Djillali Liabes University – Sidi Bel Abbes

Faculty of Economic Sciences, Commerce, and Management Sciences

Department: Financial Sciences and Accounting

Course Pack Titled:

INFORMATICS II

Addressed to : 2nd year Bachelor's Degree – Finance and Accounting - First Semester

Prepared by: Mrs. Sara BENTOUILA

Department of Financial Sciences and Accounting

Academic Year: 2025–2026

Contents

List of Figures

List of Tables

General Introduction

In the digital age, the ability to collect, structure, query, and present information has become a fundamental skill, extending far beyond the field of computer science alone. For future professionals in economics, management, and business, data mastery is an essential lever for performance, enabling informed decision-making, process optimization, and the discovery of new opportunities.

This course material, designed for the "Computer Science 2" module, aims to guide you step-by-step in acquiring this key skill. It is not simply about learning to use a piece of software, but about understanding the logic that underlies the creation of a complete management application. To do this, we will rely on two complementary pillars:

1. A rigorous design method, Merise, which will teach us how to analyze a need and model an information system in a structured and professional manner.
2. A powerful and accessible tool, Microsoft Access, which will serve as our platform to bring this design to life and build a functional database.

The objective of this course is to make you autonomous in creating a simple but complete information system. Starting from the theoretical foundations of data modeling, we will cover the construction of the essential building blocks of any application:

- **Tables**, to store data in an organized way.
- **Queries**, to extract relevant information and answer specific questions.
- **Forms**, to provide a user-friendly and secure data entry interface.
- **Reports**, to present results in the form of professional summaries.

We will conclude our journey with an introduction to automation using VBA (Visual Basic for Applications), which opens the door to more dynamic and intelligent applications.

By the end of this module, you will no longer see data as a mere accumulation of numbers, but as a strategic resource that you know how to model, manipulate, and leverage. The skills acquired here will be a major asset in your future academic and professional path.

Chapter 1

Introduction to Information Systems and Design Methods

Contents

1.1 Historical Overview of the Evolution of Information Systems (IS)

The history of information systems (IS) is inseparable from the history of computing. It reflects a continuous quest to streamline and automate information processing, thereby meeting constantly changing organizational and economic needs. The evolution of IS can be outlined in four major phases, each defined by technological disruptions and new uses[Ted14, Vol].

1.1.1 Phase 1: The Era of Mainframes and Centralized Processing (1950s–1960s)

The first business computers, which appeared in the late 1950s, were massive machines called mainframes. Reserved for large organizations due to their prohibitive cost, they required specialist teams and dedicated infrastructure[Vol, Mur22].

Technology

Information processing relied on physical media such as punched cards and magnetic tapes. Interaction with the machine was minimal and occurred offline (batch processing). The end of this period saw the emergence of the time-sharing concept, allowing multiple users to access the mainframe's resources simultaneously.

Main Objective

The automation of heavy and repetitive administrative tasks. The goal was to increase productivity by speeding up calculations.

Typical Applications

Payroll management, general accounting, and inventory management.

Access and Use

Only large corporations and government agencies could afford these expensive systems, which required teams of specialists to operate. User interaction was almost non-existent. However, **Time-Sharing** technology allowed multiple users to access a central computer simultaneously.

1.1.2 Phase 2: The Advent of Databases and Early Networks (1970s)

This decade marks a major turning point. Computing was no longer just a calculation tool but became a means to store, organize, and retrieve information to support decision-making[Ted14].

Technology

This was the birth of Database Management Systems (DBMS), which allowed data to be structured coherently. In 1970, Edgar Frank Codd published his foundational paper on the relational model, which would become the industry standard. At the same time, the first computer networks, such as ARPANET (the forerunner of the Internet), began to emerge.

Main Objective

To centralize and share data within the organization to avoid redundancy and ensure consistency.

Typical Applications

Management applications became more sophisticated, incorporating real-time query and update functions.

1.1.3 Phase 3: The Microcomputer Revolution (1980s)

The arrival of the personal computer (PC), with models like the IBM PC in 1981 and the Apple Macintosh in 1984, democratized access to computing[Ted14, tea24].

Technology

The miniaturization of components (microprocessors) made computers more accessible and easier to use thanks to graphical user interfaces and the use of the mouse. PCs connected to each other via Local Area Networks (LANs), which facilitated the sharing of resources (files, printers).

Main Objective

Individual and departmental computing. Each employee could have their own workstation for office tasks (word processing, spreadsheets). IS became genuine tools for personal and group productivity.

Typical Applications

Office suites, PC-based accounting software, and the first client-server applications.

1.1.4 Phase 4: The Era of the Internet and Distributed Computing (1990s to Present)

This period is marked by global interconnection and an explosion in information exchange[Ted14, tea24].

Technology

The development of the World Wide Web by Tim Berners-Lee in the early 1990s made the Internet accessible to the general public. The TCP/IP protocols became the universal standard for communication. The client-server architecture became widespread, later evolving into distributed architectures, cloud computing, and mobile applications.

Main Objective

Borderless communication and collaboration. The IS extended beyond the company's walls to interact with customers, suppliers, and partners. It became a major strategic advantage.

Typical Applications

Websites, e-commerce, Enterprise Resource Planning (ERP) systems, corporate social networks, Big Data, and artificial intelligence.

1.2 Introduction to the Merise Design Method

After exploring the historical evolution of information systems, it has become clear that a structured approach is essential for managing the growing complexity of IT projects. The Merise method, developed in France in the early 1980s, is one of the most comprehensive responses to this need for formalism and rigor[Avi91].

1.2.1 Definition and Objectives

Definition

Merise is a method for the analysis, design, and development of information systems. Its founding principle is the clear and systematic separation of data and processes. This distinction allows for the design of a stable and scalable information system, as an organization's data structure is generally more durable than the processes that are applied to it[RT83, Avi91].

In other words, Merise proposes to first answer the questions:

- **"What?"** : What data must the system store? (Static aspect)
- **"Who, When, How?"** : What are the processes and actions that manipulate this data? (Dynamic aspect)

Objectives

Adopting a method like Merise aims to achieve several strategic objectives to ensure the success of an IT project:

- **Master complexity** : To break down a complex problem into simpler sub-problems that are easier to analyze and solve.
- **Ensure consistency** : To guarantee that data and processes are designed in a logical and integrated manner, thereby avoiding redundancy and inconsistencies.
- **Facilitate communication** : o provide a common language (through standardized graphical models) among the different project stakeholders: end-users (domain experts), analysts, and developers (technical experts).
- **Document the system** : To produce complete and structured documentation at each stage of the project, which is essential for the future maintenance and evolution of the system.
- **Promote scalability** : Thanks to the separation of data and processes, it is easier to modify processes without having to completely restructure the database, and vice versa.

1.2.2 The Three Cycles of Merise

To organize the design process in a rigorous manner, the Merise approach is structured around three axes, or "cycles" (as shown in the Figures 1.1 and 1.2), which are followed simultaneously throughout the project. These cycles help to prioritize questions and ensure that nothing is left to chance[Bap09].

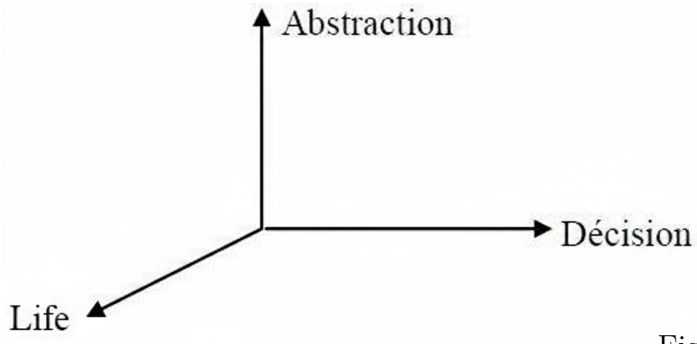


Figure 1.1: The 3 Cycles of Merise[Ben]

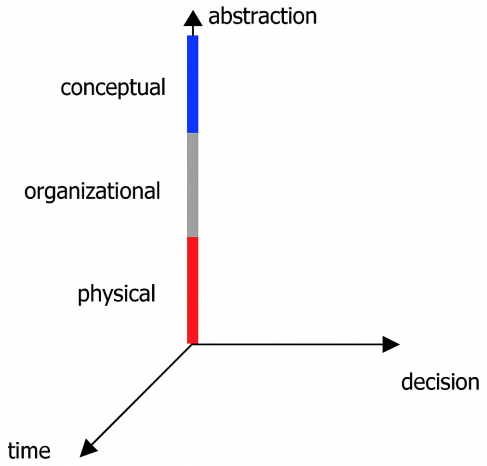


Figure 1.2: Axes of decision complexity in time and abstraction[Dar]

The Abstraction Cycle

This cycle focuses on how to specify an Information System (IS)[Sbi05, Moh13]:

- **Data memory** : is described at conceptual, logical, and physical levels.
- **Process operations** : are described conceptually, organizationally, and operationally.

Each layer is represented through a model. Changes in lower layers don't affect higher layers unless they directly impact the parameters of the layer in question. Each model is defined using formalized rules, principles, vocabulary, and syntax. Transition rules allow for automatic or semi-automatic movement from one model to another (as shown in the Figure 1.3).

The Life Cycle

This cycle describes the temporal progression of the project, from its initial idea to the end of its life. It is a classic sequential approach to project management[Sbi05, Moh13].

As shown in the Figure 1.4, the main phases of the life cycle are :

- **Design** : This phase includes the study of the existing system, analysis of needs, and detailed specification of the future system (using the levels of the abstraction cycle).
- **Implementation** : This corresponds to software development (programming), testing, and deployment of the solution.
- **Maintenance** : Once the system is in production, this phase involves its evolution, correcting anomalies, and adapting it to new needs.

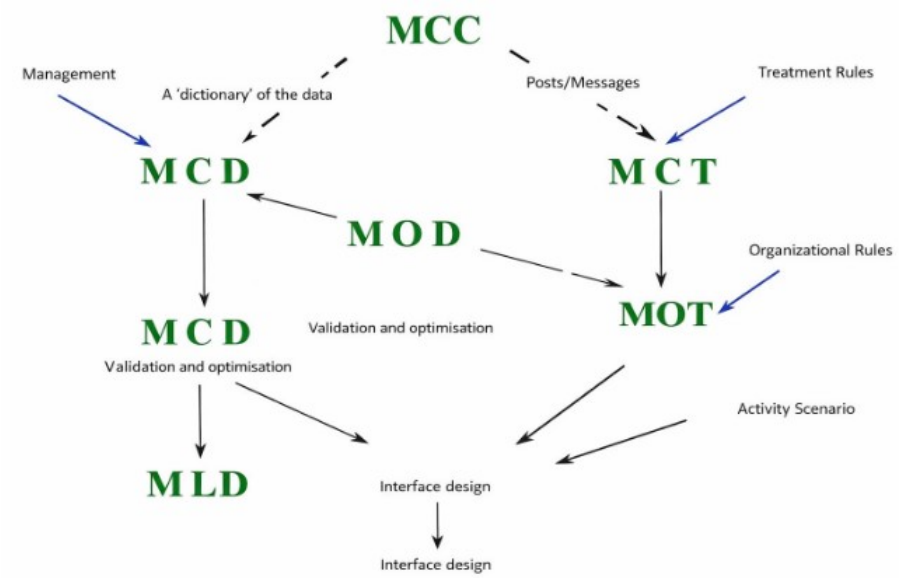


Figure 1.3: The abstraction cycle of Merise[Cite]

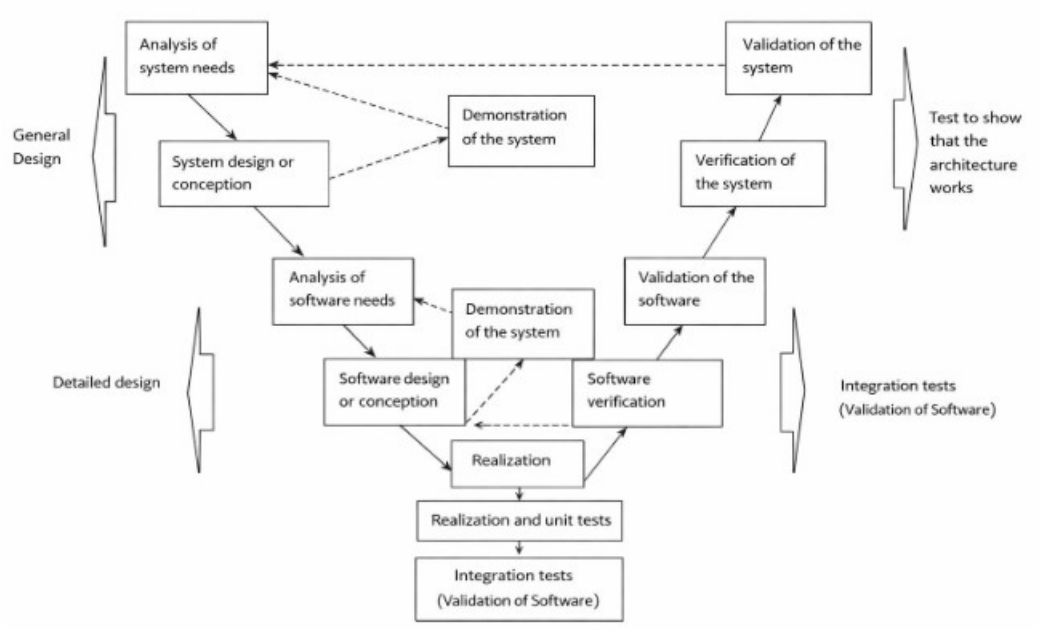


Figure 1.4: The life cycle of Merise[Cite]

The Decision Cycle

This cycle is transversal to the other two. It formalizes the project management process by identifying all the choices and validations required at each stage[Sbi05, Moh13]

Decisions are hierarchical:

- **Strategic Decisions** : Made by top management, such as launching the project or allocating the budget (Go/No Go).
- **Tactical Decisions** : Related to organizational choices or major technical options.

- **Operational Decisions** : Concern the details of implementation, such as the validation of a screen or a report by future users.

Each decision must be documented, thus ensuring the traceability and coherence of the entire project. In short, Merise structures the design of an IS by answering the questions: **WHAT? WHO? WHEN? HOW?** in a progressive, controlled, and validated manner.

1.2.3 The Stages of MERISE

The design and development process of a system in MERISE is divided into several stages[TRC89] (Figure 1.5)

- **Master Plan** : Defines the overall strategic approach and objectives for the system.
- **Preliminary Study** : Conducts an initial analysis to assess the current situation and system needs.
- **Detailed Study** : Provides an in-depth examination of system requirements and specifications.
- **Technical Study** : Outlines the technical aspects required for implementation, including hardware and software specifications.
- **Software Production** : Involves developing the necessary software components according to the design specifications.
- **Implementation** : Encompasses the deployment and operationalization of the system.

Each stage plays a crucial role in building a coherent system aligned with the organization's goals and resources.

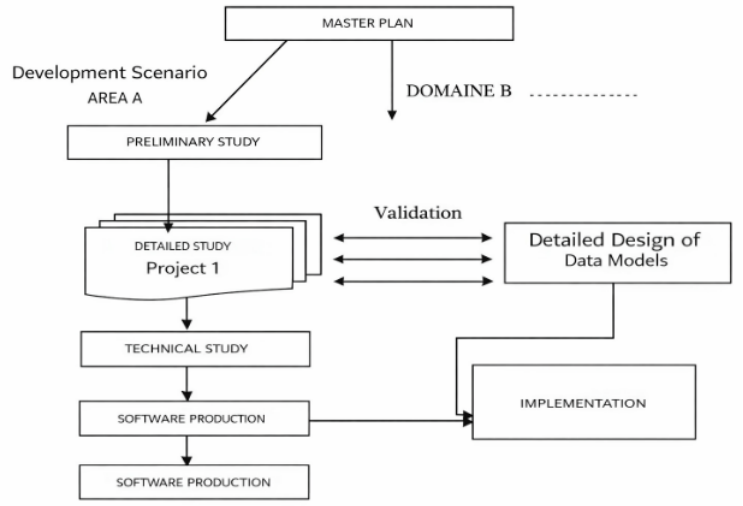


Figure 1.5: The Stages of MERISE[Ben]

1.2.4 A Multi-Level Approach

MERISE addresses various challenges at different levels, acknowledging issues like hardware or software changes, new regulations, and other contextual shifts. This results in three primary levels, each with distinct concerns[TRC89]:

- **Conceptual Level** : Defines the enterprise's main objectives and management rules, reflecting the organization's goals and constraints. This level is generally the most stable and includes functions such as personnel management, accounting, etc.
- **Logical Level** : Specifies the organizational structure required to meet objectives, including job roles, operational sequences, and the nature of various processes.
- **Technical Level** : Details the technical resources (hardware/software) necessary to implement the project. This level is subject to frequent changes.

Data and Process Models by Level

Each level consists of data and process models:

- **Conceptual Level** : Conceptual Data Model (CDM/MCD) and Conceptual Process Model (CPM/MCT).
- **Logical Level** : Logical Data Model (LDM/MLD) and logical Process Model (LPM/MLT).
- **Technical (Physical) Level** : Physical Data Model (PDM/MPD) and physical Process Model (PPM/MPT).

As shown in the Table 1.1, the main levels are:

Table 1.1: Levels and Models in the MERISE Methodology

Level	Data	Processes
Conceptual	Conceptual Data Model (MCD)	Conceptual Process Model (MCT)
Organizational/Logical	Logical Data Model (MLD)	Organizational Process Model (MOT)
Physical	Physical Data Model (MPD)	Physical Process Model (MPT)

Additional stages include:

- **Study of the Existing System** : A necessary preliminary analysis for designers unfamiliar with the domain.
- **Validation** : Ensures that each level produces documentation and project files required for review and future reference.

1.2.5 Study of the Existing System

For a designer new to the area being studied, examining the existing system is essential. This involves[TRC89]:

- **Understanding the Domain in Detail** : Gaining a comprehensive knowledge of the study area.

- **Identifying All Objectives** : Listing all goals pursued by the enterprise in this domain.

If a master plan exists, some of this work may already be done.

During the study of the current system, two types of entities provide input:

- **Workstation** : Offers detailed knowledge of specific tasks.
- **Management** : Provides a broad overview and lists all goals within the domain.

Information Collection Techniques

Several techniques may be used to gather information, including:

- **Interviews** : Direct contact facilitates a deep understanding.
- **Questionnaires**
- **Surveys**
- Other methods as needed.

The information gathered is structured for analysis. Interviews vary according to the type of entity:

1- **Management Interviews** :

- Initial understanding of the issue.
- Listing objectives.
- Identifying key workstations.
- Defining interfaces with other projects.
- Setting boundaries for the study area.

Results :

- Main objectives.
- List of workstations.
- General quantifications.
- Study boundaries.
- Constraints in terms of:
  - **Resources** (material, human, financial).
  - **Timelines** (desired completion times).
  - **Regulations** (labor law, general accounting rules, etc.).

2- **Workstation Interviews** :

- Document and describe tasks performed.
- Observe information flow.
- Learn the organization's terminology.

Results :

- Task documentation.
- Data collection.
- List of business rules.

1.3 Data Modeling (Conceptual Level)

This phase involves creating the Conceptual Data Model (CDM/MCD), which is a graphical and structured representation of the information stored by an IS. The CDM is based on two main elements: entities and associations, which give it the alternative name "Entity-Association Diagram."[Bap09, WE14]

The process of creating the CDM/MCD includes:

1. Establishing business rules (if they haven't been provided).
2. Developing the data dictionary.
3. Identifying functional dependencies among data items.
4. Creating the CDM (defining entities, associations, and cardinalities).

1.3.1 Business Rules

Before defining tables (or entities and associations in conceptual terms), it's necessary to gather the future users' requirements. From these, the developer should be able to establish the business rules governing the data to be stored[Bap09, WE14].

Example

Suppose a developer needs to digitize the information system of a library.

- For each book, the title, publication year, summary, and genre (e.g., novel, poetry, science fiction) must be known.
- A book may have zero (for anonymous works), one, or multiple authors, with details such as name, first name, date of birth, and country of origin recorded.
- Each book copy is identified by a unique reference code of letters and numbers, and each belongs to a single edition.
- Each registered user has a unique ID, along with their name, address, phone number, and email.
- Each registered user can borrow zero, one, or more books, with each loan covering one specific copy. The loan date and allowed duration (in days) are recorded.

The business rules of this system are defined as follows:

- A book may have zero (for anonymous works), one, or multiple authors
- Each book copy belongs to a single edition
- Each registered user can borrow zero, one, or more books,
- Each loan covering one specific copy.

If users do not provide clear rules, it's often the developer's responsibility to establish them. Developers may need to guide users through this process, as required.

1.3.2 Data Dictionary

This is an intermediate step, especially useful when multiple people work on the same large database. The data dictionary is a document listing all data items to be stored in the database (and that will appear in the CDM)[Bap09, WE14]. For each data item, it indicates:

- **Mnemonic Code** : A label for the data (e.g., "title_b" for book title).
- **Designation** : A description of the data item (e.g., "title of the book").
- **Data Type** :
  - **A (Alphabetic)** : Letters only.
  - **N (Numeric)** : Numbers only.
  - **AN (Alphanumeric)**: Letters and numbers.
  - **Date** : Date format (YYYY-MM-DD).
  - **Boolean** : True or False.
- **Size** : Expressed in characters or digits. For a date (e.g., YYYY-MM-DD), it would be 10 characters.
- **Remarks** : Additional comments, such as if a value must be positive.

Let's take the example of our library and the loan management system that we are responsible for computerizing. After studying the management rules, we can establish the following data dictionary (Table 1.2):

1.3.3 Functional Dependencies

A functional dependency (FD) exists when two properties (data items) P1 and P2 are related, such that each P1 value corresponds to exactly one P2 value[Sou14]. This dependency is represented as follows:

$$\mathbf{P1} \rightarrow \mathbf{P2}$$

It is said that P1 is the source of the DF and that P2 is its goal. Moreover, multiple properties can be the source, just as multiple properties can be the goal of a DF.

$$\mathbf{P1}, \mathbf{P2} \rightarrow \mathbf{P3} \qquad \mathbf{P1} \rightarrow \mathbf{P2}, \mathbf{P3} \qquad \mathbf{P1}, \mathbf{P2} \rightarrow \mathbf{P3}, \mathbf{P4}, \mathbf{P5}$$

By referring to the previous data dictionary, the following DFs can be established:

id_l → date_l, duration_l, id_r, ref_c

id_r → name_r, Fname_r, street_r, city_r, zcode_r, phone_r, mob_phone_r, email_i, birth_date_r

ref_c → id_b

id_b → title_b, year_b, summary_b, id_t, id_ed

id_t → label_t

id_ed → name_ed

id_a → name_a, Fname_a, birth_date_a, name_c

Table 1.2: Example of a data dictionary

Mnemonic Code	Designation	Type	Size	Comment
id_r	Registered numeric identifier	N		
name_r	Registered last name	A	30	
Fname_r	Registered first name	A	30	
street_r	Registered street address	AN	50	
city_r	Registered city	A	50	
zcode_r	Registered zip code	N	5	
phone_r	Registered home phone	N	15	
mob_phone_r	Registered mobile phone	AN	15	
email_r	Registered email address	AN	100	
birth_date_r	Registered birth date	Date	10	
id_b	Book numeric identifier	N		
title_b	Book title	A	30	
year_b	Book publication year	N	4	
summary_b	Book summary	A	1000	
ref_c	Book copy reference code	AN	15	
id_t	Book type identifier	N		
label_t	Book type label	AN	30	
id_ed	Edition identifier	N		
name_ed	Edition name	AN	30	
id_a	Author numeric identifier	N		
name_a	Author's last name	A	30	
Fname_a	Author's first name	A	30	
birth_date_a	Author's birth date	Date		
id_c	Country numeric identifier	N		
name_c	Country name	A	50	
id_l	Loan numeric identifier	N		
date_l	Loan date	Date		
duration_l	Authorized book loan duration	N		

These FDs (Functional Dependencies) allow us to deduce the following management rule examples:

- From a loan number, we get a loan date, a duration, the ID of the subscriber who made the loan, and the reference of the borrowed copy.
- From a copy reference, we get the ID of the corresponding book.
- From a book number, we get its title, publication year, a summary, the corresponding type ID, and its edition number.
- ...

Functional dependency properties include reflexivity, augmentation, transitivity, union, decomposition, and pseudo-transitivity[Sou14].

Reflexivity

Let X, Y be a set of properties and $Y \subset X$. Then $X \rightarrow Y$

Augmentation

$$X \to Y \Rightarrow X, Z \to Y, Z$$

Transitivity

$$X \to Y \text{ and } Y \to Z \Rightarrow X \to Z$$

Union

$$X \to Y \text{ and } X \to Z \Rightarrow X \to Y, Z$$

Decomposition

$$X \to Y, Z \Rightarrow X \to Y \text{ and } X \to Z$$

Pseudo-transitivity

$$X \to Y \text{ and } Y, W \to Z \Rightarrow X, W \to Z$$

An FD (Functional Dependency) must be:

- **Elementary** : It is the entirety of the source (left side) that must determine the target (right side) of an FD. Example: If $P1 \Rightarrow P3$, then $P1$, $P2 \Rightarrow P3$ is not elementary.
- **Direct** : The FD must not be obtained by transitivity. For example, if $P1 \Rightarrow P2$ and $P2 \Rightarrow P3$, then $P1 \Rightarrow P3$ was obtained by transitivity and is therefore not direct.

Functional Dependency Graph (FDG)

A functional dependency can also be represented using a Functional Dependency Graph (FDG)[Sou14], as shown in the Figure 1.6 :

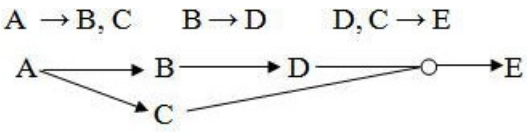


Figure 1.6: FDG Exemple[Citb]

Transitive Closure

Let F be a set of functional dependencies. The transitive closure of F, denoted F+, is obtained by adding all the transitive functional dependencies to F.

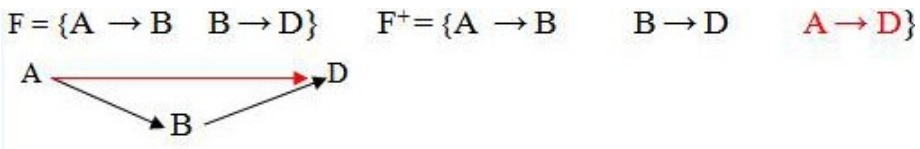


Figure 1.7: Transitive Closure Example[Citb]

Minimal Cover

A minimal cover, denoted F*, is a set of elementary and direct functional dependencies [heb22].

Let F be a set of FDs (functional dependencies). F* is said to be a minimal cover of F if and only if the closure of F is equivalent to the closure of F*, and there exists no proper subset F'* of F* whose closure is also equivalent to the closure of F[Atk88, Sou14].

1.4 Conceptual Data Model (CDM/MCD)

The main objective is the creation of a physical database (internal model) containing information related to the organization's activities. The data to be used within the organization constitutes the perceived reality[Bap09, WE14].

The transition from perceived reality to the internal model is achieved through an intermediate level: the conceptual level. The conceptual model is a synthesis of the perceptions of reality by the various actors within the organization. It is unique and independent of any technological constraints. An actor's perception will be represented as an external model.

A database design method must:

- Specify the formalisms to be used to describe the models at each level;
- Define the interfaces between the levels;
- Define tools to design each of the models.

1.4.1 Basic Concepts

An **entity** is a fundamental concept in database design that represents a distinct object or concept in the real world that is relevant to the system being modeled. It serves as a container for storing data about an object or concept, which is described by its **attributes** (or properties)[Bap09, WE14].

Key Characteristics of an Entity:

1. **Uniqueness** : Each entity must be uniquely identifiable within its context, typically through a unique attribute called the identifier or primary key.
2. **Attributes** : An entity is described by a set of attributes, which are characteristics or properties of the entity.
3. **Occurrences** : An entity can have multiple instances (or occurrences), each representing a specific instance of the object it models.
4. **Relationships** : Entities can be linked to one another through associations or relationships to define interactions between them.

The formalism of an entity is as follows (Figure 1.8):

- **Entity Name** : The label that identifies the type of object (e.g., "Author," "Book").

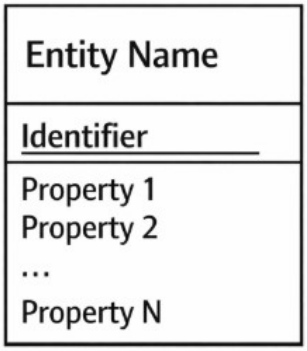


Figure 1.8: Formal Structure of an Entity[Citb]

- **Identifier** : A unique property or combination of properties that distinguishes each occurrence.
- **Attributes** : Properties that describe the entity.

So, if we take our previous data dictionary, we schematize for example an individual "Author" like this (Figure 1.9):

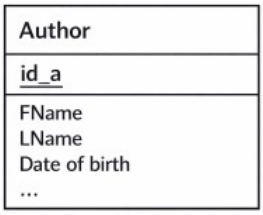


Figure 1.9: Example of an Entity: Author[Citb]

From this entity, we can derive the following management rule:

An author is identified by a unique number (id_a) and is characterized by a last name, first name, and date of birth.

If the entity "Author" contains three occurrences, the Table 1.3 might look like this:

Table 1.3: Occurrence Example

ID_A	NAME	FIRST NAME	DATE OF BIRTH
1	Didouche	Mourad	1927-07-13
2	Zighoud	Youcef	1921-02-18
3	Benbouli	Hassiba	1938-01-18

1.4.2 Association

An **association** is a concept in database design that represents a **semantic link** or relationship between two or more entities. It describes how the entities interact with one another and helps

translate additional business rules that cannot be captured through the individual definition of entities[Bap09, WE14].

Key Characteristics of an Association:

1. **Entities Involved** : Associations link one or more entities together.
2. **Cardinality** : Defines the number of instances of one entity that can be associated with instances of another entity (e.g., one-to-one, one-to-many, many-to-many).
3. **Attributes** : Some associations have their own properties that describe the relationship.

The formalities of an association are as follows (Figure 1.10):

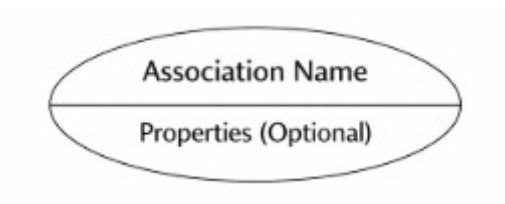


Figure 1.10: Association[Citb]

- **Association Name** : Describes the relationship (e.g., ”Writes,” ”Borrows”).
- **Entities** : The entities involved in the relationship.
- **Attributes (Optional)** : Properties specific to the association.
- **Cardinality** : Specifies the rules of the relationship (e.g., 1, M).
- **”Author writes Book”** This association links the ”Author” and ”Book” entities and captures the business rule that an author can write one or more books.

Generally, the name of the association is a verb defining the link between the individuals who are connected by it.

Exemple shown in the Figure 1.11 :

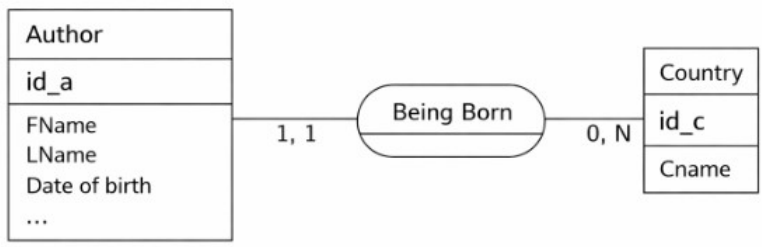


Figure 1.11: Association Exemple[Citb]

Here, the ”being born” association translates to the following two business rules:

- An author is born in exactly one country.
- In a country, there are no, one, or several authors born.

You will notice that this association is characterized by these annotations (1,1) and (0,N), which allowed us to define the previous business rules. These annotations are called **cardinalities**.

Cardinalities in the context of databases, entity-relationship (ER) diagrams, or data modeling refer to the number of instances of one entity that can or must be associated with an instance of another entity. Cardinality defines the allowable number of relationships between two entities[Bap09, WE14].

Cardinalities are expressed as pairs of numbers: (**minimum, maximum**), and they specify the rules of participation in the relationship.

Here are the common types of cardinalities:

1. **(1,1)** :
   - One-to-One relationship. Each instance of an entity is associated with exactly one instance of another entity.
   - Example: Each person has exactly one passport.
2. **(0,1)** :
   - Zero or One relationship. An entity instance can either be associated with no instance or exactly one instance of another entity.
   - Example: A student may or may not have a scholarship.
3. **(0,N)** :
   - Zero or Many relationship. An entity instance can be associated with zero or multiple instances of another entity.
   - Example: A book can have zero or more authors.
4. **(1,N)** :
   - One or Many relationship. An entity instance must be associated with at least one, but possibly many, instances of another entity.
   - Example: A teacher must teach at least one course.
5. **(M,N) (less common)** :
   - Many-to-Many relationship. Both entities can be associated with multiple instances of the other entity.
   - Example: Students can enroll in multiple courses, and each course can have multiple students.

These cardinalities help define the structure and rules of data relationships, ensuring the integrity of the system and accurate representation of real-world constraints.

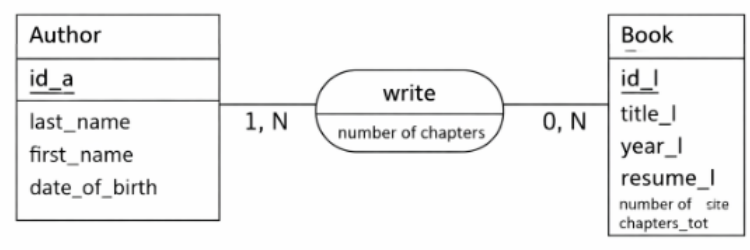


Figure 1.12: Example Cardinalities[Ben]

Exemple: Consider the following association (Figure 1.12):

Here, an author writes at least one or more books, and for each book, the number of chapters written by the author is known (the total number of chapters for each book is also known).

The "writes" association can therefore be identified by the concatenation of the *id_a* and *id_l* properties. Thus, the pair *id_a*, *id_l* must be unique for each occurrence of the association[WE14], We can also define the following functional dependency:

$$\text{id\_a, id\_l} \rightarrow \text{nb\_chapters}$$

We say that nb_chapitres (number of chapters written by an author, for a book) is a property carried by the "writes" association.

Dimension of an Association

A relation can be unary (connecting an entity to itself), binary (connecting two entities), or ternary (connecting three entities)[Bap09, WE14] (Figure 1.13).

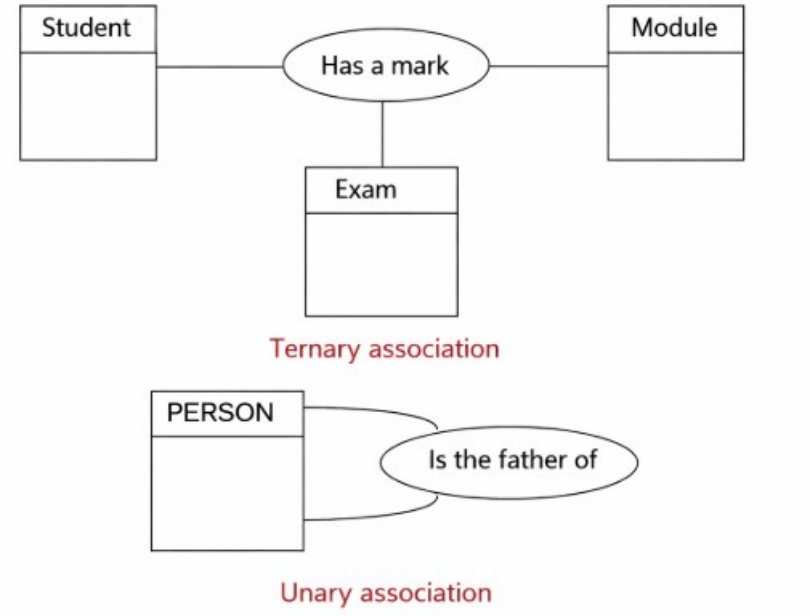


Figure 1.13: Dimension of an Association[Ben]

1.4.3 Development of the CDM/MCD

It is therefore possible to construct the complete Conceptual Data Model (CDM) from the properties in the data dictionary and from the functional dependency graph. Let the FDs (functional dependencies) defined above be:

id_l → date_l, duration_l, id_r, ref_c

id_r → name_r, Fname_r, street_r, city_r, zcode_r, phone_r, mob_phone_r, email_i, birth_date_r

ref_c → id_b

id_b → title_b, year_b, summary_b, id_t, id_ed

id_t → label_t

id_ed → name_ed

id_a → name_a, Fname_a, birth_date_a, name_c

We therefore obtain the following MCD (Figure 1.14):

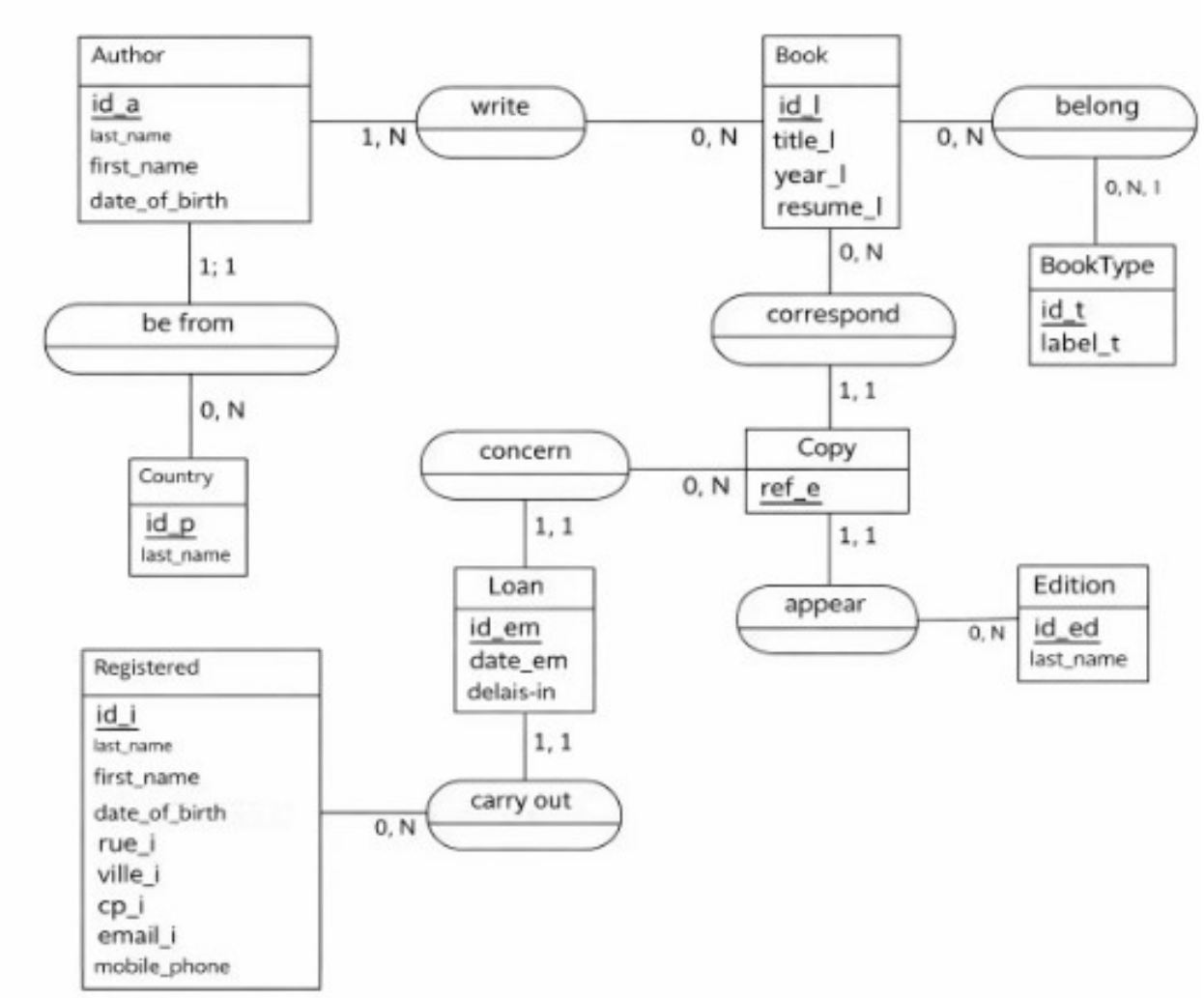


Figure 1.14: Example 01 of MCD[Ben]

1.4.4 Exemples of CDM/MCD

Problem Statement 02: Company/Employees

In a company, a department is identified by a name and characterized by a location.
An employee is characterized by a number, their name, their grade, and the department in which they work.
An employee's number is unique within a department but not across the company.

Provide the Conceptual Data Model (CDM/MCD), specifying the attributes (Figure 1.15).

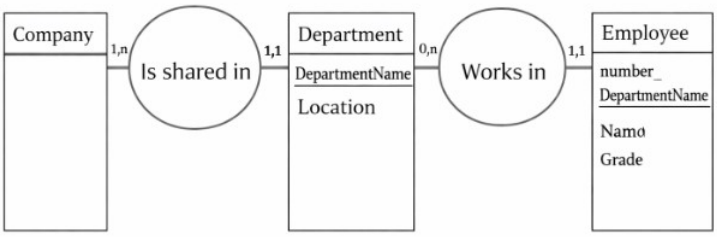


Figure 1.15: Example 02 of MCD[Citc]

Problem Statement 03: Multimedia Library

A multimedia library contains books that can be borrowed.
A book is characterized by a unique number, a title, an author, and a publisher. Furthermore, each book is described by several keywords indicating the subjects it covers.
The library has one or more copies of each book.
Each copy is identified by a number and characterized by its shelf location and purchase date.
A copy can be borrowed by a borrower. Borrowers are identified by a borrower number, and have a name and an address.
Provide the Conceptual Data Model (CDM/MCD) (Figure 1.16).

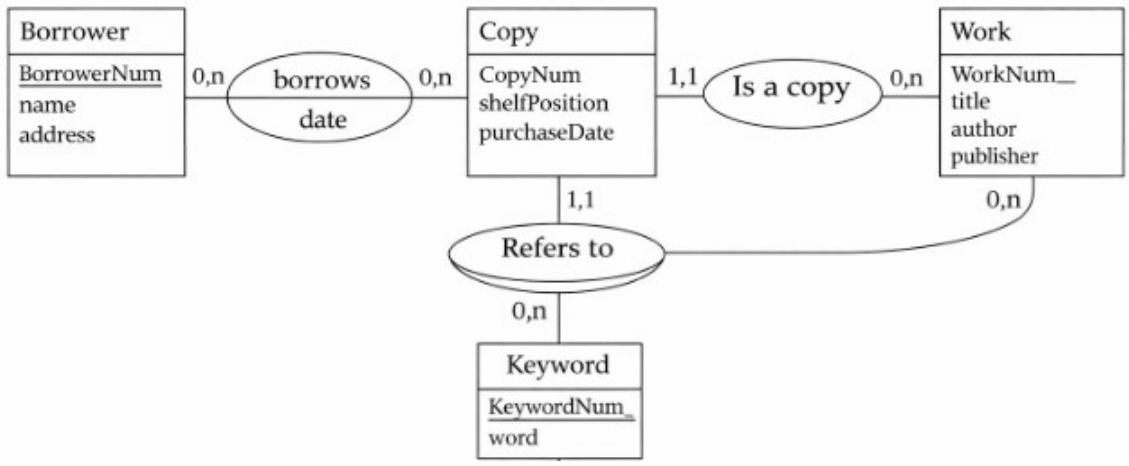


Figure 1.16: Example 03 of MCD[Citc]

1.5 Conceptual Process Model (CPM/MCT)

1.5.1 Basic Concepts

The Event

A report for the Information System (IS). It can be external (originating from the outside world) or internal (produced by the IS).

- An external event $\Longrightarrow$ Triggers a reaction from the IS in the form of an operation.
- An internal event $\Longrightarrow$ Triggers a new reaction from the IS or constitutes a result for the outside world.

An event can carry properties.

Event Type

A set of events characterized by:

- The same types of associated properties.
- The same types of actions to be performed.

Each event of this type constitutes an **occurrence** of this event type.

The Operation

A set of actions performed by the IS (Information System) in reaction to an **event** or a conjunction of events.
This set of actions cannot be interrupted by waiting for new events.
An operation produces new events as output.

Operation Type

A set of operations characterized by:

- The types of action to be taken (action = combination of elementary actions: add, modify, etc.).
- The types of contributing events.
- The types of events produced (internal or results), the emission of which is subject to emission rules that are management rules.

Synchronization

The synchronization of an operation marks the meeting point of contributing events that must occur before triggering the operation, according to a logical proposition that translates into activation rules.

Synchronization Type

The type of synchronization is characterized by: beginitemize

A list of types of contributing events.

Activation rules related to these types of events.

The synchronization of the first operation can happen without a waiting period as soon as an event occurs. However, the synchronization of any subsequent operation must correspond to a wait.

Process

The MCT (Conceptual Process Model) is generally broken down into processes.
A process is a sequence of operations included within the same activity domain.

Below (in the Figure 1.17) is the functional diagram of a MCT.

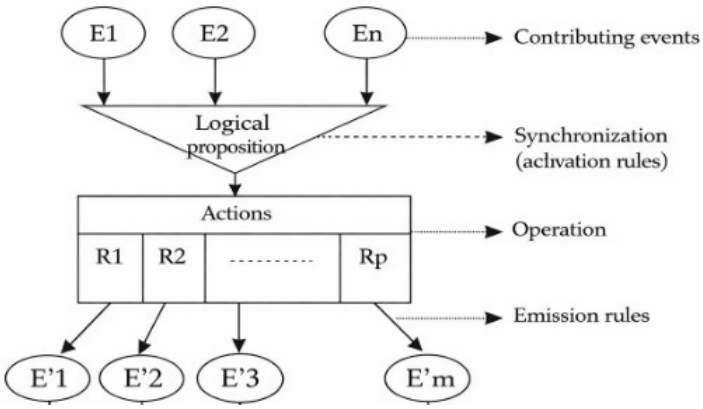


Figure 1.17: Operating Diagram of an MCT[Ben]

1.5.2 Exemples of CPM/MCT

Problem Statement 01

In a large administration, promotion requests are processed according to the following management rules:

- Management Rule 1: Every promotion request must undergo a preliminary examination to determine if it is admissible or not.
- Management Rule 2: The review of an admissible request's file can only take place after a report from the hierarchical superior.
- Management Rule 3: After the file is reviewed by the competent authority, the promotion will be granted or denied.

Solution 01

The solution is shown in the Figure 1.18 :

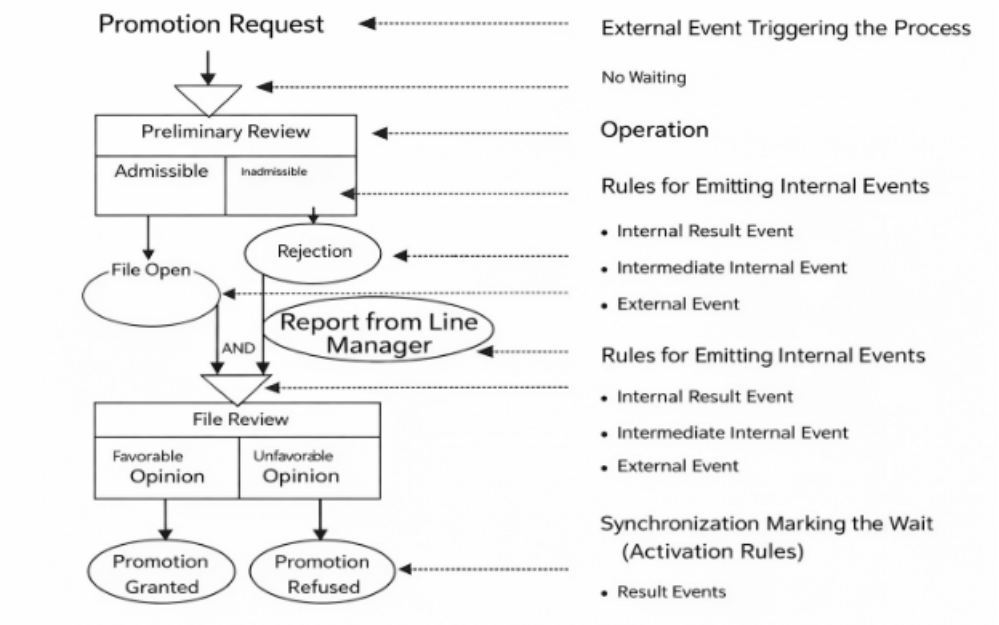


Figure 1.18: Example 01 of MCT[Ben]

Consumption of Contributing Event Occurrences In the diagram below (Figure 1.19), event occurrences are represented by tokens. These tokens are consumed, and others are produced. Each occurrence of a contributing event, which triggers the synchronization, is consumed.

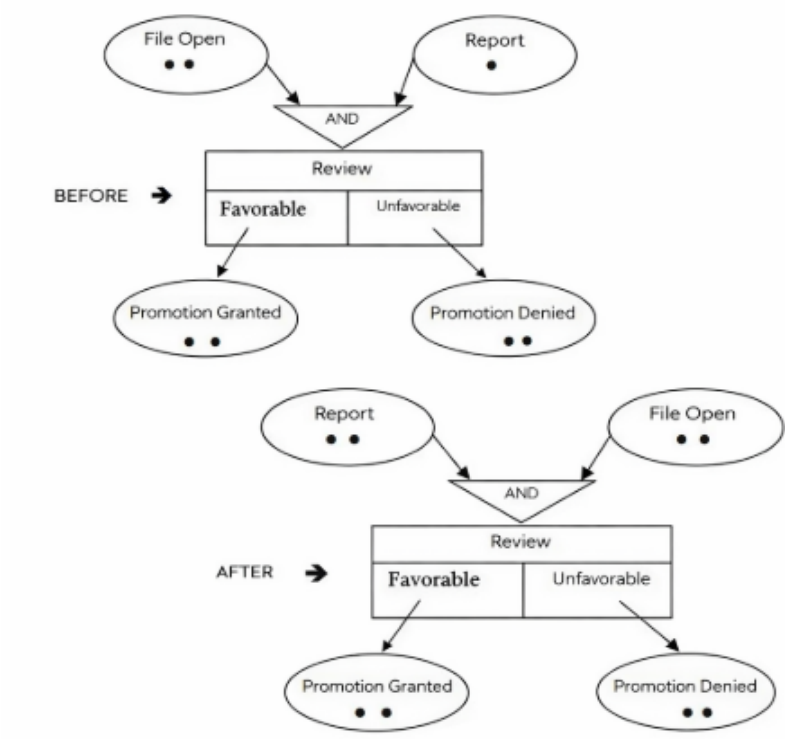


Figure 1.19: Example 01-Consumption Occurrences Contributing Events[Ben]

Problem Statement 02

The processing related to the study of a client's solvency, knowing that it is done at each reception of a purchase order and according to the following conditions based on the balance:

- if the account balance is less than 1000€, a 50
- if the balance is negative and greater than or equal to -1000€, the order is put on hold, and a letter is sent notifying that the account is in debit.
- if the balance is positive, the order is delivered with the invoice.

Solution 02

The solution is shown in the Figure 1.20 :

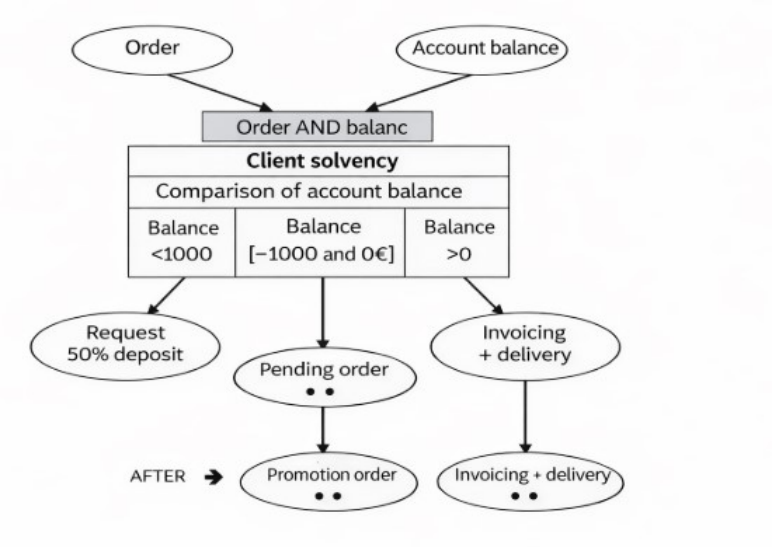


Figure 1.20: Example 02 of MCT

Problem Statement 03

When an employee submits a promotion request, their file is retrieved and examined. If their seniority and service record justify it, a report from their direct supervisor is included in the discussion after being reviewed.
A vote determines whether the opinion is favorable or not. If it is, the promotion is granted; otherwise, it is denied.
Seniority check: If the seniority or service record is unfavorable, the file is rejected.

Solution 03

The solution is shown in the Figure 1.21 :

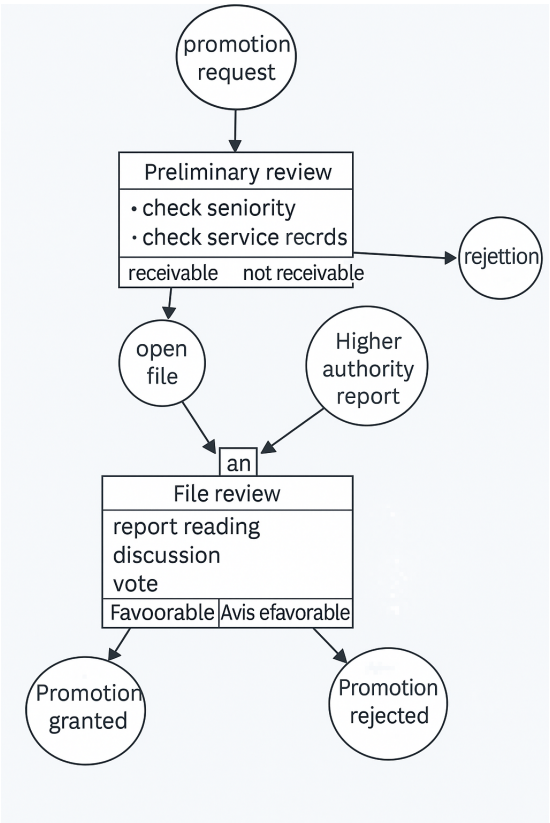


Figure 1.21: Example 03 of MCT

Problem Statement 04

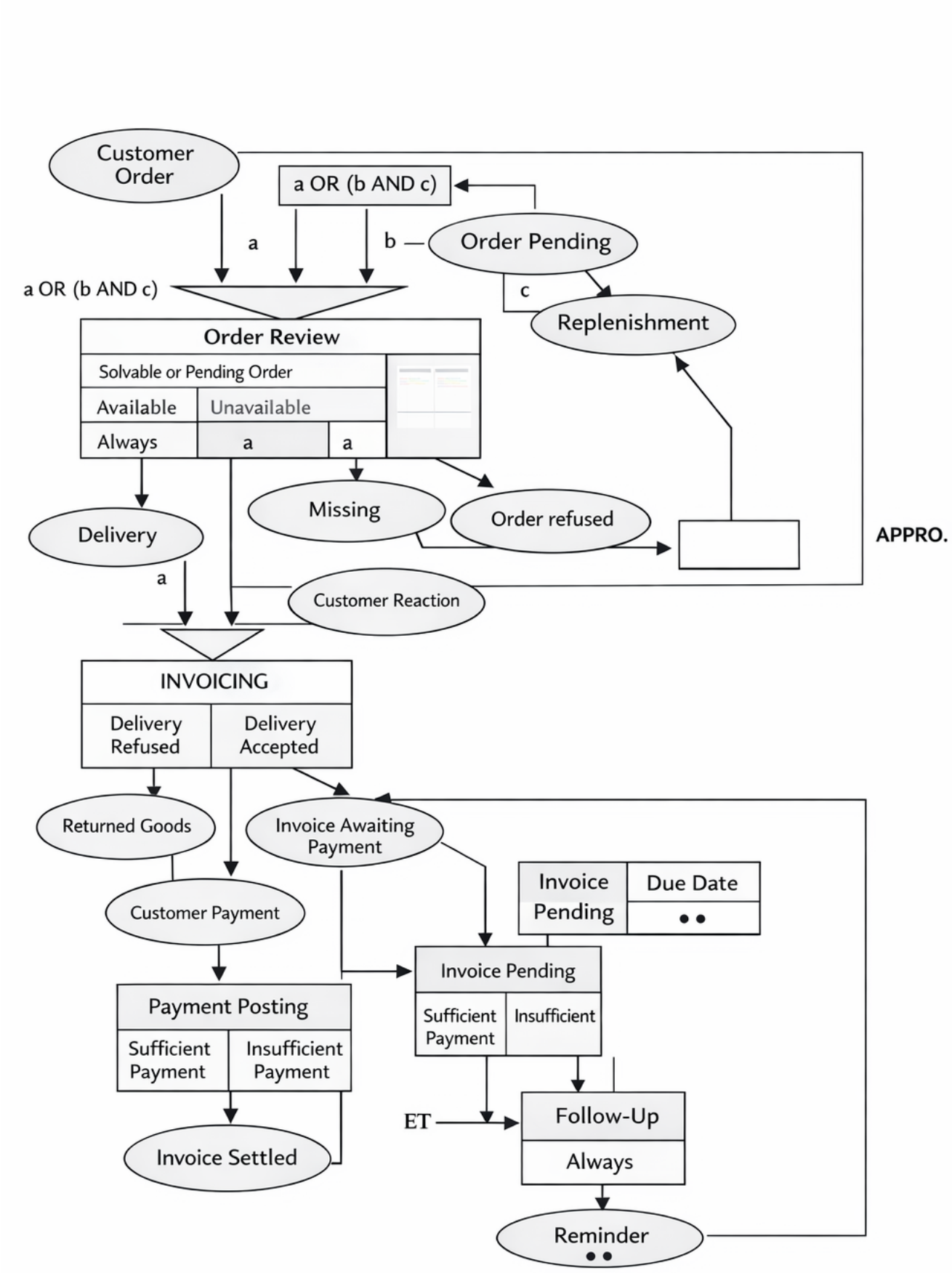
There are 3 linked processes: Customer Order Processing, Stock Management, and Procurement. The process being considered is Customer Order Processing.

Management Rules

- **M.R.1:** Any order from an insolvent customer is refused.
- **M.R.2:** Unavailable orders are put on hold and must trigger a restock from the supplier.
- **M.R.3:** Orders on hold will be declared available when the restock is sufficient.
- **M.R.4:** Available orders lead to customer delivery.
- **M.R.5:** Deliveries refused by the customer lead to a merchandise return.
- **M.R.6:** Accepted deliveries lead to invoices that are kept until full payment is complete.
- **M.R.7:** Any invoice not paid by the due date results in a reminder.

Solution 04

The solution is shown in the Figure 1.22 :


Customer Order
a OR (b AND c)
a
b
Order Pending
c
Replenishment
a OR (b AND c)
Order Review
Solvable or Pending Order
Available
Unavailable
Always
a
a
Missing
Order refused
Delivery
APPRO.
a
Customer Reaction
INVOICING
Delivery Refused
Delivery Accepted
Returned Goods
Invoice Awaiting Payment
Invoice Pending
Due Date
Customer Payment
Invoice Pending
Sufficient Payment
Insufficient
Payment Posting
Sufficient Payment
Insufficient Payment
ET
Follow-Up
Always
Invoice Settled
Reminder

Chapter 2

General Information about the Access Program

Contents

2.1 What is Microsoft Access?

Microsoft Access is a Relational Database Management System (RDBMS). It is part of the Microsoft Office suite. Unlike a simple spreadsheet program like Excel, which is optimized for calculations and analysis on limited datasets, Access is designed to organize, store, manipulate, and retrieve large amounts of information in a structured and secure manner[Mic25r].

2.1.1 Definition and Role

As an RDBMS, Access allows you to:

- **Define data** : Create tables to hold information and specify the data types for each column (text, number, date, etc.)[MON].
- **Establish relationships** : Link tables together to ensure data consistency and integrity (for example, linking a "Customers" table to an "Orders" table).
- **Manipulate data** : Add, modify, or delete information through user-friendly interfaces called forms[MON].
- **Query data** : Ask complex questions of the database using queries to extract specific information (for example, "which customers have ordered more than €100 this month?")[Jil24].
- **Present data** : Create professional reports for printing or viewing[MON].

2.1.2 Positioning relative to other tools

Access vs. Excel

Excel is ideal for simple lists, quick calculations, and charts. Access is superior when it comes to managing large volumes of interdependent data, avoiding information redundancy, and ensuring long-term data integrity.

Access vs. SQL Server

Access is a perfect desktop solution for small to medium-sized applications (SMEs, departmental projects). Systems like SQL Server or Oracle are much more powerful client-server RDBMSs, designed for large-scale enterprise applications requiring high performance, advanced security, and management of many concurrent users. However, Access can serve as a "front-end" (interface) to connect to these more robust systems[Mic25r].

In summary, Access is an all-in-one tool that combines a database engine, a graphical interface for creating objects, and a rapid application development environment. It is the ideal tool for moving from the theory of relational databases to concrete and rapid practical application.

2.2 Overview of the Access Interface

When you first launch Access, you will find a user interface that is consistent with other software in the Microsoft Office suite. It is quick and intuitive to learn. The workspace is organized around three main components that allow you to create, manage, and use your database[Mic25j].

2.2.1 The Ribbon

Located at the top of the window, the Ribbon is the central element for accessing commands. It replaces the old menus and toolbars and organizes features logically under different tabs[DA20].

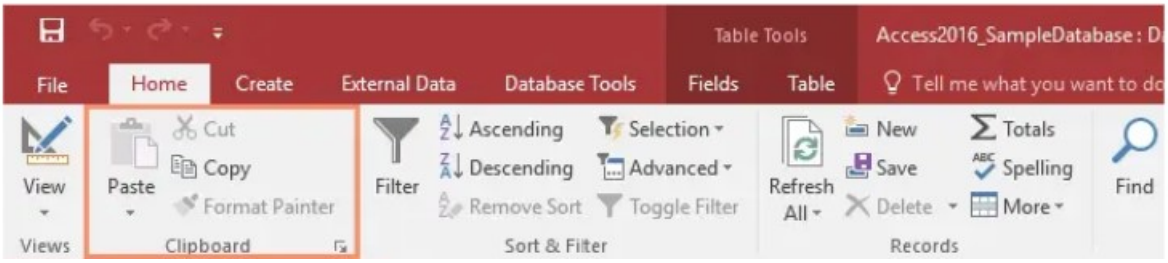


Figure 2.1: The Ribbon[Scr]

The main tabs are:

- **File** : Opens the ”Backstage” view, which allows you to manage the database file (open, save, print, compact and repair)[Mic25j].
- **Home**: Contains common commands for data manipulation (sort, filter, find, formatting).
- **Create**: This is the most important tab for development. It groups all the tools for creating the various database objects: tables, queries, forms, and reports[ano25].
- **External Data**: Allows you to import data from other sources (Excel, text files) or export data from Access to other formats.
- **Database Tools**: Provides advanced features such as managing relationships, analyzing performance, or running macros.

The Ribbon is also contextual: it displays additional tabs (e.g., ”Table Tools” or ”Form Tools”) when you select a specific object, showing only the commands relevant to that object[ano25].

2.2.2 The Navigation Pane

Located on the left side of the window (as show in Figure 2.2), the Navigation Pane is your database explorer. It lists all the objects you have created (tables, queries, forms, etc.) and allows you to open, rename, delete, or modify their design (right-click on an object)[Mic25j, ano25].

By default, objects are grouped by type (”Tables,” ”Queries,” etc.), which is the most convenient view for development. You can open an object by double-clicking it; it will then appear as a tabbed document in the main workspace[Mic25j].

2.2.3 The Workspace (or Object Window)

This is the main area in the center of the screen where objects are displayed when you open them. Access uses a tabbed document interface, which means you can have multiple tables, forms, or queries open simultaneously, each in its own tab, making it easy to navigate between them[DA20].

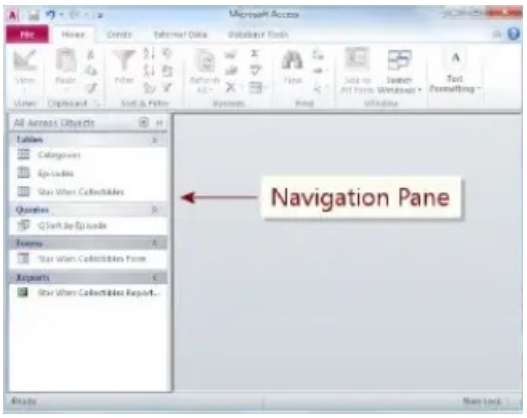


Figure 2.2: The Navigation Pane[Scr]

At the bottom of the window, the Status Bar provides contextual information and buttons to quickly switch views (for example, switching from "Datasheet View" to "Design View" for a table)[DA20].

2.3 The Main Objects of an Access Database

An Access database is a container that brings together several types of objects. Each has a specific role, but they are designed to work together to create a complete and functional application. Although others exist, we will focus on the six fundamental objects[Reg].

2.3.1 Tables

Tables are the heart of the database. They are the fundamental object where all information is physically stored. A table is organized in a two-dimensional grid[MON]:

Customers

Column

Row

ID	First Name	Last Name	Street Address	City	State
1	Tracey	am	7 East Walker Dr.	Raleigh	NC
2	Lucinda	George	789 Brewer St.	Cary	NC
3	Jerrod	Smith	211 St. George Ave.	Raleigh	NC
4	Brett	Newkirk	47 Hillsborough St.	Raleigh	NC
5	Chloe	Jones	23 Solo Ln.	Raleigh	NC
6	Quinton	Boyd	4 Cypress Cr.	Durham	NC
7	Alex	Hinton	1011 Hodge Ln.	Cary	NC
8	Nisha	Hall	123 Huntington St.	Raleigh	NC
9	Hillary	Clayton	2516 Newman	Raleigh	NC
10	Kiara	Williams	9014 Miller Ln.	Durham	NC
11	Katy	Jones	456 Denver Rd.	Cary	NC
12	Beatrix	Joslin	85 North West St.	Raleigh	NC
13	Mariah	Allen	12 Jupe	Raleigh	NC
14	Jennifer	Hill	2100 Field Ave.	Raleigh	NC
15	Jaleel	Smith	123 Hill Top Drive	Garner	NC

Figure 2.3: The Table

- **Rows**, called Records, contain information related to a single entity (e.g., a customer, a product, an order).
- **Columns**, called Fields, describe a specific characteristic of that entity (e.g., the customer's name, the product's price, the order date).

In a well-designed database, each table stores information about only one subject (e.g., a table for customers, another for products). This is the basis of normalization.

2.3.2 Queries

Queries are used to ask questions of the database. They allow you to retrieve, filter, sort, combine, and calculate data from one or more tables. A query can be seen as a question posed to the database ("Which customers live in London?"). Queries can also be used to perform actions on the data, such as adding, modifying, or deleting records in bulk[Mic25o, MON].

2.3.3 Forms

Forms are graphical interfaces designed to facilitate data entry, modification, and viewing. Instead of working directly in tables (which resemble spreadsheets), users interact with user-friendly and secure forms. These can present data from a single record at a time and include buttons, drop-down lists, and other controls to simplify actions[MON].

2.3.4 Reports

Reports are specifically designed for the presentation and printing of data. They allow information to be formatted professionally to create summaries, invoices, labels, or lists. Unlike forms, reports are not meant for data entry. Their strength lies in their ability to group and summarize information (e.g., calculating totals, subtotals, and averages)[MON].

2.3.5 Macros

Macros allow you to automate repetitive tasks without having to write programming code. A macro is a sequence of predefined actions (such as opening a form, printing a report, running a query) that can be triggered by a simple button click. It is a powerful tool for simplifying navigation and processes within the application.

2.3.6 Modules

Modules contain programming code written in Visual Basic for Applications (VBA). While macros are limited to a list of predefined actions, VBA modules offer almost unlimited flexibility and power to create complex logic, handle errors, and interact with other Office applications. They are used when macros are not sufficient.

2.4 Create Database

To create a database from a template, we first need to open MS Access and you will see the following screen in which different Access database templates are displayed (Figure 2.4).

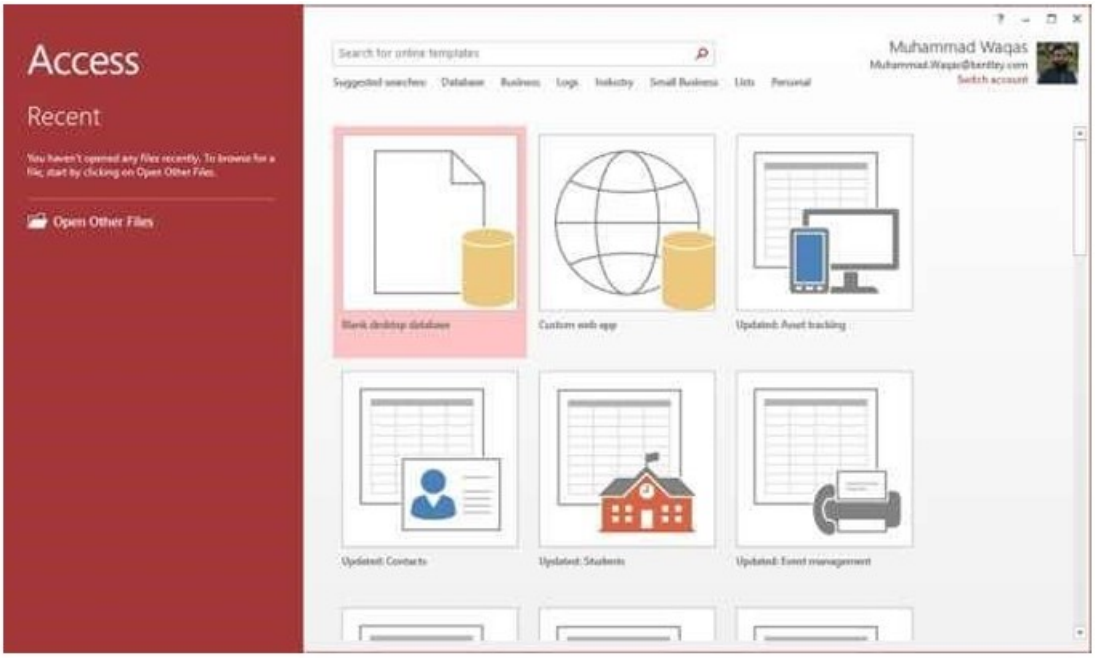


Figure 2.4: Creating a Database - Window 1[Scr]

To view the all the possible databases, you can scroll down or you can also use the search box.

Let us enter project in the search box and press Enter. You will see the database templates related to project management (Figure 2.5).

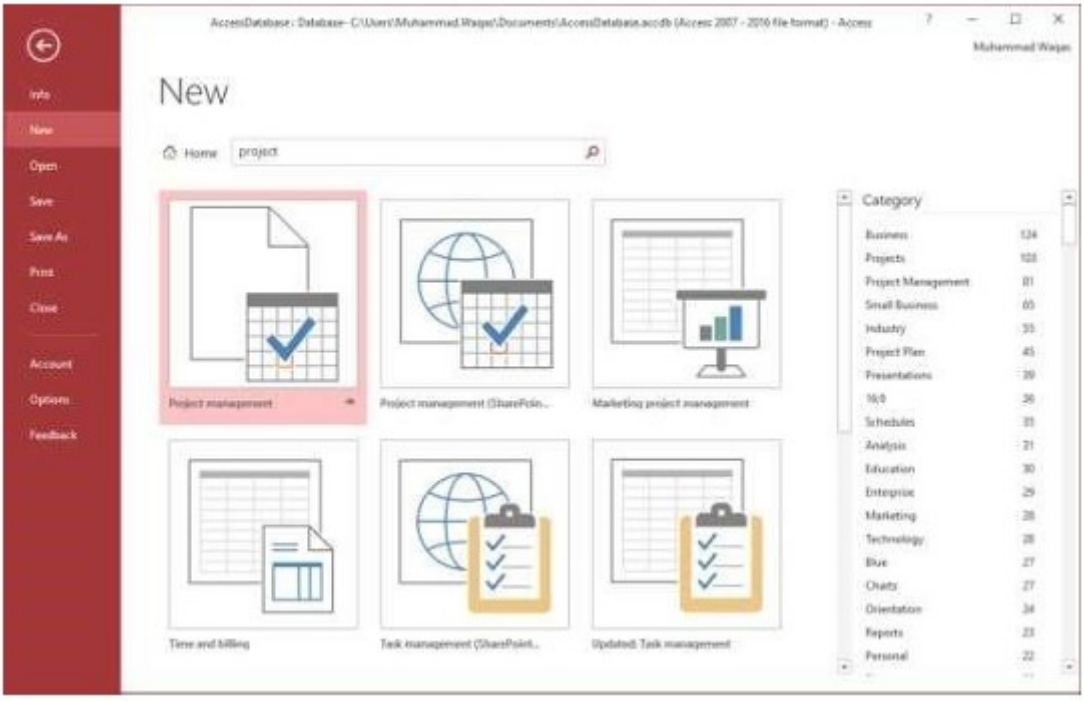


Figure 2.5: Creating a Database - Window 2[Scr]

Select the first template. You will see more information related to this template (Figure 2.6).

After selecting a template related to your requirements, enter a name in the File name field and you can also specify another location for your file if you want.

Now, press the Create option. Access will download that database template and open a new blank database as shown in the Figure 2.6.

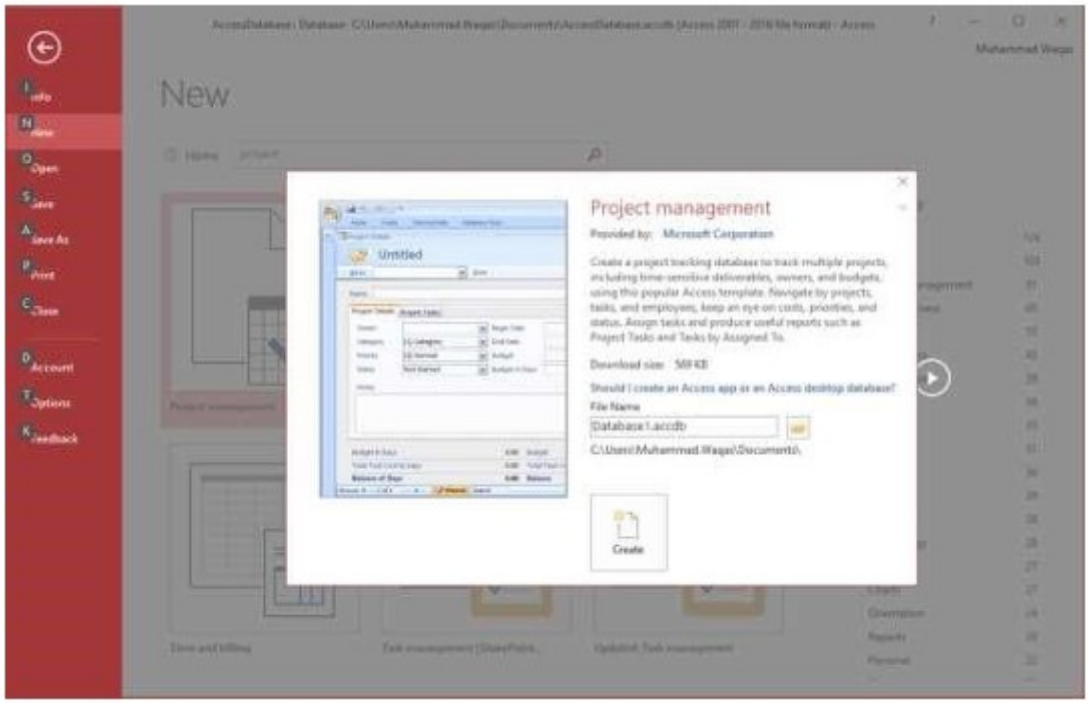


Figure 2.6: Creating a Database - Window 3[Scr]

Now, click the Navigation pane on the left side and you will see all the other objects that come with this database (Figure 2.7).

Click the Projects Navigation and select the Object Type in the menu (Figure 2.8).

You will now see all the objects types — tables, queries, etc(Figure 2.9).

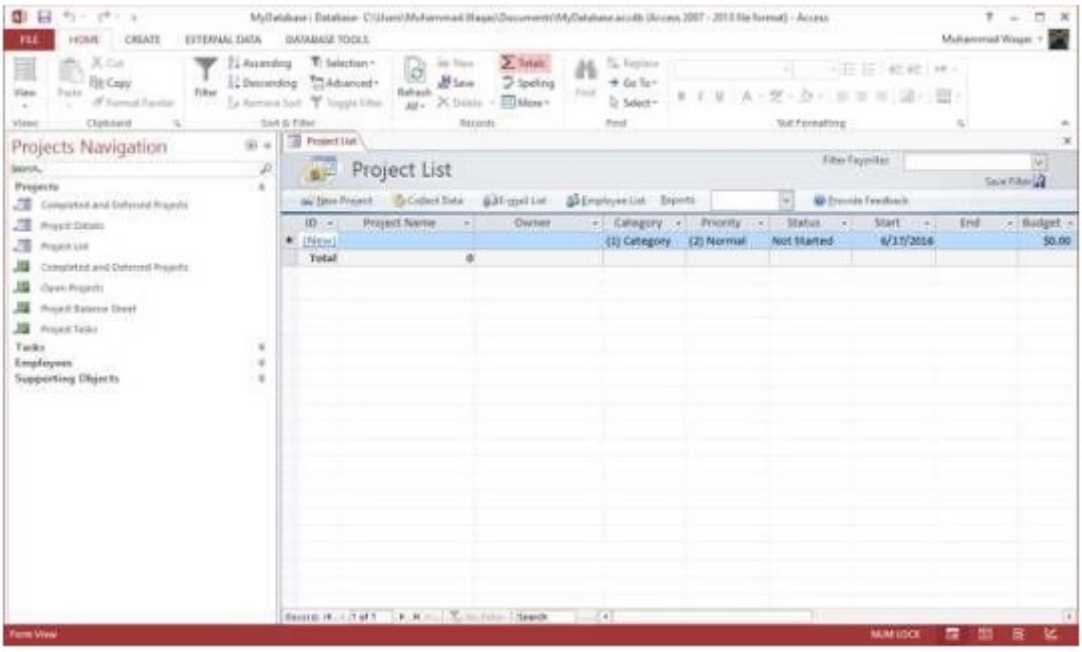


Figure 2.7: Creating a Database - Window 4[Scr]

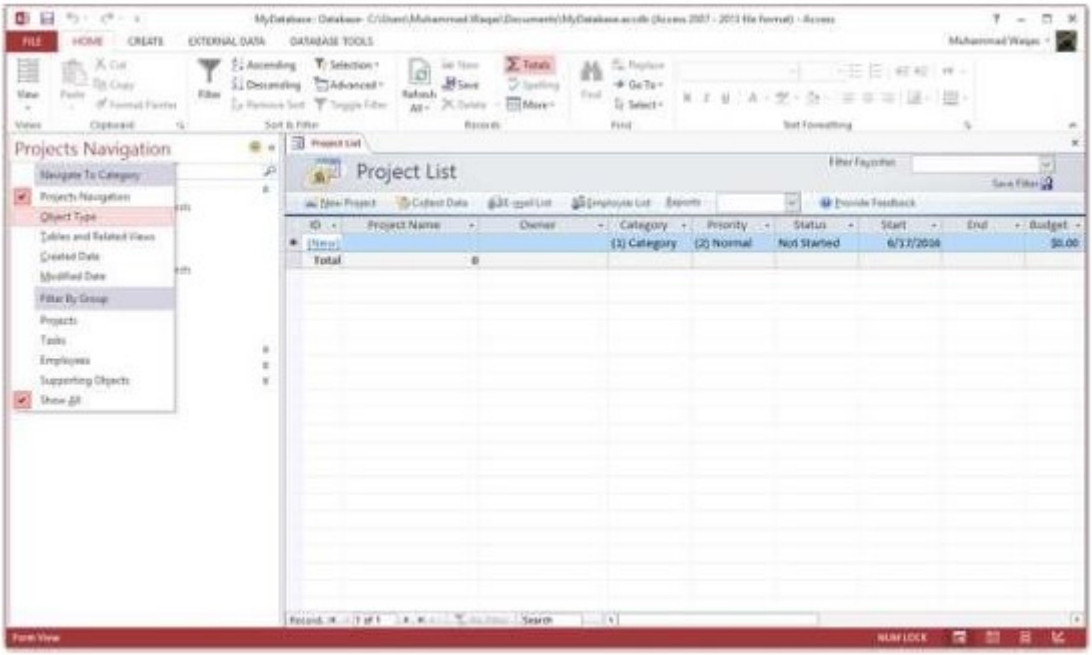


Figure 2.8: Creating a Database - Window 5[Scr]

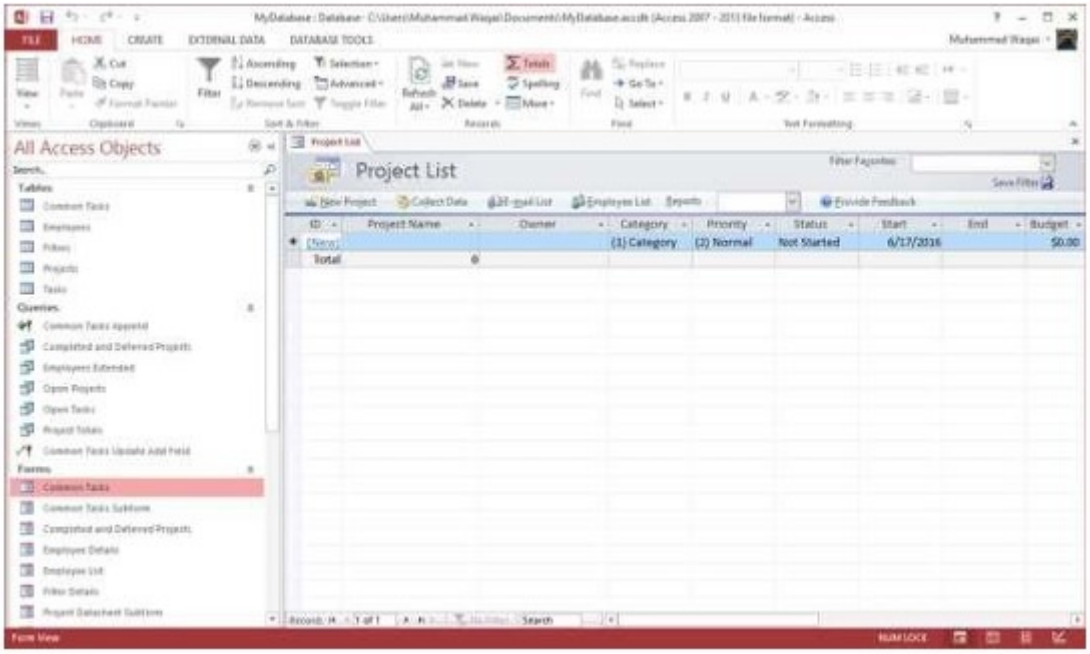


Figure 2.9: Creating a Database - Window 6[Scr]

Chapter 3

Creating Tables in the Database

Contents

Tables are the foundation of our database. A rigorous design at this stage is essential to ensure the integrity, performance, and longevity of our application. Before entering any information, we must define the structure of each table: the fields that compose it, the type of data each field can contain, and their specific properties[Mic25c].

3.1 Designing a Table in Design View

Although Access allows you to create tables directly in Datasheet View (similar to Excel), Design View is the most rigorous and comprehensive method, as it gives you complete control over the table's structure before inserting any data.

To create a table in Design View:

1. Go to the Create tab on the ribbon.
2. In the Tables group, click Table Design.

Figure 3.1: Table Design[Scr]

Access will then open a new window with a specific design grid. This grid is divided into two main areas:

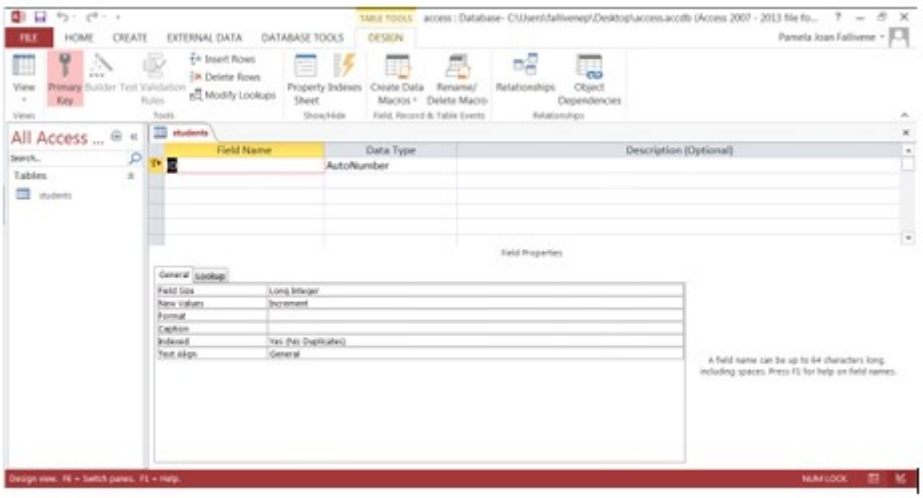


Figure 3.2: The Design Grid of a Table[Scr]

3.1.1 The Upper Grid: Field Definition

This is where you list all the fields (columns) that your table will contain. For each field, you must specify three pieces of information:

- **Field Name** : This is the name you give to the column. It should be short, descriptive, and without spaces (use underscores or CamelCase, e.g., CustomerName). This name will be used in queries, forms, and reports.
- **Data Type** : This defines the nature of the information the field will hold. Choosing the correct data type is crucial for ensuring data integrity and optimizing storage. You select the type from a drop-down list (e.g., Short Text, Number, Date/Time, etc.).
- **Description (Optional but recommended)** : This column allows you to document the purpose of the field. This is a good practice that facilitates database maintenance, for yourself or for other developers.

3.1.2 The Lower Area: Field Properties

When you select a field in the upper grid, the lower area of the screen displays the properties specific to that field. These properties allow you to refine the field's behavior and validation rules. The properties vary depending on the chosen data type, but the most common ones include[Mic25c]:

- **Field Size** : Defines the maximum number of characters (for text) or the type of number (Integer, Double, etc.).
- **Format** : Controls how the data is displayed (e.g., dd/mm/yyyy date format, € currency format).
- **Input Mask** : Enforces a strict data entry format (e.g., for a phone number).
- **Caption** : Allows you to define a more user-friendly name for the field, which will be displayed in forms and reports.
- **Default Value** : A value that will be automatically inserted into the field for any new record.
- **Validation Rule** : A validation expression to ensure that the entered data adheres to a certain rule (e.g., ¿0 for a price).
- **Required** : Specifies whether the field must contain a value.

Mastering Design View is therefore the fundamental step to building a robust and well-structured database.

3.2 Data Types

Choosing the data type for each field is one of the most important decisions when designing a table. This choice not only determines the nature of the information you can store, but it also has a direct impact on data integrity, storage space optimization, and the operations you can perform. Access offers a variety of data types, each suited for a specific purpose[Mic25f, Mic25m].

Here are the main data types you will use:

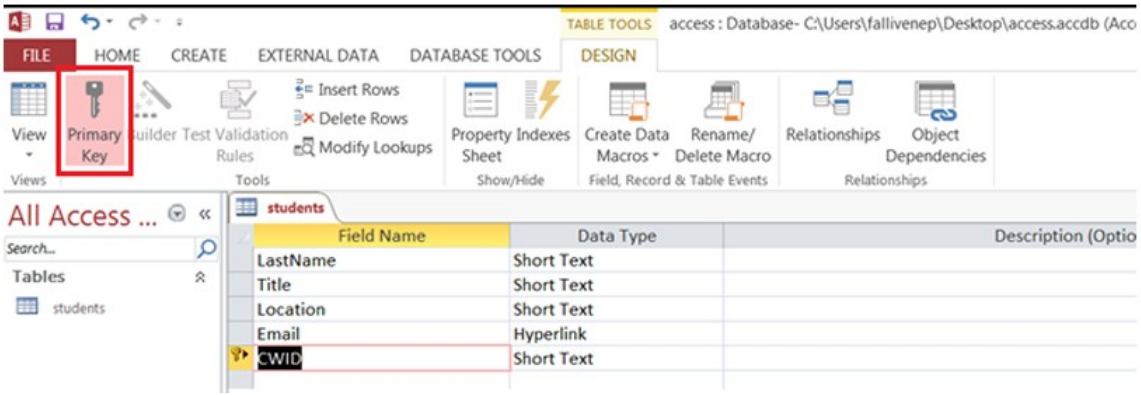


Figure 3.3: Data Types[Scr]

3.2.1 Textual Data Types

Short Text

This is the most common type. It is used for small-scale alphanumeric data (letters, numbers, and symbols)[Mic25f].

- **Use**: Names of people, addresses, postal codes, phone numbers (as they are not used in calculations).
- **Size**: Up to 255 characters.

Long Text

Formerly known as ”Memo,” this type is used for large blocks of text.

- **Use**: Detailed product descriptions, comments, meeting notes.
- **Size**: Can store up to about 1 gigabyte of text, although Access controls can only display the first 64,000 characters[Mic25f].

3.2.2 Numeric Data Types

Formerly known as ”Memo,” this type is used for large blocks of text.

Number

To be used for numerical data intended for mathematical calculations[div13].

- **Use**: Quantities in stock, student grades, ages.
- **Important Property**: The ”Field Size” property allows you to specify the type of number (Byte, Integer, Long Integer, etc.) to optimize storage.

Currency

This type is a variant of the Number type, optimized for financial values[div13].

- **Use**: Product prices, salaries, invoice amounts.
- **Advantage**: It is designed to prevent rounding errors that can occur with the Number type in financial calculations[Mic25f].

3.2.3 Temporal Data Type

Date/Time

Specifically designed to store dates, times, or a combination of both[div13].

- **Use**: Date of birth, order date, appointment time.
- **Advantage**: Allows for chronological sorting and performing calculations on dates (e.g., calculating an age or the duration between two dates).

3.2.4 Specific Data Types

AutoNumber

Generates a unique, sequential (or random) number for each new record added to the table[Mic25f].

- **Use**: Ideal for primary keys (e.g., CustomerID, OrderNum), as it guarantees the uniqueness of each record. This field cannot be edited by the user.

Yes/No

For data that can only have two states (Boolean values)[div13].

- **Use**: True/False, Yes/No, Active/Inactive.
- **Display**: Most often appears as a check box in forms and tables[Mic25f].

Attachment

Allows you to attach one or more files directly to a record, similar to an email[Mic25f].

- **Use**: Attaching a candidate's resume, a product photo, a PDF document.

Hyperlink

For storing website addresses or paths to files on a computer or network[div13].

- **Use**: Company website, email address, link to a document on a server.

These types of data can be summarized in the Table 3.1

Table 3.1: Microsoft Access Database - Data Types

Type of Data	Description	Size
Short Text	Text or combinations of text and numbers, including numbers that do not require calculating (e.g. phone numbers).	Up to 255 characters.
Long Text	Lengthy text or combinations of text and numbers.	Up to 63,999 characters.
Number	Numeric data used in mathematical calculations.	1, 2, 4, or 8 bytes (16 bytes if set to Replication ID).
Date/Time	Date and time values for the years 100 through 9999.	8 bytes.
Currency	Currency values and numeric data used in mathematical calculations involving data with one to four decimal places.	8 bytes.
AutoNumber	A unique sequential (incremented by 1) number or random number assigned by Microsoft Access whenever a new record is added to a table.	4 bytes (16 bytes if set to Replication ID).
Yes/No	Yes and No values and fields that contain only one of two values (Yes/No, True/False, or On/Off).	1 bit

3.3 Field Properties

After choosing a name and data type for each field, the next step is to refine their behavior by setting their properties. These are characteristics that control how data is entered, stored, and displayed. They provide a powerful mechanism for ensuring data integrity and consistency directly at the table level[ERN21, Mic25m].

The available properties vary depending on the selected data type, but here are the most important ones to master (Figure 3.4):

3.3.1 Format and Input Properties

Field Size

For a Short Text field, this property defines the maximum number of characters allowed (up to 255). For a Number field, it specifies the type of number (Integer, Double, etc.), which impacts the range of accepted values and the storage space. It is essential to choose the most appropriate size to optimize the database.

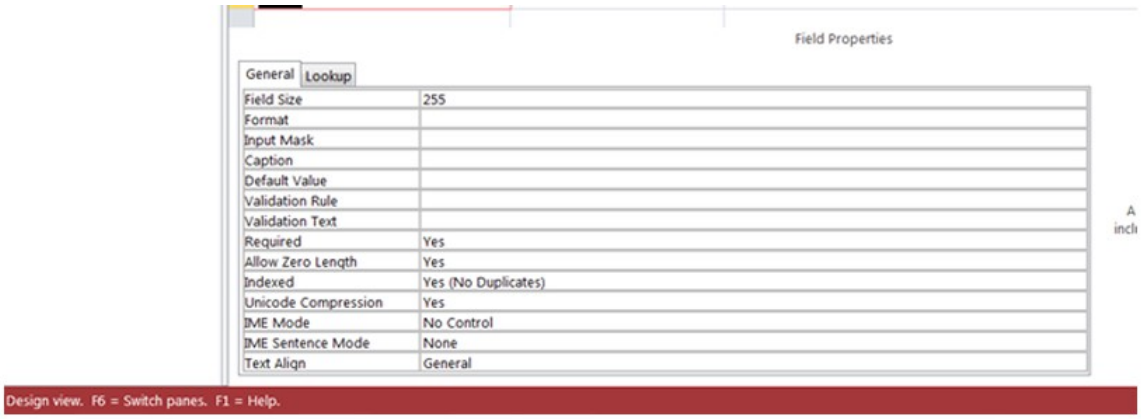


Figure 3.4: Field Properties[Scr]

Format

Controls the appearance of the data on display, without changing the data itself. For example, you can format a date as mm/dd/yyyy, display a number with two decimal places, or force text to display in uppercase.

Input Mask:

Guides the user by enforcing a strict entry format. This is very useful for phone numbers, postal codes, or any other standardized format. For example, an input mask for a phone number might look like (000) 000-0000.

Caption

Allows you to define a more explicit label for a field. This text will be used instead of the ”Field Name” (which often lacks spaces or accents) in column headers of tables, forms, and reports, thereby improving readability for the end-user.

3.3.2 Validation and Integrity Properties

Default Value

Specifies a value that will be automatically inserted into the field when a new record is created. This is useful for fields that often have the same value (e.g., the city for local customers).

Validation Rule

This is a validation rule that the entered data must meet to be accepted. For example, for a Quantity field, the rule could be > 0 to prohibit negative values. For an order date, it could be $<= Date()$ to prohibit future dates.

Validation Text

Displays a custom error message if the entered data does not comply with the rule defined in ”Validation Rule”. A clear message (e.g., ”The quantity must be greater than zero”) is more helpful to the user than Access’s standard error message.

Required

If this property is set to "Yes," the field cannot be left empty. The user is obligated to provide a value for this field when creating or modifying a record.

3.3.3 Performance Properties

Indexed

Indexing a field significantly speeds up search and sort operations on that field, much like a book's index helps to find a topic quickly. This is particularly important for foreign keys and fields frequently used in query criteria. The "Yes (No Duplicates)" option also ensures that each value in that field is unique throughout the entire table.

3.4 The Primary Key

In a relational database, every record within a table must be uniquely identifiable. This is the fundamental role of the primary key[Kin22, Mic25a].

3.4.1 Definition and Importance

A primary key is a field (or a set of fields) whose value allows each row in a table to be uniquely and reliably identified. It acts as a unique identification number for each record[Kin22, Mic25a].

Its importance is critical for several reasons:

Guaranteed Uniqueness

It prevents the creation of duplicate records. Access will never allow you to enter the same primary key value for two different records.

Basis for Relationships

The primary key is the pillar upon which relationships between tables are built. To link a "Customers" table to an "Orders" table, you use the primary key from the "Customers" table (e.g., CustomerID) as a reference in the "Orders" table (where it becomes a "foreign key")[Mic25a].

Performance

Access automatically creates an index on the primary key. This index dramatically speeds up data search, sort, and join operations[Mic25a].

3.4.2 Characteristics of a Good Primary Key

To be effective, a primary key field must follow three strict rules : [Mic25a]

1. **Unique**: Each record must have a different primary key value.
2. **Non-null**: A primary key can never be left empty (contain a Null value).

3. **Stable**: Its value should ideally never change once it has been assigned. Changing a primary key would require updating all references to that key in other tables, which is risky and complex.

3.4.3 How to Define a Primary Key in Access

There are two main approaches to choosing a primary key:

Use an existing field (natural key)

If you already have a unique and stable identifier in your data (e.g., a product number, an item code, a national insurance number), you can use it as the primary key. However, you must be absolutely certain that this identifier will never contain duplicates and will never be empty. Using a person's name, for example, is a bad idea because people can have the same name.

Create an artificial key with AutoNumber

This is the safest and most recommended method in most cases. By using the AutoNumber data type, you delegate the responsibility of generating a unique number to Access (usually an integer that increments by 1 for each new record). This key is called "artificial" or a "surrogate key" because it has no business meaning; its sole purpose is to identify the record. This guarantees that all three rules (uniqueness, non-null, and stability) are met.

To set a primary key in Design View:

- Select the field you want to use.
- On the Design tab of the ribbon, click the Primary Key icon. A small key icon will then appear to the left of the field name.

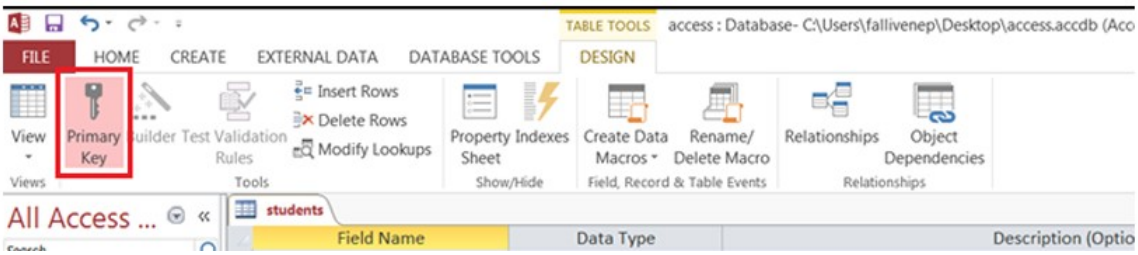


Figure 3.5: The Primary Key[Scr]

3.5 Establishing Relationships Between Tables

The true power of a relational database lies in its ability to link information distributed across different tables. Creating relationships allows you to bring this data together coherently, avoid redundancy, and ensure information integrity[Mic25i, Mic25e].

3.5.1 Why Create Relationships?

Once you have broken down your information into separate tables (one for customers, one for products, one for orders, etc.), relationships act as bridges between them. They allow Access to[Mic25i, Mic25e] :

- **Combine Data**: When you create a query, form, or report, relationships tell Access how to join records from the different tables. For example, to display a customer's name next to their order, Access uses the relationship between the Customers table and the Orders table.
- **Enforce Referential Integrity**:This is a crucial concept. By enforcing referential integrity, you are asking Access to ensure that the relationships between records remain valid. For example, it will prevent a user from creating an order for a customer that does not exist, or from deleting a customer who already has recorded orders.

3.5.2 The Relationships Window

The main tool for managing the links between tables is the Relationships Window. To access it, go to the Database Tools tab and click on Relationships.

3.5.3 How to Create a Relationship?

The process is simple and visual :

1. **Add Tables**: If the window is empty, Access will prompt you to add tables. Otherwise, click "Add Tables" on the ribbon. Select the tables you want to link and add them to the window.
2. **Identify Common Fields**: The link is made by connecting the primary key of one table to a corresponding field in another table. This corresponding field is called a foreign key. For example, you will link the CustomerID field (primary key of the Customers table) to the CustomerID field (foreign key of the Orders table).
3. **Create the Link via Drag-and-Drop**: Click on the primary key field (the "one" side of the relationship) and, without releasing the mouse button, drag it to the foreign key field (the "many" side) in the other table.
4. **Configure the Relationship**: An "Edit Relationships" dialog box will appear.
   - Verify that the two fields are correct.
   - Crucially, check the "Enforce Referential Integrity" box. This is what gives the relationship its strength.
   - You can also check "Cascade Update Related Fields" (if the primary key changes, the foreign key is updated everywhere) and "Cascade Delete Related Records" (if you delete a customer, all their orders are deleted; use with caution).
5. **Click "Create"**: Access will then draw a join line between the two tables, with a "1" symbol on the primary key side and an "∞" (infinity) symbol on the foreign key side, thus materializing a one-to-many relationship[Mic25i, Mic25e].

Once the relationships are established, the structure of your database is solid. Access will then be able to intelligently assist you in creating complex queries, forms, and reports.

3.6 Examples: Creating and Relating Tables from the Conceptual Data Model (MCD)

At the end of this chapter, let us consolidate our understanding of database design in Access with a practical example. We will use the library system conceptual data model (MCD) introduced in Chapter 1 (see diagram below):

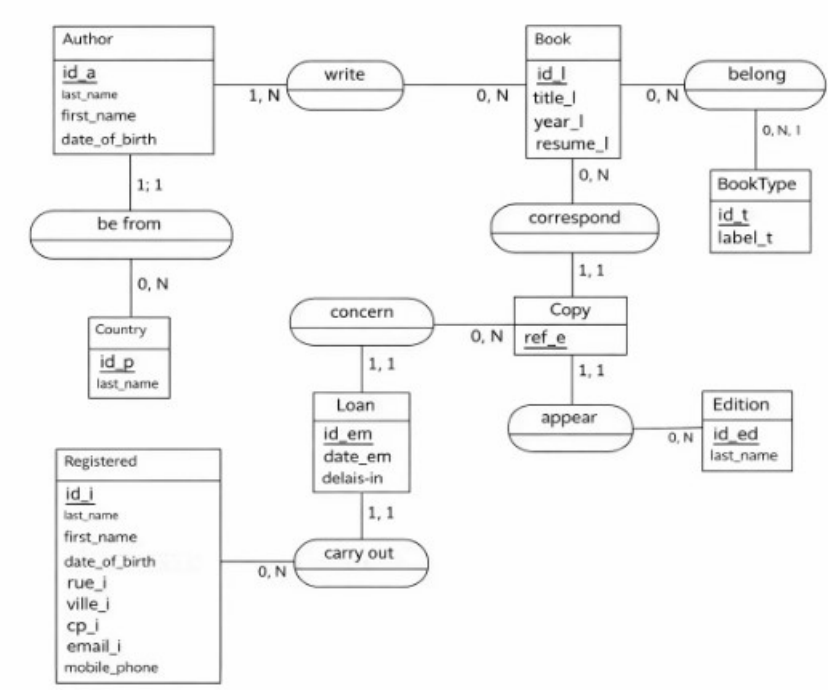


Figure 3.6: Example 01 of MCD[Ben]

3.6.1 Step 1: Table Creation in Access

The first stage is to create Access tables faithfully reflecting the entities and attributes in the MCD (from figure 3.7 to 3.14):

Author

Nom du champ	Type de données
id_a	Numérique
name_a	Texte court
Fname_a	Texte court
birth_date_a	Date/Heure

Figure 3.7: The "Author" Table[Scr]

Book

Nom du champ	Type de données
id_b	Numérique
title_b	Texte court
year_b	Numérique
summary_b	Texte long

Figure 3.8: The "Book" Table[Scr]

3.6.2 Step 2: Defining Relationships

Once all tables have been created, the next step is to establish relationships as shown in the MCD diagram (Figure 3.6):

Book | Book Type

Nom du champ	Type de données
id_t	Numérique
label_t	Texte court

Figure 3.9: The "Book Type" Table[Scr]

Copy book

Nom du champ	Type de données
ref_c	Numérique

Figure 3.10: The "Copy Book" Table[Scr]

Country

Nom du champ	Type de données
id_c	Numérique
name_c	Texte court

Figure 3.11: The "Country" Table[Scr]

Edition

Nom du champ	Type de données
id_ed	Numérique
name_ed	Texte long

Figure 3.12: The "Edition" Table[Scr]

Loan

Nom du champ	Type de données
id_l	Numérique
date_l	Date/Heure
duration_l	Numérique

Figure 3.13: The "Loan" Table[Scr]

Registered

Nom du champ	Type de données
id_r	NuméroAuto
name_r	Texte court
Fname_r	Texte court
street_r	Texte long
city_r	Texte court
zcode_r	Numérique
phone_r	Numérique
mob_phone_r	Numérique
email_r	Texte court
birth_date_r	Date/Heure

Figure 3.14: The "Registred" Table[Scr]

1. Use the Relationships window in Access (Database Tools ¿ Relationships) $\Longrightarrow$ Figure 3.15.

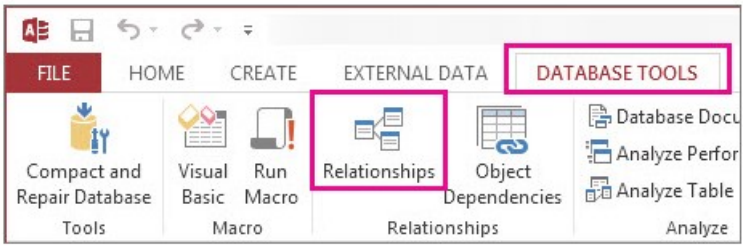


Figure 3.15: Relationships window 1[Scr]

2. Add all relevant tables to the window $\Longrightarrow$ Figure 3.16.
3. Drag primary keys to corresponding foreign keys to create relationships:
   - For example, link Author.id_a to the linking field in the associative table for the write relationship (if modeled by a join table), or directly to Book if using a foreign key.
   - Similarly, relate BookType, Edition, Country, Registered, and Copy according to the design.
4. Enforce Referential Integrity for each relationship to maintain data coherence (prevent orphaned records).

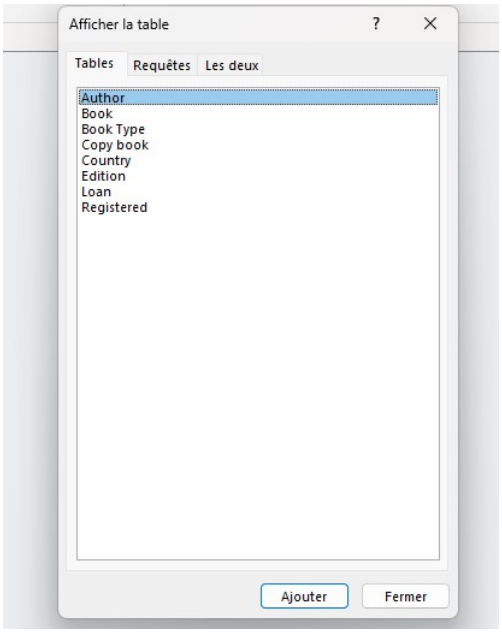


Figure 3.16: Relationships window 2[Scr]

3.6.3 Step 3: Example Relationship

1. For example, the relationship between Book and Copy is **one-to-many** (Figures 3.17, 3.18):

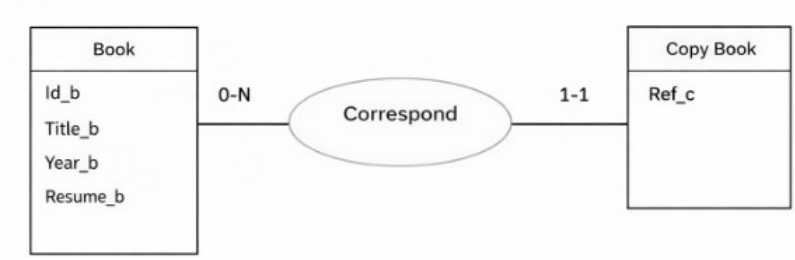


Figure 3.17: Relationships One-to-Many[Scr]

- Each Book can have multiple Copy records (Book.id_l → Copy.id_l as foreign key).
- In Access, this is established by linking the primary key in the Book table to the foreign key in the Copy table.

2. **Many-to-Many** Relationship Example (Author and Book) (Figures 3.19, 3.20):
   Now, let's illustrate a many-to-many relationship using the example of Authors and Books. In our model, each book can be written by several authors and each author can write several books.
   This relationship cannot be directly represented with just two tables; instead, we need an intermediate (junction) table to connect them.

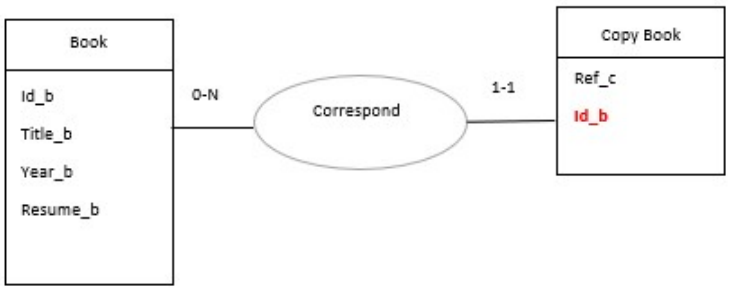


Figure 3.18: Relationships One-to-Many with the Foreign Key[Scr]

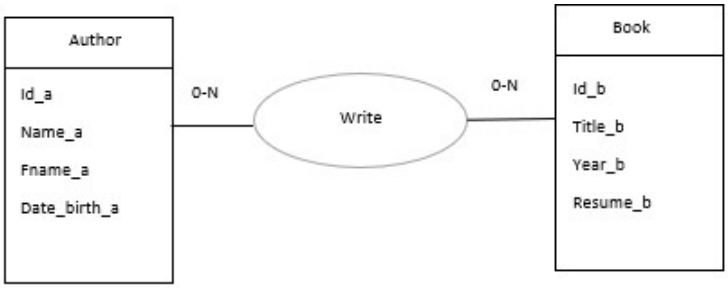


Figure 3.19: Relationships Many-to-Many[Scr]

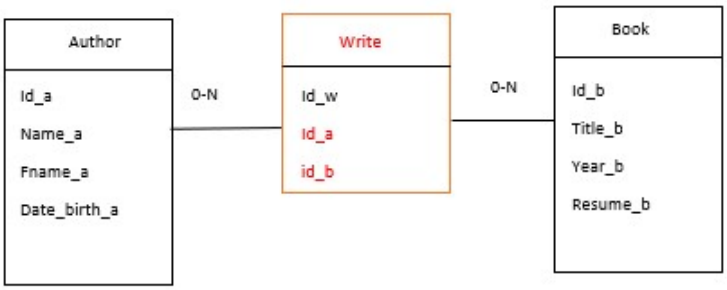


Figure 3.20: Relationships Many-to-Many with the Foreign Key[Scr]

3.6.4 Step 4: Visualizing Relationships

After creating all necessary links, the relationship diagram in Access will reflect the MCD, showing ”1” and ”∞” symbols just like the original conceptual design. This structure enables consistency checks, cascade options, and efficient querying.

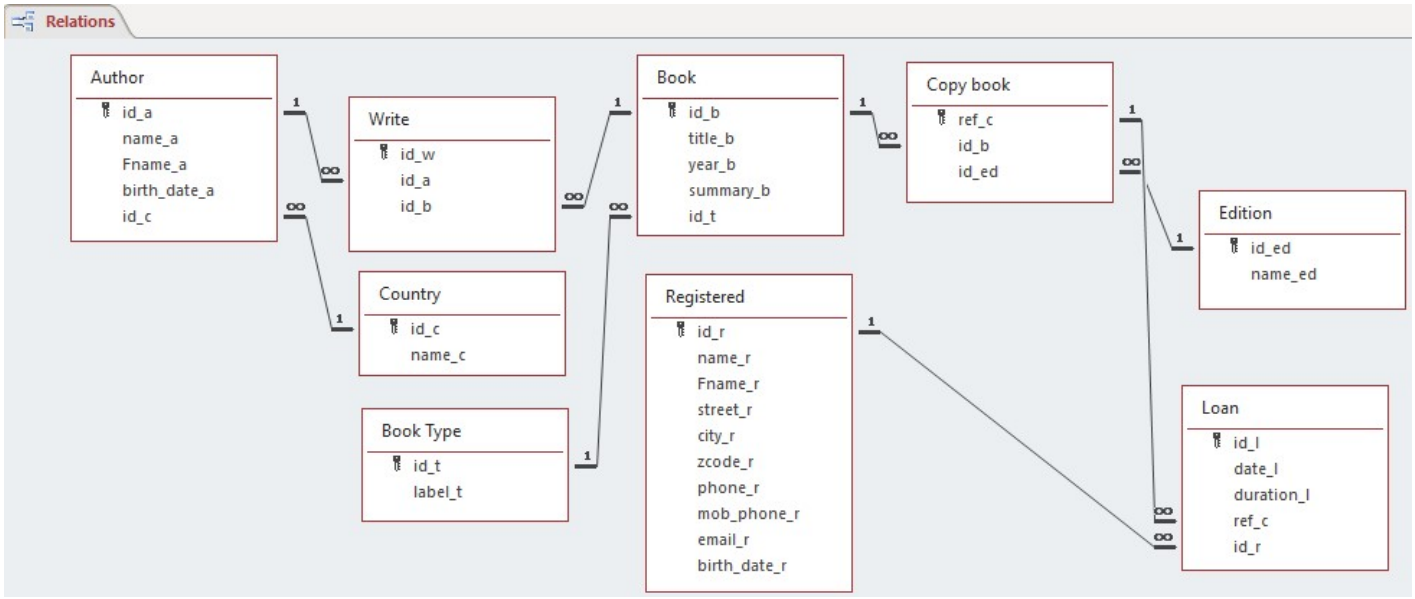


Figure 3.21: Visualizing Relationships[Scr]

Chapter 4

Create Forms

Contents

If tables are the foundation of the database, storing the information, then forms are the visible and interactive facade. They are the primary intermediary between the user and the raw data, and their proper design is essential to ensure an efficient, pleasant, and secure user experience.

4.1 Role and Utility of Forms

A form in Access is a database object that you create to serve as a user interface. Rather than forcing users to navigate the data grids of tables (which can be intimidating and a source of errors), forms offer a structured and user-friendly presentation of the information[Mic25q].

Their main roles are as follows:

4.1.1 Simplify Data Entry and Modification

The primary goal of a form is to facilitate the entry of new information and the updating of existing records. A well-designed form can[Mic25q]:

- **Guide the user**: By presenting fields in a logical order, often similar to that of a paper document (a customer file, an order form).
- **Display one record at a time**: This allows the user to focus on a single entity (a customer, a product) without being overwhelmed by hundreds of rows.
- **Use intuitive controls**: Replace manual entry with drop-down lists, check boxes, or calendars to speed up work and reduce typing errors[Joh25].

4.1.2 Control Access and Interaction with Data

Forms are an excellent way to control what users can see and do. For example, you can[Mic25q]:

- **Display only the necessary fields**: Hide technical fields (like IDs) or confidential information.
- **Lock certain fields**: Allow the viewing of some information (like a creation date) while preventing its modification.
- **Create specific views**: Design a form for the sales department that shows only sales information, and another for the accounting department with billing data.

4.1.3 Automate Actions and Create an Application

Forms do more than just display data; they can be enhanced with ”active controls” like command buttons. These buttons allow complex tasks to be automated with a single click:

- Add a new record.
- Print an invoice.
- Search for a customer.
- Open another form or a report.

By combining several forms (a main menu, an entry form, a search form, etc.), you are no longer building just a simple database, but a complete and professional management application.

4.2 Creation Methods

Access, in its goal to be both accessible for beginners and powerful for developers, offers several methods for creating a form. The choice of method will depend on the level of control you want over the final result, your comfort with the tool, and the complexity of the form you need to create.

All these methods are located on the Create tab of the ribbon, in the Forms group.

4.2.1 The "Form" Tool (Instant Creation)

This is the fastest and simplest method. It is ideal for creating a basic data entry form with a single click[Mic25b].

- **How to do it?** In the Navigation Pane, select the table or query that will contain the form's data, then click the Form icon.

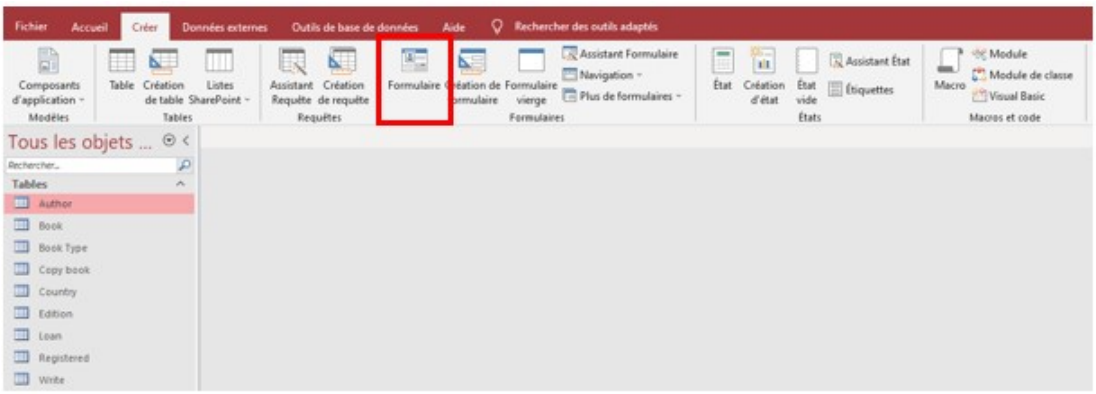


Figure 4.1: Create Form (Instant Creation) Exemple-1-[Scr]

- **Result**: Access instantly creates a form in "Layout View," displaying all the fields from the data source, one below the other (as show in Figure 4.2). If relationships exist, Access may even automatically create a subform to display the related data[Mic25b].
- **Use**: Perfect for quickly generating a draft form that you can then refine in Design View.

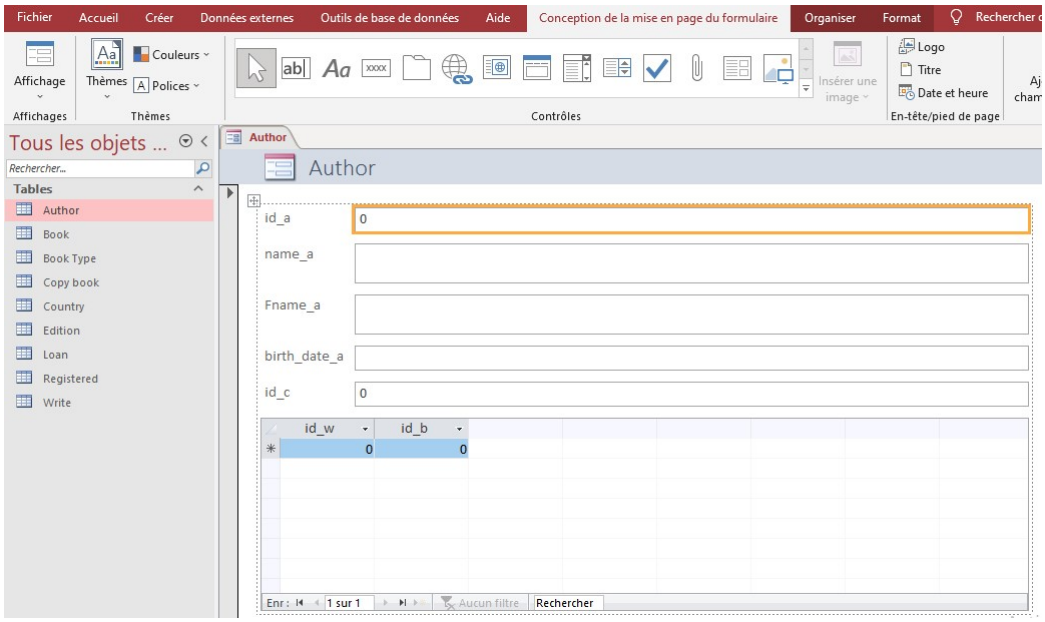


Figure 4.2: Create Form (Instant Creation)-result[Scr]

4.2.2 The Form Wizard

The Wizard is an excellent compromise between the simplicity of instant creation and the complexity of Design View. It guides you step-by-step through a series of dialog boxes to customize your form.

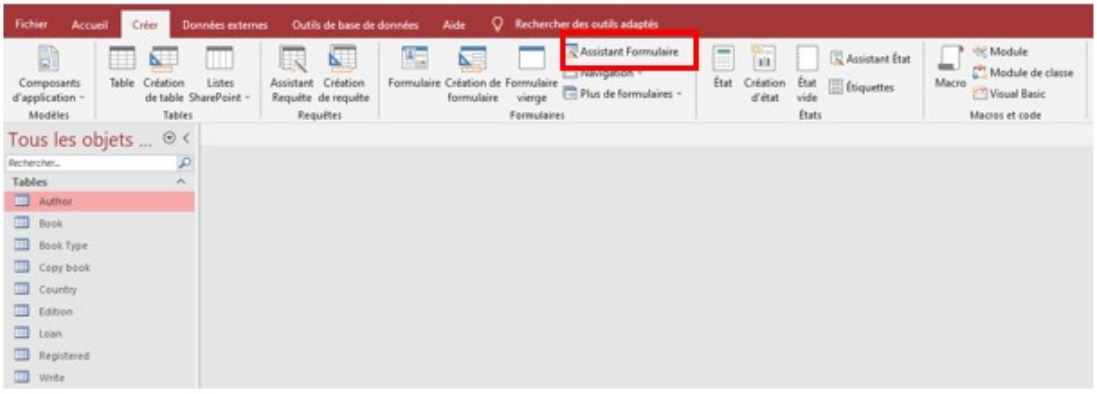


Figure 4.3: Create Form Wizard Exemple-2-[Scr]

- **How to do it?** Click on Form Wizard (as show in Figures 4.3 and 4.4).
- **Process**:
  (a) **Field Selection**: You select the source table or query, then precisely choose the fields you want to include in the form. You can even pick fields from multiple tables (if they are related).
  (b) **Layout**: You choose how the fields will be arranged (Columnar, Tabular, etc.).

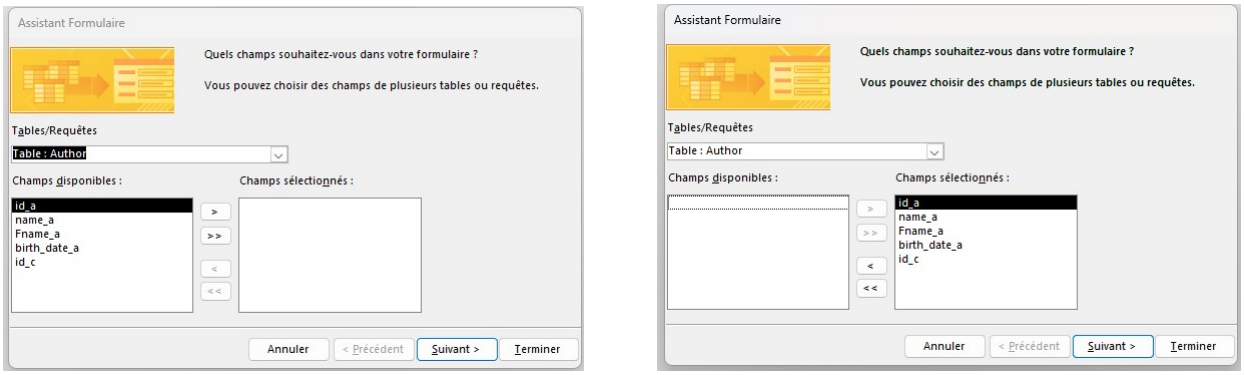


Figure 4.4: Create Form Wizard Exemple-2-[Scr]

(c) **Style**: You can apply a predefined visual style to give your form a professional look.

(d) **Name**: You give your form a name.

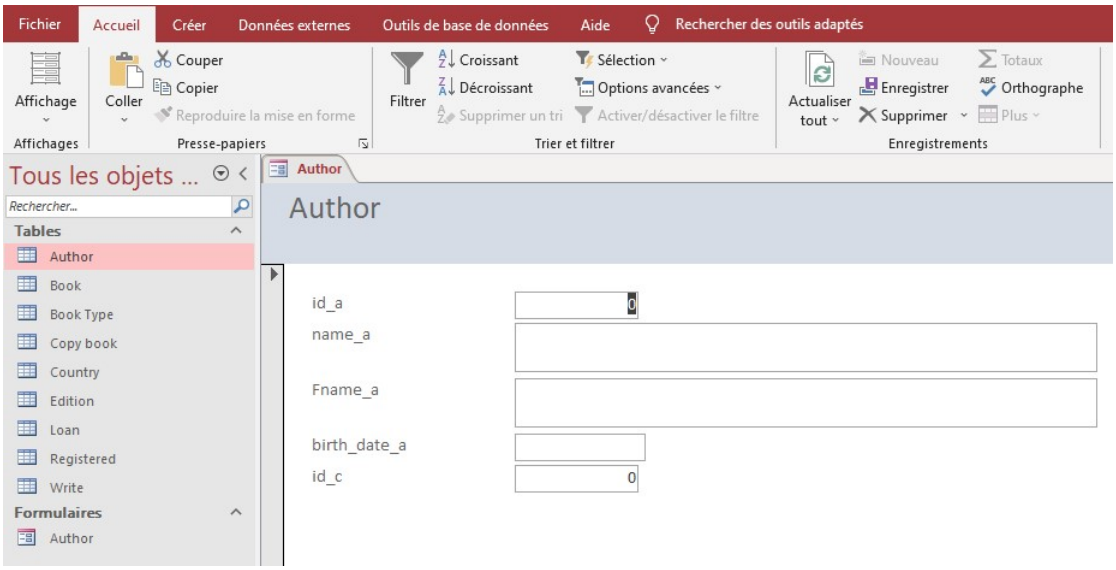


Figure 4.5: Create Form Wizard Result[Scr]

- **Use**: This is the most versatile method for most common needs, as it offers a good level of customization while being fully guided.

4.2.3 Design View and Blank Form

These two methods offer complete control over every aspect of the form. They are intended for more advanced users who want to design a custom interface[Mic25b].

- **Blank Form**: Access opens a blank form in "Layout View" and displays the Field List pane. You can then simply drag the fields of your choice from this list onto the form to add them. It's a visual and intuitive approach[Mic25b].
- **Design View**: This is the designer's blank canvas. You start with a completely empty design grid, with no link to any data source initially. You must:

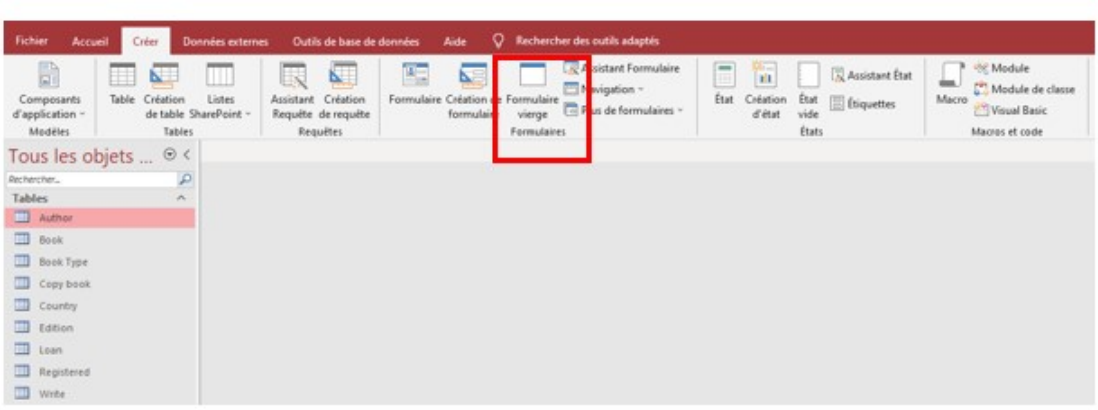


Figure 4.6: Create Blank Form Exemple-3-[Scr]

(a) Define the form's data source in its property sheet.

(b) Manually add each control (text boxes, labels, buttons, etc.) from the "Design" tab.

(c) Bind each control to its corresponding field.

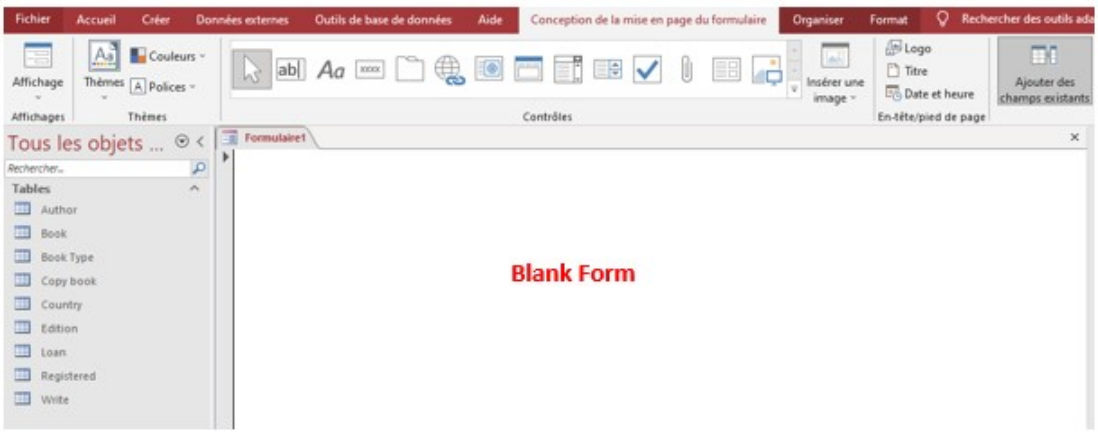


Figure 4.7: Create Blank Form Result[Scr]

- **Use**: Essential for creating complex forms, navigation menus, custom dialog boxes, or any interface that does not fit a standard template. For a progressive learning path, it is recommended to start with the "Form" tool and the Wizard, and then switch to Design View to adjust and perfect the forms generated this way.

4.3 Form Controls

Controls are the building blocks of your forms. They are the graphical objects you place on a form to display data, perform actions, or simply decorate the interface (lines, boxes, images). Mastering the different types of controls and knowing when to use them is the key to creating forms that are both functional and user-friendly[Mic25l].

There are three main families of controls :

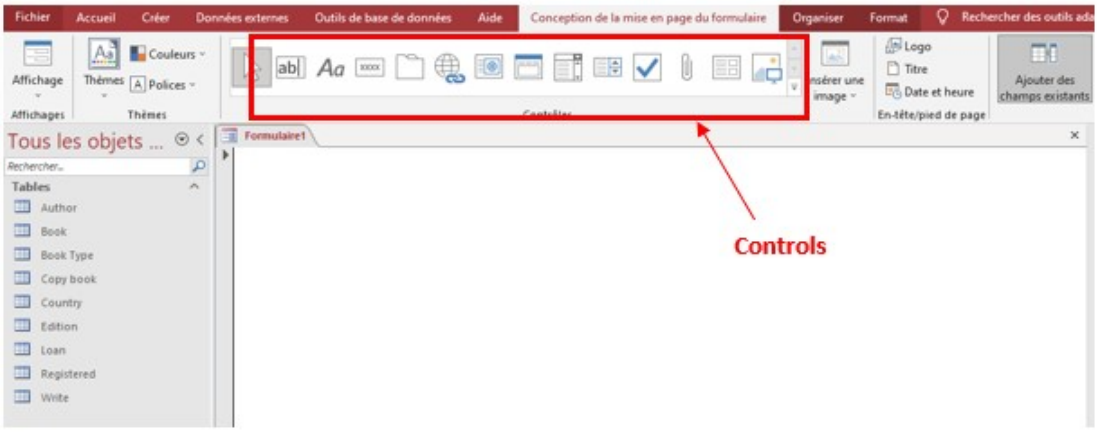


Figure 4.8: Form Controls[Scr]

(a) **Bound Controls**: A control is ”bound” when its data source is a field in a table or query. This is the most important type of control as it creates a direct link between the form and your data. Anything you enter into a bound control is automatically saved in the corresponding field of the underlying table[Mic25l].

(b) **Unbound Controls**: These controls have no data source. They ”live” only on the form. They are used to display static information (like the form's title or instructions) or graphical elements[Mic25l].

(c) **Calculated Controls**: The data source for these controls is an expression, not a field. An expression is a formula that performs a calculation. For example, a calculated control could display the total amount of an order by multiplying Quantity by UnitPrice. The value is displayed on the form but is not stored in a table[Mic25l].

4.3.1 Main Types of Controls

Here are the most commonly used controls, which can be found on the Design tab when the form is in Design View:

- **Text Box**: This is the most versatile control. It can be used to display text, numbers, dates, etc. It can be bound (to display and edit data), unbound (rare), or calculated.
- **Label**: This is an unbound control used to display static text. Each bound text box is typically accompanied by a label that indicates the field's name[Mic25l].
- **Command Button**: This is a control that, when clicked, executes an action. With the Command Button Wizard, you can easily create buttons to navigate between records, open another form, print a report, or close the application, without writing a single line of code.
- **Combo Box**: This very powerful control combines a text box and a drop-down list. It allows the user to either type a value or select it from a predefined list. It is the ideal control for managing foreign keys. For example, in an order form, a combo box can display customer names while storing the selected customer's ID in the orders table.

- **Check Box**: This is the standard control for Yes/No fields. It provides a clear and quick way to represent a Boolean value (true/false, enabled/disabled).
- **Option Group**: Allows the user to choose a single option from a limited set of mutually exclusive choices (for example, a status of "In Progress," "Completed," or "Canceled").

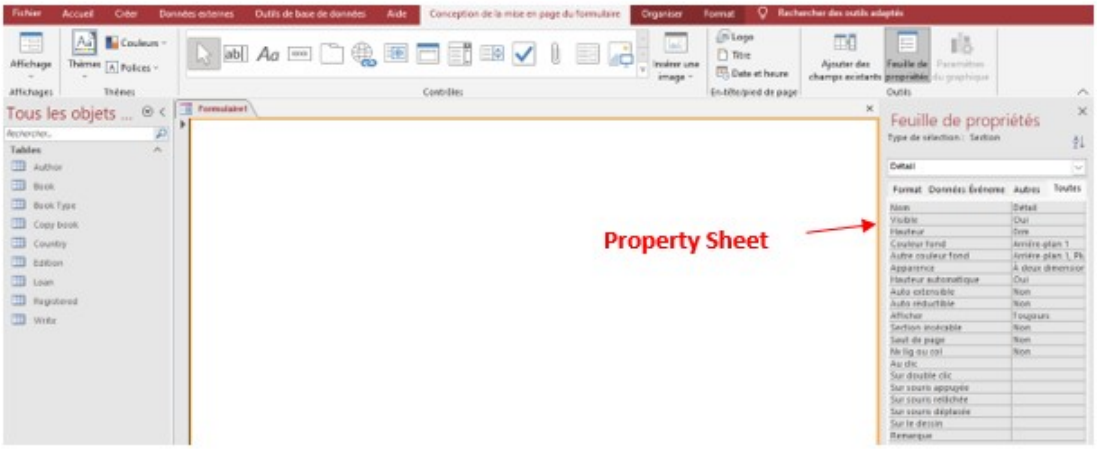


Figure 4.9: Property Sheet of Controls[Scr]

Each control has its own Property Sheet (accessible with the F4 key) that allows you to customize its appearance (color, font), behavior (locked, disabled), and data source in detail.

4.4 Improving Ergonomics

A functional form is good, but an ergonomic form is even better. Ergonomics aims to make the use of the interface as simple, efficient, and pleasant as possible for the end user. A well-thought-out form reduces training time, minimizes data entry errors, and increases overall productivity.

Here are several key techniques to improve the ergonomics of your forms in Design View:

(a) **Structure and Layout of Controls**

- **Logical Organization**: Group related fields together. For example, on a customer form, group identity information (Name, First Name), then contact details (Address, City, Phone), and finally commercial information. Using frames or the Tab Control object is excellent for visually structuring these groups.
- **Alignment and Spacing**: Use Access's alignment tools (available via a right-click or in the ribbon) to ensure your controls are perfectly aligned vertically and horizontally. Consistent spacing between controls gives an impression of professionalism and clarity.

- **Anchoring**: By default, controls do not move when the form window is resized. The anchoring property allows you to ”tie” a control to the edges of the form. For example, you can make a ”Comments” field expand along with the window, or keep a ”Close” button always in the bottom-right corner[Mic25p].

(b) **Guiding the User and Simplifying Navigation**

- **Tab Order**: When the user presses the Tab key to move from one field to another, the order of movement must be logical. This order is defined by the Tab Index property of each control. You must ensure that this numerical order matches the visual order of the fields on the form. You can easily reorganize this order by going to the Design tab and clicking on Tab Order[Luk25].
- **Hiding Unnecessary Elements**: A form intended for the end user should not display Access's default navigation elements, which can be confusing. In the form's properties, you can disable the Record Selectors and Navigation Buttons if you have already created your own navigation buttons.
- **Adding Headers and Footers**: The Form Header is the ideal place for a clear title and a logo. The Form Footer can contain common command buttons (like ”Save,” ”Cancel,” ”Close”) that will always be visible.

(c) **Providing Visual Feedback**

- **Conditional Formatting**: This powerful feature allows you to change the appearance of a control (background color, font, etc.) based on its value. For example, you can color a stock quantity in red if it falls below a certain threshold, or highlight an unpaid invoice.
- **Locking Fields**: To prevent accidental modification, fields that should not be changed (like an ID or a creation date) should have their Locked property set to ”Yes.” The mouse cursor will not even change to text-entry mode over these fields, clearly indicating to the user that they are not editable.

Chapter 5

Create Queries

Contents

After structuring the data in tables and creating forms for data entry, it's time to leverage the true intelligence of our database: **queries**. Queries are the primary tool for interrogating, analyzing, and manipulating the stored information, thereby transforming a simple collection of data into a source of answers.

5.1 Principles of Queries in Access

A query is, fundamentally, a **question asked of the database**. It is a formal request to extract specific information, perform calculations, or even modify data in bulk. In a well-designed database where information is spread across multiple tables, queries are essential for gathering and making sense of this dispersed data[Reg].

5.1.1 The Central Role of Queries

Queries are versatile and serve multiple purposes:

Finding and Filtering Data

This is their most common role. A query allows you to quickly find specific records by applying filters (called criteria).

Example: "Show all customers living in Paris."

Combining Data from Multiple Tables

Thanks to the relationships you have established, a query can pull fields from different tables and present them as a single, coherent set[Reg].

Example: "Show the name of each customer next to the name of the product they ordered."

Calculating and Summarizing Information

Queries do more than just extract data; they can also perform calculations on it. You can create new calculated fields or use aggregate functions (sum, average, count, etc.)[Reg].

Example: "Calculate the total revenue for each customer."

Serving as a Source for Forms and Reports: A form or report does not have to be based directly on a table. It is often more powerful to base it on a query. This allows you to display data that is already sorted, filtered, or calculated, making the forms and reports much more dynamic.

5.1.2 The Query Design Interface

When you create a query in "Design View" in Access, you work with a graphical grid that largely spares you from having to write SQL code (the standard language for databases). This grid allows you to:

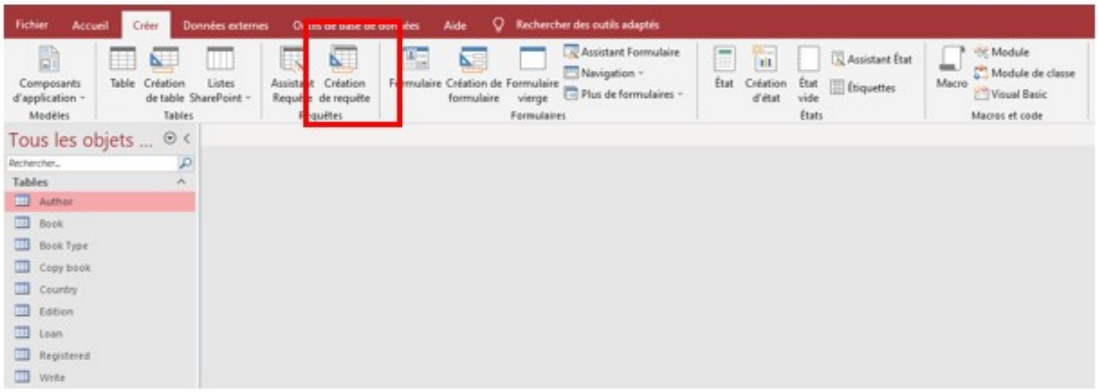


Figure 5.1: Create Queries[Scr]

(a) Choose the tables and/or queries that contain the necessary data.

(b) Select the fields you want to appear in the result.

(c) Define sort orders (ascending, descending) for one or more fields.
filter criteria to display only the records that interest you.

(d) Create calculated fields by entering expressions.

In short, queries are the engine of your database, transforming raw data into relevant and actionable information.

5.2 The Different Types of Queries

Access is not limited to a single type of query. Depending on our objective—whether it's to view, analyze, or modify data—you will use different types of queries. They can be classified into two main families: queries that display data (select queries) and those that perform actions on data (action queries)[Mic25n].

5.2.1 Select Queries

This is the most common and fundamental type of query. Its role is to extract and display data from one or more tables, without ever modifying the source data. The result of a select query is a dynamic datasheet that you can use as the basis for a form or a report[Mic25n].

Select queries come in several powerful variations:

Queries with Criteria

Allow you to filter data to display only the records that match one or more conditions (e.g., show customers from the city of "Paris" whose sales are greater than €1000), you use comparison operators (Table 5.1) and other logical operators (Table 5.2).

Table 5.1: Comparison Operators

Operator	Meaning
=	Equal
¡¿	Not Equal
¡	Less Than
¿	Greater Than
¡=	Less Than or Equal To
¿=	Greater Than or Equal To

Table 5.2: Logical Operators and Their Summarized Meanings

Logical Operator	Description	Example
`AND`	Returns TRUE if both operands are true. Symbol: `&`.	`[quantity] > 100 AND [location] = "Philadelphia"`
`OR`	Returns TRUE if at least one operand is true. Symbol: `|`.	`[quantity] > 100 OR [location] = "Philadelphia"`
`EQV`	Returns TRUE if both operands have the same value. Symbol: `<=>`.	`([A] > 10) EQV ([B] = 0)`
`NOT`	Inverts the logical value of an operand. Symbol: `~`.	`NOT [location] = "Philadelphia"`
`XOR`	Returns TRUE if operands have different values. Symbol: `^`.	`([A] > 10) XOR ([B] < 5)`
`IMP`	Implication: FALSE only if the first operand is TRUE and the second is FALSE. Symbol: `->`.	`([A] > 10) IMP ([B] = 0)`

Queries with Calculations

Allow you to create new calculated fields that do not exist in the tables. For example, a *PriceWithTax* field calculated from a *PriceWithoutTax* field (*PriceWithTax: [PriceWithoutTax] * 1.2*).

Table 5.3: Access SQL Operators and Their Meanings

Operator	Meaning	Example
Between	Selects records where the field value is within a range of values.	Between 10 And 20
In	Selects records where the field value is included in a list.	In (“Paris”,”Lyon”)
Is	Selects records where a field is empty or not.	Is Null
Like	Selects records containing approximate data.	Like “street*”
Not	Selects records that do not match the criteria.	Not In (“Paris”,”Lyon”)

Totals Queries (or Summary Queries)

Use aggregate functions (*Sum, Avg, Count, Min, Max*) to summarize data (Table 5.4). This is ideal for obtaining statistics (e.g., calculating the number of orders per customer, or total sales per month).

Table 5.4: Aggregate Operations and Their Meanings

Operation	Meaning
Count	Counts the number of values
Last	Value of the last record
StDev	Standard deviation
Max	Highest value
Min	Lowest value
Avg	Average
First	Value of the first record
Sum	Total
Var	Variance

Parameter Queries

These are interactive queries that display a dialog box asking the user to enter a value (a parameter) before they run. For example, the query might ask ”Enter a start date?” to show only orders placed after that date.

5.2.2 Action Queries

Unlike select queries, action queries modify or delete data in bulk. They are extremely powerful but must be used with the utmost caution, as their actions are irreversible.[Mic25n].

There are four types of action queries :

- **Make-Table Query**: Creates a new table based on the results of a select query. This is very useful for archiving data or creating a working copy.
- **Append Query**: Adds records to the end of an existing table. This is perfect for populating an archive table from a current data table.
- **Update Query**: Modifies data in existing records. This is the fastest way to make a change in bulk (e.g., increasing the prices of all products in a certain category by 10
- **Delete Query**: Deletes entire records from one or more tables based on criteria. This is a very dangerous operation that should be preceded by a backup.

5.2.3 Crosstab Query

his specialized query type displays data in a two-dimensional grid, similar to a PivotTable in Excel. It allows you to summarize data by grouping it along two axes (row headings and column headings) and displaying a calculated value (sum, average, etc.) at the intersection.

Example: Display the total sales (value) for each *Product* (row headings) and for each *Month* (column headings).

5.3 Using Query Design View

While the Query Wizard is useful for simple needs, Design View is the environment where you will have the most control and flexibility to build queries, whether they are simple or complex.

To access it, go to the Create tab and click Query Design. The interface that opens is composed of two main areas (Figure 5.2):

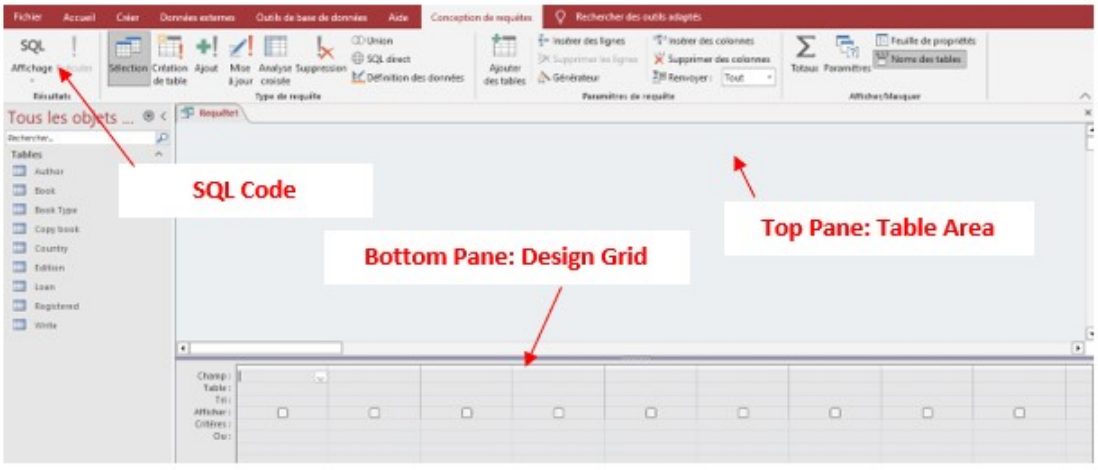


Figure 5.2: Query Design Interface[Scr]

5.3.1 The Top Pane: The Table Area

This is where you define your data sources.

- **Add Tables**: A "Show Table" dialog box appears initially, prompting you to add the tables and/or queries you need. You can add more later by right-clicking or using the "Add Tables" button on the ribbon.
- **Visualize Relationships**: If relationships exist between the added tables, Access automatically displays them as join lines. This is what allows the query to correctly link records together.

5.3.2 The Bottom Pane: The Design Grid (QBE - Query By Example)

This is the heart of query construction. It is a grid where each column represents a field that will be used in the query. For each field, several rows allow you to specify its role.

Here are the most important rows in the design grid:

- **Field**: Displays the field name. You can add a field by dragging it from a table in the top pane or by selecting it from the drop-down list in this row. This is also where you create calculated fields (e.g., PriceWithTax: [Quantity] * [UnitPrice]).
- **Table**: Indicates the field's source table. This is useful when you are using multiple tables.
- **Sort**: Allows you to sort the query results based on this field, in Ascending or Descending order.
- **Show**: A checkbox that determines whether the field should be visible in the final query result. You might need to use a field for a filter criterion without wanting to display it.
- **Criteria**: This is arguably the most powerful row. This is where you enter the filtering conditions to keep only the records you are interested in.
  - "London": to keep only records where the City field is equal to London.
  - >100: to keep only values greater than 100.
  - Between #01/01/2024# And #12/31/2024#: to filter on a date range.
- **Or**: The rows below "Criteria" allow you to define alternative conditions. A criterion on the "Criteria" line and another on the "*or*" line will be linked by a logical OR (e.g., show customers from "London" OR those whose sales are > 5000). Criteria on the same line are linked by a logical AND (e.g., customers from "London" AND whose sales are > 5000).

Once your query is built, you can view the result at any time by clicking the Run button (for action queries) or by switching to Datasheet View (for select queries) from the ribbon.

5.4 Practical Query Examples

To bring the theoretical concepts discussed earlier to life, there is nothing better than hands-on practice. This section presents a series of query examples applied to the library management database model developed in Chapter 3. Each example illustrates a specific type of query and includes its solution in SQL, which is the universal database language that Access uses in the background. These examples will serve as a starting point for building your own queries.

5.4.1 Select Queries

These queries allow for the extraction and display of data without modifying it.

Query with Criteria

Objective 1: Display all books published after the year 2000.

Graphical solution 1 in Figure 5.3:

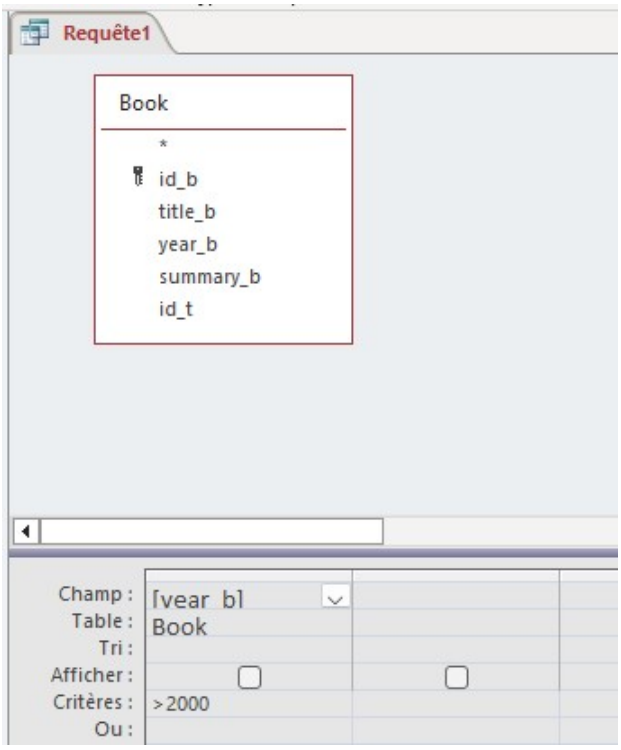


Figure 5.3: Query with Criteria Exemple 1[Scr]

SQL code 1:

```
SELECT *
FROM Book
```

WHERE year_b > 2000;

Objective 2: Find all registered members who live in ”Paris”.

Graphical solution 2 in Figure 5.4:

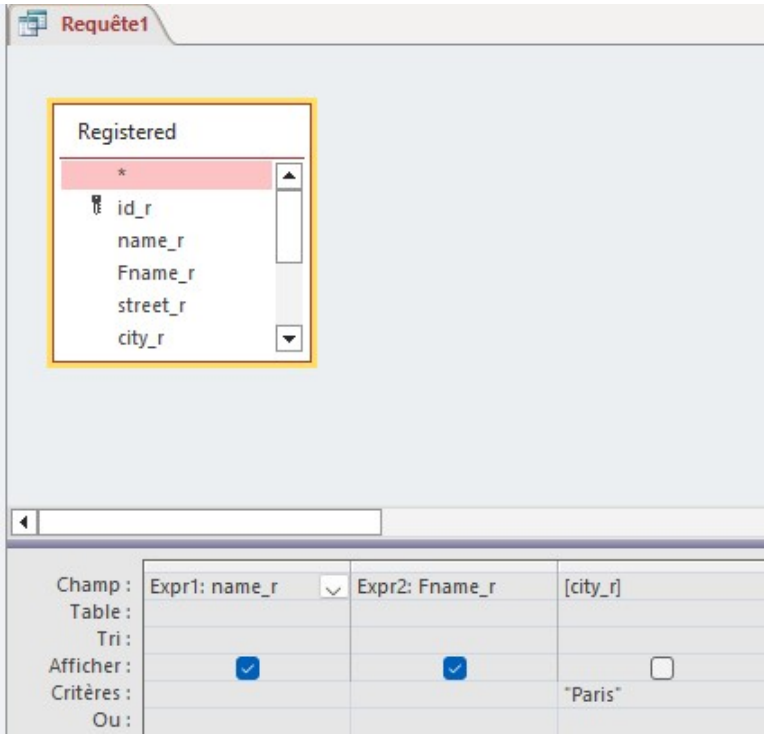


Figure 5.4: Query with Criteria Exemple 2[Scr]

SQL code 2:

SELECT name_r, Fname_r
FROM Registred
WHERE city_r = ”Paris”;

Query with Calculations

Objective 1: Display the title of each book along with its approximate age in years.

Graphical solution 1: (As show in Figure 5.5)

SQL code 1:

SELECT
title_b,
(Year(Date()) - year_b) AS BookAge FROM Book;

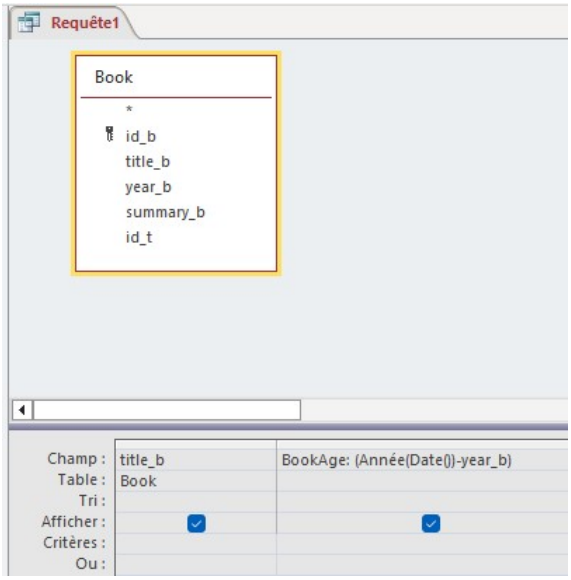


Figure 5.5: Query with Calculations Exemple 1[Scr]

Objective 2: Calculate the total amount of an order (fictional, by adding a Price field to the Book table and a Quantity field to the Loan table).

Graphical solution 2 in Figure 5.6:

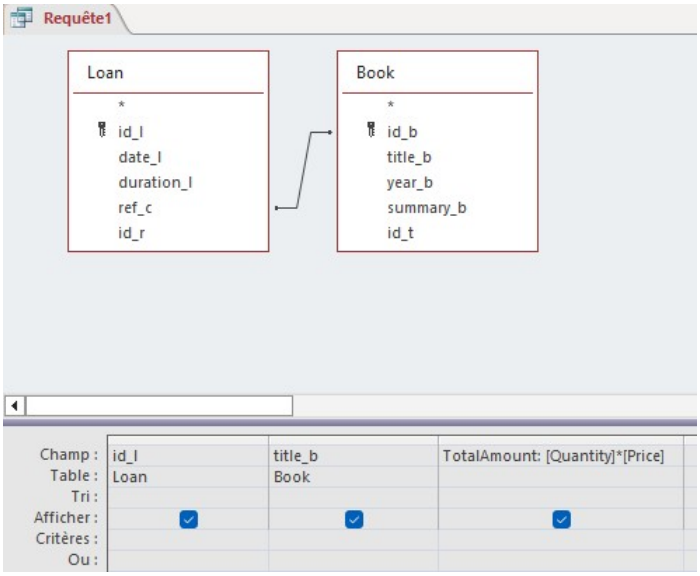


Figure 5.6: Query with Calculations Exemple 2[Scr]

SQL code 2:

```
SELECT
Loan.id_l,
Book.title_b, [Quantity] * [Price] AS TotalAmount FROM Loan INNER JOIN Book ON
Loan.ref_c = Book.id_b;
```

Totals (Summary) Query

Objective 1: Count the number of books written by each author.

Graphical solution 1:(As show in Figure 5.7)

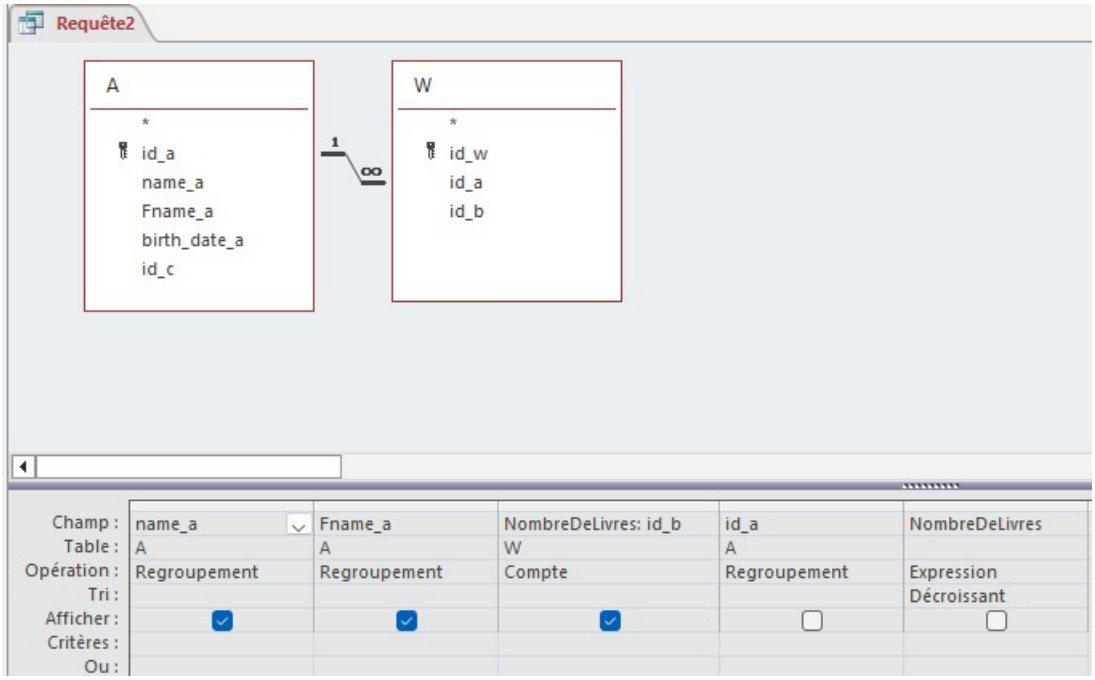


Figure 5.7: Totals (Summary) Query Exemple 1[Scr]

SQL code 1:

```
SELECT
A.name_a, A.Fname_a, COUNT(W.id_b) AS NombreDeLivres
FROM
Author AS A
INNER JOIN
Write AS W ON A.id_a = W.id_a
GROUP BY
A.id_a, A.name_a, A.Fname_a
ORDER BY
NombreDeLivres DESC;
```

To answer this query, you need to join the Author table with the ”Write” junction table.

Explanation of the query

(a) SELECT A.name_a, A.Fname_a, COUNT(W.id_b) AS NumberOfBooks:

- We select the author’s last name and first name from the Author table (aliased as A).

- We use COUNT(W.id_b) to count the number of books (id_b) associated with each author in the Write junction table (aliased as W).
- AS NumberOfBooks renames the count column for clarity.

(b) FROM Author AS A INNER JOIN Write AS W ON A.id_a = W.id_a:

- We start with the Author table.
- We perform a join (INNER JOIN) with the Write table on the common key id_a. This allows us to link each author to the records of the books they have written.

(c) GROUP BY A.id_a, A.name_a, A.Fname_a:

- Since we are using an aggregate function (COUNT), we must group the results by author. Grouping by the author's ID (id_a) ensures that each author is unique, even if multiple authors have the same name.

(d) ORDER BY NumberOfBooks DESC:

- This optional line sorts the results to display the authors who have written the most books first.

Objective 2: Calculate the total number of loans made by each registered member.

Graphical solution 2: (As show in Figure 5.8)

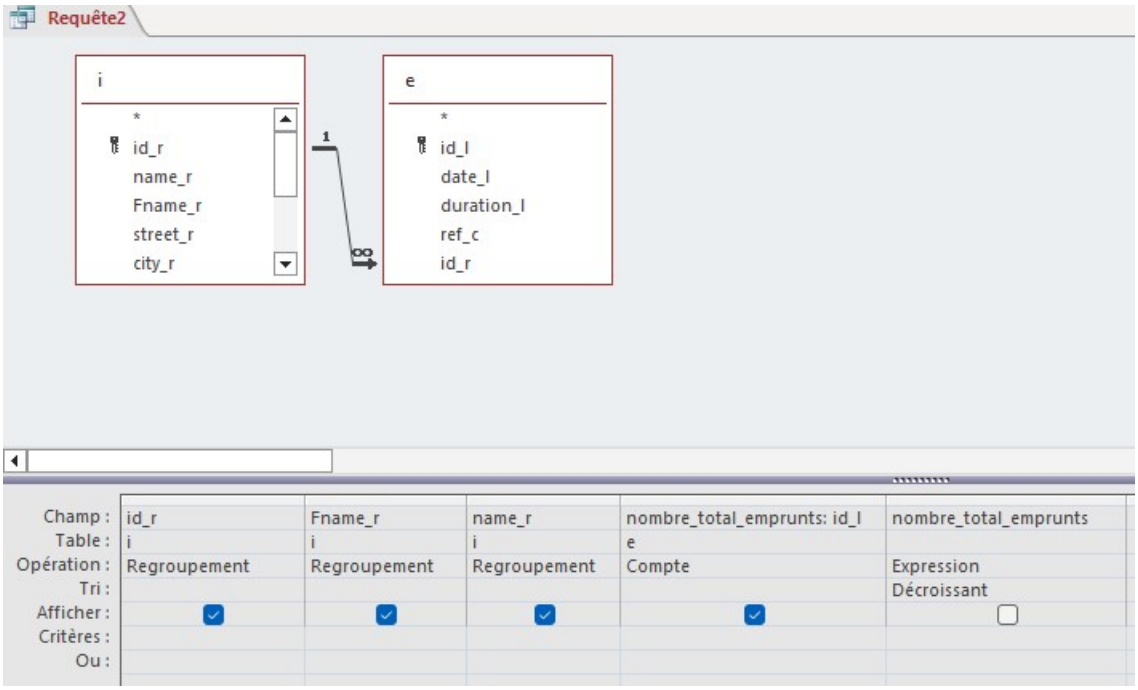


Figure 5.8: Totals (Summary) Query Exemple 2[Scr]

SQL code 2:

```
SELECT
r.id_r, r.Fname_r, r.name_r, COUNT(l.id_l) AS total_loans
FROM
Registered r
LEFT JOIN
Loan l ON r.id_r = l.id_r
GROUP BY
r.id_r, r.Fname_r, r.name_r
ORDER BY
total_loans DESC;
```

To get the desired result, you need to join the Registered table (containing information about the members) with the Loan table (containing information about the loans) and group the results by member.

Query Explanation

- SELECT r.id_r, r.Fname_r, r.name_r, COUNT(l.id_l) AS total_loans: This line selects the ID, first name, and last name of the registered person, and counts the number of loans (id_l) for each group. The result of the count is displayed in a column named total_loans.
- FROM Registered r: Specifies that the main table of the query is Registered. The alias r is used to shorten it.
- LEFT JOIN Loan l ON r.id_r = l.id_r: Performs a left join with the Loan table (using the alias l). The join is made on the id_r column, which is the foreign key linking a loan to a registered person. A LEFT JOIN is used here to ensure that all registered members are included in the result, even those who have not made any loans (their total will be 0).
- GROUP BY r.id_r, r.Fname_r, r.name_r: Groups the rows by registered person so that COUNT() can count the loans for each distinct individual.
- ORDER BY total_loans DESC: Sorts the results to first display the registered members who have made the most loans.

Parameter Query

Objective: Search for all books of a type whose label is entered by the user.

Graphical solution: (As show in Figure 5.9)

SQL code:

```
SELECT b.id_b, b.title_b, b.year_b, bt.label_t
FROM
Book AS b
```

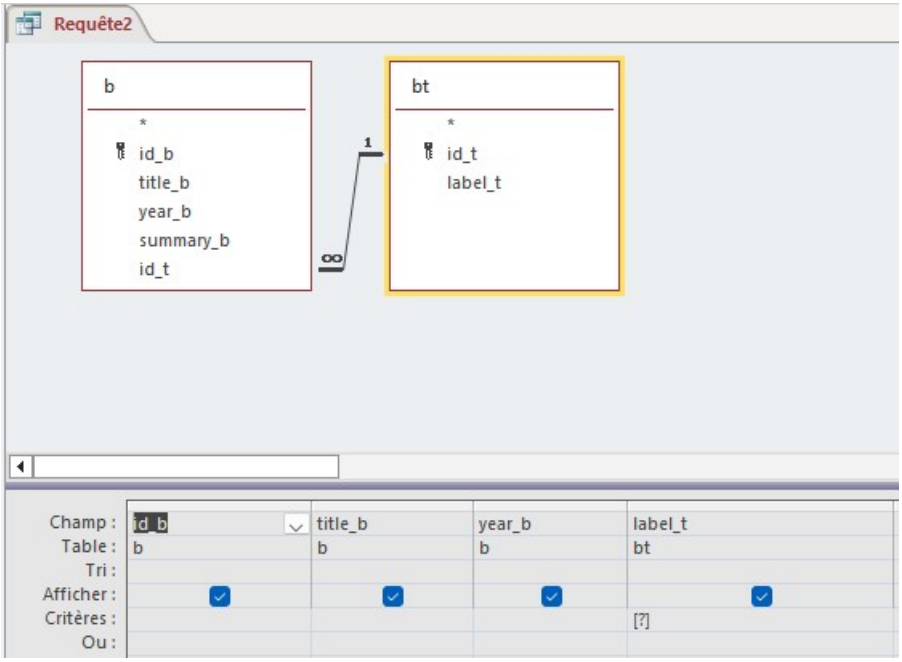


Figure 5.9: Parameter Query Exemple[Scr]

INNER JOIN
[Book Type] AS bt ON b.id_t = bt.id_t
WHERE
bt.label_t = ?;

To build this query, we need to link the Book table to the Book Type table and filter the results based on the user's input. Using a parameter (represented by ?) is essential for security and efficiency.

Query Explanation

- SELECT b.id_b, b.title_b, b.year_b, bt.label_t: This part selects the information we want to display: the book's ID, title, and year, as well as its type label.
- FROM Book AS b INNER JOIN [Book Type] AS bt ON b.id_t = bt.id_t: Here, we perform a join between the Book table (aliased as b) and the Book Type table (aliased as bt). The link is made on the common column id_t, which is the primary key in Book Type and a foreign key in Book.
- WHERE bt.label_t = ?: This is the filtering condition. It only keeps the rows where the label_t (the name of the book type) matches the value that will be provided by the user. The ? is a parameter marker.

Example of Use (in pseudo-code)
Here is how an application would use this query:

(a) The user enters the desired book type
user_value = "Science Fiction"

(b) The SQL query is prepared with the parameter marker
sql_query = 'SELECT title_b FROM Book b JOIN "Book Type" bt ON b.id_t = bt.id_t WHERE bt.label_t = ?'

(c) The query is executed by binding the user's value to the marker
results = database.execute(sql_query, [user_value])

(d) Display the results
DISPLAY results

5.4.2 Action Queries

These queries modify data in bulk. Use with caution.

Append Query

Archive old loans.

Objective: Copy all loans made before 2023 into an archive table named Loan_Archive.
Graphical solution: (As show in Figure 5.10)

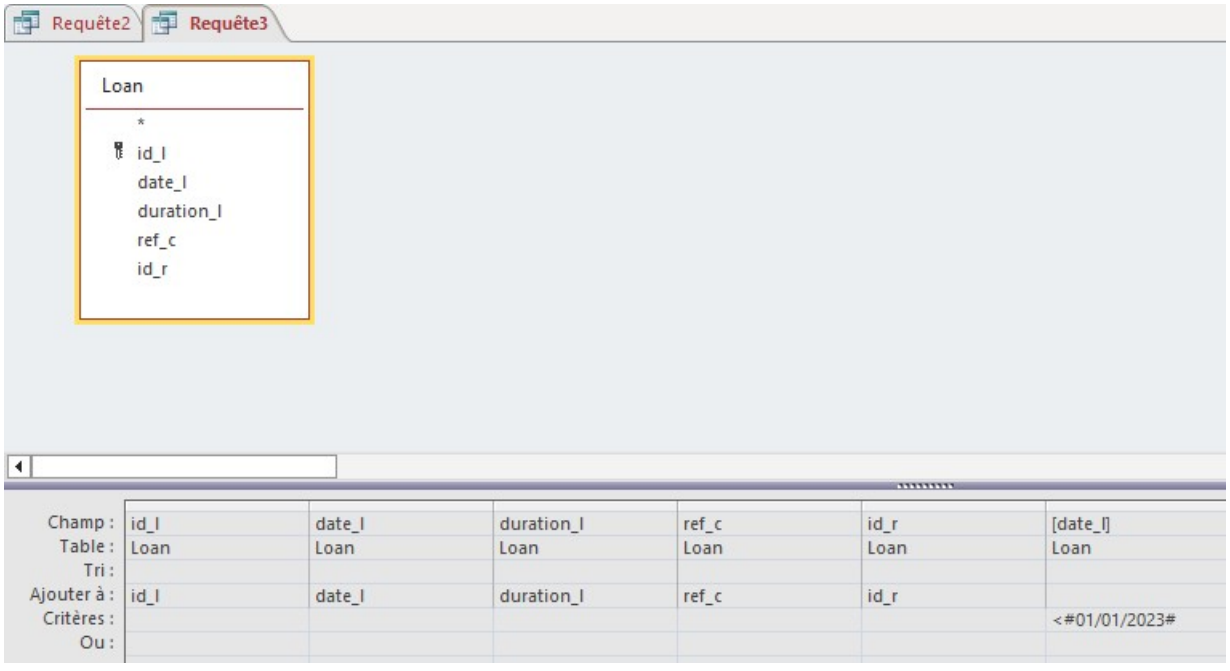


Figure 5.10: Append Query Exemple[Scr]

SQL code:

```
INSERT INTO Loan_Archive ( id_l, date_l, duration_l, ref_c, id_r )
SELECT id_l, date_l, duration_l, ref_c, id_r
FROM Loan
WHERE date_l < #01/01/2023#;
```

Update Query

Apply a change in bulk.

Objective: Increase the loan duration by 7 days for all books of the ”Novel” type.

Graphical solution: (As show in Figure 5.11)

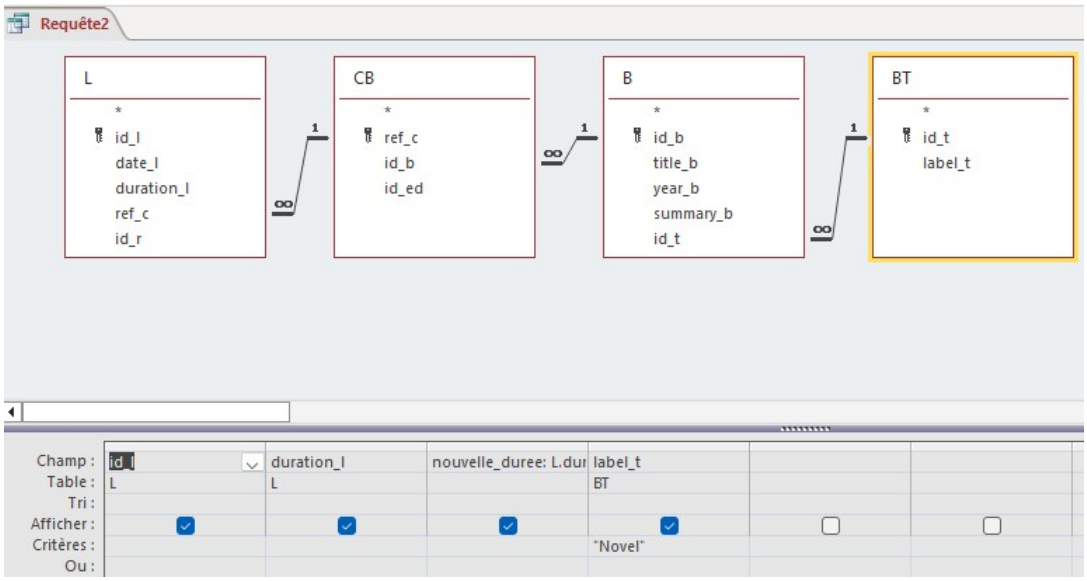


Figure 5.11: Update Query Exemple[Scr]

SQL code:

UPDATE
Loan
INNER JOIN
([Copy book]
INNER JOIN
(Book
INNER JOIN
[Book Type]
ON Book.id_t = [Book Type].id_t)
ON [Copy book].id_b = Book.id_b)
ON Loan.ref_c = [Copy book].ref_c
SET
Loan.duration_l = Loan.duration_l + 7
WHERE
[Book Type].label_t = ”Novel”;

Delete Query

Clean up data.

Objective: Delete the archived loans (those from before 2023) from the main Loan table.

Graphical solution: (As show in Figure 5.12)

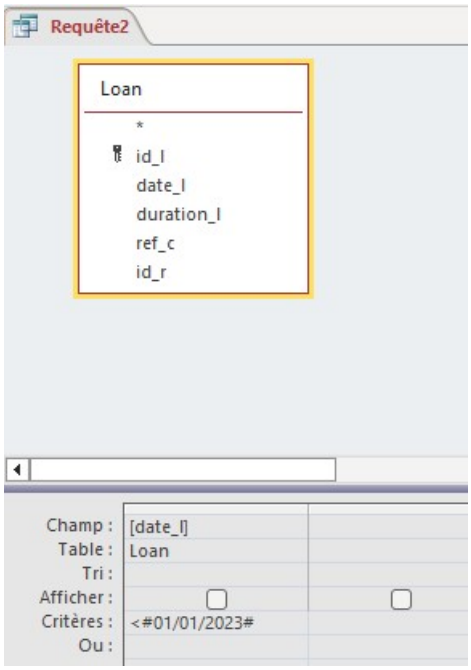


Figure 5.12: Delete Query Exemple[Scr]

SQL code:

```
SELECT *
FROM Loan
WHERE date_l < #01/01/2023#;
```

5.4.3 Crosstab Query

Objective: Display the number of books published by author's country and by publication year.

In our case, we want to see the number of books based on the author's country (as rows) and the publication year (as columns).

To do this, we need to join several tables: Country, Author, Write, and Book.

Here are two solutions: a standard *GROUP BY* query that works on almost all database systems, and a *TRANSFORM...PIVOT* query specific to Microsoft Access, which produces a true crosstab table.

Solution 1: Standard Query with GROUP BY

This query will give as a flat list where each row represents a unique combination of country and year, with the corresponding number of books. This is the most universal method.

Graphical solution 1: (As show in Figure 5.13)

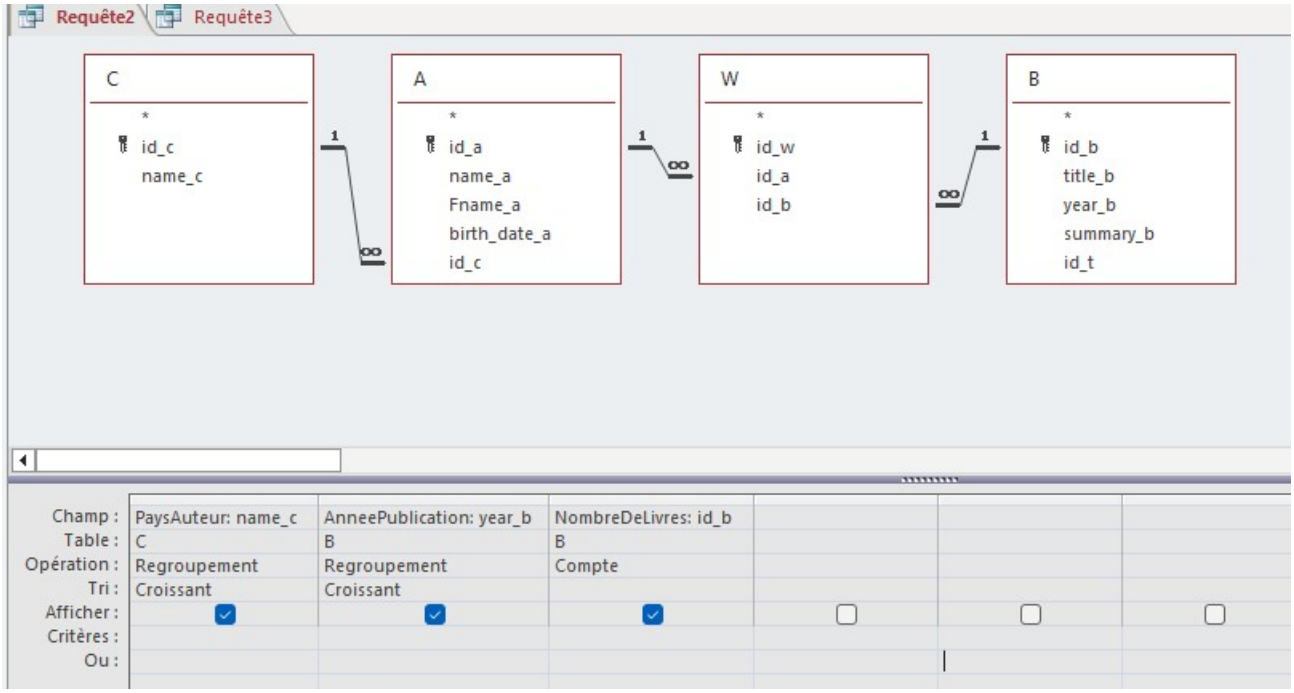


Figure 5.13: Crosstab Query with GROUP BY Exemple[Scr]

SQL code 1:

```
SELECT
C.name_c AS AuthorCountry,
B.year_b AS PublicationYear,
COUNT(B.id_b) AS NumberOfBooks
FROM
(Country AS C
INNER JOIN
Author AS A ON C.id_c = A.id_c)
INNER JOIN
(Write AS W
INNER JOIN
Book AS B ON W.id_b = B.id_b)
```

ON A.id_a = W.id_a
GROUP BY
C.name_c,
B.year_b
ORDER BY
C.name_c,
B.year_b;

Solution 2: Crosstab Query for MS Access (TRANSFORM...PIVOT)

This query is specific to Microsoft Access and directly generates a crosstab table, which exactly matches the "crosstab analysis" objective.

Graphical solution 2: (As show in Figure 5.14)

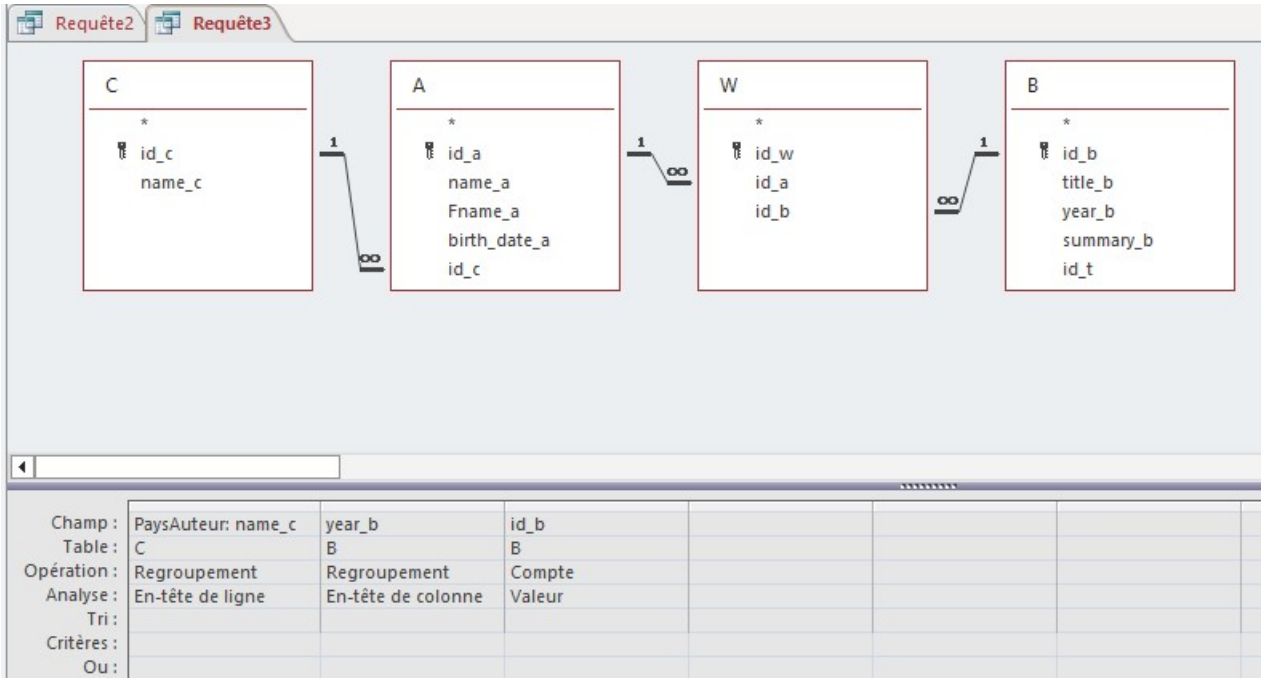


Figure 5.14: Crosstab Query with TRANSFORM...PIVOT Exemple[Scr]

SQL code 2:

TRANSFORM Count(B.id_b)
SELECT
C.name_c AS AuthorCountry
FROM
(Country AS C
INNER JOIN
Author AS A ON C.id_c = A.id_c)
INNER JOIN
(Write AS W

```
INNER JOIN
Book AS B ON W.id_b = B.id_b)
ON A.id_a = W.id_a
GROUP BY
C.name_c
PIVOT
B.year_b;
```

Explanation of the TRANSFORM...PIVOT Query

- TRANSFORM Count(B.id_b): Defines the aggregate operation that will be displayed in the center of the table (the count of books).
- SELECT C.name_c AS AuthorCountry: Defines the row headers of the table (the countries).
- FROM ...: Contains the necessary joins to link the tables, as in the standard query. The syntax with parentheses is important for MS Access.
- GROUP BY C.name_c: Specifies the field on which the rows should be grouped.
- PIVOT B.year_b: Defines the column headers. MS Access will create a column for each publication year (year_b) found in the data.

Chapter 6

Creating Reports

Contents

While tables store data, queries interrogate it, and forms facilitate its entry, reports, on the other hand, are specifically designed to present this information in a summarized, professional manner, optimized for printing or exporting. A report is the final product of your database, the one you share as a summary, invoice, labels, or catalog[Mic25k].

6.1 Introduction to Reports

A report is an Access object whose main purpose is to format, calculate, group, and summarize data for printing or converting into a portable document (like a PDF). Unlike forms, which are designed for interaction (data entry, modification), reports are optimized for read-only viewing and the final presentation of data[Mic25k].

6.1.1 The Role and Utility of Reports

Reports are essential for transforming the raw data in your database into clear, actionable information. Their main functions are:

- **Presenting data professionally**: Reports give you full control over the layout. You can add titles, logos, page numbers, headers, and footers to create polished documents that adhere to a graphic charter.
- **Summarizing and grouping information**: This is one of the major strengths of reports. You can easily group your data by one or more criteria (for example, grouping sales by region, then by salesperson) and perform calculations on each group (such as subtotals) as well as a final calculation (grand total)[Mic25k].
- **Creating various documents**: Reports are not limited to simple lists. You can use them to generate:
  - Invoices or purchase orders.
  - Product catalogs.
  - Contact or address lists.
  - Mailing labels.
  - Summary reports with charts.

6.1.2 Data Source of a Report

As with a form, the data source for a report (called Record Source in its properties) can be a table or, more powerfully, a query. Using a query as the source is the most common and recommended practice, as it allows you to provide the report with data that is already sorted, filtered, and pre-calculated, which greatly simplifies the design of the report itself.

6.2 Creating a Report

As with forms, Access offers several approaches to create a report, ranging from automatic generation to complete manual design. The choice depends on the complexity of the desired report and the degree of customization needed. All these options are found on the Create tab of the ribbon, within the Reports group.

6.2.1 The "Report" Tool (Instant Creation)

This is the most direct method for obtaining a basic report (Figure 6.1).

Figure 6.1: The "Report" Tool (Instant Creation)-Exemple[Scr]

- **How to do it?** Select a table or query in the Navigation Pane, then click the Report icon.
- **Result**: Access immediately generates a "stacked" report that includes all fields from the source (Figure 6.2). If groupings were defined in the source (for example, a sorted query), Access will reproduce them. This is an excellent starting point, but the result almost always requires manual adjustments in Layout or Design view to be truly presentable.

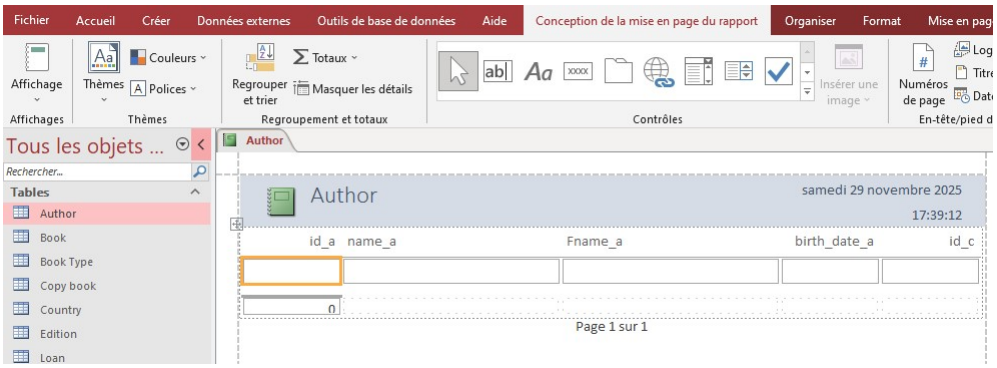


Figure 6.2: The "Report" Tool (Instant Creation)-Result[Scr]

Figure 6.3: The Report Wizard-Exemple-1-[Scr]

6.2.2 The Report Wizard

The wizard is the most recommended and versatile method. It guides you through a series of screens to configure the structure of your report before you even open it (Figure 6.3).

- **How to do it?** Click Report Wizard.
- **Guided Process**:
  (a) **Field Selection**: Choose the table(s)/query(s) and select the fields you want to appear.
  (b) **Grouping Levels**: This is the most important step. You can choose one or more fields on which to base the groupings (e.g., group by Country, then by City). The report will be structured according to this hierarchy.
  (c) **Sort Order**: For each group, you can define up to four fields to sort the records (e.g., sort customers in alphabetical order by LastName).
  (d) **Summary Options**: You can ask the wizard to calculate subtotals, averages, or other aggregates for each group.
  (e) **Layout and Orientation**: Choose a layout (Stepped, Block, Outline) and the paper orientation (Portrait or Landscape).
  (f) **Title**: Give your report a name.

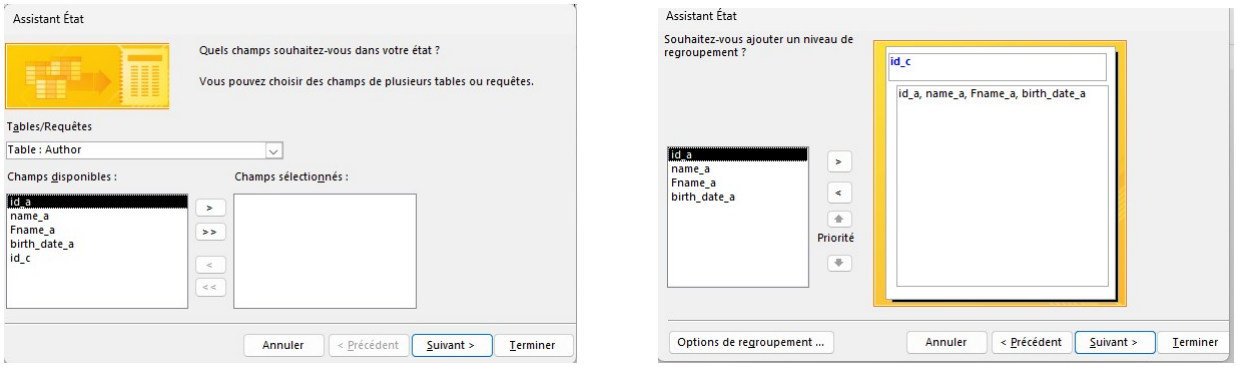


Figure 6.4: The Report Wizard-Exemple-2-[Scr]

- **Use**: Ideal for creating structured reports with groupings and subtotals effortlessly.

Figure 6.5: The Report Wizard-Result[Scr]

6.2.3 Design View and Blank Report

These methods offer absolute control and are intended for creating custom or complex reports, such as invoices or mailings (Figure 6.6).

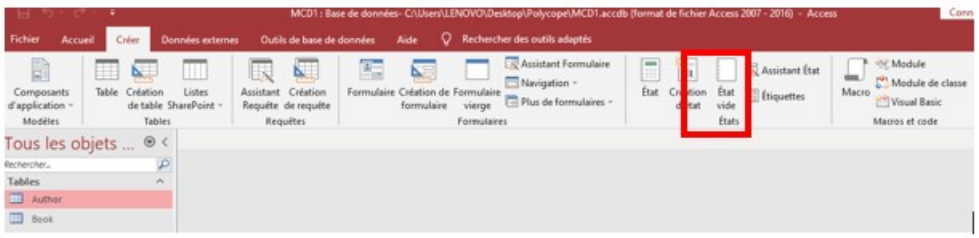


Figure 6.6: Design View and Blank Report-Exemple[Scr]

- **Blank Report**: Creates an empty report in Layout View. You can then drag fields from the ”Field List” to build your report visually.
- **Design View**: You start with a blank design grid, divided into sections (Report Header, Page Header, Detail, etc.) (Figure 6.7). It is up to you to define the data source, manually add each control (text boxes, labels), and configure groupings and sorting via the ”Group, Sort, and Total” pane. This method is the most powerful but also the most demanding.

6.3 Structure of a Report

Access reports are divided into distinct sections that control layout and presentation. The most important are:

(a) **Report/Page Header and Footer (Figure 6.8)**

Figure 6.7: Design View and Blank Report-Result[Scr]

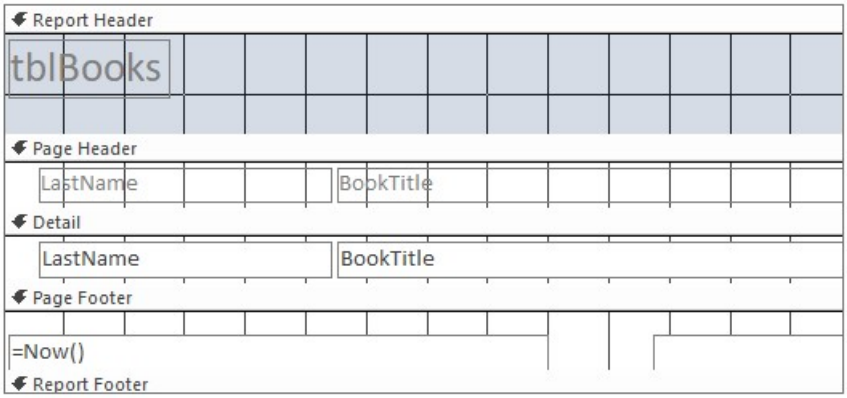


Figure 6.8: Report/Page Header and Footer[Scr]

- **Report Header** : Appears only at the top of the first page. This is the place for the main title, logos, or introductory information for the entire report. It is not repeated on subsequent pages.
- **Page Header** : Appears at the top of every page. Commonly used for column labels, dates, or page titles; ensures clarity when printing large reports.
- **Report Footer** : Shown only once, at the end of the last page. Ideal for global summaries such as totals, averages, or conclusions that apply to the entire report.
- **Page Footer** : Printed at the bottom of every page. Often used for page numbers, dates, or additional notes that help with navigation and organization[Mic25h].

(b) **Grouping and Sorting Levels**

- Access enables you to group records based on one or more fields. Each group can have:
  - **A Group Header** : Inserted before each new group in your data, perfect for labeling each subgroup (e.g. city, department).
  - **A Group Footer**: Inserted after each group's data, mainly for subtotals or group-level analyses.

- Sorting within groups organizes your records hierarchically and makes reports easier to interpret, especially with nested groups[Mic25h].

(c) **Detail Section**

- The Detail section is the core of the report. Each record in your query or table is displayed here—one row per record.
- This is where you place the data fields (text boxes, calculations) you want to display for every item in your report.

By understanding and manipulating these structural elements, you can create professional, readable reports tailored to any presentation or printing need.

6.4 Calculations in Reports

Reports are not only used to display data lists; they also allow you to perform automatic calculations (sums, averages, counts, minimum, maximum, etc.) in order to produce real summary reports. These calculations are performed using calculated controls (text boxes containing an expression) placed in the appropriate sections of the report, in particular in the group footers and the report footer.

In the group footer, expressions (for example $= Sum([Amount])$ or $= Avg([Amount])$) compute aggregates only for the records in the corresponding group. This way you obtain subtotals or averages per customer, product, region, and so on. In the report footer, the same functions now calculate grand totals over the entire report (total turnover, overall average, total number of records, etc.).

To create these calculations, you add a text box in the desired section and then enter an expression that uses Access aggregate functions: $Sum(sum)$, $Avg(average)$, $Count(count)$, Min and Max (extreme values). By combining these functions with grouping levels, you can turn a simple report into a rich analysis report, presenting both the detail of records and numeric indicators per group and for the whole dataset.

Chapter 7

Create a Menu For Applications

Contents

7.1 Introduction to Macros

7.1.1 Definition

In Microsoft Access, a macro is a simplified automation tool that allows you to execute a task or a series of tasks sequentially. Unlike VBA (Visual Basic for Applications) code, which requires writing lines of computer programming, macros are built via an intuitive visual interface. The user selects predefined actions from a list (such as ”OpenForm”, ”CloseWindow”, ”MessageBox”) and assembles them to create an execution logic.

We can think of a macro as a ”robot” that is given a list of simple instructions to follow step-by-step.

7.1.2 Utility and Benefits

The use of macros meets several essential needs in database design[Mic25d]:

- **Time saving and efficiency** : They allow for the automation of repetitive tasks. For example, instead of manually opening three forms every morning, a single macro can do it in one click.
- **Simplification for the end user** : They hide the complexity of the database. The user does not need to know the internal structure of Access; they simply click a button that triggers the macro.
- **Data security** : They allow data validation before saving or prevent certain actions if conditions are not met (for example, preventing a form from closing if a required field is empty).
- **Accessibility** : No programming knowledge is required. It is the ideal gateway to automation before tackling more complex development in VBA.

In summary, macros serve as the ”glue” between the different objects of the database (tables, forms, reports) to transform a simple storage structure into a truly functional application.

7.2 Creating and Running a Simple Macro

Creating macros in Access relies on a visual interface called the Macro Builder. We will illustrate this process with a simple example: a macro that displays a welcome message to the user[Mic25d].

7.2.1 The Design Interface (Macro Builder)

To access the creation tool (Figure 7.1):

(a) Go to the Create tab on the ribbon.

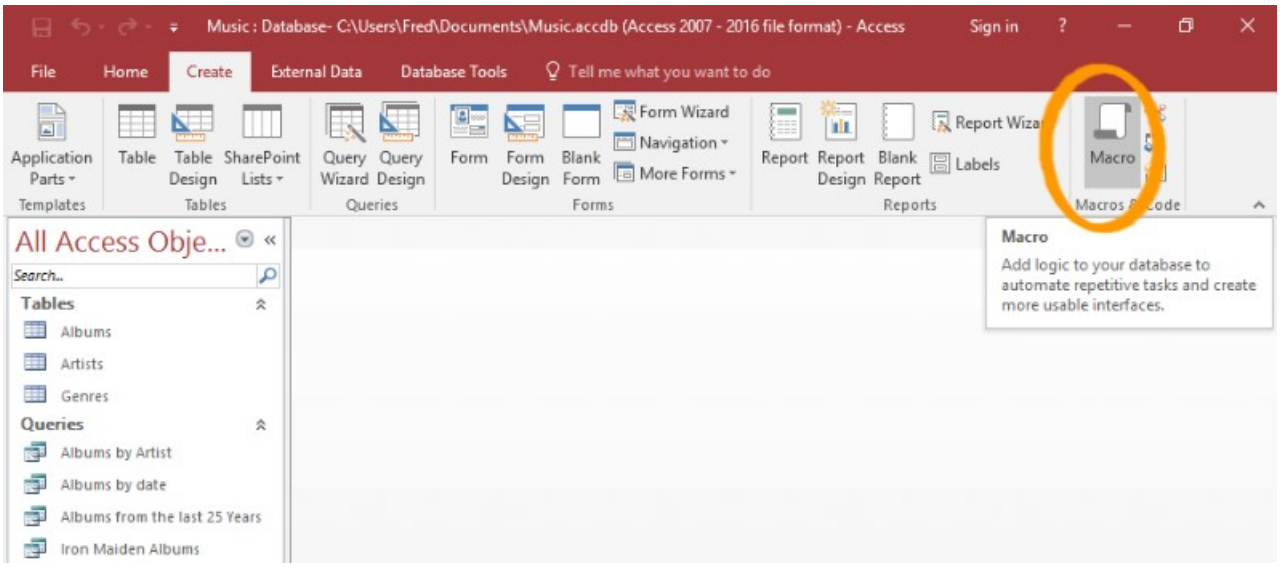


Figure 7.1: The Design Interface (Macro Builder)[Scr]

(b) In the Macros Code group, click the Macro button.

A new window opens. It is mainly composed of:

- A central area for stacking actions.
- An Action Catalog pane (on the right or via the ribbon button) that lists all available commands

7.2.2 Practical Example: Creating a ”Hello” Macro

Follow these steps to create your first automation:

(a) **Add an action**: in the ”Add New Action” dropdown list (or via the catalog), search for and select the MessageBox action (Figure 7.2).

(b) **Configure the action**: Once the action is added, configuration fields appear.

- In the Message field, type: *Hello, welcome to the application!*.
- Leave the other parameters (Beep, Type, Title) as default for now (Figure 7.3).

(c) **Save the macro**:

- Click the save icon (floppy disk) or press Ctrl + S.
- Name the macro: mcr_Hello. (The mcr_ prefix is a useful naming convention to quickly identify your objects).

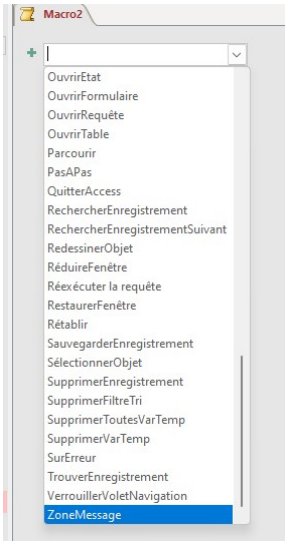


Figure 7.2: Practical Example: Creating a ”Hello” Macro 1/3[Scr]

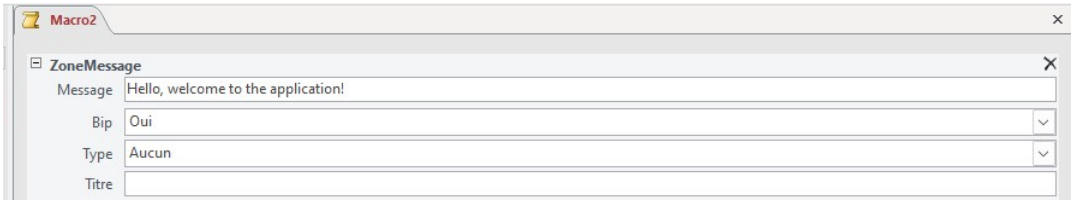


Figure 7.3: Practical Example: Creating a ”Hello” Macro 2/3[Scr]

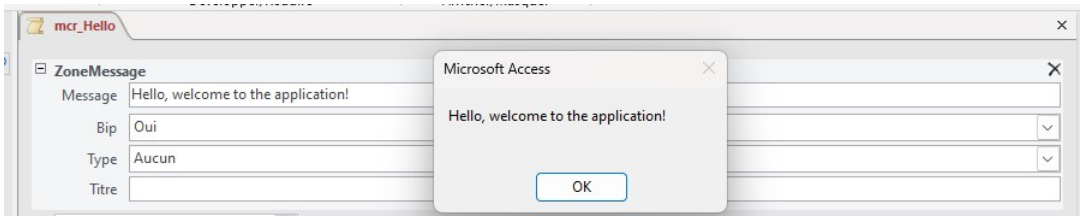


Figure 7.4: Practical Example: Creating a ”Hello” Macro 3/3[Scr]

7.2.3 Running the Macro

There are several ways to trigger the execution of a macro:

- **From Design View**: Click the Run button (red exclamation mark) located in the Macro Design tab. This is ideal for testing functionality immediately (Figure 7.4).
- **From the Navigation Pane**: Simply double-click on the macro name mcr_Hello in the object list on the left.
- **Via a button**: A macro is often associated with the ”On Click” event of a button in a form.

7.3 Creating a Navigation Menu

For a database to become a truly user-friendly application, it is essential to hide the complexity of Access objects behind a simple interface. This is the role of the Main Menu (or navigation form).

7.3.1 Designing the Main Menu

The goal is to create a ”dashboard” form that centralizes all functionalities.

Method 1: The Navigation Form (Recommended for beginners)

Access offers a dedicated tool to create modern menus with tabs[Mic25t, Citd].

(a) Go to the Create tab, Forms group.

(b) Click on Navigation and choose a layout (e.g., Vertical Tabs, Left)(Figure 7.5).

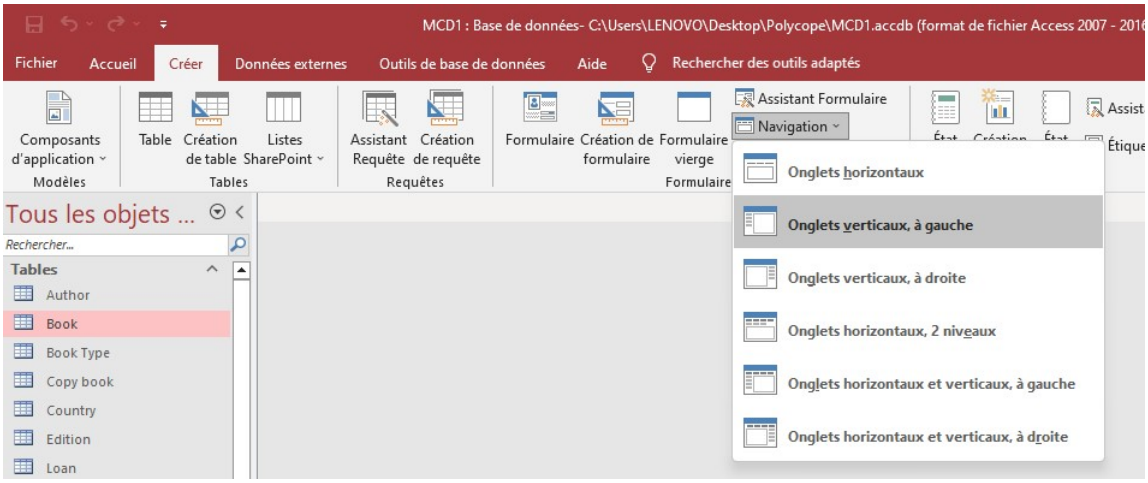


Figure 7.5: Method 1: The Navigation Form (Recommended for beginners) 1/3[Scr]

(c) An empty form appears with tabs (Figure 7.6).

(d) From the Navigation Pane on the left, drag and drop your forms (e.g., f_Author, f_Book) and reports directly onto the tabs. Access automatically creates the navigation buttons (Figure 7.7).

Method 2: The Blank Form with Buttons (More flexible)

This method allows for total customization of the design[Mic25t, Citd].

(a) Create a Blank Form in Design View (Figure 7.8).

(b) Add a title with a label (e.g., ”Library Management”).

Figure 7.6: Method 1: The Navigation Form (Recommended for beginners) 2/3[Scr]

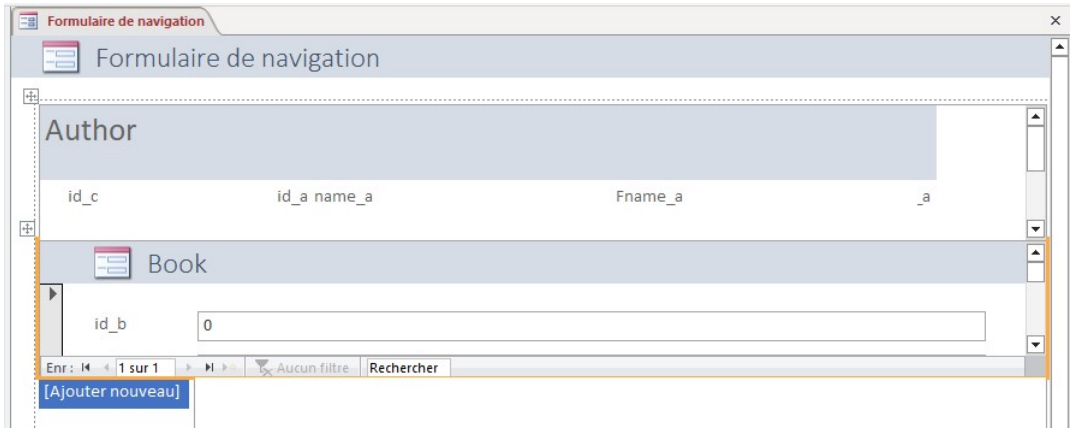


Figure 7.7: Method 1: The Navigation Form (Recommended for beginners) 3/3[Scr]

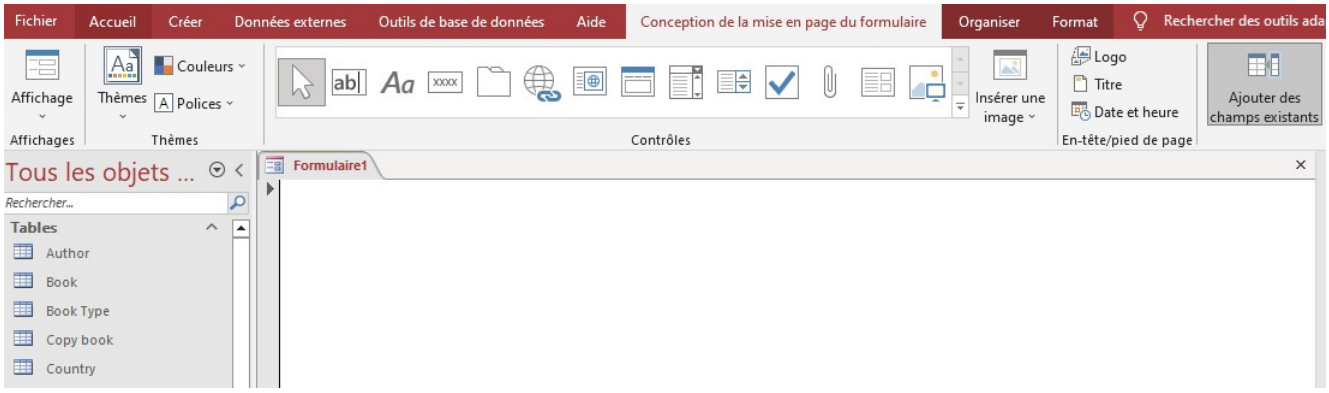


Figure 7.8: Method 2: The Blank Form with Buttons (More flexible)) 1/4[Scr]

(c) Use the Button tool (in the Design tab).

(d) The Button Wizard launches. Choose the category Form Operations > Open Form (Figure 7.9).

(e) Select the form to open (e.g., f_Book) and finish the wizard (Figure 7.10).

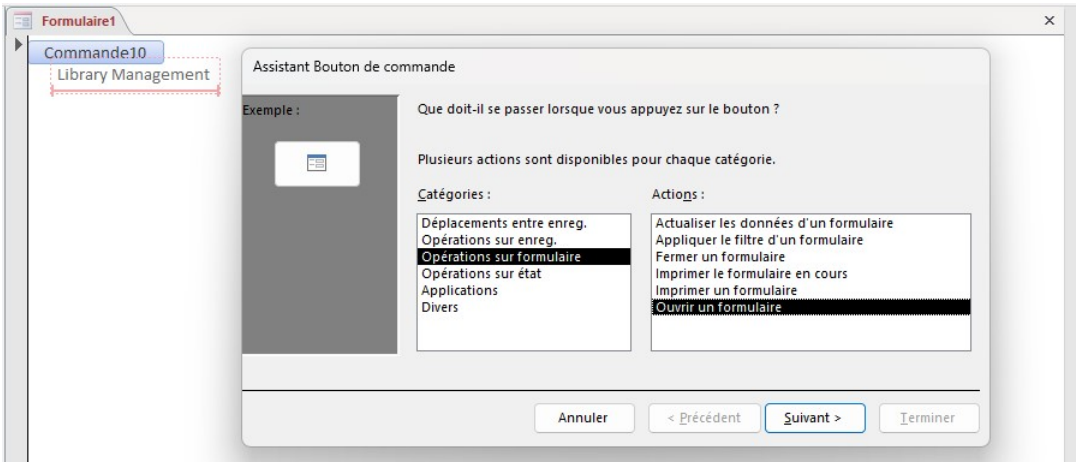


Figure 7.9: Method 2: The Blank Form with Buttons (More flexible)) 2/4[Scr]

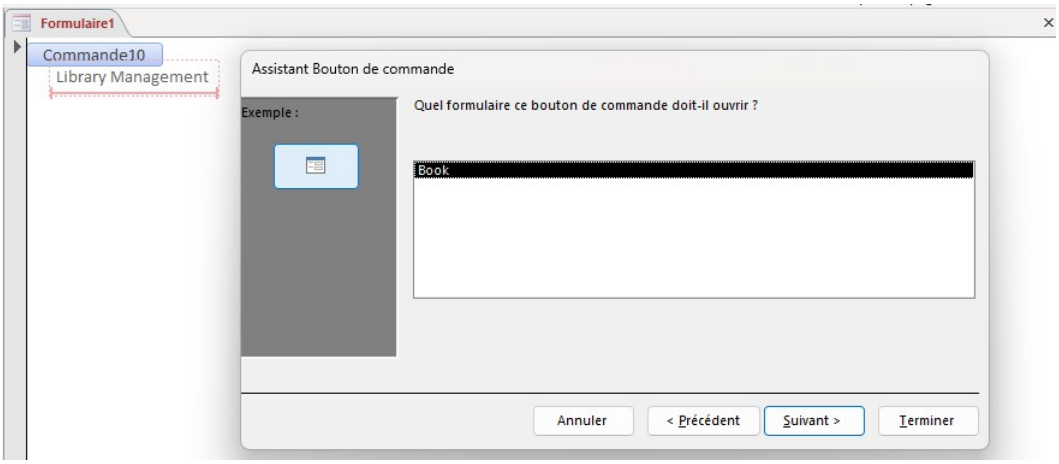


Figure 7.10: Method 2: The Blank Form with Buttons (More flexible)) 3/4[Scr]

(f) Repeat the process to create buttons for your reports (Report Operations > Preview Report) and a button to exit the application (Application > Quit Application)(Figure 7.11).

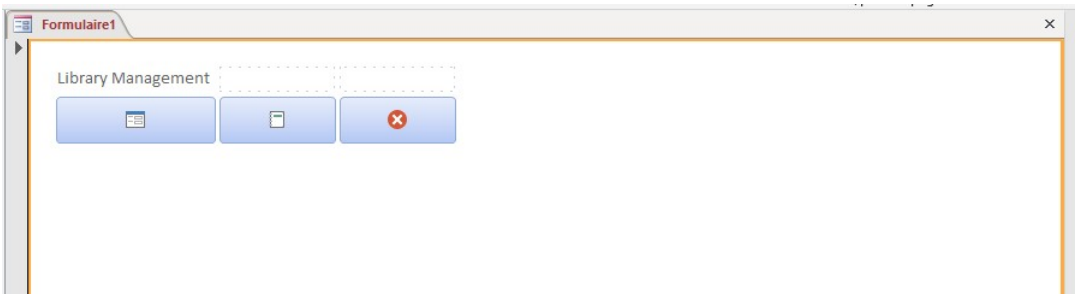


Figure 7.11: Method 2: The Blank Form with Buttons (More flexible)) 4/4[Scr]

7.3.2 Setting the Startup Menu

To ensure this menu opens automatically as soon as your application launches[Mic25t, Citd]:

(a) Click on the File tab > Options.

(b) Select the Current Database section.

(c) In the Display Form dropdown list, choose your menu form (e.g., f_MainMenu).

(d) Confirm with OK. You will need to close and reopen the database to see the change.

Thanks to this menu, the end user no longer needs to access the raw Navigation Pane; they simply navigate via the buttons you have created.

Chapter 8

Introduction to VBA Programming for Access

Contents

8.1 Introduction to VBA

8.1.1 What is VBA?

VBA stands for Visual Basic for Applications. It is a programming language developed by Microsoft and integrated into all Office suite software (Access, Excel, Word, etc.).

Unlike macros, which rely on predefined lists of actions (somewhat like building blocks), VBA is a full-fledged computer language. It allows you to write precise instructions in the form of code to control Access down to the smallest detail. It is the engine running "under the hood" of your forms and reports[Cita].

8.1.2 Why use VBA instead of macros?

Although macros are very useful for beginners, they quickly reach their limits when you want to create a professional application. VBA is essential for the following reasons[JP21]:

- **Error Handling**: With VBA, you can anticipate problems (e.g., a missing file, incorrect input) and display a clear message to the user instead of letting Access crash or show an incomprehensible error code.
- **Advanced Interactivity**: VBA allows you to modify the interface in real-time. For example, hiding a field if a box is unchecked or changing the color of a button based on the entered value—something impossible or very complex with a simple macro.
- **Custom Functions**: If Access does not have a specific calculation formula for your business needs, you can create it yourself in VBA and use it everywhere in your queries and forms.
- **Communication with other software**: VBA enables Access to "talk" to Excel (to export data), Outlook (to send an automatic email), or Word (to generate a letter), offering complete automation of your administrative processes.

In summary, while macros are perfect for simple and repetitive tasks, VBA is the indispensable tool for building robust, user-friendly, and custom-made applications.

8.2 The Visual Basic Editor (VBE)

To write VBA code, you must leave the standard Access interface and enter a dedicated development environment: the Visual Basic Editor (or VBE).

8.2.1 Accessing the VBE

There are several ways to open this interface:

- **Via the Ribbon**: In the Database Tools tab, click the Visual Basic button (often located on the far left) (Figure 8.1).

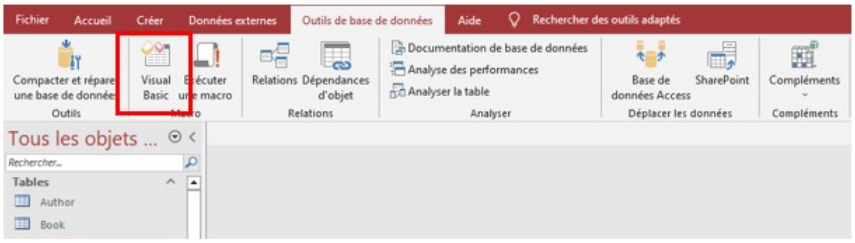


Figure 8.1: Accessing the VBE-Via the Ribbon[Scr]

- **Via Keyboard Shortcut (Recommended)**: Simply press ALT + F11. This is the fastest method and the most used by developers[JP21].

8.2.2 Interface Overview

Once opened, the VBE is presented as several windows, each with a specific role (Figure 8.2):

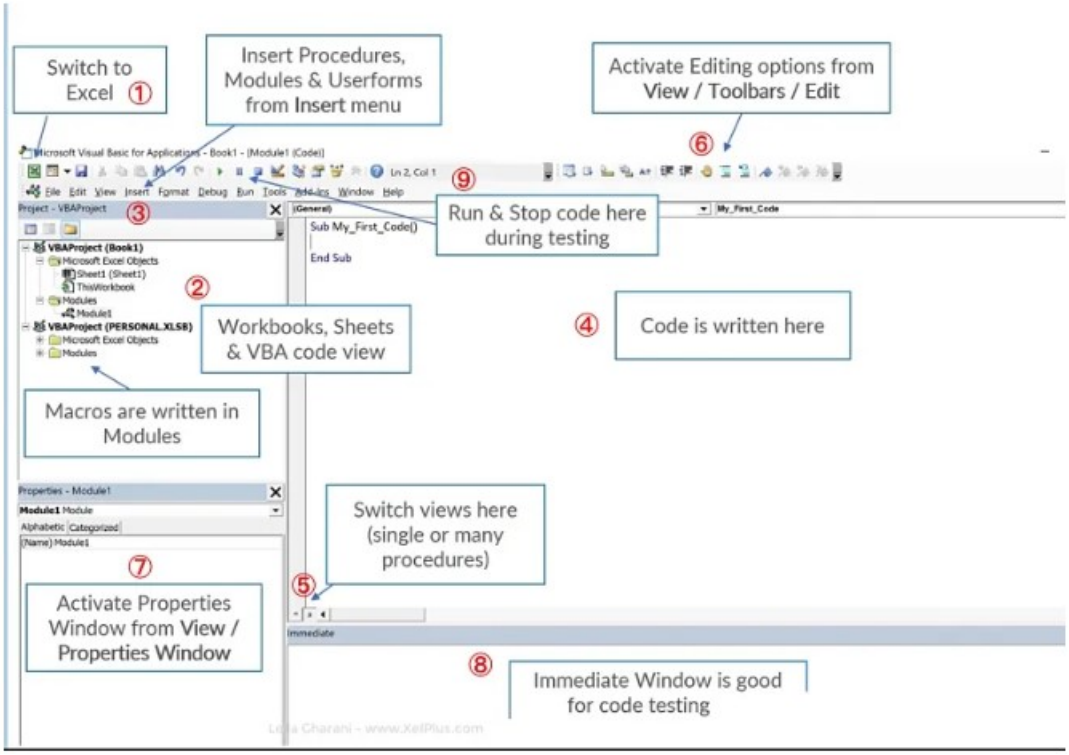


Figure 8.2: VBE Interface Overview[Lei]

- **Project Explorer**: Usually located at the top left, it displays the tree structure of your database. Here you will see all forms and reports containing code, as well as "Modules" where general code is stored. This is where you navigate between different code files[JP21, Lei].

- **Code Window**: This is the large central area. This is where you will type your VBA instructions. It functions like a smart text editor that color-codes keywords and helps you complete commands.
- **Properties Window**: Often at the bottom left (shortcut F4), it allows you to modify the characteristics of the selected object (name, color, etc.) without going back to Access[Mic25s].
- **Immediate Window**: At the bottom of the screen (shortcut CTRL + G), this area is very useful for testing a small line of code "on the fly" or for debugging your program by displaying temporary values.

Understanding this environment is the essential first step before writing your first line of code.

8.3 VBA Language Basics

Like any computer language, VBA has its own grammar and vocabulary. We will look at the three pillars that allow you to build any program: storing information (variables), decision-making (control structures), and code organization (procedures).

8.3.1 Variables

A variable is a named "box" in the computer's memory used to store temporary information (a price, a name, a date) for later use in the code.

Declaration:

In VBA, it is recommended to declare each variable using the *Dim* keyword followed by its type.

- **Syntax**: *Dim* VariableName As DataType
- **Common Examples**:

```
Dim StudentName As String ' For text
Dim Age As Integer ' For a whole number
Dim Average As Double ' For a decimal number
Dim IsEnrolled As Boolean ' For True/False
Dim BirthDate As Date ' For a date
```

If the type is not specified, VBA defaults to the Variant type, which is flexible but consumes more memory[Mic25g, MAG].

8.3.2 Control Structures

These allow you to direct the flow of the program's execution.

(a) **Conditions (If... Then... Else)** These allow code to be executed only if a condition is met.

- **Example**: Check if a student has passed.

```
If Average >= 10 Then
    MsgBox "Passed"
Else
    MsgBox "Failed"
End If
```

You can also use $ElseIf$ to test multiple conditions in sequence[Mic25s].

(b) **Loops (For... Next, Do... Loop)** These allow an action to be repeated multiple times without rewriting the code.

- **For Loop**: When the number of repetitions is known in advance.

```
For i = 1 To 10
    MsgBox "Number " & i
Next i
```

- **Do While Loop**: When repeating as long as a condition is true (e.g., reading rows in a table until the end).

8.3.3 Functions and Procedures

VBA code is broken down into blocks called "procedures". There are two main types:

- **The Sub Procedure (Subroutine)**: Performs an action but does not return a result. This is what is used to respond to a button click.
  Example: $SubShowMessage()$ which displays "Hello".
- **The Function**: Performs a calculation and returns a result. It can be used in a formula.
  Example: $FunctionCalculateVAT(PriceExclTax)$ which returns the tax amount.

The main distinction is that a Function returns a value, whereas a Sub simply executes a task[Mic25g, MAG].

8.4 The Access Object Model

To program effectively in VBA, it is essential to understand how Access is structured "from the inside". This is referred to as the object model. Imagine a tree structure (a hierarchy) where each element of your database (form, report, button, text box) is an object that can be manipulated through code.

8.4.1 Object Hierarchy

At the top of this hierarchy is the Application object (Access itself). Just below, there are main collections that group all objects of the same type:

- **Forms**: The collection of all currently open forms.
- **Reports**: The collection of all currently open reports.
- **DoCmd**: A special object that allows executing macro commands in VBA (Open, Close, Print)[JP21].

8.4.2 Manipulating Forms and Reports

To act on a specific form, you must indicate its full path in the hierarchy.

- **General Syntax**: Forms("FormName") or Forms!FormName
- **The Me Object**: This is a very powerful shortcut. If you are writing code inside a form (for example, on a button click), you can use the keyword Me to refer to "this form". It is shorter and avoids typing errors[JP21].

8.4.3 Manipulating Controls (Fields, Buttons)

To read or modify the value of a field (e.g., a text box named $txtName$), you go down one level in the hierarchy.

- **Read a value**: Variable = $Forms!f_Student!txtName.Value$
- **Modify a value**: $Forms!f_Student!txtName.Value = "NewValue"$
- **With the Me shortcut**: $Me.txtName.Value = "NewValue"$

You can also modify the visual properties of controls:

- $Me.txtName.Visible = False$ (Hides the field)
- $Me.cmdValidate.Enabled = True$ (Enables the Validate button)
- $Me.lblTitle.Caption = "Hello"$ (Changes the text of a label)

8.4.4 The DoCmd Object: The Autopilot

The *DoCmd* (Do Command) object is the VBA equivalent of macro actions. It allows you to control the interface[JP21].

- **Open a form**: $DoCmd.OpenForm"f_Home"$
- **Open a report**: $DoCmd.OpenReport"r_Invoice", acViewPreview$ (Print Preview)

- **Close the active object**: *DoCmd.Close*
- **Quit Access**: *DoCmd.Quit*

Mastering these few commands already allows you to build a fluid and dynamic navigation system, entirely controlled by code.

8.5 Practical Examples

To conclude this chapter, let's look at two very common use cases where VBA provides real added value compared to standard features.

8.5.1 Complex Data Validation

Access allows you to define simple validation rules (like "Price ¿ 0"), but for more advanced checks, VBA is essential. The key event to use is BeforeUpdate. It triggers just before the data is saved[Mic25u, JP21].

Example: Prevent entering a loan if the return date is earlier than the checkout date.

```
Private Sub Form_BeforeUpdate(Cancel As Integer)

    ' Check if the return date is entered and is earlier than the checkout date
    If Not IsNull(Me.ReturnDate) And Me.ReturnDate < Me.CheckoutDate Then

        ' Display a clear error message
        MsgBox "Error: The return date cannot be before the checkout date!", _
               vbCritical, "Invalid Date"

        ' Cancel the data saving process
        Cancel = True

        ' Set focus back to the incorrect field
        Me.ReturnDate.SetFocus

    End If

End Sub
```

Explanation: The $Cancel = True$ argument is a powerful command that stops the saving process immediately. As long as the user does not correct it, the incorrect data will not pollute the database.

8.5.2 Automating Repetitive Tasks

VBA excels at chaining actions. Let's take the example of a "New Client" button that should not only open a blank form but also pre-fill certain fields to save time[Mic25u, JP21].

Example: Button to add a client and set their city by default.

```
Private Sub btnNewClient_Click()

    ' 1. Open the Client form in Add mode (acFormAdd)
    DoCmd.OpenForm "f_Client", , , , acFormAdd

    ' 2. Pre-fill the city with "Paris" by default
    Forms!f_Client!txtCity.Value = "Paris"

    ' 3. Display a small confirmation message in the Immediate Window (for the
       developer)
    Debug.Print "New client initiated at " & Now

End Sub
```

This type of automation, although simple, considerably improves the daily user experience.

General Conclusion

At the end of this journey, we have covered all the essential steps for creating a functional and structured database application. Starting from the theoretical foundations of information systems and the Merise method, we have progressively transformed an abstract business need into a concrete and tangible technical solution using Microsoft Access.

This course material has enabled you to develop a dual skill set. On one hand, a conceptual skill, by learning to analyze, model, and structure information in a logical and coherent manner. On the other hand, a practical skill, by mastering the tools that bring this modeling to life.

You have learned to:

- Build the foundations of your application with Tables and ensure data consistency through relationships and referential integrity.
- Query this data with precision and relevance using Queries, thus transforming raw data into actionable information.
- Create ergonomic and secure user interfaces with Forms, facilitating data entry and consultation.
- Summarize and present this information professionally via Reports, ready to be printed or shared.
- Finally, glimpse the automation potential offered by the VBA language to enrich and customize your applications.

The knowledge acquired in this module extends beyond the simple framework of the Microsoft Access tool. You have assimilated a working methodology and a design logic that you can apply in many other professional and technological contexts. The ability to model a problem, structure a solution, and implement it is a fundamental asset in today's professional world.

This course has given you the keys to becoming informed users and autonomous designers of information systems. Curiosity and practice will be your best allies in deepening these concepts and exploring the more advanced features that will make you experts in data management and valorization.

Bibliography

[ano25] anovalexter. Parts of microsoft access, 2025. URL: https://www.scribd.com/presentation/900404106/Parts-of-Microsoft-Access.

[Atk88] John Atkins. A note on minimal covers. *ACM SIGMOD Record*, 17(4):16–21, 1988.

[Avi91] DE Avison. Merise: A european methodology for developing information systems. *European Journal of Information Systems*, 1(3):183–191, 1991.

[Bap09] Jean-Luc Baptiste. *Merise Guide pratique: Modélisation des données et des traitements, langage SQL*. Editions ENI, 2009.

[Ben] Mohamed Benelhadj. L'entreprise et son systeme d'information. URL: https://www.univ-constantine2.dz/CoursOnLine/Benelhadj-Mohamed/co/module_Systeme_d_Information_2.html.

[Cita] Apprendre à programmer en vba access. URL: https://www.bonbache.fr/apprendre-a-programmer-en-vba-access-170.html.

[Citb] Chapter 4: The merise method. URL: http://tele-ens.univ-oeb.dz/moodle/pluginfile.php/396773/mod_resource/content/1/Chapter%204%20Merise%20Method.pdf.

[Citc] Exercices avec corrigés d'informatique de gestion. URL: https://fsjescours.com/informatique-de-gestion-exercices-corriges/.

[Citd] Formulaire de navigation et actualisations. URL: https://www.bonbache.fr/formulaires-de-navigation-et-de-mise-a-jour-463.html.

[Cite] Merise method - a short introduction. URL: https://itselisecatherine.blogspot.com/2017/10/merise-method-part-1.html.

[DA20] Ysla Dela and D Angelica, Grace. 10 interfaces or parts of ms access application software. 2020.

[Dar] KUGEL Darius. La méthode merise. URL: https://mcours.net/cours/pdf/info1/Cours_lamethode_Merise_gxtrs.pdf#:~:text=Merise%20est%20une%20m%C3%A9thodologie%20d'informatisation%20dont%20les,cette%20m%C3%A9thode%20sont%20des%20personnalit%C3%A9s%20fran%C3%A7aises%20(Hubert.

[div13] sontakke divyag. Data types in microsoft access, 2013. URL: https://www.scribd.com/document/160968499/Data-Types-in-Microsoft-Access.

[ERN21] APPIAH ERNEST. Introduction to data types and field properties, 2021. URL: https://www.scribd.com/document/496243028Access2010IntroToDataTypesAndFieldProperties.

[heb22] Shakeel heba. Évolution historique des si, 2022. URL: https://dev.to/hebashakeel/minimal-cover-417l.

[Jil24] Lynch Jill. What is microsoft access?, 2024. URL: https://www.onlc.com/blog/what-is-microsoft-access/.

[Joh25] Terra John. What is microsoft access? database management simplified, 2025. URL: https://www.simplilearn.com/what-is-microsoft-access-article.

[JP21] ANDRÉ Jean-Philippe. Vba access programmer sous access, 2021. URL: https://www.editions-eni.fr/livre/vba-access-programmer-sous-access-9782409038129/microsoft-access-et-vba.

[Kin22] Yasar Kinza. primary key (primary keyword), 2022. URL: https://www.techtarget.com/searchdatamanagement/definition/primary-key.

[Lei] Gharani Leila. The visual basic editor (vbe).

[Luk25] Chung Luke. Microsoft access tips to avoid 17 common form design mistakes, 2025. URL: https://www.fmsinc.com/microsoftaccess/forms/common_mistakes.htm.

[MAG] MAGEFI. *Programmation en VBA (Structures de contrôle)*. Université de Bordeaux.

[Mic25a] Learn Microsoft. Add or change a table's primary key in access, 2025. URL: https://support.microsoft.com/en-us/office/add-or-change-a-table-s-primary-key-in-access-07b4a84b-0063-4d56-8b00-65f2975e4379.

[Mic25b] Learn Microsoft. Create a form in access, 2025. URL: https://support.microsoft.com/en-us/office/create-a-form-in-access-5d550a3d-92e1-4f38-9772-7e7e21e80c6b.

[Mic25c] Learn Microsoft. Create a table and add fields, 2025. URL: https://support.microsoft.com/en-us/office/create-a-table-and-add-fields-8fdc65f9-8d40-4ff5-9212-80e6545e8d87.

[Mic25d] Learn Microsoft. Create a user interface (ui) macro, 2025. URL: https://support.microsoft.com/en-us/office/create-a-user-interface-ui-macro-12590d3b-b326-4207-bfe5-19234f53f08b.

[Mic25e] Learn Microsoft. Create, edit or delete a relationship, 2025. URL: https://support.microsoft.com/en-us/office/create-edit-or-delete-a-relationship-dfa453a7-0b6d-4c34-a128-fdebc7e686af.

[Mic25f] Learn Microsoft. Data types for access desktop databases, 2025. URL: https://support.microsoft.com/en-us/office/data-types-for-access-desktop-databases-df2b83ba-cef6-436d-b679-3418f622e482.

[Mic25g] Learn Microsoft. Dim, instruction, 2025. URL: https://learn.microsoft.com/fr-fr/office/vba/language/reference/user-interface-help/dim-statement.

[Mic25h] Learn Microsoft. Guide to designing reports, 2025. URL: https://support.microsoft.com/en-us/office/guide-to-designing-reports-876a6d27-59aa-467a-8240-ee6e01597291.

[Mic25i] Learn Microsoft. Guide to table relationships, 2025. URL: https://support.microsoft.com/en-us/office/guide-to-table-relationships-30446197-4fbe-457b-b992-2f6fb812b58f.

[Mic25j] Learn Microsoft. Guide to the access user interface, 2025. URL: https://support.microsoft.com/en-us/office/guide-to-the-access-user-interface-cd8eca71-78a1-484f-897b-fc80b1ac77ec.

[Mic25k] Learn Microsoft. Introduction aux états dans access, 2025. URL: https://support.microsoft.com/fr-fr/topic/introduction-aux-%C3%A9tats-dans-access-e0869f59-7536-4d19-8e05-7158dcd3681c.

[Mic25l] Learn Microsoft. Introduction to controls, 2025. URL: https://support.microsoft.com/en-us/office/introduction-to-controls-4a8cf5f2-d739-4ae9-b1e0-510c3f4d6975.

[Mic25m] Learn Microsoft. Introduction to data types and field properties, 2025. URL: https://support.microsoft.com/en-us/office/introduction-to-data-types-and-field-properties-30ad644f-946c-442e-8bd2-be067361987c.

[Mic25n] Learn Microsoft. Introduction to queries, 2025. URL: https://support.microsoft.com/en-us/office/introduction-to-queries-a9739a09-d3ff-4f36-8ac3-5760249fb65c.

[Mic25o] Learn Microsoft. Learn the structure of an access database, 2025. URL: https://support.microsoft.com/en-us/office/learn-the-structure-of-an-access-database-001a5c05-3fea-48f1-90a0-cccaa57ba4af.

[Mic25p] Learn Microsoft. Make controls stretch, shrink, or move as you resize a form, 2025. URL: https://support.microsoft.com/en-au/office/make-controls-stretch-shrink-or-move-as-you-resize-a-form-51fd88e0-43d3-4070-a298-18ba273f4cf8.

[Mic25q] Learn Microsoft. Resources for learning about access forms, 2025. URL: https://support.microsoft.com/en-us/office/resources-for-learning-about-access-forms-c0611287-1b19-4ceb-a454-069ea6bc3806.

[Mic25r] Learn Microsoft. Update history for office ltsc 2024 and office 2024, 2025. URL: https://learn.microsoft.com/en-us/officeupdates/update-history-office-2024.

[Mic25s] Learn Microsoft. Utiliser la fenêtre propriétés, 2025. URL: https://learn.microsoft.com/fr-fr/office/vba/language/reference/user-interface-help/use-the-properties-window.

[Mic25t] Learn Microsoft. Video: Create navigation forms, 2025. URL: https://support.microsoft.com/en-us/office/video-create-navigation-forms-67773fba-1ddb-4624-a07f-fc84e8c82de8.

[Mic25u] Learn Microsoft. Événement form.beforeupdate (access), 2025. URL: https://learn.microsoft.com/fr-fr/office/vba/api/access.form.beforeupdate-event.

[Moh13] NEMICHE Mohamed. *Analyse et Conception du Système d'Information (Merise)*. Faculté Polydisciplinaire de Ouarzazate, 2013.

[MON] UNIVERSITY MONTCLAIR, STATE. Introduction to microsoft access 2016.

[Mur22] Mignerat Muriel. Minimal cover, 2022. URL: https://ecampusontario.pressbooks.pub/adm1770sandbox/chapter/evolution-historique-des-si/.

[Reg] Institute Regional, Training. Ms access.

[RT83] Arnold Rochfeld and Hubert Tardieu. Merise: An information system design and development methodology. *Information & Management*, 6(3):143–159, 1983.

[Sbi05] Boubker Sbihi. Analyse et conception d'un systeme d'information avec la methode merise: Cas d'une bibliotheque universitaire. *Resi*, 14(1), 2005.

[Scr] Screenshot. Microsoft access 2019.

[Sou14] Robert Soulé. Functional dependencies and finding a minimal cover. *línea]. Available: http://www. inf. usi. ch/faculty/soule/teaching/2014-spring/cover. pdf.[Último acceso: 2017]*, 2014.

[tea24] Pandora FMS team. Un bref historique des systèmes d'exploitation, 2024. URL: https://pandorafms.com/blog/fr/historique-des-systemes-dexploitation/.

[Ted14] Matti Tedre. *The science of computing: shaping a discipline*. CRC Press, 2014.

[TRC89] Hubert Tardieu, Arnold Rochfeld, and René Colletti. *La méthode MERISE : Principes et outils*. Editions d'Organisation, France, 1st edition, 1989.

[Vol] Michel Volle. Evolution du rôle du système d'information: du concept au processus.

[WE14] Adrienne Watt and Nelson Eng. *Database design*. BCcampus, 2014.

Appendix

دليل المادة التعليمية			
اسم المادة: إعلام آلي 2			
الميدان	العلوم الاقتصادية التسيير، وعلوم التجارية	الفرع	علوم اقتصادية
التخصص	/	المستوى	الثانية
السداسي	الثالث	السنة الجامعية	2022/2021

التعرف على المادة التعليمية			
اسم المادة	إعلام آلي 2	وحدة التعليم	الأفقية
عدد الأرصدة	01	المعامل	01
الحجم الساعي الأسبوعي	01:30	المحاضرة (عدد الساعات في الأسبوع)	/
أعمال م/تط (عدد الساعات في الأسبوع)	01:30	أعمال م/ت (عدد الساعات في الأسبوع)	/

مسؤول المادة التعليمية			
الاسم، اللقب	.....	الرتبة	.....
تحديد موقع المكتب	.....	البريد الالكتروني	.....
رقم الهاتف	.....	توقيت الدرس ومكانه	.....

وصف المادة التعليمية	
المكتسبات	• مبادئ التعامل مع الحاسوب • مبادئ الرياضيات والإحصاء • إعلام ألي 1
الهدف العام للمادة التعليمية	• تعميق فهم الطالب لبرنامج Access وتمكينه من التعامل معه واستعماله فيما يخدم
أهداف التعلم (المهارات المراد الوصول إليها)	• إنشاء قواعد البيانات • تشغيل برنامج Access • إدارة والتعامل مع قواعد البيانات • البرمجة باستخدام VBA

محتوى المادة التعليمية	
المحور الأول	لمحة تاريخية حول تطور أنظمة المعلومات وطرق تصميمها
المحور الثاني	عموميات حول برنامج Access
المحور الثالث	إنشاء الجداول في قاعدة البيانات
المحور الرابع	إنشاء النماذج
المحور الخامس	إنشاء الاستعلامات Requetés
المحور السادس	إنشاء الحالات états
المحور السابع	إنشاء قائمة للتطبيقات Création d'un menu pour l'application
المحور الثامن	البرمجة باستخدام vba

طريقة التقييم				
التقييم بالنسبة المئوية	العلامة		الوزن النسبي للتقييم	
امتحان جزئي	5	20	وزن الأعمال الموجهة والتطبيقية	% 25
أعمال موجهة (البحث : إعداد/إلقاء)	-			
أعمال تطبيقية (البحث : إعداد/إلقاء)	6			% 30
المشروع الفردي	2			% 10
الأعمال الجماعية (ضمن فريق)	-			-
خرجات ميدانية	-			-
المواظبة (الحضور / الغياب)	6			% 30
عناصر أخرى (المشاركة)	1			05%

تدرس المادة في شكل أعمال موجهة و تقييمها يكون بمراقبة مستمرة يقاس معدل المادة بمختلف الأنشطة التي تنجز في المراقبة المستمرة

معدل المادة	معدل التقيم في المراقبة المستمرة=
Moy.M	***= Note Td x %100***

المصادر والمراجع		
المرجع الأساسي الموصى به :		
عنوان المرجع	المؤلف	دار النشر و السنة
1 *VBA ACCESS 2002 PROGRAMMER SOUS ACCESS*	*Michele Amelot*	*ENI édition, 2001*
2 *J'apprends à me servir de ACCESS 2007 utilisation et programmation*	*Florence Fessy*	*IOS édition, 2010*
3 *ACCESS 2 programing by example*	*Greg M. perry*	*QUEédition, 1994*

التوزيع الزمني المرتقب لبرنامج المادة		
الأسبوع	محتوى المحاضرة	التاريخ
الأسبوع الأول	لمحة تاريخية حول تطور أنظمة المعلومات وطرق تصميمها (التركيز على طريقة merise) تعريف وشرح طريقة merise مدخل المراحل مدخل المستويات	
الأسبوع الثاني	شرح قاموس المعطيات ومختلف أنواعه (الموسع والبسيط) شرح النموذج التصوري للمعطيات مع أمثلة	
الأسبوع الثالث	النموذج المنطقي للبيانات وطرق الانتقال من النموذج التصوري الى النموذج المنطقي علاقة اب ل اب علاقة اب ل ابن علاقة ابن ل ابن	
الأسبوع الرابع	**عموميات حول برنامج Access:** تعريف برنامج أكسس تنصيب برنامج أكسس تشغيل برنامج أكسس شرح تبويبات برنامج أكسس إغلاق برنامج أكسس كيفية إنشاء قاعدة بيانات فارغة شرح le volet de navigation	
الأسبوع الخامس	**إنشاء الجداول في قاعدة البيانات** - خيار الإنشاء mode création - نوع البيانات types des données - تحديد خصائص الحقول - المفتاح - العلاقة بين الجداول - الحجز إدخال التسجيلات	
الأسبوع السادس	إنشاء النماذج	

الأسبوع السابع	إنشاء الاستعلامات requetés	
الأسبوع الثامن	إنشاء الحالات états	
الأسبوع التاسع	إنشاء الماكرو	
الأسبوع العاشر	إنشاء قائمة للتطبيقات création d'un menu pour l'application	
الأسبوع الحادي عشر	البرمجة باستخدام vba	
الأسبوع الثاني عشر	البرمجة باستخدام vba	
الأسبوع الثالث عشر	البرمجة باستخدام vba	
الأسبوع الرابع عشر	البرمجة باستخدام vba	
الأسبوع الخامس عشر	البرمجة باستخدام vba	
الأسبوع الخامس عشر	مراجعة وتطبيقات	

الأعمال الشخصية المقررة للمادة

1. القيام بحل تمارين مقدمة على شكل واجبات منزلية؛
2. استجواب تقييمي؛
3. تقييم الأسئلة التفاعلية للطلبة عبر منصة Moodle.
4. الحضور والتفاعل في منصة Moodle.
5. إنشاء دردشة ومنتدى في منصة Moodle للتعليم الالكتروني.
6. جمع بيانات حول ظاهرة ما و تبويبها حسب نوع البيانات
7. قيام كل طالب بانجاز محاكاة لما تعلمه في المادة من خلال إنشاء قاعدة بيانات أو قوائم أو تطبيقات.

مصادقات الهيئات الإدارية والبيداغوجية

رئيس القسم	مسؤول ميدان التكوين أ.د قادري علاء الدين	الأستاذ مسؤول المادة	نائب العميد الملكف بالبيداغوجيا أو مدير الدراسات

ملاحظة هامة: بعد المصادقة على دليل المادة في بداية كل سداسي يتم نشره على الموقع الرسمي للمؤسسة الجامعية