Chapter IV: Methodology for Developing an Information System (MERISE)

1. Information System Development Process

The development process of an information system (IS) is a structured set of activities aimed at achieving the objectives of an IS project defined by the organization. These activities vary according to the type of organization, the type of project, and the type of system to be developed. This process must be clearly described in order to be properly managed.

2. Software Life Cycle

The life cycle of an information system (IS) succinctly describes the phases through which an information system goes, from the initial need to the system's retirement.

3. IS Development Activities

The development of an information system (IS) involves several activities that vary depending on the type of project and the organization. The essential activities include:

- **Specification of system requirements and constraints**, and the preparation of a requirements document
- **Design of the solution**, including the creation of a model for the system to be developed
- **Implementation of the system**
- **System testing**, to verify the alignment between the system's implemented properties and the specified requirements
- **System installation** at the client's site and verification of its proper functioning
- **System maintenance**, including fault repair

4. IS Development Models

There are various models used for software development. These models aim to implement the development activities mentioned above with a certain level of organization among these activities. The most well-known models include the **Waterfall model**, the **V-model**, the **Spiral model**, and the **Incremental model**. The application of these models often involves the use of an analysis and design method. Each method has its advantages and disadvantages and is suited to a specific type of project

(industrial, management, scientific, etc.). Some of the existing methods include **MERISE**, **SADT**, **SART**, **OMT**, and **UML** (although UML is not a method but a unified modeling language).

5. The MERISE Method (in French Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise)

MERISE is a systemic method that emerged in 1979 following a project initiated in 1977 by the French Ministry of Industry. The goal was to provide companies with a design method that would enable them to successfully complete their IT projects within the planned budget and timeline.

5.1. Characteristics of Merise

The major advantages of Merise are:

1. A comprehensive approach to the IS conducted simultaneously on data and processes.
2. A description of the IS using a simple and rigorous formalism standardized by ISO for data representation (Entity-Relationship Model).
3. Separation of data and processes.

5.2. Levels of Abstraction in Merise

5.2.1. Conceptual Level

The conceptual level aims to answer the question WHAT? In other words, what needs to be done and with what data, without considering work organization or the hardware used. The two resulting models at this level are:

- **Conceptual Data Model (CDM)**
- **Conceptual Process Model (CPM)**

5.2.2. Organizational Level

The organizational level seeks to answer the questions WHO? WHERE? and WHEN? At this level, the organizational criteria of work are integrated. It considers (or proposes) the distribution of processes between humans and machines, as well as the mode of operation (real-time, batch processing). The resulting models at this level are:

- **Logical Data Model (LDM)**
- **Organizational Process Model (OPM)**

5.2.3. Operational (Physical) Level

This level focuses on implementing the results of technical decisions based on objectives and technical constraints. It answers the question HOW?

At this stage, technical solutions are studied (e.g., data storage methods, program segmentation for processes). The resulting models at this level are:

- **Physical Data Model (PDM)**
- **Physical Process Model (PPM)**

The levels of abstraction and their resulting models are summarized in the following table:

LEVEL	DATA	TREATMENTS
Conceptual *Management choice: What?*	Conceptual Data Model (CDM)	Conceptual Model of Treatments (CTM)
Organizational *Organizational choice: Who? Where? When?*	Logical Data Model (LDM)	Organizational Model of Treatments (OMT)
Physical *Technical choices: How?*	Physical Data Model (PDM)	Physical Model of Treatments (PMT)

5.3. Phased Breakdown

MERISE recommends dividing the project into four phases. This division is not specific to the Merise method but is generally advised for the execution of any IT project. Each phase corresponds to a level of abstraction. The phases are as follows:

1. **Preliminary Study / Master Plan**
2. **Detailed Study**
3. **Implementation**
4. **Deployment**
5. **Maintenance**

Breakdown of the MERISE method into stages
Master plan / Preliminary study (study of the existing)
Detailed study (in parallel by two teams if possible) CDM: Conceptual Data Model CTM: Conceptual Treatment Model OMT: Organizational Model of Treatment External views / Validation MLD: Logical Data Model
Realization(together) PDM: Physical Data Model PMT: Physical Model of Treatment
Implementation
Maintenance

6. Preliminary Study (Existing State Study)

The goal of this phase is to:

- Gain detailed knowledge of the area where a new automation is being considered.
- Gather a comprehensive list of the objectives the company pursues in this domain.
- Identify management, organizational, and technical constraints.

6.1 Gathering Existing Information:

Existing information can only be collected from two entities: "Management" and "Workstation." The primary technique used is interviews, which can be supplemented with questionnaires, surveys, documents, etc.

6.1.1 Management Interviews:

- Gain an initial understanding of the problem.
- List the objectives required.
- Identify the main workstations involved.
- Define interfaces with other projects.
- Outline the study’s scope.

6.1.2 Workstation Interviews:

- List and describe the tasks performed.
- Observe information flow.
- Learn the company's terminology.

6.2 Tasks and Formalisms Accompanying the Existing State Study

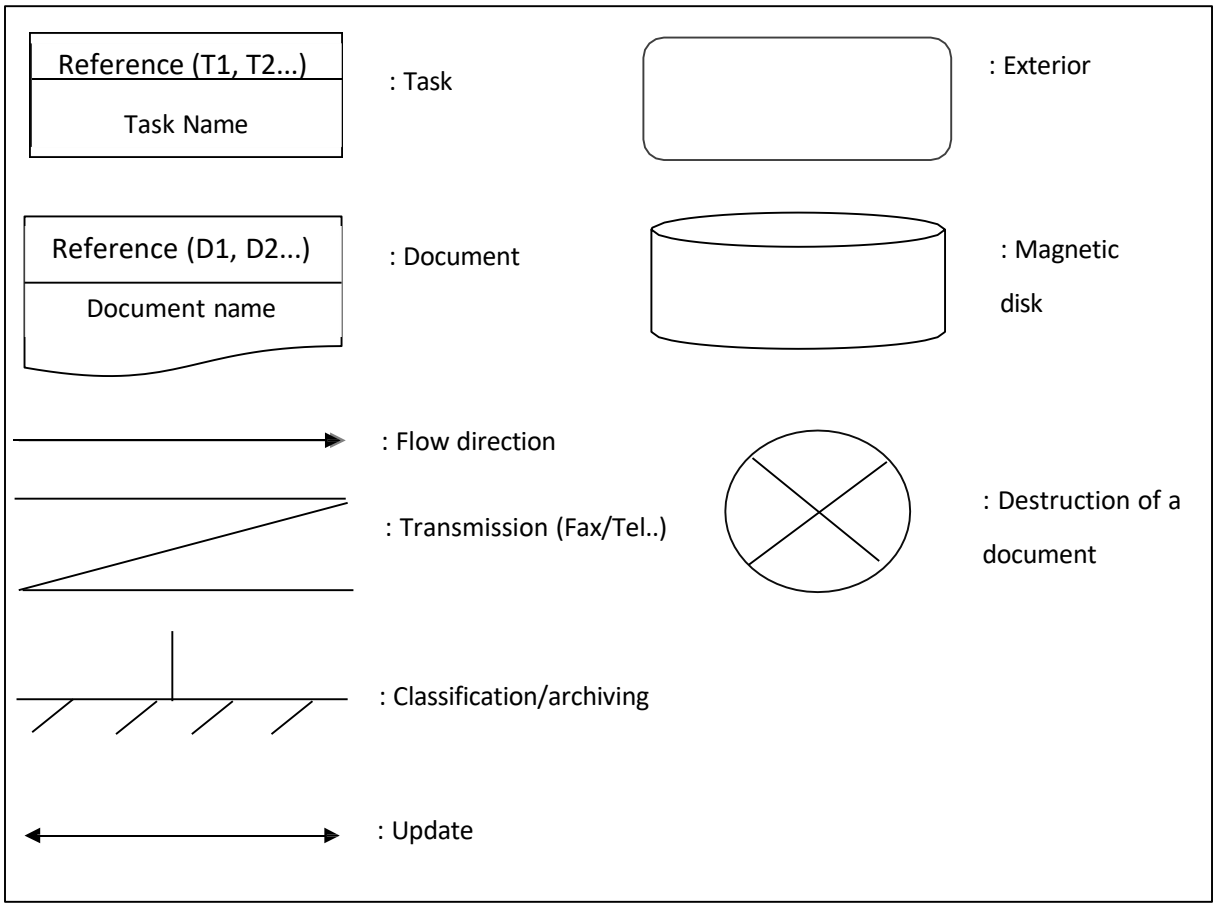
During the existing state study phase, we must perform the following tasks:

- Visualized interviews using task-document diagrams.
- Creation of a flow chart.
- Listing of rules (management, organizational, and technical).

- Creation of a data dictionary.

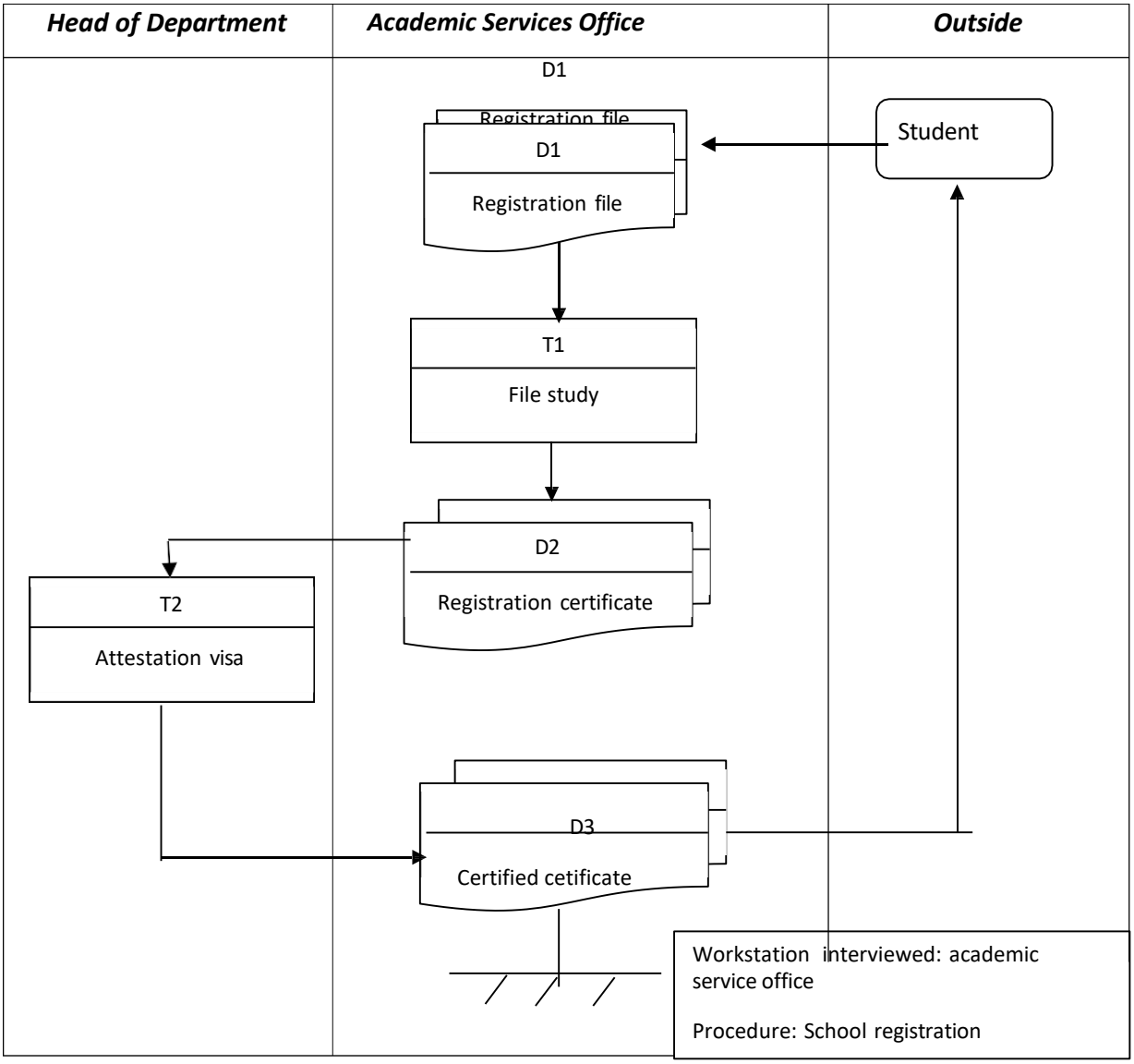
6.2.1 Task-Document Diagrams

Throughout the interviews, task-document diagrams will be constructed. These will illustrate the sequence of tasks through the documents that initiate them and those they produce. To streamline the diagrams and add specific comments related to all tasks and documents, references to these are provided separately. The symbolism used in task-document diagrams is as follows:



For organizing the diagram, external actors (if any) are placed in the rightmost column, the central column is reserved for the relevant workstation (interviewee), and the leftmost columns are reserved for actors within our study scope (if any).

Example: School registration



Document number	*Wording/role*	*Task numbers*
D1	Registration file: contains the documents necessary for registration such as….	T1
D2	Registration certificate: this is a visa-free certificate	T1, T2
D3	Certified certificate: registration certificate signed by the head of department	T2

Task number	*Description of the task*	*Workplace*	*Frequency and volume*	*Input documents*	*Output documents*
T1	Study of registration file. Check if the student's documents are compliant and complete	academic service office	At the start of the academic year	D1	D2
T2	Visa of the certificate. Before signing the certificate we must check its conformity.	Head of Department	At the start of the academic year	D2	D3

6.2.2 Flow Chart

The purpose of this diagram is to show the flow of documents among the different actors. This chart has already been described in the previous lessons.

6.2.3 Rules Inventory

Task execution and data handling within a company are always governed by a set of rules. This phase aims to identify rules that are often vaguely or ambiguously expressed in task descriptions or documents collected, and to categorize them into different levels of abstraction. Rules can be expressed in various ways:

- In common language
- Using mathematical formulas
- Through pseudocode
- With decision tables, flowcharts, etc.

There are three types of rules:

Management Rules:
These describe the "what" of the company, meaning they convey the objectives chosen and the constraints accepted by the company. They outline the actions that must be accomplished and the regulations attached to these actions (laws, regulations, calculation rules, etc.). Management rules are divided into two types:

- **Action Rules** that describe the actions the company must perform. *Examples:*
  - Every delivered product will be added to inventory.
  - A student cannot enroll in two different levels.
- **Calculation Rules** that describe how actions should be carried out. *Example:* Base salary = index × number of points.

Organizational Rules:
These describe the "where," "who," and "when," representing the organizational structure in place to achieve the set objectives. *Examples:*

- Orders must not be placed on Fridays.
- The cash register must be closed by the cashier every day at 4 p.m.

Technical Rules:

These describe the "how," representing the technical conditions for implementing tasks.

Example: Auxiliary memory capacity must be at least 20 gigabytes.

6.2.4 Data Dictionary

The Data Dictionary (D.D.) represents the set of properties included on the documents used by the different workstations within our study scope. The model used to create a D.D. might look as follows:

Code	Meaning	Type	Length	Syntax rules	Semantic rules
St_code	Student code	AN	12	2024/INF/0185	
St_first_name	Student first name	A	20		
St_last_name	Student last name	A	20		
St_birth_date	Student birth date	D	10	DD/MM/YYYY	
St_average	Student average	N	5	XX,XX	0≤ St_average ≤ 20
----------	----------	------	-------	----------	----------

Dictionary Cleansing (purification)

After creating the dictionary, it must be refined by eliminating synonyms and polysems terms.

- **Synonyms**: Different names refer to the same concept.

  Examples:

  - `Num_commande` and `Ref_commande`
  - `Agent` and `Employé`
- **Polysemes**: A single name refers to two or more distinct concepts.

  Example:

  - `Num_C` is used for both client number and order number.

7. Detailed Study

7.1 Conceptual Level

7.1.1 Data: CDM (Conceptual Data Model)

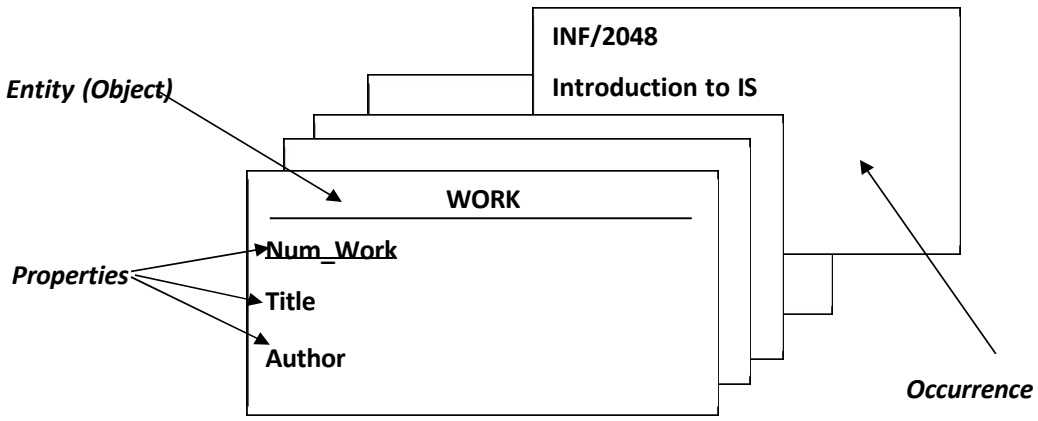
The objective of the **Conceptual Data Model** (CDM) is to provide a static, schematic representation of the data related to the management domain in question. This model is built around the following concepts:

A. Property (Attribute)

The finest level of data (referred to as "unbreakable"). It assumes that the data has been purified of polysemes and synonyms. It represents a property associated with entities and associations to describe them. The properties must align with the company's management choices.
Example: employee_name, stock_Qte, Address, birth_date, etc.

B. Entity (Object)

An object of the real world (either concrete or abstract), about which information is to be recorded and which has its own existence. An entity is composed of a collection of specific properties, where each occurrence can be unambiguously identified using a specific property called an "identifier." Each value of this identifier corresponds to a unique occurrence. "For each occurrence, all properties must take one and only one value."
Example:

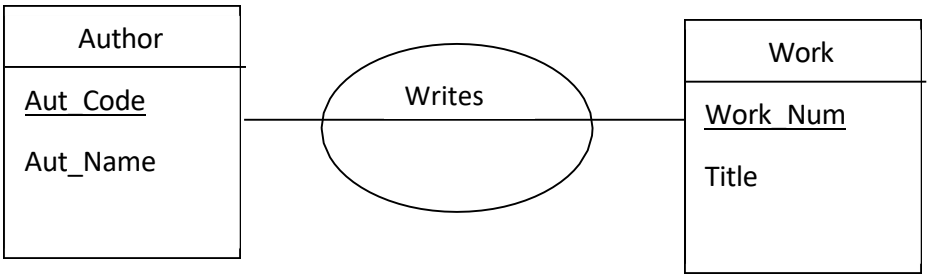


C. Association (Relationship)

An association represents the link between two or more entities, where each entity plays a specific role in the relationship. Each occurrence of an association is identified by concatenating the identifiers of the entities involved in the association. Associations are typically symbolized by conjugated verbs.

Examples:

- An author **writes** a work.
- A customer **places** an order.



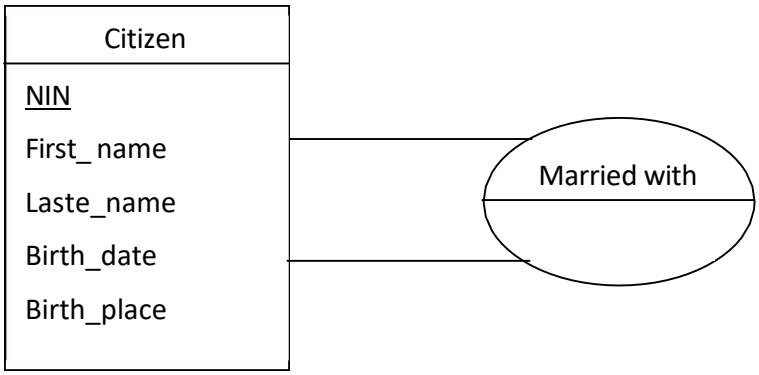
Dimensions of an Association:

The **dimension** of an association refers to the number of objects (entities) involved in the relationship. Based on the size, we can distinguish some particular types of associations:

- **Dimension = 1 → Reflexive Association**
  A reflexive association (dimension 1) connects an entity to itself. It represents a relationship where an entity is related to another instance of the same entity.

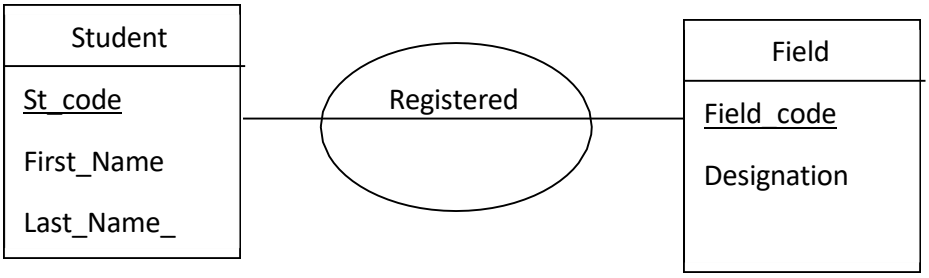
Example:

An employee **supervises** another employee. (The entity "employee" is related to another instance of "employee.")



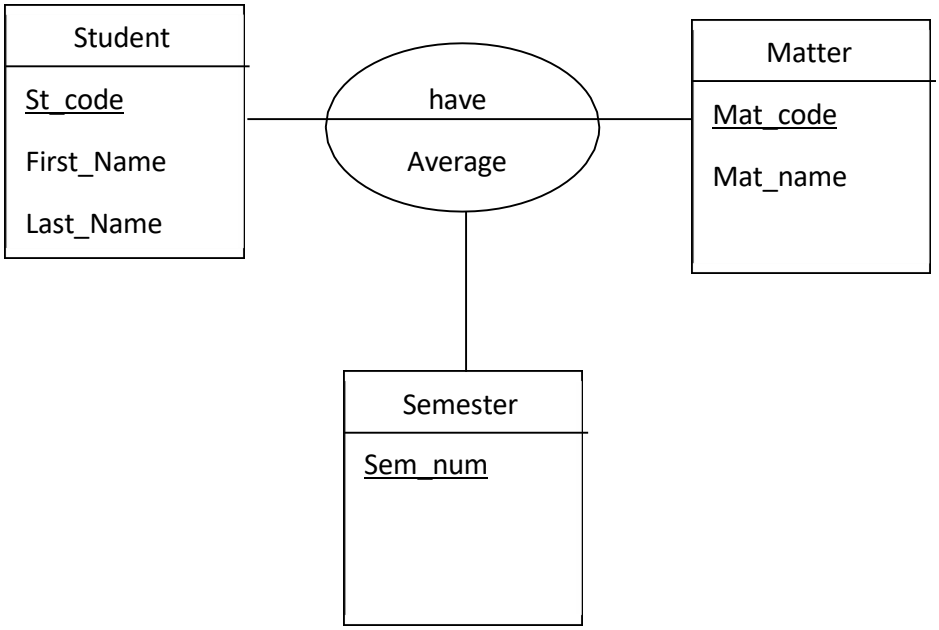
- **Dimension = 2 → Binary Association**

  A binary association connects two entities. This is the most common type of association, where two distinct entities are related to each other in a way that forms a relationship between them.



- **Dimension = 3 → ternary association**

  It connects three entities.



- **Dimension ≥ 4 → N-ary Associations**
  An N-ary association connects four or more entities. These types of associations are rare and their use should generally be avoided. Their existence typically indicates poor design work. Therefore, it is better to decompose them into a set of ternary or binary associations.

Example:

- A complex association involving entities such as **student**, **course**, **professor**, and **semester** in one relationship could be broken down into simpler binary or ternary associations.

D. Cardinality

The cardinalities of an entity measure, when traversing the set of occurrences of that entity, the minimum and maximum participation of an individual (occurrence of the individual) in the association. In other words, the cardinalities of an entity in an association express the number of times an occurrence of this entity can be involved in an occurrence of the association, both at a minimum and a maximum.

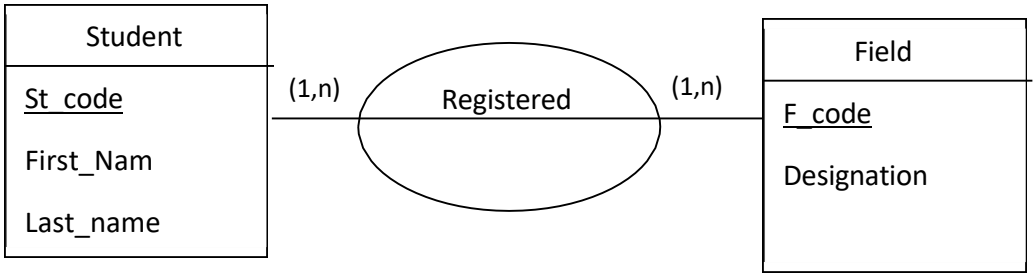
- A **minimum cardinality of "0"** allows some occurrences of the entity to not participate in the association.
- A **minimum cardinality of "1"** requires all occurrences of the entity to participate in the association.
- A **maximum cardinality of "n"** allows some occurrences of the entity to participate multiple times in the association.
- A **maximum cardinality of "1"** forbids all occurrences of the entity from participating more than once in the association.

Important:

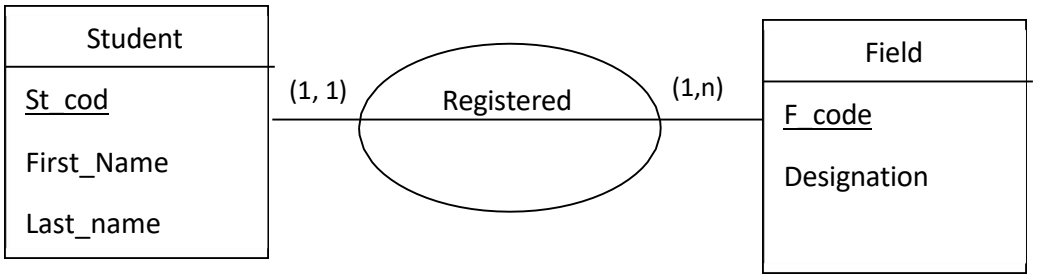
Business rules play a crucial role in determining cardinalities. Therefore, they must be carefully collected and expressed during the preliminary study phase.

Example:

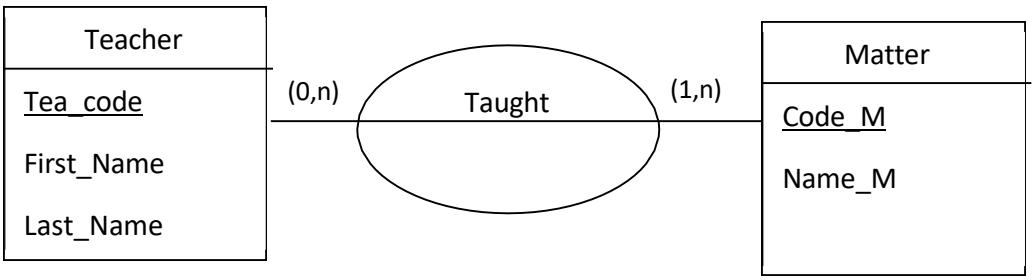
- **Business Rule 1:** A student can enroll in multiple fields of study.



- **Business Rule 2:** A student can enroll in only one field of study.



- **Business Rule 4:** Administrator teachers may be exempt from teaching.



In fact, in the vast majority of cases, only 4 combinations of values are used for cardinalities:

- **0,1**: At most one
- **1,1**: One and only one
- **1, n**: One or more
- **0, n**: Zero or more

Example: Pharmaceutical Distribution Company

Al-Shifa Company is a business specialized in the distribution of pharmaceutical products across different provinces (Wilayas). The company manages the pharmacies registered with it, where each pharmacy is identified by its commercial registration number, name, address, phone number, and email.

The company operates through a group of distribution agents, with each agent responsible for delivering orders in one or more provinces. Each agent is identified by a registration number, first name, last name, phone number, and email. Additionally, each agent declares the distribution vehicle they use, which must be appropriate for the nature of the work (e.g., utility vehicles with air conditioning). The company must record the registration number and type of each vehicle.

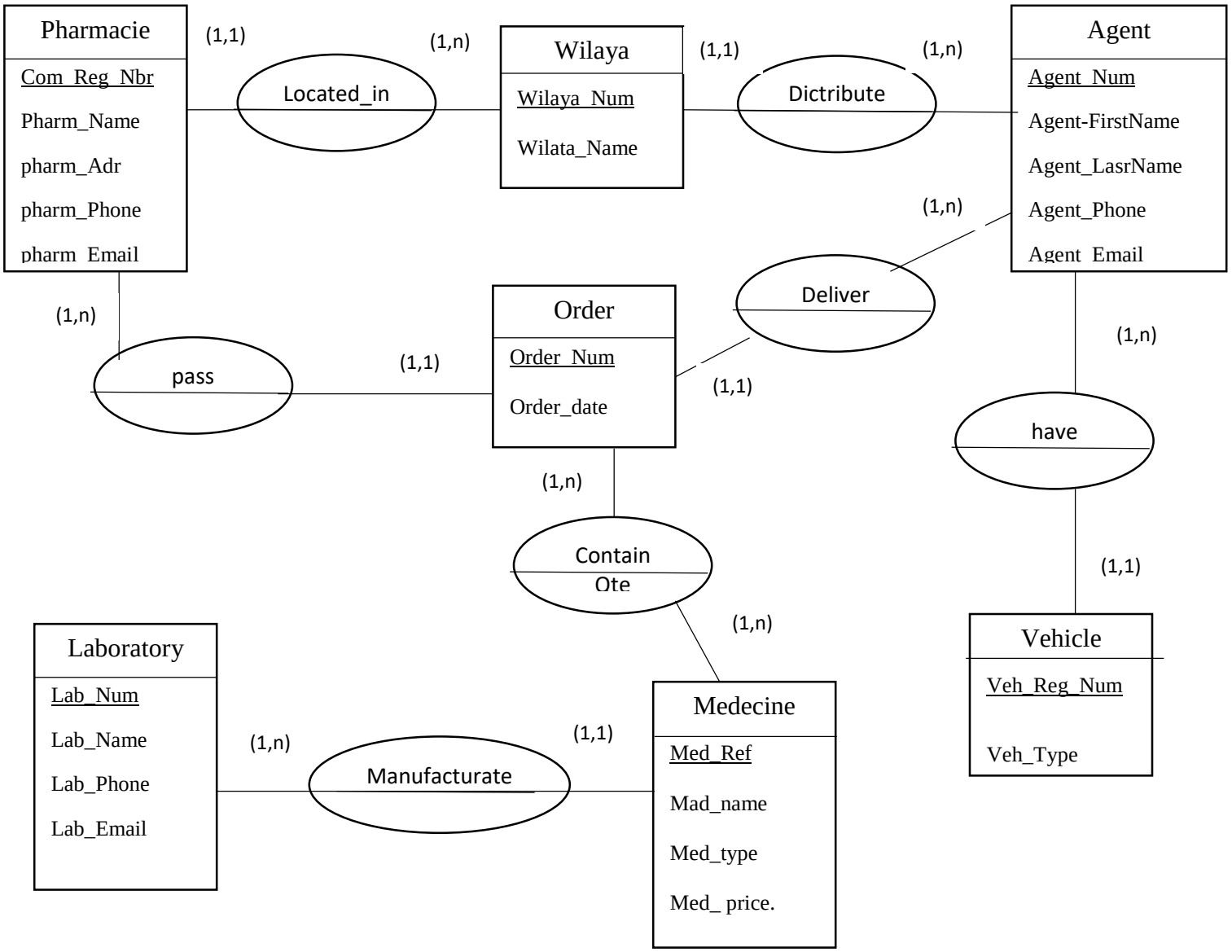
The company collaborates with pharmaceutical manufacturing laboratories, and it must maintain records of each laboratory's number, name, address, phone number, and email. Each laboratory produces several medications, and each medication is manufactured by one and only one laboratory. Each medication is identified by a reference number, name, type, and price.

Pharmacies place their orders with the company, which must keep a record of each order's number and date, along with the list of medications it includes and the desired quantities. Each order is delivered by a single Distribution Agent.

Data dictionary

Code	Meaning	Type	Length	Syntax rules	Semantic rules
Com_Reg_Nbr Pharm_Name pharm_Adr pharm_Phone pharm_Email	Commer register number Pharmacy name Pharmacy address Pharmacy telephone Pharmacy email	N A AN N AN			
Agent_Num Agent-FirstName Agent_LasrName Agent_Phone Agent_Email	Agent number Agent First name Agent Last name Agent Phone Agent Email	N A A N AN			
Veh_Reg_Num Veh_Type	Vehicule registration Vehicule type	N AN			

Wilaya_Num Wilata_Name	Wilaya Number Wilaya Name	N A			
Lab_Num Lab_Name Lab_Phone Lab_Email	Laboratory number Laboratory name Laboratory phone Laboratory Email	N A N AN			
Med_Ref Mad_name Med_type, Med_ price.	Medicine reference Medicine name Medicine type Medicine price	N AN A N			
Order_Num Order_Date Ote	Order number Order Date Quantity	N D N			



7.1.2. Treatments: CMT (Conceptual Model of Treatments)

The goal of the CMT is to represent the dynamic aspect of the domain under study, specifically the activities carried out within the domain. It is also referred to as the “Event-Result Model”: the occurrence of one or more events triggers an operation that produces a result. Its structure is based on the following concepts:

A. Event

A real occurrence that triggers the execution of one or more actions. In other words, events notify the information system that something is happening and require a response. Events can be internal or external to the information system.

Examples:

- Arrival of an order: an event that triggers the execution of the operation related to order processing.
- Receipt of an invoice: an event that triggers the execution of the operation related to invoice payment.

B. Operation

A set of actions executed without interruption, which does not depend on the occurrence of any event other than the initial trigger. An operation produces new events as output.

Example:
The operation *"order preparation"* includes the following uninterrupted actions:

- Identifying missing products and the quantities to order,
- Selecting a supplier,
- Drafting a purchase order.

C. Synchronization Rules

A boolean condition representing the business rules that events must satisfy to trigger an operation. Synchronization rules are the translation of these business rules and define the conditions under which operations are triggered. They are expressed using logical operators (primarily AND and OR).

Example:

- To initiate the creation of an order (operation), one must have:
  Either a stock shortage OR a demand to fulfill (synchronization rule).

D. Emission Rules

Emission rules represent the business rules governing the production of an operation's results.

Example:

- If the order is compliant, then...

Due to their complexity and for better readability, emission rules are generally expressed as OK, not OK, or OK (conditionally).

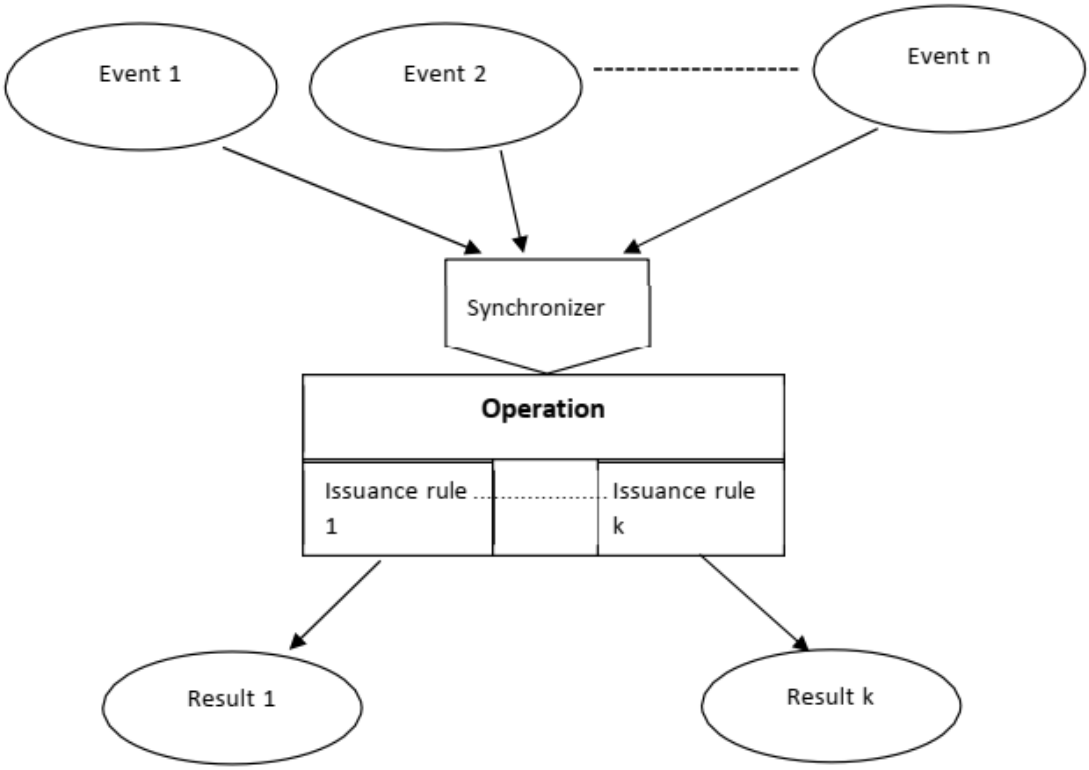
E. Result

The product of executing an operation. A result is also an event-like fact and may serve as the trigger for another operation.

Examples:

- Transmitted order: the result of the operation related to order creation.
- Approved file: the result of the operation related to file verification and validation.

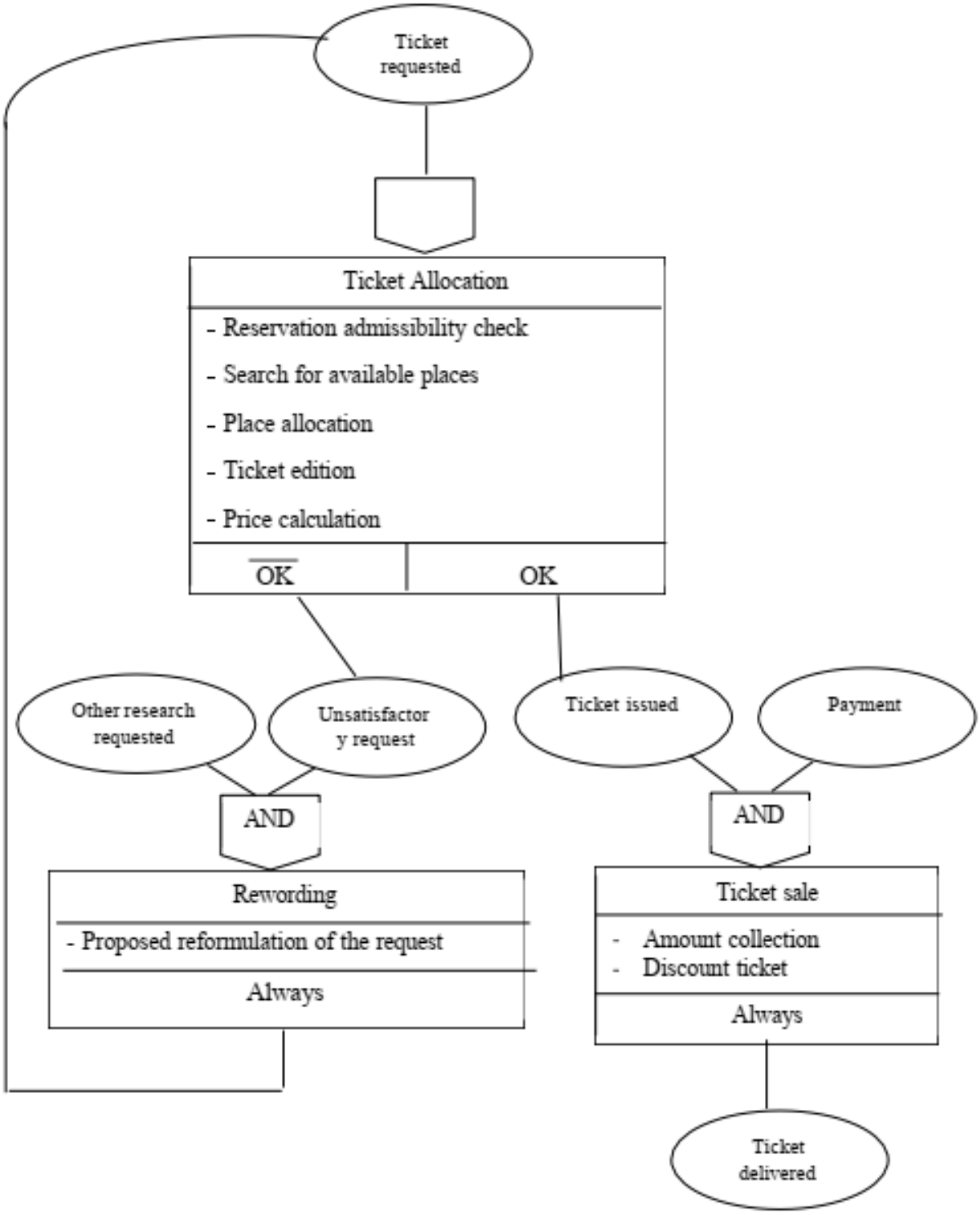
Graphic Representation



Example: Reserving seats in a theatre.

The reservation of seats in a theatre takes place according to the following management rules:

- The reservation counter can issue tickets in advance (reservations) or tickets for immediate entry,
- Place reservations are possible under certain conditions (less than two months in advance, etc.),
- For any seat allocation, a ticket must be issued and a search for available seats carried out,
- Reductions are granted upon presentation of proof (military, students, etc.),
- No ticket can be issued if payment has not been received beforehand,
- For immediate entries, tickets are issued without precise allocation of a place


Ticket requested
Ticket Allocation
- Reservation admissibility check
- Search for available places
- Place allocation
- Ticket edition
- Price calculation
OK
OK
Other research requested
Unsatisfactor y request
Ticket issued
Payment
AND
AND
Rewording
- Proposed reformulation of the request
Always
Ticket sale
- Amount collection
- Discount ticket
Always
Ticket delivered

7.2. ORGANIZATIONAL LEVEL

7.2.1. Processes: Organizational Model of Processes (OMP)

The OMP involves making technical and organizational choices for software implementation. At this level, issues related to the following are addressed:

- Hardware,
- Workstations,
- Temporal organization of processes to be performed,
- Financial and human resources.

In other words, it aims to answer the questions: Where, Who, When.

- **Where:** Concerned workstation and hardware.
- **Who:** Human and financial resources, whether the process is automated or not.
- **When:** Timing of the processes to be executed.

The **how** is not addressed at this stage.
In summary: **OMP = CMP + Location + Time + Nature**

A. Event

Concept and formalism are identical to those in the CMP: A real occurrence that triggers the execution of one or more tasks.

B. Organizational Rule

A statement of the organization in place in terms of workstations, the nature of the process, and its chronology.

C. Synchronization

Concept and formalism are identical to those in the CMP: A boolean condition representing the business rules that events must satisfy to trigger tasks.

D. Functional Procedure (FP)

A set of actions executed without interruption (based on the organization in place), not conditioned by the occurrence of any event other than the initial trigger.
For all tasks within a single FP, the workstation, the nature of the process, and its timing remain consistent.

Note: An operation at the CMP level = Σ FPs at the OMP level (at least one).

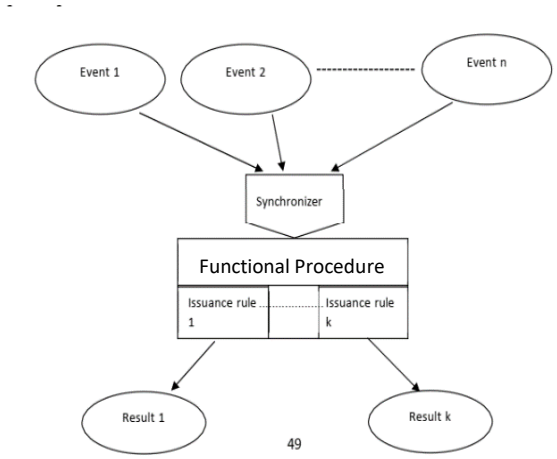
E. Emission Rule

Concept and formalism are identical to those in the CMP: A condition representing the business and organizational rules governing the production of an FP's results.

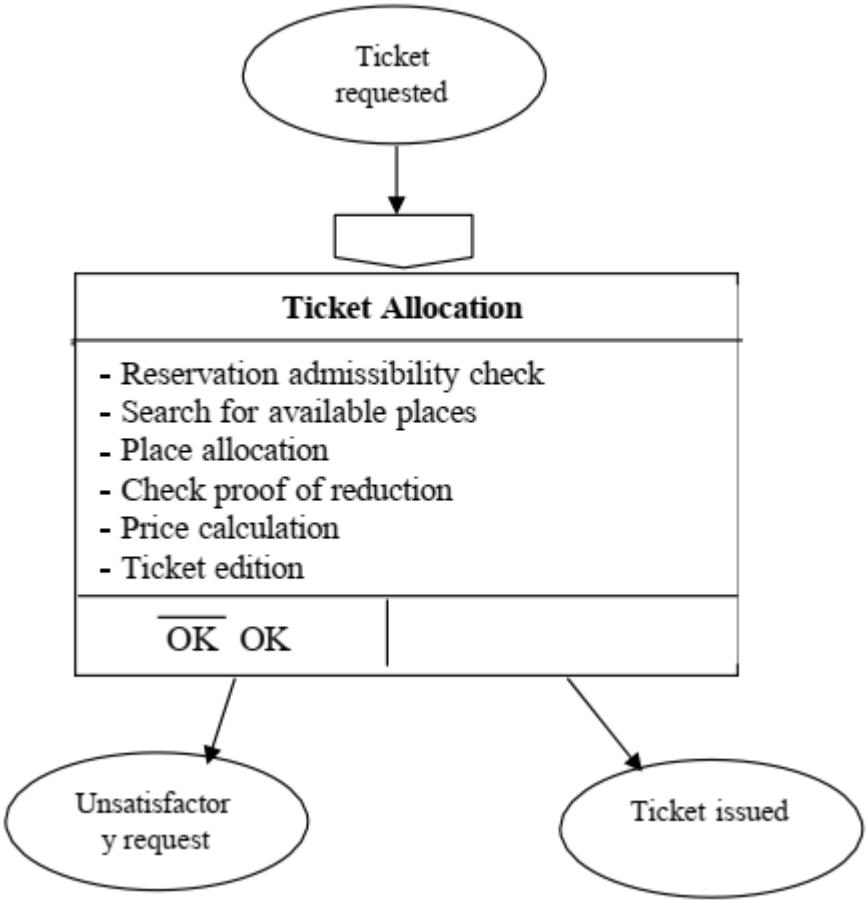
F. Result

Concept and formalism are identical to those in the CMP: The product of executing an FP. A result, being an event-like fact, may trigger another FP.

Graphical Representation

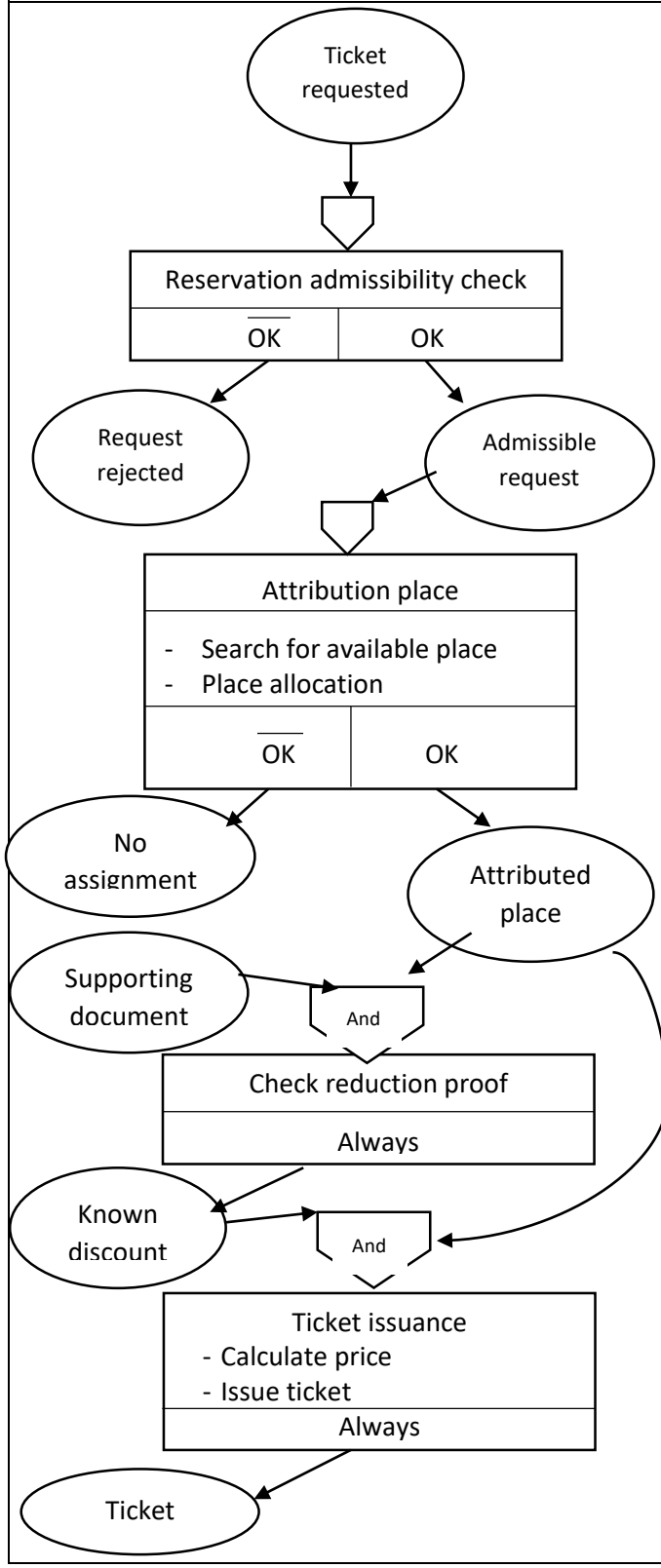
Functional sequence of procedures	***Nature of treatment***	***Workplace***	***Chronological sequence***
	Nature of the phase processing (manual, automatic)	Name of the workstation executing the phase	Phase execution period

Example: Taking the ticket allocation procedure from the previous example.



This procedure contains several actions that differ organizationally:

Action	**Workplace**	**Nature of treatment**	**Timing/Timeline**
Reservation admissibility check	Public counter	Manual (conversational)	As requests progress
Search for available places	Public counter	Automatic	As requests progress
Seat assignment	Public counter	Automatic	As requests progress
Control of proof of reduction	Public counter	Manual (conversational)	As requests progress
Price calculation	Public counter	Automatic	As requests progress
Ticket edition	Public counter	Automatic	As requests progress

Functional sequence of procedures	*Nature of treatment*	*Workplace*	*Chronological sequence*
	Manual Automatic Manual Automatic	Public counter Public counter Public counter Public counter	As request progress As request progress As request progress As request progress

7.2.2. Data: Logical Data Model (LDM)

The **CDM (Conceptual Data Model)** represents the static part of the future Information System (IS). It is expressed in a format understandable by designers rather than machines. Therefore, it must be transformed into a format that can be implemented using computer techniques. This transformation is known as the logical level, and it involves converting the CDM into a relational model by applying certain transition rules. Consequently, the following abbreviations are commonly used:

- **LDM**: Logical Data Model
- Sometimes, the following abbreviations are also used:
    - **LDRM**: Logical Data Model Relational
    - **RDM**: Relational Data Model
    - **LRDM**: Logical Relational Data Model

A. Transition Rules from CDM to a Relational LDM

Rule 1: Properties

Each property becomes an attribute of the relational model.

Rule 2: Entities

Each entity becomes a relation (at least in third normal form) in the relational model. The identifier of the entity becomes the primary key of the corresponding relation.

Rule 3: Association

- **Binary Father-Son Association**

  A binary father-son association is characterized by:
    - Dimension of 2
    - Cardinality of the entities participating in the association:
        - Father (0, N) or (1, N)
        - Son (0, 1) or (1, 1)

  This transforms as follows:

- The association disappears.
- The father entity becomes the father relation.
- The son entity becomes the son relation.
- The identifier of the father entity becomes an attribute of the son relation and is called a **foreign key**.
- The properties of the association become attributes in the son relation.

Example:

The resulting **LDM** is as follows:

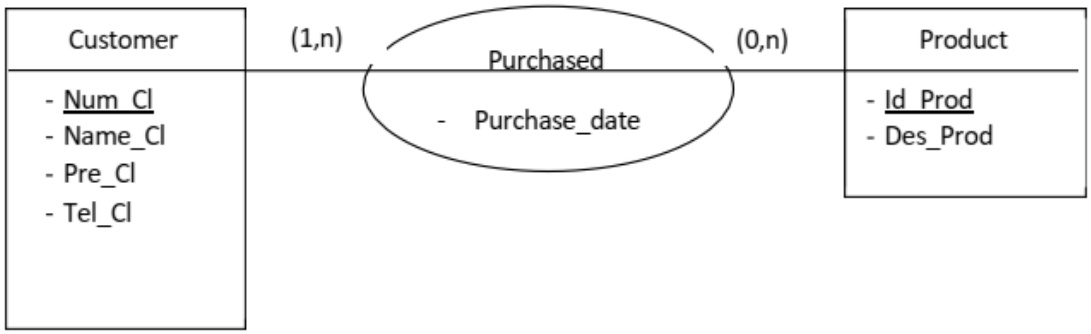
Étudiant (***Num-Etud***, Nom_Etud, Pre_Etud, Adr_Etud, ***Id_Fil****, Anne_Isc)
Filière (***Id_Fil***, Des_Fil)

Non Father-Son Binary Association

For an association with cardinality (0, N) or (1, N) for the participating entities, the transformation follows this approach:

- The entities are always converted into relations.
- The identifier of each entity becomes the primary key of the corresponding relation.
- The association itself becomes a relation with its own attributes, if any.
- The primary key of the association is the concatenation of the primary keys of the entities forming the association.

Example:



The resulting **LDM** is as follows:

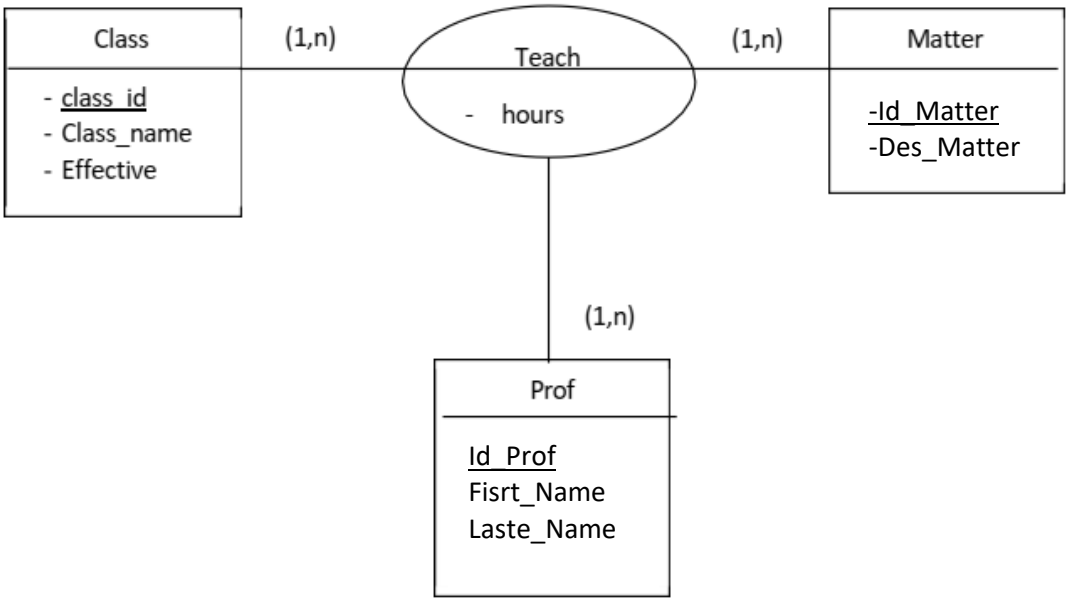
Client (Num_Cl, Nom_Cl, Pre_Cl, Tel_Cl)
Produit (Id_Prod, Des_Prod)
Achete (Num_Cl, Id_Prod, Date_achat)

N-ary Associations

This refers to associations with a dimensionality of 3 or more. They transform as follows:

- The entities involved always become relations.
- The identifier of each entity becomes the primary key of the corresponding relation.
- The association itself becomes a relation with its own attributes, if any.
- The primary key of the association is the concatenation of the primary keys of the entities forming the association.

Example:



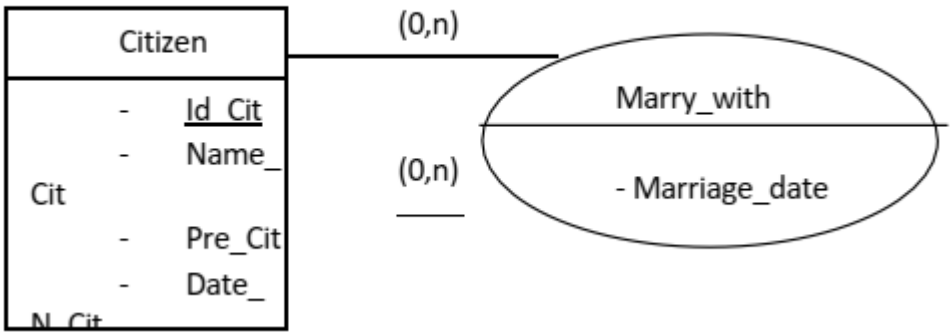
Class (***class_id***, class_name, Effective)

Matter (***Id_Matter***, Des_Matter)

Prof (Id_Prof, First_name, Laste_Name)

Teach (**Id_class, Id_Matter, Id_Prof,** hours)

- ***Reflexive association***



Citizen (**Id_Citizen**, First_Name, Laste_Name, Birth_date)

Marry_with (**Id_Cit_source, Id_Cit_target**, Date_marriage)

7.2.2. Data: Logical Data Model (LDM)

The **CDM (Conceptual Data Model)** represents the static part of the future Information System (IS). It is expressed in a format understandable by designers rather than machines. Therefore, it must be transformed into a format that can be implemented using computer techniques. This transformation is known as the logical level, and it involves converting the CDM into a relational model by applying certain transition rules. Consequently, the following abbreviations are commonly used:

- **LDM**: Logical Data Model
- Sometimes, the following abbreviations are also used:
    - **LDRM**: Logical Data Model Relational
    - **RDM**: Relational Data Model
    - **LRDM**: Logical Relational Data Model

A. Transition Rules from CDM to a Relational LDM

Rule 1: Properties

Each property becomes an attribute of the relational model.

Rule 2: Entities

Each entity becomes a relation (at least in third normal form) in the relational model. The identifier of the entity becomes the primary key of the corresponding relation.

Rule 3: Association

- **Binary Father-Son Association**

  A binary father-son association is characterized by:
    - Dimension of 2
    - Cardinality of the entities participating in the association:
        - Father (0, N) or (1, N)
        - Son (0, 1) or (1, 1)

  This transforms as follows:

- The association disappears.
- The father entity becomes the father relation.
- The son entity becomes the son relation.
- The identifier of the father entity becomes an attribute of the son relation and is called a **foreign key**.
- The properties of the association become attributes in the son relation.

Example:

The resulting **LDM** is as follows:

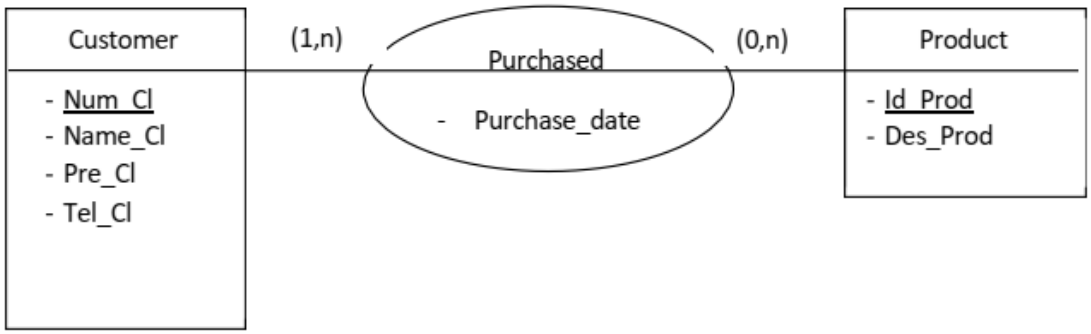
Étudiant (***Num-Etud***, Nom_Etud, Pre_Etud, Adr_Etud, ***Id_Fil****, Anne_Isc)
Filière (***Id_Fil***, Des_Fil)

Non Father-Son Binary Association

For an association with cardinality (0, N) or (1, N) for the participating entities, the transformation follows this approach:

- The entities are always converted into relations.
- The identifier of each entity becomes the primary key of the corresponding relation.
- The association itself becomes a relation with its own attributes, if any.
- The primary key of the association is the concatenation of the primary keys of the entities forming the association.

Example:



The resulting **LDM** is as follows:

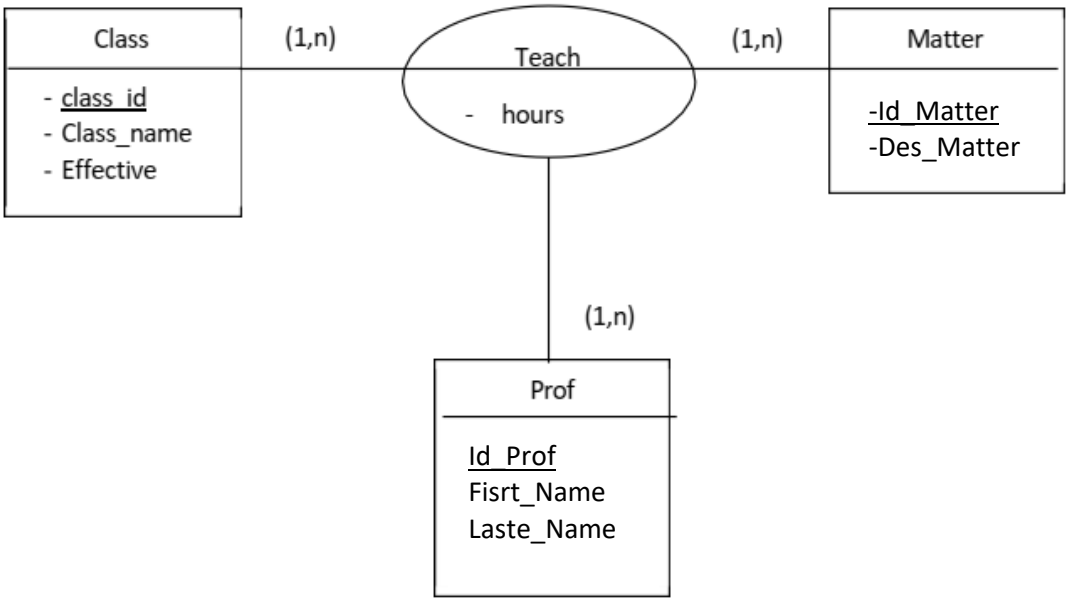
Client (Num_Cl, Nom_Cl, Pre_Cl, Tel_Cl)
Produit (Id_Prod, Des_Prod)
Achete (Num_Cl, Id_Prod, Date_achat)

N-ary Associations

This refers to associations with a dimensionality of 3 or more. They transform as follows:

- The entities involved always become relations.
- The identifier of each entity becomes the primary key of the corresponding relation.
- The association itself becomes a relation with its own attributes, if any.
- The primary key of the association is the concatenation of the primary keys of the entities forming the association.

Example:



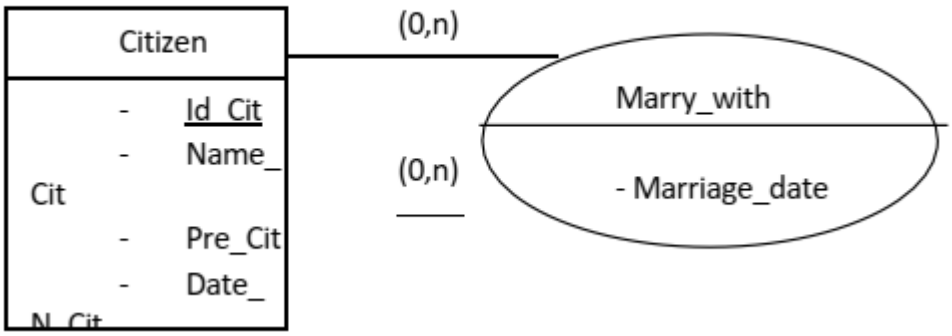
Class (***class_id***, class_name, Effective)

Matter (***Id_Matter***, Des_Matter)

Prof (Id_Prof, First_name, Laste_Name)

Teach (**Id_class, Id_Matter, Id_Prof,** hours)

- ***Reflexive association***



Citizen (**Id_Citizen**, First_Name, Laste_Name, Birth_date)

Marry_with (**Id_Cit_source, Id_Cit_target**, Date_marriage)

7.2.3. Normalization of the Relational Model

Normalization is the process of organizing data within a database to ensure:

- Non-redundancy of data
- Data integrity
- Ease of updating

When normalizing a data model, it is recommended to achieve at least the Third Normal Form (3NF).

A. First Normal Form (1NF)

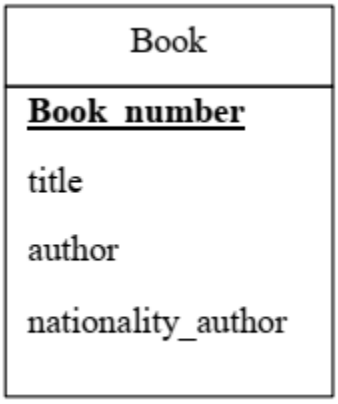
A relation is in First Normal Form (1NF) if and only if it contains only simple, elementary, or atomic values (i.e., no structured or repetitive data).

- If a table contains groups of repetitive data, these groups must be separated into new entities that store the repetitive data.
- If a column in a table contains structured data, it must be decomposed into multiple columns, each containing one element of the structured data.

Example:

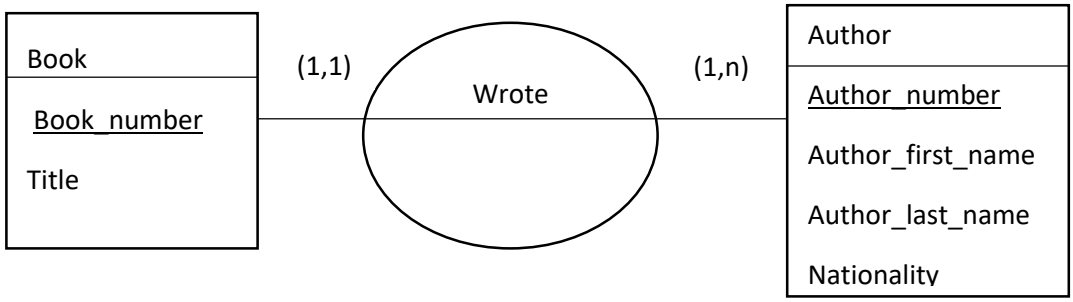
Consider the relation:

BOOKS(Book_Num, Title, Author, Author_Nationality)



Book_number	Title	Author	Nationality_author
01	EZ-ZILZEL (THE EARTHQUAKE)	Tahar Ouettar	Algerian
02	THE MARTYRS RETURN THIS WEEK	Tahar Ouettar	Algerian
03	THE ACE	Tahar Ouettar	Algerian
04	WAR AND PEACE	Leo Tolstoy	Russian
05	THE DEVIL	Leo Tolstoy	Russian

This table is not in 1NF, therefore it must be broken down as follows:



Book (Book_Num, Title, Author_Num*)

Author (Author_number, Author_first_name, Author_last_name, Nationality)

Book_number	Title	Author_number
1	EZ-ZILZEL (THEEARTHQUAKE)	1
2	THE MARTYRSRETURN THIS WEEK	1
3	THE ACE	1
4	WAR AND PEACE	2
5	THE DEVIL	2

Author_number	Author_fisr_name	Author_last_name	Nationality
1	Ouettar	Tahar	Algerian
2	Tolstoy	Leon	Russian

B. Second Normal Form (2NF)
A table or relation is in Second Normal Form (2NF) if it meets the following conditions:

1. It is in First Normal Form (1NF).
2. All non-key attributes are fully functionally dependent on the entire primary key (in the case of a composite primary key) and not on just a part of it.

Note:
The Second Normal Form applies only to tables that have a composite primary key. If a table has a single-column primary key, it automatically satisfies the condition of 2NF as there is no partial dependency.

Example:
Consider the relation:
OPERATION (Account_num*, Code_Ope*, Date_Ope*, Name, First_name, Libe_Ope, Sum)

Account_num	Code_Ope	Date_Ope	Name	First_name	Lib_Ope	Sum
125978	30	05/12/ 2020	Ali	Ahmed	Debit	15000
125978	40	09/15/ 2020	Ali	Ahmed	Credit	45000
125978	30	05/11/ 2021	Ali	Ahmed	Debit	30000
125978	40	07/09/ 2021	Ali	Ahmed	Credit	25000
151936	30	05/10/ 2020	Tarek	Salim	Debit	35000
151936	40	12/11/ 2021	Tarek	Salim	Credit	40000

If the attributes **Name** and **Firstname** depend only on **Num_Account** (and not on the entire composite key **(Num_Account, CodeOpe, DateOpe)**), the table is not in 2NF.

To normalize:

1. Remove the partial dependency by creating a new table to store **Num_Account**, **Name**, and **Firstname**.
2. Maintain the remaining attributes in the original table, ensuring they are fully dependent on the composite primary key.

Note:

- **Name** and **First Name** functionally depend on **Account_Number**.
- **Operation Label** functionally depends on **Operation Code**.

Correction:

To resolve this situation, we must decompose the relationship into three separate relations:

1. **ACCOUNT(Account_Number, Last_Name, First_Name)**
    - Stores account information, including the account number and the owner's name.
2. **LABEL (Ope_Code, Ope_Libel)**
    - Stores the mapping of operation codes to their corresponding labels.
3. **OPERATION**(Account_Number, Ope_Date, Ope_Code*, Sum)**
    - Represents individual operations, linking accounts, operation codes, and amounts.

Resulting Tables:

Account_Number	Last_Name	First_Name
125978	Ahmed	Ali
151936	Salim	Tarek

Ope_Code	Ope_Libel
30	Speed
40	Credit

Account_Number	Ope_Date	Ope_Code	Sum
125978	05/12/2020	30	15,000
125978	09/15/2020	40	45,000
125978	05/11/2021	30	30,000
125978	07/09/2021	40	25,000
151936	05/10/2020	30	35,000
151936	12/11/2021	40	40,000

C. 3rd Normal Form (3NF)

A relation is in **Third Normal Form (3NF)** if it satisfies the following conditions:

1. It is in **Second Normal Form (2NF)**.
2. Additionally, no non-key attribute is functionally dependent on any other non-key attribute.

Example:
Consider the relationship:

Flight (Flight_Num, Departure_City, Arrival_City, Flight_Date, Flight_Time, Plane_Name, Manufacturer, Number_of_Seats, Speed)

In this relation:

- **Manufacturer**, **Number_of_Seats**, and **Speed** are functionally dependent on **Plane_Name**, which is a non-key attribute.
- This violates 3NF because non-key attributes are functionally dependent on another non-key attribute (**Plane_Name**).

Correction:
To normalize the relation into 3NF, we decompose it into two relations:

1. **Flight (Flight_Num, Departure_City, Arrival_City, Flight_Date, Flight_Time, Plane_Name)**
    - This stores flight-related information and references the plane by its name.
2. **Airplane (Plane_Name, Manufacturer, Number_of_Seats, Speed)**
    - This stores information specific to the airplane.

7.3. Physical or Operational Level

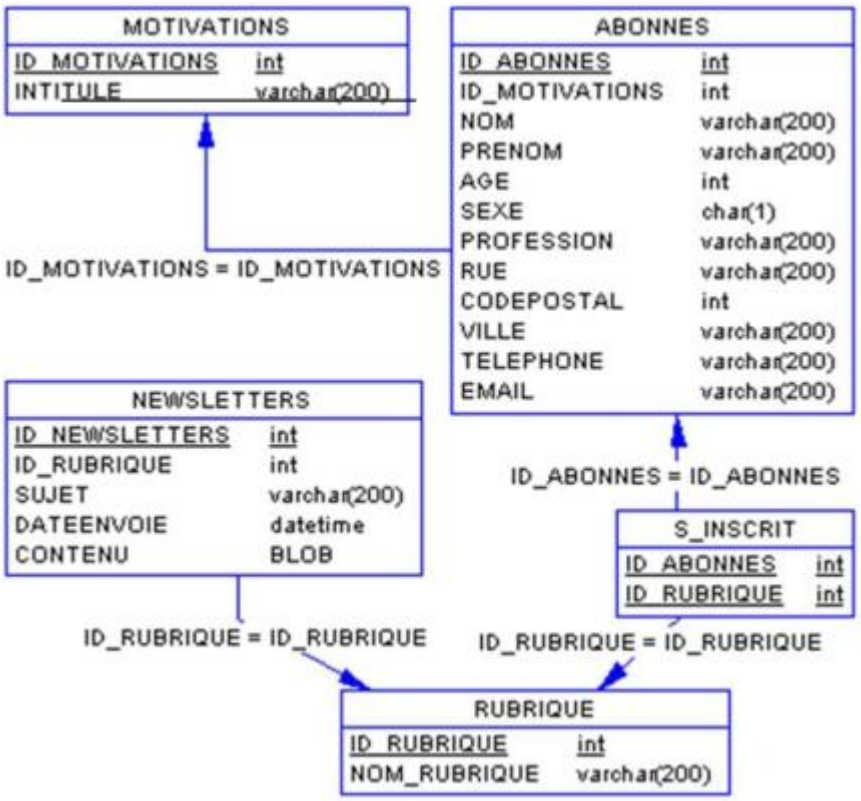
7.3.1. Data: Physical Data Model (PDM)

The **Physical Data Model (PDM)** is used to define the final structure of the database, including the various relationships between its components. It represents the organization of data while considering the specific **Database Management System (DBMS)** selected.

Typically, **Relational Database Management Systems (RDBMS)** are chosen. The result is a set of tables, each with columns that have specific data types based on the DBMS used and as declared in the **Data Dictionary**.

The language primarily used for these operations is **SQL**. Alternatively, specialized tools like **PowerAMC**, **WinDesign**, or other **Automated Graphic Tools (AGLs)** can be employed to automatically generate the database schema.

Example:



7.3.2. Treatments: Physical Model of Treatment (MPT)

The **Physical Model of Treatment (MPT)**, also referred to as the **Operational Model of Treatment (OMT)**, focuses on writing the program code for implementing the database operations. This can be automatically generated within the context of a **Software Engineering Workshop (SEW)**.

The goal of methodologies like **MERISE** is to facilitate the automatic production of code from the design phase. The MPT focuses on the internal structure of all project-related applications and prepares for development by breaking down each operation into technical modules.

Steps in the MPT process include:

- **Defining internal data:** Specify the internal data structures required by the technical module.
- **Defining module processing:** Outline the procedures or functions associated with the module.
- **Defining algorithms or pseudo-code:** Create algorithms or pseudo-code to detail the logic of operations.
- **Presenting technical processing:** Provide a clear presentation of technical workflows.
- **Specifying input information:** Identify the inputs needed for the module.
- **Specifying output information:** Define the expected outputs of the module.
- **Providing results:** Describe the final results or outcomes produced by the technical module.

This process ensures a structured and efficient approach to implementing database operations in alignment with the design specifications.