I.E.S Rodrigo Caro

Unit 3: Relational Models

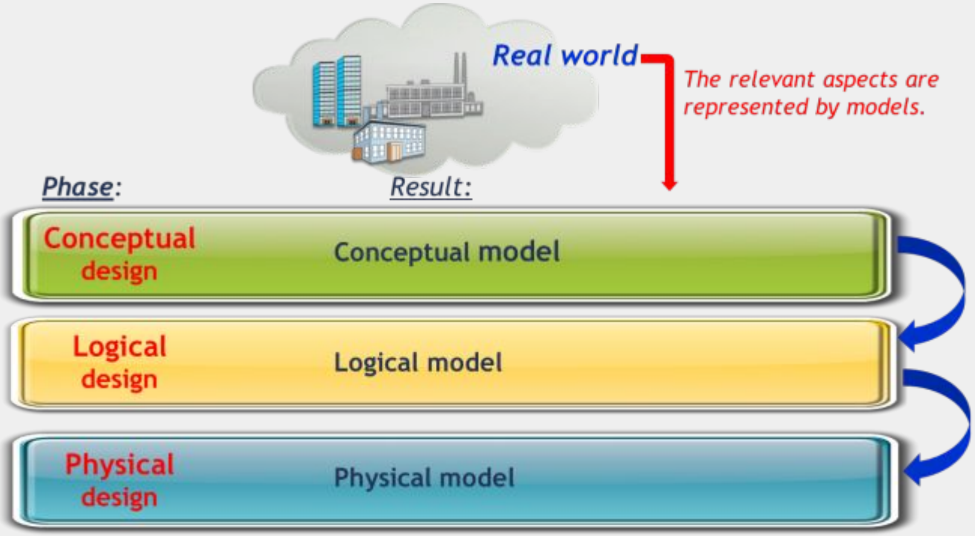
ASIR

Index

1. Relational Model: Definition
2. Elements
3. Constraints
4. Model Entities
5. Model Relationships: N:M
6. Model Relationships: 1:M
7. Model Relationships: 1:1
8. Reflexive relations
9.
10. Normalization: 1FN
11. Normalization: 2FN
12. Normalization: 3FN

Relational Model: Definition

Relational Model is a **logic model** to design databases.
Currently, the relational model is the most widespread model as it has been replacing other models such as the network or hierarchical model.
Remember, we are working at logic level. The physical storage of the data is independent of the way the data are logically organized..



Relational Model: Definition

The main reasons for its success are:

- Information is represented and manipulated in a simple way. Basically, it consists of **two dimensions interrelated tables**: consisting of **rows** (records or **tuples**) and **columns** (**attributes** or fields).
- They are based on **relational algebra**, which is a mathematical model with solid foundations.

  Basic principles of the relational model:

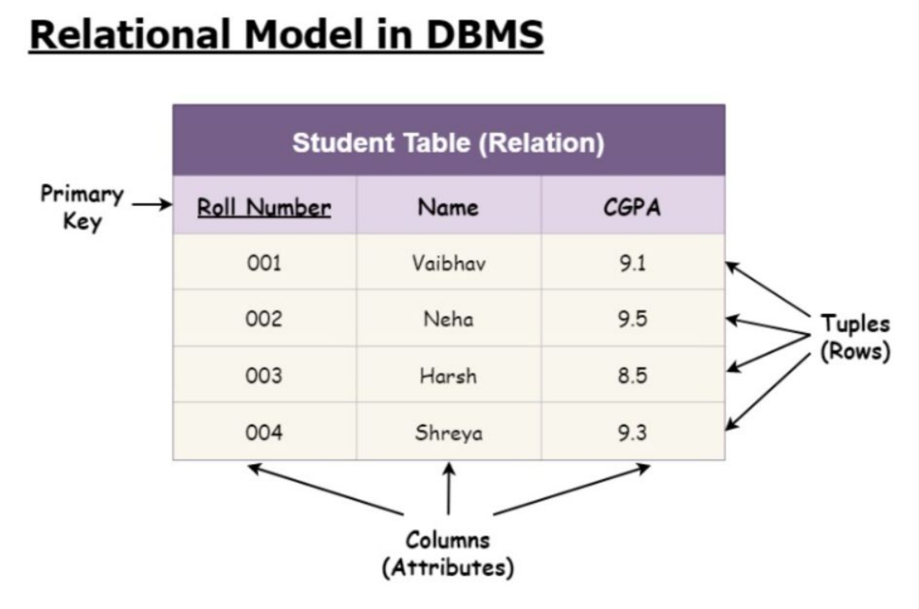
  - Relational data structure
  - Integrity rules.

Relational Model: Relationship concept

In the relational model, the relationship is definition of the structure of the table, i.e. its name and the list of attributes that compose it.

We use the primary key to distinguish one record from another.

There may be more combinations of attributes in a relationship that can uniquely identify a row (these will be called "candidate keys"), but from these only **one will be chosen** to be used as the **primary key.**



Relational Model in DBMS

Student Table (Relation)

Roll Number	Name	CGPA
001	Vaibhav	9.1
002	Neha	9.5
003	Harsh	8.5
004	Shreya	9.3

Relational Model: Elements

Relationship (table): These represent the entities for which information is to be stored in the DB. It is made up of:

- **Rows** (Records or **Tuples**): Correspond to each occurrence or value of the entity.
- **Columns** (**Attributes** or fields): Correspond to the properties of the entity.

Conditions:

- Each relation has a name and this name is different from the name of all the other relations of the same DB.
- No two attributes in the same relation have the same name.
- The order of the attributes (columns) does not matter.
- The order of the tuples (rows) does not matter.
- Each tuple is distinct from the others: there are no duplicate tuples (at least they will differ in the primary key).

Relational Model: Elements

- **Candidate key**: attribute that uniquely identifies a tuple. Any of the candidate keys could be chosen as the primary key.
- **(PK) Primary Key**: Candidate key that we choose as the identifier of the tuple. A primary key cannot assume a null value (**entity integrity**).
- **Alternative Key**: Any candidate key that is not a primary key (those that we have not chosen as the primary key).
- **(FK) Foreign Key:** The attribute or set of attributes that form the primary key of another relationship. In other words, the values present in the foreign key must correspond to values present in the corresponding primary key (**Referential Integrity**).

Relational Model: Elements

- **Domain of an attribute**: Set of values that can be assumed by that attribute.
- **Degree**: The total number of attributes which in the relation is called the degree of the relation.
- **Cardinality**: Total number of rows present in the Table.

Relational Model: Constraints

There are many types of Integrity Constraints in DBMS. Constraints on the Relational database management system is mostly divided into three main categories are:

- Domain Constraints
- Key Constraints
- Referential Integrity Constraints

Relational Model: Constraints

Domain Constraints

Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type. For example:

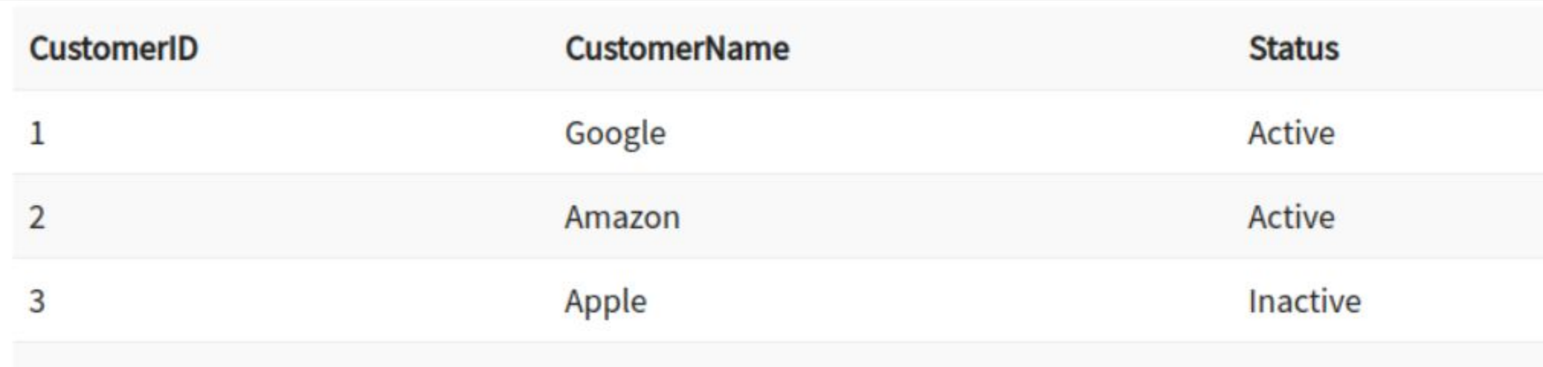
```
Create DOMAIN CustomerName
 CHECK (value not NULL)
```

The example shown demonstrates creating a domain constraint such that CustomerName is not NULL

Relational Model: Constraints

Key Constraints

An attribute that can uniquely identify a tuple in a relation is called the key of the table. The value of the attribute for different tuples in the relation has to be unique.



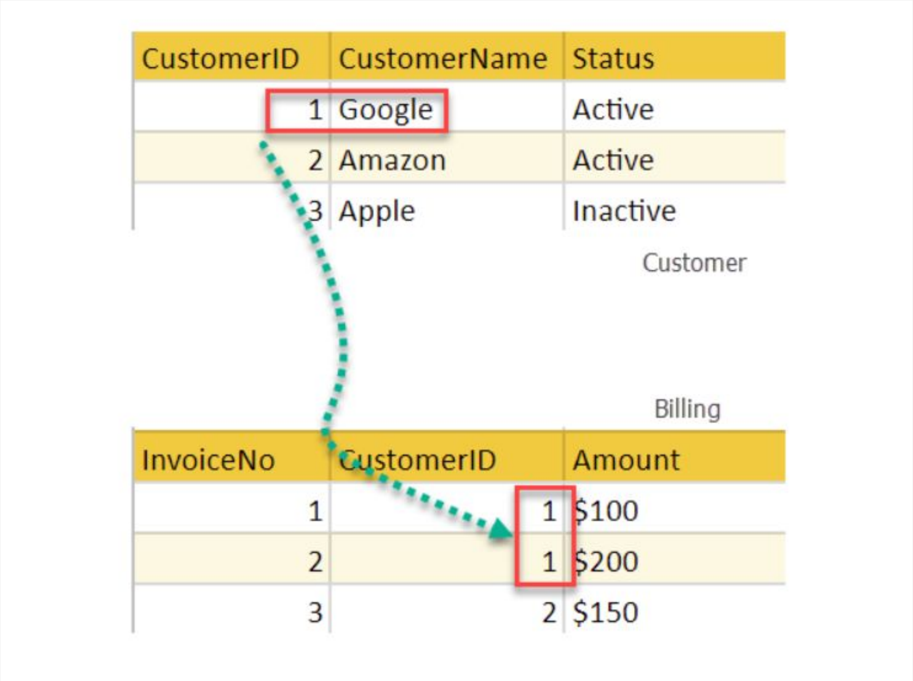
CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

CustomerID is a key attribute of Customer Table.

Relational Model: Constraints

Referential Integrity Constraints

A foreign key is an important attribute of a relation which should be referred to in other relationships. Referential integrity constraint state happens where relation refers to a key attribute of a different or same relation. However, that key element must exist in the table.



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	$100
2	1	$200
3	2	$150

From E / R to Relational Model: Entities

ER model is more semantically oriented than the relational model.

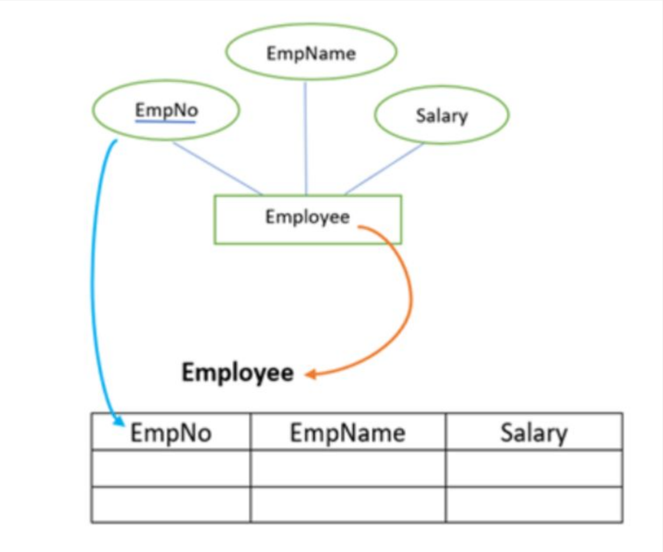
Rules:

- Each entity should be transformed into a table. The table name should be called with the name starting by T Example Entity Employee → Table TEmployee
- All simple attribute will be transformed into a field in a table.
- The **Primary** Key will be the main key to identify the relation. (It should appear underlined). The alternative **key** should be **bond**
- **Mandatory attributes** can not be null (domain constraint)

From E / R to Relational Model: Entities

- **Mandatory attributes** can not be null (domain constraint)

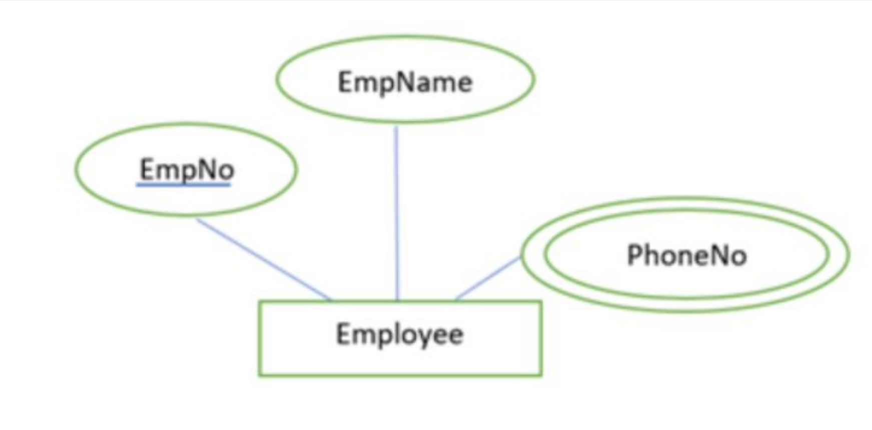
TEmployee
EmpNo
EmpName
Salary

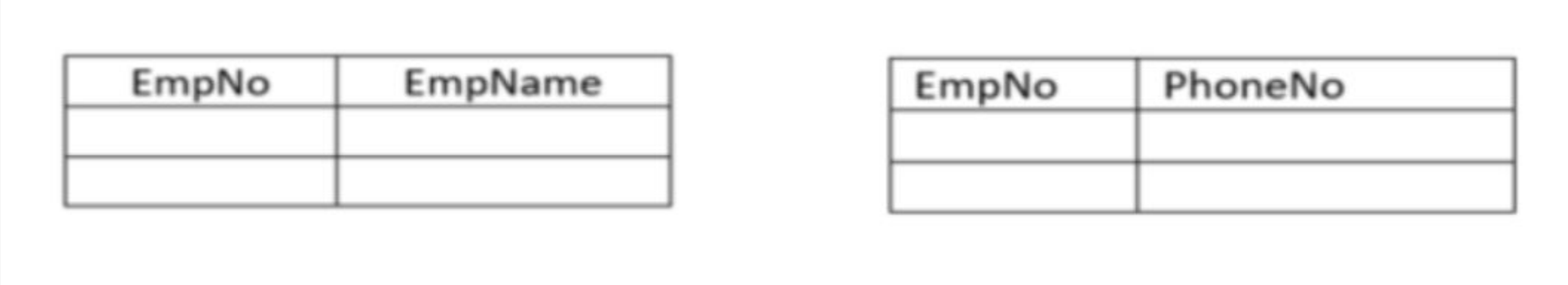


EmpNo	EmpName	Salary

From E / R to Relational Model: Entities

- **Multivalued attributes**: A new relationship is created with the primary key of the entity and the multivalued attribute, both being the primary key of the new relationship.



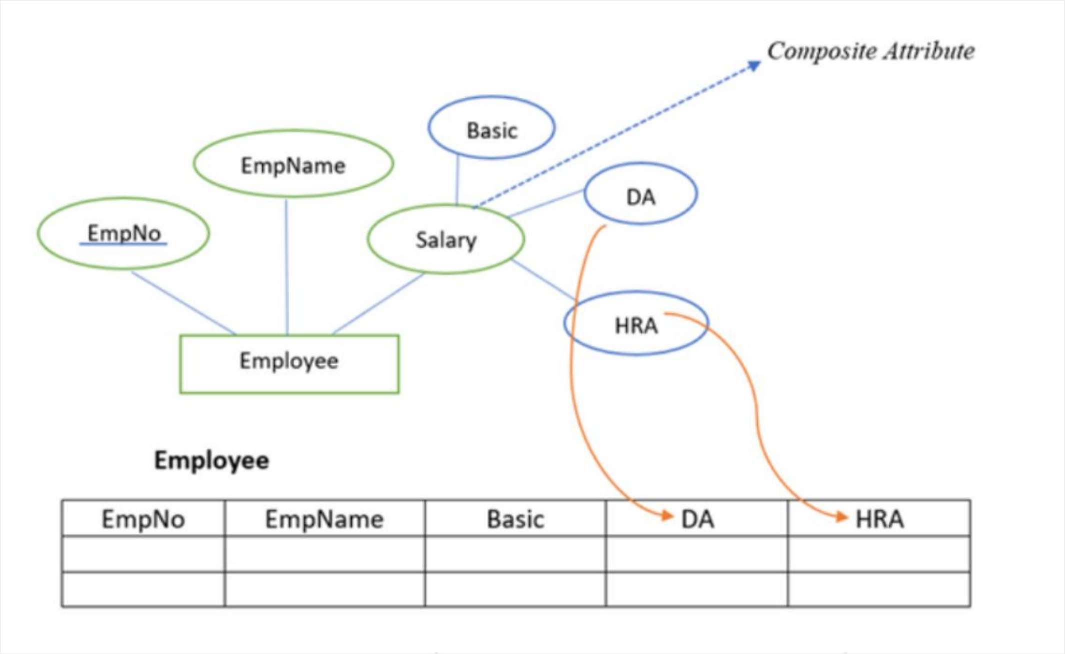


EmpNo	EmpName

EmpNo	PhoneNo

From E / R to Relational Model: Entities

- **Composite attributes** attributes shall be transformed into simple attributes, i.e. one more column in the table.



Employee

EmpNo	EmpName	Basic	DA	HRA

From E / R to Relational Model: N: M Relationships

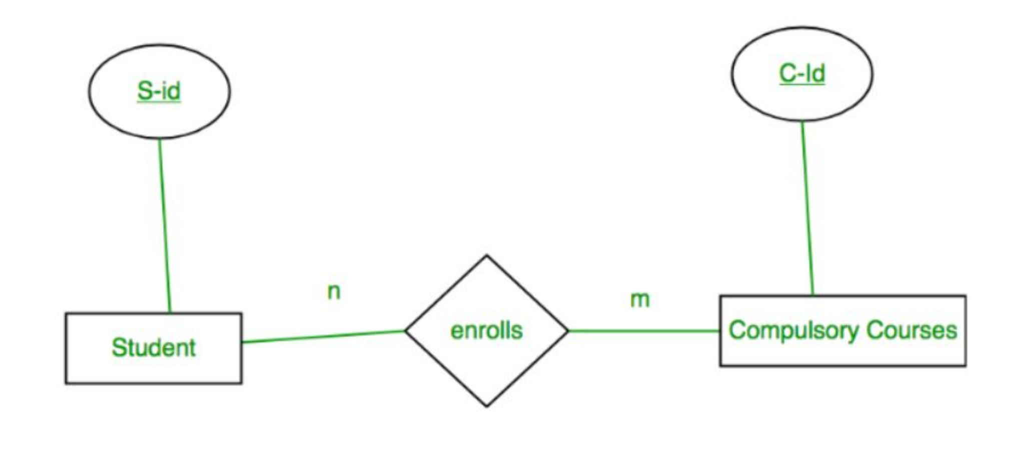
Binary N:M Relationships between entity A and entity B

Build a new table with:

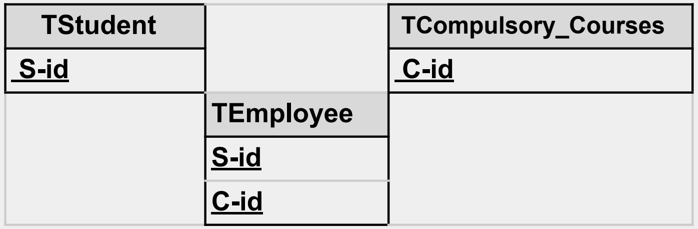
- Add a column for each primary key as Foreign Key
- The primary key will contain these foreign keys and may another column to identify the tuple.
- Relationship attributes will be added as column to this new table.

From E / R to Relational Model: N: M Relationships

Binary N:M Relationships



Relationship’s attributes should be added as a field of its table



TStudent
S-id

TEmployee
S-id
C-id

TCompulsory_Courses
C-id

From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships

For this type there are 3 possibilities:

First Scenario

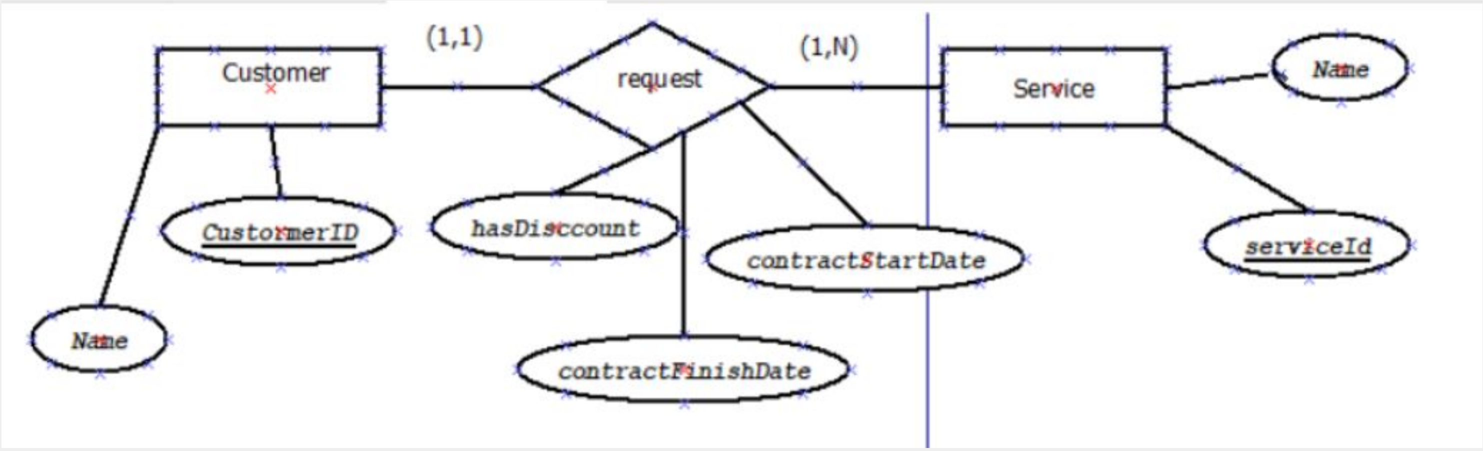
> If there is at least one **attribute for the relationship**, **build a new table** adding the two primary keys as foreigng keys and the relationship attributes and define the primary key for this new table.

From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships

For this type there are 3 possibilities:

First Scenario

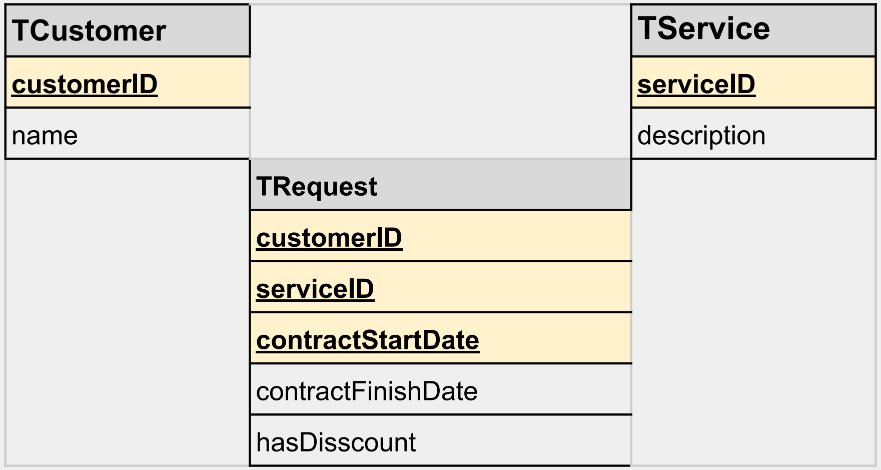


From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships

For this type there are 3 possibilities:

First Scenario



TCustomer
customerID
name

TService
serviceID
description

TRequest
customerID
serviceID
contractStartDate
contractFinishDate
hasDisscount

From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships between entity A and entity B

Second Scenario

If there **one of the cardinalities is (0,1)** (A cardinality) and the other is (0,n) (B cardinality)

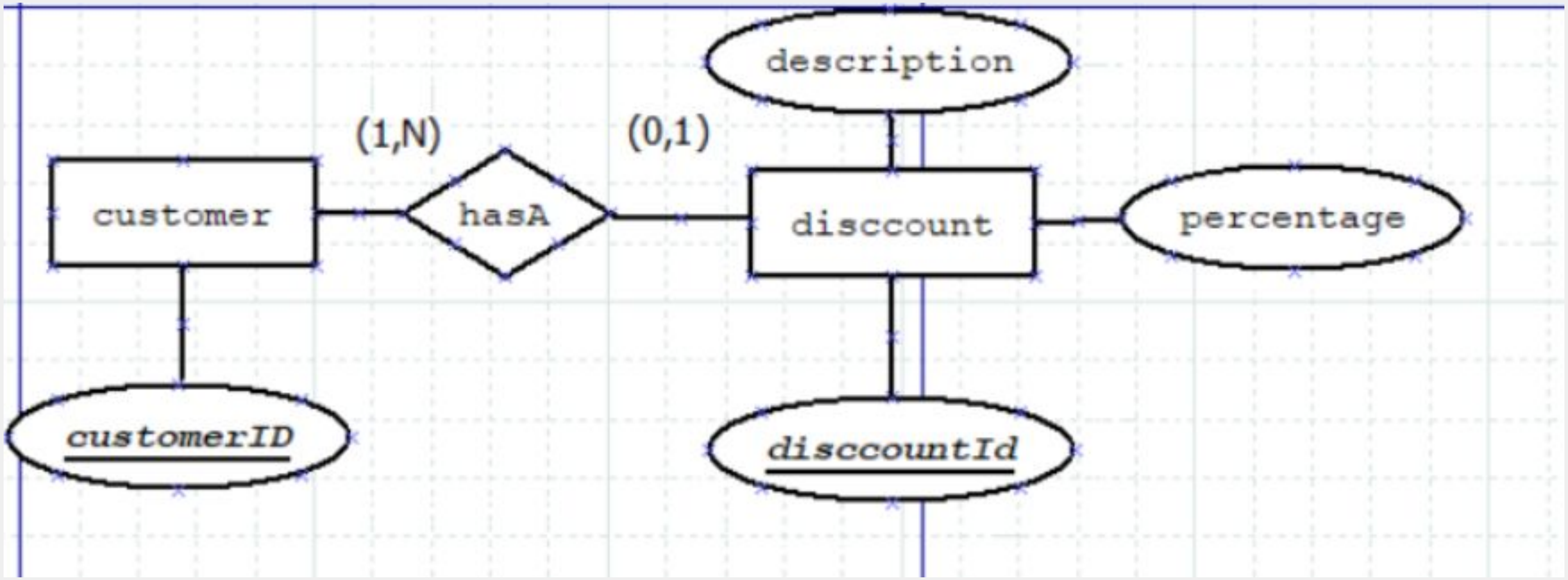
(0,1) cardinality means A entity could exist without relationship with B.

In this scenario, we should **build a new table** for the relationship adding the two primary keys as foreign keys and defining the primary key for this table. This primary key could be the combination of the two foreign keys and maybe another field more.

From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships between entity A and entity B

Second Scenario



From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships between entity A and entity B

Second Scenario

TCustomer
customerID
Name

TDisccount
disccountID
description
percentage

ThasA
customerID
disccountID
description
percentage

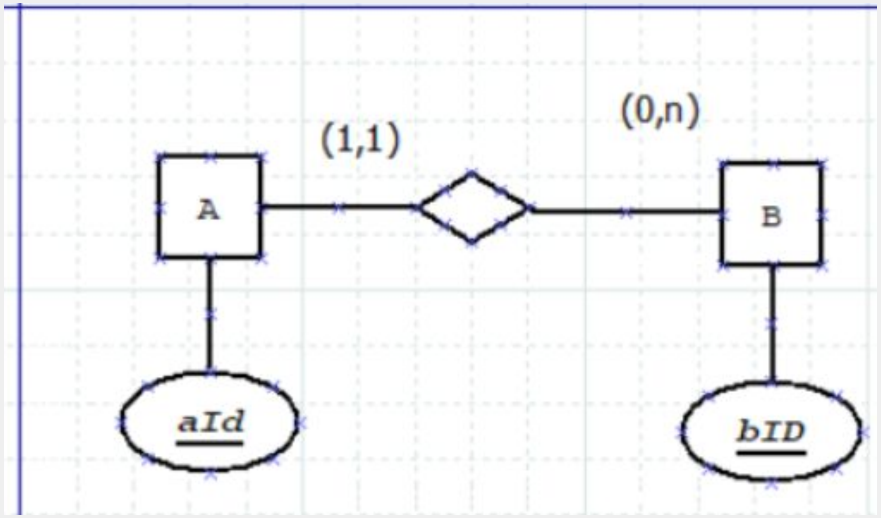
From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships between entity A and entity B

Third Scenario

If there is **no relationship** attributes **and** one of the cardinalities is **(1,1)** (entity a) and the other is (0,n) (B cardinality)

Cardinality (1,1) implies A entity will not exist without relationship

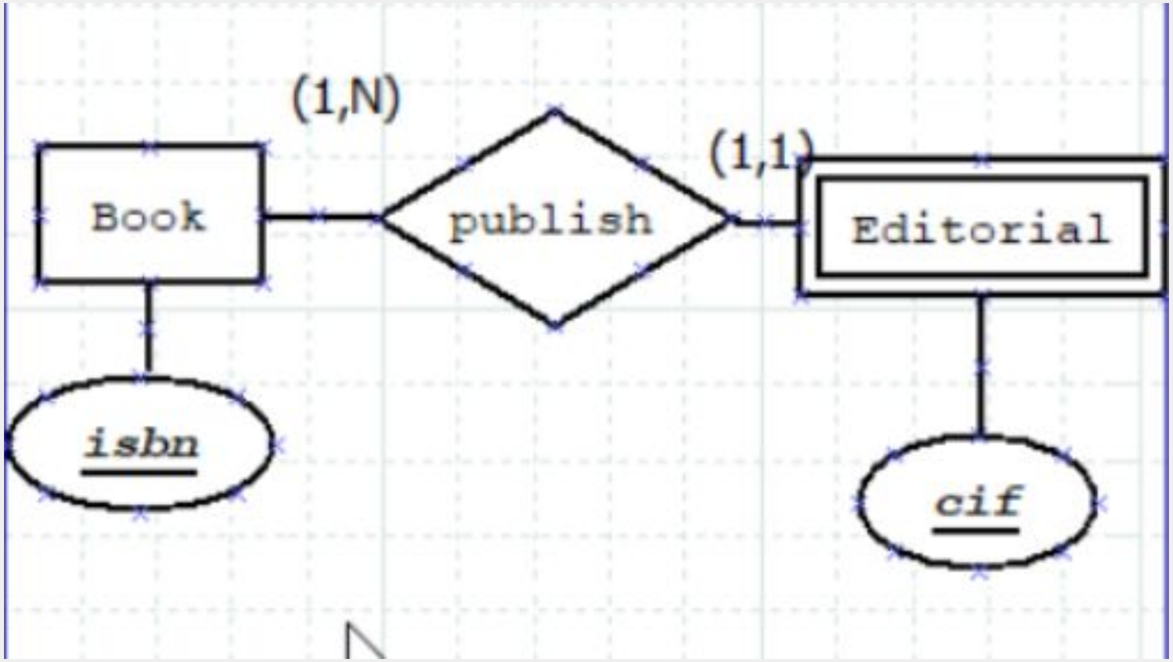


From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships between entity A and entity B

Third Scenario

For example:



From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships

Third Scenario

If there is no relationship attributes and one of the cardinalities is (**1,1**).

We should **propagate the primary key** of the entity having cardinality maximum cardinality **1 to the one with the maximum cardinality N**

From E / R to Relational Model: 1:N Relationship

Binary 1:N Relationships

Third Scenario

For our example, Book (1,n) and Editorial (1,1). We should propagate cif to TBook table as foreign key.

TBook
isbn
cif

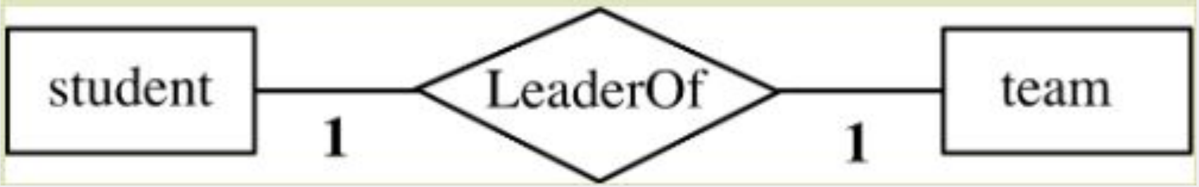
TEditorial
cif

From E / R to Relational Model: 1:1 Relationship

Binary 1:1 Relationships between entity A and entity B

The objective is to minimize empty fields

First Scenario: Both cardinalities are (0,1) or there are relationships attributes



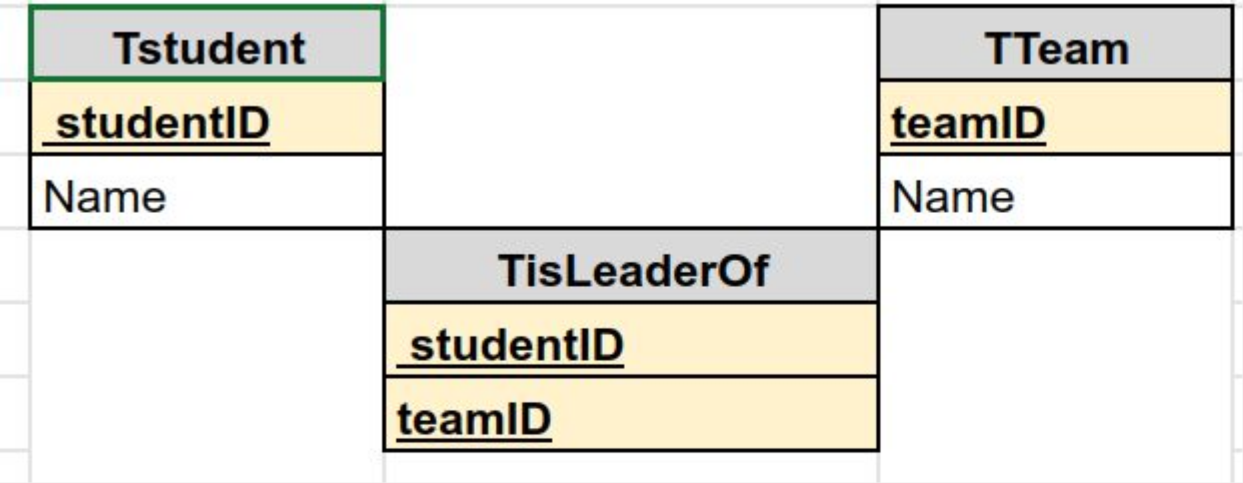
(0,1) (0,1)

A student could be the leader of a team and a team could have or not a leader.

From E / R to Relational Model: 1:1 Relationship

First Scenario: Both cardinalities are (0,1) or there are relationships attributes

In this scenario, we build **a new table** adding both primary keys. Doing that we are minimizing empty fields.



Tstudent
studentID
Name

TTeam
teamID
Name

TisLeaderOf
studentID
teamID

From E / R to Relational Model: 1:1 Relationship

Second Scenario: For all other scenarios

We should add the primary key to the other table. But in which one?

- If there is one (0,1) (entity A) cardinality and another **(1,1)** (entity B). This one implies entity A may not exist without the relation (weak entity). In this case table A will receive primary key from entity B

From E / R to Relational Model: 1:1 Relationship

Second Scenario: For all other scenarios

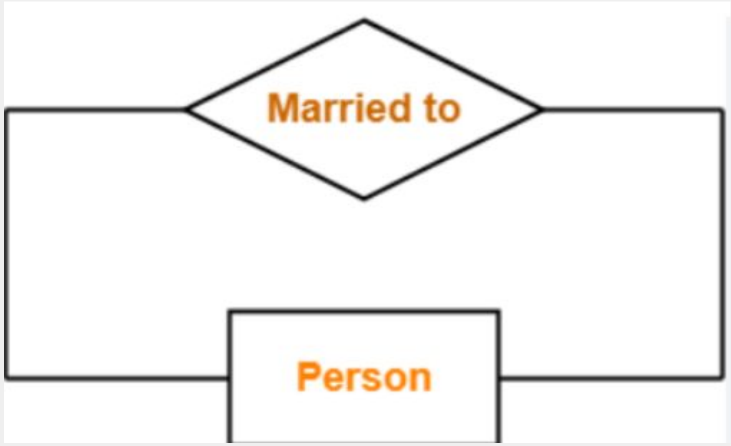
- If both are (1,1), you should review:
  - If one entity should be created before the other. In this case, the second one will receive the primary key from the first one.
  - If both entities exists without depending of the other. For this scenario, there is no rule. You could add a new table or you could manage adding the primary key as foreign key to the other

From E / R to Relational Model: Reflexive Relations

Basically, apply the **same rules as binary relations**

- **Relation 1:1** Un table with two fields: one as a primary key and the other as a foreign key.

1:1



Person		
<u>idPerson (PK)</u>	name	*marriedWith (FK)*

From E / R to Relational Model: Reflexive Relations

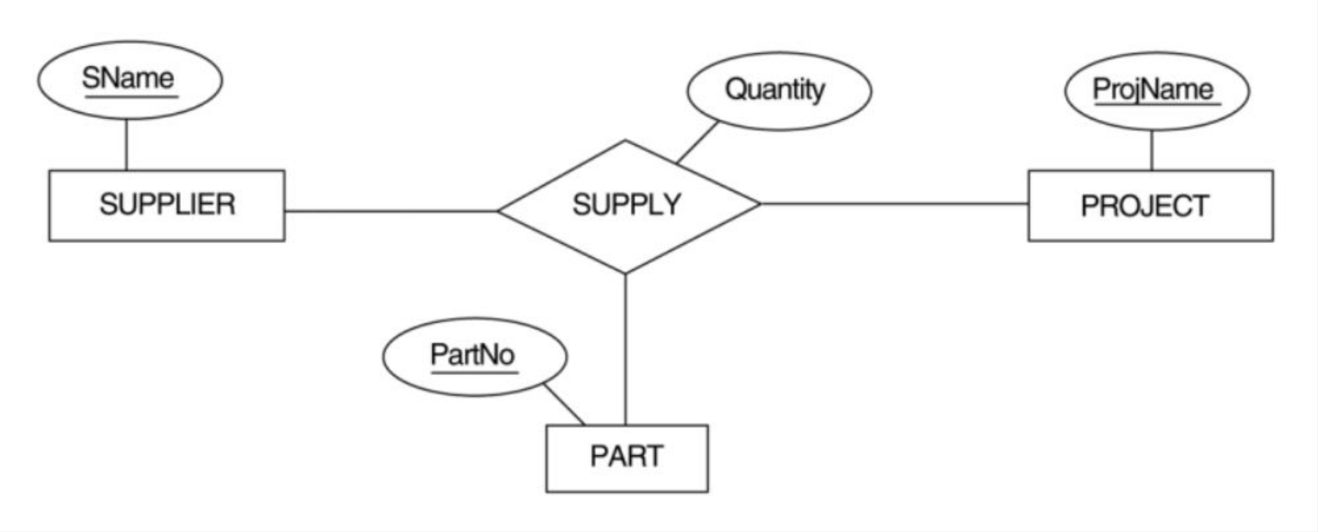
- **Relation N:M** Create a new table with the two primary keys
- **Relation 1:N** We have two cases:
  - The one in which the entity many is mandatory, we proceed as in case (1,1), propagating from primary key from 1 cardinality to N cardinality table
  - If it is not obligatory, a new table is created whose key will be that of the entity and the key is also propagated to the new table as a foreign key.

From E / R to Relational Model: Ternary or more Relationship

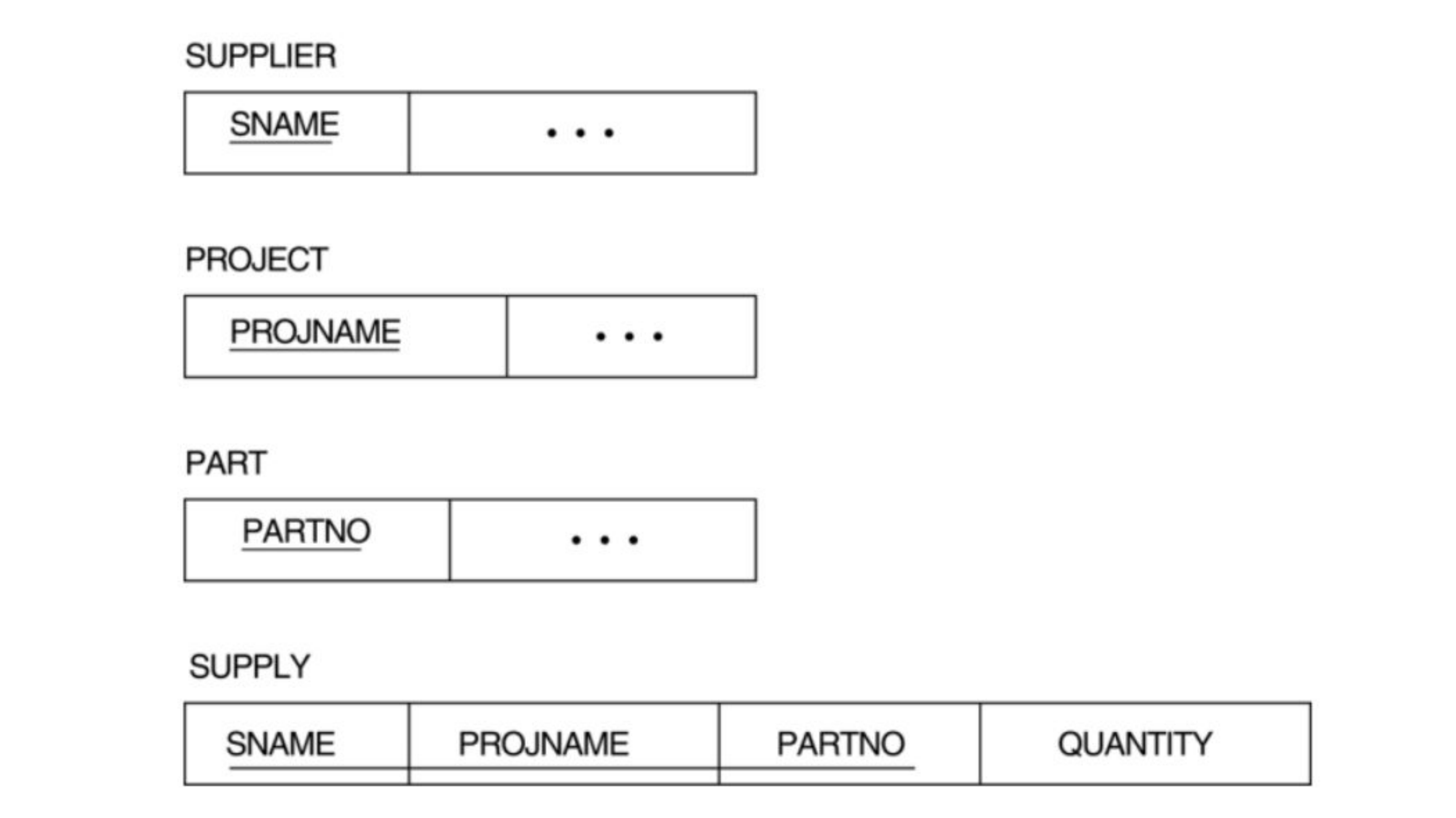
N-ary where N should be > 2

We should add a new table including all the primary keys of the relations and also relation attributes.

After that, we should review cardinality to check if it could be simplified.



From E / R to Relational Model: Ternary or more Relationship



SUPPLIER

SNAME	. . .

PROJECT

PROJNAME	. . .

PART

PARTNO	. . .

SUPPLY

SNAME	PROJNAME	PARTNO	QUANTITY

From E / R to Relational Model: Hierarchical relations

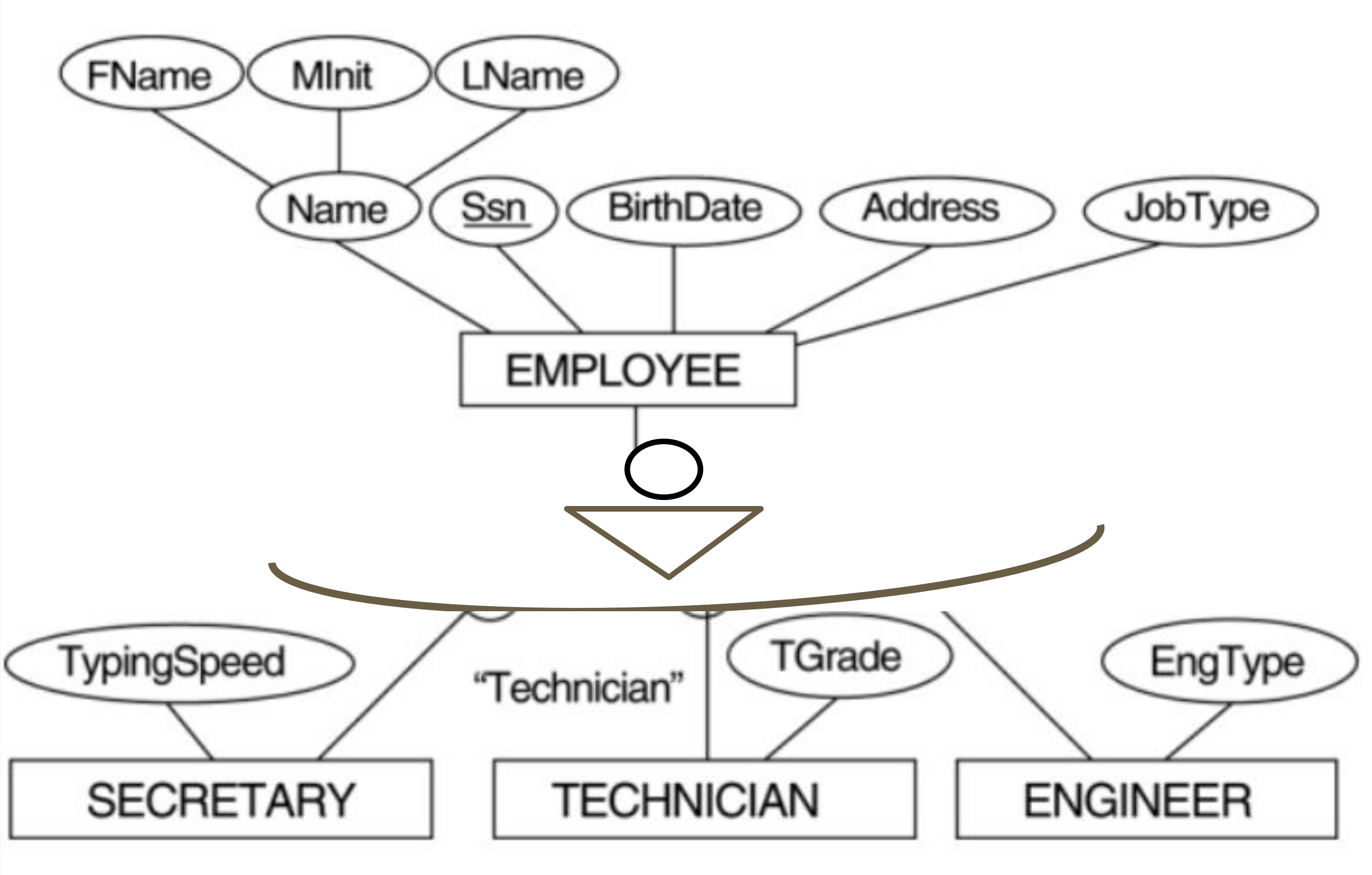
The relational model has no mechanism for the representation of hierarchical relationships. Therefore, these relationships have to be eliminated. This one should involve to lose semantics. For example:

- Exclusive Relations → No way to model with E/R. It will be managed as a restriction on the database.

To manage that there are different options:

1. First option, to build an unique table with parent attributes and all the attributes **from children entities to parent** It could imply many null attributes and big tables.
2. Second option, all the common attributes which are allocated in the parent entity will be **propagate to the children**. That solution only is possible f**or total and exclusive relations.**
3. Third option, to transform to **one 1:1 relationship between the parent and each children**.

From E / R to Relational Model: Hierarchical relations



From E / R to Relational Model: Hierarchical relations

1. First option:

   TEmployee(SSN, Name, Fname, MInit, LName, BirthDate, Address, JobType, TypingSpeed, TGrade, EngType)

2. Second option: It is not possible because it is overlapped.
3. Third option:

   Employee(SSN, Name, Fname, MInit, LName, BirthDate, Address)

   Secretary(SSN, TypingSpeed)

   Technician (SSN, TGrade)

   Engineer (SSN, EngType)

Summary Transformations from E/R to Relational

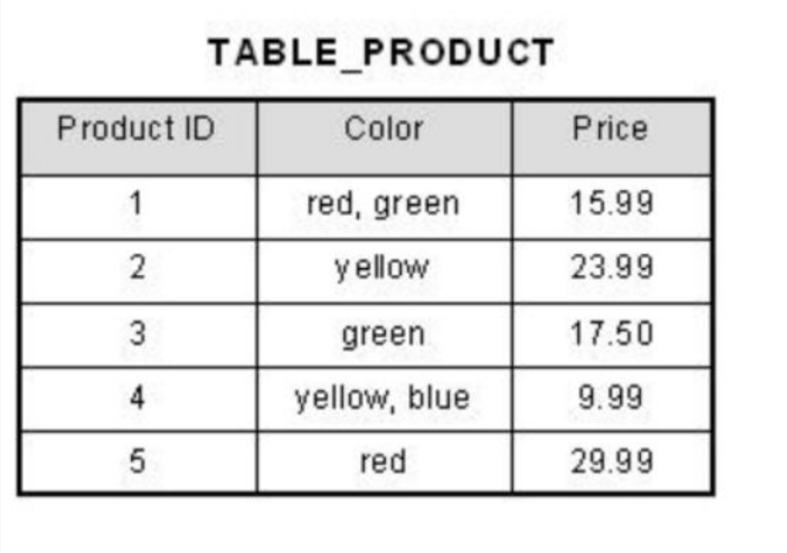
E/R Model			Relational Model
Entity			Create a new table (PK)
Attribute	Simple		New Column in Entity table Mandatory can not be null
	Composite		New Columns for each simple attributes
	Multivalued		New table with the entities' primary keys and the attribute, adding a row per value
Relations	Binary N:M Cardinality		Add a new table with the two primary keys
	Unary, Binary 1:N	At least one attribute in the relationship	Add a new table with the two primary keys
		One cardinality (0,1) (A)	Add a new table with the two primary keys
		ONe cardinaliy (1,1) (A)	Propagate the A (cardinality 1) primary key to the B table (cardinality N) as foreign key
	Unary, Binary 1:1	At least one attribute in the relationship	Add a new table with the two primary keys
		Both (0,1)	Add a new table with the two primary keys
		(0,1)A (1,1) B or both (1,1)	Propagate the B (cardinality 1,1) primary key to the A table (cardinality 0,1) as foreign key
	Ternary or more		Add a new table adding all the primary keys and after review it could be simplied

Normalization: 1FN

Normalization is the process of structuring a relational database in order to **reduce data redundancy** and **improve data integrity**.

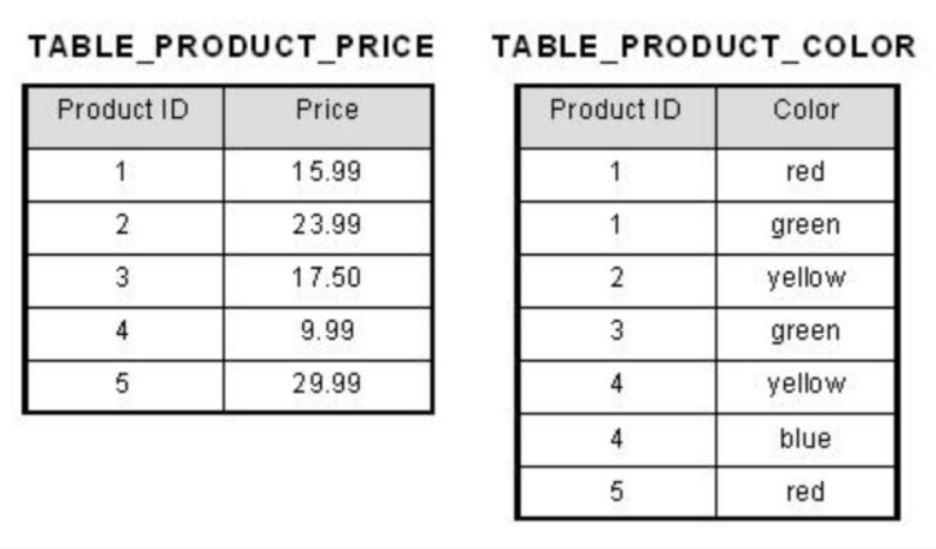
There are different levels of normalization.

- **First Normal Form (1FN)** A relationship is in first normal form, if there **no multivalued attributes**.



TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99



TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

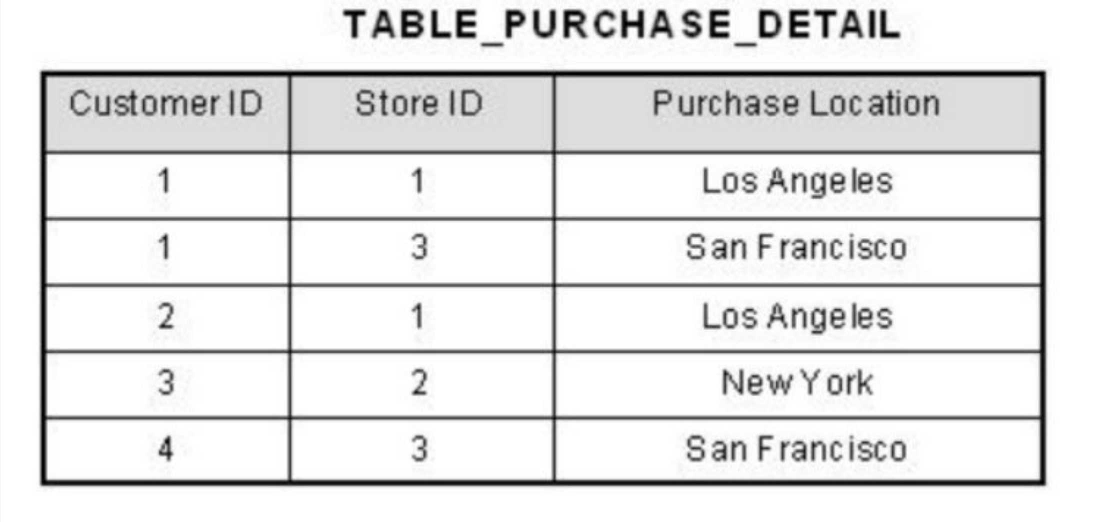
Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

Transformation to 1FN

Normalization: 2FN

- A relation is in the **Second Normal Form** (**2FN**) if it fulfills the following two requirements:
  - It is in first normal form.
  - All attributes that are **not part of the primary key are fully dependent on it**. It means that each non key field must be about the same thing as the primary key.

Normalization: 2FN

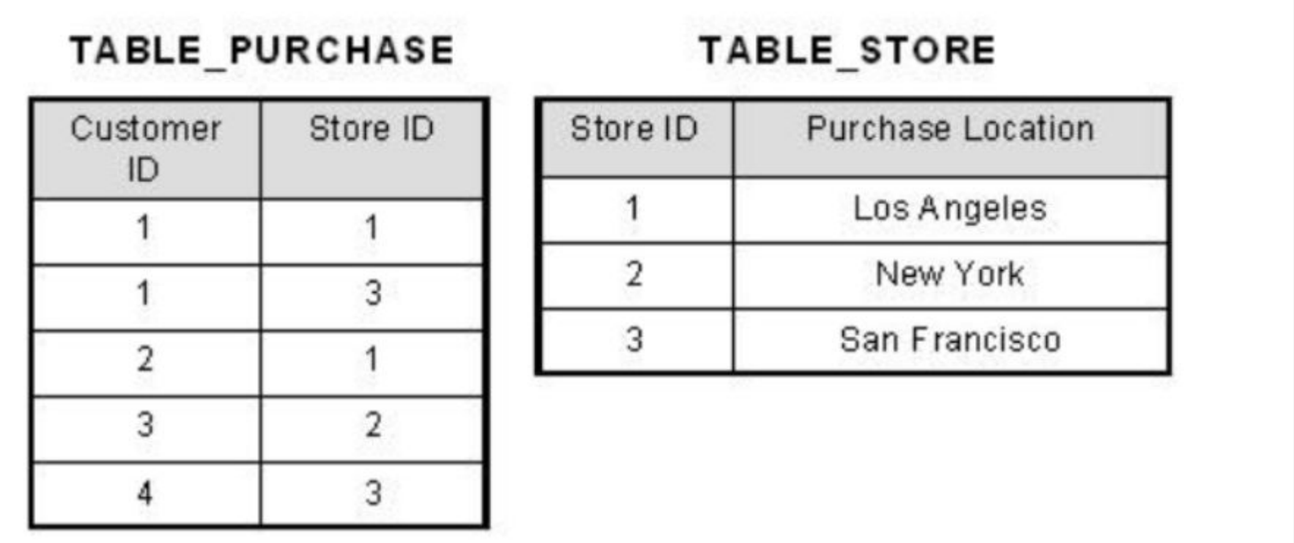


TABLE_PURCHASE_DETAIL

Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

This table has a **composite primary key [Customer ID, Store ID]**. There is a non-key attribute [Purchase Location]. In this case, **[Purchase Location] only depends on [Store ID]**, which is only part of the primary key. Therefore, this table does not satisfy second normal form.

Normalization: 2FN



TABLE_PURCHASE

Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3

TABLE_STORE

Store ID	Purchase Location
1	Los Angeles
2	New York
3	San Francisco

Transformation to 2FN

Normalization: 3FN

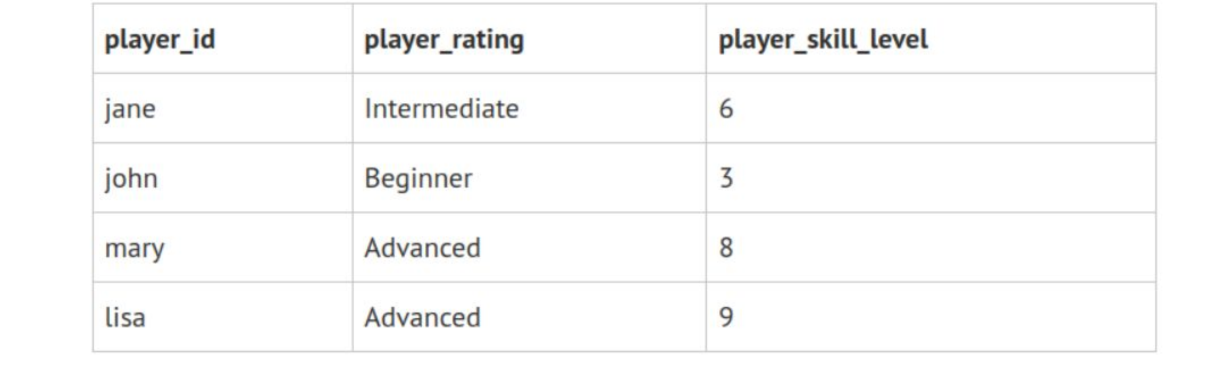
- A relation is in the **Third Normal Form** (**3FN**) if it fulfills the following two requirements:
  - It is in second normal form.
  - If all attributes that are **not part of the primary key are independent of each other**, i.e. they do not give information about other attributes of the relationship. This type of dependency is called functional dependency in the relational schema. In other words, a table is considered in third normal if the table/entity is already in the second normal form and the columns of the table/entity are non-transitively dependent on the primary key.

Normalization: 3FN

Transitive dependency. It happens when an attribute depends, in addition to the primary key (2FN), on another non-key attribute, then it is said to have transitive functional dependency.

3FN implies not to have transitive dependencies.

Normalization: 3FN



player_id	player_rating	player_skill_level
jane	Intermediate	6
john	Beginner	3
mary	Advanced	8
lisa	Advanced	9

1FN? Yes, there is not multivalued attributes

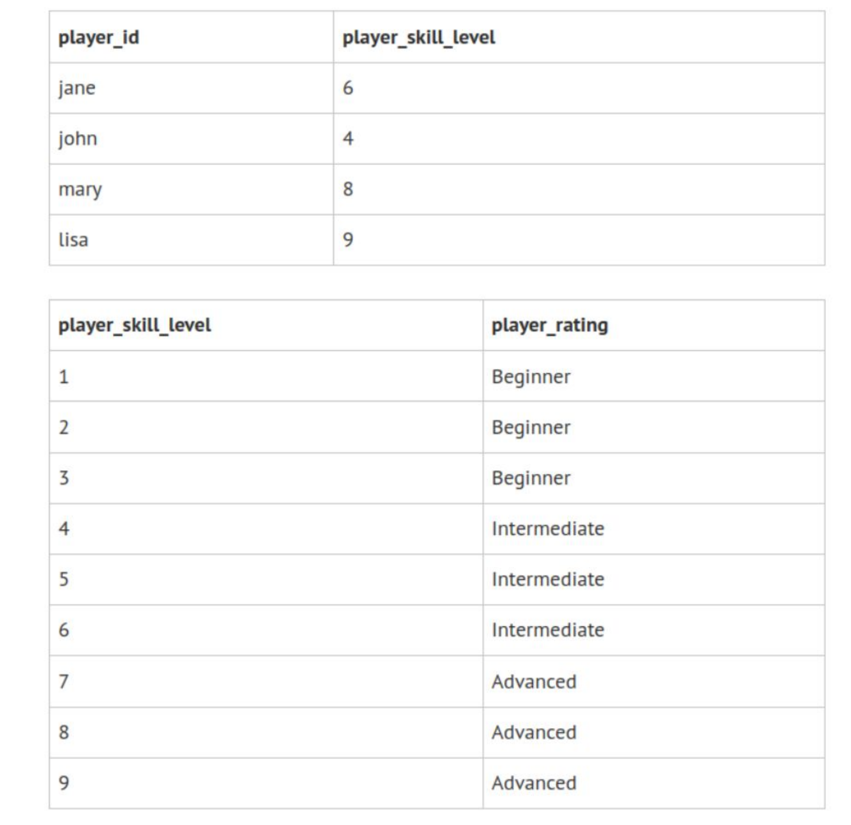
2FN? Yes, the primary key is player_id and t the rest of the attributes depend on the primary key.

3FN? Not, because player_skill_level depends on the player_rating

Normalization: 3FN

Third form Transformation

Split the in two tablets: one with the primary key and the level and a second one with the level and the rating



player_id	player_skill_level
jane	6
john	4
mary	8
lisa	9

player_skill_level	player_rating
1	Beginner
2	Beginner
3	Beginner
4	Intermediate
5	Intermediate
6	Intermediate
7	Advanced
8	Advanced
9	Advanced

References

- Rafael Lozano, Introducción a las Bases de Datos
- Arturo Mora Rioja. Base de datos. Diseño y Gestión. Editorial Sintesis Madrid 2014
- Antonio Postigo Palacios, Gestión de base de datos Editorial Paraninfo Madrid 2022
- https://www.guru99.com/relational-data-model-dbms.html
- https://www.scaler.com/topics/dbms/relational-model-in-dbms/