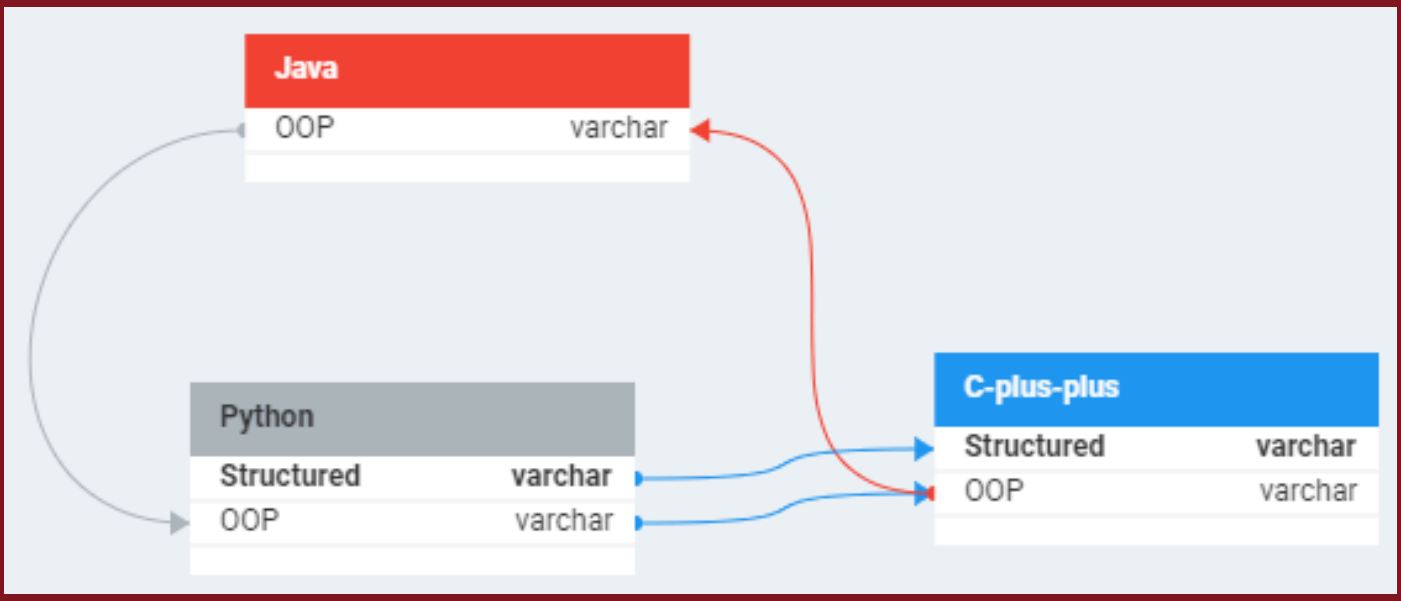
# MS Access Databases

## Theory & Practice for Beginners

CPUcademy

# Designing Databases



## Basic concepts

A **database** is an ordered collection of data from a specific subject area, organized in a way that facilitates access to it (**traditional databases** were stored as paper documents in files, and it wasn't easy and took a lot of time to access them). Documents describe the database objects (which can be people, books, cars, etc.) and also **the relationships** between the objects, e.g., who owns the car or borrowed the book.

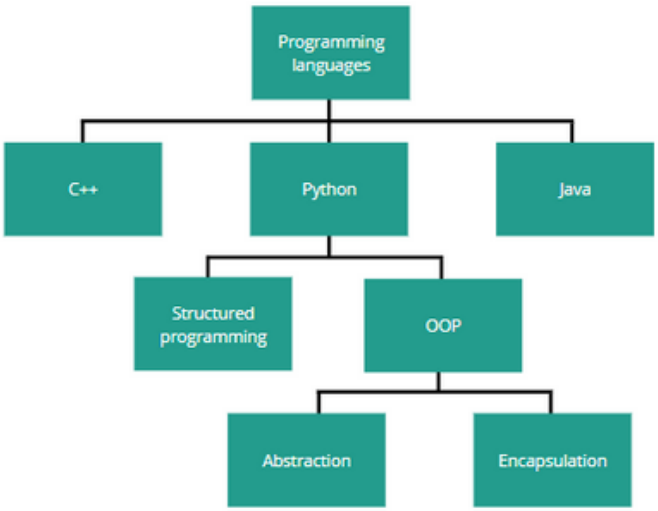
## Computer databases

**The advantages** of computer databases are quick information search, easy calculations, the ability to store large amounts of data in a small space, and fast data organization. **DBMS** (DataBase Management System) is a program that manages data in a database and enables its processing, e.g., MS Access or Libre Office BASE. **RDBMS** stands for Relational DataBase Management System (MS Access is a RDBMS). **The database system** consists of the database itself and the DBMS. We should design the database so that the user for whom the database is intended does not have to know anything about databases and can still operate it without any problems. To achieve this, the database must have a **solid design** that determines the user's needs for collecting, storing, and processing data and defining the activities that make up the operations of the database.

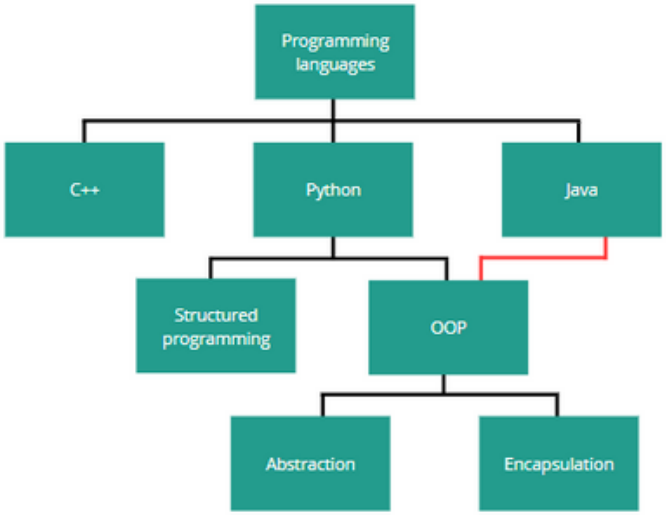
## Database models

A database model is an abstract description of how data is represented and used. **It consists of** structure, constraints, and operations. There are four models:

**1. Hierarchical model** - data has the structure of an **inverted tree**. Every item, except the main one, is connected to a **parent object**. A child object can only be created once the parent object exists, and deleting a parent object results in removing all its associated **child objects**.

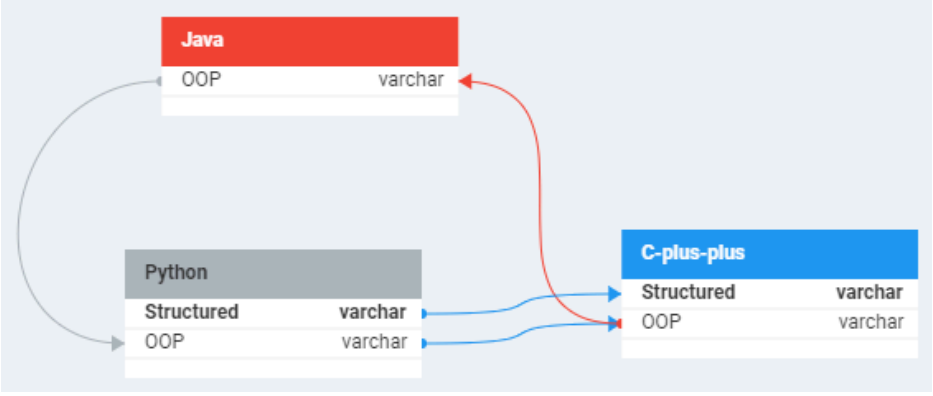



**2. Network model** - this is a modified version of the hierarchical model in which connections between objects create a network, **connections at the same level** of the data tree are allowed, and information is contained in documents and in the course of network connections.




**3. Object-oriented model** - it combines the features of computer programs created in object-oriented programming languages with the features of database applications. Data is made available in the form of objects, and their state and behavior are described using tools available in **object-oriented programming** (properties, methods, object classes). Objects are used to store data and are handled using the same methods. The properties are instances of the same class.

**4. Relational model** - it is based on a mathematical model of data organization and the concept of relationships. Data is represented in the form of tables (entities). Tables consist of **records (rows)** with the same structure. Tables create connections between themselves. These connections are called **relationships**. Due to its functionality, the relational model is most often used in database design. The creator of the theory of this model is Edgar Frank Codd (1970).



## Relational database model according to E. F. Codd

The fundamental database object (table) is a representation of relationships in terms of mathematical concepts, but one relationship can be mapped using many different tables. In this theory, table concepts (**column and row**) are replaced by **attribute and tuple**. In the relational model, it is assumed that the order of columns and rows in tables is **irrelevant** and that rows containing the same data are **identical**. At the intersection of the row and the column, there is a **field** that contains the smallest, indivisible value, i.e., the part of information that cannot be further divided for reasons of **logical consistency**. A set of attributes is called a **relation schema**, e.g., the Merchandise relation schema will be {merch_id, name, price, quantity}. Each attribute is assigned to a domain (a set of allowable attribute values). The domain of the relationship is the sum of the domains of all attributes.

### Key

The key of the relation schema is a set of attributes that can be used to uniquely identify a tuple (a tuple is a record). A given scheme may have several keys (**candidate keys**). Among the keys, one is selected and becomes the **primary key**.

### Data integrity

Data integrity means the correctness of the database structure (compliance with the schema) and the correctness of the data stored in it. Types of integrity constraints:

- **Entity (table) integrity** - each relationship schema has a primary key, and no element of the primary key can be left empty (NULL).
- **Referential integrity** - each foreign key value is equal to the primary key value of the specified tuple in the parent relationship or is NULL (undefined).
- **General constraints** - additional conditions regarding data correctness are specified by users or database administrators.

## Relational database model

The basic form of data storage is a **table**. Information is stored in these tables, which consist of **rows (records)** and **columns (attributes)**. They create connections between themselves called **relationships**. The combination "primary key value, table name, column name" gives access to **any data**. **NULL** (missing information) must be supported. **Data integrity** should be a natural feature of database design. **The benefits** of using this model include the efficiency of data storage, the certainty of data integrity (consistency), the possibility of expanding the database, the ability to easily change the database structure, and increasing the speed of data access. **The description** of such a model includes:

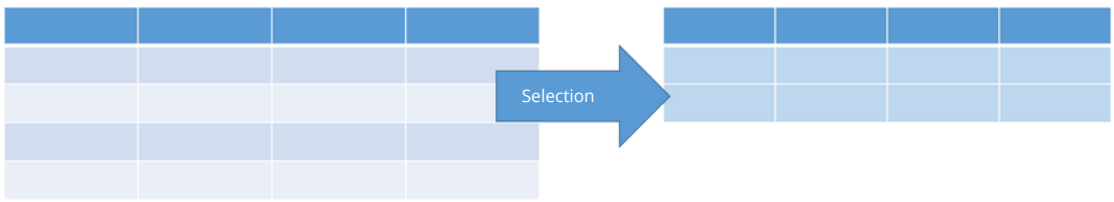
- Data structures - how and according to what principles data storage will be organized and according to what principles we will design it.
- Data manipulation language - how data in the database will be downloaded, saved, modified, and deleted.
- Data integrity - how the correctness of data storage will be ensured.

## Entity

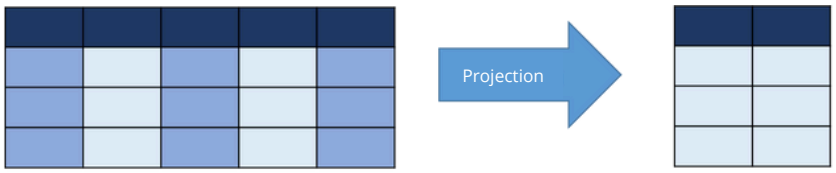
An entity is a **clear-cut element** of the real world or an imaginary object. Its specific occurrence (**instance**) must be assigned only to this one specific entity. If we want to create an entity to which, e.g., John Smith and Sara Jones, who are employees of a certain company, will be assigned, it will be called "Employees." If we want to collect data about, e.g., BMW and Mercedes, we will create a "Cars" entity.

### Types of Relational Algebra

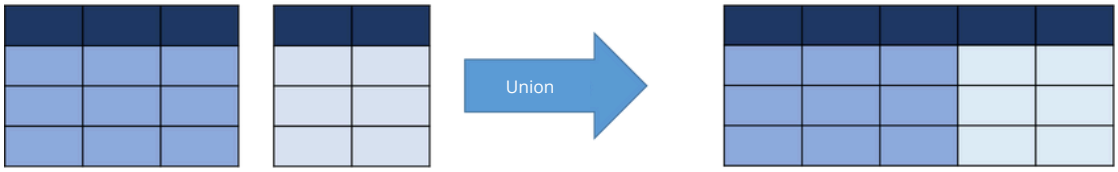
**Selection operation** - it selects tuples (from a relation) that satisfy the provided predicate.




**Projection operation** - it projects those column(s) that satisfy any given predicate.




**Union operation** - it performs binary union between two tables.




## Creating a conceptual model of the database

Conceptual database design means constructing a **data schema** regardless of the selected data model, target database management system, application programs, or programming language. Entity relationship diagrams are used to create a graphical model of a database, most often the so-called **ERD diagrams** (Entity Relationship Diagrams). Such a diagram is modeling reality using three elements: entities, their attributes, and relationships between entities. To create these diagrams, we use **CASE** (Computer Aided Software Engineering) tools. An example of this tool is *https://erd.dbdesigner.net/*.

## Attributes

An attribute is a **property of an entity**. It is of a specific type representing a specific value, e.g., a "Car" entity could have attributes such as brand (text), year (date), mileage (int), or "accident-free" - which is a logical value. The basic **attribute types** are:

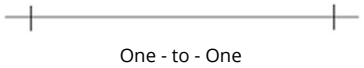
- ShortText - a string of characters,
- Number - numerical data,
- Date / Time - date and time,
- AutoNumber - a unique serial number,
- Bool - logical type (true or false).

## Relationships between sets of entities - relationship diagrams

Each relationship has two ends to which the following attributes are assigned: name, relationship degree, and whether a relationship is mandatory or optional. Degrees of relationships (types of relationships):

optional relationship

mandatory relationship



One - to - One

One - to - Many

e.g. one client has many orders

Many - to - Many

Many - to - One

## Relationships

A relationship is a logical connection between database tables. They allow you to design the database so that it consists of many thematic tables, and each piece of information is saved only once. There should be **no redundancy** (unnecessary repetition).

## Basic definitions

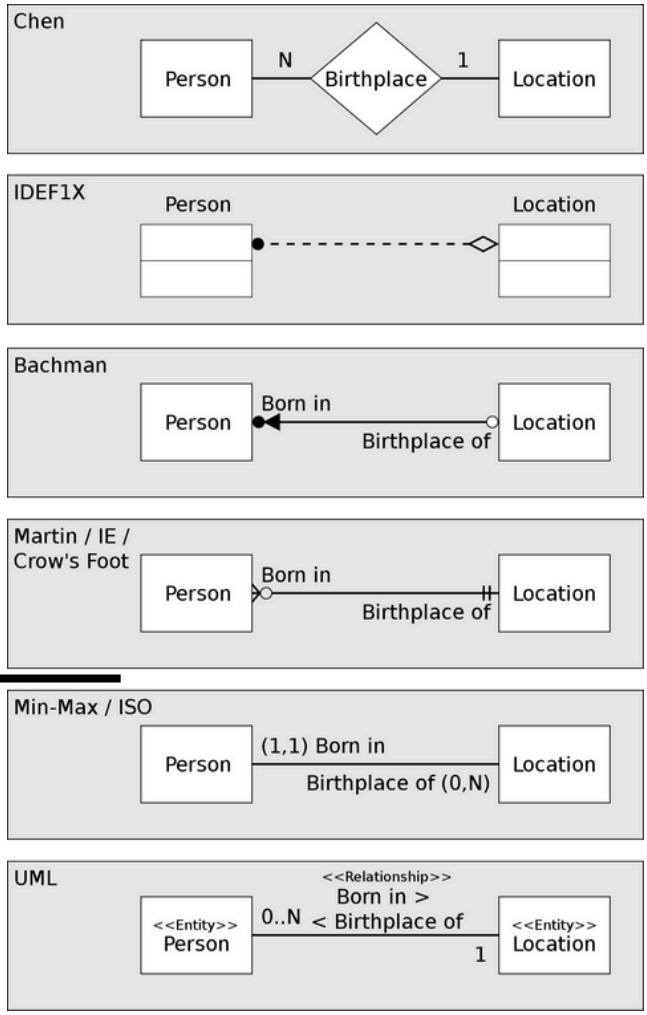
- Entity - an object, a state of phenomena, a concept, i.e., any object that we can distinguish from others, e.g., a person, a car, a book. Items that are similar to each other constitute a set of entities.
- Attributes - characteristics of the entity.
- Domain - a set of values that a given attribute can take.

We should define a **purpose** (what entities we will be dealing with), **objects** (what attributes individual entities have), and **object attributes** (what domains can take particular attributes).

A row is a record or a tuple. A column is an attribute or a field (a field is a column in a specific record). The column contains the characteristics of an entity.

## Entity relationship diagrams in various notations

A notation is a set of symbols and shapes used to describe tables and relationships in ERD diagrams.




## Tables in the relational database model

- Each table must have a unique name.
- Each column must have a unique name within the table.
- All values in a column must be of the same type.
- There cannot be two identical rows in the table.
- A table cannot exist without rows.
- The data stored in the table is based on simple types.
- The order of rows and columns does not matter.

## Primary and foreign key

- Primary key - a field (or minimum combination of fields) that takes on a different, unique value for each record. This key uniquely identifies the record. It cannot contain repetitive data and cannot be empty.
- Artificial key - a field containing a unique identification number. It functions as a primary key (it is a value created to serve only this purpose).
- Foreign key - a field that refers to the primary key of another table. It is used to create relationships between tables.

## Rules of creating relationships between tables

- Shared fields must have the same data type and field size.
- The shared field on one side must be a primary key field.
- Linking fields must contain information that matches each other.
- Shared fields do not have to have the same names.

## Relationships between tables

- They help you create queries (questions to the database) based on the data from multiple tables.
- They help you create forms and reports based on multiple tables.
- By enforcing referential integrity, they prevent detached records from being created.

## Stages of database design

- database planning,
- creating an ERD diagram,
- transformation of the conceptual model into a relational model,
- database normalization process,
- selection of structures and definition of access rules to the database.

## Table design rules

A separate table is used to describe each set of similar entities. One row corresponds to one entity. An attribute corresponds to a column. An information type is specified for each attribute.

## Database Designing

- defining the purpose of the database,
- determining the scope of information needed,
- dividing information into tables,
- dividing information elements into columns,
- selecting primary keys,
- application of normalization rules,
- improving the design,
- creating relationships between tables.

## MS Access RDBMS

### Access objects

- Table
- Query
- Form
- Report
- Macro
- Module

### Table field properties

- Field size
- Format
- The decimal place
- Input mask
- Default value
- Validation rule
- Required
- Indexing

### Data validation

- Default value
- Validation rule (for single fields and records)
- Input mask

### Default value

The default value will appear automatically with each new record. It can be a constant or an expression. A default value cannot be specified for AutoNumber and OLE Object fields.

## Table normalization

Normalization is used to check whether the designed tables have the correct structure. **The benefits** of normalization are:

- eliminating the problem of data repetition (redundancy),
- database volume optimization,
- optimizing efficient database operation,
- minimizing the risk of errors when entering data.

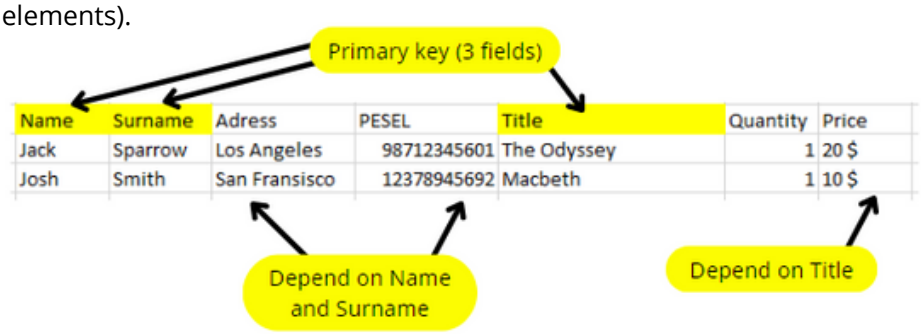
## Normalization rules

(The table examples provided do not meet the given rules.)

**The first normal form (1 NF)** - a table is in the first normal form when a single table field contains elementary information (one piece of information, e.g., not two book titles).

| Name | Surname | Adress | PESEL | Title | Quantity | Price |
|---|---|---|---|---|---|---|
| Jack | Sparrow | Los Angeles | 98712345601 | The Odyssey, The Iliad | 2 | 20 $ |
| Josh | Smith | San Fransisco | 12378945692 | Macbeth | 1 | 10 $ |

**The second normal form (2 NF)** - a table is in the second normal form if it is in the first normal form, and each of the fields not included in the primary key depends on the entire key, not on its parts (when a key is made up of several elements).




**The third normal form (3 NF)** - a table is in the third normal form if it is in the first and second normal form, and each of the fields not included in the primary key carries information directly about the key and does not refer to any other field.

| Name | Surname | Adress | PESEL | InvoiceNr. | PaymentMethod | DateOfIssue |
|---|---|---|---|---|---|---|
| Jack | Sparrow | Los Angeles | 98712345601 | 1 | card | 01.05.2024 |
| Josh | Smith | San Fransisco | 12378945692 | 2 | cash | 02.05.2024 |

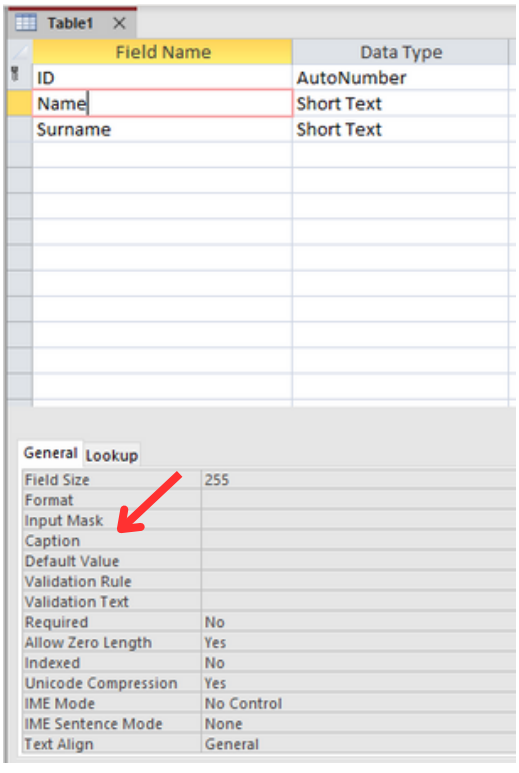
Depend on Name and Surname

Depend on InvoiceNr.

## Input mask

It can be defined for fields of the following types: Number, Date / Time, and Text. It is often used for entering postcodes and telephone numbers. You can enter it while editing a field. It can be used for three purposes:

- forcing the entry of characters of a specific type,
- automatic appearance of characters,
- limiting the number of characters that can be entered.



### Input mask symbols and examples

| Symbol | Meaning |
|---|---|
| 0 | A number must be entered, and the + and - signs cannot be used. |
| 9 | A number does not have to be entered, and the + and - signs cannot be used. |
| # | A number does not have to be entered, and you can use the signs + and -. |
| L | A letter must be entered. |
| ? | A letter does not have to be entered. |
| A | A letter or a number (an alphanumeric value) must be entered. |
| a | A letter or a number (an alphanumeric value) does not have to be entered. |
| & | Any character must be entered. |
| < | It converts the following letters to lowercase. |
| > | It converts the following letters to uppercase. |
| ! | It forces the user to fill the mask from right to left. |
| \ | It sets the next character as a one not treated as a mask code, but as a regular character. |

| Field | Mask | Meaning |
|---|---|---|
| Date | 00/00/00 | Each zero must be replaced by a number. |
| Date | 90/90/00 | Fields marked with a nine can be left blank. |
| ID number | >LLL0000000 | The ID number consists of two parts - a series marked with capital letters and the actual number consisting of seven digits. |
| Surname | >L<?????????????? | In the surname field, the first letter should be uppercase, and the rest should be lowercase. You can enter a maximum of 15 characters. |
| PESEL | 00000000000 | The mask will force you to enter all eleven digits. It will work for a text field. |
| Postcode | 00\-000 | The postcode will have 6 characters, including 5 mandatory digits, and the third position will contain the "-" sign. It will work for a text field. |
| Symbol | 00"ABC"000 | The symbol field contains two mandatory digits, then the text "ABC," and then three mandatory digits. |

We import external data to Access via *External Data / New Data Source / From File*.

### Indexes

If a table field is often used for searching or sorting, you can create an index for this field to speed up these operations. Indexes can reduce database performance when adding and updating data values. In some cases, the RDBMS creates an index automatically, e.g., for a primary key field.

### Integrity rules

Integrity rules ensure the correctness and consistency of data in the database. We define them at the level of:

- fields (data types, input masks, data validation),
- tables and relationships (by enforcing referential integrity).

**Referential integrity** is intended to prevent disconnected records and to synchronize references. After enabling "Enforce Referential Integrity":

- You cannot enter a value in a foreign key field if one does not exist in the primary key field of the linked table.
- You cannot delete a record in the primary table if there are records matching it in the linked table (unless you enable the “Cascade Delete Related Records” option when creating the relationship).
- You cannot change the primary key value if doing so would result in creating disconnected records (unless you enable "Cascade Update Related Fields" when creating the relationship).

### Validation rule (Check)

A validation rule can be defined for a single field or the entire record:

- field validation rule, e.g., “> 100 And < 200”,
- record validation rule, e.g., “[Employment_date] > [Birth_date]”.

You can define validation rules yourself or using expressions. You can type the field validation rule in the properties of the field. I will show where you can set the one for an entire record on the next page.

### Field validation rule

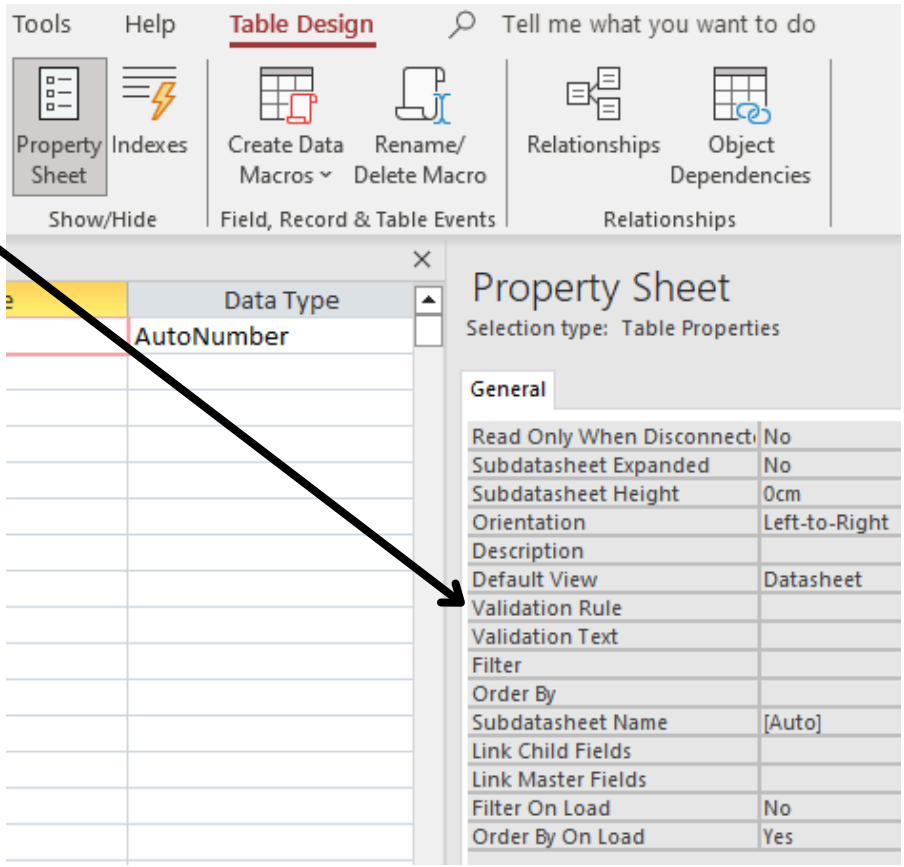
The field validation rule allows you to check the values entered in the field (the checking takes place when **leaving the field**). The field validation rule definition for the “schoolGrade” field (with number type) would be “>= 1 And <= 6”. It restricts typing to integers from 1 to 6. The field validation rule definition for the Date / Time field could be “>=# 2022-01-01# And <= #2022-12-31#”. It limits entry to dates from 2022. When specifying a field validation rule, you can specify your error message (attribute: validation rule text), e.g., “schoolGrade” might read “Please enter an integer between 1 and 6.”

**Record validation rule**

A record validation rule specifies the conditions that must be met to save the entire record (the checking takes place when **leaving the record**). A record validation rule can refer to different fields in the same table. This rule is entered in the table properties.

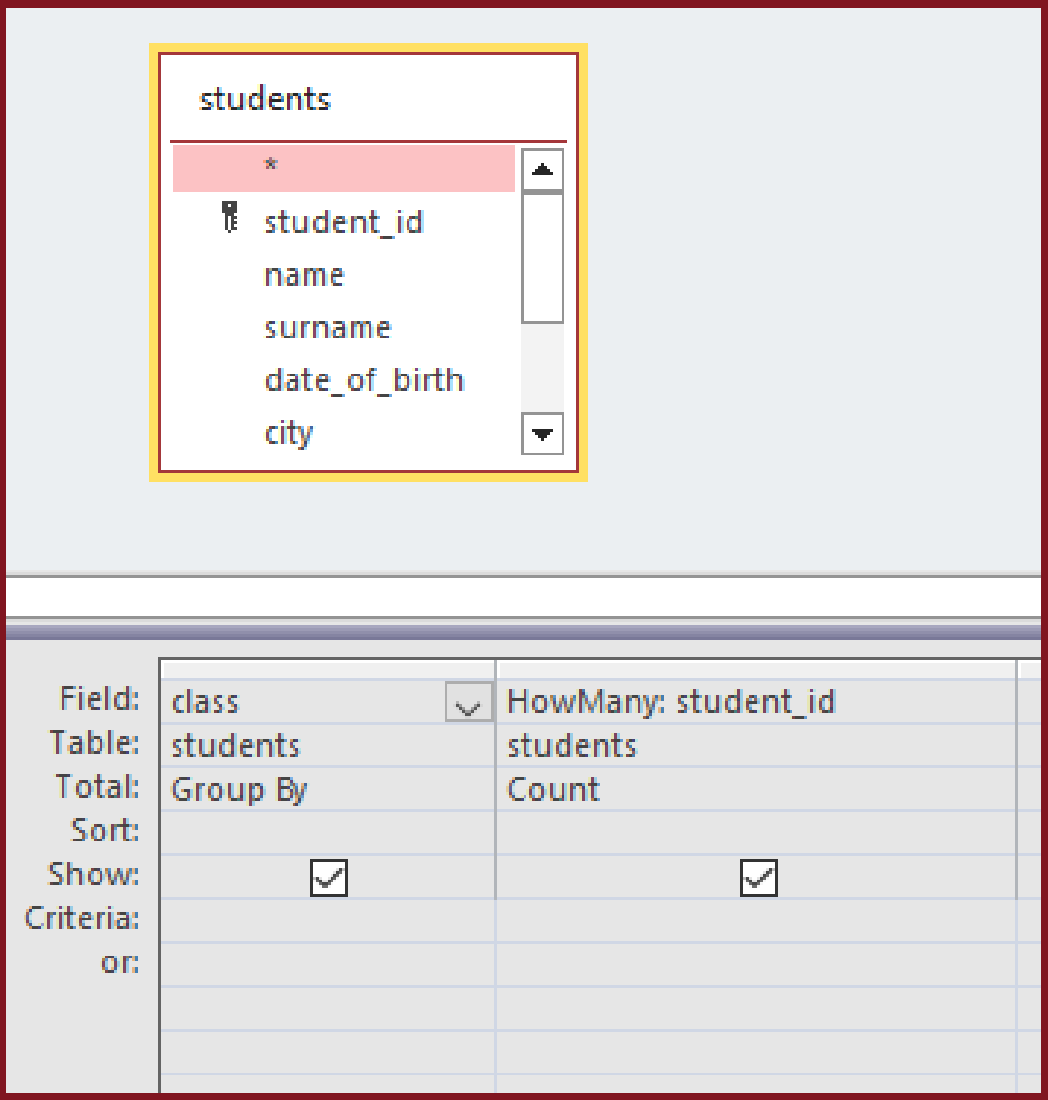
**Designing Databases in Practice**

Now, it is time to show you how to use MS Access and apply all the knowledge you have learned. You will find a short tutorial here: *https://www.youtube.com/watch?v=fBE8WfiTrpc*.




| Most used data types in MS Access | Values |
|---|---|
| ShortText | Set by default. It is a combination of text and numbers or numbers that do not require calculations (for example, telephone numbers). Size up to 255 characters. Note: Access does not reserve space for unused parts of a field. |
| LongText | It is a long text or a combination of text and numbers. When entering data manually, a maximum of 65,535 characters can be entered and displayed. |
| Date / Time | Date and time values for years between 100 and 9999. Size 8 bytes. Regardless of how date and time data is formatted, Date/Time fields store dates and times as double-precision floating-point numbers. |
| AutoNumber | A unique serial number (incremented by 1) or random number that Microsoft Access assigns when a new record is added to a table. The AutoNumber field cannot be updated. A type typically used in primary keys. Size 4 bytes. |
| Yes / No | A unique serial number (incremented by 1) or random number that Microsoft Access assigns when a new record is added to a table. The AutoNumber field cannot be updated. A type typically used in primary keys. Size 4 bytes. |
| Number | Numerical data used in mathematical calculations. Size 1, 2, 4, 8 or 16 bytes. Size property settings for the Number type and their values: **Byte** - stores numbers from 0 to 255. Size 1 byte. **Integer** - stores integers from -32,768 to 32,767. Size 2 bytes. **Long integer** - default setting. Stores numbers ranging from -2,147,483,648 to 2,147,483,647 (no fractions). Size 4 bytes. **Single precision** - Stores numbers between -3.402823E38 and -1.401298E-45 for negative values and 1.401298E-45 and 3.402823E38 for positive values. Size 4 bytes. **Double precision** - stores numbers from -1.79769313486231E308 to -4.94065645841247E324 for negative values and from 4.94065645841247E-324 to 1.79769313486231E308 for positive values. Size 8 bytes. **Decimal** – numbers between -1028-1 and 1028-1. Size 12 bytes. **Replication ID** - a unique global identifier required for replication. Size – 16 bytes. |
| Currency | Financial values, calculations accurate to 4 decimal places. Size 8 bytes. |

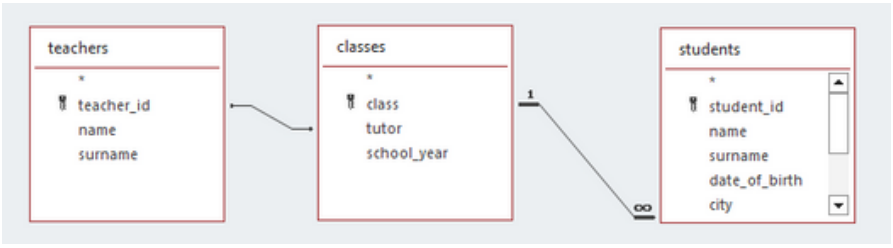
# Queries

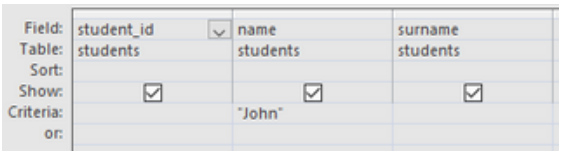

students
*
student_id
name
surname
date_of_birth
city
Field:
Table:
Total:
Sort:
Show:
Criteria:
or:
class
students
Group By
HowMany: student_id
students
Count

## Basic concepts

A query is simply **a request** for information from a database. We can give criteria to the requested data, sort and modify them, or write expressions that combine the extracted data. In MS Access, we can create them through **Query Design** or **Query Wizard**. In the first one, we set everything manually, and in the second one, we tell it what we want to do, and it designs it for us. Every query has a **Design View**, in which we can design it in graphical windows provided by MS Access, a **SQL View**, where we can write it by hand in SQL, and a **Datasheet View**, where we can see the result.

When we create a query in Query Design, we choose the tables that we want to extract data from. If we import more than one table, and they have connected fields, we have to make **connections** like when creating relationships between tables. Then, we can add the fields we want by clicking on them twice. They will show in the window below, where we can choose if we would like to **sort** by their values (ascending or descending). We can also tick if we wish for them to be displayed because we can, e.g., sort by them without showing their values. Most importantly, we can write **criteria** by which particular records will be displayed (if a field's value is accordant to it, the whole record will be shown). Of course, many fields can have criteria, and only if all of them are fulfilled, the record will be displayed.




## Criteria

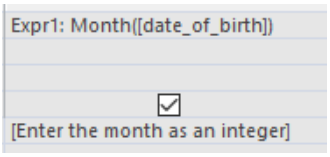
If we write just a text, e.g., **"John"**, it means we want to see only the records that have this value in the chosen field. We can use the **Or** logical operator: **"John" Or "Noah"**. If we want to display all the records that do not have **"John"** in the chosen field, we write **<> "John"**. We can compare if a value of a field is greater / lesser than something, and if yes, display the record (e.g., **>175**). To check if a field is empty or not, we use: **Is Null / Is Not Null**. In the first case, if the field is empty, the record will be displayed. While working with **Data** type fields, we can, e.g., check if a date is in the second half of 2002 like this: **Between #01.07.2002# And #31.12.2002#**. An important criterion is **Like**. It is a function that can be used to specify a lot of things (* - a string, ? - one character, # - one number):

- starts with "R": **Like "R*"**
- doesn't start with "R": **Not Like "R*"**
- starts with a letter from "A" to "D": **Like "[A-D]*"**
- doesn't start with a letter from "A" to "D": **Like "[!A-D]*"**
- ends with "R": **Like "*R"**
- the second letter is "A": **Like "?a*"**
- has five characters: **"?????"**

MS Access docs about **Like**: *https://support.microsoft.com/en-gb/office/use-like-criterion-to-locate-data-65b07c8a-b314-435a-8b48-2b911856d4f9*
MS Access docs about **criteria**: *https://support.microsoft.com/en-us/office/examples-of-query-criteria-3197228c-8684-4552-ac03-aba746fb29d8*

## Expressions

Instead of the field name, we can enter an expression. We can use predefined functions in them. In the example below, the expression uses the **Month()** function, which takes the **date_of_birth** column as an argument (the square brackets mean it is a column). The criterion is **[Enter the month as an integer]**, so upon entering the Datasheet View, a window with this instruction will pop up, and we will have to type in the month. The query will show everyone who was born in that month. Of course, we can enter "normal" criteria, e.g., 6.




I prepared a file with **over 50 labeled queries for you to try out**, and you can find it here: *https://drive.google.com/file/d/1Frusx3YlHtZ0KMW366WGys9YT8F1e-L2/view?usp=sharing*.

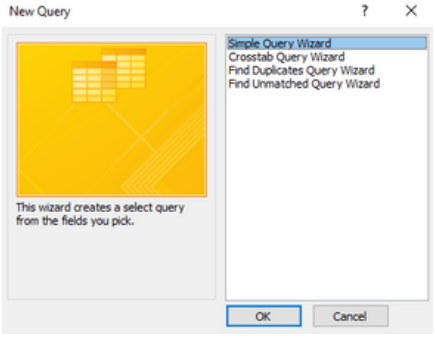
An expression doesn't have to incorporate a function. It can just, e.g., create an email for every user: "**email: [name] & "." & [surname] & "@gmail.com**" ("email" is the name of the expression). It doesn't have to have any criteria. Remember that **&** is a concatenation operator, and **And** is a logical operator (you can find a full list of operators here: *https://support.microsoft.com/en-gb/office/table-of-operators-e1bc04d5-8b76-429f-a252-e9223117d6bd*).

In the file that I've mentioned above, you will find the application of many functions, but if you wish to know more, check the docs: *https://support.microsoft.com/en-au/office/access-functions-by-category-b8b136c3-2716-4d39-94a2-658ce330ed83*.
If you don't understand what a given function does or what parameters it takes, you can right-click on the expression and choose **Build**. In this window, you have a list of all available functions, their descriptions, and parameters. If you just wish to see what you are writing, choose **Zoom**, and it will open a bigger window with a text field.

An example of an important function you have to know is **IIf()**. It can be used, e.g., like this: **IIf(Right([name];1)="n";"A";"B")**. It means that if the first letter of your name from the right side (the last one) is **n**, then the output will be **A**, and if not - **B**.

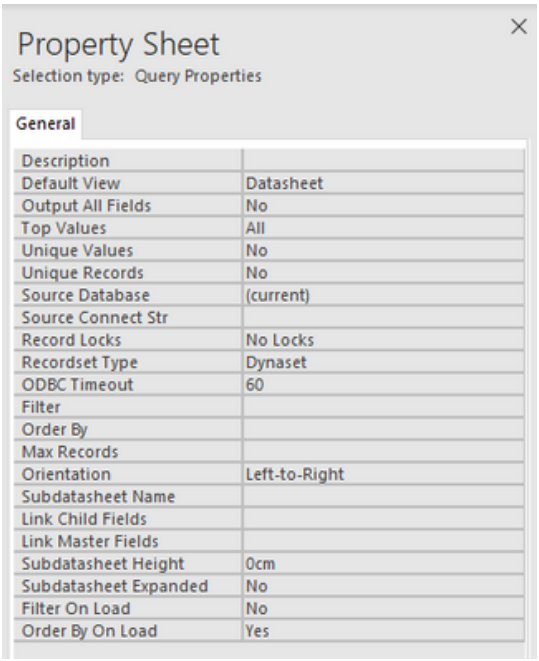
## Query Wizard

I mentioned it at the very beginning. We tell it what query we want to create, and it designs it for us.



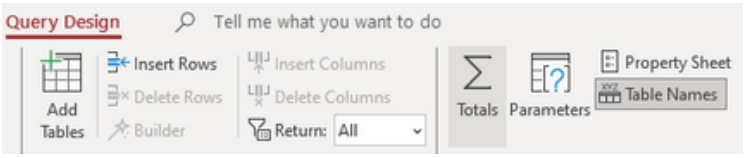
## Property Sheet

We can open a Property Sheet by right-clicking on the space where the tables are in Query Design and selecting **Properties**. We can change a lot of parameters. We can set **Unique Values** (in SQL - **DISTINCT**), e.g. when we are displaying only surnames, we might want to avoid redundancy. We can also set **Top Values** (in SQL - **TOP**), which means we will only display a given number or a percent of results from the top. Every field also has its own Property Sheet (we used the one for the whole query). We can access it by clicking on the name of the field in the table at the bottom. If the field is numerical, we can set, e.g., how many decimal places we want to see.




## Totals

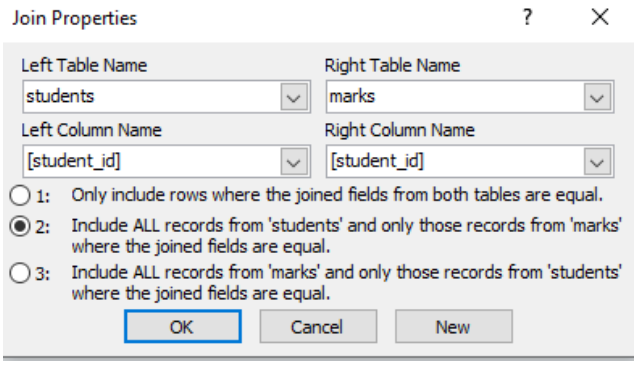
Activating the **Totals** will create a new row in the table where we make queries. We will be able to choose five options. **Sum** will sum all the elements from this column, **Avg** will calculate the average, **Min** and **Max** will find the minimum and maximum number, and **Count** will display the number of records (if a field that has this mode is empty, the record will not count). We can also select **Group By**, which means the records will be sorted according to these values.




| Field: | Sum: weight | Average: weight | Minimum: weight | Maximum: weight | Count: weight |
|---|---|---|---|---|---|
| Table: | students | students | students | students | students |
| Total: | Sum | Avg | Min | Max | Count |
| Sort: | | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ | ☑ |
| Criteria: | | | | | |
| or: | | | | | |

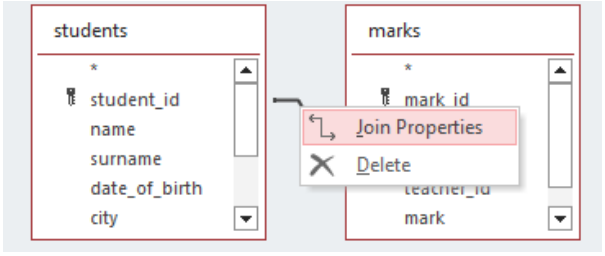
## Join Properties

When we right-click at a connection between two tables in a query, we can select **Join Properties**. There, we can choose which records we want to include in the query.

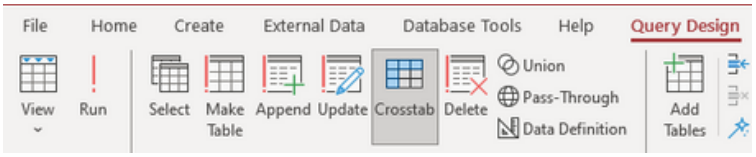



## Special queries

- A **Self Join** query is a query that has a table that is connected to itself.
- A **Subquery** is a query that is written as an expression inside of another query: (e.g., Chairman: (SELECT name & " " & surname FROM students AS u1 WHERE students.class=u1.class AND u1.function="Chairman")).
- A **Crosstab** is a query that displays data in a different format and order.




| Field: | class | Year: Year([date_of_bi | Count: student_id |
|---|---|---|---|
| Table: | students | | students |
| Total: | Group By | Group By | Count |
| Crosstab: | Row Heading | Column Heading | Value |

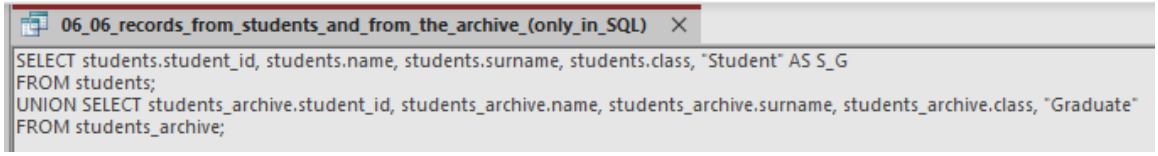



## Updating queries

Updating queries are queries that can modify the data from the tables. We create them by pressing buttons like **Crosstab**. We can design queries that: create a new table (**Make Table**), add data (**Append**), update a table (**Update**), or delete data (**Delete**). We run them by clicking on them twice or through the **Run** button.

## SQL-only queries

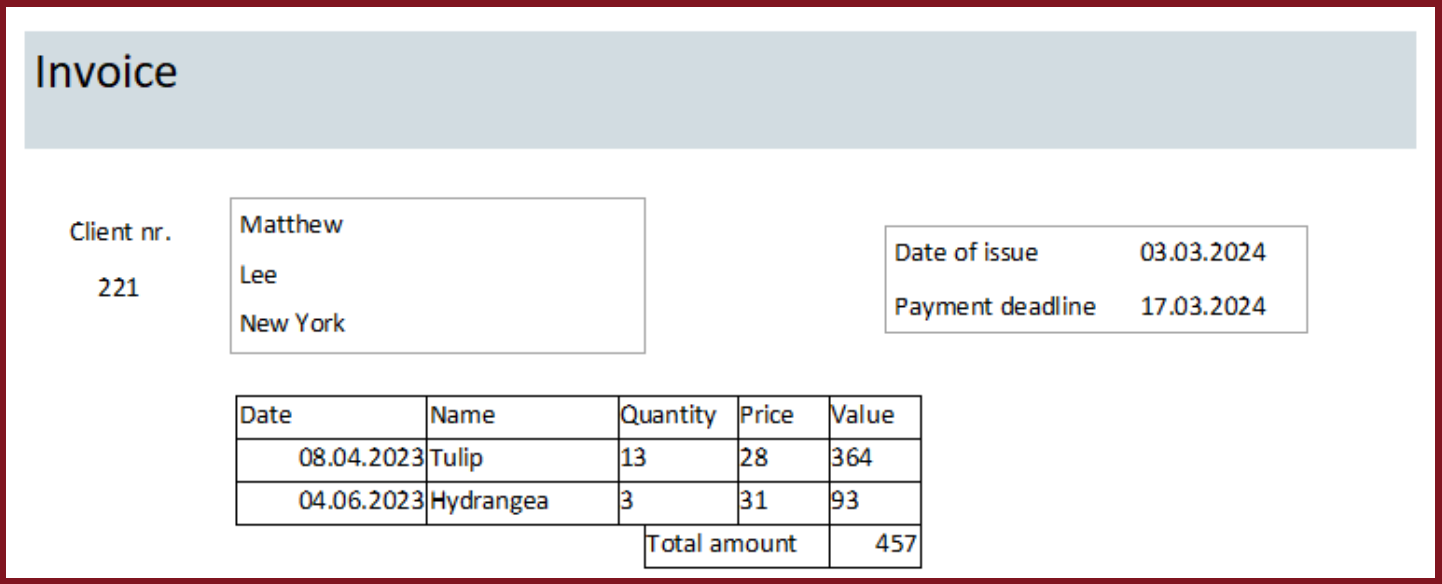
There are queries that do not exist in MS Access but only in SQL. We can use them by switching to SQL View (the Design View will no longer be available). An example of this type of query is a query that displays whole records from different tables. Remember that, in MS Access, a decimal separator is a comma, and in SQL - a dot.




## Queries in Practice

Here, you can find a short tutorial that navigates you through the file mentioned before: *https://www.youtube.com/watch?v=R7UEw3PkCyQ*.
The practice file: https://drive.google.com/file/d/1Frusx3YlHtZ0KMW366WGys9YT8F1e-L2/view?usp=sharing.

# Forms & Reports



Invoice

Client nr.
221

Matthew
Lee
New York

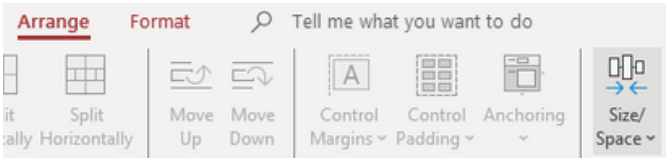
| | |
|---|---|
| Date of issue | 03.03.2024 |
| Payment deadline | 17.03.2024 |

| Date | Name | Quantity | Price | Value |
|---|---|---|---|---|
| 08.04.2023 | Tulip | 13 | 28 | 364 |
| 04.06.2023 | Hydrangea | 3 | 31 | 93 |
| | | Total amount | | 457 |

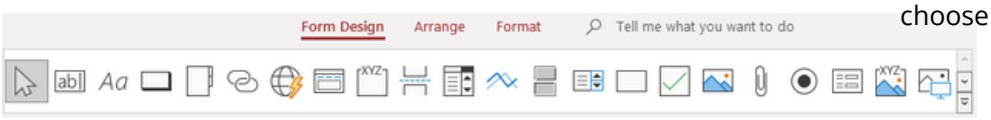
## Basic concepts

A form is a database object that you can use to create a **user interface** for a database application, and a report is a formatted **presentation** of data from a database that provides structured information. We will design them only using Form and Report Wizards. We can attach **Macros** to forms. They are a tool that allows us to automate tasks and add functionalities to forms, reports, and controls. They are coded in VBA (Visual Basic for Applications), but we will use Access's graphical interface.

When we create a form in Form Wizard, we choose the tables we want to use, the layout of the form, and its name. When we move the elements of the form, we can easily align them using the tools from the **Size / Space** tab. When we right-click on one of the sections of our form, we can choose the **fill color** to make it more appealing. At the top, we have a bar with items we can add. A **text box** contains a value taken from a table, and a **label** contains exactly the text we write (it doesn't matter if there is a column with this name). A **button** causes some action upon clicking on it. A **combo box** is a dropdown list. A **Subform / Subreport** is another form / report inside our form (inside the designated rectangle). We can also add **lines** and **rectangles** to categorize our items and make everything look better.



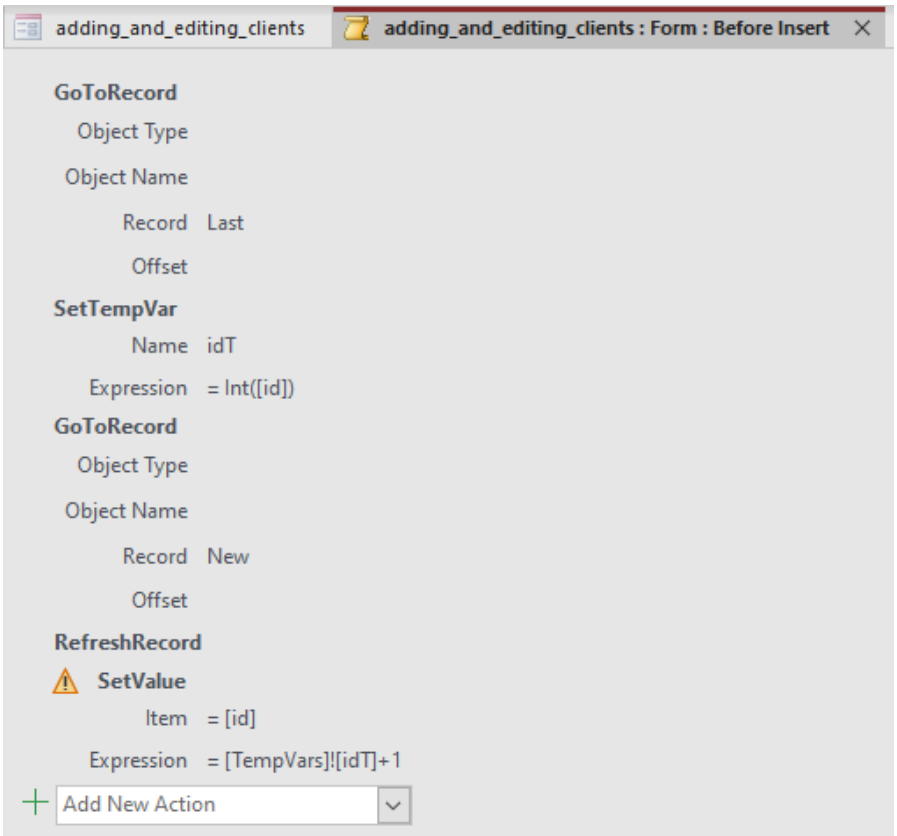

If we use the form to change values in a table, we have to save these changes. For this purpose, we can create a button, select *Form Operations / Close Form*, and name it. It will close the form and update the table. To make a **combo box**, we right-click on a text box, choose *Change to / Combo box*, and in **Row Source** in its Property Sheet add a table or a query, e.g., a query that selects values from which we can choose in this dropdown menu.



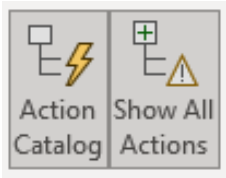

If we want to search for clients from a combo box and then fill the text boxes in this form with their data, we create a new combo box, choose "Find a record on my form based on the value I selected in my combo box", select the fields from which we want to choose, and leave everything else on default mode.

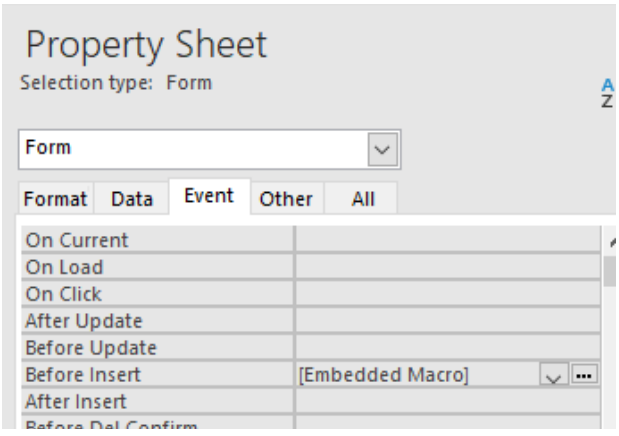
## Macros

When we add a new client to a database, we shouldn't have to know the last index in the table. To automatically assign an ID when entering other data, we can use a macro that will be called on "Before Insert" **Event** (we can set it in the Property Sheet). It looks like this:




To create a macro that uses the "SetValue" command, we have to click "Show All Actions" in the menu above because this function is considered "**dangerous**" (it can change data).

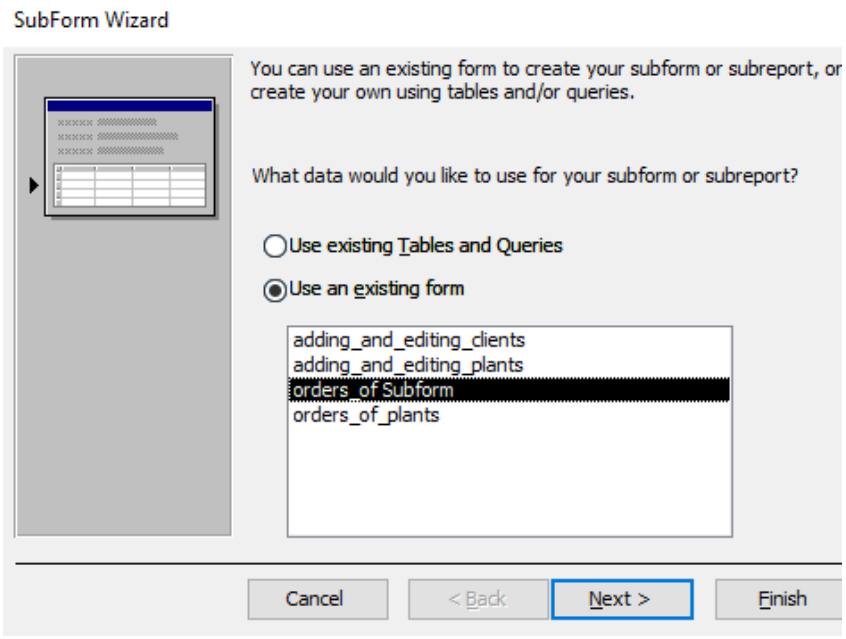


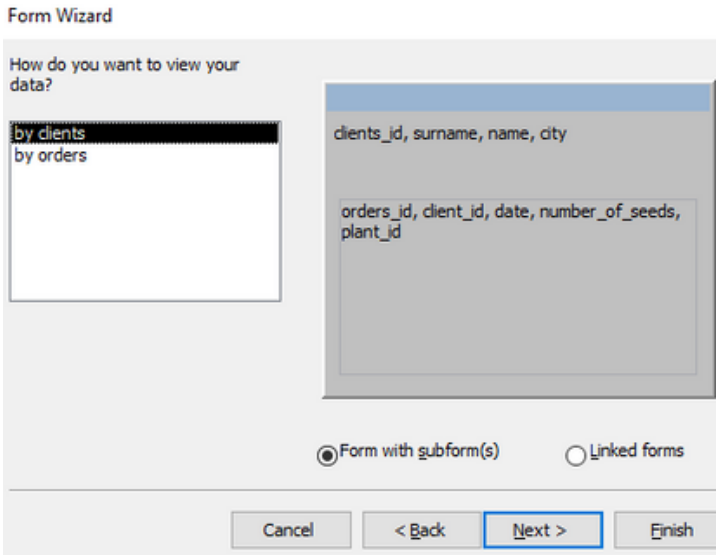




I prepared a file with **forms, reports, and macros for you to try out**, and you can find it here: *https://drive.google.com/file/d/1A3fZVnwLjCmow0B11nbNgOF-tjCbDxUc/view?usp=sharing*.

## Subforms

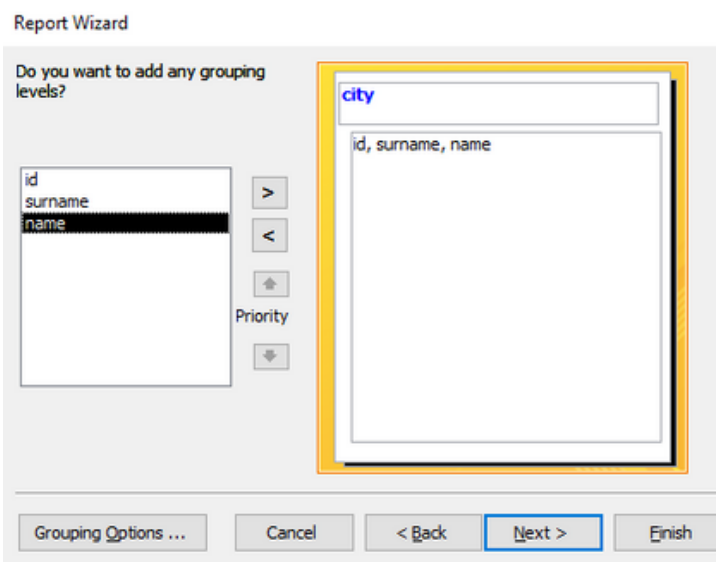
In the Form Wizard, when we select two tables and then mark from which point of view we want to view data (as shown on the right), we will get a form with another form embedded next to it. Its values will change accordingly, e.g., the orders will update in compliance with the viewed client. We can also create subforms by selecting the **Subform / Subreport** item and choosing an existing form (as shown below).



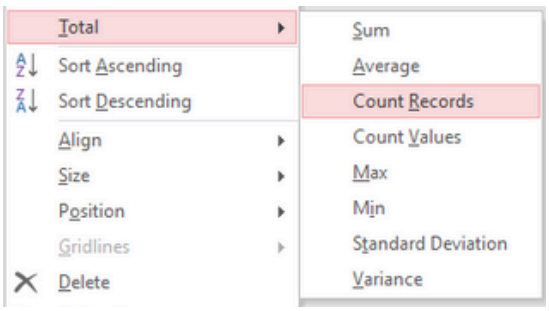



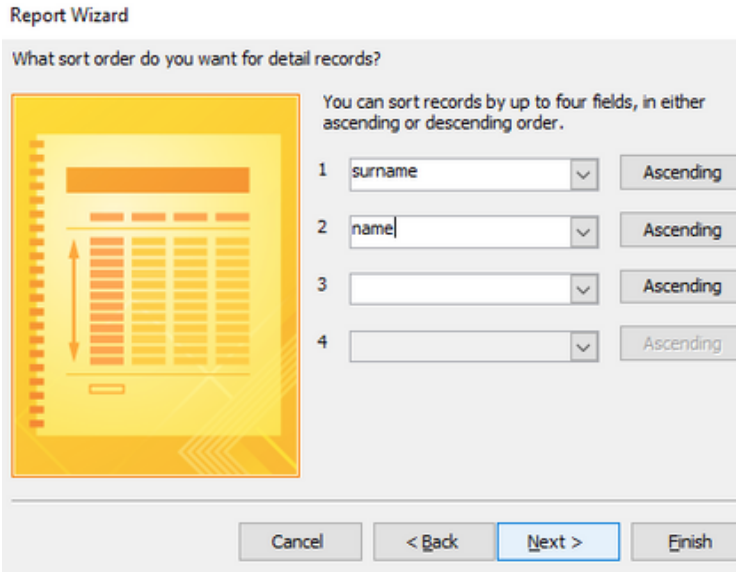

## Reports

In the Report Wizard, after we choose which tables we want to use, we can add **grouping levels**, e.g., clients can be grouped by the cities they live in. Then, we can choose by which records we want to sort these values further.

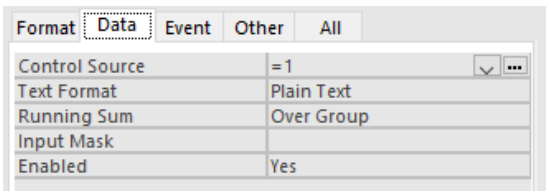



If we want to add a field that contains the count, sum, etc., of other items, we have to right-click on a text box with this value, select **Total**, and choose the desired operation. A new text box with the expression will appear.



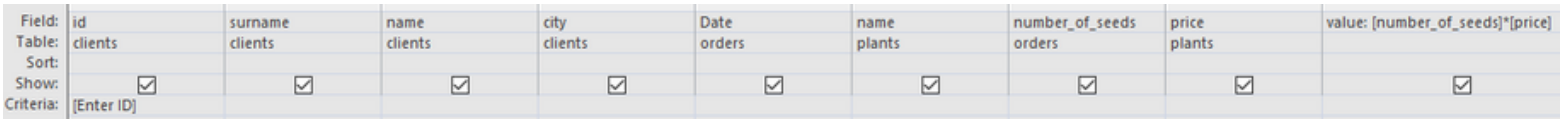



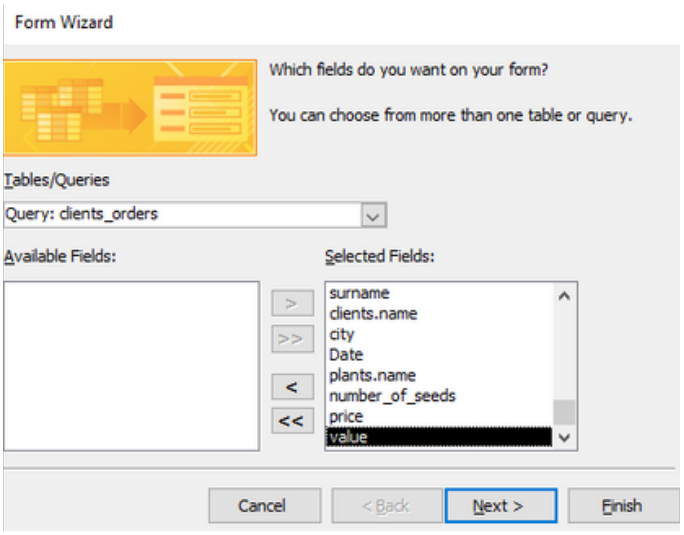

In the file that I've mentioned, there is a report that gathers clients and their orders. It looks nice when the orders are numbered separately, starting from one for each client. To make this happen, we can create a text box, write "**=1**" inside it, and select the values in the Property Sheet, as shown below.

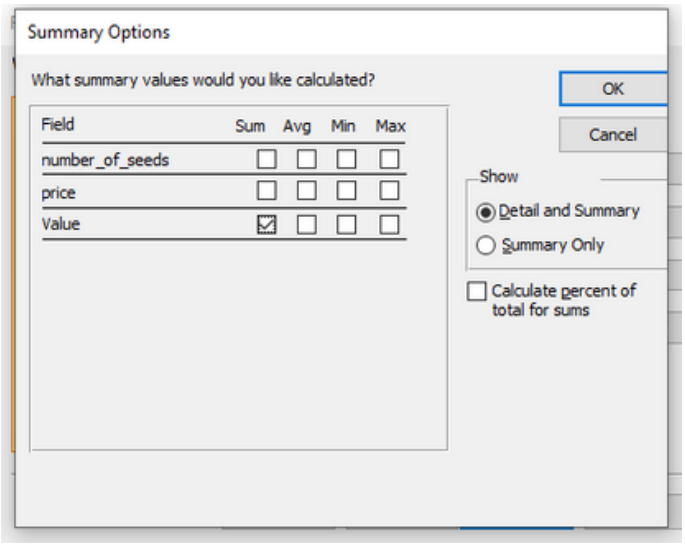


## “Invoice” exercise

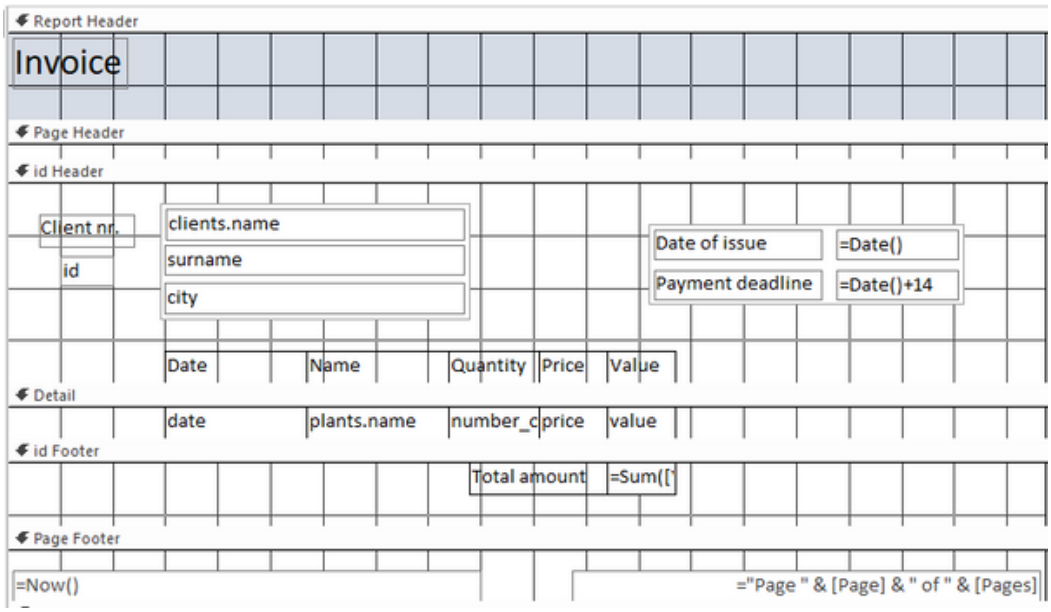
In this exercise, we will create an invoice for a selected client. All of the results of this exercise are in the file mentioned before. First, we have to make a **query** and then use it as a source while creating a report. We will sort by clients, group by clients_id, and sort by date. On the page where we chose sorting by date, in the **Summary Options**, we have to tick the Sum of Value (as shown below).

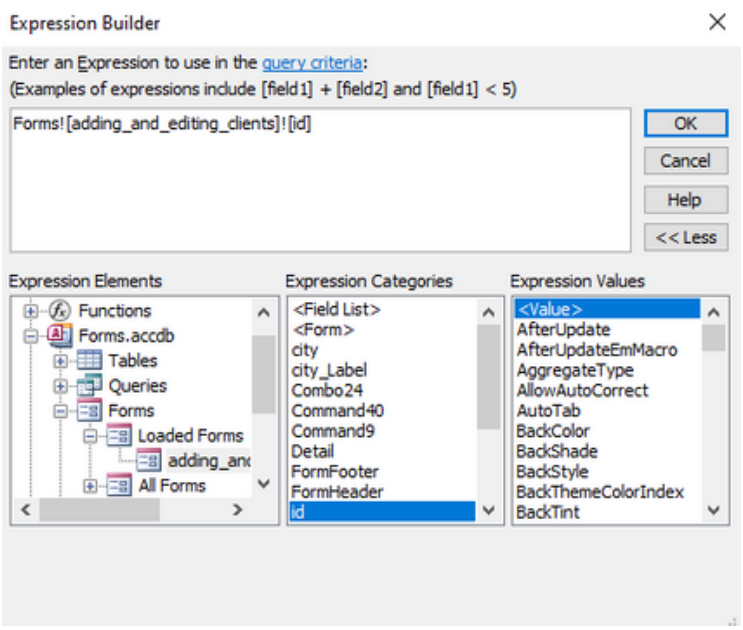



While designing the report, remember that you can delete the frame of an object by changing its **Border Style** in the Property Sheet from Solid to Transparent. Also, to delete the gray color showing up every other line, delete the "Darker 5%" from the **Alternate Back Color** in all sections. After you finish designing, create a button in a form that adds and edits clients, select "*Report Operations / Preview Report*", and choose the one with the invoice. Now, change the criteria of the ID in the query. Use the **Expression Builder**, and choose "*Forms / Loaded Forms / the form in which you added the button*". Then double-click “id” and press "OK". The generated expression will be the criteria. Now, when you click the button in the form, a report with an invoice for this particular client will be created.



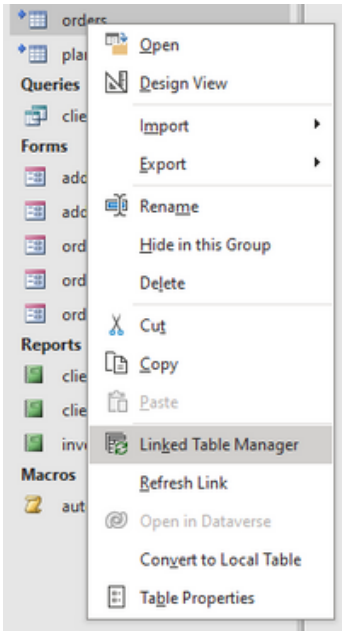




### Forms & Reports in Practice

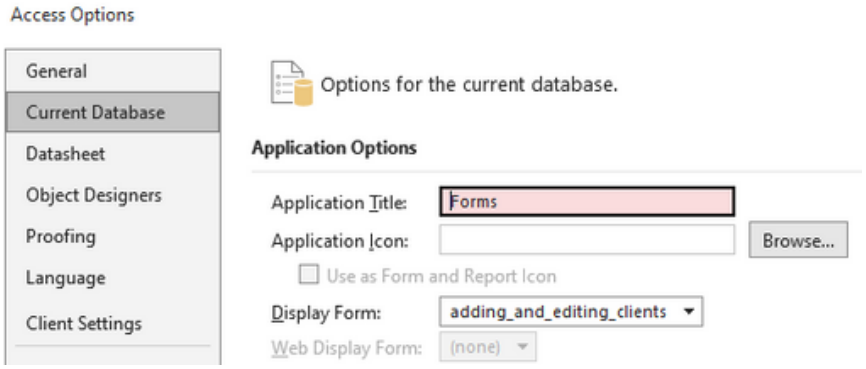
Here, you can find a short tutorial that navigates you through the file mentioned before: *https://www.youtube.com/watch?v=GWmW4O6eM2Y*.
The practice file: *https://drive.google.com/file/d/1A3fZVnwLjCmow0B11nbNgOF-tjCbDxUc/view?usp=sharing*.

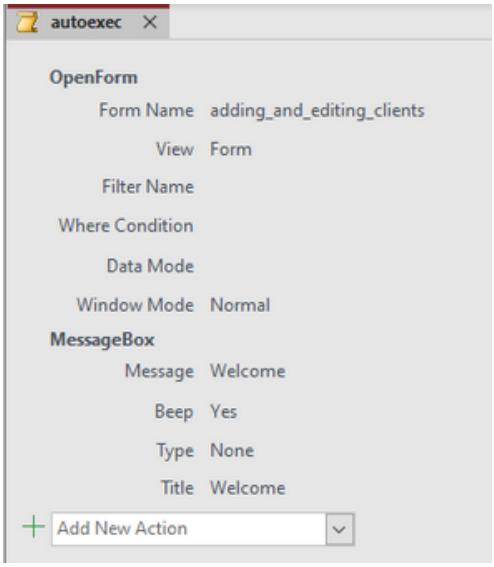
## Creating an application

We can reduce the size of a database file by clicking **Compact and Repair Database** in the **Database Tools** tab. There, we can also derive the database's tables to a different file, which we could place, e.g., in our company's local network, and use the main file only as an application. We can do it by clicking **Access Database**. When we want to change the location of the file with tables, we have to move the file and click **Relink** in the **Linked Table Manager**.


When we choose "*File / Options / Current Database*", we can name our application and select the default form that will open automatically upon entering the file. This form could contain, e.g., our main menu. Also, we can choose an icon for the file. Instead of selecting the default form there, we can also create a macro that does the same thing. The advantage we get by using a macro is that we can also add other things, e.g., a welcoming window. This macro isn't connected to any form (we create it independently in the **Create** tab), and it has to be called **autoexec**.




To add a password to our database, we have to open the file not by clicking on it but by browsing it from the Access application and selecting **Open Exclusive**. Then, in "*File / Info*", we can choose **Encrypt with Password** and type it in. Now, before entering the file, Access will ask us for a password.