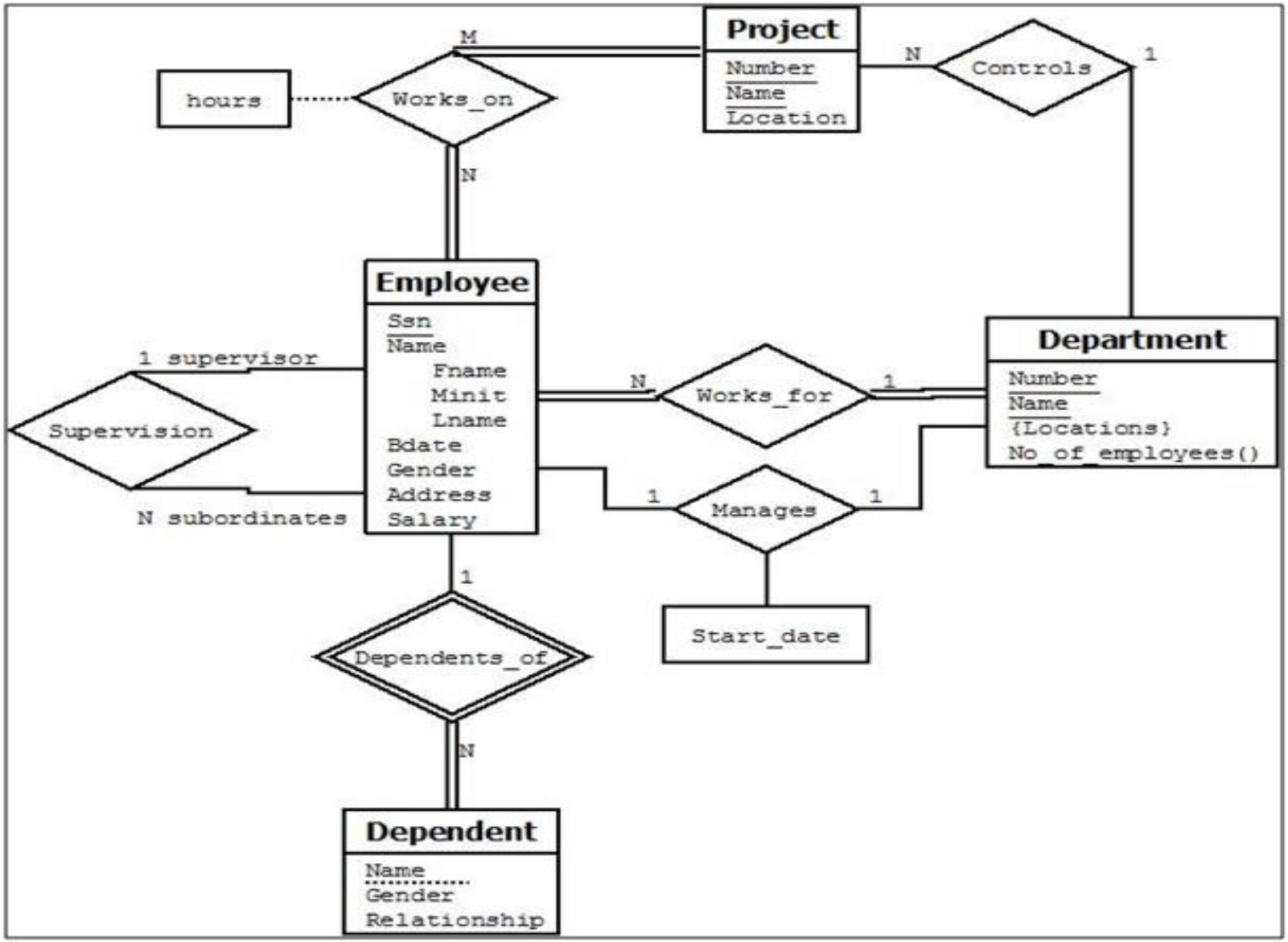
EXAMPLE COMPANY DATABASE…

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate. 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, birthdate, and relationship to employee.

EXAMPLE COMPANY DATABASE…



EXAMPLE COMPANY DATABASE…



Enhanced Entity-Relationship Model (EER)

- EER is an ER model supported with additional semantic concepts.

- Semantic concepts supported:
  - Specialization
  - Generalization
  - Aggregation

SPECIALIZATION

- Top-down design process; we designate sub-groupings within an entity type that are distinctive from other entities in the set.

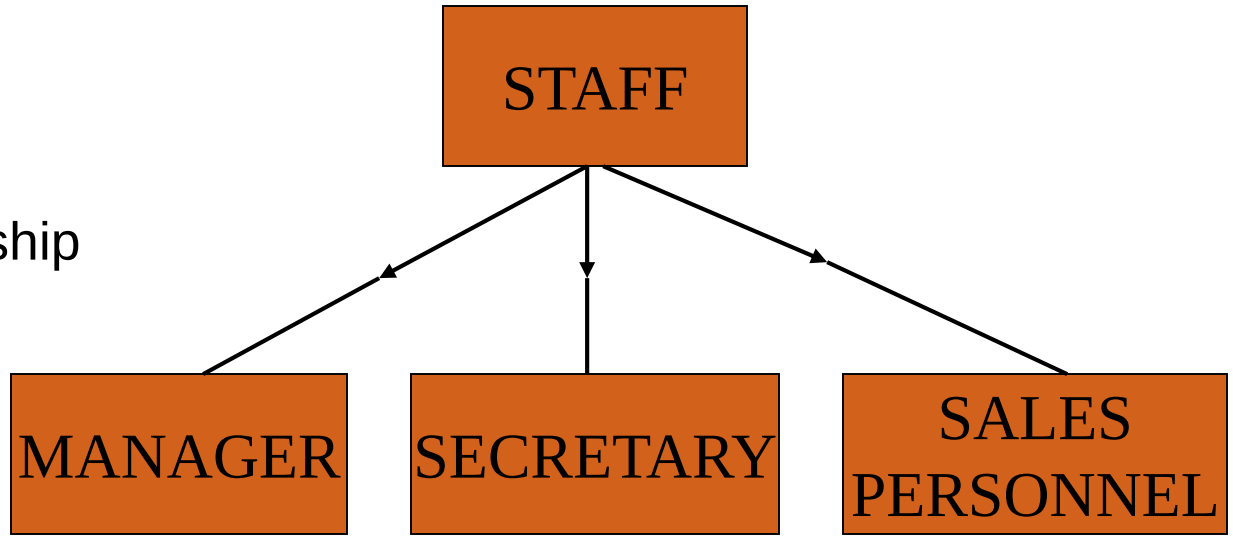
- These sub-groupings (**subclasses**) become lower-level entity types that have *attributes* or participate in *relationships* that do not apply to the higher-level entity set (**superclass**).

SPECIALIZATION/GENERALIZATION

Superclass

IS_A relationship
1:1

Subclass



Generalization

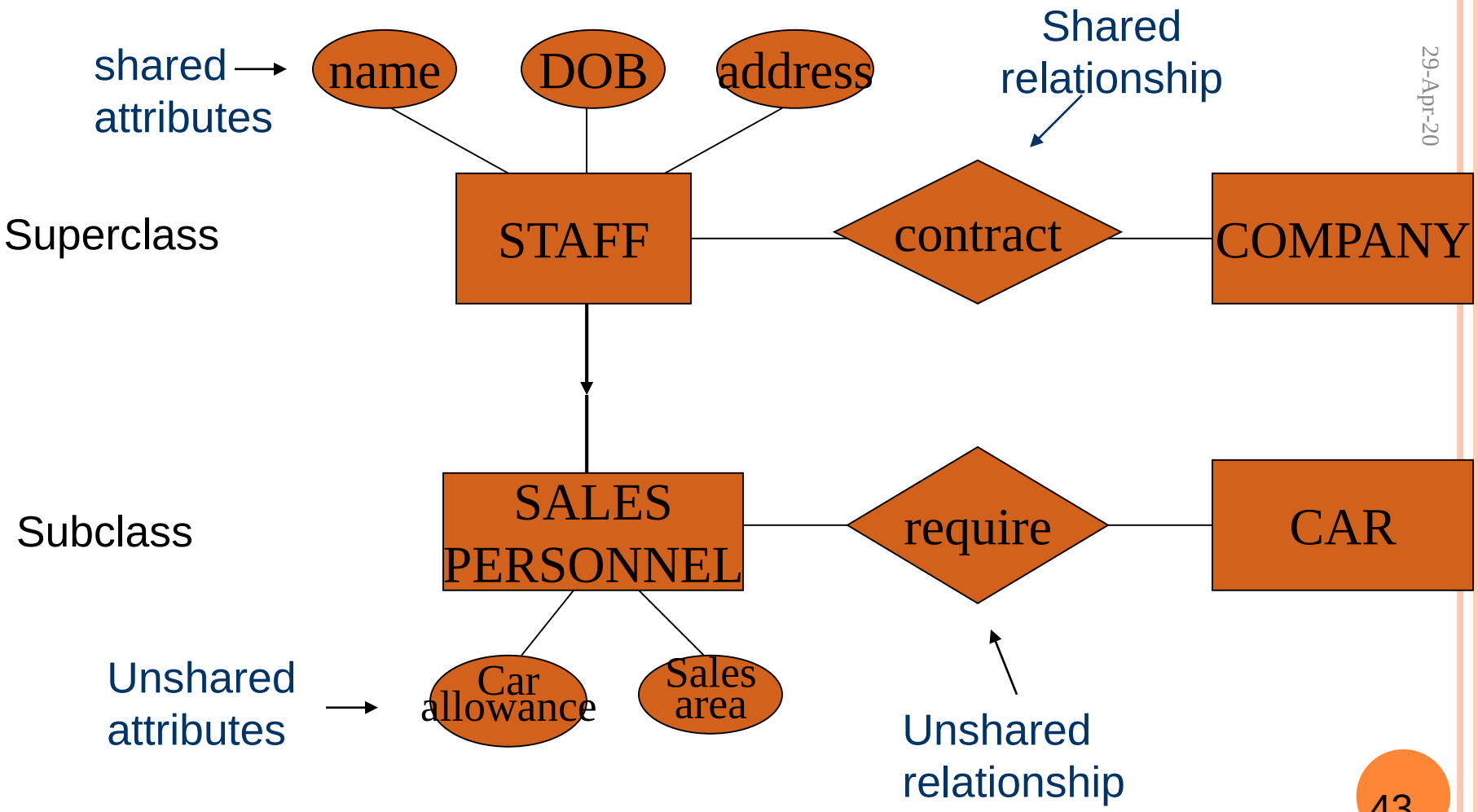
- A bottom-up design process – combine a number of entity types that share the same features into a higher-level (superclass) entity type.

- Specialization and generalization are simple inversions of each other; they are represented in an EER diagram in the same way.

INHERITANCE

A subclass entity type inherits all the attributes and relationship participation of the superclass entity type to which it is linked.

INHERITANCE



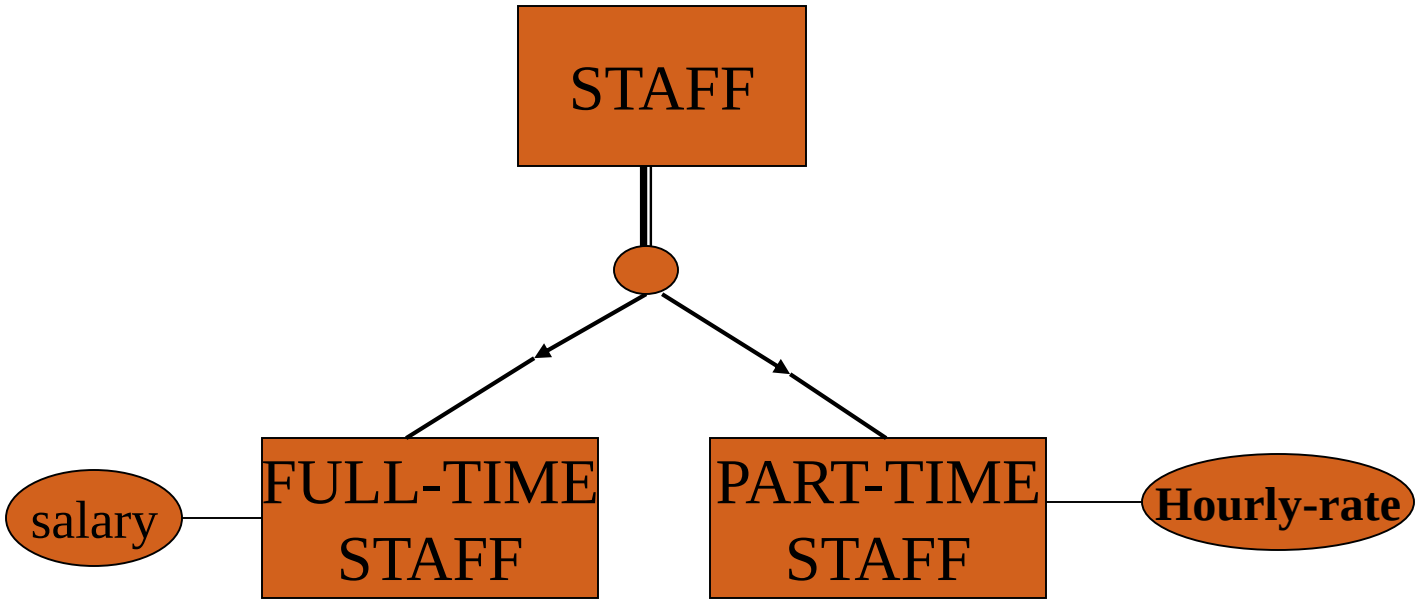
Constraints on Specialization/Generalization

- **Participation constraint** determines whether every member in the superclass must participate as a member of a subclass.

- Two types of participation constraints:
  - Mandatory (total)
  - Optional (partial)

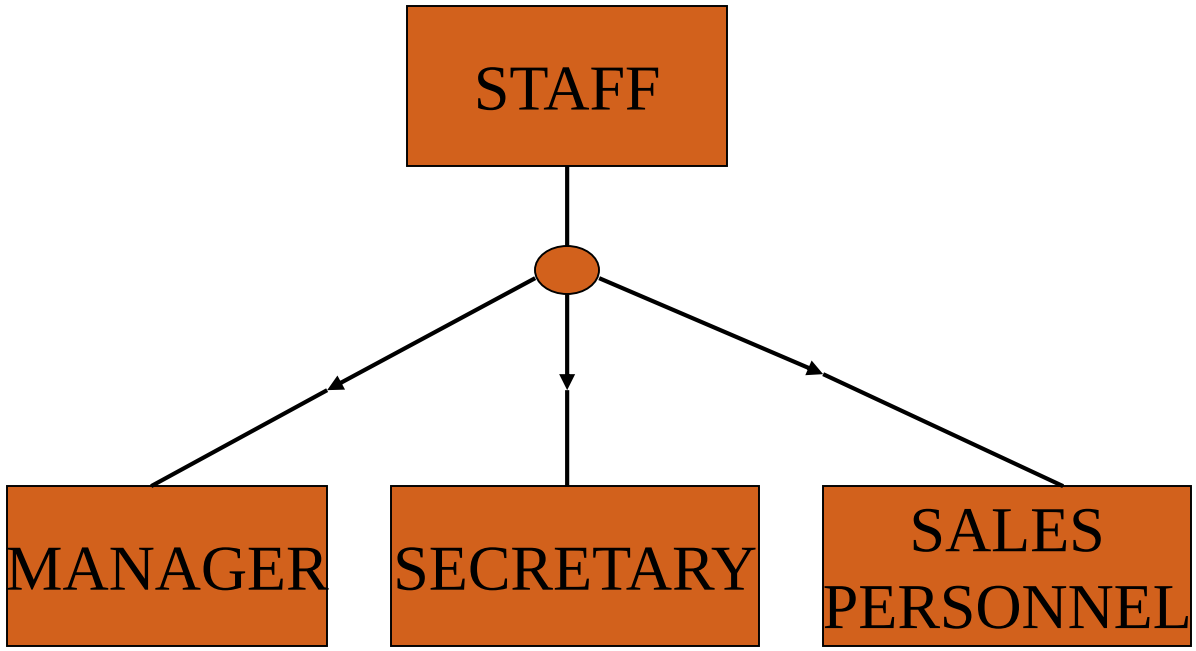
Participation Constraints

Mandatory (total) participation where every member in the superclass must also be a member of a subclass.



Participation Constraints

Optional (partial) participation where a member in the superclass need not belong to any of its subclasses.



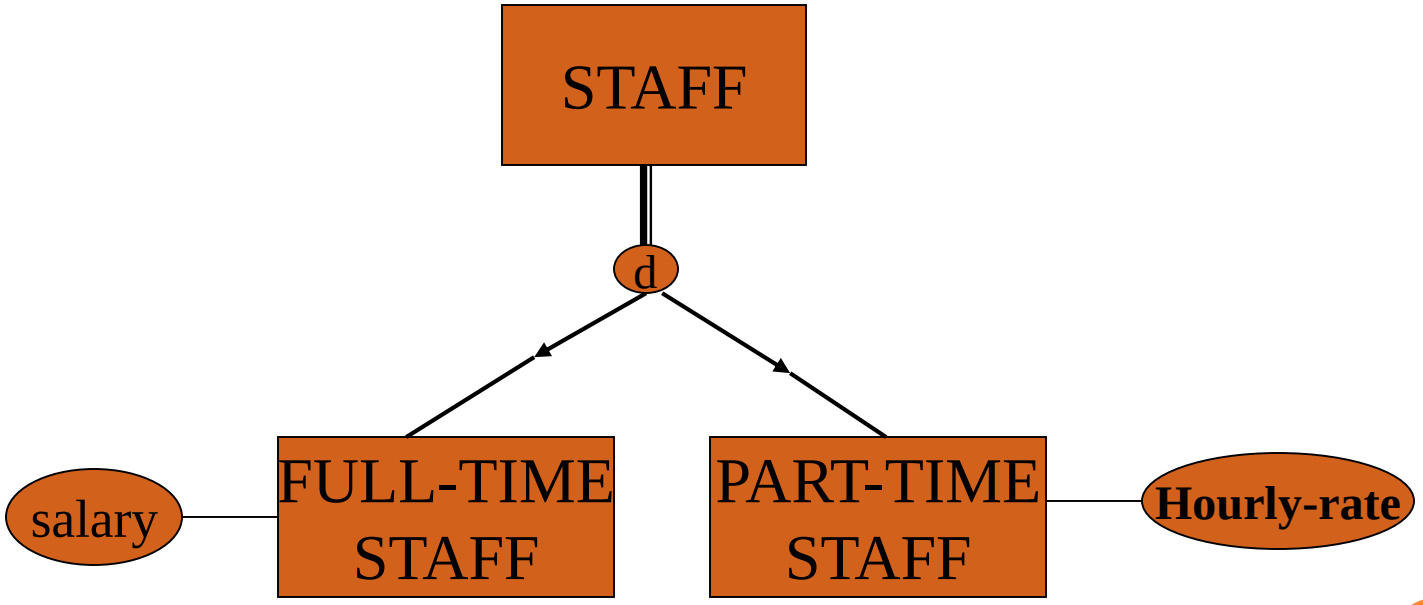
Constraints on Specialization/Generalization

- **Disjoint constraint** describes the relationship between members of the subclasses & indicates whether it is possible for a member of a subclass to be a member of one or more subclasses.

- Two types of disjoint constraints:
  - Disjoint
  - Non-Disjoint

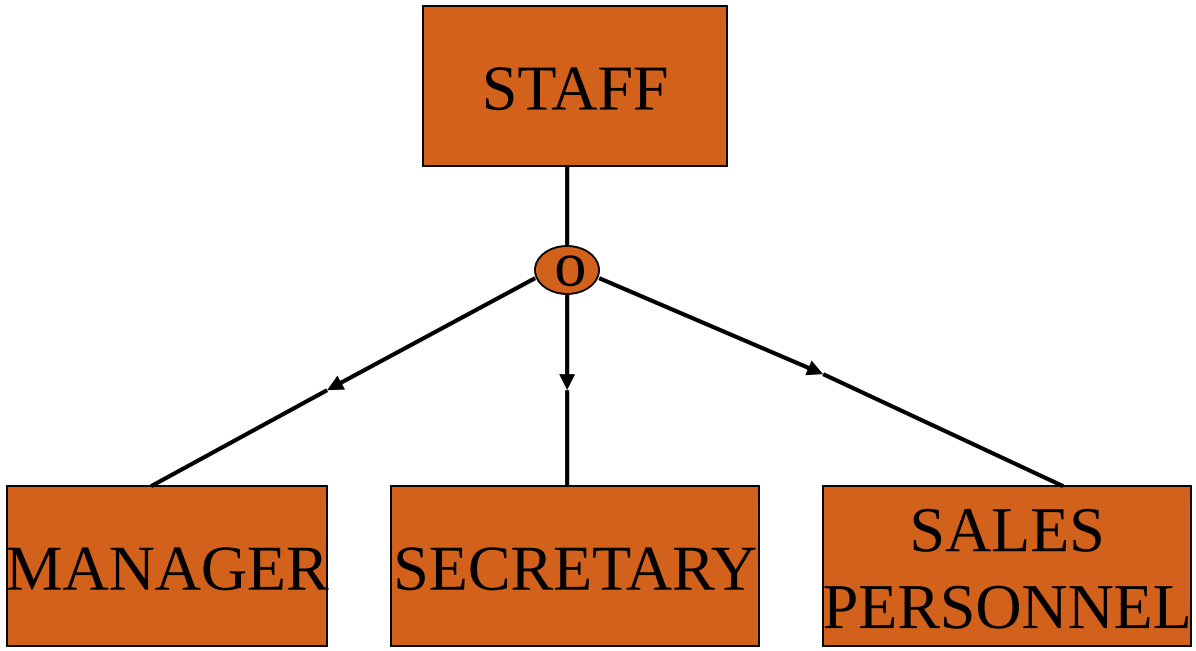
Disjoint Constraints

Disjoint constraint when an entity can be a member of only one of the subclasses of the specialization.



Disjoint Constraints

Non-disjoint constraints: an entity is a member of more than one subclass of specialization. Entity types may **overlap.**

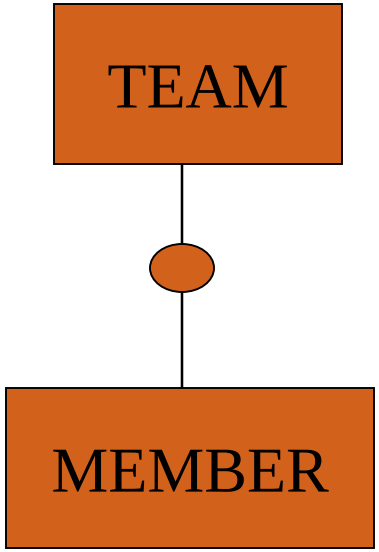


Aggregation

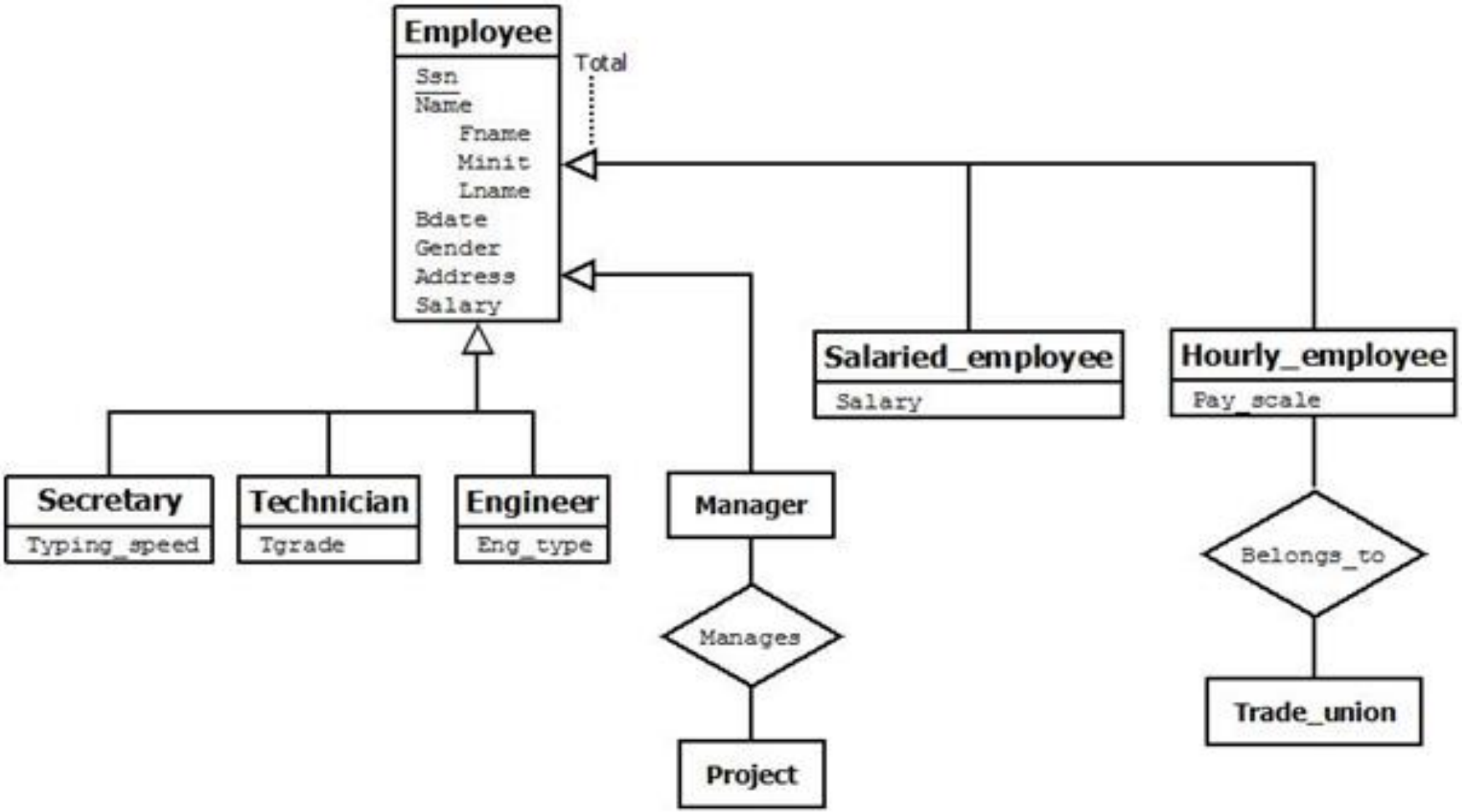
- Represents a “part-of” relationship between entity types, where one represents the ‘whole’ and the other the ‘part’.

- No inherited attributes; each entity has its own unique set of attributes.

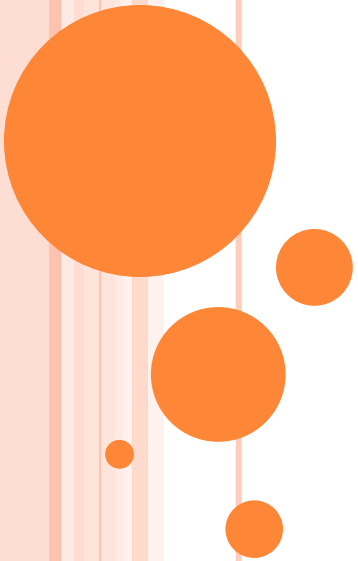
AGGREGATION



EERD-Exercise draw the following using Chen notation

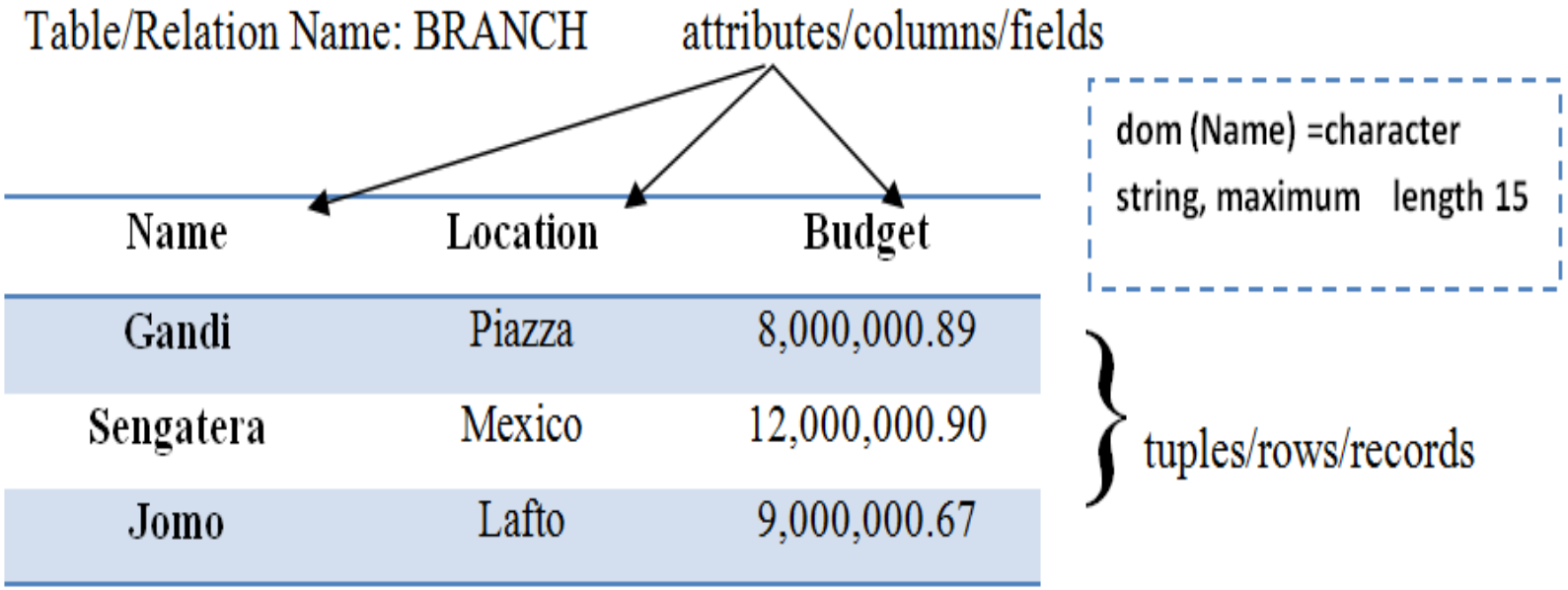


Relational Model



RELATIONAL MODEL

In the relational model, all data is logically structured within relations (tables).



Relational Data Structure

- **Relation** is a table with columns & rows. Holds information about entities.
- **Attribute** is a named column of a relation.
- **Tuple** is a row of a relation.
- **Degree** of a relation is the number of attributes it contains.
- **Cardinality** of a relation is the number of tuples it contains.

Relational Data Structure

- **Domain** is the set of allowable values for one or more attributes. Every attribute in a relation is defined on a domain.

- **Relational database** is a collection of normalized relations with distinct relation names.

DOMAINS

STUDENT

StudentNo	LName	FName	Initial	DOB	GPA
4170010	Al-Saleh	Amal	M.	04-06-78	3.91
4182000	Al-Ghanem	Nora	A.	02-12-79	4.20
4182034	Al-Fahad	Laila	A.	01-11-74	4.01

Attribute	Domain Name	Definition
StudentNo	Student Number	Digits: size 7
LName	Last Name	Character: size 15
FName	First Name	Character: size 15
Initial	Initial	Character: size 3
DOB	Date of Birth	Date: range 01-01-20, format dd-mm-yy
GPA	Great Point Average	Real: size 3, decimal 2, range 0-5

Properties of Relations

- The relation has a name that is distinct from all other relation names in the relational DB.
- Each cell of the relation contains exactly single value.
- Each attribute has a distinct name.
- The values of an attribute are all of the same domain.
- Each tuple is distinct. There are no duplicate tuples.
- The order of attributes has no significance.
- The order of tuples has no significance; theoretically.

Relational Keys

- **Candidate key (CK)** is an attribute, or set of attributes, that uniquely identifies a tuple, and no proper subset is a CK within the relation.

- **Primary Key (PK)** is the CK that is selected to identify tuples uniquely within the relation.

- **Foreign Key (FK)** is an attribute, or set of attributes, within one relation that matches the CK of some relation. Used to represent relationship between tuples of two relations.

Relational Keys

STUDENT

StudentNo	LName	FName	Initial	DOB	GPA	Dept
4170010	Al-Saleh	Amal	M.	04-06-78	3.91	D001
4182000	Al-Ghanem	Nora	A.	02-12-79	4.20	D001
4182034	Al-Fahad	Laila	A.	01-11-74	4.01	D002
4188134	Saod	Amal	F.	22-04-73	3.01	D003
4189860	Rashed	Rana	I.	30-01-78	2.31	D001

Foreign Key

Primary Key

DEPARTMENT

DeptNo	Department Name	Location
D001	Computer Science	Build # 20
D002	Business Administration	Build # 45
D003	Science	Build # 6

DB Relations

- **Relation schema** is a named relation defined by a set of attributes.

  If A_1, A_2, .., A_n are a set of attributes, then relation schema R_1 is: $R_1 (A_1, A_2, .., A_n)$

- **Relational schema** is a set of relation schemas, each with a distinct name.

  If R_1, R_2, .., R_n are a set of relation schemas, then relational schema R is: $R = \{R_1, R_2, .., R_n\}$

Relation Schema

STUDENT

StudentNo	LName	FName	Initial	DOB	GPA	Dept
4170010	Al-Saleh	Amal	M.	04-06-78	3.91	D001
4182000	Al-Ghanem	Nora	A.	02-12-79	4.20	D001
4182034	Al-Fahad	Laila	A.	01-11-74	4.01	D002
4188134	Saod	Amal	F.	22-04-73	3.01	D003
4189860	Rashed	Rana	I.	30-01-78	2.31	D001

STUDENT (StudentNo, Lname, Fname, Initial, DOB, GPA, Dept)

Relation Schema

DEPARTMENT

DeptNo	Department Name	Location
D001	Computer Science	Build # 2
D002	Business Administration	Build # 4
D003	Science	Build # 6

DEPARTMENT (DeptNo, Department Name, Location)

VIEWS

- **User View:** Describes that part of database that is relevant to a particular user or application area. (i.e. subset of the database)

- **Relation View:** Essentially some subset of the relations, *usually called "view".*

VIEWS…

- **Base relation** is a named relation corresponding to an entity in the conceptual schema, whose tuples are physically stored in the DB.

- **View** is a derived relation. Virtual, may not exist, but dynamically derived from one or more base relations.
  - The only information about a view that is stored in the database is its *structure*.

- The external model can consist of both conceptual level relations (base relations) and derived views.

VIEWS…

Views

View

STUDENT_GPA

StudentNo	GPA
4170010	3.91
4182000	4.20
4182034	4.01
4188134	3.01

Base Relation

STUDENT

StudentNo	LName	FName	Initial	DOB	GPA	Dept
4170010	Al-Saleh	Amal	M.	04-06-78	3.91	D001
4182000	Al-Ghanem	Nora	A.	02-12-79	4.20	D001
4182034	Al-Fahad	Laila	A.	01-11-74	4.01	D002
4188134	Saod	Amal	F.	22-04-73	3.01	D003

Purpose of Views

- Provides security mechanism by hiding parts of the DB from certain users.
- Customize data to user's needs, so that the same data can be seen by different users in different ways.
- Simplify complex operations. It allow you to work with data from different tables simultaneously.
- Supports logical data independence.

Relational Integrity

- **Data integrity refers to the validity, consistency, and accuracy of the data in the database.**
- **Integrity rules** are constraints that apply to all instances of the DB.
- Two integrity rules for the relational model:
  - Entity integrity
  - Referential integrity

Relational Integrity…

- **Entity Integrity**: Ensures that there are no duplicate records within the table. In a base relation, no attribute of a PK can be null.

- **Referential Integrity:** If a FK exists in a relation, either the FK value must match a CK value of some tuple in its home relation or the FK value must be wholly null.

- **Enterprise constraints:** rules specified by the users or DBA of the DB based on the ways an organization perceives and uses its data. (e.g. number of staff working in a branch is at most 20)

ER  Relational Model
(mapping ER to Relatinal model)

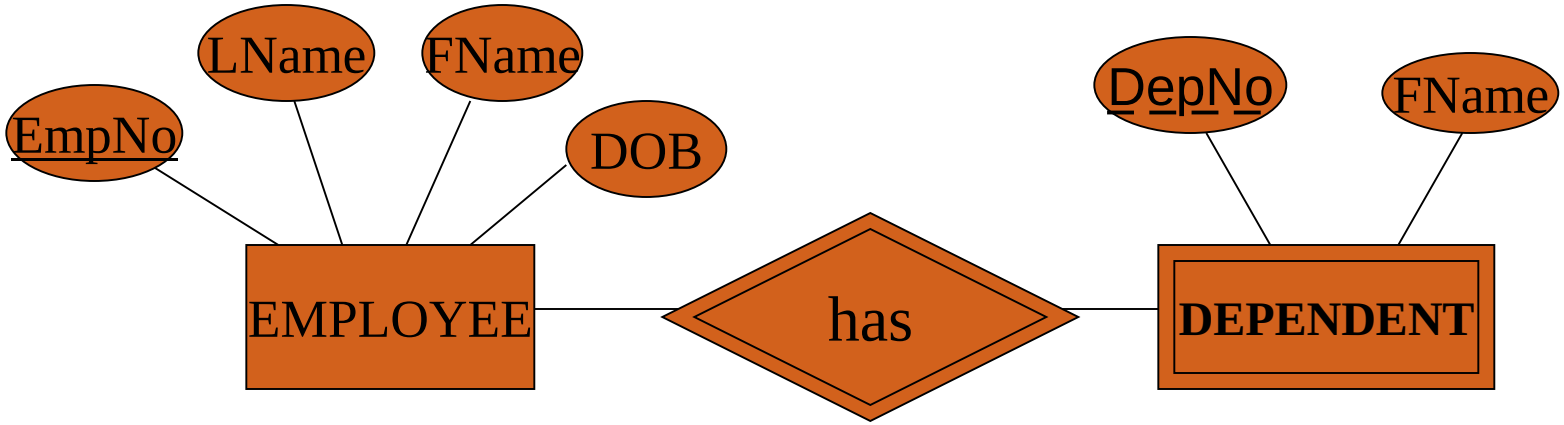
Entity Type → Relational Model

- Represent each entity type with a relation.
- Entity type attributes become the relation attributes.

STUDENT (StudentNo, Lname, Fname, Initial, DOB, GPA, Dept)
DEPARTMENT (DeptNo, Department Name, Location)

Weak Entity Type → Relational Model

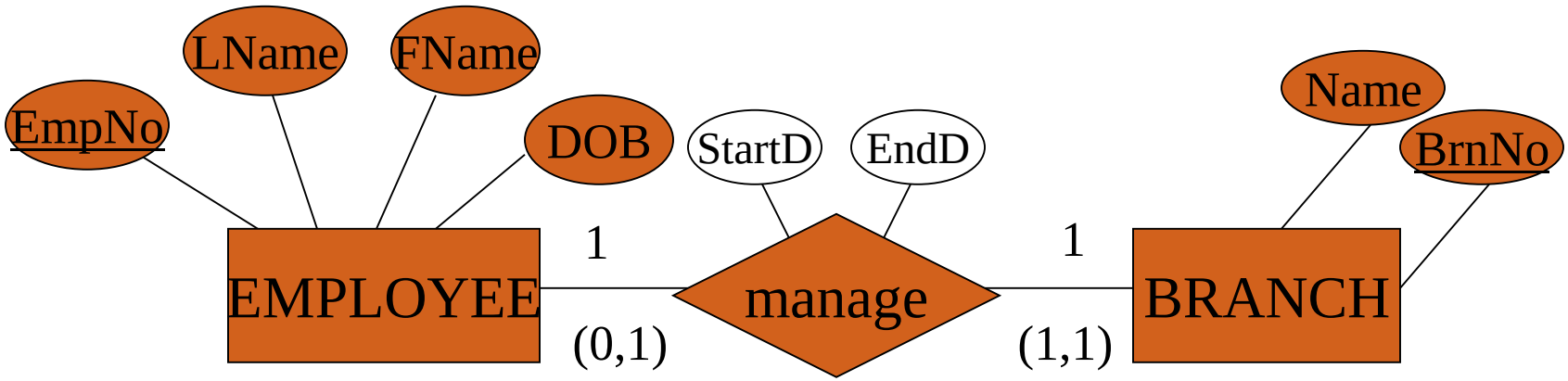
A weak entity type relation must include its key and its strong entity type PK as a FK. The combination of the two keys form the PK of the weak entity.



EMPLOYEE (EmpNo, Lname, Fname, DOB)
DEPENDENT (DepNo, EmpNo, FName)

1:1 Relationship → Relational Model

- Identify an entity type (S) (preferably total participator).
- Include the PK of the other entity (T) as a FK in S.
- Add attributes that describes the relationship to S.

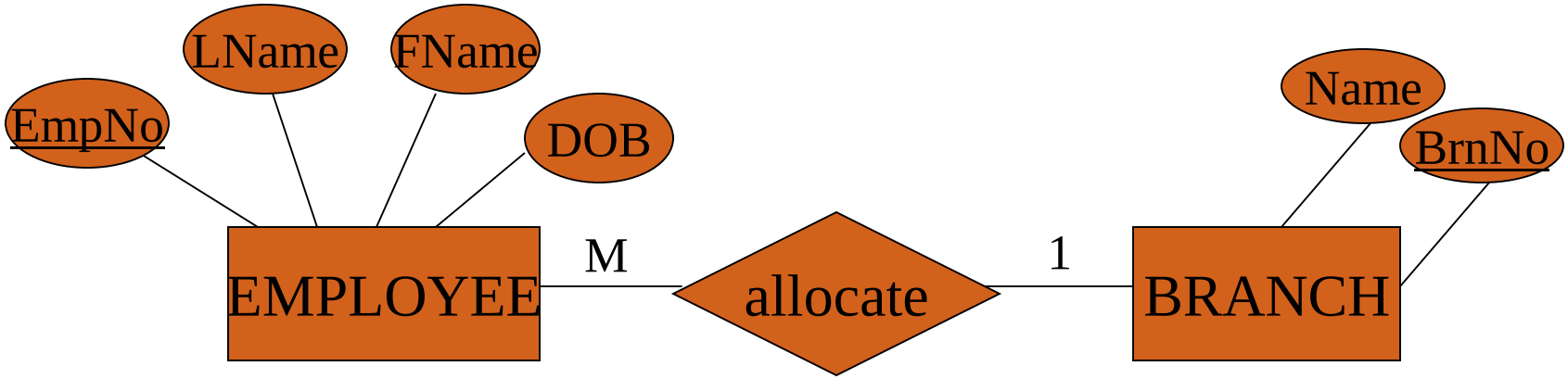


EMPLOYEE(EmpNo, Lname, Fname, DOB)

BRANCH(BrnNo, Name, EmpNo, StartDate, EndDate)

1:M Relationship → Relational Model

- Identify a participating entity type (S) on the m-side.
- Include the PK of the other entity type (T) as a FK in S.
- Add attributes that describes the relationship to S.

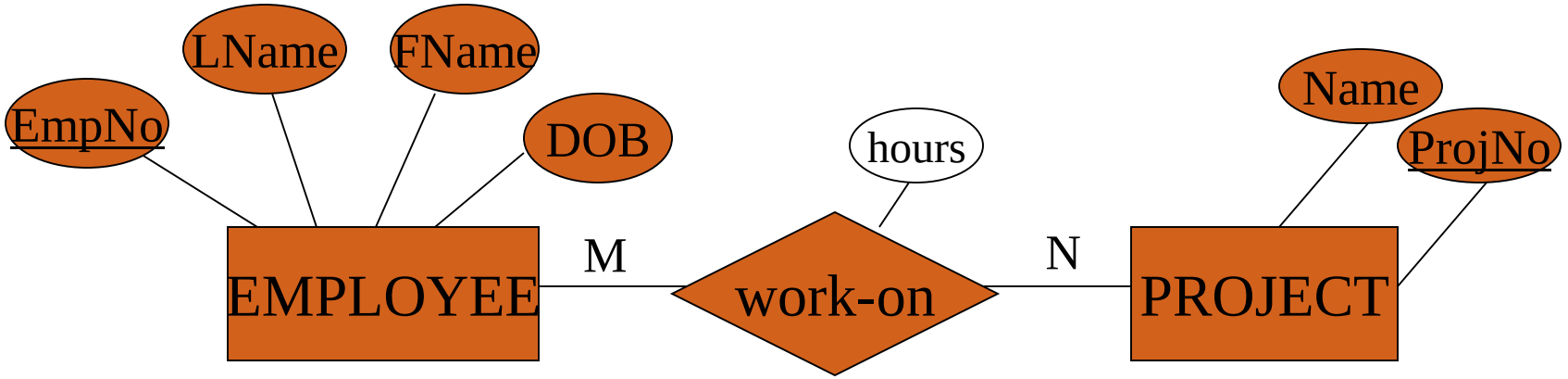


EMPLOYEE(EmpNo, Lname, Fname, DOB, BrnNo)

BRANCH(BrnNo, Name)

M:N Relationship → Relational Model

- Create a relation R to represent the relationship.
- Include the PK of participating entity types (T & S) as FK in R. The combination of the two FK will form the PK of R.
- Add attributes that describes the relationship to R.



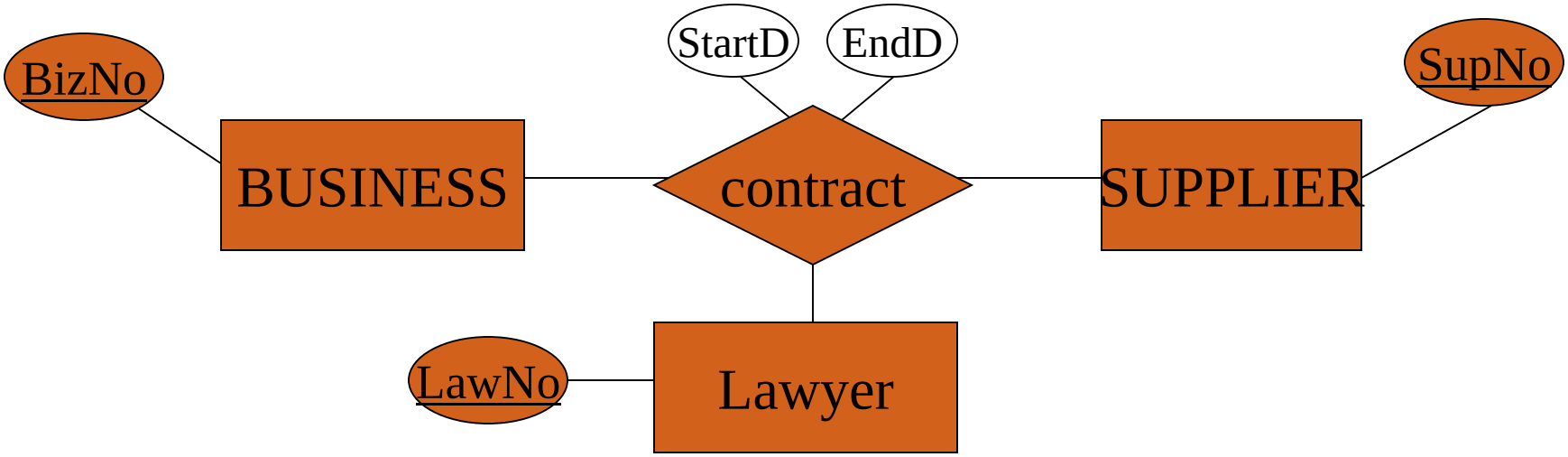
EMPLOYEE(EmpNo, Lname, Fname, DOB)

PROJECT(ProjNo, Name)

Work-on(EmpNo,ProjNo, hours)

n- ary Relationship → Relational Model

- Create a relation R to represent the relationship.
- Include the PK of the participating entities as FK in R. The combination of all FK form the PK of R.
- Add attributes that describes the relationship to R.

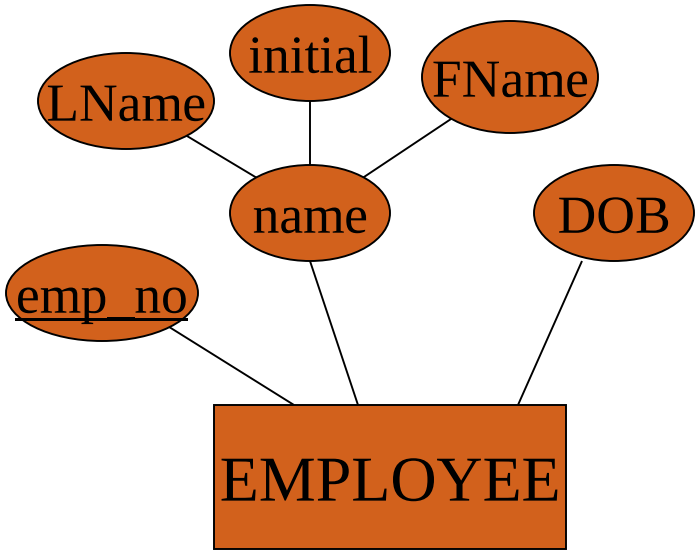


BUSINESS(BizNo) LAWYER(LawNo) SUPPLIER(SupNo)

contract(BizNo, SupNo, LawNo, StartDate, EndDate)

Composite Attribute → Relational Model

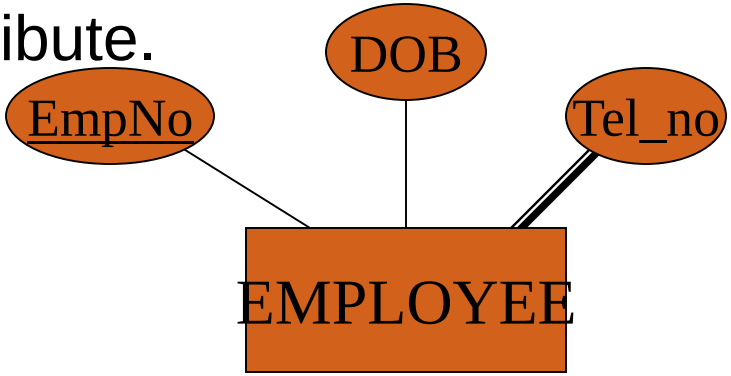
Include its simple components in the relation.



EMPLOYEE(EmpNo, Fname, initial, Lname, DOB)

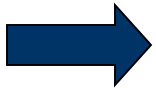
Multivalued Attribute → Relational Model

- Suppose A is a relation that contains the multivalued attribute.
- Create a relation R to represent the attribute.
- Include the PK of A as FK in R.
- The PK of R is the combination of the PK of A (FK) & the multivalued attribute.



EMPLOYEE(EmpNo, DOB)

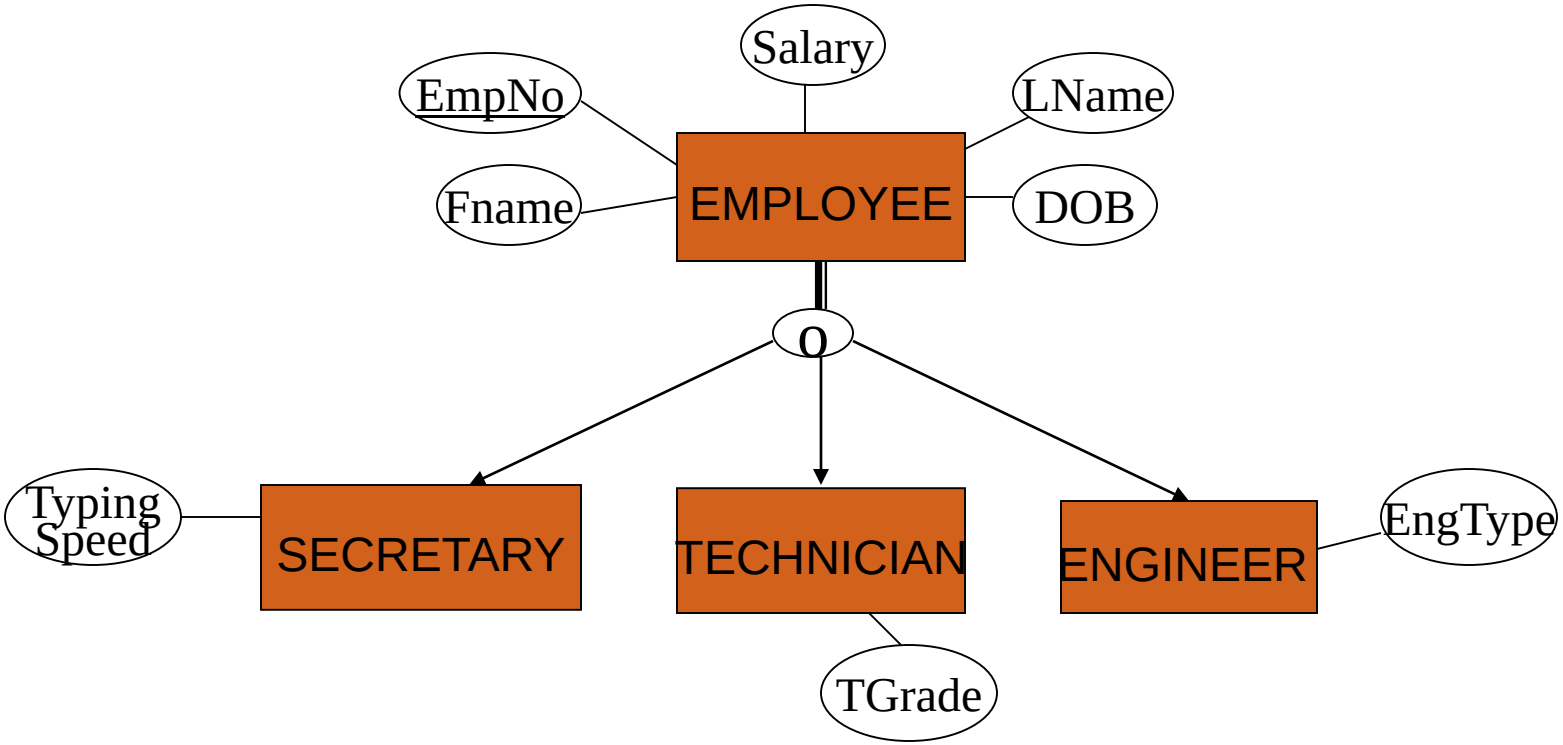
TELEPHONE(EmpNo, tel_no)

EER  Relational Model

Mandatory / Non-Disjoint

- Suppose specialization with subclasses (S_1, S_2, .., S_m) & a superclass C.
- Create a relation L to represent C with PK & attributes.
- Include the unshared attributes for each subclass S_i, $1 \leq i \leq m$.
- Add discriminator in L to distinguish the type of each tuple.

Mandatory / Non-Disjoint



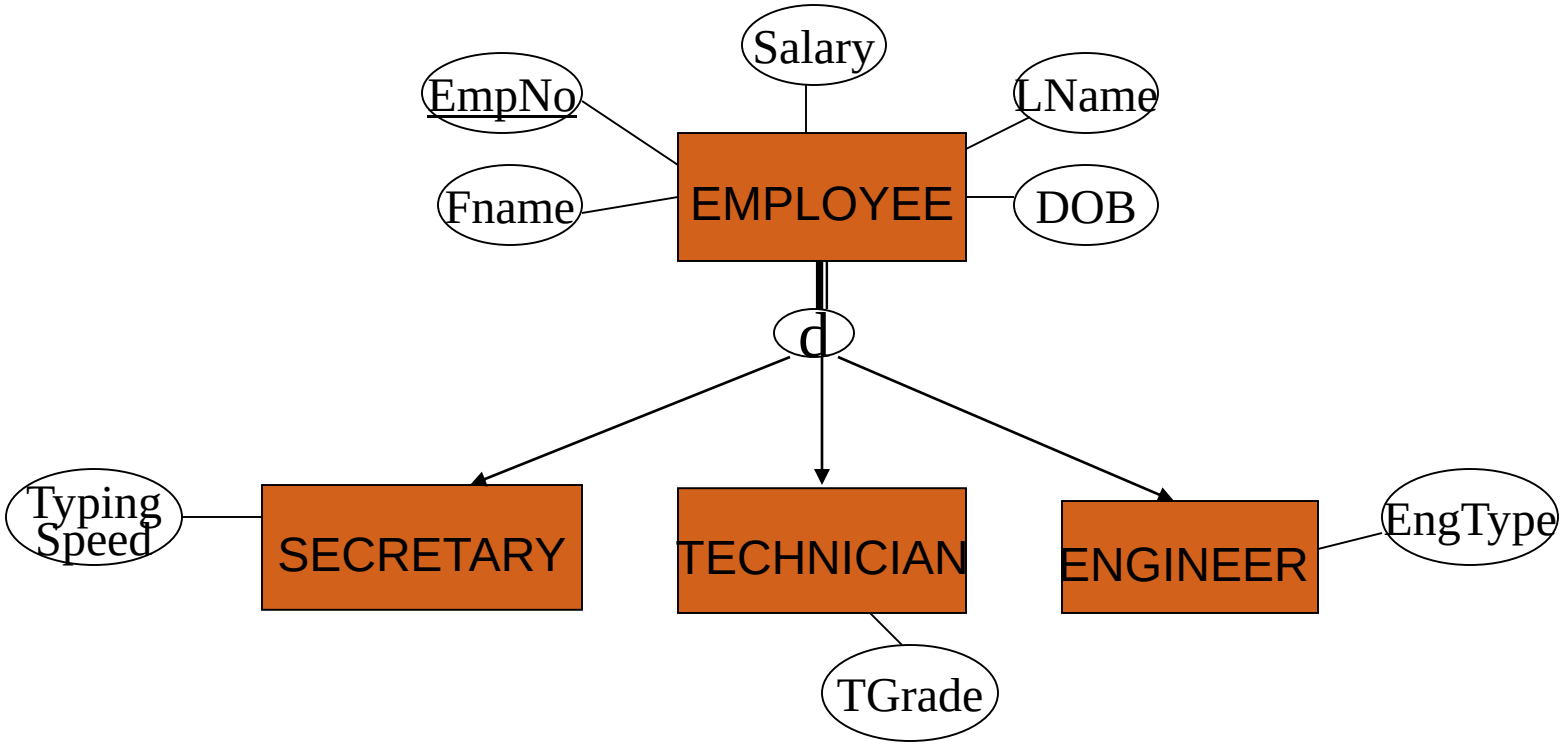
EMPLOYEE(EmpNo, Fname, Lname, DOB, Salary,TypingSpeed,TGrade, EngType, Secretary Flag, Technician Flag, Engineer Flag)

Mandatory / Disjoint

- Suppose specialization with subclasses (S_1, S_2, .., S_m) & a superclass C.

- Create a relation L_i, $1 \leq i \leq m$, to represent each combination of super/subclass.

Mandatory / Disjoint



SECRETARY(EmpNo, Fname, Lname, DOB, Salary,TypingSpeed)

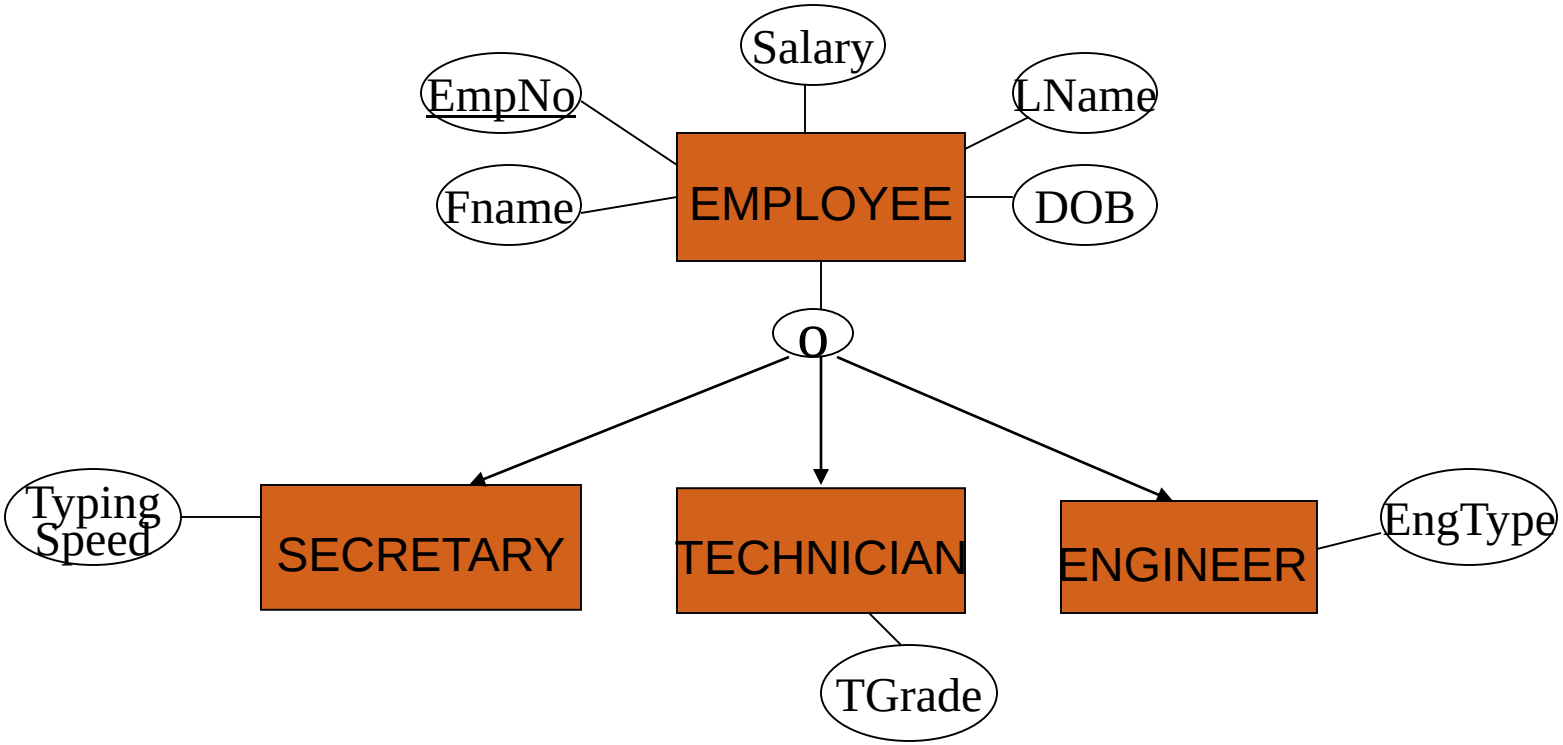
TECHNICIAN(EmpNo, Fname, Lname, DOB, Salary,Tgrade)

ENGINEER(EmpNo, Fname, Lname, DOB, Salary, EngType)

Optional / Non-Disjoint

- Suppose specialization with subclasses (S_1, S_2, .., S_m) & a superclass C.
- Create a relation L_1 to represent C with PK & attributes.
- Create a relation L_2 to represent all subclasses S_i, $1 \leq i \leq m$.
- Add discriminator in L_2 to distinguish the type of each tuple.

Optional / Non-Disjoint



EMPLOYEE(EmpNo, Fname, Lname, DOB, Salary)

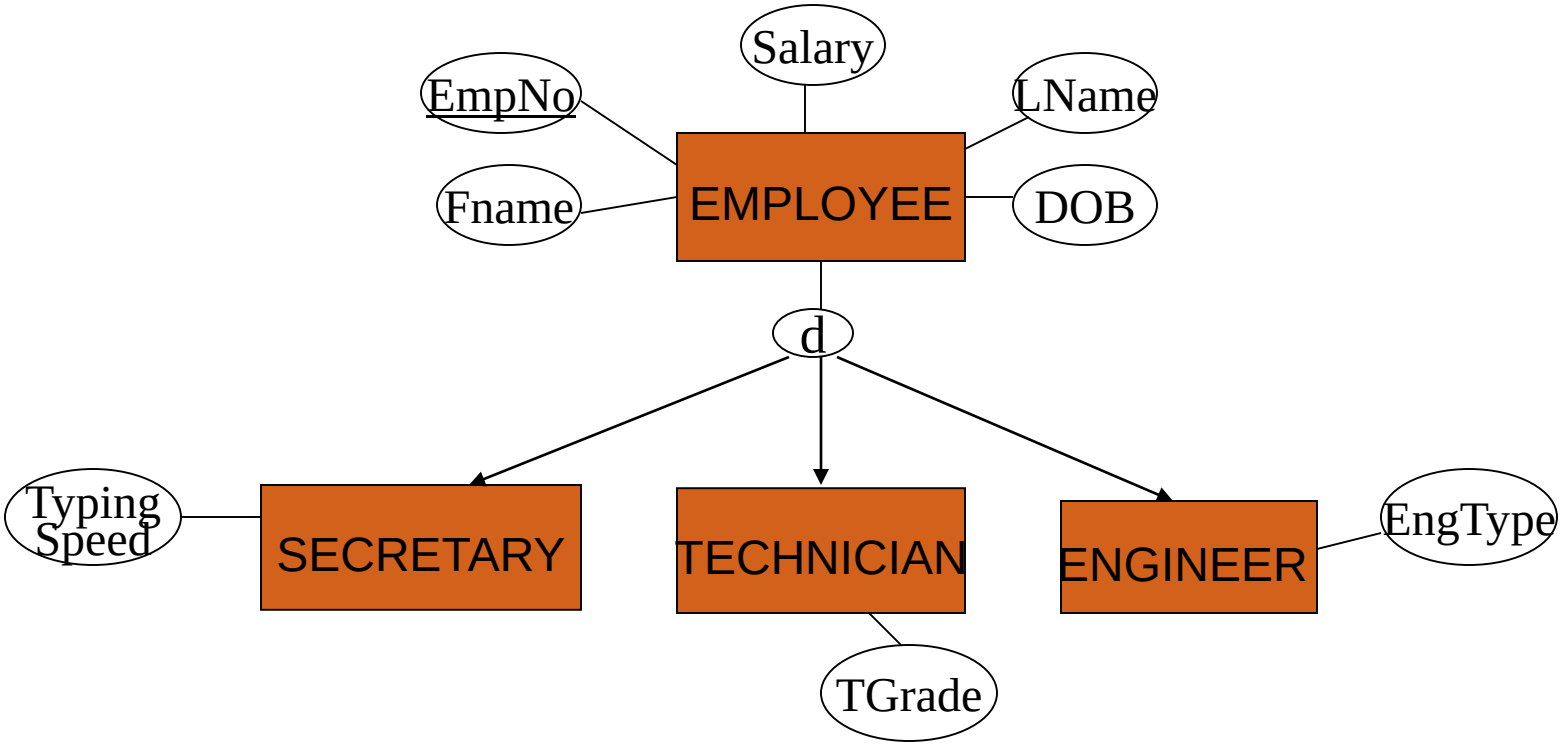
SUB-EMP(EmpNo, TypingSpeed,TGrade, EngType,

Secretary Flag, Technician Flag, Engineer Flag)

Optional / Disjoint

- Suppose specialization with subclasses (S_1, S_2, .., S_m) & a superclass C.
- Create a relation L to represent C with PK & attributes.
- Create a relation L_i to represent each subclass S_i, $1 \leq i \leq m$, and include the PK.

Optional / Disjoint



EMPLOYEE(EmpNo, Fname, Lname, DOB, Salary)

SECRETARY(EmpNo, TypingSpeed)

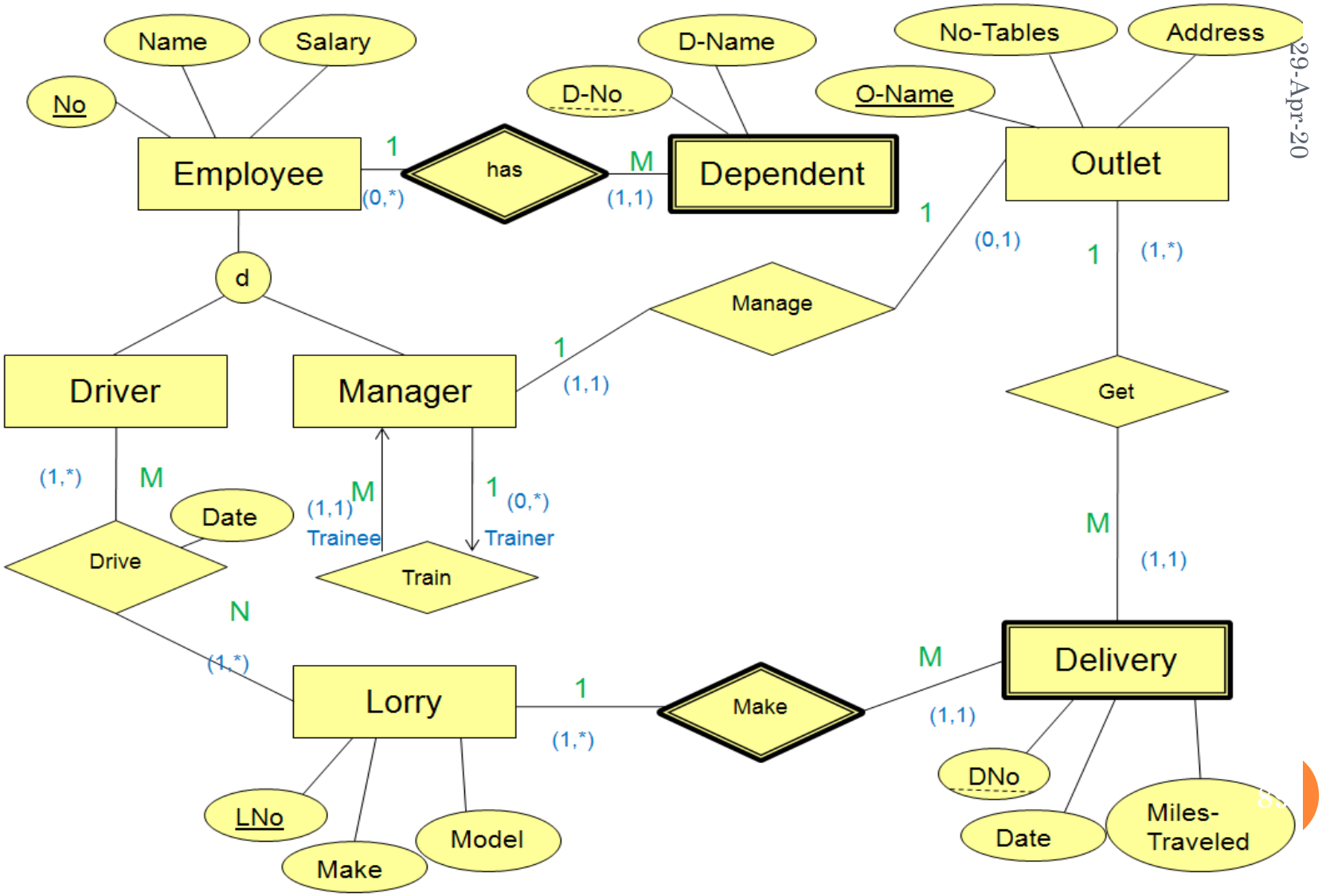
TECHNICIAN(EmpNo, Tgrade)

ENGINEER(EmoNo, EngType)

ER → Relational Model

ER Model	Relational Model
Entity Type	Relation
1:1 or 1:M Relationship Type	FK
M:N Relationship Type	Relation & 2 FK
n-ary Relationship Type	Relation & n FK
Simple attribute	Attribute
Composite attribute	Simple component attribute
Multivalued attribute	Relation and FK
Value set	Domain
Key attribute	PK

Question – Mapping EER



NORMALIZATION

Normalization

- The main objective in developing a logical data model is to create an accurate representation of the data, its relationships and constraints.
- To achieve this objective, we must identify a suitable set of relations.
- A technique we can use to help identify such relations is known as normalization.

The Purpose of Normalization

- Normalization- a technique for producing a set of relations with desirable properties, given the data requirement of an enterprise.
- Normalization reduces data redundancy in a database.
- By doing so it eliminates serious manipulation anomalies.

The Purpose of …

- Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form.
- Three normal forms were initially proposed, called first normal form (1NF), second normal form (2NF), and third normal form (3NF).
- Subsequently, a stronger definition of third normal form was introduced by R. Boyce and E.F. Codd, referred to as Boyce-Codd Normal Form (BCNF).

The purpose of …

- Higher normal forms that go beyond BCNF were introduced later such as fourth (4NF) and fifth (5NF) normal form.

- However, these later normal forms deal with practical situations that are very rare. We shall see only the first three in this training

THE PURPOSE OF …

- The process of normalization is a formal method that identifies relations based on their primary key (or candidate keys) and the functional dependencies among their attributes

- Normalization supports DB designers by presenting a series of tests, which can be applied to individual relations so that a relational schema can be normalized to a specific form to prevent the possible occurrence of update anomalies.

Data Redundancy and Update Anomalies

- A major aim of relational database design is to group attributes into relations so as to minimize data redundancy and thereby reduce the file storage space required by the implemented base relations.

- As an example to demonstrate these problems, consider the following two alternative ways of capturing data. The first alternative uses two tables, whereas the second one uses one table.

Alternative One:

Staff Relation

IDNo	fName	sAddress	Position	Salary	branchNo
SL21	Alie	Arat Killo	Manager	3000	B5
SG37	Chaltu	Megenagna	Snr Asst	1200	B3
SG14	Kiflu	Merkato	Deputy	1800	B3
SA9	Kassa	Shiro meda	Assistant	900	B7
SG5	Hirut	Arat Killo	Manager	2400	B3
SL41	Taye	Merkato	Assistant	900	B5

Branch Relation

branchNo	bAddress	btelNo
B5	Arada	55 66 77
B7	Mexico	66 77 88
B3	Merkato	77 88 99

Alternative Two:-

Staff_Branch Relation

Staff_No	fName	sAddress	position	salary	branchNo	bAddress	telNo
SL21	Alie	Arat Killo	Manager	3000	B5	Arada	55 66 77
SG37	Chaltu	Megenagna	Snr Asst	1200	B3	Merkato	77 88 99
SG14	Kiflu	Merkato	Deputy	1800	B3	Merkato	77 88 99
SA9	Kassa	Shiro Meda	Assistant	900	B7	Mexico	66 77 88
SG5	Hirut	Arat Killo	Manager	2400	B3	Merkato	77 88 99
SL41	Taye	Merkato	Assistant	900	B5	Arada	55 66 77

Insertion Anomalies

There are two main types of insertion anomalies.

These are demonstrated by using the Staff_Branch relation given earlier as follows.

1. To insert the details of new members of staff into the Staff_Branch relation, we must also insert the details of the branch at which the staff are to be posted.

   For example, to insert the details of new staff located at branch number B7, we must enter the correct details of the branch number B7 so that the branch details are consistent with the values for branch B7 in other rows of the Staff_Branch relation.

Insertion Anomalies continued…

2. To inset details of a new branch that currently has no member of staff into the Staff_Branch relation, it is necessary to enter nulls into the attributes for Staff, such as iDNo.

- However, as iDNo is the primary key for the Staff_Branch relation, attempting to enter nulls for Staff_No violates entity integrity, and it is not allowed

Deletion Anomalies

- If we delete a row from the Staf_Branch relation that represents the last member of staff located at a branch, the details about that branch are also lost from the database.

- For example, if we delete the row for staff number SA9 from the Staff_Branch relation, the details relating to branch number B7 are also deleted from the database.

- But, deleting the row for staff number SA9 from the Staff relation, the details on branch number B7 remain unaffected in the Branch relation.

Modification Anomalies

- If we want to change the value of one of the attributes of a particular branch in the Staff_Branch relation; for example, the telephone number for branch number B3, we must update the rows of all staff located at that branch.
- If this modification is not carried out on all the appropriate rows of the Staff_Branch relation, the database will become inconsistent.
- In this example, branch number B3 may appear to have different telephone numbers in different staff rows.
- **Taking these problems into consideration, it can be clearly observed that it is better to keep data by using the two tables format**.

Functional Dependencies

- One of the main concepts associated with normalization is functional dependency.

- Functional dependency - describes the relationship between attributes in a relation.

  - If A and B are attributes of relation R, B is functionally dependent on A (denoted A→B), if each value of A is associated with exactly one value of B. (A and B may each consist of one or more attributes).

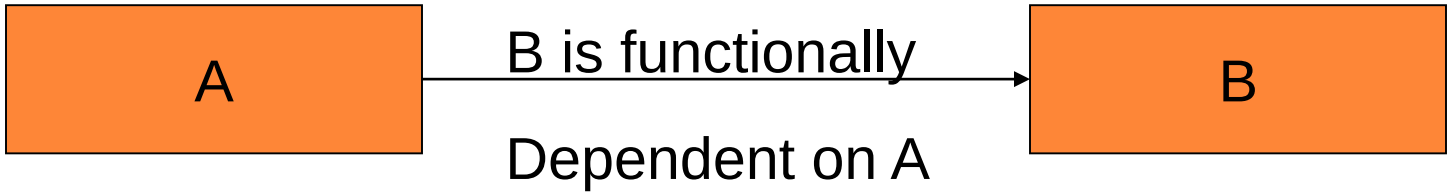
Functional Dependencies continued…

- Functional dependency is a property of the meaning or semantics of the attributes in a relation.
- The semantics indicate how attributes relate to one another, and specify the functional dependencies between attributes.
- When a functional dependency is present, the dependency is specified as a constraint between the attributes

Functional Dependencies continued…

- Consider a relation with attributes A and B, where attribute B is functionally dependent on attribute A.
  - If we know the value of A and we examine the relation that holds the dependency, we find only one value of B in all the rows that have a given value of A at any moment in time.
  - That is, when two rows have the same value of A, they also have the same value of B.
  - However, for a given value of B there may be several different value of A.

FUNCTIONAL DEPENDENCIES CONTINUED…



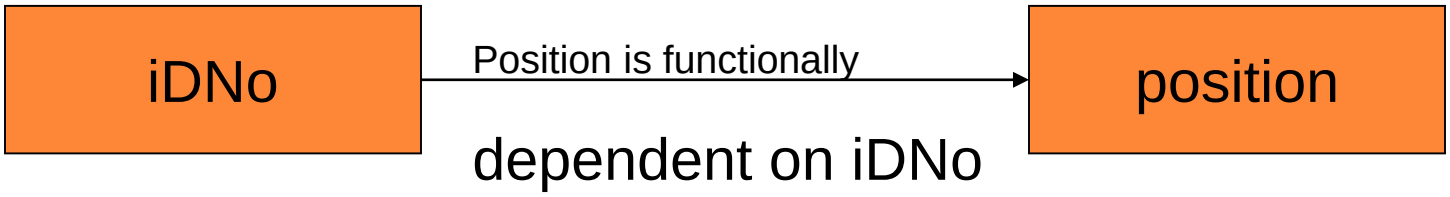
- ***Determinant*** – the determinant of a functional dependency refers to the attribute or group of attributes on the left hand side of the arrow.
- That is,when a functional dependency exists, the attribute or group of attributes on the left-hand side of the arrow is called the determinant. For example, A is the determinant of B in the above model.

EXAMPLE – FUNCTIONAL DEPENDENCIES

- Consider the attributes iDNo and position of the Staff relation given below. For a specific iDNo, for example SL21, we determine the position of that member of staff as Manager.

- In other words, the position attribute is functionally dependent on the iDNo, but the opposite is not true, iDNo is not functionally dependent on position. A member of staff holds one position, however, there may be several members of staff with the same position.

- For each id number, there is only one position, on the other hand, there are several id numbers associated with a position.



The Process of Normalization

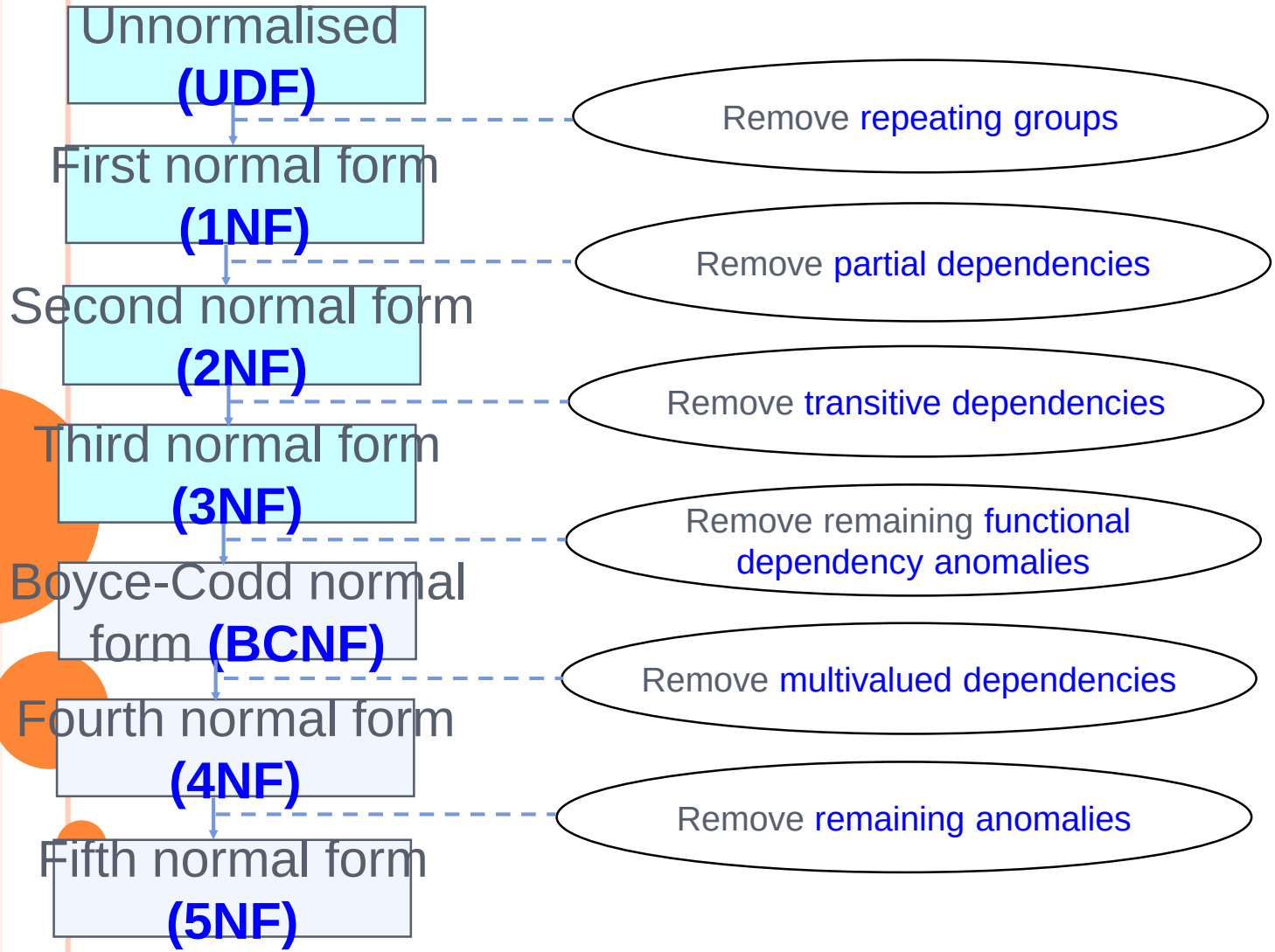
- Normalization is a formal technique for analyzing relations based on their primary key and functional dependencies.

- The technique involves a series of rules that can be used to test individual relations so that a database can be normalized to any degree.
- Normalization is often executed as a series of steps.

The Process of Normalization…

- Each step corresponds to a specific normal form that has known properties.
- The first normal form (1NF) is critical in creating appropriate relations. All the subsequent normal forms are optional.
- However, to avoid the update anomalies, it is normally recommended that we proceed to at least 3NF.

Stages of Normalization



First Normal Form (1NF)

- Unnormalized form (UNF) – a table that contains one or more repeating groups.
- 1NF – a relation in which the intersection of each row and column contains one and only one value.
- It states that the domain of an attribute must include only atomic (simple, indivisible) values.

Unnormalised Normal Form (UNF)

ORDER

Customer No: 001964
Name: Mark Campbell
Address: 1 The House
Leytonstone
E11 9ZZ

Order Number: 00012345
Order Date: 14-Feb-2002

Product Number	Product Description	Unit Price	Order Quantity	Line Total
T5060	Hook	5.00	5	25.00
PT42	Bolt	2.50	10	20.50
QZE48	Spanner	20.00	1	20.00

Order Total: 65.50

ORDER (orderNo, orderDate, custNo, custName, custAdd,
*(prodNo, prodDesc, unitPrice, ordQty, lineTotal)**, orderTotal)

Example - UNF to 1NF

ORDER (orderNo, orderDate, custNo, custName, custAdd,
(prodNo, prodDesc, unitPrice, ordQty, lineTotal),* orderTotal)

1. Remove the outermost repeating group (and any nested repeated groups it may contain) and create a new relation to

ORDER-1 (orderNo, orderDate, custNo, custName, custAdd, orderTotal)

(prodNo, prodDesc, unitPrice, ordQty, lineTotal)

2. Add to this relation a copy of the PK of the relation immediately

ORDER-1 (orderNo, orderDate, custNo, custName, custAdd, orderTotal)

(orderNo, prodNo, prodDesc, unitPrice, ordQty, lineTotal)

3. Name the new entity *(appending the number 1 to indicate 1NF)*

ORDER-LINE-1 (orderNo, prodNo, prodDesc, unitPrice, ordQty, lineTotal)

4. Determine the PK of the new entity

ORDER-LINE-1 (orderNo, prodNo, prodDesc, unitPrice, ordQty, lineTotal)

Second Normal Form (2NF)

- 2NF is based on the concept of full functional dependency.

- Full functional dependency – indicates that if A (which is primary key) and B (non key attribute) are attributes of a relation, we say B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.

- In other words, A functional dependency A→ B: is said to be a full functional dependency if removal of any attribute from A results in the dependency not being sustained any more, otherwise it is called partially dependency.

Second Normal Form (2NF)

Definition: A relation is in 2NF if, and only if, it is in 1NF and every non-key attribute is fully dependent on the primary key.

Steps from 1NF to 2NF:

- Remove the attributes that are only partially functionally dependent on the composite key, and place them in a new relation.
- Add to this relation a copy of the attribute(s) which are the determinants of these attributes. These will automatically become the primary key of this new relation.
- Name the new entity *(appending the number 2 to indicate 2NF)*
- Rename the original entity *(ending with a 2 to indicate 2NF)*

Example - 1NF to 2NF

ORDER-LINE-1 (order-no, prod-no, prod-desc, unit-price, ord-qty, line-total)

1. Remove the attributes that are only partially functionally dependent on the composite key, and place them in a new relation.

ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-desc, unit-price)

2. Add to this relation a copy of the attribute(s) which determines these attributes. These will automatically become the primary key of the new relation.

ORDER-LINE-1 (order-no, prod-no, ord-qty, line-total)

(prod-no, prod-desc, unit-price)

3. Name the new entity *(appending the number 2 to indicate 2NF)*

PRODUCT-2 (prod-no, prod-desc, unit-price)

4. Rename the original entity *(ending with a 2 to indicate 2NF)*

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)

Third Normal Form (3NF)

- 3NF is based on the idea of transitive dependency.

- Transitive dependency: A condition where A, B, and C are attributes of a relation such that if A→B and B→C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C)

  - it exists when a non-key attribute determines another non-key attribute

Transitive Dependency

Definition: A transitive dependency exists when there is an intermediate functional dependency.

Formal Notation: If **A → B** and **B → C**, then it can be stated that the following transitive dependency exists:

A → B → C

Third Normal Form (3NF)

Definition: A relation is in 3NF if, and only if, it is in 2NF and every non-key attribute is non-transitively dependent on the primary key.

29-Apr-20

Steps from 2NF to 3NF:

- Remove the attributes that are transitively dependent on non-key attribute(s), and place them in a new relation.
- Add to this relation a copy of the attribute(s) which are the determinants of these attributes. These will automatically become the primary key of this new relation.
- Name the new entity *(appending the number 3 to indicate 3NF)*
- Rename the original entity *(ending with a 3 to indicate 3NF)*

Example - 2NF to 3NF

ORDER-2 (order-no, order-date, cust-no, cust-name, cust-add, order-total

1. Remove the attributes that are transitively dependent on non-key attributes, and place them in a new relation.

ORDER-2 (order-no, order-date, cust-no, order-total

(cust-name, cust-add)

2. Add to this relation a copy of the attribute(s) which determines these attributes. These will automatically become the primary key of the new relation.

ORDER-2 (order-no, order-date, cust-no, order-total

(cust-no, cust-name, cust-add)

3. Name the new entity *(appending the number 3 to indicate 3NF)*

CUSTOMER-3 (cust-no, cust-name, cust-add)

4. Rename the original entity *(ending with a 3 to indicate 3NF)*

ORDER-3 (order-no, order-date, cust-no, order-total

The resulting Relations which are each in 3NF are the following:-

ORDER-3 (order-no, order-date, cust-no, order-total)

CUSTOMER-3 (cust-no, cust-name, cust-add)

PRODUCT-2 (prod-no, prod-desc, unit-price)

ORDER-LINE-2 (order-no, prod-no, ord-qty, line-total)