UNIVERSITY of RWANDA

College of Science and Technology

School of ICT

Computer Science Department

Year 2

Lecturer: Muhayimana Odette
E-mail: muhayiodette06@gmail.com
Phone: 0788888571

CHAPIII: RELATIONAL MODEL

3.1 CODD's Rules

- In 1985, E.Frank Codd published a list of rules that became a standard way of evaluating a relational system.
- Those rules represent relational ideal and remain a goal for relational database designers.
- There are 12 rules, but there are no systems that will satisfy every rule

1. **Information Rule** : Data should be presented to the user in the tabular form
2. **Guaranteed Access Rule**: Every data element shouldbe unambiguously accessible
3. **Systematic Treatment of Null Values**: Null value are supported for representing missing information in a systematic way.

3.1 CODD's Rules

4. **Dynamic online Catalog Based on the Relational Model**: users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.
5. **Comprehensive Data Sublanguage Rule**: A database supports a clearly defined language to define the database, view the definition, manipulate the data and restrict some data values to maintain integrity
6. **View Updating Rule**: All views that are theoretically updatable must be updatable by the system
7. **High-level Insert, Update and Delete**: All records in a file must be able to be added, deleted , or updated with singular commands
8. **Physical Data Independence**: Changes in how data are stored or retrieved should not affect how a user accesses the data

3.1 CODD's Rules

9. **Logical Data Independence**: A user's view of data should be unaffected by its actual form in files.

10. **Integrity Independence**: Constraints on user input should exist to maintain data integrity

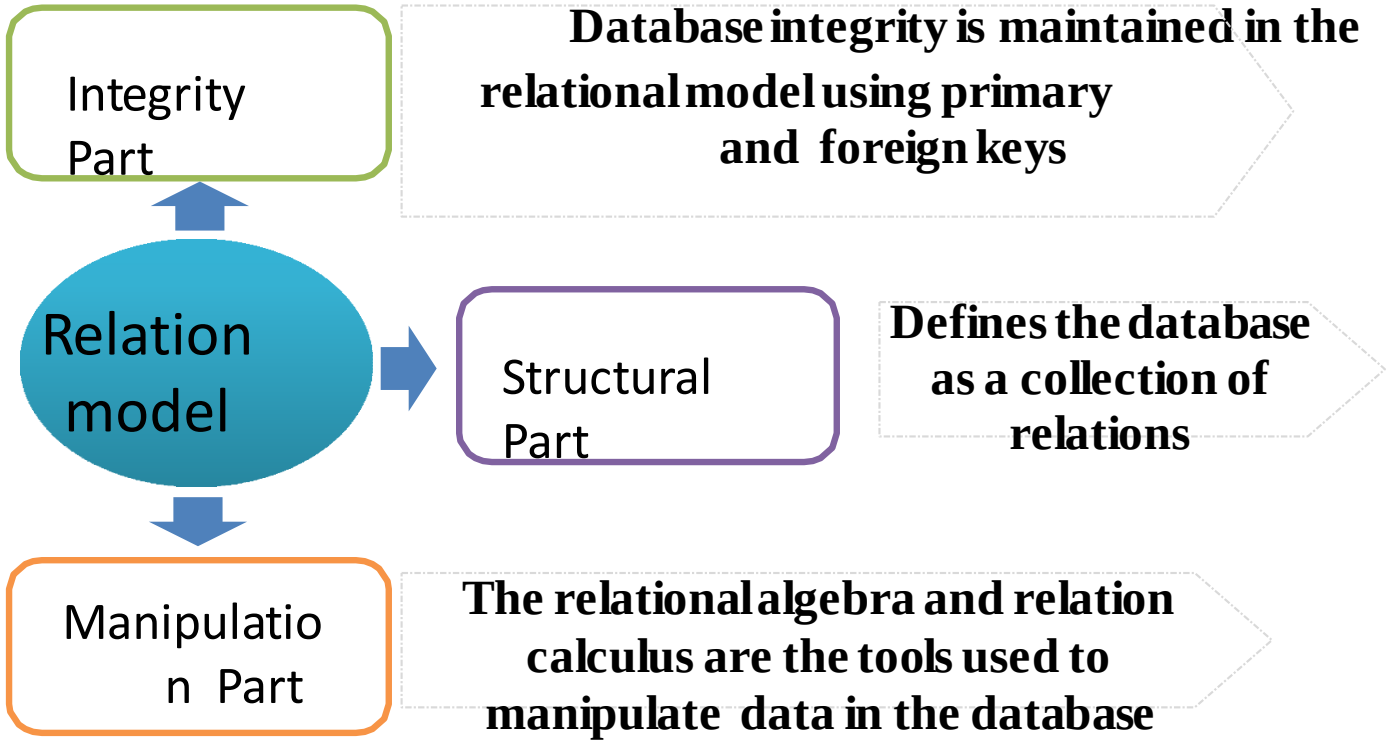
11. **Distribution Independence**: A database design should allow for distribution of data over several computer sites

12. **Non-subversion Rule**: Data fields that affect the organization of the database cannot be changed

13. **Rule Zero or Foundation Rule**: A relational database management system must manage its stored data using only its relational capabilities

3.2 Relational Data Model

- The relational model uses a *collection of tables* to represent both data and the relationships among those data
- Relational model is a combination of three components:



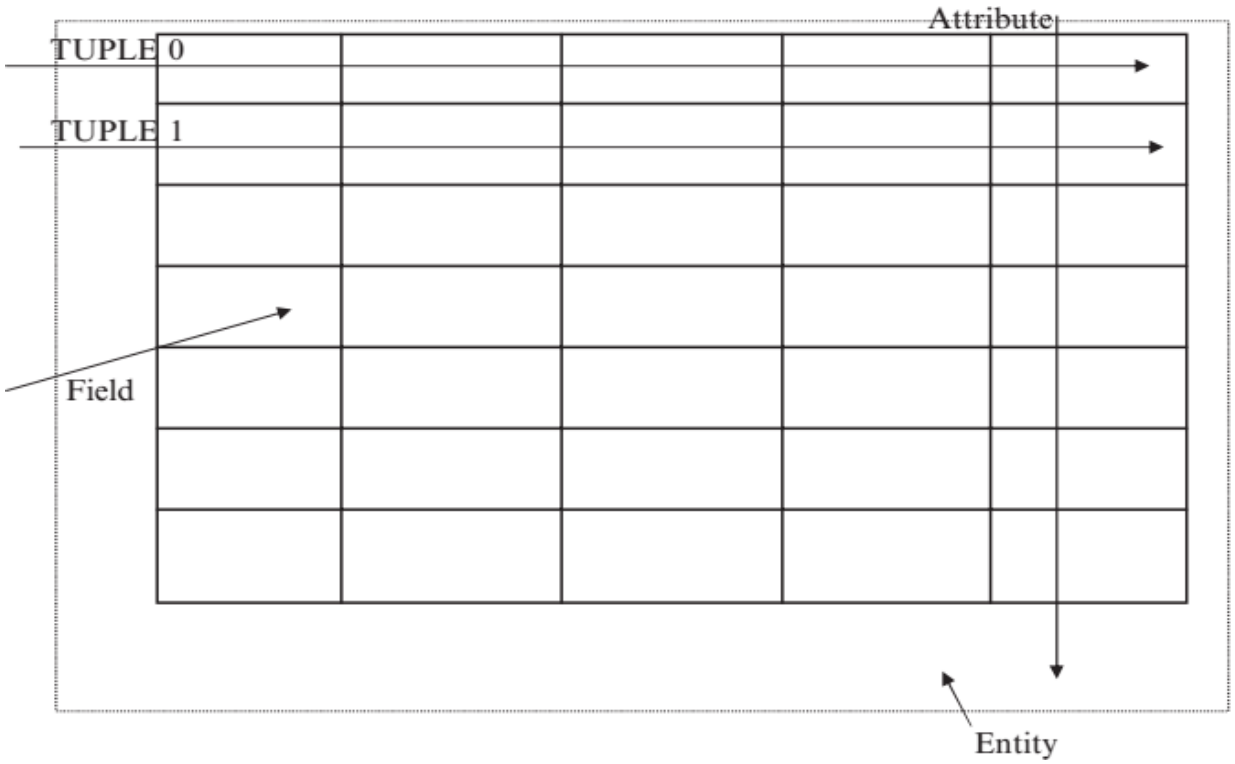
3.2 Relational Data Model

- Key features of relational data model:
  - Each row in the table is called ***tuple***
  - Each column in the table is called ***attribute***
  - The intersection of row with the column will have ***data value***
  - In relational model rows can be in any order
  - In relational model attributes can be in any order
  - **By definition, all rows in a relation are distinct. No two rows can be exactly the same**
  - Relation must have a key.
    - Keys can be a set of attributes
  - For each column of a table there is a set of possible values called ***its domain.***
    - **Domain** is the set of valid values for an attribute
    - The domain contains all possible values that can appear under that domain
  - ***Degree of the relation*** is the number of attributes (columns)in the relation
  - ***Cardinality of the relation*** is the number of tuples (rows) in the relation

3.2 Relational Data Model

- Terms commonly used by user, model and programmers:

User	Model	Programmer
✓ Row	✓ Tuple	✓ Record
✓ Column	✓ Attribute	✓ Field
✓ Table	✓ Relation	✓ File



3.2 Relational Data Model

- For a table to be a relation, the following rules must be held
  - The intersection of the row with the column should be contain single value (atomic value)
  - All data in a column are of same type (same domain)
  - Each column has a unique name (column order not significant)
  - Not two rows are identical (row order not significant)

- Example of Relational model (~**MOVIE**)

Movie Name	Director	Actor	Actress
Titanic	James Cameron	Leonardo	Kate Winslet
Autograph	Cheran	Cheran	Gopita
Roja	Maniratnam	Aravind	Madubala

Degree of relation = 4
Cardinality of relation = 3

Transforming the EER diagram into relations

The steps:

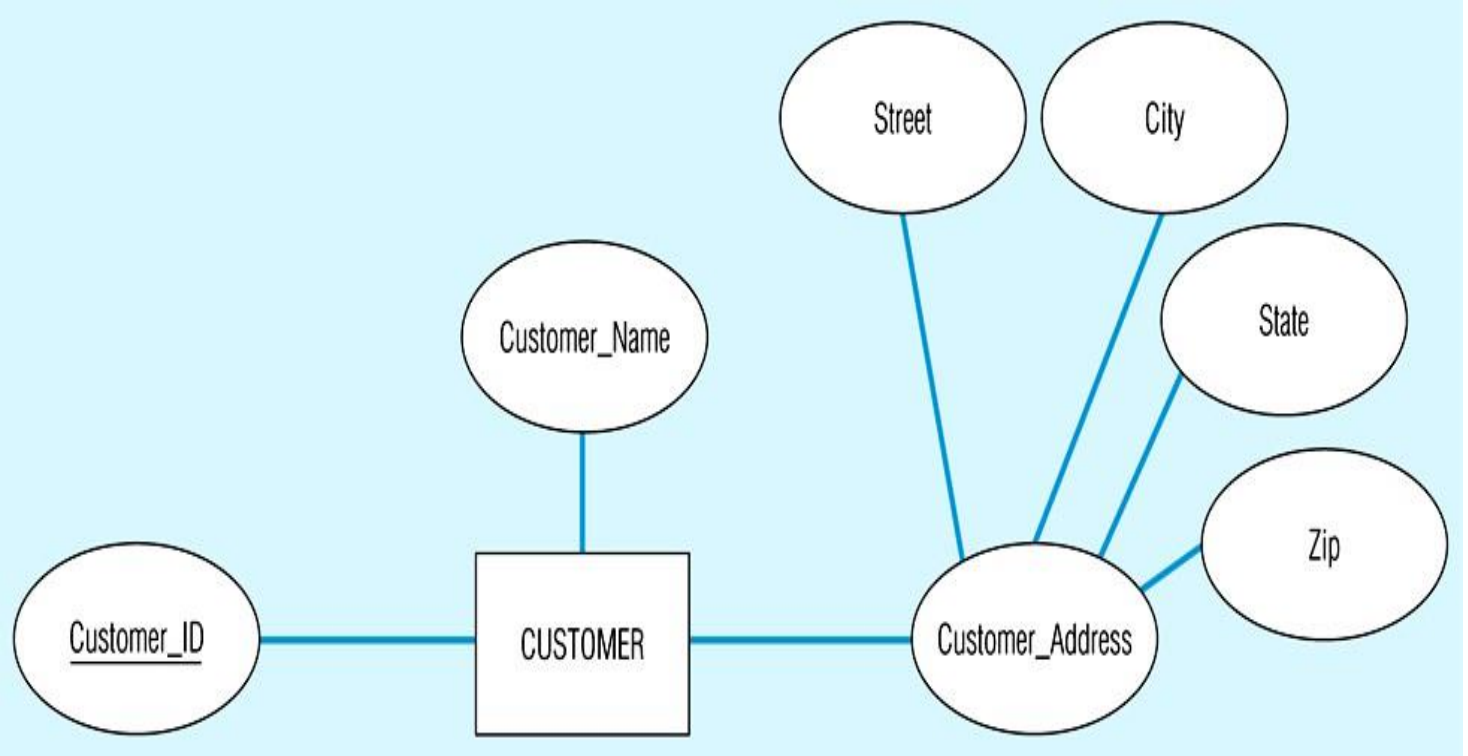
- Map regular entities
- Map weak entities
- Map binary relationships
- Map associative entities
- Map unary relationships
- Map ternary relationships
- Map supertype/subtype relationships

Transforming E-R diagrams into relations

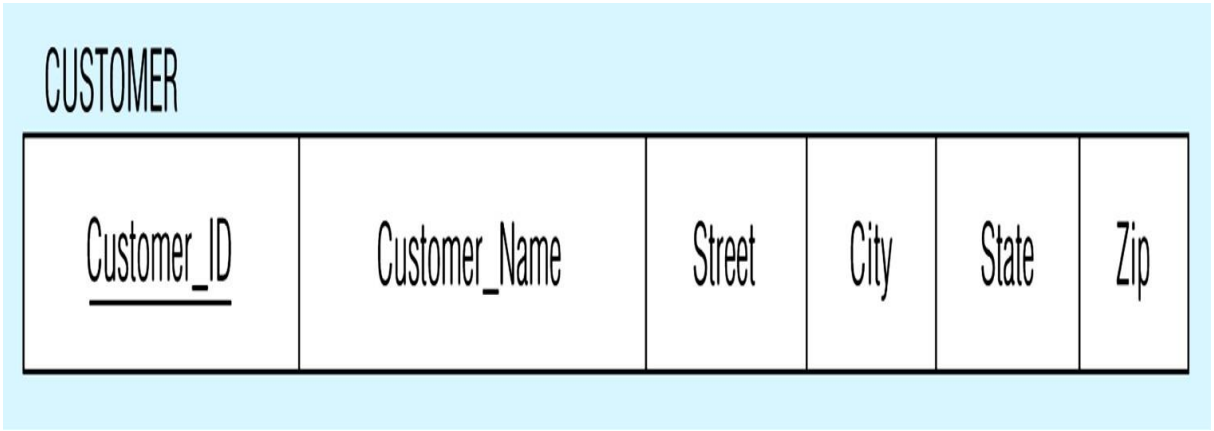
Mapping regular entities to relations

- Composite attributes: use only their simple, component attributes
- Multi-valued attributes: become a separate relation with a foreign key taken from the superior entity

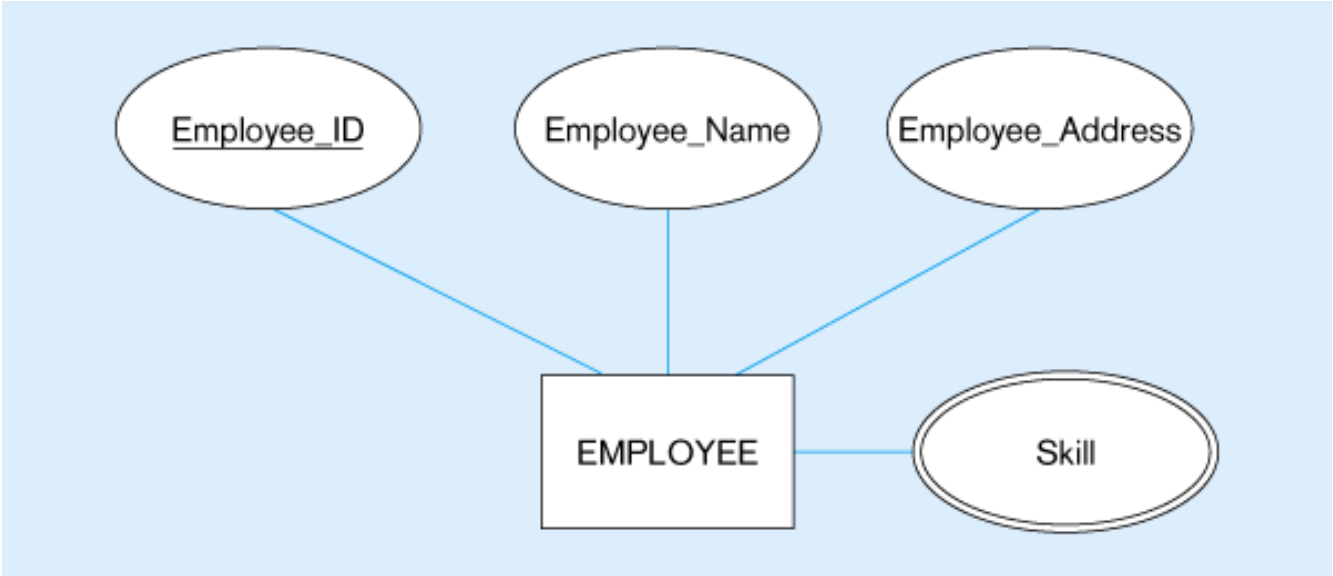
Mapping a composite attribute



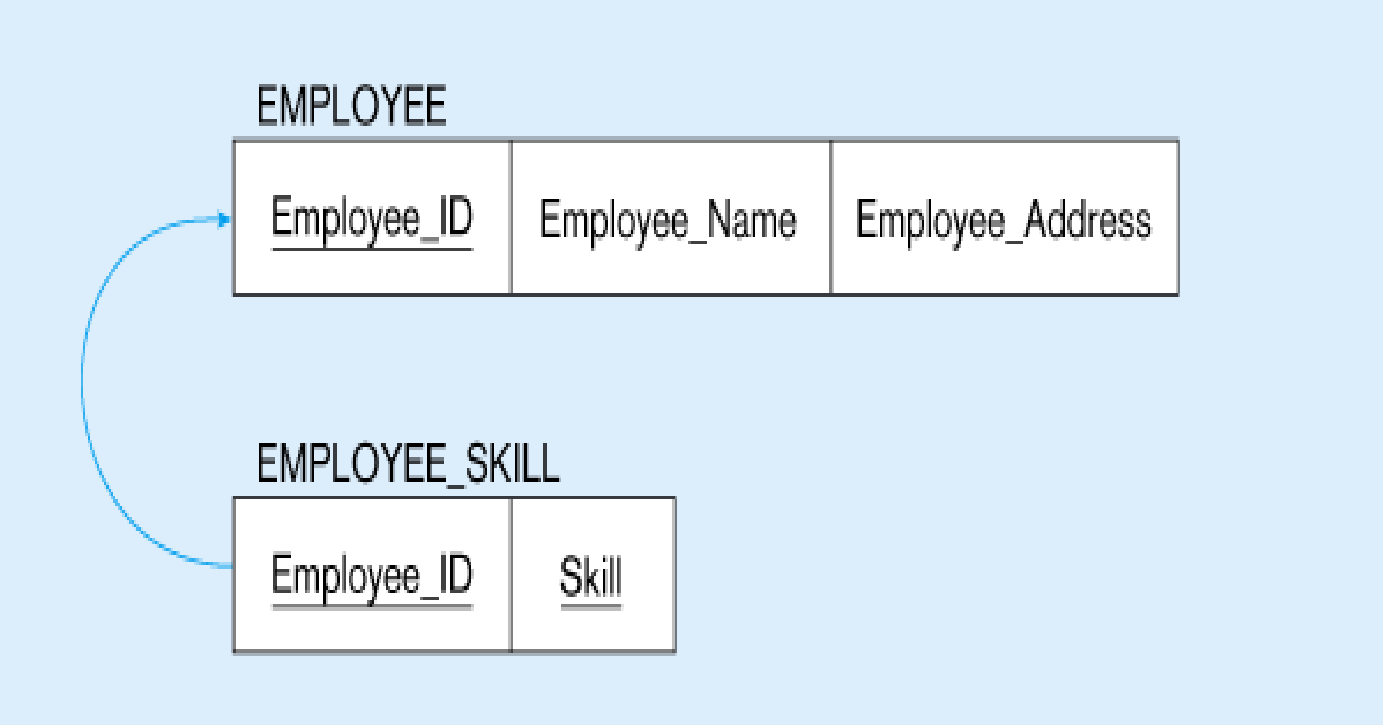
Looks like this using relational schema notation



Mapping a multivalued attribute



Looks like this using relational schema notation



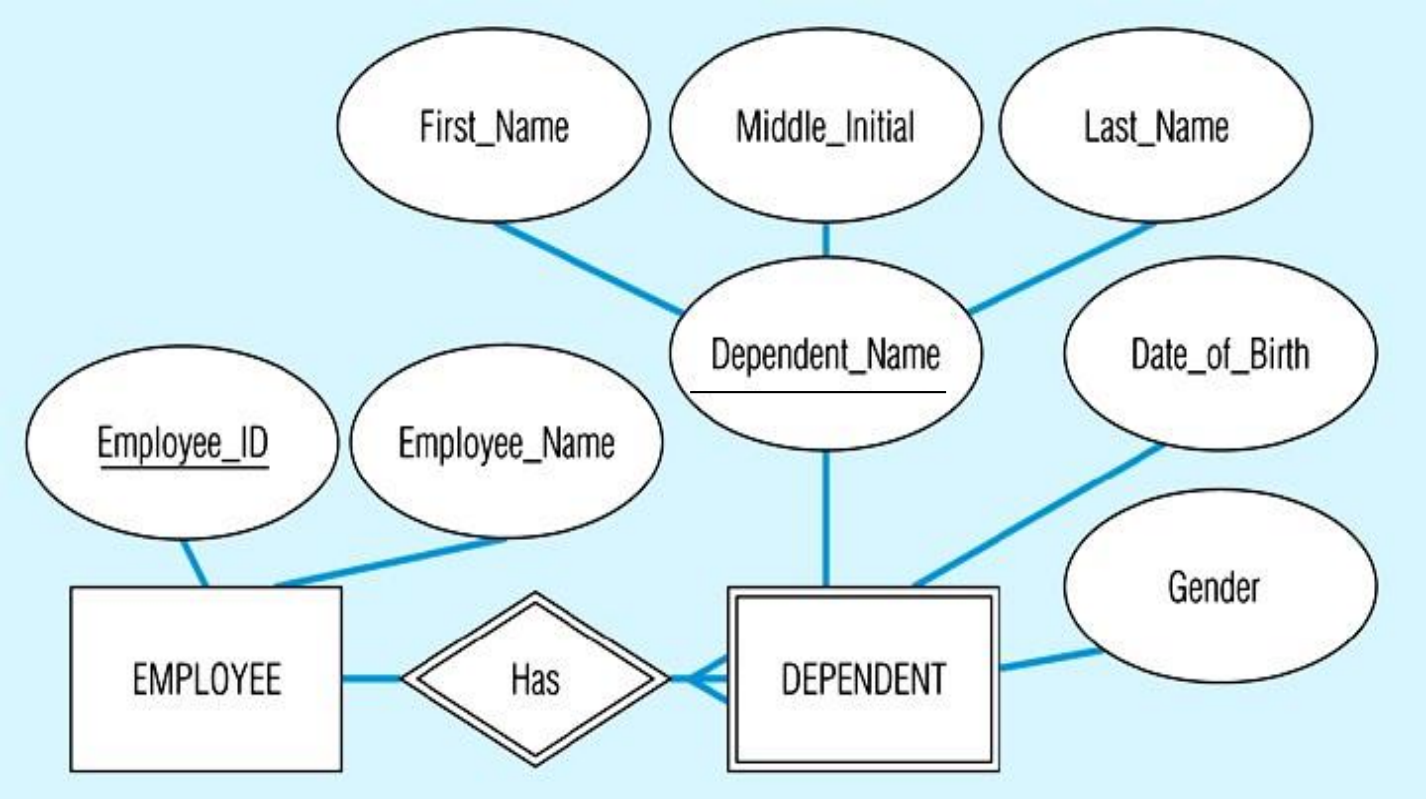
1–to–many relationship between original entity and new relation

Transforming E-R diagrams into relations

Mapping weak entities

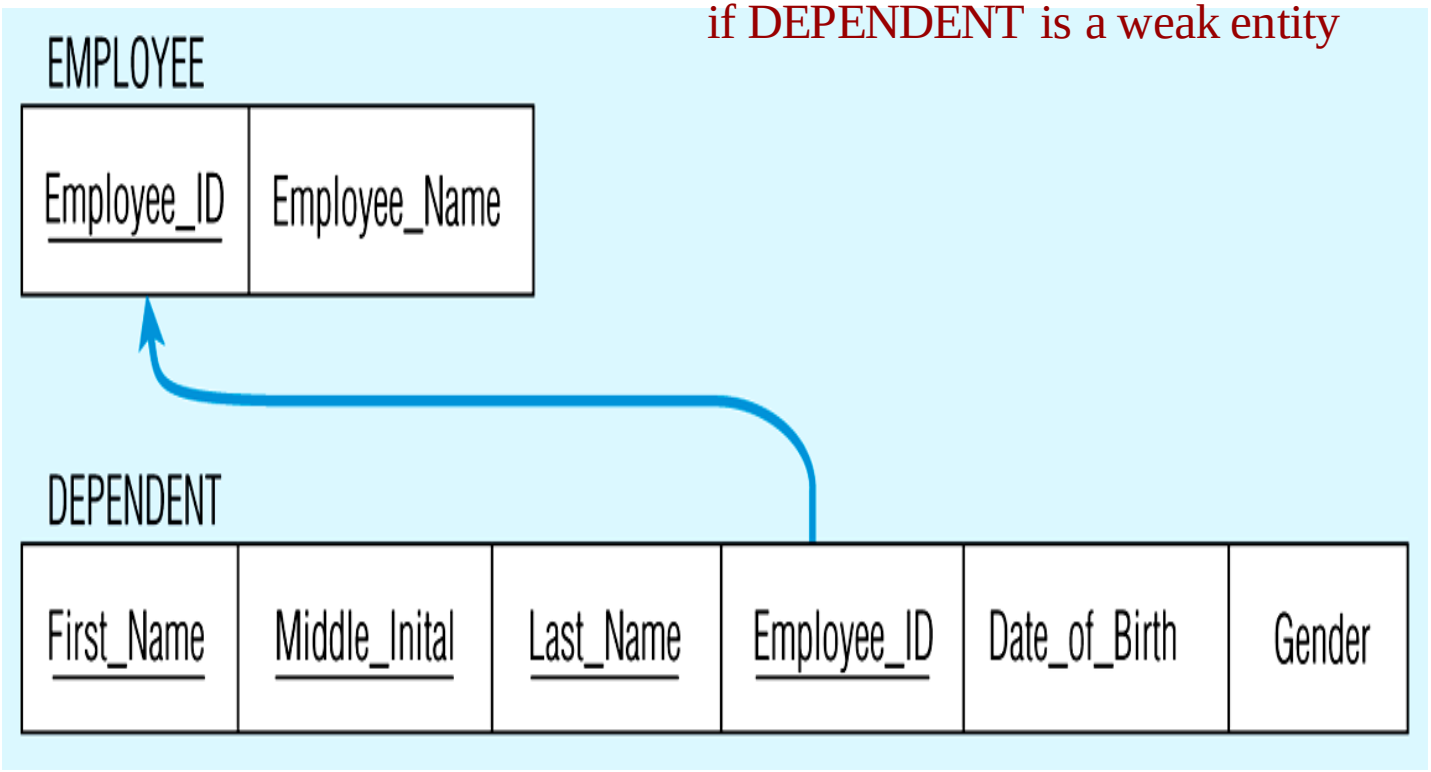
- Becomes a separate relation with a foreign key taken from the superior entity

Example of mapping a weak entity



Looks like this using relational schema notation

NOTE: the domain constraint for the foreign key should NOT allow *null* value if DEPENDENT is a weak entity

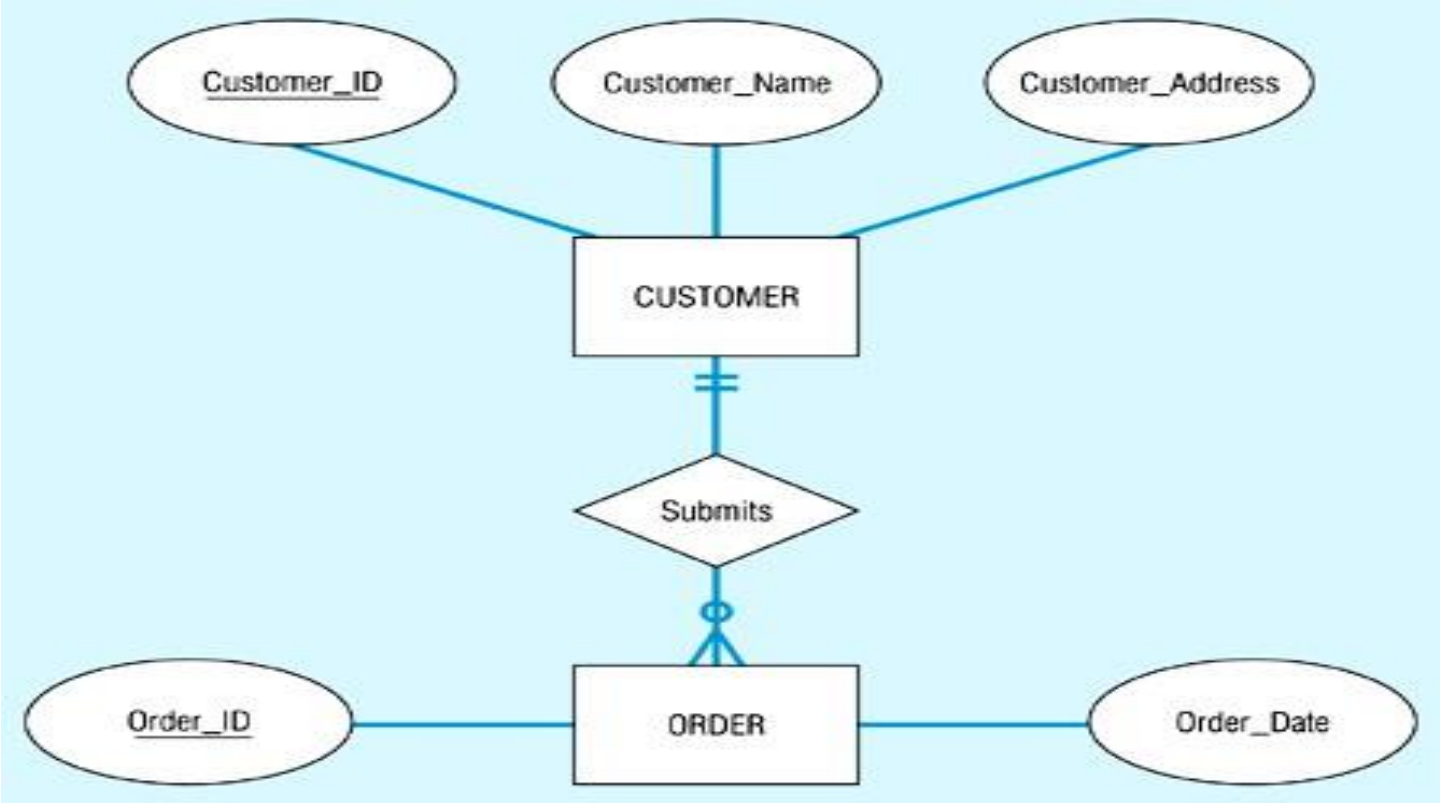


Transforming E-R diagrams into relations

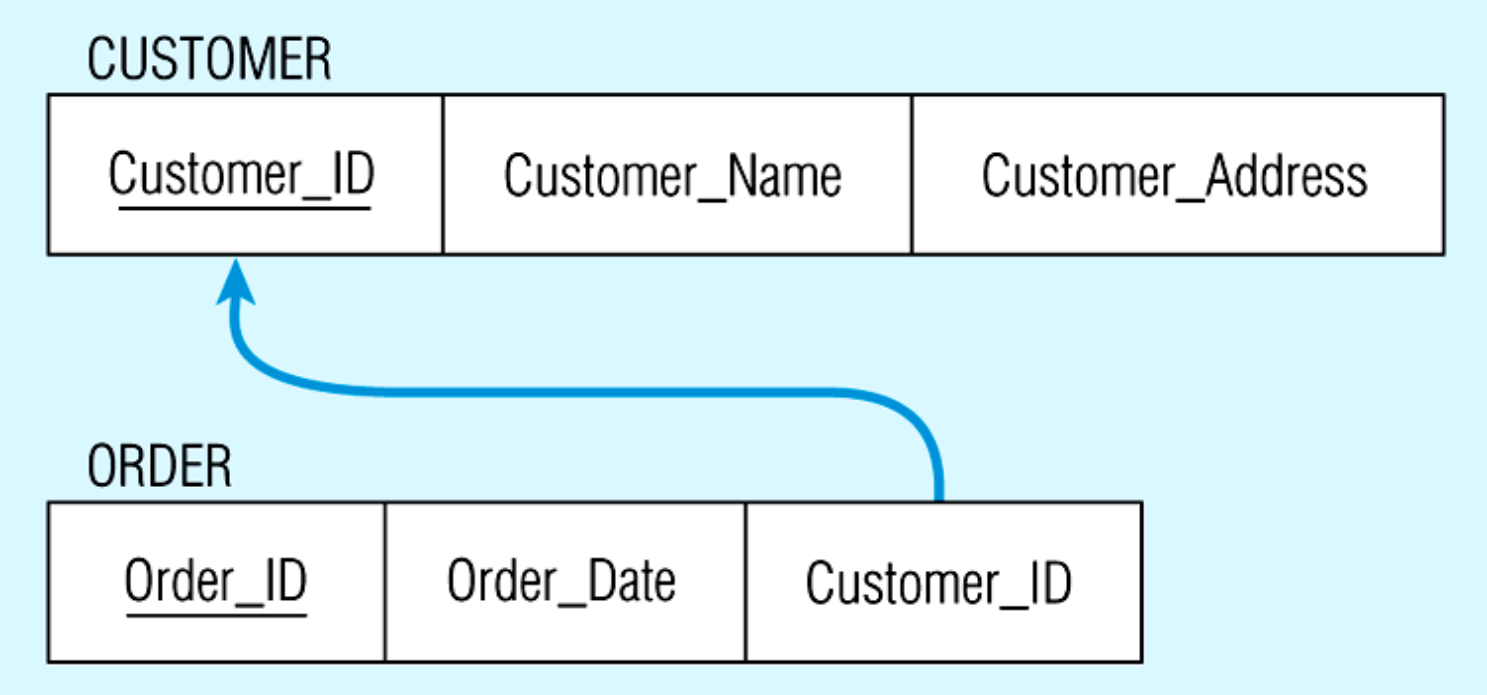
Mapping binary relationships

- One-to-many - primary key on the one side becomes a foreign key on the many side
- Many-to-many - create a new relation (associative entity) with the primary keys of the two entities as its primary key
- One-to-one - primary key on the mandatory side becomes a foreign key on the optional side

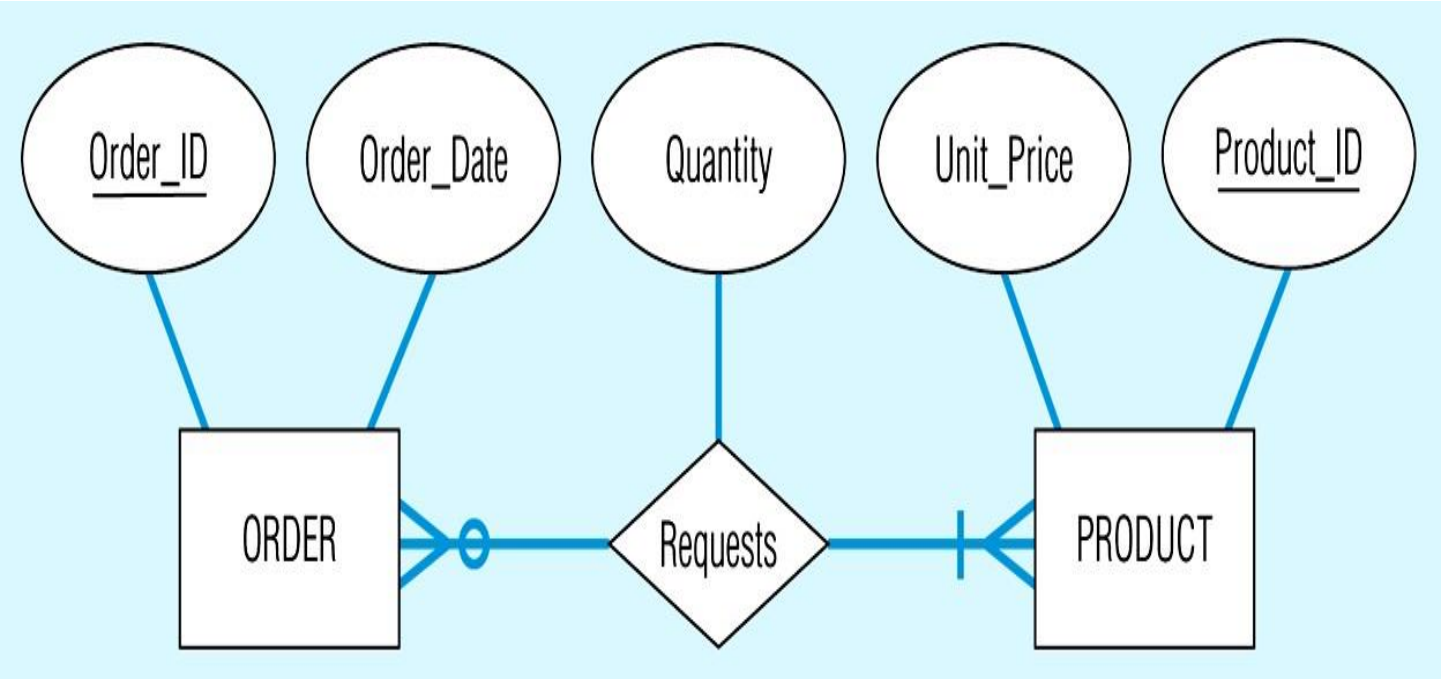
Example of mapping a 1:M relationship



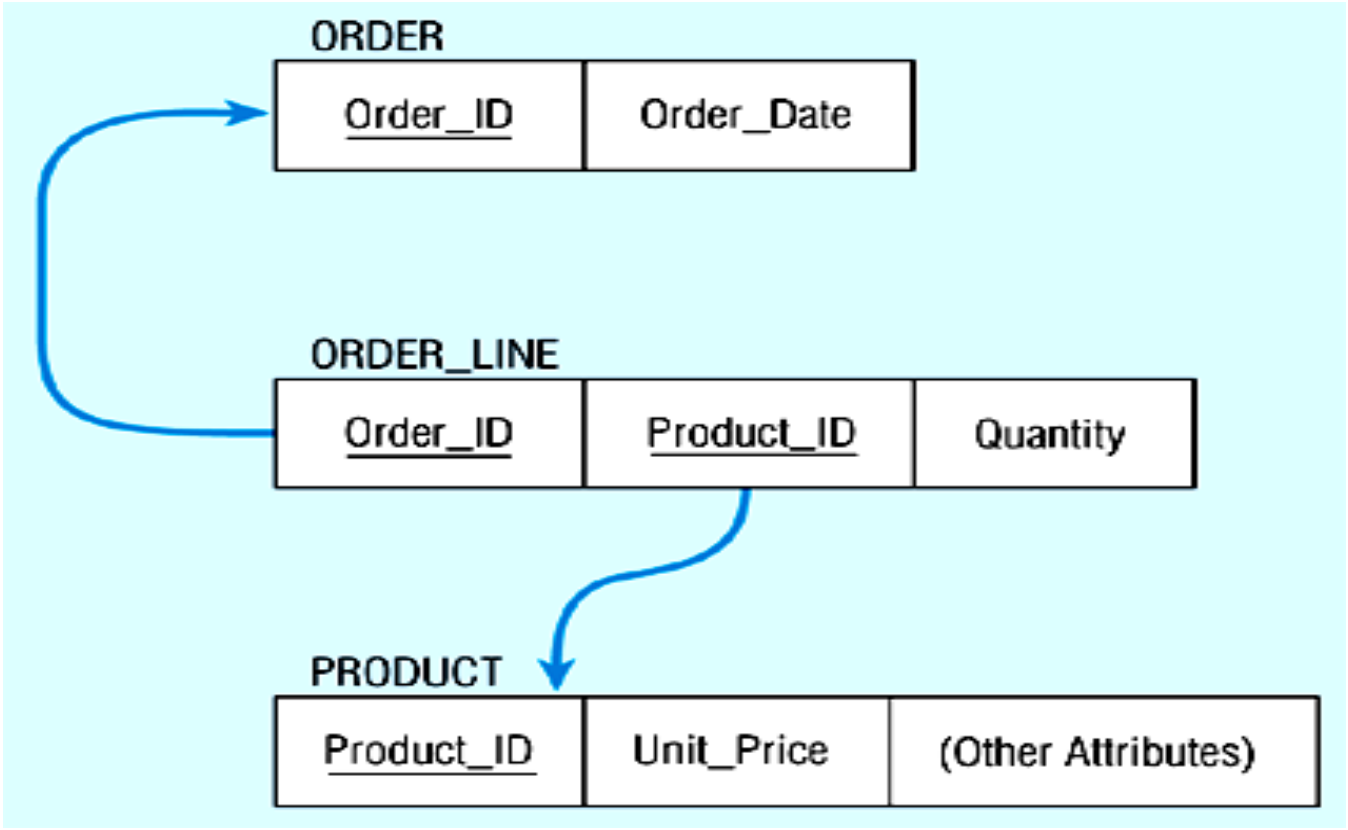
Looks like this using relational schema notation



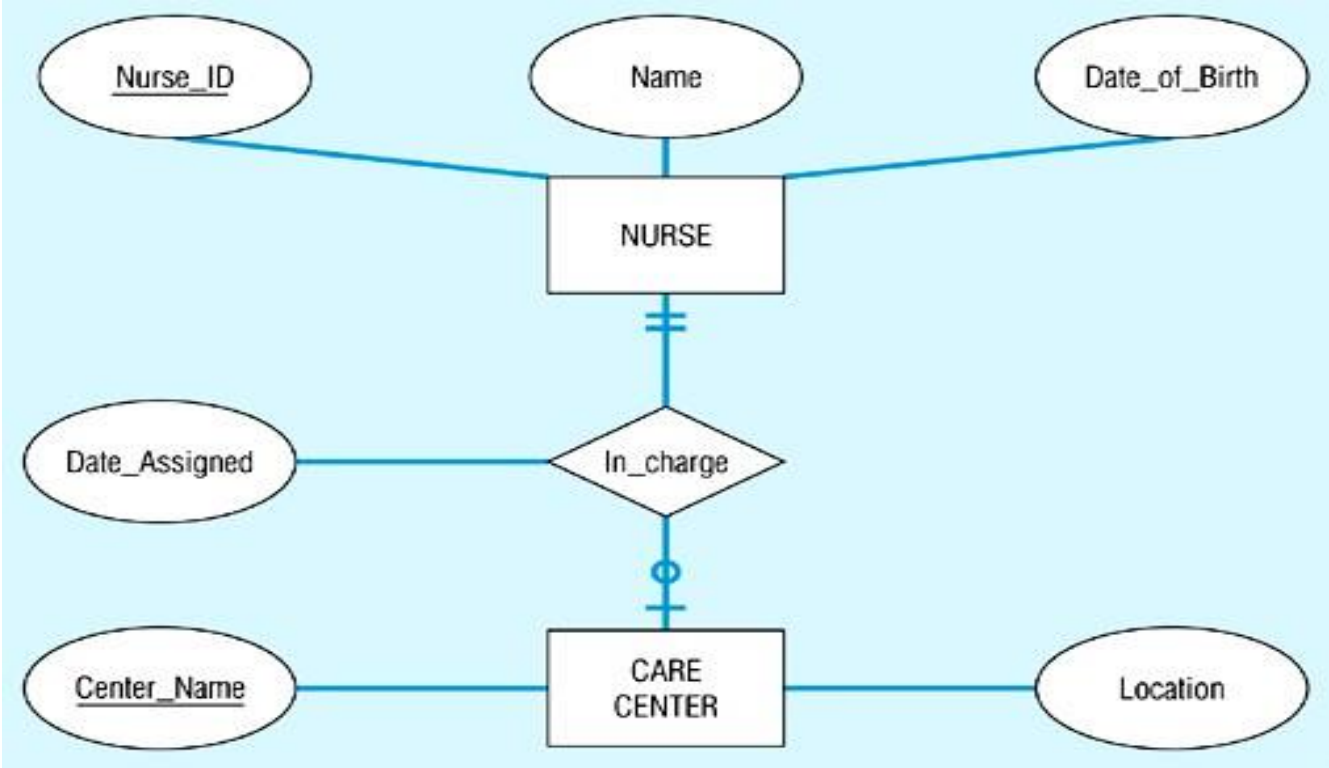
Example of mapping an M:M relationship



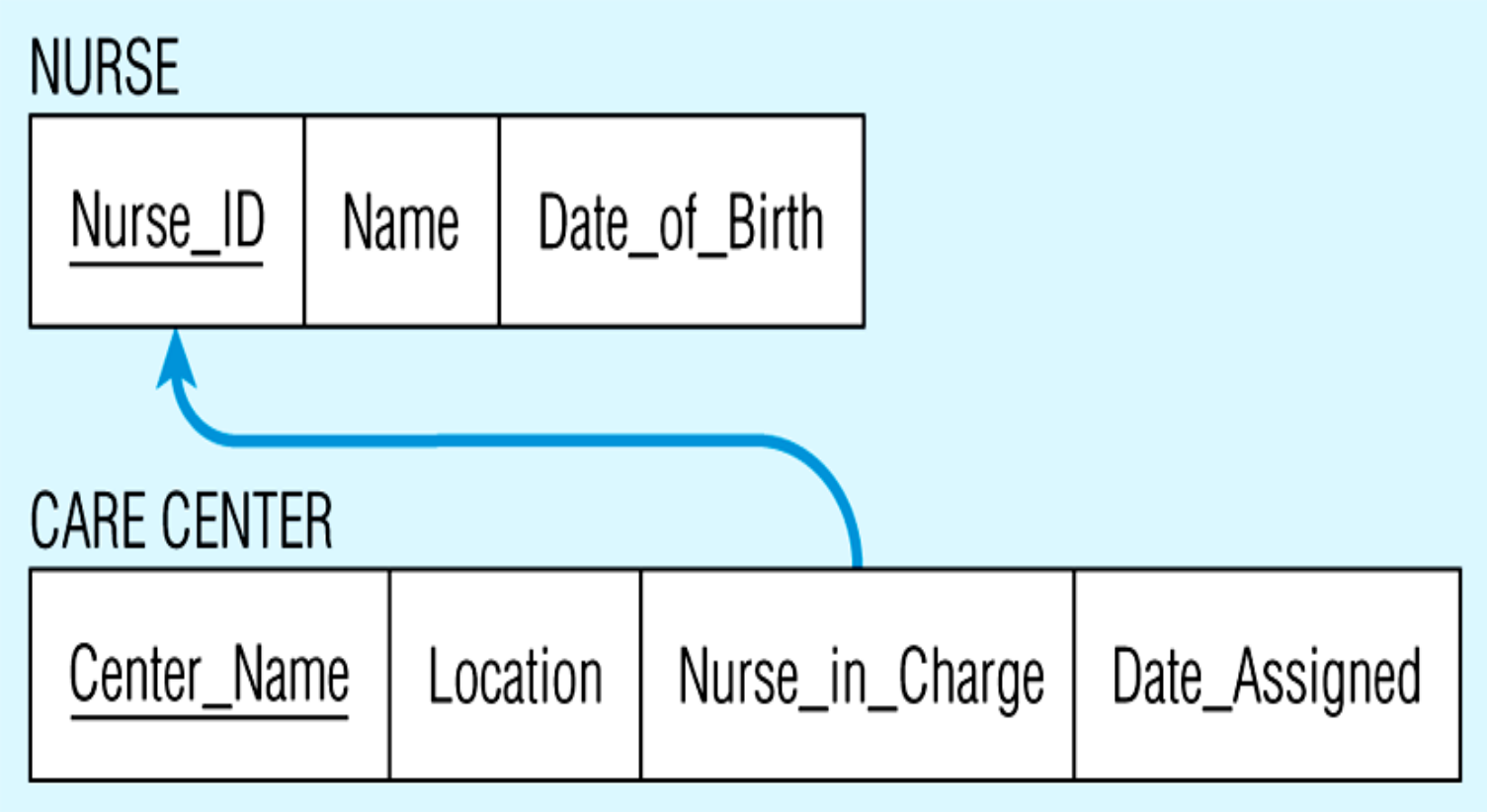
Looks like this using relational schema notation



Mapping a binary 1:1 relationship



Looks like this using relational schema notation

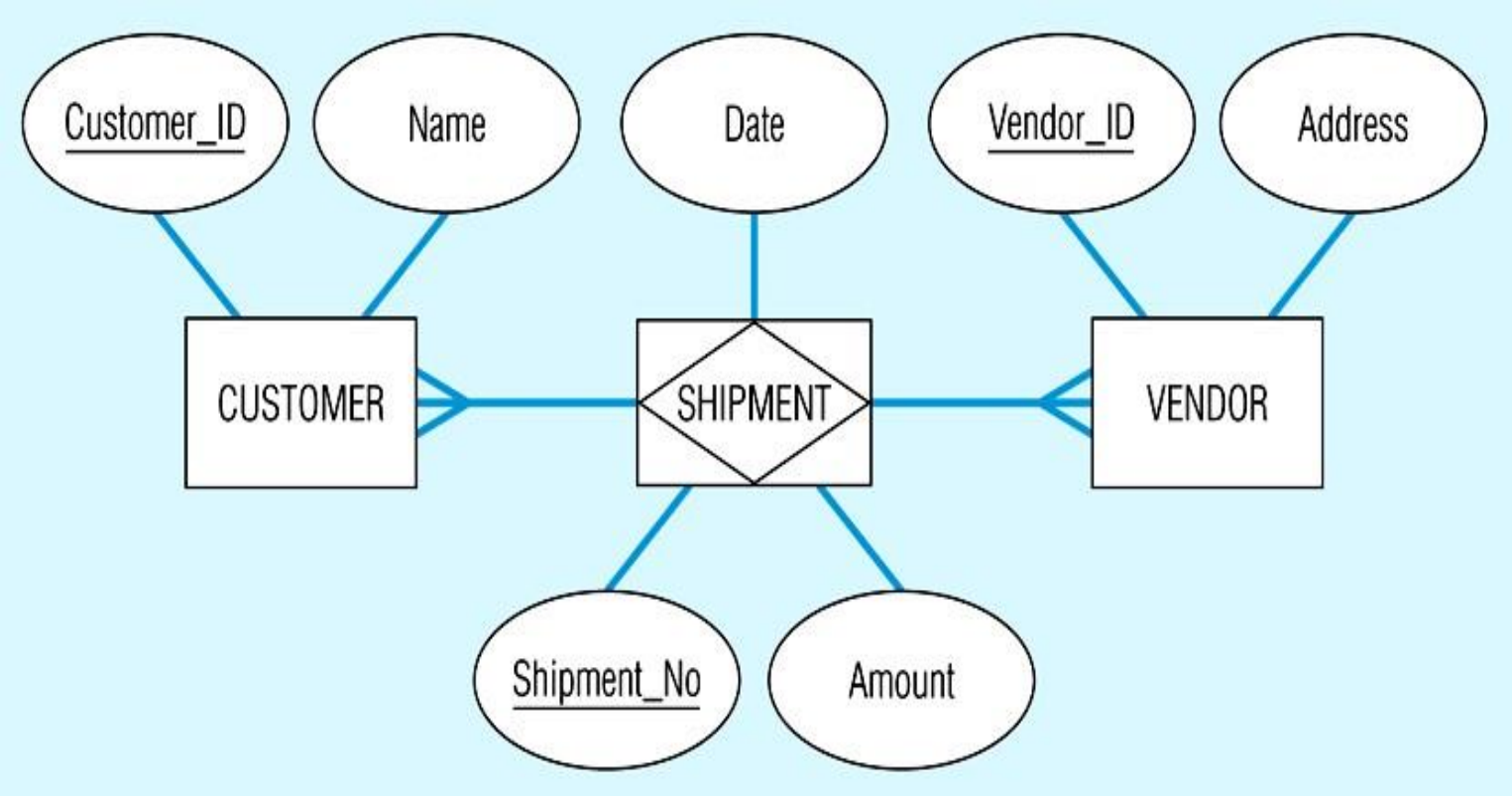


Transforming E-R diagrams into relations

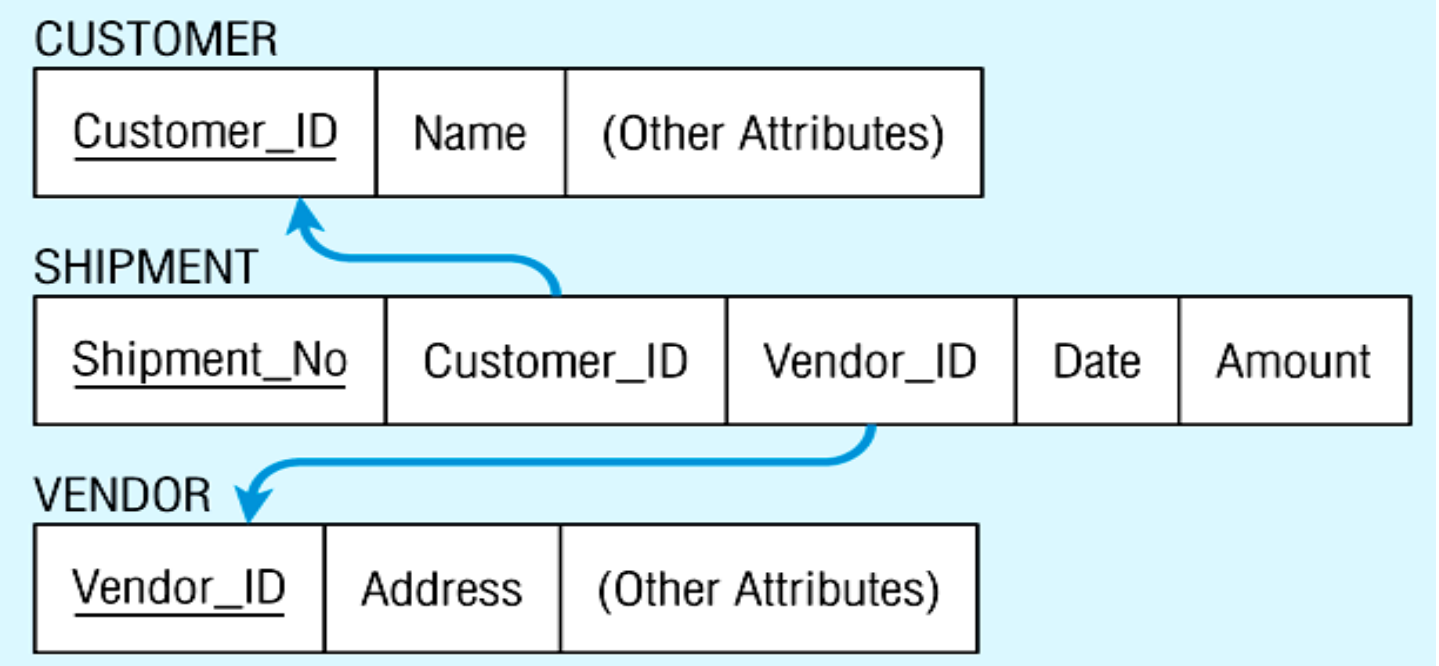
Mapping associative entities

- Identifier not assigned
  - Default primary key for the association relation is the primary keys of the two entities
- Identifier assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

Mapping an associative entity with an identifier



Looks like this using relational schema notation

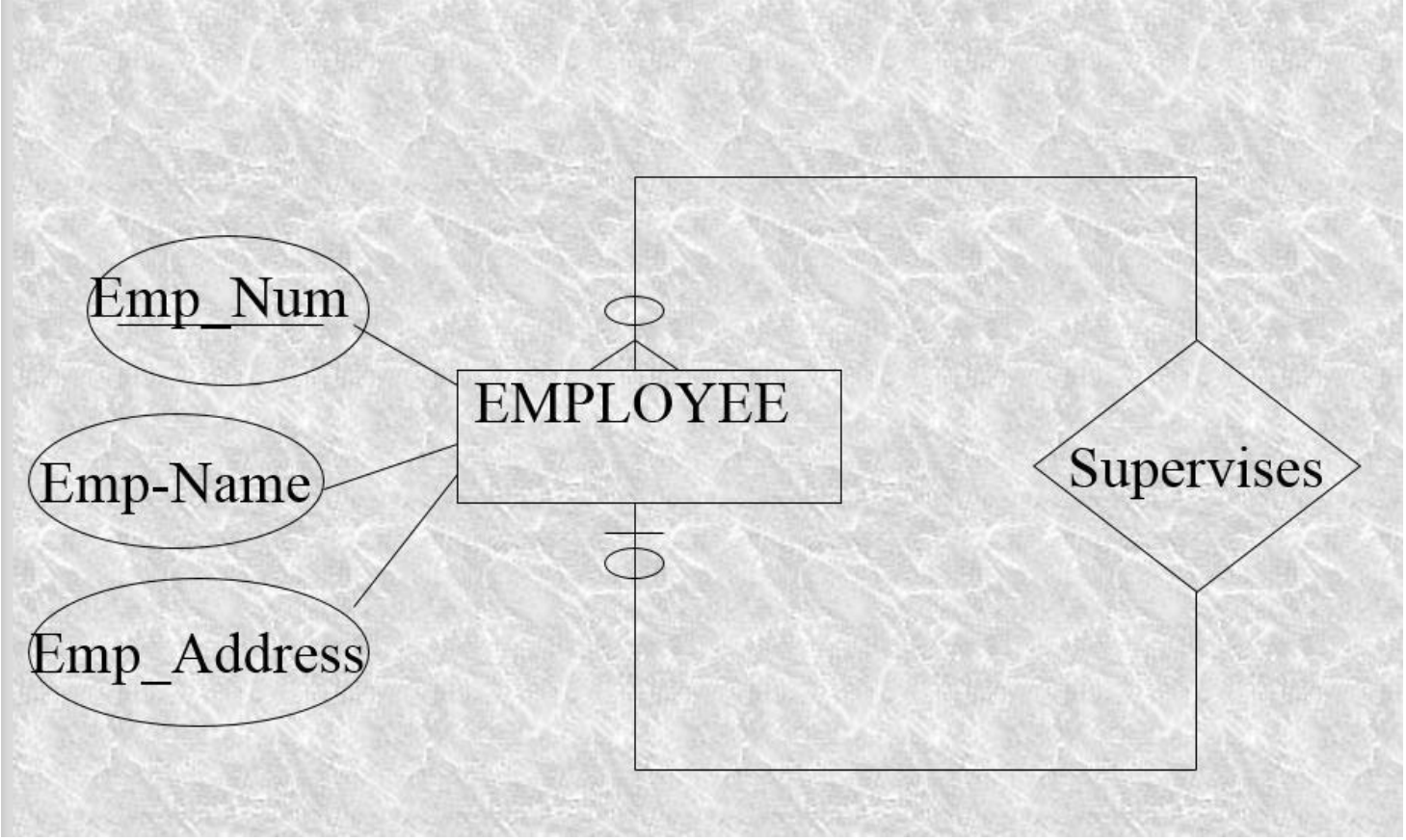


Transforming E-R diagrams into relations

Mapping unary relationships

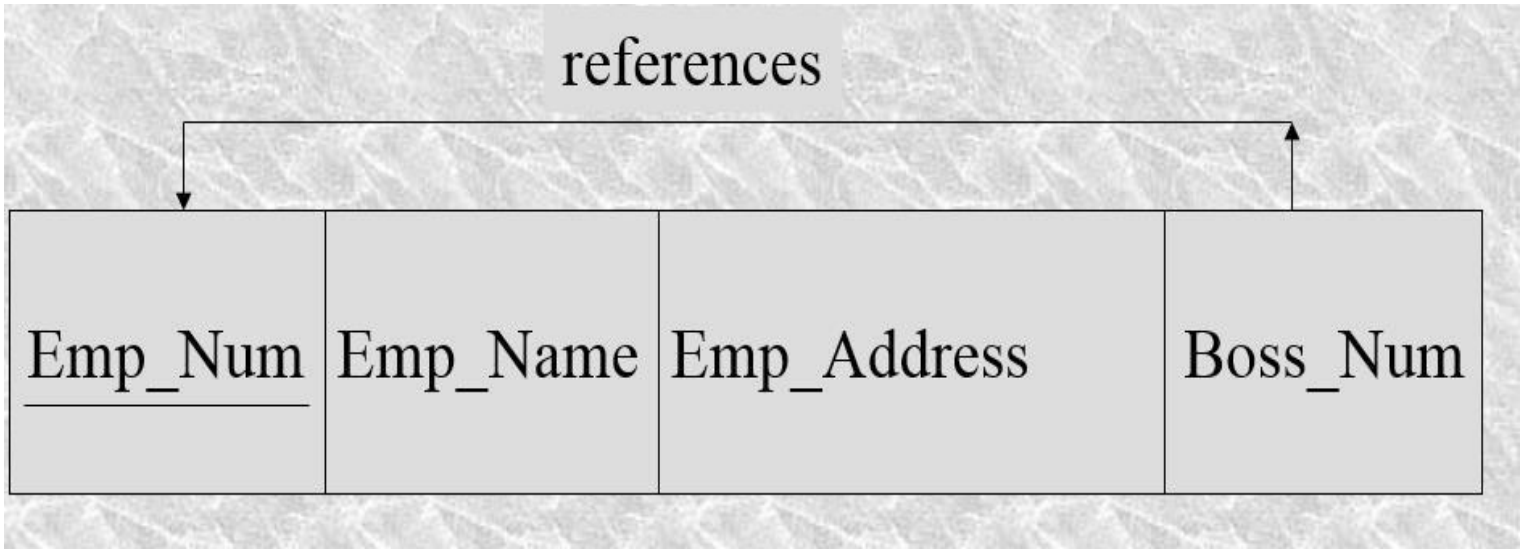
- One-to-many - recursive foreign key in the same relation
- Many-to-many - two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

For example...

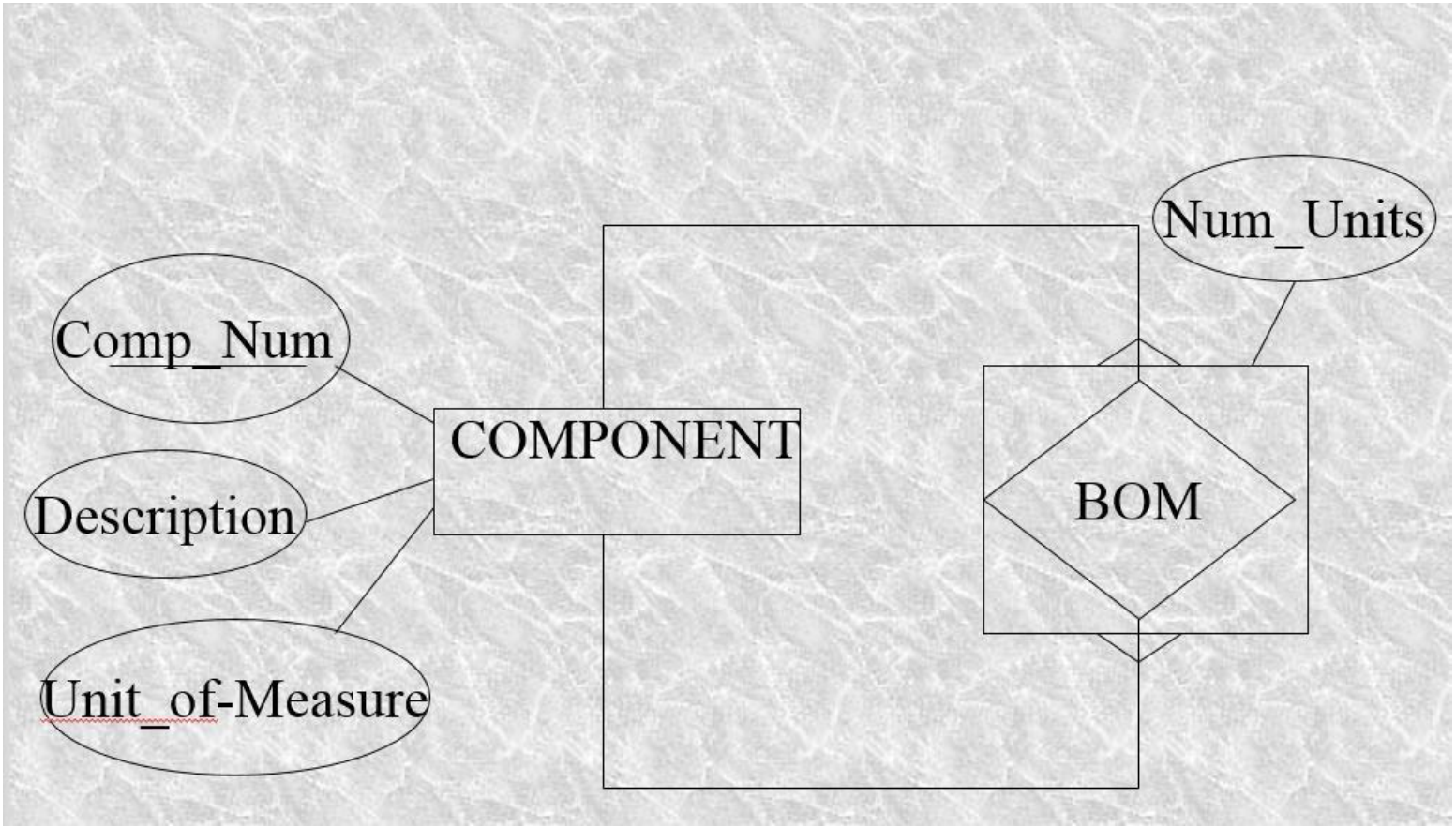


Would look like...

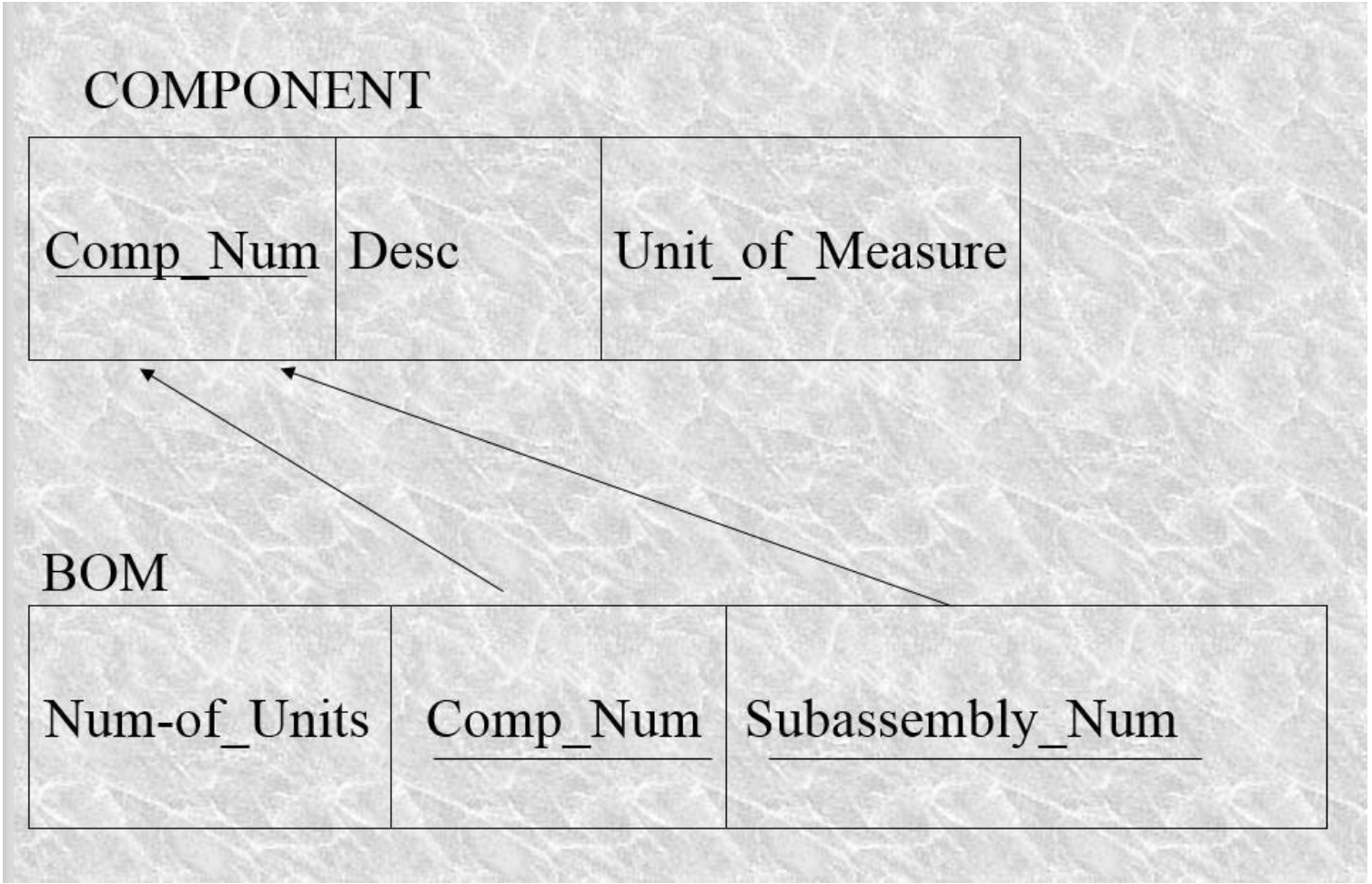
EMPLOYEE relation with recursive foreign key



And..


Num_Units
Comp_Num
COMPONENT
BOM
Description
Unit_of-Measure

Would look like…

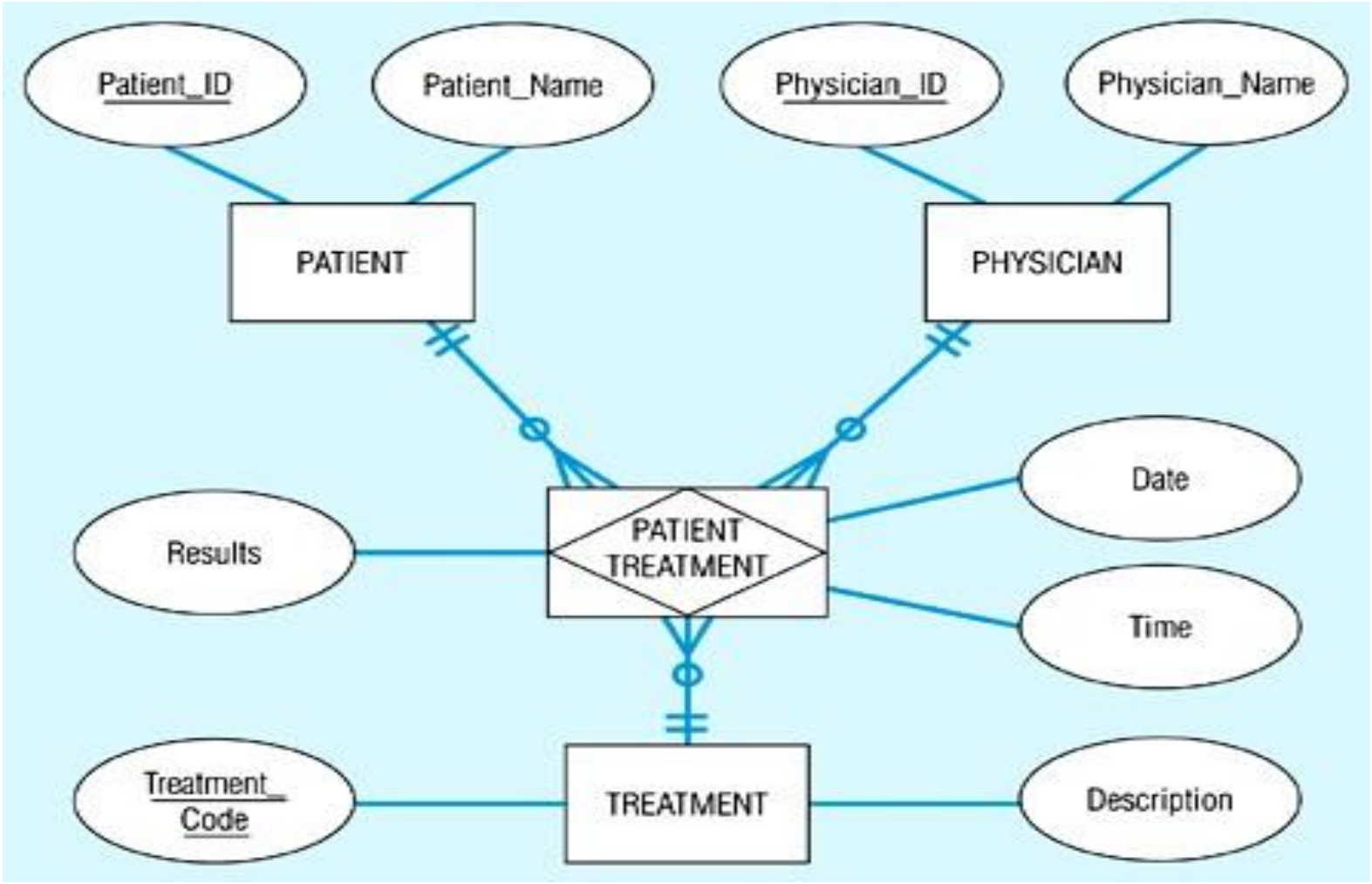


Transforming E-R diagrams into relations

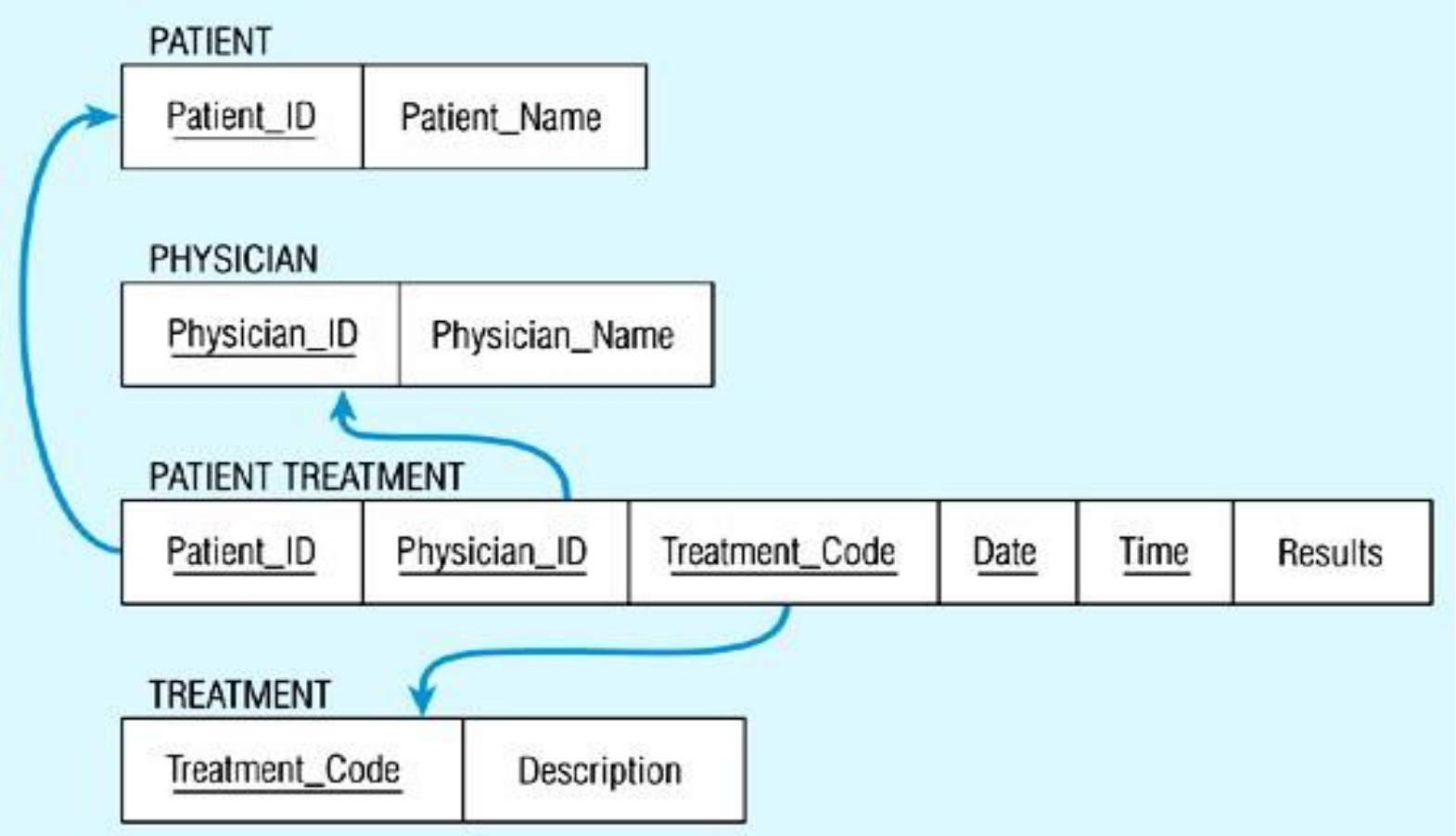
Mapping ternary (and n-ary) relationships

- One relation for each entity and one for the associative entity

Mapping a ternary relationship



Looks like this using relational schema notation

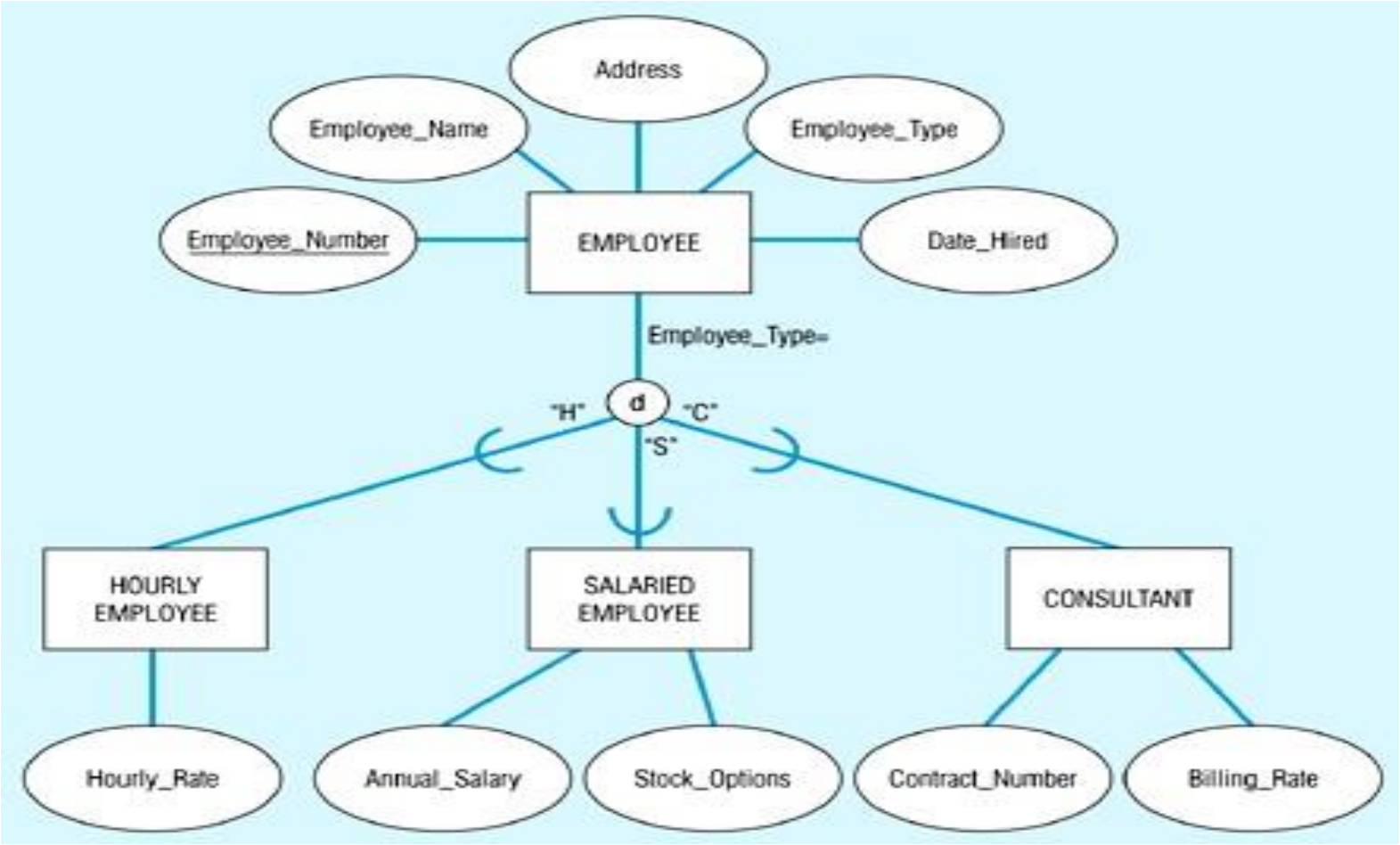


Transforming E-R diagrams into relations

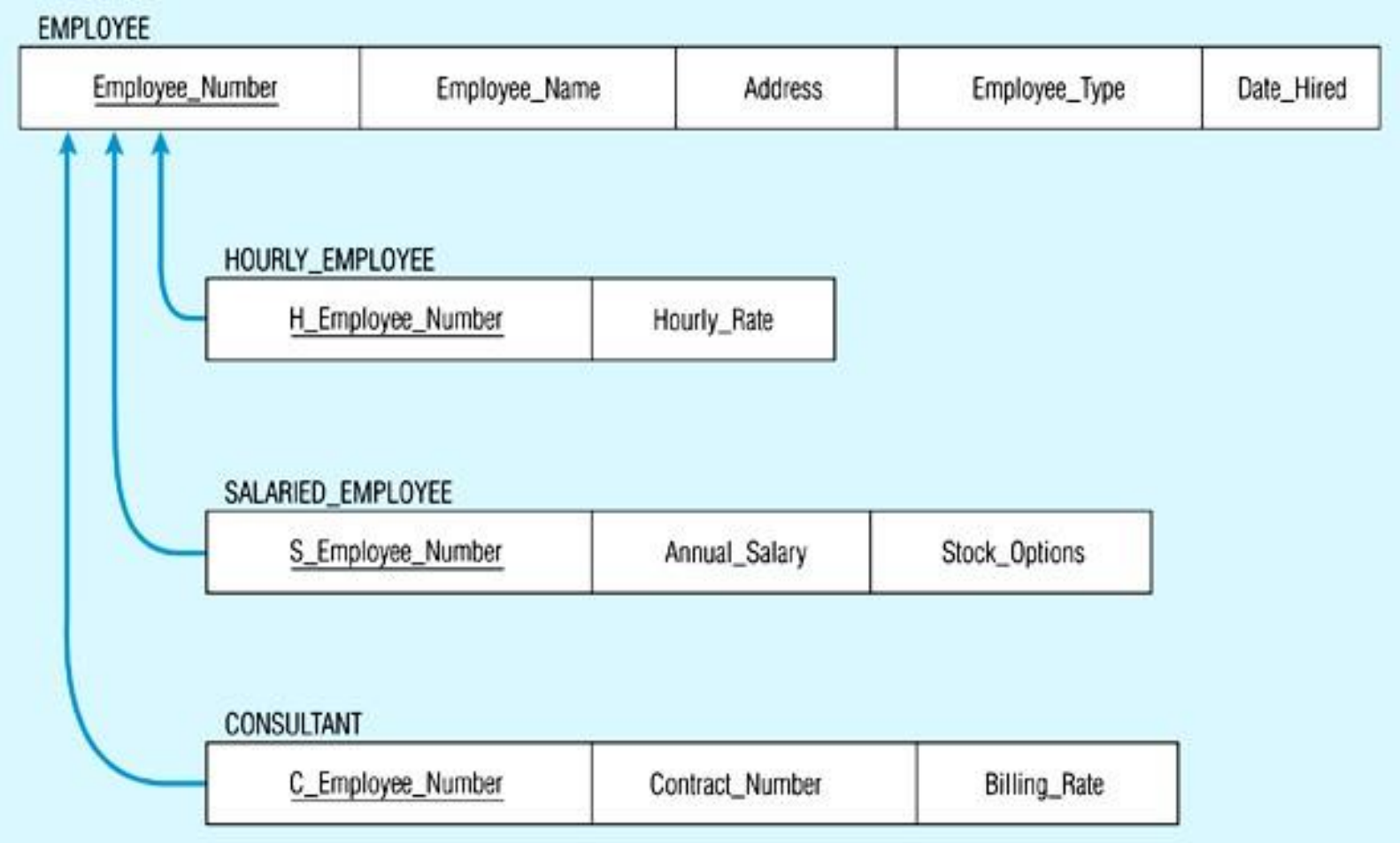
Mapping Supertype/subtype relationships

- Create a separate relation for the supertype and each of the subtypes
- Assign common attributes to supertype
- Assign primary key and unique attributes to each subtype
- Assign an attribute of the supertype to act as subtype discriminator

Mapping Supertype/subtype relationships



Would look like this...



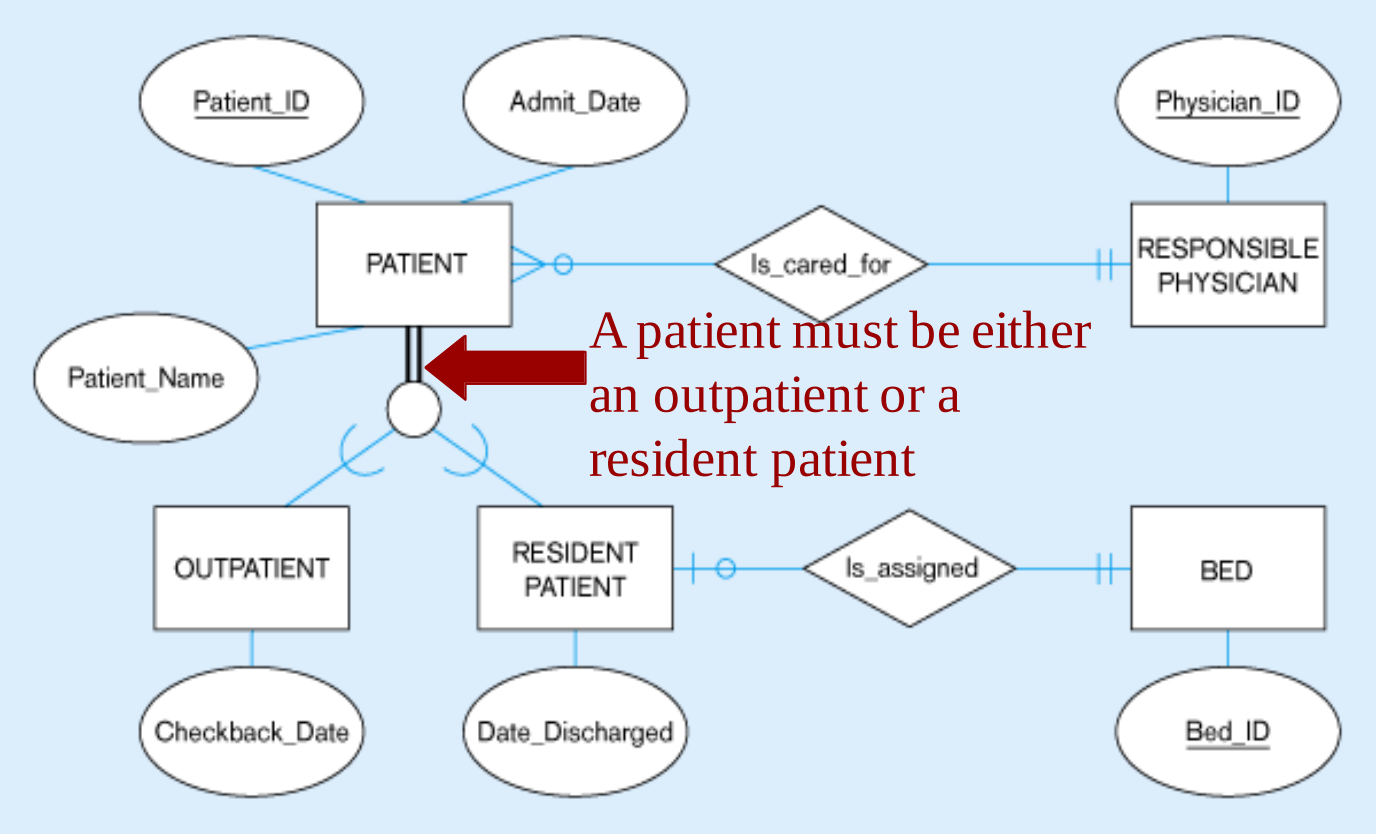
Constraints in Supertype/Completeness Constraint

Completeness Constraints: Whether an instance of a supertype ***must*** also be a member of at least one subtype

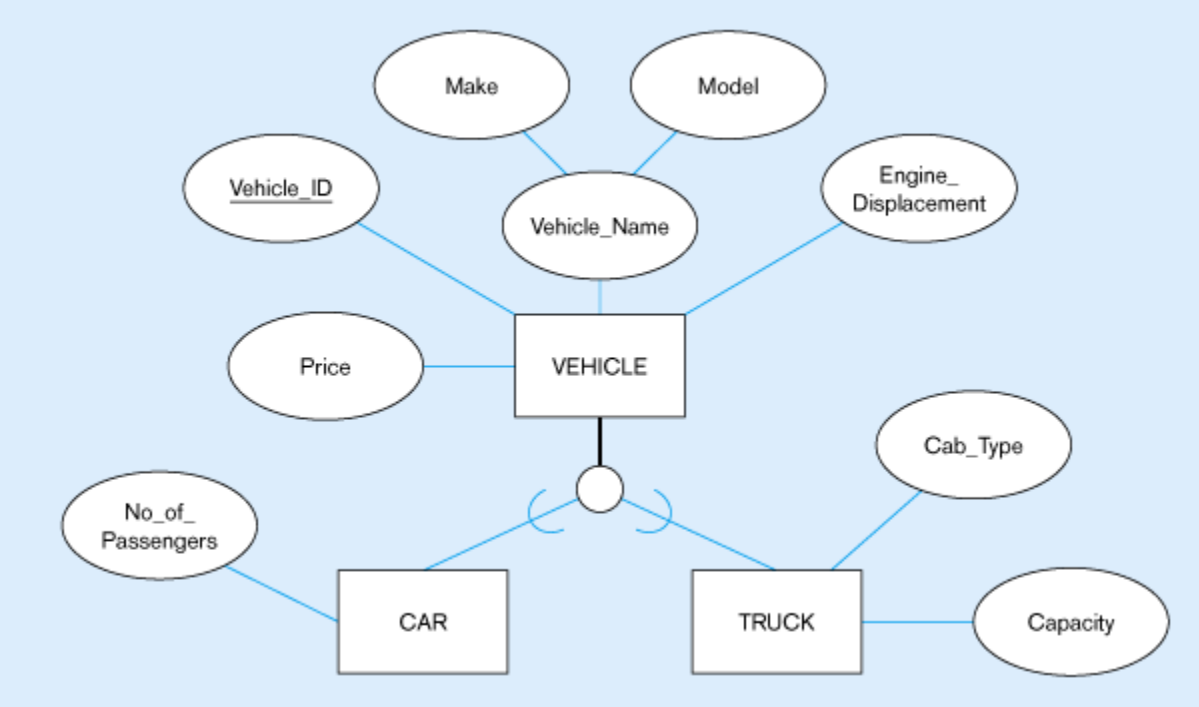
Total Specialization Rule: Yes (double line)
Partial Specialization Rule: No (single line)

Examples of completeness constraints

Total specialization rule



Partial specialization rule



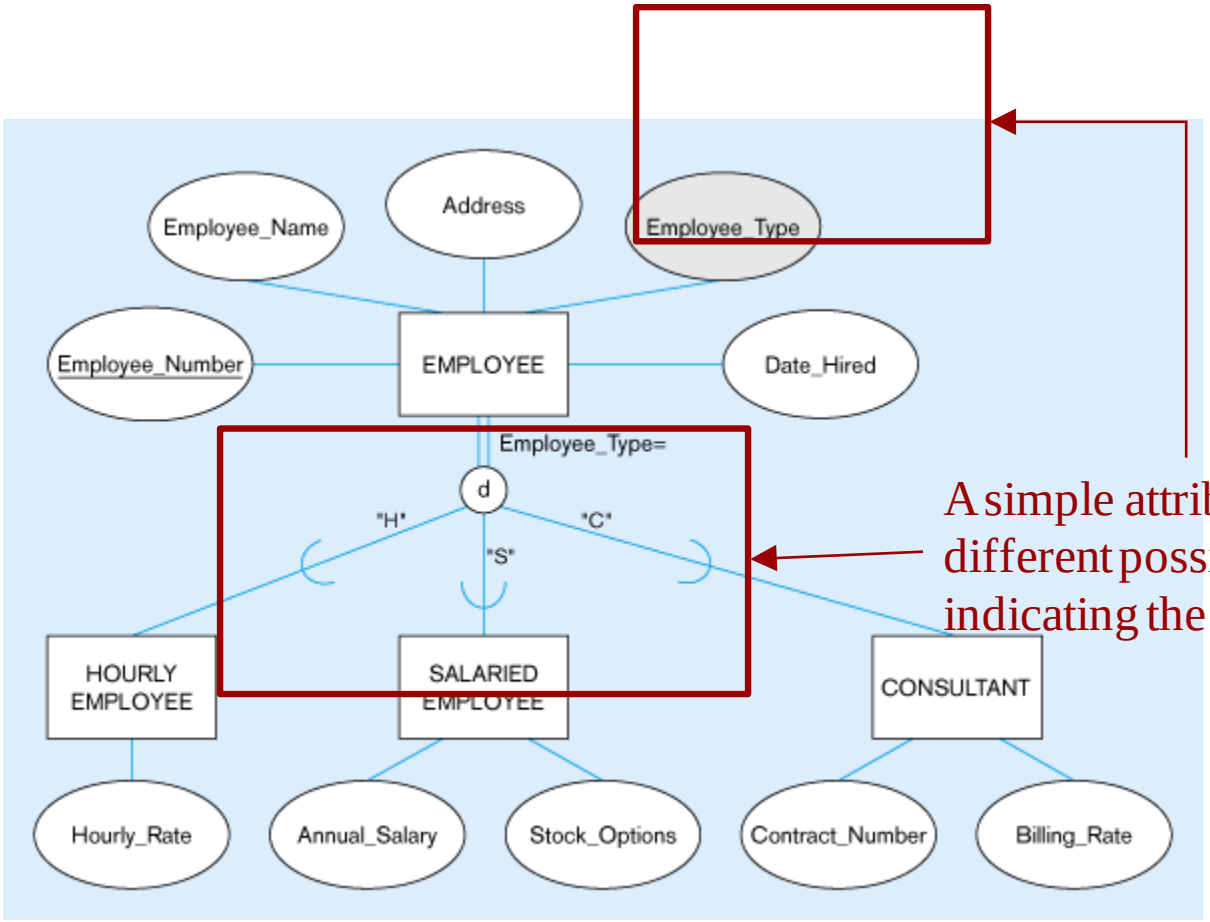
Constraints in Supertype/Disjointness constraint

Disjointness Constraints: Whether an instance of a supertype may *simultaneously* be a member of two (or more) subtypes

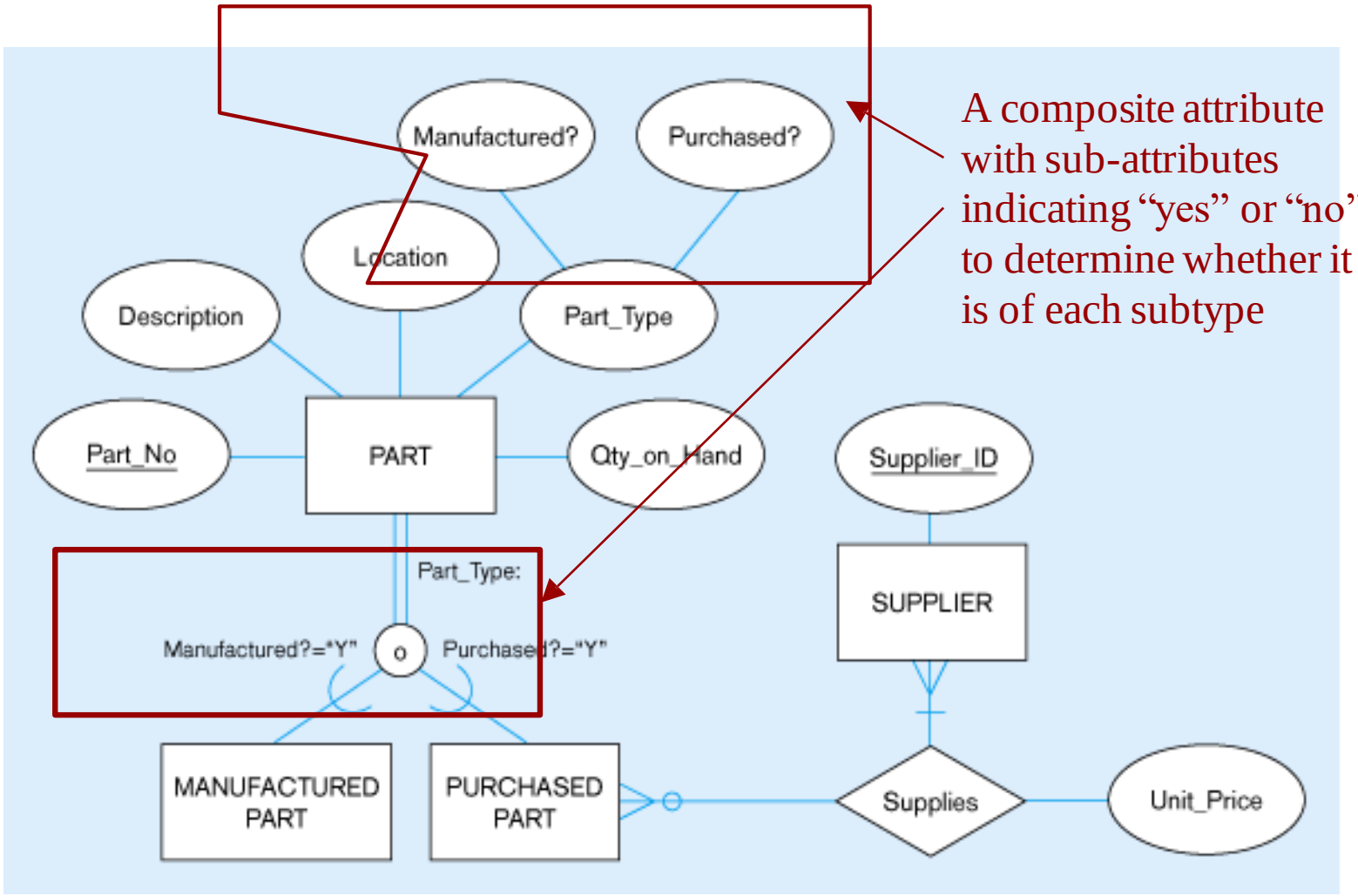
Disjoint Rule: An instance of the supertype can be only ONE of the subtypes

Overlap Rule: An instance of the supertype could be more than one of the subtypes

Introducing a subtype discriminator (*disjoint* rule)



Subtype discriminator (***overlap*** rule)



3.4 Relational Integrity

- Data integrity constraints refer to the accuracy and correctness of data in the database
- It provides a mechanism to maintain data consistency for operations like INSERT, UPDATE, and DELETE
- The different types of data integrity constraints are
  - Entity Integrity~ A primary key cannot accept an null value
  - Null Integrity~ Data value is not known temporarily
  - Domain Integrity Constraint~ Domain specifies its own name, data type, and logical size
  - Referential Integrity~ either each FK value must match a PK value in another relation or the FK value must be null

3.5 Relational Algebra

- Is a theoretical language with operations that work on one or more relations to define another relation without changing the original relation
- Is an abstract language, the queries formulated in relational algebra are not intended to be executed on a computer.
- Knowledge about relational algebra allows us to understand query execution and optimization in RDMS

3.5 Relational Algebra Operations

- Set operation and database operations

Relational Algebra Operation

- ✓Selection
- ✓ Projection
- ✓rename
- ✓Union
- ✓ Intersection
- ✓ Difference
- ✓ Cartesian product
- ✓ Join

3.5 Relational Algebra Operations

- Set operation and database operations

➢ **Set operations**

– Unary and Binary operations

- Unary involves one operand, whereas binary operation involves two operands
- Selection and projection are unary operations
- Union, difference, Cartesian product, and Join operations are binary operations

➢ Database operations

– Selection Operation

- Works on a single relation R and defines a relation that contains only those tuples of R that satisfied condition (predicate)
- Can be considered as *row wise filtering*
- Syntax: $\sigma_{Predicate}(R)$
  - R-Relation, Predicate-Condition

3.5 Relational Algebra Operations

- Example , consider the STUDENT shown below

STUDENT

Student Roll. No	Name	GPA
001	Aravind	7.2
002	Anand	7.5
003	Balu	8.2
004	Chitra	8.0
005	Deepa	8.5
006	Govind	7.2
007	Hari	6.5

- List the Roll.no, Name, and GPA of those students who are having GPA of above 8.0
- Query expressed in relational algebra as $\sigma_{GPA} \succ 8\,(Student)$

Student Roll. No	Name	GPA
003	Balu	8.2
005	Deepa	8.5

3.5 Relational Algebra Operations

- Projection Operation
  - It works on a single relation R and defines a relation that contains a vertical subject of R, extracting the values of specified attributesand elimination duplicates.
  - It's considered as a column wise filtering

  - Syntax: $\prod_{a1,a2,....an}(R)$
    - Where a1, a2, …..an are attributes and R stands forrelation
  - Consider the relation STAFF, with the attributes given below

STAFF

Staff No	Name	Gender	Date of birth	Salary
SL21	Raghavan	M	1-5-76	15,000
SL22	Raghu	M	1-5-77	12,000
SL55	Babu	M	1-6-76	12,500
SL66	Kingsly	M	1-8-78	10,000

  - Produce the list of salaries for all showing only the name and salary detail $\prod_{Name,salary}(Staff)$

3.5 Relational Algebra Operations

- Output for the query

Name	Salary
Raghavan	15,000
Raghu	12,000
Babu	12,500
Kingsly	10,000

- Rename operation (ρ)
  - Returns an existing relation under a new name
  - $\rho_{A/B}(R)$ is the relation R with its attribute name changed from B to A
  - Consider the relation BATSMAN with the attributes name, nation and BA

BATSMAN

Name	Nation	BA
Sachin Tendulkar	India	45.5
Brian Lara	West Indies	43.5
Inzamamulhaq	Pakistan	42.5

  - After renaming BATSMAN relation's attribute ρ (R)

BATSMAN

Name	Nation	Batting average
Sachin Tendulkar	India	45.5
Brian Lara	West Indies	43.5
Inzamamulhaq	Pakistan	42.5

Batting average/BA

3.5 Relational Algebra Operations

- Union Operation
  - The union of two relations R and S defines a relation that contains all the tuples of R or S or both R and S, duplicate tuples being eliminated
  - Its relational algebra expression $R \cup S$
  - Consider those relations

Customer 1

Name	City
Anand	Coimbatore
Aravind	Chennai
Gopu	Tirunelveli
Helan	Palayankottai

Customer 2

Name	City
Gopu	Tirunelveli
Balu	Kumbakonam
Rahu	Chidambaram
Helan	Palayamkottai

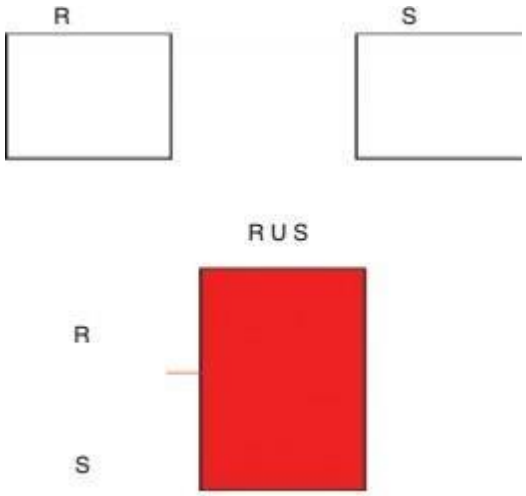


Fig. 3.4. Union of two relations R and S

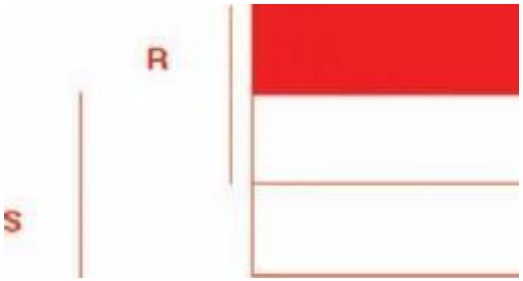
- Result of Customer 1 U customer 2

Customer 1 ∪ Customer 2

Name	City
Anand	Coimbatore
Aravind	Chennai
Balu	Kumbakonam
Gopu	Tirunelveli
Rahu	Chidambaram
Helan	Palayamkottai

3.5 Relational Algebra Operations

- Difference Operation
  - Defines a relation consisting of the set of all tuples that are in relation R but not in S
  - The difference between two relations R and S is denoted by $R - S$
  - Its illustration is shown below



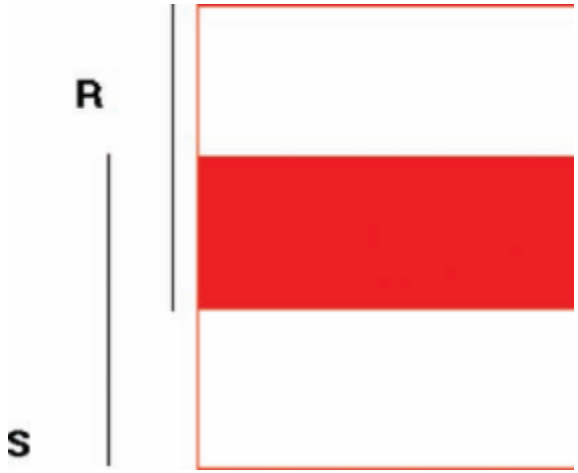
  - From the shown relations

Customer 1 – Customer 2

Name	City
Anand	Coimbatore
Aravind	Chennai

3.5 Relational Algebra Operations

- Intersection Operation
  - Defines a relation consisting of the set of all tuples that are in both R and S.
  - The intersection of two relations R and S is denoted by $R \cap S$
  - Its illustration is shown below



- From the shown relations

Customer 1 ∩ Customer 2

Name	City
Gopu	Tirunelveli
Helan	Palayamkottai

3.5 Relational Algebra Operations

- Division Operation
  - It extracts records and fields from one table on the basis of data in the second table
  - The division of the relation R by the relation S is denoted by $R \div S$, where $R \div S$ is given by:

$$R \div S = \Pi_{R--S(r)} - \Pi_{R--S((\Pi_{R--S(r)} X\ S)-r)}$$

  - Example for illustration is shown below

Student

Name	Mark
Arul	97
Banu	100
Christi	98
Dinesh	100
Krishna	95
Ravi	95
Lakshmi	98

Mark

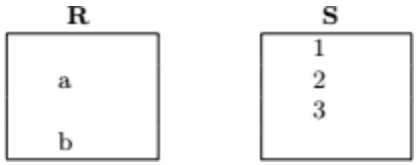
Mark
100

Answer

Name
Banu
Dinesh

3.5 Relational Algebra Operations

- Cartesian Product Operation
  - It defines a relation that is the concatenation of every tuples of relation R with every tuples of relation S.
  - The result of Cartesian product contains all attributes from both relations R and S.
  - Cartesian product between the two relations R and S is denoted by $R \times S$
  - Example for illustration is shown below



  - Determine $R \times S$

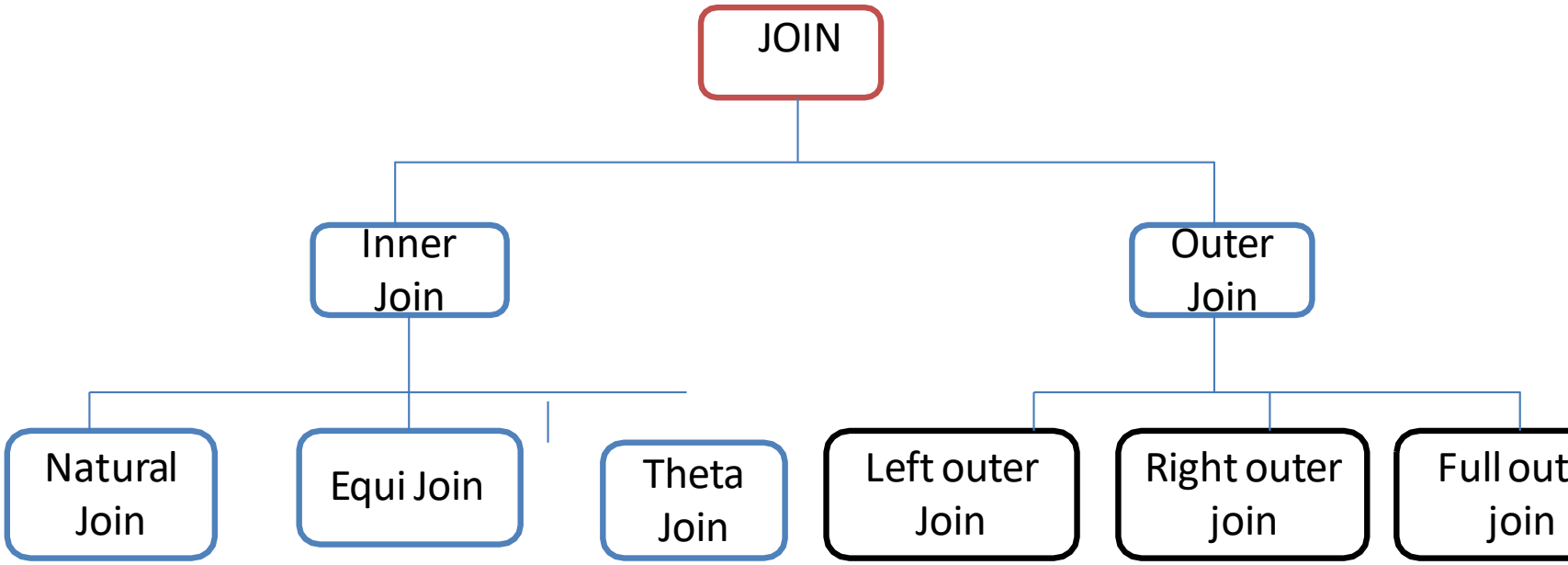
R	S
a	1
a	2
a	3
b	1
b	2
b	3

Note:

No. of tuples in R × S = 2 * 3 = 6
No. of attributes in R × S = 2

3.5 Relational Algebra Operations

- Join Operations
  - Join operation combines two relations to form a new relation.
  - The tables should be joined based on a common column and condition.
  - The common column should be compatible in terms of domain.



3.5 Relational Algebra Operations

a. Natural Join

- ✓ The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, have the **same name** and **domain** in both relations.
- ✓ If this is not the case, a renaming operation is applied first.
- ✓ Acts on those matching attributes where the values of attributes in both relation is same.

Employee

Employee ID	Designation	Dept Number
C100	Lecturer	E1
C101	Assistant Professor	E2
C102	Professor	C1

⋈

Department

Dept name	Dept Number
Electrical	E1
Computer	C1

Employee ⋈ Department

Employee ID	Designation	Dept Number	Dept name
C100	Lecturer	E1	Electrical
C102	Professor	C1	Computer

3.5 Relational Algebra Operations

b. Equi Join

- ✓ The most common use of join involves join conditions with equality comparisons only.
- ✓ Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
- ✓ In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have *identical values* in every tuple.
- ✓ The JOIN seen in the previous example was EQUIJOIN.

- ✓ Example

STAFF

Staff No	Job	Dept
1	salesman	100
2	draftsman	101

DEPT

Dept	Name
100	marketing
101	civil

- ✓ Answer for the earlier query is equi-join of STAFF and DEPAT

STAFF EQUI JOIN DEPARTMENT

Staff No	Job	dept	dept	Name
1	salesman	100	100	marketing
2	draftsman	101	101	civil

3.5 Relational Algebra Operations

c. Theta Join

- A conditional join in which we impose condition other than equality condition
- If equality condition is imposed, then theta join become equi join.
- The symbol stands for the comparison operators: <,>,>=,<=.
- Its expression $\sigma_{\theta}(R \times S)$ *or* R1 ⋈θ R2
- Its illustration

Student

SID	Name	Std
101	Alex	10
102	Maria	11

[*Table: Student Relation*]

Subjects

Class	Subject
10	Math
10	English
11	Music
11	Sports

[*Table: Subjects Relation*]

Student_Detail = STUDENT ⋈Student.Std = Subject.Class SUBJECT

Student_detail

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

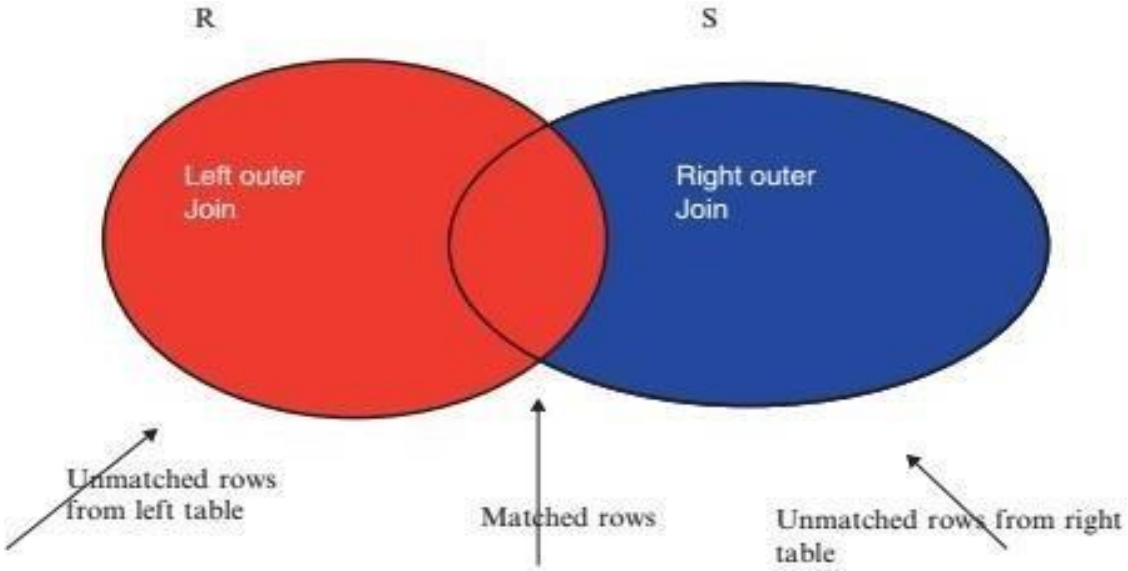
[*Table: Output of theta join*]

3.5 Relational Algebra Operations

d. Outer Join

- ✓ Matched pairs are retained unmatched values in other tables are left null

1. Full outer Join 
2. Left outer Join 
3. Right outer Join 



3.5 Relational Algebra Operations

Outer Join

PEOPLE ⟕ PEOPLE. Food = MENU. Food MENU

Name	Age	People.Food	Menu.Food	Day
Raja	21	Idly	Idly	Tuesday
Ravi	22	Dosa	Dosa	Wednesday
Rani	20	Pizza	NULL	NULL
Devi	21	Pongal	Pongal	Monday

Table 3.2. Right outer join of PEOPLE and MENU relation

PEOPLE ⟖ PEOPLE.Food = Menu.Food MENU

Name	Age	People.Food	Menu.Food	Day
Devi	21	Pongal	Pongal	Monday
Raja	21	Idly	Idly	Tuesday
Ravi	22	Dosa	Dosa	Wednesday
NULL	NULL	NULL	Fried rice	Thursday
NULL	NULL	NULL	Parotta	Friday

PEOPLE

Name	Age	Food
Raja	21	Idly
Ravi	22	Dosa
Rani	20	Pizza
Devi	21	Pongal

MENU

Food	Day
Pongal	Monday
Idly	Tuesday
Dosa	Wednesday
Fried rice	Thursday
Parotta	Friday

d. Left outer Join

All tuples of left relation R, are included in the resulting relation and if there exists tuples in R without any matching tuple in S then put NULL in S's attributes

e. Right outer Join

All tuples of the Right relation S, are included in the resulting relation and if there exists tuples in S without any matching in R, then put NULL in R's attributes

3.5 Relational Algebra Operations

f. Full outer Join

- ✓ All tuples of both participating relations are included in the resulting relation and if there is no matching tuples for both relations, their respective unmatched attributes are made NULL

Left	
A	**B**
100	Database
101	Mechanics
102	Electronics

[*Table: Left Relation*]

Right	
A	**B**
100	Alex
102	Maya
104	Mira

[*Table: Right Relation*]

Courses		HoD	
A	**B**	**C**	**D**
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya
---	---	104	Mira

[*Table: Full outer join output*]

3.6. Advantages of Relational Algebra

- ✓ Has a solid mathematical background
- ✓ The background is the basic of many interesting developments and theorems

3.7. Limitations of Relational Algebra

- ✓ It can't do arithmetic
- ✓ It can not sort or print results in various formats
- ✓ It can not modify the database
- ✓ It can not compute transitive closure

Consider the query, Find all direct and indirect relatives of Gopal? It is not possible to express such kind of query in relational algebra. Here transitive means, if the person A is related to the person B and if the person B is related to the person C means indirectly the person A is related to the person C. But relational algebra cannot express the transitive closure.

3.8. Relational calculus

- Relational calculus query specifies *what* is to be retrieved rather than *how* to retrieve it.
  - No description of how to evaluate a query.
- In first-order logic (or predicate calculus), *predicate* is a truth-valued function with arguments.
- When we substitute values for the arguments, function yields an expression, called a *proposition*, which can be either true or false.

3.8. Relational calculus

- If predicate contains a variable (e.g. '*x* is a member of staff'), there must be a range for *x*.
- When we substitute some values of this range for *x*, proposition may be true; for other values, it may be false.
- When applied to databases, relational calculus has forms: *tuple* and *domain*.

Tuple Relational Calculus

- Interested in finding tuples for which a predicate is true. Based on use of tuple variables.
- Tuple variable is a variable that 'ranges over' a named relation: i.e., variable whose only permitted values are tuples of the relation.
- Specify range of a tuple variable S as the Staff relation as:

  Staff(S)

- To find set of all tuples S such that P(S) is true:

  {S | P(S)}

Tuple Relational Calculus

To find details of all staff earning more than £10,000:

$$\{S \mid Staff(S) \wedge S.salary > 10000\}$$

To find a particular attribute, such as salary, write:

$$\{S.salary \mid Staff(S) \wedge S.salary > 10000\}$$

3.10. Domain Relational Calculus

Query has the form:

$$\{\langle x1, x2, \ldots, xn \rangle \mid p(\langle x1, x2, \ldots, xn \rangle)\}$$

- Answer includes all tuples $\langle x1, x2, \ldots, xn \rangle$ that make the formula $p(\langle x1, x2, \ldots, xn \rangle)$ be true.

Example: Find all sailors with a rating above 7

$$\{\langle I, N, T, A \rangle \mid (I, N, T, A) \in Sailors \wedge T > 7\}$$

Giving each attribute a variable name

Ensures that I, N, T, and A are restricted to be fields of the same tuple

3.11. Relational algebra vs calculus

Relational algebra is procedural language that can be used to tell the DBMS how to build a new relation from one or more relation in database

Relational calculus is non-procedural language that can be used to formulate the definition of relation in terms of one or more database relation

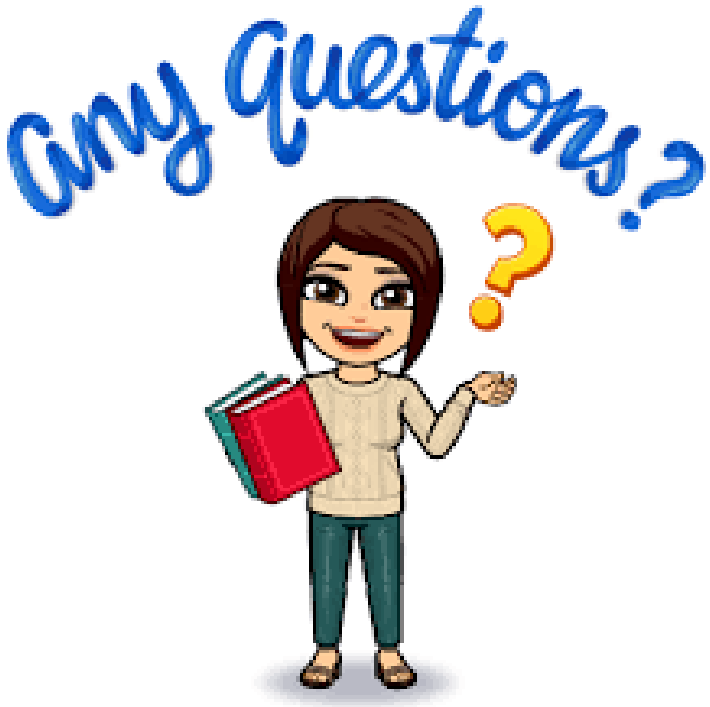
Relational algebra: User has to specify what is required and what are the procedures or steps to obtain output

Relational calculus: user just specifies what is required and need not to specify how to obtain it.

3.12.List of ERD Modeling Language tools

Name	Creator	Platform / OS	First public release	Latest stable release	Open source	Software license	Programming language used
ArgoUML	Tigris.org	Cross-platform (Java)	1998-04	2011-12-15[1]	Yes	EPL	Java
Astah	Change Vision, Inc.	Windows, OS X, Linux	2009-10-19	2015-06-25	No	Commercial, Free trial, Free edition (Community version)	Java, C++, C#
ATL	Obeo, INRIA Free software community	Cross-platform (Java)	Unknown	2010-06-23	Yes	EPL	Java
Borland Together	Borland	Cross-platform (Java)	Unknown	2008	No	Commercial	Unknown
BOUML	Bruno Pagès	Cross-platform	2005-02-26	2016-06-04	No	Commercial starting from v5.0,[2] GPL before v5.0	C++/Qt and Java ("plug-out")
CaseComplete	Serlio Software	Windows	2004	2013-04	No	Commercial	C#
ConceptDraw PRO	CS Odessa	Windows, OS X	1993	2010 (v9)	No	Commercial	Unknown
Dia	Alexander Larsson/GNOME Office	Cross-platform (GTK+)	2004?	2012-07-05	Yes	GPL	C
Eclipse UML2 Tools[3]	Eclipse Foundation	Cross-platform (Java)	2007	2009-06-19	Yes	EPL?	Java
Edraw Max	Edrawsoft	Windows, Linux, OS X	2004	2015-03	No	Commercial	C++
Enterprise Architect	Sparx Systems	Windows (Supports Linux and OS X installation)	2000	2015-06-18	No	Commercial	C++
Gliffy	Gliffy	Chrome, Safari, Firefox, Internet Explorer 9+	2006-08-01	2015-01 (v. 5.1)	No	Commercial, Free trial	HTML5 and Javascript
Lucidchart	Lucid Software	Windows, OS X, Linux, Solaris	2008-12	2014-10-07	No	Commercial / Free (educational)	HTML5 and Javascript
MagicDraw	No Magic	Cross-platform (Java)	1998	2014-11-17 (v18.1)	No	Commercial	Java
Microsoft Visio	Microsoft	Windows	1992	2016 (v16.0)	No	Commercial	Unknown


THANK
YOU


Any questions?