

Pavol Jozef Šafárik University in Košice
Faculty of Medicine



Introduction to MS Access

Databases not only for medical students

Jaroslav Majerník
Andrea Kačmariková



Košice 2025

This university textbook was created with the contribution of the Cultural and Educational Grant Agency of the Ministry of Education, Research, Development and Youth of the Slovak Republic as part of the KEGA project no. 003UPJŠ-4/2023, *"Innovation of the teaching process in medical and non-medical study programs using medical simulation tools and virtual reality"*.

Introduction to MS Access Databases not only for medical students

University textbook

Authors:

Jaroslav Majerník and Andrea Kačmariková

Department of Medical Informatics and Simulator Medicine, Faculty of Medicine, Pavol Jozef Šafárik University in Košice

Reviewers:

Assoc. Prof. Ing. Miroslav Kvaššay, PhD.

Department of Informatics, Faculty of Management and Informatics, University of Žilina

prof. Ing. Radovan Hudák, PhD.

Department of Biomedical Engineering and Measurement, Faculty of Mechanical Engineering, Technical University in Košice

This text is published under the Creative Commons 4.0 license - CC BY NC ND ("Attribution - Do not use commercially - Do not process").



The authors are responsible for the professional and linguistic aspects of this university textbook. The manuscript has not undergone editorial or linguistic editing.

Available at: <https://portal.lf.upjs.sk/articles.php?aid=214>
<https://unibook.upjs.sk>

Publication date: 08. 09. 2025

DOI: <https://doi.org/10.33542/IMA-0425-5>

ISBN 978-80-574-0425-5 (e-publication)

Preface

The university textbook entitled *"Introduction to MS Access – Databases not only for medical students"* is intended for medical students, students of nontechnical universities and all beginners in working with databases. It introduces the readers to the issues of database data processing and explains the basic principles that need to be known when creating relational types of databases. It is therefore suitable for those who have not yet worked with databases, or who are starting to think about managing their data using their own, perhaps even their first, database. However, even more experienced computer users can also find useful information here. The university textbook is written in such a way that its content can be mastered with basic knowledge of working with a computer and programs running in the environment of one of the versions of the MS Windows operating system.

The subject of relational databases, which also includes MS Access databases, is explained here using examples implemented in the MS Access 2019 application. However, the foundational principles discussed are similar and applicable to other versions of this software product. In the individual chapters, you will learn when it is advantageous to use databases, but also how to design and create your own database. The most commonly applied user procedures, methods of designing and creating the main database objects, working with them, as well as the possibilities for their optimization from the perspectives of functionality and purposefulness are explained here. At the same time, the reader will simultaneously gain a sufficient understanding of the capabilities of the MS Access software application. Based on the information provided here, it can be assumed that even a complete beginner can become a full-fledged user of this office application.

Authors

Contents

Introduction	9
1 Databases	13
1.1 Data organization models in databases	13
1.2 Why MS Access	14
1.3 Installing MS Access from Microsoft 365	16
1.4 Database design	17
2 MS Access	21
2.1 Launching and closing the program	21
2.2 New database	22
2.3 Environment and controls	24
2.4 Database objects	26
3 Tables	29
3.1 Creating and viewing a table	30
3.2 Database structure	31
3.3 Personal Data table	33
3.3.1 Design view of the Personal Data table	33
3.3.2 Data types of database fields	35
3.3.3 General properties of the fields of the Personal Data table	37
3.4 Examinations table	46
3.4.1 Design view of the Examinations Table	47
3.4.2 General properties of the fields of the Examinations table	48
3.5 Biometric data table	51
3.5.1 Design view of the Biometric Data table	52
3.5.2 General properties of the fields of the Biometric Data table	53
3.6 Primary key of database table	59
3.6.1 Primary key of the Personal Data table	59
3.6.2 Primary key of the Examinations table	61
3.6.3 Primary key of the Biometric Data table	61
3.6.4 Troubleshooting selected issues when setting up the primary key . . .	62

3.7	Relationships between tables	63
3.7.1	Types of relationships between tables	63
3.7.1.1	Relationship 1:1	64
3.7.1.2	Relationship 1:N	64
3.7.1.3	Relationship M:N	64
3.7.2	Relationship between the tables Personal Data and Examinations . .	65
3.7.3	Relationship between the tables Personal Data and Biometric Data .	69
3.7.4	Modifications of relationships	70
3.8	Datasheet view of the Personal Data table	71
3.9	Datasheet view of the Examinations table	74
3.10	Datasheet view of the Biometric Data table	76
3.11	Import of external data	77
4	Forms	85
4.1	Automatic form creation	86
4.2	Form view of the Personal data form	88
4.3	Custom form design	90
4.4	Creating forms using the Form Wizard	91
4.4.1	Columnar layout of form fields	92
4.4.2	Tabular layout of form fields	94
4.4.3	Datasheet layout of form fields	95
4.4.4	Justified layout of form fields	95
4.5	Form Design View	96
4.5.1	Form header design	98
4.5.2	Form details design	100
4.5.3	Subform	103
4.5.4	Form footer design	106
4.5.4.1	Go to next record button	106
4.5.4.2	Go to previous record button	107
4.5.4.3	Add new record button	108
4.6	Advanced forms	109
4.6.1	Form with subform	109
4.6.2	Linked forms	112
4.7	Navigation form	114
5	Data selection and sorting	117
5.1	Sorting and filtering	117
5.2	Record filtering types	119

6	Queries	123
6.1	Advanced filter or sorting	124
6.2	Creating queries using the wizard	125
6.2.1	Simple query	126
6.2.1.1	Patients from Kosice	126
6.2.1.2	Patients born before 2000	131
6.2.1.3	Patients with the initial letter M in their surname	132
6.2.1.4	Selection query from multiple tables	133
6.2.2	Crosstab query	135
6.2.3	Find duplicates query	138
6.2.4	Find unmatched records query	141
6.3	Parametric query	144
6.4	Action queries	145
7	Reports	147
7.1	Creating and displaying reports	147
7.2	Automatic report creation	149
7.3	Creating reports using the wizard	151
7.3.1	List of examinations	151
7.3.2	Personal data by health insurance	155
7.4	Labels	157
	Bibliography	163
	Index	165

Introduction

With the continuous development of information technologies and the expansion of the network services portfolio, there is also a continuous increase in the amount of processed information that surrounds us in various areas of life and in various forms. In general, we encounter information on a daily basis, both in verbal, visual and digital form. The information generated every day, at least those necessary for us, must be stored, organized, and always available when needed.

Health-relevant information, including that related to solving research problems, is particularly valuable in the field of medicine and healthcare. Without it, for example, it is not possible to establish the correct diagnosis of a patient, draw the right conclusions or recommendations in clinical experiments, etc. Most of us have become accustomed to storing valuable information in tables. We often use available spreadsheet processors for this, but when there is a lot of information and data, sorting, processing, and providing it becomes difficult, because not only do we lose track of it, but also important information about the relationships between them becomes inaccessible. In such a case, it is appropriate to change the way of thinking about the way of working with data and consider using applications and solutions from database systems group.

From the user's point of view, a database system represents a complex system, which is most often composed of a series of multiple tables, forms, queries, reports, but also other useful objects and functions for advanced work with processed data. There are various database systems, whose environments differ, but their basic principles and the essence of the procedures used remain common.

The aim of this university textbook, that serves as accompanying texts for practical exercises, is to provide medical students with the teaching material necessary for studying subjects that explain and clarify the principles of data processing in healthcare and medicine, which include, for example, the subjects such as Medical Informatics, Informatics for Dental Medicine, as well as many others included in the curricula of individual study programs.

Students will become familiar with the Microsoft Access database system (MS Access), learn to independently create new and modify existing database objects, correctly enter and modify data into the relevant fields of individual tables, change the properties of these fields, or use data from tables using other database objects. An important part of working with a database is also the approach to selected problems from routine clinical practice through resolving simple and illustrative examples, with the main aim still being to explain and

understand the background and principles of working with data, regardless of the system or application in which they are processed. This is intended, on the one hand, to contribute to increasing the use of information systems in healthcare, eliminating concerns about their use among healthcare professionals, but also to establish prerequisites for formulating correct requirements for the development of new functions in information systems that take into account the needs and requirements of healthcare professionals and the healthcare system as a whole.

Individual chapters of this university textbook are dedicated to basic database objects, the intention of explaining the meaning and procedure for creating a given object in the form of solved examples with a detailed descriptions of the most important control and functional elements. For better clarity and reproducibility of the described procedures, the text parts are supplemented with visual documentation of specific phases of design, use and modification of the relevant database objects.

We recommend that the reader, while reading this publication, practice the procedures listed and actively create objects in his own database. Each chapter devoted to the basic objects of a relational database also includes separate practical tasks that the reader can use to expand the functionality and practical applicability of the sample database. We also draw the reader's attention to the review questions that are listed at the end of each such chapter.

Chapter 1 (page 13) explains the basic meaning of databases. It briefly describes some traditional solutions for data organization models in databases as used in various fields with the corresponding computing equipment. It also explains to regular users of office suite applications that they do not need to worry about databases and that they can start with their first database in the MS Access environment.

A general description and introduction to the MS Access workspace are given in the Chapter 2 (page 21). In this chapter, the reader will find, for example, basic information on launching the application, displaying the most commonly used control elements, how to create a new database, using predefined database templates, a brief description of database objects used when working with, as well as many other features and options offered by the MS Access software application.

Chapter 3 (page 29) focuses on the basic database object, the database table. The chapter describes, for example, what to consider when creating database tables, how to edit and modify them in the design view, or how to insert data into table fields using their datasheet view. It also explains how to create an input mask for fields of various data types, how to correctly define other properties of individual fields, what the primary key of a database table is, and how to create relationships, i.e., mutual connections between database tables and their records. Additionally, the reader will learn how to set the primary key for a database table and how the properties of the field change when it becomes the primary key of a database table. The chapter will also address solutions to selected problems that may arise in designing the primary key. Besides the primary key, the possible relational

relationships between tables in an MS Access database are defined in this chapter alongside the individual tables.

Information (structured data) is entered into the database more easily and conveniently through forms. The next Chapter 4 (page 85), is therefore dedicated to these objects of relational databases. It explains to the reader how to create various types of forms, defines the display possibilities of forms, and also describes the basic design of several forms so that the database user can work with them comfortably efficiently. When creating one's own form design, individual commands aiding in formatting, arranging, and designing control elements and fields are provided. It describes not only how to insert and edit database records through a form, but also what the advantages of forms and subforms are.

From the extensive data stored in the database, or in its tables, we can select and display those that meet specific requirements or need to be worked with at a given moment by filtering and sorting. Thus, Chapter 5 (page 117) explains how to use the sorting and filtering functions of database records. It defines the individual types of filters and their advantages and disadvantages when filtering information contained in database systems. For sorting and filtering frequently used and analyzed data, queries are mainly used instead of repeatedly defined filter functions in database systems.

Queries are separate objects of database systems designed to select information, data from various database tables, gather them, and display them according to specified specific criteria. We will learn more about queries in Chapter 6 (page 123) of these university textbook for practical exercises. We will also learn how to create various types of queries and correctly set criteria for individual database fields.

In the last chapter, Chapter 7 (page 147), we will look at reports that are inseparably linked to database systems. We will learn how to create reports using the report wizard and create our own report design and labels, which can be used, for example, to label samples of biological material taken from patients. We will also describe and explain the individual types of report views and learn how to check a report before printing or sending it outside the database.

Two fundamental terms, whose meanings should not be confused with each other, will be frequently used throughout the chapters of this university textbook. These are the terms "information" and "data". We will understand information to mean everything that gives us or someone or something a message about things or events that have happened, are happening, or will happen. From an computer science perspective, the basic units of information are bit (*b*) and Byte (*B*). On the other hand, data will be understood as any element with informational value, usually processed by a computer. These are various values, numbers, characters, symbols, graphs, etc. In other words, the information consists of data.

To study the use of the MS Access database system when working with patient data, video files are available on the Portal of UPJŠ LF. The link to the video is directly in the text. Thanks to the hypertext link to the UPJŠ LF Portal, viewing the tutorials will be easy and fast.

Explanations:

In this university textbook, the presentation of the topics is continuously supplemented by additional notes, references to illustrative video demonstrations, assignments of practical tasks for further database work, and review questions for verifying the reader's knowledge from theoretical and practical insights. Such information is graphically differentiated according to the following visual convention.

**Note**

Additional information and suggestions for reflecting on the presented solutions.

**Video**

Link to video demonstrations related to the described practical solution to the problem.

**Practical Task**

Several practical tasks for independent practice.

**Review Questions**

Summary review questions to verify understanding of the explained topic.

Chapter 1

Databases

The term "database" can simply be understood as an organized collection of data, obtained for a specific purpose, and typically arranged in a structured digital form. Databases collect data that can be further processed and used according to the needs and requirements of their users. There are several definitions of the term database. One of them is the following.

A database is a collection of information organized according to certain criteria, most often in the form of a table, to allow the most advanced manipulation of this information.

Historically, it can be said that people have always worked with information, but it was the advent and improvement of computers that elevated their processing to the level we know today. And it is constantly improving.

Databases were used, for example, in the form of various card files and catalogs, which represented a source of data intended for processing. Examples include library card files, patient health records, product catalogs, etc. However, we encounter data processing in almost all our activities. We have bank accounts in banks, we find real estate in cadastral lists, we are registered as customers with various merchants, we compile lists of tasks, contacts, books, movies, music. Examples can be given from almost every area, and it does not matter what kind of information is involved, but the important thing is that as soon as such information is carefully recorded in a database, it can then be processed and evaluated using computing resources, and therefore in many cases fully automatically.

1.1 Data organization models in databases

Databases and information systems utilizing information stored in databases, the principle of their operation, as well as the demands on the hardware equipment itself, depend on the used models of logical organization of stored and processed data. From the perspective of the history of computer-supported data processing, several models can be characterized that have been and are being developed and used in connection with the evolution of computing technology and the data processing methods themselves.

- **Hierarchical model** – its basis is a tree structure similar to the structure of a hard disk directory, where subdirectories are created in the main directory, further subdirectories are created in them, etc. By mapping the branches created in this way, the desired information can be found, i.e. data that is in a certain relationship to the initial concept. The disadvantage of the hierarchical model is that it does not allow the creation of simple relationships between data stored in different branches of the tree, and in case of changes in requirements, the entire database structures must be revised.
- **Network model** – uses pointers to express relationships between database items, which form a network structure. Unlike the hierarchical model, it can relatively easily implement relationships of type $M : N$, i.e. multiple data or records (M) relate to multiple other data or records (N).
- **Relational model** – is a very widespread type, where all data in the database is represented in a uniform way, namely as values stored in tables. The fixed structure of the record presents one row of a table composed of multiple fields (columns). Relationships between tables are created through a "common field" describing also the database structure. One of the advantages is the rapid updating of data, as it is sufficient to enter/correct data in one object of the database, and the change will be reflected throughout the entire database, i.e. in every object of the database system that processes the data and provides it to the user.
- **Object-oriented model** – a modern database model based on object programming principles. Database records contain, in addition to the data itself, their own methods that operate with this data. Object-oriented databases efficiently reflect the requirements of connecting images, sound, text, and graphics.

In general, software that stores information or data in tables and can simultaneously work with multiple tables, is called a database system or database management system. It consists of a system for data organization and a database. A database management system is software that provides all the required features of a database system and manipulates data. A large amount of information is stored, processed and sorted in a database system. For example, the database system of a health insurance company records information about contracted doctors, but also about their patients.

1.2 Why MS Access

MS Access represents a database system designed for working with databases, which, due to its sophisticated user interface, is suitable for a wide range of regular users. They can use it even without needing to know anything about the principles of database operation, using pre-created templates. Of course, MS Access supports collaboration with other applications of the Microsoft Office or Microsoft 365 packages, which, for example, allow data to be

transferred without needing to manually rewrite it. Among other things, it also enables advanced data collection, such as through e-mail or web forms.

The basis of every database is data. In relational databases, which include MS Access databases, data is stored in tables. It might seem that data could be recorded "more simply" into a table in the MS Word text editor or using the MS Excel spreadsheet processor. For example, information about patients and their examinations can be written into the table rows one below the other, adding a new row (a record) with each examination. Certainly, this is possible, and to some extent, it might suit some users with smaller data collection requirements.

On the other hand, the situation is not as simple as it might seem. We must realize that the data (information stored in the table) represents only one part of the database. Other components of the database ensure that the data needed at any given time is extracted from the database. Only by combining these components can we obtain a system that provides us with all the advantages of computer/machine data processing.

If we were to record individual records in MS Word, MS Excel, or other similar editors, it could happen that during subsequent data processing and evaluation, we would spend a lot of time searching for records or correcting possible typos in the data stored multiple times across records. Similarly, the data entry itself could be lengthy depending on the size of the database, as many data would be recorded repeatedly (duplicately). To explain why creating a database in such editors is insufficient, we will provide the following example.

Imagine that we are recording patients and their examination records. In a table of MS Word or MS Excel, for each patient visit, we should record information such as their first name, surname, title, address, phone number, date of birth, insurance number (personal identification number), examination date, diagnosis, findings, prescription, etc. Therefore, one row of the table represents one visit, and with each new patient, we will add a new row to the table. It is already apparent that if the patient comes for another examination, it will be necessary to repeatedly fill out all his/her data (first name, surname, title, address, etc.) in the next row of the table. Over time, this will undoubtedly become very inconvenient, not to mention that typing errors could occur, which will be very difficult to identify as the number of records grows. For example, an error in the name or insurance number (personal identification number) might lead the user (during advanced computer processing) to treat records as examinations of two different patients. This way, several erroneous data entries could appear in the database, ultimately negatively affecting the reliability and credibility of the processed information.

To avoid such inconveniences, data in relational databases is stored in several separate tables, between which logical relationships, i.e., relations, are defined. Our previous example can then be implemented by creating two separate tables. The first table will contain the names and contacts of patients, and the second their examinations. Personal data of the patient (name, surname, title, etc.) will be recorded only once, regardless of the number of examinations the patient undergoes. The examinations table will instead only record current

information about the examination, i.e., examination date, findings, etc. If we define a field in both tables to uniquely identify individual records (such as insurance number, personal identification number, social security number, record number/patient number, etc.), then we can link these tables using a relationship (see Chapter 3).

It might seem like a complicated way to solve a problem, but if we make, for example, a mistake in the patient's name, it will just mean having an incorrect name in the database. However, it will not affect the reliability of the database because all examinations will still belong to the same patient. Moreover, the mistake in this name can be corrected at any time, and at one place in the database in one record. The change will automatically be reflected in all examination records the patient has undergone.

Relational databases have many other advantages that users appreciate when the need to process a larger amount of data. These include significantly faster data search possibilities, options for creating various analyses and reports, more efficient ways of data retrieval, options for data sharing among colleagues, and considerably larger volumes of stored data. MS Access, as one of the several available office database tools, provides regular users with much better ways to work with information than standard text editors or spreadsheet processors.

1.3 Installing MS Access from Microsoft 365

The database system MS Access is available not only as a standard component of the MS Office package for students but also within the Microsoft 365. Its installation on devices with the MS Windows operating system (i.e., MS Access from Microsoft 365 is for PC only) is possible directly from the student account that students of Pavol Jozef Šafárik University in Košice receive upon enrollment. All students are assigned a unified login in the form of `AIS_ID@upjs.sk`, and in electronic mail communication, they also use an alias in the form of `firstname.surname@student.upjs.sk`. After logging into their student account on the Microsoftonline¹ website (M365 Copilot), through account information or directly via the **App** - App Launcher (Figure 1.1), they have access to Microsoft 365 apps. By clicking on the **Apps** link, they will be taken to the home page, where they will also find an **Install apps** link in the upper right corner to install licensed components of this office suite.

Students can install applications that are part of Microsoft 365 on PC or Mac devices, as well as on tablets or smartphones. However, MS Access is only available for PC, i.e. a personal computer or laptop with the MS Windows operating system. To install individual products or all licensed components, it is necessary to follow the installation wizard's instructions. The installed Microsoft 365 applications will be part of the **Start** menu after the installation is completed, including the MS Access database system.

¹Microsoftonline, <https://login.microsoftonline.com/> or alternatively <https://www.office.com/>

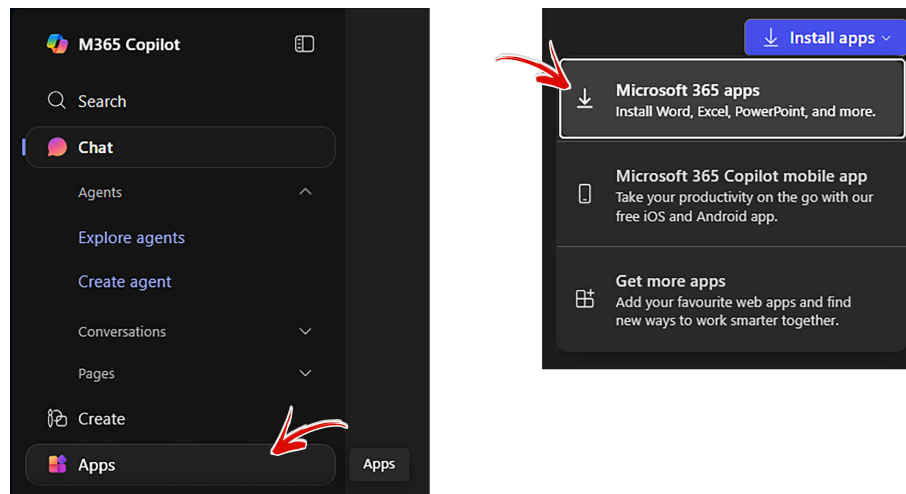


Figure 1.1: Access to Microsoft 365 apps.

1.4 Database design

In order for information to be processed and evaluated efficiently, it is not enough to just have a more or less sophisticated database tool – an application/program. A very important factor influencing the functionality and ultimately the satisfaction of users is the design of the database itself. A well-designed database ensures not only the fulfilment of all user requirements, but also offers possibilities for modification and changes that may arise during its practical usage.

It is difficult to establish a unique procedure for database design. However, it is possible to recommend adhering to several basic steps and principles. The user should primarily consider the very purpose of the database. It is necessary to answer several basic questions. *What information will be recorded? What is the purpose of collecting this information? Who will use the database? Will it be a single person or several collaborators? How and when will the database be used?*

Furthermore, it is necessary to characterize in detail all the required information. All data related to the purpose of the database should be recorded, either based on the existing experiences of the user, where the information has already been processed, for example in paper form, or based on the expected benefits the resulting database is to fulfil and provide. If we want to gather information on patients' health conditions, then we should characterize items similar to those found in health records. Individual items can be written down on paper, perhaps in order of importance. Each recorded item can represent a potential column (field) in the database table.

After completing the list of all required items, it is necessary to define tables. In each database, the information should be divided in such a way that it is not recorded redundantly and ensuring that it is accurate and complete. This is achieved by breaking down the information into several categories, which then form the basis for creating individual tables. For example, patient records could be divided into personal data, emergency information,

vaccinations, examinations, etc. The database structure should be designed so that if changes to the data are needed, these changes are made in only one place (not in different tables). For instance, the address of a patient in the patient list, the blood type or allergies in emergency related information, etc. It is also necessary to define the fields of the tables considering future possibilities for sorting and filtering data. Therefore, instead of entering the patient's name in one column, it is better to use for example three columns marked as title, first name, and surname. Or similarly, an address can be registered in columns including street, description number (street number), postal code, town or city, country, state etc.

Each database table should contain a record identifier, i.e. the so-called primary key (also referred to as the main key). In practice, it is one of the table fields, or a group of fields, that uniquely identify each row (record). MS Access uses primary key fields, among other things, to quickly link data from multiple tables or to merge data. Duplicate values cannot exist in the table's primary key. Therefore, we use a field whose values are unique and do not change as the primary key of the table. In a database that uses more than one table, the primary key of a table can be used as a reference data in other tables. For identifying patients, an insurance number or personal identification number can be used as the primary key, and for defining patient examinations, a random unique number can be used, which will be different for each outpatient visit, etc.

Creating tables does not end the design of a database. It is necessary to define their mutual relationships (relations), i.e. to create their mutual interconnection. To establish a mutual relationship between two tables, we use the fields of these tables. For example, if the primary key in the patient list is the insurance number, then this field is also added as another column in the examinations table, where it represents a so-called "foreign key" (a 1 : N relationship can be created in this way, see Chapter 3). MS Access can then use the patient's identification in the patient list to find records of his/her examinations in the examination list.

After completing the design of all necessary tables, fields, and relationships, it is advisable to fill the tables with sample data and try to work with stored information. This way, we can identify potential design issues, absence of important data, undesirable duplicates, redundancy of data, improvement possibilities, etc. By identifying these factors in a timely manner, we can prevent later complications and ensure optimization and possible enhancements to the resulting database. The database can be further supplemented with other objects (forms, queries, reports, macros or modules) and gradually optimized to the desired appearance and functionality with a clear and intuitive user interface (environment). All steps in designing the basic structure of the database in the MS Access environment will be characterized in detail in the following chapters of this publication and explained using specific examples.



Review questions

1. What do we mean (understand) by the term database?
2. What is the difference between a database system and a database?
3. What is a relational database?
4. What are the advantages of a relational database?
5. What is the main difference between a spreadsheet and a database system?
6. How can records in a database be identified?

Chapter 2

MS Access

The database system MS Access can be obtained as part of the Microsoft Office suite (various versions) or Microsoft 365 (Chapter 1.3). In this publication, the topic of databases in MS Access is explained using the 2019 version, with the basic principles and rules being valid for other versions of this software product, with a very few exceptions.

2.1 Launching and closing the program

In general, there are multiple ways to launch the MS Access database system, just as it applies to most applications running under the MS Windows operating system. Moreover, each user has their own habits of managing their computer, using installed programs or documents. Therefore, it is difficult to specify a so-called "universal" guide to launch the MS Access database tool, but probably the most common way is through the **Start** menu (by clicking the **Start** button or pressing the **Windows** key on the keyboard and typing the text **access**) of the MS Windows operating system, for example, as shown in Figure 2.1.

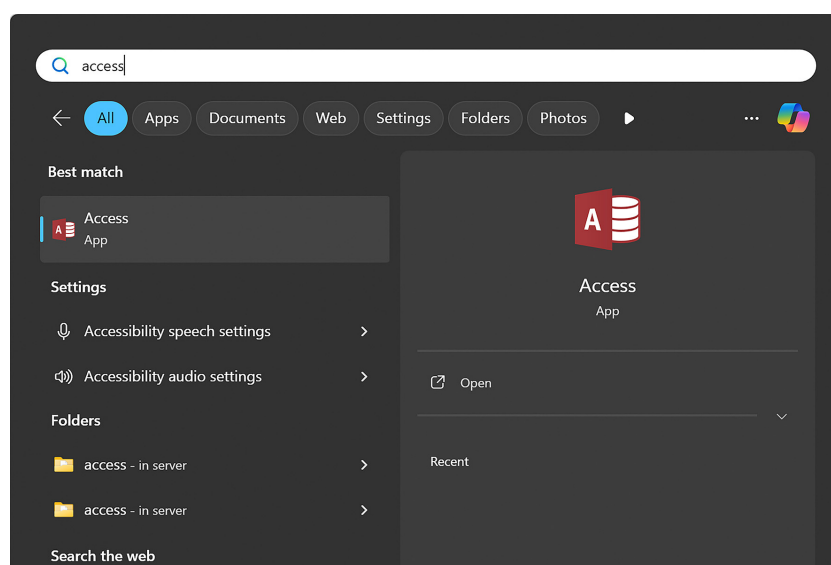


Figure 2.1: Launching MS Access using the Start menu of the MS Windows operating system.

The names of the displayed items may of course vary, depending on the version of the MS Office suite, operating system, as well as their language versions. Access to MS Access can be facilitated by, for example, creating a desktop shortcut or pinning the program to the taskbar of the operating system.

Upon launching the program, the initial (introductory) window of MS Access (Figure 2.2) appears, where it is possible to create a new database, use one of the database templates, or open an already existing user's database (if the user has already worked with MS Access).

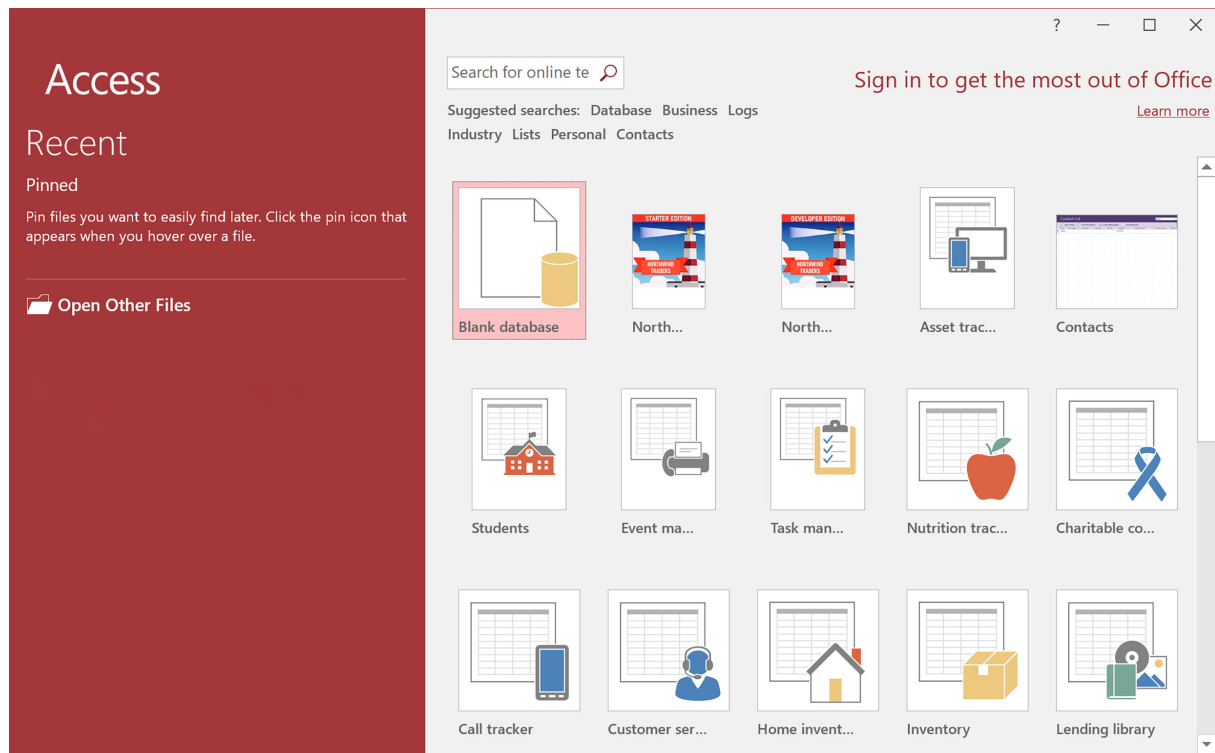


Figure 2.2: Initial MS Access window for creating a new or opening an existing database.

Ending work in MS Access is similar to other applications and can be done in several ways that are well-known to regular users:

- confirm the **C**lose menu option on the **F**ile tab,
- press the **X** button in the main window header, or
- use the ALT+F4 keyboard shortcut.

2.2 New database

To create a new empty computer database (Figure 2.4), it is necessary to left-click on the "Blank Database" option in the MS Access initial window menu (Figure 2.2), and in the subsequently displayed window (Figure 2.3) enter the desired name for the new database file, define its location on the computer's hard drive (in case we want to change the predefined folder where the new database should be saved), and confirm by clicking the **C**reate button.

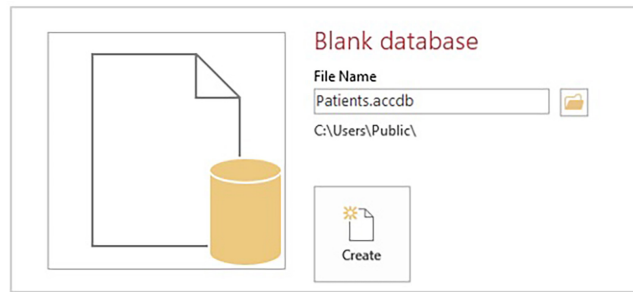


Figure 2.3: Confirmation of the name and location of the new blank database in MS Access.

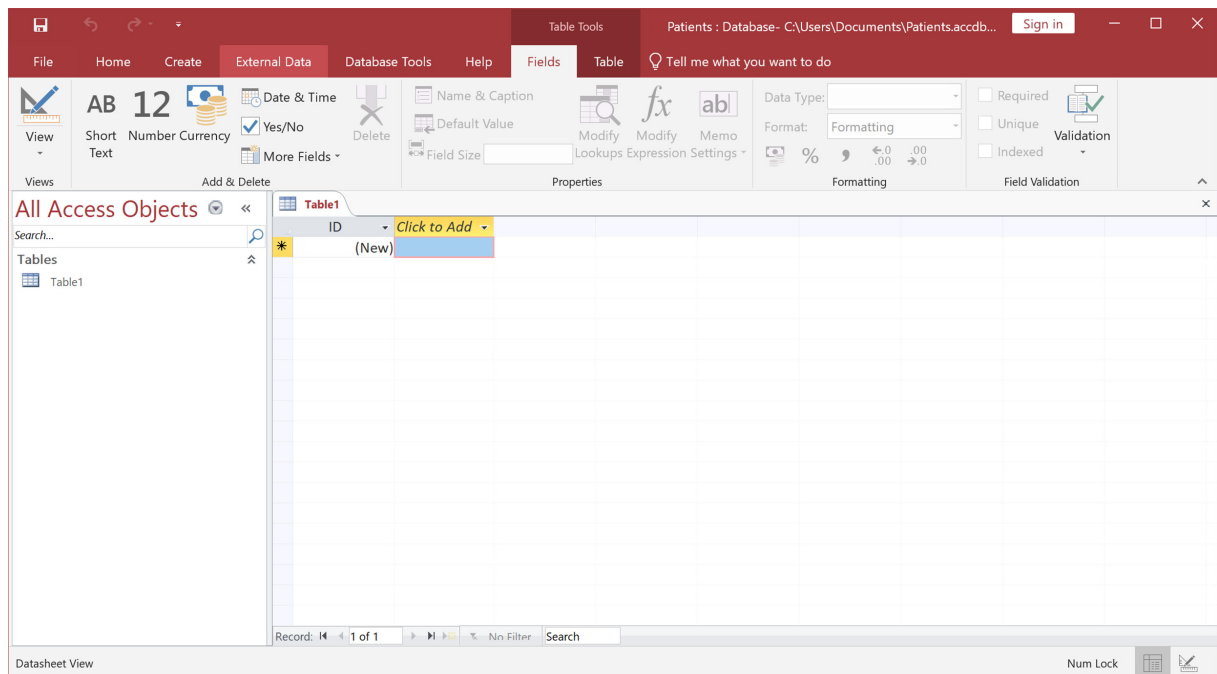


Figure 2.4: New blank database in MS Access.

The database created this way is ready for use and contains only a single database object (see Chapter 2.4), which is an empty table named "Table1." This method of creating a new blank database is suitable for users who want to create a complete database structure based on their own designs, requirements and ideas, i.e. they define all database objects themselves, their number, structure, properties, data connections, work with them, etc.

When creating a new database, the user also has the option to select one of the predefined templates, which already contain tables for data entry, as well as other database objects, including forms, queries or reports that allow, for example, browsing, sorting, filtering or clear printing of the original or processed data as required. In the initial MS Access window (Figure 2.2), it is enough to select the appropriate template and similarly as with an empty database file, enter the file name and choose its storage location (if it is different from that listed under the name of the newly created database file). Then it is enough to confirm the **Create** button, and work with the new database can begin. Database templates are displayed in the initial MS Access window (Figure 2.2), but if the user's computer is online

(i.e. connected to the Internet), many more database templates can be obtained, for example, from Microsoft Support².

If we want to open an existing database, then on the **File** tab (Figure 2.4), we select the **Open** menu and use the standard dialog box to find the database file we want to work with. A list of several recently used (opened) database files can also be found in the **Recent Activities** located on the left down side of the initial MS Access window (Figure 2.2).

MS Access 2019 databases (and other recent versions too) are stored as a single file with the **.accdb** extension, for example, **database1.accdb**. Older databases, i.e. databases created in versions like MS Access 2003 or earlier, have the **.mdb** extension, but current versions of MS Access (including 2019 or 2021) can fully work with them.



Video

Creating the Patients.accdb file

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

2.3 Environment and controls

The environment and controls of MS Access are similar to those of other MS Office suite applications. The main window of MS Access, for example as shown in Figure 2.4, is divided into several parts and encloses the program's workspace.

At the top of the MS Access workspace is a customizable **Quick Access Toolbar** (Figure 2.5). It contains commands that are independent of the activity being performed and do not change while working with the database.



Figure 2.5: Quick Access Toolbar.

The user can place buttons on this toolbar representing commands that are used most frequently. Simply press the **Customize Quick Access Toolbar** button located at the end of the toolbar (Figure 2.6) and select one or more of the available commands. A list of all commands can be found in the **More Commands...** section, where it is also possible to change the order of the individual displayed buttons. Similarly, these commands can be removed from the Quick Access Toolbar.

Unlike most applications, the main menu of MS Access is not static, that is, the **Ribbon** (Figure 2.7) and its options change depending on which object the user is working with and what activity they are currently performing. The Ribbon represents the so-called command interface and contains tabs that group related commands together. The main command tabs are **File**, **Home**, **Create**, **External Data** and **Database Tools**. Additional tabs are automatically added to these tabs depending on the currently used database object. For

²Microsoft Support, <https://support.microsoft.com>

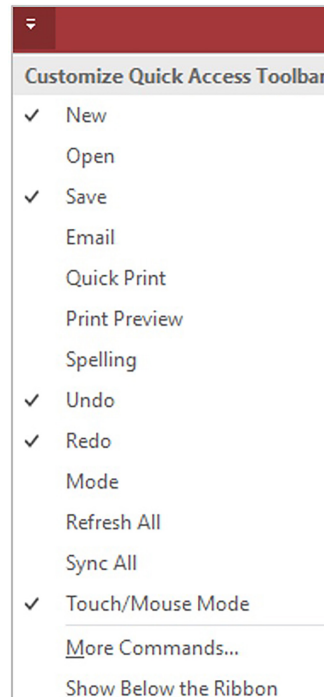


Figure 2.6: Add or remove commands from the Quick Access Toolbar.

example, in Figure 2.7, when working with a database table, these are the Table Tools tabs **Fields** and **Table**. The meaning of specific commands on the Ribbon will be explained with examples in following chapters of this publication.

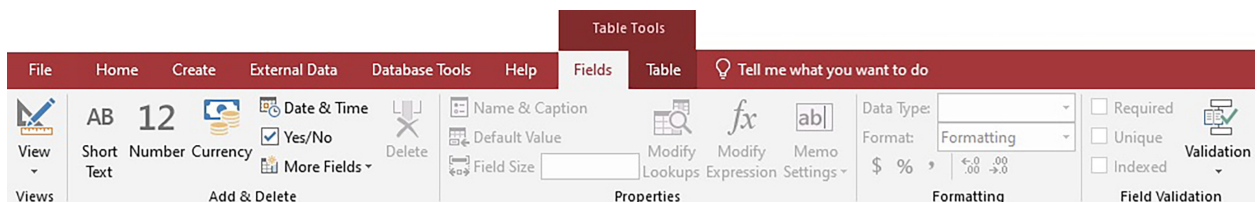


Figure 2.7: Ribbon.

Another important element of the user interface is the **Navigation pane** of database objects (Figure 2.8), which is located on the left side of the MS Access workspace.



Figure 2.8: Navigation pane.

The navigation pane displays database objects (see a brief description in Chapter 2.4) and allows working with these objects. All created database objects are divided into categories that can be shown or hidden. To achieve more space for working with database objects, it is possible to hide (and show again) the entire navigation pane, either by pressing the button « located in the upper right corner of the navigation pane (Figure 2.8) or by pressing the F11 key on the keyboard.

Database objects are usually displayed in the form of cards in the workspace of the MS Access environment. For example, as seen in Figure 2.4, where a new empty table called "Table1" is open, or in Figure 2.9, where multiple different database objects are open.

ID	Date of Exar	PIN	Symptoms	Diagnosis	Allergy	Prescription	Date of che	Click to Add
1	10/7/2024	111111/1111	headache	Migraine		Valetol	10/10/2024	
2	10/14/2024	222222/2222	sore throat	Angina	peanuts	ATB	10/17/2024	
3	10/14/2024	555555/5555	pain in hand	broken wrist		Tramal	10/31/2024	
4	10/14/2024	444444/4444	pain in eyes	inflammatory	cold	eye drops	10/20/2024	
5	11/11/2024	111111/1111	fever	Angina	dust	ATB	11/15/2024	
6	11/11/2024	666666/6666	headache	Hypertension		Concor	11/29/2024	
*(New)	1/13/2025							

Figure 2.9: Tab strip of open database objects.

The working environment of MS Access, like many other applications, also displays a Status bar (Figure 2.10).

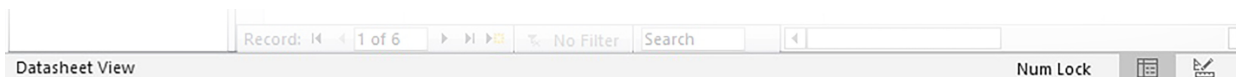


Figure 2.10: Status bar.

The status bar is by default located at the bottom of the MS Access application window and displays status messages, work tips, activity indicators, and other information. Similarly to other applications, it also contains several features, such as the options to switch between views of the currently used object.

2.4 Database objects

Databases in MS Access are made up of objects, including tables, forms, queries, reports, macros, and modules. Each object serves its specific function, and in a particular database, depending on its purpose and the scope of processed information, multiple objects of the same type can be created.

Table

A table is a fundamental object of a relational database, without which it is not possible to create any MS Access database. It is used to store data, and therefore, when a new database is created, the table is always the first object to be created (see Chapter 2.2). Typically,

several tables are created in one database to neatly store interrelated data. These tables are later linked as needed (see Chapter 3). Each table consists of rows and columns. One row of the table represents one database record, such as a record about a patient, a record about medications, a record about examinations, etc. The columns of the table represent fields, i.e. data of the record. In a patient's record, this could be, for example, first name, surname, degree/title, date of birth, etc. Data stored in tables can be of various types (text, number, date, etc.), but the data of one field (data in one column) are always of the same type.

Form

A form is a database object that displays data from one or more tables, or queries, in a clear format (see Chapter 4). Generally, a form is an extension of a table that provides a different view of the data stored in the database. In addition to viewing data, it also allows editing and entering new records. Using forms facilitates working with data stored in tables and also reduces the risk of errors occurring in data entry, which can be cumbersome and inconvenient in large databases with many fields and records in the tables themselves. Entering data into form fields is clear, and the user records data into precisely defined fields, usually in such a way that, unlike table cells, they can see the entire edited content.

Query

A query is a database object that we use to select data from one or more tables, or to perform an action with the data (see Chapter 6). It works similarly to a filter, where based on predefined criteria, we select only the information we need from the database. A query can also be used to join two or more tables, as well as to control, add, change or delete data from the database. For example, we use it to select patients with a certain blood type, create a list of prescribed medications, list examinations of a specific patient, etc.

Report

A report allows data stored in database tables or query results to be prepared in a form suitable for presentation in electronic or printed form (see Chapter 7). This database object offers users the option to select database content and settings for clear printing.

Macro

Macro is a tool that can be used for the automation of repetitive tasks. A macro can also be used to add various functions to forms, reports, or their controls. It represents a list of codes (events) that are executed when it is run, where the user does not have to write programming language codes but selects from a list of commands.

Module

A module is an object created using programming code, i.e. in the case of MS Access, it is a program written in the object-oriented programming language Visual Basic, which works with database data. Unlike a macro composed of a sequence of usually simple steps performed when a specific event is triggered, a module is more complex, containing "programmatically" more demanding calculations.

**Review questions**

1. In what ways can we create a database?
2. Describe the MS Access environment.
3. What is the purpose of the Ribbon?
4. What is the purpose of the Navigation pane?
5. What is the purpose of the Status bar?
6. What are database objects and what are they used for?
7. Which is the main database object and why?

Chapter 3

Tables

In this chapter, most readers will begin creating the structure and content of their first database, which will also serve as an example through which the individual functions and capabilities of the MS Access software tool will be gradually explained. The sample database will be intended for simple record-keeping of patients and their examinations. It is important to note right at the beginning that the database will be created so that every reader can handle its creation. For this reason, the scope of recorded information will be minimized and the specification of some parameters simplified. However, more experienced readers can create and expand individual objects according to their own ideas.

Before we start creating the first database objects (in this case tables), it is necessary to create a new database. Since this publication aims to explain the basic principles of relational databases, we will prioritize creating an empty database over creating one from a template, which we would ultimately have to modify and add elements needed for our purposes.

We can create a new database, for example, as explained in Chapter 2.1 and Chapter 2.2. For our purposes, we will use the name "Patients", i.e. the database will be stored on the computer as a file with this name, for example in the folder **C:\Databases\Patient.accdb**. In the thus-created empty database, we can start working, i.e. creating all the necessary objects, defining their control elements and properties, according to our ideas and requirements, or according to the analysis of the database users' needs and the design of its structure, which we previously made, for example on paper.

The design of the database structure, especially the structure of its tables, is important for the proper functioning of the entire database system. In the design of the table(s), account should be taken in particular of the scope and type of data processed, since it is often very difficult to change the structure of the tables once the data have been filled in. Mistakes may occur, or even data loss, and sometimes (if a backup is not made) reverting to the originally designed table structure is impossible.

We already know that a database table is the main source of data and that it consists of a set of rows and columns. One row in a table contains an instance of one entity (a set of related data), which is usually handled as an indivisible whole. The data in one row of

a table is referred to as a **record**. Each such record consists of data arranged side by side, i.e. in the table columns. A column in which the data is of the same type is called a **field** of the database table.

In each relational database, there are usually multiple database tables with related and unrelated data. Database tables with related data are interconnected through relations (mutual **relationships**). Simple searching, sorting and calculations from the data closely depend on how the data is stored in the database table. It also affects the size of the database file, the processing speed of calculations, intermediate results, etc. The more detailed the preliminary analysis of the design and description of the parameters of each database table is done, the easier it is to navigate the database table, not only when entering new records, but also when applying various sorting mechanisms or executing calculation tasks. When designing database tables, it is recommended, among other things, to follow basic principles:

- fields in a table should only contain related information, such as only contact details, only vaccination information, only laboratory test information, etc.,
- design fields so that they do not accumulate multiple data, for example, address should be divided into several separate fields (street, building number, zip code, city, etc.),
- do not create fields with repeating content, or do not create the same fields in multiple tables, except for fields needed to create relationships between tables,
- each table should have a field that uniquely identifies the record, i.e. it cannot contain duplicate values.

3.1 Creating and viewing a table

In MS Access, a table can be created in several ways, just like other database objects. As mentioned in previous chapters, a table represents a fundamental database object. Therefore, the first table is automatically created and displayed when a new empty database is created (Figure 2.4). This table is empty and has only a single field named ID. Even though such a table is ready for data entry, i.e. it is displayed in datasheet view, it is advisable to first define its fields, their properties, and any links to other tables that are part of the proposed database.

The datasheet is not the only view of a database table in MS Access. Tables can be viewed as:

- *Datasheet* (data view) – allows searching, editing, adding or deleting data of the table, with table rows corresponding to database records and columns to individual fields of the record,
- *Design* (design view) – allows creating and altering the structure of a table by defining the names, data types, descriptions and properties of its fields (columns), without displaying any data stored in the table, meaning the details of stored records cannot be seen here.

These table views can be changed using the **View** menu found, for example, on the **Home** tab of the ribbon (Figure 3.1).

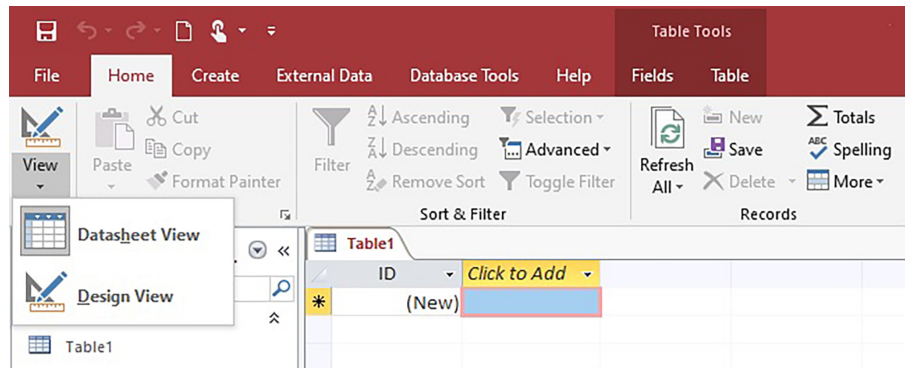




Figure 3.1: Menu for changing the database table view.

The database table can be opened in the desired view via the local menu called up above the table in the navigation pane objects list (Figure 2.8). It is also possible to switch the view of an already open table in the local menu of the table tab on the tab strip of open database objects (Figure 2.9), or using the buttons   located on the right side of the status bar (Figure 2.10).

3.2 Database structure

The structure of the resulting sample database designed for basic work with a outpatient clinic will be divided into several tables. In the first step of implementation, to illustrate and simplify the explanation of the issue, three basic tables will be specified, which we will name: "Personal Data", "Examinations" and "Biometric Data". Subsequently, it will be possible to add more tables to the database to ensure that its final structure serves the intended purpose and that the data in the individual tables represent logical and interrelated units.

A patient's health record, or a record of their examination, should generally include both the identification of the patient and information about current health issues and their solutions during a doctor's visit, information needed for emergency medicine, a list of vaccinations, laboratory examinations and their results, and many other categories of data related to the provision of healthcare. In the first table "Personal Data", we define fields that will identify individual patients, ensuring that these are not duplicated in other tables. The table can therefore, contain fields such as: First Name, Surname, Academic Degree, Personal ID Number, Date of Birth, Gender, Street, City, Zip Code, Country, Phone, etc.

The list of fields can, of course, be much more extensive, and it is possible to extend it with fields such as Maiden Name, ID Card Number, Passport Number, Nationality, Marital Status, Employment, Insurance Number, Health Insurance Company, Insurance Country, etc. The patient data in this table will be recorded and stored only once (in one place). In case of any change, such as an address, it is sufficient to

change (update) only one record of the table, and this change will automatically be reflected in all related outputs of the database.

**Note**

If we need to store multiple addresses for a patient, we would create a separate table of addresses to which we would add fields such as **Address Type** (permanent residence, temporary residence, relatives, general practitioner, etc.), and information about which of these addresses is contact or current.

In the "Examinations" table, we will record information about doctor visits, with each patient able to have any number of visits (examinations). The table may contain, for example, the following fields: **Date of Examination**, **Main Diagnosis**, **Secondary Diagnoses**, **Finding** (for description of health problem, treatment procedures and methods used, further treatment, etc.), **Medications**, **Recommendations**, **Date of Follow-up Examination** (control visit), **Extra Payment**, **Note**, etc. Here, too, individual components of the medical record can be divided into multiple fields, and any other related items belonging to this list characterizing the examination record can be defined.

To "identify" the patient to whom a record in the examination table belongs (since names or addresses of patients are not in the examination table), we add a **Patient ID** field to the table, a field that is unique to the patient (such as a personal identification number or insurance number). Additionally, we will include an **Examination ID** field in the list of fields, which will serve as an identifier for completed examinations (e.g., examination order). Thus, the database system will know (after defining appropriate links between created tables, as it is explained in the following chapters) that, for example, examinations numbered 1, 36, and 124 belong to patient Peter Brown, examinations numbered 5, 44, 78, and 214 belong to patient Mario Johnson Smith, etc.

**Note**


It is advisable to divide the examination record into multiple fields, rather than recording everything as a block of text, as it allows for advanced data handling based on individual fields. For example, you can search for patients with a specific diagnosis, track number of visits over a certain period, map the list of prescribed medications, etc.

In the third table of the database, we will record the biometric characteristics of patients. The potential fields of the "Biometric Data" table could contain or calculate fields such as **Height**, **Weight**, **Waist Circumference**, **Hip Circumference**, **BMI** or **WHR** indexes, **Body Surface Area**, etc. Unlike the list of examinations, where a patient may undergo several times, we assume that one patient will have only one record in this table, since they cannot have different simultaneous weights, heights, etc.

**Note**

If archiving biometric data values is required, for example, for tracking development over time, a field characterizing the date of data acquisition can be added, and the table can be defined to allow one patient to have multiple records.

3.3 Personal Data table

We will create the table "Personal Data" in our `Patients.accdb` database from an already existing empty table "Table1" (Figure 2.4), which we obtained automatically when creating a new empty database. For the first use of the "Table1" table, just press the **Save** button , enter the desired table name in the "Save As" dialog box, and confirm by clicking **OK** (Figure 3.2).

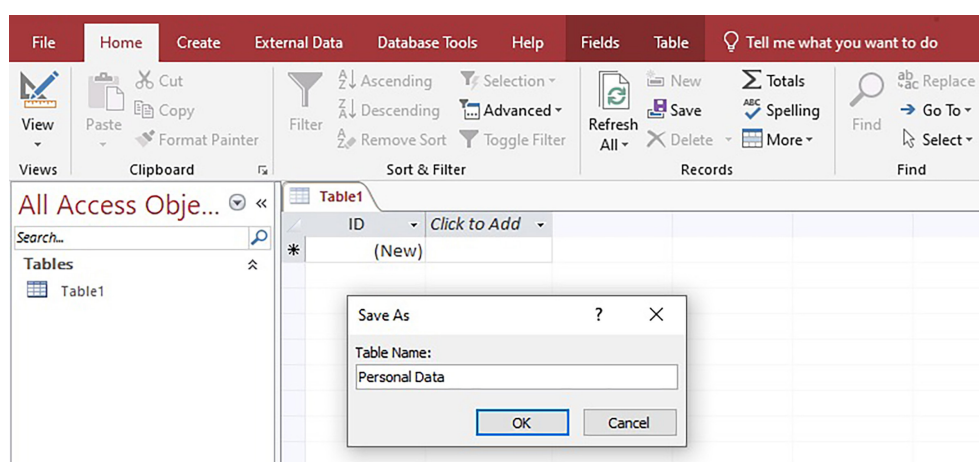


Figure 3.2: Dialog box for saving the table.

We will specify the fields and their properties in the design view of the table, which we can access, for example, by confirming the **Design View** option in the **View** menu on the **Home** tab (Figure 3.1). If we have not worked with the "Table1" table and have not saved it in the manner mentioned above, we will be prompted to save it or change the table name as shown in Figure 3.2. We thus save our first table with the name "Personal Data" and we will know that it contains a list of patients with their identification data.



Note

It is advisable to choose the names of individual database objects so that they characterize their meaning or content. For example, in a database with six tables, it is not suitable to leave the names "Table1", "Table2", ... "Table6", because over time we will lose track of what is stored in them, and we will certainly need to add other objects to the database over time, which can complicate orientation in sometimes already quite complex database structures. Suitable names are, for example, "TableBooks" or "Books" for book records, "TableOrders" or "Orders" for orders, etc.

3.3.1 Design view of the Personal Data table

The database table "Personal Data" is displayed in the design view according to the above procedure (Figure 3.3). The design view does not allow us to see the data stored in the database table, but it allows us to clearly define field names, data types in these fields, their description, and properties.

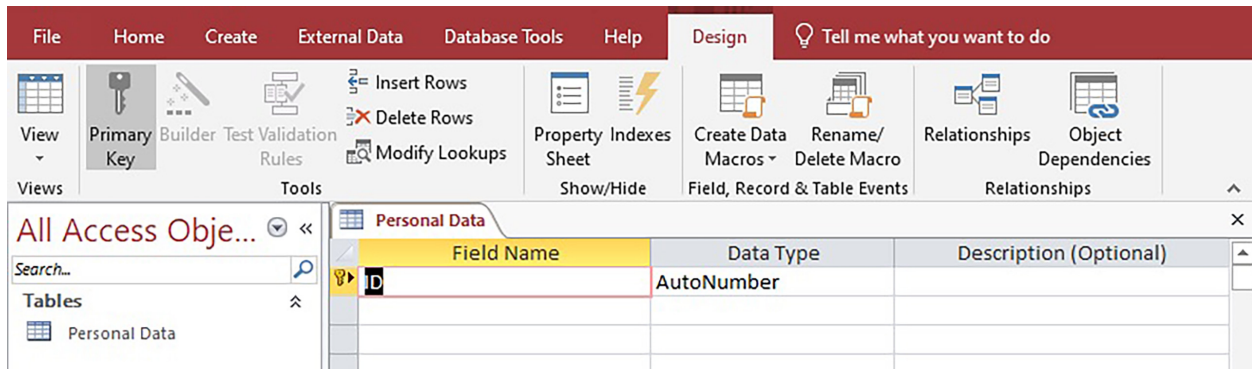


Figure 3.3: Design view of the Personal Data table.

Based on previous considerations about the database structure (Chapter 3.2), we gradually enter the names of all the fields required for recording patients into the "Field Name" column. We can keep the name of the first field as ID, or we can change it to ID record, followed by Academic Degree, Name, Surname, etc. An example of a list of fields in the "Personal Data" table is shown in Figure 3.4.


Personal Data	
ID	
Name	
Surname	
 Personal Identification Number	
Gender	
Date of Birth	
Address	
City	
ZIP code	
Health insurance company	
Telephone number	

Figure 3.4: List of selected database fields in the Personal Data table.

**Note**

Field names can be any string of characters, but we recommend choosing shorter names, without spaces or characters with diacritics (for readability in this textbook we do not strictly adhere). Such names would be functional, but problems could arise when used in processing by other objects, functions, or calculations. For example, for personal identification number, we can use the field name PIN, for blood type group, for example, Blood_Gr, etc. If we want to separate characters or words in field names, we recommend using an underscore mark instead of a space.

For the new database fields created in this way, in addition to their names, we also gradually define the data type, provide a brief description (suitable, for example, as a hint for the meaning or purpose of using the data in a given field), and also specify individual properties.

3.3.2 Data types of database fields

Database records will contain various data, such as text, numbers, dates, etc. To ensure that each field records only specific types of data, it is necessary to define what type of data these fields should accept. In the table's design view, it is possible to define the following data types by selecting from a list (Figure 3.5).

- **Short text** – typically the default data type value and can contain text information, or combinations of text and numbers, as well as numbers not used in calculations (such as phone numbers). Up to 255 characters can be entered into a field of this type.
- **Long text** – a field allowing for the recording of more extensive texts or combinations of text and numbers. The field range is limited to 63,999 characters.
- **Number** – numerical data used, for example, in mathematical calculations.
- **Date and time** – a field designated for entering dates and time in various formats. It is possible to record years from 100 to 9999.
- **Currency** – currency values for entering financial amounts and numerical data suitable for mathematical calculations.
- **AutoNumber** – a unique number automatically assigned by MS Access to each new table record. It can be defined as a sequence (a number that is incremented by 1) or randomly generated.
- **Yes/No** – a field suitable for specifying only one of two possible values. For example, Yes/No, True/False, On/Off, etc.
- **OLE Object** – an object inserted into the database table or linked with the table (*Object Linking and Embedding*). This can be an MS Excel workbook, an MS Word document, or an image, sound, etc.
- **Hyperlink** – text or a combination of text and numbers stored as text and used as a hyperlink address, such as a URL (*Uniform Resource Locator*) address of a website.
- **Attachment** – various types of files (images, documents, charts, etc.). Offers better flexibility than an OLE Object and more efficient disk space usage.
- **Calculated** – a field whose value is the result of a calculation of numerical data stored in the database.
- **Lookup Wizard...** – creates a list of values from which data can be selected into the database table. For example, for the **Gender** field, it can be a list with options "male", "female" and "unspecified".

The data type of fields is defined in our "Personal Data" table in the second column of the design view (Figure 3.3) by selecting from the list of offered options. We will leave the ID field as the "AutoNumber" type. Then, we set the **First name** as "Short text", **Surname** as "Short text", **PIN** as "Number" (later we will change it to "Short text" and use the field properties to set only the format of specific ID number formats), **Gender** as "Short text" (later we will

use the "Lookup Wizard..." to define a list of possible values), **Date of Birth** as "Date and time", **Address** as "Short text", **City** as "Short text", **Zip code** as "Number" (the issue of a leading zero, which does not display by default, can be resolved by using "Short text" data type), **Health insurance company** as "Short text", and **Telephone number** as "Short text". An example of data type definitions in the particular fields is shown in Figure 3.5.

Field Name	Data Type	Description (Optional)
ID	AutoNumber	
Name	Short Text	Write name of the patient
Surname	Short Text	Write surname of the patient
Personal Identification Number	Short Text	Write PIN of the patient in form XXXXXX/XXX(X)
Gender	Short Text	Write gender of the patient (male/female)
Date of Birth	Date/Time	Write patient's date of birth
Address	Short Text	Write permanent address of the patient (street, number)
City	Short Text	Write city
ZIP code	Short Text	Write ZIP code of the city
Health insurance company	Short Text	Write name of health care insurance company
Telephone number	Short Text	Write telephone number in form prefix and number (+421 XXX XXX XXX)

Field Properties	
General	Lookup
Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Text Align	General

The data type determines the kind of values that users can store in the field. Press F1 for help on data types.

Figure 3.5: Design of Personal Data table fields.

Descriptions of database table fields are not required, but it is advisable to add them in the third column of the design view. Later, for example, when entering data into the table, this description will be displayed in the status bar, which can facilitate database users' work with data and orientation in the table.

The datasheet of the "Personal Data" table, where we can start recording the first patient entries, is displayed using the **View** menu (Figure 3.1) located, for example, on the **Home** tab. The database system also prompts us to save the changes made when switching the view. The datasheet of the "Personal Data" table (Figure 3.6) will already contain all the fields we have defined.

ID	Name	Surname	Personal ID	Gender	Date of Birth	Address	City	ZIP code	Health insurance	Telephone number
(New)										

Figure 3.6: Datasheet view of the table Personal Data with newly defined field names.

3.3.3 General properties of the fields of the Personal Data table

Records of patient identification data contain various information, and for their correct entry and storage in the database table, it is often necessary to set some properties of the given field. According to the selected data type of the respective field, it is possible to define their properties in the table design view (usually it is sufficient to set a few selected properties, according to the requirements for the data processed in the respective field), which are displayed on the **General** tab (for example, Figure 3.5 below for data type "Short text" or Figure 3.7 for data type "Number").

General Lookup	
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	No
Indexed	No
Text Align	General

Figure 3.7: General properties of database fields of the "Number" type.

Before we start with patient records, we recommend defining the complete structure of the database, i.e. all tables, their fields, but also the properties of these fields that we have not yet specified, and the relationships between table records. For simplicity, we will limit the initial structure design of the database to three tables.

We define the general properties of a specific field (the field we are working with) at the lower part of the table design view. In the "Personal Data" table for the ID field, we leave the general properties set to "Field Size: Long Integer", "New Values: Increment", and "Indexed: yes (no duplicates)". This way, a new record will always be assigned the next sequential number, even if one of the records was deleted and the previous position is free.

We can add an **Academic Degree** field to the table, which is "Short text", and set the property "Field Size: 10". It means the user can enter a maximum of 10 characters in this field. This value can be any number that corresponds to the maximum expected length of the string of characters defining the academic degree of the patient. Additional general properties for the academic degree do not need to be set as this field is usually not mandatory.



Note

For the Academic Degree field a list of all degrees can be created from which the database user will select the one belonging to particular patient. This list can be created identically to the option list for the Gender field. Alternatively, multiple separate fields can be created for degrees, such as one for degrees given before the name and one for degrees given after the patient's surname.

The patient's **First Name** and **Surname** should be recorded in the patient record (we should not have a registered patient without a name) and should be entered correctly, i.e. it should be entered with a capital initial letter. Such data format can be achieved in several ways, one of which is to use the "Input Mask" property, whose value we set, for example, in the form ">L<?????????????". The input mask properties can be defined using reserved characters, the list of which is given in Table 3.1. The above mentioned input mask for the name will ensure that when typing the name, the first letter is mandatory, will be uppercase, and followed by several lowercase letters (it will not be possible to enter numbers even though the data type "Short text" allows it), the number of which is limited by the number of question marks in the input mask (one question mark represents one letter).

Table 3.1: List of characters used to define the input mask.

Character	Description
0	Required digit (0 to 9), must be entered.
9	Optional digit (0 to 9), does not need to be entered.
#	Allows entering a digit, space, or plus and minus symbols. If a position with this character is skipped, a space will be entered.
L	Required letter (a to z), must be entered.
?	Optional letter (a to z), does not need to be entered.
A	Required letter (a to z) or digit (0 to 9).
a	Optional letter (a to z) or digit (0 to 9).
&	Required character or space.
C	Optional character or space.
. , : ; - /	Placeholder thousand and decimal place characters, date and time separators.
>	All following characters are converted to uppercase.
<	All following characters are converted to lowercase.
!	Forces the input mask to fill characters from left to right instead of right to left.
\	Character immediately following will be displayed as entered.
""	Characters enclosed in quotation marks will be displayed as entered.




Note

A complete input mask consists of three parts separated by semicolons. For example, >L<?????????????;;. The first part is mandatory and consists of characters (it may also contain embedded characters such as parentheses, dots, and hyphens) defining the required data format. The second and third parts of the input mask are optional. The second part contains information on whether the input characters are stored in the database with the data (value 0) or not stored, just displayed (value 1). The third part specifies a character or space (the default is an underscore) that serves as a placeholder symbol displayed when editing the field's value.

Input masks are very useful, but since they do not allow any exceptions, they are not suitable for all situations. Sometimes it may be necessary to enter data in the respective field that does not match the defined input mask. In such a case, MS Access will not allow saving of this data.

In the "Personal Data" database table, there can be a case where we have two or even more patients with the same first and last names. Therefore, a unique identifier needs to be used to differentiate them, which can be, for example, a unique registration number of a personal document, insurance card, personal identification number, etc. The most commonly used identifier currently is the patient's ID number, which is why, if we use it, we will know which patient it is. The personal identification number (PIN) contains both numerical data about the day, month, and year of birth and information about gender and also an extension – assigned digits that distinguish people born on the same calendar day and are usually separated by a slash.

Since we want to use the form of the personal identification number that includes the special character slash, we chose the "Short text" data type for this field, while only numbers will be recorded here. The length of the PIN must be equal to ten digits, or nine digits for those patients born before 1953 with an extension composed of three digits only. The last digit, the tenth, will therefore not be mandatory. We will not verify other properties of the personal identification numbers at this stage of the sample database design for clarity. Therefore, we will enter the personal identification number using an input mask, whose value we set to 000000\ /0009;0;*. Besides directly entering the expression to define the input mask, it is also possible to use the **Input Mask Wizard** (Figure 3.8), which we open by pressing the icon  located at the end of the line of this property on the **General** tab of design view of the table (e.g. it is shown in Figure 3.7).

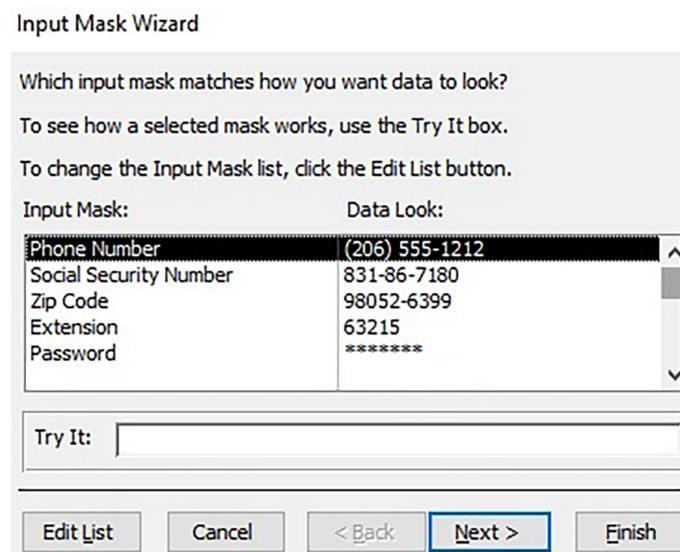

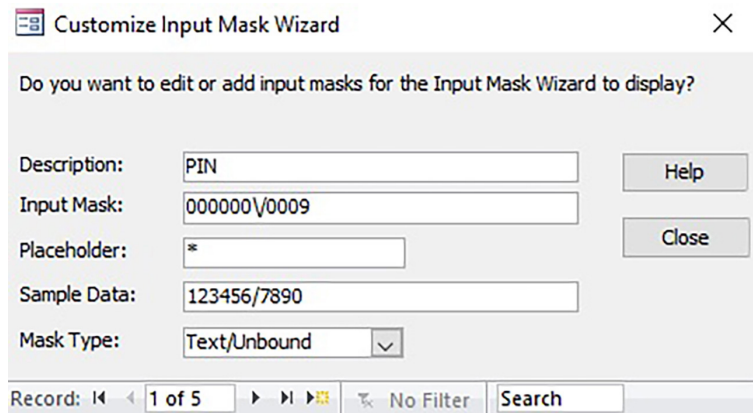


Figure 3.8: Input Mask Wizard.

The Input Mask Wizard allows you to browse and modify existing input masks, as well as create new ones. To create a new or modify an existing input mask, confirm the **Edit list** button, which opens the **Customize Input Mask Wizard** dialog box (Figure 3.9). We create a new input mask by moving to a new record using the button  located in the row for switching existing records.



Customize Input Mask Wizard

Do you want to edit or add input masks for the Input Mask Wizard to display?

Description: PIN

Input Mask: 000000\0009

Placeholder: *

Sample Data: 123456/7890

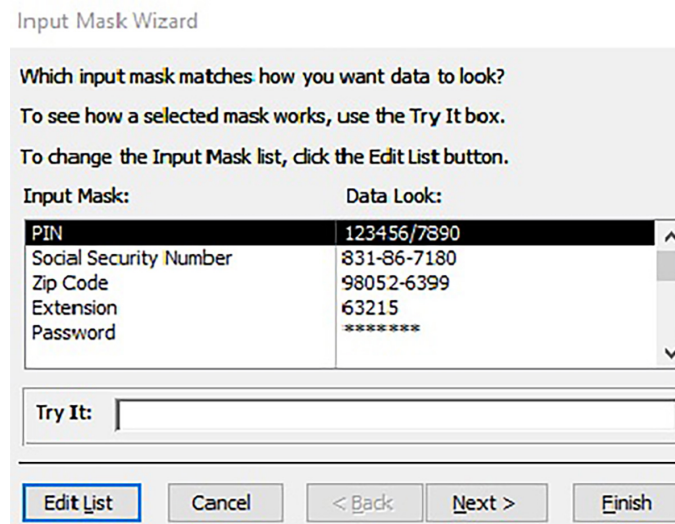
Mask Type: Text/Unbound

Buttons: Help, Close

Record: 1 of 5, No Filter, Search

Figure 3.9: New input mask for the PIN field.

In the individual fields of the input mask, gradually enter Description (any name), Input mask in the form 000000\0009, Placeholder (optional, default is an underscore), and Sample Data (any data sample that matches the input mask settings). We leave the mask type with the text/unbound value and confirm the **Close** button, which returns us to the Input Mask Wizard window, where we already have the newly created input mask for the personal identification number (PIN) (Figure 3.10).



Input Mask Wizard

Which input mask matches how you want data to look?

To see how a selected mask works, use the Try It box.

To change the Input Mask list, click the Edit List button.

Input Mask:	Data Look:
PIN	123456/7890
Social Security Number	831-86-7180
Zip Code	98052-6399
Extension	63215
Password	*****

Try It:

Buttons: Edit List, Cancel, < Back, Next >, Finish

Figure 3.10: Selecting the input mask for the PIN field.

The input mask we want to use must first be selected with the left mouse button or arrow keys on the keyboard (the mask selected in the black frame), and in the next steps of the Input Mask Wizard, which we access through the **Next** button, we gradually check or modify the entered input mask, try entering data into the selected mask, define whether we want data to be saved with the used symbols or without the used symbols (for the PIN, this is the slash) and complete the input mask settings by confirming the **Finish** button. The information about the defined input mask for the personal identification number will be listed in the properties list of this field, as shown in Figure 3.11.

General Lookup	
Field Size	255
Format	
Input Mask	000000\0009;0;*
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Text Align	General

Figure 3.11: General properties of the PIN field.

**Video****Input mask**

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

The field **Gender** is supposed to contain only information corresponding to whether the patient is male or female. This field may remain unfilled in cases where the patient's gender information is not available or not obvious from other identification details of the patient. To prevent other information, such as names, addresses, etc., from being accidentally stored in the gender field, it is advisable to create a list of acceptable values from which the user can select the appropriate option when working with a record in the table.

The easiest way to create a list of options is by using a wizard. In the design view of the "Personal Data" table, select the **Gender** field and change the data type from "Short Text" to "Lookup Wizard...", which also opens its dialog box (Figure 3.12).

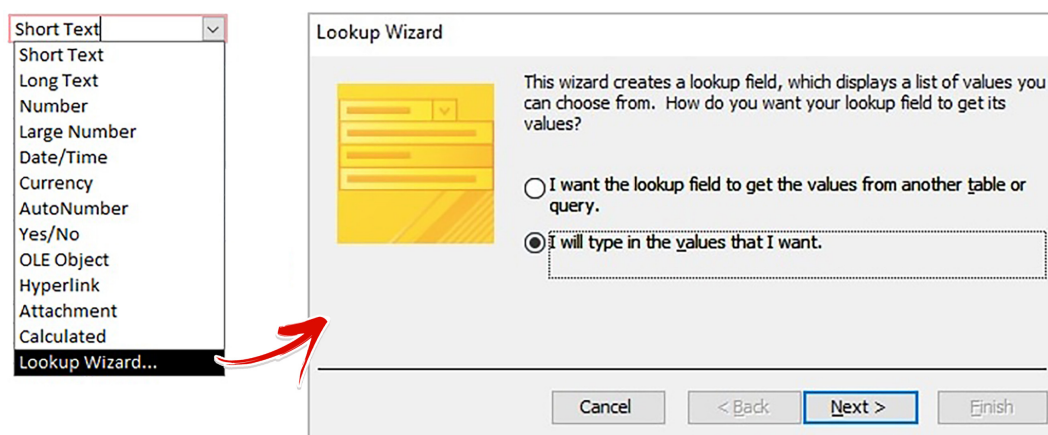


Figure 3.12: Initial window of the Lookup Wizard.

The values or items that will make up the list of the given field can be entered by loading from existing sources (tables or queries, including external files containing the values for the list) or entered manually. To create a list of values for the **Gender** field, choose the "I will type in the values that I want" option and in the next step, after pressing the **Next** button,

enter in the first column (only one column is needed for gender) the names valid for the gender, as shown in Figure 3.13.

Figure 3.13: Defining values for the lookup field.

If you want the **Gender** field to contain only the values defined by the newly created list, then in the next step of the Lookup Wizard, restrict the list of lookup field values to this list (check the "Limit To List" checkbox, Figure 3.14).

Figure 3.14: Limiting the list of values for a lookup field.


After saving the changes made in the design view of the "Personal Data" table and switching to its datasheet view, a list of values for selecting the gender name will be available in the **Gender** field (Figure 3.15).

Figure 3.15: Gender lookup field.

**Note**

When working with table records and using the keyboard, simply enter the first letter of the value in the lookup field, and the list will highlight the first item that begins with the given letter.

The patient's **Date of Birth** will be recorded in a numerical format with the sequence representing day, month, and year separated by a period (slash if preferred) and a space. We recommend setting its general property "Format" to "Short Date".

To verify the correctness of the date of birth, for example, to avoid entering the date of a patient who is not yet born, it is useful to use another property, which is the "Validation Rule". We will require that the date of birth is before (existing patient) or matches the current date (newborn). The validation rule setup is done using the "Expression Builder" (Figure 3.16), which is activated by pressing the icon  located at the end of the "Validation Rule" property line.

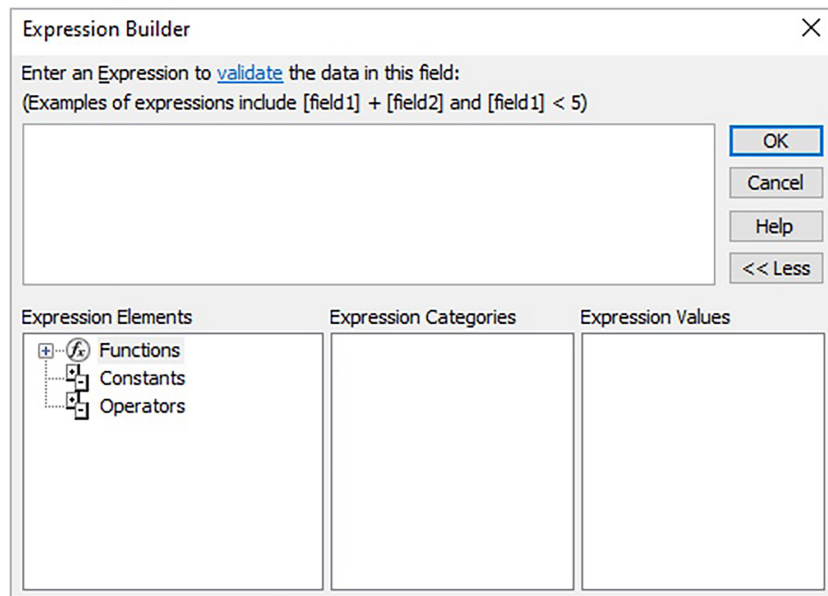


Figure 3.16: Expression Builder for validation rule of database table field.

The top of the "Expression Builder" window is the space for entering the validation rule. Its bottom is composed of three areas where functions, constants, and operators can be selected for the validation rule (expression). These are further divided into categories and values for clarity.

From the operators, select the value `<=`, clicking it twice with the left mouse button to transfer it to the expression. Continue by selecting "Functions: Built-in Functions", choose "Expression Category: Date and Time", and then "Expression Value: Date" (current date retrieved from the operating system, Figure 3.17). Double-clicking on the expression value "Date" transfers the text `Date` into the validation rule, and we confirm its use by clicking the **OK** button. In the design view of the "Personal Data" database table, the value `<=Date()` is set in the "Validation Rule" line.

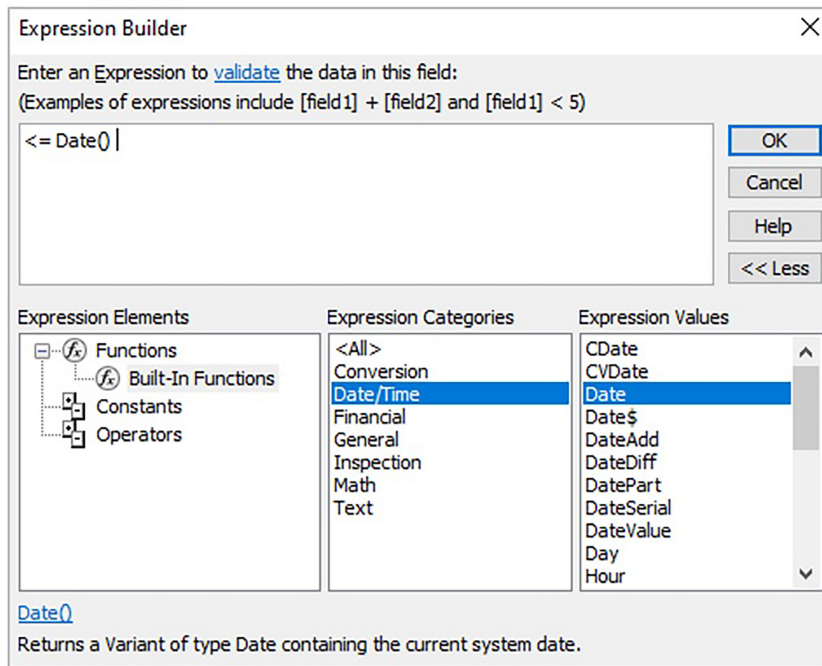


Figure 3.17: Expression Builder for validation rule of Date of Birth.

If we want to inform the database user about an incorrect value (one that does not meet the validation rule), then write an explanation into the "Validation Text" property line. For example, the text: *"The patient has not been born yet. Check and correct the date of birth"*. If a future date of birth is entered during patient registration, then a dialog box with this warning will appear. Until the validated data is evaluated as correct, it will not be possible to continue writing data into other fields of the table or save the record.

The **Address** field will store patient residency information, including street and house number. From the properties of this field, it is possible to use "Field Size" and reduce the value from 255 to, for example, 35 characters, allowing the user to enter only a limited number of characters, including letters and numbers, which are expected to suffice for entering the full street name and house number.

For the **City** field, besides field size, we can also set a "Default Value" property. Usually, we enter the name of the city where the clinic is located, as it is assumed that the majority of patients will be from this city or its nearby area, the designated city being the district. Using the default value can speed up the administrative process when registering new patients since the **City** field will already be filled in in each new record. If some patients are from another city, the default data can be simply overwritten directly in the datasheet view.



Practical Task

Set the Zip Code field (postcode) to accept only numbers, which must be exactly five digits long, with a space between the third and fourth digits. Use an input mask, similar to what was used for the personal identification number (PIN).

**Note**

The fields for a patient's residence are significantly reduced for the purposes of the sample database. For permanent residence address documentation, individual fields for the street, house number, street number, Zip code, city, and state can be created. The names of villages and their zip code can also be loaded from external files or imported into the database, e.g. from official lists of towns and municipalities.

Patients are generally insured by one of the health insurance companies operating in particular country, and foreign patients may be insured by other foreign insurance companies. To avoid constantly typing their names when registering new patients, we can create a list of insurance companies from which the user can select the relevant one.


**Practical Task**

Create a list of names of existing insurance companies for the Health Insurance Company field. The "Lookup Wizard..." data type or the **Lookup** tab located next to the **General** tab in the table design view (such as Figure 3.5, 3.7, or 3.11) might be used for this.

**Note**

For larger databases, it is advisable to create a separate health insurance table in which patients would have records of all health insurance and insurance companies with which they have been insured, including, for example, the insurance number, insurance company code, start date, end date of insurance, and possibly the insurance state code, etc.

Another field for which the "Input Mask" property can be used is the **Telephone number** field. When creating an input mask for the telephone number, it is first necessary to choose its uniform format, i.e. the format in which the telephone contact will be displayed and stored in the database table. Careful consideration of restrictions needs to be taken to ensure that the input mask does not prevent the storage of a real phone number. As an example, we will create a phone number format including the country code, along with the special character + (a separate country prefix field may also be created). For this field, we use the "Short Text" data type and proceed similarly when creating an input mask for the personal identification number.

The "Input mask" for the **Telephone Number** field is set via the "Input Mask Wizard" (activated by pressing the button ) and its "Edit list" option. In the "Customize Input Mask Wizard" dialog box (Figure 3.18), type the input mask after the prefix +421 using the digit 9. This digit does not indicate an operator number but that the position of the digit can be empty (underscore) or filled, for example, in case of a fixed line number like +421 _55 123 123. For the position of further digits, enter zero, meaning that the digits must be filled. In the "Data Preview" field, enter an arbitrary example phone number, and close the "Customize Input Mask Wizard" dialog box by pressing the **Close** button.

Customize Input Mask Wizard

Do you want to edit or add input masks for the Input Mask Wizard to display?

Description: Help

Input Mask: Close

Placeholder:

Sample Data:

Mask Type:

Record: 1 of 5 No Filter Search

Figure 3.18: Creating an input mask for the Telephone Number with Slovakia's international prefix.

In the following steps of the wizard, select the mask for the defined phone number format and gradually set (**Next** button) possibly changing the placeholder character, testing the input mask functionality, and deciding whether you want the phone number to be stored with or without symbols in the mask. Complete using the input mask for the respective field by pressing the **Finish** button. Find the input mask value in the list of general properties defined, for example, as shown in Figure 3.19.

General Lookup	
Field Size	255
Format	
Input Mask	*+421 *900\ 000\ 000;0;*
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Text Align	General

Figure 3.19: General properties of the Telephone Number field.



Note

In case it is necessary to document seven digits after the city prefix, you can extend the input mask by adding the digit 9 to its end. The seventh digit, for example, in mobile phone numbers will thus not be mandatory, but in local telephone numbers (landlines), the database user may enter and store it.

3.4 Examinations table

In the second table of the sample database, we will record patient visits and information related to their current health problems and their management. The "Examinations" table

will be created by selecting the **Table** option on the **Create** tab of the Ribbon (Figure 3.20). The table can also be created by opening its design view, after selecting the **Table Design** option.

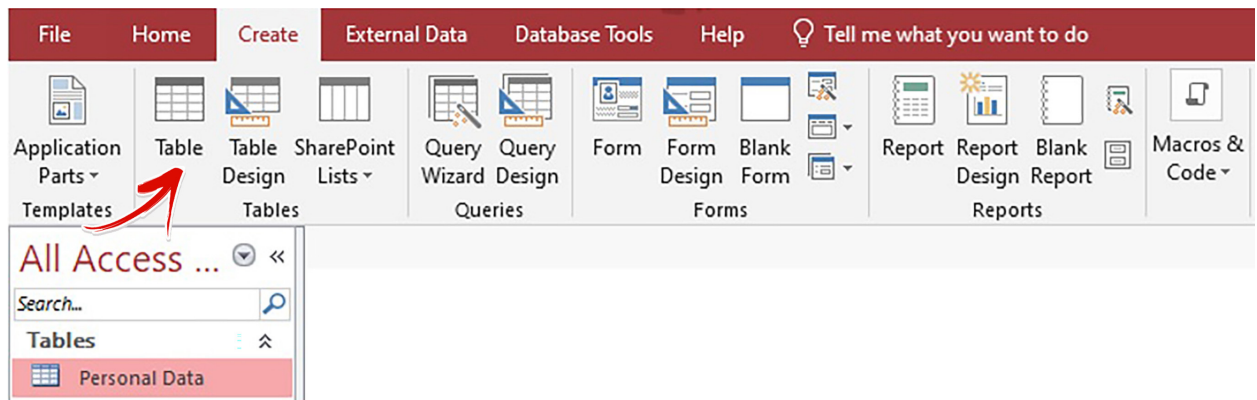


Figure 3.20: The Table menu for creating a new database object from the Create tab.

A new empty table will be generated and opened, again with the default name "Table1" and with one automatically inserted ID field. Using the **View** menu located on the **Home** tab, we switch the table's datasheet view to design view and in the **Save As** dialog box, we change its name to "Examinations" and confirm by pressing **OK** button. Similarly to the "Personal Data" table, we define all the required fields according to the previous database structure design (see Chapter 3.2).

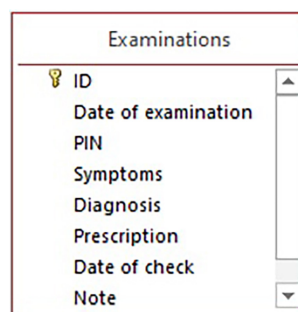


Figure 3.21: List of selected database fields in the Examinations table.

3.4.1 Design view of the Examinations Table

The first field in the list named ID can be renamed if needed, e.g. to **Examination_ID** for better identification, keeping the data type "AutoNumber" for automatic numbering of individual patient examinations. Then add new fields, for example, **Date of Examination** (DoE) and define its data type as "Date/Time", **PIN** as "Short Text", **Symptoms** as "Long Text" for recording more detailed reports (more than 255 characters), **Diagnosis** as "Short Text", **Prescription** as "Short Text", **Date of Check** (DoCh) as "Date/Time", **Note** as "Short Text", etc. (similarly as it is shown in Figure 3.22).

Field Name	Data Type	Description (Optional)
ID Examination	AutoNumber	
Date of Examination	Date/Time	Write date of examination
PIN	Short Text	Write patient's PIN in form XXXXXX/XXX(X)
Symptoms	Long Text	Write patient's symptoms
Diagnosis	Short Text	Write main diagnosis
Prescription	Short Text	Write name of the medicine
Date of check	Date/Time	Write date of new appointment
Note	Short Text	Write additional information

Figure 3.22: Design of database fields in the Examinations table.




Note

The Personal identification number (PIN) field here does not present duplication of recorded data, but was added to the "Examinations" table to later link records of the "Personal Data" and "Examinations" tables. For this reason, it should be defined similarly to how it is in the "Personal Data" table.

3.4.2 General properties of the fields of the Examinations table

We will the selected fields used as the base of the "Examinations" table records so that working with records, as in the "Personal Data" table, is as simple as possible and so that the records contain information in a unified format. This will be necessary for further automated processing, analyzing, or evaluating stored data.

The field identifying individual examination records, examination ID, will contain unique numeric values assigned automatically, so unless we wish to use our own numbering forms for these database records, there is no need to adjust its other properties.

Patient **Date of Examination** will be recorded uniformly throughout the sample database in the format containing day, month, and year. Therefore, define the data type as "Date/Time" and set the "Format" property to "Short Date". Assuming that records from individual patient visits to the doctor are made in real-time, i.e. at the time when the patient is undergoing an examination, set the "Default Value" property so that every new record always contains the current date. To do so, you can use "Expression Builder" by clicking on the icon  located at the end of line of this property, or enter the expression directly into the "Default Value" field. This default value will shorten the recording time of examination records and prevent errors in record-keeping from incorrect date entries or typographical errors that can be done by the user (operator of the database).

The "Expression Builder" for the "Default Value" property should contain an expression inserting the current date into each new record. Using the operators, select the expression value = and double-click it to transfer it into the built expression. Then using "Functions: Built-in Functions", select "Expression Category: Date/Time" and "Expression Value: Date" (Figure 3.23), i.e. a function that loads the current system date into the new record. Confirm the set expression value with **OK** button.

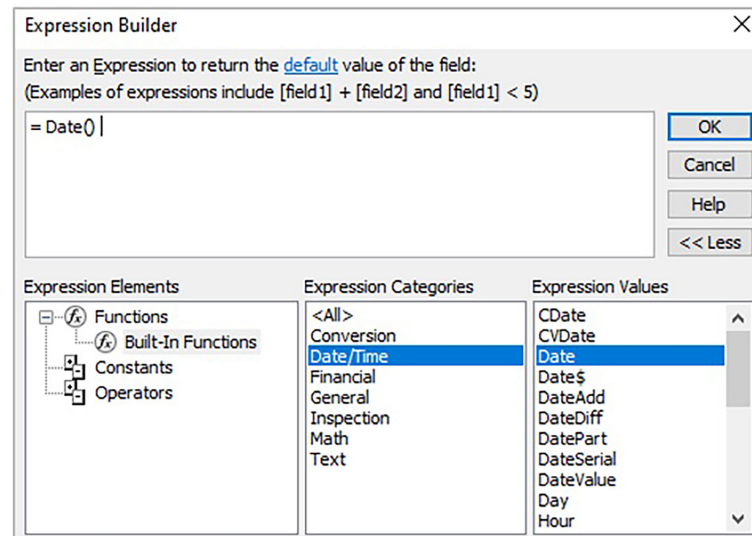


Figure 3.23: Expression Builder and setting the default value for Date of Examination field.

As specified above, to correctly assign examination records to a specific patient, it is also necessary to create a PIN field in the "Examinations" database table. Create and set the general properties of the PIN database field identically, as laid out in the Chapter 3.3.3, or by copying the field from the "Personal Data" database table (Figure 3.24).

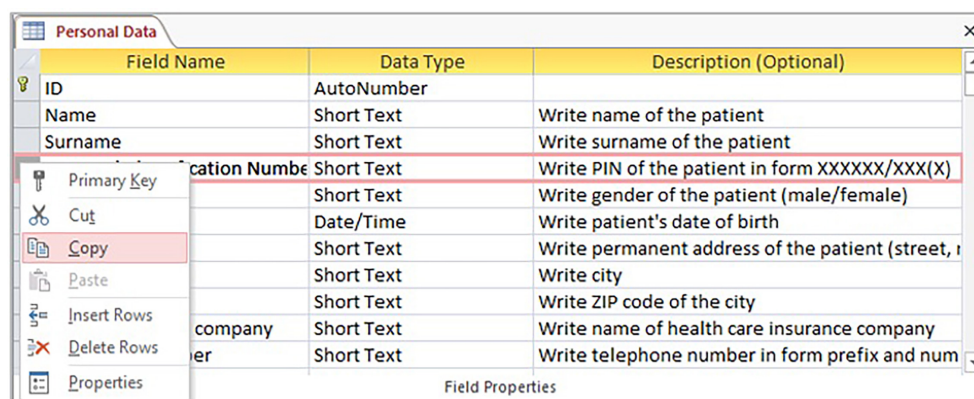


Figure 3.24: Copying PIN field from the "Personal Data" database table.

To copy a database field from one database table to another, first right-clicking the gray square of the PIN field in the design view of the "Personal Data" table and select the **Copy** options. Then, move to the second table, i.e. in this case, the "Examinations" database table, and in its design view, paste the copied PIN field into an empty row by selecting **Paste** option from the right-click local menu (Figure 3.25). By pasting the PIN field into the "Examinations" table, all its properties will be copied as set in the "Personal Data" table.

For the **Symptoms** database field, which is expected to store the most extensive information obtained during a patient's examination, including the description of solving their health problem, no property will be set. The database user can record a description of the finding without restrictions.

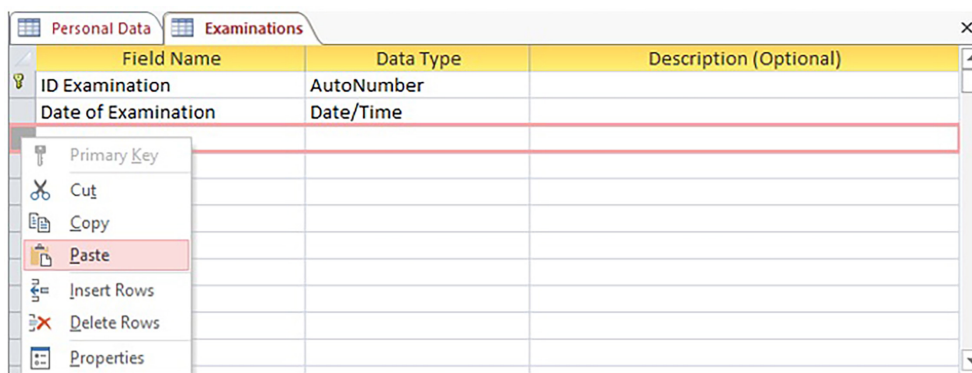


Figure 3.25: Pasting the PIN field into the "Examinations" database table.

Health records use codes for diagnosis marking according to the International Classification of Diseases (ICD-10 and ICD-11, for more information, visit website of WHO³). Each diagnosis is uniquely defined and cannot be confused when exchanging information, for example between healthcare facilities at national and international levels.



Note

In comprehensive examination records, it is possible to create multiple fields for diagnoses, e.g. main and secondary, or traumatic and causal, etc. To avoid manually recording codes of diagnoses and their descriptions (full diagnosis name), it is possible to insert the current version of the International Classification of Diseases into the database from an external file, from which users will subsequently select diagnoses.



Practical Task

Create an input mask for the Diagnosis field that only accepts diagnosis codes in the format of the International Classification of Diseases version 10 (ICD-10), i.e. in the format LNN.N, where L is a letter and N is a digit.

The **Prescription** field is used to store information about prescribed drugs or active substances and possibly about dosing, i.e. information on when and what amount of medication the patient should take. For the **Prescription** field, as designed in the sample database proposal, there is no need to further specify or limit general properties.




Note

The **Prescription** field could be replaced by a separate table that records all prescribed patients' medications, including further information such as a list of all categorized drugs, possibilities of repeat prescription, confirmation of medication dispensation in a pharmacy, etc.

For the **Date of Check** field set the "Date/Time" data type and the general field property "Validation Rule". Assume that the correct control date should be at earliest tomorrow or

³World Health Organization, International Classification of Diseases, <https://www.who.int/standards/classifications/classification-of-diseases>

later. Use "Expression Builder" activated by clicking the button  located at the end of the "Validation Rule" line. Select the value of expression > and double-click on it. Through "Function: Built-in Functions" select „Expression category: Date and time" and "Expression Value: Date". We confirm the insertion of the expression for checking the **Date of Check** value with the **OK** button (Figure 3.26).

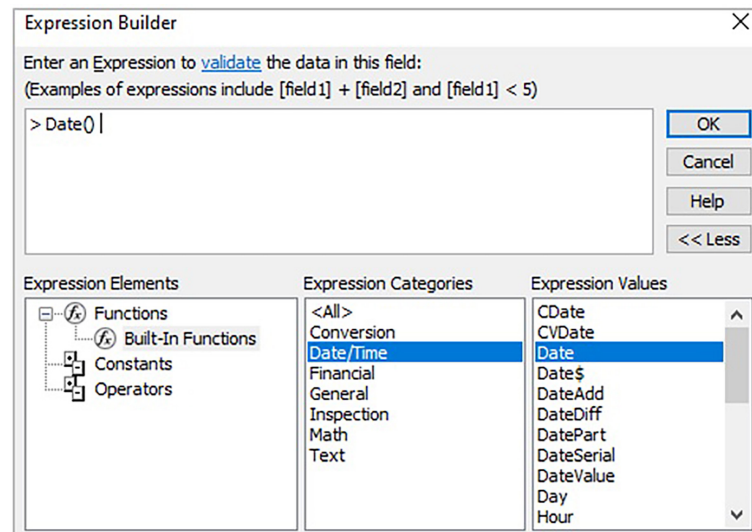



Figure 3.26: Validation rule for the **Date of Check** field.

If the validation rule identifies an inappropriate date while entering the date of check, for example, a past date, notify the user of an error with a system message containing the text entered in the "Validation Text" property. You can provide any explanatory text here, e.g. *"The earliest date of check can be tomorrow. Please verify the date and correct it."*

The **Note** field can be used to provide additional information regarding a patient's current examination. Such fields are typically found in all database tables to store additional information not belonging to the other fields of the table. Since anything could be entered here, no property will be set for this field.

All set properties of the "Examinations" table fields are saved by clicking the button  located in the upper-left corner of the MS Access application, or you will be automatically prompted to save changes when switching the table's design view to datasheet view. After creating and saving the fields in the "Examinations" database table, set a primary key for individual tables and create relationships between them before recording the first records (see the following Chapters 3.6 and 3.7).

3.5 Biometric data table

In the next database table, we will record the biometric data of our registered patients. We already know how to create a new database table, for example, via the **Create** tab and its **Table** option. Just like in previous cases, a new empty table named "Table1" will be created.

We will save this database table under the name "Biometric Data" upon the first switch from datasheet view to design view and define the following fields in it.

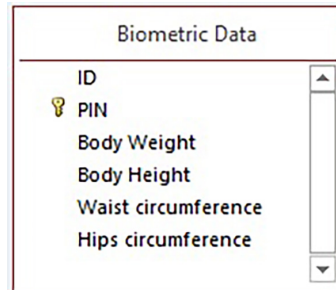


Figure 3.27: Selected fields of the Biometric Data table.

3.5.1 Design view of the Biometric Data table

The design of individual fields is carried out with regard to their expected content using the same procedure as with previous tables. For the ID field, we will keep the data type "AutoNumber", so each record has its own unique identifier and the user does not have to enter values into this field manually.

To know which patient the biometric data stored in the "Biometric Data" table belongs to, we will also add a PIN field to this database table, with properties identical to those in the previous two tables.

Fields marked as Body Weight, Body Height, Waist circumference, and Hip circumference will store corresponding numerical information in appropriate units, so we will set their data type as "Number" (see Figure 3.28).

Biometric Data			
	Field Name	Data Type	Description (Optional)
	ID	AutoNumber	
🔑	PIN	Short Text	Write patient's PIN in form XXXXXX/XXX(X)
	Body Weight	Number	Write parameter in kilograms
	Body Height	Number	Write parameter in meters
	Waist circumference	Number	Write parameter in centimeters
	Hips circumference	Number	Write parameter in centimeters

Figure 3.28: Design view of the Biometric Data table.



Note

Biometric data could be collected as part of patient examinations, or as an extension of the "Examinations" table. In such a case, we would add, for example, a Biometric Data ID field to the "Examinations" table and link the tables (see Chapter 3.7). By long-term monitoring of the patient, attending physicians would have a view of the development and changes of his/her recorded biometric data, and not just the current or the last recorded values.

3.5.2 General properties of the fields of the Biometric Data table

The correctness of values recorded in the field PIN is ensured mainly by the "Input Mask" property, which is set to the value 000000\~/0009 according to the procedures mentioned in Chapter 3.3.3 or Chapter 3.4.2.

For the **Body Weight** field with a data type of "Number", we set the general properties "Field Size", "Format", "Decimal Places", and "Validation Rule" with "Validation Text". The "Field Size" is set to **Single**, which allows entering numerical values with a floating decimal point (others, like **Decimal** - decimal numbers, can also be used). The "Format" is defined with the value **Fixed**, ensuring that any allowed entered value is not rounded to a whole number, but the display form will have decimal places. For the **Body Weight** field, which we will record in kilograms, one decimal place will suffice.

To ensure that weight values correspond to real patients' weights, we set the range of permitted values in the "Validation Rule", i.e. neither negative numbers nor weights of several hundreds or thousands of kilograms should be accepted. We then set the minimum and maximum allowed weight value for patients (later, these values can be changed if necessary). Let us consider initial values for minimum human weight of 2 kg and for maximum weight of 250 kg.

Using the "Validation Rule" property and the "Expression Builder" (Figure 3.29), we set the interval of allowed patient weight values. In the Expression Builder window select "Operators: Between", click twice on the selected operator with the left mouse button and the text: **Between** «Expression» **And** «Expression» is inserted into the expression in the upper part of the Expression Builder. Click on the first «Expression» and enter the value 2, similarly click on the second «Expression» and change it to the value 250. Values are entered without units, i.e. without kg. Confirm the use of the validation rule expression by pressing the **OK** button.

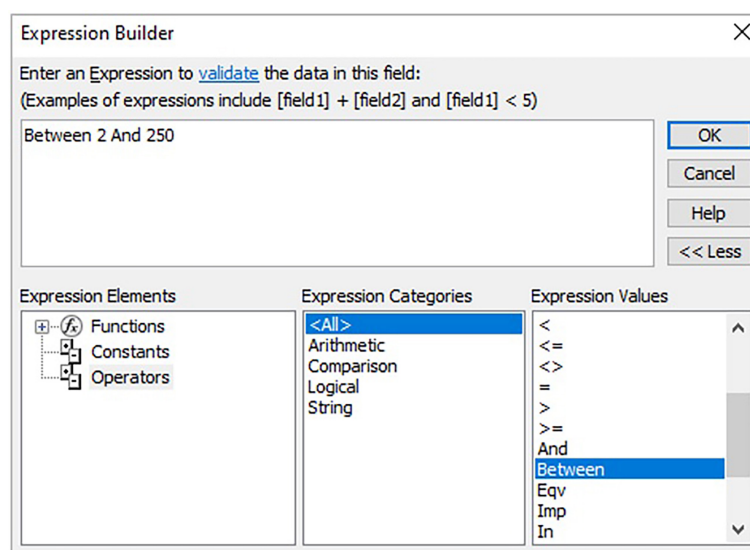


Figure 3.29: Expression Builder for the validation rule of the **Body Weight**.

In the "Validation Text" property row write an explanation that, in case the validation rule does not apply, it will inform the user that the entered value is not permitted. The validation text should be defined to clearly inform the user of the issue, i.e. why the value in the Body Weight field is unacceptable, and guide them how to solve it. For example, *"The patient's weight cannot be less than 2 and greater than 250 kg. Please check and enter the correct value in kilograms."* or *"Enter the correct weight!!!"*. The overview of modified general properties for the **Weight** field is shown in Figure 3.30.

General	Lookup
Field Size	Single
Format	Fixed
Decimal Places	0
Input Mask	
Caption	
Default Value	0
Validation Rule	Between 2 And 250
Validation Text	Enter correct Body Weight !!!
Required	No
Indexed	No <input type="checkbox"/>
Text Align	General

Figure 3.30: General properties of the Body Weight field.



Note

The range of values for the validation rule can also be written using other operators known, for instance, from mathematics. Then the validation rule could contain the following expression: ≥ 2 And ≤ 250 , which would work the same way as using the operator **Between**.

The **Body Height** field can be defined similarly to the **Body Weight** field, with the patient's height being entered and stored in the database table in meters (we can also opt for centimeters and suitably adjust property settings). The data type is set to "Number", and in the general properties of the **Body Height** field, we adjust property values as follows: "Field Size: Single", "Format: Fixed", "Decimal Places: 2". In the validation rule, we set the range of permissible parameter height values, for example, from 0.45 m to 2.3 m. The validation rule will then have the expression form: **Between 0.45 And 2.3**. Again, values are entered without units, i.e. in this case without m. We suitably adjust the information indicated in the validation text that will be displayed if the user enters the patient's height value outside this interval. An example of modified general properties of the **Body Height** field is shown in Figure 3.31.

The **Waist Circumference** and **Hip Circumference** of our patients will be recorded in centimeters. We presume that we will be using approximately the same value ranges (to cover both lean and obese patients), and therefore these two fields will have identical settings. Both fields will be of the "Number" type, and as in previous fields, we use the "Field Size" value **Single** and the "Format" value **Fixed**. The number of "Decimal Places" is set to 0 (zero).

General		Lookup
Field Size	Single	
Format	Fixed	
Decimal Places	2	
Input Mask		
Caption		
Default Value	0	
Validation Rule	Between 0.45 And 2.3	
Validation Text	Enter correct Body Height !!!	
Required	No	
Indexed	No	
Text Align	General	

Figure 3.31: General properties of the Body Height field.

Restriction of permissible waist and hip circumference values is also ensured here by the validation rule, which specifies the range, for example, from 30 to 250 centimeters (other values can be chosen based on real data). The validation rule will contain the expression **Between 30 And 250**, and the validation text might be, for example "*Enter correct waist circumference!!!*" or "*Enter correct hip circumference!!!*", as illustrated in Figure 3.32.

General		Lookup
Field Size	Single	
Format	Fixed	
Decimal Places	0	
Input Mask		
Caption		
Default Value	0	
Validation Rule	Between 30 And 250	
Validation Text	Enter correct Hip Circumference !!!	
Required	No	
Indexed	No	
Text Align	General	

Figure 3.32: General properties of the Hip Circumference field.



Practical Task

Add more fields identifying the patient's body structure to the database table "Biometric Data" and suitably set their properties. For example, add fields such as Chest Circumference, Eye Color, or others that might be useful in patient management.

From the biometric data obtained during patient examinations and saved into structured database table records, further useful characteristics can be determined or calculated. For instance, from the body weight and body height values of a patient, we can calculate the Body Mass Index – BMI, using the following well known formula:

$$BMI = \frac{w}{h^2} \quad (3.1)$$

where w is the value of body weight in kilograms and h is the value of body height in meters.



Note

If we work with the patient's body height in centimeters, then in the denominator of the equation 3.1, first divide the body height by 100 and then square it.

If we want the BMI value to be part of the "Biometric Data" table and automatically calculated, we will create a new field in the table design view with the name BMI and set the data type to "Calculated" (Figure 3.33).

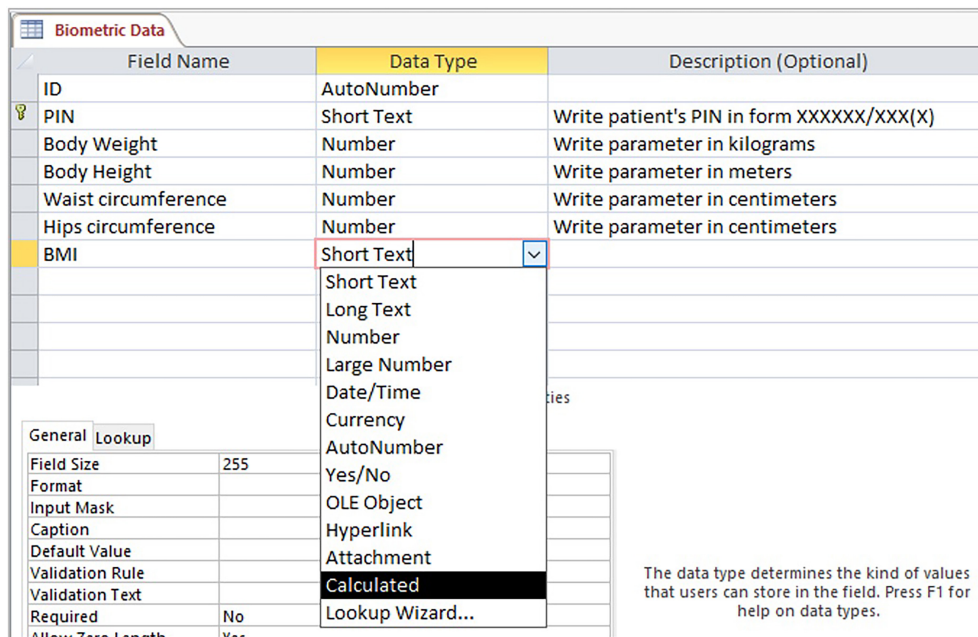


Figure 3.33: The value in the BMI field will be calculated.

After selecting the "Calculated" data type, the "Expression Builder" opens, where the "Biometric Data" table and its fields are available within the expression categories, from which values can be selected and added to the created expressions (Figure 3.34).

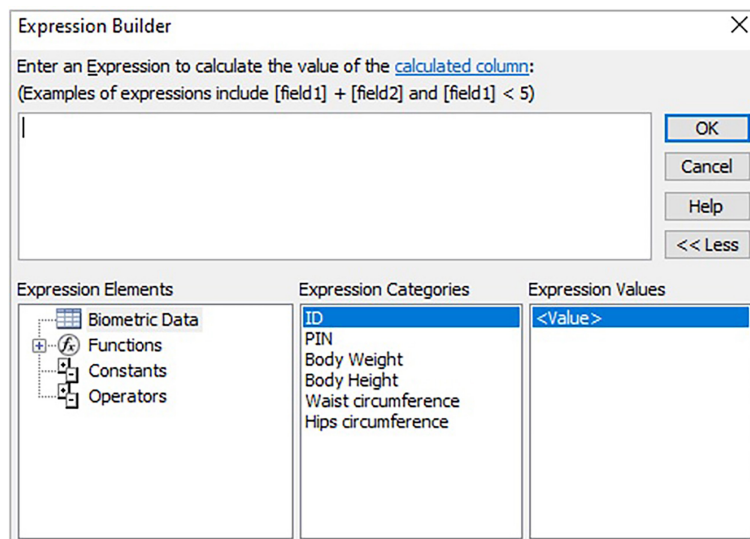


Figure 3.34: The Biometric Data table as part of expression elements.

According to equation 3.1, we use weight and height values so that the body mass index value is calculated in the BMI field for particular patient. In the expression categories, double-click on **Body Weight**, and this field is written into the expression builder window.

Next select the $/$ operator, confirm with a double-click (or enter it from the keyboard), and double-click to add the **Body Height** field from the Biometric Data table in the expression categories. Since the height value must be squared, we use the $^$ operator followed by the number 2. Thus, we compiled the expression to calculate BMI (Figure 3.35), the use of which is confirmed by pressing the **OK** button.

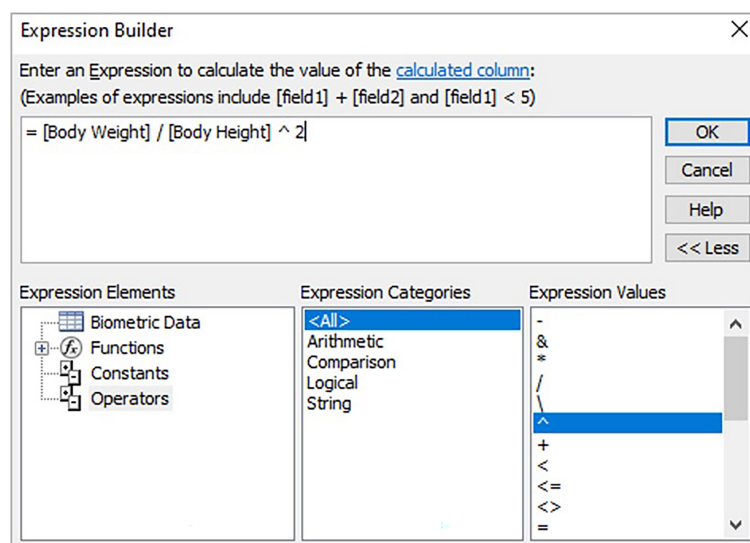



Figure 3.35: Expression Builder for the BMI field.

To ensure that the calculated BMI field result is readable in the database table's datasheet view, it is appropriate to set additional general properties for the field, such as "Result Type" with the value **Single**, "Format" with the value **Fixed**, and the number of decimal places, e.g. 2 (Figure 3.36). All changes for general properties of the "Biometric Data" table, and thus also for the BMI field, can be saved continuously with a button , or saved after all changes by switching from the table design view to its datasheet view.

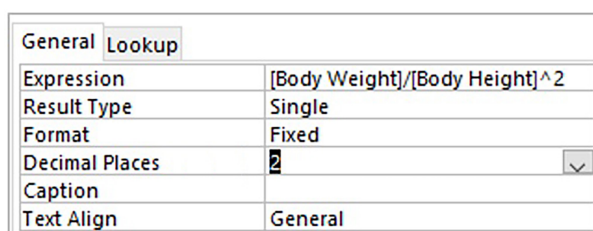


Figure 3.36: General properties of the BMI field.



Video

Creating a Calculated BMI Field

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

Another automatically calculated field that we will add to the fields of the "Biometric Data" table will contain values calculated from the numerical data of existing fields **Waist**

Circumference and Hip Circumference. By dividing these two characteristics, we obtain a dimensionless coefficient referred to as WHR (*Waist to Hip Ratio*), calculated as follows:

$$WHR = \frac{C_w}{C_h} \quad (3.2)$$

where C_w is the value of waist circumference and C_h is the value of hip circumference, while both characteristics are given in the same units, for example, centimeters.

To set the WHR calculation into a separate field of the "Biometric Data" table, we can proceed analogously to the case of BMI. In design view, we will gradually add a new WHR field, which will be of type "Calculated", enter the expression for the calculation, and adjust the properties for displaying and storing the WHR value. An example setting of the WHR field properties is shown in Figure 3.37.

General	
Expression	[Waist circumference]/[Hips circumference]
Result Type	Single
Format	Fixed
Decimal Places	2
Caption	
Text Align	General

Figure 3.37: General properties of the WHR field.

In the design view of the database table "Biometric Data", we have created and stored fields as shown in Figure 3.38.

Field Name	Data Type	Description (Optional)
ID	AutoNumber	
PIN	Short Text	Write patient's PIN in form XXXXXX/XXX(X)
Body Weight	Number	Write parameter in kilograms
Body Height	Number	Write parameter in meters
Waist circumference	Number	Write parameter in centimeters
Hips circumference	Number	Write parameter in centimeters
BMI	Calculated	
WHR	Calculated	

Figure 3.38: Design view of the extended Biometric Data table.

Other characteristics that could be calculated from recorded patient data include Body Surface Area – BSA. One of the most commonly used formulas is the so-called Du Bois formula:

$$BSA = w^{0,425} \cdot h^{0,725} \cdot 0,007184 \quad (3.3)$$

where w is the value of body weight in kg and h is the body height in cm . The resulting BSA value is in m^2 .

**Note**

In our sample database, we have already created three tables, which are currently empty and also independent, i.e. the data we would record in them would not be dependent on each other in any way, respectively the individual records in them would not be interconnected. Using an identical procedure, it is possible to create additional tables in the database for recording various groups of data, for example, according to the considerations mentioned in Chapter 1.4.

**Practical Task**

Add a new table "Urgent Information" to the database, in which, for example, fields such as Blood Group, Rh factor, Permanent Medications, Allergies, Infectious Diseases, etc., which may be important in further patient management, will be recorded.

In the next steps of designing the sample database, we will gradually set the primary keys for individual tables and create relationships between them.

3.6 Primary key of database table

In previous chapters, three basic tables of a sample database were created. However, a more complex database usually contains many more tables, allowing users to take advantage of digital data processing. As the number of records in individual tables increases, there is also a need for their identification. For this reason, it is necessary to ensure that each table contains a field, or a group of fields, that guarantees the unique identification of each record. Such a field is called the **primary** or **main key** of the table. The primary key is therefore the identifier of the table records, where each value of particular field is unique (not repeated), and thus it is not possible to find two identical records in the given table.

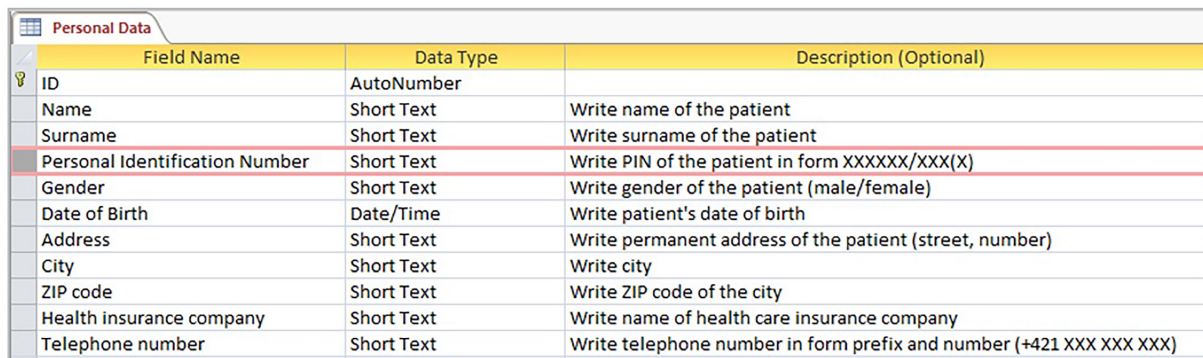
A database table can have only one primary key. When creating a new table, MS Access automatically creates a primary key field (ID field), which we can either keep depending on the nature of the group of information processed in the respective table, or we can create our own field with unique data and set this field as the primary key.

3.6.1 Primary key of the Personal Data table

Table "Personal Data" contains fields from which it is relatively easy to identify the one we should use as the primary key. It cannot be the patient's name, as we would not be able to have two patients with the same name in the records. Similarly, the primary key cannot be a surname or date of birth because we could not store two patients with the same surname or patients born on the same day in this table. Therefore, a suitable field for the primary key of the "Personal Data" table is the personal identification number (PIN), whose values are unique, and using it cannot lead to patient confusion, i.e. there can be two patients with the

same name, but not two patients with the same personal identification number (although such administrative errors have occurred in practice, but were corrected upon identification).

Looking at the "Personal Data" table in design view (like in Figure 3.5 or 3.39), we see that the primary key is marked as the ID field (a small yellow key located before the field name). It is the first field in the list, obtained automatically after creating a new table (Figure 3.3). If we want to make the **Personal Identification Number** the primary key, then in the design view of the table, we select this field with the mouse or keyboard arrows, or select the entire row by clicking the gray square before the field name (Figure 3.39), and confirm the **Primary Key** option in the **Tools** menu on the **Design** tab (Figure 3.40).



Field Name	Data Type	Description (Optional)
ID	AutoNumber	
Name	Short Text	Write name of the patient
Surname	Short Text	Write surname of the patient
Personal Identification Number	Short Text	Write PIN of the patient in form XXXXXX/XXX(X)
Gender	Short Text	Write gender of the patient (male/female)
Date of Birth	Date/Time	Write patient's date of birth
Address	Short Text	Write permanent address of the patient (street, number)
City	Short Text	Write city
ZIP code	Short Text	Write ZIP code of the city
Health insurance company	Short Text	Write name of health care insurance company
Telephone number	Short Text	Write telephone number in form prefix and number (+421 XXX XXX XXX)

Figure 3.39: Selecting the database PIN field.

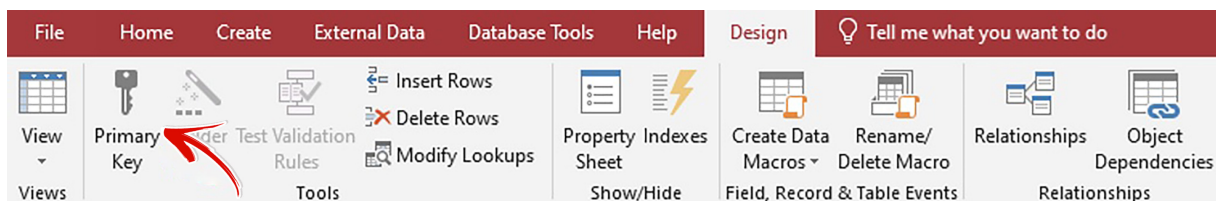
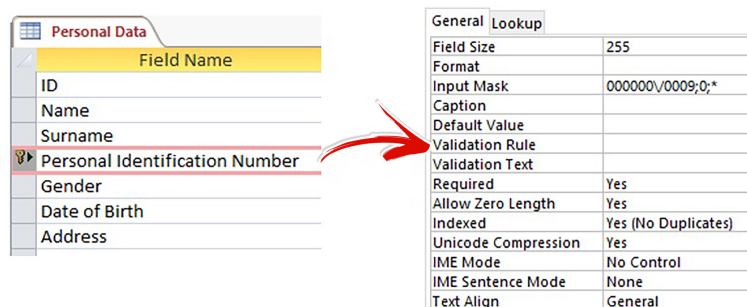


Figure 3.40: Primary Key option in the Design tab.

The primary key symbol moves from the ID field to the **Personal Identification Number** field. This setting also changes some general properties of the **Personal Identification Number** field, as the field marked as the primary key is automatically required, indexed, and without duplicates as shown in Figure 3.41.



Field Name	
ID	
Name	
Surname	
Personal Identification Number	
Gender	
Date of Birth	
Address	

General	
Field Size	255
Format	
Input Mask	000000\0009;0;*
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	Yes
Allow Zero Length	Yes
Indexed	Yes (No Duplicates)
Unicode Compression	Yes
IME Mode	No Control
IME Sentence Mode	None
Text Align	General

Figure 3.41: The PIN field designated as the Primary Key of the Personal Data table.

**Note**

Another way to designate any of the fields in a table as the primary key is to right-click on the respective field in design view and select the **Primary Key** option from the local menu.

**Video**

Primary key of the table

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

3.6.2 Primary key of the Examinations table

In the database table "Examinations", the primary key will be the **Examination ID** field, which is of type "AutoNumber". MS Access will directly manage the uniqueness of data in this field with each new record (see Figure 3.42).

Examinations		
Field Name	Data Type	Description (Optional)
ID Examination	AutoNumber	
Date of examination	Date/Time	Write date of examination
PIN	Short Text	Write patient's PIN in form XXXXXX/XXX(X)
Symptoms	Long Text	Write patient's symptoms
Diagnosis	Short Text	Write main diagnosis
Prescription	Short Text	Write name of the medicine
Date of check	Date/Time	Write date of new appointment
Note	Short Text	Write additional information

Figure 3.42: Examination ID field designated as the Primary Key of the Examinations table.

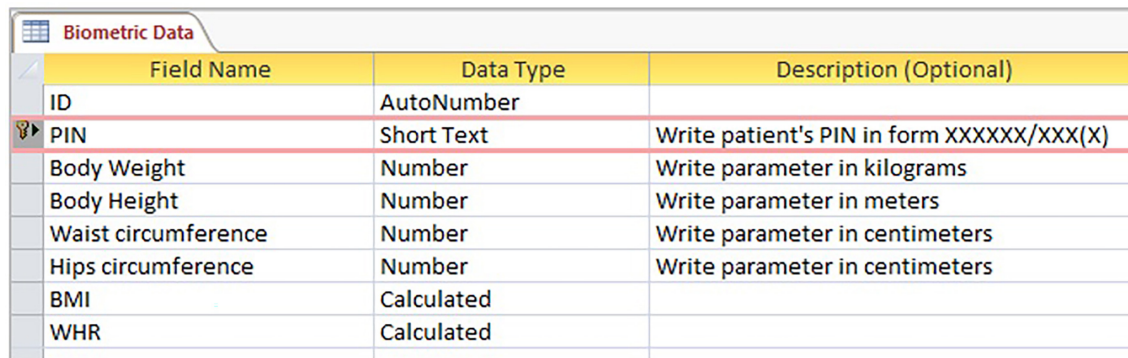
Fields such as **Date of Examination**, **Diagnosis** or **Symptoms** cannot be the primary key of the "Examinations" table, as in such a case we could perform only one examination per day, or have only one examination with a particular diagnosis or finding. Similarly, the **Personal Identification Number** field cannot be the primary key in this table. Otherwise, it would mean that the given patient could be examined only once, or it would be possible to record only one of their examinations. Furthermore, this field was added to the "Examinations" table to establish a link between examination records and records from the "Personal Data" table, i.e. to identify examinations of a specific patient.

If we were to add more fields to the "Examinations" table, whose data would be linked to a specific examination, than we would most likely also not be able to use them as the primary key of this table.

3.6.3 Primary key of the Biometric Data table


In the simplified design of this sample database, it was considered that with each visit of a patient to the clinic, the information about the patient's biometric data would be updated.

Therefore, one patient, and thus also one Personal Identification Number in the "Biometric Data" table, will always correspond to only one current record. From this, it follows that a suitable primary key for the "Biometric Data" database table will be the **Personal Identification Number** field. In design view, we set the **Personal Identification Number** field as the primary key of this table, following the known procedure as mentioned above.



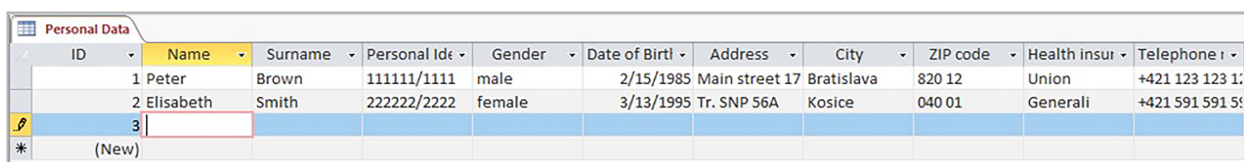
Field Name	Data Type	Description (Optional)
ID	AutoNumber	
PIN	Short Text	Write patient's PIN in form XXXXXX/XXX(X)
Body Weight	Number	Write parameter in kilograms
Body Height	Number	Write parameter in meters
Waist circumference	Number	Write parameter in centimeters
Hips circumference	Number	Write parameter in centimeters
BMI	Calculated	
WHR	Calculated	

Figure 3.43: PIN field designated as the Primary Key of the Biometric Data table.

Again, by changing the primary key of the table, some general properties of the PIN field are automatically adjusted so that personal identification number is mandatory, indexed, and without duplicates. We save the settings we made by clicking the **Save** button  or confirming when switching from the table's design view to its datasheet view.

3.6.4 Troubleshooting selected issues when setting up the primary key

The Primary Key of a table is a required (mandatory) field, meaning that when creating a record, a value must be entered into this field. The field must not be left empty. If we define or change the primary key afterward and already have records in the table where the value is missing in the field intended to be used as the primary key, then this change cannot be saved. In Figure 3.44, there is an example where there is one empty record (ID 3) in the database table, with only the ID used during registration and other data not filled, indicating one patient recorded without any identification details.



ID	Name	Surname	Personal Id	Gender	Date of Birth	Address	City	ZIP code	Health insur	Telephone
1	Peter	Brown	111111/1111	male	2/15/1985	Main street 17	Bratislava	820 12	Union	+421 123 123 123
2	Elisabeth	Smith	222222/2222	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali	+421 591 591 591
3										
*	(New)									

Figure 3.44: Empty record in the Personal Data table.

If we want to change the primary key from the ID field to the **Personal Identification Number** field, it is necessary first to fill in the PIN for patient into the empty record, or remove this empty record (right-click over the gray square of the record and confirm "Delete record" in the local menu).

Another common issue is duplicate values in the field intended to be used as the primary key. This occurs again when there are already records entered, and the primary key is to be set/changed afterward. An example of such a situation is a "Personal Data" database table with records as in Figure 3.45, where the PIN appears twice (for instance, when two or more patients are listed with the same **Personal Identification Number** value, or when a patient is recorded multiple times).

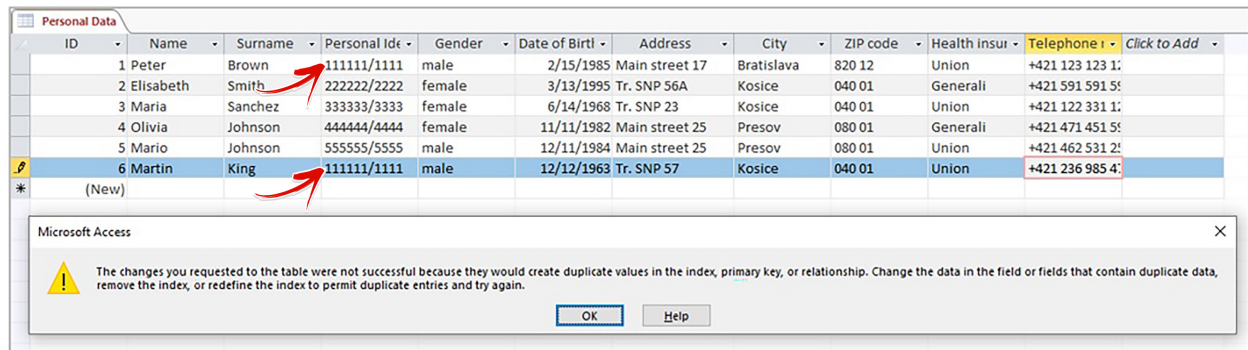


Figure 3.45: Duplicate records in the Personal Data database table and an error message when trying to use the PIN as the primary key.

With the existence of duplicate value(s) in the database field intended to be the primary key, the data must first be corrected in the table's datasheet view so that no value is repeated, and only then assign the primary key to the field in the table design view.

3.7 Relationships between tables

As explained in previous chapters, creating a single database table with a large number of fields and various information, for example, about patients, would be unclear and very complex in terms of maintaining and processing data. Therefore, the processed information is divided into several smaller, related groups of data and stored in separate database tables.

For the created tables (two or more) to form a relational database, relationships between them need to be established. That is, relationships between the individual tables in the database allow records in different tables to be linked.

3.7.1 Types of relationships between tables

In relational databases, several types of relationships between two database tables are distinguished:

- no mutual relationship between tables (their records do not relate in any way),
- one-to-one relationship (1:1) between tables,
- one-to-many relationship (1:N) between tables,
- many-to-many relationship (M:N) between tables.

3.7.1.1 Relationship 1:1

A one-to-one (1:1) relationship is a relationship in which one record from one table corresponds to at most one record in the other table. If we create or find such a relationship in a database, we should first consider whether we need it and if it is possible to combine the data from these tables into a single separate table. This situation often arises as a result of poor database design, where tables contain similar data.

However, if we need to separate the data, because one table will have many empty fields that are typical (useful) only for specific records, it is possible to use such a relationship for clarity. This can occur for example with specific parameters of diagnostic devices, supplementary information on medications that relate only to a few products, notes on a patient's personal data, etc.

Creating a 1:1 relationship requires both tables to contain a common field (column) and for the values in them to be unique. This is accomplished by using the same primary key in both tables, which will be used to link them. As a result, part of the record data is stored in one table and part in the other table. However, the user usually enters this information through a single form (see Chapter 4), so their administrative work in the database (system or application, whose data source is the given database) is not affected or made more complex or time-consuming by this division.

3.7.1.2 Relationship 1:N

The relationship between two tables of type one-to-many is the most commonly used relationship in relational databases. It is used between tables where one record in one table corresponds to multiple records in the other table. This type of relationship ensures that each piece of data is stored in the database only once, i.e. in one place, which is one of the main advantages of databases.

An example is a list of drug manufacturers, where each drug is made by one manufacturer, and each manufacturer can produce several types of drugs; a list of patients, where each patient can receive several vaccinations; a list of health insurance companies, where each insurance company may have several invoices issued, etc.

Creating a one-to-many relationship between two tables requires both tables to contain a common field (column), where the values in this field in one table are unique, ideally as the primary key of this table (it can also be another field where duplicate values are not allowed), and in the other table, the value of this field can theoretically occur infinitely many times. The primary key of the second table is usually another field.

3.7.1.3 Relationship M:N

A many-to-many relationship is defined when multiple records in one table correspond to multiple records in another table. Such a relationship is not usually defined directly but requires creating an auxiliary, so-called junction table that contains fields such as **connection**

ID (identifier and primary key of this junction table) and two fields (columns), the primary keys of both the first and second tables.

An example of an M:N relationship can be an surgery operation planning, i.e. the connection between a list of doctors and a list of surgeries. To explain, one doctor can perform multiple operations (not at the same time – an operation schedule) and vice versa, one operation can be carried out by surgical teams consisting of several doctors. The first table would be the list of doctors (contains, for example, doctor code, title, first name, last name, etc.), and the second table would represent the list of surgeries (contains surgery ID, surgery room, date and time of surgery, instrumentation, etc.). To create an M:N relationship between the doctors' table and surgeries table, a third – junction table is created, which contains only the doctor code and surgery ID. Then, a 1:N relationship is created between the doctors' table and the junction table, and another 1:N relationship between the surgery table and the junction table. This ensures the tables of doctors and operations are in an M:N relation.

The need for creating a junction table can be replaced by adding the second table's primary key field to the first, setting values through lookup (similarly as, for example, for the field **Gender** in section 3.3.3), and allowing multiple values in one field (for example by selecting this option in a wizard, see Figure 3.14). However, fields with multiple values cause problems when transferring data to other systems like SQL servers that do not support them. Therefore, if there is an expectation of sharing database data in more robust systems, fields with multiple value capability should preferably not be used.

3.7.2 Relationship between the tables **Personal Data** and **Examinations**

Each registered patient can have more than one record related to a doctor's visit (current illness, check-up examination, etc.) in the "Examinations" table. Therefore, we create a one-to-many relationship between the "Personal Data" and "Examinations" tables. After closing all database objects on the MS Access workspace, we open the **Database Tools** tab and confirm the **Relationships** option in the **Relationships** menu category (Figure 3.46).

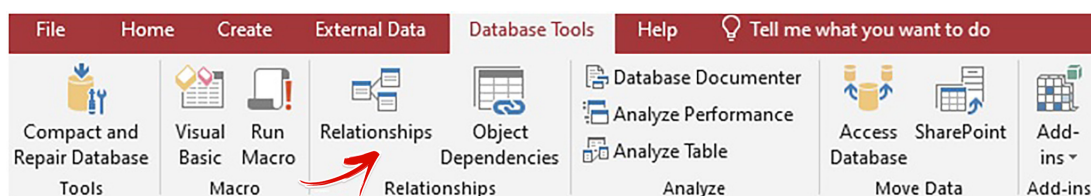


Figure 3.46: Relationships menu located on the Database Tools tab.



Note

When creating or modifying mutual relationships between database tables, it is necessary that these tables and other database objects that are based on these tables are closed. Ideally, the workspace environment of MS Access should be empty. Otherwise, it will not be possible to create/modify the relationship and MS Access will report an error that the database object/objects are in use.

Selecting the **Relationships** menu will open an empty **Relationships** tab (we have not yet created any relationship) and the "Show Table" window on it, from which it is possible to add tables or database queries to the relationship tab, between which we want to create relationships. If we have closed the "Table View" window, or have opened the Relationships tab repeatedly due to the need to edit existing relationships or add new ones, then on the **Design** tab, we confirm the **Show Table** menu (Figure 3.47).

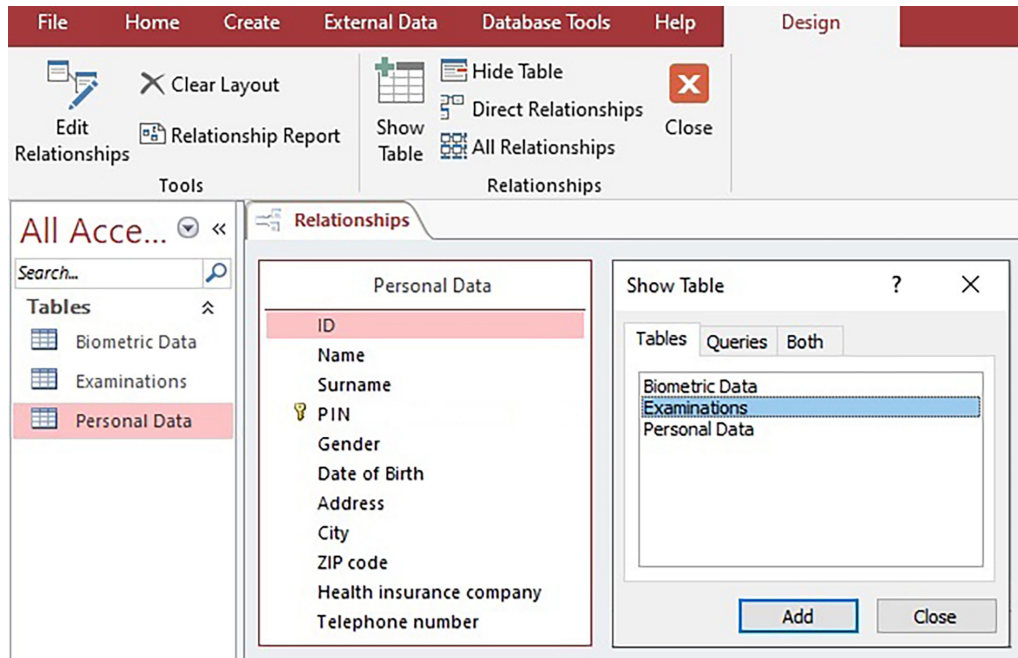


Figure 3.47: Editing relationships between database objects.

The **Show Table** option can also be obtained by clicking anywhere on the empty space of the mutual relationships window with the right mouse button and selecting it from the local menu (Figure 3.48).

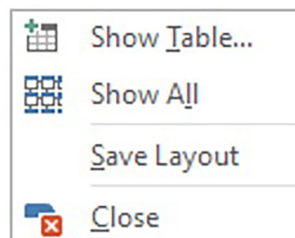


Figure 3.48: Local menu of the Relationships window.

From the "Show Table" window, we gradually add (using drag and drop method or the **Add** button) the database tables that we want to link (connect and thus establish the relationship between them) to the workspace of the relationship card. In this case, add the "Personal Data" and "Examinations" tables. You can close the "Table View" window, and the relationships card will contain both tables with lists of their fields, allowing you to adjust each table's size and position with the mouse (Figure 3.49).

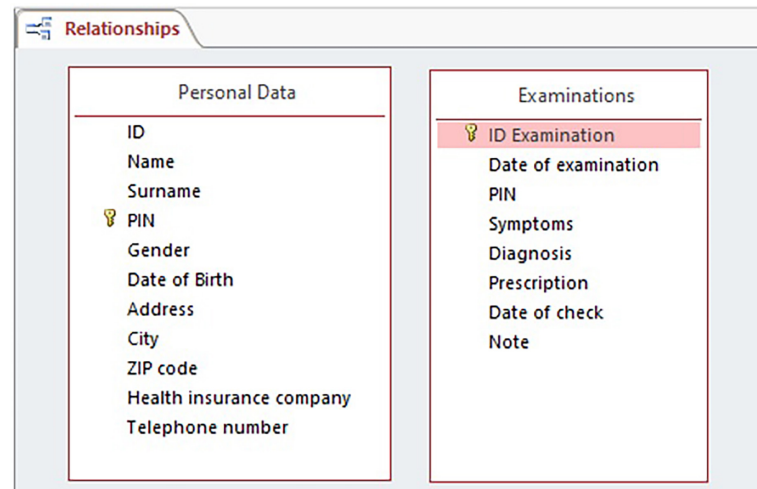


Figure 3.49: Tables added to the Relationships tab workspace.

Creating a relationship involves selecting the PIN field from the main table (so-called parent table) "Personal Data" and dragging it to the corresponding PIN field in the related table (so/called child table) "Examinations" using the drag-and-drop method. When we release the left mouse button, a dialog box for editing relationships opens (Figure 3.50).

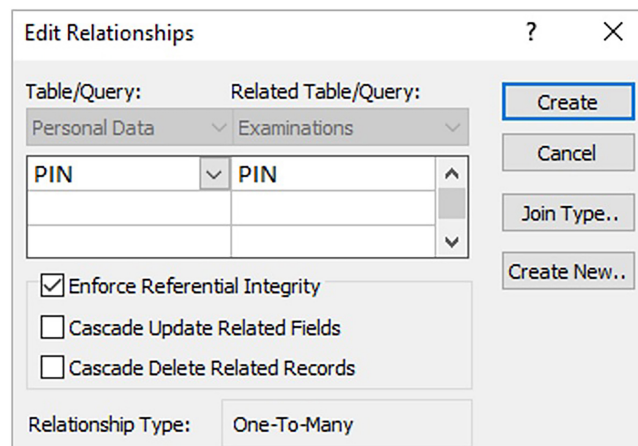


Figure 3.50: Relationship editing dialog box.

This dialog **Edit Relationships** displays the linked tables and the fields used to create the relationship (only fields of the same data type can be joined, otherwise the records will still be evaluated as unrelated). If incorrect fields were initially chosen, they can be changed in this dialog box by selecting from the field list of the relevant table.

The options located below this list of fields for the two linked tables are also important. The options are:

- *Enforce Referential Integrity* – automatically checks that relationship rules are maintained. For instance, it should not be possible to delete a diagnosis from the "Diagnoses" table if it is used in the "Examinations" table, or perform an examination for a patient not listed in the "Personal Data" table.

- *Cascade Update Related Fields* – the function that, when a record in one table is changed, ensures that the corresponding records in the other related table are automatically updated. It is only available if the "Enforce Referential Integrity" option is enabled.
- *Cascade Delete Related Records:* – the function that, when deleting a record in one table, ensures the automatic deletion of the corresponding records in the other related table. For example, if we delete a patient from the "Personal Data" table, it is possible to automatically delete all examinations belonging to this patient from the "Examinations" table. The use of this option must be carefully considered so that we do not delete data in the database that we will still need. The option is available only if "Enforce Referential Integrity" is enabled.

In the relationship between the "Personal Data" and "Examinations" tables, we select the first option ("Enforce Referential Integrity") and confirm the creation of the relationship by clicking the **Create** button. The dialog box "Edit Relationships" will be closed and a one-to-many (1:N) relationship will be created between the "Personal Data" and "Examinations" tables. From this point, it will only be possible to record entries in the "Examinations" table for patients who are registered in our database. This means that the patient must first be registered in the patient list (in the "Personal Data" table) before their examination information can be recorded (in the "Examinations" table). By enforcing referential integrity, symbols 1 and ∞ will be displayed at the ends of the tables' connection, indicating the type of relationship as shown in Figure 3.51.

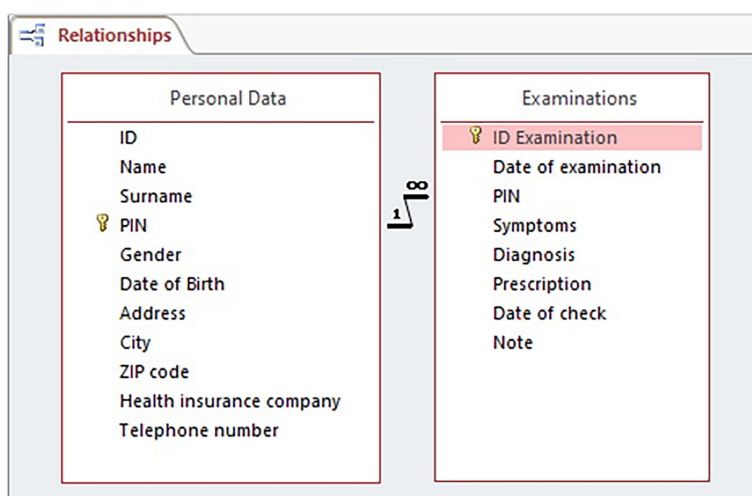


Figure 3.51: Relationship 1:N between tables Personal Data and Examinations.



Video

Relationship

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

3.7.3 Relationship between the tables Personal Data and Biometric Data

The group of data contained in the "Biometric Data" table is designed in such a way that each patient corresponds to only one record in this table. Therefore, in our sample database, we will link the database tables "Personal Data" and "Biometric Data" with a one-to-one relationship.

To create the relationship, ensure the tables are closed, and add the "Biometric Data" table through the "Show Table" window to the relationship card (Figure 3.47). The main table (parent table) is the "Personal Data" table, and we create the relationship from the "Personal Data" table to the related table (child table) "Biometric Data". In the "Personal Data" table field block, select the field PIN and drag it over the PIN field of the "Biometric Data" table. After releasing the left mouse button, the "Edit Relationships" dialog box will open (Figure 3.52).

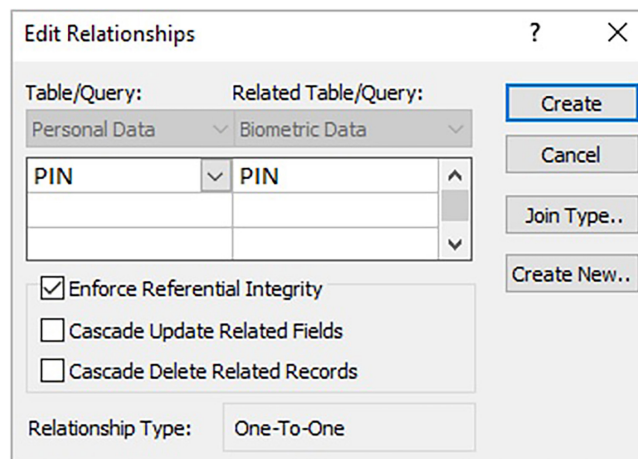


Figure 3.52: Dialogue box to edit relationship of Personal Data and Biometric Data tables.

In the relationship between the database tables "Personal Data" and "Biometric Data" we enforce referential integrity and confirm the creation of the relationship by pressing the **OK** button.

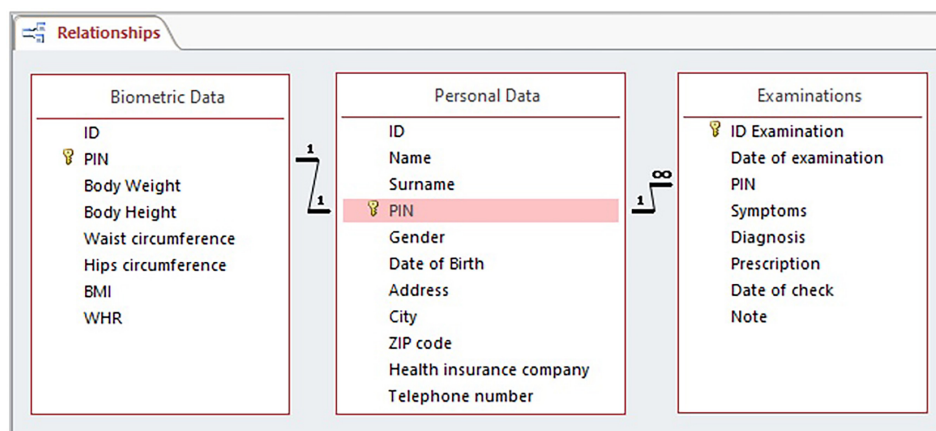


Figure 3.53: Relationships between database tables.

On the "Relationships" tab (Figure 3.53), we see the linking of database tables with a thick line and numbers indicating the type of mutual relationship, i.e. referential integrity is enforced. The relationship between the "Examinations" and "Biometric Data" tables does not need to be created, since all patient data can be accessed through already existing mutual relationships. In other words, it is not necessary to create links between each table and each other.



Note

A personal identification number (PIN) was used to create both sample relationships. However, relationships can be created using any fields from database tables. It is important that the data compared from the fields of two tables are of the same data type, and that it is possible to automatically evaluate (in the background operation of a relational database) whether the values match or not.

3.7.4 Modifications of relationships

In case it is necessary to modify an existing relationship, make sure that all database objects are closed, and then reopen the relationships tab window (**Database Tools – Relationships**). Double-click on the relationship you want to edit (the line connecting two tables) with the left mouse button, or right-click and confirm the **Edit Relationship** option from the local menu (Figure 3.54).

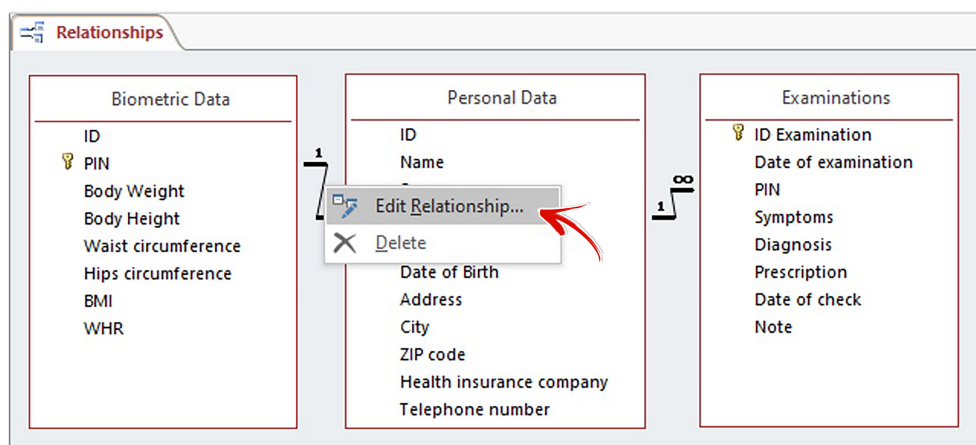



Figure 3.54: Modifying the relationship between database tables.

The dialog box for editing relationships opens (e.g. Figure 3.52), where we make all necessary changes, confirm them with the **OK** button, and save the changes on the Relationships tab by confirming the **Save** button .

Conversely, if during the design or while using the database it turns out that one of the relationships needs to be removed, select it with the mouse and press the **DELETE** key on the keyboard, or open the local menu over the relationship with the right mouse button, from which confirm the **Delete** option (Figure 3.55).

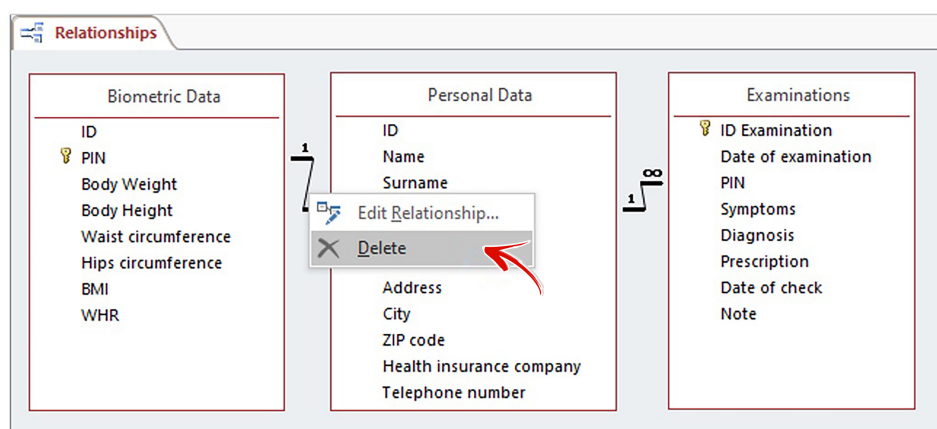


Figure 3.55: Removing a relationship between database tables.

Confirming the removal of a relationship is necessary, as a verification message with the text: *Are you sure you want permanently delete the selected relationship from the database?*, will notify us to prevent any unwanted removal of the relationship (Figure 3.56).

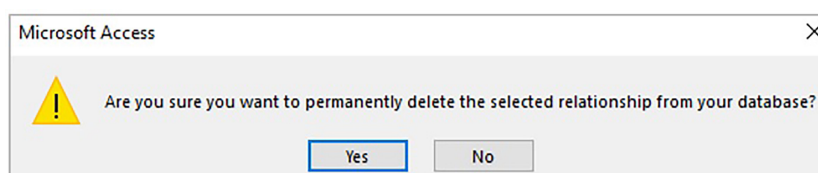


Figure 3.56: Prompt to confirm the removal of the relationship.

3.8 Datasheet view of the Personal Data table

In previous steps, we have designed the fields of the database table "Personal Data", set appropriate data types for them and specified selected general properties. We also linked the "Personal Data" table with other database tables. With this, we have completed the basic design of the tables and we can now start using them and entering the relevant records into them.

In the "Personal Data" table, we can start recording the first patients and entering their personal data into it. If the table is in design view, then on the **Home** tab, we confirm the **Datasheet View** from the **View** menu (Figure 3.57).

If the "Personal Data" table is closed, simply double-click it with the left mouse button in the database objects list on the navigation pane, and it will open and display in its datasheet view (default table view).

To register a new record, in this case the first patient, start, for example, by entering the name of the patient. By entering the first character (in any field of the table), the record will be marked as edited (a pencil icon appears in a gray square at the beginning of the record) and an ID number is automatically assigned to the record, which cannot be changed

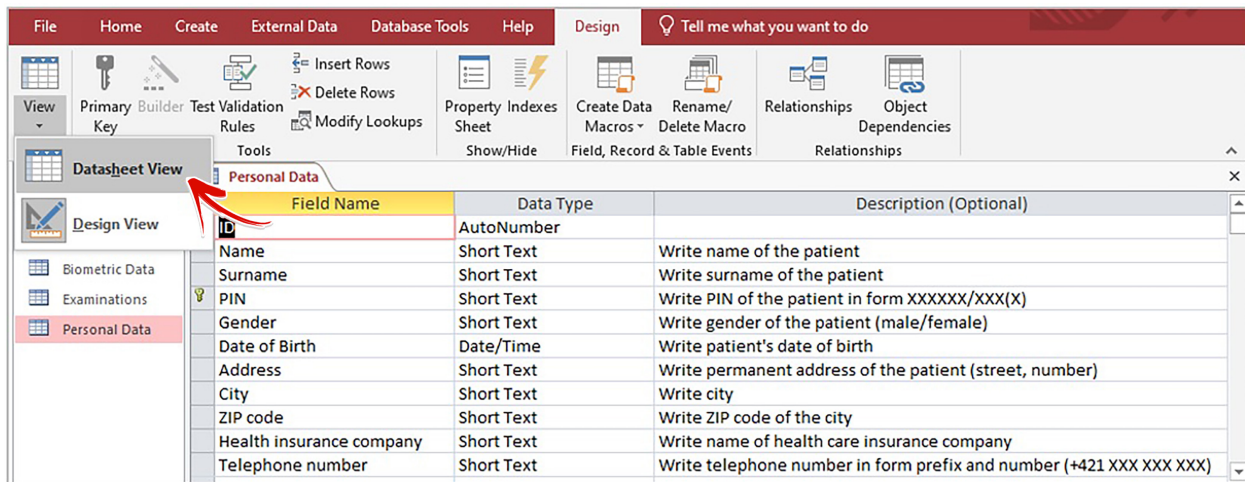


Figure 3.57: Change the view of the Personal Data table.

or edited (auto-numbering property), as shown in Figure 3.58. If a saved record is deleted, its ID number will not be reused.

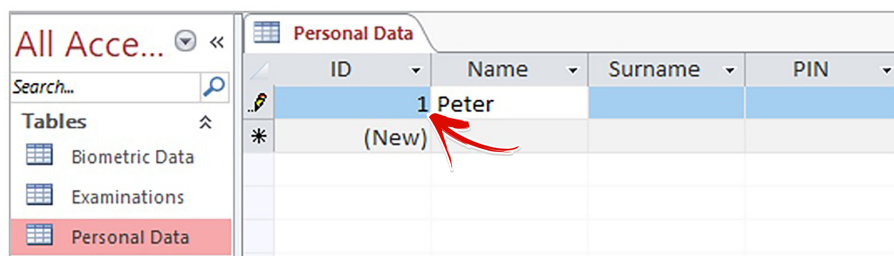


Figure 3.58: Registration of the first record in the Personal Data table.

The next fields of the record can be navigated either with the mouse cursor or by using the TAB key on the keyboard, which moves the cursor to the next table cells.

Fields with selection options, i.e. fields that contain a list of values (lists created when designing a database table, for example, for gender – see Figure 3.59) are filled by selecting a value from the given field's menu, or by entering the initial letter of the desired value, and the first option starting with this letter is selected. You can continue with the second, third, or even other letters if the list is extensive and contains similar items starting with the same characters. Once the desired value is highlighted, just press the TAB key, the selected value from the list is used, and the cursor moves to the next field.

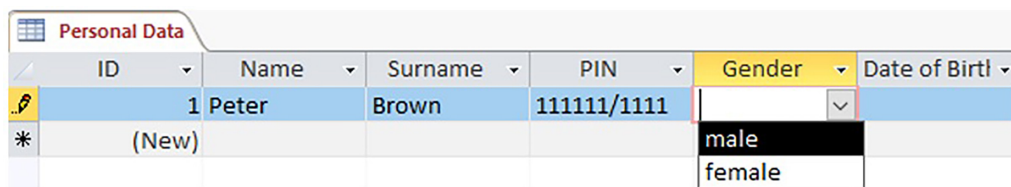


Figure 3.59: Selecting a value for gender from the list.

If the default column width of a field is insufficient, i.e. the information is not fully visible, than this field can be expanded with the mouse by holding the left button down on the edge of two adjacent fields in the table header and moving it to the right (or left in case of decreasing the width) until reaching the desired width. It is also possible to expand a field width in the datasheet view of the table to the width of the largest value contained in the field by double-clicking the left mouse button on the edge of two adjacent fields.

Database fields defined with the data type "Date and Time" automatically have a calendar icon embedded in the cell menu (Figure 3.60). Therefore, date values can be entered in these fields manually by typing the numbers in the chosen date format or by selecting the date from the calendar, which is familiar from various common office applications.

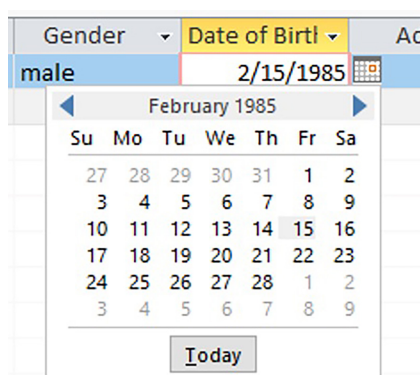



Figure 3.60: Selecting a date using a calendar.

In each field, data corresponding to the purpose of the field should be entered. Users can read hints for each field in the status bar, where a description of the active field is displayed as it was prepared during the design phase in the table design view.

Records in database tables are saved automatically when moving to another table record or when closing the table datasheet view (provided all required fields have been filled). However, records can also be saved manually by pressing the **Save** button  located on the Quick Access Toolbar or from the table's local menu, which can be accessed by right-clicking on its name (Figure 3.61).

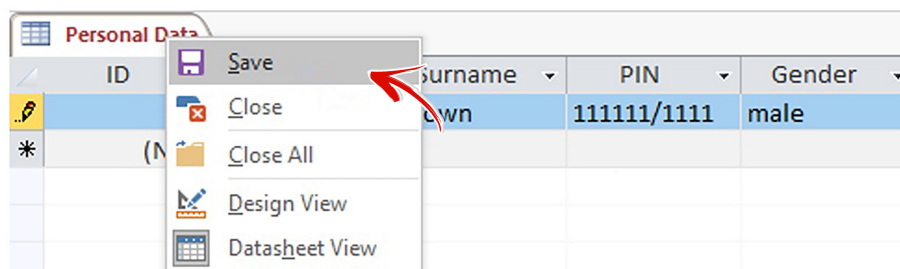


Figure 3.61: Saving a database table record.

**Note**

There should be no empty records in database tables. Currently, it is possible to store records in the "Personal Data" table only with a specified PIN (the ID is generated automatically). To avoid empty records in database tables, it is advisable to identify the most important information and mark their fields in the design view as required by setting the "Required" property to yes. On the other hand, it is not advisable to mark all fields as required, as there may be cases where the values are unknown/unavailable, but the user will still have to fill in these fields to save the record.

**Practical Task**

In the database table "Personal data" set up general field property "Required" for fields First Name and Surname. Also, register 10 new records in the "Personal Data" database table, i.e. register ten new patients.

3.9 Datasheet view of the Examinations table

In the database table "Examinations" (Figure 3.62), we will record entries from outpatient treatments of patients who are registered in the "Personal Data" database table. If a new patient arrives for treatment, it will not be possible to save the entry in the "Examinations" table until they are registered in the "Personal Data" table. MS Access will notify us of this with a message indicating that a related record is missing in the "Personal Data" table.

Examinations					
Date of exam	PIN	Symptoms	Diagnosis	Prescription	Date of check
1/16/2025	222222/2222	Patient feels tired, has difficulty breathing. Cough: 1 week, T:38.5°C	Flu	antibiotics	1/22/2025
1/16/2025	666666/6666	Patient complains of leg pain and swelling	Lower limb varicose veins	Lioton gel, Detralex	
1/16/2025	333333/3333	Patient has difficulty breathing, cough and fever are present	Bronchitis	antibiotics	1/30/2025
1/16/2025	444444/4444	Patient has inflammation and tearing of the eyes	Eyelid inflammatory	Oftalmoseptonex ointment	1/20/2025
* 2/6/2025					

Figure 3.62: Datasheet view of the Examinations table.

The fields in the "Examinations" table contain extensive text information, which becomes less clear in narrow cells. Increasing the field width usually does not suffice to make all recorded information, such as a **Prescription**, **Symptoms**, etc., visible. To facilitate working with text, correcting potential errors, or even formatting text, it is possible to use the active cell Zoom. The cell zoom in the table acts as a separate window, like a basic text editor, and we open it for the given cell by pressing the SHIFT+F2 key combination. The zoom opened over the **Symptoms** field is shown in Figure 3.63.

possible to record new examinations in the subdatasheet of the "Personal Data" table, with all records and values saved only in the table they belong to.

In the patient records, the "Examinations" table may not be a subdatasheet, but it can be changed to a subdatasheet of another table (with which the "Personal Data" table has a relationship), for example, by opening the **More** menu located in the **Records** category on the **Home** tab and the **Subdatasheet** menu within it (Figure 3.65). The subdatasheet can also be removed here, or its records can be expanded or collapsed in bulk.

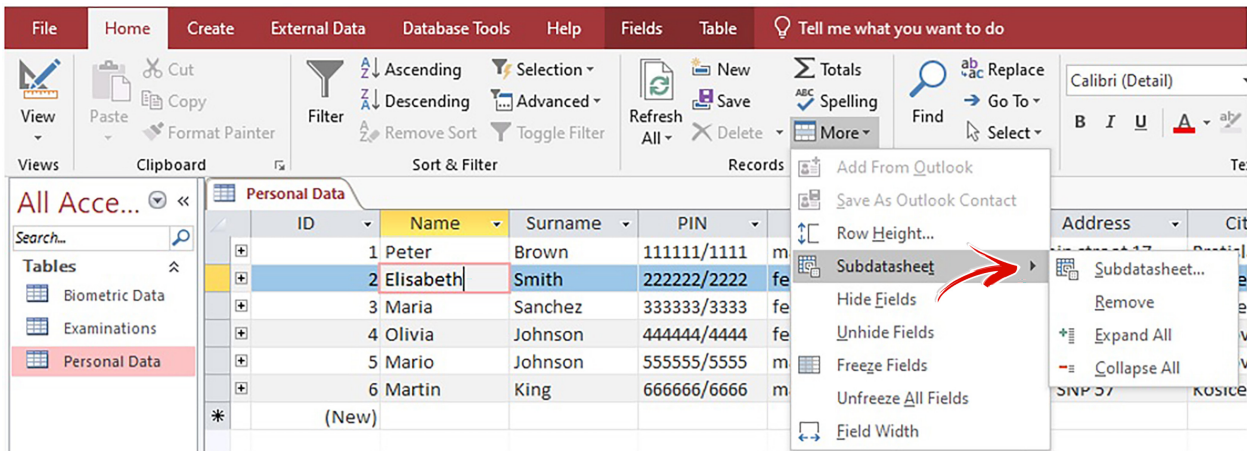


Figure 3.65: Options to change subdatasheets.

3.10 Datasheet view of the Biometric Data table

Working with the "Biometric Data" table is very intuitive and relatively well-structured, as we mainly record simple numerical data there. After establishing a relationship with the "Personal Data" table, we ensured that biometric data can only be recorded for already registered patients, and that each patient can have at most one record here. There is no need to expand the mandatory fields beyond the automatic numbering of the ID and the primary key PIN, since it is unknown which data we will know for individual patients and which we will not.



Practical Task

Enter the data of all registered patients into the "Biometric Data" database table by adding as many entries to the "Biometric Data" table as there are patients in the "Personal Data" table.

Even in the "Biometric Data" database table, it is possible to find and use related table subdatasheets, depending on how the mutual relationships were created between these tables in our model database. The datasheet view of the "Biometric Data" table with subdatasheets is shown in Figure 3.66.

Biometric Data									
ID	PIN	Body Weigh	Body Height	Waist circum	Hips circum	BMI	WHR	Click to Add	
1	111111/1111	56	1.75	86	90	18.29	0.96		
2	222222/2222	89	1.95	102	100	23.41	1.02		
ID	Name	Surname	Gender	Date of Birth	Address	City	ZIP code	Health insur	
2	Elisabeth	Smith	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali	
ID Exam	Date of exam	Symptoms			Diagnosis	Prescription	Date of check		
	1/16/2025	Patient feels tired, has difficulty breathing. Cough: 1 week, T:38.5°C			Flu	antibiotics	1/22/2025		
*	(New)	2/6/2025							
*	(New)								
3	333333/3333	112	1.98	105	117	28.57	0.90		
4	444444/4444	64	1.72	88	100	21.63	0.88		

Figure 3.66: Datasheet view of the Biometric Data table with subdatasheets.

**Note**

You can change the order, add, or remove subdatasheets in specific tables using **Subdatasheets** menu in **Home** tab under **Records** category in **More** menu.

When working with multiple database objects simultaneously (for example, having multiple tables open) and inserting new records into a table (for example, "Examinations") through an embedded table (e.g. "Examinations" in the "Personal Data" table), newly added records do not appear when checking records in the target table (e.g. "Examinations"). This is because the table was opened before new records were entered into it through another database object. In such a case, simply locate and confirm the **Refresh All** menu in the **Home** tab under the **Records** category (Figure 3.67) to refresh all open objects, meaning all data stored in the respective tables will be loaded. Alternatively, to view all records, you can close and reopen the given table.

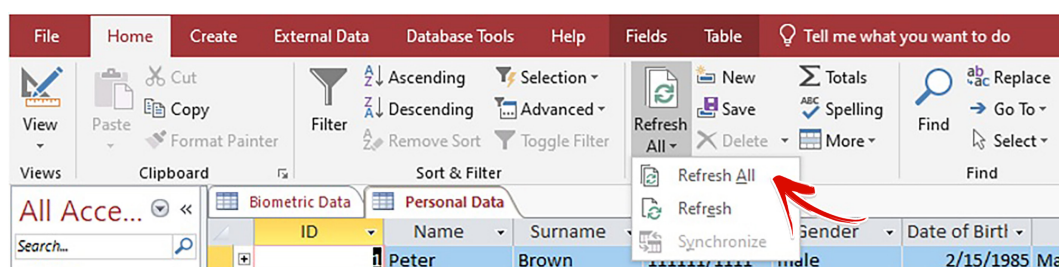


Figure 3.67: The Refresh All menu for reloading the content of open database objects.

3.11 Import of external data

To facilitate the preparation of the database structure and streamline the work of its users, tables can also be created by importing data from external files and using them as sources of values for lookup fields. These are often various registers and lists, such as lists of categorized drugs, ZIP codes of cities, lists of diseases, etc., whose manual entry would be laborious, prone to errors, and time-consuming.

We will demonstrate the possibility to create and use such a table using the example of the International Classification of Diseases (ICD)) version 11. According to the International Classification of Diseases, each disease has a unique code and name. Individual categories of diseases also include additional information. Therefore, the table, which we will call the abbreviation ICD11-en, will contain only two columns (fields), namely **Code** and **Disease Name**. Both fields will be defined as the data type "Short Text".

Since the ICD itself contains more than thirtysix thousand items, the data into the database table will be imported from an external source. In the first step, we will need the data source itself. We can find it on World Health Organization website⁴. We save the records to a computer, for example in the form of an MS Excel workbook (other formats are also possible), and modify them as needed into the required format (two columns, merging sheets, etc.). The resulting ICD workbook can be assembled as shown in Figure 3.68.

	A	B
1	Code	Disease Name
2		Certain infectious or parasitic diseases
3		- Gastroenteritis or colitis of infectious origin
4		- - Bacterial intestinal infections
5	1A00	- - - Cholera
6	1A01	- - - Intestinal infection due to other Vibrio
7	1A02	- - - Intestinal infections due to Shigella
8	1A03	- - - Intestinal infections due to Escherichia coli
9	1A03.0	- - - Enteropathogenic Escherichia coli infection
10	1A03.1	- - - Enterotoxigenic Escherichia coli infection
11	1A03.2	- - - Enteroinvasive Escherichia coli infection
12	1A03.3	- - - Enterohaemorrhagic Escherichia coli infection
13	1A03.Y	- - - Intestinal infections due to other specified Escherichia coli
14	1A03.Z	- - - Intestinal infections due to Escherichia coli, unspecified
15	1A04	- - - Intestinal infections due to Clostridioides difficile

Figure 3.68: Structure of the International Classification of Diseases table in an MS Excel workbook.

Then, on the **External Data** tab in MS Access, in the **New Data Source** group, open **From file** and confirm **Excel** option (Figure 3.69).

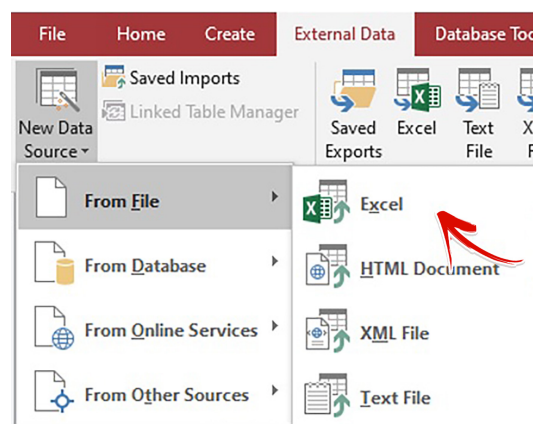


Figure 3.69: Menu of the External Data tab.

⁴World Health Organization, International Statistical Classification of Diseases and Related Health Problems (ICD), <https://icd.who.int/browse/2025-01/mms/en>

A dialog box will appear (Figure 3.70), in which use the **Browse** button to find the path to our data source, i.e. our MS Excel workbook file with the table for the International Classification of Diseases (e.g. "C:\Users\Documents\ICD11-en.xlsx"). In the same dialog box, we also choose one of three options that determines how and where to import this external data into our current database.

Figure 3.70: Dialog box for loading external data.

It is possible to save external data in the current database in the following ways.

- **Import the source data into a new table in the current database** – the data will be imported into a new table, and the user will be able to define its properties. A copy of the source data will be created, which will be a full-fledged part of the database. If the source data is changed after this import, these changes will not be reflected in the database table.
- **Append a copy of the records to the table** – the data will be appended to one of the existing tables in the database. If the database does not contain any tables (e.g. a new database), then this option will not be active. Changes to the source data after this import will not be applied to the database.
- **Link to the data source by creating a linked table** – MS Access will create a table containing a link to the source data. Such data will not be editable in the database, but if the source data changes, this change will also be reflected in the linked table.

We leave the first option selected and confirm the selection with the **OK** button, which starts the **Import Spreadsheet Wizard** from the worksheet (Figure 3.71).

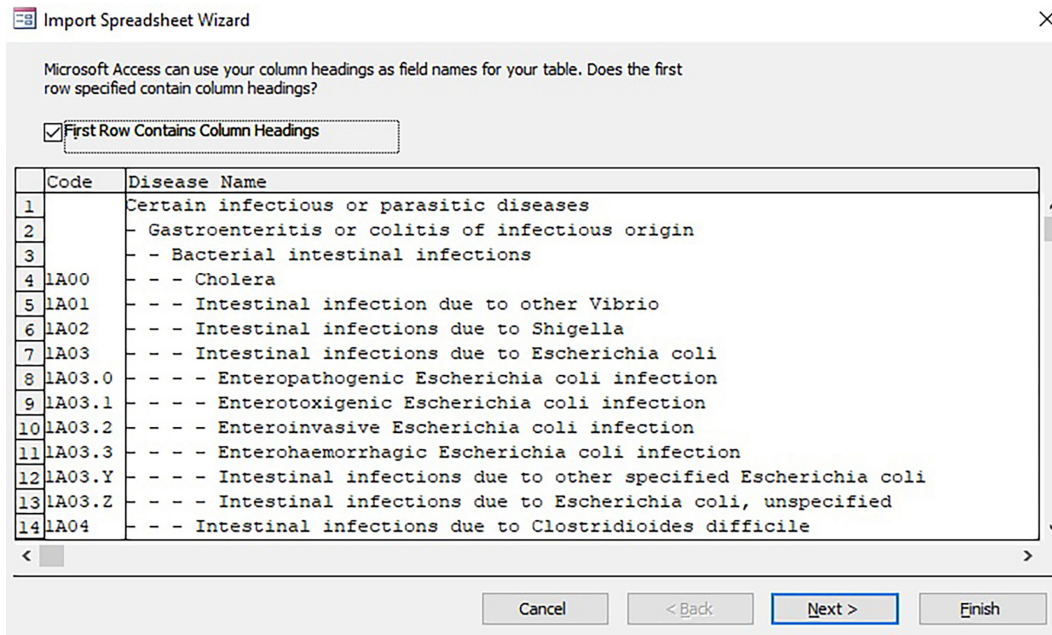


Figure 3.71: Import Spreadsheet Wizard.

The first row of the imported data contains field names, so we check the option "First row contains column headings". MS Access will use the same field names in the newly created table. However, these names can be changed during or after the data import is completed. Click the **Next** button to move to the next step of the wizard, where it will be possible to adjust the properties of the imported data (Figure 3.72).

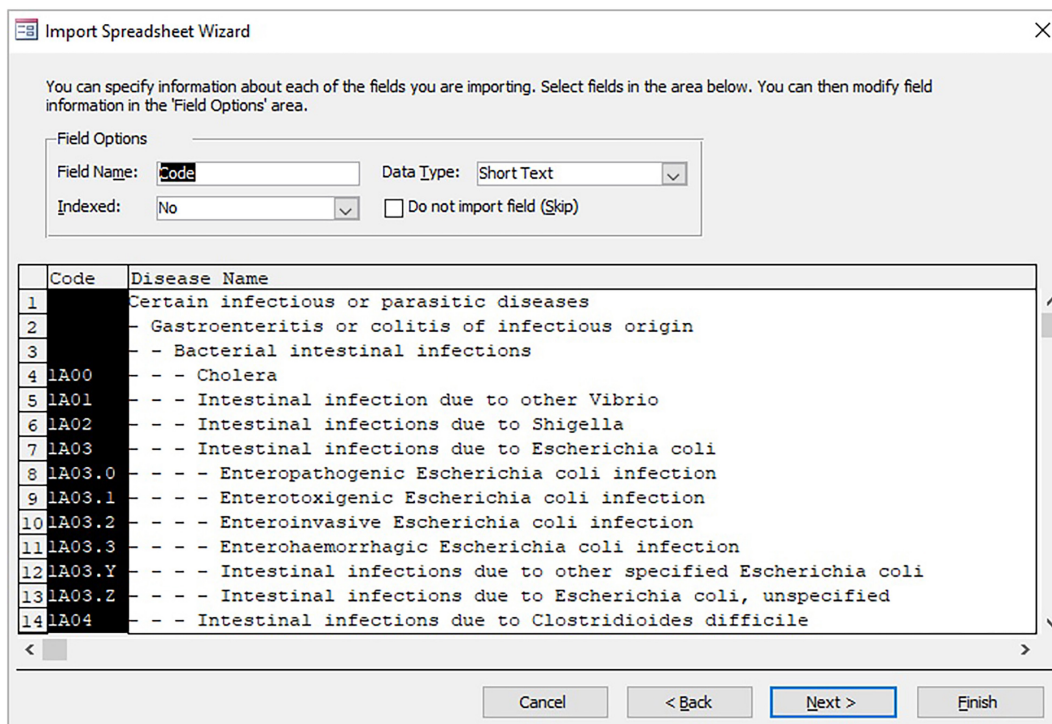


Figure 3.72: Edit properties of imported fields.

In the lower part of the window, click on the column whose properties you want to check or possibly modify (the column will be highlighted in black), and in the upper part of the window marked as **Field options**, adjust the desired parameters. For each imported field, gradually check:

- **Field Name** – change it if you do not want to use the original name taken from the source data,
- **Data Type** – MS Access identifies and suggests the most appropriate data type according to the input values, choosing the wrong type can cause some values to be ignored during import,
- **Indexed** – for the Code field, change it to "Yes (No Duplicates)" for faster search options,
- **Do Not Import Field (Skip)** – check this if you do not want to import the column (in our table, we want to import both columns).

After finishing selecting the imported fields and possibly adjusting their properties, click the **Next** button, and in the next step, decide if and what primary key you will use in the new database table (Figure 3.73).

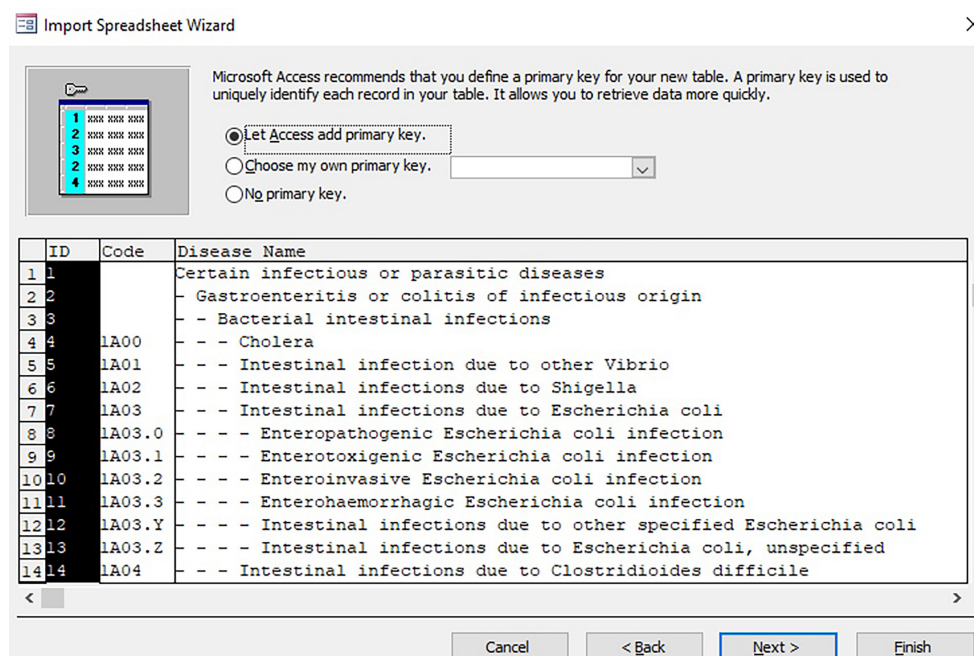


Figure 3.73: Defining the primary key of imported data.

It is possible to select one of three options.

- **Let Access add primary key** – MS Access adds an additional AutoNumber field and will automatically insert unique identifier values for all imported records.
- **Choose my own primary key** – select one of the imported fields as the primary key. In our table, the field Code can be used for this purpose.

- **No primary key** – select this, if the new database table is not supposed to have a defined primary key.

In our sample import, the field **Code** will be selected as the primary key of the new table, so we confirm the **Next** button. In the final step of the import wizard from the spreadsheet, enter the name of the target table into the field **Import to Table**. MS Access usually offers the name of the worksheet from the MS Excel workbook from which the data is imported. We will use a name according to the content of the table, i.e. ICD11 (Figure 3.74).

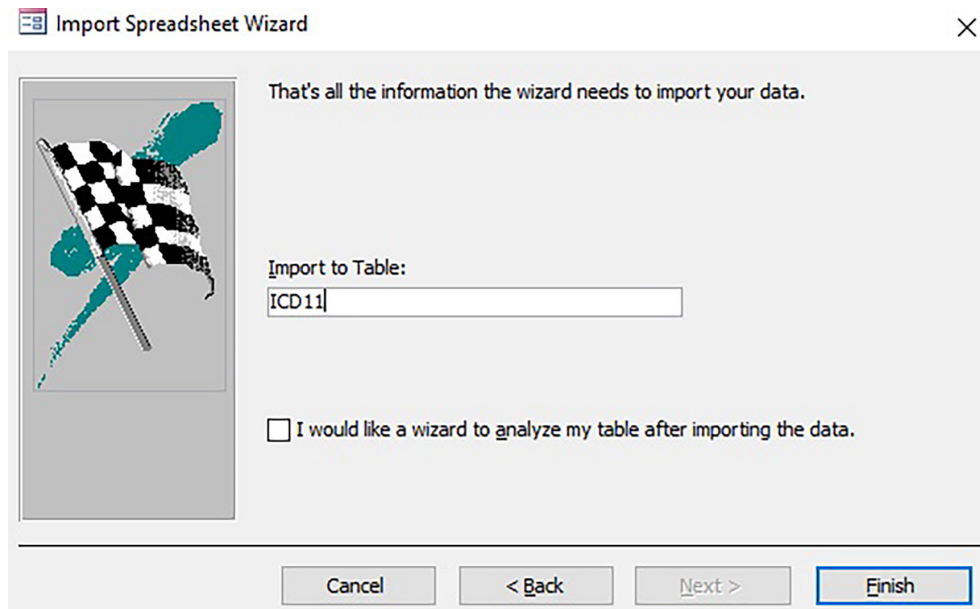


Figure 3.74: Choosing the name of the new table.

If the answer to the item "I would like a wizard to analyze my table after importing the data" is yes, then also check this option (verifying if the imported data does not duplicate). Finally, import the data into the database using the **Finish** button.

If MS Access was able to import all or at least some data, the wizard will display a page with the status of the import operation. It is also possible to save detailed import information as a specification, making any repeated import with the same settings easier. If the data import is not successful, MS Access will display an appropriate message and propose a solution to fix the problem.

Our import of the source data (import we conducted) went smoothly and the newly created table contains more than eleven thousand records – diagnoses according to the International Classification of Diseases. Its datasheet view is shown in Figure 3.75.

In case we want database users to select patient diagnosis information in the "Examinations" table from the list and not input them manually, then in the "Examinations" table design view, we change the **Diagnosis** field data type to "Lookup Wizard..." and follow it to use the "ICD11" table as the data source for the lookup (similar to the process of creating a list for the gender field, but we do not enter values manually, see Chapter 3.3.3). The lookup wizard also creates a relationship between the tables "Examinations" and "ICD11".

ICD11		
ID	Code	Disease Name
1		Certain infectious or parasitic diseases
2		- Gastroenteritis or colitis of infectious origin
3		-- Bacterial intestinal infections
4	1A00	--- Cholera
5	1A01	--- Intestinal infection due to other Vibrio
6	1A02	--- Intestinal infections due to Shigella
7	1A03	--- Intestinal infections due to Escherichia coli
8	1A03.0	---- Enteropathogenic Escherichia coli infection
9	1A03.1	---- Enterotoxigenic Escherichia coli infection
10	1A03.2	---- Enteroinvasive Escherichia coli infection

Figure 3.75: Datasheet view of the ICD11 table created by importing from an MS Excel table.

The properties of the diagnosis lookup field can be modified if necessary, in the design view of the "Examinations" table in the Lookup properties tab, as shown in Figure 3.76.

General Lookup	
Display Control	Combo Box
Row Source Type	Table/Query
Row Source	SELECT [ICD11].[ID], [ICD11].[Code], [ICD11].[Disease Name] FROM ICD11;
Bound Column	1
Column Count	3
Column Heads	Yes
Column Widths	0";1.5";10"
List Rows	16
List Width	2"
Limit To List	Yes
Allow Multiple Values	No
Allow Value List Edits	No
List Items Edit Form	
Show Only Row Source Values	No

Figure 3.76: Properties of the diagnosis selection lookup field.

In the datasheet view of the "Examinations" table, it will only be possible to select diagnoses from the imported International Classification of Diseases list (Figure 3.77).

Diagnosis	
Code	Disease Name
	Certain infectious or parasitic diseases
	- Gastroenteritis or colitis of infectious origin
	-- Bacterial intestinal infections
1A00	--- Cholera
1A01	--- Intestinal infection due to other Vibrio
1A02	--- Intestinal infections due to Shigella
1A03	--- Intestinal infections due to Escherichia coli
1A03.0	---- Enteropathogenic Escherichia coli infection
1A03.1	---- Enterotoxigenic Escherichia coli infection
1A03.2	---- Enteroinvasive Escherichia coli infection

Figure 3.77: Selecting a diagnosis in the Examinations table.

**Review Questions**

1. What principles do we follow in designing a database table?
2. What does a database table consist of?
3. What is a database field?
4. What is a record in a database table?
5. What views of a database table do we know?
6. List at least five general properties of a field and state what they are used for.
7. In what cases do we use an input mask for a database field?
8. What is the purpose of a validation rule for a database field?
9. What do we call a field that does not contain duplicate values and also identifies records in the table?
10. What types of relationships in relational databases do we know?
11. By what database tool do we create a relationship between two tables?
12. Is it possible to create a relationship between tables whose primary key is not the same database field?
13. What is the purpose of referential integrity in relationships?
14. What does the M:N relationship mean?
15. What prerequisites must database tables meet in order to link them?

Chapter 4

Forms

At the end of the previous chapter, we tried to populate the tables of a sample database with initial data – records, verifying that with the correct database structure design and field property settings, data management is relatively straightforward. However, as records in individual tables increase, the clarity of the information begins to diminish, and tables that exceed the size of one screen cause complications for users, at least in terms of quick orientation in their records.

Therefore, in real systems and applications, a graphical environment is created for users to access the desired information without having to enter the source tables. In MS Access relational databases, such a user interface is typically a form or group of forms.

A database form represents a separate object that facilitates the work with data and generally displays information about a required record or group of records more quickly and clearly. During the form design phase, it is possible to format the display of database fields and their descriptions, create custom form layouts with any number of fields, even combined from various database tables (or queries). In ergonomically well-designed forms, the data is then more readable, organized, and more efficiently handled.

According to the form controls used, which mainly include fields displaying data, we distinguish between bound and unbound forms. A bound form is a form that is directly connected to the source data, i.e. to a database table or query, allowing these data to be displayed, edited, and new ones added. On the other hand, an unbound form is not directly connected to a database table or query, i.e. source objects. However, it usually consists of various other controls, such as labels, buttons, images, etc., which have their functions and are intended for working with the system or application, which is based on a database. In this chapter, we will mainly focus on bound forms, but several of them will also use unbound objects – controls that streamline working with data in the form.

Forms can be created in various ways. If a form is to be used for working with data from a single object (table, query), it can be created automatically from the given source. However, the resulting form typically needs to be minimally adjusted in design to match the rest of the application's environment. A form can also be generated using a form wizard, which simplifies the design process, such as selecting the type of field layout, inserting subforms,

etc. In design view, forms can also be created manually by defining all objects individually according to the form designer's preferences.

Unlike tables, where two views (datasheet and design) were available, for forms there are three types of views available:

- **Form view** – allows editing and entering new data,
- **Design view** – allows changes to form design, setting properties, and functioning of existing elements, and adding new objects and fields,
- **Layout view** – allows optimization of form control properties, but unlike design view, it also displays data loaded from source objects. It is used for fine-tuning the form design but does not allow editing or adding new data/records.

4.1 Automatic form creation

One of the quickest ways to create a bound form is to automatically generate it using the **Form** menu. First, in the navigation pane, select a data source, such as the database table "Personal Data", and then confirm the **Form** option on the **Create** tab in the **Forms** menu (Figure 4.1).

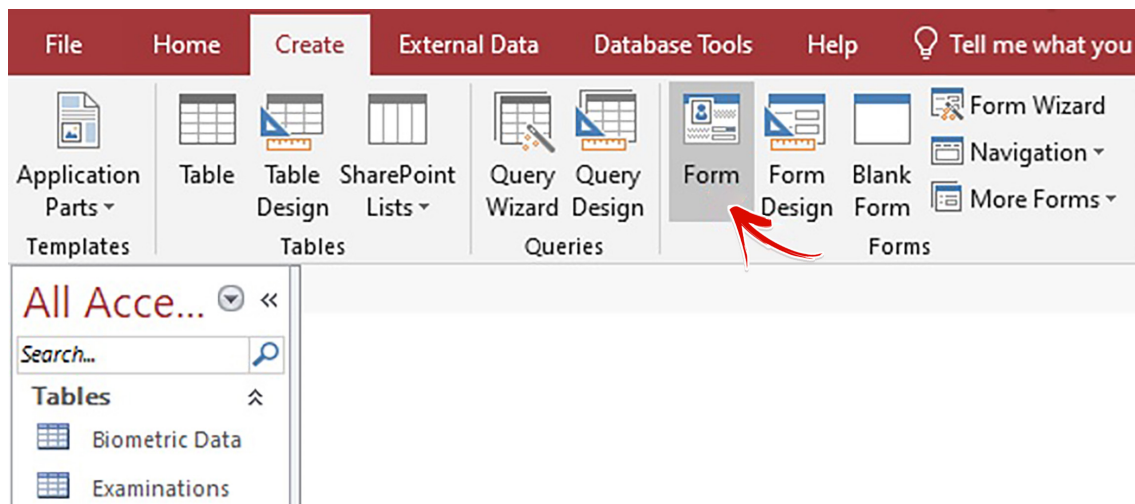


Figure 4.1: Creation of an automatically generated form for the Personal Data table.

On the workspace of MS Access, a new bound form from the "Personal Data" database table is generated and opened in layout view, meaning it will display data loaded from the source table (it will not be possible to edit or add new data) and, if necessary, changes to its design can be made. Such an automatically generated form will not yet be saved in the list of database objects in the navigation pane (if we are satisfied with the functionality and design, we will save it later). After making any necessary design adjustments, the form's view can be changed, for example, on the **Home** tab in the **View** menu (Figure 4.2).

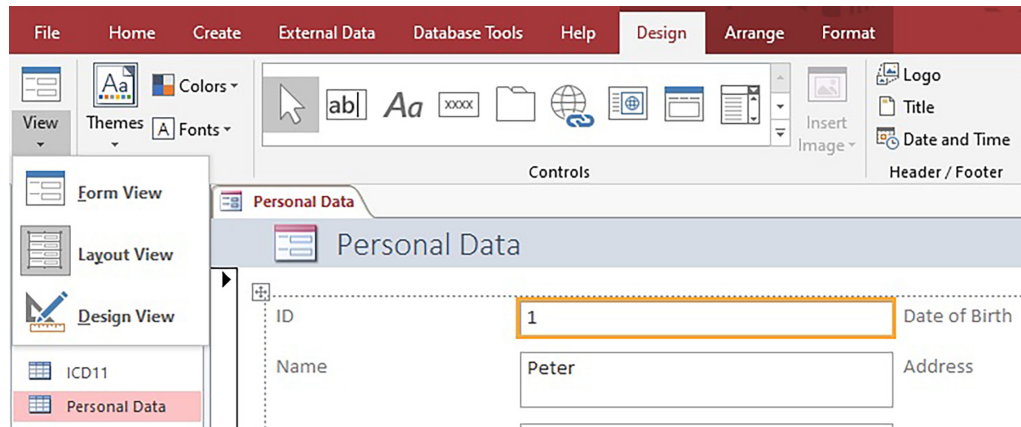


Figure 4.2: Changing form views using the View menu.

To change the form view, it is also possible to use the local menu obtained by right-clicking above the name of the form (Personal Data) forming the tab on the MS Access workplace, or clicking on the the viewing options located in the bottom right corner of the status bar (Figure 4.3).

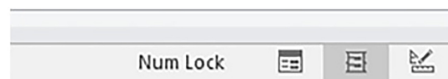



Figure 4.3: Options for switching between form views in the status bar.

The "Personal Data" form contains all the database fields present in the "Personal Data" table. Since a one-to-many relationship has been specified between the "Personal Data" and "Examinations" database tables, a subform is generated on the "Personal Data" form displaying as a datasheet containing all recorded examination entries of the patient whose personal details are shown by the "Personal Data" form (i.e. not a complete list of all examinations of all patients, see Figure 4.5).

**Note**

If we want to insert new records into the database using this automatically generated form, the form must first be saved. Saving the form and any changes to its settings can be done progressively by pressing the **Save** button  located on the Quick Access Toolbar, or during view changes if any design or functional adjustments were made to the form.

The save dialog will appear automatically during the first view change or the first save of such automatically generated form, where the user decides either to save it or not. Our first form will be saved under the name "Personal Data" (Figure 4.4), and later, if it will no longer be needed, it can be deleted anytime just like any other database object.

After saving the form, a new tab named "Forms" will appear in the navigation pane (the ways to display database objects in the navigation pane can be changed in navigation settings), where we can also find our first form named "Personal Data".

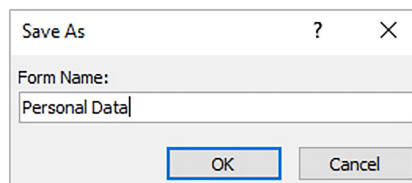


Figure 4.4: Saving the Personal Data form.

4.2 Form view of the Personal data form

In case the "Personal Data" form is closed, locate it in the list of objects in the navigation pane and open it by double-clicking with the left mouse button, or using the **Open** option from the local menu (right-click on the name of the form). The "Personal Data" form will open in form view, which is the default view for forms. In form view, we can edit data and add new records (Figure 4.5).

ID Exam	Date of exam	Symptoms	Diagnosis	Prescription	Date of check	Note
*	(New)	1/21/2025				

Figure 4.5: Form view of the automatically created Personal Data form.



After opening the "Personal Data" form, the record of our first patient is loaded and displayed in the relevant personal data fields. We can access records of other patients using the arrow buttons located in the status bar at the bottom of the form (Figure 4.6).



Figure 4.6: Options for browsing records located in the form status bar.

**Note**

Our first automatically generated form also includes a datasheet of related records recorded in the "Examinations" table. Therefore, it is possible to identify two status bars in one form, but the "inner" one belongs to the examinations sheet.

If we need to register a new patient in the database, it is not necessary to go through all the records, but we use the **New (blank) record** button . The form will move us to the end of the list and display blank fields ready for registering a new patient. All the attributes of the individual fields that were set in the source database table "Personal Data" are retained and apply to the fields displayed in the "Personal Data" form. After filling in the patient's data (at least the required fields), the record can be saved by moving to the previous or next record, i.e. record of another patient, or by using the **Save** button  located on the Quick Access Toolbar.

**Note**

We do not add a new entry to the database using a form by overwriting displayed data, but use the **New (blank) record** button where all fields are empty. Otherwise, it would only change the existing patient's data.

**Practical Task**

Register five new patients to the sample database via the "Personal Data" form, i.e. add five new records with patient identification data.

Newly entered and saved patient records (through the "Personal Data" form) can be found in the database table "Personal Data" (all data changes are always saved into the source tables, forms only display them and allow working with them), for example, as shown with records with ID 10 and 11 in Figure 4.7.

Personal Data										
ID	Name	Surname	PIN	Gender	Date of Birth	Address	City	ZIP code	Health insur	Telephone number
10	Dylan	Mckenzie	101010/1010	male	12/3/1999	Tovern street 6	Bratislava	820 11	self-payer	+421 556 622 332
1	Peter	Brown	111111/1111	male	2/15/1985	Main street 17	Bratislava	820 12	Union	+421 123 123 123
11	Barbara	King	121212/1212	female	12/23/2001	Main street 95	Bratislava	820 11	Generali	+421 626 262 626
2	Elisabeth	Smith	222222/2222	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali	+421 591 591 595
3	Maria	Sanchez	333333/3333	female	6/14/1968	Tr. SNP 23	Kosice	040 01	Union	+421 122 331 122
4	Olivia	Johnson	444444/4444	female	11/11/1982	Main street 25	Presov	080 01	Generali	+421 471 451 592
5	Mario	Johnson	555555/5555	male	12/11/1984	Main street 25	Presov	080 01	Union	+421 462 531 254
6	Martin	King	666666/6666	male	12/12/1963	Tr. SNP 57	Kosice	040 01	Union	+421 236 985 474
7	Sarah	Newton	777777/7777	female	12/1/2000	University street 1	Kosice	040 01	Union	+421 111 222 333
8	Mariah	Black	888888/8888	female	1/1/1982	University street 3	Kosice	040 01	self-payer	+421 222 255 558
9	Michael	Omar	999999/9999	male	4/11/1969	Tr. SNP 5	Presov	080 01	self-payer	+421 111 122 211

Figure 4.7: Personal Data table extended by new records entered via the form.

If the "Personal Data" database table was open and new records added via the form are not displayed (visible) in it, then refresh the data in open objects with the **Refresh All** button located on the **Home** tab in the **Records** menu (Figure 4.8).

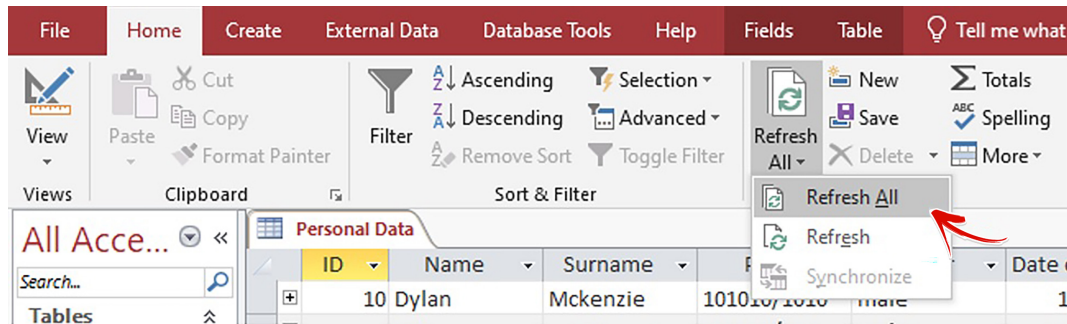


Figure 4.8: Refreshing records in Personal Data table.

4.3 Custom form design

In case the automatically generated forms do not meet our needs, or if the finalization of their adjustments is complex or time-consuming, it is possible to design a custom form by first creating a blank form and then defining its objects exactly according to our preferences. A blank form can be created, for example, by selecting the **Form Design** option in the **Create** tab under the **Forms** menu (Figure 4.9).

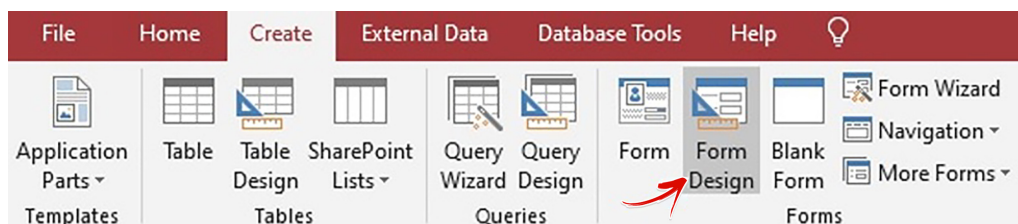


Figure 4.9: Creating a blank form in design view.

In a blank form, it is possible to design the detail section (Figure 4.10), i.e. area where data is displayed and managed, as well as the form header and footer (suitable for labels, control buttons, etc.). Fields from individual database tables can be added to the form's detail section using the **Add Existing Fields** option in the **Design** tab's **Tools** menu.

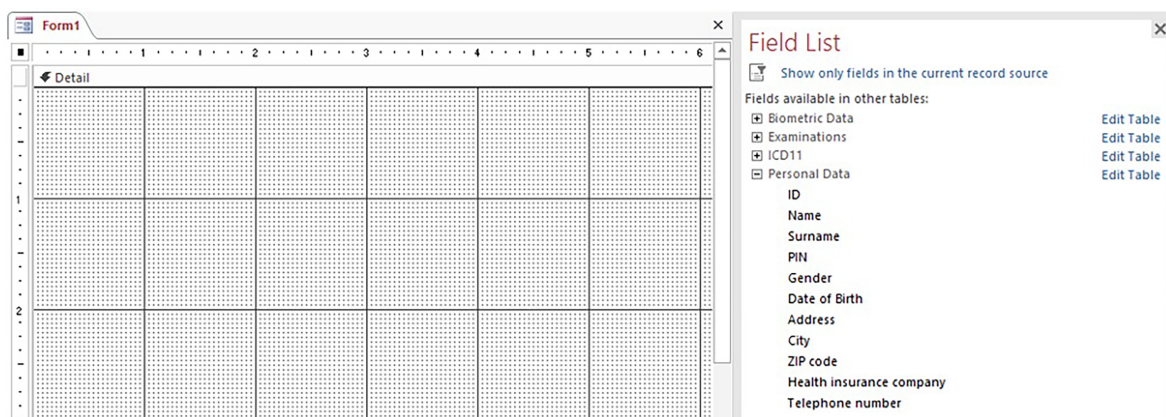


Figure 4.10: Design view of a blank form and a list of database table fields.

The field list contains all the database tables with their fields, which can be added to the designed form by double-clicking the left mouse button or using the standard drag-and-drop method. Database fields can be combined, but it is important to consider the relationships between tables and their fields to avoid potential functionality problems when using them in form view.

Alternatively, a blank form can be opened using the **Blank Form** option located in the **Create** tab under the **Forms** menu (Figure 4.11). This type of form will be opened in layout view.

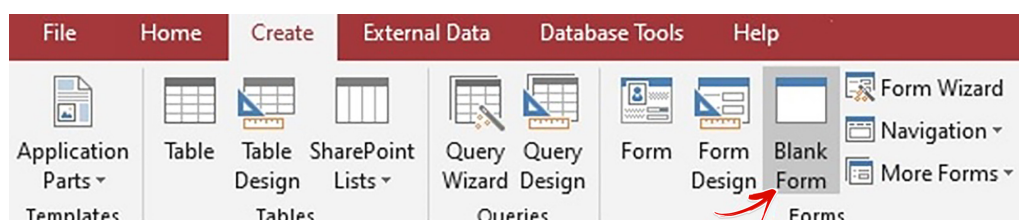


Figure 4.11: Creating a blank form in layout view.

Creating a form this way requires some experience in form design (form editing options will be explained in the following chapters), so it is advisable to start creating initial forms using a wizard (Chapter 4.4).

4.4 Creating forms using the Form Wizard

The "Form Wizard" is a useful tool suitable for familiarizing oneself with the form environment, the possibilities of various object layouts used in forms, and the ways to edit them. The Form Wizard allows for the creation of simple forms, with field selection possible from a single source (table or query) as well as multiple database sources. In a few intuitive steps, the form creator is guided through a process where they decide which fields they want and how they want them arranged in the form. The **Form Wizard** is part of the **Forms** menu on tab's **Create** menu (Figure 4.12).

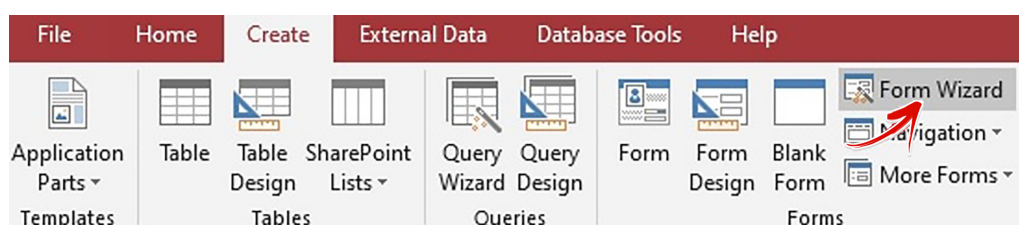


Figure 4.12: Form Wizard menu on the Create tab.

Basic form field layouts selected from a single database source (such as a single database table) using the form wizard include columnar, tabular, datasheet, and justified layouts.

4.4.1 Columnar layout of form fields

The sequence of steps for creating forms with different field layouts will be identical. First, on the **Create** tab in the **Forms** menu, use the **Form Wizard** option, and then the first dialog box will appear as shown in Figure 4.13.

Figure 4.13: Selecting fields from database tables and queries.



Note

For clarity and to test the functionality of the forms and their editing possibilities, we recommend creating forms with different field layouts based on the same selection of the source table. This will not only allow us to see how each form looks, but later, for example, when creating more forms, we will be able to more easily decide which type suits us better.

In the first dialog box of the wizard (Figure 4.13), we select a table (in our case, we first use the "Personal Data" table) and from the "Available fields" section, we use a button **>>** to move all fields from the "Personal Data" table to the "Selected fields" section. Fields can be selected (or removed) individually by double-clicking the left mouse button on the field name or by selecting the desired field and using a button **>**. We confirm the selection of fields with the **Next** button and in the next step of the wizard, we set the preferred type of form layout. We have four above-mentioned layout types available, from which we select the first option (Columnar) and confirm it with the **Next** button (Figure 4.14).

In the final step of the wizard, we define the name of the form. The default name offered is the name of the table from which fields the form is created. In our case, we choose the name "Personal Data columnar layout" to make it easier to identify the tested forms, and

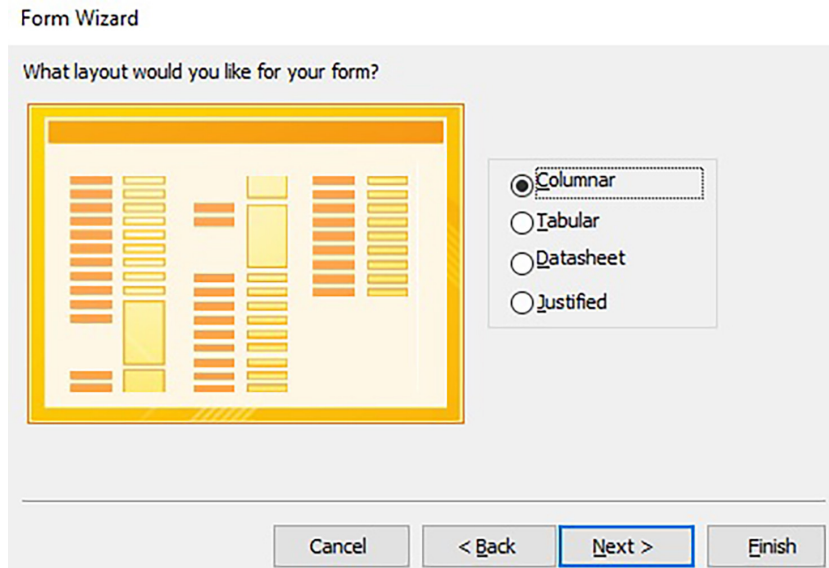


Figure 4.14: Selecting the field layout in the form.

we select the option "Open the form to view or enter information", i.e. form view of this new database object. We confirm the form name and the selection of the form view with the **Finish** button (Figure 4.15).

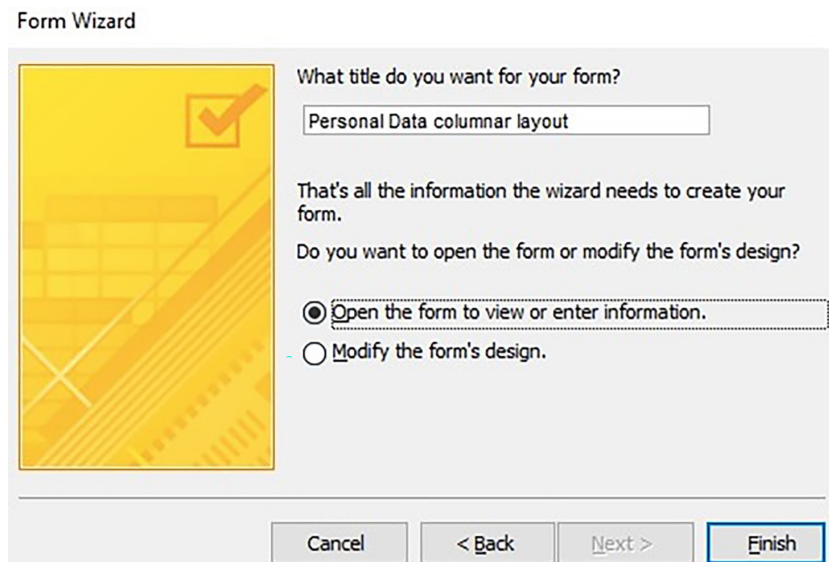


Figure 4.15: Completing the form creation using the wizard.

A newly created form "Patients Data columnar layout" will appear on the MS Access workplace, which will be displayed in the form view (Figure 4.16). The fields that were selected for the form are arranged one below the other in a single column. On the left side, the names of the fields are listed one below the other, and next to them, the corresponding fields in which data of one (usually the first) of the registered patients is loaded.

Personal Data columnar layout

Personal Data columnar layout

ID

Name

Surname

PIN

Gender

Date of Birth

Address

City

ZIP code

Health insurance company

Telephone number

Figure 4.16: Form view of the columnar layout of the newly created form.

4.4.2 Tabular layout of form fields

To create a form from the data of a one table with a tabular layout, we repeat the process of creating a new database object using the form wizard. On the **Create** tab in the **Forms** menu, use the **Form Wizard** option (Figure 4.12). As the data source, again select the "Personal Data" table and move all the fields from the available fields to the selected fields (Figure 4.13). In the second step of the wizard, set "Tabular" field layout and confirm with the **Next** button (Figure 4.14). In the third step of the form wizard, enter the form name, for example, "Personal Data tabular layout", set the option "Open the form to view or enter information", and confirm its creation with the **Finish** button (Figure 4.15). A form with a tabular layout of the "Personal Data" table fields is generated, as shown in Figure 4.17.

Personal Data tabular layout

Personal Data tabular layout

ID	Name	Surname	PIN	Gender	Date of Birth	Address	City	ZIP code	Health insurance cc
1	Peter	Brown	111111/1111	male	2/15/1985	Main street 17	Bratislava	820 12	Union
2	Elisabeth	Smith	222222/2222	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali
3	Maria	Sanchez	333333/3333	female	6/14/1968	Tr. SNP 23	Kosice	040 01	Union
4	Olivia	Johnson	444444/4444	female	11/11/1982	Main street 25	Presov	080 01	Generali
5	Mario	Johnson	555555/5555	male	12/11/1984	Main street 25	Presov	080 01	Union
6	Martin	King	666666/6666	male	12/12/1963	Tr. SNP 57	Kosice	040 01	Union
7	Sarah	Newton	777777/7777	female	12/1/2000	University street 17	Kosice	040 01	Union

Figure 4.17: Form view of the tabular layout of the newly created form.

The database fields are arranged side by side in the tabular view of the form, the field labels are placed in the form header, and multiple records (not just the record of one patient) are listed one below the other.

4.4.3 Datasheet layout of form fields

We will repeat the procedure using the form wizard as with the previous two layouts. On the **Create** tab in the **Forms** menu, we use the **Form Wizard** option (Figure 4.12). As the data source, we again select the "Personal Data" table and move all fields from the available fields into the selected fields (Figure 4.13). In the second step of the wizard, we set the layout type to "Datasheet" and confirm with the **Next** button (Figure 4.14). In the third step, we enter a form name, for example, "Personal Data datasheet", set the option "Open the form to view or enter information" and confirm with the **Finish** button (Figure 4.15). A form with a datasheet field layout will be generated as shown in Figure 4.18.

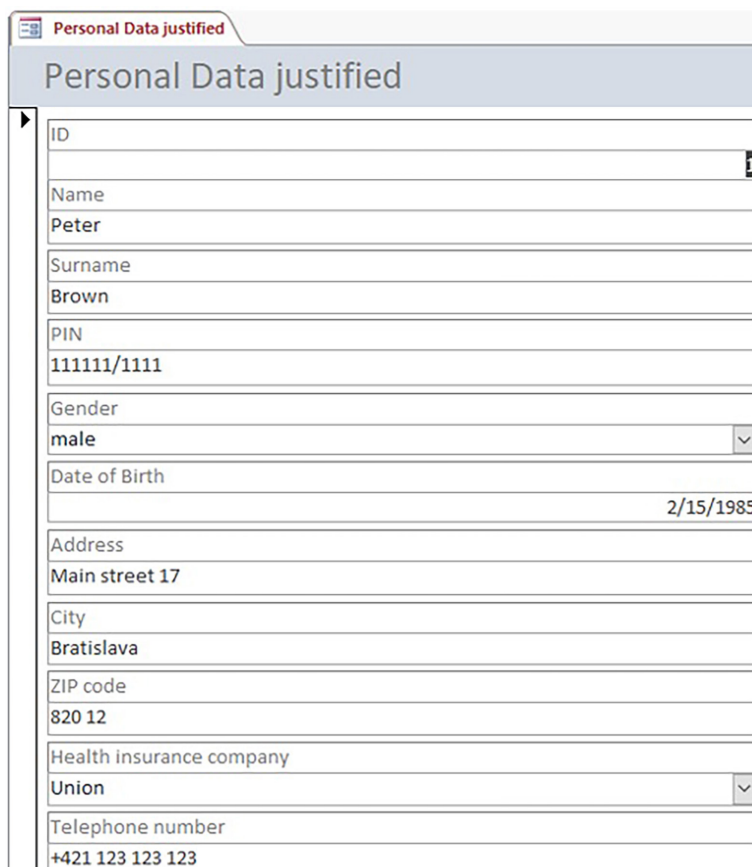
Personal Data datasheet										
ID	Name	Surname	PIN	Gender	Date of Birth	Address	City	ZIP code	Health insurance	Telephone number
1	Peter	Brown	111111/1111	male	2/15/1985	Main street 17	Bratislava	820 12	Union	+421 123 123 123
2	Elisabeth	Smith	222222/2222	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali	+421 591 591 595
3	Maria	Sanchez	333333/3333	female	6/14/1968	Tr. SNP 23	Kosice	040 01	Union	+421 122 331 122
4	Olivia	Johnson	444444/4444	female	11/11/1982	Main street 25	Presov	080 01	Generali	+421 471 451 592
5	Mario	Johnson	555555/5555	male	12/11/1984	Main street 25	Presov	080 01	Union	+421 462 531 254
6	Martin	King	666666/6666	male	12/12/1963	Tr. SNP 57	Kosice	040 01	Union	+421 236 985 474
7	Sarah	Newton	777777/7777	female	12/1/2000	University street 17	Kosice	040 01	Union	+421 111 222 333
8	Mariah	Black	888888/8888	female	1/1/1982	University street 35	Kosice	040 01	self-payer	+421 222 255 558
9	Michael	Omar	999999/9999	male	4/11/1969	Tr. SNP 5	Presov	080 01	self-payer	+421 111 122 211
10	Dylan	Mckenzie	101010/1010	male	12/3/1999	Tovern street 6	Bratislava	820 11	Union	+421 556 622 332
11	Barbara	King	121212/1212	female	12/23/2001	Main street 95	Bratislava	820 11	Generali	+421 626 262 626
* (New)										

Figure 4.18: Form view of the datasheet layout of the newly created form.

A datasheet type form is suitable in cases where it is necessary to display or work with multiple records simultaneously (it does not have to contain all fields of the source table), for example as a subform that displays all related records.

4.4.4 Justified layout of form fields

Justified layout of the form aims to utilize the entire form area so that the fields are displayed as clearly as possible, with the size of the generated field for reading data from the source table determined by the data type and properties defined for each field in the given table. To generate a form with justified fields, we repeat the aforementioned procedure. On the **Create** tab in the **Forms** menu, we use the **Form Wizard** option (Figure 4.12). As the data source, we select the "Personal Data" table and move all available fields to the selected fields (Figure 4.13). In the second step of the wizard, we set the layout type to "Justified" and confirm with the **Next** button (Figure 4.14). In the third step, we enter a form name, for example, "Personal Data justified", set the option "Open the form to view or enter information" and confirm with the **Finish** button (Figure 4.15). A form with justified fields (arranged by edges) is generated, as shown in Figure 4.19.



Personal Data justified	
ID	
Name	Peter
Surname	Brown
PIN	111111/1111
Gender	male
Date of Birth	2/15/1985
Address	Main street 17
City	Bratislava
ZIP code	820 12
Health insurance company	Union
Telephone number	+421 123 123 123

Figure 4.19: Form view of the justified layout of the newly created form.



Note

When creating forms and selecting the fields that the form should contain, it is important to keep in mind that the form will also be used for adding new records. Therefore, if not all fields of the given table are selected, the required ones should certainly be included. Otherwise, the form will require the entry of mandatory data to save a new record, but the user will not have a field available in which to enter it.

4.5 Form Design View

Similar to the design view for tables, the Form Design View allows us to define the properties, operation, and appearance of all objects placed in forms. Auto-generated forms or forms created using a wizard can be edited to meet our expectations and requirements concerning the entire database's functionality. We will demonstrate the modifications that can be made in the Form Design View using one of the forms created for the "Personal Data" table data.

For example, we open the "Personal Data columnar layout" form in Form Design View by selecting **Design View** in the **View** option of the **Home** tab (see, for example, Figure 4.2), and the form we are about to edit will appear in its Design View. In the Form Design View, the "Form Design Tools" panel activates, offering three menu tabs: **Design**, **Arrange**, and

Format (Figure 4.20), where you can find intuitive menus for adding and editing the form and its objects.

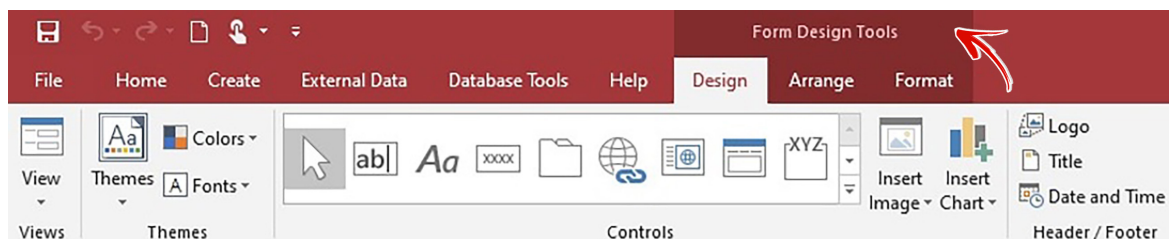


Figure 4.20: Form Design Tools.

To simplify working with the form design, it is possible to activate the grid in the background of its workspace. This can assist with resizing block fields, aligning and moving fields, etc. To activate or deactivate it, right-click on the form's workspace and select the grid option (Figure 4.21).

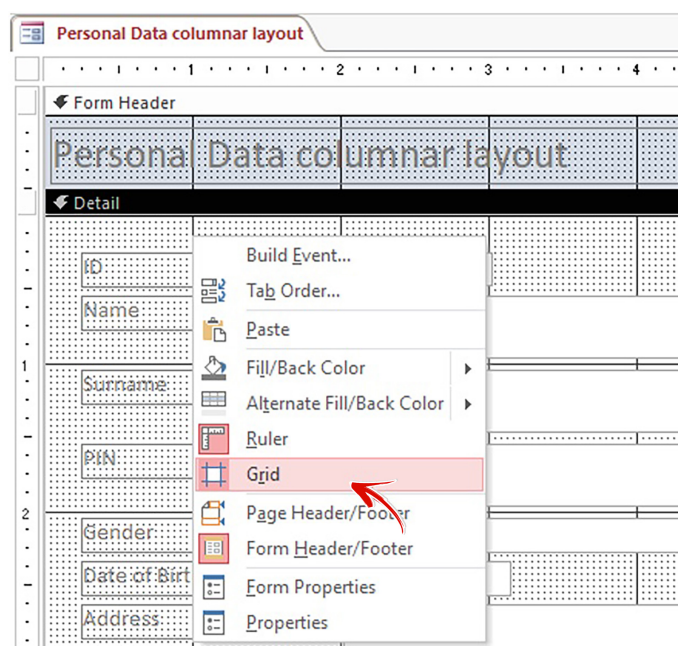


Figure 4.21: Grid activation/deactivation in the Form Design View.

The form in Design View is divided into the form header, detail (body of the form), and form footer. The size of these form sections can be altered by changing the position of the bars that separate them (Detail bar, Footer bar).



Note

Larger forms may also contain a page header and page footer, i.e. information that can repeat as the form's content scrolls, depending on the displayed data and the number of pages the form contains. The form header and form footer are displayed only once, at the beginning and end of the form, respectively.

4.5.1 Form header design

Form header created using the form wizard contains only one object, which is a label with the form's name (Figure 4.22). Changing the name used in the form header "Personal Data columnar layout" is very simple. Click on the label with the form name with the left mouse button, which will select it and highlight it in orange. Clicking the left mouse button again will enter the label text edit mode, allowing you to change or rewrite the text, for example, to Patient Records, Patient's registration form or a similar name that reflects the primary purpose for which the form is intended. This is only a change of the label name in the form header, not a change of the form's entire name.

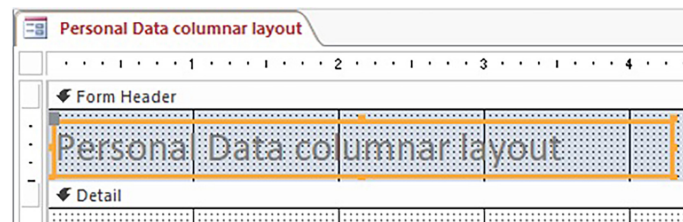


Figure 4.22: Label in Form Header.



Note

If we want to rename the entire form, close the form object and right-click on it in the navigation pane. From the local menu select the **Rename** option and enter the desired form name (Figure 4.23).

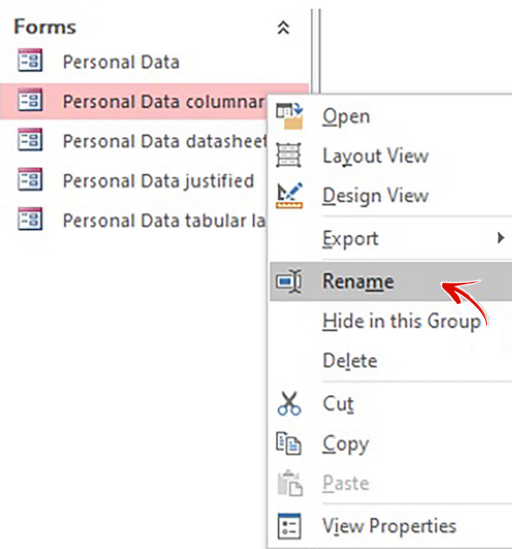





Figure 4.23: Rename the form.

Font formatting is identical to what we know from other office applications. We can format the form name in the header of the form in design view through "Form Design Tools", where we open the **Format** tab and gradually change the font type, color, or size

according to our requirements and preferences. The text in the label can be centered  and the selected label field can be placed in the center of the form header.

To change the background of the form header, left-click on the background area of the header (we always edit the object that is currently selected, i.e. highlighted in orange frame) and use the  option located on the **Format** tab to define the preferred color. This procedure can be used to define the fill color of any form object.

In the form header, it is also advisable to include additional information (unbound), such as the date and time, clinic logo, address, etc. The date and time of opening the form can be set through **Form Design Tools**, the **Design** tab, the **Header/Footer** category, using **Date and Time**  option (Figure 4.24). The position of the fields in the form header with this information can be adjusted according to personal preference by simply selecting the object with the mouse and moving it to the desired place.

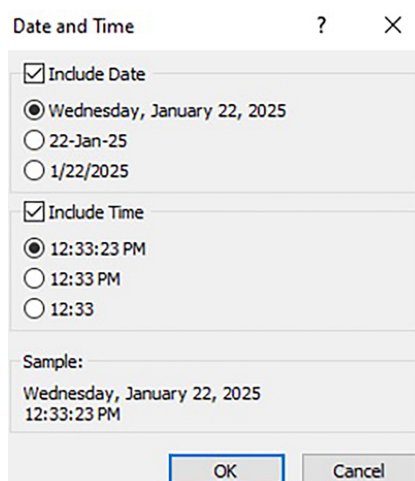





Figure 4.24: Dialog box for inserting Date and Time.

The clinic logo (saved as a separate file - an image stored on the hard drive) can also be inserted into the form header through **Form Design Tools**, the **Design** tab, the **Header/Footer** category, and selecting Logo  option. You can also add the institution name, clinic name, or doctor's name to the logo using the "Text Box" control . The form header design is saved when switching the form view, or continuously using the **Save** button . An example of a modified form header is shown in Figure 4.25.

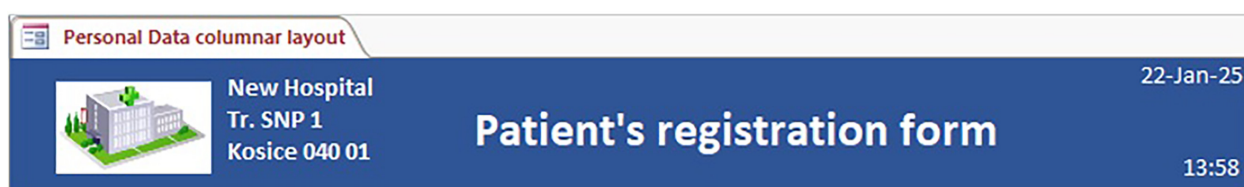


Figure 4.25: Example of a modified form header in form view.

4.5.2 Form details design

The details area (or the body of the form) usually represents the largest space on the form, which users of the database work with, because it is used to display most of the processed information. Automatically generated fields rarely have dimensions that are optimal for the specific type of displayed data. Some fields are generated to be too large, so their space will never be fully utilized (for example, `ZIP code` will always display only 5 characters, but takes up the width of the entire form), or some fields are very small, and to display the entire information, it is necessary to scroll or navigate through their contents.

By adjusting the size of the respective fields (depending on the chosen form layout type), we can achieve, for example, uniformity of width or height of the fields, their alignment from the top or from the left edge of the form, etc. By changing the dimensions of the objects on the form, it is also possible to gain space for additional fields and objects that we want to add to the form.

If we want to change the width of any of the fields/objects, we proceed by first selecting the field/object with the mouse (by clicking on it with the left mouse button) and then clicking and holding the left mouse button over one of the squares located on the orange outline (left or right, Figure 4.26).

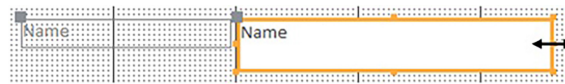


Figure 4.26: Changing the dimensions of objects in design view of forms.

Moving the mouse right or left changes the width of the field/object. We proceed identically in the case of changing the height. When changing the width of multiple fields at the same time (similarly when changing other properties), it is necessary to select multiple fields, for example, by pressing and holding the `SHIFT` key and left-clicking to select fields/objects whose properties we want to change in bulk. The marked objects are then enlarged or reduced as needed at once.

When moving fields around the form and aligning them, the presence of the displayed grid can be utilized. The field we want to move is highlighted with the mouse (or several fields simultaneously) and moved while holding the orange outline (Figure 4.27).

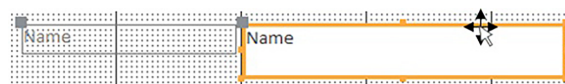





Figure 4.27: Moving objects in the design view of forms.

Each bound field, i.e. a field that represents data loaded from the created table, also has its label in the form. The label moves together with the field. However, if we want to move only the field or only the label of that field, we must grab and move the small gray square located in the top left corner of the field or its label.

Custom layout design of form objects in the details area might look, for example, like the one shown in Figure 4.28.

Detail	
ID	Date of Birth
Name	Address
Surname	City
PIN	ZIP code
Gender	Health insurance company
	Telephone number

Figure 4.28: Moving database fields of the form.

From the perspective of further editing and to highlight field descriptions, we recommend using appropriate font formatting. On the **Format** tab, use commands to change the type and size of the font , font color , or we can use commands for formatting control elements, such as shape fill  or shape outline .

All options for editing form objects can also be found in the property sheet, which can be turned on or off in the local menus of objects as well as on the **Design** tab of form design tools. We can also find special effects for database fields here. We select the field or fields for which we want to set a special effect and choose one of the offered options (Figure 4.29).

Property Sheet




Selection type: Text Box

Name

Format Data Event Other All

Format	
Decimal Places	Auto
Visible	Yes
Show Date Picker	For dates
Width	1.7708"
Height	0.4167"
Top	0.4583"
Left	2.0208"
Back Style	Normal
Back Color	Background 1
Border Style	Solid
Border Width	Hairline
Border Color	Background 1, Darker 35%
Special Effect	Flat
Scroll Bars	
Font Name	Raised
Font Size	Sunken
Text Align	Etched
Font Weight	Shadowed
Font Underline	Chiseled

Figure 4.29: Setting special effects for database fields.

On the **Design** tab, it is also possible to find and use controls that enliven and clarify the form design. To outline a group of fields, a rectangle  can be used. We draw a rectangle around the **Name**, **Surname**, **PIN** and **Gender** fields. First, we select **Rectangle** in the **Controls** category on the **Design** tab, then click on the spot in the form where the top left corner of the rectangle should be, hold down the left mouse button, and move towards the location where the bottom right corner of the rectangle should be. Release the left mouse button and adjust the position and dimensions of the rectangle if necessary. For setting additional properties of the rectangle, we can use shape fill , shape outline , and others from the **Format** tab. An overview of the controls on the **Design** tab is shown in Figure 4.30.

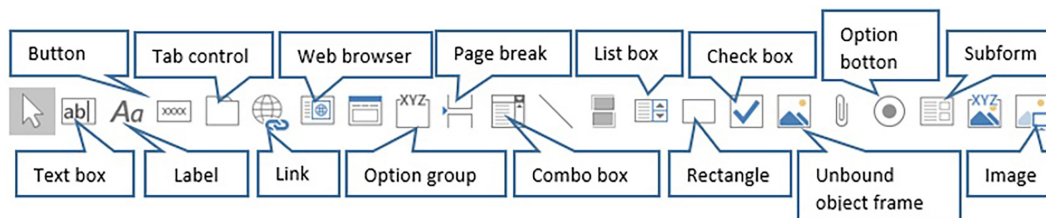


Figure 4.30: Overview of the controls on the Design tab.



Note


We design the form and form objects to be user-friendly, clear, and easy to use. Individual fields should also be color-matched (the same) in all forms so that it is clear that it is one database - one software product.

An example of a form design change is shown in Figure 4.31.

Figure 4.31: Change in the appearance of the header and detail section of the form.

The details section can also include information from other related database sources. We add them to the existing form as a subform.

4.5.3 Subform

A subform is added to a form in its design view. In our case, for patients, the subform will contain fields with additional information stored in one of the existing tables. We will add records of previous examinations of patients to the "Patient Registration" form and place them under personal information. We use the **Subform/Subreport** option , which is part of the controls on the **Design** tab, and in the "Details" section of the form, click where we want to place the subform. This also opens the "Subform Wizard" (Figure 4.32).

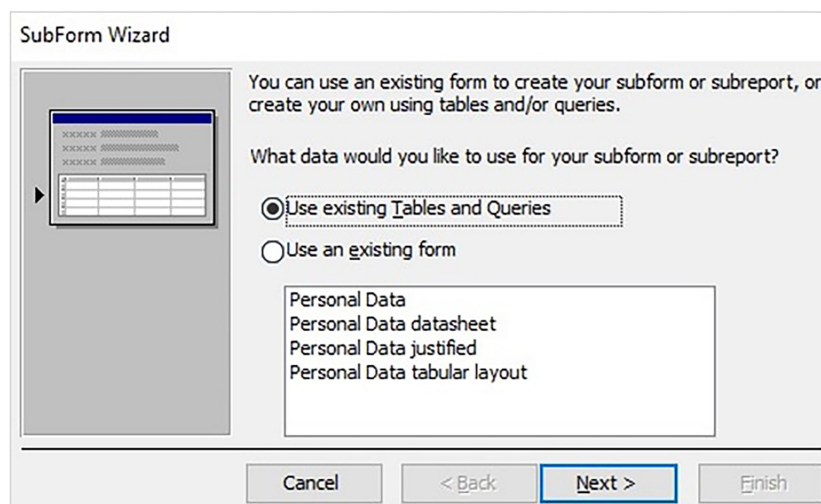


Figure 4.32: Subform Wizard.

In the subform wizard window, select "Use existing Tables and Queries" and click **Next** to proceed to the next step of the wizard (Figure 4.33).

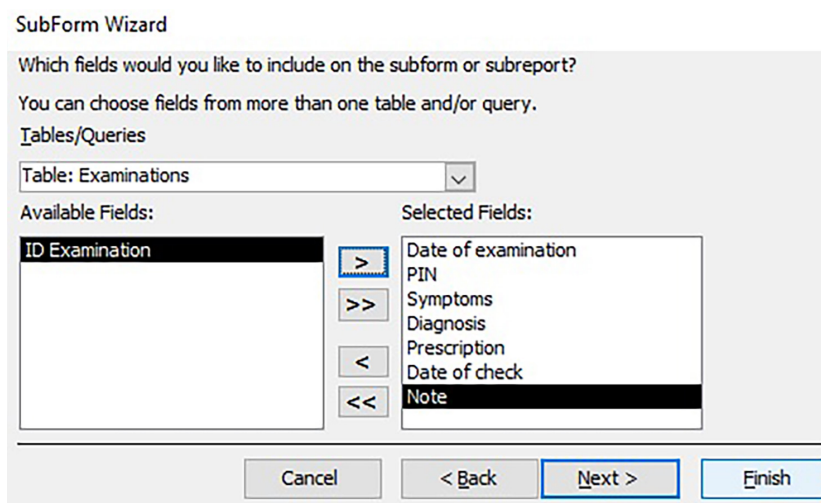


Figure 4.33: Selecting fields for the subform.

**Note**

We can also use one of the existing forms, which are usually already graphically and functionally adjusted, as a subform. In such a case, we use the second option of the subform wizard, which is "Use existing form" (Figure 4.32).

As a data source for the subform, we choose the "Examinations" table, move the required fields from the available fields to the selected fields group, and continue creating the subform by clicking **Next** button. In the next step of the subform wizard, we set up the link between the form and the subform (Figure 4.34).

SubForm Wizard

Would you like to define which fields link your main form to this subform yourself, or choose from the list below?

☐ Choose from a list. ☒ Define my own.

Form/report fields: Subform/subreport fields:

PIN PIN

Show Examinations for each record in Personal Data using PIN

Cancel < Back **Next >** Finish

Figure 4.34: Linking the form to the subform.

We select the option "Define my own" and in the "Form/report fields" section choose the field PIN (a database field from the "Personal Data" table, from which the "Registration form" is created). In the "Subform/subreport fields" section, from the available fields of the "Examinations" table, again choose the field PIN (a database field from the "Examinations" table, from which we are creating a subform). Confirm the link setting by clicking **Next** button and move to the window of the last step of the subform wizard (Figure 4.35), enter the name of the subform, for example, "Examinations", and finish creating the subform using the **Finish** button.

SubForm Wizard

What name would you like for your subform or subreport?

Examinations

Those are all the answers the wizard needs to create your subform or subreport.

Cancel < Back **Next >** **Finish**

Figure 4.35: Entering the subform's name.

In the navigation pane under the category **Forms**, we have a new object – a subform named "Examinations". Place the created subform "Examinations" in the form's design view to a suitable position, adjust the field widths so that the fields of the subform are readable in form view (Figure 4.36).

New Hospital
Tr. SNP 1
Kosice 040 01

Patient's registration form

06-Feb-25
12:16

ID: 2

Name: Elisabeth

Surname: Smith

PIN: 222222/2222

Gender: female

Date of Birth: 3/13/1995

Permanent address

Address: Tr. SNP 56A

City: Kosice

ZIP code: 040 01

Health insurance company: Generali

Telephone number: +421 591 591 595

Examinations

Date of exami	PIN	Symptoms	Diagnosis	Prescription
1/16/2025	222222/2222	Patient feels tired, has difficulty breathing. Cough: 1 week, T:38.5°C	Flu	antibiotics
* 2/6/2025	222222/2222			

Record: 2 of 2


Figure 4.36: Form view of the form with the subform.



Video

Creating a Form

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

In the "Patient's registration form", we can work with patient records while simultaneously viewing information about their examinations and recording new ones. Records are always saved automatically to the source tables when moving to another record or continuously when the **Save** button  is used.





Practical Task

Enter examination records for any five patients through the registration form created above, i.e. add five new examination records to the database using the "Examinations" subform.

4.5.4 Form footer design

In most forms, the footer is not utilized, but useful information can be displayed here, such as the number of records, the order of the current record, or you can add custom control buttons. In our sample database, we will use the footer space to insert buttons that allow us to navigate between records. These will enable us to view records using our defined controls (there will be no need to use the standard record navigation bar).

4.5.4.1 Go to next record button

We will create a custom button using the **Button** control  found in the controls on the **Design** tab. In the design view of the "Patient's registration form", click on the boundary of the form footer section and increase its height to the desired level. Then select the **Button** control  from the **Design** tab and click on the location in the form footer where the button should be placed (buttons can also be added to details or header sections). On the workspace, the "Command Button Wizard" dialog box will appear (Figure 4.37).

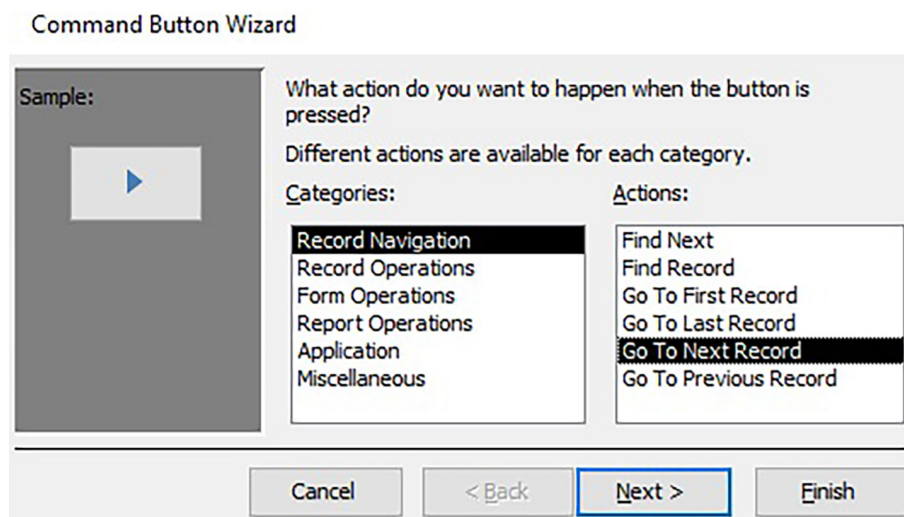


Figure 4.37: Command Button Wizard.

On the left side, select the "Record Navigation" category and "Go To Next Record" from the actions displayed on the right side. Continue with the generation of the new control button by clicking **Next** button.

In the next step of the button wizard (Figure 4.38) define what should be displayed on the button, i.e. text or an picture. If you prefer buttons with text, use text such as "Next" or "Next Record". However, illustrative buttons with pictures are more commonly used. It is possible to use standard pictures or custom ones stored on the computer's hard drive. After choosing how to display the button, confirm its creation with the **Finish** button.

In the design view of the form footer, a button will be added (Figure 4.39), whose position and design we can modify if necessary (e.g. using commands from the **Format** tab).

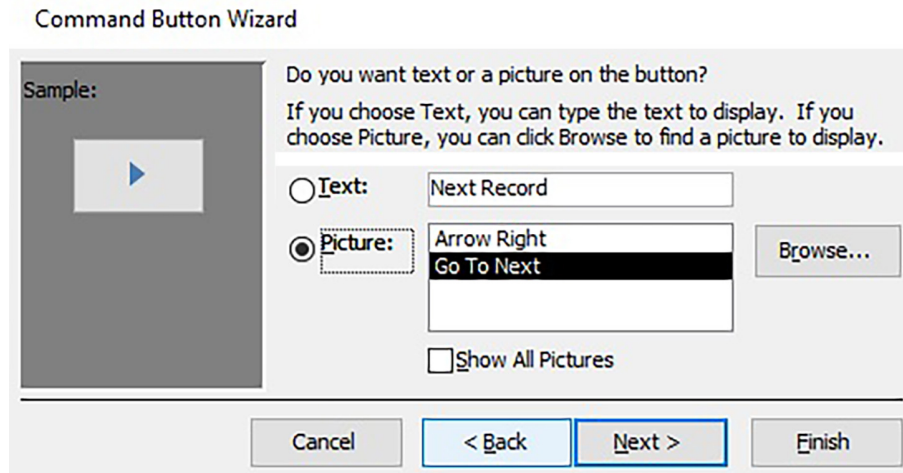



Figure 4.38: Button description setting "Go To Next Record".



Figure 4.39: Button "Go To Next Record".

4.5.4.2 Go to previous record button

To create a button that allows navigation to the previous record, proceed similarly to the previous Chapter 4.5.4.1. In the **Design** tab between **Controls**, use the **Button** control  and click in the form footer where the button should be placed (Figure 4.40).

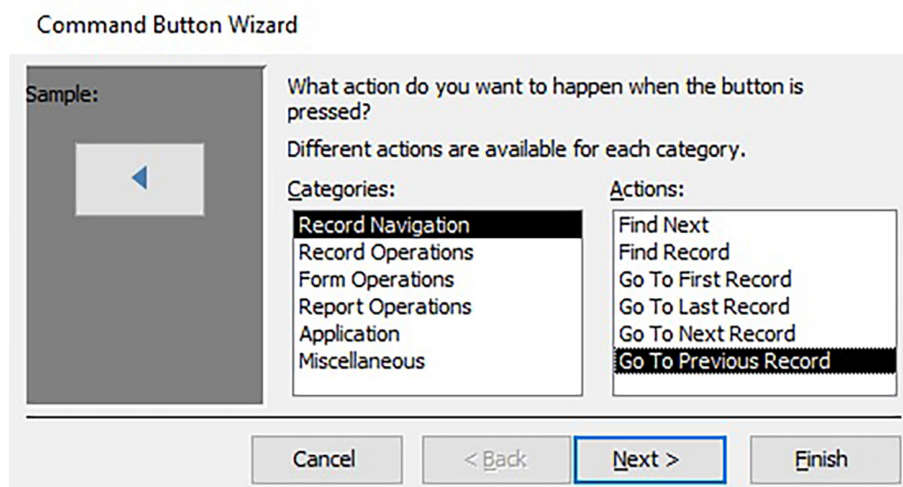


Figure 4.40: Command Button Wizard "Go To Previous Record".

In the Command Button Wizard window, select the category "Record Navigation" and the action "Go To Previous Record", confirm with the **Next** button and move to the next step, where again define the text or picture used on the created button (Figure 4.41).

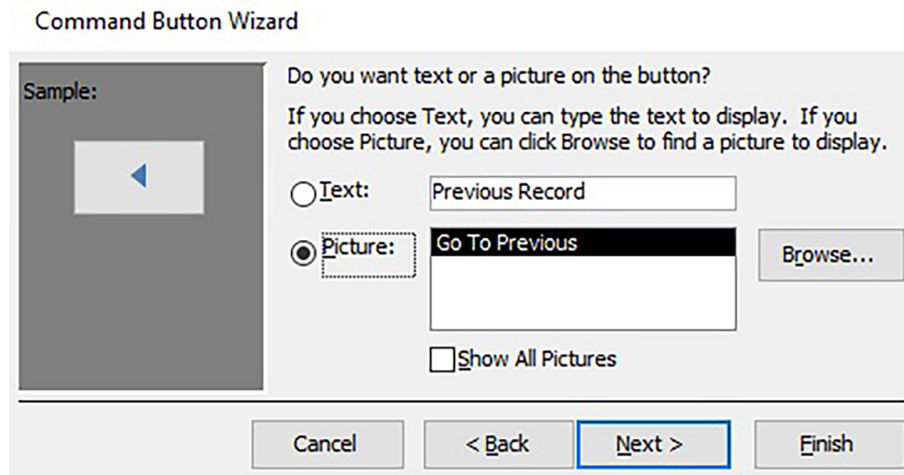



Figure 4.41: Button description setting "Go To Previous Record".

**Practical Task**

Create and add buttons "Go To First Record" and "Go To Last Record" to the Patient's registration form.

4.5.4.3 Add new record button

In cases where we want to record new patients and to avoid navigating through all records, we use the "New Record" button, which takes us to the end of the patient list. To create it, use the **Design** tab again, the **Controls** menu, and the **Button** control . In the footer section of the form, click on the location where the button should be placed, and then in the command button wizard, select the category "Record Operations" and the action "Add New Record" (Figure 4.42).

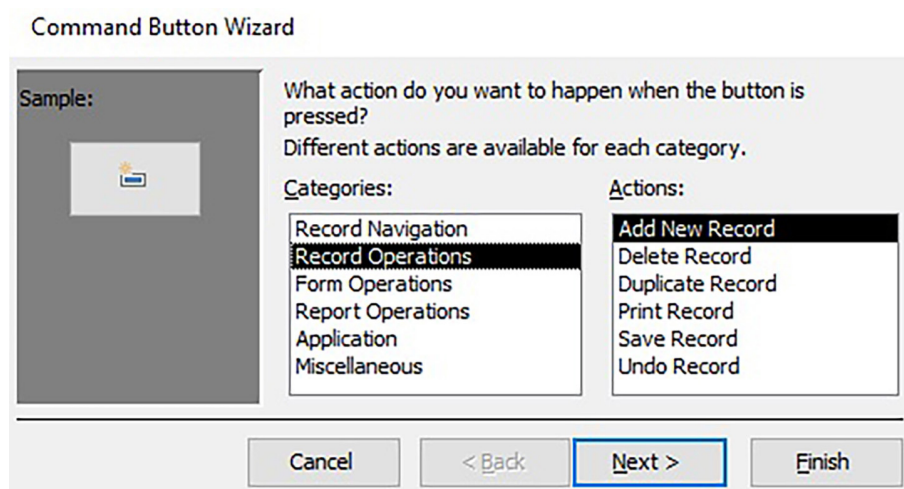


Figure 4.42: Command Button Wizard "Add New Record".

In the next step, set the button label, such as "Registration," and click **Finish** button (Figure 4.43).

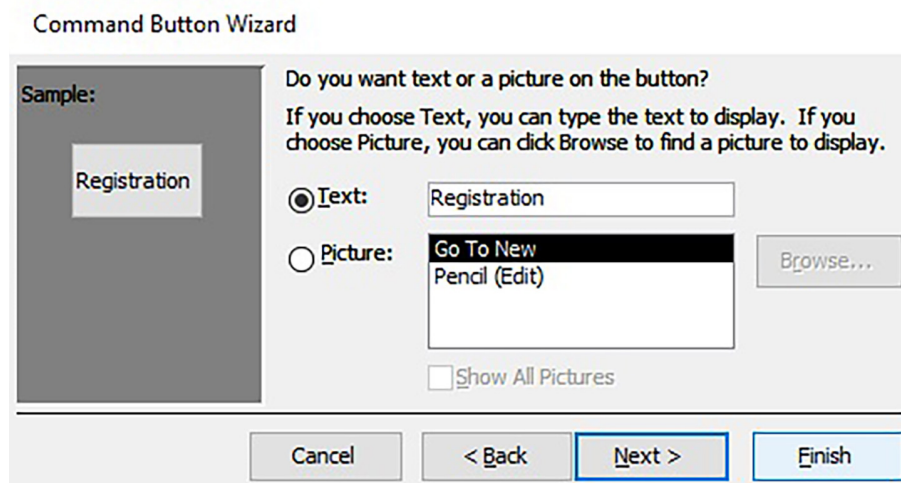


Figure 4.43: Button description setting "Registration".

**Video**

Add New Record Button

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

If we want to have other buttons on the form, like save record, print record, and others, we proceed identically according to the previous steps. To verify the functionality of the created buttons, open the form in form view and check if the form responds correctly when using them.

**Practical Task**

Using the "New Record - Registration" button on the "Patient's registration form" register five new patients to the database.

4.6 Advanced forms

Forms containing fields from multiple related sources – advanced forms, can also be created using the form wizard in a manner similar to what was explained in Chapter 4.4. In this chapter, we will explain how to use the form wizard to create a form with a subform and linked forms.

4.6.1 Form with subform

For illustration, we will create a form similar to that from previous chapters, where we will be able to record personal data of patients and their respective examinations. In the **Create** tab in the **Forms** menu, select the **Form Wizard** option.

In the first window of the wizard, choose the "Personal Data" table and move all the fields (or only some of them, if not all are needed) from the "Available fields" section to the "Selected fields" section and continue by changing the database table to the "Examinations"

table (Figure 4.44) and move all fields of this table (or only some that are needed on the subform) from the "Available fields" to the "Selected fields" section, i.e. the "Selected fields" contains database fields from two related tables. Continue to the next step of form wizard by clicking the **Next** button.

Form Wizard

Which fields do you want on your form?
You can choose from more than one table or query.

Tables/Queries
Table: Examinations

Available Fields:

- ID Examination
- Date of examination
- PIN
- Symptoms
- Diagnosis
- Prescription
- Date of check
- Note

Selected Fields:

- PIN
- Gender
- Date of Birth
- Address
- City
- ZIP code
- Health insurance company
- Telephone number

Buttons: Cancel, < Back, **Next >**, Finish

Figure 4.44: Selecting fields for a form with a subform from two database tables.

In the next step (Figure 4.45), to view the data, select the option by database table "Personal Data" (we will see the patient and all his examinations, otherwise we would see one examination and who it belongs to, but we would not know how many examinations the patient has) and select option "Form with subform(s)".

Form Wizard

How do you want to view your data?

by Personal Data
by Examinations

ID, Name, Surname, Personal Data_PIN, Gender, Date of Birth, Address, City, ZIP code, Health insurance company, Telephone number

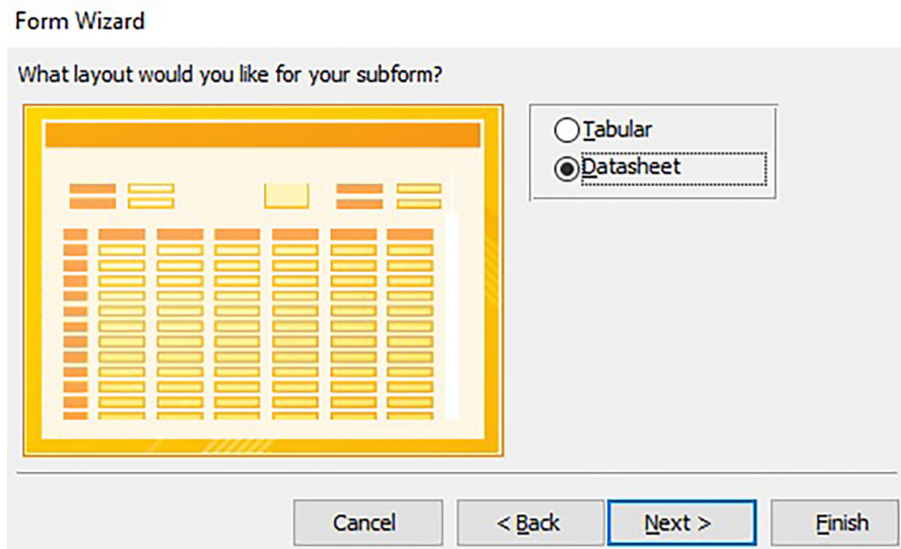
ID Examination, Date of examination, Examinations_PIN, Symptoms, Diagnosis, Prescription, Date of check, Note

☒ Form with subform(s) ☐ Linked forms

Buttons: Cancel, < Back, **Next >**, Finish

Figure 4.45: Setting up the view of the advanced form with a subform.

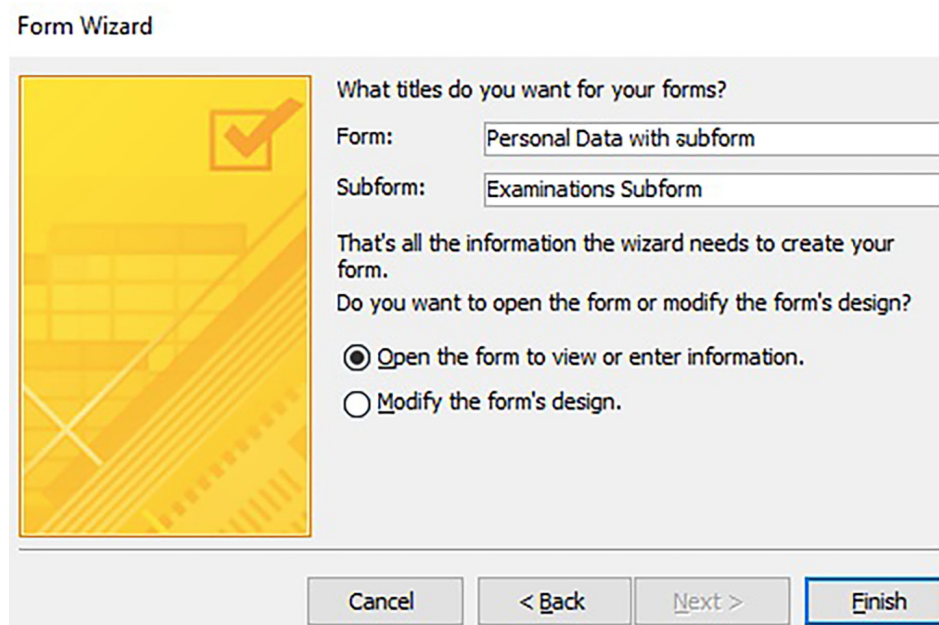
Continue with the **Next** button to the next step of the wizard, in which choose the table layout of fields in the subform (Figure 4.46) (it is also possible to choose the standard datasheet view) and continue to the next step.



The 'Form Wizard' dialog box is titled 'What layout would you like for your subform?'. It features a large preview window on the left showing a grid of 10 columns and 10 rows of yellow rectangular fields. To the right of the preview, there are two radio button options: 'Tabular' (unselected) and 'Datasheet' (selected). At the bottom of the dialog, there are four buttons: 'Cancel', '< Back', 'Next >' (highlighted with a blue border), and 'Finish'.

Figure 4.46: Setting the layout of fields in the subform of the advanced form with a subform.

In the last step of the form wizard (Figure 4.47) assign a name to the form, e.g. "Personal Data with subform" and a name for the subform, e.g. "Examinations subform". If you want to enter new records into the form after creating the object, choose the "Open form to view or enter information" option. In case you want to edit the design of the form and subform before displaying it, then check the "Modify the form's design" option and confirm with the **Finish** button.



The 'Form Wizard' dialog box is titled 'What titles do you want for your forms?'. It has a large yellow graphic on the left with a checkmark icon. On the right, there are two text input fields: 'Form:' with the value 'Personal Data with subform' and 'Subform:' with the value 'Examinations Subform'. Below these fields, a message states: 'That's all the information the wizard needs to create your form. Do you want to open the form or modify the form's design?'. There are two radio button options: 'Open the form to view or enter information.' (selected) and 'Modify the form's design.' (unselected). At the bottom, there are four buttons: 'Cancel', '< Back', 'Next >', and 'Finish' (highlighted with a blue border).

Figure 4.47: Specifying form and subform names.

The generated form with a subform using the form wizard opened in form view is illustrated in Figure 4.48.

Figure 4.48: Form view of the Personal Data form with the Examinations subform.



Video

Form with subform

(See Portal UPJŠ LF, <https://portal1.lf.upjs.sk/clanky.php?aid=57>)

4.6.2 Linked forms

The second type of advanced forms is linked forms. When creating them using the wizard, proceed in the same way as with a form with a subform. First, on the **Create** tab in the **Forms** menu select the **Form Wizard** option and in its first step, move the desired fields from the "Personal Data" database table and the "Examinations" database table to the "Selected Fields" section (like in Figure 4.44).

In the second step of the wizard (Figure 4.49), again display the selected fields according to the "Personal Data" database table, i.e. the main form will be the form created from the fields of the "Personal Data" database table. In the lower right part of the wizard window, mark the "Linked Forms" option, i.e. a link to the form with examination data will be created on the main form.

By clicking the **Next** button, proceed to the next step (like in Figure 4.47), where the name of the form is entered, for example, "Personal Data Linked form", and the name of the subform, for example, "Examinations". Then choose the option "Open the main form to view and enter information" to see how the new linked forms will look, and click the **Finish** button.

The "Personal Data Linked form" (main form) opens in its form view and will allow working with the records of individual patient identification data. In the header of the

Form Wizard

How do you want to view your data?

by Personal Data
by Examinations

ID, Name, Surname, Personal Data_PIN, Gender, Date of Birth, Address, City, ZIP code, Health insurance company, Telephone number

ID Examination, Date of examination, Examinations_PIN, Symptoms, Diagnosis, Prescription, Date of check, Note

☐ Form with subform(s) ☒ **Linked forms**

Cancel < Back **Next >** Finish

Figure 4.49: Setting up the view of the advanced form as linked forms.

"Personal Data Linked form" find a button named "Examinations", which allows to open the linked form with examinations for the currently viewed/edited patient record (Figure 4.50).

Personal Data Linked form

Examinations

ID

Name Peter

Figure 4.50: Form view of part of the linked forms and button overlay by form label.

Since the "Examinations" button is covered by the form label (Personal Data Linked form), it appears to be non-functional. This issue of the automatically generated button for opening/closing the linked form can be resolved by moving the "Personal Data Linked form" label away from the "Examinations" button (in design view), and saving the change. After switching the form back to form view, the "Examinations" button will be active and will link us to the patient's examination data form when used (Figure 4.51).

Examinations

ation f examination PIN	Symptoms	Diagnosis	Prescription
1/16/2025 222222/2222	Patient feels tired, has difficulty breathing. Coughed:	Flu	antibiotics
2/6/2025			

Figure 4.51: Generated Examinations form in form view.

**Note**

The **Examinations** button in the header of the "Personal Data Linked form" also serves to close the linked "Examinations" form, i.e. if the "Examinations" form is open and we press the **Examinations** button, the "Examinations" form will close.

The width of some fields in the examinations form is insufficient, and the data is not readable, for example, the Date of Examination. However, we can edit the form design in its layout or design view.

**Video**

Linked Forms

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

**Practical Task**

In the design view of the "Examinations" form, adjust the fields so that the data is readable.

4.7 Navigation form

A navigation form is also a useful way to group multiple forms (or other database objects), such as those frequently used by users. It contains navigation controls, i.e. groups of tabs or buttons, allowing convenient work with the database and its resources. The advantage of a navigation form includes, among others, saving space in the database's workspace tabs in MS Access.

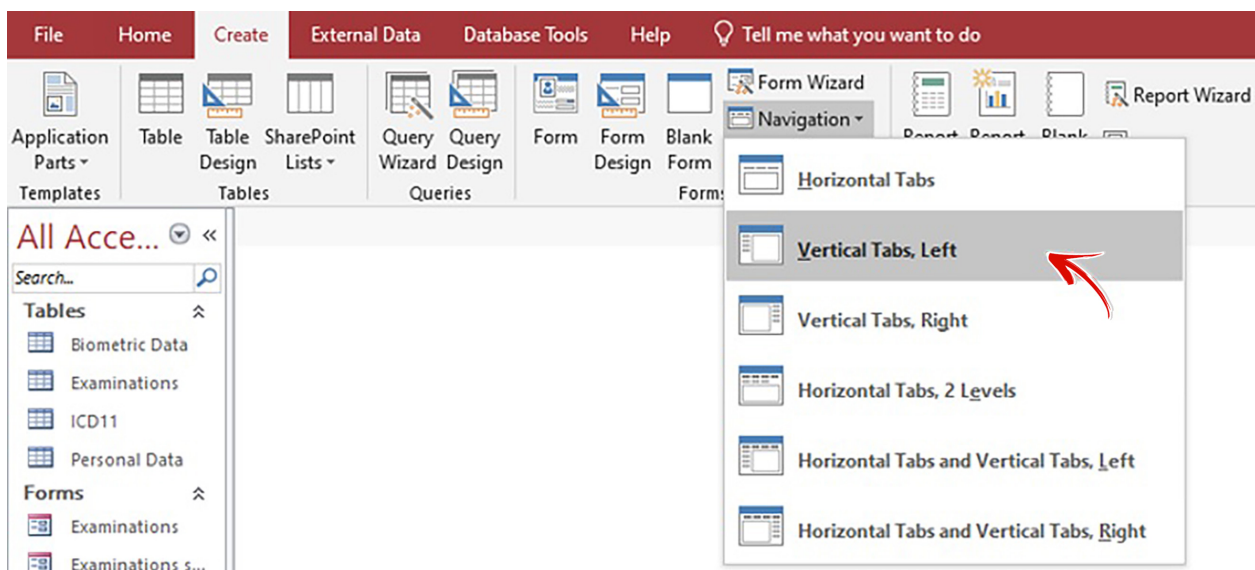


Figure 4.52: Creating a navigation form.

Users often have multiple database objects open in their workflow. In our case, these could be, for example, a registration form, a list of available medications, a prescription

form, etc. These are database objects that doctors most often need and use in their daily work. Under one tab in a navigation form, we can have multiple objects stored (depending on the complexity of the database design and user interface).

To create a navigation form, use the **Create** tab, select **Forms** category, and then the **Navigation** option. Several design options for the navigation form are available, such as "Vertical Tabs, Left" (Figure 4.52).

Once we use one of the types of navigation forms, its layout is generated, and we progressively drag and drop the objects we want to use in the navigation form from the navigation pane into the "Add New" panel. In our sample navigation form, we add the most commonly used forms, thus obtaining one form that includes other previously adjusted forms.

The screenshot displays a 'Patient's registration form' with a navigation pane on the left. The navigation pane shows 'Registration form' as the active tab, with 'Examinations' listed below it. The main form area is divided into several sections. The top section contains a header with 'New Hospital Tr. SNP 1 Kosice 040 01' and a date '04-Feb-25'. Below this, the form is organized into fields for patient information: ID (10), Name (Dylan), Surname (Mckenzie), PIN (101010/1010), Gender (male), Date of Birth (12/3/1999), Permanent address (Address: Tovern street 6, City: Bratislava, ZIP code: 820 11), Health insurance company (Union), and Telephone number (+421 556 622 332). The 'Examinations' section features a table with columns: Date of exami, PIN, Symptoms, Diagnosis, and Prescription. The table has one row with data: 2/4/2025, 101010/1010, and empty cells for Symptoms, Diagnosis, and Prescription. The form also includes a 'Registration' button and a 'Records' section showing '1 of 1' records.

Figure 4.53: Example of a navigation form.

In the navigation pane, select the "Patient's registration form" and drag it into the "Add New" section. Then, select another form and drag it into the "Add New" section again. Repeat this process until all the required forms are in the navigation form. Save the form, for example, under the name "Navigation Form". The form you created will be found among other forms in the navigation pane. Upon opening the navigation form in form view, you have a list of built-in forms on the left, which activate upon clicking their name. An example

of a navigation form with links to the "Patient's registration form" and "Examinations" forms is shown in 4.53.

**Video**

Navigation Form

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

**Practical Task**

Adjust the design of the navigation form to align with the design of individual forms, i.e. use the same font, colors, special effects, etc.

**Kontrolné otázky**

1. What are the main advantages of forms?
2. In what ways can a form be created?
3. Describe the different form views and state their differences.
4. What are the types of field layouts for creating simple forms?
5. What are the main parts that a form consists of?
6. In which form view is it possible to create form buttons?
7. What tabs are available in the form design tools and what are their purposes?
8. What types of advanced forms do we know?
9. How would you explain the term subform?
10. What are the advantages of navigation form?

Chapter 5

Data selection and sorting

Our sample database contains a large amount of structured data stored in several basic tables (created using the procedure described in previous chapters). It can be assumed that as the days and weeks of use increase, the volume of processed information will grow (in real applications and systems, this is one of their main purposes), and manual examination of records, or searching for specific information in tables, will become increasingly challenging. Fortunately, data stored in a structured form in database tables can be easily sorted by organizing and filtering them.

5.1 Sorting and filtering

A simple and quick way to find desired information together is to sort it, either alphabetically or numerically. If there are too many records and we are interested only in some of them, we can filter them according to our requirements and display only those we are interested in at any given moment. To find the required information, it is necessary to set appropriate criteria, which we will input as a filter applied to a specific field or group of fields. Finding relevant records can be realized using the **Sort & Filter** options available on the **Home** tab of the ribbon (Figure 5.1).

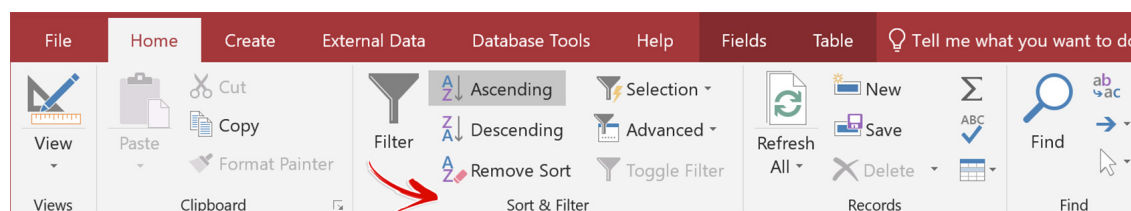


Figure 5.1: "Sort & Filter" menu.

The simplest way to get an overview of the records in a database table is to organize them according to one of the stored values. If we want to alphabetically sort patient records by surname, simply left-click on any patient's surname and confirm on the **Home** tab in the **Sort & Filter** menu the **Sort A to Z** function. All records will be displayed sorted by the patients' surnames. Similarly, sorting functions (ascending and descending) can be applied

to any fields in the database sources (tables and query data sheets). Sorting can be canceled using the **Remove Sort** button (function).

When using sorting functions, all records remain visible. On the other hand, filter functions select and display only those records from a group that meet the specified selection criteria. Other records are hidden, but they are still part of the database or its source table, meaning they are not deleted.

The filter function, like the sorting function, is applied to a database field in which we want to search and display the desired data/records. After setting the criteria for the given database field, the display of filtered records on the desktop is temporary (unless filter settings are saved), and the filtering result changes with each change of criteria applied to the respective database fields.

We can apply a filter function to a designated field directly on the **Home** tab; we only need to position the cursor in the field by which we want to filter the records. For example, put the cursor in any cell of the **Surname** field of the "Personal Data" database table, and activate the **Filter** button. A local menu for sorting and filtering records will appear next to the field with the patients' surnames, as shown in Figure 5.2.

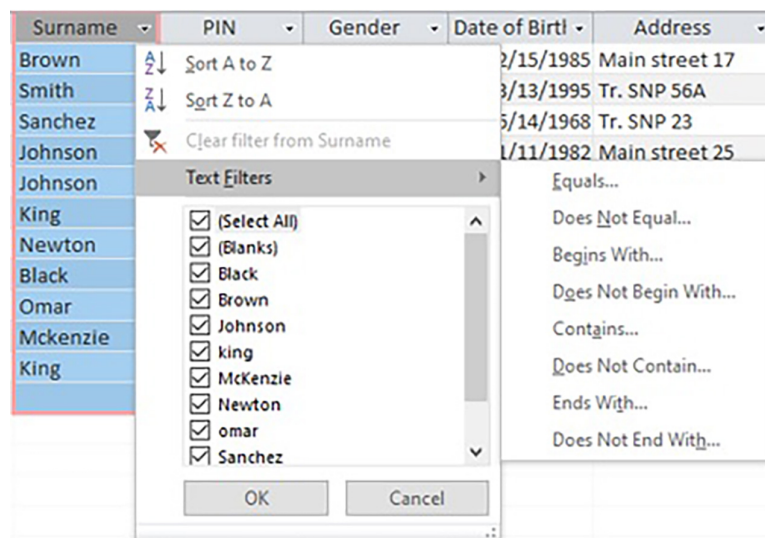


Figure 5.2: Filter for field Surname.

The filter list contains all the values present in the **Surname** field, allowing us to select the surname(s) we want to display and confirm by clicking **OK**. Depending on the data type of the specific field, additional filter options based on recorded values can be used. For the field **Surname**, these are text filters that allow us to search for text that equals the entered expression (character string), does not equal it, starts with it, does not start with it, contains it, does not contain it, ends with it, or does not end with it. For other fields with different data types, these may include numerical filters, date filters, etc.

Filters can be cleared in various ways, for instance, by confirming the **Toggle Filter** button on the **Home** tab (Figure 5.1), confirming the **Clear Filter from Surname** option (Figure 5.2), or selecting **Select All** from the **Surname** field's local menu (Figure 5.2).

The sorting and filtering functions are also part of the local menu obtained by right-clicking over the respective field. Sorting and filtering functions can be combined, meaning we can set different filters and sorting for multiple database fields simultaneously.



Practical Task

In the "Personal Data" database table, find all patients with a surname starting with the letter M who are insured by the Union health insurance company.

5.2 Record filtering types

Depending on the method of defining the filtering criteria and the scope of records that are of interest to the database user, it is possible to define several types of filters.

Filter by selection: represents a type of record filtering, which allows searching for data in a selected database field containing specific data. On the **Home** tab, in the **Sort & Filter** menu, you can use the **Select** option to filter records by selection, which includes the options: "Equals", "Does not equal", "Contains", and "Does not contain". Figure 5.3 illustrates the use of the filter by selection, where the selection criterion is the display of records of patients who are male. The field to which the filter has been applied also has a filter icon displayed in its header.

ID	Name	Surname	PIN	Gender	Date of Birth
1	Peter	Brown	111111/1111	male	15. 2. 19
5	Mario	Johnson	555555/5555	male	11. 12. 19
6	Martin	King	666666/6666	male	12. 12. 19
9	Michael	Omar	999999/9999	male	11. 4. 19
10	Dylan	Mckenzie	101010/1010	male	3. 12. 19

Figure 5.3: Filter by selection.

Filter out of selection: is based on the principle of finding and displaying a list of all records that do not meet the specified criteria, i.e. it is more efficient to define selection by "Show all patients outside Košice", rather than defining all the cities and towns where our patients may come from, or "Show all examinations except radiological", rather than defining all options for various types of examinations, etc.



Practical Task

Display a list of all registered patients except those insured by the Generali health insurance company.

For the possibility of defining more complex filtering criteria, the **Advanced** menu on the **Home** tab under the **Sort & Filter** section can be used. Advanced filter options include: "Filter by Form", "Apply Filter/Sort", and "Advanced Filter/Sort" (Figure 5.4).

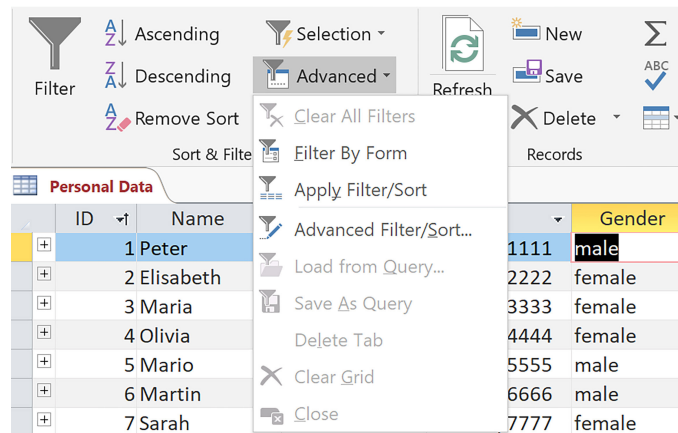


Figure 5.4: Advanced filters.

Filter by form: displays a table with a header containing field names and a single blank record row where filtering criteria can be defined. It is possible to specify the criteria for multiple fields (Figure 5.5).

Personal Data: Filter by Form							
ID	Name	Surname	PIN	Gender	Date of Birth	Address	City
		Like "K*"		"male"			

Figure 5.5: Example of setting filtering criteria for database fields.

For given fields, set criteria (e.g. surnames starting with the letter K and gender is male), and run/apply the filter with the **Toggle Filter** button. The result appears in the database table (Figure 5.6).

Personal Data								
ID	Name	Surname	PIN	Gender	Date of Birth	Address	City	
6	Martin	King	666666/6666	male	12/12/1963	Tr. SNP 57	Kosice	
*	(New)							

Figure 5.6: Result of filtering by form.

Information about records being filtered in the datasheet is available in the status bar, such as shown in Figure 5.7.



Figure 5.7: Status bar after the filter is applied.

The filter can be turned off by clicking the **Toggle Filter** option on the **Home** tab or by turning off the **Filtered** button in the database table's status bar.

Table 5.1 lists several examples of criteria used to filter records in datasheets of database tables or queries.

Table 5.1: Examples of filtering criteria for database records.

Criteria	Usage
<i>Type Field – Text</i>	
Like "M*"	- searches for text starting with the letter M
Not Like "M*"	- searches for text not starting with the letter M
Like "*tion"	- searches for text ending with the string "tion", such as "vaccination", etc. .
"yes"	- searches for text equal to the value yes
Not "Košice"	- searches for text not equal to the value Košice
Is Null	- searches for empty fields (no values entered)
Is Not Null	- searches for filled fields (values were entered)
"Paralen" Or "Ibalgin"	- searches for text that contains only the values Paralen or Ibalgin
<i>Type Field - Number</i>	
100	- find values equal to number 100
>100 (<100)	- find values greater than 100 (less than 100)
>=100 (<=100)	- find values greater than or equal to 100 (less than or equal to 100)
>25 And <26 (Between 25 And 26)	- find values in the range above 25 to 26 (including values 25 and 26)
<i>Type Field – Data and Time</i>	
#dd. mm. rrrr#	- find records with a specific date
<#dd. mm. rrrr#	- find records with dates before the specified date
>#dd. mm. rrrr#	- find records with dates after the specified date
Between #dd. mm. rrrr# And #dd. mm. rrrr#	- find the period between two dates, including the boundary dates
Date()+1 (Date()-1)	- find records with tomorrow's (yesterday's) dates, for example, checking patients scheduled for examination

**Video**

Filter by Form

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)**Practical Task**

Using the advanced filter, find female patients who have permanent residence in Košice (City: Košice, Gender: female).

Filters allow us to filter information stored in only one database table, or a datasheet of a table or query. If we need to find information from two or more tables (data sources) simultaneously, then we prefer to use a database object, which is a query.

**Review Questions**

1. What is the sorting function useful for? Give some examples.
2. What is a filter?
3. Can filter and sorting be combined?
4. What types of filters do we know?
5. What is a disadvantage of filters?
6. What will be the result of filtering after entering the criterion >#01.01.2020# into the field "Date of birth"?
7. How many filters can we set for one database field?

Chapter 6

Queries





When working with extensive databases containing large amounts of data, we encounter situations where certain groups of data are used repeatedly or more frequently than others. Similarly, such groups of data may, and often do, come from various source tables. In these cases, sorting and filtering records in a specific table will no longer suffice, or in the best case, such "data mining" will be laborious and time-consuming. In situations where it is needed to repeatedly search and combine data from multiple tables into different database outputs, or even for further processing and analysis, for example in external applications, it is necessary to use another object of the MS Access relational database, which is a **Query**.

A query is a separate database object that allows users to select data from the entire database and also display only the fields users choose for the query (it is called a selection query). However, queries can also be used to modify existing data, add new data, or remove unnecessary data. This group of queries is also called as action queries. Queries can also be used to perform various calculations and generate the desired summary reports. We can obtain these, for example, using so-called cross-tab queries. The individual types of queries and their modifications will be addressed in the following subchapters.

The result of individual types of queries is a separate tab or window on the desktop with a standard datasheet containing selected records according to specified criteria. Each query can also be used as the input (data source) of another query, i.e. queries are not just created from tables, but if appropriate, also from existing queries. Generally, it can be said that queries are used wherever it is necessary to frequently select records according to more complex criteria or to have various current summary overviews available.

Like other database object types, queries also have different views:

- **Datasheet view** – allows working with records that are the result of a query and are displayed to the user in the form of a standard datasheet,
- **SQL view** – used to define selection criteria and input sorting options for records using SQL commands,
- **Design view** – allows creating and editing a query by specifying its criteria and defining sorting options for records.

Change of query display is possible in the same way as with previous database objects, i.e. via the **View** menu  located for example on the **Home** tab or through the status bar and icons located on its right side   .

6.1 Advanced filter or sorting

The simplest way to learn how to write selection criteria, or at least verify their correct writing at the beginning of work with queries, is to apply the desired filter to some of the fields in a table (or query) datasheet and view them in the design view of the filter using the **Advanced Filter/Sort** feature located in the **Home** tab's **Sort & Filter** menu under **Advanced** button (Figure 6.1).

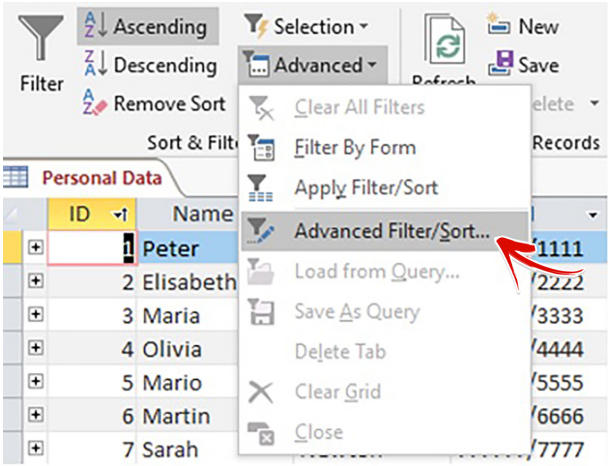


Figure 6.1: Advanced Filter/Sort menu.

In the "Personal Data" database table, we can apply filtering and sorting functions, for example, by selecting all male patients from Košice born before the year 2010 (refer to steps according to Chapter 5) and arrange them alphabetically by surname. We will display the filter settings of the "Personal Data" table using the **Advanced Filter/Sort** option and check the criteria entries in individual fields (Figure 6.2).

Field:	Name	Surname	PIN	City	Gender	Date of Birth
Sort:		Ascending				
Criteria:				"Kosice"	"male"	<#1/1/2010#
or:						

Figure 6.2: Advanced Filter/Sort settings of the Personal Data table.

In the filter settings, you can see that in the first row, we define the field or fields for which we want to apply the filter or sort function. In the second row, we determine whether the records in the filter output should be sorted or not. If so, we choose from ascending or descending options. If records are not to be sorted according to the field, the value is empty. In the third and possibly further rows (if there are more criteria), we enter filtering criteria according to the given field.

If we make changes in the filter settings, then these changes are applied, and we can view the output of the advanced filter in the datasheet by confirming the **Apply Filter/Sort** option in the **Advanced** menu (Figure 6.1). However, advanced filter settings can also be saved as a query. If we open the **Advanced** menu, the **Save as Query** option is active now, and we use it to create our first query. We can name it, for example, "Males born before 2010 in Kosice" and save it by clicking **OK** button. A new category for queries will be added in the navigation pane's object list, including our first query. Whenever we use it, it will always display the current list of males born before 2010 in Kosice. Using the advanced filter and saving its settings is thus the simplest and quickest way to create selection queries.



Note

A query created by saving the settings of an advanced filter and sort displays in its output all fields of the source object (table or query) and not just those for which filtering or sorting criteria are set. However, further adjustments to the query can be made in its design view.

6.2 Creating queries using the wizard

Queries containing information from one or more database sources are usually created using the **Queries** option located on the **Create** tab, which includes options for creating a query using the wizard or by using the design view of a blank query, where we manually define all the required settings (Figure 6.3).

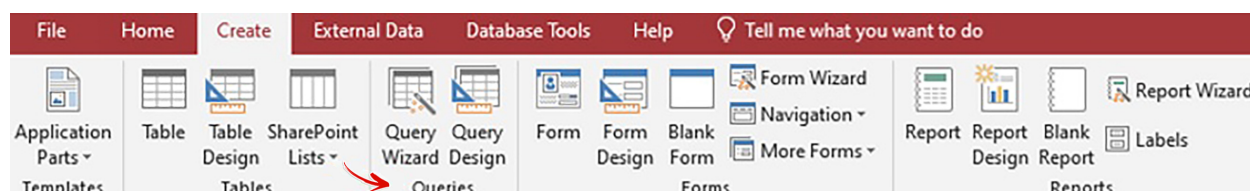


Figure 6.3: Queries option on the Create tab.

Using the **Query Design** option can be somewhat more challenging at the beginning of working with queries, and therefore, there is a higher probability of errors in query design. Thus, we recommend starting first with creating queries using the **Query Wizard** and practicing their modification in the design view.

Using the query wizard, we can create the following types of queries (Figure 6.4):

- **Simple Query** – a query that most closely resembles filters applied in tables, but the output of the query can contain fields from different database tables and queries,
- **Crosstab Query** – displays summary overviews or dependencies of data in selected database fields,
- **Find Duplicates Query** – searches for records with duplicate entries/values, i.e. those that are not unique,

- **Find Unmatched Query** – searches for records for which there are no corresponding records in the linked/related table(s).

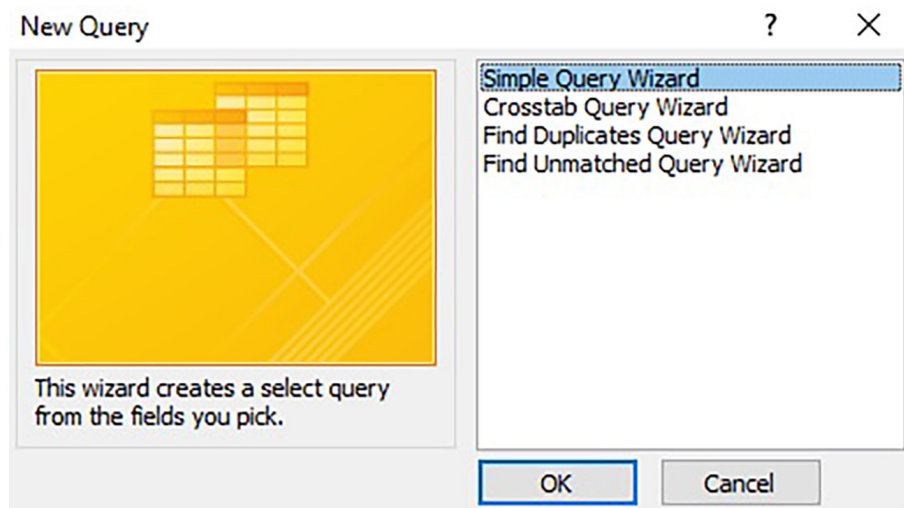


Figure 6.4: Option to create a new query using the Query Wizard.

6.2.1 Simple query

6.2.1.1 Patients from Kosice

We will create a simple selection query that will result in a list of patients from Kosice. On the **Create** tab in the **Queries** menu, choose the **Query Wizard** option to select the Simple Query Wizard (Figure 6.4). Confirm **OK** button, and in the next step, select the database table "Personal Data" and move the required fields for the query from the "Available fields" section to the "Selected fields" section (Figure 6.5).

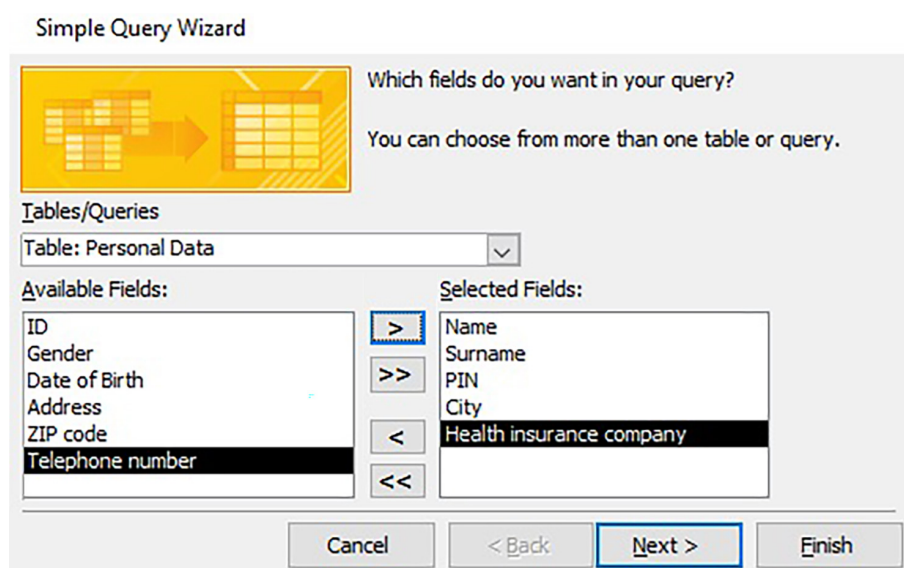


Figure 6.5: Field selection for the Patients from Kosice query.

In the next step of the wizard, decide whether you want to display record details (we keep this option) or a summary, i.e. in the case of numerical data, it is possible to sum, average, find the smallest or largest value. In the last step of the wizard, enter the query name, e.g. "Patients from Kosice", set the option "Open query to view information", and complete the creation of a simple selection query by confirming with the **Finish** button (Figure 6.6).

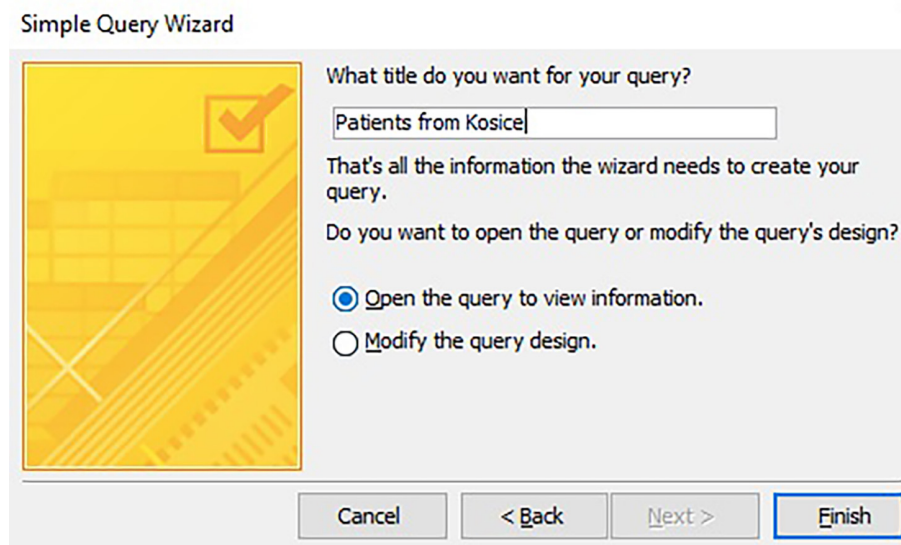


Figure 6.6: Entering the name for the Patients from Kosice query.

The query opens in a datasheet view, and since no selection criteria for records were defined in the query wizard (the wizard does not offer these settings), the query output shows all recorded entries from the "Personal Data" database table with the fields selected in the query wizard (Figure 6.7).

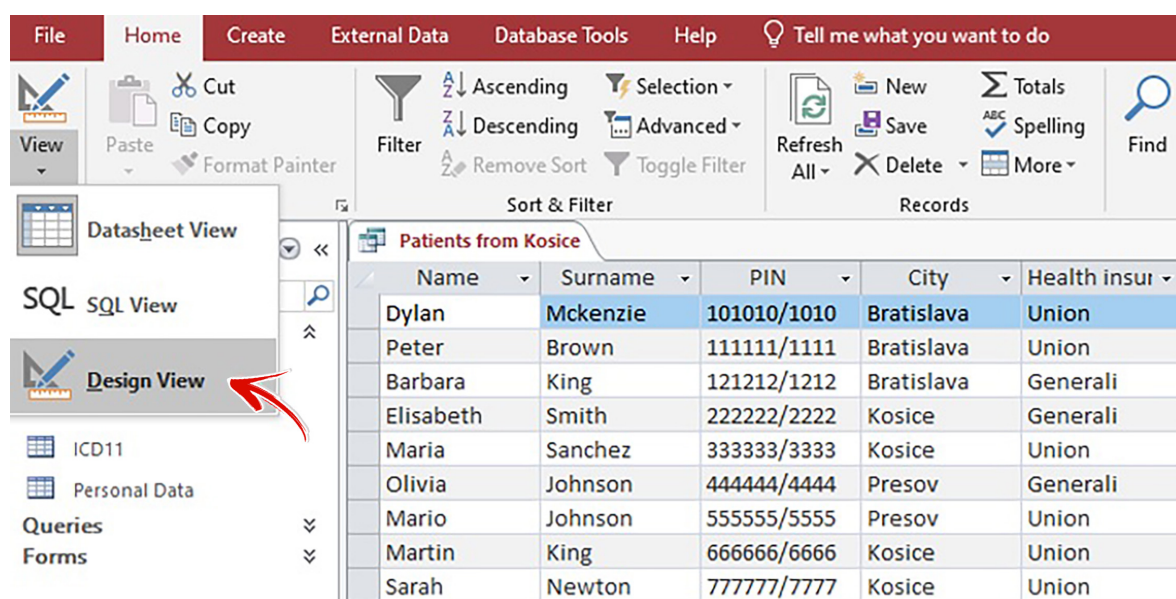


Figure 6.7: Datasheet view of the Patients from Kosice query and view change options.

If we want to set criteria for some fields in a query (in our case for the field **City** to find records with the value "Kosice"), we need to open the query in its design view. To switch the query view, we use well-known options, such as the **View** menu located on the **Home** tab (Figure 6.7) or the view buttons located in the bottom-right corner of the status bar. In the query design view (Figure 6.8), we see a workspace split into two parts. At the top, there is the database table "Personal Data", listed with its fields, and at the bottom, there is a space with query settings. It displays the fields selected for the query, without criteria or sorting functions set. Unlike filter settings, such as the advanced filter shown in Figure 6.2, query settings also include the option to select a data source "Table" and a "Show" checkbox, which allows the values of the field to be shown or hidden in the query output.

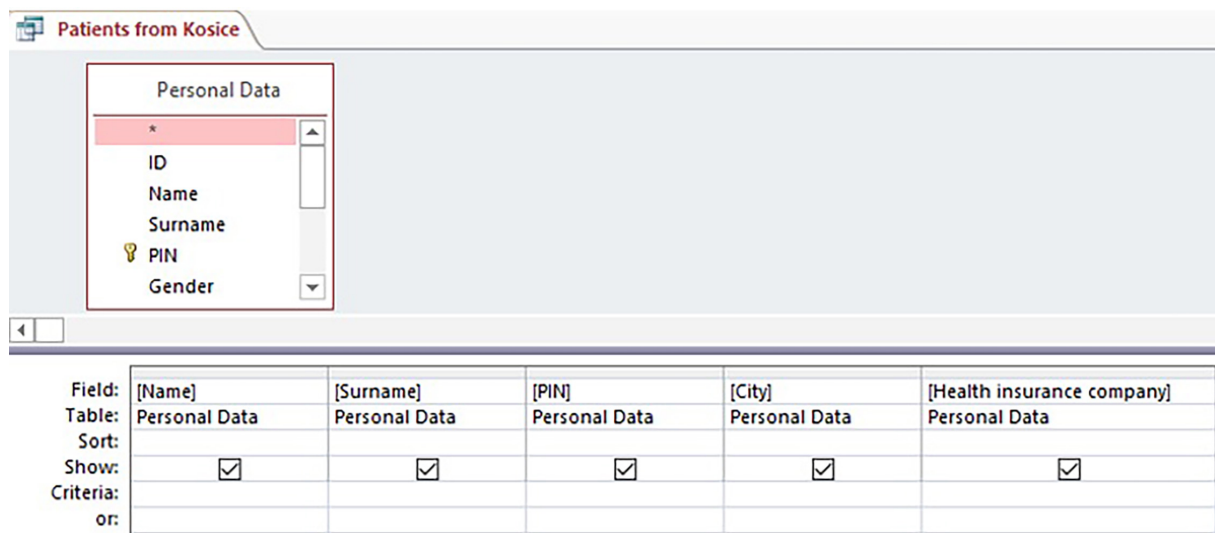


Figure 6.8: Design view of Patients from Kosice query.

If we need to add more fields to the query from the database table, we do this by selecting fields from the field list in the next empty columns in the query settings, which appears when we click into the first row. A field can also be added to the query by double-clicking the left mouse button on the field name in the top section of the query workspace or by selecting the field in the database table and dragging the field name to the bottom section of the query window. Conversely, if we need to delete a field from the query, we select this field by clicking the left mouse button on the gray strip above the field name in the query's lower section. The entire column is then highlighted in black (such marked fields can also be moved), and pressing the **DELETE** key on the keyboard will remove it from the query.

Criteria for individual fields are entered in the same way as in advanced filters into the criteria rows. In addition to other rules, the following apply to entering criteria:

- *Text* can simply be written into the correct database field. For example, write "Kosice" into the **City** field criteria. After entering the text and leaving the cell, MS Access will automatically add the necessary quotes.
- *Logical operators* are part of the criteria (e.g.. =, >, <> (not equal), >=, <=, **Between**, **And**, **Like**, **Not**, **Or**).

- *Date* is entered in the date format defined when creating the table and is enclosed in # symbols (e.g. #01.01.2021#).
- *Parameter* is enclosed in square brackets [].

**Note**

If you are not sure how to enter a criterion or if a query is not functioning correctly, you can verify the correctness of the selection criterion entry by using the advanced filter and checking its entry in the design view (see Chapter 6.1).

For our example, enter the city name, i.e. Kosice, into the *City* field and the "Criteria" row (Figure 6.9).

Field:	[Name]	[Surname]	[PIN]	[City]	[Health insurance company]
Table:	Personal Data	Personal Data	Personal Data	Personal Data	Personal Data
Sort:					
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				"Kosice"	
or:					

Figure 6.9: Criteria setting for the *City* field.**Note**

If multiple values for a single field criterion are needed, additional values can be entered into the **or** rows.

Field:	[City]
Table:	Personal Data
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	"Kosice"
or:	"Bratislava"

Figure 6.10: Two values for the *City* field criterion.

After entering the criteria, save the query in design view and retrieve the query output by switching its view to datasheet or clicking the **Run** button in the **Results** menu on the **Design** tab (Figure 6.11).

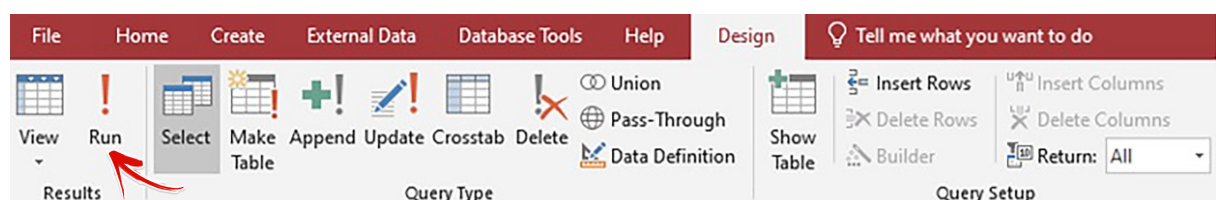


Figure 6.11: Running the query Patients from Kosice.

The "Patients from Kosice" query opens in datasheet view, this time showing only records relevant to the set criterion for the *City* field (Figure 6.12).

Patients from Kosice				
Name	Surname	PIN	City	Health insur
Elisabeth	Smith	222222/2222	Kosice	Generali
Maria	Sanchez	333333/3333	Kosice	Union
Martin	King	666666/6666	Kosice	Union
Sarah	Newton	777777/7777	Kosice	Union
Mariah	Black	888888/8888	Kosice	self-payer
*				

Figure 6.12: Output of Patients from Kosice query.

Besides the list of all patients meeting the criteria at the time the query is used, new records can be added to the query, since it works like a standard datasheet. In reality, a record (provided that the query contains required fields, otherwise it cannot be saved) is stored in the source table and not in the query. If we add a new patient in the query who is not from Kosice, after pressing the **Refresh All** button on the **Home** tab or rerunning the query, this record will not appear in the query (still remains in the "Personal Data" table).

This query can also be used as a data source for a new query. For example, when creating a new query using the query wizard, it will already be available (Figure 6.13).

Which fields do you want in your query?

You can choose from more than one table or query.

Tables/Queries

- Table: Personal Data
- Table: Biometric Data
- Table: Examinations
- Table: ICD11
- Table: Personal Data
- Query: Patients from Kosice

Fields:

Cancel < Back Next > Finish

Figure 6.13: Query Patients from Kosice as a new data source.



Video

Simple query

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)



Practical Task

Create a query with multiple criteria values for the **City** field and verify its functionality by viewing the data in the query's datasheet.

6.2.1.2 Patients born before 2000

Assume we need to work with records of patients of a certain age, for example, those born till the year 2000. Therefore, we will define the selection criterion for the **Date of Birth** field using the same procedure as when creating the previous query. We will use the **Create** tab, **Query Wizard** option, and **Simple Query Wizard** again (Figure 6.3 and 6.4).

In the second step of the simple query wizard, we will select the "Personal Data" database table as the data source and move the required fields (all or at least mandatory ones) from the available fields to the selected fields (Figure 6.14).

Simple Query Wizard

Which fields do you want in your query?
You can choose from more than one table or query.

Tables/Queries
Table: Personal Data

Available Fields:
ID
Gender
Address
ZIP code
Telephone number

Selected Fields:
Name
Surname
PIN
Date of Birth
City
Health insurance company

Buttons: Cancel, < Back, Next >, Finish

Figure 6.14: Selecting fields for the Patients born before 2000 query.

We confirm the selected fields with the **Next** button, choose the type of record detail view, and in the final step of the wizard, we define the name of the query, for example, "Patients born before 2000" (we do not use dots in names). After completing the last step of the wizard, the query opens in data view, and records of all registered patients in the database are available. Therefore, we will display the query in design view and enter the criterion in the form of <#1/1/2000# for the **Date of Birth** field (Figure 6.15).

Field:	[Name]	[Surname]	[PIN]	[Date of Birth]	[City]
Table:	Personal Data	Personal Data	Personal Data	Personal Data	Personal Data
Sort:					
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:				<#1/1/2000#	
or:					

Figure 6.15: Criteria for Date of Birth field.



Note

When entering criteria, it is necessary to maintain the data format. If we had the Date of Birth in the form of 11.12.1989, for example, we would write the criterion for the Date of Birth field in the form of <#1.1.2000#.

After setting the selection criterion, we run the query with the **Run** command located on the **Design** tab (Figure 6.11). The result is a query in datasheet view with a list of all patients born before 2000 (Figure 6.16).

Patients born before 2000						
Name	Surname	PIN	Date of Birth	City	Health insur	
Dylan	Mckenzie	101010/1010	12/3/1999	Bratislava	Union	
Peter	Brown	111111/1111	2/15/1985	Bratislava	Union	
Elisabeth	Smith	222222/2222	3/13/1995	Kosice	Generali	
Maria	Sanchez	333333/3333	6/14/1968	Kosice	Union	
Olivia	Johnson	444444/4444	11/11/1982	Presov	Generali	
Mario	Johnson	555555/5555	12/11/1984	Presov	Union	
Martin	King	666666/6666	12/12/1963	Kosice	Union	
Mariah	Black	888888/8888	1/1/1982	Kosice	self-payer	
Michael	Omar	999999/9999	4/11/1969	Presov	self-payer	
*						

Figure 6.16: Datasheet view of the Patients born before 2000 query.

6.2.1.3 Patients with the initial letter M in their surname

As an example of another simple selection query, we will create a query that will display information about patients with surname starting with the letter M. In the **Create** tab, in the **Queries** and **Query Wizard** menu, select the **Simple Query Wizard** option. From the "Personal Data" table, move the required fields from the available fields to the selected fields, e.g. as shown in Figure 6.17.

Simple Query Wizard

Which fields do you want in your query?
You can choose from more than one table or query.

Tables/Queries
Table: Personal Data

Available Fields:

- ID
- Gender
- Date of Birth
- Address
- ZIP code
- Telephone number

Selected Fields:

- Name
- Surname
- PIN
- City
- Health insurance company

Buttons: Cancel, < Back, Next >, Finish

Figure 6.17: Field selection for the Surname starts with M query.

Confirm the field selection by clicking **Next** button, choose the record detail view type, and in the last step of the wizard, define the query name, such as "Surname starts with M". After completing the last step of the wizard, the query opens in datasheet view. All the

records of registered patients in the database are available. View the query in design view and enter the criterion in the form Like "M*" in the Surname field (Figure 6.18).

Field:	[Name]	[Surname]	[PIN]	[City]	[Health insurance company]
Table:	Personal Data	Personal Data	Personal Data	Personal Data	Personal Data
Sort:					
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		Like "M*"			
or:					

Figure 6.18: Criteria for the Surname field.

Display the query output by switching from design view to datasheet view or by clicking **Run** option on the **Design** tab. The query's datasheet will display all records of patient's surname starting with M (Figure 6.19).

Surname starts with M				
Name	Surname	PIN	City	Health insurance company
Dylan	Mckenzie	101010/1010	Bratislava	Union
*				

Figure 6.19: Datasheet view of the Surname starts with M query.



Practical Task

1. Create a query that finds patients born in the years 2001 to 2010 inclusive. Use the Between operator.
2. Create a query with a list of examinations where a specific medication was prescribed so that there is at least one record in the query. Choose the medication name according to your records in the "Examinations" database table.

6.2.1.4 Selection query from multiple tables

Simple queries are often the data source for other queries, in which various criteria, record sorting, summary overviews, etc. are subsequently defined. In such queries, information found in different tables is often also required. If there is a mutual relationship between some tables, it is advantageous to create one query that will contain all fields from two or more tables and then use it as a source for other queries. For example, to create a cross-tab query (see the next Chapter 6.2.2) where we are interested in an overview of regions (patients' residences) and the occurrence of infectious diagnoses (diseases) in these regions. If we did not have information from two source tables (cities from the "Personal Data" table and diagnoses from the "Examinations" table) combined in one query, we would not be able to conveniently create a cross-tab query using the query wizard. Similarly, it would be impossible to list several other cases where we want output from a query (or later also the reports) to contain information from multiple related database tables.

We will demonstrate creating a query containing fields from multiple related database tables with an example where we want to have in one query all the data from the "Personal

Data" table and the "Examinations" table. We use the **Create** tab, **Query Wizard** option, and **Simple Query Wizard** (Figure 6.3 and 6.4).

In the second step of the wizard, select the "Personal Data" database table as the data source and move all its fields from the available fields to the selected fields. Then change the data source to the "Examinations" database table and again move all its fields to the selected fields (Figure 6.20).

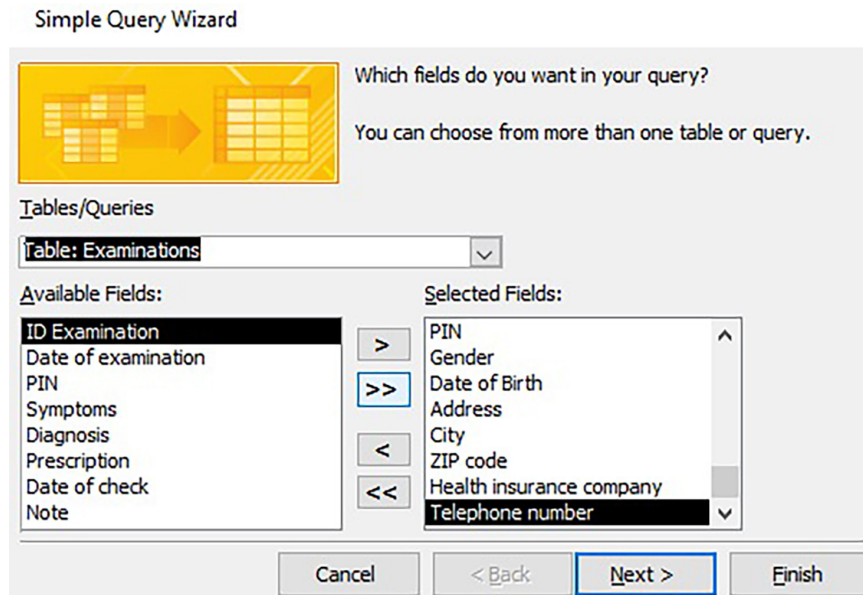


Figure 6.20: Selection of fields from multiple tables.

Confirm the selection of all fields from two related tables by clicking **Next** button, and choose the type of record detail view in the next step, so you have all the information available for queries that will be created from this query

In the last step of the simple query wizard, enter a suitable name, such as "Personal Data and Examinations", and finish creating the query by confirming with the **Finish** button. The newly created simple query opens in data view and will contain detailed records of all registered patients who have at least one examination recorded in the database (Figure 6.21, showing only part of the query fields).

Personal Data and Examinations									
ID	Name	Surname	PIN	Gender	Date of Birth	Address	City	ZIP code	Health insur
2	Elisabeth	Smith	222222/2222	female	3/13/1995	Tr. SNP 56A	Kosice	040 01	Generali
3	Maria	Sanchez	333333/3333	female	6/14/1968	Tr. SNP 23	Kosice	040 01	Union
4	Olivia	Johnson	444444/4444	female	11/11/1982	Main street 25	Presov	080 01	Generali
4	Olivia	Johnson	444444/4444	female	11/11/1982	Main street 25	Presov	080 01	Generali
6	Martin	King	666666/6666	male	12/12/1963	Tr. SNP 57	Kosice	040 01	Union
* (New)									

Figure 6.21: Datasheet view of the Personal Data and Examinations query.

If some registered patients do not have a related record in the "Examinations" table (have not been examined yet), such patients do not appear in the output of the "Personal Data

and Examinations" query (this is not an error of the query). On the contrary, if a patient has undergone examinations multiple times, they appear in the query as many times as there are related records in the "Examinations" table. For example, in Figure 6.21, the data of patient with PIN equal to 444444/4444 appear twice in the query output, i.e. the patient has stored records from two examinations (fields that did not fit in the figure). In the simple selection query "Personal Data and Examinations", we will not set any criteria, but as we mentioned above, we will use it as a data source for other queries where we will require information from both tables simultaneously.

6.2.2 Crosstab query

Crosstab queries are typically used when we want to compare data in database fields, display their totals, averages of numerical values, etc. The fastest way to create them is by using the crosstab query wizard.

The principle of a crosstab query is best explained by an example where we want to generate a summary of our patients' residences with information about the number of men and women (male and female patients) from each city. On the **Create** tab in the **Queries** and **Query Wizard** menu, select the **Crosstab Query Wizard** option (analogous to procedure we mentioned in previous queries according to Figures 6.3 and 6.4) and confirm with the **OK** button. The second step of the cross-query wizard will be displayed, as illustrated in Figure 6.22.

Crosstab Query Wizard

Which table or query contains the fields you want for the crosstab query results?

To include fields from more than one table, create a query containing all the fields you need and then use this query to make the crosstab query.

Table: Biometric Data
Table: Examinations
Table: Personal Data

View
☒ Tables ☐ Queries ☐ Both

Sample:

	Header1	Header2	Header3
TOTAL			

Cancel < Back Next > Finish

Figure 6.22: Selecting the source object for the crosstab query.

We choose the data source from which we will select the database fields **City** (as residence information) and **Gender**, i.e. in our case, it will be the "Personal Data" table where this information is stored, and the result of the crosstab will be the evaluation of the contents of its fields. We move to the next step of the wizard with the **Next** button (Figure 6.23).

Crosstab Query Wizard

Which fields' values do you want as row headings?

You can select up to three fields.

Select fields in the order you want information sorted. For example, you could sort and group values by Country and then Region.

Available Fields:

- ID
- Name
- Surname
- PIN
- Gender
- Date of Birth
- Address
- ZIP code
- Health insurance company
- Telephone number

Selected Fields:

- City

Sample:

City	Header1	Header2	Header3
City1	TOTAL		
City2			
City3			
City4			

Cancel < Back **Next >** Finish

Figure 6.23: Setting the row header of the crosstab query.

For the header rows of our crosstab query, we select the **City** field by moving it from available to selected fields. The crosstab query will have as many rows as there are unique cities and towns listed in the records of patients' personal data. We confirm the selection by pressing the **Next** button, thus moving to the next step of the wizard (Figure 6.24).

Crosstab Query Wizard

Which field's values do you want as column headings?

For example, you would select Employee Name to see each employee's name as a column heading.

Available Fields:

- ID
- Name
- Surname
- PIN
- Gender
- Date of Birth
- Address
- ZIP code
- Health insurance company
- Telephone number

Sample:

City	Gender1	Gender2	Gender3
City1	TOTAL		
City2			
City3			
City4			

Cancel < Back **Next >** Finish

Figure 6.24: Setting the column headers of the crosstab query.

For the column headers of the crosstab query, we mark the **Gender** field (Figure 6.24). This will achieve our cross query having as many columns as there are options in the records of the "Personal Data" table in the Gender field, i.e. one labeled "male" and another labeled "female" (a potential third column could be labeled "undetermined" if such values were present in the table).

The most important setting of the crosstab query is determining the information that will be indicated at the intersections of rows and columns, which we reach with the **Next** button in the next step of the crosstab query wizard (Figure 6.25).

Crosstab Query Wizard

What number do you want calculated for each column and row intersection?
For example, you could calculate the sum of the field Order Amount for each employee (column) by country and region (row).

Do you want to summarize each row?
☒ Yes, include row sums.

Fields:
ID
Name
Surname
PIN
Date of Birth
Address
ZIP code
Health insurance company
Telephone number

Functions:
Avg
Count
First
Last
Max
Min
StDev
Sum
Var

Sample:

City	Gender1	Gender2	Gender3
City1	Count(ID)		
City2			
City3			
City4			

Cancel < Back **Next >** Finish

Figure 6.25: Setting the intersection of rows and columns of the crosstab query.

For the intersection of rows and columns, we select the **ID** field, as each record has its ordinal number. We mark the function as "Count", which counts the ID values belonging to the respective row and column of the crosstab query to find out how many female and male patients there are in a given city. We also select the option to include row sum, which ensures that the total number of patients from a given city is indicated in the crosstab query.



Note

To determine the calculated value indicated as the intersection of a row and column, we recommend using some of the mandatory fields, as this ensures the information is complete and no record is missing.

In the last step of the crosstab query wizard (Figure 6.26), which we reach by pressing the **Next** button, choose the name of the new crosstab query. An appropriate name is, for example, "Crosstab Query City vs Gender" (in names, we do not use dots, for example, if there are abbreviations in the name). To view the output of the crosstab query after creating it, we choose the "View Query" option and finalize the crosstab query design with the **Finish** button.

A new crosstab query will open on the desktop in datasheet view showing the number of patients from a given city and also information on how many of that number are males and how many are females (Figure 6.27).



Video

Crosstab query

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)



Figure 6.26: Name of the crosstab query.

City	Total Of ID	female	male
Bratislava	3	1	2
Kosice	5	4	1
Presov	3	1	2

Figure 6.27: Output of the crosstab query City vs Gender.



Practical Task

Create a crosstab query "City vs Health Insurance".

6.2.3 Find duplicates query

If we need to find or search for records where some information is repeated, i.e. if a field or a group of fields contains the same data, we can use a query with duplicate items. This way, for example, we can create a list of patients who have the same diagnosis, have similar symptoms, use the same medications, or were examined on the same day, etc.

We will test the functioning of the query with duplicate items by creating an overview of patients with the same prescription. The data source for this query will be the "Examinations" database table, which records patient visits to the clinic with descriptions of the solution to their health problems. Again, we start with the **Create** tab, where we select the **Query Wizard** option in the **Queries** menu and choose the **Find Duplicate Query Wizard** option (according to Figures 6.3 and 6.4).

In the second step of the wizard (Figure 6.28), we select the required data source. We can display data sources such as tables and queries separately or together (for clarity, depending on the database size and the number of objects it contains), as we have the option to display tables, display queries, or display both options. We choose "Table" and designate the "Examinations" table as the data source.

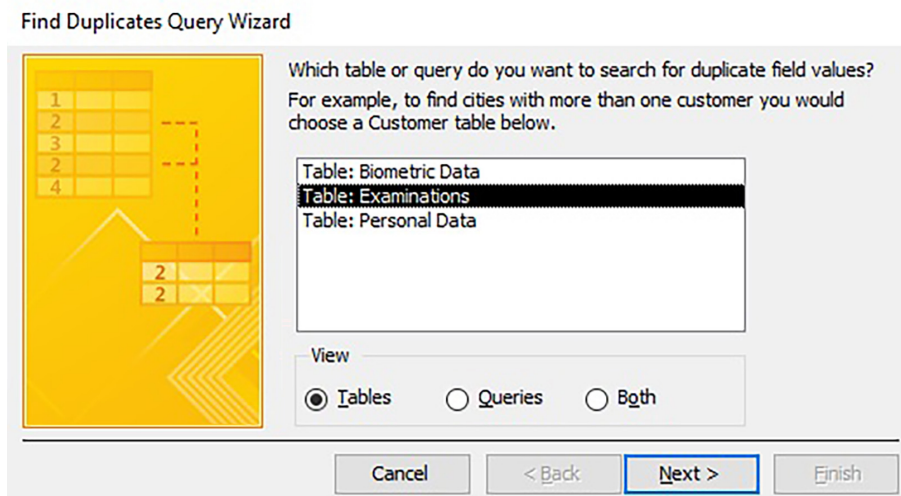


Figure 6.28: Selecting the source for the find duplicates query.

**Note**

If we want the output of the query with duplicate items, aside from the data from the "Examinations" table, to include patient names and surnames (which are in the "Personal Data" table) for better patient identification, we select a query that contains fields from both tables as the data source.

From the available fields, we move the **Prescription** field to the "Duplicate-values fields" section (Figure 6.29). If we select multiple fields, the query will search for records where there is a matching content simultaneously in all of the selected fields (e.g. patients with the same first name and surname).

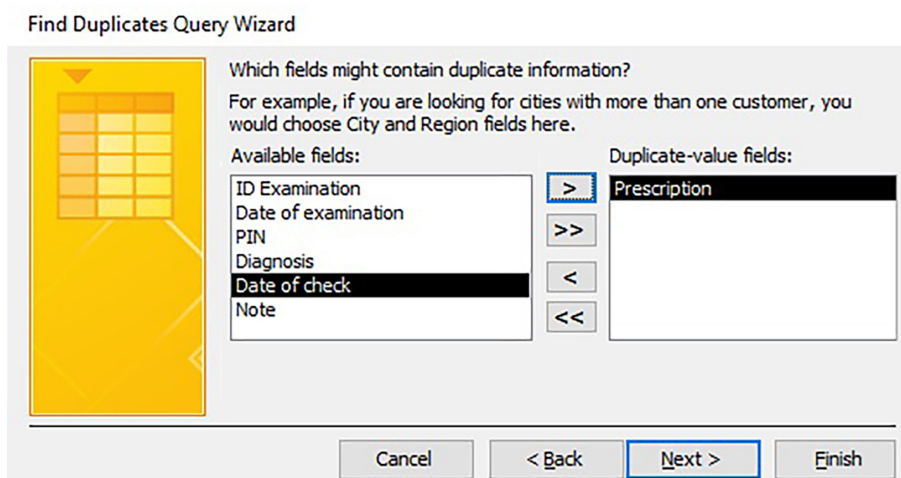


Figure 6.29: Selecting field with duplicate values.

After selecting the field that may contain duplicate values, we proceed with the **Next** button to the next step of the find duplicates query wizard (Figure 6.30). If we want to display additional data along with the field containing duplicate values in the output,

we move them to the "Additional Query Fields" section and proceed to the final step of creating the query with duplicate items, where we choose an appropriate name, such as **Same Prescription** and make it display with the **Finish** button (Figure 6.31).

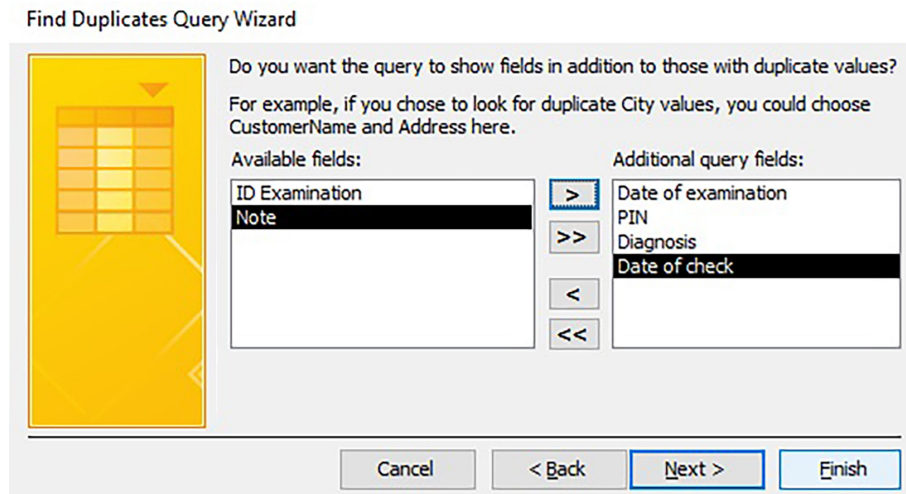


Figure 6.30: Additional query fields for the find duplicates query.

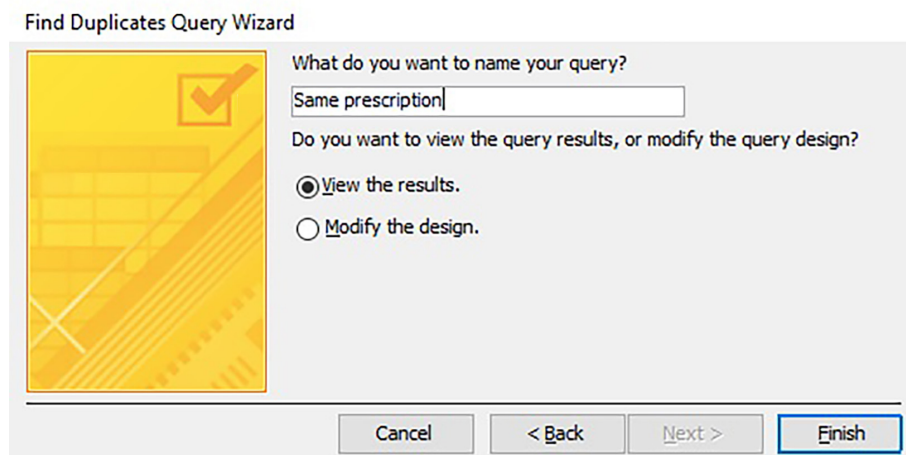


Figure 6.31: Query name for the find duplicates query.

The result is a query containing a list of examinations and their records where the duplicates in prescriptions were identified (Figure 6.32). Such a list can be further sorted and filtered or define specific criteria such as a particular medication or combination of drugs.

Same prescription					
Prescription	Date of exam	PIN	Diagnosis	Date of check	
Antibiotics	1/16/2025	333333/3333	Bronchitis	1/30/2025	
Antibiotics	1/16/2025	222222/2222	Flu	1/22/2025	
*	2/6/2025				

Figure 6.32: Query with duplicate item Same prescription.

**Video****Find Duplicate Query**

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

**Practical Task**

Add new records to the "Examinations" table to have more data. Also, create a query that will find a list of patients with the same diagnosis.

6.2.4 Find unmatched records query

This type of query is used to evaluate the contents of database tables with mutual relationships, i.e. to compare records in such tables. In our case, we will use the query for unmatched records to create a list of patients who are registered in our database but have not yet been examined or do not have any examination record listed in the "Examinations" database table.

We will also create this query using the **Query Wizard** and its **Find Unmatched Query Wizard** option (Figure 6.4). As a data source, select the "Tables" option in the first step of the wizard, mark the "Personal Data" database table (Figure 6.33), and proceed to create the query with the **Next** button.

Find Unmatched Query Wizard

The query you create will list records in the table you select below that have no related records in the table you select on the next screen. For example, you can find customers that have no orders.

Which table or query contains records you want in the query results?

Table: Biometric Data
Table: Examinations
Table: Personal Data

View

☒ Tables ☐ Queries ☐ Both

Cancel < Back Next > Finish

Figure 6.33: Selecting the source database table for the find unmatched query.

In the next step, we select the table where related records should be located. We therefore select the "Examinations" table (Figure 6.34), as we want to know which patients have no examinations record.

**Note**

Depending on the complexity of the created database and the amount of collected data, we could similarly search for patients without vaccination, patients without preventive examination, prescriptions not picked up at the pharmacy, etc.

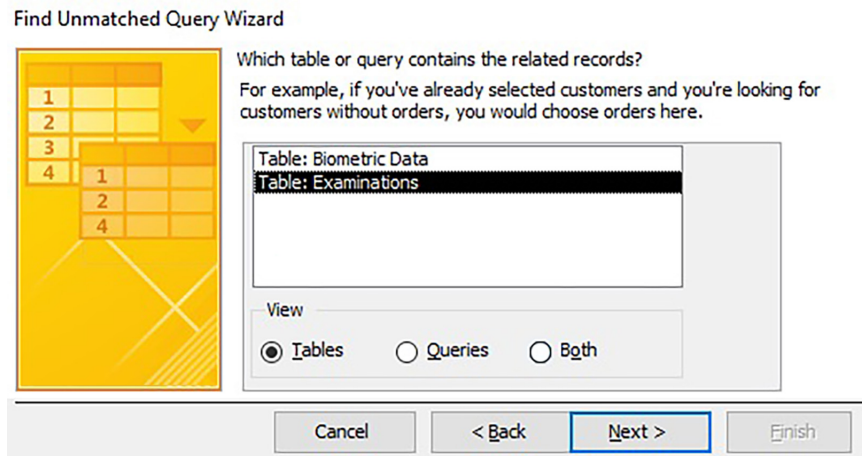


Figure 6.34: Selecting the related database table for the find unmatched query.

The used database tables have a defined mutual relationship, so in the next step, we just check which field is common to both tables (it is the **Personal Identification Number**) and continue with the **Next** button (Figure 6.35).

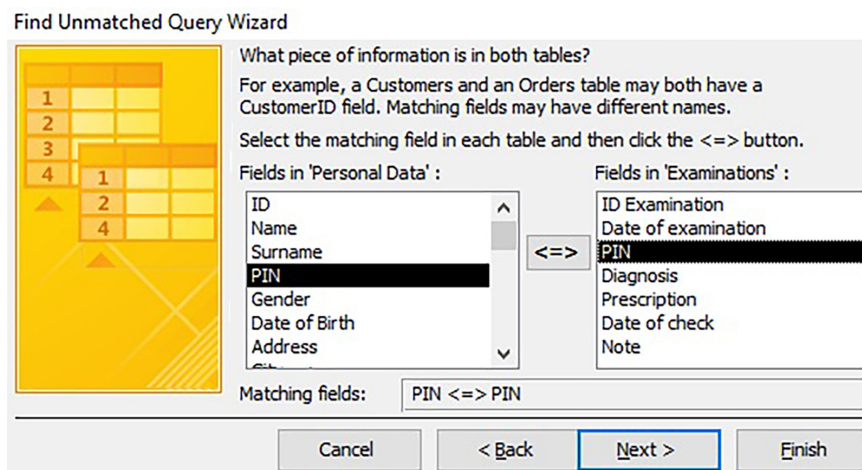


Figure 6.35: Setting up the matching fields.

The output of the query will contain fields from the main table, i.e. the "Personal Data" table, as we are looking for patients who have no examinations. We choose the fields we want to display in our list, for example **Name**, **Surname**, **PIN**, **Gender**, **Date of Birth**, **City**, or we select all of them in case we would use the query to record new entries (Figure 6.36).

To complete the process of creating the search query for unmatched records, we give it a name, such as "Patients without examination", select the "Show Results" option, and confirm with the **Finish** button (Figure 6.37).

The result of the query is a list of all our patients in the datasheet view who do not yet have any examination recorded (Figure 6.38).

If we add new patients to the database, and they do not yet have any examination, then such new patients will also be part of the output of this query. Conversely, the query will stop showing those patients for whom we add the record in the "Examinations" table.

Find Unmatched Query Wizard

What fields do you want to see in the query results?

Available fields:

- ID
- Address
- ZIP code
- Telephone number

Selected fields:

- Name
- Surname
- PIN
- Gender
- Date of Birth
- City
- Health insurance company

Buttons: Cancel, < Back, Next >, Finish

Figure 6.36: Selecting fields to be displayed in the output of the find unmatched query.

Find Unmatched Query Wizard

What would you like to name your query?

Patients without examination

That's all the information the wizard needs to create your query.

Do you want to view the query results, or modify the query design?

☒ View the results.

☐ Modify the design.

Buttons: Cancel, < Back, Next >, Finish

Figure 6.37: Name of query Patients without examination.

Patients without examination						
Name	Surname	PIN	Gender	Date of Birth	City	Health insur
Dylan	Mckenzie	101010/1010	male	12/3/1999	Bratislava	Union
Peter	Brown	111111/1111	male	2/15/1985	Bratislava	Union
Barbara	King	121212/1212	female	12/23/2001	Bratislava	Generali
Mario	Johnson	555555/5555	male	12/11/1984	Presov	Union
Sarah	Newton	777777/7777	female	12/1/2000	Kosice	Union
Mariah	Black	888888/8888	female	1/1/1982	Kosice	self-payer
Michael	Omar	999999/9999	male	4/11/1969	Presov	self-payer
*						

Figure 6.38: Datasheet view of the Patients without examination query.

6.3 Parametric query

Queries that we create to retrieve selected data from one or more tables are adjusted in design views. In the design view, we also specify criteria by which the data is filtered, displayed in details, and possibly used in summary views. The criterion of each query can be set, for example, to show examinations from a specific day, prescriptions with a specific drug, overweight patients, etc. In cases where it is appropriate, criteria can be entered using so-called parametric entry. This means that the user enters the criterion value at the time they want to run the query.

The query creator (the author of the MS Access relational database) creates a parametric query by placing square brackets [] in the expression of the criteria instead of a specific value. Let us imagine that we want to obtain records of a particular patient using a query. One way would be to enter their personal identification number into the criteria of the query, and the query would select only records assigned to this personal identification number. However, if the user wants to display records of another patient, they would have to change this personal identification number in the query's design view (for each patient). Therefore, we write the criteria parametrically, and the user is then prompted to enter the personal identification number of the patient whose records they want to see at the time the particular query is used.

To explain the functioning of the parametric criterion, let us return to the query "Personal Data and Examinations", which we have created in Chapter 6.2.1.4, and which shows all fields from the tables "Personal Data" and "Examinations". Its output is therefore many records belonging to different patients. However, when working with a specific patient, we usually need only records of that patient. Therefore, let us display the query "Personal Data and Examinations" in design view and in the field **Personal Identification Number** belonging to the table "Personal Data", write the criterion in the form: [Enter the patient's personal identification number] (Figure 6.39).

Field:	[Name]	[Surname]	[PIN]	[Gender]	[Date of Birth]	[Address]
Table:	Personal Data	Personal Data	Personal Data	Personal Data	Personal Data	Personal Data
Sort:						
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			[Enter the patient's PIN]			
or:						

Figure 6.39: Criterion for parametric query.

Through the **Run** command on the **Design** tab or by changing the query view to datasheet view, save and run the query thus adjusted. A dialog box will open in which we need to enter the personal identification number of the patient whose records we want to obtain. Let us remember to adhere to the format of the personal identification number (or other fields and data types, if we use the parametric form of criteria for them), as setting the input mask does not apply here (Figure 6.40).

Figure 6.40: Personal identification number for the parametric query.

The result of the query is a list of information about the specific patient and all their examinations, for example, as shown in Figure 6.41.

Parametric query							
Name	Surname	PIN	Gender	Date of Birth	Address	City	
Maria	Sanchez	333333/3333	female	6/14/1968	Tr. SNP 23	Kosice	
*							

Figure 6.41: Record of a patient with a working personal identification number of 333333/3333.

The patient from our sample database with the personal identification number 333333/3333 had only one examination recorded at the time the query was run. If we use the query later (understand weeks or months), for example, after two more examinations, all three records will be displayed in the output.

Every time we use or run this query, the database will require entering the personal identification number of the patient whose records we want to see. This way, we have one query usable for all our patients. If we enter a personal identification number that is not in the database, or the patient does not have a recorded examination, the result will be an empty datasheet of the query.



Video

Parametric query

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)



Practical Task

Create a parametric query "Insurance", which will result in a list of patients insured by the same health insurance company.

6.4 Action queries

In the previous chapters dedicated to various queries, we mentioned that they could also be used to modify existing data or even create new records. Apart from the standard selection or crosstab queries, we also include so-called action queries. We can create them through the query design view. In the toolbar, the **Query Tools** activate, with the **Design** tab and

the **Query Type** options, allowing the creation of other types of action queries, as shown in Figure 6.42.

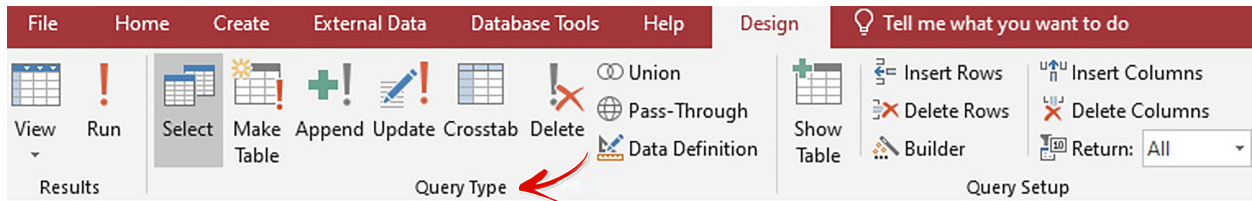


Figure 6.42: Query Types.

An action query can be designed as:

- **make-table query** – saves the query result into a new table,
- **append query** – adds records resulting from the query to the table,
- **update query** – changes data already entered in the database,
- **delete query** – deletes all records in the database containing data valid for the specified criteria.

Action queries have the advantage of modifying a large amount of data in the database per the specified criteria, which would otherwise need to be done manually.



Review Questions

1. What are database queries for?
2. What is the difference between a filter and a query?
3. What types of queries do we know?
4. What is the role of a cross query?
5. How do we create a query with duplicate items?
6. What query views do we know?
7. Through which view and how do we enter criteria for a parameter query?
8. Give an example of a parameter query.
9. What are action queries?
10. Can a query be used to create a new object?

Chapter 7

Reports

Lists of patients for the health insurance company, patients treated in a given month, list of prescribed medications, patient medical record excerpts, list of procedures for a certain period, and other similar lists, i.e. reports, can be created from the data stored in the database. Reports represent independent database objects, in which the user defines what fields will be part of the report, how the data will be formatted, grouped and organized, which summary characteristics will be calculated and displayed in electronic or printed version, etc. Just like with forms, it is also possible to create custom report designs and manage their functionality and design.

7.1 Creating and displaying reports

Reports are created similarly to other database objects via menu located in the **Create** tab (Figure 7.1).

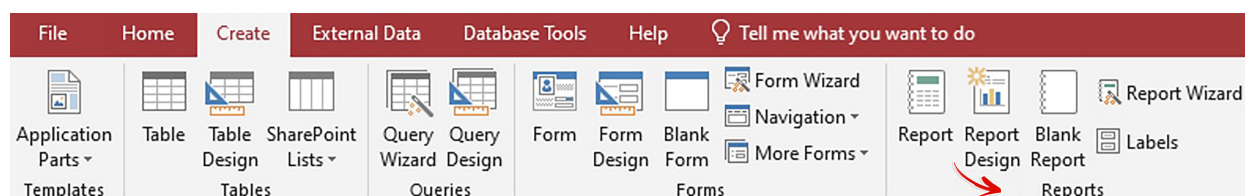


Figure 7.1: Reports menu in the Create tab.

The fastest way to create a report is the **Report** option, which automatically creates an output report from the selected database table or query containing all fields of this source object. The **Report Design** option creates a new empty report in design view, into which you need to insert fields from existing tables or queries, as well as other control elements that are to be used in the report. A similar option for creating a report is also the **Blank Report** menu, which creates a new report in layout view, and through the "Field List" menu, the desired fields are selected and manually added into the report.

The most commonly used choice for creating a report is the **Report Wizard**, which allows us to create basic reports using dialog boxes so that they are laid out according to our

ideas and needs by selecting the required fields from one or more database sources. When creating a report using the **Report Wizard**, we most often encounter three main types of report layouts and their objects, which are:

- **Stepped layout** – places each field on separate row, one after the other, with the first column containing field names and the next column containing field data,
- **Block layout** – uses a table layout, transforming each field into a separate column in the details section, and field names are placed in the report page header section,
- **Outline** – a layout that arranges information from the source object in the smallest possible space, one row contain several fields, the data fills the entire width of the page without large gaps.

When working with reports, it is important to consider the orientation of the sheet on which the report will be printed. For a "Portrait" orientation and A4 page size, it is advisable to place five to six fields side by side, depending on the nature of the displayed information. For a larger number of database fields, "Landscape" orientation is recommended.

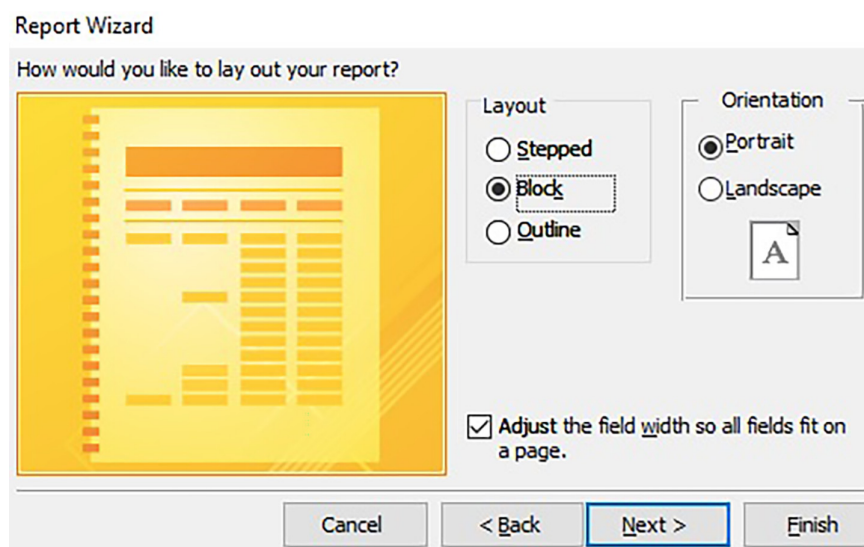
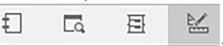


Figure 7.2: Standard report layout types.

You can change the view of an already created and opened report via the **Home** tab and the **View** menu (as shown in Figure 7.3) or via the bottom right corner of the MS Access status bar .

The **Report view** is the basic report view that displays its output in digital form or prepares it for printing. **Print Preview** is a standard preview option commonly used by office applications to check the layout of report content before actually printing it on the selected paper format. **Layout view**, similar like in forms, allows changes to the layout of elements and their appearance, while simultaneously displaying loaded data but not allowing you to edit it. **Design view** shows report controls, allows changes to their properties, and is appropriate for creating a custom report design.

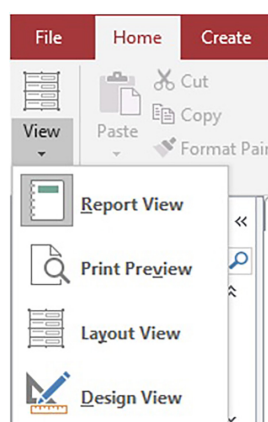


Figure 7.3: View menu for changing report view.

7.2 Automatic report creation

To create an automatic report, select the data source (table or query) in the navigation pane, from which you want to create the report, or use the data source you are currently working with (already open object). For the first report, we are going to create a report for biometric data. Select the "Biometric Data" table and confirm the **Report** option in the **Reports** menu on the **Create** tab (Figure 7.4).

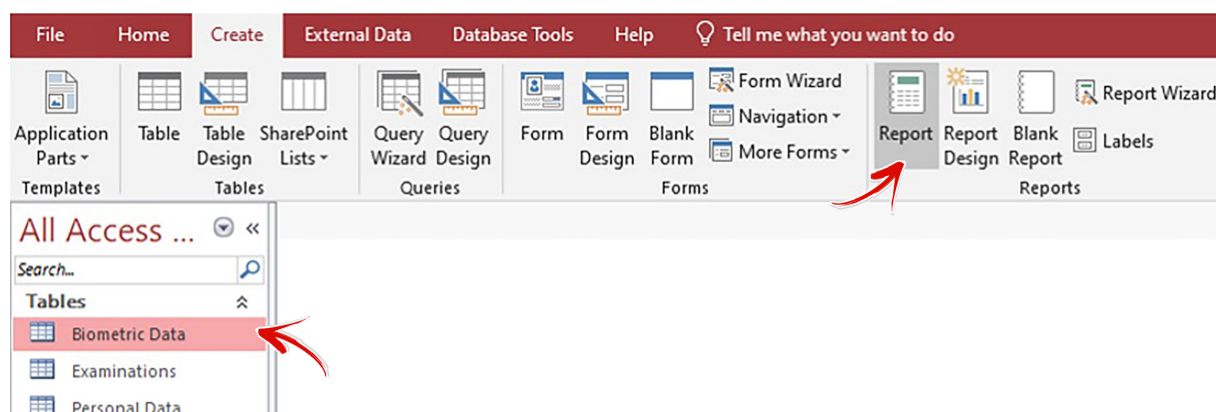


Figure 7.4: Selection a database table to create an automatic report.

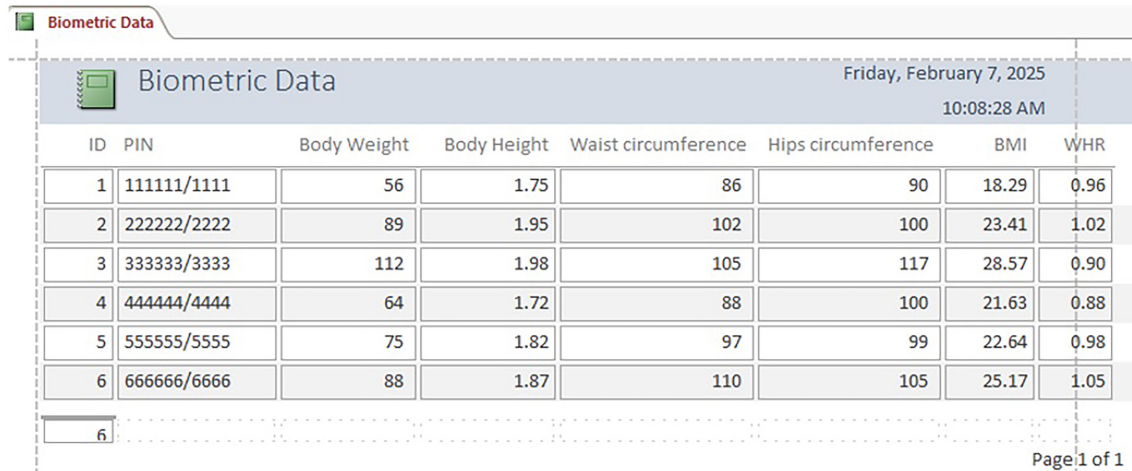
A "Biometric Data" report is generated, that is open in layout view to see the data and allows adjustments if necessary (Figure 7.5).



Note

When designing and editing reports, we use the knowledge from editing forms as this part of the work is identical. However, unlike forms, we use fewer color elements since many reports will be printed (if they must be printed - considering environmental and ecological aspects) in black and white.

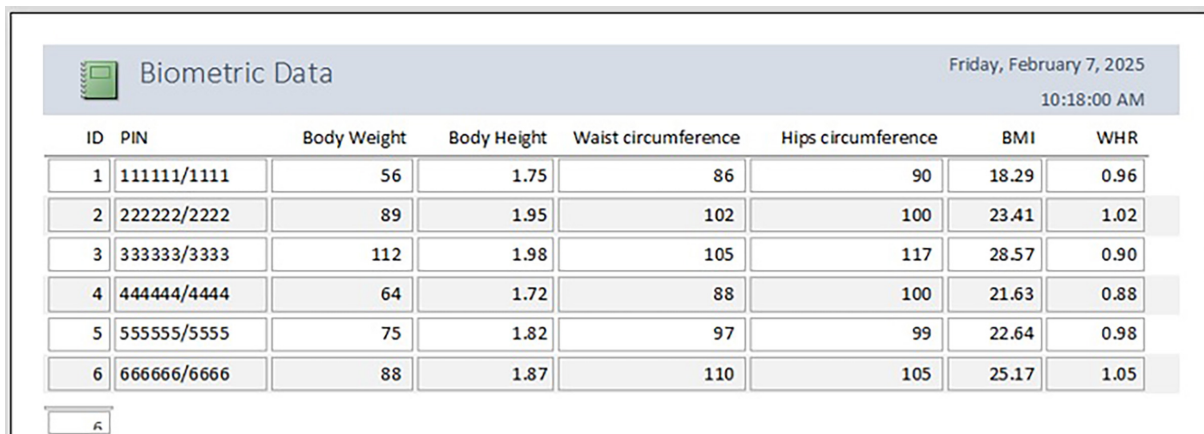
The report sheet orientation is set to "Portrait", and it is visible that fields arranged side by side do not fit across one page width.



ID	PIN	Body Weight	Body Height	Waist circumference	Hips circumference	BMI	WHR
1	111111/1111	56	1.75	86	90	18.29	0.96
2	222222/2222	89	1.95	102	100	23.41	1.02
3	333333/3333	112	1.98	105	117	28.57	0.90
4	444444/4444	64	1.72	88	100	21.63	0.88
5	555555/5555	75	1.82	97	99	22.64	0.98
6	666666/6666	88	1.87	110	105	25.17	1.05


Figure 7.5: Generated Biometric Data report in layout view.

Some basic adjustments in the report design view can be made, for example, by changing the orientation of the report sheet to **Landscape** in the **Page Layout** menu on the **Report Layout Tools** and its **Page Setup** tab. In either the layout view or the design view of the report, adjust the width of the fields so that they fit the width of the page and data is fully visible and readable. To check the design changes made, use the report preview before printing (Figure 7.6).



ID	PIN	Body Weight	Body Height	Waist circumference	Hips circumference	BMI	WHR
1	111111/1111	56	1.75	86	90	18.29	0.96
2	222222/2222	89	1.95	102	100	23.41	1.02
3	333333/3333	112	1.98	105	117	28.57	0.90
4	444444/4444	64	1.72	88	100	21.63	0.88
5	555555/5555	75	1.82	97	99	22.64	0.98
6	666666/6666	88	1.87	110	105	25.17	1.05

Figure 7.6: Print preview before printing Biometric Data report.

In this view, you can see what the printed report will look like. After setting the page parameters, checking data readability, and adjusting the size of database fields, you can print the report through the **Print** menu .

An automatically generated report is not saved yet, so when you first close the report, MS Access will notify you that the report is not saved and you can decide whether to save it as a new database object or not.

7.3 Creating reports using the wizard

Using the report wizard, we will create two reports, and based on these examples, it will be possible to analogically create and modify additional reports for various purposes.

7.3.1 List of examinations

In this section, we will first demonstrate creating a report containing a list of patient examinations sorted by date of examination. On the **Create** tab in the **Reports** menu, confirm the **Report Wizard** option (Figure 7.1). Select the database table "Examinations" and move the required fields to the "Selected Fields" section (Figure 7.7).

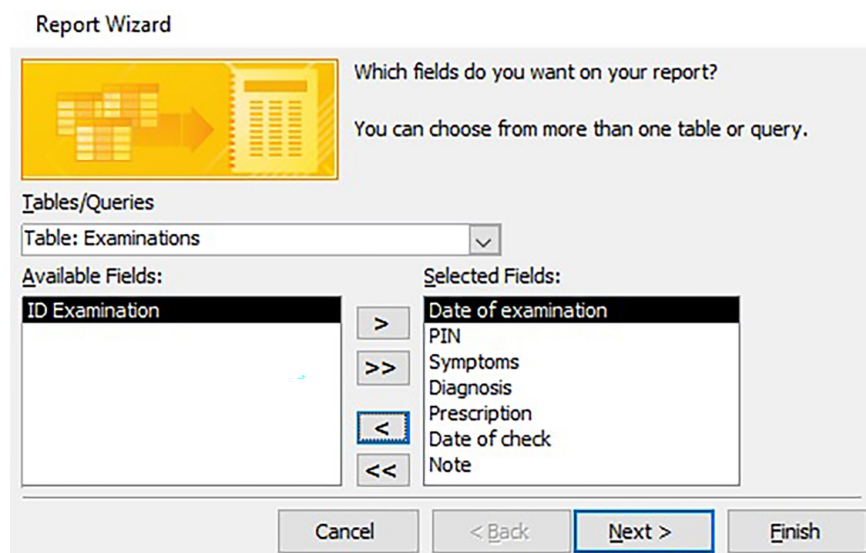


Figure 7.7: Selection of fields for Patient's examination by month report.

In the next step, we will design the field grouping levels by which examination information will be displayed (Figure 7.8).

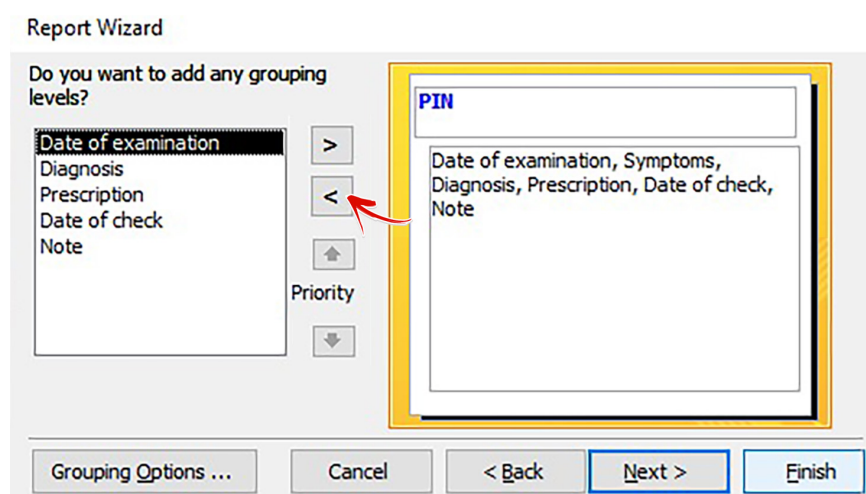


Figure 7.8: Setting the main grouping level.

In the sample, we have the main grouping level of patients sorted by the field **Personal Identification Number**. However, the report we want should contain a list of patient examinations sorted by date of examination. Therefore, swap the **Personal identification Number** and **Date of examination** fields using the arrow buttons in the middle of the dialog box. First, select the PIN field in the sample on the right side of the dialog box with the mouse and move it back to the source table fields using the "<" arrow. Then click on the **Date of examination** field and move it from the available fields to the sample, i.e. use the ">" arrow. After the field swap, we see in the sample report the field "Date of examination by Month" as the main grouping level (Figure 7.9).

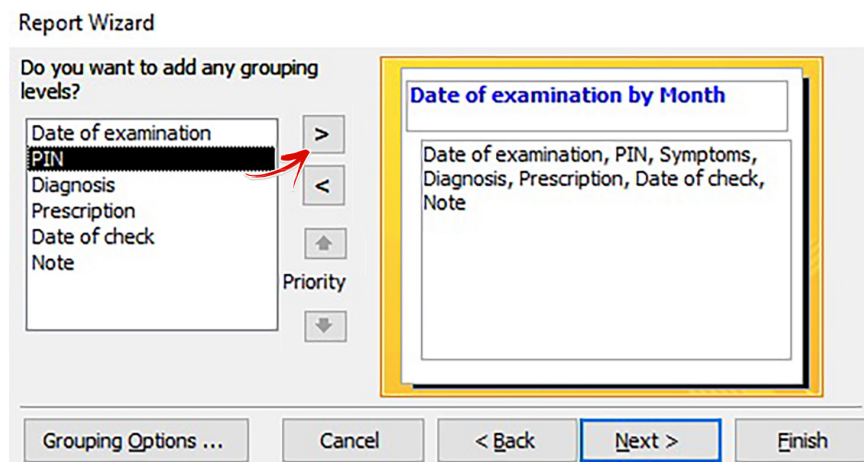


Figure 7.9: Adding Patient's examinations by months as main grouping level.

In the next step of the report wizard, choose the order of other fields you selected in the report and by which you want to sort records for the given month. You can sort records by one to four fields in ascending or descending order, with the highest priority being the first level (Figure 7.10).

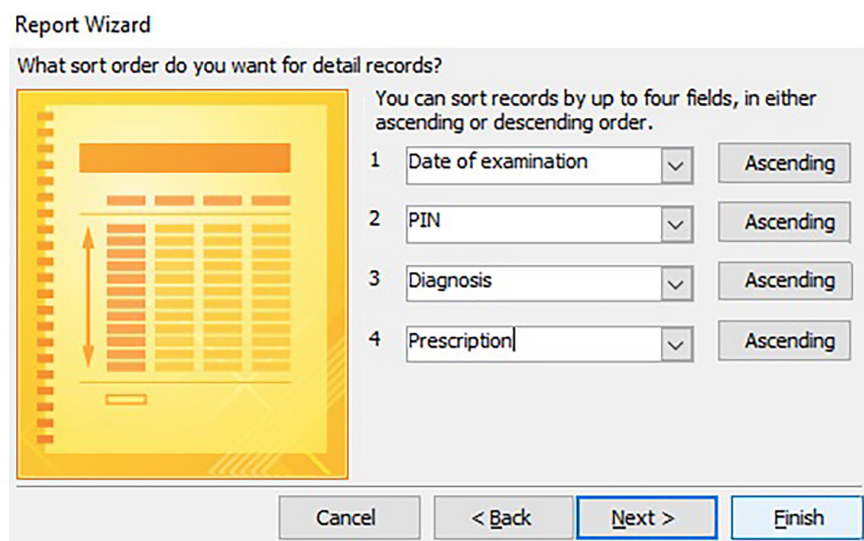


Figure 7.10: Record sorting setting in the report by available fields.

In the last step of the report wizard, choose the type of layout and the orientation of the report pages (Figure 7.11). If the report contains more than five fields, it is recommended to set the report orientation to "Landscape", or after generating the report, rearrange it in the design view.

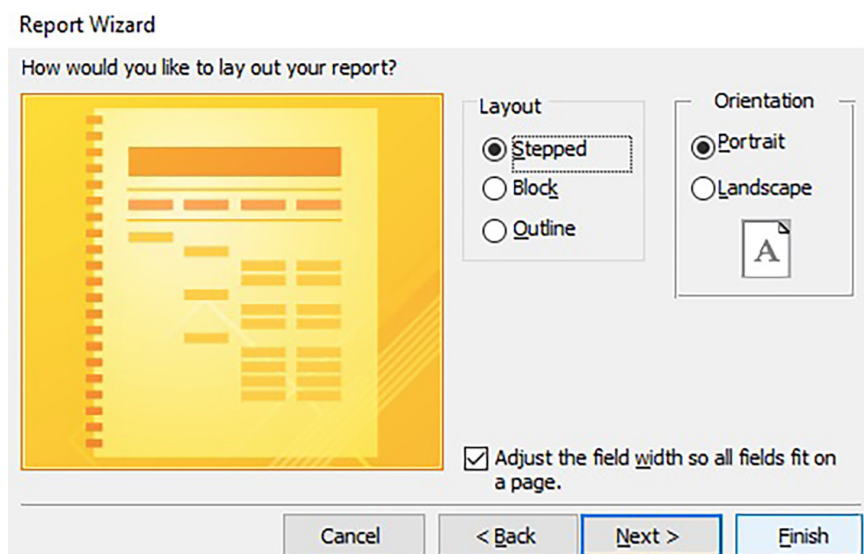


Figure 7.11: Layout and orientation of the report.

Finally, we enter an appropriate report name, for example Patient's examination by months and select the "View Report" option. We confirm the creation of the report by clicking **Finish** button (Figure 7.12).

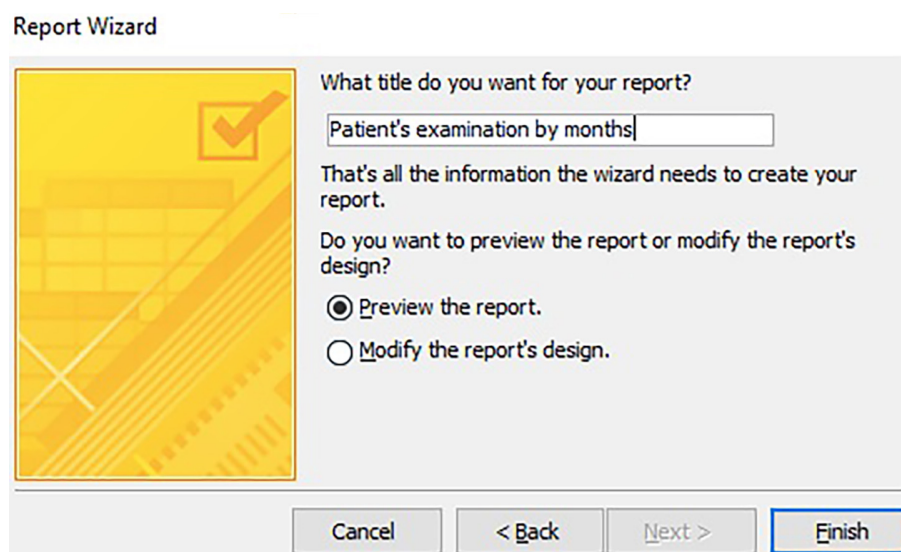


Figure 7.12: Completing the report wizard.

The desired report is generated and displayed in print preview (Figure 7.13). The new report will be saved and also available as another object in the report category of the navigation pane. Almost all reports (like forms), which are automatically created or created with

a wizard, need to be aesthetically adjusted, as some fields are too large for their content, while others are too small to make the entire information visible.

Patient's examinations by months						
Date of exami	of examination	PIN	Diagnosis	Prescription	Symptoms	heck
January 2025						
	1/16/2025	222222/2222	Flu	Antibiotics	Patient feels tired, ### has difficulty breathing. Cough: 1 week, T:38.5°C	
	1/16/2025	333333/3333	Bronchitis	Antibiotics	Patient has difficulty breathing, cough and fever are present	####
	1/16/2025	444444/4444	Eyelid inflammator	Oftalmoseptonex	Patient has inflammation and tearing of the eyes	####
	1/16/2025	666666/6666	Lower limb varicos	Lioton gel, Detrale	Patient complains of leg pain and swelling	
February 202						
	2/6/2025	444444/4444	Migraine	Valetol	Patient has headache	

Figure 7.13: Generated Patient's examination by months report in print preview.



Video

Report Patient's examination by months

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

In the print preview, we can see that several field names and the data within the fields generated by the wizard in the report are unreadable. Therefore, it is necessary to at least enlarge/reduce the width/height of such database fields, and usually we will make several other design changes to the report. We open the report in its design view and follow the same procedures as when designing and editing forms.

In the report header, we change, for example, the name, size, and color of the font. We can also insert the clinic's logo and its name along with the address. In the page header, we expand narrow database fields, change the background of the report, use special effects, etc. In the page footer, there is information about the time and date of the report opening/usage. After making the adjustments, we recommend saving them regularly, we open the report in print preview. If all the field names and data are readable, we can print the report if necessary through the **Print** menu . The adjusted report design may look like the one shown in Figure 7.14.

Patient's examinations by months							
Month	Date of examination	PIN	Symptoms	Diagnosis	Prescription	Date of check	Note
January 2025	1/16/2025	222222/2222	Patient feels tired, has difficulty breathing. Cough: 1 week, T:38.5°C	Flu	Antibiotics	1/22/2025	hot drinks, rest at home
	1/16/2025	333333/3333	Patient has difficulty breathing, cough and fever are present	Bronchitis	Antibiotics	1/30/2025	hot drinks, rest at home
	1/16/2025	444444/4444	Patient has inflammation and tearing of the eyes	Eyelid inflammator	Oftalmoseptonex	1/20/2025	
	1/16/2025	666666/6666	Patient complains of leg pain and swelling	Lower limb varicos	Lioton gel, Detrale		
February 2025	2/6/2025	444444/4444	Patient has headache	Migraine	Valetol		

Figure 7.14: Modified design of Patient's examination by months report.

**Note**

In similar summarization reports, it is advisable to add the patient's name and surname to the mentioned information, which will facilitate communication with the patient and better identification of patients and their records. To create a report containing fields from multiple tables and displaying details of these fields, we use a query containing fields from related tables, for example, a query from previous chapters titled "Personal Data and Examinations".

7.3.2 Personal data by health insurance

As a second example of a report created using the wizard, we will create a report that includes a list of patients grouped by health insurance company.

Report Wizard

Which fields do you want on your report?

You can choose from more than one table or query.

Tables/Queries
Table: Personal Data

Available Fields:

- ID
- Date of Birth
- Address
- City
- ZIP code
- Telephone number

Selected Fields:

- Name
- Surname
- PIN
- Gender
- Health insurance company

Buttons: > >> < <<

Buttons: Cancel < Back Next > Finish

Figure 7.15: Selection of fields for Personal Data by Health insurance company report.

On the **Create** tab in the **Reports** menu, confirm the **Report Wizard** option (Figure 7.1) and select the "Personal Data" table as the data source in the first step. Move fields such as **Name**, **Surname**, **PIN**, **Gender** and **Health insurance Company** to the "Selected Fields" section (Figure 7.15).

In the next wizard step, define the grouping level for records consisting of selected fields. Choose the **Health insurance company** field as the main grouping level for patients, ensuring patients from the same insurance company are grouped together in the report output, followed by other insurance companies and their patients, etc. (Figure 7.16).

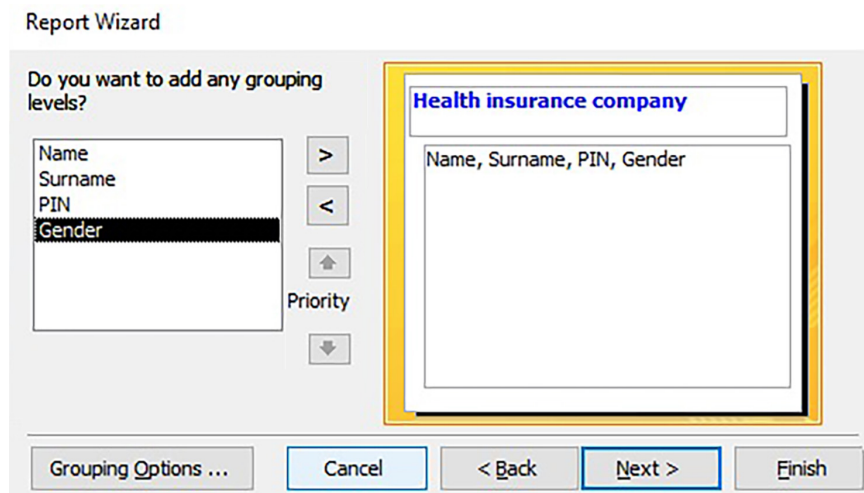


Figure 7.16: Adding Health insurance company as main grouping level.

In the subsequent step, define the order of the records in the report. For organization by personal identification number, add it to the primary sort order. To arrange records by surnames, use the **Surname** field for the first level and the **Name** field for the second to alphabetically organize individuals with identical surname.

Continue by setting the appropriate layout for fields in the report and its orientation (portrait or landscape) for possible printing. In the final step, choose an appropriate report name, such as "Personal Data by Health insurance company" in our example. Keep the "View Report" option selected and confirm creation with the **Finish** button.

A new report will be generated and displayed as a print preview. Design adjustments can be refined in layout view or design view, with the result possibly appearing as shown in Figure 7.17.



Note

Once again, when creating reports, you can also use queries as data sources, in addition to database tables. Therefore, if you want to create a report where fields from multiple tables are included, such as adding blood type, date of last vaccination, or allergies to records, we recommend first creating a query and then using it as the data source for the report.


 <div> New Hospital Tr. SNP 1 Kosice </div> <div> Personal data by health insurance company </div>				
Insurance company	Surname	Name	PIN	Gender
Generali				
	Johnson	Olivia	444444/4444	female
	King	Barbara	121212/1212	female
	Smith	Elisabeth	222222/2222	female
self-payer				
	Black	Mariah	888888/8888	female
	Omar	Michael	999999/9999	male
Union				
	Brown	Peter	111111/1111	male
	Johnson	Mario	555555/5555	male
	King	Martin	666666/6666	male
	Mckenzie	Dylan	101010/1010	male
	Newton	Sarah	777777/7777	female
	Sanchez	Maria	333333/3333	female

Figure 7.17: Custom report design in print preview.



Practical Task

Create a report that includes data on the patient's Name, Surname, Personal Identification Number, City and their BMI. Categorize the records by the patient's place of residence.

7.4 Labels

In addition to typical overview reports displaying details loaded from source objects, it is possible to create other specific reports. For example, a report that allows printing patient data, diagnosis, and prescribed medications on a standard prescription form (if a printed version is required), information about incapacity for work on forms for the employer and insurance company, etc. Various labels and name tags can also be printed, for example, to label collected samples.

To create labels or tags, we use the **Create** tab, **Reports** menu and the **Labels** option (see Figure 7.4). Labels, similar to the report wizard, allow us to create name tags or labels through dialog boxes where we set their size, font attributes, and desired database fields. Using the label wizard we create a group of labels, for example, to use for labeling patients' blood samples. First, define the label size settings and the number of labels in one row. We should use available label sizes, i.e. printed labels will not need trimming. In our case, we have labels with a size of 34x27 mm for one label and 3 in one row (Figure 7.18).

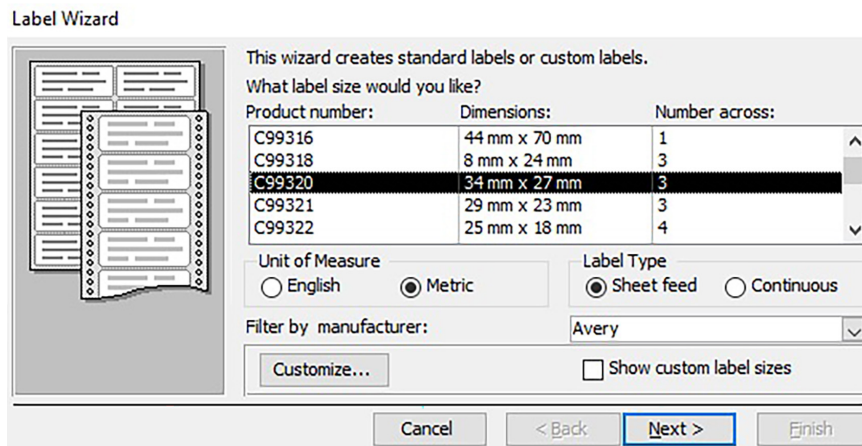


Figure 7.18: Setting label size and the number of labels per row.

Confirm the selection with the **Next** button, which also moves us to the step of choosing font settings to be used when printing labels (Figure 7.19).

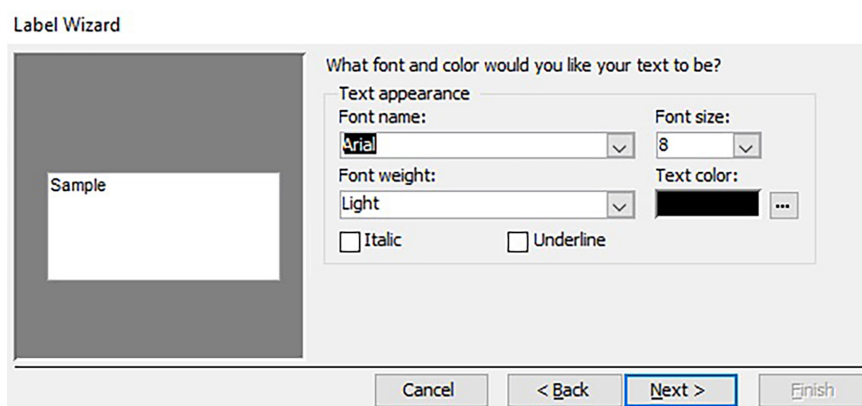


Figure 7.19: Defining font type, size, and color for labels.

Proceed to the selection of fields and their arrangement on labels with the **Next** button (Figure 7.20).

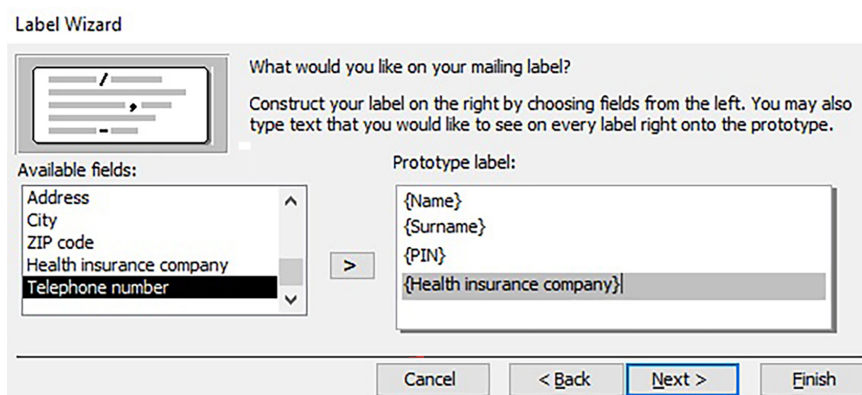


Figure 7.20: Field selection for label description.

When inserting individual fields into the sample description (double-click the left mouse button or the arrow button ">"), do not forget to add spaces between the database fields (e.g., between the name and surname, otherwise they will be joined). Use the ENTER key to move to the next line in the sample description.

In the next step, no available fields are selected for sorting records, but in case of many labels, they could be sorted, for example, by patients' surnames. Confirm the settings with the **Next** button, and in the final step, give a name to the label report, e.g., "Labels Personal Data" (Figure 7.21).

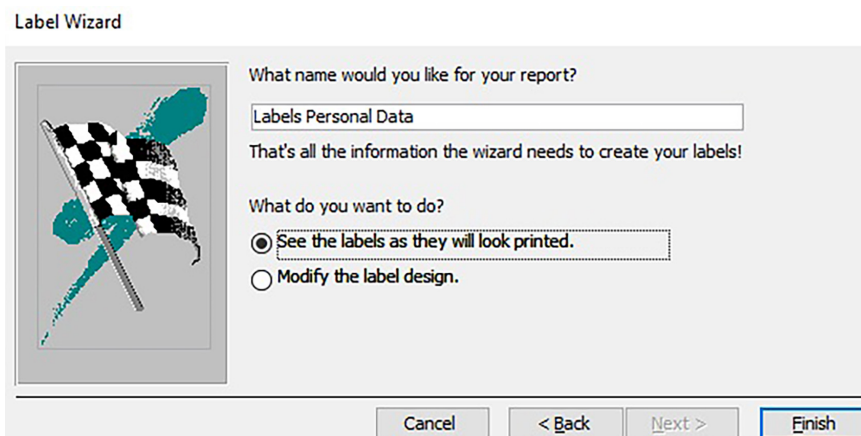


Figure 7.21: Assigning a name to the label report.

The "Labels Personal Data" opens in print preview where we check the content, accuracy, and completeness of the required information (Figure 7.22). However, check that the changes are not necessary, as data data may overflow into the next line and labels created in this way would generally not be suitable for printing.

Mariah Black 888888/8888 self-payer	Peter Brown 111111/1111 Union	Olivia Johnson 444444/4444 Generali
Mario Johnson 555555/5555 Union	Martin King 666666/6666 Union	barbara king 121212/1212 Generali
Dylan McKenzie 101010/1010 Union	Sarah Newton 777777/7777 Union	michael omar 999999/9999 self-payer
Maria Sanchez 333333/3333 Union	Elisabeth Smith 222222/2222 Generali	

Figure 7.22: Patient labels in print preview.

Open the labels in design view and make some common formal adjustments. For example, increase the width of all fields, set the font to bold for the database field **Name** and **Surname**, center the text, etc. To check the output, switch the report view back to print preview and from the **Page Setup** menu in the **Columns** tab, you can adjust the spacing of columns as needed so that the labels on the sheet are evenly distributed, then confirm the changes with the **OK** button (Figure 7.23).

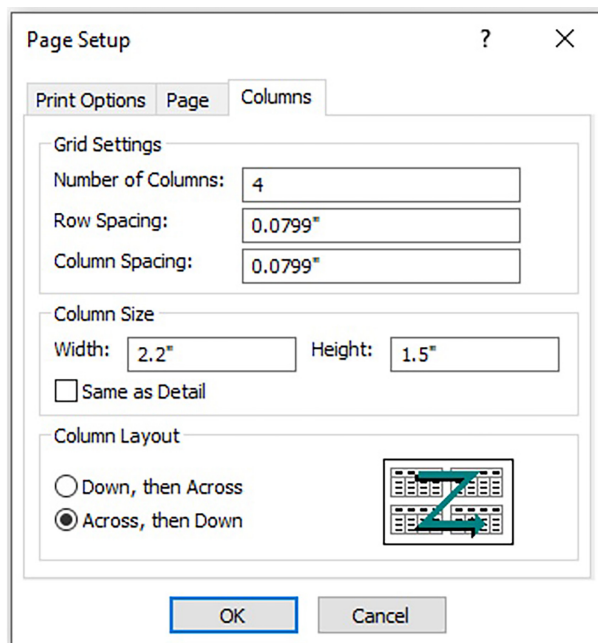


Figure 7.23: Setting column spacing.

Save the changes in the report and you can reuse it if needed, even with new patients who can be added to the database later. An example of edited labels before printing is shown in Figure 7.24. Insert paper with adhesive labels (stickers) into the printer and after printing, stick them on the test tubes with patients' samples.

<p>Mariah Black 888888/8888 self-payer</p>	<p>Peter Brown 111111/1111 Union</p>	<p>Olivia Johnson 444444/4444 Generali</p>	<p>Mario Johnson 555555/5555 Union</p>
<p>Martin King 666666/6666 Union</p>	<p>Barbara King 121212/1212 Generali</p>	<p>Dylan McKenzie 101010/1010 Union</p>	<p>Sarah Newton 777777/7777 Union</p>
<p>Michael Omar 999999/9999 self-payer</p>	<p>Maria Sanchez 333333/3333 Union</p>	<p>Elisabeth Smith 222222/2222 Generali</p>	

Figure 7.24: Print preview of patient labels after editing.

**Video****Patient Labels**

(See Portal UPJŠ LF, <https://portal.lf.upjs.sk/clanky.php?aid=57>)

**Note**

It is not always necessary to create/print labels for all patients. You can create a query that selects only a certain group of patients, for example, from a specific city, with a certain diagnosis, insured by a given insurance company, etc., and use this query to create labels.

**Practical Task**

Create labels for marking samples from women patients with names and surnames and any three other database fields.

**Review Questions**

1. Which database objects do we use to create a printed report?
2. Define individual types of report views.
3. Which report view allows us to check the readability of field names and data?
4. Through which view can we create our own customized report design?
5. What is the difference between the options to create a Report and create Labels?

Bibliography

- [1] Belko, P.: *Access 2013*. Brno: Computer Press, 2014. ISBN 978-80-251-4125-0.
- [2] Breil, B., Fritz, F., Thiemann, V. & Dugas, M.: *Multidisciplinary education in medical informatics – A course for medical and informatics students*. Studies in Health Technology and Informatics, 2010, 160(1) 581-584. DOI: 10.3233/978-1-60750-588-4-581.
- [3] Dhru, N.: *Office 365 for healthcare professionals: Improving patient care through collaboration, compliance, and productivity*, (2018), DOI: 10.1007/978-1-4842-3549-2.
- [4] Firsova, S. A., Ryabukhina, E. A.: *Integrative method of teaching information modeling in practical health service based on microsoft access queries*, Integration of Education, 2016, 20(2), 264-280. DOI: 10.15507/1991-9468.083.020.201602.264-280.
- [5] Fridsma, D. B.: *Health informatics: A required skill for 21st century clinicians*. BMJ (Online), 2018, 362, k3043-k3043. DOI: 10.1136/bmj.k3043.
- [6] Goh, P. S.: *eLearning or technology enhanced learning in medical education – Hope, not hype*. Medical Teacher, 2016, 38(9), 957-958. DOI: 10.3109/0142159X.2016.1147538.
- [7] Haux, R., Ammenwerth, E., Haber, A., Hühner-Bloder, G., Knaup-Gregori, P., Lechleitner, G., Leiner, F., Weber, R., Winter, A. & Wolff, A. C.: *Medical informatics education needs information system practicums in health care settings: Experiences and lessons learned from 32 practicums at four universities in two countries*. Methods of Information in Medicine, 2006, 45(3), 294-299. DOI: 10.1055/s-0038-1634073.
- [8] Held, B.: *Access VBA*. Brno : Computer Press, 2006. ISBN 80-25111-12-1.
- [9] Hovenga, E. J. S.: *Globalisation of health and medical informatics education - what are the issues?* International Journal of Medical Informatics, 2004, 73(2), 101-109, DOI: 10.1016/j.ijmedinf.2003.11.004.
- [10] Jidkov, L., Alexander, M., Bark, P., Williams, J. G., Kay, J., Taylor, P., Hemingway, H. & Banerjee, A.: *Health informatics competencies in postgraduate medical education and training in the UK: A mixed methods study*. BMJ Open, 2019, 9(3), e025460-e025460. DOI: 10.1136/bmjopen-2018-025460.
- [11] Kruczek, A.: *Microsoft Access 2010: Podrobná uživatelská příručka*. Brno : Computer Press, 2010. ISBN 978-80-251-3289-0.
- [12] Law, G. C., Apfelbacher, C., Posadzki, P. P., Kemp, S. & Tudor Car, L.: *Choice of outcomes and measurement instruments in randomised trials on eLearning in medical education: A systematic mapping review protocol*. Systematic Reviews, 2018, 7(1), 75-75. DOI: 10.1186/s13643-018-0739-0.

- [13] Lungeanu, D., Tractenberg, R. E., Bersan, O. S. & Mihalas, G. I.: Towards the integration of medical informatics education for clinicians into the medical curriculum. *Studies in Health Technology and Informatics*, 2009, 150, 936-940. DOI: 10.3233/978-1-60750-044-5-936.
- [14] Masic, I. & Pandza, H.: *Medical informatics education - past, today and future*. *European Journal for Biomedical Informatics*, 2018, 14(2) DOI: 10.24105/ejbi.2018.14.2.8.
- [15] Matiaško, K. a kol.: *Databázové systémy – 1. diel*. Žilina : EDIS, 2018. ISBN 978-80-554-1488-1.
- [16] Matiaško, K. a kol.: *Databázové systémy – 2. diel*. Žilina : EDIS, 2018. ISBN 978-80-554-1489-8.
- [17] Morgado, F.: *Introducing Microsoft Access Using Macro Programming Techniques: An Introduction to Desktop Database Development by Example*, DOI: 10.1007/978-1-4842-6555-0.
- [18] Pecinovský, J.: *Excel a Access 2010 – efektivní zpracování dat na počítači*. Praha : Grada, 2011. ISBN 978-80-247-7416-9.
- [19] Röhrig, R., Stausberg, J., Dugas, M.: *Development of national competency-based learning objectives "medical informatics" for undergraduate medical education*. *Methods of Information in Medicine*, 2013, 52(3), 184-188. DOI: 10.3414/ME13-04-0001.
- [20] Shufnarovych, M. A.: *Some prospects for the use of modern information technology*. *Scientific Bulletin of UNFU*, 02/2017, 27(1), 222-225. DOI: 10.15421/40270151.
- [21] Spišáková, M.: *Databázový systém MS Access pre stredné odborné školy*. Bratislava : Slovenské pedagogické nakladateľstvo – Mladé letá, s r. o., 2017. ISBN 978-80-10-03097-2.
- [22] Voglová, B.: *Excel a Access*. Praha : Grada, 2004. ISBN 978-80-247-6325-5.
- [23] Winter, A., Hilgers, R. D., HofestÄdt, R., Knaup-Gregori, P., Ose, C. & Trimmer, A.: *More than four decades of medical informatics education for medical students in germany*. *Methods of Information in Medicine*, 2013, 52(3), 181-183. DOI: 10.1055/s-0038-1627057.
- [24] Zdeněk, M.: *Access v příkladech: Praktická učebnice databázového programu*. Brno : Computer Press, 2006. ISBN 80-86686-55-8.
- [25] Zvarova, J.: *Medical decision support and medical informatics education: Roots, methods and applications in czechoslovakia and the czech republic*. *Yearbook of Medical Informatics*, 2013, 8, 206-212. DOI: 10.1055/s-0038-1638857.
- [26] Pomoc a vzdelávanie pre Access: <https://support.microsoft.com/sk-sk/access>.

Index

A

action query, 123, 145
advanced filter, 124
advanced form, 109
automatic report, 149
autonumber, 35

B

BMI, body mass index, 55
bound form, 85
BSA, body surface area, 58
button

- new record, 108
- next record, 106
- previous record, 107

C

calculated value, 35
calendar, 73
cell zoom, 74
command button wizard, 106
copy field, 49
cross-tab query, 123
crosstab query, 135
currency, 35

D

data selection, 117
data sorting, 117
data type

- attachment, 35
- autonumber, 35
- calculated, 35, 56
- currency, 35
- date and time, 35
- hyperlink, 35

- long text, 35
- lookup wizard..., 35
- number, 35, 37, 52
- OLE object, 35
- short text, 35
- yes/no, 35

data

- import, 79
- refresh, 77

data, 11

database system, 14

database, 13

datasheet view, 30, 123

date and time, 35

default value, 44, 48

delete record, 62

design view, 30, 86, 123, 148

duplicate items, 138

E

expression builder, 43, 48

F

field size, 44

field

- copy, 49
- paste, 49

field, 17, 27, 30

filter

- advanced, 120
- by selection, 119
- out of selection, 119

filter, 11, 118

form body, 100

form design, 90

form details, 100
 form footer, 106
 form grid, 97
 form header, 98
 form view, 86
 form with subform, 109
 form wizard, 85, 91
 form

- advanced, 109
- blank, 91
- bound, 85
- columnar, 92
- datasheet, 95
- design view, 96
- justified, 95
- layout, 92
- linked, 112
- navigation, 114
- tabular, 94
- unbound, 85

form, 11, 27, 85

H

hyperlink, 35

I

ICD, international classification
 of diseases, 78

information, 11

input mask wizard, 39

input mask, 38

installation, 16

L

label, 157

layout view, 86, 148

layout

- columnar, 92
- datasheet, 95
- justified, 95
- tabular, 94

linked forms, 112

long text, 35

lookup wizard, 35, 41

M

macro, 27

main key, 18, 59

module, 27

MS Access, 10, 21

N

name tag, 157

navigation form, 114

navigation pane, 25

number, 35, 37

O

OLE object, 35

P

parametric query, 144

paste field, 49

primary key, 18, 59

print preview, 148

property sheet, 101

Q

query wizard, 125

query

- action, 145
- duplicate items, 138
- parametric, 144
- unmatched records, 141

query, 11, 27, 123

quick access toolbar, 24

R

record, 15, 18, 27, 30

referential integrity, 67

refresh, 77

relational database, 63

relationship

- 1:1, 64
- 1:N, 64

- delete, 70
- edit, 70
- M:N, 64

relationship, 30, 65

rename form, 98

report view, 148

report wizard, 148

report

- block layout, 148
- outline layout, 148
- stepped layout, 148
- wizard, 151

report, 11, 27, 147

ribbon, 24

S

selection query, 123

short date, 43, 48

short text, 35

sort records, 117

sorting, 11

SQL view, 123

status bar, 26, 87, 120

subdatasheet, 75

subform wizard, 103

subform, 11, 87, 103, 109

T

table, 10, 15, 26

toolbar

- navigation pane, 25
- quick access, 24
- ribbon, 24

U

unbound form, 85

unmatched records, 141

V

validation rule, 43, 51

validation text, 44, 51

view

- datasheet, 30
- design, 30, 86, 96, 148
- form, 86
- layout, 86, 148
- print preview, 148
- report, 148

W

WHO, world health organization, 78

WHR, waist to hip ratio, 58

wizard

- form, 91
- query, 125
- report, 151
- subform, 103

Y

yes/no, 35

Z

zoom, 74

Introduction to MS Access
Databases not only for medical students
University textbook

Authors: doc. Ing. Jaroslav Majerník, PhD.
Ing. Andrea Kačmariková, PhD.

Publisher: Pavol Jozef Šafárik University in Košice
ŠafárikPress Publishing

Year: 2025

Pages: 168

Author's sheets: 14,8

Edition: first

DOI: <https://doi.org/10.33542/IMA-0425-5>
ISBN 978-80-574-0425-5 (e-publication)

