

THE RUDIMENTS AND BASIC PRINCIPLES OF DATABASE ADMINISTRATION AND DESIGN

It is essential for the acquisition of the usage of the MS Access to be familiar with the key notions, and terms. Therefore, in this chapter we will examine the above mentioned questions in more detail.

DATA AND INFORMATION

Information is not the same as data, but rather a kind of meaning of the data. Data is, in contrast to information, objective. Database administration is used for storing facts in databases, and to present information in such form that carry information for the user. Therefore, data is understood here as a series of signs that become information during the processing of the data.

The data file is a coherent mass of facts, which includes all the data that are required for the realization of a given goal.

“The datum is a piece of knowledge that can be interpreted (it is perceptible, sensible, comprehensible, and understandable).” (Halassy 1994, p. 8)

“Information is a datum that becomes a newly interpreted knowledge.” (Halassy 1994, p. 9)

THE DATABASE

Under the term **database** we usually mean a group of data that is stored on the basis of a system, which data do not necessarily get stored on a computer. In order to define what a database precisely is we need to get familiar with a few notions.

The Types of Information Handling

First method: we can store our data in different stocks, and establish a connection between them with the help of a programme. For instance, Dbase or Clipper.

Another method is when we store our data in text mode. This is the formulation of knowledge in a text-like way, instead of a data-like form. We call the tool itself a text manager, and the text managing system is a database.

The third option is the **database administration**.

A database is a collection of the different phenomena that we are interested in, in an organised unit. The point is that a database is neither a data bank, nor the unorganised unit of files.

Steps of a mode-based database administration:

- We create the entities along with their properties.
- We establish the relationship between them.
- The rest is dealt with by the database handler.

The **database** is a collection of data, which stores the data required for a given task in an organised way, grants access to them, and at the same time safeguards the integrity of the units, and protects them from any harm.

Base Concept

“We call the thing-that-something that we want to describe with the help of our knowledge an entity.” (Halassy 1994, 24)

The specific entities are known as **entity occurrences**.

“We call the thing-that-something that we use to describe the phenomena, which we are interested in a property.” (Halassy 1994, 28)

The concrete value of a property is the **property value**. The **property value set** is the term used for all the values present in a specific time.

Both entity and property are relative terms. A thing can be both an entity and a property at the same time. It is up to us to decide, which properties should be treated as individual entities.

Primary key: the property of an entity, which takes up a different value for every entity occurrence, is known as the atom or identifier of the entity. It is also known as primary key.

We call the relation between the entities a **relationship**. When it comes to practice, it is not self-evident what kind of relationship we create between the entities, and it is a difficult task at the same time. This is the core of both database designing and its hardships.

THE DATABASE MANAGEMENT SYSTEM

The systems designed to make easier the management of the databases is called **database management systems**. The database management systems are used for recording, storage, and management of the data in a database.

Access is also a database management system. We can use it to make new databases, or add, delete, modify, or search for data in the already existing bases.

THE MAIN FUNCTIONS OF THE DATABASE MANAGEMENT SYSTEMS

- Make new databases
- Define the content of the DBs (databases)
- Store data
- Query data
- Protect data
- Encrypt data
- Handling of access rights
- Synchronise accesses
- Organisation of physical data structures

As we can see, the database management system is software, which grants us access to a database and takes care of the maintenance functions.

THE THREE LEVELS OF THE DATABASE

‘DESIGNER’S BLINDNESS’

Both computing designers and users are tool oriented. That means, they think in the data structure that is supported by their currently used database manager. The problem occurs when they change to a new system. Then they have to start everything over. However, the database has a **device-independent** approach. We differentiate between the **notional**, **logical** and **physical** level of the DB. The structure of the DB needs to be formed in three steps.

THE NOTIONAL AND LOGICAL STRUCTURE OF THE DB

Designing and modifying the DB should always occur on the required level. During the designing phase there are two possible attitudes:

- There are some, who use the file manager type of systems, like dBase, Paradox, or FoxPro. These can be accessed through a file-server structure in a network system. They have, primarily, historical importance.
- There are others, who design more complex systems, and the aim is to achieve a multi-user network environment. They use client-server architecture; in addition, they may further develop it to produce multilayer applications. Such systems are, for example, Oracle, MSSQL, IBM DB2, PostgreSQL, MySQL.

“We call the data structure *conceptual*, when they reflect the phenomena, its characteristics and its relations according to reality, and at the same time reflecting the natural concepts.” (Halassy 1994, 45)

There are many factors that can influence what the structure of the database should look like:

- **Technical factor:** it is often the case that you have to get accustomed to the possibilities of the database manager. The designed data model may not be true to nature.
- **Accessibility:** it may happen that we need to modify a good structure, because of the privacy of the data.
- **Efficiency:** we may need to choose a data structure with a single report instead of one with multi-report, because our database manager supports this structure much better.

Therefore, we call those data structures that meet the technical, accessibility and efficiency criteria **logical structures**. The best solution of realization is when the principal planning follows the conceptual database designing.

THE PHYSICAL STRUCTURE OF THE DB

A database is only acceptable when the physical manifestation matches reality. However, there are some problems that may emerge:

- **Assertion:** the input data must be valid. For instance, if we add a date, its structure and value must be valid.
- **Data presentation:** when we give the type and size of a datum it is called data presentation. There are types like textual, imagery, logical, numerical, etc. These should be handled separately for each has its own executable operation.
- **Organisation of data and way of storage:** the more modern a database manager is, the less attention is required for the way the files are stored.

THE NAMES OF THE PHYSICAL DATA STRUCTURE

“We call the conscious order of storage, access and presentation of knowledge on a store **physical data structure**.” (Halassy 1994, 49)

Data table, file: it matches the definition of the entity. The system handles the data in a table form, and thus we need to think in tables.

Field, column: this is the same as property. We refer to the name of the given property with **field name**, and to the property occurrence with **field value**. Field is also referred to as **column**.

Record, row: same as **entity occurrence**. Record is the value that can be found in a row of a report. These are only concerning one entity.

Elementary item: values appearing in the cells of a table

Entity: that of which we store the data about. We consider a person, for instance, an entity.

Attribute: a property, characteristic feature of the entity. A person’s characteristic feat could be its height.

Entity type: all the properties of the entity as a whole.

Entity occurrence: the concrete properties of the entity. For example, Opel Astra, 10 years old, 1400 cm³, blue coloured

FUNDAMENTAL STRUCTURES

It happens in many cases that an individual data file, or a table does not have enough data to identify certain information. In such cases it may become necessary to handle the data files as a whole, according to a database structure, also known as data model. There are many data models existing. However, only three of them are widely known..

DATA MODELS

Hierarchic data model: it stores the data in a hierarchic structure, which is similar to a tree. All the nodes of the tree represent one record-type. There is a so called 'parent-child' relation between the data. Each datum has as many 'descendants' as you wish, but can have only one 'ancestor'. Its main advantage is being easily describable and easy to make. Today this database is out-of-date.

Network data model: the further development of the hierarchic model. In this model it is possible to establish a system of relations between the data as one please. A datum can have many ancestors. The model's main disadvantages are it's complicated relations and great demand for storage place. It appears in the environment of computers with a great capacity.

Relational data model: we store the data-sets with different types, yet with some common feature in individual tables. Such tables have a field, which contains the same datum that serves as a link between the tables. Currently this is the most widespread data model. This data model is supported by the most widely known database manager programs, like dBase, Clipper, FoxPro, **Access**, Oracle, MSSQL, MySQL.

Object-oriented data model: a reliable database can be designed and produced with object-oriented technology. This model is not so widespread at the moment.

The Access is based on a relational data model; therefore we should examine this model in more detail.

We call those programs that store, organise, and search for data in tables (relations) on the computer relational database managers.

A relational database is a type of database which consists of more than one interconnected tables. The relational database manager system is capable of interconnecting data tables with each other on a logical basis, and search for the common information inside these tables.

In order to call a table a **relation** it needs to meet the following criteria:

- It cannot have two identical rows.
- Each and every column has its own name.
- The sequence of the rows and columns is optional.

The **relational databases** usually contain more than one logically interconnected table.

There is a set relation between the tables. It is very important during the design stage of the process that we construct these relations carefully.

The basis of the structure of the relational database is the **normalisation** (see later), which refers to the method used for determining the optimal place of the data.

THE THREE FACTORS OF THE DATA MODEL

The data model has three factors: **entity, property, relation**. These are all equal members of the data model; therefore none of them is superior to the other.

- The properties of the entity are known as its internal structure.
- The relations of the entity are the entities external structure.

THE ROLE OF THE PROPERTIES

The properties have four roles:

- **Atom/identifier or primary key**: the given property clearly identifies the entity occurrence.
- **Descriptive**: those properties that, considering the entity occurrence, are not unique. Most of the properties of an entity are like this.
- **Switcher or foreign key**: a property that is being identifier in one entity and descriptive in another. It could ensure the relation between the two entities.
- **Super key**: if the relation has one column, which clearly identifies every single record.

The roles of the properties are not of the same importance. “We call the function of the property within the entity its relative role, and the most important relative role its absolute role.” (Halassy, 1994, 75)

Relative means that the task of the property depends on, which entity contains it. Another thing is, a property can have the same relative and absolute role.

“Two entities only have a relationship with each other, if one of them as a switcher property contains the others identifier property.” (Halassy 1994, 76)

Requirements a primary key must meet:

- All entities must have an identifier.
- The identifier's value cannot be empty or unknown in any entity occurrence.
- Every entity can have only one identifier property.
- “The same property can only be used by one entity.” (Halassy 1994, 74)

TYPES OF RELATIONS

1 to 1 relation: in this relation an item of table A has exactly one item from table B that it has a connection with. This relation is quite rare, simply because two entities can be easily merged. It is generally used to fix temporary problems. Also known as mutual relation.

1 to more relation: if to an entity's 1 occurrence the other entity's more than one (N) occurrence can be connected then we call these type of relationships 1:N or 1 to more relation. We call the entity with a linking property that has an identifier role a *superior*, and the one with a switcher role an *inferior*. We also call this a hierarchic or inhomogeneous relation.

More to more relation: when two entities have a relation in which, to entity A's 1 occurrence there are more than one entity connected from entity B's 1 occurrence then we refer to this situation as N:M, or more to more relation. We also refer to these situations as network relations. If there is an existing N:M relation between two entities we can dissolve it by introducing a third entity, which will lead to two 1:N relations.

NORMALISATION

Dependencies

- **Functional dependency:** if we can dedicate to a property's any kind of value, which exists in one system to another property type only one value. For example: to one identity number there can be only one name associated, but to the same name there can be many identity numbers related. 1 to more relation.
- **Mutual functional dependency:** if the above mentioned requirement is true in both 'directions'. For instance: registration number – engine number. 1 to 1 relation.
- **Functionally independents:** if the previous relation between the two property types does not exist. An example: the hair colour of an employee and the company's premise.
- **Transitive functional dependency:** if within an entity type one descriptive property type's concrete values determine other descriptive property values.

Normal Forms

1NF (Initial Normal Form): if all rows of the columns within one relation have one, and only one value, and the sequence of the values within each row is the same then all rows are different. There are at least one or two properties, which can clearly differentiate between the rows.

2NF (Second Normal Form): if the relation is in 1NF, and all the values that are not keys are functionally completely depend on the primary key.

3NF (Third Normal Form): if the relation is in 2NF, and the attributes only depend on a primary or alternative key. If attribute "Bs" value depend on attribute "As" value, furthermore attribute "Cs" value transitively depend on "As" value. It is the indispensable requirement of the third normal form to remove such transitive dependencies. If the table of the database is not in a 3NF format, then it should be separated into two tables in a way that the individual tables will be in the form of 3NF.

For instance: a shop that is renting tools can summarize its trade in an exercise book in the following way:

Data	Name	Address	Tool	Category	Price
05.02.1997	Géza Nagy	Eper street 5.	polisher	small	500
	Nóra Kós	Nap street 3.	welding-machine	medium	1000
06.02.1997	Géza Nagy	Eper street 5.	paint-sprayer	medium	1000
	Pál Szabó	Fő street 1.	lawnmower	big	2000
	Nóra Kós	Nap street 3.	chainsaw	big	2000

This is not a relation yet, since it does not meet the requirement of having one column value in one row.

It will become INF-like if we fill in the date in every row. In this case the table will contain too much redundancy, because the same dates appear more than once. This leads to the following **anomalies**:

- **Deletion anomaly**: with the deletion of an unwanted date a useful data will also disappear.
- **Modification anomaly**: because of the modification of a datum we need to modify the content of many fields.
- **Inscription anomaly**: when adding a new datum to the table we cannot fill in all the fields

In this form the primary key can be the **name**, **tool**, and the **category**, because the name determines the address, the tool the category, and category the price.

Date
Name
Address
Tool

Category
Price

2NF, if we make multiple individual tables that are partially dependent on the primary keys.

1. Table

Serial number	Date	Name	Tool
----------------------	------	------	------

2. Table

Tool	Category	Price
-------------	----------	-------

3. Table

Name	Address
-------------	---------

There is still a deletion anomaly, because if we delete one of the Tools then it will also delete Category and Price.

3NF, this requires handling the second table as two separate tables.

2. a. Table

Tool	Category
-------------	----------

2. b. Table

Category	Price
-----------------	-------

In the above given example the field 'Serial Number' clearly determines all the rows of the relation, that is why it can be used as a super key.

Of course, all the four tables must be used together, because in most cases it is only possible to read out information of a database by using multiple tables at once. The common use of the tables is made possible by a single field that can be found in all other tables. This field is the **relationship field**.

HOMOGENEOUS STRUCTURES

In the previous subchapter we spoke about inhomogeneous structures. In case of inhomogeneous structures we can talk about a relation between two different entity types. There are relationships when the entity's occurrence is in relation with themselves.

The backward pointer entity relation (employees, bosses)

Let us imagine, that we want to record who is whose boss at our company.

Solutions:

In the same entity (Person) we record the Bosses' property right next to the Names property. We call this backward pointer relation.

PERSON	
Personal Name	Boss Name
Feri	Pista
Józsi	Pista
Laci	Ottó
Pista	Ottó
Ottó	Peti

A problem occurs if, for example Pista quits...everywhere where the deletion appeared the name of the new boss must be written in. We can solve this problem with the use of two tables, which have multiple one-to-many relationships. Let us introduce an entity with the name Hierarchy, which is a similar person-to-person linkage. What will be put into pairs is not the names, but the person's identifiers, and these will have a relation with the modified PERSON entity.

PEOPLE	
Person ID	Person Name
1	Feri
2	Józsi
3	Laci
4	Pista
5	Ottó

HIERARCHY	
Boss ID	Employee ID
4	1
4	2
5	3
5	4
24	5

Why is the second solution better ?

- The database managers are supporting the one-to-many relation, because it fits their logic better. If a person quits, then the database manager will automatically cease the boss-employee relation, due to this one-to-many relation.
- This structure allows more general relationships to be established, because it lets us make an employee and more than one boss relation.

The Family Tree and the Spouse Relation

Let us examine the following task: at our company, we have decided to take everything we produce into an inventory. The aim of this inventory is to let us know what kind of parts were used for the construction of a machine, and that the given component was used for which machines.

One of the tables will contain what type of machine or part we have. This will be the PARTS entity. The other entity – STRUCTURE – always contains the relationship of two things. For instance, we write in the WhatIsIt column “car” and next to this in the OfWhat column the type of part used for the construction. We continue this process until all the components of the car are enlisted.

PARTS	
Part	Name of part
1	Car
2	Engine
3	Bodywork
4	Frame
5	Piston
6	Spark plug

STRUCTURE		
WhatIsIt	OfWhat	Piece
1	2	1
1	2	1
1	2	1
2	5	4
2	6	4

The above given relationship system is called family tree, or homogeneous network system. The PARTS entity is being connected to the STRUCTURE entity by the Part property through the WhatIsIT and the OfWhat properties.

The Spouse Relation

Our objective is to record who is with whom in a marital relation. Let us introduce a SPOUSE entity right next to the already existing PERSON entity.

PERSON	
Person ID	Person Name
1	Feri
2	Józsi

3	Manci
4	Juci
5	Ottó

Spouse		
Husband Code	Wife Code	When
1	2	1997
2	4	1998
5	23	1996

There is no superfluous repetition in our solution. The date of marriage can be recorded. Its advantage is that the relationships can be transparently handled. The most important thing to see is that we solved the problem with the use of a double one-to-many relation.

In conclusion, we can say that all of the homogeneous structures can be realized with a family tree structure.

DATABASE ADMINISTRATION

Data management: the recording, modification, deletion, display on the monitor, and in a list, save, etc. of the data. No new knowledge arises.

“We call those activities that are performed on data (by a computer), during which no new knowledge arises, **data management operation**.” (Halassy 1994, 117)

Data processing: new data, knowledge arises with the help of the already existing data.

“We call those activities performed on data (by a computer), during which new datum arises, **data processing operation**”. (Halassy 1994, 118)

Data management is possible without data processing; however, it is not true the other way around. If a database is carefully and well designed, then anything can be found, and worked out from it.

Derived data: datum, which can be calculated from the basic data. These data are not stored in the database.

THE ESSENCE OF THE DATABASE

IPO approach: once, users believed that data processing by a computer works along the following pattern: you have an input, a processing, and finally the output. This approach is considered out-of-date nowadays.

Database approach: the processing of data has an optimal chain. This is based on the relationships between data. Database means a basic database, where everything is recorded. According to the approach we have an early input, and delayed output. The early input means that every datum is stored. This way they will become useful during the delayed output as results.

“The ‘engine’ of the information systems is not the data processing, but the carefully determined data management based on the database structure.” (Halassy 1994, 126)

THE DESIGNING OF THE DATABASE

If we want to make a well functioning database, then we need to think through the problem that needs to be solved carefully. It is important to determine which property of an entity we want to store. According to the decision taken will we define the entity type and the structure of the database.

THE MAIN STEPS OF DATABASE DESIGNING

There are seven suggested steps in designing a database:

1. **Requirement handling:** here is where we determine the aim of the database. Consider what kind of information would we like to get from the database. We need to know which data will be stored of an entity.
2. **Determination of entities, and reports:** after sorting out the collected data we need to organise them into an information system. The information system is dealing with entities. The physical storage of the entities happens in one table. The entity instances are recorded in the rows of the table (the records), while the attributes get into the fields of the records (the columns). We should store all data in one table. This is required because when it comes to later modifications we only need to refresh the data at one place. The data considering one theme should be stored in one table.

3. **Determination of field, and attributes:** this is where we design the details of the table. Our objective is to determine the fields that build up the table. We can classify the attributes in different ways:
- a. **simple**, i.e. it cannot be dismantled any further, and complex.
 - b. The **complex** attribute consists of many simple attributes.
 - c. **one-valued**: it can only take one value at every occurrence.
 - d. The **many-valued** can take more than one value at every occurrence.
 - e. the database stores the values of the **stored** attribute. The **derived** value can be determined on the basis of other attributes.
4. **Determination of identifiers:** the data stored in the tables needs to be unambiguously identified. All the tables, which require the identification of each of their records, need a primary key. The **primary key** is a type of identifier of which values cannot be repeated in the given table. The primary key has an important role in the relational databases. With it we can increase the efficiency, speed up search and the collection of data.

In Access we can use three types of primary keys:

1. **counter type:** this is the most frequently used. During this process we need to create a Counter type field. Access will generate a unique serial number for each new record.
 2. **primary key consisting of one field:** the key is not a counter type, if it does not contain a single value that repeats itself; for instance tax number.
 3. **primary key consisting of many fields:** we make this type of key with the use of many fields. This occurs when we cannot maintain the uniqueness of a single field.
5. **Determination of relationships:** we connect the records of the tables with the help of the primary key. Relation means the connection of two entities. The quantity of relations can be divided into three groups, but since we have already discussed entity relations in one of the previous chapters, we will only give a list of the possibilities here:
1. **One-to-one relation**
 2. **One-to-many relation**
 3. **Many-to-many relation**

6. **Testing:** After the designing of the fields, tables and relations we need to check whether there is an error remaining. During the initial phase it is easier to modify the design of the database, then after it is uploaded with data.
7. **Data input:** after finishing with the necessary repairs we can finally upload the data into the existing tables. Furthermore, we can form the other objectives as well. There is a possibility to make forms, reports, and queries (see in detail later).

DATAMODEL ERRORS

- **Open logical overlapping:** if we add a property, for example, Address to multiple entities, and it means the same thing in all of them then we need to record this datum many times. This is open logical overlapping. This can cause a lot of problems subsequently. On the one hand, It occupies too much way space; on the other hand, it leads to maintenance problems. The open logical overlapping causes data repetition; therefore, it is a **redundancy**.
- **Seeming logical overlapping:** if the Address property can be found in multiple entities, but means a different address in all of them then we are talking about seeming logical overlapping. The problem here is that the unanimity is disturbed. Such overlapping must be repaired. A simple way to do this is to modify the names of the properties so they will become more expressive. We call the seeming logical overlapping **homonymy**, since this leads to mistake information.
- **Hidden logical overlapping:** if a property in two entities expresses the same, but their names are different. These kinds of errors can make things very difficult, when it comes to reviewing the data model, since they create a small chaos. This type of overlapping means the existence of two similar things with different names, therefore they are also called **synonymy**.
- **The lack of logical overlapping:** if two entities cannot be connected, because we have not created switcher fields then we are talking about the lack of logical overlapping. The lack of the realization of a relationship is called **inconnectivity**.
- **Physical overlapping:** if there is a property within an entity in which these are repeated in the occurrences. It is called 'physical', because it concerns the repetition of concrete data. The repetition of the values of the switcher role property is not a physical overlapping.

FIRST STEPS IN DATABASE ADMINISTRATION

NOVELTIES IN MICROSOFT ACCESS 2007

Similarly to the rest of the programs of Microsoft Office 2007 Access got a brand new look, too. The menus are replaced by strips here as well.

The launch screen is entirely new, now we are facing a simplified, transparent user interface, where we can choose between different templates, get to know the novelties of the program, or load already existing databases. Here we can find the list of all the previously opened files, too, thus enabling us to load them in faster without searching through the hard disk.

It is easily noticeable from the start screen that the Microsoft Office Online site plays a greater role than before, since we can download different materials from there. Therefore, those users, who have an internet connection are in advantage, for the templates and most of the hint files can be downloaded from there.

The previously known very helpful application - Northwind.mdb - is also available in the 2007 version in a brand new format. It uses .accdb extension.

II) Survey

1) Templates

- database templates
- field and column templates

a) Database templates

Each template is a whole data account application with its pre-determined tables, forms, reports, macros, and relationships. The templates were developed in a way that allows immediate usage, so we can easily start to use the database. If a template does not fit our taste we can reconfigure and reshape it to our needs. The program comes with a built-in database template collection in stores, and we can download extra templates from the Microsoft Office Online webpage.

b) Field and column templates

The field templates are field plans already provided with name, data type, duration, and pre-set properties. We can simply drag the selected fields from the Field templates job window to the data sheet. These are based on XML scheme definitions files, so we can create our own standard definitions.

Furthermore, the program has table templates as well, which help you with the tables most frequently used by the database manager. One such template is the Business card album, which already contains the most often used fields, like Surname, First name, and Address. The field properties are pre-set, too so that using the table can begin immediately. Other table templates: Jobs, Problems, Events, and Tools.

2) User Interface

The user interface is built up by several items, many of which are determining the relation with the product. The aim of the new design was to help with the effective handling of Access, and to help find the most necessary orders in a quicker way.

The most significant novelty of the user interface is the menu strip, which is part of the Microsoft Office Fluent user interface. The menu strip is the strip reaching through the upper part of the application window, which contains the orders organised into groups. The different pages of the strip organise the orders in a rational way. The main pages of the menu strip: Start page, New, Exterior data, Database tools. On each of the pages there are groups of orders associated with different functions, among which are the other novelties of the user interface.

The most significant new items of the Office Access 2007's user interface:

- **Microsoft Office Access – the first steps:** this page appears first, when we launch the application.
- **Office Fluent interface's menu strip:** the section above the program window, where we can choose the orders.
- **Order page:** orders grouped by rationality
- **Context-sensitive order page:** the order page appearing on the screen depending on the object in use, or the task being undertaken.
- **Collection:** a control that displays the possible options in a visual form, which is used for displaying the predictable result.

- **Quick access toolkit:** a uniform and general toolkit, which appears on the menu strip and gives you the necessary orders with only one click.
- **Navigation window:** found on the left side of the window it displays the database objects. It replaces the Database window of the previous versions.
- **Document pages:** the tables, enquiries, forms, reports, and macros appear as document pages in the program.
- **Status bar:** a strip found in the lowest part of the program window, which displays buttons for changing the view, or giving state information.
- **Mini window:** an item above an object, which appears in a transparent form above the text selected, and helps with the formatting of the text.

a) First steps:

This is the screen that appears at every start-up. From here, we can create new databases, create a database with the help of a template, or open an existing database, too. From here, we can directly visit the Microsoft Office Online webpage from where we can download new templates and such.

Opening of a new database:

- Launch the application. The first steps page will appear.
- Select the New database item in the New database group.
- Write a file name into the File name field of the New database.
- Click on the Create button.
- This will create the new database, and a new table will open up in Data sheet view.

Creation of new database on the basis of recommended template:

- Launch the program
- Choose a template in the online templates group on the First steps page
- Write in the desired name into the File name field

- If we want to connect to the Windows SharePoint Services website then put a tick in the box next to “Create and attach database to Windows SharePoint Services website”.
- Click on the Create button, or the Upload button.

Creation of new database on the basis of Microsoft Office Online template:

- Launch the program
- On the First steps page choose a category in the Template categories window, and when the templates of the category appear, choose one.
- Write in the desired name into the File name field.
- Click on the Download button.
- Access automatically downloads the template, creates the database based on the template, stores it in our Documents folder, and opens the database.

b) Menu strip:

The menu strip is the primary replacer of the menus and toolkits, and the main command interface of Access 2007. Its great advantage is that it collects into one place those tasks and entrance points, which earlier required menus and toolkits to display.

The menu strip consists of pages, which contain orders. These are the following: Homepage, Create, Exterior data, Database tools. The orders of the menu strip consider the currently active objective.

We can also use keyboard shortcuts with the menu strip. The keyboard shortcuts of the previous version can still be used. On the other hand, the menu accelerator of the previous versions is replaced by the key access system. This is a small size system, which uses either a letter or a tag made of a combination of letters, which appear on the menu strip, and indicate which keyboard shortcut activates the control item below them.

We can execute orders in many ways. The quickest and most direct way of doing so is by using the keyboard shortcut associated with the order.

Order page	Common operations
Homepage	Choosing another view
	Copy and paste from clipboard
	The properties of the actual font type
	Setting the actual font alignment
	Use of Rich Text formatting for Recording type fields
	Operations related to records (Refreshing, New, Save, Delete, Summary, Spell checking, Others)
	Sorting and filtering of records
	Search of records
Creation	Creation of a new table
	Creation of a new table on the basis of a table template
	Creation of a list on a SharePoint website, and create a table, which is in relation with the new list in the actual database
	Creation of a new table in Designer view
	Creation of a new form on the basis of an active table or enquiry
	Creation of a new report or diagram
	Creation of a new report on the basis of an active table or enquiry
	Creation of a new enquiry, macro, module, or class module
External data	Importing and switching of external data
	Exporting of data
	Collecting and refreshing data with e-mail
	Operations with offline SharePoint lists
Database tools	Launching Visual Basic Editor or running a macro
	Creation and examining of table relationships
	View/hide object dependency on the property page
	Running and examining performance of the database documentator
	Transferring data to a Microsoft SQL Server or into an Access database (only tables)

	Running attached table manager
	Handling of Access piggyback files

c) Context-sensitive order pages:

Beside the usual order pages, the Access 2007 program has a new interface item, which is called context-sensitive order page. It contains orders and services, which are needed in a particular situation.

d) Collections:

Another novelty of the Office 2007's user interface is the controller item, known as the collection. The collection controller item works together with the menu strip due to its design. It does not display the orders, but the results of the usage. Its essence is to give a visual overview.

e) Quick access toolkit:

By default, it is the small area next to the menu strip, which makes it possible to get access to the orders with one click. By default, those orders can be found there, which are the most frequently used, for example Save, Undo. It can be customised with those orders that we use the most often.

f) Navigation window:

When we open a database, or create a new one then the names of the database object will appear on the navigation window. Database objects are: tables, forms, reports, macros, and modules.

g) Document pages:

In the 2007 version of Access the database objects can be displayed not only in windows, but also in document pages. The document pages can be turned on and off with the program's settings.

h) Status bar:

It appears on the bottom of the window and gives space to the status messages, to the description of properties, to the process markers, etc. There are two extra functions in the status bar of Office Access 2007. These are the View and Zooming buttons.

With the help of the controller items of the status bar we can quickly switch between the different views of the active window.

i) Mini window:

In the previous versions of Access, when it came to text formatting, using the menu or displaying the Formatting toolkit was often required. In 2007 the text can be formatted more quickly with the help of the mini window. If we select a text section for formatting the mini window will automatically appear above the text.

3) New safety functions

The Office Access 2007 will simplify the securing and opening of safe databases with a new security model.

Security novelties in Access 2007:

The data still can be viewed, even if we do not allow the frozen Microsoft Visual Basic for Applications (or VBA) program codes and components. In Office Access 2003, if we have changed the security level to a high value then we had to sign the program codes, and had to mark the data as reliable, for viewing the data of the databases.

Easier usage. If we place a database file in a reliable place, for instance, a folder marked as safe, or a network place then these can be opened without warning messages and enabling frozen content. The same happens if we want to open a file that was created in a previous version. Of course, this happens only if the given file is signed digitally, and marked as reliable.

The Security Centre. All the security settings of Access can be found in one place. We can create and modify reliable place, and change the security settings. Furthermore, it is capable of evaluating the components of the database, and decide whether the given database can be safely opened, or should it freeze it, and leave it to the user.

Less warning messages. By default, in 2007 if we open a database from a not safe place then the only tool we will see is the Message strip. If we consider the given file to be reliable then we can allow the frozen components, like modified queries, macros, ActiveX controllers, terms and VBA program codes, with the Message strip.

New possibilities for signing and disseminating files. In the previous versions we could do this with the help of the Visual Basic Editor for some components of the database. In Access 2007 we can pack up a database then sign and disseminate the pack. If we unpack the database from a signed pack to a reliable place then the message strip will not appear. The same happens if we unpack the database to an unreliable place. However, in that case if we had packed in and signed a database that contained an unreliable or invalid digital signature, then we have to mark it as reliable on the Message strip on every occasion we open it. The Office Access 2007 protects the databases in file format with a stronger algorithm, by using the database password. We can encrypt our data by encrypting our database. By doing so, we can prevent unwanted users from accessing our data. The range of macro operations has extended as well, with operations at the freezing of the database. These macros have an error flagging ability; moreover, we can embed macros into forms, reports or controller properties, which worked in the previous versions of Access with VBA program modules.

1. Security structure

The Access is a collection of database objects, like tables, forms, reports, enquiries and macros, which often require the presence of each other to function. To make sure that the data are secure, the Access 2007 and the Data Security Centre perform more safety testing when we open the database.

The process:

If we open an .accdb or an .accde file, Access will hand over its position to the Data Security Centre. If the place is secure the database will open with all functions working. If we open a database from a previous version then the program hands over the position of the database, and if there is, the details of the digital signature used on the database as well.

Based on these data, the Data Security Centre analyses the reliability of the database then gives order to Access to open the database.

If the Data Security Centre freezes a content during the opening of the database, the Message strip will appear.

If we open a database from a previous version and the database is not signed, or it is unreliable then by default, Access will freeze the executable contents.

2. The restricted operating mode:

If the Data Security Centre marks a database as unreliable then Access 2007 will open it in restricted operating mode.

The following content is frozen by the program:

- VBA program codes and the references found in them.
- The unsafe operations found in the macros. All operations are classified as “unsafe” that allow a user to modify the database or get access to resources outside the database.
- Different types of enquiries:
 - Modified enquiries: With these, data can be uploaded, refreshed, or deleted.
 - Data Definition Language (DDL) enquiries: these can be used to create and modify database objects.
 - SQL transmitter enquiries
- ActiveX controller items

During the opening of the given database Access may try to load piggyback files. When a piggyback files has loaded, or the wizard starts, Access will notify the Data Security Centre, which makes further security decisions, and allows the given object or operation. In most cases the content can be allowed with the help of the Message strip. The piggyback files are exceptions to this rule. We can give permission for them at the Data Security Centre/Piggyback files page, if we tick the “All application piggyback files must be signed by a reliable publisher” square.

3. Packing, signing and disseminating databases:

Access 2007 makes it easier to sign and disseminate the databases. An .accdb or an .accdbe file can be packed in, authenticated by a digital signature, and published for other users.

Creation of signed pack:

- Open the database you want to pack and sign.
- Click on the Microsoft Office button/Publish button/Preparation and signing order.
- Then the “Choose Certificate” dialog box will appear.
- Choose a digital certificate.

- Then the “Create signed Microsoft Office Access pack” dialog box will appear.
- In the “List of place” select the place of the signed database pack.
- Give the pack’s name in the “File name” field then click on the “Create” button.

Unpack and use of signed pack:

- Click on the Microsoft Office button/Open order.
- Choose Microsoft Office Access signed packs (.accde) in the list of File types.
- To search for the folder containing the .accde file use the Place list.
- We can choose between the following possibilities:
 - We trust in the digital certificate of the installation pack.
 - Click on the Open button/”All contents from the publisher are reliable” item.
 - The “Unpack Location of the Database” dialog box will appear.
 - Select the location where you want to unpack the database. We can give a New Name to the unpacked database in the “File Name” field.

4. Encryption of database with password:

The encryption tool of Access 2007 is the combination and development of two former tools: the encryption and database password. If we encrypt the database with the help of database password then it will become illegible for other tools.

- Open the database you want to encrypt in Exclusive mode.
- Database tools page/Database tools group/click on the “Encryption with password” button
- “Set Database Password” dialog box will appear.
- Write the password in the Password field then write it in again into the Testing field.

Decryption and opening of database:

- Open the encrypted database.
- “Password must be given” dialog box will appear.
- Write in the password into the Database Password field then press the OK button.

Remove password:

- Database Tools page/Database Tools group/click on the Database Decryption button.
- “Delete Database Password” dialog box will appear.
- Write the password in the Password field then press the OK button.

Presentation of Microsoft Access 2007

The presentation of the program can be demonstrated the best and in the easiest way through a specific exercise. The task is a movie rental system, where the storage of data is the following:

MEMBERS (M_AZ, name, address, beginning date of membership)

MOVIES (MOV_AZ, title, distributor, type, genre, language, duration, length of making)

RENT (M_AZ, MOV_AZ, date of rental)

GENRE (G_AZ, genre)

TYPE (T_AZ, type)

DETAILS OF THE MOVIE (Director, stars and guest stars)

1. First steps

At first, run Access 2007 that can be done in three ways. You either choose it from Start menu/ programs/ Microsoft office, or a double click on the icon on the desk. It is also possible to run it from our existing database, by finding extensions .accdb or.mdb.

After opening, the first window is “Getting started with Microsoft Office”. Here we may create an empty database or a new database, which is suggested to follow either the online model or the provided pattern. We can also open previous database.

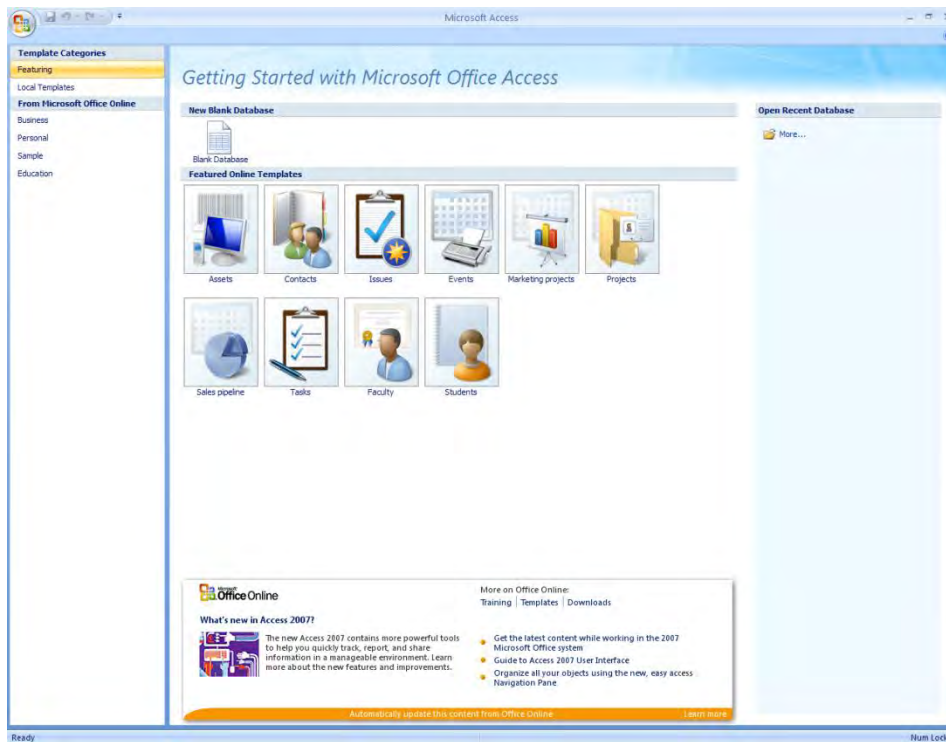


Illustration #1 – “Getting started with Microsoft Office “ sheet

To create a new database, at first click on the ‘Blank database’ icon on the upper left corner of the ‘getting started’ sheet, then state the name of the database in the empty place that appears on the right side of the screen and labeled as Filename. Finally press the create button. (In our case the name of the Database is going to be ‘Catalogue’).

Views of the Object

We may open the chosen object in to different views. With ‘design view’ we can check and modify the structure of the documents and the features of the elements. By clicking on the Open command the entries of the document become revealed. In this case we can operate with the records.

Type of Object	Views
Table	Design view, PivotTable view, PivotChart view, Datasheet view

Query	Design view, PivotTable view, PivotChart view, Datasheet view, SQL view
Form	Design view, PivotTable view, PivotChart view, Datasheet view, Formview
Report	Design view, PivotTable view, PivotChart view, Datasheet view

2. Tables

The most important parts of the database are the tables as the data is stored in them. The construction of the tables follows the well-known pattern, the columns represents the categories, or fields, while the rows gives place to the units or records.

There are several ways to create a new table. If our database had come from a pattern, some tables had already been created. The creation can take place in design view, by using a pattern or adding the documents manually. The icons that are necessary for creating a new table can be found in the row ‘Create’ under the field ‘Table’.



Using the design view is a manual way of creation. It is a rather long and complicated method. To create, we use the table design icon, under the row ‘Create’ and then define the table.

The fields can be listed in the upper left part of the given window in design view under the label: ‘field name’.

We can define the field name by clicking in the empty place of the ‘Field name’ row and write the appropriate name. After that we can set the format from the other row’s list which is called ‘data type’. The third row may be used for short notes or descriptions. At the

bottom of the window under the label 'Field Properties' we define the remaining features of the field:

- **Filed size:** In case of 'Number' we determine the type of it and in case of text we can specify the number of characters that can be written in the given field.
- **Decimal places:** by 'number' formats it provides the number of decimals.
- **Format:** In case of number it applies for the appearance i.e. general, currency, percent etc.
- **Caption:** The label for the field when used on a form in datasheet view.
- **Default value:** A value that is automatically entered in a field for new records.
- **Validation rule:** An expression that limits the values that can be entered in a field.
- **Validation text:** The error message that appears when you enter a value prohibited by the validation rule.
- **Required:** we can set whether the data entry is required or not in this field.
- **Allow zero length:** whether zero-length strings in the field are allowed

Indexed: here we have three possibilities. If we choose 'no', then the field will not be indexed. If we choose 'yes' (duplicates OK), we will prohibit duplicate values in the field. If we choose 'yes' (no duplicates), then the field will be indexed, and duplicate values are allowed that enables the possibility of unambiguous mapping within the units.

-

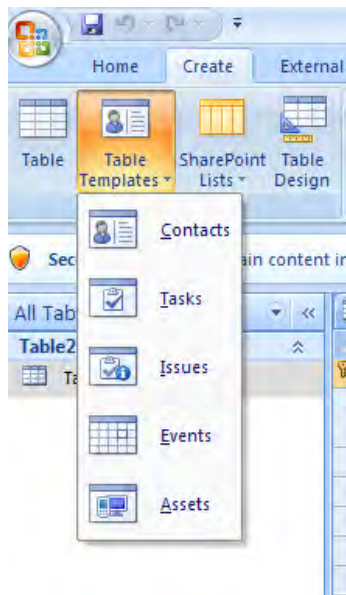
If we are done with filling in the fields close the table. The program offers us to save it, then click yes and label the table. Create the tables of the database. During this process we

have to set the primary key. We may do it by placing the mouse on the appropriate field and choose the Primary key icon by clicking with the right button.

[illegible]

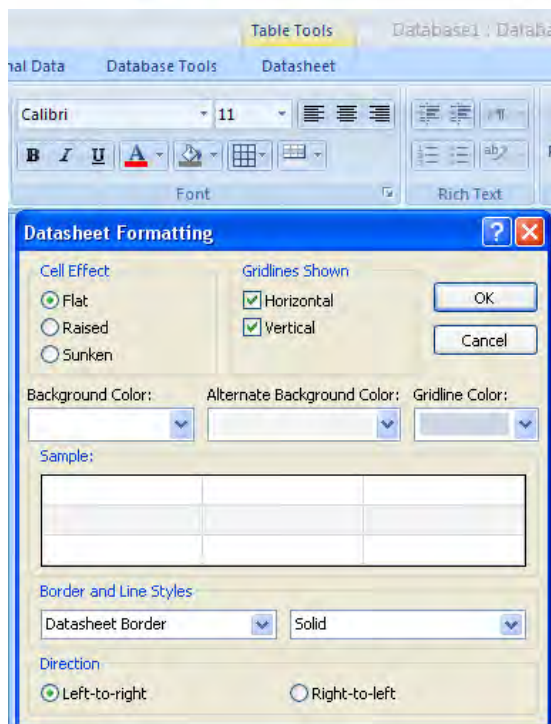
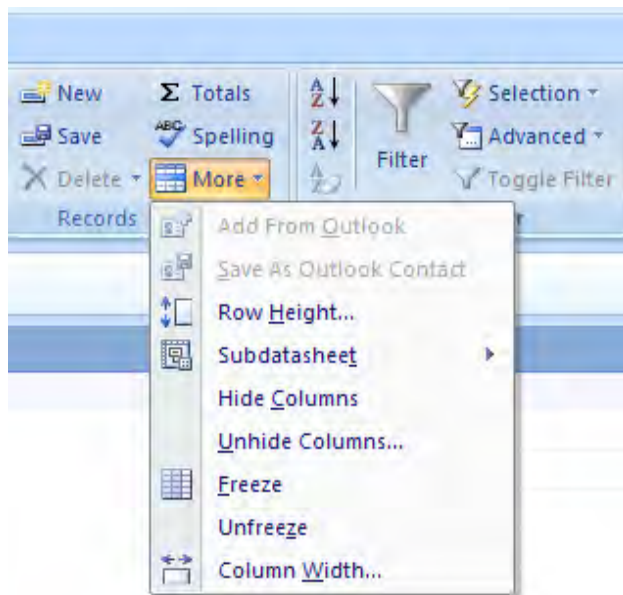
3. Creating a table

We can also use patterns to create tables. The only problem is that in most cases we may not get the table which suits our requirements the best therefore it might be necessary to modify some parts of it in design view later.



4. Table templates

The new-made tables can be formatted and the appearance can be modified. Choose the table from the navigation window and open it. From the 'home' icon, we can set the character type, size, color and also the height and width of the field.



6. set row height and column width

5. Font and formatting

The easiest way of data inputs into the database is direct typing. In order to do this we have to display the chosen table. A simple way of moving between the fields and records is to click on the chosen unit of the table or we use the arrows. It is also possible to change the structure of the tables, if we recognize subsequently that any field had been created wrong. However, the modifications must be handled carefully as they may affect the already stored data. Logically deleting a field means the delete of its contents. Changing the

formats can also result in the loss of data. We can transform numbers to text, but it is not true the other way around. For modification choose the wanted table of the database then chose the 'design view' option.

a, Handling records

If we are operating such functions that may affect more records at the same time, hence the records have to be marked. We can do it by clicking on the square on the left side of the record. In case of marking more than one record, place the mouse to the first record then hold the button and move the cursor until the last one. If we want to modify the content of a specific record just move the mouse to the wanted cell and retype the data. When deleting a record, at first mark the record and press the 'del' button. There are further possibilities in handling the record, by clicking with right mouse button on the square before the record we can open the local menu and learn about additional functions.

b, Tasks with the fields

If we would like to copy a field, click on the square on the left side of the field in design view in order to mark the row of the field. Then choose 'copy' from local menu, click at the new place then choose 'paste' from the menu and finally rename the given field. On the other hand, if we just want to move it, select the 'cut' option from local menu and insert the content to the right place.

It is quite easy to insert a new field in design view. If we would need an extra field at the end, just fill in the empty field after the last filled one. In case we would like one in the middle, move the mouse before the field where we would like to insert the new one, then select the 'insert rows' option from local menu. Field can also be added in datasheet view. In this case click on the chosen place in the row of the fieldname with right mouse button, then choose the 'insert column' option from the local menu.

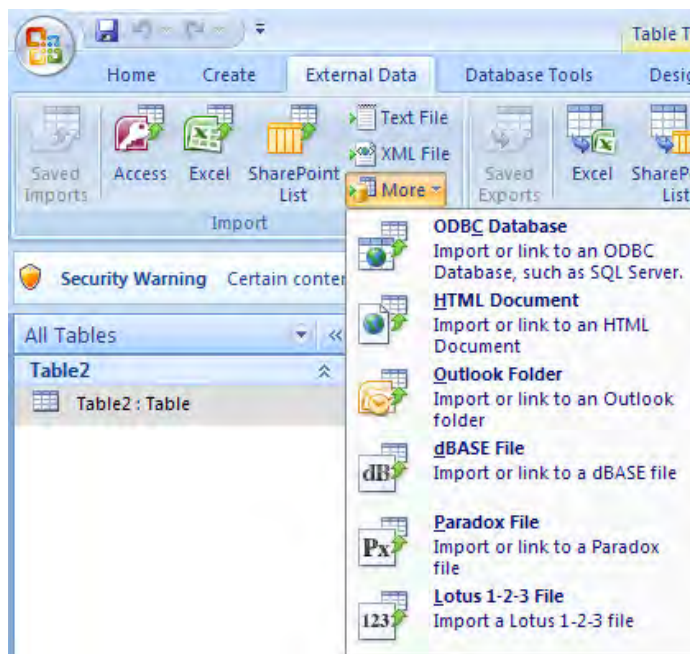
You may also use Design view to erase a field. Place the mouse on the row to delete and choose the Delete row option from local menu after pressing right mouse button. A field

can also be deleted by clicking on the square on the left side of it and pressing delete button. In datasheet view the process is the following we click on the chosen field and open the local menu by pressing the right mouse, then choose the delete columns option.

We can choose the field name in both datasheet view and design view. In case of the first one, click on the name and rewrite it. In the second case click on the name by right button and choose the rename option from local menu. Changing the data type of the field is only available in design view. Click on that specific unit scroll down the list in the data type column and choose a new one. In case of a change in data type other changes may become necessary.

Importing external data

Using importation we can copy data from another database.



The range of importable data is wide as it can be seen above. In some cases it is done automatically (access, dbase, paradox), while in other cases some few additional steps are required to import the data successfully.

Indexes, defining keys

After filling the fields and defining their features the next step is to set the indexes and the primary key of the table. Using indexes makes the arrangement and the search easier, but it slows down the data input and modification. Using indexes requires foresight. An index can be made on the basis of one or two fields. The most common method is to connect one field to one index. the process of indexing may be done in design view. After choosing the field we can set the index by scrolling down the list.

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
Smart Tags	

III. Relationships

There are three different parts of relationships. These are the one – one type, the one – many type and the many – many type. In case of one – one connections exactly one unit's belongs to each instance of another unit. For example: marital status. With one - many relationships to each instance of one specific unit may belong more than one instances of another. The most general type is the many – many relationship where both units can be linked to more instances of each other's.

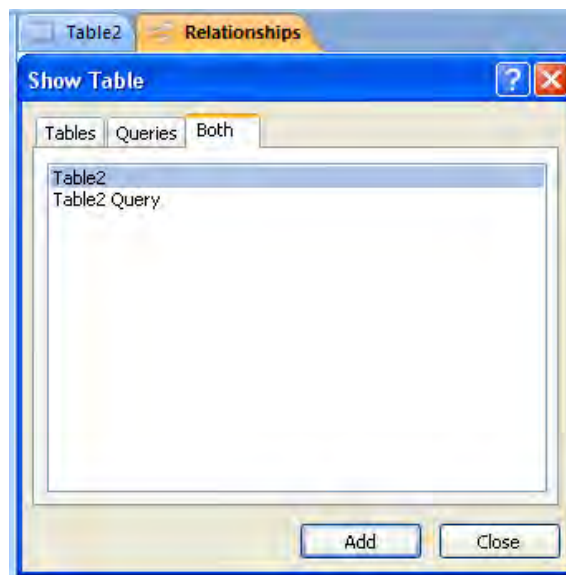
If we would like to determine which records should be included in the query we have to define the features of join. Access supports three types of it:

The first option is when only those lines are included where the joint fields are the same in all tables.

The second possibility is when all the records of the original table are implied but in case of joint tables, only those count where the joint fields are identical.

The third way is the reverse of the previous one that is only those record are included from the original table where the joint fields are the same, but every record is included from joint tables.

To create relationships we must choose the relationships icon of Microsoft Access from Database Tools menu. If we have not established any relationship yet add the table to the panels. To do this we have to choose those tables that possess relationship and press add button.



7. illustration

We may add tables to our relationships if we choose the table design option from local menu. After seeing all the tables on the relationship panel we can begin to build up our relationships. Before that the open tables have to be closed. For new relationship draw the related field of the table to refer to the related table of the referred field, therefore a window appears where we can set the properties of the relationship.

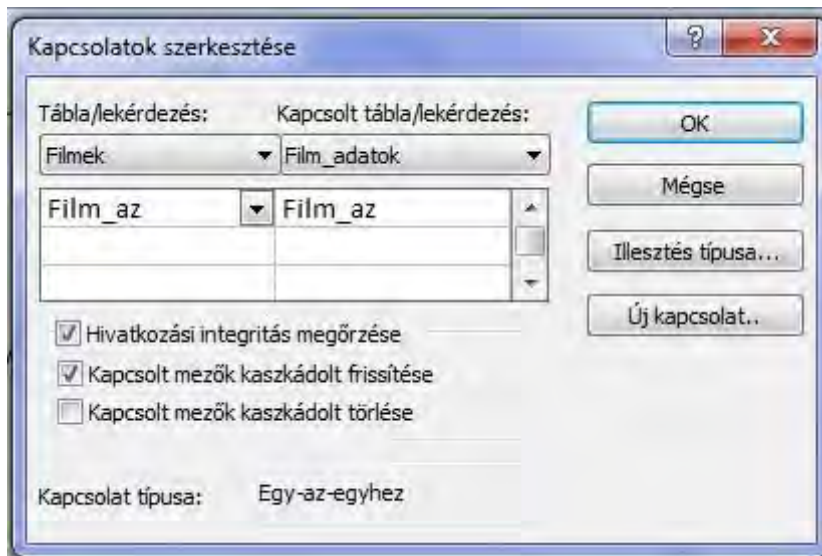


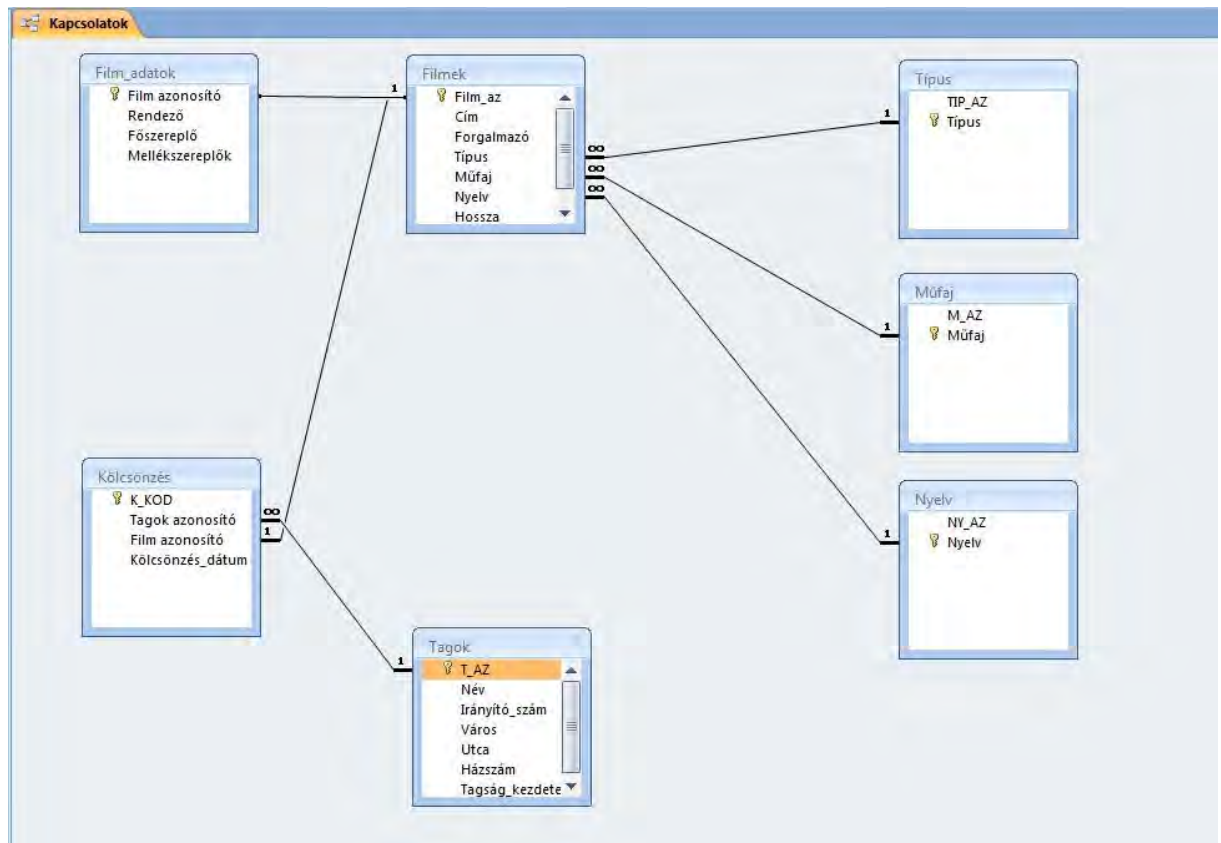
Illustration #8.

When setting the properties the following ones have to be included:

- **Preserving the integrity of the query:** the related data could not be deleted accidentally and independently from each other.
- **Cascading refresh of the related fields:** On the primary table in case of a change in the primary key the Access automatically refresh the primary key to the new values on all related tables.
- **Cascade delete of related fields:** On the primary table in case of the delete of records the related records are also deleted on the related fields.

The properties of a relationship can be changed subsequently. In this case press twice on the related line and modify the properties on the panel. Obviously we may also delete our relationships by choosing the 'del' option on the related line. The local menu can also be used to modify or delete relationships.

If our relationships are set in case of opening a table every record gets an extra + before it. here we can develop the data belonging to the given record but localized on another table.



#9. Relationships.

IV. Operations in the database

We can execute easy operations in our database like searching, replacing, filtering, screening, and refreshing.

a, Find

It can be used to find a record in a field. First place the mouse on top of the field in which we would like to search then choose the Find icon from the main page.

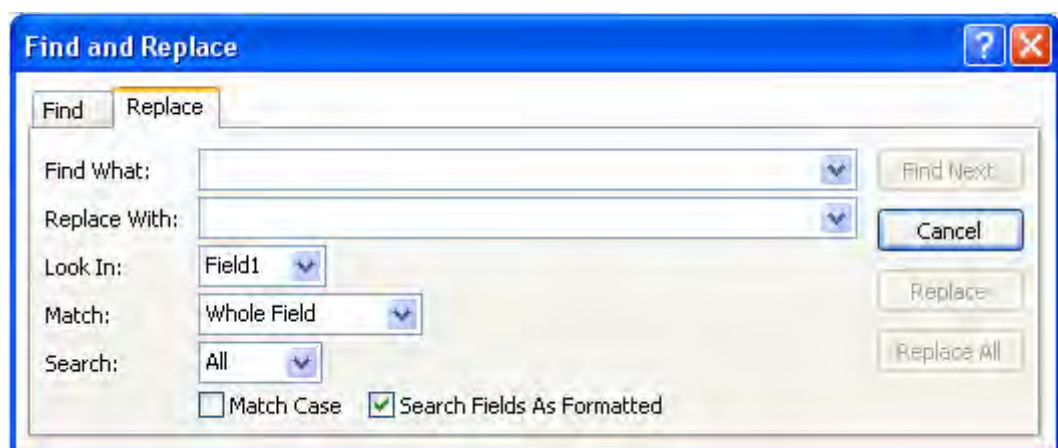


#10. Find

Type the wanted data to the Find What field then set the appropriate options that can be the following: Look in, Match, Search. By pressing the Find Next button the Access looks for the first record that contains the wanted word.

b, Replace

With this option we may replace every frequently appeared, identical record with another data. Again place the mouse onto the field where we would like to replace and choose the replace icon.

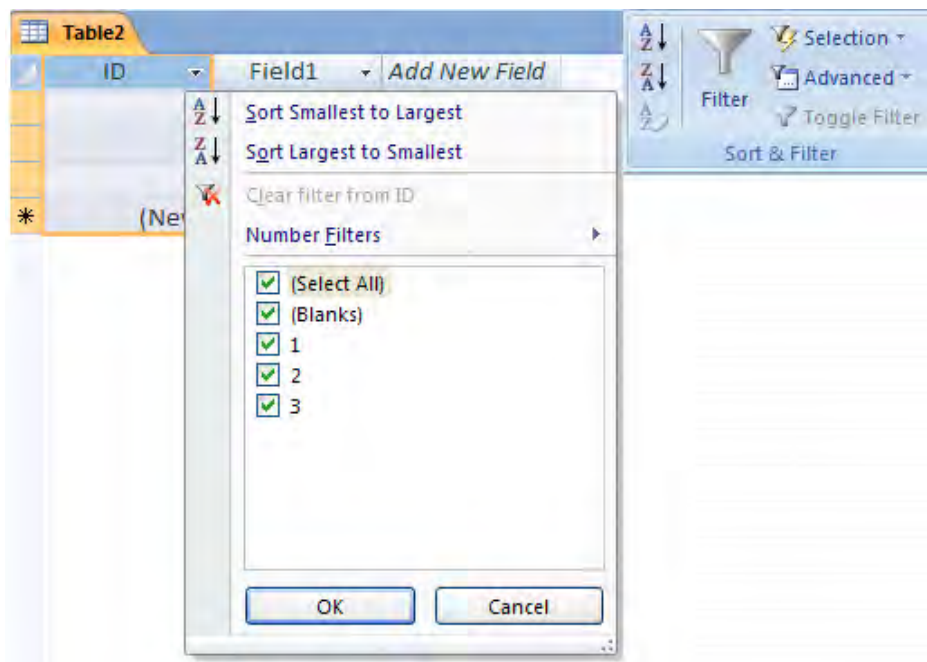


#11 replace

Type the data to replace and also the new data and set the different options. By clicking on the Find Next button the program search for the next record that contains the preset conditions. Then we can choose the replace option to replace the specific one but if we choose the replace all button, then all the records containing the given data are replaced.

c, Sorting

Sorting can be done in two different ways, one option is to choose Sorter from home page either the 'sort smallest to largest' or 'sort largest to smallest'. Another possibility is to click on the little triangle next to the name of the field and from the list we choose the appropriate filter.

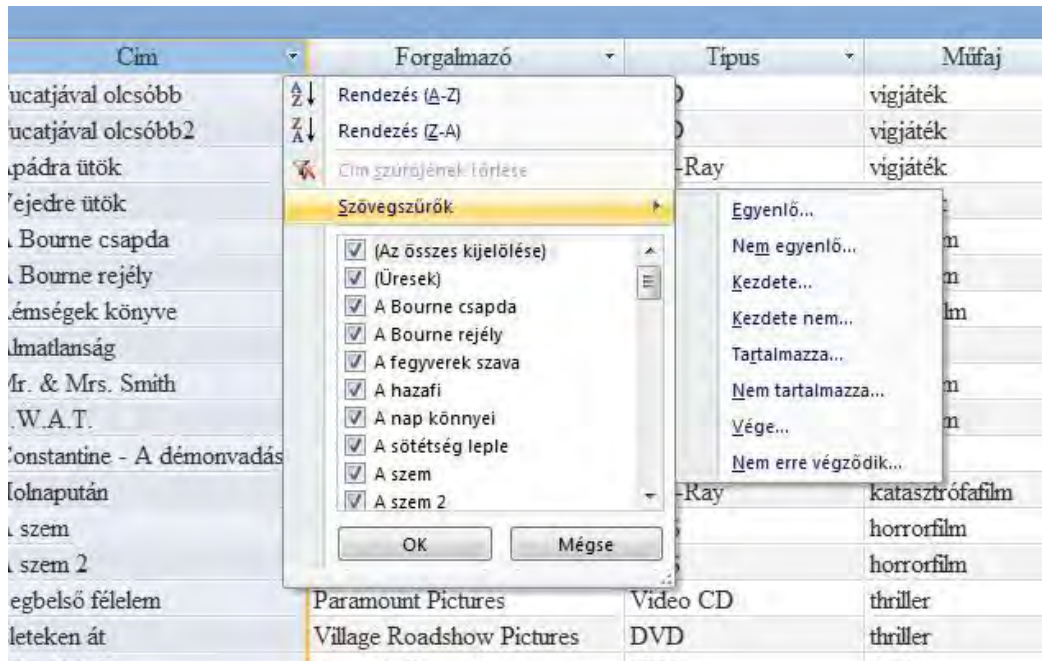


#12 Filter

d, Filter

We have the possibility to see only those records that fulfills the criteria. We can do it from the Sort and Filter option also from the main page. We may also start screening by using the little arrow next to the field name. Then we can choose one or more of the database, give further conditions or even delete the filtering process. By using specific filters for numbers, texts we can set unique conditions, so essentially we can create any kinds of assumptions. There is also a way to screen according to Form, hereby a screen-form is

opened where all screening information can be applied. Turning on and off screening mode can be done from the ‘sort and filter’ option of main page.



e, refresh

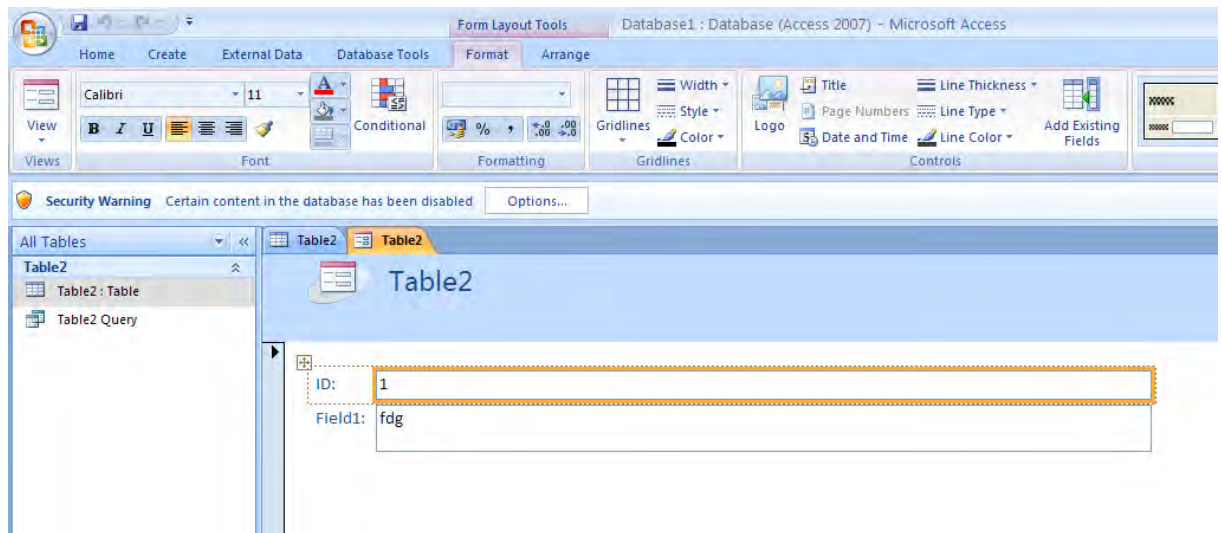
After certain operations it may happen that not the supposed data appears on the table. Then we need a refresh which can be done by clicking on the ‘refresh all’ icon.



Forms

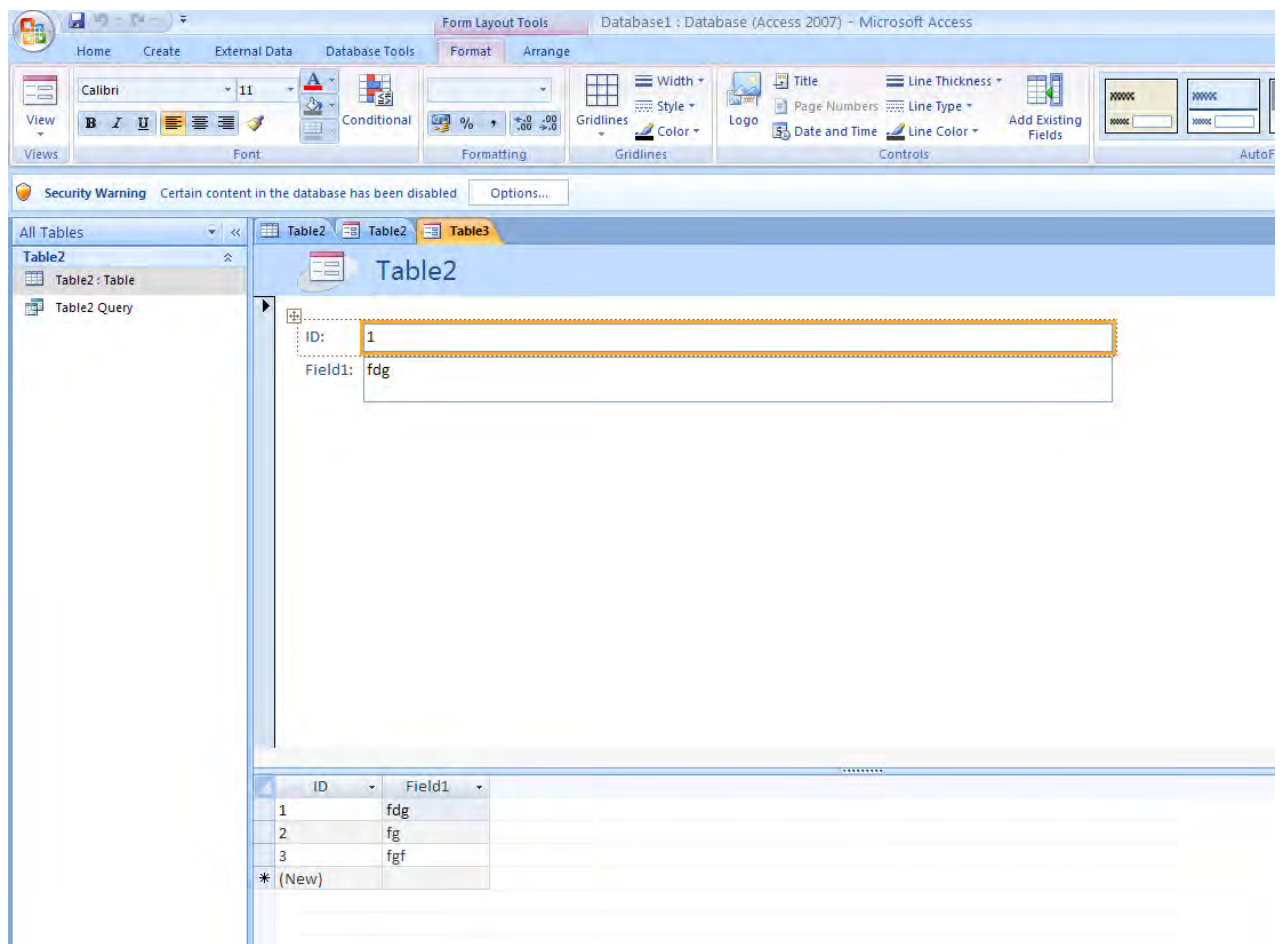
a. Creating forms automatically

This is the most practical way of creating forms. It can be done by using the Form icon on the main page.



#13 Creating forms

If the given form is appropriate save it. We may create a similar form with the 'split form' icon but then the table also appears at the bottom of the form.

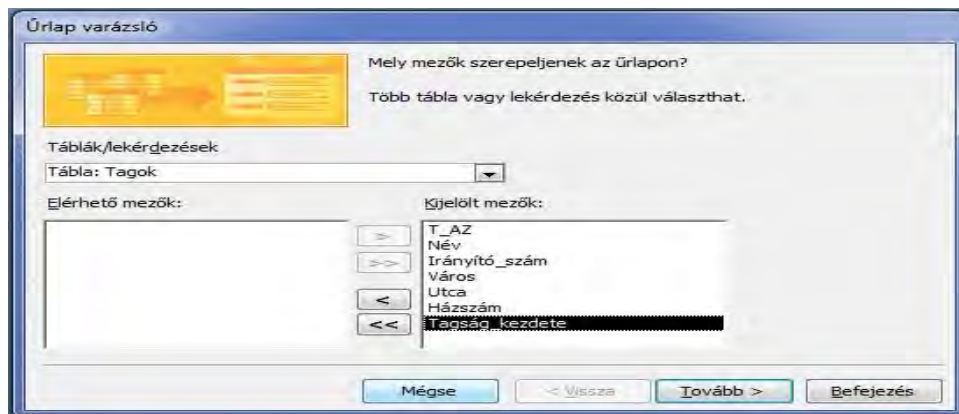


#14 Split form

We can create Forms containing even more units, pivot tables, pivot charts, empty forms or by scrolling down the More Forms option, also datasheets.

b) Creating form by using wizard


We can do this choosing the Create tab, scrolling the More forms button and clicking on the Form Wizard button. In the appearing window we choose the data source we want to use as the base for our form and add the fields. We are using the Members table here.



Click on the Next button to set the layout, which can be Columnar, Tabular, Datasheet and Justified. Click on Next again.

Úrlap varázsló

Milyen szerkezetű legyen az űrlap?



☒ Oszlopos
☐ Táblázatos
☐ Adatlap
☐ Sorkizárt

Mégse < Vissza Tovább > Befejezés

In the next window we can choose the style of the form, then we can give it a name and choose whether we want to open the form to view, to enter information, or to modify the form's design.

Úrlap varázsló

Milyen stílust szeretne?

Úrlapformátumok:

- Access 2003
- Access 2007
- Áramlás
- Aspektus
- Fényűző
- Hegycsúcs
- Iroda**
- Lendület
- Loggia
- Medián
- Metró
- Modul
- Műhely
- Nanforduló



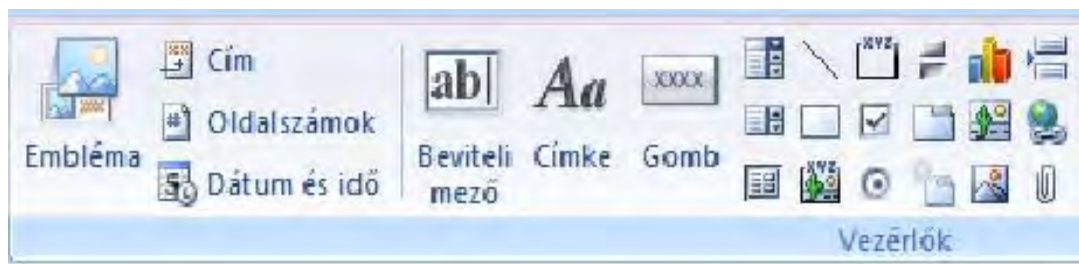
Mégse < Vissza Tovább > Befejezés

c) Creating form in Design View

We can create forms from the beginning, building it up ourselves. On the Create tab choose the Form Design button then add the fields to the form by clicking on the Existing Field button in the right hand corner and dragging the field onto the list. Then set their properties and parameters, save and close. In this view many elements for adding and viewing can be used. These are the following:

1. Logo: Insert a picture into your form or report to be used as a logo.
2. Title: Display a title in a form.
3. Page numbers
4. Date and time
5. Text Box: Add general data
6. Label
7. Button: Form Control
8. Combo Box:
9. Option Group
10. Toggle Button

11. Insert Chart
12. Insert or Remove Page Break
13. List Box
14. Rectangle
15. Check Box: Choose yes or no
16. Tab Control: Create subforms with tabs
17. Unbound Object Frame
18. Insert Hyperlink
19. Subform
20. Bound Object Frame:
21. Option Button:
22. Insert Page
23. Insert picture
24. Attachment



We can modify our existing forms by changing to Design View in the opened form. In Design View we can see the ruler, the gridlines, and the icons of the Form Elements.

d) Form styles

Access gives us the opportunity to design the forms automatically using styles. We can do this on the Arrange Tab choosing the AutoFormat button.

e) Changing the properties of an element

We can do it by right-clicking on the element we want to modify then choosing the Properties from the appearing list or by switching on the Property Sheet icon. The appearing Property Sheet panel offers many setup opportunities that can be divided into

four groups. These groups can be defined by choosing the appropriate tabs. The All tab shows all properties at the same time. If we change the Visible field to 'No' on the Format tab then the given element will not appear in normal form view only in Design View. The Allow fields on the Data tab do not allow data entry for the element if 'No' is chosen.

f) Creating a diagram

We can create special forms that view diagrams based on data from the database. This can be done by clicking on the Diagram button on the Design tab. In the first step the table on which the diagram will be based on has to be chosen. Then we need to choose the fields and the type of the diagram. Finally, the field names and the title of the diagram have to be given.

2. Queries

With queries we can link tables, filter records and list our data. With the help of queries we can filter our data, even if they are in different linked tables. This can be done in a graphic way or by using SQL language. The result is given in tables which can work as bases for reports and forms. In queries we can use mathematical procedures, functions or we can give various conditions.

Queries in Access 2007 can be created in two views and visualized in a different one.

Design View: the tables used by the query, their fields and relationships and the QBE grid appears

SQL View: we can view the query's definition in SQL language

Datasheet View: we can view and modify the resulting data of the query

a) Types of Queries

- Select Query: this is the most common query used as a base for the rest of the queries. This query creates a select query from the fields you pick, sorts and filters data, composes summaries and groups.
- Crosstab Query: *displays data in a compact, spreadsheet-like format*. It creates separate columns from repeated values of a field. If a chosen column contains more than one identical element then the identical elements form separate columns by using the repeated values as headers of the columns and the corresponding data will form rows. By the meeting points of the columns and rows will be those values that come from a third field. The disadvantage of this query is that the data cannot be modified.
- Make-Table Query: Executes a predetermined procedure on the data. The new table contains the resulting data of the query.
- Update Query: We use it to modify records when we only want to modify a certain group of records.
- Append Query: We can use it to add records to existing tables. We are able to archive and extend existing tables with this query.
- Delete Query: This query enables us to delete complete records using given parameters.
- SQL Query: Every previously mentioned query is a SQL query but there are special cases when unique queries written in SQL language are needed.
- Join Query: It displays data queried from several tables jointly.
- Aggregate Query: We can execute commands for ODBC databases in order to access these directly in SQL language. The ODBC is a standard database accessing method that makes it possible to access any data from any database irrespectively of which system handles them.
- Parameter Query: We can create modify or delete tables.

b) Defining conditions

If we do not want to see every item as the result of a query, but only certain parts, we have to use conditions. These can be operators and relationships.

Comparison Operators:

We execute comparison with the help of these operators. We can do these with relation symbols: =, <, >, <=, >=. These can be used with numbers and texts.

Logical Operators:

These can be used for True/False type data. Like the AND, OR and NOT.

AND: „expression1 AND expression2”, it is true if both conditions are true.

OR: „expression1 OR expression2”, it is true if any condition is true.

NOT: „NOT expression, it is true if the given condition is false in the expression.

Arithmetic Operators:

To execute common mathematical operations.

Special Operators:

LIKE: this operator can be used as a filter.

BETWEEN ... AND: determines whether a numeric or date value is found within a range.

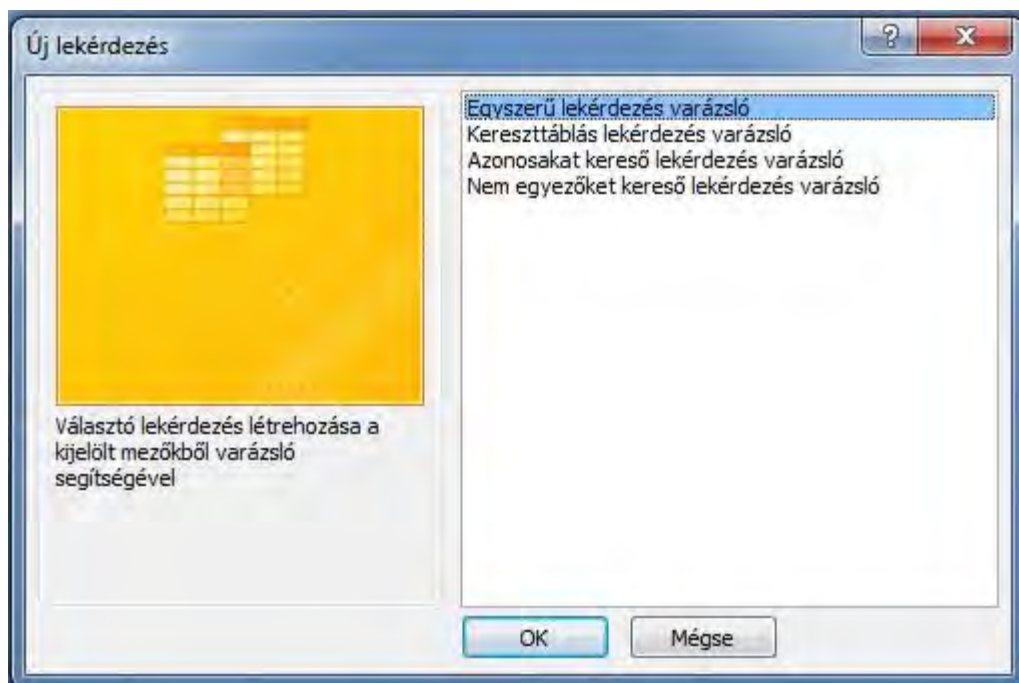
IN: substitutes many OR operators. IN (expression 1, expression2, ... expressionn)

ISNULL: selects the records where the given field is empty.

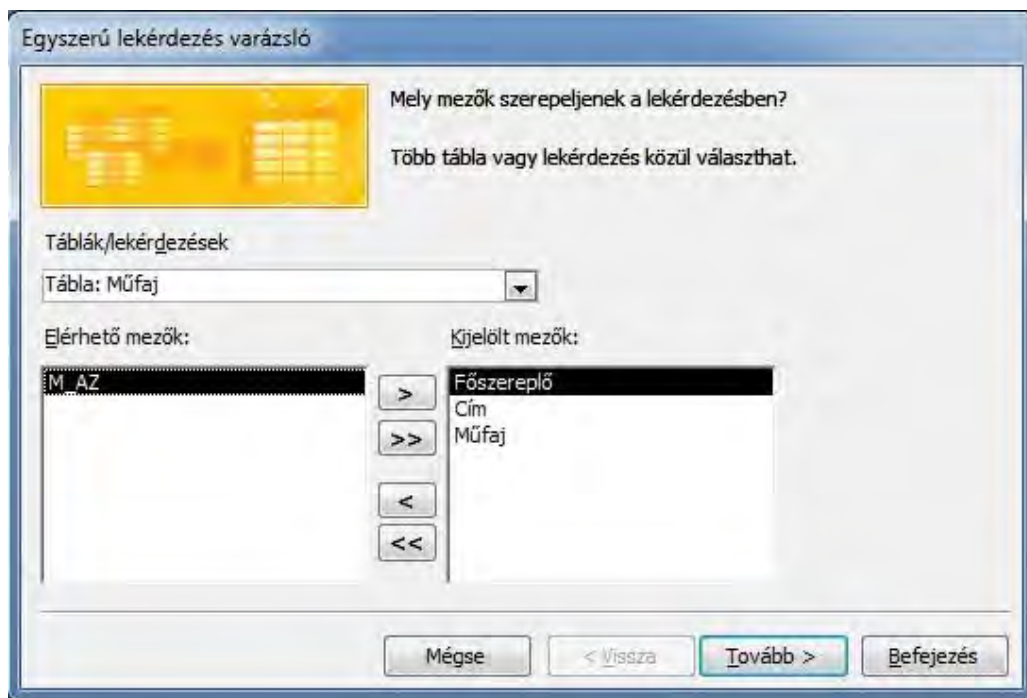
ISNOTNULL: selects the records where the given field is not empty.

c) Creating Queries

We can create Queries in Design View but it easier to use the Query Wizard. We can find it on the Create Tab by clicking on the Query Wizard. The first step when creating a query is to determine the type of the Query.



The next step is to choose the tables and their fields.



The last step is to define the type and the name of the Query.

Egyszerű lekérdezés varázsló

Milyen címet szeretne adni a lekérdezésnek?
 Film_adatok Lekérdezés

A varázslónak ezekre az információkra volt szüksége a lekérdezés elkészítéséhez.

Meg kívánja nyitni a lekérdezést, vagy a lekérdezéstervet szeretné módosítani?

☒ Lekérdezés megnyitása az adatok megtekintése céljából
☐ A lekérdezésterv módosítása

Mégse < Vissza Tovább > Befejezés

d) Modifying Queries:

If the complete query is not like the one we desired to create it is possible to modify it. We can do this by changing the view of the running query to Design View or by selecting the query and clicking the Design button on the Home tab. In this case the tables forming the query appear in the upper half and below appears the QBE grid which can be designed.

Film_adatok Lekérdezés

Műfaj: M_AZ, Műfaj

Filmek: Film_az, Cím, Forgalmazó, Tipus, Műfaj, Nyelv

Film_adatok: Film azonosító, Rendező, Főszereplő, Mellékszereplők

Mező:	Film_adatok	Cím	Filmek	Műfaj	Műfaj										
Tábla:	Film_adatok		Filmek	Műfaj	Műfaj										
Rendezés:															
Megjelenítés:	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Feltétel:															
Vagy:															

e) Adding or deleting tables:

If we want to add a table to our query go to the Create tab and choose the Display Table icon and click on the desired table we want to add to the query.

It is easy to delete a table, just click on the table and hit Del.

f) Adding and deleting fields:

We can add a field by choosing the field of the desired table and dragging it into the appropriate place of the QBE grid. Another option is to click on an empty item's Field or Table item and choose the desired field or table from the drop-down menu.

To delete a field click on the check box above the QBE grid and hit Del.

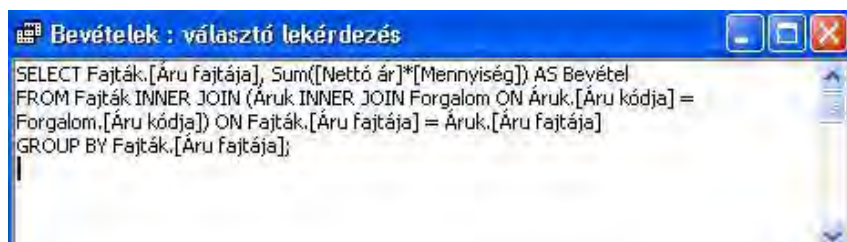
SQL Query

A SQL (Structured Query Language) Query is a query which is created using SQL commands. As a result of the spread of database management software a standard language needed to be created which enables to execute the different operations in the different programs.

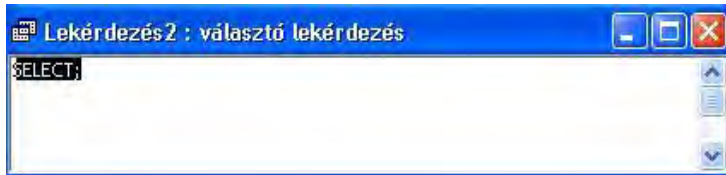
Using SQL

To execute more complex queries sometimes it is essential to write a SQL command into the Condition row of the QBE grid. The SQL does not contain control commands therefore it is not suitable for programming.

Every query can be viewed in SQL View. Execute the SQL View command in the View Menu.



If we want to create a query using SQL command without using the QBE grid, we need to choose the New Query in Design View on the Create tab without adding tables. Then change into SQL View. A window appears where we can edit the SQL commands.



The SQL commands always start with a keyword and are continued with parameters and finished with clauses. Add a semicolon at the end of the commands.

So, the command is introduced by the Select which can be followed by an optional Predicate.

Predicates:

- ALL: as a result every record that meets the conditions appears.
- DISTINCT: only one appears in case of identical rows.
- DISTINCTROW: a record which is identical to another one does not appear again.
- TOP: the number of records that appears is identical to the number which we write after TOP. It does not differentiate between two equal values. With the PERCENT option we can display percentage.

After the predicate the tables and fields displayed in the query are listed. All fields marked with an asterisk * appears in the query. With Table name* every query of the particular table appears in the query. If we want a name for our column other than the name of the field we need to add the name after AS.

After listing the field names we list the names of the used tables in a new row introduced by the FROM keyword. Separate these with commas. If the table we intend to use is in a different database we have to supply its name and access path within quotation marks.

Clauses

- WHERE: We list the criteria after the keyword. e.g. WHERE Irányár>10000
- ORDER BY: Specifies how to sort the results. In descending order provide the DESC keyword after the field name. e.g. ORDER BY AVG(Irányár) DESC
- HAVING: we use this clause when we make conditions on records grouped in a query

Most important functions:

- SUM: Returns the sum of a set of values contained in a specified field on a query.
- MIN, MAX: Return the minimum or maximum of a set of values contained in a specified field on a query.
- AVG: Calculates the arithmetic mean of a set of values contained in a specified field on a query.
- COUNT: Calculates the number of records returned by a query.

In the case of a multi table query we have to define through which fields the tables are connected. The connecting fields are linked with an equal sign, for more relationships we use the AND operator.

Statements:

- UPDATE: we modify the values of the existing fields of a table. We specify the name of the table we want to modify after the UPDATE statement. This is followed by the SET keyword, then write the formula into the next row which is the basis of the modification
- SELECT INTO: we can create a make-table query with it. Here we have to specify the fields which we want to copy into the new table. This has to be followed by the INTO statement and the name of the new table.
- DELETE: we use it to create a delete query. After the statement the name of the table that we want to be deleted preceded by FROM has to be given. WHERE is followed by the criteria of the records to be deleted.

- **INSERT INTO:** we can add a new record into an existing table. After INSERT INTO the name of the table to which we want to add records is given. Then SELECT followed by the name of the field, after FROM comes the connection of the tables and after WHERE the conditions.

I finish presenting the SQL language here. To get to know it better I suggest that you view in SQL view the queries either on the QBE grid or created by the wizard as many times as possible. After studying these try to understand what you have seen.

Types of SQL queries

Join Query: It collects the fields of several tables or queries into one field. After creating the query go to the Query menu from there to the SQL-specific submenu and click on Join.

Aggregate query: it sends commands that the server accepts directly to ODBC databases. We can query records and modify data for instance. After creating the query go to the Query menu from there to the SQL-specific submenu and click on Aggregate. The ODBC is a database whose data can be attached, exported and imported with the relating driver.

Parameter query: it is for creating or modifying objects like the MS Access and MS SQL Server tables. After creating the query go to the Query menu from there to the SQL-specific submenu and click on Parameter.

Subquery: this type of query consists of a SQL SELECT statement within a select or update query. So, it is embedded into a SELECT, SELECT...INTO, INSERT...INTO, DELETE or UPDATE statement. These statements are written into the Field row of the gridlines of the query. If we want to add a criterion to a field we write it into the Criteria row. We use the sub-query to check whether certain elements of sub-query exist or not. Words we use: EXISTS, NOT EXISTS, ANY, IN, ALL.

Its syntax: comparison[ANY|ALL|SOME] (SQL statement)

expression[NOT] IN (SQL statement)

[NOT] EXISTS (SQL statement)

Parts of a sub-query:

Item	Explanation
Comparison	It compares the expression with the result of the sub-query.
Expression	We search the set of the result of the sub-query with it.
SQL statement	A SELECT statement that follows the format and rules of other SELECT statements.

3. Reports

The bases of the reports are the queries. We can create Reports in three ways: using the Report tool, by using the Report Wizard and by using the Blank Report tool.

a) Creating a report by the Report tool

Choose the table or query on which you want to base the report then on the Create tab, in the Reports group, click Report. In the appearing design view we only have to do the formatting and modifying.

b) Creating a report using the Report Wizard

On the Create tab, in the Reports group, click Report Wizard. In the first step we have to add the table or query on which we want to base the report or the fields we want to display in the report. The second step is to define how we want the data to be displayed. On the following panel we can set the group levels then the order of appearance, the orientation, the layout, and the style. Finally, on the last panel we can give a name to our report.

Jelentés vázsló

Szeretne hozzáadni csoportszinteket?

Cím
Forgalmazó
Főszereplő
Mellékszereplők

>
<
*
Prioritás
*

Cím, Forgalmazó, Főszereplő,
Mellékszereplők


Csoportosítási beállítások...

Mégse < Vissza **Tovább >** Befejezés

Figure 24: Report Generation 2nd step

Jelentés vázsló

Milyen rendezési sorrendet szeretne használni?



A rekordokat legfeljebb négy mező szerint rendezheti, növekvő vagy csökkenő sorrendben.

1	Cím	▼	Növekvő
2	Forgalmazó	▼	Növekvő
3		▼	Növekvő
4		▼	Növekvő

Mégse < Vissza **Tovább >** Befejezés

Figure 25: Report Generation 3rd step

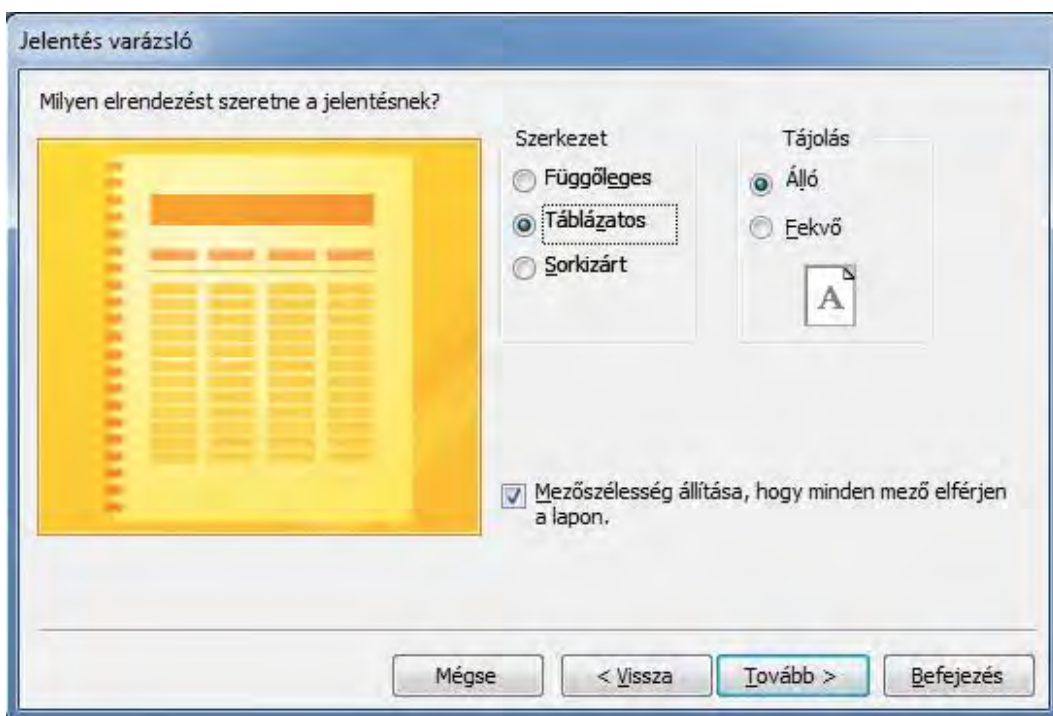


Figure 26: Report Generation 4th step

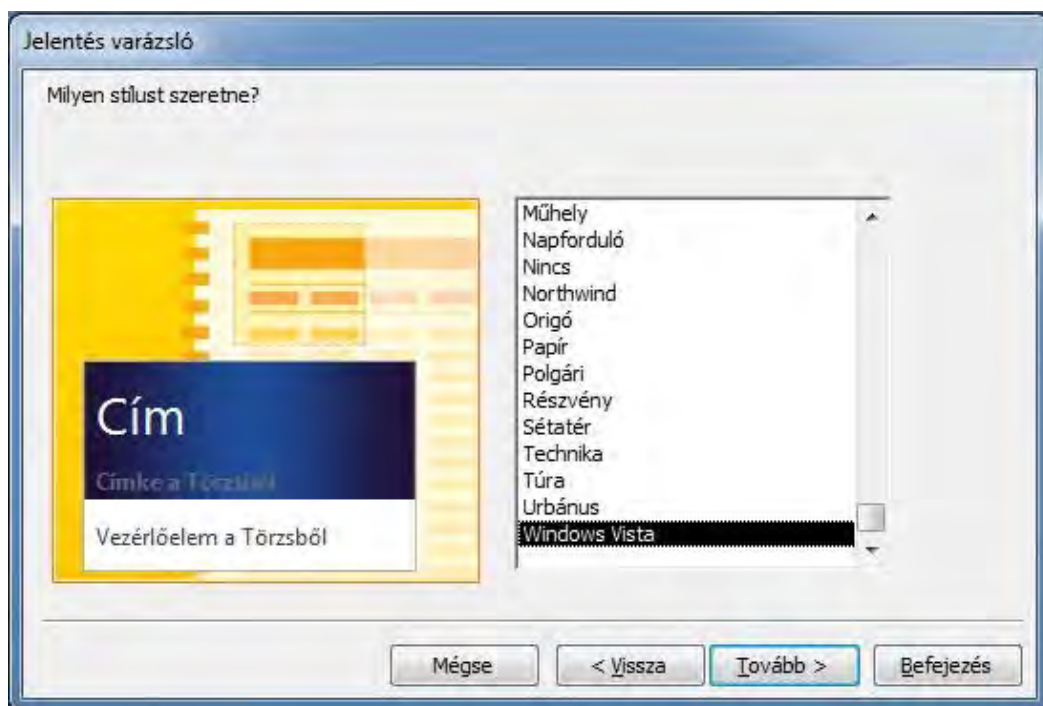


Figure 27: Report Generation 5th step

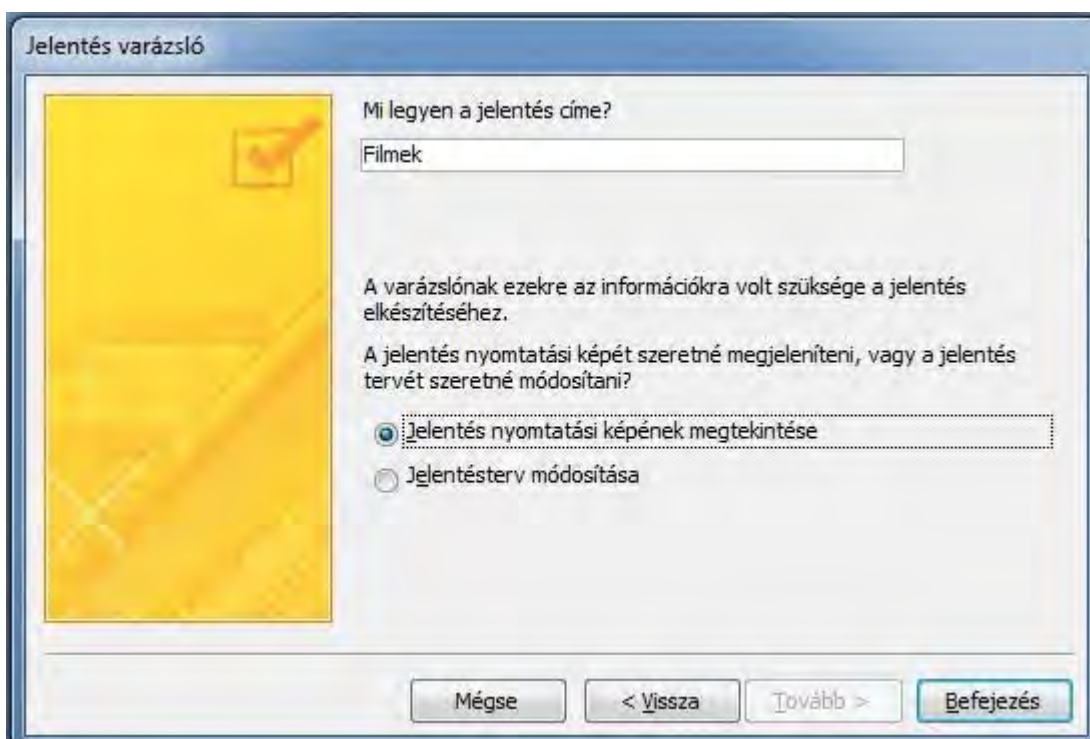


Figure 28: Report Generation 6th step

a) Report generation with unique designing:

It is easier to generate a report with a wizard or automatically, however, we have the possibility to generate a report using unique designing. To do this, click on the Blank Report icon on the Creation strip then drag the fields you want to use, into the blank report from the field list on the right side.

b) Modification of a report:

If we want to modify our report then select the report you want to modify then switch to the Editor view.

The easiest way of adding a new field is switching in the Add Fields button then dragging the field you want to use from the list on the right to the panel.

We can create special reports, which will display diagrams on the basis of the data in the database. To do this, click on the Diagram button on the Designing strip.

We can move the items of the report as easily as dragging the edge. We can modify the size of the items by dragging the squares on their edges and corners. We can also change the properties of the items by bringing up the local menu and selecting the option we want to use.

If we want to give a different frame to the report elements then we can do so by using the Special Effects button on the Design strip. Here is where we can change the line thickness, line type and line colour of the items. There is a possibility to automatically format the reports on the basis of styles. We can do this with the Automatic Formatting icon of the Layout strip.

We can also use unique fonts and highlighting. We find this in the Font field of the Designing strip.

We can use different formatting for those records that meet certain requirements by clicking on the Conditional icon in the Designing strip.

a) Macros:

A macro is basically a pre-defined sequence of events, which allow more complex and multiple staged sequences of activities to be done with a single click or with the press of a button. The editing of macros under Access is significantly different from the macro recording technique of Word or Excel.

a) Creating a new macro:

We can create macros by clicking on the Macro button on the Creation strip. During the definition the macro we will see three columns on the panel. The first column is the operation, the second its argument, and the third is a note connected to it. What is important for us is the area in the lower part of the window, which contains settings for the property. This is where we can determinate the environmental parameters of each operation. We can also create conditional structures for the macros by clicking on the Criteria icon on the Designing strip.

b) Modification of macros:

Open the macro in the Designing view. All rows can be modified in the macro code, if we select a specific row then select the new order or parameter from the pull-down menu. We can also insert a new row, if we select the place where we want to do this, and click on the Insert Rows icon on the Designing strip. To delete a row, click on the empty square at the beginning of a row then press the Del button, or choose the Delete Row icon from the Designing strip.

c) Running a macro:

To run a macro, click on its name twice in the navigation window. This could happen automatically, since if we switch a macro to an event then with the occurrence of the event, the macro will start.

d) Switching a macro to an event:

Click the right mouse button on the suitable item in the designing view then select the Properties option from the local menu. Click on the white field on the Event page's suitable item then choose the written macro from the pull-down menu, or if it is not ready yet, click on the three dots. This will start the macro editing.

PRINT OPTIONS

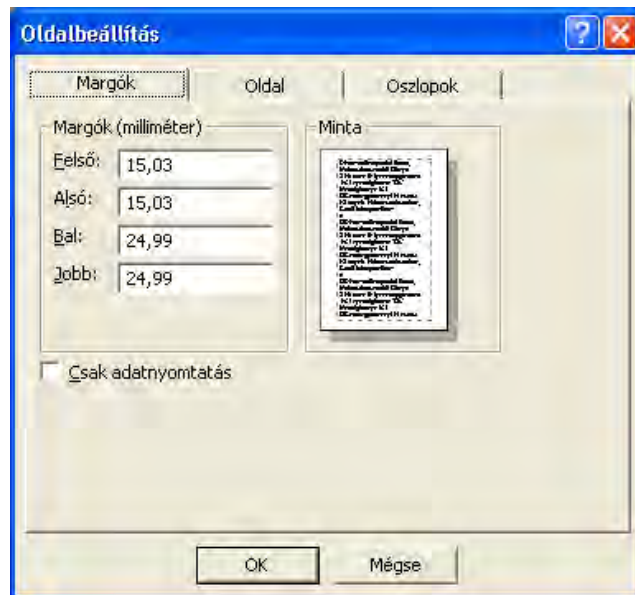
We can also put our data stored in the database on paper. All objects can be printed in the same way. This can be a simple table or a report with summing operations.

We can print the objects in the Designing, Form or Data Sheet view. We can run printing by clicking on the **Printing** icon in the toolkit, or launch it from **File** menu by clicking on the **Printing** order. We can bring forth the print options window with the Ctrl+P letter combination.

SETTING PRINT PARAMETERS

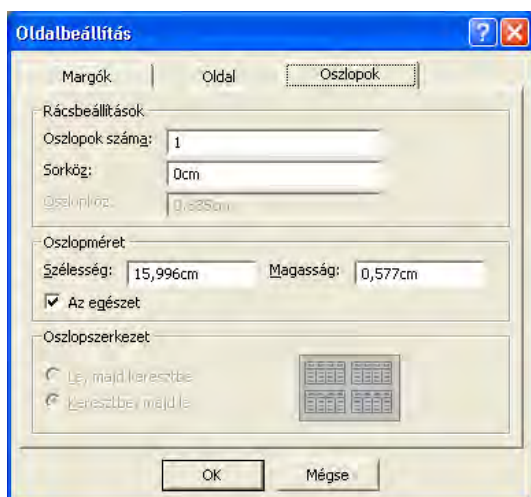
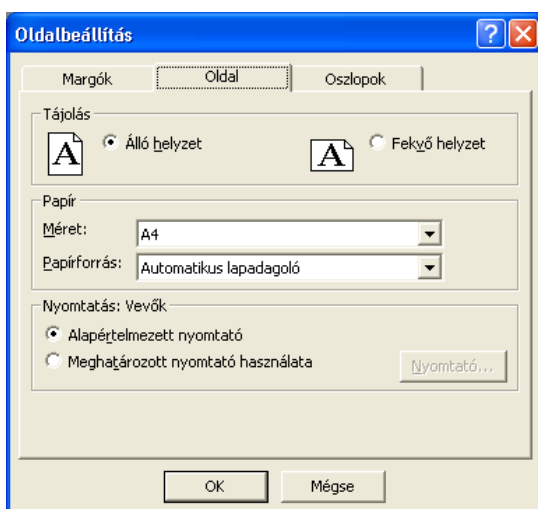
There are a few settings that need to be done before starting printing. For example: number of copies, quality, orientation, margins, layout.

After opening the object give the Page Setup order from the File menu.



On the **Margins** panel the margins. Only choose the **Print File** option if you want to omit the gridlines, graphics, etc. during the printing of a form or a report.

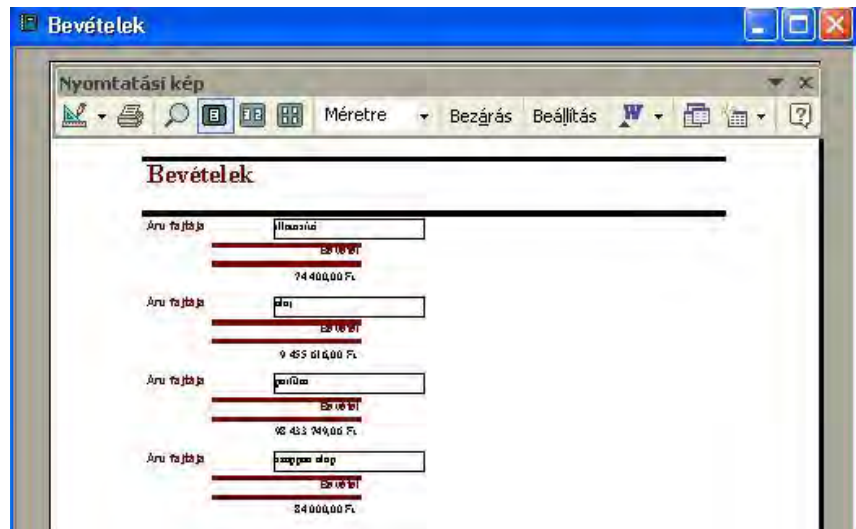
On the **Side** page the orientation, paper size and the attributes of the printer can be given. There is a possibility to select a different printer, which is not the default one. Choose the **Use Specific Printer** Radio button.



On the Columns page it only appears when it comes to forms, reports and macros. We can modify the number of columns, their sizes, the layout and the printing sequence.

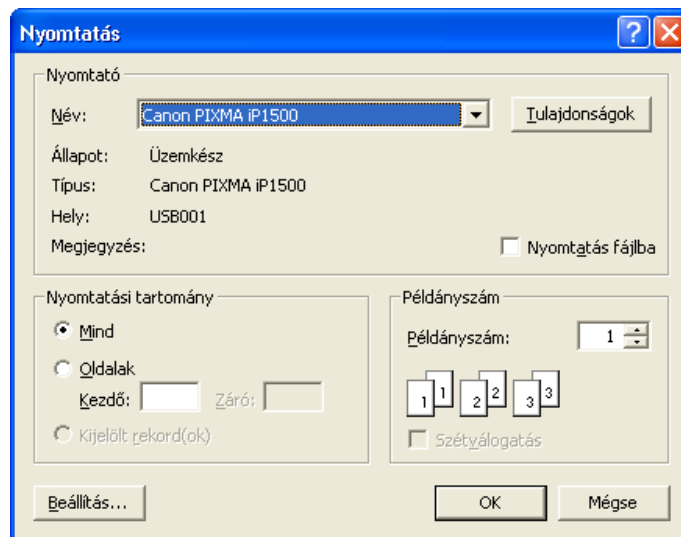
PRINTER LAYOUT

Before starting printing it is a good idea to check the settings. We can do this with the **Printer Layout** order in **File** menu. This operating mode has its own toolkit. This is where zooming, shrinking, sorting, etc. takes place.



PRINTING

We can display the **Print** panel with the **Print** order from the **File** menu, or with the Ctrl+P letter combination. By clicking on the **Properties** button we can modify the printing setting based on the printer's type. By pressing the **Settings** button a panel will appear in accordance with the page setup.



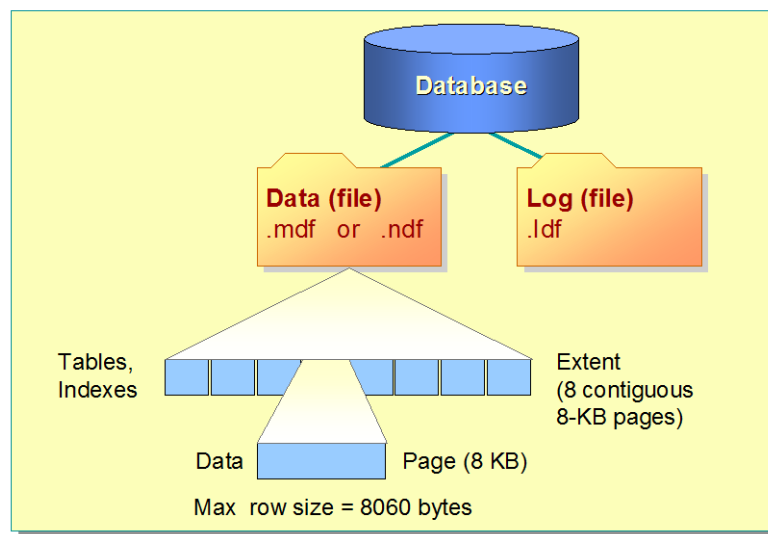
MSSQL Server

Commands

You can define a database by using SQL Server Enterprise Manager or the CREATE DATABASE statement in SQL Query Analyzer. The process of defining a database also creates a transaction log for that database.

Information about each database in SQL Server is stored in the sysdatabases table in the master database. Therefore, you must use the master database to define a database when you use Transact-SQL.

Defining a database is a process of specifying the name of the database and designating the size and location of the database files. When the new database is created, it is a duplicate of the model database. Any options or settings in the model database are copied into the new database.



When creating a database, it is important to understand how SQL Server stores data so that you can calculate and specify the amount of disk space to allocate for the database. Consider the following facts and guidelines about data storage:

All databases have a primary data file, identified by the .mdf file name extension, and one or more transaction log files, identified by the .ldf file name extension. A database also may have secondary data files, which are identified by the .ndf file name extension. These

physical files have both operating system file names and logical file names that you can use in Transact-SQL statements.

A database creation of the syntax:

```
CREATE DATABASE database_name
[ON
    { [PRIMARY] (NAME = logical_file_name,
        FILENAME = 'os_file_name'
        [, SIZE = size]
        [, MAXSIZE = {max_size|UNLIMITED}]
        [, FILEGROWTH = growth_increment] )
    } [,...n]
]
[LOG ON
    { (NAME = logical_file_name,
        FILENAME = 'os_file_name'
        [, SIZE = size]
        [, MAXSIZE = {max_size|UNLIMITED}]
        [, FILEGROWTH = growth_increment] )
    } [,...n]
]
[COLLATE collation_name]
```

Here's an example:

The name of the database: Sample, 10MB datafile and 3 MB log file.

```
CREATE DATABASE Sample
ON
    PRIMARY ( NAME=SampleData,
        FILENAME='c:\Program Files\
            Microsoft SQL Server\MSSQL\Data\Sample.mdf',
        SIZE=10MB,
        MAXSIZE=15MB,
        FILEGROWTH=20%)
LOG ON
    ( NAME=SampleLog,
        FILENAME='c:\Program Files\
            Microsoft SQL Server\MSSQL\Data\Sample.ldf',
        SIZE=3MB,
        MAXSIZE=5MB,
        FILEGROWTH=1MB)
COLLATE SQL_Latin1_General_CP1_CI_AS
```

When you create a database, you can set the following parameters:

PRIMARY

This parameter specifies the files in the primary filegroup. The primary filegroup contains all of the database system tables. It also contains all objects not assigned to user filegroups. Every database has one primary data file.

FILENAME

This parameter specifies the operating system file name and path for the file. The path in the `os_file_name` must specify a folder on the server on which SQL Server is installed.

SIZE

This parameter specifies the size of the data or log file. You can specify sizes in megabytes (MB)—the default value—or kilobytes (KB). The minimum size is 512 KB for both the data and log file. The size specified for the primary data file must be at least as large as the primary file of the model database. When adding a data file or log file, the default value is 1 MB.

MAXSIZE

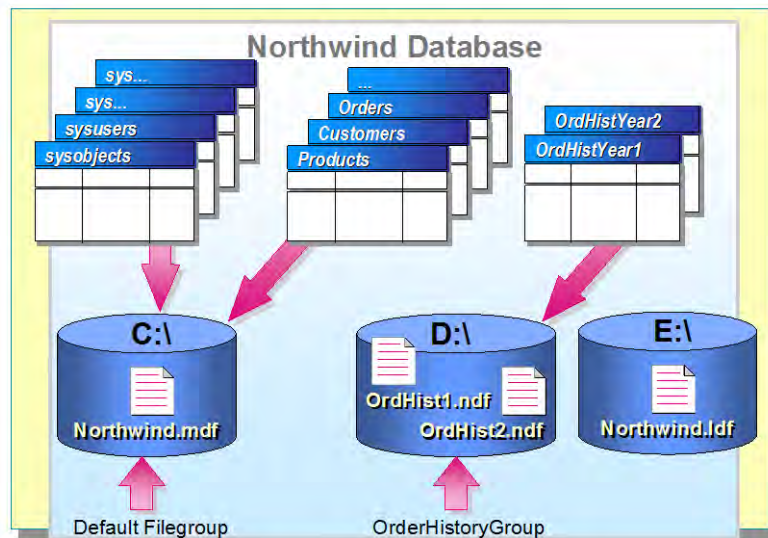
This parameter specifies the maximum size to which the file can grow. You can specify sizes in megabytes—the default value—or kilobytes. If you do not specify a size, the file grows until the disk is full.

FILEGROWTH

This parameter specifies the growth increment of the file. The FILEGROWTH setting for a file cannot exceed the MAXSIZE setting. A value of 0 indicates no growth. The value can be specified in megabytes—the default—in kilobytes, or as a percentage (%). The default value if FILEGROWTH is not specified is 10 percent, and the minimum value is 64 KB (one extent). The specified size is rounded to the nearest 64 KB.

Creating Filegroups

If your hardware setup includes multiple disk drives, you can locate specific objects and files on individual disks, grouping your database files into filegroups. Filegroups are named collections of files. SQL Server includes one filegroup as a default. You can create additional filegroups by using either the CREATE DATABASE or ALTER DATABASE statement.



With filegroups, you can locate specific objects on a specific file. In the illustration, the OrdHist1.ndf and OrdHist2.ndf files are placed on separate disk to separate files that are heavily queried from those that are heavily modified and to reduce disk drive contention. System administrators also can back up and restore individual files or filegroups instead of backing up or restoring an entire database. Backing up files or filegroups is necessary on large databases to have an effective backup and restore strategy.

The following example creates a user-defined filegroup in the Northwind database and adds a secondary data file to the user-defined filegroup.

```
ALTER DATABASE Northwind
ADD FILEGROUP OrderHistoryGroup
GO

ALTER DATABASE Northwind
ADD FILE
( NAME = 'OrdHistYear1',
  FILENAME = 'c:\ Program Files\
    Microsoft SQL Server\MSSQL\Data\OrdHist1.ndf',
  SIZE = 5MB),
TO FILEGROUP OrderHistoryGroup
```

Managing Databases

When data files grow, or when data modification activity increases, you may need to expand the size of the data or log files. You can manage database growth by using SQL Server Enterprise Manager or the ALTER DATABASE statement. You must be in the master database to use the ALTER DATABASE statement.

You can control the size of the database by:

- Configuring the database and log files to grow automatically.
- Manually increasing or decreasing the current or maximum size of existing database and log files.
- Manually adding secondary database and log files.

You can create secondary database files to expand the size of a database. Use secondary database files to place data files on separate physical disks when you do not use the disk-striping capabilities of RAID systems.

```
ALTER DATABASE database
{ ADD FILE < filespec > [ ,...n ] [ TO FILEGROUP filegroup_name ]
| ADD LOG FILE < filespec > [ ,...n ]
| REMOVE FILE logical_file_name [ WITH DELETE ]
| ADD FILEGROUP filegroup_name
| REMOVE FILEGROUP filegroup_name
| MODIFY FILE < filespec >
| MODIFY NAME = new_dbname
| MODIFY FILEGROUP filegroup_name
  {filegroup_property | NAME = new_filegroup_name }
| SET < optionspec > [ ,...n ] [ WITH < termination > ]
| COLLATE < collation_name >
}
```

The following example increases the current log size and adds a secondary data file to the Sample database.

```
ALTER DATABASE Sample
    MODIFY FILE ( NAME = 'SampleLog',
    SIZE = 15MB)

GO

ALTER DATABASE Sample
    ADD FILE

(NAME = 'SampleData2' ,
FILENAME='c:\Program Files\
Microsoft SQL Server\MSSQL\Data\Sample2.ndf',
SIZE=15MB ,
MAXSIZE=20MB)
```


Expanding a Transaction Log

When a database grows, or when data modification activity increases, you may need to expand the transaction log.

If your transaction log runs out of space, SQL Server cannot record transactions and does not allow changes to your database.

You can monitor the transaction log with SQL Server Enterprise Manager, the DBCC SQLPERF (LOGSPACE) statement, or Microsoft Windows® 2000 System Monitor.

You can monitor the transaction logs of individual databases by using SQL Server:Database object counters in System Monitor. These counters include ones listed in the following table.

Shrinking a Database or File

When too much space is allocated, or when space requirements decrease, you can shrink an entire database or specific data files in a database.

You can shrink an entire database by using SQL Server Enterprise Manager or by executing the Database Consistency Checker (DBCC) statement SHRINKDATABASE. This shrinks the size of all data files in the database.

SQL Server shrinks log files by using a deferred shrink operation, and does so as if all of the log files existed in one contiguous log pool. Log files are reset when the log is truncated; SQL Server attempts to shrink the truncated log files to as close to the targeted size as possible.

```
DBCC SHRINKDATABASE (database_name [, target_percent] [, {NOTRUNCATE | TRUNCATEONLY}])
```

This example shrinks the size of the SampleData so that the file will have free space of 30 percent.

```
DBCC SHRINKDATABASE (SampleData, 30)
```

You can shrink a data file in a database by using SQL Server Enterprise Manager or by executing the DBCC statement SHRINKFILE.

```
DBCC SHRINKFILE ({file_name | file_id} [, target_size] [,
{ EMPTYFILE | NOTRUNCATE | TRUNCATEONLY}])
```

This example shrinks the size of the sample data file to 10 MB.

```
DBCC SHRINKFILE (Sample, 10)
```

Autoshrink is not enabled by default. By setting the autoshrink database option to true, you can set a database option to recover unused space automatically. You can also change this option with SQL Server Enterprise Manager.

Dropping Databases

You can drop a database when you no longer need it. Dropping a database deletes the database and the disk files that the database uses.

You can drop databases by using SQL Server Enterprise Manager or by executing the DROP DATABASE statement.

```
DROP DATABASE database_name [,...n]
```

This example drops multiple databases by using one statement.

```
DROP DATABASE Northwind, pubs
```

When you drop a database, consider the following facts and guidelines:

With SQL Server Enterprise Manager, you can drop only one database at a time. With Transact-SQL, you can drop several databases at once. After you drop a database, every login ID that used that particular database as its default database will not have a default database.

Database management operations

Before you can create a table, you must define the data types for the table. Data types specify the type of information (characters, numbers, or dates) that a column can hold, as well as how the data is stored. Microsoft® SQL Server™ 2000 supplies various system data types. SQL Server also allows user-defined data types that are based on system data types.

Common data types	SQL Server system-supplied data types	ANSI synonym	Number of bytes
Integer	int	<i>integer</i>	4
	bigint	–	8
	smallint, tinyint	–	2, 1
Exact numeric	decimal[(p[, s])] numeric[(p[, s])]	<i>dec</i> –	2–17
Approximate numeric	float[(n)]	<i>double precision,</i> <i>float[(n)]</i> for <i>n=8-15</i>	8
	real	<i>float[(n)]</i> for <i>n=1-7</i>	4
Monetary	money, smallmoney	–	8, 4
Date and time	Datetime, smalldatetime	–	8 4
Character	char[(n)]	<i>character[(n)]</i>	0–8000
	varchar[(n)]	<i>char VARYING[(n)]</i>	
	text	<i>character VARYING[(n)]</i> –	0–2 GB
Unicode character	nchar[(n)] nvarchar[(n)] ntext	–	0–8000 (4000 karakter) 0–2 GB
Binary	binary[(n)]	–	0–8000
	varbinary[(n)]	<i>binary VARYING[(n)]</i>	
Image	image	–	0–2 GB
Global identifier	uniqueidentifier	–	16
Special	bit, cursor, uniqueidentifier	–	1, 0–8
	timestamp	rowversion	8
	sysname	–	256
	table	–	
	sql_variant	–	0–8016

Creating tables

After you define all of the data types for your table, you can create tables, add and drop columns, and generate column values.

When you create a table, you must specify the table name, column names, and column data types. Column names must be unique to a specific table, but you can use the same column

name in different tables within the same database. You must specify a data type for each column.

Syntax:

```
CREATE TABLE table_name
  column_name data type [COLLATE<collation_name>]
  [NULL | NOT NULL]
  | column_name AS computed_column_expression
  [,...n]
```

The following example creates the `dbo.CategoriesNew` table, specifying the columns of the table, a data type for each column, and whether that column allows null values.

```
CREATE TABLE dbo.CategoriesNew
(
  CategoryID      int IDENTITY (1, 1) NOT NULL,
  CategoryName    nvarchar(15) NOT NULL,
  Description      ntext NULL,
  Picture          image NULL)

```

Dropping a table removes the table definition and all data, as well as the permission specifications for that table.

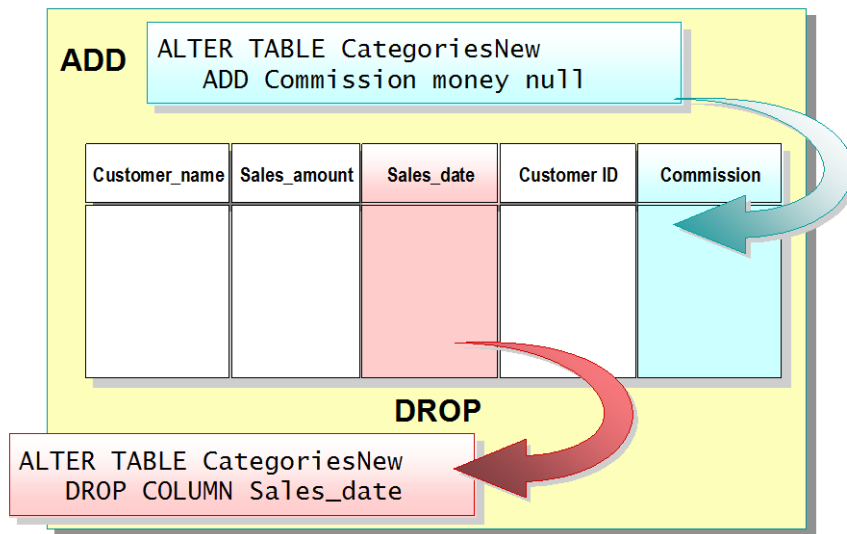
Before you can drop a table, you should remove any dependencies between the table and other objects. To view existing dependencies, execute the `sp_depends` system stored procedure.

Syntax:

```
DROP TABLE table_name [,...n]
```

Adding and Dropping a Column

Adding and dropping columns are two ways to modify tables.



Syntax:

```
ALTER TABLE table
    { | [ALTER COLUMN column_name ]
        | {
            ADD
                {
                    <column_definition> ::=
                        column_name data_type
                        { [NULL | NOT NULL]
                | DROP column column_name } [,...n]
```

The type of information that you specify when you add a column is similar to that which you supply when you create a table.

This example adds a column that allows null values.

```
ALTER TABLE CategoriesNew
ADD Commission money null
```

Dropped columns are unrecoverable. Therefore, be certain that you want to remove a column before doing so.

This example drops a column from a table.

```
ALTER TABLE CategoriesNew
DROP COLUMN Sales_date
```

All indexes and constraints that are based on a column must be removed before you drop the column.

Generating Column Values

Several features allow you to generate column values: the Identity property, the NEWID function, and the uniqueidentifier data type.

You can use the Identity property to create columns (referred to as identity columns) that contain system-generated sequential values identifying each row inserted into a table. An identity column is often used for primary key values.

Having SQL Server automatically provide key values can reduce costs and improve performance. It simplifies programming, keeps primary key values short, and reduces user-transaction bottlenecks.

```
CREATE TABLE table
(column_name data_type
[ IDENTITY [(seed, increment)]] NOT NULL )
```

Consider the following requirements for using the **Identity** property:

- Only one identity column is allowed per table.
- It must be used with integer (int, bigint, smallint, or tinyint), numeric, or decimal data types. The numeric and decimal data types must be specified with a scale of 0.
- It cannot be updated.
- You can use the IDENTITYCOL keyword in place of the column name in a query. This allows you to reference the column in the table having the Identity property without having to know the column name.
- It does not allow null values.

You can retrieve data from identity columns using the **@@identity** global variable, which determines the value of the last row inserted into an identity column during a session.

This example creates a table with two columns, StudentId and Name. The Identity property is used to increment the value automatically in each row added to the StudentId column. The seed is set to 100, and the increment value is 5. The values in the column would be 100, 105, 110, 115, and so on. Using 5 as an increment value allows you to insert records between the values at a later time.

```
CREATE TABLE Class
(StudentID int IDENTITY(100, 5) NOT NULL,
Name varchar(16))
```

Using the NEWID Function and the uniqueidentifier Data Type

The uniqueidentifier data type and the NEWID function are two features that are used together. Use these features when data is collated from many tables into a larger table and when uniqueness among all records must be maintained:

- The uniqueidentifier data type stores a unique identification number as a 16-byte binary string. This data type is used for storing a globally unique identifier (GUID).
- The NEWID function creates a unique identifier number that can store a GUID by using the uniqueidentifier data type.
- The uniqueidentifier data type does not automatically generate new IDs for inserted rows the way the Identity property does. To get new uniqueidentifier values, you must define a table with a DEFAULT constraint that specifies the NEWID function. When you use an INSERT statement, you must also specify the NEWID function.

In this example, the Customer table customer ID column is created with a uniqueidentifier data type, with a default value generated by the NEWID function. A unique value for the CustID column will be generated for each new and existing row.

```
CREATE TABLE Customer
(CustID uniqueidentifier NOT NULL DEFAULT NEWID(),
CustName char(30) NOT NULL)
```

Generating Scripts

When you create objects in a database, it is important to save all object definitions in a script file.

You can use SQL Server Enterprise Manager to document an existing database structure (schema) by generating it as one or more Transact-SQL scripts. These Transact-SQL scripts contain descriptions of the statements that were used to create a database and its objects.

You can generate:

- An entire database into a single script file.
- Table-only schema for one, some, or all tables in a database into one or more script files.
- Table and index schema into one script file, stored procedures into another script file, and defaults and rules into yet another script file.

Data is based on applying the following syntax:

A new record into table:

```
INSERT INTO table_name
```

```
VALUES [<values>]
```

In the following example we write a new record into the Categories table.

```
INSERT INTO Categories  
VALUES („Smith”, „$500”, „ID520444”, „none”)
```

We must also make sure that we are referring to an existing table. Volume and sequence of the data in () must same as in the table. Only if you want to add some values we must give name of the columns.

```
INSERT INTO Categories (‘Customer_name’, ‘CustomerID’)  
VALUES („Smith”, „ID520444”)
```

Of course, it is possible to delete data.

Synax:

```
DELETE FROM <table_name> WHERE <condition>
```

The WHERE clause must specify the conditions under which the deletion of rows you want to achieve.

```
DELETE FROM Categories WHERE ‘Customer_name’ = ‘Smith’
```

Finally, let's see how to modify data.

Syntax:

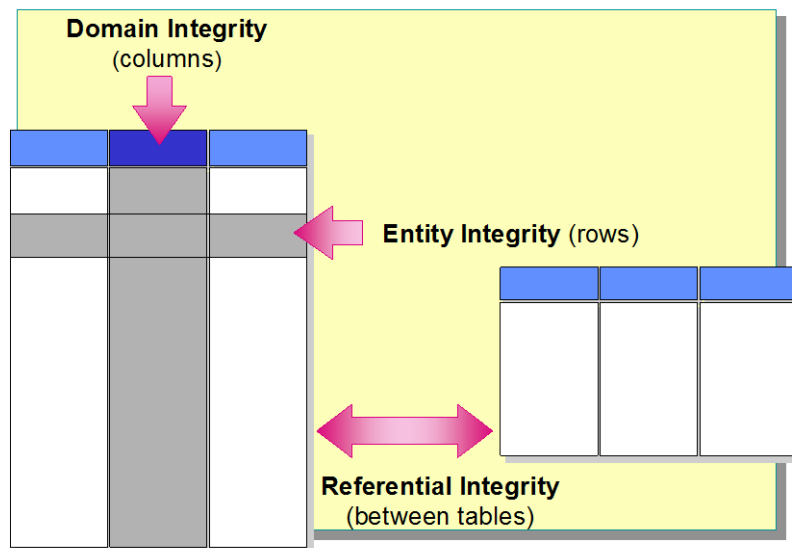
```
UPDATE <table_name> SET column_name='values'  
[, column_name='values',...]
```

In the following example we modify the Customer_name. The new name will be 'Bush', and the value of primary key is 17.

```
UPDATE Categories SET Customer_name = ‘Bush’  
WHERE CatID = 17
```


Types of Data Integrity

An important step in database planning is deciding the best way to enforce the integrity of the data. Data integrity refers to the consistency and accuracy of data that is stored in a database. The different types of data integrity are as follows.



Domain Integrity

Domain (or column) integrity specifies a set of data values that are valid for a column and determines whether null values are allowed. Domain integrity is often enforced through the use of validity checking and can also be enforced by restricting the data type, format, or range of possible values allowed in a column.

Entity Integrity

Entity (or table) integrity requires that all rows in a table have a unique identifier, known as the primary key value. Whether the primary key value can be changed, or whether the whole row can be deleted, depends on the level of integrity required between the primary key and any other tables.

Referential Integrity

Referential integrity ensures that the relationships among the primary keys (in the referenced table) and foreign keys (in the referencing tables) are always maintained. A row in a referenced table cannot be deleted, nor the primary key changed, if a foreign key refers to the row, unless the cascade action is permitted. You can define referential integrity relationships within the same table or between separate tables.

Defining Constraints

Constraints are the preferred method of enforcing data integrity. This section discusses how to determine the type of constraint to use, what type of data integrity that each type of constraint enforces, and how to define constraints.

Type of integrity	Constraint type
Domain	DEFAULT
	CHECK
	REFERENTIAL
Entity	PRIMARY KEY
	UNIQUE
Referential	FOREIGN KEY
	CHECK

Constraints are an ANSI-standard method of enforcing data integrity. Each type of data integrity — domain, entity, and referential — is enforced with separate types of constraints. Constraints ensure that valid data values are entered in columns and that relationships are maintained between tables. The following table describes the different types of constraints.

You create constraints by using the CREATE TABLE or ALTER TABLE statement.

You can add constraints to a table with existing data, and you can place constraints on single or multiple columns:

- If the constraint applies to a single column, it is called a column-level constraint.
- If a constraint references multiple columns, it is called a table-level constraint, even if it does not reference all columns in the table.

```
CREATE TABLE table_name
( { < column_definition >
  | < table_constraint > } [ ,...n ])
< column_definition > ::= { column_name data_type }
    [ [ DEFAULT constant_expression ]
    [ < column_constraint > ] [ ,...n]
< column_constraint > ::=
    [ CONSTRAINT constraint_name ]
        | [ { PRIMARY KEY | UNIQUE }
            [ CLUSTERED | NONCLUSTERED ] ]
        | [ [ FOREIGN KEY ]
            REFERENCES ref_table [ ( ref_column ) ]
```

```

        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]]
    | CHECK ( logical_expression ) }
< table_constraint > ::=
    [ CONSTRAINT constraint_name ]
    { [ { PRIMARY KEY | UNIQUE }
      [ CLUSTERED | NONCLUSTERED ]
      { ( column [ ASC | DESC ] [ ,...n ] ) } ]
    | FOREIGN KEY
      [ ( column [ ,...n ] ) ]
      REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
      [ ON DELETE { CASCADE | NO ACTION } ]
      [ ON UPDATE { CASCADE | NO ACTION } ]
    | CHECK ( search_conditions ) }

```

This example creates the Products table, defines columns, and defines constraints at both the column and table level.

```

USE Northwind
CREATE TABLE dbo.Products
(
    ProductID      int IDENTITY (1,1) NOT NULL,
    ProductName    nvarchar (40) NOT NULL,
    SupplierID     int      NULL,
    CategoryID     int      NULL,
    QuantityPerUnit nvarchar (20) NULL,
    UnitPrice      money     NULL      CONSTRAINT DF_Products_UnitPrice  DEFAULT(0),
    UnitsInStock   smallint  NULL      CONSTRAINT DF_Products_UnitsInStock
DEFAULT(0),
    UnitsOnOrder   smallint  NULL      CONSTRAINT DF_Products_UnitsOnOrder
DEFAULT(0),
    ReorderLevel   smallint  NULL      CONSTRAINT DF_Products_ReorderLevel
DEFAULT(0),
    Discontinued   bit       NOT NULL  CONSTRAINT DF_Products_Discontinued
DEFAULT(0),

    CONSTRAINT PK_Products PRIMARY KEY CLUSTERED (ProductID),

    CONSTRAINT FK_Products_Categories FOREIGN KEY (CategoryID)
REFERENCES dbo.Categories (CategoryID) ON UPDATE CASCADE,
    CONSTRAINT FK_Products_Suppliers  FOREIGN KEY (SupplierID)
REFERENCES dbo.Suppliers  (SupplierID) ON DELETE CASCADE,

    CONSTRAINT CK_Products_UnitPrice CHECK (UnitPrice >= 0),
    CONSTRAINT CK_ReorderLevel       CHECK (ReorderLevel >= 0),
    CONSTRAINT CK_UnitsInStock       CHECK (UnitsInStock >= 0),
    CONSTRAINT CK_UnitsOnOrder       CHECK (UnitsOnOrder >= 0)
)

```

Types of Constraints

DEFAULT Constraints

A DEFAULT constraint enters a value in a column when one is not specified in an INSERT statement. DEFAULT constraints enforce domain integrity.

```
[CONSTRAINT constraint_name]
    DEFAULT constant_expression
```

This example adds a DEFAULT constraint that inserts the UNKNOWN value in the dbo.Customers table if a contact name is not provided.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT DF_contactname DEFAULT 'UNKNOWN' FOR ContactName
```

CHECK Constraints

A CHECK constraint restricts the data that users can enter into a particular column to specific values. CHECK constraints are similar to WHERE clauses in that you can specify the conditions under which data will be accepted.

```
[CONSTRAINT constraint_name]
    CHECK (logical_expression)
```

This example adds a CHECK constraint to ensure that a birth date conforms to an acceptable range of dates.

```
USE Northwind
ALTER TABLE dbo.Employees
ADD
CONSTRAINT CK_birthdate
CHECK (BirthDate > '01-01-1900' AND BirthDate < getdate())
```

PRIMARY KEY Constraints

A PRIMARY KEY constraint defines a primary key on a table that uniquely identifies a row. It enforces entity integrity.

```
[CONSTRAINT constraint_name]
    PRIMARY KEY [CLUSTERED | NONCLUSTERED]
    { ( column[,...n] ) }
```

This example adds a constraint that specifies that the primary key value of the dbo.Customers table is the customer identification and indicates that a nonclustered index will be created to enforce the constraint.

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT PK_Customers
PRIMARY KEY NONCLUSTERED (CustomerID)
```

UNIQUE Constraints

A UNIQUE constraint specifies that two rows in a column cannot have the same value. This constraint enforces entity integrity with a unique index.

A UNIQUE constraint is helpful when you already have a primary key, such as an employee number, but you want to guarantee that other identifiers, such as an employee's driver's license number, are also unique.

```
[CONSTRAINT constraint_name]
    UNIQUE [CLUSTERED | NONCLUSTERED]
        { ( column[,...n] ) }
```

This example creates a UNIQUE constraint on the company name in the dbo.Suppliers table.

```
USE Northwind
ALTER TABLE dbo.Suppliers
ADD
CONSTRAINT U_CompanyName
UNIQUE NONCLUSTERED (CompanyName)
```

FOREIGN KEY Constraints

A FOREIGN KEY constraint enforces referential integrity. The FOREIGN KEY constraint defines a reference to a column with a PRIMARY KEY or UNIQUE constraint in the same, or another table.

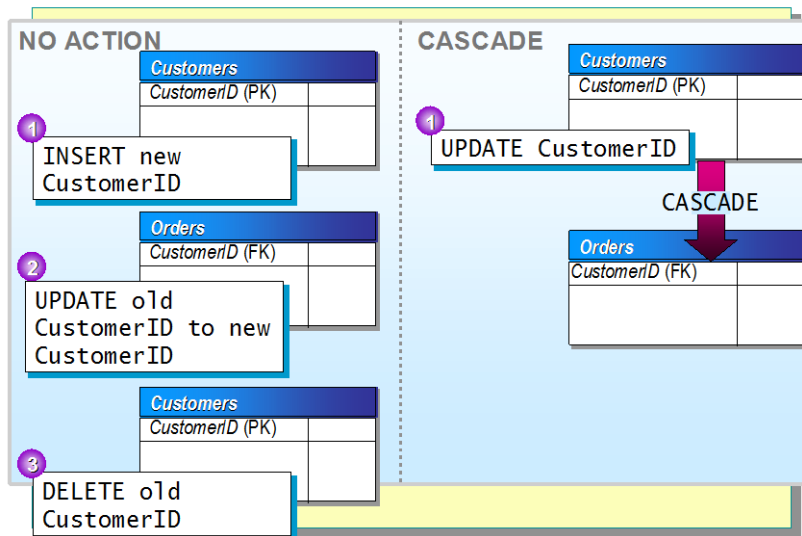
```
[CONSTRAINT constraint_name]
    [FOREIGN KEY] [(column[,...n])]
        REFERENCES ref_table [(ref_column [,...n])].
```

This example uses a FOREIGN KEY constraint to ensure that customer identification in the dbo.Orders table is associated with a valid identification in the dbo.Customers table.

```
USE Northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
FOREIGN KEY (CustomerID)
REFERENCES dbo.Customers(CustomerID)
```

Cascading Referential Integrity

The FOREIGN KEY constraint includes a CASCADE option that allows any change to a column value that defines a UNIQUE or PRIMARY KEY constraint to automatically propagate the change to the foreign key value. This action is referred to as cascading referential integrity.



The REFERENCES clauses of the CREATE TABLE and ALTER TABLE statements support ON DELETE and ON UPDATE clauses. These clauses allow you to specify the CASCADE or NO ACTION option.

```
[CONSTRAINT constraint_name]
    [FOREIGN KEY] [(column[,...n])]
        REFERENCES ref_table [(ref_column [,...n])].
    [ ON DELETE { CASCADE | NO ACTION } ]
    [ ON UPDATE { CASCADE | NO ACTION } ]
```

NO ACTION specifies that any attempt to delete or update a key referenced by foreign keys in other tables raises an error and the change is rolled back. NO ACTION is the default.

If CASCADE is defined and a row is changed in the parent table, the corresponding row is then changed in the referencing table.

Disabling Constraint Checking on Existing Data

When you define a constraint on a table that already contains data, SQL Server checks the data automatically to verify that it meets the constraint requirements. However, you can disable constraint checking on existing data when you add a constraint to the table.

```
ALTER TABLE table
[WITH CHECK|WITH NOCHECK]
ADD CONSTRAINT constraint
    [FOREIGN KEY] [(column[,...n])]
    REFERENCES ref_table [(ref_col [,...n])]
    [CHECK (search_conditions)]
```

In this example, you add a FOREIGN KEY constraint that verifies that all employees are associated with a valid manager. The constraint is not enforced on existing data at the time that the constraint is added.

```
USE Northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
ADD CONSTRAINT FK_Employees_Employees
FOREIGN KEY (ReportsTo)
REFERENCES dbo.Employees(EmployeeID)
```

Indexes in the Database

Using indexes can greatly improve database performance. This section introduces basic index concepts and discusses when and why indexes are used.

How Data Is Accessed

SQL Server accesses data in one of two ways:

- Scanning all of the data pages of tables—called a table scan. When SQL Server performs a table scan, it:
 - Starts at the beginning of the table.
 - Scans from page-to-page through all of the rows in the table.
 - Extracts the rows that meet the criteria of the query.
- Using indexes. When SQL Server uses an index, it:
 - Traverses the index tree structure to find rows that the query requests.
 - Extracts only the needed rows that meet the criteria of the query.

SQL Server first determines whether an index exists or not. Then, the query optimizer, the component responsible for generating the optimum execution plan for a query, determines whether scanning a table or using the index is more efficient for accessing data.

Creating Indexes

Now that you are familiar with the different index architectures, we will discuss creating and dropping indexes and obtaining information on existing indexes.

You create indexes by using the CREATE INDEX statement and can remove them by using the DROP INDEX statement.

Using the CREATE INDEX Statement

Use the CREATE INDEX statement to create indexes. You also can use the Create Index Wizard in SQL Server Enterprise Manager. When you create an index on one or more columns in a table, consider the following facts and guidelines:

- SQL Server automatically creates indexes when a PRIMARY KEY or UNIQUE constraint is created on a table. Defining a PRIMARY KEY or UNIQUE constraint is preferred over creating standard indexes.
- You must be the table owner to execute the CREATE INDEX statement.
- Indexes can be created on views.
- SQL Server stores index information in the sysindexes system table.
- Before you create an index on a column, determine whether indexes already exist on that column.
- Keep your indexes small by defining them on columns that are small in size. Typically, smaller indexes are more efficient than indexes with larger key values.
- Select columns on the basis of uniqueness so that each key value identifies a small number of rows.
- When you create a clustered index, all existing nonclustered indexes are rebuilt.

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]  
INDEX index_name ON { table | view } ( column [ ASC | DESC ] [ ,...n ] )  
[WITH  
[PAD_INDEX ]  
[[,] FILLFACTOR = fillfactor ]  
[[,] IGNORE_DUP_KEY ]  
[[,] DROP_EXISTING ]  
[[,] STATISTICS_NORECOMPUTE ]  
[[,] SORT_IN_TEMPDB ]  
]  
[ON filegroup ]
```

This example creates a clustered index on the LastName column in the Employees table.

```
CREATE CLUSTERED INDEX CL_lastname  
ON employees(lastname)
```

Use the DROP INDEX statement to remove an index on a table.

```
DROP INDEX 'table.index | view.index' [, ...n ]
```

This example drops the cl_lastname index from the Member table.

```
USE Northwind  
DROP INDEX employees.CL_lastname
```

Creating Unique Indexes

A unique index ensures that all data in an indexed column is unique and does not contain duplicate values.

Unique indexes ensure that data in indexed columns is unique. If the table has a PRIMARY KEY or UNIQUE constraint, SQL Server automatically creates a unique index when you execute the CREATE TABLE or ALTER TABLE statement.

This example creates a unique, nonclustered index named U_CustID on the Customers table. The index is built on the CustomerID column. The value in the CustomerID column must be a unique value for each row of the table.

```
USE Northwind
CREATE UNIQUE NONCLUSTERED INDEX U_CustID
ON customers(CustomerID)
```

Maintaining Indexes

You must maintain indexes after you create them to ensure optimal performance. Over time, data becomes fragmented. You manage data fragmentation according to your organization's environment.

SQL Server provides an Index Tuning Wizard that tracks the usage of your indexes automatically and assists with maintaining and creating indexes that perform optimally.

You also can use various options and tools to help you rebuild indexes and verify index optimization.

DBCC INDEXDEFRAG Statement

As data in a table changes, the indexes on the table sometimes become fragmented. The DBCC INDEXDEFRAG statement can defragment the leaf level of clustered and nonclustered indexes on tables and views. Defragmenting arranges the pages so that the physical order of the pages matches the left-to-right logical order of the leaf nodes. This rearrangement improves index-scanning performance.

Index Defragmenting vs. Index Rebuilding

The time required to defragment is related to the amount of fragmentation. A very fragmented index might require more time to defragment than to rebuild. A relatively unfragmented index defragments faster than rebuilding a new index.

Using the DBCC INDEXDEFRAG statement does not improve performance when indexes are physically defragmented on disk. To physically defragment an index, rebuild the index.

```
DBCC INDEXDEFRAG
    ( { database_name | database_id | 0 }
      , { table_name | table_id | 'view_name' | view_id }
      , { index_name | index_id }
    ) [ WITH NO_INFOMSGS ]
```

This example executes the DBCC INDEXDEFRAG statement on the mem_no_CL index of the Member table in the credit database.

```
DBCC INDEXDEFRAG(credit, member, mem_no_CL)
```

Triggers

A trigger is a special kind of stored procedure that executes whenever an attempt is made to modify data in a table that the trigger protects. Triggers are tied to specific tables.

Triggers are best used to maintain low-level data integrity, not to return query results. The primary benefit of triggers is that they can contain complex processing logic. Triggers can cascade changes through related tables in a database, enforce more complex data integrity than a CHECK constraint, define custom error messages, maintain denormalized data, and compare before and after states of data under modification.

Defining Triggers

Create triggers by using the CREATE TRIGGER statement. The statement specifies the table on which a trigger is defined, the events for which the trigger executes, and the particular instructions for the trigger.

```
CREATE TRIGGER [owner.] trigger_name
ON [owner.] table_name
[WITH ENCRYPTION]
{FOR | AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE}
AS
[IF UPDATE (column_name)...]
[{AND | OR} UPDATE (column_name)...]
sql_statements}
```

When a FOR UPDATE action is specified, the IF UPDATE (column_name) clause can be used to focus action on a specific column that is updated.

Both FOR and AFTER are equivalent syntax creating the same type of trigger, which fires after the initiating (INSERT, UPDATE, or DELETE) action.

INSTEAD OF triggers cancel the triggering action and perform a new function instead.

When you create a trigger, information about the trigger is inserted into the sysobjects and syscomments system tables. If a trigger is created with the same name as an existing trigger, the new trigger will overwrite the original one.

The following example creates a trigger on the Employees table that prevents users from deleting more than one employee at a time. The trigger fires every time a record or group of records are deleted from the table. The trigger checks the number of records being deleted by querying the Deleted table. If more than one record is being deleted, the trigger returns a custom error message and rolls back the transaction.

```
Use Northwind
GO

CREATE TRIGGER Empl_Delete ON NewEmployees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 1
BEGIN
    RAISERROR(
        'You cannot delete more than one employee at a time.',
        16, 1)
    ROLLBACK TRANSACTION
END
```

The following DELETE statement fires the trigger and prevents the transaction.
DELETE FROM Employees WHERE EmployeeID > 6

The following DELETE statement fires the trigger and allows the transaction.
DELETE FROM Employees WHERE EmployeeID = 6

Altering and Dropping Triggers

If you must change the definition of an existing trigger, you can alter it without having to drop it.

```
ALTER TRIGGER trigger_name
ON table
[WITH ENCRYPTION]
{{FOR {[,] [DELETE] [,] [UPDATE] [,] [INSERT]}}
[NOT FOR REPLICATION]
AS
sql_statement [...n] }
|
{FOR {[,] [INSERT] [,] [UPDATE]}}
[NOT FOR REPLICATION]
AS
IF UPDATE (column)
[{AND | OR} UPDATE (column) [,...n]]
sql_statement [...n] }
}
```

This example alters the delete trigger created in the previous example. New trigger content is provided, which changes the delete limit from one record to six records.

```
Use Northwind
GO

CREATE TRIGGER Emp1_Delete ON Employees
FOR DELETE
AS
IF (SELECT COUNT(*) FROM Deleted) > 6
BEGIN
    RAISERROR(
        'You cannot delete more than six employees at a time.',
        16, 1)
    ROLLBACK TRANSACTION
END
```

You can disable or enable a specific trigger, or all triggers on a table. When a trigger is disabled, it is still defined for the table; however, when an INSERT, UPDATE, or DELETE statement is executed against the table, the actions in the trigger are not performed until the trigger is re-enabled.

You can enable or disable triggers in the ALTER TABLE statement.

```
ALTER TABLE table
    {ENABLE | DISABLE} TRIGGER
    {ALL | trigger_name[,...n]}
```

Dropping a Trigger

You can remove a trigger by dropping it. Triggers are dropped automatically whenever their associated tables are dropped.

Permission to drop a trigger defaults to the table owner and is non-transferable. However, members of the system administrators (sysadmin) and database owner (db_owner) roles can drop any object by specifying the owner in the DROP TRIGGER statement.

```
DROP TRIGGER trigger_name
```

The trigger in this example was created to update a column (UnitsInStock) in the Products table whenever a product is ordered (whenever a record is inserted into the Order Details table). The new value is set to the previous value minus the ordered amount.

```
USE Northwind
CREATE TRIGGER OrdDet_Insert
ON [Order Details]
FOR INSERT
AS
```

```

UPDATE P SET
UnitsInStock = (P.UnitsInStock - I.Quantity)
FROM Products AS P INNER JOIN Inserted AS I
ON P.ProductID = I.ProductID

```

The trigger in this example was created to update the Discontinued column in the Products table whenever a category is deleted (whenever a record is deleted from the Categories table). All affected products are marked as 1, indicating they are discontinued.

```

USE Northwind
CREATE TRIGGER Category_Delete
ON Categories
FOR DELETE
AS
UPDATE P SET Discontinued = 1
FROM Products AS P INNER JOIN deleted AS d
ON P.CategoryID = d.CategoryID

```

This example prevents a user from modifying the EmployeeID column in the Employees table.

```

USE Northwind
GO
CREATE TRIGGER Employee_Update
ON Employees
FOR UPDATE
AS
IF UPDATE (EmployeeID)
BEGIN TRANSACTION
RAISERROR ('Transaction cannot be processed.\
***** Employee ID number cannot be modified.', 10, 1)
ROLLBACK TRANSACTION
END

```

Some useful system functions

@@identity

The system functions, system data can be provided, without select. We can ask back the value of identity filed with the @@identity function. This function give me the last value, which the system generated in the table. There is the best if we use it in the stored procedure.

```

Create table test (id int identity primary key)
DECLARE @ident int
INSERT INTO test DEFAULT VALUES
Set @ident = @@identity

```

@@rowcount

This system function returns the last query affected rows number.

```
Declare @rc int
Select * from test
Select @rc = @@rowcount
Print 'the rows count: '+cast(@rc as varchar(10))
```

Cursors

The cursor is used to a query we can go through the line. Cursors within the stored procedure are used for query processing.

```
DECLARE @keycol AS INT, @filler AS CHAR(200);
DECLARE C CURSOR
  FOR SELECT keycol, filler FROM dbo.T1;
OPEN C
FETCH NEXT FROM C INTO @keycol, @filler;
WHILE @@fetch_status = 0
BEGIN
  --      Process data here FETCH NEXT FROM C INTO @keycol, @filler;
END
CLOSE C;
DEALLOCATE C;
```

```
create procedure getSales
as
declare @location varchar(120)
declare @sale_name varchar(120)
declare saleCursor cursor for
select Name, Location
from Sale
open saleCursor -- most hajtódik csak végre a SELECT utasítás
fetch next from saleCursor -- a FETCH utasítással tudjuk kiolvasni a sorokat
into @sale_name, @location
print 'SaleName = ' + convert(nvarchar,@sale_name)
print 'Location = ' + convert(nvarchar,@location)
while @@fetch_status = 0
begin
  fetch next from saleCursor
  into @sale_name, @location
  print 'SaleName = ' + convert(nvarchar,@sale_name)
  print 'Location = ' + convert(nvarchar,@location)
end
close saleCursor
deallocate saleCursor
```

Stored Procedures

We can create stored procedures much the same way as any other object in a database. The syntax for creating the following:

```
Create procedure|proc name  
[Params name data type] [= default value]  
[OUT Params name data type [= default value]]  
AS  
<code>
```

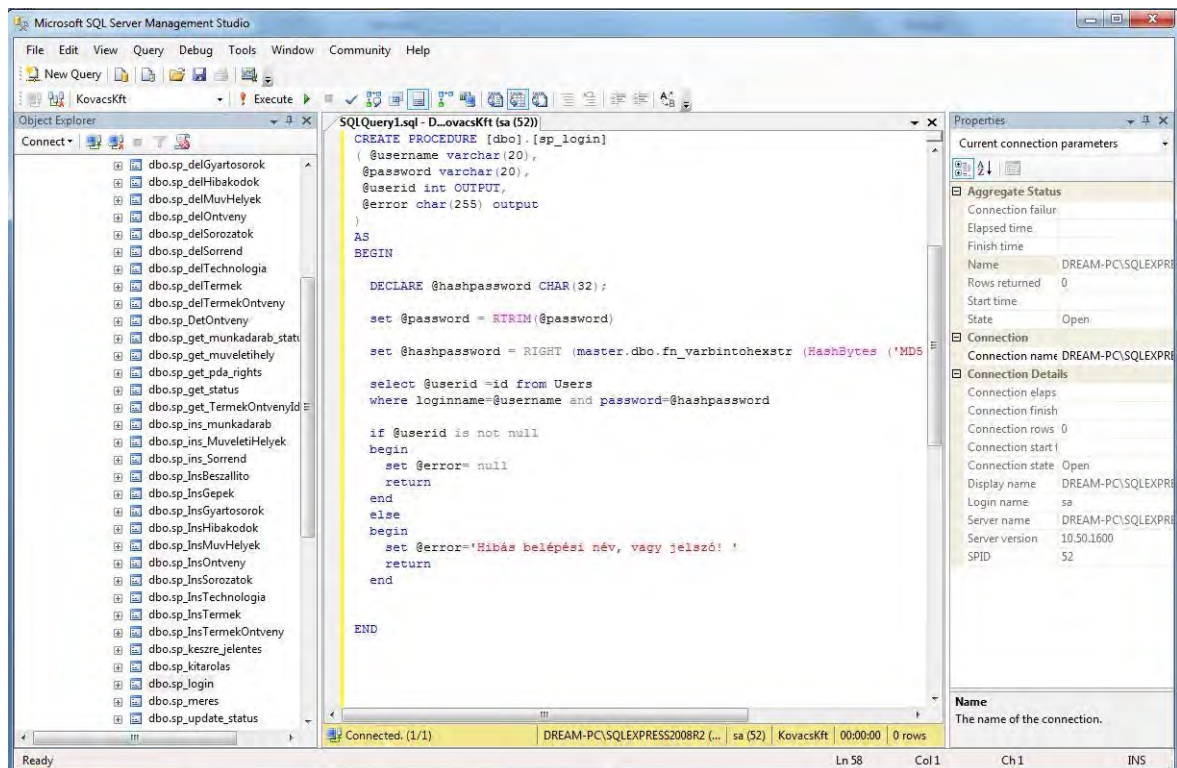
For example, a very basic level of stored procedure:

```
Create proc spShip  
AS  
select * from Ship
```

The created stored procedures are stored in the database compiled. Running:

```
exec spShip
```

The stored procedure name is followed by a list of parameters, which can be omitted. The parameters make flexible the using of stored procedures. We can create the procedures simply in SQL Server Management Studio.



Modify the stored procedure

It's possible with the ALTER PROCEDURE command. The procedure must exist.

```
Alter procedure|proc name  
[Params name data type] [= default value]  
[OUT Params name data type [= default value]]  
AS  
<code>
```

For example:

```
Alter proc spShip  
AS  
select * from Ship order by 1
```

Existing procedure can be deleted from the database with this command:

```
DROP PROCEDURE <name of procedure>
```

Parameters

The parameters increase the usability of stored procedures. Stored procedures accept data in the form of input parameters. The result – calculated data, search results etc – is returned by output parameters.

The parameters declaration is needed the name and the data type. It's possible default value and direction. IN or OUT.

Syntax:

```
@name of parameter [AS] data type [= default|NULL] [VARYING] [OUTPUT|OUT]
```

The first character of the name of parameter must be '@' and the name can't contain spaces.

```
Create procedure spInsertShipper  
@CompanyName nvarchar(40),  
@Phone nvarchar(24)  
AS  
INSERT INTO Shippers VALUES  
(@CompanyName, @Phone)
```

This procedure inserts a new record into the Shippers table. The table is in our sample database, Northwind.

Running:

```
exec spInsertShipper 'Apple Trading Co.', '0655/555-555'
```

It is necessary that the number, type and order of the actual parameters comply with the characteristics of formal parameters. A lot of errors can be avoided by using Default values.

```
Create Proc spIns
  @custom int = 1,
  @nev varchar(30),
  @kapott_id int OUT
AS
Insert into ord values (@custom, @nev)
Select @kapott_id = @@identity
```

```
CREATE PROCEDURE [dbo].[sp_get_TermekOntvenyId]
(
    @TermekId int,
    @OntvenyId int,
    @TeOnId int OUTPUT
)
AS
BEGIN
    Select @TeOnId = T0Id from tzsTermekOntveny
        where (TermekId = @TermekId) and (OntvenyId = @OntvenyId)

    if (@TeOnId is null)
    BEGIN
        set @TeOnId = -1
        return
    END
END
```