Курс Информатика част II

Файла може да се изтегли от адрес
http://zhelezov.atwebpages.com/Informatics2_Lectures.pdf с ограничение Read Only

Съдържание

1. Увод

В много случаи при създаване на локални и интернет приложения се налага продължително съхраняване на информация с възможност за нейното извличане и актуализиране от потребителите на сайта. Въпреки че по принцип е възможно да се съхранява такава информация и в текстови файлове, за предпочитане е да се работи с бази от данни, за които програмните среди предлагат добре развити средства за проектиране. В настоящия курс се изучават моделите на базите от данни (БД), системите за тяхното управление, SQL като език за интегрирани заявки към БД, и по-конкретно MS Access като система за управление на бази от данни.
По-долу са дадени дефиниции на някои основни понятия, свързани с базите от данни

2. Основни понятия и дефиниции

2.1. База от данни

Базата от данни (БД) представлява колекция[1] от логически свързани данни в конкретна предметна област, които са структурирани по определен начин, и се използват за задоволяване на информационните потребности на потребителите.

2.2. Системи за управление на бази от данни

Системата за управление на бази от данни СУБД (на английски: Database management system, DBMS) е набор от компютърни програми, контролиращи изграждането, поддръжката и използването на бази от данни. Примери за такива системи са Oracle, Microsoft SQL Server, PostgreSQL, Access. Някои от СУБД са потребителски ориентирани програмни среди, а други са по-скоро езици за създаване и описване на бази от данни. Програмата Microsoft Access например е типичен пример на потребителски ориентирана програма за управление на бази от данни.

[1] Систематизирана съвкупност от обекти, обединени по определен признак, притежаващи вътрешна цялост и принадлежащи на определен собственик

Макар че в Microsoft Access е възможно и програмиране с езика SQL, малки бази от данни могат да бъдат създавани и без програмиране като се използват инструментите, които Access предоставя.

3. Модели на данни

Всяка база от данни се състои от обекти, които са свързани помежду си. Различните начини, използвани за описване на връзките между данните, се отразяват на организацията на данните, наречена модел на данните или логическа структура на БД.

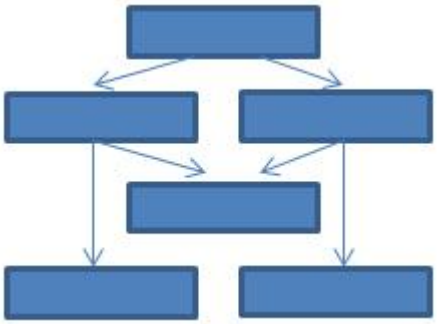
3.1. Йерархичен модел

Това е най-старата форма на модел на база данни. Той е разработен от IBM за IMS (система за управление на информацията). В йерархичният модел всички записи са в отношение на йерархична подчиненост – всеки запис може да има произволен брой наследници, но само един родител. Това позволява данните да бъдат представени дървовидно. Корен на дървото е една таблица. Търсенето на данни става като се обхожда дървото. Предимство на йерархичния модел е бързият и опростен достъп до данните, недостатък – съхраняване на излишна информация за осъществяване на връзките между данните., което води до необходимост от повече памет, усложняване на софтуера и др.

3.2. Мрежов модел

В мрежовият модел обектите се обединяват в мрежа, в която всеки запис в БД може да се свърже с всеки друг запис от БД. Всяка съвкупност от записи може да се включи към една или повече мрежи. Предимство – бърз

достъп до данните, недостатък – работата с мрежова БД е сложна и изисква предварително специална подготовка.



3.3. Модел на релационните бази от данни

Най-разпространеният модел за организация на база от данни е релациония модел. За пръв път той е формулиран от д-р E. F. Codd през 1970г. Д-р Код формулира известните дванадесет правила за този релационен модел, (който всъщност за тринадесет, тъй като номерирането им започва от 0 и завършва на 12). Сърцевината на релационният модел е вложена най-вече в първите три правила:

0) Релационните системи за управление на бази от данни (СУБД) трябва да могат да управляват напълно БД чрез техните релационни възможности.

1) **Правило за информацията**: Цялата информация за дадена релационна БД (РБД), включително имената на таблиците и колоните, се представя явно като стойности в таблиците.

2) **Гарантиран достъп**: Всяка стойност в РБД задължително трябва да бъде достъпна чрез комбинация от име на таблицата, стойност на първичен ключ и име на колона.

При релационния модел данните са разделени в множества, които имат структурата на таблица. Релационната таблица се състои от отделни елементи с данни, наречени полета. Едно множество от полета образува *кортеж* - запис в таблицата. По-долу ще разгледаме по-подробно елементите на релационният модел.

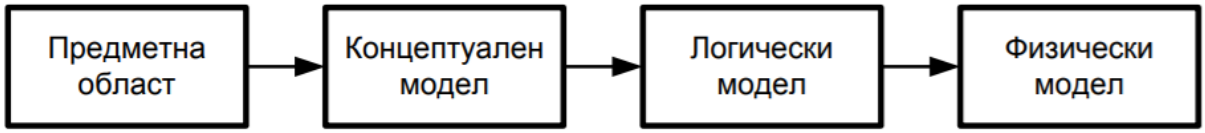
3.4. Обектноориентиран модел

Данните се разделят на класове от обекти, различаващи се по структура на включените обекти, по действията, извършвани с тях и по събитията, в които участват. Този модел е и най-новият. Той е най-близък до обектно ориентираното програмиране. Съвременните интегрирани среди за разработка (на английски: integrated development environment, IDE) на софтуерни приложения като Microsoft Visual Studio например, предоставят помощни средства за получаване на обектноориентиран модел на БД от нейния релационен модел – като например Object Relation Manager (ORM).

4. Проектиране на бази данни

Процесът на проектиране на БД включва:

1) проектиране на концептуален модел – представяне на всички обекти и взаимовръзките между тях;
2) проектиране на логически модел – представяне на обектите и взаимовръзките с помощта на някой от моделите на данни: мрежов, йерархичен или релационен; представяне на графически модели, изхождайки от модела на данни;
3) проектиране на физически модел – проектиране на методите за съхраняване и достъп до данните, а също и на методите за достъп до операционната система (ОС);



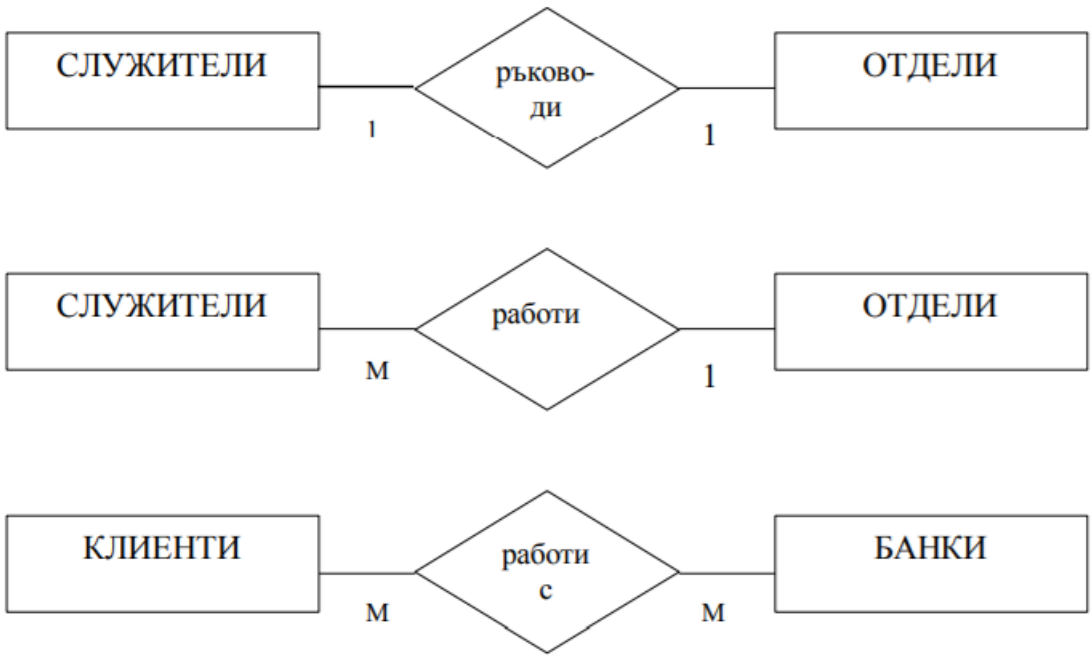
Концептуална схема ще наричаме универсално представяне на структурата на данните в рамките на определена предметна област, независимо от крайната реализация на базата данни и апаратната платформа.

4.1. Концептуален модел „Същности-връзки” (ER-модел)

Един от най-популярните концептуални модели на данни е моделът „Същности-връзки”, накратко наричан ER-модел (Entity-Relationship). Моделът е предложен от Питър Чен (Chen) през 1976 г. Моделирането на предметната област се базира на използване на графични диаграми, в

които обектите и връзките между тях се представят с графични символи. Моделът ER е доразвит от множество разработчици и днес съществуват редица негови разновидности: модел на ORACLE; модел на Vantage Team Builder; модел на Баркер; метод IDEF1, разработен от Т. Ремей (T.Ramey), и неговата доразвита версия IDEF1X (използва се в ERwin, Design/IDEF и др.); метод IE (Information Engineering) и т. н. Разликите между отделните модели се състоят в графичното изобразяване на обектите, атрибутите и връзките, както и в степента на отразяването на особеностите на обектите, атрибутите и връзките.

В диаграма ER (същности – връзки) обектите се представят като именувани правоъгълници, връзките (отношенията) - чрез именован ромб, който е свързан с класовете обекти чрез линии, а атрибутите (свойствата) – чрез именувана елипса [4] (фиг. 2).



Фиг. 2 Примери за представяне на различни видове връзки между обекти чрез диаграма „същности-връзки”

Между две същности е възможно да съществуват следните видове връзки:

- Един към един (1:1) - На един екземпляр от едната същност съответства точно 1 екземпляр от другата същност и обратно. Примери: клас и класен ръководител; директор и училище

- Един към много (1:M) На един екземпляр от едната същност съответстват много (повече от един) екземпляри от другата същност. На един екземпляр от втората същност съответства един екземпляр от първата същност. Примери: ученици и класни ръководители (един ученик има само един класен ръководител, но един класен ръководител ръководи много ученици); книги и издателства (определена книга се издава от точно едно издателство, но едно издателство издава много книги)
- Много към много (N:M) На един екземпляр от едната същност съответстват много (повече от един) екземпляри от другата същност и обратно. Това е най-често срещания вид връзка. Примери: ученици – учители (един ученик се обучава от няколко учители и един учител обучава няколко ученици); книги- автори; артисти – филми.

Преход от концептуален модел към логически модел

За преход от концептуален модел към логически модел се използват определени формални правила. Например, за преход от ER модел към релационна схема се използват следните правила:

- **Стъпка 1**. Всяка проста същност се превръща в таблица. Простата същност е същност, която не е подтип и няма подтипове. Името на същността става име на таблицата.
- **Стъпка 2**. Всеки атрибут (свойство) на същността става възможна колона със същото име; може да се избере и поточен формат. Стълбовете, съответстващи на незадължителните атрибути, могат да съдържат неопределени стойности, а стълбовете, съответстващи на задължителните атрибути – не могат.
- **Стъпка 3**. Компонентите на уникалния идентификатор на същността се превръщат в първичен ключ на таблицата. Ако има няколко възможни уникални идентификатора, избира се най-използвания.
- **Стъпка 4**. Връзките много към много (и един към много) стават външни ключове, като се правят копия на уникалния идентификатор от страната „един” и съответните колони изграждат външния ключ. Незадължителните връзки съответстват на колони, допускащи

неопределени стойности, а задължителните връзки – на колони, недопускащи неопределени стойности.

- **Стъпка 5**. Създават се индекси за първичния ключ (като уникален индекс), за външните ключове и за тези атрибути, на които се предполага, че ще се базират заявките.

4.2. Етапи при проектиране на релационни БД

Проектирането на една релационна БД включва следните етапи:
1) Определяне на целите, на които ще служи БД.
2) Определяне на таблиците, които са нужни.
3) Определяне на полетата във всяка таблица.
4) Определяне на първичните и външните ключове.
5) Определяне на връзките между таблиците

4.2.1. Определяне на целите

Изграждането на една БД зависи от целите, които си поставяме.

Пример 1: Разработва се БД за стоките, които продава малка дистрибуторска фирма. Всички данни се съхраняват на един от компютрите в офиса. Достъпът до данните се осъществява от няколко служители чрез локална мрежа. Тук акцентът ще се постави върху бързия достъп до данните и възможността за едновременна работа на няколко потребители. Надеждната защита на данните от неоторизиран достъп не е от съществено значение.

Акценти: бърз достъп, възможност за едновременна работа

Пример 2: Разработва се БД за голяма фирма с офиси по целия свят. В този случай трябва да се осигурят: надеждна защита на данните и различни права на достъп за различните категории служители; дублиране на критичната информация на няколко места в мрежата, бърз достъп до данните и др.

Акценти: надеждна защита на данните, различни права на потребители

4.2.2. Определяне на таблиците

Обикновено в една таблица се записват данните, които се отнасят за един клас от обекти. Колко таблици ще има в една РБД, зависи от броя на описваните класове от обекти. Например, в БД за работата на библиотека ще има таблици за: книгите, авторите и читателите; в БД за дейността на едно училище ще има таблици за: учениците, класовете, учителите, предметите, оценките и др.

При неправилно определяне на таблиците е възможно възникването на следните проблеми, наречени *аномалии*:

1. **Аномалия на излишеството** – едни и същи данни се повтарят многократно, изразходвайки излишни ресурси за съхраняването си;
2. **Аномалия на обновяването** – ако се промени някой от атрибутите, който е дублиран, то е необходимо да се обновяват всички записи, съдържащи този атрибут, за да се осигури непротиворечивост на данните;
3. **Аномалия на включването** – нов запис може да бъде включен, само ако всички стойности на атрибутите му са дефинирани;
4. **Аномалия на изключването** – изтриването на едни данни води до изтриването на други (напр., ако се изключат всички стоки на даден доставчик, данните за доставчика се изтриват);

За да изясним горните проблеми, ще разгледаме следния пример. В една училищна информационна система, за всеки ученик трябва да има информация в кой клас е и кой е класният му ръководител.

Код	Име ученик	Клас	Кл. ръководител
1	Асен Стоянов	10а	Руска Василева
2	Борис Велев	10а	Руска Василева
3	Васил Ганев	10а	Руска Василева
4	Ангел Балев	10б	Лиляна Минчева
5	Георги Славов	10б	Лиляна Минчева
6	Станчо Моллов	10б	Лиляна Минчева

В тая примерна таблица се наблюдават три аномалии:

Аномалия на излишеството: Ако включим двете полета в записа на ученик, то идентификаторът на класа и името на класният ръководител ще се среща толкова пъти в таблицата, колкото са учениците в класа .
Аномалия на обновяването: Ако класният ръководител си смени името, то трябва да бъде променено във всички записи, които го съдържат, а ако пропуснем промяната в някой запис, ще се получи противоречие.
Аномалия на включването: Ако не знаем името на учителя, не можем да включим записите на учениците, на които той е класен ръководител.
Аномалия на изключването: Ако се изтрият записите на всички ученици, на които учителят е класен ръководител, данните за класният ръководител се загубват.

4.3. Нормализация на БД. Нормални форми.

За да се премахнат аномалиите, се използва т. нар. нормализация на релационната схема.

Нормализацията е преобразуване на релационната схема, при което върху нея се налагат известни ограничения, водещи до отстраняване на някои нежелателни свойства (аномалии).

Първа нормална форма:
Изисква атомарност на данните в полетата на таблиците. Във всяка колона от таблицата и съответно във всяко поле от записите трябва да се съдържа само по една стойност.
Втора нормална форма:
Всеки елемент от всеки запис в коя да е таблица с БД трябва да се идентифицира с целия първичен ключ на таблицата
Трета нормална форма:
Третата нормална форма изисква не само всяка колона без ключ да зависи от целия първичен ключ, но и колоните без ключ да са независими една от друга.

Казано по друг начин, всяка колона без ключ трябва да зависи единствено от първичния ключ. Не трябва да съществуват функционални зависимости между стойностите на колоните. Например не трябва да се съхраняват отделно цена на стока без ДДС и цена с ДДС, след като едната се получава от другата с изчисление.

За нормализация на релациите в трите нормални форми се използват два основни метода: синтез и декомпозиция (разбиване на таблицата на няколко свързани таблици).

Да разгледаме последователно прилагането на нормализацията към горният пример.

Привеждане в първа нормална форма:

За привеждане в първа нормална форма е необходимо всяка колона от таблицата да съдържа само по една стойност. Така таблицата добива вида:

Код	Име _на_ученик	Фамилия _на_ученик	Клас	Име_на _ръководител	Фамилия_на _ръководител
1	Асен	Стоянов	10а	Руска	Василева
2	Борис	Велев	10а	Руска	Василева
3	Васил	Ганев	10а	Руска	Василева
4	Ангел	Балев	10б	Лиляна	Минчева
5	Георги	Славов	10б	Лиляна	Минчева
6	Станчо	Моллов	10б	Лиляна	Минчева

Привеждане във втора нормална форма:

За привеждане във втора нормална форма е необходимо всички записи трябва да се селектират еднозначно само с един първичен ключ. Горната таблица не отговаря на това условие. В нея за селектиране на ученик трябва да се използва полето „код”, а за селектиране на класен ръководител – полето „Клас”. Това показва, че е необходимо да се приложи метода на декомпозицията. Таблицата трябва да се раздели на две таблици – за ученици, и за класни ръководители, както следва:

Номер	Име	Фамилия	Клас
1	Асен	Стоянов	10а
2	Борис	Велев	10а
3	Васил	Ганев	10а
4	Ангел	Балев	10б
5	Георги	Славов	10б
6	Станчо	Моллов	10б

Таблица Ученици

Клас	Име	Фамилия
10а	Руска	Василева
10б	Лиляна	Минчева

Таблица Ръководители

Така се избягват:

Аномалия на излишеството: В таблицата „Ученици” не се записва името на класния ръководител, а само номера на класа. В таблицата „Ръководители” данните са всеки класен ръководител се записват само веднъж.

Аномалия на обновяването: Ако се смени класният ръководител, ще се промени само един запис в таблица „Ръководители”. В таблицата „Ученици” промени няма да се правят.

Аномалия на включването: Не е необходимо да се знае името на класния ръководител за да се включи запис за нов ученик.

Аномалия на изключването: Ако по някаква причина се изтрият записите на всички ученици, данните за класния ръководител се запазват.

4.4. Видове връзки между таблиците.

За правилното функциониране на една система в БД трябва да има изградени релации (връзки) между таблиците. Таблиците могат да се използват и без създадени в БД връзки (релации) между тях, но дефинирането на релации в базата от данни улеснява поддържането на логическата свързаност между данните. Например ако в една таблица се съхранява информация за служители, а в друга – за техните телефонни номера, ако е създадена връзка между тях при изтриване на служител от първата таблица сървърът на БД ще изтрие (при включена опция за това) всички негови телефонни номера от втората таблица.

Таблиците могат да бъдат свързани, когато полето, което е първичен ключ на една от таблиците, съществува също и в другата таблица. Между две таблици са възможни три типа релационни връзки, които се създават и използуват по различен начин.

Едно към едно (1:1) – всеки запис от една таблица може да се свърже само с един запис от друга таблица. Релацията се извършва чрез първичния ключ на едната таблица и първичен ключ (или уникален индекс) на втората таблица. Пример за такава връзка е връзката между таблица, съдържаща данни за служители и таблица, съдържаща допълнителна информация за тези служители, които са ръководители на отдели. Таблици, които са свързани с релация „Едно към едно” по принцип могат да се обединят в

една таблица, но това не винаги е добро решение. Например ако таблица „Служители” се обедини с таблица „Ръководители” в общата таблица, за тези служители, които не са ръководители в съответните полета няма да се съдържат данни, следователно обемът на обединената таблица ще бъде неоправдано голям.

Едно към много (1:n) – всеки запис от една таблица може да се свърже с много записи от друга таблица. Реферираната таблица се нарича главна (или родителска) таблица, а таблицата с външния ключ се нарича свързана (или дъщерна) таблица. Външният ключ на свързаната таблица се свързва с първичния ключ на главната таблица. Пример за такава връзка е връзката между таблицата Ръководители (като главна) и таблицата Ученици (като свързана. Връзката между тях е чрез колоната Клас (първичен ключ за таблица „Ръководители”) и съответната колона Клас от таблица „Ученици”.

Много към много (m:n) – най-сложният вид релация. Тази релация не се осъществява директно, а чрез допълнителна таблица, която е свързана с външни ключове с първата и с втората таблица. Пример за това е например връзката между таблици „Доставчици” и „Клиенти”. Един доставчик може да доставя стока за много клиенти, и един клиент може да има много доставчици. Допълнителната таблица съдържа двойки „идентификатор на доставчик”-„идентификатор на клиент”, които са свързани съответно с първичните ключове на таблиците „Доставчици” и „Клиенти”.

5. Система за управление на бази данни Microsoft Access

Microsoft Access е система за управление на бази данни (СУБД) на Microsoft, която комбинира релационния Microsoft Jet Database Engine с графичен потребителски интерфейс и инструменти за разработка на софтуер. Той е включен в на пакета от приложения на Microsoft Office, и се предлага и като самостоятелен продукт.

Microsoft Access съхранява данни в собствен формат, базиран на Access Jet Database Engine. Той може също да импортира или да свързва директно към данни, съхранявани в други приложения и бази данни.

Разработчиците на софтуер, архитектите на данни и опитните потребители могат да използват Microsoft Access за разработване на

0приложен софтуер. Подобно на други приложения на Microsoft Office, Access (както и другите продукти от Microsoft Office) включва поддържка на скриптовият обектно-базиран език за програмиране Visual Basic for Applications (VBA), който предоставя достъп до различни обекти, включително наследения DAO (обекти за достъп до данни), ActiveX обекти от данни и много други компоненти на ActiveX. Визуалните обекти, използвани във формуляри и отчети, излагат своите методи и свойства в програмната среда на VBA, а модулите на VBA кода могат да декларират и да извикват функции на операционната система Windows.

5.1. Microsoft Access Runtime

Microsoft предлага безплатни версии за изпълнение на Microsoft Access, които позволяват на потребителите да стартират настолно приложение на Access, без да е необходимо да купуват или инсталират версия на Microsoft Access. Това всъщност позволява на разработчиците на Access да създават бази данни, които могат свободно да се разпространяват до неограничен брой крайни потребители. Тези версии на Access 2007 и по-нови версии могат да бъдат изтеглени безплатно от Microsoft.

5.2. Свойства на полетата в MS Access

Всяко поле в таблиците на Access има свойства и тези свойства определят характеристиките и поведението на полето.

5.2.1. Типове данни в таблиците на MS Access

Най-важното свойство на поле е неговият тип данни. Типът данни на полето определя какъв вид данни може да съхраняват в него.

Важно е да се запомни, че за всяка колона от таблица на релационна БД (в това число MS Access) се дефинира тип на данните и във всички записи в таблицата, в полетата от дадената колона може да се съдържа само този тип данни.

MS Access поддържа различни видове данни, всеки със специфично предназначение.

Тип	Описание	Размер
Short Text	Текст или комбинации от текст и цифри, включително числа, които не изискват изчисляване (например телефонни номера)	до 255 символа
Long Text	Дълъг текст или комбинации от текст и цифри	до 63 999 символа
Number	Числови данни, използвани при математически изчисления	1, 2, 4, или 8 байта
Date/Time	Стойности за дата и час в интервал от 100 до 9999.	8 байта
Currency	Стойности на валута и цифрови данни, използвани при математически изчис-ления, включващи данни с един до четири знака след десетичната запетая	8 байта
AutoNumber	Уникален последователен (увеличаване с 1) номер или произволен номер, присвоен от Microsoft Access, когато в таблица се добавя нов запис	4 байта
Yes/No	Да и Не стойности и полета, които съдържат само една от двете стойности (Yes/No, Вярно/ Невярно или Включено/Изключено).	1 bit.

5.2.2. Свойства на полетата с които се налагат ограничения

Освен типа на данните, за полетата в таблиците на MS Access могат да се задават допълнителни свойства, с които дефинират допълнителни изисквания към данните, които могат да се въвеждат в полетата на таблицата. Ако при създаване на таблицата разработчикът не зададе стойност на някое свойство, Access задава стойност по подразбиране. По-долу е даден списък от свойства, на които най често се налага да се задават стойности, различни от подразбиращите се.

Field Size: задава максималния брой символи, които могат да бъдат въведени в текстово поле или подтип числови данни за поле от тип Number

Format: задава как данните да се извеждат в полето на таблицата.
Input Mask: задава правилата, които трябва да се спазват при въвеждане на данни.
Default Value: задава стойността по подразбиране, която ще се записва в полето на таблицата всеки път, когато се въведе нов запис.
Required: задава (със стойност Yes или No) дали е необходимо да се въведе стойност в това поле на таблицата.
Caption: задава надпис на колоната при извеждане на съдържанието на таблицата. Ако не бъде зададен, като надпис се извежда името на колоната.

5.2.3. Свойства за числови полета (тип Number)

За полетата от тип Number чрез допълнително свойството **Field Size** се задава типа на числовите данни, които могат да се въвеждат в даденото поле от таблица на Access:

Byte за цели числа без знак с размер един байт (0 ÷ 255)
Integer за цели числа с размер два байта (-32768 ÷ 32767)
Long Integer за цели числа с размер четири байта ($-2^{31} \div 2^{31}$)
Single за числа с размер четири байта с плаваща запетая
Double за числа с размер осем байта с плаваща запетая
Decimal десетични числа с фиксирана запетая с максимум 28 разряда
Currency специален вид за числа с десетична запетая, с до 4 цифри отдясно на десетичната запетая и до 15 отляво с общ с размер осем байта. Въведен е за финансови данни.

5.2.4. Свойства, с които се управлява извеждането на таблиците

Форматирането и вида на извеждане на таблиците в MS Access се управлява чрез няколко свойства:

Caption: задава надпис на колоната при извеждане на съдържанието на таблицата. Ако не бъде зададен, като надпис се извежда името на колоната.

5.3. Създаване на таблици в Access БД.

Таблиците са основните обекти в релационните БД. В тях се съдържат данните във вид на записи в съответствие със структурата на таблицата. В съответствие с релационният модел структурата на таблиците (както и на всички други обекти в БД) се съхранява в системни таблици, които по подразбиране не са достъпни за потребителя на БД.

Структурата на таблиците в релационните БД (в това число и на Access) включва:

дефиниции на колони (columns)

дефиниции на връзки (relations)

дефиниции на ограничения (constraints)

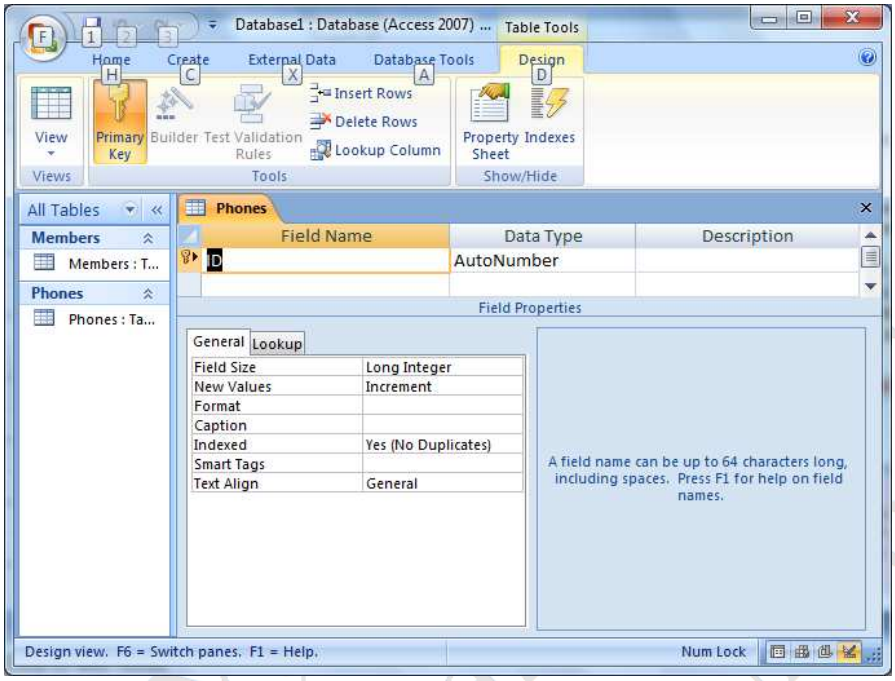
5.3.1. Създаване на таблица в БД с инструмента Design

За да се създаде таблица с инструмента Design е необходимо да се въведат за всяка колона от таблицата задължителните параметри: име (Field name) и тип на данните (Data Type). Освен задължителните параметри колоните имат и набор от незадължителни (с подразбиращи се стойности), например размер (обем) на данните (Field size), подразбираща се стойност (Default Value), Format (форматиране при извеждане в прозорец на Access) и др.

За създаване на нова таблица, от менюто се избира функция Create→New.

Access предоставя два изгледа (т.е. два вида графичен интерфейс) за създаване на таблици – табличен (DataSheet View) и дизайнерски (Design View). Изгледът Design View) предоставя повече удобства на разработчика и затова се предпочита. Превключването между двата изгледа става чрез бутона View от менюто DataSheet.

При избиране на изглед Design View се извежда прозорец за създаване или променяне на колони в таблицата.



Фиг. 1 Прозорец на изглед Design View за създаване на таблица

За всяка колона от таблицата се въвежда име, избира се тип и размер на данните и се задават стойности на незадължителните (с подразбиращи се стойности) параметри. След като се добавят всички колони и се зададат техните параметри, създадената (или променена) таблица се записва чрез функцията от менюто „Start button”→Save или (някои версии на Access) File→Save.

Важно е да се запомни, че при отварянето на таблица в изглед Design View или Datasheet View тя се заключва, с което се забраняват всички други операции за променяне на нейната структура. Поради това прозорецът на изгледа Design View (или Datasheet View) винаги трябва да се затваря след приключване на операциите с него, за да се премахне заключването на таблицата.

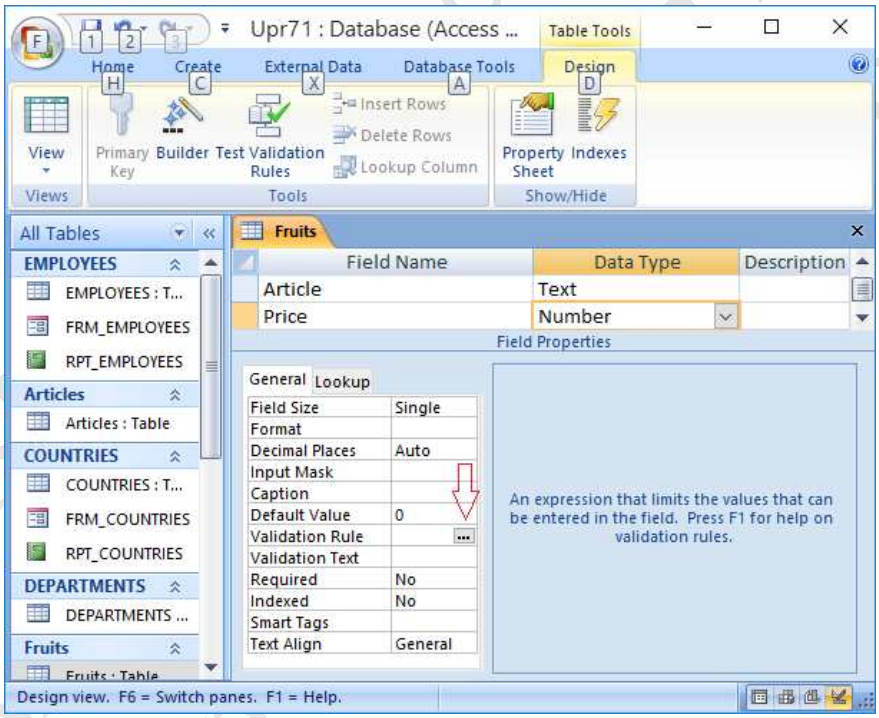
5.3.2. Проверка на валидността на данните при въвеждане

Access предоставя възможност за проверка на валидността на данните при въвеждането им в таблица от БД. За целта се използва свойството Validation Rule (правило за валидизация). Правилото за валидизация може

да се запише като формула (критерий) с използване на следните оператори за сравнение:

> за по-голямо	<= за по-малко или равно
>= за по-голямо или равно	= за равно
< за по-малко	<> за различно

Например критерият > 0 за колона “Price” ще означава, че в колоната могат да се въвеждат само стойности по-големи от 0.



Фиг. 2 Добавяне на правило за валидизация (Validation Rule) за колона от таблица

5.4. Изчисления в MS Access

Когато използваме Microsoft Access, често трябва да работим със стойности, които не са записани директно в базата от данни (БД). Например ако в БД се съхраняват данни за продажби, стойността на данъка върху тях може да бъде изчислена и не е необходимо да се съхранява в БД. Също така може да се изчисли общата сума, получена от продажбите, а също и сумите за отделните групи стоки. Тези стойности се изчисляват като използват изрази.

5.4.1. Изрази – формули за изчисляване на стойност

Изразите в Access (както и в езиците за програмиране) са формули за изчисляване на стойност, написани по определен синтаксис.

В Access изразите могат да включват вградени или дефинирани от потребителя функции, идентификатори, оператори, константи и кръгли скоби. По-долу тези елементи на изразите са разгледани по-подробно.

5.4.2. Оператори

Операторите в Access са символи за обозначаване на операции. Константите и променливите, функциите и идентификатори които участват при изпълнението на операциите се наричат операнди. Основните групи оператори са: аритметични, логически, за сравнение и символни.

Аритметични оператори

Access предоставя оператори за основните аритметични операции.

Оператор	Операция
+	Сумиране
-	Изваждане
*	Умножение
/	Делене
Mod	Модул (остатък от целочислено делене)
-	(при един операнд) Променяне на знака на стойността

Оператори за сравнение

Операторите за сравнение са най-често използваните оператори в SQL, тъй като те участват при изчисляването на условия, които се включват в повечето SQL команди. Операциите за сравнение могат да се изпълняват за различен тип операнди, но винаги връщат логически резултат TRUE или FALSE. Левият операнд на операторите за сравнение по правило е

единична стойност, получена в общия случай от изчисляването на израз. Десният операнд освен единична стойност може да бъде списък, интервал или шаблон. Дадената по-долу таблица съдържа операторите за сравнение на Access.

=	Равно
<>	Различно
>	По голямо
>=	По голямо или равно
<	По малко
<=	По малко или равно

Например ако за дадена колона от таблица от тип Integer (т.е за цели числа) зададем подразбираща се стойност Default = 6>5 при добавяне на запис Access ще получи от операцията сравнение стойност True, от която след преобразуване в цяло число ще се получи подразбираща се стойност -1.

Логически оператори

Access притежава оператори за основните логически операции: логическо умножение AND, логическо събиране OR и логическо отрицание NOT. Резултатите от

X	Y	Z
false	false	false
false	true	false
true	false	false
true	true	true

3a) логическо умножение (И)

X	Y	Z
false	false	false
false	true	true
true	false	true
true	true	true

3b) логическо събиране (ИЛИ)

X	Z
false	true
true	false

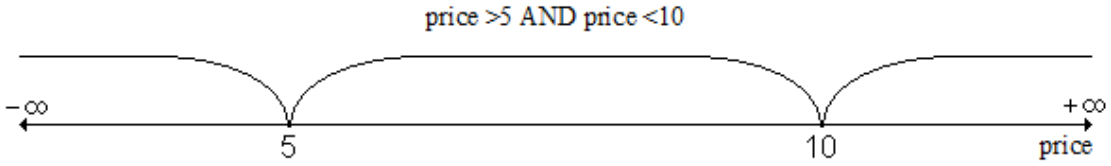
3c) логическо отрицание (НЕ)

Логически операции – таблици на истинност.

Те участват в изрази, с които се изчисляват по сложни условия, както е демонстрирано в следващите примери:

```
price >5 AND price <10
```

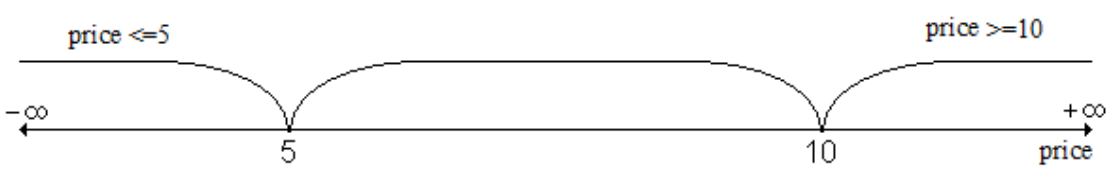
Този израз ще върне логическа стойност TRUE ако стойността на полето price е едновременно по-голяма от 5 и по-малка от 10. Можем да си представим стойностите на price, за които от горният израз ще се получи логическа истина като интервал върху числовата ос, както е показано на следващата фигура:



Да разгледаме израз, който използва логическият оператор OR:

```
price <=5 OR price >=10
```

Този израз ще върне логическа стойност TRUE ако стойността на полето price е или по-малка или равна на 5 или по-голяма или равна на 10. От този израз ще се получи логическа истина True ако стойността на price е в един от два непресичащи се интервала върху числовата ос, както е показано на следващата фигура.



Очевидно вторият израз ще върне стойност, точно обратна стойността, получена от първия израз. Това означава, че втория израз може да бъде заместен от инвертирания с оператор NOT резултат от първия израз.
NOT (price >5 AND price <10)

5.5. Идентификатори

В изразите чрез идентификаторите се получава достъп до полета или колони от таблици в БД. Може да се каже, че в изразите на Access

идентификаторите съответстват на променливите в езиците за програмиране. Синтаксисът (правилото за записване) на идентификатор с който се получава достъп до поле от запис на таблица е:

[име_на_таблица]![име_на_поле]

Например от изразът

[Users]![Fname]+" "+[Users]![Lname]

Пример 1 Израз, в който се използват идентификатори за полета от таблица

се получава текст (символен низ) които съдържа име и фамилия на служител от таблица Users. В израза се слепват текста от полето Fname, празен интервал " " и текста от полето Lname.

Тогава, когато името на таблица или колона съдържа само букви, цифри и символи "_" (долна черта) квадратните скоби могат и да не се поставят, т.е. изразът може да се запише и като Users!Fname+" "+Users!Lname.

в общият случай е следното:

[име_на_колекция]![име_на_обект].[име_на_свойство]

където:

колекция	е структура, която съдържа обекти. Например таблицата е колекция от колони, а записът (редът) в таблицата е колекция от полета
обект	е структура, която притежава свойства.

В примера колекцията е записът от таблицата Users, а обектите –полетата Fname и Lname.

5.6. Вградени функции

Функциите в Access са процедури които изпълняват зададен код и връщат резултат. Access предоставя голям набор от вградени функции за различни видове операции. Най-общо тези функции могат да се разделят на две групи:

- Скаларни (едноредови) функции – това са функции, които извършват операции със стойностите от всеки един запис и връщат отделен резултат за всеки запис. Например функцията UCase преобразува символите-букви на зададен символен низ в главни букви и връща получения низ.
- Агрегатни (обобщаващи) функции - това са функции, които извършват операции със стойностите на едно или няколко полета от всички извлечени записи и връщат един резултат за всички обработвани записи.

5.6.1. Едноредови (скаларни) функции

Access предоставя скаларни функции за операции с различни типове данни – за операции с дата и време, за аритметични и символни операции.

Текстови функции

Име	Описание
Asc(string)	Връща ASCII кода на първият символ от string
Chr(code)	Връща символа за зададения ASCII код
Оператор &	Слепва два символни низа и връща общ низ. При слепване на низ с Null връща низа
Оператор +	Слепва два символни низа и връща общ низ. ! При слепване на низ с Null връща Null
Format(data, format)	Форматира символен низ по зададен формат. Пример: Format(Date, "yyyy/mm/dd")
InStr([start,] string, search)	Връща позицията на първото появяване на search в зададена последователност от символи в символен низ string.
InstrRev([start,] string, search)	Връща позицията на първото появяване на зададена последователност от символи, но при търсене от края към началото
LCase(string)	Връща зададен низ като замества главните букви с малки.
Left(string, lenght)	Извлича зададен брой символи от низ (започвайки отляво)
Len(string)	Връща дължината на символен низ в брой символи
Mid(string, start,	Извлича зададен брой символи от низ

[length])	(започвайки от зададена позиция)
Replace (string, find, replacement, [start, [count]])	Заменя подниз *find* в низ *string*, с друг подниз *replacement*, определен брой пъти *count* от начална позиция *start*.
Right(string, lenght)	Извлича зададен брой символи от низ (започвайки отдясно)
Space	Връща низ от зададен брой празни интервали
Split(string [,delimiter])	Разделя символен низ string в масив от поднизове по зададен разделител delimiter
Str(number)	Връща зададено число като символен низ
StrComp(string1, string2, compare)	Сравнява символни низове. compare задава начина на сравнение на символите: 1-двоично, 2-текстово
StrConv(string, conversion)	Конвертира символен низ string по зададен тип на конверсия conversion: 0 =в главни букви, 1=в малки букви, 2=първа главна останалите малки
StrReverse(string)	Връща символите на зададен низ в обратен ред
Trim(string), Ltrim, RTrim	Премахват празните интервали от символен низ. Ltrim – в началото, Rtrim – в края, Trim – и в началото и в края
UCase(string)	Връща зададен низ като замества малките букви с главни.

Функции за дата и време

Име	Описание
Date()	Връща текущата системна дата
DateAdd(interval, number, date)	Добавя интервал от време/дата към зададена дата и след това връща датата. Интервалът се задава като: yyyy = Year, q = Quarter, m = month, y = Day of the year, d = Day, w = Weekday, ww = Week, h = hour, n = Minute, s = Second. Пример: DateAdd("m", 1, "31-Jan-95") прибавя месец към датата.
DateDiff(interval, date1, date2)	Връща разликата между две дати за зададен интервал. Интервалът се задава както за функция DateAdd
DatePart	Връща определена част от датата (като цяло число)

DateSerial	Връща дата от зададена част от нея (стойност за година, месец и ден)
DateValue(string)	Преобразува символен низ в дата
Day(date)	Връща деня от месеца за дадена дата
Format	Форматира стойност на датата с посочения формат
Hour(date)	Връща часовата част от време/или дата и време
Minute(date)	Връща минутите от време/или дата и време
Month(date)	Връща месецът (като число) от дадена дата
MonthName	Връща името на месеца от зададено число
Now()	Връща текущата дата и час въз основа на системната дата и час на компютъра
Second(date)	Връща секундите от време/или дата и време
Time()	Връща текущото системно време
TimeSerial	Връща време от посочените части (час, минута и втора стойност)
TimeValue(string)	Преобразува символен низ във време
Weekday(date)	Връща номера на деня от седмицата за дадена дата
WeekdayName(date)	Връща името на деня от седмицата за даден номер на ден
Year(date)	Връща годината (като число) от дадена дата

Числови функции

Abs(x)	Връща абсолютната стойност на число
Atn(x)	Връща стойността на функция аркус тангенс
Sin(x)	Връща стойността на функция синус по зададен ъгъл
Cos(x)	Връща стойността на функция косинус по зададен ъгъл
Exp(x)	Връща стойността на функция exp(x) по зададено число x
Format	Форматира числова стойност по зададен формат
Int(x)	Връща цялата част на число
Randomize	Инициализира генератора на случайни числа (използван от Rnd)
Rnd()	Връща псевдослучайно число
Round(x,n)	Закръглява число x до определен брой десетични знаци n
Sgn(x)	Връща знакът на число
Sqr(x)	Връща квадратния корен от зададено число

Val(string)	Преобразува символен низ в число

5.6.2. Агрегатни (обобщаващи) функции

Агрегатните функции изчисляват обобщаваща информация за дадена извадка от записи. Access предоставя следните агрегатни функции:

COUNT(expr)	изчислява броя на извлечените записи
SUM(expr)	намира сумата от стойности на зададена колона в записите
MAX(expr)	намира най-голямата стойност на зададена колона
MIN(expr)	намира най-малката стойност на зададена колона
AVG(expr)	намира средната стойност на зададена колона
Var(expr)	Пресмята дисперсията (отклонение от средната стойност) на стойностите в полето.

където expr представлява символен низ, идентифициращ колоната, която съдържа числовите данни, или в общият случай израз (т.е. формула) за изчисляване на стойност за всеки запис от извадката. Пример: AGV(price) ще иачисли средната стойност за цялата колона price или за всяка от групите при изпълнение на заявка с групиране.

5.6.3. Използване на функции в таблици на MS Access

Едноредовите (скаларни) функции могат да се използват за задаване на подразбираща се стойност на полетата в зададена колона от таблица. Например ако за подразбираща се стойност (Default) на колона от таблица зададем =Date() то при добавяне на запис към таблицата в тази колона ще се записва като подразбираща се стойност стойността на текущата дата.

5.7. Заявки в MS Access

Заявките са модулите в Microsoft Access, чрез които се извличат данни, съхранени в таблиците. Заявките не съдържат данни в себе си, а само код за извличане на данните и описание на начина, по който те да бъдат показани на потребителя.

Какво може да се направи чрез заявки:

1. Чрез заявките се прави извличане (с възможност за задаване на критерий) и подреждане на данните.
2. Чрез заявките могат да се свържат две или повече таблици и да представят извлечените данни под формата на таблица.
3. Заявките могат да ползват като източници на данни не само таблици, но и други заявки, създадени преди това на базата на съществуващи таблици или заявки.
4. Заявките са основния вид модули, в които могат да се генерират вторични данни. Чрез заявките могат да се получат и извеждат не само първични данни - такива, които са съхранени в таблиците, но и производни (вторични) данни - такива които са генерирани чрез ползване на формули (изрази) или функции (вградени или потребителски).
5. В MS Access могат да се създават два вида заявки - заявки за избор на данни - такива, които само показват данните по определен начин и изпълними заявки - такива които променят, допълват или изтриват вече съществуващи данни или създават нови таблици.

Всички заявки в релационните бази от данни се реализират чрез език за структурирани заявки (SQL). В Microsoft Access съществува помощна програма, която чрез своя интерфейс дава възможност за създаване на заявки без да е необходимо да се пише код на SQL, но в БД заявката се съхранява като SQL код. Кодът на заявките на Access е почти напълно съвместим с SQL кода на почти всички широко разпространени релационни системи са управление на бази данни (РСУБД) и заявка създадена в една система (с някои синтактични корекции), по принцип може да работи и на други сървъри за БД, ако структурата на данните е същата. Това, което създава най-големи проблеми при преобразуването на заявки за изпълнението им на други сървъри за БД е разликата във вградените функции – и като имена и като входни параметри.

5.8. Заявки за селектиране на данни

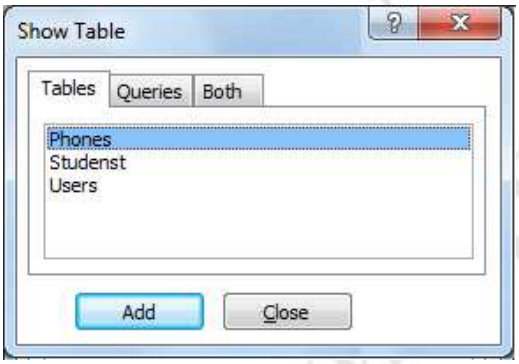
Чрез заявките за селектиране на данни се извличат данни от една или повече таблици от БД. Към тази група принадлежат селективните, параметричните, кръстосаните (crosstab) и обобщаващите (с групиране) заявки.

5.8.1. Създаване на селективни заявки в изглед Query Design

Изгледът Design е удобен инструмент за създаване на заявки. Той предоставя графичен интерфейс за добавяне и свързване на таблици, подобен на инструмента Relationships (с който се създават връзки между таблиците) и работна таблица, в която се добавят колоните, които участват в заявката и критериите, които използват техните стойности.

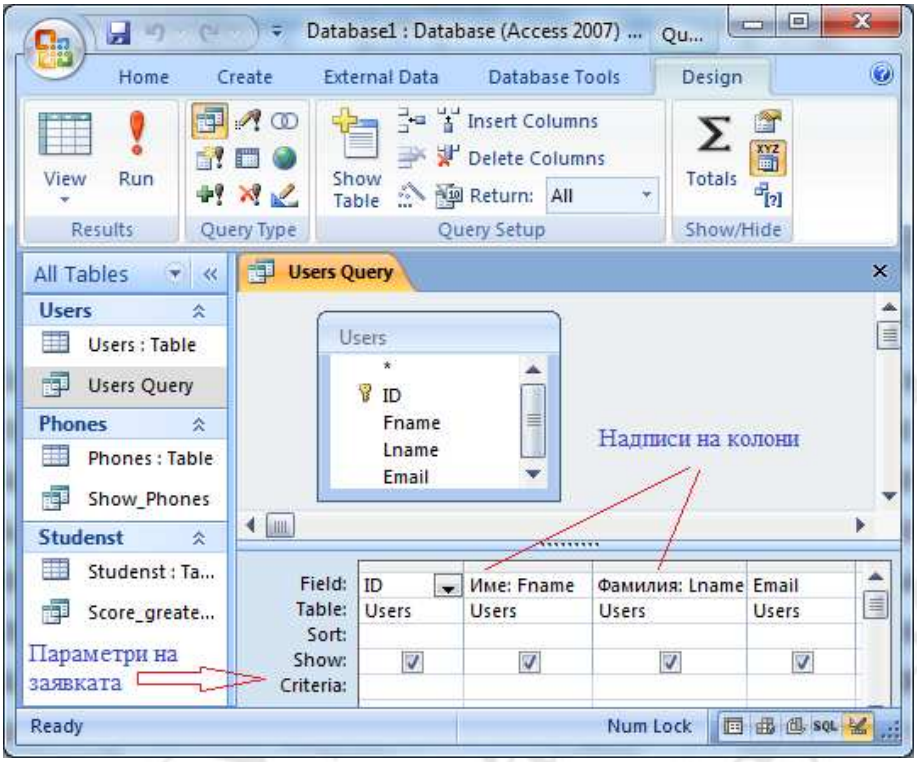
За създаване на заявка в изглед Design, от менюто Create се избира Query Design. Извежда се работна област и прозорец за добавяне на таблици и създадени преди това заявки.

.



Фиг. 3 Прозорец за добавяне на таблици и заявки

Добавените таблици и заявки се извеждат в работната област. Важно е да се отбележи, че ако има създадени връзки между таблиците (например с инструмента Relationships) те се изобразяват в прозореца на Query Design. При създаване на заявка, чрез която ще се извличат данни от две или повече таблици, могат да се създават връзки между таблиците и в прозореца на Query Design по същият начин, както и с инструмента Relationships - чрез провлачване с натиснат ляв бутон на мишката на колона от първата към колона от втората таблица.



Фиг. 4 Работна област за създаване на зявки в изглед Design

Под областта, в която се изобразяват в графичен вид таблиците се извеждат в табличен вид параметри на заявката както следва:

- **Field** - в този ред се въвеждат колоните от таблицата или колони, съдържащи производни данни. Колоните от таблицата могат да се попълнят чрез теглене, чрез избор от отварящ се списък или чрез двойно кликване върху полето в списъка на таблицата. Ако в реда се съдържа дадено поле само веднъж, надписът на колоната е същият като името на полето. Ако дадена колона се добави втори път, е необходимо да и се даде друг надпис, в противен случай Access генерира сам надпис, който започва с “Expr” Надписът на колоната може да се смени, като се направи етикет на полето. Например ако искаме колоната с Fname да има надпис „Име” пишем: Fname:Име. В една заявка всяка колона трябва да има уникално име.
- **Table** - в този ред се изписва името на таблицата, чиито полета се ползват. Ако една и съща таблица е включена в заявката повече от веднъж, и се използват колони от различни включвания, към името на таблицата се добавя номерация „_X, например Users, Users_1, Users_2 и т. н. В полетата, които съдържат производни данни полето Table е празно.
- **Sort** - указва каква е посоката на сортиране на данните: Ascending (нарастващ ред), Descending (намаляващ ред) или несортирани данни.

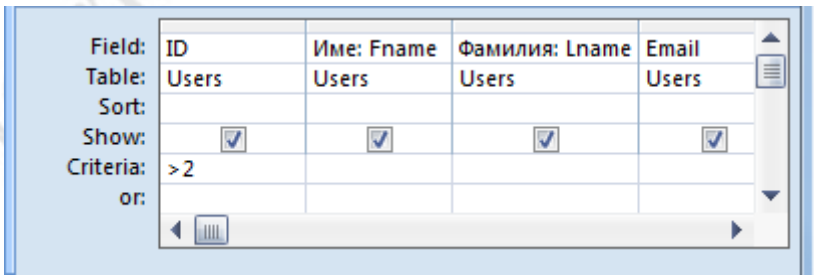
- **Show** - съдържа елемент за маркиране (check box), с който се указва дали колоната да се вижда при отваряне на заявката. Причината в заявката да се включи колона, която няма да се вижда при отваряне е, че за данните от тази колона може да се зададе критерий.
- **Criteria** - съдържа условието, на което трябва да отговарят данните от съответната колона.
- **or** - съдържа допълнително условие, на което трябва да отговарят данните в съответната колона. Двете условия – **Criteria** и **or** се обединяват с логическа операция ИЛИ (OR), което означава, че е достатъчно да се изпълнява едно от двете условия, за да се изведат данни от заявката.

5.8.2. Добавяне на критерии на заявки в изглед Query Design

В изглед Query Design критериите на заявките се записват в редовете „Criteria" и „or" за колоните, включени в заявката. Критерият за колона се състои от оператор и стойност за сравнение. Операторите, които могат да се използват са:

>	за по-голямо
>=	за по-голямо или равно
<	за по-малко
<=	за по-малко или равно
=	за равно
<>	за различно

Например критерият > 2 за колона "ID" ще означава, че след филтриране ще останат само тези редове от таблицата, които в колоната "ID" имат стойност по-голяма от 2.

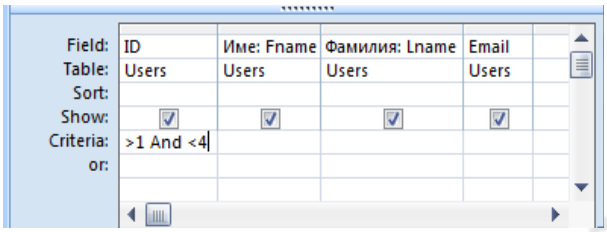


Фиг. 5 Добавяне на критерий за колона в изглед Query Design

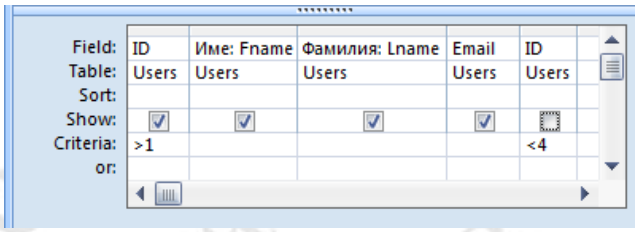
За получаване на по-сложни критерии за колона могат да се използват логически оператори AND, OR и NOT:

AND (И) – обединява две условия като общото условие се изпълнява ако и двете условия се изпълняват. Например критерият „> 1 AND < 4” ще се изпълнява за числени стойности, които са по-големи от 1 и по-малки от 4.

Важно е да се запомни, че всички критерии за колони, които са зададени на реда Criteria се обединяват с операция AND т.е. всички условия трябва да са изпълнени за да се извлекат данни от заявката.



Критерий за колона ID с използване на оператор AND



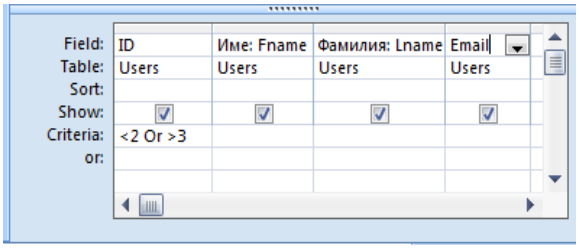
Критерий с използване на две колони за колона ID

Фиг. 6 Област върху числовата ос, за която се изпълнява критерият „> 1 AND <4”

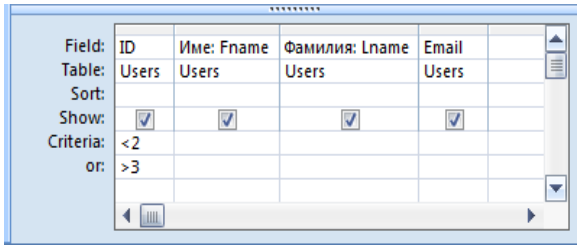
Дадените по-горе примери за критерии за колона ID са еквивалентни, но в левият пример критериите „>1” и „<4” са обединени с оператор AND, а в десният пример критериите „>1” и „<4” са зададени за две отделни колони ID, но за да не се изведе втората колона при отваряне на заявката, за нея е премахнато маркирането в реда Show.

Интересно е да се отбележи, че когато дясната заявка се съхрани и се затвори прозореца на Query Design, при следващо отваряне в изглед Query Design заявката ще изглежда така, както е показана вляво. Това е така, защото заявките се съхраняват като SQL код, а не във вида, в който се извеждат в изгледа Query Design.

OR (ИЛИ) – обединява две условия като общото условие се изпълнява ако поне едно от двете условия се изпълнява. Например критерият „< 2 OR >3” ще се изпълнява за числени стойности, които са по-малки от 2 или по-големи от 3. Той също може да се зададе по два начина – с използване на оператор OR или с използване на реда “or” както е показано в следващите примери



Критерий за колона ID с използване на оператор OR



Критерий за колона ID с използване на редове “Criteria” и “or”

Фиг. 7 Области върху числовата ос, за които се изпълнява критерият „< 2 OR >3”

Дадените по-горе примери за критерии за колона ID са еквивалентни, но в левият пример критериите „<2” и „>3” са обединени с оператор OR, а в десният пример критериите „<2” и „>3” са зададени поотделно в редовете “Criteria” и “or”.

И в тоя случай когато дясната заявка се съхрани и се затвори прозореца на Query Design, при следващо отваряне в изглед Query Design заявката ще изглежда така, както е показана вляво. Това (както и за примера с оператор AND) е така, защото заявките се съхраняват като SQL код, а не във вида, в който се извеждат в изгледа Query Design.

5.8.3. Създаване на заявки с изчисляеми колони в изглед Query Design

Съгласно релационният модел между данните в различни колони не трябва да съществуват функционални зависимости. Например ако в една колона са записани стойности на числова променлива x, то в друга колона не трябва да се съдържат стойности, които са функция на x. В заявките обаче понякога се налага да се извеждат колони, в които стойностите се изчисляват. За да се получи такава колона в заявката, в реда Field се записва името на колоната, и след разделител двоеточие „:” се записва израз за изчисляване на стойността. Например ако в заявка, която използва данните от таблица Users с колони Fname и Lname добавим колона с име

Full_Name: [Fname]+" "+[Lname]

в колоната ще се извеждат името и фамилията на потребителите, записани

в таблицата като символен низ, получен чрез слепване на Fname, Lname и празен интервал.

По подобен начин могат да се получат и други изчисляеми колони.

Синтаксисът (т.е. правилото за записване) на името на изчисляемата колона и изразът, чрез който се изчислява стойността във всяко поле от колоната се вижда от примера:

име_на_ изчисляемата_колона:*израз_за_изчисляване*

5.8.4. Създаване на параметрични заявки

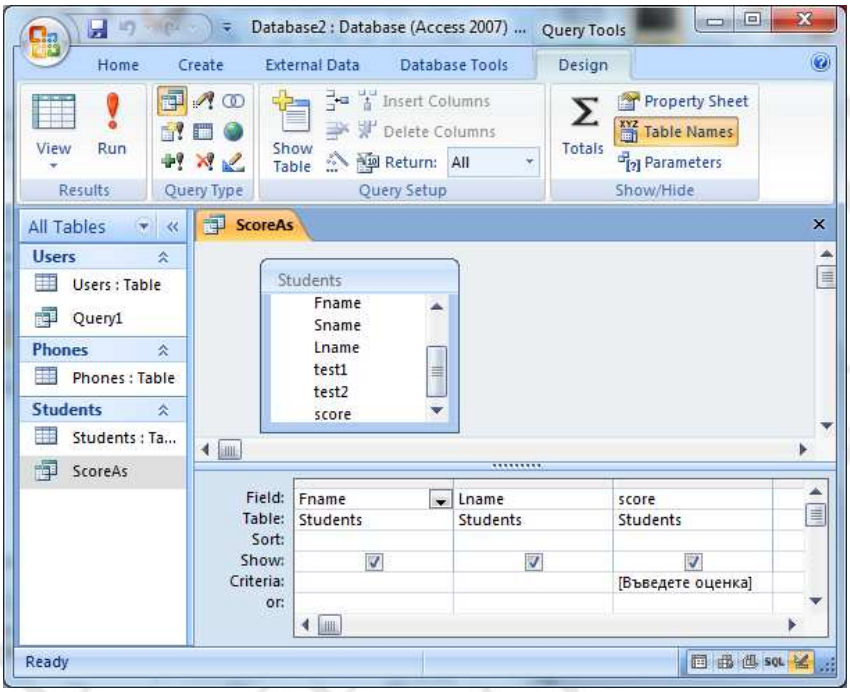
Параметричните заявки са заявки, с които на потребителят се дава възможност чрез диалогов прозорец да въведе стойности на параметри. Параметърът е част от информация, която предоставяте на заявка веднага, когато я стартирате. Параметрите могат да се използват самостоятелно или като част от по-голям израз за формиране на критерий в заявката.

По правило стойностите на параметри се използват в критерии за извличане на данни. Може да се проектира заявка, която да подканва потребителя да въведе стойност на един параметър, като например цена на стока, или повече – например две дати. За всеки параметър параметричната заявка показва отделен диалогов прозорец, с който подканва потребителя да въведе стойност за параметъра.

5.8.4.1. Добавяне на параметър към селективна заявка

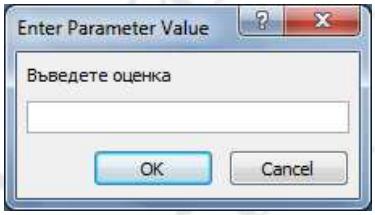
За добавяне на параметър заявката се отваря в изглед за проектиране и в реда Criteria, в който се задава критерий за избраната колона, се записва вместо стойност текст в квадратни скоби. Текстът трябва да съдържа пояснение за това какъв параметър (като предназначение и стойност) се

очаква да въведе потребителят на заявката. Например ако искаме да създадем заявка ScoreAs, която да извлича от таблица Students имената и факултетните номера на студенти, които имат оценка 4, то в реда Criteria за колоната Score въвеждаме [Въведете оценка].



Фиг. 8 Задаване на параметър на селективна заявка

Когато се отвори параметризираната заявка, подканата се показва в диа логов прозорец без квадратните скоби.

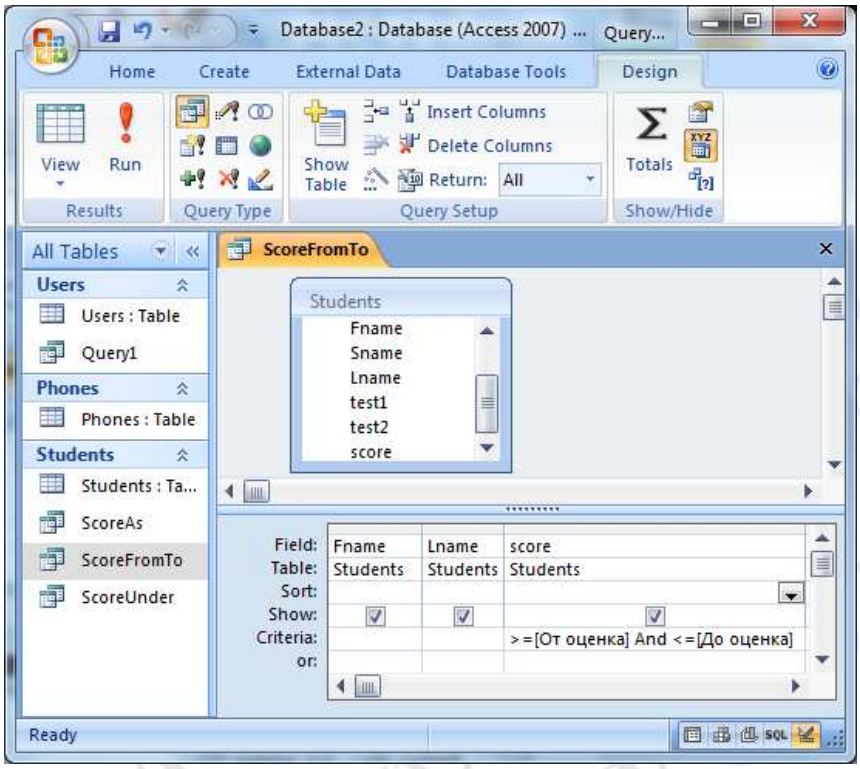


Фиг. 9 Въвеждане на параметър при изпълнение на параметрична заявка

Важно е да се запомни, че параметрите могат да се използват не само като единични стойности в критерий за съвпадение, както е в предният пример, а в изрази, с които се задават по-сложни критерии , включващи оператори за сравнение. Например можем да създадем заявка с два параметъра, която да извлича от таблица Students имената и факултетните номера на студенти, които имат оценка в зададен интервал. За целта въвеждаме в реда Criteria за колоната Score следният критерий:

>= [От оценка] And <= [До оценка]

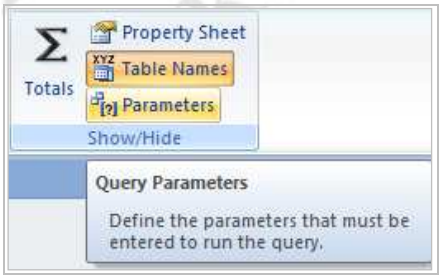
както е показано на следващата фигура



Фиг. 10 Параметрична заявка с два параметъра

5.8.4.2. Проверка на типа на въведената стойност за параметър

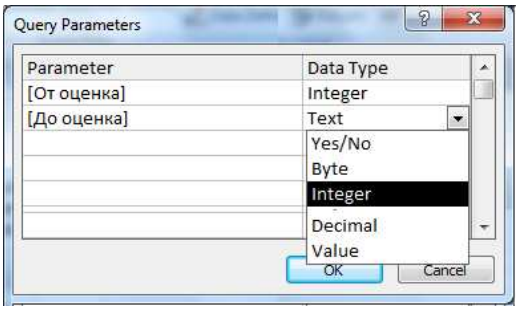
По подразбиране при въвеждане на стойност на параметър не се прави проверка за типа на въведените данни. В резултат на това ако потребителят при изпълнение на параметрична заявка с много параметри въведе например вместо число текст като стойност на първият параметър, съобщение за грешка ще се изведе едва при изпълнението на заявката и всички параметри трябва да се въвеждат отново.



Фиг. 11 Функция за задаване типа на параметри

Access дава възможност за проверка на типа на въведената стойност на параметър, която се изпълнява веднага, след като параметърът бъде въведен. За целта се използва функцията „Parameters” от групата “Show/ Hide” на менюто Desing.

При щракване с мишката върху функцията „Parameters” се извежда прозорец, в който се въвеждат параметрите (така, както са зададени в реда Criteria) и се избира от списъчен обект вляво типът на данните за всеки от тях.



Фиг. 12 Задаване на тип на параметрите

След като е зададен тип на данните за параметрите ако потребителят въведе данни от невалиден тип се извежда съобщение за грешка. Например ако за параметричната заявка ScoreFromTo от предният пример се въведе като стойност на първият параметър текст вместо число, ще се изведе съобщение за грешка:

След затварянето на прозореца се очаква от потребителя да въведе валидна стойност на параметъра.

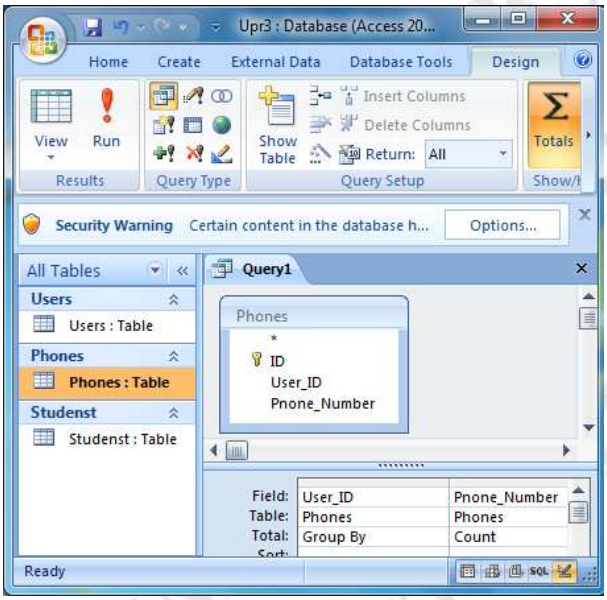
5.8.5. Създаване на заявки с групиране

Групирането се използва за организиране на получените от заявката записи (редове) в групи по определен критерий и обобщаване на данни за тези групи. За да се извърши групиране трябва да се избере колона (или, в общият случай израз, в който участват стойности от полетата на колони) по чиято стойност записите се разделят на групи.Обикновено групирането се използва в комбинация с една или няколко от агрегатните функции COUNT, SUM, MAX, MIN и AVG. Целта е агрегатната функция да се приложи върху групите от записи, а не върху всички извлечени от заявката записи. Access предоставя две възможности за създаване на заявки с групиране: чрез използване на изглед Query Design или чрез използване на помощната програма Query Wizard.

Създаване на заявки с групиране в изглед Query Design

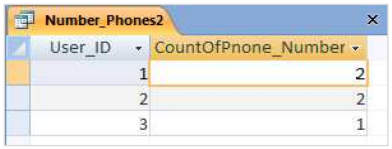
Да разгледаме един пример: Искаме да получим заявка, която да извежда от таблица Phones броят телефони, който притежават клиентите и фамилиите на притежателите. За целта създаваме заявка в изглед Query Design, в която включваме таблиците Users (която съдържа данни за клиентите) и Phones, която съдържа данни за техните телефони. Колоната, по която селектират потребителите е ID за таблица Users и свързаната с нея колона User_ID в таблица Phones. Групирането по тази колона ще раздели извлечените от заявката записи на групи по потребители.

За целта включваме изтегляме върху реда Fields от таблицата с параметри (ТП) в долната половина на прозореца колоните User_ID и Phone_Number от таблица Phones.



Щракваме с мишката върху бутона „Totals” (∑) в менюто Design и в таблицата с параметри се появява ред „Total” с текст „Group By” във всяка колона.

Важно е да се запомни, че текст „Group By” трябва да остане само за тези колони, по които се извършва групиране, т.е за тези колони, по чиито стойности записите се разделят на групи. В примера разделяме на групи записите, получени от таблица Phones по номер на притежател, т.е. по съдържанието на колоната User_ID. Така групата за всеки номер на притежател ще съдържа толкова телефонни номера, колкото телефона той притежава. В колоните от заявката, по които не се извършва групиране, трябва да се получи по една числова стойност за всяка от групите. Това се прави с някоя от функциите за групиране (агрегатни функции) Count, Sum, Max, Min или Avg. В примера в колоната „Phone_Number” искаме да получим броя на телефоните за всеки притежател, затова щракваме на бутона “∨” на реда “Total” и от падащото меню избираме Count.



Ако отворим заявката с “Open” (след съхраняване на заявката и затваряне на прозореца на Query Design) ще се изведе следното съдържание.

Access поставя име по подразбиране на изчисляемите колони от заявките. В повечето случаи те са твърде дълги и неудобни за използване и се налага да бъдат променени. Както беше дадено и по-горе, заглавие (caption) на колона се добавя по следния синтаксис:

заглавие**:**име_на_колона

В примера за колоната Phone_number можем да добавим заглавие „Брой телефони” като добавим текста на заглавието и символ двоеточие „:”

```
Брой Телефони:Pnone_Number
```

Фиг. 13 Добавяне на заглавие към колона с агрегатна функция в заявка с групиране

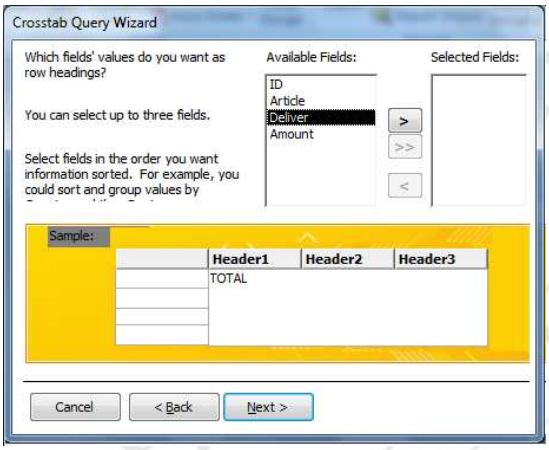
Access извежда текста на заглавието преди символа двоеточие във вида, в който е записано. В SQL на Access заглавията се поставят в квадратни скоби, но в Design View квадратните скоби се използват за параметрични заявки и не трябва да се поставят в заглавия.

Важно е да се запомни, че в заявките с групиране „Group By” трябва да се зададе за всички колони, за които не се задава обобщаваща (агрегатна) функция. В някои случаи заявката трябва да изведе данни в колони, по които не можем да правим групиране (т.е. не можем да им зададем „Group By”) но не можем и да им зададем агрегатна функция. В такива случаи се използва свързване между заявка с групиране и заявка без групиране, чрез която се добавят допълнителните данни.

5.8.6. Създаване на кръстосани заявки (Crosstab query) чрез инструмента Query Wizard

Кръстосаните заявки подпомагат анализа на влиянието на един тип информация върху друг тип. При изпълнение на такава заявка се създава двумерна таблица, в която стойностите от едната колона се извеждат в първата колона на Crosstab таблицата като заглавия за редовете, а стойностите от втората колона – в първия ред на Crosstab таблицата като заглавия на колони. Търсената информация се намира на в пресечната клетка на реда и колоната като обобщаваща информация – брой, средна

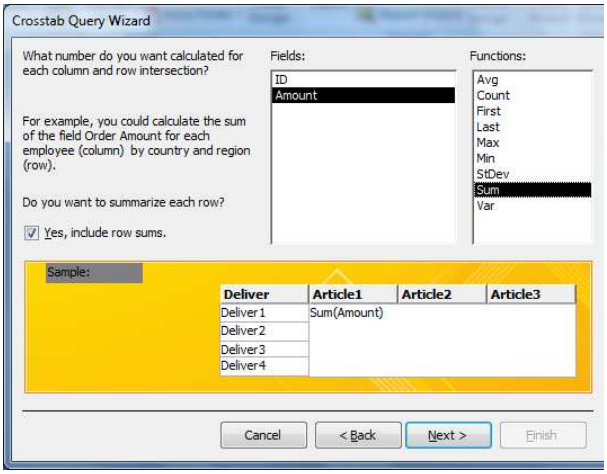
стойност, минимална, максимална и т.н. Очевидно е, че тази информация се получава също чрез групиране, но информацията се предствя в друг вид. По-точно, в първата колона са имената на групите (например вид стока), в първият ред – видът данните, за които се търси информация (например наличности), а в пресечната точка – обобщаващата информация (например сума от наличности за видове стока). Като пример да разгледаме таблица Articles, която съдържа колони Article (за имена на стоки) и Amount за доставени количества. За създаване на заявката щракваме на бутона Query Wizard от менюто Design и от прозореца избираме Crosstab Query Wizard. Отваря се прозорец, от който избираме като източник на данни таблицата Articles.



На следващата стъпка избираме колоната от таблицата, стойностите от която ще се изведат в първата колона на Crosstab таблицата (по тези стойности ще се извършва обобщаване). В примера това е колона Deliver.

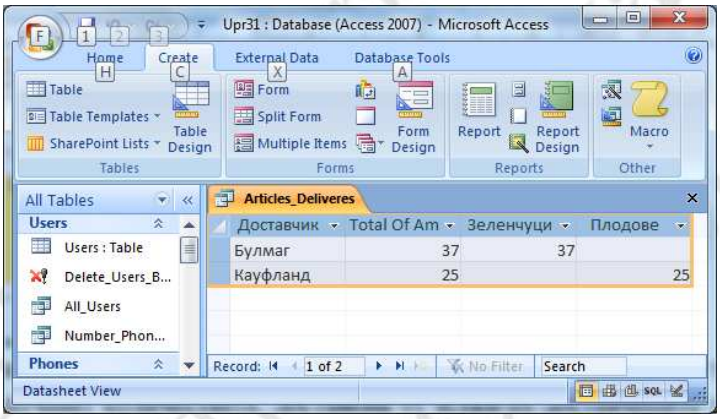
На следващата стъпка се избира колоната, чийто стойности ще се разполагат в първия ред на Crosstab таблицата. В примера това е колоната Article.

На следващата стъпка се избират колоната и функцията, по която се прави обобщаване. По правило това е колона, която съдържа числови стойности. В примера това е колоната Amount (количество), а функцията за обобщаване е Sum.



Така в пресечните клетки за редове и колони от Crosstab таблицата се получават количествата, доставени от всеки от доставчиците за всяка една от стоките.

В примера се получава следния резултат:



Crosstab таблицата съдържа количествата, доставени за всяка стока от всеки един от доставчиците.

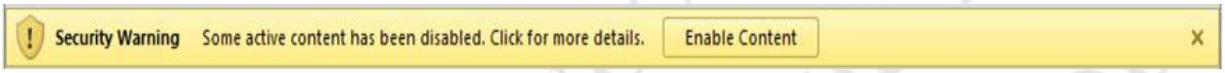
5.9. Заявки за създаване, променяне и актуализиране

Към тази група принадлежат заявките за създаване на таблица и за добавяне, актуализиране и изтриване на записи в съществуваща таблица. Заявките за създаване и променяне на обекти в БД (в това число и таблици) в английско езичната литература се обозначават със съкращението DDL (Data Definition Language), докато тези, с които се манипулират или селектират данните се обозначават със съкращението DML (Data Manipulation Language). DCL – за управление на БД.

5.9.1. Настройки за разрешаване на заявки за създаване, променяне и актуализиране

Заявките за селектиране на данни в Access не изискват допълнителни настройки за сигурност. За изпълнението на заявки за създаване,

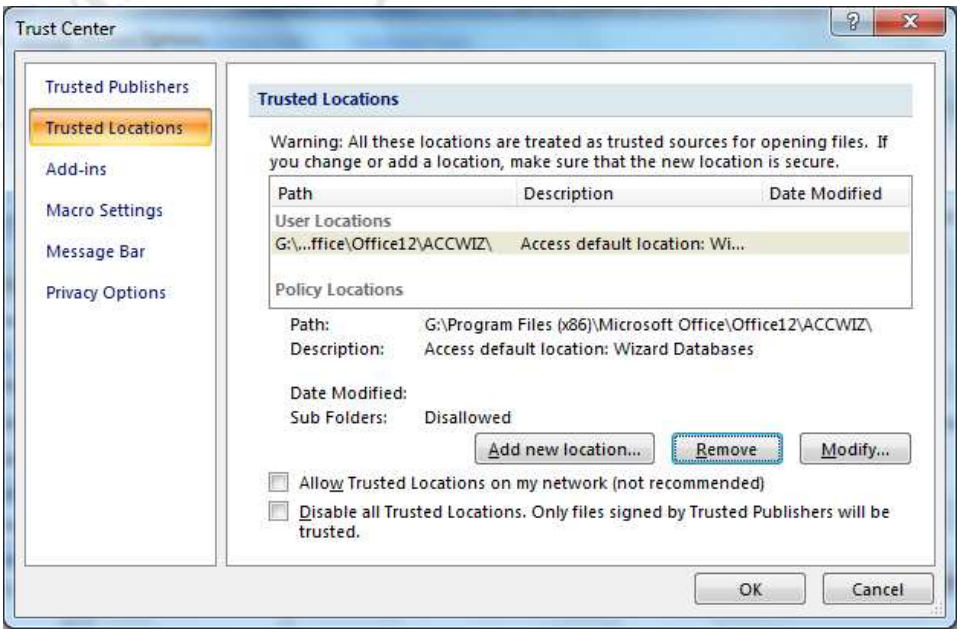
променяне и актуализиране на обекти в базата от данни (БД) е необходимо допълнително разрешение от притежателят на базата от данни, а с което той да потвърди, че счита БД за надежден източник на данни. По подразбиране при инсталиране на MS Access (като част от Microsoft Office) на потребителският компютър, подразбиращото се местоположение за БД на Access (напр. „C:\Program Files (x86)\Microsoft Office\Office12\ACCWIZ\") се обявява като сигурно (надеждно). Ако обаче създадете БД и запишете файла XXXX.accdb в друга директория, при опит за изпълнение на заявки, които не са селективни, в долната лента на прозореца ще се изведе прозорец със съобщение:



или съобщение:

"The action or event has been blocked by Disabled Mode."

В първият случай за да се разреши използването на заявки, които не са селективни е достатъчно да се щракне на бутона "Enable Content" на прозорец със съобщение. Ако обаче се изведе второто съобщение, е необходимо директорията да се добави в списъка на надеждните локации „Trusted locations" в настройките на Access. За целта от бутона „Office Start" (или от менюто File) се избира „Access Options" и от менюто вдясно на прозореца „Access Options" се избира „Trust Center" и се щраква на бутона „Trust Center Settings". Извежда се прозорец Trusted Locations, в който чрез бутона

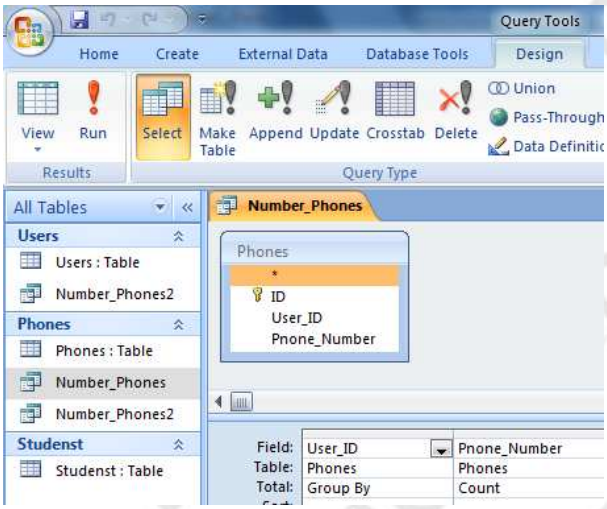


Фиг. 14 Добавяне на директория в списъка на надежните локации за БД

„Add new location” се добавя пътя до директорията, в която сме записали файла на БД.

5.9.2. Заявки за създаване на таблица по данни от селективна заявка

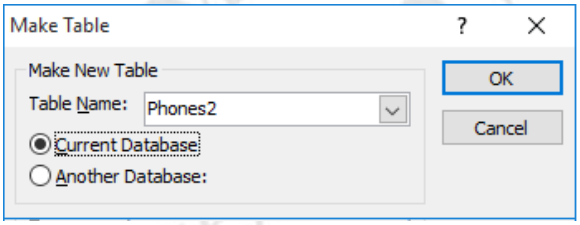
Чрез такава заявка се създава таблица, която съдържа данните, получени при изпълнението на селективна заявка.



Фиг. 15 Създаване на таблица по данни от заявка

За създаването на такава заявка се отваря селективната заявка в изглед Query Design и се щраква на бутона „Make Table” в менюто Design. Извежда се диалогов прозорец, в които операторът трябва да въведе името на таблицата и да избере дали тя да се създаде в текущата или в друга БД. След потвърждаване с бутона ОК Access генерира SQL кода за създаване на таблицата.

За да се изпълни той обаче, е необходимо да се щракне върху бутона „!” (Run). Извежда се диалогов прозорец със съобщение от вида:



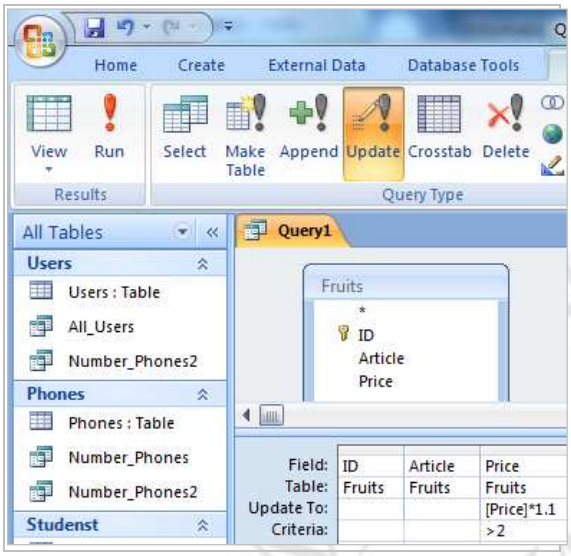
Фиг. 16 Задаване на име на таблицата

Фиг. 17 диалогов прозорец с информация за таблицата

Ако операторът щракне на бутона “Yes” заявката се изпълнява и се създава таблица със същият брой и тип колони както получените от селективната заявка и в нея се записват извлечените чрез селективната заявка данни.

5.9.3. Заявка за променяне на записи в таблица от БД

Използва се в случаите, когато една и съща промяна трябва да се направи за голям брой записи. За създаването на такава заявка се отваря или се създава в изглед Query Design селективна заявка, която извлича всички записи, които трябва да бъдат променени и в менюто Design се щраква на бутона „Update”.



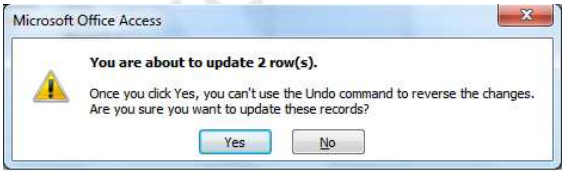
Фиг. 18 Създаване на заявка за променяне на данни в таблица

В таблицата със параметри на заявката се появява ред „Update To”. В него, в колоната, чийто данни ще се променят, се записва формула, чрез която да се получат новите стойности на полетата от колоната, след което се щраква на бутона “!” (Run) за да се изпълни заявката. По правило във формулата за изчисляване на новите стойности участват стойности на полета от текущия запис на таблицата.

Във формулата имената на тези полета трябва да се записват в квадратни скоби, както е показано във фигурата по-горе.

Например ако таблицата има колона Price, която съдържа цени на стоки, и искаме да умножим всички цени по 1.1 (т.е. да ги увеличим с 10%), в полето от реда Update To” за колоната трябва да запишем формулата *=[Price]*1.1* . Ако е необходимо да се променят само част от записите, извлечени от заявката, в реда Criteria може да се добави критерий, както е показано във фигурата.

След като се щракне на бутона “!” (Run) се извежда диалогов прозорец със съобщение за броя записи, които ще бъдат променени. Ако операторът потвърди с бутона “Yes” заявката се изпълнява. Ако превключим на изглед SQL View можем да видим SQL кода, който е генерирал Access за изпълнение на тази заявка.



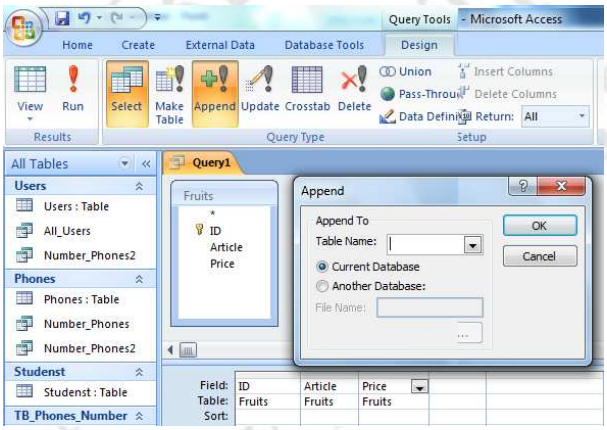

```
UPDATE Fruits SET Fruits.Price = [Price]*1.1
WHERE (((Fruits.Price)>2));
```

Фиг. 19 Пример за SQL код, генериран от Access за изпълнение на заявка Update

Заявката може да бъде съхранена и използвана повторно, но при всяко следващо отваряне Access извежда диалогов прозорец с въпрос дали искаме заявката да се изпълни. Ако операторът потвърди с бутона “Yes” заявката се изпълнява и данните се променят отново по зададените формули за изчисляване на новите стойности. Например ако отново изпълним горната заявка, цените в колона Price, за които се изпълнява зададеният критерий “>2” ще се увеличат с отново с 10%.

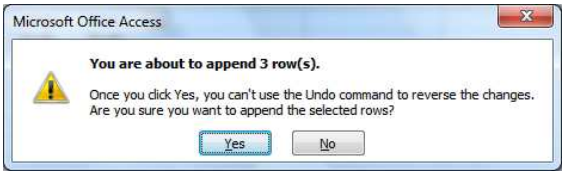
5.9.4. Заявка за добавяне на записи в таблица от БД

Този вид заявки са полезни тогава, когато е необходимо да се копират записи от една таблица в друга, например при периодично създаване на резервно копие на данните. За целта първо трябва да се създаде таблица-приемник със същата структура (брой колони от същият тип данни) както заявката-източник. Създава се заявката-източник в изглед Design View и се щраква на бутона „Append” в менюто Design. Извежда се диалогов прозорец, в който трябва да се въведе или избере името на таблицата-приемник от текущата или от друга БД. След потвърждаване с бутона “OK” Access създава кода на заявката. След като се щракне на бутона “!” (Run) се извежда диалогов прозорец със съобщение за броя записи, които ще бъдат копирани в таблицата-приемник.



Фиг. 20 Създаване на заявка за копиране на записи от една таблица в друга

Ако операторът потвърди с бутона “Yes”, заявката се изпълнява.

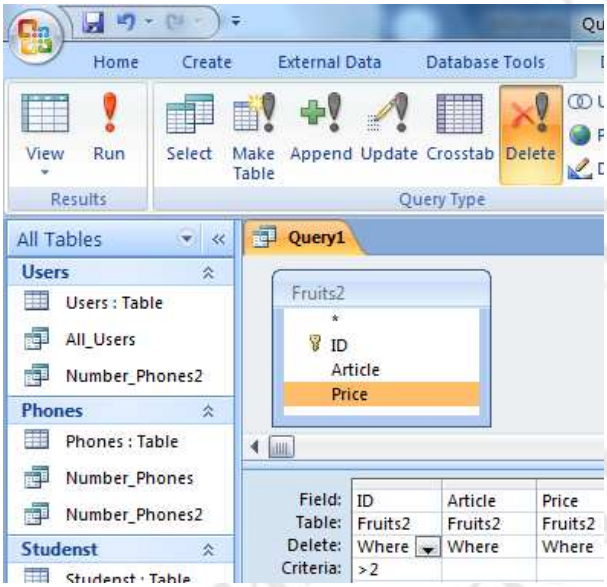


Ако превключим на изглед SQL View можем да видим SQL кода, който е генерирал Access за изпълнение на тази заявка:

```
INSERT INTO Fruits2 ( ID, Article, Price )
SELECT Fruits.ID, Fruits.Article, Fruits.Price FROM Fruits;
```

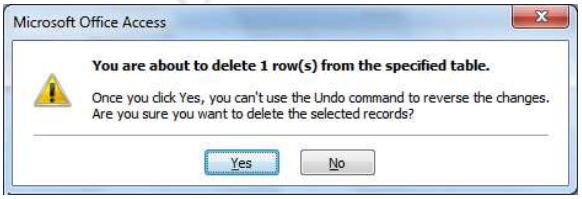
5.9.5. Заявка за изтриване на записи от таблица

Използва се в случаите, когато трябва да се изтрият записи от таблица по зададен критерий. За създаването на такава заявка първо се отваря или се създава в изглед Query Design селективна заявка, която извлича всички записи, които трябва да бъдат изтрити.



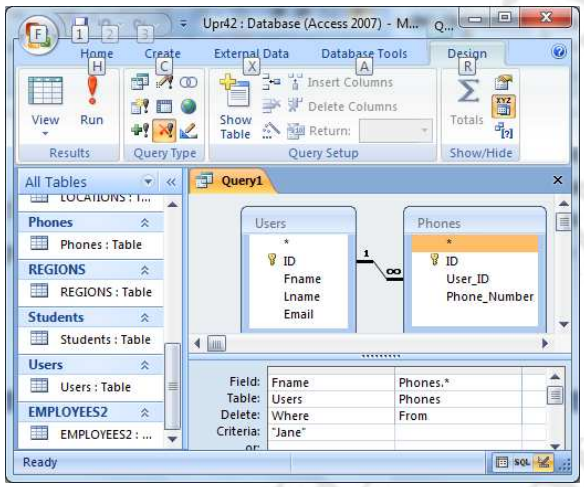
Фиг. 21 Създаване на заявка за изтриване на група записи от таблица

В таблицата в параметри на заявката (в долната част на прозореца) трябва да се включат (чрез провлачване от графичното изображение на таблицата) всички колони, за които ще се задават критерии. За създаването на кода на заявката се щраква на бутона „Delete” от менюто Design. В таблицата с параметри на заявката се появява ред „Delete”, а в полетата от реда по подразбиране се извежда клаузата Where. В реда Criteria на таблицата с параметрите, за всяка от колоните могат да се въведат критерии,по които да се селектират редовете, които ще изтрие заявката. Ако критерии не се зададат, ще бъдат изтрити всички записи от таблицата. За да се изпълни заявката се щраква на бутона ! (Run) от менюто Design. извежда диалогов прозорец със съобщение за броя записи, които ще бъдат изтрити. Ако операторът потвърди с бутона “Yes”, заявката се изпълнява. Ако превключим на изглед SQL View можем да видим SQL кода, който е генерирал Access за изпълнение на тази заявка. В примера заявката изтрива записи от таблица Fruits2, като за колоната ID е зададен критерий “>2” и за нея Access генерира следния SQL код:



```
DELETE Fruits2.ID, Fruits2.Article, Fruits2.Price
FROM Fruits2 WHERE (((Fruits2.ID)>2));
```

Полезно е да се има предвид и това, че в заявка за изтриване на записи може да се използва свързване между таблиците, но в тоя случай е необходимо да се укаже от коя таблица ще се изтриват записи. За целта в колона от таблицата за параметри на заявката се провлачва символът „*” от изображението на таблицата в тогава в реда Delete за тази колона се появява клауза From (вместо клауза Where за задаване на критерий). Като пример да разгледаме заявка за изтриването на записи от таблицата Phones, за които в свързаната таблица Users в колоната Lname има стойност “Jane”, т.е. заявката трябва да изтрие от таблица Phones всички телефони за клиент с име Jane.



Фиг. 22 Заявка за изтриване на записи от свързани таблици

За целта добавяме таблиците Users и Phones в прозореца на инструмента Query Desing. Между тях се изобразява връзка „едно към много” тъй като тя е създадена в БД. За да можем да зададем критерия ="Jane", провлачваме от таблица Users и поставяме в таблицата с параметри на заявката колоната Fname. За да укажем, че ще се изтриват записи от таблица Phones провлачваме от таблица Phones и поставяме в таблицата с параметри на заявката символа „*”. Когато щракнем с мишката на бутона „Delete” от менюто Design, в таблицата с параметри на заявката се появява ред “Delete”, в който в колоната „Fname” се извежда клаузата Where, а в колоната „Phones.*” – клаузата From. За да се изпълни заявката, както и в предният пример се щраква на бутона ! (Run) от менюто Design. Извежда се диалогов прозорец с информация колко записа ще бъдет изтрити, и ако потребителят потвърди с щракване на бутона “Yes” записите от таблица Phones, за които в колоната Fname от таблица Users има стойност "Jane" ще бъдат изтрити. Ако се превключи в изглед SQL View се вижда, че Access генерира следния SQL код:

```
DELETE Users.Fname, Phones.*
FROM Users INNER JOIN Phones ON Users.ID = Phones.User_ID
```

```
    WHERE (((Users.Fname)="Jane"));
```

В тази SQL команда в списъка от колони след командата DELETE таблица Users участва само с една колона, докато таблица Phones – с всичките си колони (това означава символът “*” след името на таблицата). От това сървърът за БД разбира, че трябва да изтрие записи само от таблица Phones.

https://www.tutorialspoint.com/ms_access/index.htm MS Access Tutorial
https://mkala.rpa-mu.ru/Media/mkala/UMP/OPD-kolledj/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0.pdf Лабораторный практикум по курсу «информатика и математика»
https://support.microsoft.com/en-us/office/download-and-install-microsoft-365-access-runtime-185c5a32-8ba9-491e-ac76-91cbe3ea09c9 Download and install Microsoft 365 Access Runtime
https://download.cnet.com/Access-2007-Download-Access-Runtime/3001-2070_4-10726092.html
https://www.guru99.com/ms-access-tutorial.html Microsoft Access Tutorial: Learn MS Access with Example
https://www.yourofficeanywhere.co.uk/info-hub/microsoft-access-online-in-browser/
https://access-templates.com/tag/student+database.html Начальной Курс для Access
http://www.it.sou-dolnichiflik.com/BD_query.pdf Заявки в Microsoft Access - създаване и редактиране
https://asenevtsi.com/ebooks/A2007/T045.htm ВИДОВЕ ЗАЯВКИ

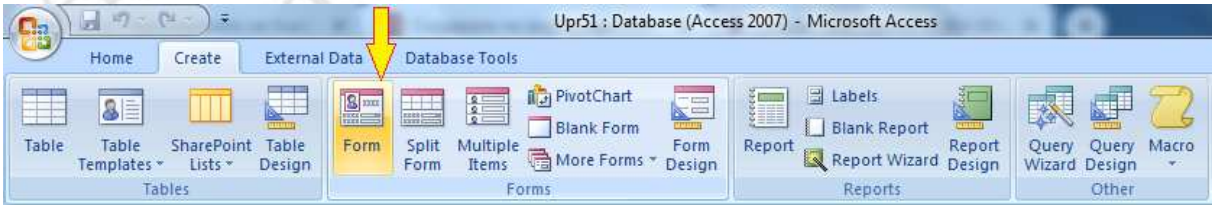
5.10. Формуляри в MS Access

Формулярът в Access е обект на базата от данни, който се използва за създаване на потребителски интерфейс за операции с данните. Свързан (bound) е този формуляр, който е пряко свързан с източник на данни, като таблица или заявка, и може да се използва за въвеждане, редактиране или показване на данни от този източник на данни. Като алтернатива разработчикът може да създаде „несвързан“ формуляр, който не се свързва директно с източник на данни, но който може да съдържа командни бутони, етикети или други контроли, които са му необходими за работа с приложението.

В настоящият курс ние ще разглеждаме основно свързаните формуляри. Чрез свързаните формуляри може да се контролира достъпът до данни, като например кои полета или редове данни се показват. Например, за някои потребители достъпът до определени таблици може да се ограничи само до няколко полета. Предоставянето на тези потребители на формуляр, който съдържа само тези полета, ги улеснява при използването на базата данни. Освен това чрез добавяне на бутони и други потребителски контроли и код, който да се извиква за изпълнение чрез тях могат да се автоматизират често извършвани действия.

5.10.1. Създаване на формуляр в изглед Layout View

За да създадем формуляр в изглед Layout View (изглед за оформление) отваряме базата от данни и избираме от прозореца All Tables таблицата, за която ще създаваме формуляр. Отваряме менюто “Create” (Създаване) и избираме „Form”.



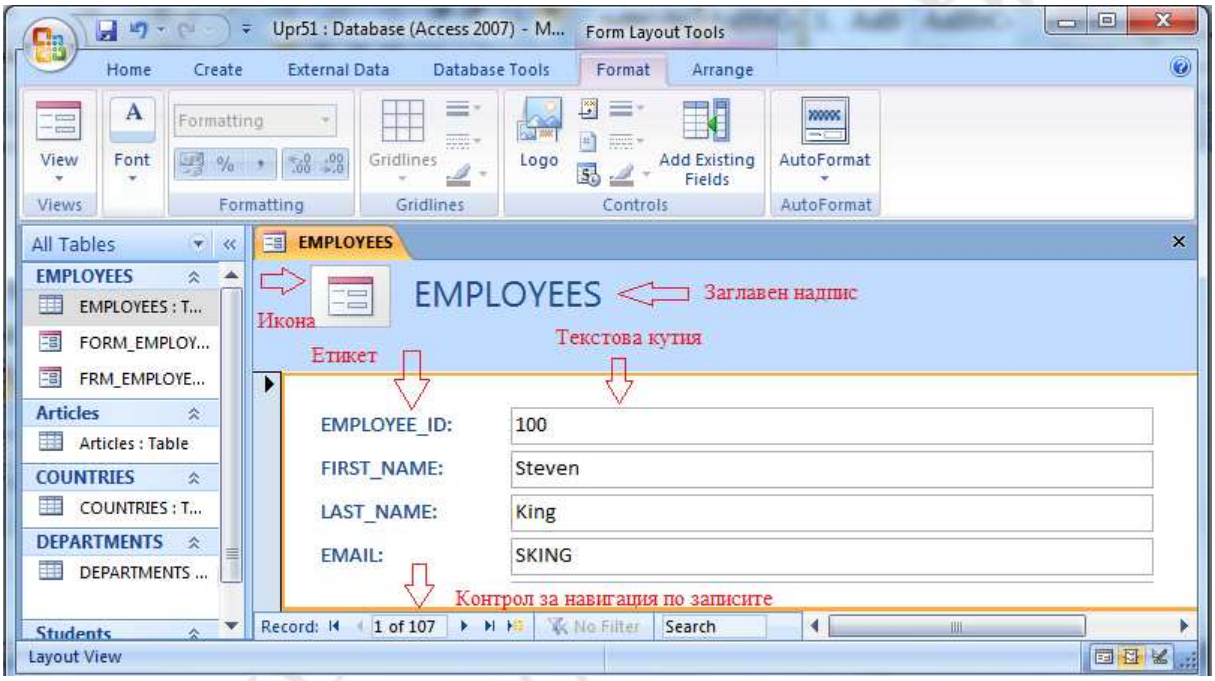
Фиг. 23 Създаване на прост формуляр чрез менюто Create

Access създава формуляр, който съдържа:

- Заглавна част (Form_Header) с икона (контрол Image с име Auto_Logo0) и заглавен надпис (Auto_Title0).

- Работна секция (Detail), която съдържа за всяка колона от таблицата двойка обекти (т.н. „контроли") – етикет и текстова кутия. В етикета се извежда името (или заглавният надпис, ако е зададен) на колоната, а в текстовата кутия – съдържанието на полето от тази колона за избраният запис.

В затварящата част (FormFooter) се извежда контрол за навигация по записите от таблицата с бутони за първи, предидущ, следващ и последен запис.



Фиг. 24 Обекти (контроли) на формулярите

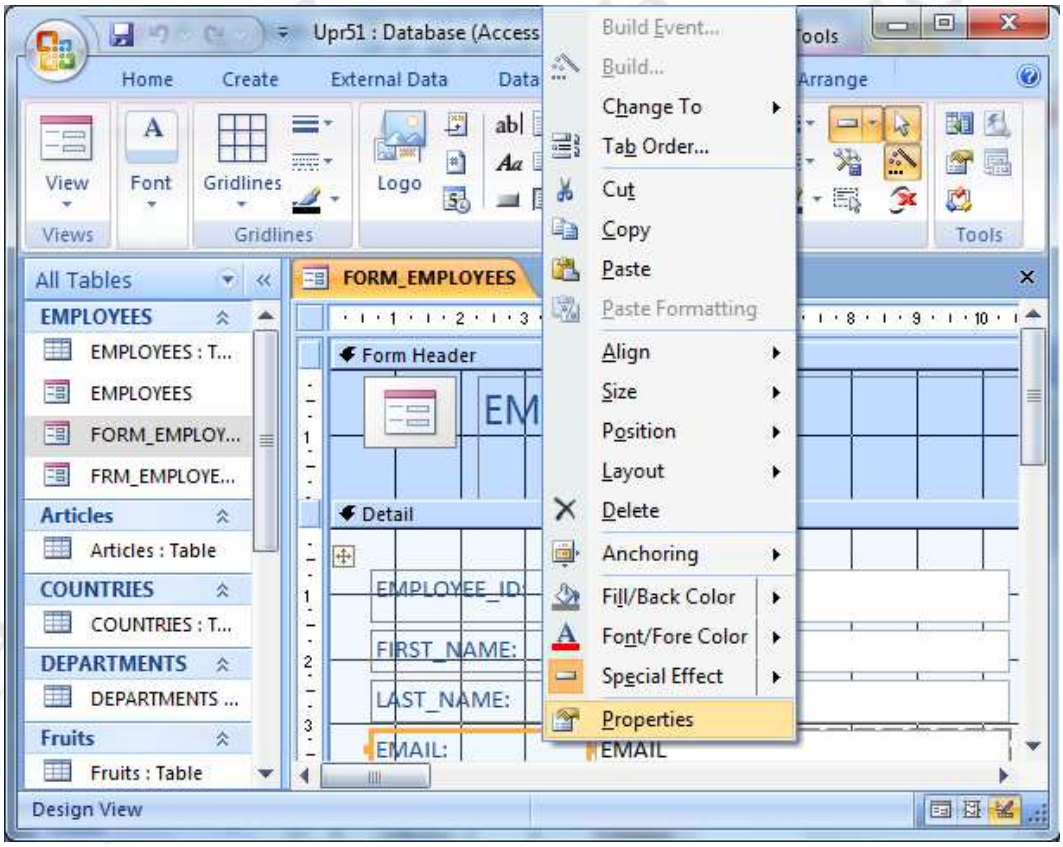
5.10.2. Контролите ActiveX controls

Думата „контрол" е кратко, „побългарено" наименование на ActiveX control. Контролът ActiveX control е програмен обект, който има графичен интерфейс и определено поведение при действия на потребителя (т.н. събития). Той може да се използва многократно от приложните програми в рамките на компютър или между компютри в мрежа. Технологията за създаване на ActiveX контроли е част от цялостния набор от технологии на Microsoft, основен от които е Component Object Model (COM). Microsoft предоставя библиотеки от ActiveX контроли с различно предназначение – Label, Text Box, Picture, Image, Scroll Bar.

В Access ActiveX контролите се използват във формулярите за създаване на графичен интерфейс (надписи, изображения), за извеждане и променяне на съдържанието на таблици от БД. Във формуляра, показан на фигурата, в иконата (обект Image) се извежда статично изображение, в етикетите (обект Label) се извеждат имената на колоните, а в текстовите кутии (обект TextBox) се извежда съдържанието на полетата от текущия запис.

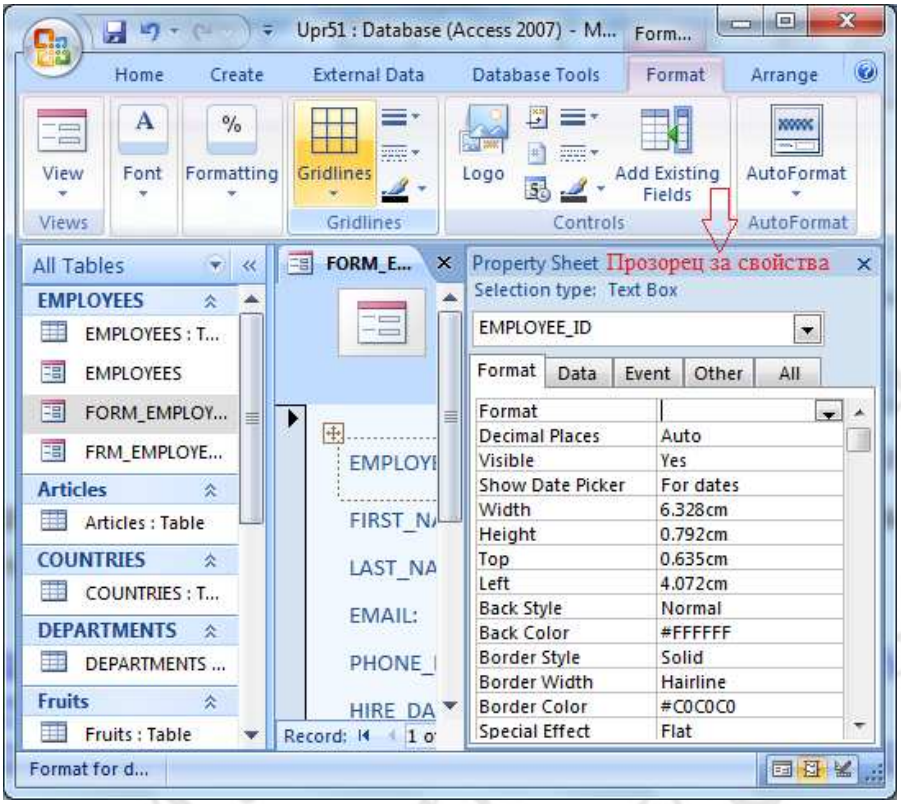
5.10.3. Променяне на свойствата на контролите

ActiveX контролите предоставят голям брой свойства (properties), с които може да се променя вида, в който контролът се извежда във формуляра. За променяне на свойствата на контрол е необходимо да се щракне с десният бутон на мишката върху контрола и от плаващото меню да се избере „Properties”.



Фиг. 25 Отваряне на прозореца за свойства на контрол

Вдясно се отваря прозорец в който в първа страница се извеждат имената на свойствата и текущите им стойности.



Фиг. 26 Прозорец за свойства на контрол

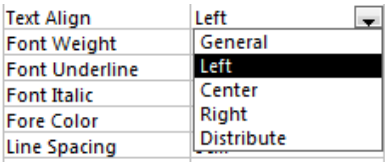
Най-горе се извежда типа на контрола Selection Type (във фигурата по-горе това е Text Box – текстова кутия). В списъчният контрол по-долу се извежда името на контрола (в примера EMPLOYEE_ID).

5.10.3.1. Страница Format на прозореца за свойства на контрол

В страницата Format се извеждат свойства за форматиране, например:

За размери и позиция	Width (дължина), Height (височина), Left (хоризонтална) и Height (вертикална) – координати на горния ляв ъгъл
За рамка	BorderStyle (вид на линията), BorderWidth (дебелина на линията) , BorderColor (цвят на линията)
За текст	FontName (име на шрифта), FontSize (размер на шрифта), TextAlign (подравняване спрямо рамката)

За някои от свойствата възможните стойности се избират от списък.



Фиг. 27 Списък за избор на подравняване

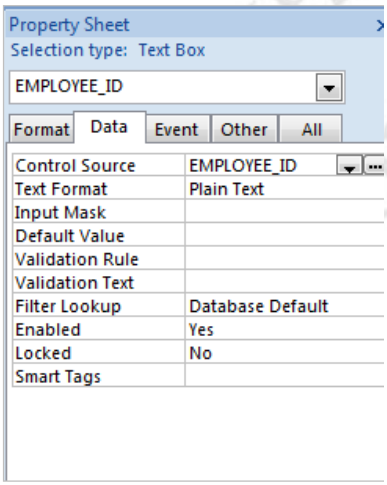
Фиг. 28 Прозорец за избор на цвят

Например за свойството TextAlign (подравняване на текста спрямо рамката) се избира стойност от списък. За целта се щраква на бутона и се избира една от стойностите от списъка, който се извежда.

В други случаи се избира стойност от диалогов прозорец. В такъв случай при щракване с мишката върху реда от таблицата за съответното свойство се появява бутон . Например за стойност на свойството BorderColor (цвят на рамка) може да се избере цвят от списък с имена на цветове като се щракне на бутона или цвят от диалогов прозорец с палитра от цветове, който се извежда когато се щракне на бутона .

5.10.3.2. Страница Data на прозореца за свойства на контрол

В страницата Data се извежда информация за източника на данни за контрола, ако контролът е свързан към базата от данни.

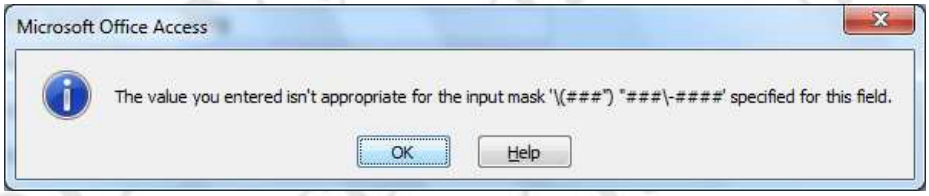


За несвързаните контроли (например контрол Label, който съдържа статичен текст), в страницата Data не се извежда нищо.

Свойство “Control Source”: Задава източника на данни за контрола. В примера контролът EMPLOYEE_ID е свързан към колоната EMPLOYEE_ID от таблицата Employees, за която е създаден формуляра. Това е зададено като стойност на свойството “Control Source” вдясно. От падащия списък, който се извежда когато се щракне с мишката на бутона може да се избере друга колона, към която да се свърже контрола, но ако формулярът е генериран автоматично, по правило стойността на свойството “Control Source” не трябва да се променя. С бутонът се отваря прозорец на помощна

програма Expression Builder за създаване на израз (формула), чрез който да се изчислява стойността за контрола.

Свойство „Input Mask”: Маските за въвеждане са полезни за операции по въвеждане на данни, като например маска за въвеждане на поле за телефонен номер, което ви показва как точно да въведете нов номер: (___) ___-____. Маската се задава като последователност от определени символи, с които се указва какъв символ може да се постави на всяка позиция от въвежданият от контрола текст. Като пример да разгледаме маската за въвеждане на телефонен номер (###) ###-####. Със символа “#” се указва, че на дадената позиция може да стои само цифра от 0 до 9, а символите "(", ")", и "-" изискват точно съвпадение, т.е. трябва да се въведат така, както са записани в маската. Ако в текстовата кутия се въведе текст, който не съответства на маската се извежда събщение в диалогов прозорец и се очаква коригиране на въведеният текст.



За примера при зададена маска (###) ###-#### на позициите на символа”#” контролът ще приема само въвеждане на цифри.

В маските могат да се използват следните символи:

0	цифра (от 0 до 9, задължително въвеждане).
9	цифра (от 0 до 9, незадължително въвеждане).
#	Цифра или празен интервал (от 0 до 9, незадължително въвеждане). Празните интервали се извеждат при редактиране но не се записват.
L	буква (от A до Z, задължително въвеждане).
?	буква (A to Z, незадължително въвеждане).
A	буква или цифра (задължително въвеждане).
a	буква или цифра (незадължително въвеждане).
&	всеки символ или празен интервал (задължително въвеждане).
C	всеки символ или празен интервал (незадължително въвеждане).

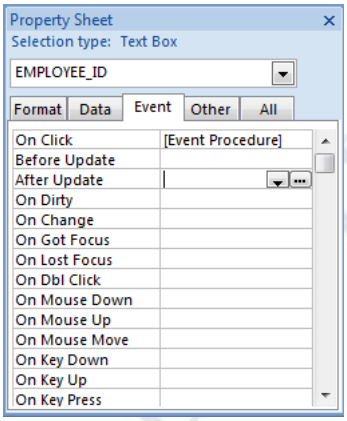
Свойство Enabled: По подразбиране има свойство Yes, което означава, че контролът е активен. При No контролът не е активен, изобразява се със сив цвят и не приема курсора на мишката.

Свойство Locked (заключен): По подразбиране има свойство No, което означава, че контролът е активен и в неко може да се въвежда. При Yes

контролът не е активен, но приема курсора на мишката и съдържанието му може да се копира.

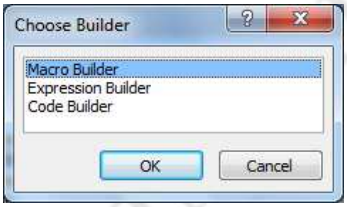
5.10.3.3. Страница Event на прозореца за свойства на контрол

В страницата Event се извеждат събитията (действия на оператора или на операционната система) и имената на процедурите (с код, създаден от разработчика), които се изпълняват при възникване на съответните събития. Например събитието OnClick настъпва тогава, когато потребителят натисне бутон на мишката когато курсорът е върху контрола.



Фиг. 29 Страница за събития Event

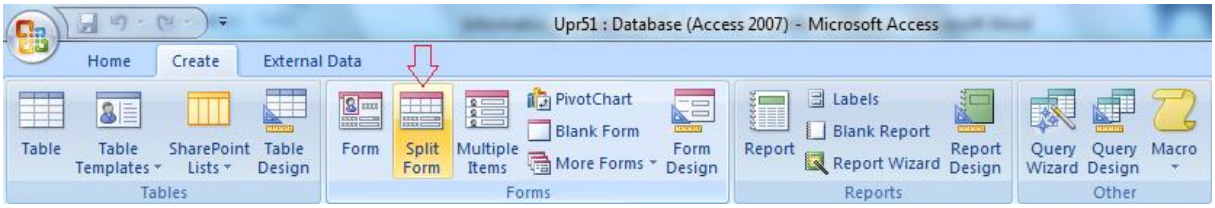
За да се създаде процедура за дадено събитие се щраква с мишката в дясната клетка срещу името на събитието в таблицата със събития.



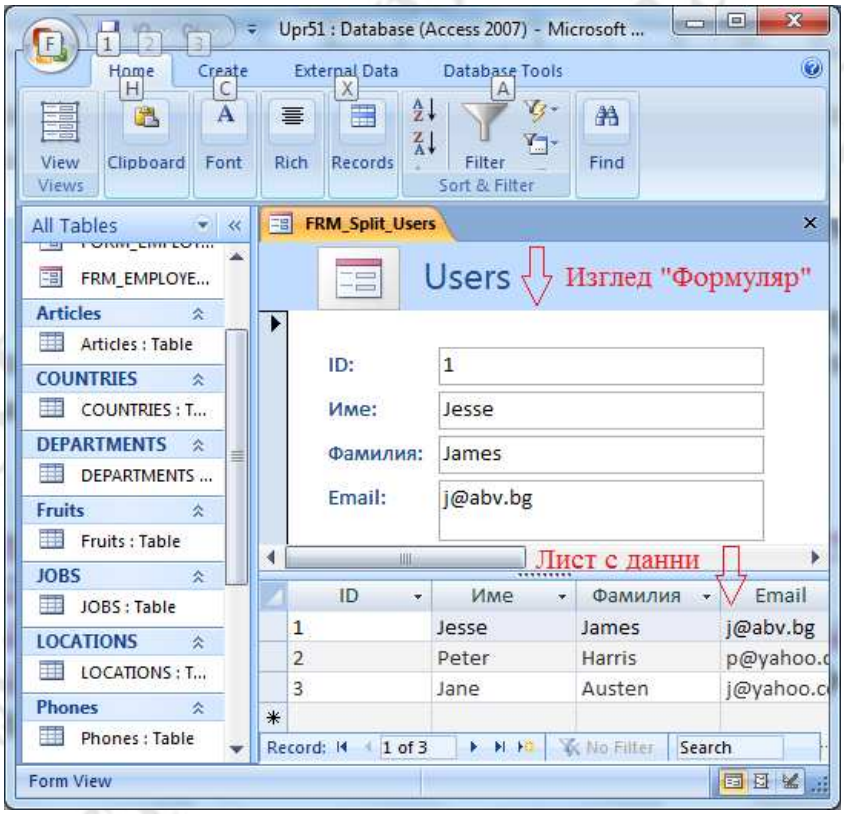
Извежда се бутон с три точки ..., при щракване върху който се отваря прозорец за избор на помощна програма за създаване на събитийна процедура. Ако за дадено събитие е създадена процедура, в таблицата срещу името на събитието се извежда текст [Event Procedure].

5.10.4. Създаване на разделен формуляр Split Form

Разделеният формуляр, за разлика от простият формуляр, дава два изгледа на данните едновременно – изглед "Формуляр" и изглед "Лист с данни". За да създадем разделен формуляр отваряме базата от данни и избираме от прозореца All Tables таблицата, за която ще създаваме формуляр. Отваряме менюто “Create” (Създаване) и избираме „Split Form” (Разделен формуляр).



В разделеният формуляр двата изгледа са свързани към един и същ източник на данни и са постоянно синхронизирани един с друг. Избирането на поле в едната част от формуляра избира същото поле в другата му част. Потребителят може да добавя, редактира или изтрива данни от всяка част (стига източникът на записа да може да се актуализира и ако формулярът не е конфигуриран да не допуска тези действия).

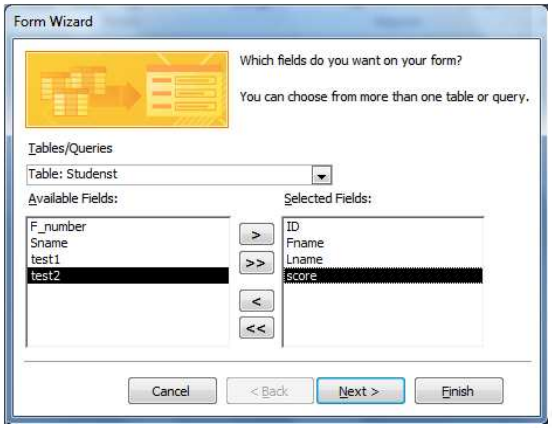


Фиг. 30 Разделен формуляр

Работата с разделени формуляри ви предоставя предимствата на двата типа формуляри в един формуляр. Например можете да използвате частта на лист с данни във формуляра за бързо намиране на запис и след това да използвате формулярната част, за да прегледате или редактирате записа.

5.10.5. Създаване на формуляр с инструмента Form Wizard

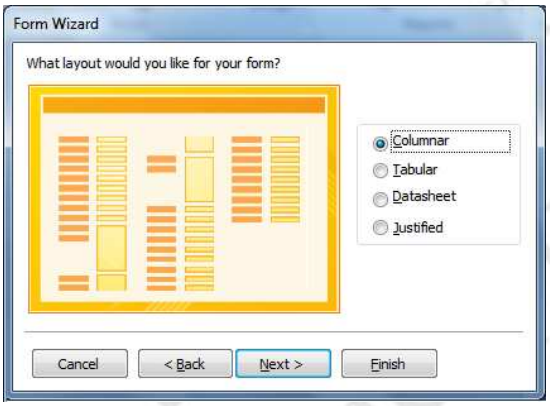
Инструментът Form Wizard дава възможност за създаване на формуляр в диалогов режим, като на всяка стъпка се избират различни елементи от формуляра.



Фиг. 31 Избор на източник на данни за формуляр

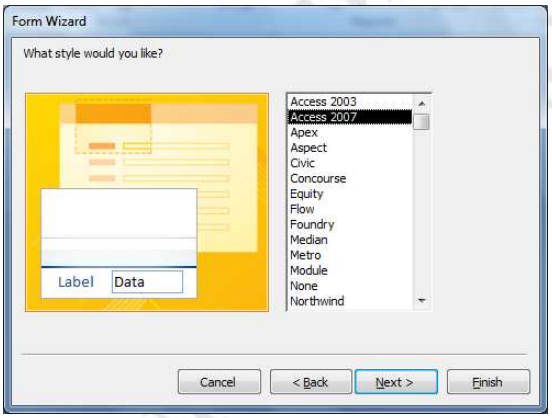
На първа стъпка се избира източникът да данни за формуляра (таблица или заявка) и с бутона > се прехвърлят вдясно имената на колоните, от които ще се извеждат данни.

На следващата стъпка се избира един от няколко възможни подреждания



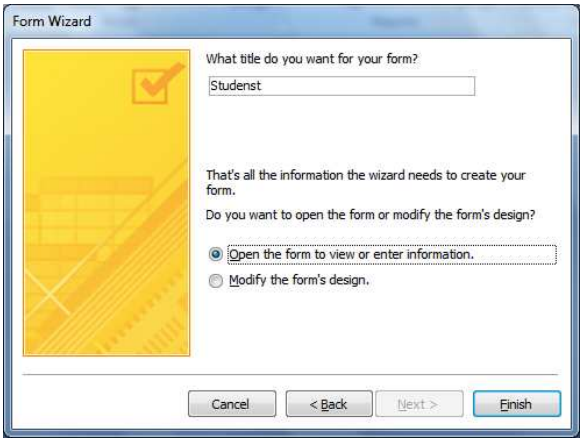
на елементите – по колони (Columnar), с отместване с табулации (Tabular), като таблица на Excel (Datasheet) или с размери според съдържанието (Justified).

След като се избере едно от тези подреждания и се щракне с мишката на



бутона "Next" се извежда прозорец за избор на стила на контролите в отчета (цвят на рамката, фонов цвят, цвят на текста).

На последната стъпка от помощната програма Form Wizard се дава на опе-



ратора възможност да зададе име на формуляра и да избере дали той да бъде отворен за извеждане на информация или за променяне.

5.10.6. Използване на списъчен контрол ComboBox във формуляр

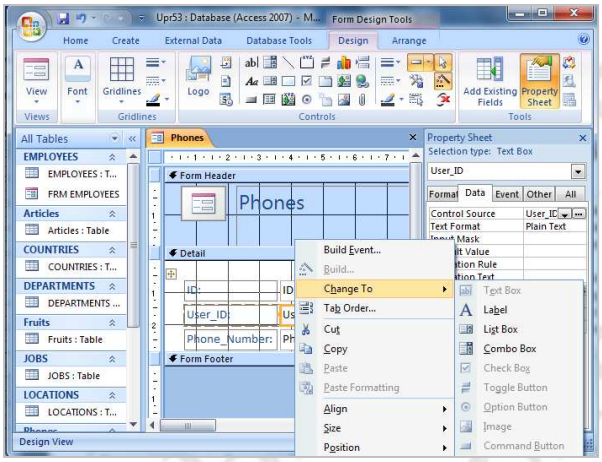
При въвеждане на данни в таблица, която е свързана с друга, главна таблица, в колоната, чрез която се извършва връзката (т.н. Външен ключ Foreign Key) не могат да се въвеждат произволни стойности, а само такива, които ги има в главната таблица. Ако се въведе стойност, която я няма в свързаната колона от главната таблица (по правило това е първичният ключ Primary Key), сървърът генерира грешка и отказва да съхрани записа. Например ако в таблицата Users има записи за трима потребители с първичен ключ ID = 1, 2 и 3, а в таблица Phones са записани телефони на тези потребители и колоната User_ID е свързана с колоната ID от таблица Users, то ако се опитаме да въведем в колоната User_ID на таблица Phones стойност, която не е 1, 2 или 3, ще се генерира грешка

Фиг. 32 Съобщение за грешка при опит за въвеждане на невалидна стойност в поле от запис при свързани таблици

Тогава, когато потребителят на БД при въвеждане на данни в колона може само да избира от списък от стойности (както е в случай че се въвеждат данни в свързана колона), е по-добре във формуляра да му се предоставя списъкът от допустимите стойности, вместо текстова кутия, както е по подразбиране. Такава възможност предоставят списъчните контроли ListBox и ComboBox. Предпочита се използването на ComboBox,

защото той при избиране на стойност извежда падащ списък, а когато не е избран е с размер на текстова кутия.

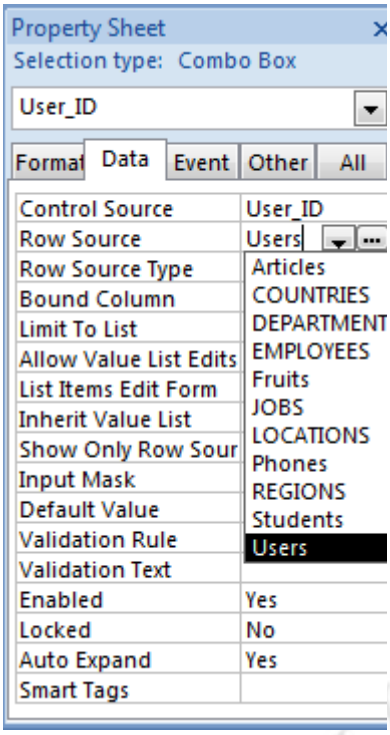
Като пример за използване на ComboBox ще създадем формуляр за таблица Phones в който стойността на User_ID ще се избира от стойностите в свързаната колона ID на таблица Users. За целта отваряме създаденият формуляр FRM_Phones в изглед Design View (Изглед на дизайна). Щракваме с десния бутон на мишката върху текстовата кутия за User_ID и от плаващото меню избираме Change To →Combo Box. В прозореца Properties (Свойства) се отваря страница "Data".



Фиг. 33 Заместване на текстова кутия със списъчен контрол

- Свойството "**Control_Source**" има стойност User_ID – името на колоната от таблицата, към която е свързан контрола.
- Свойството "**Row Source Type**" задава вида на източника на данни за контрола ComboBox и по подразбиране има стойност "Table/Query (Таблица/Заявка)".
- Свойството "**Bound_Column**" задава индекса на колоната от източника (по правило главната таблица), от която се получава стойността за запис в колоната-приемник User_ID. По подразбиране има стойност 1, т.е. данните ще се вземат от първата колона (ID) на таблица Users.

Свойството "**Control Source**" трябва да се свърже към колоната ID на главната таблица (в примера таблицата Users), към която е свързан контрола. От тази колона се получава списъкът от стойности, от които потребителят избира стойността, която се получава от контрола.

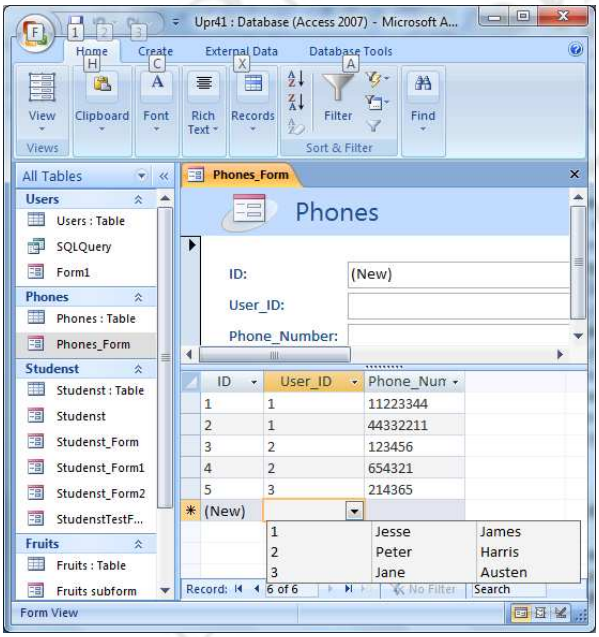


Фиг. 34 Задаване на свойства на списъчен контрол

Свойството "**Row Source**" трябва да се свърже към таблицата Users, от която ще се попълнят колоните в падащият списък. За целта щракваме с мишката върху бутона и от падащия списък избираме таблицата Users. Така в контрола ComboBox ще се попълни съдържанието на избраната таблица Users.

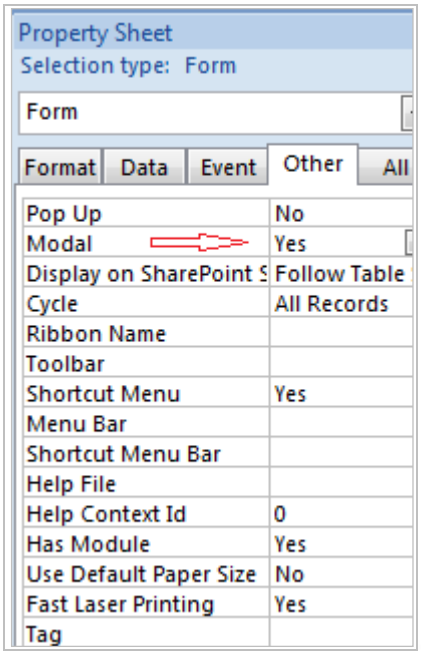
Ако искаме да се попълнят данните само от част от колоните (евентуално и за част от записите по зададен критерий) можем да създадем от таблицата заявка като щракнем на бутона .

Друго свойство, което можем да зададем, е свойството "Column_Count" - броят колони от източника (таблица Users) които да се извеждат в контрола ComboBox . В случая е най-подходящо да се извеждат стойности от три колони: ID, Fname и Lname. Така, когато избира стойността на номера на клиента (User_ID) потребителят ще вижда и неговото име и фамилия, както е показано на фигурата вляво. За да зададем броя колони, които да се извеждат в списъчният обект отваряме страницата Format на прозореца Properties и задаваме стойност 3 на свойството "Column_Count".



5.10.7. Използване на формуляр като модален прозорец

Свойството Modal на формуляр указва дали формулярът се отваря като модален прозорец. Когато формуляр се отвори като модален прозорец, се забранява отварянето на други прозорци в Access докато не се затвори отвореният формуляр.



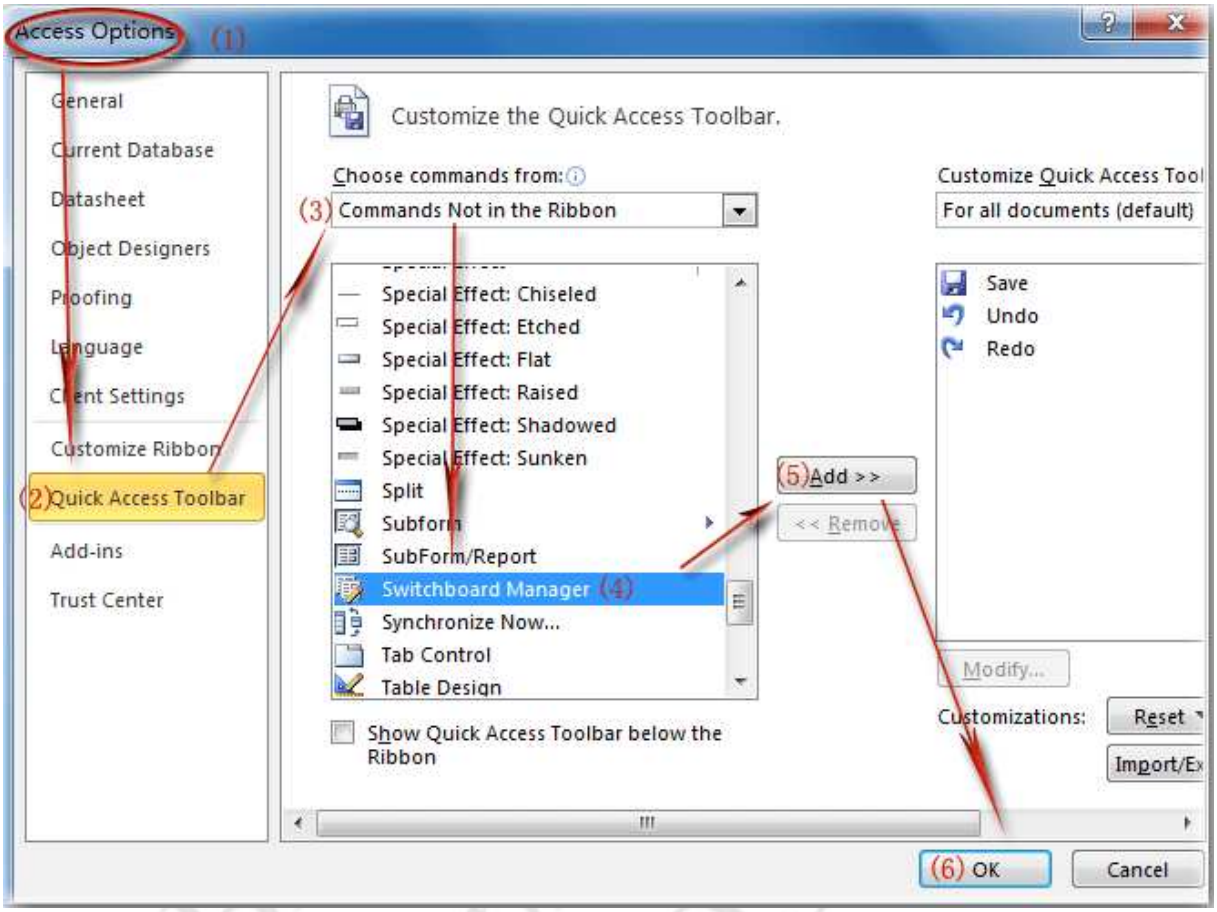
Това понякога е необходимо за да се избегне отварянето на много обекти в БД (таблици, формуляри, отчети) когато променянето на данните в един обект трябва да промени данните в друг обект. За да се отваря формулярът в изглед “Form View” като модален прозорец го отваряме в изглед “Design View” и в прозореца Properties на страница „Others” променяме свойството Modal в “Yes”.

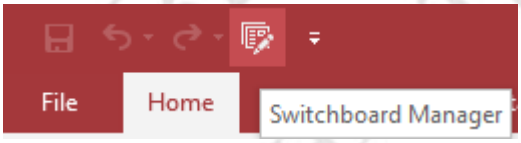
5.11. Специални видове формуляри

MS Access дава възможност за създаване на някои формуляри с предназначение, различно от това на разгледаните дотук формуляри за създаване на потребителски интерфейс за операции с данните.

5.11.1. Създаване на команден панел (Switchboard)

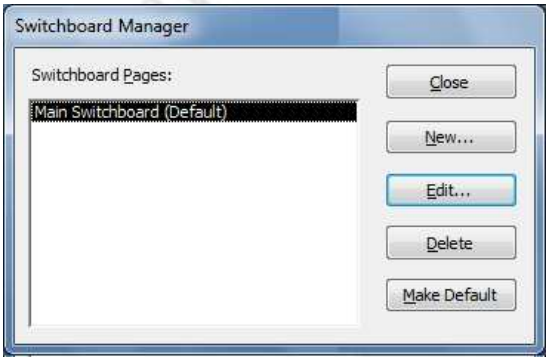
Командният панел е формуляр, на който разработчикът може да постави бутони за отваряне на обекти от БД и пояснителен текст към тях. Добавянето на команден панел към приложение на Access дава възможност на потребителите по-бързо да намерят необходимите им обекти в базата от данни.
За създаване на команден панел се използва инструментът Switchboard Manager от менюто Database Tools.





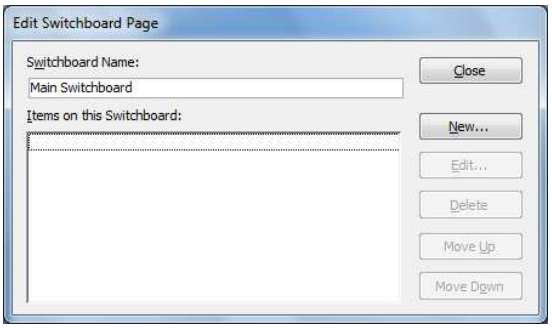
При стартирането му се извежда прозорец

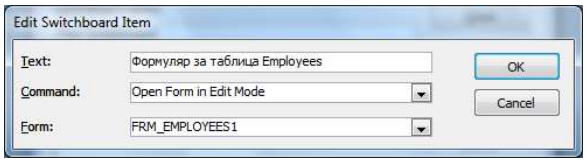
След като разработчикът потвърди създаването на команден панел, панелът се създава и в прозореца All Tables се добавя група Switchboard



Извежда се прозорец на инструмента Switchboard Manager, чрез който разработчикът може да добави към панела бутони за отваряне на обекти (таблици, формуляри, отчети) от БД.

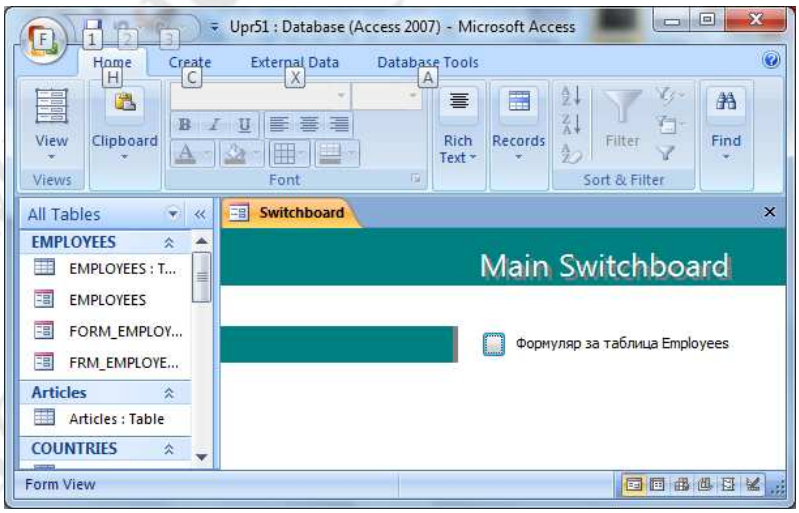
За редактиране на съдържанието на панела (добавяне на бутон за отваряне на обект) се щраква на бутона „Edit”. Извежда се прозорец, в който може да се промени името на панела и да се добавят бутони за отваряне на





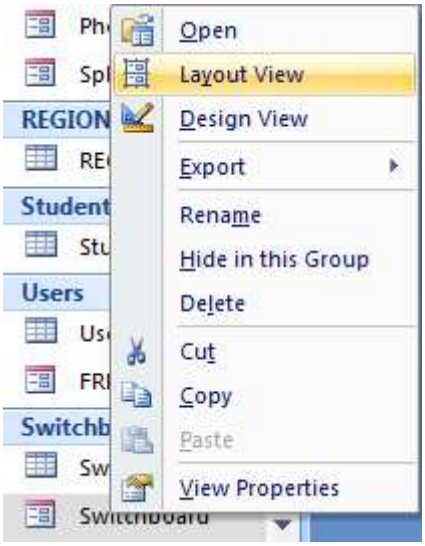
Фиг. 35 Добавяне на бутони към Switchboard панела

обекти. За добавяне на бутон се щраква върху бутона „New”. Извежда се прозорец (фигурата вдясно), в който се въвежда пояснителен текст (Text), избира се действие (Command) и обект от падащ списък. В примера е въведен пояснителен текст „Формуляр за таблица Employees”, избрана е команда „Open Form in Edit Mode” и като обект е избран формулярът „FRM_EMPLOYEES1”. След като се потвърди добавянето с бутона “OK” прозорецът се затваря и бутонът (по-точно пояснителният му текст) се добавя към списъкът от задачи на Switchboard страницата. След затварянето на прозореца за страницата и прозореца за Switchboard панела, панелът може да бъде отворен с Open за да се провери неговото функциониране. По подразбиране се създава Switchboard страница със следният дизайн:



Фиг. 36 Вид на Switchboard панел с бутон за отваряне на формуляр

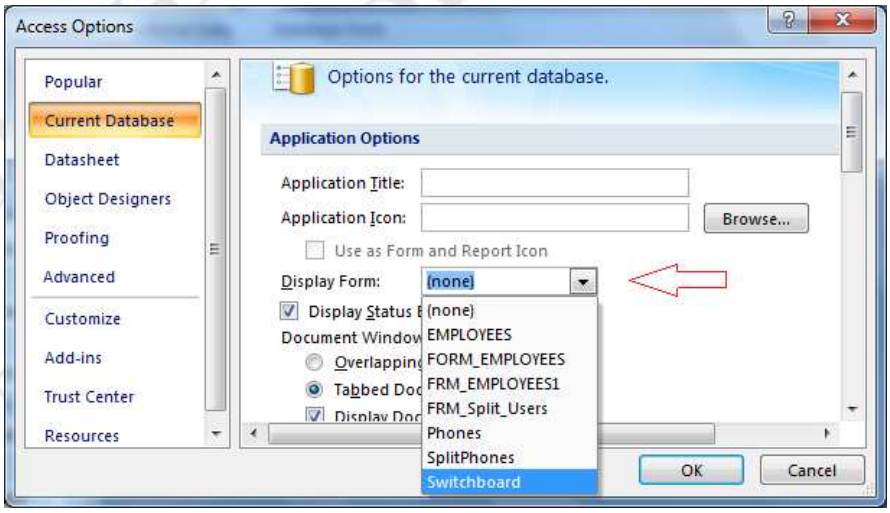
При щракване с мишката върху бутона или върху пояснителният текст се



изпълнява зададеното действие (в примера – отваря се формулярът за таблица Employees). Панелът може да бъде отворен в изглед “Layout View”или в изглед “Design View” за променяне на дизайна или съдържанието на панела.

5.11.2. Настройка за автоматично отваряне на Switchboard панел

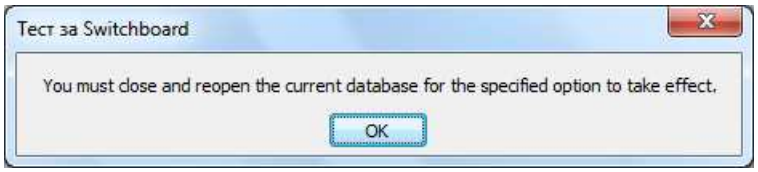
За да се улесни работата на потребителят на СУБД Access с обектите от БД е желателно панелът за навигация да се отваря автоматично при отваряне на БД. За целта от бутона „Office Start” (или от менюто File) се избира „Access Options” и от менюто вдясно на прозореца „Access Options” се избира “Current Database”. Формулярът (в случая Switchboard) който ще се отваря автоматично при отварянето на БД се избира от падащият списък „Display Form”, както е показано на фигурата по-долу.



Фиг. 37 Настройка за автоматично отваряне на панел за навигация

След като се избере от списъка формулярът, който да се отваря автоматично и се щракне на бутона “OK” за потвърждение, се извежда

съобщение, че БД трябва да се затвори и да се отвори отново за да се активира зададената опция



Две допълнителни настройки, които могат да се зададат чрез този прозорец са:

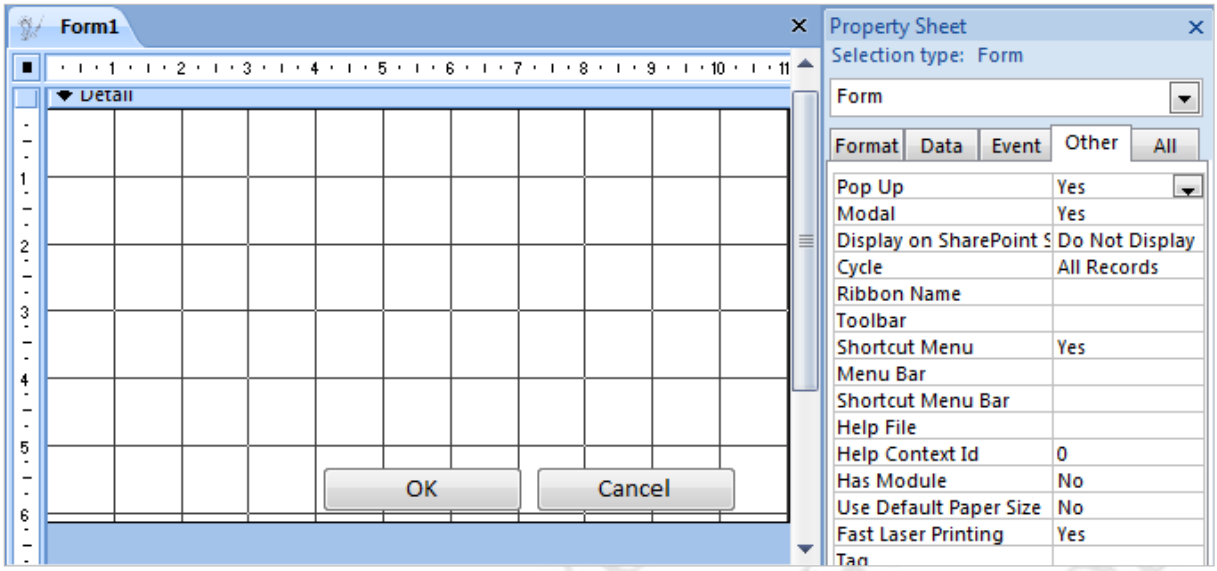
Application Title:	надпис в лентата на прозореца на Access – по подразбиране се извежда името на файна за БД
Application Icon:	Графичен файл с разширение .ico или .bmp , който да се извежда като икона по подразбиране за формулярите и отчетите в БД.

За да се извежда иконата в раздела на документа по подразбиране за всеки формуляр или отчет е необходимо да се селектира и опцията „Use as Form and Report Icon”.

5.11.3. Диалогов формуляр (Dialog Box Form)

Формулярите за диалогов прозорец (диалогови формуляри) могат да се използват в Access за показване на резултати от търсене, което се извършва от потребителя. Може също да се използва от крайния потребител за избор на специфични условия за извличане на данни. Подобно на формуляра за превключватели, формулярът за диалогов прозорец е несвързан, така че чрез него не може да се променят данните в таблиците на Access.

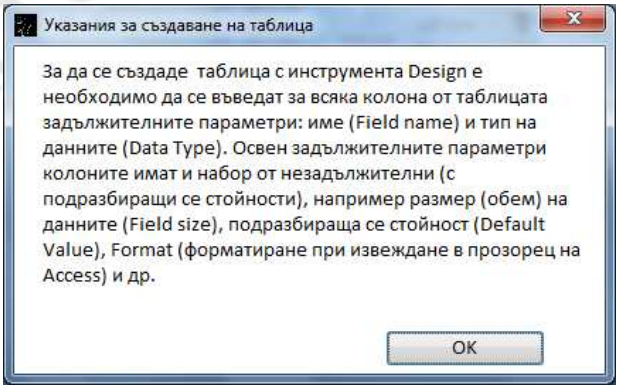
За създаване на диалогов формуляр от менюто Create се избира “More Forms” и от плаващото меню се избира Modal Dialog. Създава се несвързан формуляр, който съдържа само два бутона. По-долу е показан вида на формуляра в изглед Design View.



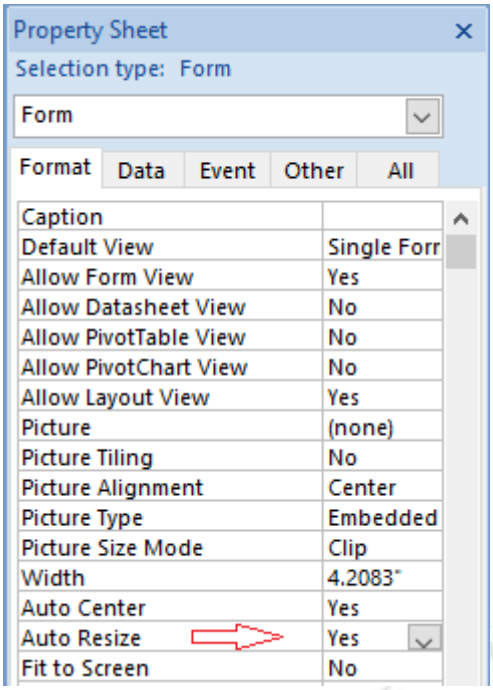
Фиг. 38 Начален вид на диалогов формуляр в изглед Design View

Както се вижда от фигурата, първоначално диалоговият формуляр съдържа само два бутона – OK и Cancel, но от менюто Design разработчикът може да добави в секцията Detail допълнителни контроли. Например ако в секцията Detail се добави контрол Label, в него може да се изведе статичен (т.е. постоянно зададен) текст и диалоговият формуляр може да с използва за извеждане на помощна информация при отваряне на някой от обектите (таблица, формуляр, отчет) в БД.

По подразбиране диалоговият формуляр се създава като модален прозорец (свойство Modal=Yes), който се извежда пред другите прозорци (свойство Pop Up=Yes). Ако формулярът се използва за извеждане на помощна информация, той не трябва да бъде модален (свойство Modal=No), за да може да има достъп и до другите отворени прозорци, но е добре да се извежда пред другите прозорци, т.е, стойността на свойството “Pop Up” да остане “Yes”.



Фиг. 39 Пример за използване на диалогов формуляр за извеждане на помощна информация



В изглед Design размерът на формуляра може да се промени чрез провлачване на рамката с натиснат ляв бутон на мишката. Но ако при това провлачване някои контроли останат извън новият размер, при отпускане на бутона формулярът автоматично се преоразмерява така че да побере контролите. За да се изведе формулярът при отваряне с Open с променениият размер, е необходимо на свойството **Auto Resize** да се зададе стойност Yes.

https://bg.eferrit.com/%D1%81%D1%8A%D0%B7%D0%B4%D0%B0%D0%B2%D0%B0%D0%BD%D0%B5-%D0%BD%D0%B0-%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8F%D1%80%D0%B8-%D0%B2-microsoft-access-2007/ Създаване на формуляри в Microsoft Access 2007

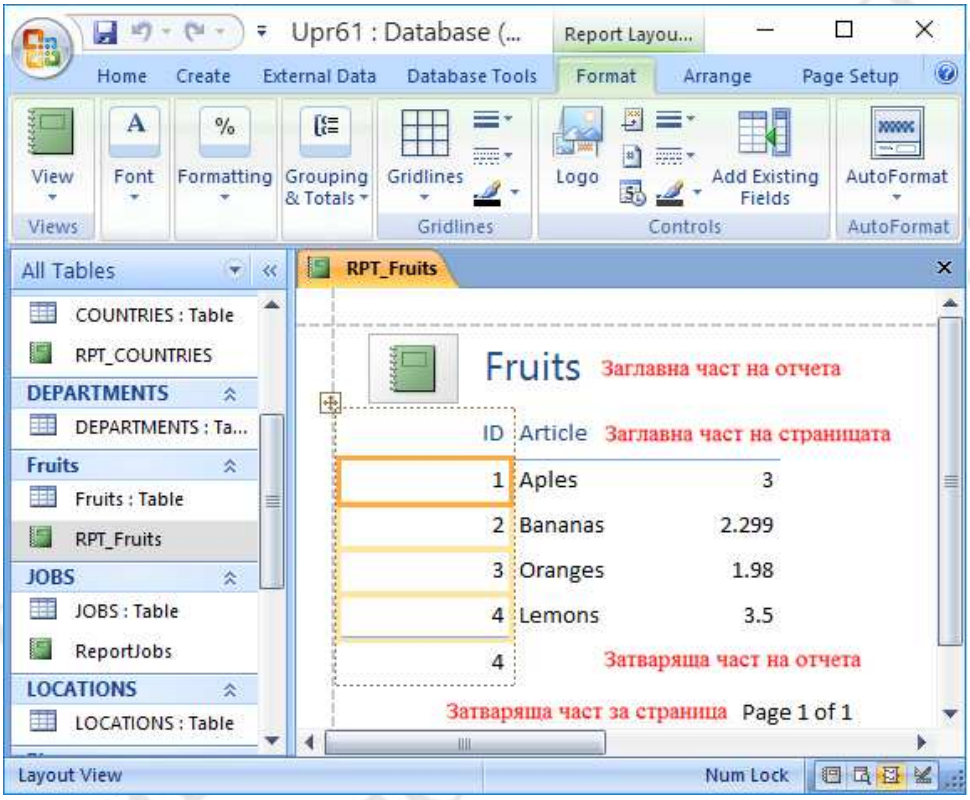
https://www.techwalla.com/articles/types-of-forms-in-microsoft-access Types of Forms in Microsoft Access

5.12. Отчети в MS Access

Отчетите са документи, които се генерират от СУБД за да се получи оперативна информация от съхраняваните данни. Чрез отчетите се представя обобщена информация, съдържаща се в други обекти в базата данни. Отчетите могат да бъдат разпечатани и предадени като документи на различни потребители. За да представят информацията в прегледен вид, е необходимо отчетите да бъдат подходящо структурирани.

5.12.1. Структура на отчетите

Отчетите се се състоят от секции, които могат да се променят в изгледа за проектиране Design View. Разбирането на начина на работа на всяка секция е полезно за да може да се създават по-добри отчети. Например секцията, в която разработчикът е поставил изчисляеми контроли, определя какъв резултат ще бъде получен при изчисленията. Резултатите от изчисленията могат да бъдат различни за различните секции.



Фиг. 40 Структура на отчет

Следващият списък съдържа описание на типовете секции и тяхната употреба.

Раздел	Позиция при отпечатване	Предназначение на секцията
Заглавна част на отчета (Report Header)	В началото на отчета.	В заглавната част на отчета се извежда информация, която обикновено се поставя на титулна страница, например емблема, заглавие или дата. Когато поставите изчисляема контрола, използваща агрегатната функция Sum в горния колонтитул на отчета, изчислената сума е за целия отчет. Заглавна част на отчета се

Раздел	Позиция при отпечатване	Предназначение на секцията
		отпечатва преди заглавната част на страница.
Заглавна част на страница (Page Header)	В началото на всяка страница.	Можете да използвате заглавната част на страницата за да повторите заглавието на отчета на всяка страница, когато отчета се извежда на няколко страници.
Заглавна част на група (Group Header)	В началото на всяка нова група записи.	Използвайте заглавната част на група за извеждане на името на групата. Например в отчет, който е групиран по продукти, използвайте горния колонтитул на група, за да отпечатате името на продукта. Когато поставите изчисляема контрола, която използва агрегатната функция Sum в заглавната част на група, сумата е за текущата група. В даден отчет може да имате няколко заглавни части на групи в зависимост от това, колко нива на групиране сте добавили. За повече информация относно създаването на горни и долни колонтитули на група вж. раздела "Добавяне на групиране, сортиране или общи суми".
Секция Details	Извежда се по веднъж за всеки запис от източника.	Това е мястото, където се поставят контролите, съставящи основния текст на отчета.
Затваряща част за група	В края на всяка група записи.	Използвайте затваряща част за група за да отпечатате обобщена информация за група. В даден отчет може да имате няколко долни колонтитули на група в зависимост от това, колко нива на групиране сте добавили.
Затваряща част за страница	В края на всяка страница.	Използвайте затваряща част за страница, за да отпечатате номерата на страниците или информация за всяка страница.

Раздел	Позиция при отпечатване	Предназначение на секцията
Затваряща част на отчета	В края на отчета.	Използвайте затваряща част на отчета за да отпечатате общи суми или друга обобщена информация за целия отчет.
Забележка: В изглед на проектиране (Design View) затваряща част на отчета се показва под затваряща част на страницата. Обаче във всички други изгледи (например изглед на оформление или когато отчетът се отпечатва или визуализира), затваряща част на отчета се показва *над* затваряща част на страницата, непосредствено след последната затваряща част на група или на реда с подробна информация на последната страница.		

5.12.2. Създаване на отчети в Access

Access предоставя три инструмента за създаване на отчети:

Отчет (Report) - Създава прост табличен отчет, съдържащ всички полета в източника на записи, който сте избрали в навигационния екран.

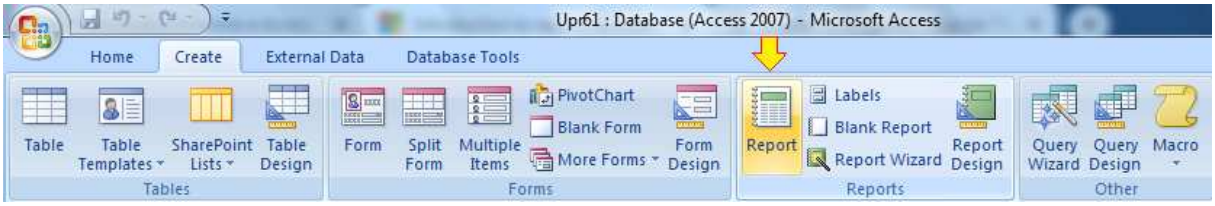
Съветник за отчети (Report Wizard) – Помощна програма за постъпково създаване на отчети.

Празен отчет (Blank Report) – създава празен отчет, за който разработчикът трябва да зададе източник на данни и да постави контроли в секциите на отчета.

Важно е да се отбележи, че отчетите на Access не са статични документи. При всяко отваряне на отчет за разглеждане и отпечатване (изглед Report View) той се попълва с текущото съдържание на източника на данни (таблица или заявка) на Access, а също с информация за текущата дата и време. Така че отчетите на Access са динамични документи, които се попълват със съдържание при отварянето им.

5.12.3. Създаване на табличен отчет

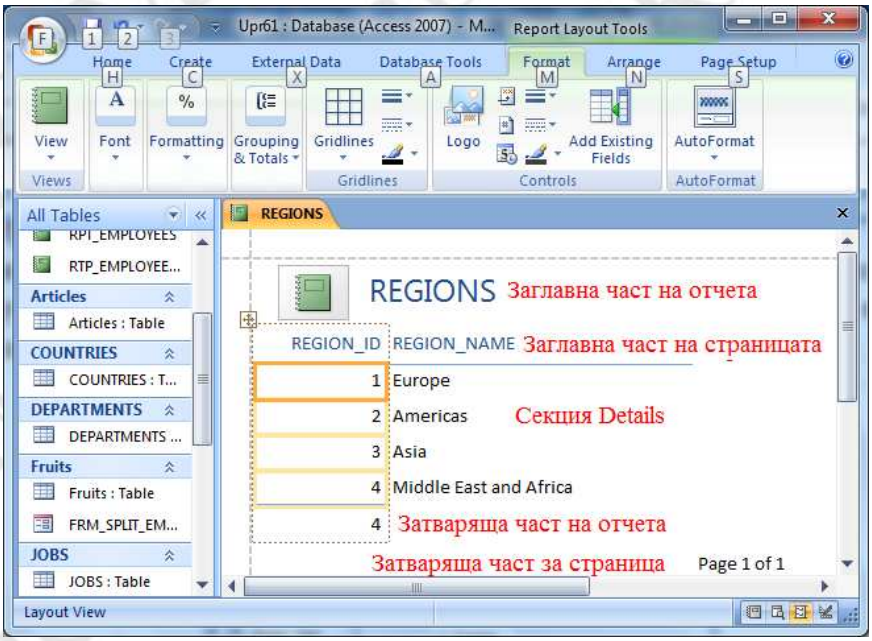
За създаване на табличен отчет разработчикът трябва да да избере от прозореца “All Tables” вдясно източника на данни за отчета (таблица или заявка) и да щракне на бутона „Report” от менюто Create (Създаване).



Фиг. 41 Създаване на табличен отчет

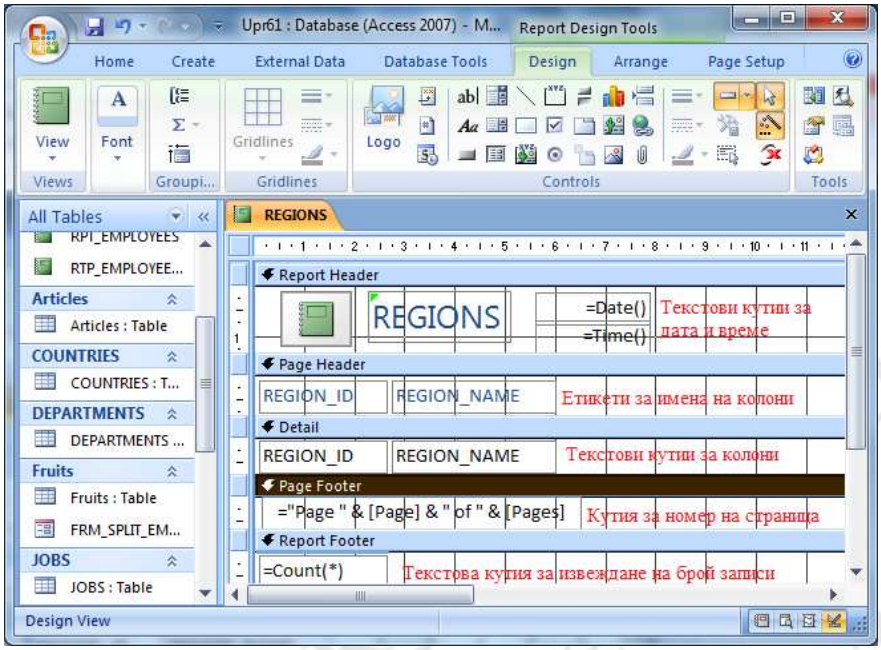
Създава се отчет, който извежда:

- Заглавна част на отчета ReportHeader с икона и името на таблицата
- Заглавна част на страницата PageHeaderSection с имена на колоните
- Секция Details, която съдържа таблица с извлечените от източника записи
- Затваряща част за страница PageFooterSection, която съдържа текстова кутия Text2, в която се извежда номера на страницата и общият брой страници на отчета.
- Затваряща част на отчета ReportFooterSection, която съдържа текстова кутия AccessTotalsXXXX (където XXXX е името на първата колона), в която се извежда общият брой записи.



Фиг. 42 Вид на табличен отчет в изглед за оформление

Отчетът, показан на фигурата, използва като източник на данни таблицата Regions. Той е в изглед за оформление (Layout View), и поради това затварящата част на отчета се извежда преди затварящата част на страницата. Ако превключим в изглед за проектиране (Design View), както е показано на следващата фигура, двете затварящи части ще си разменят местата.



Фиг. 43 Вид на табличен отчет в изглед за дизайн

В изгледа за дизайн, показан на фигурата по-горе се виждат имената на секциите на отчета и ActiveX контролите, включени е тях.

В заглавната част на отчета ReportHeader са включени контрол Image за извеждане на икона, контрол Label за извеждане на заглавен надпис и две текстови кутии TextBox, в които, чрез вградените функции на MS Access Date() и Time() се извеждат текущата дата и време.

В заглавната част на страницата PageHeader са включени контроли Label, в които се извеждат (като статичен текст) имената на колоните от източника (в примера имената на колоните на таблицата Regions).

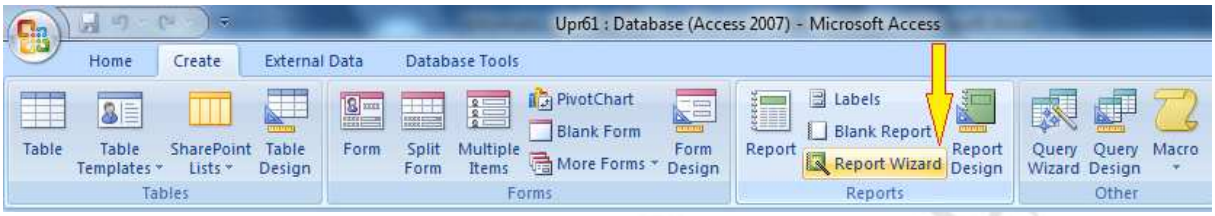
В секцията Details са включени текстови кутии, в които се извеждат съдържанията на полетата от един запис от източника. При отваряне на отчета за разглеждане и отпечатване в изглед Отчет (Report View) секцията Details се извежда поотделно за всеки запис от източника.

В затварящата част за страницата PageFooter се съдържа текстова кутия, в която, чрез свойствата Page и Pages на VBA обекта Report се извеждат номерът на текущата страница на отчета и общият брой страници.

В затварящата част за отчета ReportFooter се съдържа текстова кутия, в която, чрез обобщаваща (агрегатна) функция на SQL Count(*) се извежда броя на записите в отчета.

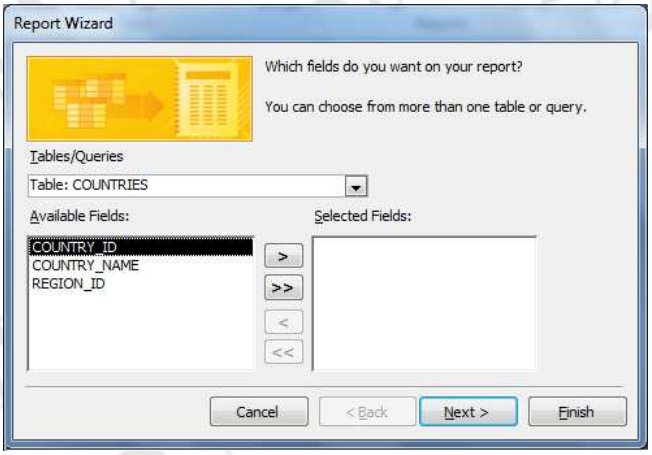
5.12.4. Създаване на отчет със съветника за отчети (Report Wizard)

За създаване на отчет със съветника за отчети (Report Wizard) разработчикът трябва да да избере от прозореца “All Tables” вдясно източника на данни за отчета (таблица или заявка) и да щракне на бутона „Report Wizard” от менюто Create (Създаване).



Фиг. 44 Създаване на отчет с инструмента Report Wizard

Помощната програма Report Wizard се изпълнява на няколко стъпки. По време на първата стъпка се отваря прозорец, от който се избират източника на данни и колоните, които ще се извеждат в отчета. По подразбиране за източник на данни се предлага таблицата или заявката, която е била селектирана преди стартиране на Report Wizard.

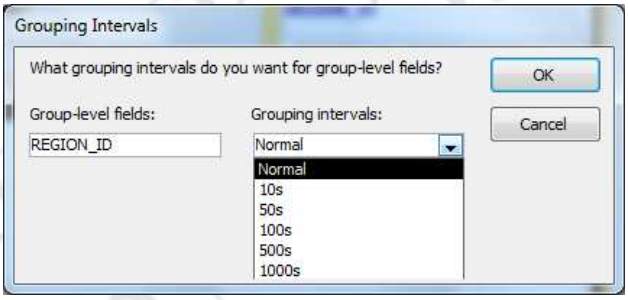


Фиг. 45 Избор на източник на данни и колони за Report Wizard

На втората стъпка могат да се изберат една или няколко колони, по които да се извършва групиране при извеждане на данните в отчета.

Фиг. 46 Избор на колони за групиране за Report Wizard

В примера е избрана колона за групиране REGION_ID. Чрез бутона „Gruping Options" Report Wizard дава възможност за избор на интервали от стойности, както е показано на следващата фигура. Това има смисъл да се прави тогава, когато в колоната, по която се извършва групиране има много на брой различни стойности и искаме да ги групираме „по на едро", т.е. да намалим броя на групите чрез обединяване на стойностите в интервали.



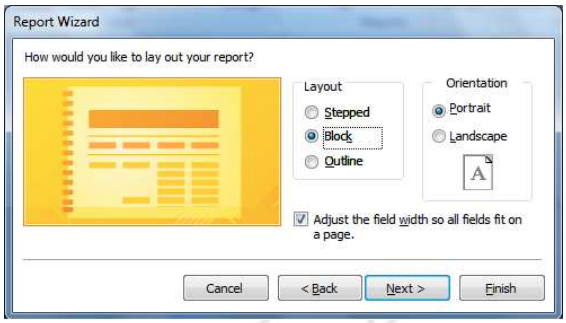
Фиг. 47 Избор на интервали за групиране за Report Wizard

На следващата стъпка може да се избере сортиране на записите в отчета

Фиг. 48 Сортиране на записите в отчета с Report Wizard

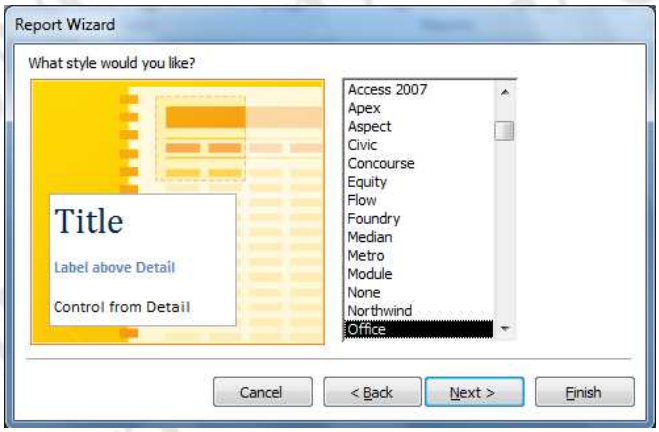
по стойностите в една или няколко колони в нарастващ (Acsending) или намаляващ (Desending) ред.

На следващата стъпка може да се избере едно от три предлагани подреждания: Stepped, Block и Outline. Разликата между тях е в разстоянията на отделните блокове, получени в резултат на групирането (ако има такова). Най-икономично се използва листа при подреждането Block.



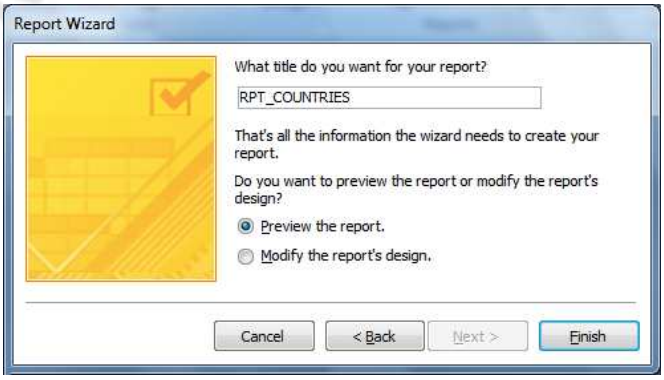
Фиг. 49 Избор на вид подреждане на групите в отчета

На следващата стъпка може да се избере стил на отчета от зададен списък със стилове. Стиловете предлагат различни шрифтове и цветове на текста на заглавния надпис, заглавията на колоните и рамките на контролите.



Фиг. 50 Избор на стил на отчета

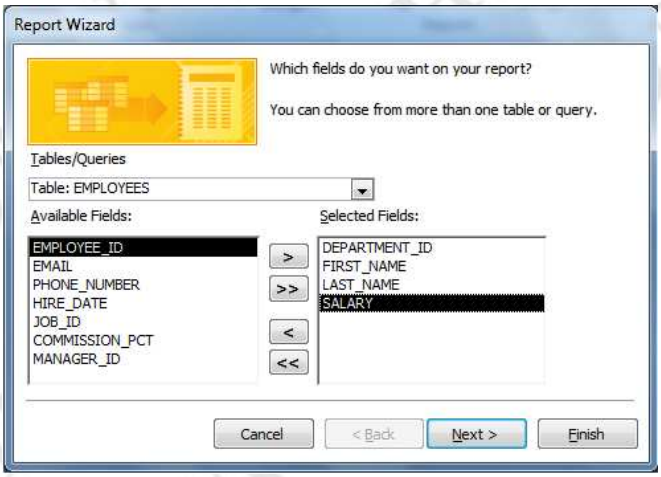
На следващата стъпка може да се избере име на отчета и да се отвори създаденият отчет за разглеждане.



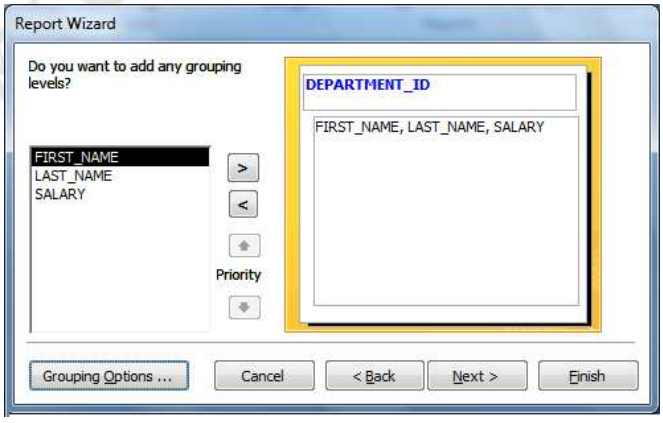
Фиг. 51 Последна стъпка - задаване на име на отчета

5.12.5. Създаване на отчет с обобщаващи (агрегатни) функции

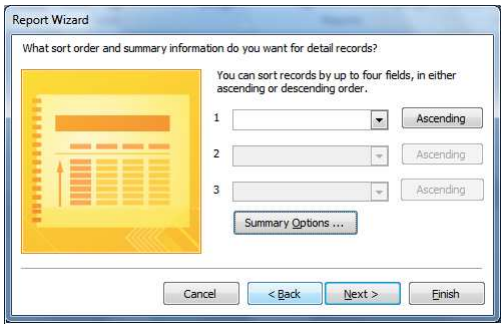
При създаване на отчет по данни от свързани таблици в много случаи е полезно записите да се групират по стойностите на колона, с която таблицата се свързва с друга (главна) таблица и да се изчислят обобщаващи стойности за групите от записи. Като пример ще създадем отчет за служителите от таблица Employees, групирани по номер на департамент Department_ID. За целта селектираме таблицата Departments в прозореца All Tables вляво и щракваме с мишката бърху бутона Report Wizard. На първата стъпка избираме и прехвърляме в десният списък колоните Department_ID, First_Name, Last_Name и Salary.



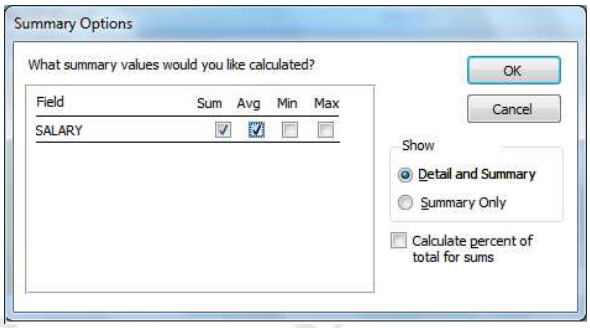
На следващата стъпка избираме от списъка вляво колоната Department_ID и я прехвърляме вдясно като колона за групиране с бутона „>”.



На следващата стъпка можем да изберем, но също така можем да добавим към отчета обобщаващи (агрегатни) функции.



Фиг. 52 Задаване на сортиране



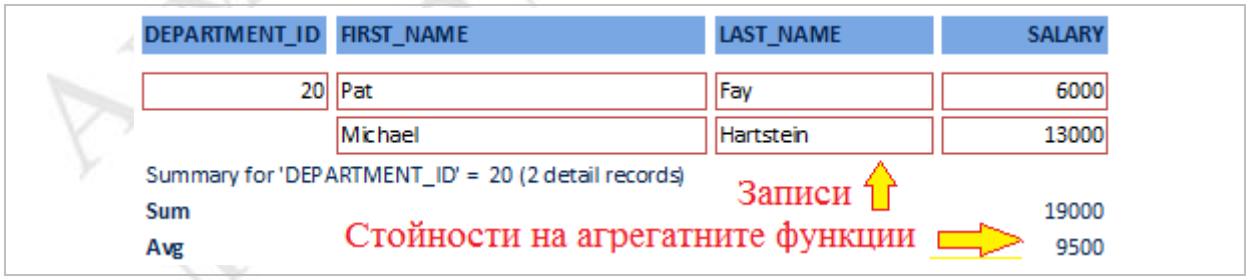
Фиг. 53 Избор на агрегатни функции

За целта се щраква на бутона „Sumary Options” от прозореца за избор на колони за сортиране на записите в отчета. Отваря се прозорец, от който можем да изберем агегатни функции за числовите колони в заявката. Освен това има две опции за отчета:

„Detail and Summary” за извеждане на записите (секция Detail) и отделно стойностите на агрегатните функции (Sumary).

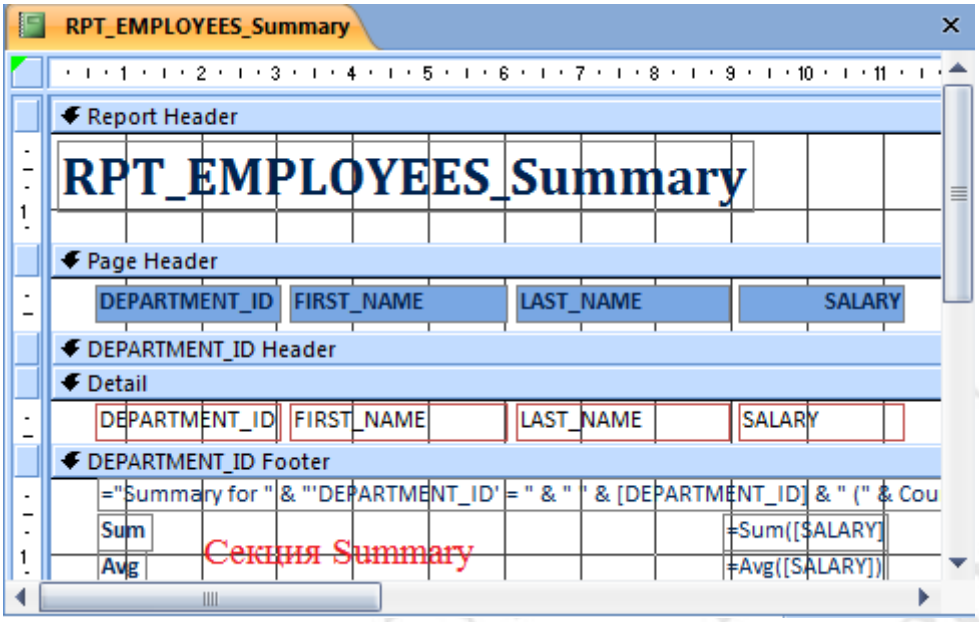
„Summary Only” за извеждане само на стойностите на агрегатните функции.

На следващите стъпки се избира вид подреждане и се задава име на заявката, както в предния пример. В полученият отчет записите са разделени на групи по стойността на избраната колона за групиране Depatrment_ID. Под всяка група се извеждат стойностите на агрегатните функции за групата.



Фиг. 54 Вид на група записи и група Summary в отчет с използване на агрегатни функции

Структурата на полученият отчет се вижда по-добре когато е в „Изглед за проектиране” (Design View).



Фиг. 55 Вид на отчет с използване на агрегатни функции в изглед Design View

Секцията Detail съдържа текстовите кутии, в които се извеждат стойностите на полетата от записите, които се извеждата в отчета. Следващата секция (в примера „Department_ID Footer”) е затварящата част на секцията, в която се извеждат данните от групиране. В нея вдясно са текстовите кутии, в които се извеждат сумата и средната стойност на Salary от записите в групата.

В изгледа Design View е най-удобно да се правят промени в текста и форматирането на контролите на отчета. Например може да се промени текста

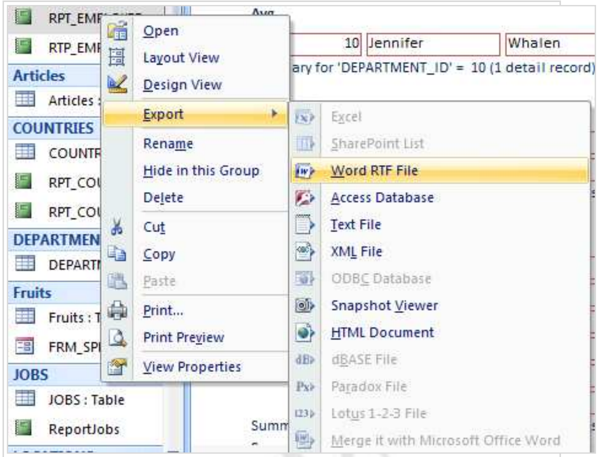
```
="Summary for " & "'DEPARTMENT_ID' = " & " " & [DEPARTMENT_ID] & " (" & Count(*) & " " & IIf(Count(*)=1;"detail record";"detail records") & ")"
```

в текстовата кутия в секцията „Department_ID Footer” с по-разбираем текст на кирилица. Заглавният надпис в секция Report Header също е добре да бъде променен.

5.12.6. Експортиране на отчет като статичен документ

Както беше посочено по-горе, отчетите са динамични документи, съдържанието на които се генерира от текущото съдържание на БД при отварянето им за разглеждане. Налага се обаче да се съхраняват отчети за да показват информация за определени дата и време на създаване. За тази цел може да се използва функцията за експортиране, която предоставя

Access. Чрез нея генериран в даден момент отчет може да се съхрани като статичен документ в .rtf формат, HTML формат или като обикновен текстов файл. За целта се щраква с мишката върху отчета в прозореца „All Tables” вляво и от плаващото меню се избира
Export→<вид на документа>



Фиг. 56 Експортиране на отчет

Онлайн източници:
http://fbm.uni-ruse.bg/d/itu/Uprajnenija/6%D0%BD%D0%B0%D1%81%D0%BE%D0%BA%D0%B8-%D0%B7%D0%B0-%D0%BE%D1%82%D1%87%D0%B5%D1%82%D0%B8.pdf Създаване на отчети в Access 2010
https://support.office.com/bg-bg/article/Въведение-вотчетите-6e640524-3536-4ccc-83ed-7142d028440f Въведение в отчетите в Access
https://support.microsoft.com/en-us/office/introduction-to-reports-in-access-e0869f59-7536-4d19-8e05-7158dcd3681c Introduction to reports in Access
https://www.google.com/search?q=MS+Access+How+to+add+group+footer&oq=MS+Access+How+to+add+group+footer&aqs=chrome..69i57j0i22i30l3.26287j0j15&sourceid=chrome&ie=UTF-8

5.13. Макроси в MS Access

Макросите в MS Access са програмни модули с код на Visual Basic for Applications (VBA), които дават възможност за автоматизиране на задачите и за добавяне на функционалност към формулярите, отчетите и контролите.

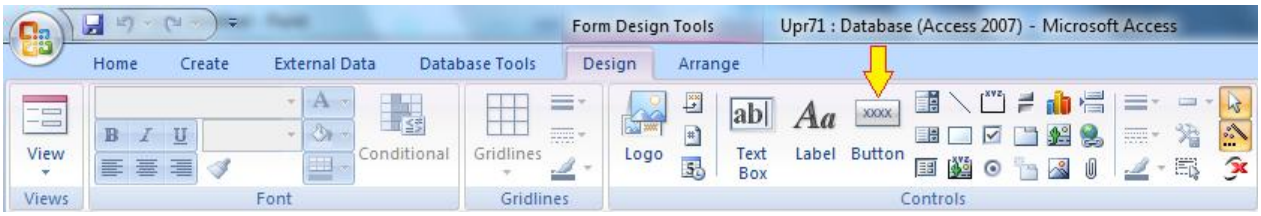
Макросите в Access работят малко по-различно от макросите в Word или Excel, където (чрез инструмента Macro Recorder) потребителят може да запише последователност от действия и ги възпроизвежда по-късно.

В MS Access макросите за достъп са изградени от набор от предварително дефинирани действия, което ви позволява да автоматизирате често срещани задачи и да добавите функционалност към контроли или обекти. Макросите могат да бъдат самостоятелни обекти, видими от навигационния екран, или вградени директно във формуляр или отчет. След като разработчикът създаде обекти на база данни, като таблици, формуляри и отчети, чрез макросите всички тези обекти могат да се свързват заедно за да създаде просто приложение за база данни, което всеки може да използва или дори да модифицира с относително малко обучение.

Макросите осигуряват начин за изпълнение на команди, без да е необходимо да пишете или дори да знаете VBA код, и има много неща, които бихте могли да постигнете само с макроси.

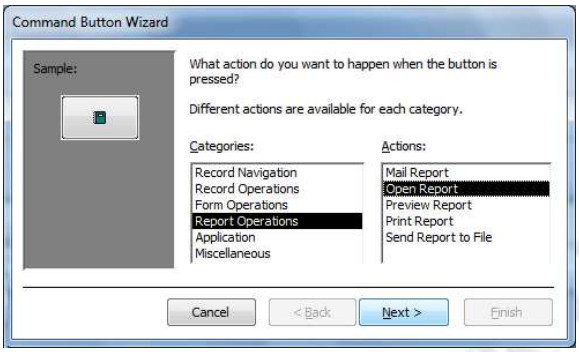
5.13.1. Създаване на макрос чрез инструмента Command Button Wizard

Инструментът Command Buton Wizard дава възможност за създаване на макрос във формуляр, който да се стартира чрез бутон и да изпълнява определено действие. За да се създаде макрос с този инструмент се отваря формулярът (в примера FRM_Regions) в изглед Design View , щраква се с мишката върху иконата за бутон от менюто Design (вижте фигурата по-долу) и с натиснат ляв бутон на мишката се изчертава правоъгълника



, в който да се изобрази бутона. Бутонът се изобразява като правоъгълник с текст, който е името на ActiveX контрола, който се добавя към формуляра. Текста върху бутона в последствие може да се промени като се запише нов текст като стойност на свойството Caption на контрола.

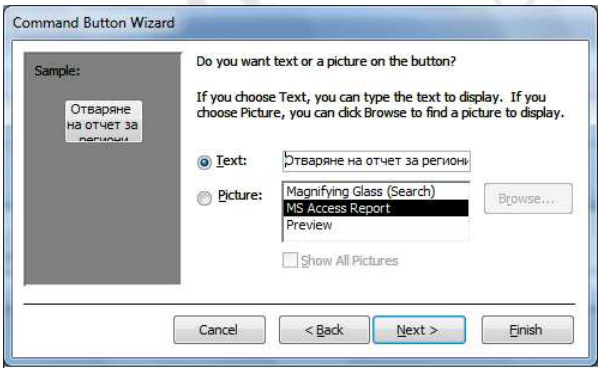
Когато се отпусне бутона на мишката се извежда прозореца за първата стъпка на помощната програма Command Button Wizard



На първата стъпка на помощната програма се избира от списъка вляво категорията на действието, а от списъка вдясно – вида на събитието от избраната категория. На фигурата е избрана категорията „Report Operations” (операции за отчети), а вдясно – операцията „Open Report” (отваряне на отчет).

След като щракнем на бутона “Next” се отваря следващият прозорец на Command Button Wizard, от който (за примера) се избира отчета, който ще се отваря с бутона, а в общият случай се избира обекта на БД, за който ще се изпълнява избраната операция.

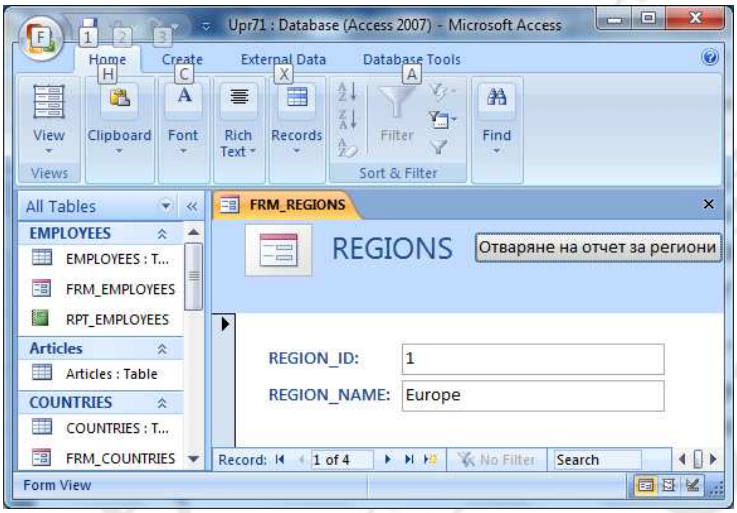
След като щракнем на бутона “Next” се отваря третият прозорец на



Command Button Wizard, от който се избира вида на бутона – с надпис (Text) или с изображение (Picture). Ако изберем „Text” (както е показано на фигурата) можем да въведем в текстовата кутия вдясно текста върху бутона.

Ако изберем „Picture”, можем да изберем изображение за бутона от списъка вдясно, или да изберем графичен файл с изображение чрез бутона Browse, който отваря диалогов прозорец за избор на файл.

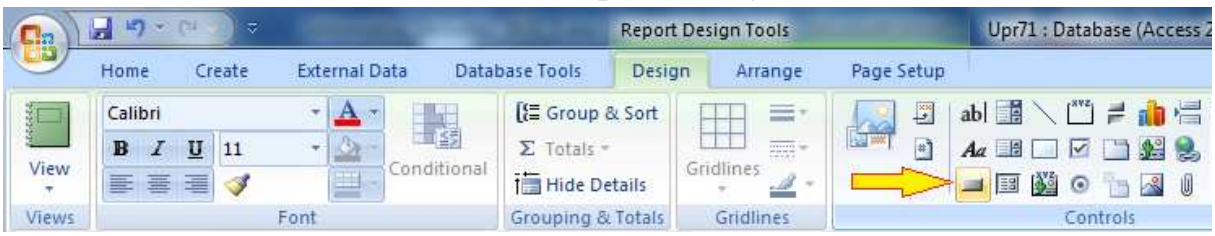
След като щракнем на бутона “Next” се отваря четвъртият, последен прозорец на Command Button Wizard, в който можем да въведем в текстовата кутия името на бутона като ActiveX обект и да завършим процедурата с щракване на бутона “Finish”. Ако отворим формуляра FRM_Regions за разглеждане и щракнем на добавеният бутон „Отваряне на отчет за региони” ще се отвори отчетът RPT_Regions.

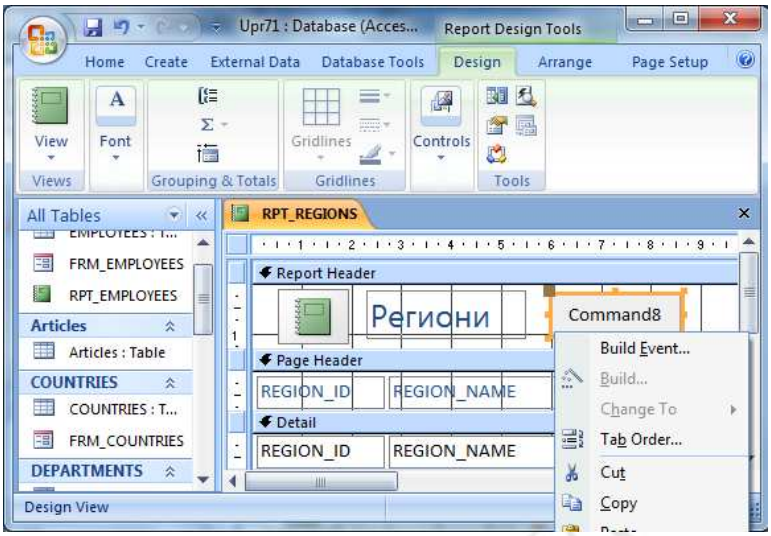


Фиг. 57 Вид на табличен формуляр с добавен бутон чрез инструмента Command Button Wizard

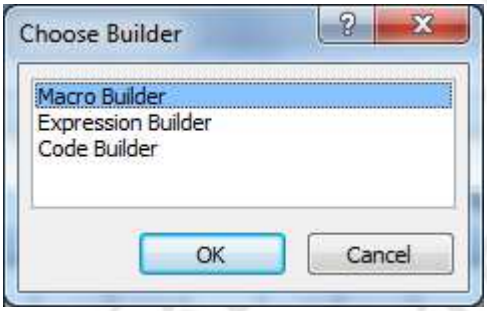
5.13.2. Създаване на макрос чрез инструмента Macro Builder

Инструментът Macro Builder дава възможност да се добави действие към бутон, включен в отчет. За да се създаде макрос с този инструмент се отваря отчетът (в примера RPT_Regions) в изглед Design View , щраква се с мишката върху иконата за бутон от менюто Design (вижте фигурата по-долу) и с натиснат ляв бутон на мишката се изчертава правоъгълника, в който да се изобрази бутона. Бутонът се изобразява като правогълник с текст, който е името на ActiveX контрола, който се добавя към отчета. Текста върху бутона в последствие може да се промени като се запише нов текст като стойност на свойството Caption на бутона.

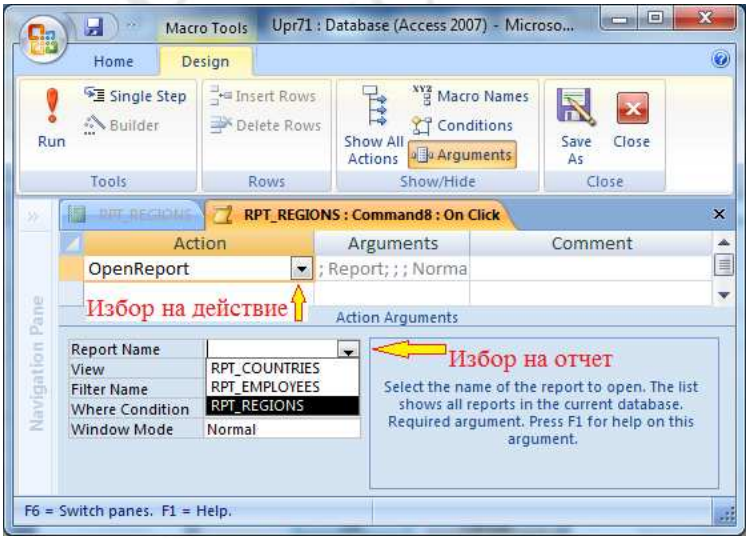




За да стартираме инструмента Macro Builder щракваме с десният бутон на мишката върху бутона (както е показано на фигурата вляво) и от плаващото меню избираме Build Event. Отваря се диалогов прозорец, от който



се избира Macro Builder. Отваря се прозорец, от който се избира действие за бутона. На фигурата по-долу е избрано действието „Open Report", а от списъчният обект за Report Name е избрано



Фиг. 58 Избор на действие за бутон

името на отчета (RPT_Countries) който да се отвори чрез бутона. След затваряне на прозореца с бутона "Close" действието вече е свързано с бутона и, когато се отвори отчета RPT_Regions за разглеждане и се щракне на добавеният бутон, ще се отвори отчета RPT_Countries.

5.14. Visual Basic for Applications

Visual Basic for Applications (VBA) е програмен език, създаден от Microsoft, който се използва в много продукти на Microsoft като MS. Word, MS. Excel, MS. Access, MS. Outlook и др. VBA е създаден с цел да помогне на потребителите на Microsoft да разширяват функционалността на приложенията от Microsoft Office чрез писане на код и така да управляват данните по по-ефикасен и ефективен начин.

В настоящият курс ще разгледаме елементите на езикът за програмиране VBA и ще дадем примери за неговото използване в MS Access.

5.15. Основни елементи на VBA

Създаването на програми на кой да е език за програмиране изисква познаване и правилно използване на елементите на езика – символи, ключови думи, константи, променливи, функции, обекти, коментари. Изучаването на тези програмни елементи, тяхното дефиниране и използване, правилата (синтаксиса) за записване на командите и структурата на програмите са начален етап за изучаване на всеки език за програмиране. Ще разгледаме последователно тези елементи и изисквания. Основните елементи на VBA включват типове данни, константи, променливи, конструкции за условно разклонение, конструкции за цикли, класове (типове) и обекти.

5.16. Типове данни

В компютърните науки и компютърното програмиране типът данни или просто *типът* е атрибут на данните, който указва на компилатора или интерпретатора как програмистът възнамерява да използва данните. Повечето езици за програмиране поддържат основни типове данни за целочислени числа (с различни размери), числа с плаваща запетая (които приближават реалните числа), символи и булеви (логически) стойности. Типът данни ограничава стойностите, които променливите или функциите, могат да приемат. Този тип данни дефинира операциите, които могат да бъдат извършени с данните, значението на данните и начина, по който стойностите на този тип могат да се съхраняват.

Най-често използваните типове данни във VBA са:

Byte:	Съхранява положителни стойности от 0 до 255. Този тип

	използва 1 байт от паметта.
Integer:	Съхранява както отрицателни, така и положителни недесетични стойности, вариращи от -32,768 до 32,767. Този тип ще използва 2 байта размер на паметта.
Long:	Съхранява както отрицателни, така и положителни недесетични стойности, вариращи от -2,147,483,648 до 2,147,483,647. Той използва 4 байта размер на паметта.
Single	Съхранява както отрицателни, така и положителни десетични стойности, вариращи от -3.402823E38 до -1.401298E-45 за отрицателни стойности и 1.401298E-45 до 3.402823E38 за положителни стойности. Използва 4 байта размер на паметта.
Double:	Съхранява както отрицателни, така и положителни десетични стойности, вариращи от -1.79769313486231E308 до -4.94065645841247E-324 за отрицателни стойности и 4.94065645841247E-324 до 1.79769313486232E308 за положителни стойности. Той използва 8 байта размер на паметта.
Date:	Съхранява стойности за дата / час. Той използва 8 байта от паметта.
Boolean:	Съхранява логически стойности - вярно или невярно. Този тип консумира 2 байта размер на паметта.
Currency:	Съхранява валутни стойности, вариращи от -922,337,203,685,477.5808 до 922,337,203,685,477.5807. Той използва 8 байта от паметта.
Variant:	Съхранява всякакви стойности. Той използва 16 байта или повече от този на размера на паметта.
String:	Съхранява текстови стойности. Използването му на памет зависи от дължината на текста за съхранение.

5.16.1. Константи

Всяка програма съдържа постоянна информация, която не се променя по време на нейното изпълнение. Такъв тип информация се представя посредством константи.

D1. Константата е величина, която не може да променя стойността си по време на изпълнение на програмата.и се определя от елементите {тип, стойност}.

В VBA константите се включват в самите команди. Ето един пример за константа, включена в команда:

```
x= 3.1416
```

С тази команда в променливата x се записва числото 3.1416, което, както сте се досетили е константата π, зададена с точност до четвъртия знак след десетичната точка. Независимо от това колко пъти ще се изпълни тази команда в програмата, резултатът винаги ще бъде един и същ, тъй като вдясно от оператора за присвояване "=" стои константата 3.1416.

Използване на подобни константи многократно в текста на програмите се счита за лош стил на програмиране. По-добре е константите, които имат някакво специално значение за програмата да се декларират.

Деклариране на константи – команда Const

Командата за деклариране на константи Const има следният синтаксис

[Public | Private] Const име_на _константа [As type] = израз

където:

Public	Optional. Ключова дума, използвана на ниво модул, за деклариране на константи, които са достъпни за всички процедури във всички модули. Не се допуска в процедурите.
Private	Optional. Ключова дума, използвана на ниво модул за деклариране на константи, които са достъпни само в модула, където е направена декларацията. Не се допуска в процедурите.
име_на _константа	Required. Името на константата; следва стандартните конвенции за именуване на променливи.
type	Optional. Тип данни на константата; може да бъде Byte, Boolean, Integer, Long, Currency, Single, Double, Decimal (not currently supported), Date, String, or Variant. Use a separate As type clause for each constant being declared.
израз (expression)	Required. Литерал, друга константа или всяка комбинация, която включва всички аритметични или логически

	оператори с изключение на Is.

По подразбиране константите са Private (частни). В рамките на процедурите константите винаги са частни; видимостта им не може да бъде променена. В стандартните модули видимостта по подразбиране на константите на ниво модул може да бъде променена с помощта на ключовата дума Public. В модулите на класа обаче константите могат да бъдат само частни и видимостта им не може да се променя с помощта на ключовата дума Public.

Не може да се използват променливи, дефинирани от потребителя функции или вградени функции на VBA (като Chr) в изрази, присвоени на константи.

Константите, декларирани в процедура Sub, Function или Property са локални за тази процедура. Константа, декларирана извън процедура, е дефинирана в целия модул, в който е декларирана. Можете да използвате константи навсякъде, където можете да използвате израз.

Примери:
Const MyVar = 459 ' частна (Private) по подразбиране
Public Const MyString = "HELP" ' публична, достъпна във всички модули

5.16.2. Променливи

Освен константи всеки език за програмиране включва променливи, които представят информацията, която се променя по време на изпълнение на програмата.

D2 Променливата е величина, която може да променя стойността си по време на изпълнение на програмата и се определя от елементите {име, тип, стойност}.

Стойността, съхранявана в променливата се използва чрез нейното име. За всяка променлива при изпълнение на програмата се заделя място в оперативната памет на компютъра. При избор на име на променлива трябва да се спазват следните правила:

Имената на променливите не трябва да започват с цифра и могат да съдържат букви, цифри и символи за подчертаване “_” и “$”.

Имената на променливите трябва да са уникални (различни) за всички променливи, декларирани като глобални (извън функции) или вътре в дадена функция. VBA, за разлика от С-подобните езици, не прави разлика

между малките и големи букви в имената на променливите. Например за VBA x и X са една и съща променлива, докато например за C++ и JavaScript това са две различни променливи.

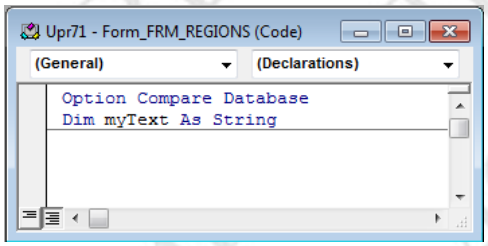
5.16.2.1. Деклариране на променливи на локално ниво

На локално ниво променливите се декларират в процедура между Sub и End Sub или във функция между Function и End Function. Чрез деклариране на променливи на локално ниво, обхватът на променливите е ограничен само до процедурата, в която е декларирана. Това означава, че променливата не може да бъде използвана или достъпна извън процедурата или функцията.
Синтаксис:

Dim *име_на_променлива* As *тип_на_данните*

5.16.2.2. Деклариране на променливи на глобално ниво

На глобално ниво променливите се декларират в глобалната секция на формуляр или в глобален модул. Ако се декларира променлива в глобална секция на формуляр (както е показано на фигурата вляво), тя става достъпна за всички процедури и функции (Sub и Function) във формуляра.

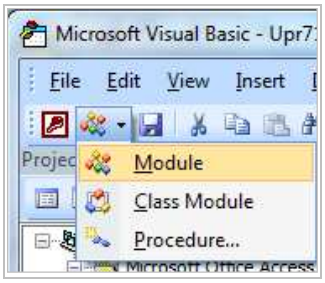


Фиг. 59 Деклариране на променлива в глобална секция на формуляр

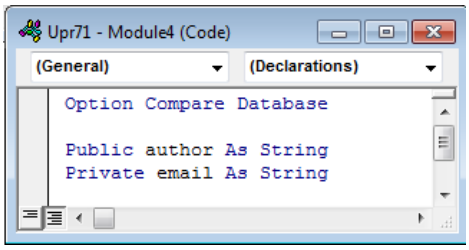
Ако се декларира променлива в глобален модул, тя трябва да бъде декларирана с ключова дума Private (за да е достъпна само в модула) или с ключова дума Public за да е достъпна навсякъде в приложението (т.е. навсякъде в кода, който се съдържа във файла за БД).

5.16.2.3. Добавяне на глобален модул към приложението на Access

За добавяне на глобален модул към приложението на Access е необходи-



мо да се отвори прозорецът на VBA и да се щракне на стрелката надолу на бутона "Insert Module" и да се избере от падащото меню Module. Отваря се прозорец за въвеждане на кода на модула. Променливите, процедурите и функциите, които се декларира в глобалните модули с ключова дума Public са достъпни в цялото приложение на Access.

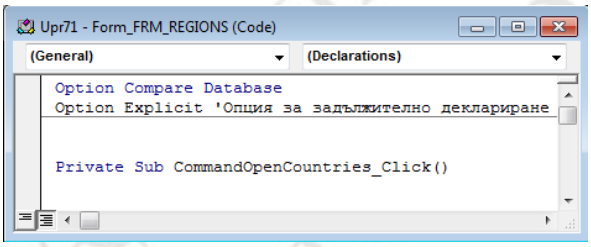


Фиг. 60 Деклариране на променливи в глобален модул

В примера вляво в глобалният модул Module4 са декларирани с ключови думи Public и Private една глобална и една локална променлива.

5.16.2.4. Option Explicit – опция за задължително деклариране

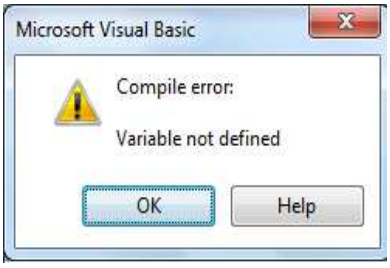
По подразбиране декларирането на променливи във VBA не е задължително, но става задължително, ако в глобалната секция на формуляр или на глобален модул се постави инструкцията Option Explicit.



Фиг. 61 Инструкция Option Explicit на ниво формуляр

На фигурата вляво е показано включването на Option Explicit в глобалната секция на формуляр, в резултат на което в кода на формуляра става задължително декларирането на променливи на глобално и локално ниво

Като резултата ако в модула или в някоя процедура или функция има недекларирана променлива, при изпълнението на кода се извежда прозорец



със съобщение "Variable not defined" и в дизайнера на VBA се отваря за редактиране кода на модула.

Използването на Option Explicit се препоръчва, тъй като така се избягва дублирането на имена на променливи на глобално и локално ниво,

което (ако не е зададена Option Explicit) може да доведе до трудно откриваеми грешки.

5.16.3. Операции, оператори и операнди.

Всяка компютърна програма в съответствие с предназначението си извършва зададени операции над стойностите на променливи и константи. Основните елементи на една операция са операторите и операндите.

D4 Операторите във VBA са символи за обозначаване на операции. Константите и променливите[2], които участват при изпълнението на операциите се наричат операнди.

Например при изпълнение на операцията събиране x + y операнди са променливите x и y, а операторът е „+”.

Важно е да се запомни, че при всяка операция се получава резултат, типът на който се определя от вида на оператора и типа на операндите.

● Например при операцията събиране се получава числова стойност, но ако оперантите са цели числа (тип Integer) резултата също е от тип Integer. Ако операндите са реални числа (тип Single) резултата също е от тип Single.

Таблица на операторите

Дадената по-долу таблица съдържа обозначението на оператора, типът на операндите в реда на записването им, кратко описание на операцията и приоритетът на операцията при изчисляване на стойността на изрази.

Оператор	**Тип на операндите**	**Операция**	**Приоритет**
,	всеки тип	Изчислява стойността на два операнда. Връща стойността на втория операнд.	1 (най-нисък)
=	променлива, всеки тип	Присвояване на стойност	2
? :	логически, всеки тип	Условен оператор (тернарен)	3
OR	логически	Логическо събиране (ИЛИ)	4
AND	логически	Логическо умножение (И)	5

[2] Като операнди в изразите могат да се използват масиви и функции

&	символни	Слепване на два символни низа	6
=	всеки тип	Проверка за равенство/идентичност	7
<, <=, >, >=, <>	Число или символен низ	Оператори за сравнение	8
+, -	Число	Събиране/Изваждане	9
*, /, Mod, ^	число	Умножение/Делене/Модул/Степен	10
new	Конструктор	Създаване на нов обект (унарен)	11
( )	Функция, списък от аргументи	Извикване на функция	
.	Обект, атрибут/метод	Достъп до атрибут или метод на обект	

5.16.4. Конструкции за управление.

Всяка една програма, в съответствие с алгоритъма си, съдържа последователност от инструкции и конструкции за управление. Конструкциите за управление реализират условните разклонения в алгоритъма на програмата. От това следва, че всички разклонени програми съдържат конструкции за управление. VBA предоставя на програмиста две конструкции, които осъществяват условно разклонение – **If-Then-Else** и **Select Case** и три специализирани конструкции за организиране на цикли – **For-Loop**, **While-Loop** и **Do-While**.

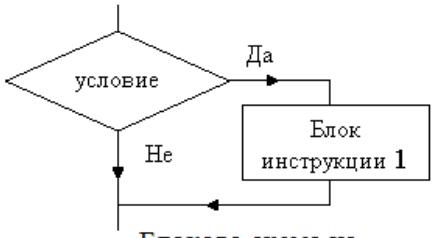
5.16.5. Конструкция за условно разклонение If-Then-Else

Тази конструкция се използва тогава, когато трябва да се изпълни една инструкция или блок от инструкции само в случай, че се изпълнява зададено условие. Клаузата Else дава възможност да се изпълни друга инструкция или блок от инструкции в случай, че не се изпълнява зададеното условие. Конструкцията има следния синтаксис:
Синтаксис:
Едноредов:

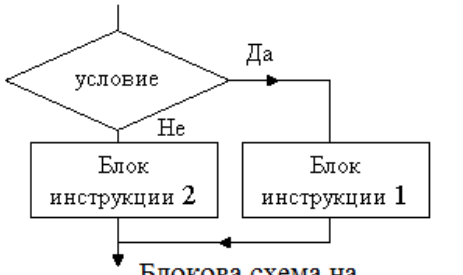
If условие Then [блок_команди_1] [Else блок_команди_2]

Блоков:

```
If condition Then
  [ команди ]
[ ElseIf condition-n Then
  [ elseif_команди ]]
[ Else
  [ else_команди ]]
End If
```



Фиг. 2 Блокова схема на конструкция IF-Then



Фиг. 3 Блокова схема на конструкция IF-Then-Else

Когато при изчисляване на условието се получи логическа истина се изпълнява „блок команди 1", а в противен случай, ако е включена клауза Else, се изпълнява „блок команди 2"

Да разгледаме един пример:

```
If Len(Region_Name.Text) > 4 Then 'проверка дали текста е над 4 символа
   Region_Name.BackColor = vbYellow 'ако да, задава жълт цвят на фона
   Else
   Region_Name.BackColor = vbGreen   'ако не, задава зелен цвят
End If
```

Фиг. 62 Конструкция IF-Then-Else за задаване на цвят на текстова кутия в зависимост от дължината на текста

В примера от свойството Text чрез вградената функция Len се получава дължината на текста в текстовата кутия Region_Name и ако дължината на текста е по-голяма от 4 , чрез свойството BackColor се задава жълт фонов цвят на текстовата кутия, а в противен случай – зелен цвят. За задаване на цветовете се използват вградените константи за цветове vbYellow и vbGreen.

5.17. Събитийни процедури в Access

Събитийните процедури в Access са VBA Sub процедури, които се извикват за изпълнение при възникване на определено събитие в обект на Access. Събитийните процедури съдържат блок от VBA команди и могат да бъдат създадени като макрос с инструмента Macro Builder или чрез

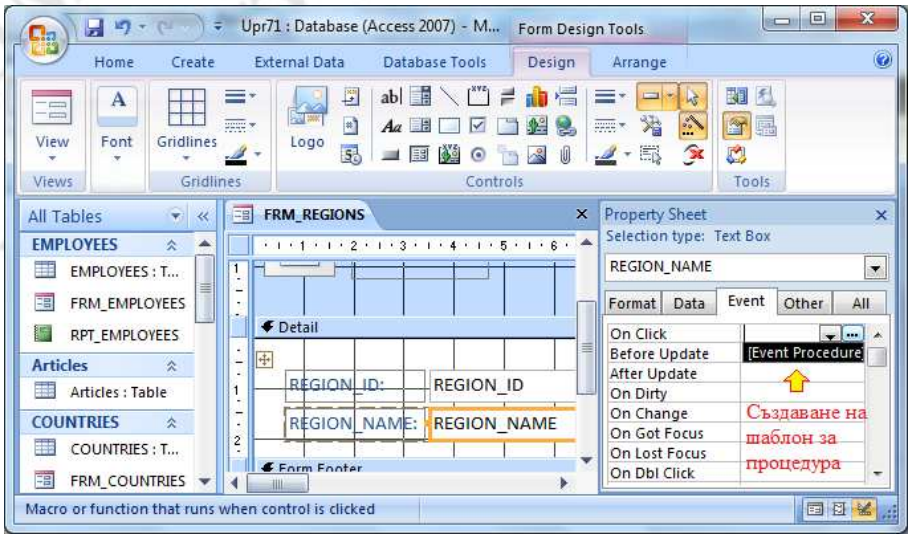
директно въвеждане на командите в прозорец на Code Builder. Всяка събитийна процедура се привързва към определено събитие на обект от приложението на Access. Има определено правило (конвенция) за имената на събитийните процедури:

Private Sub *име-на-обект_име-на-събитие*([*параметри*])

Например, процедура на събитие с име cmdClear_Click() ще бъде изпълнена при възникване на събитие Click за обект cmdClear, а това ще се случи тогава, когато потребителят щракне върху контрол (напр. бутон) с име cmdClear. Някои събитийни процедури имат параметри, други не. Например събитийната процедура KeyDown (натискане на бутон от клавиатурата) има параметри KeyCode (код на натиснат бутон) и Shift (число, което показва дали е натиснат бутон Shift или Control), докато събитийната процедура Click (щракване с ляв бутон на мишката върху обект) няма параметри.

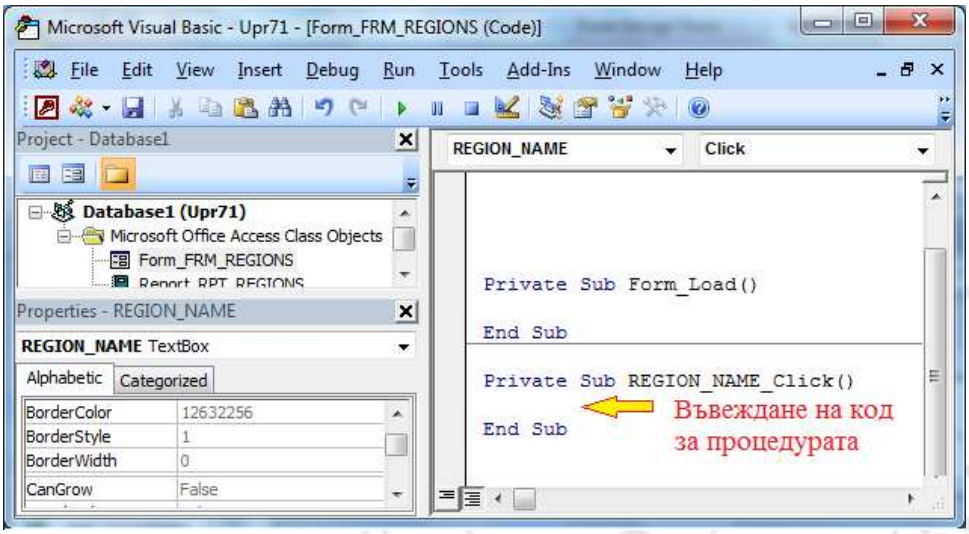
5.17.1. Създаване на събитийната процедура чрез въвеждане на код

За създаване на събитийната процедура, която да се изпълнява при възникване на определено събитие за обект от приложение на Access (контрол, формуляр, отчет) се селектира обекта и в страницата Event от прозореца Property Sheet се щраква върху на избраното събитие (например Click). От бутона ▾ се избира Event Procedure и се щраква върху бутона ... .



Фиг. 63 Създаване на шаблон за събитийна процедура чрез страница Event на прозореца Property Sheet

Отваря се прозереца на Visual Basic и в него – генерираният от Access шаблон за събитийната процедура



Фиг. 64 Прозорец на Visual Basic с шаблон за събитийна процедура

След като се въведе кода на процедурата, от менюто File се избира „Close and Return to Microsoft Access" (затваряне и връщане в приложението на Access). Ако кодът не съдържа грешки (по-точно ако не генерира грешка по време на изпълнение), при възникване на събитието той ще се изпълни. В противен случай пви възникване на събитието ще се изведе диалогов прозорец със съобщение за грешка и ще се отвори прозорецът на Visual Basic за да бъде коригиран кодът.

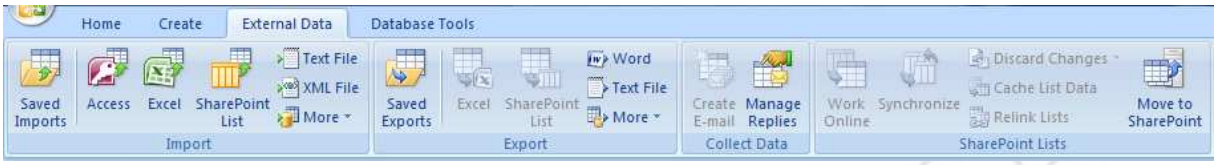
Като пример можем да използваме в събитийната процедура кода от примера за конструкция IF-THEN-Else:

```
Private Sub Region_Name_Click()
If Len(Region_Name.Text) > 4 Then 'проверка дали текста е над 4 символа
   Region_Name.BackColor = vbYellow 'ако да, задава жълт цвят на фона
   Else
   Region_Name.BackColor = vbGreen   'ако не, задава зелен цвят
End If
End Sub
```

Фиг. 65 Код на събитийна процедура за събитие Click на текстова кутия Region_Name

5.18. Използване на данни от външни източници в MS Access

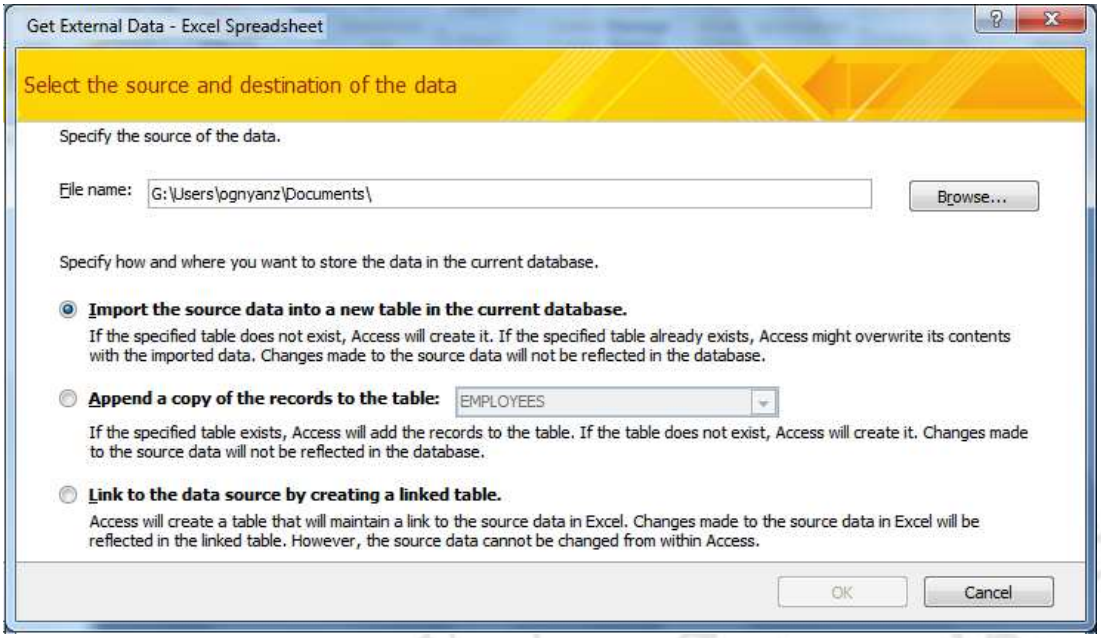
MS Access дава възможност за внасяне в БД на данни от различни източници – бази от данни, електронни таблици и файлове с различни формати за съхраняване на данни. Тези функции са включени в менюто „External Data”.



5.18.1. Внасяне на данни от електронни таблици

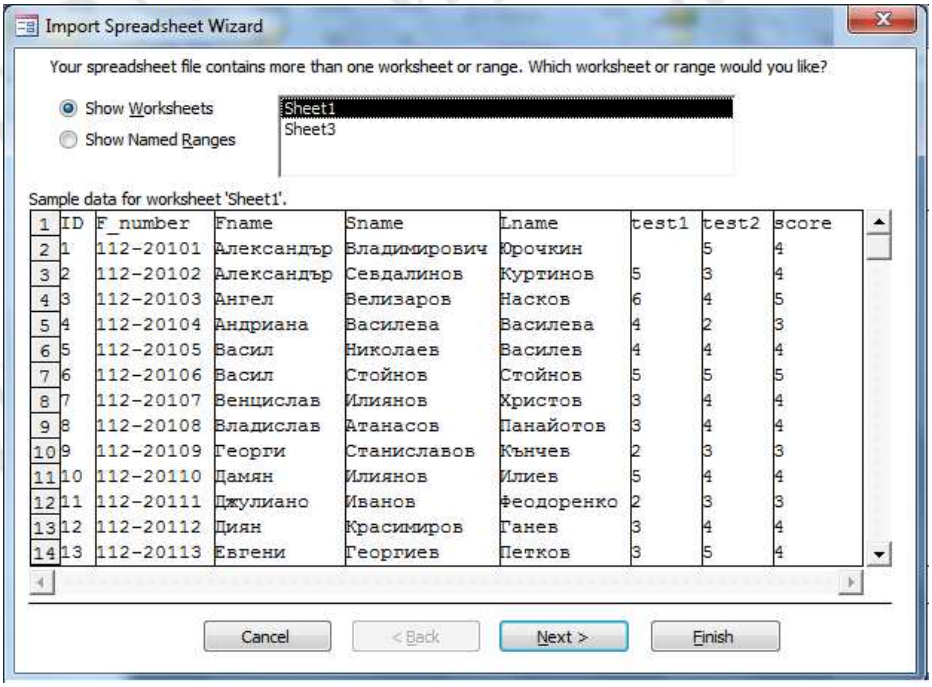
MS Access позволява внасяне (импортиране) на данни от документ на Excel, като дава възможност да се определи дали да се вмъкне само част от таблица или целият файл. Като пример ще разгледаме импортирането на страница (Sheet) от документ на Excel като таблица в БД на Access. За целта отваряме менюто „External Data” и избираме Excel. Отваря се прозорец, от който, след като се щракне на бутона Browse, се избира .xlsx файла с документа на Excel, от който ще се импортират данни. Предоставят се три опции:

- Import the source data into a new table in current database – импортиране на данните в нова таблица в текущата база от данни. Ако таблица с това име не съществува, тя се създава, а ако съществува – се надписва (заменя) нейното съдържание с импортираните данни.
- Append a copy of the records to the table – добавяне на записи от източника към съществуваща таблица. Ако таблица с това име не съществува, тя се създава.
- Link to the data source by creating a linked table – създаване на връзка към източника на данни чрез създаване на свързана таблица. В този случай при промяна на данните в таблицата на Excel промените се отразяват в свързаната таблица.



Фиг. 66 Избиране на източник и приемник на данни при импортиране

След като се избере файла с документа на Excel и се щракне на бутона OK се отваря прозорец, от който се избира листа от документа на Excel, от който ще се импортират данни и данните в него

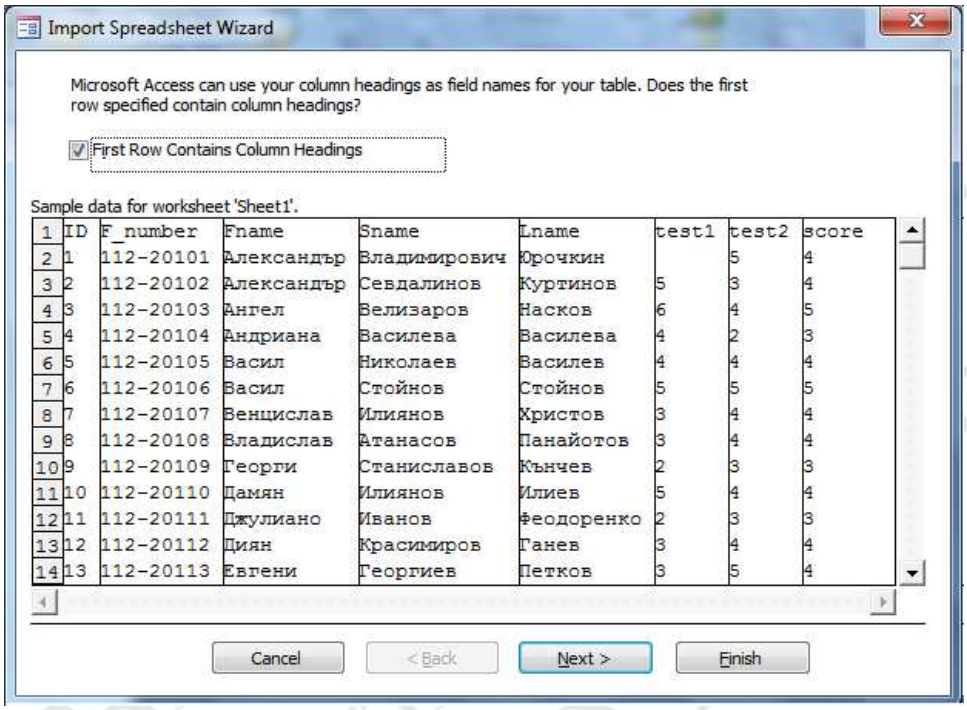


	ID	F_number	Fname	Sname	Lname	test1	test2	score
1	1	112-20101	Александър	Владимирович	Урочкин		5	4
2	2	112-20102	Александър	Севдалинов	Куртинов	5	3	4
3	3	112-20103	Ангел	Белизаров	Насков	6	4	5
4	4	112-20104	Андриана	Василева	Василева	4	2	3
5	5	112-20105	Васил	Николаев	Василев	4	4	4
6	6	112-20106	Васил	Стойнов	Стойнов	5	5	5
7	7	112-20107	Венцислав	Илиянов	Христов	3	4	4
8	8	112-20108	Владислав	Атанасов	Панайотов	3	4	4
9	9	112-20109	Георги	Станиславов	Кънчев	2	3	3
10	10	112-20110	Дамян	Илиянов	Илиев	5	4	4
11	11	112-20111	Джулиано	Иванов	Феодоренко	2	3	3
12	12	112-20112	Диян	Красимиров	Ганев	3	4	4
13	13	112-20113	Евгени	Георгиев	Петков	3	5	4

Фиг. 67 Избиране на лист от Excel с данни за импортиране

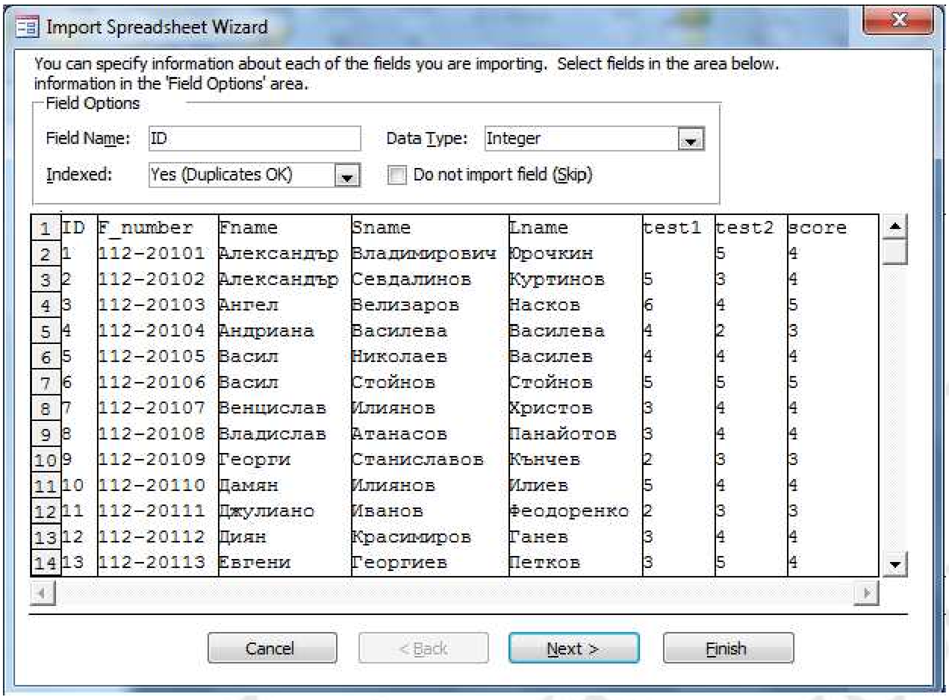
След като се избере листа от Excel с данни за импортиране и се щракне на бутона OK се отваря прозорец, в който се указва дали в първият ред от таблицата в листа съдържа имената на колоните. Ако е маркирана опцията „First Row Contains Column Headings”, то имената на колоните на таблицата, която ще се добави към БД на Access се получават от първият

ред в листа наExcel. В противен случай при импортирането се приема, че първият ред също съдържа данни и Access генерира автоматично имена на колоните.



	ID	F_number	Fname	Sname	Lname	test1	test2	score
1	1	112-20101	Александър	Владимирович	Юрочкин		5	4
2	2	112-20102	Александър	Севдалинов	Куртинов	5	3	4
3	3	112-20103	Ангел	Велизаров	Насков	6	4	5
4	4	112-20104	Андриана	Василева	Василева	4	2	3
5	5	112-20105	Васил	Николаев	Василев	4	4	4
6	6	112-20106	Васил	Стойнов	Стойнов	5	5	5
7	7	112-20107	Венцислав	Илиянов	Христов	3	4	4
8	8	112-20108	Владислав	Атанасов	Панайотов	3	4	4
9	9	112-20109	Георги	Станиславов	Кънчев	2	3	3
10	10	112-20110	Дамян	Илиянов	Илиев	5	4	4
11	11	112-20111	Джулиано	Иванов	Феодоренко	2	3	3
12	12	112-20112	Диян	Красимиров	Ганев	3	4	4
13	13	112-20113	Евгени	Георгиев	Петков	3	5	4

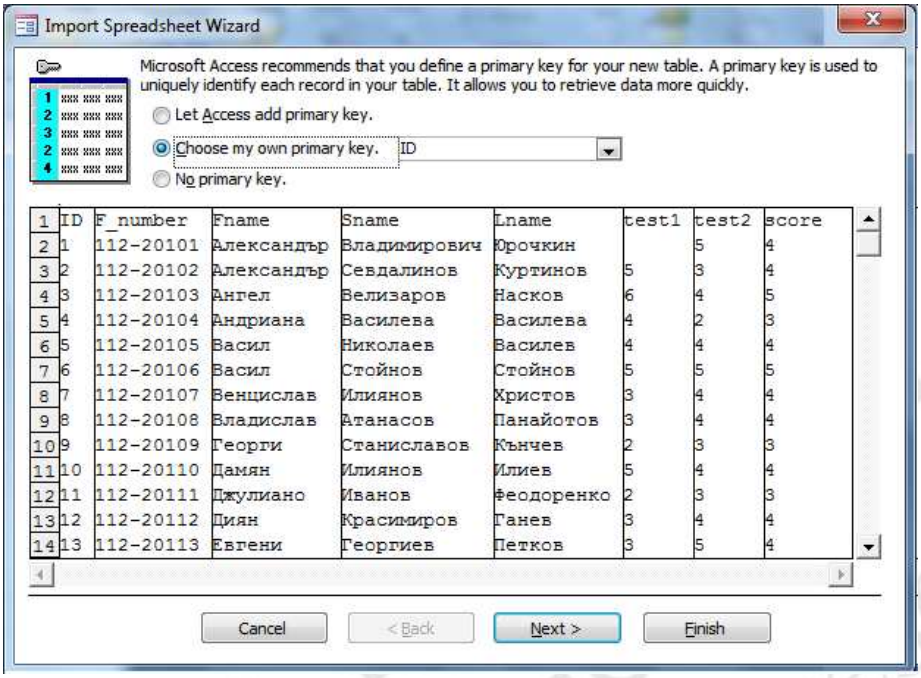
След като се избере как да се получат имената на колоните и се щракне на бутона OK се отваря прозорец, чрез който може да се избере типа на данните в колоните. Access предлага тип double за колони, които съдържат само числови стойности, и тип text за колони, които съдържат текст. Разработчикът може да избере по-прецизно типът на данните. Например ако една колона съдържа само цели числа, то вместо тип double за нея може да се зададе тип integer.



След като се избере типа на данните в колоните и се щракне на бутона OK се отваря прозорец, чрез който може да се избере начинът на получаване на първичният ключ на таблицата. Access предлага две възможности:

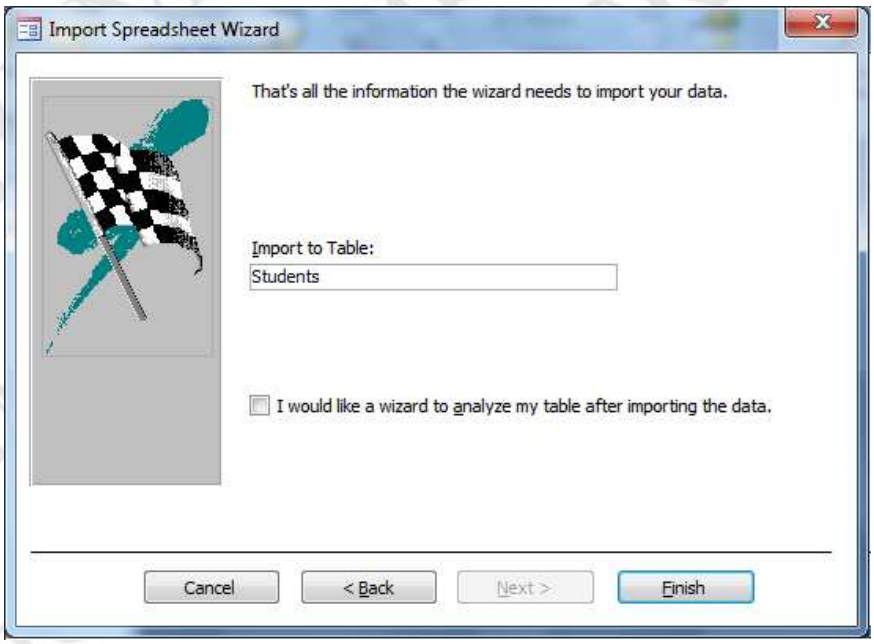
- Let Access add primary key – Access да добави колона със стойности за първичен ключ
- Choose my own primary key – да се използва една от съществуващите колони като колона с първичен ключ
- No primary key – да няма първичен ключ

Последната възможност не е за предпочитане, тъй като съгласно релационният модел е желателно всяка таблица да има първичен ключ.



	ID	F_number	Fname	Sname	Lname	test1	test2	score
1	1	112-20101	Александър	Владимирович	Юрочкин		5	4
2	2	112-20102	Александър	Севдалинов	Куртинов	5	3	4
3	3	112-20103	Ангел	Велизаров	Насков	6	4	5
4	4	112-20104	Андриана	Василева	Василева	4	2	3
5	5	112-20105	Васил	Николаев	Василев	4	4	4
6	6	112-20106	Васил	Стойнов	Стойнов	5	5	5
7	7	112-20107	Венцислав	Илиянов	Христов	3	4	4
8	8	112-20108	Владислав	Атанасов	Панайотов	3	4	4
9	9	112-20109	Георги	Станиславов	Кънчев	2	3	3
10	10	112-20110	Дамян	Илиянов	Илиев	5	4	4
11	11	112-20111	Джулиано	Иванов	Феодоренко	2	3	3
12	12	112-20112	Диян	Красимиров	Ганев	3	4	4
13	13	112-20113	Евгени	Георгиев	Петков	3	5	4

На последната стъпка от помощната програма за импортиране на данни се задава името на таблицата, която се създава в БД на Access.



5.19. Експортиране на данни от MS Access

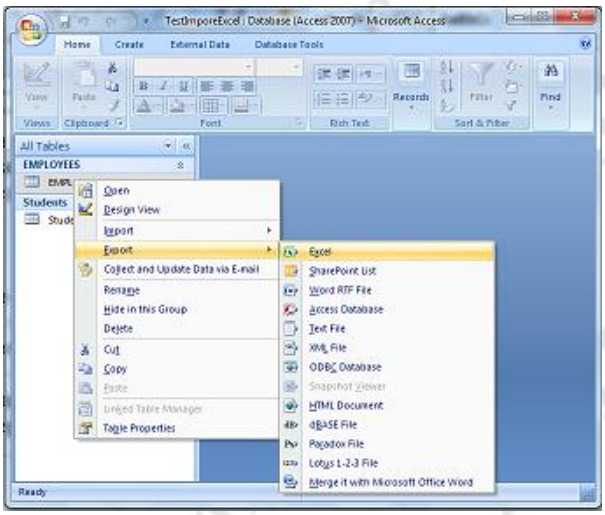
Експортирането е операция, обратна на импортирането Докато при импортирането в БД на Access се внасят данни от външни източници, то при експортирането данни от Access се записват във формат, който дава

възможност за работа с тях чрез друг програмен продукт. Като пример ще разгледаме експортиране на данни от Access в документ на Excel.

5.19.1. Експортиране на данни към документ на Excel

При експортиране данни към Excel, Access създава копие на избраните данни и след това съхранява копираните данни във файл, който може да бъде отворен в Excel. Ако често се налага да се копират данни от Access в Excel, може да се запише за бъдеща употреба подробна информация за операцията за експортиране и дори да се планира автоматично изпълнение на операцията за експортиране през определени интервали от време.

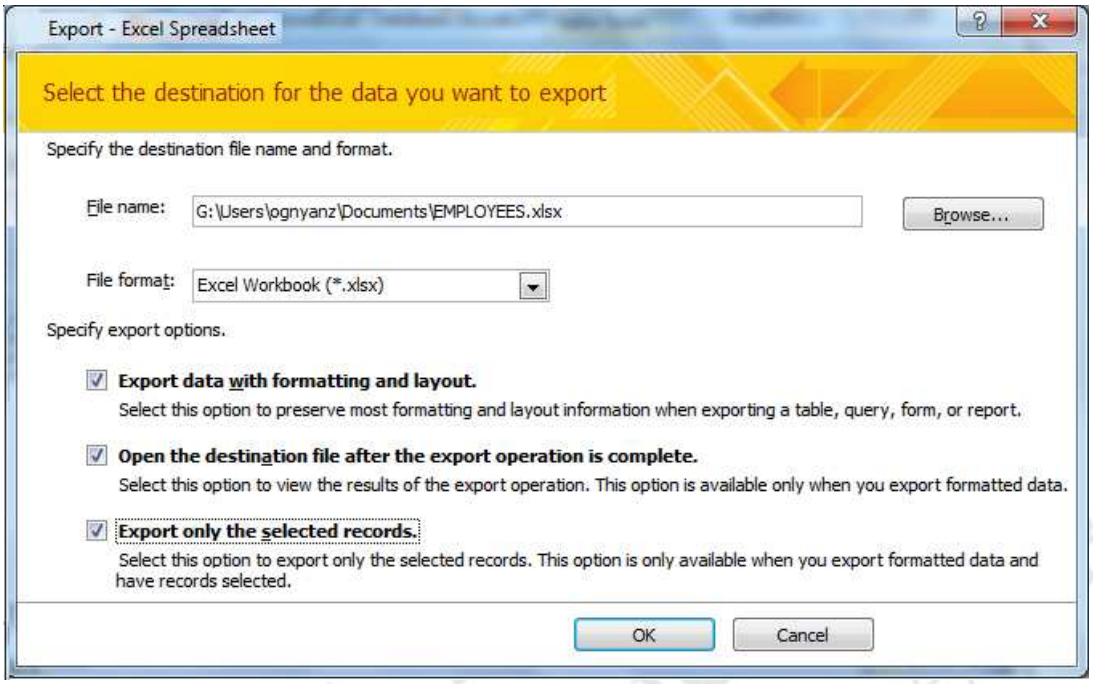
За да стартираме процедурата за експортиране на данни към Excel щракваме с десният бутон на мишката върху таблицата в прозореца „All Tables" и от плаващото меню избираме Export→Excel.



Фиг. 68 Стартиране на процедура за експортиране на таблица в документ на Excel

Отваря се прозорец, в който Access предоставя три опции:

- Export data with formatting and layout – указва при селектирането да се запазят зададените форматирания. Желателно е да използвате тази опция за да си спестите форматирането на експортираната таблица.
- Open the destination file after the export operation is complete – указва да се отвори документът на Excel след експортирането на данните.
- Export only selected records – експортиране само на селектираните записи



Фиг. 69 Задаване на опции при експортитане на данните в Excel

След като се изберат опциите и се щракне на бутона OK данните се експортират и (ако е избрана опцията „Open the destination file...") се стартира Excel и в него се отваря създаденият при експортирането документ.

5.20. SQL

SQL (Structured Query Language) – език за структурирани заявки, е разработен в края на 70-те години от IBM за техния продукт DB2. SQL се е утвърдил като стандартен език за команди към СУБД, който се поддържа на практика от всички сървъри за бази от данни. Посредством SQL програмистът на бази от данни може да извършва:

- Извличане на информация от БД.
- Актуализиране на съдържанието на БД
- Променяне на структурата на БД (DDL)
- Променяне на системните настройки за сигурността (DCL)
- Създаване на потребители и задаване на права на потребителите

В БД на MS Access заявките се съхраняват като SQL код. Access предоставя удобен графичен интерфейс за създаване на заявки към БД, но също така дава възможност за създаване на заявки чрез непосредствено въвеждане на SQL код. Освен това операторът може да превключва между изгледите Design View и SQL View и да да разглежда и евентуално да

коригира SQL кода, генериран от Access при използване на изгледа Design View .

Обемът на курса не дава възможност за разглеждане на всички възможности на езика SQL, затова са разгледани основните елементи на SQL и най-често използваните команди за операции с БД като са дадени и примери за използването им.

5.20.1. Оператори, изрази и условия на SQL

SQL не е език за програмиране с общо предназначение, както например C, Java или PHP. Въпреки това той притежава основните елементи на езиците за програмиране: константи, променливи, изрази, конструкции за управление, пълен набор от оператори за основните операции , както и набор от вградени функции с различно предназначение. В тази глава се разглеждат основните елементи на SQL във връзка с използването на SQL команди при операции с бази от данни в PHP приложенията.

1) Оператори за сравнение
2) Аритметични оператори
3) Логически оператори
4) Оператори за операции със символни низове (конкатенация, quote)
5) SET оператори за комбиниране на резултати от заявки

Аритметичните и логическите оператори бяха разгледани в 5.4.1 Тук ще разгледаме по-подробно операторите за сравнение LIKE и IN

5.20.2. Оператори за сравнение

Операторите за сравнение са най-често използваните оператори в SQL, тъй като те участват при изчисляването на условия, които се включват в повечето SQL команди. Операциите за сравнение могат да се изпълняват за различен тип операнди, но винаги връщат *логически резултат* TRUE или FALSE. Левият операнд на операторите за сравнение по правило е единична стойност, получена в общия случай от изчисляването на израз. Десният операнд освен единична стойност може да бъде списък, интервал или шаблон. Дадената по-долу таблица съдържа операторите за сравнение на SQL.

=	Равно

<>	Различно
>	По голямо
>=	По голямо или равно
<	По малко
<=	По малко или равно
in	Равно на елемент от списъка
not in	Различно от всички елементи от списъка
between	Между две зададени стойности (включително граничните)
not between	Извън интервала от две зададени стойности
is null	Незададена стойност
is not null	Зададена стойност
like	Като зададена стойност, Символът. “*” в шаблона дава съвпадение при коя да е последователност от символи. Символът “?”дава съвпадение за един произволна буква на дадената позиция в низа, а символът “#” – за произволна цифра, [s1-s2] за последователност от символи от s1 до s2, [!s1-s2] извън последователност от символи
not like	Не е като зададен шаблон. Значението на символите “%” и “_” е същото като за оператора like.

таблица 1 Оператори за сравнение на SQL

5.20.2.1. Оператор LIKE

Операторът LIKE сравнява символен низ със зададен шаблон и връща логически резултат. LIKE е много мощен оператор за сравнение, чрез който могат да се задават условия, за който, без използване на този оператор би трябвало да се използват много по-сложни изрази.

Синтаксис:

израз Like “шаблон”

където:

израз (expression) е SQL формула за изчисляване на стойност, използвана в кауза WHERE

шаблон (pattern) е символна константа или символен низ, спрямо който се сравнява изразът

Операторът Like се използва в клауза Where за задаване на условие, по което се селектират записи, които съответстват на зададен шаблон. Като шаблон може да се задава точна стойност (например като „Smith“) или да

се използват заместващи символи, за да се селектира диапазон от стойности (например като „Sm*“).

Използване в шаблоните на обобщаващи символи *, ? , # и област [b_1-b_2]

Символът „*” замества последователност от символи. Например със шаблонът „Sm*“ ще се селектират всички символни низове, които започват със „Sm“, например Smith, Smile и т.н. Символът „*” може да се поставя и в началото и в края на шаблона. Например със шаблонът „*st*” ще се селектират символни низове, които съдържат последователността от символи на произволна позиция, например „stone”, “asterics” и т.н, докато със щаблонът “*st” ще се селектират само тези символни низове, които завършват на “st”, например “best”, “first” и т.н.

Символът „?” замества в шаблона една буква. Например със шаблонът „b?st“ ще се селектират всички символни низове от четири символа, които имат като първи символ “b” и като трети и четвърти символ “st”, а на второ място може да стои коя да е буква, напр. “bist”, “best” и т.н.

Символът „#” замества в шаблона една цифра. Например със шаблонът „1#34*“ ще се селектират всички символни низове от четири символа, които имат като първи символ “1” и като трети и четвърти символ “34”, а на второ място може да стои коя да е цифра, напр. “1234”, “1534” и т.н.

Област от символи в шаблон на оператор LIKE се задава във вида [b_1-b_2], където b_1 и b_2 са символи (букви или цифри), като b_1 трябва да бъде преди b_2 в кодовата таблица. Например с шаблонът “[A-Z]*” ще се селектират всички символни низове, които започват със главните букви от латинската азбука. С шаблона “[Z-A]*” обаче няма да се селектира нищо, тъй като символът “A” не е след символа “Z” в кодовата таблица.

Отрицание в шаблон на оператор LIKE се задава във вида [!b_1-b_2], като символът „!” преди първият символ от областта означава, че съвпадение ще се получи

В шаблонът на оператор LIKE може да се използва и комбинация от обобщаващи символи и области. Например със шаблона „1#34“ ще се селектират всички символни низове от четири символа, които имат като първи символ “1” и като трети и четвърти символ “34”, а на второ място може да стои коя да е цифра, например „123456”, „153477777” и т.н.

5.20.2.2. Оператор IN

Операторът IN проверява дали стойността на зададен израз е равна на някоя от стойностите в зададен списък.
Синтаксис:
израз [Not] In(стойност1, стойност2, …)

Ако стойността, получена при изчисляване на израза е намерена в списъка със стойности, операторът In връща True; в противен случай се връща False. Ако незадължителната клауза Not е включена, резултатът се инвертира, т.е. ако стойността е намерена в списъка, операторът Not In връща False. Да разгледаме един пример:

```
SELECT * FROM Users WHERE ID IN (2,3,4);
```

Пример 2 Използване на оператор IN в SQL команда Select

В примера чрез оператора IN се проверява дали в полето ID на текущия запис се съдържа някоя от стойностите, включени в списъка. Ако се открие съвпадение, командата Select извлича записа.

5.20.3. Извличане на записи – SQL команда SELECT

SQL командата *Select е най-често изпълняваната SQL команда, тъй като приложенията, независимо дали изпълняват други операции с БД, на практика винаги визуализират извадки от съхраняваната в БД информация. Командата Select има следния синтаксис:

```
SELECT [predicate] {* | списък от полета }
FROM списък от таблици
[ WHERE условие ]
[ GROUP BY  списък от полета за групиране ]
[ HAVING условие за групиране ]
[ ORDER BY списък от колони за сортиране ]
[ ASC | DESC ] /* Сортиране в нарастващ/намаляващ ред */
```

където:

predicate	(незадължителен) – определя кои записи ще се извеждат.

* SQL не прави разлика между малки и главни букви в синтаксиса на командите

	Допустими стойности са: ALL – (подразбира се) извеждат се всички записи DISTINGT- извежда се само един от повтарящите се в извадката записи DISTINGT ROW- извежда се само един от повтарящите се в цялата таблица записи
* | списък от полета	Извеждат се всички (*) или зададен списък от полета. Когато се извеждат полета от няколко таблици, преди името на полето може да се запише и името на таблицата, следвана от “.”
FROM	След клаузата се записва списък от имена на таблици, от които се извличат записи.
WHERE условие	След клаузата се записва условие. Ще бъдат извлечени само записите, за които условието има стойност логическа истина - TRUE
GROUP BY списък от полета	След клаузата се записва списък от полета, по които се групират стойностите на полетата от записите
HAVING условие за групиране	След клаузата се записва условие, по което се извличат записите. Използва се само с клауза GROUP BY
ORDER BY	Задава списък от колоните, по които се извършва сортиране на извлечените записи.
ASC | DESC	Определя реда на сортиране чрез клаузата ORDER BY. ASC задава нарастващ ред, DESC – намаляващ.

Да разгледаме един пример за SQL команда Select:

```
SELECT Fname, Lname, score AS Успех FROM Studenst;
```

Пример 3 Извличане на записи с SQL команда SELECT

Командата извлича съдържанието на полетата Fname, Lname и score от всички записи от таблица students като на колоната score е зададено заглавие (caption) “Успех”.

5.20.4. Задаване на условие – клауза Where

Клаузата Where указва кои записи от таблиците, изброени в клаузата FROM, ще се използват при изпълнение на SQL командите SELECT, UPDATE или DELETE. WHERE е незадължителна клауза, но когато е включена, задължително трябва да бъде след клаузата FROM. След клаузата Where се записва израз, от който се получава логическа стойност TRUE (истина) за тези записи, които трябва да се получат от заявката и FALSE (неистина) за тези, които не трябва да се извличат.

Да разгледаме един пример за SQL команда Select с клауза Where.

```
SELECT Fname, Lname, score AS Успех FROM students WHERE score > 5
```

Пример 4 Извличане на записи с SQL команда SELECT със задаване на условие

Командата извлича от таблица students записи със съдържанието на полетата ime, fnom и result , за които стойността в полето result е по-голяма от 5.

5.20.5. Групиране в SQL команда SELECT. Клауза Group By

Клаузата GROUP BY в оператор SELECT се използва за организиране на получените от заявката записи (редове) в групи по определен критерий и обобщаване на данни за тези групи. Обикновено групирането се използва в комбинация с агрегатни функции. Целта е агрегатната функция да се приложи върху групите от записи, а не върху всички извлечени от заявката записи.

Важно е да се запомни, че агрегатните функции могат да се използват заедно с полета в списъка на SQL командата SELECT само ако се извършва групиране на записите по тези полета чрез клаузата GROUP BY.

Да разгледаме един пример: Искаме да получим средната заплата на служителите от таблица Employees, групирана по длъжности. За целта изпълняваме заявката:

```
SELECT AVG(SALARY), JOB_ID FROM Employees GROUP BY JOB_ID;
```

AVG(SALARY)	JOB_ID
5760	IT_PROG

12008	AC_MGR
8300	AC_ACCOUNT
7280	ST_MAN

В резултата се получава средна заплата поотделно за всеки идентификатор на длъжност.

При използване на агрегатна функция можем да включим в списъка на командата SELECT няколко колони, но всички те трябва да са включени в списъка на клаузата GROUP BY. Ако например изпълним заявката

```
SELECT AVG(SALARY), JOB_ID, Department_ID FROM Employees
GROUP BY JOB_ID;
```

ще получим грешка "*Not a GROUP BY expression*". За да се изпълни вярно командата е необходимо да се извърши групиране и по номер на департамент Department_ID както е в командата:

```
SELECT AVG(SALARY), JOB_ID, Department_ID FROM Employees
GROUP BY JOB_ID, Department_ID;
```

5.20.6. Задаване на условие при групиране. Клауза HAVING

При групирането могат да се задават условия чрез клаузата Having, следвана от логически израз (т.е. чрез израз, които връща логическа стойност TRUE или FALSE), но в този израз могат да участват само тези колони, които са включени в списъка за групиране (т.е. след клаузата GROUP BY). Например, ако искаме да изпълним заявка, с която да получим сумата от заплатите по отдели само за служителите на длъжност IT_PROG, командата SELECT с клауза HAVING ще има вида

```
SELECT SUM(SALARY) FROM Employees GROUP BY JOB_ID HAVING
JOB_ID='IT_PROG';
```

Пример 5 Задаване на условие при групиране с клауза HAVING

В случая обаче, тъй като с условието се селектира само една група, същият резултат може да се получи и без групиране, със задаване на условие с клауза WHERE със заявката:

```
SELECT SUM(SALARY) FROM Employees WHERE JOB_ID='IT_PROG';
```

Ако обаче селектираме повече от една група и искаме да получим чрез обобщаваща (агрегатна) функция отделни резултати за всяка група, е

необходимо да се използва групиране и условие, зададено с клауза Having, както е в следващият пример:

```
SELECT JOB_ID, SUM(SALARY) AS [Общо заплати] FROM Employees
GROUP BY JOB_ID HAVING JOB_ID='IT_PROG' OR JOB_ID='SA_MAN';
```

В примера за колоната, в която се извежда резултата от агрегатната функция SUM е зададено заглавие (caption) “Общо заплати”.

5.20.7. Вложени заявки

Вложени заявки (или подзаявките) са заявки, чиито резултати се предават като аргумент на друга заявка. Подзаявките позволяват свързването на няколко заявки в една. Предимство – използването на подзаявки дава възможност да се представят сложните заявки към много таблици като свързани по-прости заявки, които при създаването на SQL кода на заявката могат първоначално да се тестват самостоятелно.

Подзаявките се използват на мястото на израз в заявка и се ограждат в скоби (). На мястото на израза подзаявката връща единична стойност или списък от стойности. Да разгледаме един пример:

```
SELECT department_name FROM departments
WHERE location_id = (SELECT location_id FROM locations WHERE city =
'Seattle');
```

Пример 6 Използване на подзаявка в команда SELECT

В примера със заявката се извличат имената на департаментите, които се намират в град Сиатъл. Информацията за локациите се съдържа в таблица LOCATIONS а в таблица DEPARTMENTS се съдържат само идентификаторите на локациите. От подзаявката се получава идентификатора на град Сиатъл и той се използва в условието WHERE на външната команда SELECT.

Полезно е да се запомни, че в много случаи вместо използване на подзаявки може да се използва свързване на таблици, и в тоя случай заявката ще се изпълнява по-бързо. Например същия резултат, както в горната заявка можем да получим и със следната заявка, която използва вътрешно свързване (JOIN) между таблиците

```
SELECT department_name FROM departments D
JOIN locations L ON D.location_id=L.location_id AND city = 'Seattle';
```

Пример 7 Използване на свързване на таблици вместо подзаявка

Има случаи обаче, в които в сложни заявкп се налага използването могоредови (агрегатни) функции и при използването на подзаявка се избягва необходимостта от групиране във външната заявка. Ако например е необходимо да се получат от таблица EMPLOYEES имената на служителите, които получават максималната заплата, може да се използва следната заявка

```
SELECT first_name, last_name, Salary AS MAX_Salary FROM employees
WHERE Salary=(SELECT MAX(Salary) FROM employees);
```

Пример 8 Използване на подзаявка, която включва агрегатна функция

В примера от подзаявката се получава максималната заплата, а външната заявка извлича от таблица EMPLOYEES имената на служителите с тази стойност на Salary. Ако агрегатната функция MAX беше включена във външната заявка, трябваше да се извърши групиране по колони first_name и last_name и нямаше да се получи желания резултат.

5.20.8. Използване на подзаявки, които извличат списък от стойности

Подзаявки, които извличат стойности се използват съвместно с оператор IN, с който се проверява дали зададена стойност се съдържа в списък. Например ако заявката трябва да извлече от таблица Employees имената на всички служители на длъжност Manager тя трябва да използва подзаявка, с която да получи от таблица JOBS всички идентификатори на такива служители и да извлече от таблица Employees само тези записи, за които стойностите на JOB_ID се съдържат в списъка. Това може да стане със следната заявка:

```
SELECT first_name, last_name FROM employees where job_id IN
(SELECT job_id FROM jobs WHERE JOB_TITLE LIKE '*Manager*');
```

Пример 9 Използване на подзаявка, която връща списък

Същия резултат, както в горната заявка можем да получим обаче и със заявка, която използва вътрешно свързване (JOIN) между таблиците.

Операторът IN допуска и използване на отрицание NOT. В тоя случай условието в клауза WHERE ще се изпълнява тогава, когато стойността на полето или израза вляво на оператора NOT IN не се съдържа в списъка вдясно. Например заявката

```
SELECT first_name,last_name FROM employees where job_id NOT IN
(SELECT job_id FROM jobs WHERE JOB_TITLE LIKE '*Manager*');
```

Пример 10 Използване оператор NOT IN с подзаявка, която връща списък

ще върне от таблица Employees имената на служителите, чиито длъжности не съдържат низа “Manager”, т.е. длъжностите, чиито идентификатори не се съдържат в списъка, който връща подзаявката.

5.20.9. Използване на подзаявки в списъка от колони на команда SELECT

Подзаявка може да се включи и в списъка от колони (или изрази) на команда SELECT. В такъв случай колоните от извлечената от подзаявката динамична таблица се включват като част от колоните на извлечената от външната команда SELECT таблица. Това може да се използва например в случай когато получената от подзаявката стойност участва в израз във външната заявка, както е в следващия пример:

```
SELECT first_name,last_name,
ROUND(Salary/(SELECT MAX(Salary) FROM employees)*100) & "%"
AS "Percent_From_MAX" FROM employees;
```

Пример 11 Използване на подзаявка в израз, включен в списъка от колони на команда SELECT

В примера подзаявката извлича максималната стойност на Salary от таблица Employees, която се използва в израз за изчисляване на това, колко процента от максималната стойност е стойността на Salary за всеки от

извлечените записи. Получената стойност се закръглява до цяло число с функция ROUND. Обърнете внимание на използването на оператора за слепване на символни изрази &. Чрез него изчислените проценти се преобразуват в символен низ и към него се слепва символа “%”.

5.20.10. Използване на подзаявки, които извличат динамична таблица

Подзаявка, която извлича таблица, се използва на мястото на таблица в клаузата FROM или с оператор JOIN при свързване на таблици. В тоя случай получената от подзаявката таблица е само за четене (Not Updatable) следователно резултата от нейното изпълнение е подобен на този, който се получава от изпълнението на съхранена заявка (Query). Да разгледаме един пример:

```
SELECT first_name, last_name, MAX_Salary FROM employees E
JOIN (SELECT MAX(Salary) AS MAX_Salary FROM employees) M ON
E.salary=M.MAX_Salary;
```

Пример 12 Използване на подзаявка, която извлича таблица

В примера от подзаявката се получава динамична таблица с псевдоним M, която се свързва с таблицата Employees (с псевдоним E) посредством оператор JOIN.

5.20.11. Обединяване на резултати от заявки. Оператори UNION и UNION ALL

Операторът UNION комбинира резултатите от две SQL заявки в обща виртуална таблица. За да може да се изпълни това обединение е необходимо двете заявки да връщат записи с един и същ брой колони от съвместими типове. Разликата между UNION и UNION ALL е, че операторът UNION премахва дублиращите се записи, докато UNION ALL ги запазва. Да разгледаме два примера, от които се вижда разликата:

```
SELECT first_name, last_name FROM employees
WHERE Salary >15000
```

```
UNION
SELECT first_name, last_name FROM employees
WHERE Salary >16000;
```

Пример 13 Обединение на заявки с оператор UNION

Както се вижда от зададените условия на заявките в примера, в резултатите им ще има повтарящи се записи. Операторът UNION обаче премахва повторенията и се получават само три записа (резултата вляво), докато ако се използва оператор UNION ALL записите се повтарят (резултата вдясно).

Steven	King
Neena	Kochhar
Lex	De Haan

Резултат от обединение с operator UNION

Steven	King
Neena	Kochhar
Lex	De Haan
Steven	King
Neena	Kochhar
Lex	De Haan

Резултат от обединение с operator UNION ALL

Ако типовете данни от колоните от двете заявки не са съвместими, Access, (за разлика от SQL Server и Oracle) не се генерира грешка. Например ако изпълним заявката

```
SELECT first_name, last_name, Salary FROM employees
WHERE Salary >15000
UNION
SELECT first_name, last_name, Email FROM employees
WHERE Salary >16000;
```

въпреки различните типове данни от колони Salary и Email не се генерира грешка, а Access преобразува типа Number на данните от колона Salary в тип Text.

Онлайн източници

https://stackoverflow.com/questions/10393087/expressing-basic-access-query-criteria-as-regular-expressions Expressing basic Access query criteria as regular expressions

Онлайн източници

https://www.worldbestlearningcenter.com/index_files/VBA-Access-Declaring-variables.htm VBA for Access 2007 Tutorials

https://support.microsoft.com/en-us/office/move-data-from-excel-to-access-90c35a40-bcc3-46d9-aa7f-4106f78850b4 Move data from Excel to Access

https://quick-adviser.com/how-do-i-open-microsoft-access-runtime/#How_do_I_open_Microsoft_Access_runtime How do I open Microsoft Access runtime?

https://support.microsoft.com/en-us/office/guide-to-expression-syntax-ebc770bc-8486-4adc-a9ec-7427cce39a90#bmoverview Guide to expression syntax

https://support.microsoft.com/en-us/office/use-parameters-in-queries-forms-and-reports-8209eb5c-1589-42e2-9b20-4181f4c7a356#bmform_param_5 Use parameters in queries, forms, and reports

https://support.microsoft.com/en-us/office/introduction-to-data-types-and-field-properties-30ad644f-946c-442e-8bd2-be067361987c Introduction to data types and field properties

https://courses.lumenlearning.com/wm-computerapplicationsmgrs-2/chapter/field-properties/ Field Properties