

Z.T.Məhərrəmov
V.H.Abdullayev

VERİLƏNLƏR BAZALARI

(ADO TEXNOLOGİYASI İLƏ MÜDAXİLƏ)

DƏRS VƏSİTİ

**Z.T. Məhərrəmov
V.H.Abdullayev**

**VERİLƏNLƏR BAZALARI
(ADO TEXNOLOGİYASI İLƏ MÜDAXİLƏ)**

DƏRS VƏSƏİTİ

BAKİ – 2019

Elmi redaktor:

Əli Nağıyev
Sumqayıt Dövlət Universitetinin
“Texniki kibernetika” kafedrasının professoru, t.e.d.

Rəyçilər:

Taleh Əsgərov
MAA-nın Aerokosmik informasiya sistemləri
kafedrasının dosenti, t.ü.f.d.,

Tahir Əlizadə
AMEA İdarəetmə Sistemləri İnstitutunun
Qeyri-stasionar proseslərin real vaxt rejimində
analizi laboratoriyasının müdiri, t.e.n.

Məhərrəmov Z.T., Abdullayev V.H. Verilənlər bazaları
(ADO texnologiyası ilə müdaxilə). Dərs vəsaiti. Bakı: “Elm”, 2019.
- 228 səh.

İSBN 978-9952-523-10-3

Dərs vəsaiti müxtəlif verilənlər bazalarına ADO texnologiyası ilə müdaxilə üsullarının öyrədilməsinə həsr edilmişdir. Kitabda ADO mexanizmi və Delphi və Visual Studio C# dillərində mövcud olan ADO komponentləri və SQL sorğu dili ətraflı izah edilmişdir. Ms Access, Paradox, Excel və Microsoft SQL Server-də yaradılmış verilənlər bazalarına ADO texnologiyası ilə Delphi və Visual Studio C# proqramlaşdırma dillərindən qoşulma metodikası nümayiş etdirilmiş və həmin bazaların cədvəllərinin istifadəsi ilə əlavələr layihələndirilmişdir. Bu məsələni həll etmək üçün Ms Access 2010 və Microsoft SQL Server-də verilənlər bazasının yaradılması və, habelə, SQL sorğu dili əhatə edilmişdir.

Kitabdan 050632, 050655, 050631, 050656 ixtisaslarında təhsil alan tələbələr, verilənlər bazalarının layihələndirilməsi ilə məşğul olan magistrələr, digər mütəxəssislər və müəllimlər istifadə edə bilərlər.

655 (07) – 2019

© **Z.T.Məhərrəmov, V.H.Abdullayev, 2019**

M ü n d ə r i c a t

Birinci fəsil. Verilənlər bazasının layihələndirilməsinin nəzəri əsasları 7

1.1. Əsas anlayışlar.....	7
1.2. Verilənlər bazasının növləri	8
1.2.1. İyerarxiyalı verilənlər bazası.....	9
1.2.2. Şəbəkə verilənlər bazası.....	10
1.2.3. Relyasiyalı verilənlər bazası	11
1.2.4. Obyektyönlü verilənlər bazası	12
1.3. İnformasiya obyektləri və onların seçilməsi.....	12
1.4. Verilənlər bazası cədvəlləri	15
1.4.1. İlkın açarlar	17
1.4.2. İndeksler.....	18
1.4.3. Cədvəl formatları	20
1.5. Cədvəllərarası relyasiya münasibətləri	22
1.5.1. Birin–birə münasibəti.....	22
1.5.2. Birin – çoxa münasibəti	23
1.5.3. Çoxun – çoxa münasibəti.....	24
1.5.4. Bir cədvəlin yazıları arasında əlaqə	25
1.5.5. İstinad tamlığı	27
1.6. Verilənlər bazasının normallaşdırılması	30
1.6.1. “Tədris prosesi” predmet sahəsinin informasiya– məntiq modeli əsasında normallaşdırma.....	34
1.6.2. “Anbar” predmet sahəsinin informasiya– məntiq modeli əsasında normallaşdırma.....	43
1.6.2.1. Birinci normal forma	43
1.6.2.2. İkinci normal forma	47
1.6.2.3. Üçüncü normal forma	51
1.6.2.4. Tranzaksiya anlayışı	53

1.6. 3. “Futbol üzrə ölkə çempionatı” predmet sahəsinin informasiya–məntiq modeli əsasında normallaşdırma	54
1.7. Verilənlər bazasını idarəetmə sistemləri	58
1.8. Bəzi VBİS–lərin imkan və xarakteristikalarının qısa xülasəsi	63

İkinci fəsil. Microsoft Access 2010–da cədvəllərin yaradılması..... 71

2.1. Microsoft Access VBİS–in arxitekturası	71
2.2. Cədvəllərin yaradılması və verilənlərin tipləri	72
2.2.1. Konstruktor rejimində cədvəlin yaradılması	75
2.3. “Tədris prosesi” predmet sahəsinin cədvəllərinin yaradılması	77
2.3.1. Cədvələ verilənlərin daxil edilməsi və onlara baxış	83
2.3.2. Cədvəllər arasında əlaqələrin yaradılması	85

Üçüncü fəsil. SQL dili..... 88

3.1. SQL dilinin xüsusiyyətləri	88
3.2. SELECT operatoru	91
3.2.1. WHERE operandı	95
3.2.2. ORDER BY operandı	99
3.2.3. Sorgularda hesablamalar	100
3.2.3.1. Hesablanan sahələr	100
3.2.3.2. Aqreqat funksiyalar	102
3.2.4. GROUP BY operandı	103
3.2.5. Cədvəllərin birləşdirilməsi	104
3.2.6. Parametrlı sorgular	109
3.2.6.1. Parametrlərə qiymətlərin Params xassəsi vasitəsi ilə proqram yolu ilə verilməsi	111
3.2.7. Verilənlərin modifikasiyası	116
3.2.7.1. INSERT operatoru	117
3.2.7.2. UPDATE operatoru	117
3.2.7.3. DELETE operatoru	118

Dördüncü fəsil. Delphi–nin digər verilənlər bazası cədvəlləri ilə əlaqəsi	120
4.1. Verilənlərə müdaxilə mexanizmləri	120
4.2. ADO ilə iş.....	126
4.3. Delphi–də dəstəklənən müdaxilə mexanizmləri və ADO ilə iş üçün komponentlər	128
4.3.1. ADO komponentləri	130
4.3.1.1. TADOConnection komponenti	130
4.3.1.2. TADODataSet komponenti.....	132
4.3.1.3. TADOCommand komponenti.....	133
4.3.1.4. TADOTable komponenti	133
4.3.1.5. TADOQuery komponenti	134
4.3.1.6. TADOStoredProc komponenti	135
4.3.1.7. TRDSCollection komponenti	135
4.4. Delphi ilə Ms Access cədvəlləri arasında əlaqə	135
4.5. Access cədvəlləri üzərində filtrləmə əməliyyatı və sorğuların yaradılması.....	140
4.5.1. SQL Builder ilə sorğuların yaradılması.....	142
4.6. Delphi ilə Ms Excel cədvəlləri arasında əlaqə.....	148
4.7. Paradox cədvəllərinin ADO vasitəsi ilə Delphi əlavələrinə qoşulması.....	152
4.8. ADO və VB haqqında bəzi məlumatların alınması ...	153
Beşinci fəsil. Microsoft SQL Server-də verilənlər bazasının yaradılması	155
5.1. Serverə qoşulma	155
5.2. Verilənlər bazasının yaradılması	157
5.2.1. Verilənlərin tipləri.....	158
5.2.2. Cədvəllərin yaradılması	159
5.2.4. Cədvəlin açılması və verilənlərin daxil edilməsi	163
Altıncı fəsil. Visual Studio mühitində ADO ilə verilənlər bazalarına integrasiya.....	168
6.1. ADO.NET texnologiyası	168
6.2. Ms Access-də yaradılmış verilənlər bazasına	

qoşulma	172
6.3. Visual Studio mühitində ADO ilə iş üçün	
komponentlər	189
6.3.1. VBİS-lərə müdaxilə üsulları	189
6.3.1.1. ADO.NET müdaxilə metodu	191
6.3.1.1. 1. Çox səviyyəli sistemlər.....	192
6.3.1.1. 2. Əlaqəsi kəsilmiş sistemlər	192
6.3.1.1. 3. Verilənlərin paylanmış emalı	193
6.3.1.1. 4. Verilənlər provayderi.....	194
6.3.2. ADO komponentləri	194
6.3.2.1. dataSet verilənlər mənbəyi	196
6.3.2.2. dataGrid komponenti	198
6.3.2.3. sqlDataAdapret verilənlər adapteri ..	205
6.3.2.4. bindingSource komponenti.....	218
ƏDƏBİYYAT	228

Birinci fəsil

VERİLƏNLƏR BAZASININ LAYİHƏLƏNDİRİLMƏSİNİN NƏZƏRİ ƏSASLARI

1.1. Əsas anlayışlar

Verilənlər bazası (VB) – predmet sahəsinin obyektlərinin strukturunu və əlaqələrini əks etdirən və kompyuterin yaddaşında saxlanılan müəyyən qarşılıqlı əlaqəli verilənlər yığımıdır.

Predmet sahəsi – əsas göstəricilərini VB-də əks etdirmək istədiyimiz real dünyanın bir hissəsidir. Predmet sahəsi sonsuzdur və həm çox zəruri, həm də az əhəmiyyət kəsb edən anlayış və verilənlərdən ibarətdir. Predmet sahəsi kimi tədris prosesini, bankların fəaliyyətini, anbarları, arxivləri və s. göstərmək olar.

Predmet sahəsinin modeli – predmet sahəsi haqqında biliklərimizdir. Belə biliklər ekspertin beyində qeyri–formal biliklər şəklində və ya hər hansı vasitə ilə formalaşdırılmış şəkildə ola bilər. VB layihələndirildikdə predmet sahəsinin modelini xüsusi qrafik təsvirlə ifadə etmək daha səmərəli olur. Predmet sahəsini təsvir etmək üçün çoxlu metodlar mövcuddur.

Verilənlər bazasının əsas anlayışlarından biri informasiyadır. **İnformasiya** dedikdə hər hansı hadisə, proses, obyekt haqqında istənilən məlumat başa düşülür.

Verilənlər – informasiyanın dialektik tərkib hissəsi olmaqla, adətən, nitq, mətn və əyani informasiyaya xas olan adi sərbəst strukturlu informasiyadan fərqli olaraq, daha ciddi, *xüsusi formada təsvir olunmuş informasiya növüdür*. Belə təsvir forması verilənlərin toplanılmasını, saxlanılmasını avtomatlaşdırmağa imkan verir və sonradan bu verilənlər adamlar və ya informasiya vasitələri ilə emal edilir. Kompyuter texnologiyasında verilənlər – kompyuterdə saxlanmaq, emal olunmaq və habelə əlaqə kanalları

ilə ötürülmək üçün rahat formada diskret və qeyd olunmuş informasiyadır.

Bəzi hallarda “verilənlər” anlayışını “məlumat” anlayışı ilə əvəz etməyə cəhd olunur, bu tamamilə yanlış bir mülahizədir. VB layihələndirildikdə verilənlərin əsasən aşağıdakı tiplərindən istifadə edilir:

- ədədi xarakterli;
- simvol və ya hərf-rəqəm;
- vaxt və tarix;
- mətn;
- ikilik;
- hiperistinad.

Verilənlərin məntiqi – predmet sahəsinin anlayışlarını, onların qarşılıqlı əlaqələrini və habelə, verilənlər üzərinə qoyulan məhdudiyyətləri təsvir edir.

Verilənlərin fiziki modeli dedikdə verilənlərin məntiqi modeli yaradıldıqda hər hansı bir verilənlər bazasını idarəetmə sistemində qəbul edilmiş formada verilənlərin cədvəllərlə təsviri, atributların cədvəl sütunları ilə təsvir edilməsi, açar atributlar üçün unikal indekslərin yaradılması və s. başa düşülür.

Verilənlər bazalarını idarəetmə sistemləri (VBİS) – VB yaratmaq, bazanın verilənləri üzərində müxtəlif axtarış, sıralama, yeniləmə, sorğu və s. kimi əməliyyatları yerinə yetirməyə imkan verən proqram vasitələri kompleksidir.

1.2. Verilənlər bazasının növləri

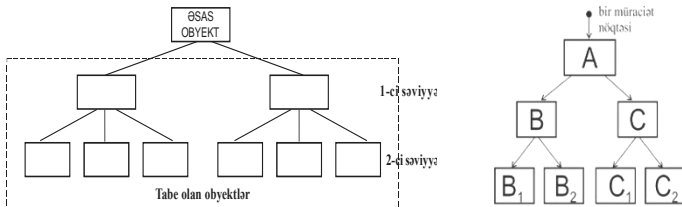
İstənilən verilənlər bazasının əsasını verilənlər modelləri təşkil edir. **Verilənlər modeli** verilənlərin strukturu və onlar üzərində yerinə yetirilən əməliyyatlar yığımıdır. Verilənlərin əsas modelləri aşağıdakılardır:

- iyerarxiya modeli ;
- şəbəkə modeli;
- relyasiya modeli;
- obyektlyönlü model.

Bu model növlərinə uyğun olaraq verilənlər bazalarını da iyerarxiyalı, şəbəkə, relyasiya və obyektlyönlü VB adlandırırlar.

1.2.1. İyerarxiyalı verilənlər bazası

İyerarxiyalı VB əsasən 60-cı illərin əvvəllərində istifadə edilmişdir. Onlar adi ağac şəklində təsvir olunur. Ağacların zirvələri müxtəlif iyerarxik səviyyələrdə yerləşir (şəkil 1.1). Verilənlər iki kateqoriyaya bölünür: *əsas* və *təbə olan*. Beləliklə, bir obyekt əsas obyekt, digərləri isə təbə olan obyektlər olur.



Şəkil 1.1. İyerarxiyalı VB sxemi

İyerarxiya modellərində yalnız əsas obyektə birbaşa müraciət etmək olar, yerdə qalan obyektlərə müraciət yalnız modelin zirvəsində duran obyekt vasitəsi ilə mümkündür. Məsələn, B_2 obyektinə müraciət yalnız $A \rightarrow B \rightarrow B_2$ əlaqəsi ilə mümkündür, yəni iyerarxiyalı VB-də yazılara birbaşa müraciət mümkün deyildir. İyerarxiya modelləri üzərinə kifayət qədər məhdudiyyətlər və qaydalar qoyulur. Bunların bəziləri aşağıdakılardır:

- bütün növ əlaqələr ($1:1$, $1:M$, $M:N$) funksional olmalıdır;
- iyerarxiya ağacı dövrü olmayan, istiqamətlənməmiş qraf təşkil edir;
- ağac əsas (köklü) ağacdən və alt ağaclardan ibarətdir;
- ağacın hər bir buğumu (düynü) – yalnız bir valideyn buğumla əlaqəlidir (hər bir buğumun yalnız bir valideyni var, bir valideynin isə istənilən qədər törəməsi (buğumu) ola bilər);
- ağacın budaqları “ilkin–törəmə” tipli əlaqəyə uyğundur;
- hər bir törəmə buğuma yalnız ilkin buğum vasitəsi ilə müraciət etmək olar.

İyerarxiya modelinin çatışmayan cəhətləri aşağıdakılardır:

• verilənlər üzərində əməliyyatlar yuxarıdan aşağıya prinsipi ilə yerinə yetirilir. Əksinə axtarış isə çox çətindir və hətta əksər hallarda heç mümkün deyildir;

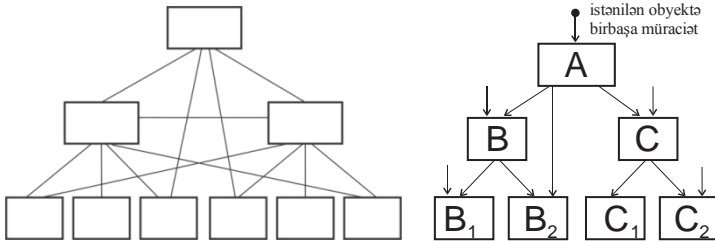
• məntiqi səviyyədə verilənlərin təkrarı baş verir;

• **M:N** münasibətini təsvir etmək üçün ağaclar təkrarlanmalıdır;

• valideyn və varis arasında informasiya tamlığına avtomatik əməl olunur: heç bir varis valideynsiz mövcud ola bilməz. Ona görə də valideynsiz varisi saxlamaq mümkün olmur və ilkin buğumu pozduqda bütün varislər də pozulur.

1.2.2. Şəbəkə verilənlər bazası

İyerarxiya modeli ilə, demək olar ki, eyni zamanda verilənlərin şəbəkə modeli tətbiq edildi. Şəbəkə modeli iyerarxiya modelinin genişlənmiş formasıdır. Belə modelin əsas məqsədi iyerarxiya modelində olan çatışmazlıqları aradan qaldırmaq idi. Şəbəkə modeli də qraflar şəklində təsvir olunur, lakin burada istənilən obyekt həm əsas, həm də tabe olan obyekt ola bilər (şəkil 1.2).



Şəkil 1.2. VB–nin şəbəkə modeli

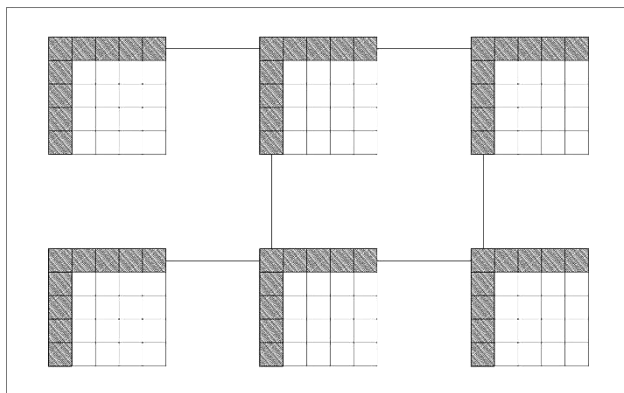
Qraflar nəzəriyyəsi nöqtəyi–nəzərindən şəbəkə modelinə ixtiyari qraf uyğun gəlir. İyerarxiya modelində hər bir varis obyektin yalnız bir əcdad obyektini ola bildiyi halda, şəbəkə modelində varis obyekt istənilən qədər əcdada malik ola bilər. Obyektlər arasında əlaqə əcdad və varis arasında istənilən qədər əlaqədən ibarət olur. Beləliklə, şəbəkə modelində hər bir obyektə birbaşa müraciət etmək mümkündür (məsələn, B_2 obyektinə A və

B obyektlərinin əlaqəsi vasitəsi ilə deyil, birbaşa müraciət oluna bilər.

Şəbəkə modellərinin strukturunun sərt olması və verilənlərin təsvirinin mürəkkəb olması əksər hallarda bu modeldən imtina edilməsinə səbəb oldu.

1.2.3. Relyasiyalı verilənlər bazası

Məhz relyasiyalı verilənlər bazası 70-ci illərdən başlayaraq proqramlaşdırmada geniş tətbiq olunmağa başladı. Bu modeli *IBM* firmasının əməkdaşı Edqar Kodd təklif etmişdir. Model öz adını “*relation*” ingilis sözündən götürmüşdür ki, tərcüməsi “*münasibət*”, “*əlaqə*” deməkdir. Relyasiyalı VB-də obyektlər və onların qarşılıqlı əlaqələri sətir və sütunlardan ibarət cədvəllərdə təsvir olunur (şəkil 1.3). Belə VB-nin əsas üstünlüyü onun sadəliyi və strukturunun çevikliyidir.



Şəkil 1.3. Relyasiyalı VB sxemi

Hər bir cədvəl verilənlər bazasının bir obyektini təşkil edir. Belə bazanı kompüterdə yaratmaq çox asan olur. Biz növbəti bölmələrdə ancaq relyasiyalı VB–dən söhbət açacağıq.

Beləliklə, VB-ni təşkil edən cədvəllər disklərdə, ayrıca bir qovluqda saxlanır. Bütövlükdə bu qovluq VB-ni təşkil edir, onun tərkibinə daxil olan cədvəllər isə ya bir faylda, ya da bir neçə ayrı-ayrı fayllarda yadda saxlanır. Bu fayllar üzərində əməliyyat sisteminin yol verə biləcəyi istənilən əməliyyatları aparmaq olar.

Növbəti bölmələrdə relyasiyalı baza haqqında daha vacib bilikləri əhatə edəcəyik.

1.2.4. Obyektyönlü verilənlər bazası

Obyektyönlü verilənlər bazası şəbəkə və relyasiya modellərini birləşdirməklə mürəkkəb strukturlu verilənlərdən ibarət böyük VB yaratmaq üçün istifadə edilir. Bu model 80-ci illərin ortalarından tətbiq edilməyə başlamışdır. Lakin bu istiqamət son illərdə daha çox tətbiq edilir. Verilənlərin obyektyönlü modelində real dünyanın istənilən mətləbi yalnız bir anlayışla – *obyektlə* təsvir olunur. Obyekt dedikdə onun vəziyyəti və davranışı yada düşür. Obyektin vəziyyəti onun xassələrinin qiymətləri – *atributları* ilə müəyyən olunur. Obyektin davranışı isə obyekt üzərində əməliyyatlar yerinə yetirə bilən *metodlarla* müəyyən olunur. Eyni xassəli və eyni davranışlı obyektlər *siniflərdə* qruplaşdırılır. Obyekt bir və ya bir neçə sinfin nüsxəsi olur. Obyektyönlü modellər xassə və metodları irsi mənimsəmə xüsusiyyətinə malikdir.

Hal-hazırda obyektyönlü VBİS-lər sahəsində tədqiqatlar aparılır. Müasir dövrdə istifadə olunan obyektyönlü VBİS-lərə **O2, EXODUS, POSTGRES, ORION-1, ORION-2, ORION-1SX, db40** və s. sistemləri misal göstərmək olar.

1.3. İnformasiya obyektləri və onların seçilməsi

İnformasiya obyekti – real obyektin, prosesin və ya hadisənin informasiya təsviridir. İnformasiya obyekti predmet sahəsinin miqdarı və keyfiyyət xarakteristikalarından ibarət məntiqi qarşılıqlı əlaqəli rekvizitlər yığımından əmələ gəlir. Məsələn, “*Tədris prosesi*” predmet sahəsi üçün *TƏLƏBƏ, MÜƏLLİM, KAFEDRA* və s. informasiya obyektləridir. İnformasiya obyektləri predmet sahəsinin təsviri əsasında rekvizitlər arasında funksional asılılıqları müəyyən etməklə seçilir. İnformasiya obyektlərinin rekvizitlər yığımı normallaşdırmanın tələblərini təmin etməlidir. Hər bir informasiya obyektinin unikal adı olmalıdır, məsələn, *TƏLƏBƏ, MÜƏLLİM, KAFEDRA* və s.

İnformasiya obyektinin nüsxələri olur. Məsələn, *TƏLƏBƏ* obyektinin nüsxəsi konkret tələbədən ibarətdir. Nüsxə rekvizitlərin konkret qiymətlər yığımından əmələ gəlir və informasiya obyektinin *açarının* qiyməti ilə birmənalı təyin olunmalıdır. İnformasiya obyektinin açarı isə bir və ya bir neçə açar rekvizitlərdən ibarət olur. Beləliklə, rekvizitlər açar və təsviredici rekvizitlərə bölünür. Təsviredici rekvizitlər açardan funksional asılı olur.

Rekvizitlərin funksional asılılığını araşdırdıqda hesabı asılılıqlar nəzərə alınmır. Məsələn, məhsulun qiymətinin onun miqdarından asılılığını nəzərə almaq lazım deyildir, onu sonradan hesablamaq olar. Elə funksional asılılıqlar aşkar edilməlidir ki, təsviredici və açar rekvizitlər arasında əlaqələr müəyyən edilə bilsin və bunların əsasında hər bir rekvizitin tərkibi aşkar edilsin.

Verilənlər modelini qrafik təsvir etdikdə hər bir informasiya obyektini düzbucaqlı ilə təsvir olunur. Düzbucaqlıların daxilində obyektin və açar identifikatorun adları yazılır. Şəkil 1.4–də *QRUP* və *TƏLƏBƏ* informasiya obyektlərinin belə təsviri göstərilmişdir. Şəkildən görüldüyü kimi, *NQ* (*Qrupun nömrəsi*) *QRUP* obyektinin sadə açarı, *NQ+TN* (*Tələbənin nömrəsi*) isə *TƏLƏBƏ* obyektinin tərkibli açarıdır.

<i>QRUP</i>	<i>TƏLƏBƏ</i>
<i>QN</i>	<i>QN + TN</i>

Şəkil 1.4. İnformasiya obyektinin təsviri

Hər bir informasiya obyektini aşağıdakı normallaşdırma (normallaşdırma ilə bir az sonra ətraflı tanış olacağıq) tələblərinə cavab verməlidir:

- informasiya obyektini unikal açara malik olmalıdır;
- açar sadə və ya tərkibli olmalıdır;
- bütün təsviredici rekvizitlər arasında qarşılıqlı əlaqələr olmalıdır, yəni onlar arasında heç bir funksional asılılıq olmamalıdır;
- tərkibli açara daxil olan bütün rekvizitlər arasında da asılılıqlar olmamalıdır;

- hər bir təsviredici rekvizit açıqdan funksional, yəni tam asılı olmalıdır, başqa sözlə, hər bir açar qiymətinə yalnız bir təsviredici rekvizit uyğun gəlməlidir;

- tərkibli açarlı təsviredici rekvizitlər tamamilə bu açarı əmələ gətirən bütün rekvizitlər yığımından asılı olmalıdır;

- hər bir təsviredici rekvizit açıqdan tranzitiv olaraq, yəni digər aralıq rekvizit vasitəsi ilə asılı olmamalıdır.

Tranzitiv asılılıq mövcud olduqda bir rekvizit əvəzinə iki rekvizit əmələ gətirməklə rekvizitlər yığımını parçalamaq olar.

Predmet sahəsinin normallaşdırmanın tələblərinə cavab verən informasiya obyektlərini seçmək üçün iki yanaşma mövcuddur: intuitiv və formal.

İntuitiv yanaşmada informasiya obyektlərini asanlıqla aşkar etmək olur, lakin bu zaman alınan informasiya – məntiq modelini sonrakı mərhələlərdə yenidən işləmək tələb olunur. Bu özünü xüsusən obyektlər arasındakı çoxun–çoxa (**M:N**) münasibətində büruzə verir. Əgər kifayət qədər təcrübə yoxdursa, bu daha ciddi səhvlərə gətirir. Normallaşdırmanın tələblərinin yoxlanılması nəticəsində informasiya obyektlərinin yenidən seçilməsi zərurəti ortaya çıxır.

Formal yanaşmanın nəzəri əsasları Amerika alimi C. Martin tərəfindən işlənmişdir. Belə yanaşmada VB-də saxlanılacaq sənədlər (cədvəllər), onların rekvizitləri və habelə bu sənədlərin formaları aşkar edilməlidir. Rekvizitlərin siyahısını cədvəl 1.1-də göstərildiyi kimi təsvir etmək olar.

Cədvəl 1.1. *QRUP* informasiya obyektinin rekvizitlərinin funksional asılılığı

Cədvəl	Rekvizitlərin adı	Rekvizitin kodu	Funksional asılılıqlar
<i>QRUP</i>	Qrupun nömrəsi	QN	<pre> graph TD QN --> SAY QN --> OBAL SAY <--> OBAL </pre>
	Tələbələrin sayı	SAY	
	Qrupun orta balı	OBAL	

Bundan sonra isə aşağıdakılar müəyyənləşdirilməlidir:

1. Predmet sahəsinin təsviri və sənədlərin (cədvəllərin) formasının təhlilinə əsasən rekvizitlər arasında funksional əlaqələr müəyyənləşdirilir. Hər bir rekvizitin digərlərindən funksional asılılığı təyin olunur və əgər belə asılılıq olarsa, həmin rekvizitdən digərinə (açar rekvizitə) xətt çəkilir və asılı rekvizitə tərəf oxla istiqamətləndirilir.

2. Bütün rekvizitləri təsviredici və açar rekvizitləri kimi iki qrupa bölərək onlar arasında əlaqələri müəyyən etmək. Bunun üçün rekvizitlər arasında aşkar edilmiş funksional asılılıqlar təhlil edilməlidir. Birinci qrupa digər rekvizitlərdən asılı olan rekvizitləri daxil etmək və onların hər biri üçün hansı rekvizitlərdən asılı olmasını göstərmək lazımdır. Asılı olan rekvizitlər açar rekvizitləri adlanan ikinci qrupu əmələ gətirir. Əgər tranzitiv asılılıq mövcud olarsa, onda rekvizitlər eyni zamanda həm asılı, həm də açar olur, ona görə də onlar ayrı–ayrı qruplara daxil edilir.

3. İnformasiya obyektlərini yaratmaq. Bunun üçün bir və ya bir neçə rekvizitlərdən eyni qayda ilə asılı olan rekvizitləri təsviredici rekvizitlər qrupuna birləşdirmək lazımdır. Hər qrupa açar rekvizitləri üçün ümumi olan rekvizitlər daxil edilir.

İnformasiya obyekti seçildikdən sonra onun yekun təsvirini yaratmaq lazımdır.

1.4. Verilənlər bazası cədvəlləri

VB–də verilənlər cədvəllərdə saxlanır. Ona görə də VB–də saxlanılan informasiyanın ölçü vahidi cədvəldir. Hər bir cədvəl sətir və sütunlardan ibarət olur. Sətirlər hər hansı bir hadisənin obyektinin nüsxəsinə, sütunlar isə hər hansı bir hadisənin atributlarına (əlamətlərinə, xarakteristikalarına, parametrlərinə) uyğun gəlir. Cədvəl 1.2–də “*Tədris prosesi*” predmet sahəsinin *QRUP* cədvəlinin (obyekt, hadisə) nümunəsi göstərilmişdir. Bu cədvəlin sütunlarında “*Qrupun nömrəsi*”, “*Tələbələrin sayı*” və “*Qrupun orta balı*” kimi parametrlər təsvir olunmuşdur. Sətirlər isə konkret hadisə – qrup haqqında verilənləri əks etdirir. VB termini ilə desək sütunlara – *sahələr*, sətirlərə isə *yazılar* deyilir.

Cədvəl 1.2. *QRUP* cədvəli

Qrupun nömrəsi	Tələbələrin sayı	Qrupun orta balı
50	25	92
58	20	94
62	17	84
65	12	96
80	15	87

VB–nin ayrı–ayrı cədvəlləri arasında əlaqələr mövcud olur. Məsələn, yuxarıdakı cədvəldə “*Qrupun nömrəsi*” sahəsi cədvəl 1.3–də göstərilmiş cədvəldə tamamlana bilər.

Cədvəl 1.3. *TƏLƏBƏ* cədvəli nümunəsi

Qrupun nömrəsi	Qrupda tələbənin nömrəsi	Tələbənin soyadı, adı, atasının adı	Təvəllüdü	Tələbənin ünvanı	Daxil olduğu bal
50	1	Abdullayev Y	03.02.96	Bakı, S. Vurğun, 65	677
50	2	Manafov K	07.08.97	Bakı, Bülbül pr, 10	660
62	1	Qafarov Z.	01.09.96	Quba, Amsar kəndi	653
62	2	Qulamov A	17.12.98	Gəncə, H. Əliyev, 63	695
80	1	Arazov M	25.04.97	Astara, H. Aslanov, 17	643
55	3	Qurbanov M	12.12.99	Qazax, M. Vidadi, 56	570
66	2	Hüseynov F	01.05.98	Tovuz, M. Sabir, 12	550
70	1	Bağirov K	15.06.97	Gəncə, N. Gəncəvi, 14	697

Gördüyümüz kimi, bu cədvəllər bir–biri ilə “*Qrupun nömrəsi*” sahəsi ilə əlaqəlidir və ona görə də belə cədvəllərə *relyasiyalı cədvəllər* deyilir. Əlaqəli cədvəllər bir–biri ilə *əsas (master)* və *tabe olan (detail)* prinsipi ilə qarşılıqlı əlaqədə olur. Əsas cədvələ adətən *valideyn* cədvəl, tabe olan cədvələ isə *varis (törəmə)* cədvəl deyirlər.

1.4.1. İlkın açarlar

Hər bir cədvəldə ilkın açar ola bilər. İlkın açar bir və ya bir neçə sahədən ibarət ola bilər. İlkın açarın yaradılmasından əsas məqsəd yazını birqiymətli təyin etməkdir. VB–də ilkın açarın qiyməti unikal olmalıdır, yəni təkrarlanmamalıdır, başqa sözlə, açar olan sahələrdə eyni qiymətli iki və daha artıq yazılar ola bilməz.

İlkın açarın mövcudluğu cədvəllər arasında əlaqə, münasibət yaratmağa imkan verir. *QRUP* cədvəlində “*Qrupun nömrəsi*” sahəsi belə açar ola bilər. Bu açarın köməyi ilə iki cədvəl arasında əlaqə yaradaraq aydınlaşdırıla bilər ki, məsələn, *50–ci qrupda tələbə Abdullayev Y.K. 03.02.1996–cı ildə anadan olmuşdur, “Bakı, S.Vurğun, 65” ünvanında yaşayır və ali məktəbə 677 balla daxil olmuşdur.*

İlkın açarlar elə sahələrə tətbiq edilməlidir ki, bu sahələrdə yazılar təkrarlanmasın. Məsələn, tələbələrin, əməkdaşların və s. soyadları, adları açar sahəsi kimi istifadə edilə bilməz, çünki, eyni soyadlı və adlı çoxlu tələbə, əməkdaş ola bilər. Məsələn, *QRUP* cədvəlində *Tələbələrin sayı, Orta bal* sahələri, *TƏLƏBƏ* cədvəlində *Soyadı, ünvan* sahələri açar sahəsi ola bilməz.

VB–də *ikinci açardan* da istifadə edilir. İlkın açardan fərqli olaraq ikinci açar elə sahələr üçün yaradılır ki, orada yazılar təkrarlana bilər, başqa sözlə, ikinci açar unikal açar olmur. İlkın açarın qiyməti ilə yazının yalnız yeganə bir nüsxəsini tapmaq mümkün olduğu halda, ikinci açarla bir neçə nüsxə tapıla bilər. Bu açarların fərqləri məhz bundadır.

Elə hallar ola bilər ki, ilkın açar kimi bir neçə sahədən istifadə edilsin. Məsələn, şəxsiyyət vəsiqəsinin seriyası və nömrəsi sahələrini ayrı–ayrılıqda ilkın açar kimi istifadə etmək olmaz. Çünki, eyni seriyalı və eyni nömrəli (hər birini ayrılıqda götürdükdə) istənilən qədər vəsiqələr mövcuddur. Lakin bu iki sahəni bir ilkın açar kimi istifadə etdikdə yazı nüsxəsi birqiymətli təyin olunacaqdır (çünki eyni seriya və nömrəli ikinci bir şəxsiyyət vəsiqəsi mövcud ola bilməz).

1.4.2. İndekslər

VB cədvəllərinin hər birində yüzlərlə, minlərlə yazılar ola bilər. Adətən bu yazılar cədvəldə ixtiyari ardıcılıqla, nizamsız qaydada yerləşir. Bu yazılar içərisindən hər hansı bir kriteriyə uyğun parametri axtardıqda böyük vaxt tələb olunur. Belə axtarışı sürətləndirmək və ümumilikdə VB–nin məhsuldarlığını artırmaq məqsədi ilə indekslərdən istifadə edilir. İndeksin tətbiqi nəticəsində yazılar artma və ya azalma sırası ilə düzülür. İndeks bir və ya bir neçə sahəyə tətbiq oluna bilər. İndeks sahələri adətən cədvəllərə sorğular edildikdə yaradılır. VB–də indekslərsiz də ötürmək olar, lakin yuxarıda qeyd etdiyimiz kimi, indekslərin tətbiqi VB–nin səmərəliliyini, məhsuldarlığını əhəmiyyətli dərəcədə artırır. Ona görə də biz təkid edirik ki, gələcəkdə VB yaratdıqda indekslərdən hökmən istifadə edəsiniz.

Təcrübələr göstərir ki, tələbələr indeksin mahiyyətini, əhəmiyyətini tam dərk etmirlər. Ona görə də bu sualı ətraflı aydınlaşdırmaq. Tutaq ki, cədvəl 1.3–də göstərilmiş *TƏLƏBƏ* cədvəlində bizi orta balı 697 olan tələbə maraqlandırır. Adi halda (indeks tətbiq edilmədikdə) biz bu tələbəni aşağıdakı qayda ilə axtaracağıq:

1. cədvəlin birinci sətirini seçirik;
2. bu sətirdəki balı yadda saxlayırıq;
3. bu qiyməti axtardığımız qiymətlə müqayisə edirik: əgər onlar bərabər deyilsə, növbəti sətərə keçirik.

Bizim cədvəl üçün belə müqayisələrin sayı 8–ə bərabər olacaqdır. Əgər belə sətirlərin sayı 100, 1000, 10000 olarsa, təsəvvür edirsinizmi nə qədər müqayisələr aparmaq lazımdır?

İndi isə indeksin tətbiqi ilə axtarışın necə yerinə yetirilməsinə baxaq. İzahat xatirinə cədvəlimizi cədvəl 1.4–də göstərilən formada yazaq. İndi bu cədvəlin *Bal* sütununa indeks tətbiq edək, onda yeni cədvəl (cədvəl 1.5) alacağıq.

Massivlərdə, siyahılarda, statistik seçmələrdə və s. hər hansı bir parametrin seçilməsi üçün müəyyən axtarış üsullarından istifadə edilir. Belə axtarış üsulları çoxdur, biz ikili (binar) axtarış üsulunun tətbiqinə baxaq. *İkili axtarış üsuluna* görə baxdığınız

seçməni iki bərabər hissəyə bölək. Onda cədvəl 1.6 və cədvəl 1.7-ni alırıq.

Cədvəl 1.4. İlkin cədvəl

Sıra №	Bal
1	677
2	660
3	653
4	695
5	643
5	570
7	550
8	697

Cədvəl 1.5. İndeks tətbiq edildikdən sonrakı cədvəl

Sıra №	Bal
1	550
2	570
3	643
4	653
5	660
6	677
7	695
8	697

Cədvəl 1.6. Birinci hissə

Sıra №	Bal
7	550
6	570
5	643
3	653

Cədvəl 1.7. İkinci hissə

Sıra №	Bal
2	660
1	677
4	695
8	697

Sonra 697 ballı tələbənin hansı hissəyə aid olduğunu yoxlayırıq. Bunun üçün birinci hissənin sonuncu yazısını (653) ikinci hissənin isə birinci yazısını (660) axtarılan ədədlə (697) müqayisə edirik. Əgər axtarılan ədəd bu iki ədədin arasında olarsa, onda axtarış dayandırılır – deməli, belə ballı tələbə yoxdur. Əks halda isə biz axtarışı ikinci hissədən davam etdiririk (çünki, $697 > 660$). Ona görə də cədvəlin ikinci hissəsini yenidən yarı bölürük (cədvəl 1.8 və cədvəl 1.9).

Cədvəl 1.8. Yenidən yarıya bölünmüş cədvəl

Sıra №	Bal
2	660
1	677

Cədvəl 1.9. Yenidən yarıya bölünmüş cədvəl

Sıra №	Bal
4	695
8	697

Aydındır ki, biz axtarışı ikinci seçmədə davam etdirməliyik, həmin seçməni yenidən iki hissəyə böldükdə hər seçmədə yalnız bir element qalır ki, onlardan biri bizim axtardığımız tələbəyə uyğun gəlir.

Gördüyümüz kimi, biz 8 əməliyyat əvəzinə cəmi 3 əməliyyatla axtarılan nəticəni əldə etdik!

İndekslər eyni zamanda bir neçə sahələrə də tətbiq oluna bilər. Vacib deyildir ki, indeksli sahələr hökmən ədəd mahiyyətli olsun, yazıları da əlifba sırası ilə sıralamaq olar.

İndekslərin tətbiqinin çatışmayan cəhəti, zənnimizcə, ondan ibarətdir ki, VBİS indekslər üçün ayrı fayl yaradır.

1.4.3. Cədvəl formatları

VB yaratdıqda VB-nin özünün və onun cədvəllərinin adlarında Azərbaycan dilinin, kiril əlifbasının və s. simvollarından istifadə oluna bilər. Lakin elə hallar var ki, adların belə formatları sonralar problemlər yaradır (məsələn, SQL-sorğu dilini istifadə etdikdə, kliyent-server arxitekturasında və s.). Ona görə də cədvəl adları kimi klaviaturanın yol verdiyi latın əlifbası simvollarından istifadə etmək daha məqsədəuyğundur. Sadəcə olaraq belə formal qaydalara da əməl etmək olar: cədvəl formasının qarşısına “*F*” hərfi, sorğular qarşısına “*S*” hərfi, hesabatlar qarşısına “*H*” hərfi və s. yazmaq.

Verilənlər bazasının və ya onun cədvəllərinin adlarının genişlənməmiş hissələri VBİS-dən asılı olaraq müxtəlif olur.

Ms Access VBİS-də yaradılan baza yalnız bir fayldan ibarət olur. Bu faylın genişlənməmiş hissəsi Ms Access 2003 və daha aşağı versiyalarda *.mdb*, Ms Access 2007 və 2010 versiyalarında isə *.accdb* olur.

dBase VBİS fərdi kompyuterlər üçün nəzərdə tutulmuş ilk proqramdır. Belə VB-ə daxil olan cədvəllərin formatları cədvəl 1.10-da göstərilmişdir.

Cədvəl 1.10. **dBase** VB–nin cədvəllərinin formatları

Faylın genişlənmiş hissəsi	Faylın məzmunu
.dbf .dbt	<i>dBase</i> – cədvəl faylları Böyük ikilik verilənlər (BLOB – <i>Binary Large Object</i>), o cümlədən <i>MEMO</i> – və <i>OLE</i> –sahə faylları
.mdx .ndx	<i>dBase</i> tərəfindən dəstəklənən indeks faylları Dəstəklənməyən indeks faylları, belə fayllar proqram kodları ilə yaradılır

dBase VB cədvəllərinin sahələrinin adları hərflə başlamaqla hərf–rəqəm simvollarından (10–dan çox olmamaqla) ibarət olmalıdır; xüsusi simvollar və boşluq (probel) simvoluna yol verilmir. *dBase* VB–nin çatışmayan cəhəti ondan ibarətdir ki, cədvəllər arasında əlaqə tamlığına nəzarət yoxdur və verilənlər mühafizə olunmur.

İndi isə **Paradox** VB–ə baxaq. O, *dBase* ilə müqayisədə daha çox inkişaf etmişdir. *Paradox* cədvəllərinin əsas genişlənmiş hissələri cədvəl 1.11–də göstərilmişdir.

Cədvəl 1.11. **Paradox** VB–nin cədvəllərinin genişlənmiş hissələri

Faylın genişlənmiş hissəsi	Faylın məzmunu
.db .mb .px .xg* və .yg* .val	<i>Paradox</i> – cədvəl faylları Böyük ikilik verilənlər (<i>BLOB</i>) Açar (əsas indeks) faylları İndeks faylları Daxil edilən verilənlərin tiplərinin və əlaqə tamlığının yoxlanılması üçün parametrlər faylları
.tv və .fam	Database Desktop əlavəsində cədvəllərin təsvir olunma formatları
.net	Şəbəkədə cədvəllərə müraciətə nəzarət üçün tətbiq edilən fayllar

Paradox VB cədvəllərinin sahələrinin tiplərinə, sahə adlarına və s. qoyulan tələblərə kitabın Delphi–yə aid hissəsində ətraflı baxacağıq.

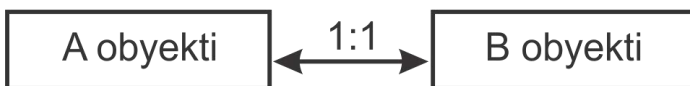
1.5. Cədvəllərarası relyasiya münasibətləri

VB–nin cədvəlləri arasında əlaqə yaratmaq üçün relyasiya münasibətlərindən istifadə edilir. Cədvəllərarası əlaqələr açar sahələri vasitəsi ilə yerinə yetirilir. Unutmayın ki, yalnız eyni tipli sahələr bir–biri ilə əlaqələndirilə bilər.

Relyasiyalı cədvəllər arasında “*birin–birə*” (**1:1**), “*birin–çoxa*” (**1:M**) və “*çoxun–çoxa*” (**M:N**) münasibətləri yaradıla bilər.

1.5.1. Birin–birə münasibəti

Bu əlaqə qrafik olaraq şəkil 1.5–də göstərilmişdir.



Şəkil 1.5. **1:1** münasibəti

Birin–birə münasibəti o zaman yaradılır ki, əsas cədvəldə bir yazıya varis cədvəldə bir yazı uyğun gəlir. Bu münasibət adətən az istifadə edilir. Onu o zaman istifadə edirlər ki, bir cədvəldə həddən artıq ikinci dərəcəli məlumat əks olunur, ona görə də bu cədvəli iki cədvəle bölüb onlar arasında **1:1** münasibəti yaradırlar (cədvəl 1.12).

1:1 münasibəti ona gətirir ki, bir neçə cədvəldən oxuma əməliyyatı icra olunur, bu isə öz növbəsində zəruri informasiyanı əldə etməyi ləngidir. Bundan başqa **1:1** əlaqəli cədvəlləri olan VB normallaşdırma tələblərinə cavab vermir.

Cədvəl 1.12. 1:1 əlaqəsində olan cədvəllər

Əməkdaşlar cədvəli				Əməkdaşlar haqqında məlumat cədvəli		
Nö	S.A.A.	Vəzifəsi	Şöbə		Nö	Təvəllüdün sayı
1.	İsmayılov A.A.	Mühəndis	10	→	1.	1952 3
2.	Qocayev B.S.	Mühəndis	20	→	2.	1960 2
3.	Qarayev S.T.	Mühasib	10	→	3.	1970 1
...	→

1.5.2. Birin – çox münasibəti

Qrafik olaraq belə əlaqə şəkil 1.6 –da göstərilmişdir.



Şəkil 1.6. 1:M münasibəti

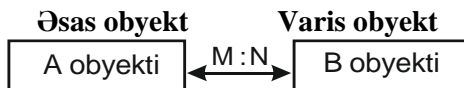
Birin – çox münasibəti ən çox tətbiq edilən əlaqədir. Belə əlaqədə əsas cədvəlin bir sətrinə varis (təbə olan) cədvəlin bir neçə sətri uyğun gəlir. Məsələn, cədvəl 1.2-də göstərilən *QRUP* cədvəlinin “*Qrupun nömrəsi*” sahəsinin hər bir sətrinə cədvəl 1.3-də göstərilmiş *TƏLƏBƏ* cədvəlinin eyniadlı sahəsində bir neçə sətir uyğun gələ bilər. Sizin diqqətinizi “*gələ bilər*” şərti fəlinə

yönəltmək istəyirik: bu o deməkdir ki, bu potensial olaraq belədir, ola bilər ki, əsas cədvəldə yazı olsun, varis cədvəldə isə hal-hazırda ona uyğun yazı olmasın, necə ki, 58 və 65-ci qruplar yoxdur (gələcəkdə bu yazılar ola bilər).

1:M əlaqəsinin iki halı vardır: birinci halda sərt əlaqəli cədvəllər yaradılır, yəni əsas cədvəlin hər bir yarısına varis cədvəlin hökmən bir yazısı uyğun gəlməlidir; ikinci halda isə bizim misalımızda göstərdiyimiz kimi, əsas cədvəlin bəzi yazılarına varis cədvəlin yazıları uyğun olmaya bilər. Belə hallar **1:1** əlaqəsinə də aiddir.

1.5.3. Çoxun – çoxə münasibəti

Belə münasibətin qrafik təsviri şəkil 1.7-də göstərilmişdir



Şəkil 1.7. **M:N** münasibəti

Şəkil 1.8-də isə çoxun–çoxə əlaqəli cədvəllər göstərilmişdir.

QRUP və *FƏNLƏR* cədvəli

Qrup №	Fənn	Müəllimin №
78	Proqramlaş.	10
62	Proqramlaş.	11
78	Sistem nəz.i	10
82	Fəlsəfə	48
78	Sosiologiya	48
...

MÜƏLLİM cədvəli

Müəllimin №	S.a.a.	Kafedra
10	Əfəndiyev A	100
11	Abdullayev Q	101
48	Məmmədov Z	102
60	Əliyev A.	103
70	Qulamov Q	104
...

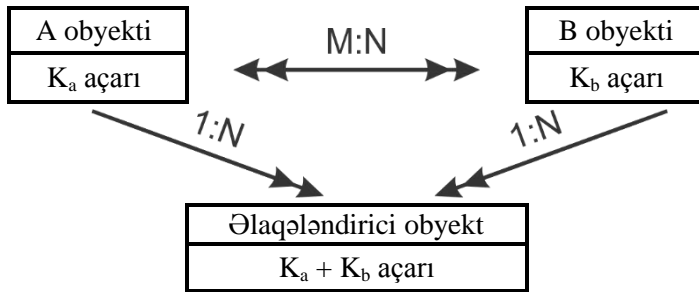


Şəkil 1.8. Çoxun–çoxa əlaqəsi

Hər bir qrupa bir neçə müəllim uyğun gəlir. Hər bir müəllim isə həm bir neçə fənni tədris edə bilər, həm də müxtəlif qruplarda dərs deyə bilər.

Bəzi VBİS–lər **M:N** əlaqəsini dəstəkləmir. Ona görə də belə əlaqəni əlavə köməkçi cədvəl vasitəsi ilə **1:M** əlaqəsinə gətirmək olar. Bu köməkçi cədvələ əlaqələndirici cədvəl deyilir.

Şəkil 1.9–da **M:N** münasibətinin **1:M** münasibətinə gətirilməsinin qrafik təsviri göstərilmişdir.

Şəkil 1.9. **M:N** əlaqəsinin **1:N** əlaqəsinə gətirilməsi

1.5.4. Bir cədvəlin yazıları arasında əlaqə

Eyni bir cədvəlin müxtəlif yazıları arasında da əlaqə ola bilər.

Tutaq ki, relyasiyalı VB–də ixtiyari səviyyələrdən ibarət ağacvari strukturlu verilənləri, məsələn, təşkilatın aşağıdakı strukturunu (şəkil 1.10) yadda saxlamaq lazımdır.

Bu halda elə cədvəl yaradılmalıdır ki, təşkilatın hər bölməsinə bir yazı uyğun gəlsin (cədvəl 1.13).

Avtomatlaşdırma departamenti
Texniki idarə
<i>Şəbəkəyə xidmət şöbəsi</i>
<i>Təmir şöbəsi</i>
<i>ATS</i>
Proqram sistemləri idarəsi
<i>İstismar şöbəsi</i>
<i>İnformasiya qrupu</i>
<i>İnzibati qrup</i>
<i>Dispetçer bürosu</i>
<i>Tədqiqat şöbəsi</i>

Şəkil 1.10. Təşkilatın strukturu

Cədvəl 1.13. Bir cədvəlin yazıları arasında əlaqə

Bölmənin nömrəsi	Bölmənin adı	Yuxarı səviyyədə yerləşən bölmənin nömrəsi
1	Avtomatlaşdırma departamenti	
2	Texniki idarə	1
3	Proqram sistemləri idarəsi	1
4	Şəbəkəyə xidmət şöbəsi	2
5	Təmir şöbəsi	2
6	ATS	2
7	İstismar şöbəsi	3
8	Tədqiqat şöbəsi	3
9	İnformasiya qrupu	7
10	İnzibati qrup	7
11	Dispetçer bürosu	10

1.5.5. İstinad tamlığı

Yuxarıda qeyd etdiyimiz kimi, VB-də ən çox rast gəlinən əlaqə birin-çoxa əlaqəsidir. Belə əlaqəyə nümunə olaraq şəkil 1.11-də göstərilən cədvəllərə baxaq.

<i>QRUP</i> cədvəli			<i>TƏLƏBƏ</i> cədvəli					
Qrup №	Tələbələrin sayı	Orta bal	Qrup №	Qrupda tələbənin №	S.A.A.	Təvəllüdü	Ünvanı	Orta bala
50	25	85	50	1	Rzayev
60	20	92	50	2	Ayazov
70	18	83	50	3	Əliyev
75	15	79	60	1	Cəfərov
			70	5	Qocayev
			75	14	Qulamov

Şəkil 1.11. Cədvəllərarası əlaqə

Şəkildən göründüyü kimi, *TƏLƏBƏ* varis cədvəli *QRUP* əsas cədvəli ilə *Qrup №* sahəsi ilə əlaqələndirilmişdir. *Qrup №* sahəsi hər iki cədvəl üçün əlaqələndirici sahədir.

Bu cədvəllərə verilənlər daxil edildikdə və ya cədvəl verilənlərini dəyişdirdikdə əsas və varis cədvəllərin yazıları arasında əlaqə itə bilər. Bu aşağıdakı iki halda mümkündür:

- əsas cədvəldə əlaqə sahəsinin qiyməti dəyişdirilir, lakin varis cədvəldə müvafiq yazılar dəyişmir;

- varis cədvəldə əlaqə sahəsinin hər hansı bir yazısı dəyişdirilir, lakin müvafiq dəyişiklik əsas cədvəldə aparılmır.

QRUP cədvəli			TƏLƏBƏ cədvəli					
Qrup №	Tələbələrin sayı	Orta bal	Qrup №	Qrupda tələbələrin №	S.a.a.	Təvəllüd ü	Ünvanı	Orta bal
80	25	85	50	1	Rzayev
60	20	92	50	2	Ayazov
70	18	83	50	3	Əliyev
75	15	79	60	1	Cəfərov
			70	5	Qocayev
			75	14	Qulamov

Şəkil 1.12. VB tamlığının pozulması

Birinci hala baxaq. Şəkil 1.12-də *QRUP* cədvəlinin *Qrup №* sahəsində 50 ədədi 80 ilə əvəz edilmişdir, lakin *TƏLƏBƏ* cədvəlində yazıların qiymətləri olduğu kimi qalmışdır. Bunun nəticəsində aşağıdakı anlaşılmazlıqlar baş verir:

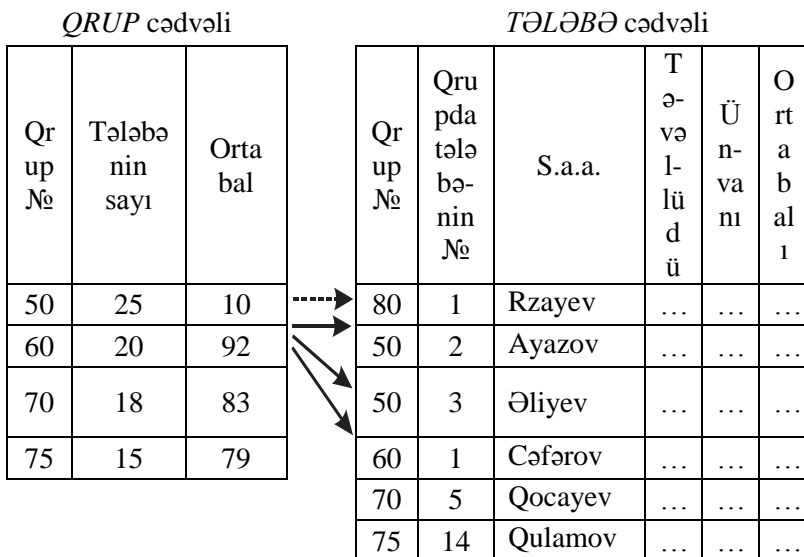
- *TƏLƏBƏ* varis cədvəlində 80-ci qrup haqqında heç bir məlumat yoxdur;

- *TƏLƏBƏ* varis cədvəlinin yazılarında 50-ci qrup haqqında məlumat var, lakin bu barədə əsas cədvəldə heç bir məlumat yoxdur.

İkinci hala baxaq. *TƏLƏBƏ* cədvəlinin əlaqə sahəsinin yazılarından birini dəyişib 50 əvəzinə 80 yazaq (şəkil 1.13). Bu halda da aşağıdakı anlaşılmazlıqlar baş verəcəkdir:

- *TƏLƏBƏ* varis cədvəlində 50-ci qrup haqqında tam məlumat yoxdur;

- *TƏLƏBƏ* cədvəlinin yazılarının birində 80-ci qrup haqqında məlumat olduğu halda, əsas cədvəldə bu qrup ümumiyyətlə yoxdur.



Şəkil 1.13. VB təhliliyənin pozulmasının başqa halı

Hər iki halda biz verilənlər bazasının istinad təhliliyənin pozulmasını müşahidə edirik. Bu o deməkdir ki, bazada saxlanılan informasiya dürüst deyildir.

VBİS adətən istinad təhliliyənin pozulmasının qarşısını alır. İstinad təhliliyi dedikdə bütövlükdə VB–nin ayrı–ayrı cədvəlləri arasında əlaqələr yığılı başa düşülür. Bu əlaqələrdən heç olmazsa biri pozularsa, onda VB–nin istinad təhliliyənin pozulmuş sayılır və belə VB–də saxlanılan informasiya etibarsız hesab olunur. İstinad təhliliyənin bütün maliyyətinə tam qavramaq üçün belə misala da baxaq. Təsəvvür edək ki, hər hansı bir tələbə bu və ya digər səbəbdən universitetdən xaric edilmişdir. Biz onu tələbələrin siyahısının saxlandığı hər hansı bir cədvəldən pozuruq, digər cədvəllərdən isə pozmururuq və o, digər cədvəllərdə tələbə kimi universitetdə qalır (əlbəttə, VB–də). Bu isə qətiyyənlə yolverilməzdir.

İstinad təhliliyəni qorumaq üçün aşağıdakı əməliyyatları yerinə yetirmək lazımdır:

- əsas cədvəlin əlaqə sahəsinin yazılarını dəyişdirdikdə varis cədvəlin də müvafiq yazılarını sinxron dəyişdirmək lazımdır;

- əsas cədvəldə yazıları pozduqda varis cədvəldə də uyğun yazıları sinxron pozmaq lazımdır.

Əsasən varis cədvəllərdə yazıların eyni zamanda dəyişdirilməsi və ya pozulması *kaskad dəyişdirmə* və *kaskad pozulma* adlanır.

İstinad tamlığını realizasiya etmək üçün, adətən, varis cədvəldə *xarici açar* yaradırlar ki, bu açar varis cədvəlin əlaqə sahəsinə daxil olur. Bu açar varis cədvəl üçün ilkin açar olur və ona görə də əsas cədvəlin ilkin açarı ilə eyni olmalıdır.

1.6. Verilənlər bazasının normallaşdırılması

Artıq bizə məlumdur ki, VB–nin layihələndirilməsinin ən əsas məsələsi verilənlərin strukturunun müəyyənləşdirilməsi, yəni cədvəllərin tərkibini və onlar arasında əlaqələri müəyyən etməkdən ibarətdir. Bu struktur isə səmərəli olmaqla yanaşı aşağıdakıları təmir etməlidir:

- verilənlərə cəld müdaxilə etmək;
- verilənlərin təkrarlanmasına yol verməmək;
- verilənlərin tamlığına nail olmaq.

Verilənlərin strukturunu müəyyənləşdirmək üçün üç yanaşmadan istifadə etmək olar:

1. Həll ediləcək məsələnin obyektləri haqqında informasiyanı bir cədvəldə toplamaq və sonradan, normallaşdırma prinsipləri əsasında, həmin cədvəli bir–biri ilə qarşılıqlı əlaqəli bir neçə cədvəle bölmək;

2. Sistem haqqında bilikləri (ilkin verilənlərin tipləri, onların əlaqəsi) tərtib etmək, sonra isə ya CASE vasitələrinin və ya VBİS–in köməyi ilə VB sxemini yaratmaq;

3. Sistemli təhlil əsasında informasiyanı strukturlaşdırmaq.

VB–nin layihələndirilməsi klassik üsulla yerinə yetirilə bilər, yəni VB–ni layihələndirən şəxs informasiya obyektlərini özü seçir və sonra əl ilə onu verilənlərin tələb olunan strukturuna gətirir. Bununla yanaşı, layihələndirmə üçün **CASE** – **sistem** adlandırılan sistemdən də istifadə etmək olar ki, bu sistem nəinki yalnız VB–ni

layihələndirmə prosesinin, habelə bütövlükdə informasiya sisteminin işlənməsini avtomatlaşdırmağa imkan verir.

VB–nin layihələndirilməsinin əsas məsələlərindən biri VB–nin normallaşdırılmasıdır. VB–nin normallaşdırılması – VB–də informasiya artıqlığına yol verilməməsidir. Normallaşdırma metodu verilənlərin relyasiya modelinin kifayət qədər mürəkkəb nəzəriyyəsinə əsaslanır.

VB–nin strukturunun işlənməsində verilənlərin artıqlığı (izafiliyi) və anomaliyası kimi problemlərlə qarşılaşa bilərik.

Verilənlərin izafiliyi dedikdə VB–də verilənlərin təkrar edilməsi başa düşülür. Bu zaman verilənlərin sadə (qeyri–izafi) və izafi təkrarlaması baş verə bilər.

Verilənlərin izafiliyi öz növbəsində müxtəlif *anomaliyaya* – VB–nin tamlığının pozulmasına gətirir. Anomaliya üç növ olur:

- pozulma;
- yeniləşdirmə;
- daxil etmə.

Qeyri–izafi təkrarlama təbii prosesdir və yolveriləbiləndir, buna misal kimi cədvəl 1.14–də təsvir edilmiş verilənləri göstərə bilərik.

Cədvəl 1.14. Qeyri–izafi təkrarlama

Müəllim	Telefon
Qurbanov A.M.	125
Quliyev T.R.	125
Arazov A.N.	335
Dəmirov Z.K.	336
Şeydayev Ə.M.	125

Cədvəldən görünür ki, üç müəllimin telefon nömrəsi eynidir, bu çox təbiidir, ola bilər ki, müəllimlər eyni kabinetdə otururlar. Beləliklə, cədvəldə telefon nömrələri təkrarlanır, lakin hər bir müəllim üçün bu nömrə unikaldir. Əgər bu təkrarlanan nömrələrdən birini pozsaq (cədvəlin uyğun sətrini), onda *Qurbanov A.M.*, *Quliyev T.R.* və *Şeydayev Ə.M.* haqqında informasiya itəcəkdir ki, buna da *pozulma anomaliyası* deyilir.

Əgər kabinetdə telefon nömrəsi dəyişərsə, onda bu dəyişikliyi bütün müəllimlər üçün yerinə yetirmək lazımdır. Əgər

hər hansı bir müəllim üçün bu dəyişikliyi etməsək, onda *yeniləşdirmə anomaliyası* baş verəcəkdir.

Daxiletmə anomaliyası isə cədvələ yeni sətir (yazı) əlavə etdikdə baş verir. Belə ki, sahəyə elə verilən daxil edilə bilər ki, o yolveriləbilən diapazona daxil olmasın və ya sahəyə hökmən verilən daxil edilməlidir (sahə boş qala bilməz), biz isə onu boş saxlayırıq.

İndi isə verilənlərin izafi təkrarlanmasına aid misala baxaq. Yuxarıdakı cədvələ müəllimlərin oturduqları kabinetlərinin nömrələrindən ibarət daha bir sütun əlavə edək (cədvəl 1.15)

Cədvəl 1.15. Verilənlərin izafi təkrarı

Müəllim	Kabinet	Telefon
Qurbanov A.M.	10	125
Quliyev T.R.	10	—
Arazov A.N.	20	335
Dəmirov Z.K.	18	336
Şeydayev Ə.M.	10	—

Cədvəldə eyni kabinet üçün yalnız bir nömrə, yəni müəllim *Qurbanov A.M.*-in telefon nömrəsi göstərilmişdir. Digər eyni telefon nömrəli müəllimlər üçün sahədə defis işarəsi qoyulmuşdur (əksər VBİS-lər buna yol vermir, ona görə də “—” əvəzinə **Nil** yazılır). Cədvəlin belə strukturu aşağıdakı problemləri yaradır:

- ixtiyari müəllimin telefonunu tapmaq üçün gərək digər sütunda yerləşən kabinetin nömrəsi üzrə axtarış aparasan;
- sahələrdə defis işarəsinin olub-olmamasından asılı olmayaraq bu cədvəl yaddaşda eyni yer tutacaqdır;
- cədvəldən kabinetin telefon nömrəsinin göstərildiyi sətir (*Qurbanov A.M.*) pozduqda yerdə qalan müəllimlərin telefon nömrələri haqqında informasiya itəcəkdir.

Əgər defis əvəzinə telefon nömrələri yazsaq, izafi təkrarlama yenə də qalacaqdır. Bu təkrarlama yalnız cədvəli bir–biri ilə

əlaqələndirilmiş iki əsas və varis cədvələ bölməklə azad olmaq olar (cədvəl 1.16 və 1.17).

Cədvəl 1.16. MÜƏLLİMLƏR

Müəllim	Kabinet
Qurbanov A.M.	10
Quliyev T.R.	10
Arazov A.N.	20
Dəmirov Z.K.	18
Şeydayev Ə.M.	10

Cədvəl 1.17.
KABİNET

Kabinet	Tel efon
10	125
20	335
18	336

Bu cədvəllər bir-biri ilə *Kabinet* sahəsi ilə əlaqələndirilmişdir. Hər hansı müəllimin telefon nömrəsini tapmaq üçün əsas cədvəldən onun soyadına əsasən kabinetin nömrəsi, kabinetin nömrəsinə əsasən isə varis cədvəldən telefon nömrəsi tapılır.

Əsas cədvəlin bir neçə varis cədvələ bölünməsi prosesi VB–nin *normallaşdırılması* adlanır. Bu halda verilənlərin izafiliyi azalır, lakin cədvəl verilənlərinə müraciət vaxtı artır.

Cədvəllərin normallaşdırılması bir neçə addıma yerinə yetirilir. Hər addımda bir normal formadan daha yüksək normalı formaya keçilir. Hər normal forma müəyyən tip funksional asılılıqları məhdudlaşdırır, uyğun anomaliyalı aradan qaldırır.

VB–nin beş normal forması mövcuddur:

- birinci normal forma;
- ikinci normal forma;
- üçüncü normal forma;
- gücləndirilmiş üçüncü normal forma və ya *Boys–Kodd* normal forması;
- dördüncü normal forma;
- beşinci normal forma.

Layihələndirmə VB–nin informasiya obyektlərinin və bu obyektlərin atributlarının müəyyənləşdirilməsindən başlayır. Atributların hamısı əvvəlcə bir cədvəldə yerləşdirilir, sonra isə ardıcıl olaraq normallaşdırılır. Təcrübə göstərir ki, adətən beş yox, birinci üç normal forma istifadə edilir.

VB–nin normallaşdırılmasını tədris prosesinin VB–nin yaradılması nümunəsində izah edək.

1.6.1. “Tədris prosesi” predmet sahəsinin informasiya–məntiq modeli əsasında normallaşdırma

Biz predmet sahəsi olaraq “*Tədris prosesi*” götürsək də, əslində, dekanlığın iş prosesinə aid VB–nin yaradılması ilə məşğul olacağıq. Məlumdur ki, dekanlığın əsas fəaliyyəti qruplarla, tələbə və kafedralarla bağlıdır. Onda layihələndirəcəyimiz VB–nin rekvizitləri kimi *TƏLƏBƏ*, *QRUP* və *KAFEDRA* qəbul edə bilərik ki, onların atributları aşağıdakılar olacaqdır:

1. Kafedranın adı
2. Telefon
3. Kafedra müdiri
4. Kafedranın kodu
5. Kafedra müdirinin şəkli
6. Kafedranın müəllimləri
7. Müəllimlərin tabel nömrələri
8. Müəllimlərin elmi adları
9. Müəllimlərin elmi dərəcələri
10. Kafedrada tədris olunan fənlər
11. Məşğələ növü
12. Məşğələ növü üzrə saatlar
13. Qrupun nömrəsi
14. Qrupda tələbələrin sayı
15. Tələbənin soyadı, adı və atasının adı
16. Tələbənin təvəllüdü
17. Tələbənin ünvanı
18. Tələbənin müvəffəqiyyət balı
19. Semestrlərin sayı
20. Fənnin adı
21. Fənnin kodu
22. Tədris saatlarının miqdarı
23. Mühazirə saatları
24. Praktiki məşğələ saatları

25. Fənni tədris edən müəllim
26. Müəllimin elmi adı
27. Müəllimin elmi dərəcəsi
28. Kafedra
29. Tələbənin qiyməti
30. Daxil olduqda tələbənin balı

Bu əsas atributlardan başqa, predmet sahəsində zəruri olan ilkin sənədlər – verilənlər, uçot məlumatları aşkar edilməlidir.

Məlumat xarakterli informasiya “*Qrup tələbələrinin siyahısı*”, “*Kafedranın müəllimlərinin siyahısı*” və “*Tədris edilən fənlər*” kimi sənədlərdə təsvir olunur. Bu sənədlər şəkil 1.14–1.16–da göstərilmişdir.

___ №-li qrup tələbələrinin siyahısı

Tələbənin nömrəsi	Soyadı, adı, atasının adı	Təvəllüdü	Ünvanı	Daxil olduqda topladığı bal

Tələbələrin sayı /hesablanır/

Qrupa daxil olduqda orta balı /hesablanır/

Şəkil 1.14. “*Tələbələrin siyahısı*” sənədinin forması

Kafedranın müəllimlərinin siyahısı

Kafedranın adı _____

Kafedranın kodu _____ Telefon _____

Kafedranın müdiri _____

Tabel nömrəsi	Soyadı, adı, atasının adı	Elmi dərəcəsi	Elmi rütbəsi

Şəkil 1.15. “*Müəllimlərin siyahısı*” sənədinin forması

Qrupda keçilən məşğələlərin planı

Qrup № _____ semestr _____

Fənnin adı	Fənnin kodu	Müəllimin soyadı, adı, atasının adı	Müəllimin tabel №	Məşğələ növü	Saat

Şəkil 1.16. “Cari semestr də keçilən fənlərin siyahısı” sənədinin forması

Tədris prosesinə aid uçot informasiyası isə cari semestr üçün məşğələlərin keçirilməsi planında təsvir oluna bilər. Bundan başqa, ən vacib sənəd imtahan cədvəli olduğu üçün digər uçot sənədi – doldurulmuş imtahan cədvəli olacaqdır. Bu sənədin forması şəkil 1.17-də göstərilmişdir.

İmtahan cədvəli

Fənnin adı _____ Qrup _____

Müəllim _____

İmtahan növü _____ Tarix _____

Sıra №	Tələbənin soyadı, adı, atasının adı	Qiymət	Müəllimin imzası

Şəkil 1.17. “İmtahan cədvəli” sənədinin forması

Predmet sahəsinin sənədləri nəinki verilənlərin strukturunu aşkar etməyə imkan verir, onlar həm də giriş–çıxış formalarını yaratmaq üçün vacibdir.

İndi isə “*Kafedra müəllimlərinin siyahısı*” sənədinin rekvizitləri arasında funksional asılılıqları müəyyən edək. Bu rekvizitləri cədvəl 1.18–ə yazaq.

Cədvəl 1.18. Rekvizitlərin funksional asılılığı

Sənəd	Rekvizitlərin sərlövhəsi	Rekvizitin adı	Funksional asılılıq
Kafedranın müəllimlərinin siyahısı	Kaf. kodu	KAFK	
	Kaf. adı	KAFA	
	Telefon	TEL	
	Müdür	MUD	
	Foto	FOTO	
	Tabel №	TABN	
	Soyadı, adı, atasının adı	SAA	
	Elmi dərəcəsi	ED	
	Elmi adı	ER	

Sənədlərin təhlili göstərir ki, *Kafedranın adı* (KAFA), *Telefon* (TEL), *Müdür* (MUD) rekvizitləri təsviredici rekvizitlərdir, onların hər biri açar rekvizit olan *Kafedranın kodu* (KAFK) rekvizitindən asılıdır və bu açar rekvizit (KAFK) kafedranın müəllimləri siyahısının ümumi identifikatorudur.

Soyadı, adı, atasının adı (SAA), *Elmi dərəcəsi* (ED), *Elmi adı* (ER) rekvizitləri *Tabel №* (TABN) açar rekviziti ilə birmənalı təyin olunur.

KAFK və TABN funksional əlaqəsi ilə TABN əsas açar sahəsinin bir qiymətinə KAFK asılı rekvizitinin bir qiyməti uyğun gəlir. TABN rekviziti müəllim rekviziti üçün təsviredici rekvizitdir. Əgər belə bir əlaqə yaradılmasaydı, onda sənədin bütün rekvizitlər çoxluğu bir–biri ilə heç bir əlaqəsi olmayan iki alt çoxluğa bölünməli idi ki, bu da bir sənədin rekvizitləri üçün yolverilməzdir.

“*Kafedranın müəllimlərinin siyahısı*” sənədinin rekvizitləri arasında müəyyən edilmiş funksional asılılıqlar cədvəl 1.18 –də göstərilmişdir.

Qeyd edək ki, KAFK rekviziti eyni zamanda bir əlaqədə təsviredici, başqa bir əlaqədə isə açar rekvizit olur. Beləliklə, biz

burada tranzitiv asılılıqla rastlaşırıq (növbəti bölmələrdə bu asılılıqla tanış olacağıq). *KAFA* rekviziti *KAFK* rekviziti vasitəsi ilə *TABN* rekvizitindən asılıdır.

Rekvizitlərin funksional asılılığını araşdırmağa davam edək. Birinci asılı olan (təsviredici) *KAFK* rekviziti üçün *TABN* rekvizitini açar kimi təyin edirik. Sonra ikinci asılı (təsviredici) *KAFA* rekvizitini tapıb onun üçün *KAFK* açarını müəyyən edirik. Analoji olaraq *TEL* təsviredici rekvizitini tapıb, onun üçün *KAFK* açarını müəyyən edirik və s. Aşkar edilmiş təsviredici və açar rekvizitlərinin uyğunluğu cədvəl 1.19–da göstərilmişdir.

Cədvəl 1.19. Təsviredici və açar rekvizitlərinin uyğunluğu

Təsviredici (asılı) rekvizitlər	Açar rekvizitlər
KAFK	TABN
KAFA	KAFK
TEL	KAFK
MUD	KAFK
FOTO	KAFK
SAA	TABN
ED	TABN
ER	TABN

Eyni açar rekvizitlərindən asılı olan rekvizitləri qruplaşdıraraq və açar rekvizitlərinə uyğun olaraq bir informasiya obyektində birləşdirək. Gələcəkdə, Ms Access–də VB–ni yaratdıqda, bizə bu rekvizitlərin tipləri də lazım olacaqdır. Ona görə də cədvəl 1.20–də rekvizitlərin tipləri də göstərilmişdir. Tiplər haqqında növbəti fəsildə izahatlar verəcəyik.

Gördüyünüz kimi, “*Kafedranın müəllimlərinin siyahısı*” sənədini təhlil etdikdən sonra *KAFEDRA* və *MÜƏLLİM* kimi iki informasiya obyektini seçdik.

Analoji qayda ilə “*Qrup tələbələrinin siyahısı*” sənədini təhlil edərək *QRUP* və *TƏLƏBƏ* kimi məlumat xarakterli iki informasiya obyektini seçə bilərik. Qrup obyektini qrupda tələbələrin sayı və orta keçid balı ilə xarakterizə olunur. Qrupu birmənalı tanımaq üçün onun nömrəsindən istifadə edilir.

Cədvəl 1.20. Rekvizitlərin “Kafedra müəllimlərinin siyahısı” informasiya obyektində birləşdirilməsi

Obyektin rekvizitləri	Açarın tipi	Rekvizitin tipi	İnformasiya obyektinin adı	Obyektin məzmunu
<u>TABN</u>	sadə, unikal	sayğac	<i>MÜƏLLİM</i>	Kafedranın müəllimləri haqqında məlumat
KAFK		mətn		
SAA		mətn		
ED		mətn		
ER		mətn		
<u>KAFK</u>	sadə, unikal	sayğac	<i>KAFEDRA</i>	Kafedra haqqında məlumat
KAFA		mətn		
TEL		mətn		
MUD		mətn		
FOTO		OLE		

TƏLƏBƏ obyektinin rekvizitləri təsviredici rekvizitlərdən ibarətdir. Bu rekvizitlər tələbənin soyadı, adı, atasının adından, təvəllüdündən və ünvanından ibarətdir. Tələbəni tanımaq üçün onun soyadından istifadə etmək olar. Lakin yuxarıda qeyd etdiyimiz kimi, qrupun siyahısında onun sıra nömrəsini qəbul etmək daha yaxşı olar. Onda bütün ali məktəb tələbələri arasında tələbəni birmənalı tanımaq üçün tərkibli açardan istifadə etmək lazım gəlir, belə ki, bu açar *Qrupun nömrəsindən* və *Qrupda tələbənin sıra nömrəsindən* ibarət olacaqdır. Bu isə tələbənin birbaşa hansı qrupa aid olmasını birmənalı təyin etməyə imkan verir.

Məlumat xarakterli obyektə həmçinin *FƏNN* informasiya obyektini də aiddir. Bu obyekt fənnin adı, ümumi saatların miqdarı, mühazirəyə ayrılan saatlar, praktiki məşğələyə ayrılan saatlar, semestrlərin sayı və s. ilə xarakterizə olunur. Açar kimi fənnin kodunun istifadə edilməsi məqsədəuyğundur.

QRUP, *TƏLƏBƏ* və *FƏNN* informasiya obyektlərinin rekvizitləri cədvəl 1.21–də göstərilmişdir.

Cədvəl 1.21. Tələbə və fənlər haqqında məlumat xarakterli obyektlər

İnformasiya obyekti	Rekvizitin adı	Rekvizitin işarəsi	Açarın tipi	Sahənin tipi
<i>QRUP</i>	Qrupun nömrəsi	<u>QN</u>	unikal, sadə	mətn
	Tələbələrin sayı	SAY		ədəd
	Qrupun orta balı	OBAL		ədəd
<i>TƏLƏBƏ</i>	Qrupun nömrəsi	<u>QN</u>	unikal, tərkibli	mətn
	Tələbənin sıra nömrəsi	TN		ədəd
	Soyadı, adı, atasının adı	SAA		mətn
	Təvəllüdü	AIL		tarix
	Ünvan	ADR		mətn
	Daxil olduğu bal	OBAL		ədəd
<i>FƏNN</i>	Fənnin kodu	<u>FK</u>	unikal, sadə	sayğac
	Fənnin adı	FA		mətn
	Ümumi saatların miqdarı	SAAT		ədəd
	Mühazirə saatları	MUH		ədəd
	Praktiki məşğələ saatları	PR		ədəd
	Semestrlərin sayı	SS		ədəd
	Fənnin proqramı	PROQ		MEMO

İndi isə “*Qrupda keçilən məşğələlərin planı*” cədvəlini təhlil edək. Buradakı informasiya uçot xarakterlidir və cari semestrə hər qrupda keçilən məşğələlərin saatlarının miqdarından ibarətdir. Əsas miqdar göstəricisi olan saat – qrupun nömrəsindən, öyrənilən fənnin kodundan, müəllimdən və məşğələ növündən asılıdır. Eyni zamanda qrupun orta balı da yenə qrupdan, fəndən və s. asılıdır. Bu sənədin rekvizitləri arasında əlaqələri müəyyənləşdirdikdən sonra, biz, *TƏDRİS* informasiya obyektini seçə bilərik. Analoji qayda ilə “*imtahan cədvəli*” sənədini təhlil edərək *MÜVƏFFƏQİYYƏT* informasiya obyektini seçə bilərik. Bu informasiya obyektlərinin xarakteristikaları cədvəl 1.22-də göstərilmişdir.

MÜVƏFFƏQİYYƏT informasiya obyekti tələbənin *TƏDRİS* obyektində əks olunan hər bir məşğələ növü üzrə yekun qiyməti haqqında informasiyanı VB-də yadda saxlamağa imkan verir. Belə qiymətləndirmə bir tərəfdən tələbənin identifikatoru ilə ($QN+TN$), digər tərəfdən isə məşğələ identifikatoru ($QN+FK+TABN+DN$) ilə təyin olunur. Beləliklə, onların birləşdirilməsi *MÜVƏFFƏQİYYƏT* obyektinin unikal identifikatorunu yaradır.

İndi isə informasiya obyektləri arasında əlaqələri araşdıraraq.

QRUP – TƏLƏBƏ obyektləri arasında əlaqə birin–çoxa münasibəti ilə xarakterizə olunur. Çünki, bir qrupda çoxlu sayda tələbələr oxuyur, lakin, bir tələbə yalnız bir qrupa aiddir. Onlar arasında əlaqə *QRUP* obyektinin unikal identifikatoru olan *Qrupun nömrəsi* ilə həyata keçirilir və bu identifikator *TƏLƏBƏ* obyektinin tərkibli identifikatoruna daxil olur ($QN+TN$).

Eyni əlaqə həm də *KAFEDRA – MÜƏLLİM* obyektlərinə aiddir və bu əlaqə *KAFEDRA* obyektinin unikal açarı olan kafedranın kodu (*KAFK*) ilə həyata keçirilir.

Hər semestr ərzində müəyyən müəllim tərəfindən müxtəlif fənlər tədris olunur. Hər bir məşğələ konkret qrup üçün planlaşdırıldığından *QRUP* və *TƏDRİS* obyektləri arasında da birin–çoxa əlaqəsi təyin olunur.

Hər bir fənn üzrə müxtəlif qruplarda müxtəlif müəllimlər tərəfindən çoxlu dərslər keçirildiyi üçün *FƏNN – TƏDRİS* obyektləri arasında birin–çoxa əlaqələri müəyyənləşdirilir. Eyni əlaqə *MÜƏLLİM – FƏNN* obyektləri arasında da mövcud olur.

Cədvəl 1.22. Uçot informasiya obyektləri

İnforma- siya obyekti	Rekvizitin adı	Rekvizitin işarəsi	Rekvizi- tin tipi	Açarın tipi
<i>TƏDRİS</i>	Qrupun nömrəsi	<u>QN</u>	mətn	unikal tərkibli açar
	Fənnin kodu	<u>FK</u>	ədəd	unikal tərkibli açar
	Müəllimin tabel nömrəsi	<u>TABN</u>	ədəd	unikal tərkibli açar
	Məşğələ növü	<u>DN</u>	mətn	unikal tərkibli açar
	Saatların miqdarı	SAAT	ədəd	
	Fənn üzrə qrupun orta balı (hesablanan)	OBAL	ədəd	
<i>FƏNN</i>	Qrupun nömrəsi	<u>QN</u>	mətn	unikal tərkibli açar
	Tələbənin sıra nömrəsi	<u>TN</u>	ədəd	unikal tərkibli açar
	Fənnin kodu	<u>FK</u>	ədəd	unikal tərkibli açar
	Müəllimin tabel nömrəsi	<u>TABN</u>	ədəd	unikal tərkibli açar
	Məşğələ növü	DN	mətn	unikal tərkibli açar
	Qiymət	QİY	ədəd	

Qeyd edək ki, *QRUP – FƏNN*, *QRUP – MÜƏLLİM* və *FƏNN – MÜƏLLİM* obyektləri arasında çoxun–çoxa münasibəti mövcuddur. Yuxarıda qeyd etdiyimiz kimi, VBİS–lərdə belə əlaqəni realizə etmək mümkün olmadığı üçün, biz, normallaşdırma tələblərinə uyğun olaraq, *TƏDRİS* obyektini əlavə etməklə obyektlər arasında birin–çoxa əlaqəsini yarada bildik. *TƏDRİS* obyektini faktiki olaraq obyektlər arasında çoxun–çoxa münasibətində əlaqələndirici obyekt rolunu oynayır.

Cədvəl 1.23–də bir–biri ilə əlaqə nöqteyi–nəzərindən hansı obyektin əsas və hansı obyektin varis obyekt olması göstərilmişdir.

Cədvəl 1.23. Əsas və varis obyektlər

Əlaqə nömrəsi	Əsas obyekt	Varis obyekt	Əlaqə növü
1	<i>QRUP</i>	<i>TƏLƏBƏ</i>	1:M
2	<i>KAFEDRA</i>	<i>MÜƏLLİM</i>	1:M
3	<i>QRUP</i>	<i>TƏDRİS</i>	1:M
4	<i>FƏNN</i>	<i>TƏDRİS</i>	1:M
5	<i>MÜƏLLİM</i>	<i>TƏDRİS</i>	1:M
6	<i>TƏLƏBƏ</i>	<i>MÜVƏFFƏQİYYƏT</i>	1:M
7	<i>TƏDRİS</i>	<i>MÜVƏFFƏQİYYƏT</i>	1:M

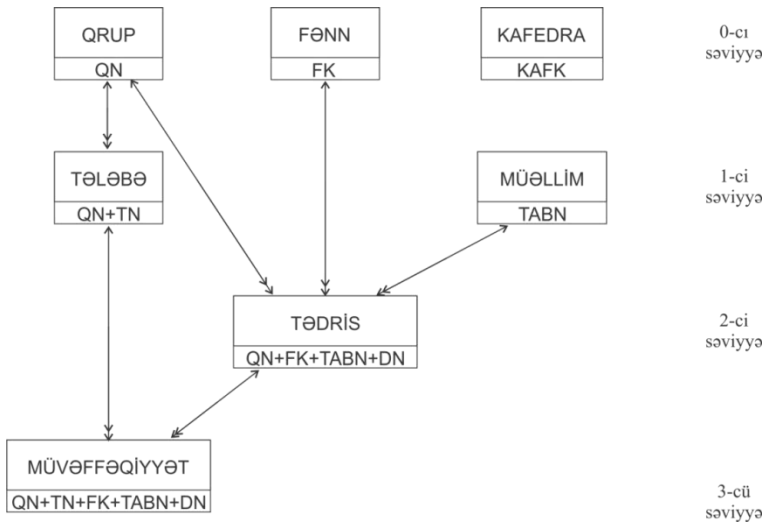
Nəhayət, bütün bu tədqiqatlardan sonra, “*Tədris prosesi*” predmet sahəsinin informasiya–məntiq modelini səviyyələr üzrə şəkil 1.18–də göstərilmiş sxem ilə təsvir edə bilərik.

Şəkil 1.19–da isə bu modelin Ms Access–də yaradılmış sxemi göstərilmişdir ki, burada obyektlərin hansı sahələrlə əlaqələndirilməsi daha dəqiq təsvir edilmişdir.

1.6.2. “Anbar” predmet sahəsinin informasiya–məntiq modeli əsasında normallaşdırma

1.6.2.1. Birinci normal forma

Birinci normal formanın tələbi ondan ibarətdir ki, VB–nin cədvəllərinin sahələri bölünməz olsun və məzmununda təkrarlanan qrup olmasın.

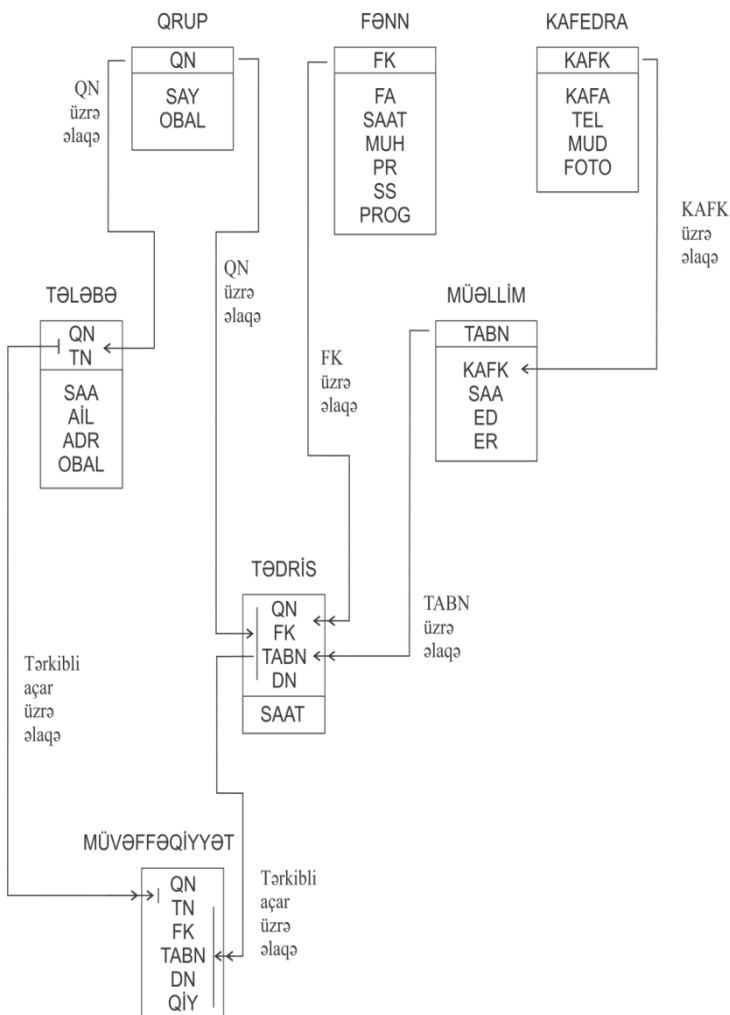


Şəkil 1.18. “*Tədris prosesi*” predmet sahəsinin informasiya–məntiq modeli

Sahənin bölünməzliyi o deməkdir ki, onun qiymətini daha kiçik hissələrə bölmək mümkün olmasın. Məsələn, əgər “*Bölmə*” sahəsində həm fakültənin həm də kafedranın adı yazılmışdırsa, deməli bölünməzlik tələbi pozulmuşdur, fakültənin və kafedranın adları üçün ayrı–ayrı sahələr müəyyən edilməlidir. Eləcə də soyadı, ad və ata adları ayrı–ayrı sahələrdə yerləşdirilməlidir.

Təkrarlanma isə o deməkdir ki, müxtəlif sahələrdə eyni mənə kəsb edən qiymətlər yerləşir. Məsələn, aylar üzrə dörd malın satışı haqqında statistika aparmaq üçün hər mal üzrə bir sahə yaratmaq olar (şəkil 1.20).

VB cədvəllərində, əlbəttə, belə təkrarçılığa yol vermək olmaz, çünki malların sayı dörd yox, yüzlərlə ola bilər və ya əvvəlcədən onların miqdarı tamamilə müəyyən olmaya bilər. Bu halda yalnız bir “*Mal*” sahəsi yaratmaqla bütün mal növlərini həmin sahədə saxlamaq lazımdır.



Şəkil 1.19. “Tədris prosesi” predmet sahəsinin Ms Access–də təsvir olunan modeli

Satış statistikas

<i>a)</i>	İl	<i>b)</i>	İl
	Ay		Ay
	1-ci mal		Mal
	2-ci mal		
	3-cü mal		
	4-cü mal		

Şəkil 1.20. Təkrarlanan qruplar *a)* və normal forma *b)*

Cədvəllərin normallaşdırılmasını “Anbar” predmet–sahəsinin informasiya–məntiq modelinin işlənməsi üzərində izah edək. Anbardan buraxılan malların uçotunu şəkil 1.21–də göstərilmiş qaimə nümunəsində izləyək.

Qaimə №

<u>Tarix</u>		<u>Alıcı</u>		<u>Ünvan</u>
18.02.2014		Azal		Bakı, Binə qəsəbədi
Buraxılan mal	Miqdarı	Ölçü vahidi	Bir vahid. qiyməti	Ümumi dəyər
Mühərrik Transformator Saygac	5	ədəd	8000	40 000
	200	ədəd	500	100 000
	1000	ədəd	100	100 000
			Cəmi	240 000

Şəkil 1.21. Qaimə nümunəsi

Qaimədə verilənləri bir cədvələ yığaq. Gələcəkdə bizə sahələr üzrə də statistika lazım ola biləcəyi üçün “*Ünvan*” sahəsini iki yerə bölərək yeni “*Şəhər*” sahəsi yaradaq. Bu həm də birinci normal formanın tələbinə uyğun gəlir. Məlumdur ki, hər bir alıcı eyni bir gündə müxtəlif miqdarda mal ala bilər, ona görə də təkrarlanan qrupun əmələ gəlməməsi üçün hər bir malın buraxılışını ayrıca bir yazıda qeyd edək. Nəticədə şəkil 1.22–də göstərilən cədvəli alacağıq.

Anbarlardan buraxılan mallar

Tarix
Alıcı
Şəhər
Ünvan
Mal
Ölçü vahidi
Bir vahidin qiyməti
Verilən malın miqdarı
Ümumi dəyəri
Qaimənin nömrəsi

Şəkil 1.22. Bölünməz və təkrarlanmayan sahəli cədvəl

1.6.2.2. İkinci normal forma

İkinci normal forma tələb edir ki, bütün sahələr ilkin açardan asılı olsun, başqa gözlə, ilkin açar yazını birqiymətli təyin etsin və izafi olmasın. Hansı sahələr ki, ilkin açarın yalnız bir hissəsindən asılıdır, onlar başqa cədvəllərin sahələrində yerləşdirilməlidir.

“*Anbar*” misalını ikinci normal formanın tələblərinə uyğunlaşdıraraq. Qaimənin tarixi və qaimənin nömrəsi ayrı–ayrılıqda yazını təyin edə bilməz, çünki onlar eyni bir qaiməyə aid bütün yazılar üçün eyni olacaqdır (şəkil 1.22–də göstərilən qaimə bir neçə yazılarla təsvir olunur). Ona görə də “*Mal*” sahəsinə ilkin açar daxil edək. Bu zaman nəzərdə tuturuq ki, bir qaimə üzrə yalnız bir adda mal buraxıla bilər, yəni ola bilməz ki, eyni bir mal üçün iki sətirdən ibarət qaimə tərtib edilsin. Şəkil 1.23–də ilkin açarın

tərkibində sahələr ayrıldıqdan sonra, cədvəlin strukturu göstərilmişdir (bu sahələr cədvəldə xətlə ayrılmışdır).

Cədvəldən asanlıqla görmək olar ki, yaratdığımız ilkin açar izafi açardır, yəni “*Qaimənin nömrəsi*” sahəsi tarixi və alıcını birmənalı təyin etməyə imkan verir. Bu qaimə üçün digər tarix və alıcı ola bilməz. “*Mal*” sahəsi isə “*Qaimənin nömrəsi*” sahəsi ilə birlikdə yazını birmənalı tanımağa imkan verir. Onda bu mülahizələri nəzərə almaqla cədvəlimizi şəkil 1.24–də göstəriləndiyi strukturlu təsvir edə bilərik.

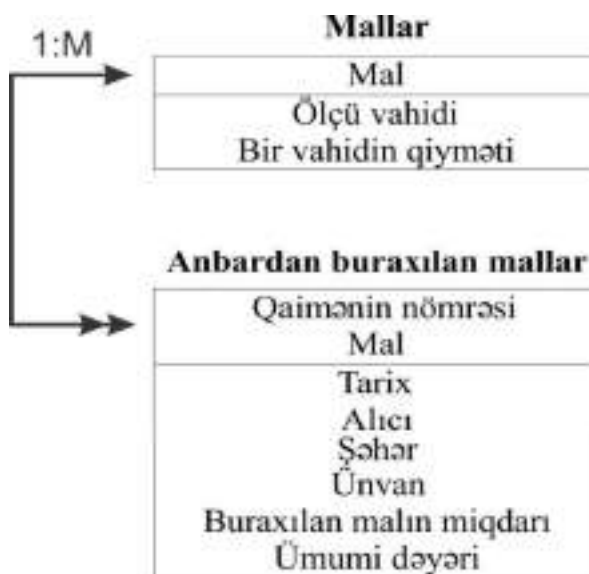
Anbardan buraxılan mallar	
Tarix	
Alıcı	
Qaimənin nömrəsi	
Mal	
Şəhər	
Ünvan	
Ölçü vahidi	
Bir vahidin qiyməti	
Buraxılan malın miqdarı	
Ümumi dəyəri	

Şəkil 1.23. İzafi ilkin açarlı cədvəl

İkinci normal formanın birinci tələbi ödənildi, lakin ikinci tələb hələ ödənməmişdir. Bu tələbə görə, yazının bütün sahələrinin qiymətləri bütövlükdə ilkin açarın qiymətindən birmənalı asılı olmalıdır və elə hal ola bilməz ki, bəzi sahələr açarın bir hissəsindən asılı olsun. Şəkil 1.24–də göstərdiyimiz cədvəldə “*Ölçü vahidi*”, “*Bir vahidin qiyməti*” sahələri ilkin açara daxil olan “*Mal*” sahəsindən asılıdır. Ona görə də bu sahələri ayıraraq yeni “*Mallar*” cədvəlində yerləşdirək və cədvəllərarası əlaqələri müəyyənləşdirək. Bir mal bir neçə qaimədə iştirak edə bildiyi üçün “*Mallar*” və “*Anbardan buraxılan mallar*” cədvəlləri arasında **1:M** münasibəti olacaqdır (şəkil 1.25).

Anbardan buraxılan mallar
Qaimənin nömrəsi
Mal
Tarix
Alıcı
Şəhər
Ünvan
Ölçü vahidi
Bir vahidin qiyməti
Buraxılan malın miqdarı
Ümumi dəyəri

Şəkil 1.24. İzafi olmayan açıqlı cədvəl



Şəkil 1.25. Mal cədvəlinin ayrılması

Sonuncu cədvəlin təhlili göstərir ki, “Alıcı” sahəsinin də “Qaimənin nömrəsi” və “Mal” sahələrindən heç bir asılılığı

yoxdur. Ona görə də bu sahəni və ondan asılı olan “Şəhər” və “Ünvan” sahələrini yeni “Alıcılar” cədvəlinə yerləşdiririk (şəkil 1.26).



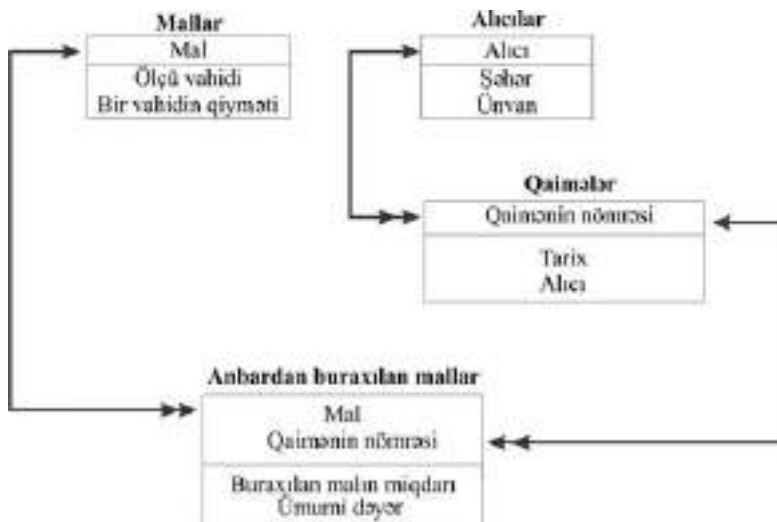
Şəkil 1.26. Alıcılar cədvəlinin ayrılması

“Anbardan buraxılan mallar” cədvəlini yenidən təhlil etdikdə görünür ki, “Tarix” sahəsi yalnız “Qaimənin nömrəsi” sahəsindən asılıdır, ona görə hər iki sahəni yeni “Qaimələr” cədvəlində yerləşdirək (şəkil 1.27).



Şəkil 1.27. Qaimələr cədvəlinin ayrılması

İndi isə bu cədvəllər arasında əlaqələri müəyyən edək. Bir alıcıya bir neçə qaimədə rast gəlinə bilər. Ona görə də “*Alicılar*” və “*Qaimələr*” cədvəli arasında birin-çoxa münasibəti olacaqdır. Bu əlaqə “*Alicı*” sahəsi ilə yerinə yetirilir. Bir qaiməyə bir neçə mal uyğun gələ bilər. Ona görə də “*Qaimələr*” və “*Anbardan buraxılan mallar*” cədvəlləri arasında “*Qaimənin nömrəsi*” sahəsi üzrə birin-çoxa əlaqəsi yaradılır (şəkil 1.28).

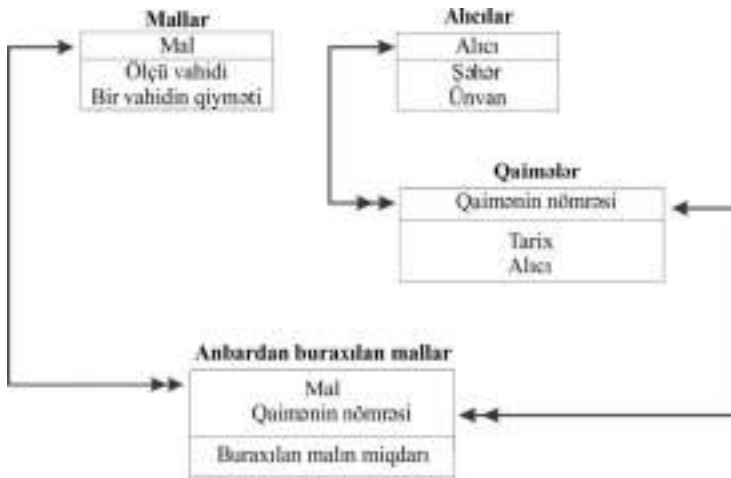


Şəkil 1.28. Cədvəllər arasında əlaqələr

1.6.2.3. Üçüncü normal forma

“*Anbardan buraxılan mallar*” cədvəlinə nəzər yetirdikdə görünür ki, “*Ümumi dəyər*” sahəsi “*Buraxılan malların miqdarı*” sahəsindən asılıdır. Bu isə üçüncü normal formanın tələblərini pozur. Həmin sahənin qiymətini belə hesablamaq olar: “*Buraxılan malın miqdarı*” sahəsinin qiymətini “*Mallar*” cədvəlinin “*Bir vahidin qiyməti*” sahəsinin qiymətinə vurmaq. VBİS-lərdə belə sahələr *hesablanan sahələr* adlanır və onları ilkin cədvəllərdən çıxarıb yekun cədvəllərdə təsvir etmək olar. Ona görə də bu sahəni uyğun cədvəldən çıxarmaq lazımdır. Beləliklə, biz “*Anbar*”

predmet sahəsinin informasiya–məntiq modelini şəkil 1.29–da göstərilədiyi kimi təsvir edə bilərik.



Şəkil 1.29. “Anbar” predmet sahəsinin informasiya–məntiq modeli

Cədvəlin normallaşdırılması prosesinə təkrarən nəzər salsaq, bütün ardıcılıqları izləsək, görərik ki, normallaşdırma nəticəsində izafi informasiyalar aradan qaldırıldı. Lakin, bununla yanaşı, cədvəllərdə yalnız bir element artıqdır ki, bu da əsas və varis cədvəldə təkrarlanan əlaqə sahələridir. Eyni zamanda, belə sahələr cədvəllərdə saxlanmadığı üçün disk yaddaşına qənaət olunur.

Normallaşdırmanın belə üstünlüklərinə baxmayaraq, onların çatışmaz cəhətləri də mövcuddur. Rekvizitlərin sayı çoxaldıqca normallaşdırma nəticəsində cədvəllərin sayı da artacaqdır. Böyük sistemlərin, təşkilatların verilənlər bazaları layihələndirildikdə bir–biri ilə əlaqəli yüzlərlə cədvəllər mövcud olur. İnsanın təfəkkürü bütün qarşılıqlı əlaqələri nəzərə almaqla yüzlərlə cədvəli təhlil etməyə imkan vermədiyi üçün normallaşdırma nəticəsində daha da artan cədvəllər arasında verilənlərin tamlığı azala bilər.

Normallaşdırmanın digər çatışmazlığı ondan ibarətdir ki, VB–yə sorğular zamanı bir neçə cədvəldən verilənləri oxumaq tələb olunur ki, bu da vaxt itkisinə gətirir. Bu xüsusən o vaxt bariz özünü göstərir ki, cədvəl həddən çox böyük olur və bu səbəbdən də verilənlərin VB–də və diskdə fragmentləşməsi baş verir.

Beləliklə, böyük həcmli verilənlərlə işlədikdə normallaşdırma tələbləri ilə sistemin sürətinin yaxşılaşdırılması arasında barışdırıcı mövqe axtarmaq lazımdır.

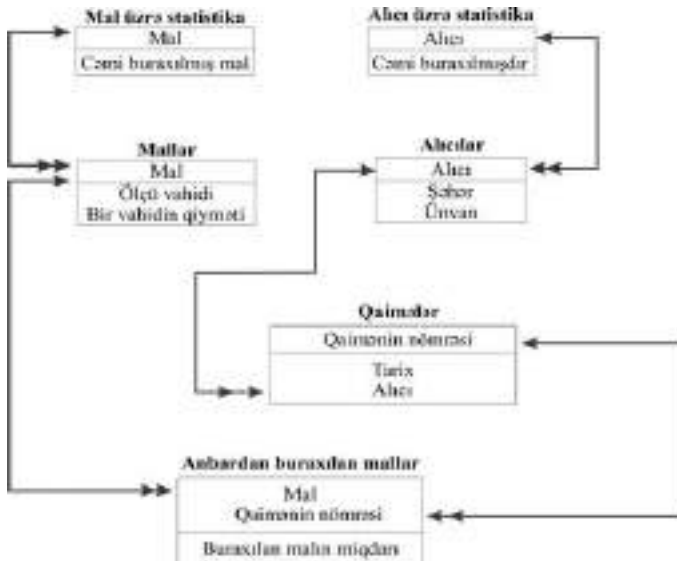
1.6.2.4. Tranzaksiya anlayışı

VB-ni bir tam vəziyyətdən başqa bir tam vəziyyətə və əksinə çevirən təsirlər *tranzaksiya* adlanır.

Əgər tranzaksiya çərçivəsində VB-ə edilən dəyişikliklərdən hər hansı biri müvəffəqiyyətlə yerinə yetirilməzsə, onda VB-ə edilən dəyişiklikdən imtina olunmalı, tranzaksiyaya qədər mövcud olan vəziyyətə qaydılmalıdır. Bunu misalla izah edək. “*Anbar*” predmet sahəsinin informasiya-məntiq modelinə daha iki cədvəl əlavə edək:

- “*Mal üzrə statistika*” adlandırdığımız cədvəldə ilin əvvəlindən anbardan buraxılmış hər bir mal haqqında yekun məlumat yerləşir;

- “*Alıcı üzrə statistika*” cədvəli isə ilin əvvəlindən hər bir alıcıya buraxılmış malların yekun qiymətlərindən ibarətdir (şəkil 1.30).



Şəkil 1.30. Statistikanın əlavə edilməsi

Qəbul edək ki, VB–ə mal sərfi haqqında məlumat daxil etməliyik. Onda tranzaksiya aşağıdakı əməliyyatlardan ibarət olacaqdır:

- “*Anbardan buraxılan mallar*” cədvəlinə yazının əlavə edilməsi;

- “*Mal üzrə statistika*” cədvəlində həmin mala uyğun yazının axtarılması və “*Cəmi buraxılmış mal*” sahəsinin “*Buraxılan malın miqdarı*” qədər artırılması; əgər “*Mal üzrə statistika*” cədvəlində həmin mala aid yazı yoxdursa, onda o əlavə edilməlidir;

- “*Alıcı üzrə statistika*” cədvəlində alıcı üzrə yazının axtarılması; buraxılmış malın dəyərinin hesablanması və “*Cəmi buraxılmışdır*” sahəsinin həmin qiymət qədər artırılması; əgər “*Alıcı üzrə statistika*” cədvəlində alıcı üzrə yazı yoxdursa, onda o əlavə edilməlidir.

Əgər tranzaksiya zamanı hər hansı bir səbəbdən, bu dəyişikliklərdən hər hansı birini yerinə yetirmək mümkün olmazsa, onda bütün digər əməliyyatların nəticələrini ləğv etmək lazımdır. Məhz bu əməliyyat tranzaksiya adlanır.

1.6. 3. “Futbol üzrə ölkə çempionatı” predmet sahəsinin informasiya–məntiq modeli əsasında normallaşdırma

Daha bir misala baxaq. “*Futbol üzrə ölkə çempionatı*” haqqında informasiya saxlamaq üçün VB layihələndirək. Bazada matçın başlanması tarixi, yarışda iştirak edən komandalar və vurulan qollar haqqında məlumat saxlanacaqdır. Əvvəlcə bütün verilənləri (rekvizitləri) bir cədvəldə birləşdirərək onun strukturunu (sahələrini) belə müəyyənləşdirək (şəkil 1.31).

Matç

Matçın başlanması tarixi; Ev sahibinin komandası; Qonaq komanda; Qol vurmuş oyunçu; Komandanın əlaməti; Vaxt

Şəkil 1.31. İlkin cədvəl

Komandanın (ev sahibi və qonaq) əlamətləri kimi onun adını, şəhəri və məşqçinin soyadını göstərəcəyik ki, bu verilənlər komandanı birmənalı tanımağa imkan verir. Qol vurmuş oyunçu üçün onun soyadını, hansı komandaya məxsus olmasını isə hər hansı bir əlamətlə, məsələn, e – ev sahibi komandası üçün, q – qonaq komanda üçün göstərmək olar.

Təcrübədə, əlbəttə, çempionat haqqında VB daha mürəkkəb struktura malik olur, bizim isə məqsədimiz informasiya–məntiq modelini tərtib etmək olduğu üçün daha sadə strukturlu VB layihələndirəcəyik.

İndi isə VB cədvəllərini normallaşdıraraq. Qeyd etdiyimiz kimi, birinci normal forma tələb edir ki, sahələrdə informasiya bölünməz olsun və cədvəldə təkrarlanan sahələr qrupu olmasın. İlk cədvəlin iki və üçüncü sahələrində komandanın adı, şəhər və məşqçi haqqında məlumat saxlanır, məsələn, *Neftçi*, *Bakı*, *Ə.Cavadov*. Bu, əlbəttə, birinci tələbə cavab vermir – komanda, şəhər və məşqçi haqqında informasiya bölünməlidir, yəni bu informasiya komandanın adı, şəhər və məşqçi kimi üç ayrı sahəyə bölünə bilər və bölünməlidir.

İkinci normallaşdırma tələbinə görə eyni mənalı bir neçə sahə olmamalıdır. Bizim cədvəlimizdə belə sahələr yoxdur, ona görə də normallaşdırmanın ikinci tələbi ödənilir. İlk baxışda adama elə gələ bilər ki, ev sahibi və qonaq komandalar sahələrində verilənlər təkrarlanır. Doğrudan da, bu sahələrdə verilənlər təkrarlanır, lakin onlar funksional təyinatlarına görə tamamilə fərqlənir və ona görə də onları təkrarlanan hesab etməmək olar.

İkinci normal formanın tələblərinə görə, cədvəl birinci normal formanın tələblərini ödəməlidir və istənilən açar olmayan sahə açar sahələri ilə birmənalı tanınmalıdır. Birinci normal formanın tələblərinə uyğunlaşdırılmış yazılar unikal deyil, verilənlərin təkrarlanması baş verir. Məsələn, əgər bir dəqiqə ərzində futbolçu bir neçə qol vurarsa, onda cədvəl eyni yazılardan ibarət olacaqdır. Yazıları unikal etmək üçün cədvələ açar sahəsi olan “*Matçın kodu*” sahəsini əlavə edək. Onda cədvəlin strukturu şəkil 1.32–də göstərilirdiyi kimi olacaqdır.

Matç

Matçın kodu
Matçın tarixi
Ev sahibi komandasının adı
Ev sahibi komandasının şəhəri
Ev sahibinin komandasının məşqçisinin soyadı
Qonaq komandanın adı
Qonaq komandanın şəhəri
Qonaq komandanın məşqçisinin soyadı
Qol vuran oyunçu
Komandanın əlaməti
Vaxt

Şəkil 1.32. Unikal açar sahəsinin əlavə edilməsi

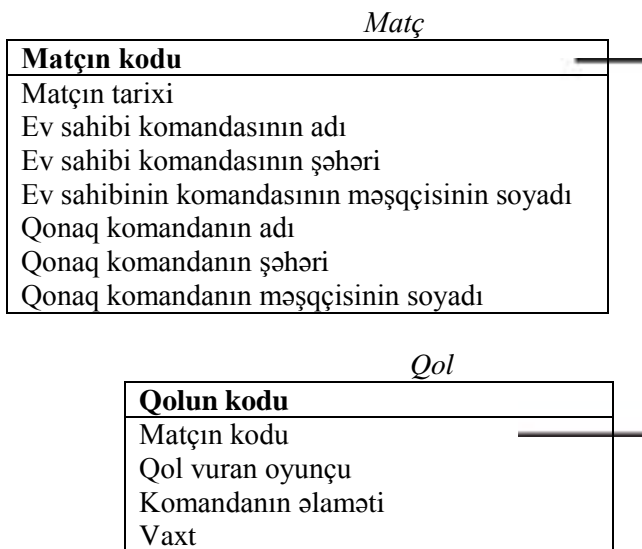
Sonuncu cədvəldə verilənlərin izafi təkrarına yol verilir, belə ki, hər bir qol üçün matçın tarixi, komandalar haqqında informasiya təkrarlanır. Ona görə də cədvəli iki yerə bölərək birində matç haqqında məlumatları, digərində isə hər bir matçda vurulmuş qollar haqqında məlumatlar yerləşdirək. Bu cədvəllərin strukturu şəkil 1.33–də göstərilmişdir.

Cədvəllər arasında əlaqə unikal açar sahəsi olan “*Matçın kodu*” sahəsi ilə yerinə yetirilir. Yazıların unikallığına nail olmaq üçün *Qol* cədvəlinə “*Qolun kodu*” açar sahəsi daxil edilmişdir.

Cədvəlimizi üçüncü normal formanın tələblərinə uyğunlaşdıraq. Bu tələbə əsasən cədvəl ikinci normal formanın tələblərinə cavab verməli və heç bir açar olmayan sahə digər açar olmayan sahələrlə birqiymətli tanınmamalıdır, başqa sözlə, açar olmayan sahələr bir–birdən asılı olmamalıdır.

Cədvəli bu tələbə uyğunlaşdırmaq üçün açardan asılı olmayan sahələrin mövcud olub–olmamasını aşkar etməliyik və əgər belə sahələr aşkar olunarsa, onları ayıraraq başqa cədvəldə yerləşdirməliyik.

Matç cədvəlində belə sahə komandaların məşqçilərinin soyadıdır ki, *komandanın adı* və *komandanın şəhəri* sahələrinin qiymətləri ilə birmənalı müəyyən olunur (nəzərdə tutulur ki, çempionat dövründə məşqçi dəyişdirilməyəcəkdir). *Matç* cədvəlini iki cədvəle ayıraraq. Yeni cədvəle komanda haqqında verilənləri daxil edək (şəkil 1.34).

Şəkil 1.33. *Matç* və *Qol* cədvəllərinin strukturu

Şəkil 1.34. “Futbol üzrə ölkə çempionatı” predmet sahəsinin informasiya–məntiq modeli

Cədvəldə sahə adları əvəzinə sahə kodları yazılmışdır, bu yazıların ölçüsünü azaltmaq məqsədi ilə edilmişdir. Cədvəllər biri–biri ilə **1:M** münasibəti ilə əlaqələndirilmişdir.

1.7. Verilənlər bazasını idarəetmə sistemləri

İlk verilənlər bazasını idarəetmə sistemləri (**VBİS**, ingiliscə **DBMS – Database Management System**) 1968-ci ildə *IBM* firması tərəfindən yaradılmış **IMS** və 1969-cu ildə *Software AG* firması tərəfindən yaradılmış **ADABA** sistemləridir. Hal-hazırda bir neçə minlərlə VBİS-lər mövcuddur və onların sayı durmadan artmaqdadır.

Yuxarı səviyyə VBİS-nin əsas funksiyalarına informasiyanın saxlanması, dəyişdirilməsi və emalını və habelə müxtəlif çıxış sənədlərinin alınmasını aid etmək olar.

Aşağı səviyyə VBİS-nin isə funksiyalarına aşağıdakıları aid etmək olar:

1. xarici yaddaş qurğularında verilənlərin idarə edilməsi;
2. operativ yaddaş buferinin idarə edilməsi;
3. tranzaksiyaların idarə edilməsi;
4. verilənlər bazalarında baş verən dəyişikliklər haqqında jurnalda qeydiyyatların aparılması;
5. verilənlər bazasının tamlığının və təhlükəsizliyinin təmin edilməsi.

Ümumiyyətlə, VBİS-lər aşağıdakı əməliyyatları yerinə yetirməyə imkan verir:

- real dünya hadisələrinin predmeti haqqında informasiyanı təsvir etmək;
- informasiyanı xarici yaddaş qurğularında saxlamaq;
- informasiyanı telekommunikasiya kanalları ilə ötürmək və almaq;
- verilmiş kriteriyə uyğun informasiyanı axtarmaq və seçmək;
- müxtəlif mürəkkəb riyazi çevirmələri yerinə yetirmək;
- verilənlər bazasında saxlanan informasiya əsasında hesablamalar yerinə yetirmək;
- eyni bir obyektə aid olan və müxtəlif verilənlər bazalarında saxlanan informasiyanı birləşdirmək;
- verilənlər bazasından informasiyanı çap etmək;
- qrafik obyektləri çəkmək və onlara düzəlişlər etmək;
- yeni obyektlər haqqında informasiya əlavə etmək;
- verilənlər bazasında informasiyaya düzəlişlər etmək.

VBİS–in üç əsas növü vardır:

- ❖ universal təyinatlı sənaye VBİS–ləri;
- ❖ xüsusi təyinatlı sənaye VBİS–ləri;
- ❖ konkret sifarişçi üçün işlənən VBİS–lər.

Xüsusi təyinatlı VBİS–lər konkret sahələrin mühasibat, anbar, bank və s. verilənlər bazasını idarə etmək üçün yaradılır.

Universal VBİS–lərin tətbiq sahələrinin dəqiq sərhədləri yoxdur, onlar hər bir sahəyə tətbiq oluna bilər və elə bu səbəbdən onlar kifayət qədər mürəkkəb olur və istifadəçilərdən xüsusi biliklərin olmasını tələb edir.

Həm xüsusi təyinatlı, həm də universal sənaye VBİS–ləri nisbətən ucuzdur, kifayət qədər etibarlıdır və istənilən təcili işlərdə tətbiq oluna bilər. Bunun əksinə olaraq, xüsusi sifarişli VBİS–lər baha başa gəlir, onların işə hazırlanması və sazlanması çox vaxt tələb edir (bir neçə aydan bir neçə ilədək). Bununla yanaşı, sənaye VBİS–dən fərqli olaraq, *sifarişçi VBİS–lər* sifarişçinin işlərinin bütün xüsusiyyətlərini maksimum nəzərə alır, onların intyerfeysləri sifarişçi üçün intuitiv aydın olur və onlardan xüsusi biliklər tələb etmir.

VBİS–ləri verilənlərin modelinə görə təsnif etsək, onların aşağıdakı növlərini göstərə bilərik:

- ❖ iyerarxiyalı;
- ❖ şəbəkəli;
- ❖ relyasiyalı;
- ❖ obyektönlü;
- ❖ obyekt–relyasiyalı.

Verilənlərin bu modellərini əvvəlki bölmələrdə izah etdiyimiz üçün burada əlavə şərhə ehtiyac görmürük, lakin biz əsasən relyasiyalı VB ilə işləyəcəyimizə görə, relyasiyalı cədvəllərdə hansı informasiyanın saxlanıldığına bir daha diqqətinizi cəlb edirik.

Beləliklə, relyasiya cədvəlləri aşağıdakılardan ibarətdir:

- eyni tipli obyektlər yığını haqqında informasiya;
 - hər hansı predmet sahəsinə aid olan bütün obyektlər yığını haqqında informasiya;
 - konkret obyekt haqqında informasiya.
- Cədvəl sətirləri aşağıdakılardan ibarətdir:
- eyni tipli obyektlər yığını haqqında informasiya;

- hər hansı predmet sahəsinə aid olan bütün obyektlər yığımı haqqında informasiya;

- konkret obyekt haqqında informasiya.

Cədvəlin sütunları aşağıdakılardan ibarətdir:

- eyni tipli obyektlər yığımı haqqında informasiya;

- hər hansı predmet sahəsinə aid olan bütün obyektlər yığımı haqqında informasiya;

- obyektin konkret nüsxəsi haqqında informasiya;

- bütün eynitipli obyektlər üçün informasiya atributlarından birinin qiymətlər yığımı.

Paylanma dərəcəsinə görə VBİS–ləri belə təsnif etmək olar:

- ❖ lokal VBİS–lər (lokal VBİS–lərin bütün tərkib hissələri bir kompüterdə yerləşir);

- ❖ şəbəkə VBİS–lər (VBİS–lərin tərkib hissələri iki və daha çox kompüterdə yerləşə bilər).

Lokal VBİS–in bütün komponentləri, başqa sözlə, serverin özü və verilənlərdən ibarət cədvəllər istifadəçinin kompüterində yerləşir. Eyni bir VB ilə bir neçə istifadəçinin eyni zamanda işləyə bilməsi üçün hər bir istifadəçinin kompüterində lokal VB–nin surəti olmalıdır. Bu tip VBİS–lərin ən ciddi problemi ondan ibarət olur ki, verilənlərin surətləri sinxronlaşdırılmalıdır, məhz elə bu səbəbdən lokal VBİS–lər praktiki olaraq tətbiq edilmir.

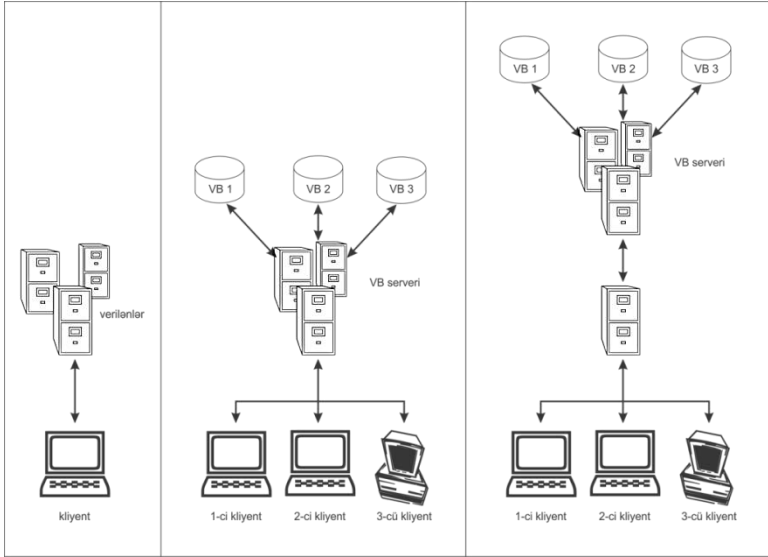
Şəbəkə VBİS–lərdə isə istifadəçinin kompüterində yalnız VB serveri ilə şəbəkə vasitəsi ilə qarşılıqlı əlaqədə olan kliyent proqramı yerləşir. Verilənlər bazası çox mənzilli arxitekturala malik ola bilər, yəni VBİS–lər bir, iki və üç mənzilli ola bilər (şəkil 1.35). Bir mənzilli arxitekturala yalnız bir mənzil (kliyent) istifadə edilir ki, o, verilənləri idarə edir və vizuallaşdırır.

İki mənzilli arxitekturala idarəetmənin əksər hissəsini VB serveri öz üzərinə götürür, kliyent isə verilənləri istifadəçi üçün rahat olan formada əks etdirməklə məşğul olur.

Üç mənzilli VBİS–lərdə isə əlavə serveri olan aralıq mənzil də mövcud olur ki, bu server VB serveri ilə kliyent arasında vasitəçi rolunu oynayır. Əlavə serveri verilənləri idarə etməyə aid bütün qayğılardan kliyenti azad edir və VB serveri ilə əlaqə yaradır.

Beləliklə, bütün hesablama yükü əlavə serverinin öhdəsinə düşür, kliyentin işi isə çox asanlaşır, yalnız soruşulan verilənləri

alır. Belə kliyəntə misal kimi internet–brauzeri göstərmək olar. Veb–əlavə ilə işlədikdə istifadəçi yalnız brauzerdə istinadı seçir və onunla əlaqədar sənədi əldə edir.



Şəkil 1.35. VBİS arxitekturaları: *solda* – bir mərhələli, *ortada* – iki mərhələli, *sağda* – üç mərhələli

Şəbəkə VBİS–ləri öz növbəsində üç növə bölünür:

- ❖ fayl–server;
- ❖ klijent–server;
- ❖ paylanmış.

Bu sistemlərin əsas atributları şəbəkədir ki, o, kompüterlərin aparat əlaqələrini təmin edir və çoxlu sayda istifadəçilərin eyni verilənlərlə korporativ işləməsinə imkan yaradır.

Fayl–server VBİS–də bütün verilənlər adətən şəbəkəyə həmişə qoşulu olan, məhz bu məqsəd üçün ayrılmış çox güclü kompüterin bir və ya bir neçə qovluğunda saxlanılır. Belə kompüter fayl–server adlanır ki, VBİS–in adı da buradan

götürülmüşdür. Belə VBİS-lərin yaradılmasının və onlara xidmətin olması onların şübhəsiz ən üstün cəhətidir, kifayətdir ki, lokal şəbəkə yaradılsın və şəbəkə əməliyyat sistemi qurulsun. Aşkar hiss etmək mümkündür ki, lokal VBİS-lə fayl-server VBİS-i arasında xüsusi fərqlər mövcud deyildir: onların hər ikisində VBİS-in bütün tərkib hissələri (verilənlərdən başqa) kliyentin kompyuterində saxlanılır.

Arxitekturasına görə fayl-server VBİS-i bir mərhələli olur, lakin bəzi hallarda əlavə serveri də istifadə oluna bilər. Fayl-server sistemlərinin çatışmaz cəhəti ondadır ki, şəbəkə həddən artıq yüklənir. Məsələn, əgər kliyentə hər hansı bir firma haqqında məlumat axtarmaq lazım gələrsə, onda əvvəlcə yüzlərlə firma haqqında məlumatdan ibarət fayl şəbəkə ilə ötürülür, yalnız bundan sonra, yəni verilənlərin sürəti yaradıldıqdan sonra axtarılan yazı tapılır. Hal-hazırda fayl-server texnologiyası köhnəlmiş hesab edilir, onun böyük informasiya sistemlərində istifadə edilməsi isə onun çatışmaz cəhətidir. Belə VBİS-lərə misal olaraq **Ms Access**, **Paradox**, **dBase**, **FoxPro**, **Visual FoxPro** göstərmək olar.

Kliyənt-server (iki mərhələli) sistemləri şəbəkənin yükünü əhəmiyyətli dərəcə azaldır, belə ki, kliyent xüsusi verilənlər bazası serveri vasitəsi ilə verilənlərlə ünsiyyətdə olur; VB serveri verilənlər kompyuterində yerləşir. VB serveri kliyentdən sorğunu qəbul edir, lazım olan yazını verilənlərin içərisində axtarır və onu kliyentə ötürür. Beləliklə, şəbəkə ilə nisbətən qısa sorğu və yeganə zəruri yazı ötürülür (hətta uyğun fayl yüz minlərlə yazılardan ibarət olsa da), serverə sorğu xüsusi Strukturlaşdırılmış sorğu dilində (**Structured Query Language – SQL**) tərtib edilir. Ona görə də əksər hallarda VB serverlərə SQL-serverlər də deyirlər. VB serverləri müxtəlif firmalar tərəfindən hazırlanan nisbətən mürəkkəb proqramlardan ibarət olur. Belə proqramlara misal olaraq **Microsoft SQL Server**, **Sybase** korporasiyasının **Sybase SQL Server**, **Oracle** korporasiyasının **Oracle**, **IBM** korporasiyasının **DB2** və s. sistemlərini göstərmək olar. Delphi-nin tərkibinə daxil olan, **Borland** korporasiyasının hazırladığı **InterBase** serveri də SQL-serverdir.

Kliyənt-server VBİS-i yüzlərlə və minlərlə kliyent yerlərində işləyə bilər. Kliyənt-server texnologiyasının üstünlükləri aşağıdakılardır:

- şəbəkənin potensial az yüklənməsi;
- mərkəzləşdirilmiş idarəetmənin rahatlığı;
- yüksək etibarlılığı və əlverişli olması;
- yüksək təhlükəsizliyi.

Kliyənt–server VBİS–lərinə aşağıdakıları aid etmək olar: **Oracle, Firebird, Interbase, IBM DB2, Informix, Ms SQL Server, Sybase Adaptive Server, Enreprise, PostgreSQL, MySQL, Cahce, ЛИНТЕР**.

Paylanmış VBİS–lər onlarla və yüzlərlə VB serverlərindən ibarət ola bilər, kliyənt yerlərinin sayı isə on və yüz minlərə qədər çata bilər. Adətən belə VBİS–lər ayrı–ayrı bölmələri böyük sahələrə səpələnmiş dövlət müəssisələrində işləyir. Bunlara müxtəlif dövlətlərin Müdafiə Nazirliyi və Daxili İşlər Nazirliyinin bölmələrini aid etmək olar.

Paylanmış VBİS–də bir neçə server bir–birini təkrarlaya bilər. Bu o məqsədlə edilir ki, bəzi həyati vacib informasiyanı korlamamaq üçün imtina və yarımçıq dayanma ehtimalını sonsuz kiçik həddə çatdırmağa nail olunsun. İnternetin sürətli inkişafı paylanmış VBİS–lərə marağı artırdı. İnternetin imkanlarından istifadə edərək nəinki dövlət miqyaslı müəssisələr, habelə nisbətən böyük olmayan kommersiya müəssisələri öz əməkdaşlarının evdə və ezamiyyətlərdə korporativ verilənlərlə işini təmin etmək üçün paylanmış VBİS–lər yaradırlar. Yeri gəlmişkən qeyd edək ki, *Oracle* müstəsna olaraq proqram təminatını yaradan ikinci ən gəlirli korporasiyadır (birinci yerdə, əlbəttə, *Microsoft* korporasiyası dayanır).

1.8. Bəzi VBİS–lərin imkan və xarakteristikalarının qısa xülasəsi

VBİS–ləri iki hissəyə ayırmaq olar:

- ❖ açıq ilkin kodlu, pulsuz (*Postgre SQL, MySQL*);
- ❖ kommersiya (*Oracle, MsSQL*).

Açıq ilkin kodlu proqramlar çox geniş yayılmışdır. Bunların içərisində ən geniş yayılanları isə **PostgreSQL** və **PostGIS** VBİS–dir. *PostgreSQL* 1986–1994–cü illərdə Kaliforniyada Maykl Stounbrekerin rəhbərliyi ilə yaradılmışdır. O, obyekt–relyasiyalı

verilənlər bazasıdır, verilənlərin həndəsi tipi (nöqtə, xətlər, poliqonlar və s.) ilə işləməyə imkan verir və sadə sorğuları yerinə yetirir.

PostGIS isə *PostgreSQL*-in genişlənmiş variantıdır. Bu, mükəmməl geoinformasiya VBİS-dir, bütün vektor formatlarını, fəza indekslərini, koordinatların dəfələrlə təsvirini və qrafik informasiya ilə iş üçün zəruri olan digər funksiyaları dəstəkləyir.

Postgre SQL VBİS-in üstün cəhətləri bunlardır:

- fəza verilənləri ilə iş üçün kitabxananın mövcud olması;
- Internet şəbəkəsi ilə tam işləmək imkanı;
- *SSL protokolu* ilə verilənləri şifrləməklə şəbəkədə informasiyanı mühafizə etmə imkanı;
- eyni verilənlər çoxluğu ilə bir neçə istifadəçinin birgə işləyə bilməsi;
- verilənlər mübadiləsinə sadələşdirməyə imkan verən *Open GIS Consortium* standartları ilə tam uyğunlaşma;
- topologiyanın səhəhliyinə nəzarət səhv daxiletmə ehtimalını azaldır.

Fəza verilənləri – fəza obyektləri haqqında ədədi verilənlərdir; belə verilənlərin tərkibinə obyektlərin fəza və qeyri-fəza atributlarının yeri və xassələri haqqında məlumat daxildir.

Fəza obyekti – real obyektin ədədi təsviridir və onun tərkibinə obyektin özü və ya onun yeri və xassələri, xarakteristikaları, atributları yığımları daxildir. Fəza obyektlərinin dörd tipi vardır:

- nöqtə;
- xətt;
- sahə (poliqon), kontur;
- səth.

Yaxın zamanlarda fəza verilənlərini saxlamaq və analiz etmək üçün İsveç şirkəti tərəfindən açıq kodlu daha bir **MySQL** VBİS-i yaradıldı. Bu sistem də kartoqrafik informasiya ilə işləməyə imkan verir.

Hal-hazırda *MySQL* VBİS *Oracle* korporasiyası tərəfindən inkişaf etdirilir. Bu VBİS-in əsas xüsusiyyəti onun ən müxtəlif cədvəllərlə işləyə bilməsidir. Açıq arxitektura malik olduğu üçün daimi olaraq yeni cədvəl tipləri meydana gəlir.

MySQL VBİS-in üstün cəhətləri bunlardır:

- çoxaxınlılıq; eyni zamanda bir neçə sorğunu dəstəkləyir;
- bir gedişdə bir sıra verilənləri birləşdirməklə əlaqələrin optimallaşdırılması;
- qeyd olunmuş və dəyişən uzunluqlu yazılar;
- ilkin kodun tərkibinə *ODBC* drayveri daxildir;
- çevik parollar və üstünlüklər sistemi;
- cədvəldə 16–a qədər açar; hər bir açarın 15–ə qədər sahəsi ola bilər;
- açar sahələrinin və xüsusi sahələrin operatorlarda dəstəklənməsi;
- 1–4 bayt uzunluğunda ədədlərin (*ints, float, double, fixed*), dəyişkən uzunluqlu sətirlərin və vaxt nişanlarının dəstəklənməsi;
- **C** və **Pearl** dillərində interfeys;
- axınlara əsaslanan cəld yaddaş sistemi;
- cədvəllərin yoxlanılması və təmiri utilitləri (*isamchk*);
- bütün verilənlər **ISO8859_1** formatında yadda saxlanılır;
- sətirlər üzərində yerinə yetirilən bütün əməliyyatlarda simvollar registrlərinə məhəl qoymur;
- psevdonimlər həm cədvəllərə, həm də cədvəllərin ayrı–ayrı sütunlarına tətbiq oluna bilər;
- açar və sahələri əlavə etmək, pozmaq da daxil olmaqla cədvəllərin asanlıqla idarə edilməsi.

MySQL hazırda *Oracle* VBİS–i ilə yanaşı ən geniş istifadə olunan sistemdir.

Geniş yayılmış VBİS və habelə proqramlaşdırma dillərindən biri də **dBase** VBİS ailəsidir. Bu ailənin ilk VBİS–i **dBaseII** 1980–ci ildə *Ashton–Tate* şirkəti tərəfindən buraxılmışdır. Sonralar meydana gələn **dBaseIII** və **dBaseIV** versiyaları ən çox yayılmış VBİS–lər olmuşdur.

dBase verilənlərinin formatları qapalı olmadığından 80–ci illərin ortalarında bir sıra şirkətlər öz dil dialekti və versiyaları olan sistemlər istehsal etdilər. Nəticədə *dBase* sisteminə oxşayan proqramlar – *FoxPro* (müasir *Visual FoxPro*), *Arago*, *Force*, *dbFast*, *Clipper*, *xBase++*, *FlagShip*, *Recital* və s. meydana gəldi. Onların hamısını birlikdə **xBase** adlandırırlar.

IBM firmasının yaratdığı VBİS–lərdən biri **DB2**–dir. İlk versiya 1993–cü ildə meydana gəlmişdir. Xarakteristikaları

yaxşılaşdırmaq üçün bir sıra mexanizmlər daxil edilmişdir ki, bunlar sistemin məhsuldarlığını yaxşılaşdırmışdır, məsələn, buferləşdirmə mexanizmi, resurs və monitoringə nəzarət alqoritmləri, diskə oxuma/yazma əməliyyatını paralelləşdirmə, diskə asinxron yazı, indekslərin ilkin oxunuşu, statistikanın toplanması və s. *DB2* optimallaşdırıcısı qrafik sorğu modeli (*QGM–Query Graph Model*) quraraq sorğuları optimal şəkllə çevirir və sorğuların optimal yerinə yetirilmə planını tərtib edir. *QGM*-in genişlənmə bilməsi *DB2*-ə yeni əlavələr, o cümlədən, obyektlyönlü genişlənmə əlavə etməyə imkan verir. Hazırda *DB2 UNIX* və **Windows NT** (və *Windows*-un daha yeni versiyalarında) əməliyyat sistemində icra oluna bilər.

IBM firmasının digər məhsulu **Informix** VBİS ailəsidir. Bu sistem 1981-ci ildə Rocer Sill və Lora King tərəfindən işlənmişdir. *Informix*-in tərkibinə onların özlərinin yaratdıqları *Informer* adlanan dili də daxil idi.

Informix dinamik serveri yüksək istismar xarakteristikalarına malikdir. *Informix*-in VB serverləri relyasiyalı və obyekt–relyasiyalıdır. Bütün müasir serverlər *Informix Dynamic Scalable Architecture (DSA)* arxitekturası üzrə yerinə yetirilmişdir və **SQL** dilində **ANSI** standartlarını dəstəkləyir.

Informix–Dynamic Server böyük və həddən çox böyük həcmli verilənlərlə işləmək üçün baza serveridir. *Informix DS* demək olar ki, bütün *Unix*–platformalar və habelə *Ms Windows NT* üçün mövcuddur. 2010-cu ildə *IBM Informix–Dynamic Server* sisteminin bütün əsas platformaları üçün yeni **IDS 11.70 Panther** versiyasını buraxmışdır.

InterBase VBİS-i 1985-ci ildə Cim Starki, onun həyat yoldaşı Ann Xarrison və Don Depalma tərəfindən *Groton Database Systems* şirkətində hazırlanmışdır. Bu sistemin sonrakı versiyaları *Borland* şirkəti tərəfindən yaradılmışdır. *InterBase* sistemi əsasən hərbi və xüsusi təyinatlı sahələrdə, məsələn, Amerikanın *MLPRS* yaylım atəşi sisteminin idarəetmə sistemində (elə indi də) və habelə *Boinq* şirkətində qanadların möhkəmliyini yoxlamaq üçün xüsusi hesabatlarda istifadə edilir.

Hal–hazırda **InterBase 2009** versiyası sonuncu versiyadır, bu versiya *Unicode* kodlarını və *AES/DES* şifrələmələrini dəstəkləyir. Sonuncu versiyanın əsas müsbət cəhətləri odur ki,

sistem tələbləri azdır, eyni zamanda bir neçə prosessorla işləyə bilir, üstəgəl inkişaf etmiş monitoring sistemi, müvəqqəti cədvəllər, jurnalla təmin etmə və s. mövcuddur. *InterBase – Linux, Ms Windows, Unix və Solaris* sistemlərini dəstəkləyir.

Araşdıracağımız növbəti VBİS *Microsoft* şirkətinin məhsulu olan məşhur **Microsoft SQL Server** sistemidir. İstifadə olunan əsas sorğu dili *Microsoft*-la birgə *Sybase* şirkəti tərəfindən yaradılan **Transact-SQL** sorğu dilidir. Kliyent-server arxitekturalı VBİS-lər içərisində ən güclü sistemlərdən biridir. Bu VBİS verilənlərin paylanmış emalı sistemlərinə qoyulan tələblərə – verilənlərin çoxaldılması, paralel emalı, nisbətən ucuz aparat platformasında böyük verilənlər bazalarını dəstəkləmək və s. – cavab verir.

Microsoft SQL Server birbaşa istifadəçi əlavələrini yaratmaq üçün nəzərdə tutulmamışdır, lakin VB-ni idarəetmə funksiyalarını yerinə yetirir.

SQL Server istənilən tip – ənənəvi və coğrafi verilənləri idarə etməyə imkan verir. Bu isə yeni nəsil əlavələrin yaradılmasına yol açır. *SQL Server* istənilən həcmli və mürəkkəb verilənləri təhlil etməyə, hesabat tərtib etməyə imkan verir.

SQL Server VB informasiyanı digər formatlara, o cümlədən *Oracle, DB2, Sybase, Ms Access* və digər VBİS-lərin formatlarına çevirə bilir.

SQL Server əlavələrə heç bir dəyişiklik etmədən verilənlər bazasını, verilənlər fayllarını və ya jurnallar fayllarını şifrəleyir. Bunun hesabına şifrələnmiş verilənlərdə həm diapazon üzrə, həm də qeyri-səlis kriteri üzrə axtarış və habelə şəxsiyyəti məlum olmayan istifadəçidən alınmış, mühafizə olunan verilənlərdə axtarış mümkün olur.

SQL Serevr OLAP (Online Analytical Processing) sorğularını və verilənlərin aktualaşdırılmasını daha cəld yerinə yetirməyə imkan verir. *SQL Server*-in tərkibinə hesabatların tərtibi, emalı və formatlaşdırılması kimi yüksək məhsuldarlıqlı mexanizm daxil edilmişdir. Genişlənə bilən arxitektura və açıq interfeyslər müxtəlif mühitlərdə hesabatların hazırlanmasını asanlaşdırır. *SQL server* VBİS-i *Word* redaktoru ilə rahat qarşılıqlı əlaqədə ola bildiyi üçün istifadəçilər hesabatları birbaşa *Microsoft Office Word* redaktorunda oxuya bilirlər.

Nəhayət, sonda onu da qeyd edək ki, bu kitabın yazıldığı dövrdə artıq *Microsoft* firması yeni *Microsoft SQL Server 2014* sınaq versiyasının yaradıldığını elan etmişdir.

Dünya VBİS bazasında ənənəvi olaraq üç şirkət – **IBM**, **Microsoft** və **Oracle** – üstünlük təşkil edir. Biz *Oracle* korporasiyasının *MySQL* VBİS-i ilə tanış olduq. İndi isə onun çox məşhur olan eyniadlı VBİS-in xarakteristikalarına baxaq.

Oracle şirkəti 1977-ci ildə, Redvudda (Kaliforniya ştatı) onun hazırkı prezidenti Leri Elison və Robert Maynor tərəfindən yaradılmışdır. **Oracle** VBİS-in çoxlu versiyaları mövcuddur, 25–30 il ərzində təkmilləşərək **Oracle 9i** versiyası yaradılmışdır. Bu gün *Oracle* VBİS-i 80-dən çox variantda əməliyyat sistemlərini dəstəkləyir. 2010-cu ilin statistikasına əsasən bu VBİS digər VBİS-lər içərisində Rusiya bazarının 60 faizini, dünya VBİS bazarının isə 30 faizini tutur.

Oracle VBİS-in 4 variantı mövcuddur:

- ❖ **Oracle Database Enterprise Edition,**
- ❖ **Oracle Database Standart Edition,**
- ❖ **Oracle Database Personal Edition,**
- ❖ **noutbuklar üçün sadələşdirilmiş variant.**

Bütün bu variantların əsasını eyni kodlar təşkil edir, demək olar ki, eyni funksiyaları yerinə yetirirlər. Yalnız *Oracle Database Enterprise Edition* variantında bəzi funksiyalar var ki, o biri variantlarda mövcud deyildir. Bu variantları qısaca nəzərdən keçirək.

Oracle Database Enterprise Edition tam funksiyalı VBİS-dir, ancaq aparat resurslarına görə imkanları bir az məhduddur. Əslində *Oracle Database Enterprise Edition* VBİS-nin tərkibinə verilənlərin təhlükəsiz saxlanması, emalı və yekun təsviri üzrə ən yeni texnologiyalar daxil edilmişdir. Geniş miqyaslaşdırma imkanı verilənlər bazası serverinin sutkada 24 saat, həftədə 7 gün və ildə 365 gün fasiləsiz işini təmin edir və inkişaf etmiş ehtiyat surət yaratma vasitələri strateji vacib informasiyanın itkisinin qarşısını ala bilər.

Oracle Database Standart Edition. Bu VBİS *Oracle Database Enterprise Edition* variantına nisbətən bir az məhduddur ki, bu da onların qiymətlərində özünü göstərir. Serverlə dördədən artıq prosessoru dəstəkləyə bilmir. *Oracle Database Standart*

Edition böyük müəssisələrin böyük olmayan təşkilatlarında, işçi qruplarında və ya bölmələrində informasiya sistemləri yaratmaq üçün ən yaxşı həll variantıdır.

Oracle Database Personel Edition. Bu VBİS *Windows NT/2000*, *Windows 95/98/ME* əməliyyat sistemlərində baza yaratmaq, öyrətmə və əlavələri istifadə etmək üçün biristifadəçili sistemdir. *Windows NT/2000* əməliyyat sistemi üçün mövcud olan *Oracle Database Personel Edition* variantının imkanları *Oracle Database Enterprise Edition* variantının bütün imkanları ilə tam eynidir. *Windows 95/98/ME* sistemləri üçün isə onun imkanı kifayət qədər məhduddur.

Oracle Lite. Çox sadələşdirilmiş mobil VBİS–dir, noutbuk və cib kompüterlərində saxlanılan korporativ verilənlər bazalarında informasiyanı sinxronlaşdırmağa imkan verir. **Oracle 8i Lite** versiyasında verilənlərə müdaxilə standart interfeyslər (*ODBC*, *OCVI*, *JDBC*) vasitəsi ilə dəstəkləndiyinə görə adi vasitələrlə əlavələr yaratmağa imkan verir.

Oracle – serverin əsas fərqli xüsusiyyətlərindən biri müxtəlif tipli verilənlərin saxlanması və emalı imkanıdır. Belə funksionallıq VBİS–in nüvəsində cəmlənmişdir və **Oracle Database**–in tərkibində olan *InterMedia* modulu ilə dəstəklənir. Bu modul müxtəlif formalı axtarışlar da daxil olmaqla mətn sənədləri ilə işləməyə imkan verir. O, həm də kontekst axtarışları, 20–dən çox formatlı qrafik obrazlarla, audio – və video – informasiyalarla işləməyi təmin edir. Bundan başqa, *Oracle* həm də yeni tip verilənləri konstruksiya etməyə imkan verir.

Oracle Database Enterprise Edition variantına daxil edilmiş **Oracle Spatial** – *verilənlər bazası komponenti* fəza obyektlərini VB–də saxlamağa imkan verir. Əlavə *Spatial*–a müraciət etməklə həndəsi fiquru interpretasiya edən verilənlərlə deyil, birbaşa həndəsi fiqurun özü ilə işləyir. Bundan başqa, *Oracle Spatial* verilənlərin tamlığını, koordinat sisteminin çevrilməsini və digər baza məntiqini təmin edir.

Oracle VBİS–in miqrasiya siyasətini də qeyd etmək yerinə düşərdi. Məlumdur ki, VBİS–lərin köhnə versiyalardan daha yeni versiyalara keçirilməsi kifayət qədər böyük zəhmət tələb edir. *Oracle* yeni məhsulunu buraxdıqda aşağıdan–yuxarıya uyğunluğuna xüsusi diqqət yetirir və bu səbəbdən belə keçid çox

ağrısız başa gəlir. Hələ bundan başqa, *Oracle* digər firmaların VBİS-nin verilənlərinin *Oracle* VBİS-ə çevrilməsinin də qayğısına qalmışdır. Bunun üçün *Oracle* pulsuz olaraq xüsusi vasitə təklif edir. *Oracle – Migration Workbench* qrafik interfeysini əldə etməklə addım-addım, yarımavtomatik rejimdə asan olmayan miqrasiya prosedurunu asanlıqla həyata keçirmək mümkün olur.

Hələlik qarşısında ciddi rəqibi olmayan *Oracle* VBİS-in təəssüflər olsun ki, çatışmayan cəhəti də vardır: yüksək istismar xarakteristikalarına nail olmaq üçün çox savadlı mütəxəssislər və mütəxəssis – inzibatçılar tələb olunur (*əgər bunu çatışmazlıq hesab etmək olarsa!*).

Digər VBİS **Visual FoxPro** sistemidir ki, o *FoxPro* sistemi əsasında yaradılmışdır. Bu VBİS yüksək sürəti ilə fərqlənir, *xBase* və **SQL** dillərini istifadə etməklə obyektlyönlü proqramlaşdırma dilinə malikdir. Bu dilin dialekti bir sıra VBİS-lərin dilinə daxildir. Yüksək səviyyəli obyekt modelinə malikdir. Hesablama şəbəkələrində istifadə edildikdə bir və çoxistifadəçili rejimlərdə işləməyə imkan verir. Əsasən bir müəssisə miqyasında müxtəlif platformalarda əlavələr üçün tətbiq edilir.

Hal-hazırda çox geniş istifadə edilən VBİS-lərdən biri də **Ms Access**-dir. O, *Microsoft Office* paketinə daxildir. Onun üstünlükləri aşağıdakılardır: əksər istifadəçilər bu sistemi tanıyır, verilənlərin yüksək dayanıqlığını təmin edir, öyrənilməsi asandır, peşəkar olmayan istifadəçilər istifadə edə bilər, müxtəlif formatlı verilənlər bazasından hesabat yaratmağa imkan verir. *Ms Access* müxtəlif verilənlər əsasında ixtiyari formalı hesabat hazırlamağa və qeyri-kommersiya əlavələrini yaratmaq üçün nəzərdə tutulmuşdur.

İ k i n c i f ə s i l

MICROSOFT ACCESS 2010–da CƏDVƏLLƏRİN YARADILMASI

2.1. Microsoft Access VBİS–in arxitekturası

Microsoft Office paketinin tərkibinə daxil olan **Microsoft Access 2010** (bundan sonra–Access) *Windows* əməliyyat sisteminin müasir əlavəsidir. Access verilənlər bazasında əsas obyektlər cədvəllər, sorğular, formalar, hesabatlar, makrosalar və modullardır. Digər VBİS–lərdə adətən, verilənlər bazası anlayışı yalnız verilənlərin saxlanıldığı fayllara aid edilir. Ms Access–də isə verilənlər bazasına bazada saxlanılan verilənlərlə əlaqədar bütün obyektlər aiddir. Access verilənlər bazasının bu obyektlərini qısaca izah edək.

Cədvəl – verilənlərin saxlanıldığı və istifadə edildiyi obyektidir. Hər bir cədvəl müəyyən tip obyekt haqqında, məsələn, müəllim haqqında informasiyadan ibarət olur. Cədvəl sahə (sütun) və yazılardan (sətirlərdən) ibarət olur. *Cədvəlin sütunlarında* müxtəlif xarakterli verilənlər, məsələn, müəllimin soyadı, atasının adı, elmi dərəcəsi, elmi rütbəsi, ünvanı, telefon nömrəsi və s. saxlanılır. *Sətirlərdə* isə hər hansı obyekt haqqında, məsələn, konkret müəllim haqqında informasiya yazılır:

İmanov R.T. T.e.d. professor Bakı S.Vurğun,100 999 88 77

Cədvəllərə müdaxilə etmək üçün açar və indekslər müəyyənləşdirilir.

Sorğu – bir və ya bir neçə cədvəldən istifadəçi üçün zəruri olan verilənləri əldə etməyə imkan verən obyektidir. Sorğu yaratmaq üçün əsasən **SQL** dilindən istifadə edilir. Sorğu cədvəldən müəyyən verilənləri seçmək, onları yeniləşdirmək, pozmaq və ya cədvələ verilənlər daxil etmək məqsədi ilə yaradılır. Sorğu vasitəsi ilə hətta bir və ya bir neçə mövcud cədvəl verilənlərindən istifadə edərək yeni cədvəl də yaratmaq olar.

Forma – əsasən verilənləri cədvəllərə daxil etmək, onları ekranda əks etdirmək və əlavənin işini idarə etmək üçün obyektidir. Formanı çap da etmək olar. Əslində forma real həyatda istifadə edilən hər hansı bir sənədin elektron formasıdır, məsələn, nəqliyyat müəssisələrində sürücülərə verilən yol vərəqi belə formaya misal ola bilər. Sürücü yol vərəqəni müəssisəyə təqdim etdikdən sonra bu vərəqdəki verilənləri məhz vərəqdə göstərilən ardıcılıqla kompüterə daxil etmək üçün həmin vərəqin elektron forması yaradılır.

Hesabat – hər hansı bir sənəd, məsələn, imtahan cədvəli və s. yaratmaq üçün obyektidir. Hesabatı da çap etmək mümkündür.

Makros – məsələni avtomatlaşdırmaq üçün makroəmr və ya maktoəmlər yığımından ibarət obyektidir.

Modul – alt proqramlar və funksiyalardan ibarət kitabxanalar yaratmağa imkan verən obyektidir; sonralar bu kitabxanalardan bütün əlavələrdə istifadə etmək olar. Modul kodlarından istifadə etməklə daxiletmə səhvlərini emal etmək, dəyişənləri elan və tətbiq etmək, dövrlər təşkil etmək və s. kimi məsələləri həll etmək olar.

2.2. Cədvəllərin yaradılması və verilənlərin tipləri

Access *Start/Programs/Microsoft Office/Microsoft Access 2010* (Пуск/Программы/Microsoft Office/Microsoft Access 2010) əmrlər ardıcılığını icra etməklə yüklənir. Access yükləndikdən sonra şəkil 2.1-də göstərilən dialoq pəncərəsi açılır ki, bu pəncərədə artıq mövcud olan bazanı açmaq və ya yeni baza yaratmaq olar.

Access-də baza yaratmaq aşağıdakı əməlləri icra etmək lazımdır (şəkil 2.1):

- *Available Templates* (Доступные шаблоны) bölməsində *Blank Database* (Новая база данных) piktoqramını seçmək;

- *Blank Database* (Новая база данных) oblastında (şəkildə sağ tərəfdə) *File Name* (Имя файла) sahəsinə yaradacağınız bazanın adını daxil edin (gələcəkdə problem yaranmamaq üçün təkidlə tövsiyə edirik ki, faylın adında, cədvəl sahələrinin adlarında və s. yalnız latın (ingilis) hərfləri və rəqəmlərdən istifadə edəsiniz).

Əgər faylın adında genişlənmiş hissəni göstərməsəniz, o, avtomatik olaraq əlavə ediləcəkdir. **Ms Access 2003** və daha aşağı versiyalarda VB fayllarının genişlənmiş hissəsi **.mdb** olduğu halda, **Ms Access 2007** versiyasından başlayaraq genişlənmiş hissə **.accdb**-dir. Əgər faylı öz istədiyiniz qovluqda saxlamaq istəyirsinizsə, onda *File Name (Имя файла)* əmrinin yanındakı qovluq nişanı üzərində mausun düyməsini basıb, lazım olan qovluğu seçib *OK* düyməsini basın;

• *Create (Создать)* düyməsini basın. Bundan sonra Access *Table1 (Таблица1)* adlı ilk boş cədvəli yaradacaq və onu cədvəl rejimində açacaqdır.



Şəkil 2.1. Microsoft Access 2010 pəncərəsi

Access-in əsas pəncərəsində *Lent* yerləşir ki, onun üzərində müəyyən əməlləri icra edən idarəetmə düymələri (piktoqramlar şəklində) yerləşir. *Lent*-in altında sənədlərin səhifələrindən ibarət sətir, sol yuxarı küncdə Access (A nişanı) düyməsi yerləşir.

Lent (Ribbon) – *Ms Office 2003* versiyasına qədər olan əlavələr pəncərələrində yerləşən menyu və alətlər panelini əvəz edir. *Lent*-in üstün cəhəti ondan ibarətdir ki, məsələnin həlli üçün

lazım olan menyü və alətlər paneli bir qrupda birləşdirilmişdir, halbuki, həmin vasitələr əvvəlki versiyalarda menyularda, alətlər panellərində, məsələlər oblastında və istifadəçinin digər interfeys komponentlərində yerləşirdi. Bunun hesabına zəruri olan əmri indi müxtəlif yerlərdə axtarmağa ehtiyac qalmır.

Access-də *Lent-in (Ribbon)* əsas səhifələri bunlardır:

- ❖ **Home (Главная),**
- ❖ **Create (Создание),**
- ❖ **External Data (Внешние данные),**
- ❖ **Database Tools (Работа с базами данных).**

İndi isə ilk *Table1 (Таблица1)* cədvəlindən izahımızı davam etdirək. Yeni cədvəl yaratdıqda kursör *Click to Add (Щелкните для добавления)* sütununun boş xanasında yerləşir. Cədvəlin strukturu verilənləri daxil etdikdə yaradılır – hər bir yeni sütun əlavə etdikdə cədvəldə yeni sahə əmələ gəlir. Sahəyə daxil edilən verilənin tipini Access özü avtomatik olaraq müəyyənləşdirir, məsələn, sahəyə *15.08.2014* daxil edilərsə, Access onu tarix kimi qəbul edir və həmin sahəyə *Date/Time (Дата/время)* tipi verir. Əgər daxil edilən verilənlərin tipini dəqiq müəyyənləşdirə bilmirsə, onda Access həmin sahəyə *Text (Текстовый)* tipi təyin edir.

Verilənlərin tipini aşkar şəkildə vermək üçün *Click to Add (Щелкните для добавления)* düyməsi üzərində mausun düyməsini basıb açılan siyahıdan müvafiq tipi seçmək lazımdır.

Cədvəl *Save (Сохранить)* əmri ilə yadda saxlanır. Əgər *Close (Закрыть)* düyməsi basılırsa, onda cədvəlin yadda saxlanması haqqında dialoq pəncərəsi peyda olacaqdır: *Yes (Да)* düyməsini basdıqda cədvəl yadda saxlanılacaq, *No (Нет)* düyməsini basdıqda cədvəl yadda saxlanmayacaq, *Cancel (Отменить)* düyməsini basdıqda isə cədvəl açıq qalacaq, yəni cədvəllə iş davam etdiriləcəkdir.

Əgər cədvələ sahələr *Click to Add (Щелкните для добавления)* düyməsini basmaqla əlavə edilərsə, onda Access sahələri *Field1, Field2 (Поле1, Поле2)* və s. adlandıracaqdır. Sahələrin (sütunların) adlarını dəyişdirmək üçün sütunun sərlövhəsi üzərində mausun düyməsi iki dəfə basıb, yeni adı daxil etmək lazımdır.

Sütunu pozmaq üçün onun sərlövhəsində mausun düyməsini basıb kontekst menyudan (mausun sağ düyməsini basmaqla açılan

Cədvəlin hər bir sahəsi üçün *Field Name* (Имя поля) sütununda ad daxil edilir, sonra isə *DateType* (Тип данных) siyahısından verilənlərin tipi seçilir.

Ms Access 2010 versiyasında verilənlərin 10 tipi mövcuddur, hər tipin özünəməxsus təyinatı vardır. Cədvəl 2.1 –də bu tiplər sadalanmış, hər tipə uyğun sahələrdə hansı verilənlərin saxlanması və bu tiplər üzərinə qoyulan məhdudiyyət şərtləri göstərilmişdir. Yadda saxlamaq lazımdır ki, Ms Access 2010 verilənlər bazasında faylın maksimal ölçüsü 2 qiqabaytdır.

Cədvəl 2.1. Ms Access verilənlərinin tipləri

Sahə verilənlərin tipləri	Sahədə saxlanılan verilənlər	Sahənin ölçüsü
<i>Text</i> (Текстовый – Mətn)	Hərflər-qəşqəm simvolları, o cümlədən mətnlər.	255–ə qədər simvol
<i>Memo</i> (Поле MEMO – MEMO sahəsi)	Ölçüsü 255 simvoldan çox olan mətnlər	1024x1024x1024 sayda simvollar; bu simvollar 2Gb yaddaş tələb edir
<i>Numbering</i> (Числовой–ədəd)	Hesablamalarda istifadə edilən tam və kəsr ədədlər	1, 2, 4, 8 və ya 12 bayt
<i>Date/Time</i> (Дата/Время – Tarix/Vaxt)	Tarix və vaxt qiymətləri	8 bayt
<i>Currency</i> (Денежный)	Pul qiymətləri	8 bayt
<i>Auto Number</i> (Счетчик–sayğac)	Unikal ədədlər; adətən ilkin açar kimi istifadə edilir.	4 bayt
<i>Yes/No</i> (Логический–məntiqi)	Məntiqi qiymətlər	1 bit
<i>OLE Object</i> (Поле объекта OLE – OLE obyektı)	Şəkillər, sənədlər, diaqramlar	1 Qiqabayta qədər

<i>sahəsi)</i>		
<i>Attachment</i> (<i>Вложение–</i> <i>daxilolma</i>)	İkili fayllar, o cümlədən, rəqəmsal təsvirlər (şəkil və digər təsvirlər)	Sıxlaşdırılmış daxiletmələr üçün 2 qiqabayt
<i>Hiperlink</i> (<i>Гиперссылка –</i> <i>hiperistinad</i>)	Hiperistinadlar, o cümlədən Web– ünvanlar	1024x1024x1024 sayda simvollar; bu simvollar 2Gb yaddaş tələb edir
<i>Lookup Wizard</i> (<i>Мастер</i> <i>подстановок –</i> <i>əvəzetmə ustası</i>)	Açılan siyahıdan sahəyə daxil edilən qiymət	Sahənin ölçüsü mətn sahəsinin (<i>Text</i>) ölçüsünə bərabər olur.

Cədvəli cədvəl və ya konstruktor rejimlərində yaratdıqda sahələrə daxil edilən qiymətlərə nəzarət etmək və sahələrin xassələrini müəyyənləşdirmək olar. Bunu yalnız konstruktor rejimində etmək olar. Sahələrin xassələrini göstərmək üçün *Data Type* (*Тип данных*) sütununda, sahənin adı qarşısında mausun düyməsini basmaq lazımdır. Xassələr *Field Properties* (*Свойства поля*) sahəsində təsvir olunur.

Konstruktor rejimində sahəni pozmaq üçün pozulacaq sahəni (sətiri) seçib (sətrin qarşısında mausun düyməsini basmaqla) klaviatüradan *Delete* klavişini basmaq və ya sətrin kontekst menyusundan *Delete Rows* (*Удалить строки*) əmrini icra etmək lazımdır.

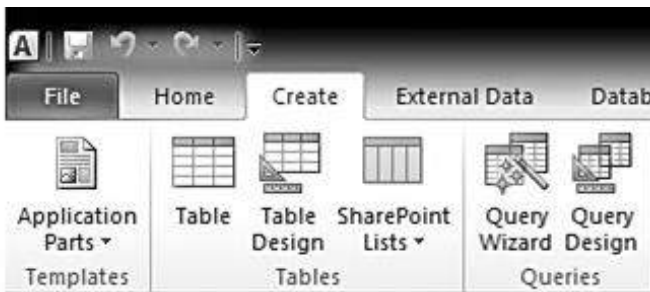
Yadda saxlayın ki, yalnız digər cədvəllərlə əlaqəsi olmayan sahəni pozmaq olar.

2.3. “Tədris prosesi” predmet sahəsinin cədvəllərinin yaradılması

Bizim məqsədimiz Access-də verilənlər bazasının yaradılması yox, hazır VB-nin ADO texnologiyası vasitəsi ilə Delphi və Visual C# dillərinin integrasiyasını öyrənməkdir. Lakin ilkin tanışlıq üçün “Tədris prosesi” predmet sahəsinin hər hansı bir

cədvəlinin yaradılması qaydaları ilə tanış olaq. “Tədris prosesi” predmet sahəsinin və ümumiyyətlə Ms Access 2010-da verilənlər bazasının layihələndirilməsini dərindən öyrənmək istəyənlərə isə xüsusi olaraq bu məqsədlə yazılmış dərs vəsaitindən istifadə etməyi tövsiyə edirik (bax: Məhərrəmov Z.T. Verilənlər bazası. Ali məktəb tələbələri üçün dərs vəsaiti. Bakı, 2015. – 436 s. Sayt: <https://www.kitabyurdu.org/kitab/it/2137-zakir-meherremov-verilenler-bazasi.html>).

İndi isə bu cədvəllərdən hər hansı birini, məsələn, *QRUP* cədvəlini konstruktor rejimində yaradaq. Bunun üçün ilkin açılan Access pəncərəsində *Blank Database* (Новая база данных) seçib *File Name* (Имя файла) sahəsinə *DEKAN* yazıb (yaradacağımız bazanın adı) *Create* (Создать) düyməsini basın. Açılan pəncərədə *Lent-in Create* (Создание) səhifəsindən *Table Design* (Конструктор таблиц) rejimini seçirik (şəkil 2.3).



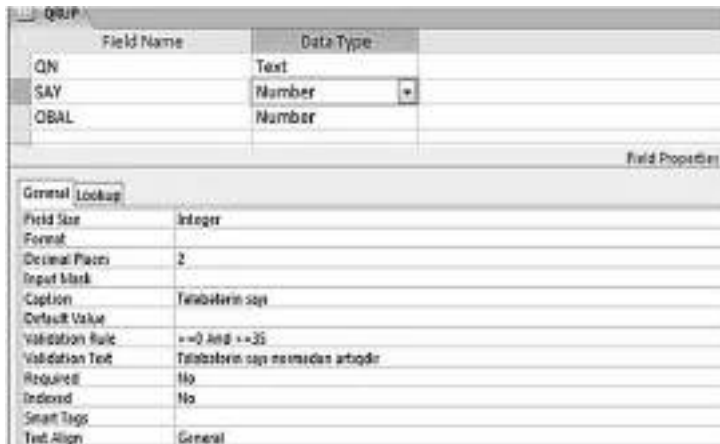
Şəkil 2.3. Konstruktor rejiminin çağırılması

Konstruktor pəncərəsində *Field Name* (Имя поля) sahəsinə sahələrin adlarını – *QN*, *SAY* və *OBAL* yazırıq, *Data Type* (Тип данных) sahəsindən isə onların tiplərini, yəni *QN* üçün – *Text* (Текстовый), *SAY* və *OBAL* üçün isə *Number* (Числовой) seçirik.

Field Properties (Свойства поля) alt pəncərəsinin *General* (Общие) səhifəsindən (şəkil 2.4) isə hər bir sahəyə uyğun xassələr müəyyənləşdiririk, yəni:

❖ *QN* sahəsi üçün:

- *Field Size* (Размер поля) – 6;
- *Caption* (Подпись) – *Grup №*;
- ❖ *SAY* sahəsi üçün:
 - *Field Size* (Размер поля) – Целое;
 - *Decimal Places* (Число десятичных знаков) – 2;
 - *Caption* (Подпись) – *Tələbələrin sayı*;
 - *Validation Rule* (Условие на значение) :
 ≥ 0 And ≤ 35 ;
 - *Validation Text* (Сообщение об ошибке) – *Tələbələrin sayı normadan artıqdır*;

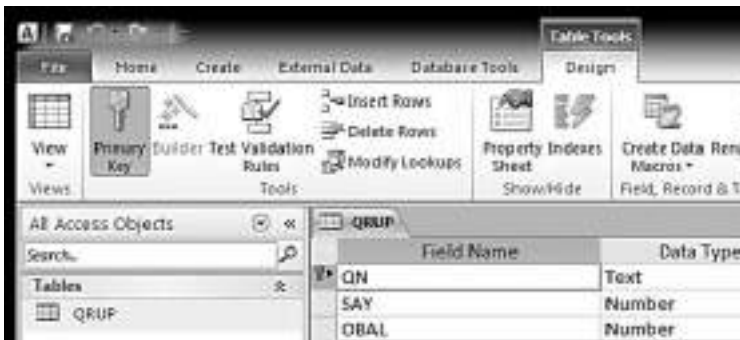


Səkil 2.4. Sahə tiplərinin xassələri

- ❖ *OBAL* sahəsi üçün:
 - *Field Size* (Размер поля) – *Одинарное с плавающей точкой*;
 - *Format* (Формат поля) – *Fixed* (Фиксированный);
 - *Decimal Places* (Число десятичных знаков) – 2;
 - *Caption* (Подпись) – *Orta bal*;
 - *Validation Rule* (Условие на значение):
 ≥ 200 And ≤ 700 ;

• *Validation Text* (Сообщение об ошибке) – Bal səhv daxil edilib.

Bu xassələrin necə təyin edilməsini öyrənək. Hər şeydən əvvəl, QN sahəsini açar sahəsi kimi təyin etmək üçün onu seçib, Lent-də yerləşən *Primary Key* (Ключевое поле) düyməsini basmaq lazımdır – sahənin adı qarşısında açar şəkli pəyda olacaqdır (şəkil 2.5).

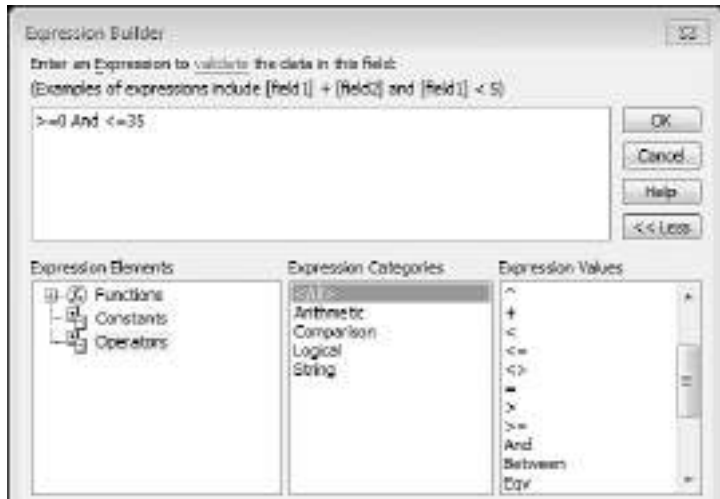


Şəkil 2.5. İlk açarın təyini

Caption (Подпись) xassəsinin qarşısında yazılmış məlumat cədvəl sahələrinin sərlövhələrində (*Table-Таблица* rejimində) təsvir olunacaqdır. Yadda saxlayın ki, VB-də, obyektəyönlü proqramlaşdırmada sahələrin, komponentlərin və s. iki xassəsi – *Name* (Имя-ад) və *Caption* (Подпись, Заголовок – sərlövhə) mövcuddur. *Name* sahənin, komponentin unikal adını müəyyənləşdirir, sahələr üzərində əməliyyatlar apardıqda kompüter bu adlardan istifadə edir, sərlövhə isə yalnız cədvəl başlıqlarında təsvir olunur.

Sahənin *Validation Rule* (Условие на значение) xassəsi sahəyə daxil edilən qiymətlərə nəzarət funksiyasını yerinə yetirir (özünü əminsizsə bu xassəyə qiymət verməyə bilərsiniz). Nəzarət şərtini daxil etmək üçün bu xassənin qarşısındakı üç nöqtə təsvirli düyməni basmaq lazımdır – bu zaman ekranda *İfade*

qurucusu (*Expression Builder–Построитель выражений*) pəncərəsi görünəcəkdir (şəkil 2.6).



Şəkil 2.6. İfadə qurucusu pəncərəsi

İfadələri yazdıqda yalnız pəncərədə göstərilən funksiya, sabit və operatorlardan (=, >, <, >=, **and**, **or** və s.) istifadə edin, boşluq işarələri qoymayın. İfadəni yazıb **OK** düyməsini basın – ifadə sahədə təsvir olunacaqdır (şəkil 2.6).

Əgər *Validation Rule* (Условие на значение) xassəsi qarşısında şərt yazılmışdırsa, onda bu şərtin ödənilib–ödənməməsi haqqında əlamət *Səhv haqqında məlumat* (Validation Text – Сообщение об ошибке) xassəsinə yazılmalıdır. Hər iki xassə cədvələ səhv verilənlər daxil edildikdə öz təsirini göstərəcəkdir.

MUELLIM və *FENN* cədvəllərinin açar sütununda *Təkrarlana bilməz* (Yes, No Duplicates–Совпадение не допускается) xassəsi yazılmışdır. Bu xassənin qiyməti *Field Properties* (Свойства поля) alt pəncərəsinin *General* (Общие) səhifəsində *Indexed* (Индексированное поле) xassəsi ilə təyin olunur. Bu xassəyə aşağıdakı üç qiymətdən bitini seçmək olar:

- *No* (Hem) – indeks sahəsi deyil;

• *Yes, Duplicates* – *Да (Допускается совпадение)* – indeksli sahə, verilənlər təkrarlana bilər;

• *Yes, No Duplicates* – *Да (Совпадение не допускается)* – indeksli sahə, verilənlər təkrarlana bilməz.


Yaratdığımız cədvəli yadda saxlayaraq onu *QRUP* adlandıraraq (şəkil 2.7).

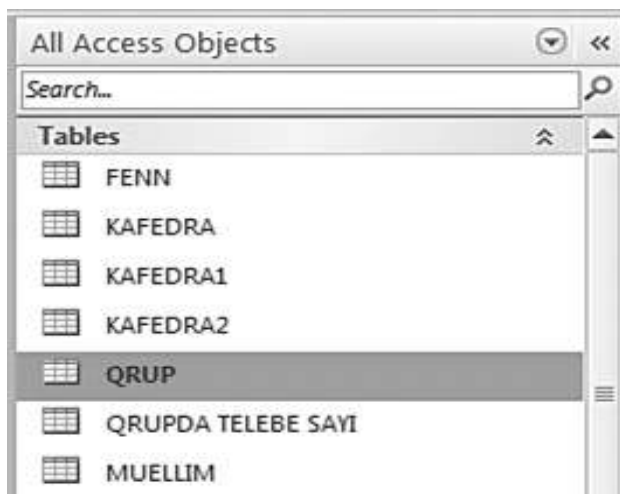


Şəkil 2.7. Cədvəlin yadda saxlanması

QRUP cədvəli Access-in *All Access Objects* (*Все объекты Access*) naviqasiya oblastında görünəcəkdir (şəkil 2.8).

Access-in naviqasiya oblastında (*All Access Objects*–*Все объекты Access*) *KAFEDRA* cədvəli üzərində mausun düyməsini iki dəfə basmaqla (və ya kontekst menyudan *Open*–*Открыть* əmrini icra etməklə) onu cədvəl rejimində açaq. Cədvələ onun sahələrinə uyğun verilənlər daxil edək. Cədvəlin bir sahəsindən digərinə keçmək üçün *Tab* və ya *Sağa* (→) klavişlərini, geriye qayıtmaq üçün isə *Shift+Tab* və ya *Sola* (←) klavişlərini basmaq lazımdır. Digər sahəyə keçməzdən əvvəl sahəyə daxiletməni ləğv

etmək üçün *Esc* klavişini və ya *Cəld müdaxilə panelindən* (*Quick Access Toolbar*) *İmtina* (*Undo–Отменить* ) düyməsini basmaq lazımdır. Növbəti yazıya (sətrə) keçmək üçün *Enter* və ya *Aşağıya* (↓) klavişlərini basmaq, əvvəlki sətrə qayıtmaq üçün isə *Yuxarıya* (↑) klavişini basmaq lazımdır. Bütün bu əməliyyatları müvafiq sahə və ya sətirlərdə mausun düyməsini basmaqla da icra etmək olar. Sahələrə verilənləri daxil etdikdə Access özü onların sahə tiplərinə uyğunluğunu yoxlayır və sahələrə səhv verilənlərin daxil edilməsinə icazə vermir.



Şəkil 2.8. Access–in naviqasiya oblasti

2.3.1. Cədvələ verilənlərin daxil edilməsi və onlara baxış

Yeni sətir həmişə cədvəlin sonuna əlavə edilir. İstənilən yazını (sətiri) seçmək üçün mausun göstəricisini sətirin başlanğıcında yerləşdirib, sətrə yönəlmiş qalın ox şəkli almasına nail olmaq lazımdır. Bundan sonra mausun düyməsini basdıqda sətir seçiləcəkdir. Hər hansı sətiri pozmaq lazım gələrsə, onu seçib kontekst menyudan *Delete Record* (*Удалить строку*) əmri icra olunmalıdır. Pozulmuş yazını bərpa etmək mümkün olmur.

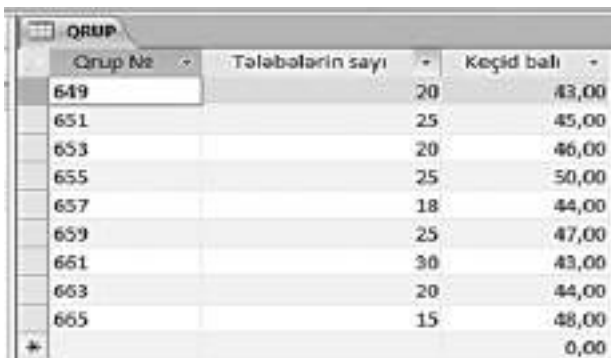
Unutmayın ki, cədvəl sahələrində mətn tipli verilənlər sahənin sol tərəfində, digər verilənlər isə sağ tərəfdə yerləşir (*Ms Excel*-də olduğu kimi). Sahələrdə verilənlər təkrarlanarsa, *Copy–Paste* əməllərindən istifadə etməklə cədvəllərin doldurulmasını sürətləndirmək olar.

Nümunə üçün şəkil 2.9-da verilənlərlə doldurulmuş *KAFEDRA* cədvəli və şəkil 2.10-da *QRUP* cədvəli təsvir olunmuşdur. Sütunların sərlövhələrində *Caption* (*Подпись*) xassəsinə verilmiş qiymət təsvir olunur; əgər bu xassəyə qiymət verilməmişdirsə, onda sütunların sərlövhələrində sahənin adına verilən qiymət təsvir olunur. Bu halda sütunların sərlövhələrinə istədiyiniz adları vermək üçün sütunun sərlövhəsində mausun düyməsini iki dəfə basıb istədiyiniz adı yazı bilərsiniz.



Kəp	Kafedranın adı	TEL	Kaf. mədini	Kaf. mədininin yəli	Click to Add
01	Informatika	555-44-30	Abdrazov T.P.		Bitməmiş Image
02	Bişçilik	555-44-31	Bəliyev A.F.		Bitməmiş Image
03	Fizika	555-44-32	İbrahimov K.L.		Bitməmiş Image
04	Tarix	555-44-33	Mehdiyev S.E.		Bitməmiş Image
05	Kimya	555-44-34	Səhəbzov E.E.		Bitməmiş Image
06	Fəlsəfə	555-44-35	Vəliyeva A.R.		Bitməmiş Image
07	İctimai	555-44-36	Zeynəlov S.L.		Bitməmiş Image

Şəkil 2.9. *KAFEDRA* cədvəli




Qrup №	Tələbələrin sayı	Keçid balı
649	20	43,00
651	25	45,00
653	20	46,00
655	25	50,00
657	18	44,00
659	25	47,00
661	30	43,00
663	20	44,00
665	15	48,00
		0,00

Şəkil 2.10. *QRUP* cədvəli

2.3.2. Cədvəllər arasında əlaqələrin yaradılması

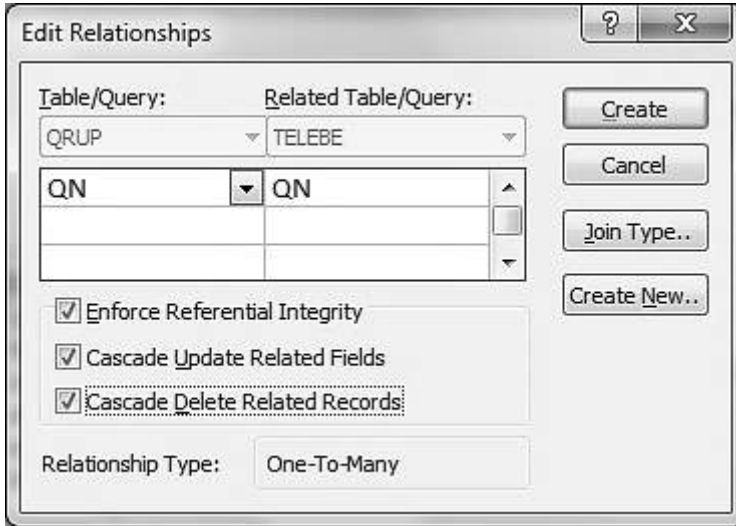
Birinci fəsildə cədvəllər arasında əlaqələrin yaradılmasının vacibliyi və bu əlaqələrin formaları haqqında geniş məlumat verdik. İndi isə hər bir cədvəl arasında praktiki olaraq əlaqələr yaratmaqla məşğul olaq. Hər şeydən əvvəl qeyd edək ki, əlaqələndiriləcək sahələr hökmən eyni tipli olmalıdır. Cədvəllər arasında əlaqə müvafiq cədvəllərin əlaqə açarları ilə yaradılır. Bu zaman əlaqə açarı həmişə əsas cədvəlin unikal açarı olur. Təbii olan cədvəldə isə əlaqə açarı ya unikal açarın bir hissəsi, ya da ilkin açarın tərkibinə daxil olmayan sahə (məsələn, *MUELLIM* cədvəlində *Kafedranın kodu*) olur. Təbii olan cədvəlin əlaqə açarı *xarici açar* adlanır. Cədvəllər arasında əlaqə yaradıldıqdan sonra müxtəlif cədvəllərdən məlumatları əks etdirən sorğular, formalar və hesabatlar yaratmaq olur.

Nümunə üçün sadə açarlardan ibarət cədvəllər arasında, məsələn, *QRUP* və *TELEBE* cədvəlləri arasında əlaqə yaradaq. Bunun üçün istənilən rejimdə açıq olan cədvəlləri bağlayıb, *Lent–in Database Tools (Работа с базами данных)* qrupundan *Relationships (Схема данных)*  düyməsini basırıq. Açılan *Relationships (Схема данных)* pəncərəsinin boş sahəsində kontekst menyunun *Show table...(Добавить таблицу...)* əmrini icra edib, yeni açılan pəncərədə əlaqə yaradılacaq cədvəllərin adlarını seçib, *Add (Добавить)* düyməsini basmaq lazımdır. Bundan sonra mausun göstəricisini *QRUP* cədvəlinin *QN* sahəsində yerləşdirib, onu *TELEBE* cədvəlinin *QN* sahəsinin üzərinə dərəcə mausun düyməsini buraxırıq. Şəkil 2.11–də göstərilən *Edit Relationships (Изменение связей)* pəncərəsi peyda olacaqdır. Bu pəncərənin *Münasibətin tipi (Relationship Type: – Тип отношения:)* sətirində birin–çoxa əlaqəsi müəyyənləşdirilmişdir. Cədvəlin aşağıdakı digər üç rejiminə də aydınlıq gətirək:

- *Enforce Referential Integrity (Обеспечение целостности данных)* – verilənlərin tamlığının təmini;

- *Cascade Update Related Fields (Каскадное обновление связанных полей)* – əlaqələndirilən sahələrin kaskad yeniləşdirilməsi;

• *Cascade Delete Related Records (Каскадное удаление связанных записей)* – əlaqələndirilən yazıların kaskad pozulması.

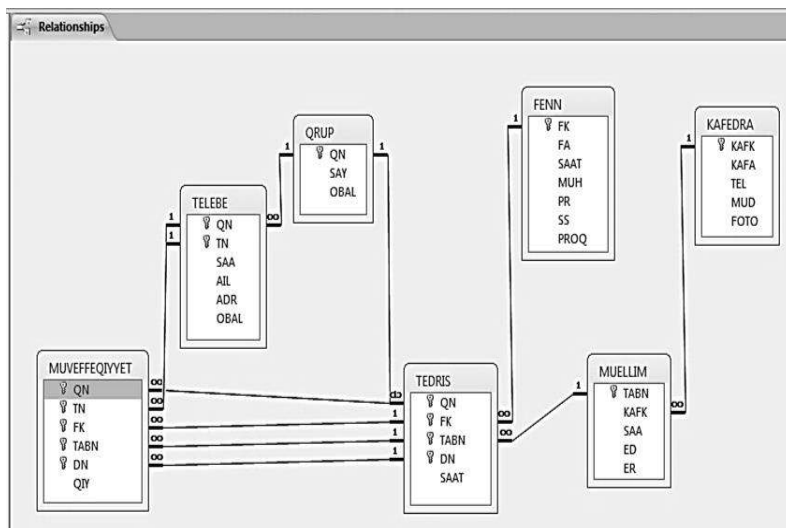


Şəkil 2.11. Cədvəllər arasında əlaqənin yaradılması

Birinci fəsildə bu rejimlər haqqında məlumat verdiyimiz üçün onları təkrarlamadan hər üç rejimin qarşısında bayraq işarəsi qoyuruq və *Create (OK)* düyməsini basırıq. Əgər cədvəl verilənlər düzgün daxil edilmişdirsə, onda cədvəl sahələri xətlə birləşdiriləcək və sxemdə **1:∞** əlaqəsi göstəriləcəkdir: bu birin çoxla əlaqəsidir.

Analoji qayda ilə *KAFEDRA*→*MUELLIM* (açar sahəsi *KAFK*), *FENN*→*TEDRIS* (açar sahəsi *FK*), *MUELLIM*→*TEDRIS* (açar sahəsi *TABN*) və *QRUP*→*TEDRIS* (açar sahəsi *QN*) cədvəlləri arasında sadə açarlarla əlaqə yaradırıq.

Nəticədə birinci fəsildə tərtib etdiyimiz “Tədris prosesi”nin informasiya–məntiq modelinin Access–də yaradılmış, şəkil 2.12–də göstərilmiş sxemini alırıq.



Şəkil 2.12. “Tədris prosesi”nin informasiya–məntiq modelinin Access–də yaradılmış sxemi

Ü ç ü n c ü f ə s i l

SQL DİLİ

Bu fəsildə **SQL** – strukturlaşdırılmış sorğu dili şərh ediləcək və onun imkanlarını Delphi-də verilənlər bazası əlavələri yaratmaqla öyrənəcəyik. Burada da izahatlar “Tədris prosesi” verilənlər bazası üzərində yerinə yetiriləcək. Lakin, Delphi Paradox cədvəlləri ilə işlədiyi üçün həmin baza məhz bu dildə yaradılmışdır. Bu fəsil yazıldıqda nəzərdə tutulmuşdur ki, oxucu Delphi-də verilənlər bazasının layihələndirilməsini bilir (bax: Məhərrəmov Z.T. Verilənlər bazası. Ali məktəb tələbələri üçün dərs vəsaiti. Bakı, 2015. – 436 s. Sayt: <https://www.kitabyurdu.org/kitab/it/2137-zakir-meherremov-verilenler-bazasi.html>).

SQL dili isə bu dillə ümumiyyətlə tanış olmayanlar və onu yenidən öyrənməyə başlayanlar üçün qısa formada şərh edilmişdir. Bu dili peşəkar səviyyədə öyrənmək üçün digər mənbələrə müraciət etmək lazımdır.

3.1. SQL dilinin xüsusiyyətləri

Verilənlərə relyasiya üsulu ilə müdaxilə yazılar qrupu üzərində əməliyyatlara əsaslanır. Bu əməliyyatları yerinə yetirmək üçün **SQL** – strukturlaşdırılmış sorğu dilinin vasitələrindən istifadə edilir. Ona görə də relyasiya üsulu ilə müdaxiləni SQL–yönlü üsul da adlandırırlar.

SQL – informasiyaya müdaxiləni təmin edən dildir və relyasiya verilənlər bazasını idarə etməyə imkan verir. **SQL** dili – **Structured Query Language** – *Strukturlaşdırılmış sorğu dili* adlanır və “*ESKÜEL*” kimi tələffüz olunur. **SQL** dili IBM firmasında E. Kodd tərəfindən *DB/2* verilənlər bazasını idarə sistemi üçün yaradılmışdır.

SQL dili müxtəlif platformalarda istifadə edilir, müxtəlif formatlı relyasiya verilənlər bazaları ilə işləməyə imkan verir. İndiyə kimi bu dilin bir neçə standartları buraxılmışdır: **SQL 86**, **SQL 89**, **SQL 92** və **SQL 99**. Standartlar iki təşkilat tərəfindən işlənir və qəbul edilir: **ANSI** (Amerika milli standartlar institutu) və **ISO** (Beynəlxalq standartlar təşkilatı).

SQL dili, qeyd etdiyimiz kimi, verilənlər bazalarına sorğular yaratmaq üçün nəzərdə tutulmuşdur və o, digər proqramlaşdırma dillərindən, o cümlədən, Delphi-dən fərqlənir. Hər şeydən əvvəl o, prosedur proqramlaşdırma dili deyildir, onun dövr, keçid, budaqlanma və s. kimi operatorları yoxdur. Hər bir sorğu verilənlərlə nə etməyi müəyyən edir, öz-özünə yerinə yetirilir və təlimatlar ardıcılığından ibarət deyildir. Məsələn, QRUP cədvəlinin bütün məzmunu əldə etmək üçün sadə sorğu

```
SELECT * FROM QRUP
```

sətirdən ibarət olacaqdır. Bu zaman sorğunun interpretasiyası və yerinə yetirilməsi üzrə bütün əməliyyatlar VBİS tərəfindən yerinə yetirilir, əlavə isə sorğuları göndərir və nəticələri təsvir etdirir. Baxdığımız halda **SQL** dilinin əsas operatorlarından biri olan **SELECT** operatoru təbiiq edilmişdir. **SQL** dilində cəmi 3 qrup operatorlar mövcuddur:

❖ **Data Manipulation Language (DML)** – *verilənləri idarəetmə operatorları*. Bu operatorlar verilənləri axtarmaq, pozmaq, dəyişdirmək və yadda saxlamaq üçündür. Onlar aşağıdakılardır:

- **SELECT** – *bir və ya bir neçə cədvəldən verilənləri seçmək;*
- **UPDATE** – *cədvəldə sahənin qiymətini dəyişmək;*
- **INSERT** – *cədvələ sətir əlavə etmək;*
- **DELETE** – *cədvəldən sətiri pozmaq.*

❖ **Data Definition Language (DDL)** – *verilənləri təyin edən operatorlar*. Bu operatorlar VB obyektlərini yaratmaq və onların strukturunu dəyişdirmək üçündür. Onlar aşağıdakılardır:

- **CREATE DATABASE** – *verilənlər bazası yaratmaq;*
- **CREATE TABLE** – *cədvəl yaratmaq;*
- **CREATE VIEW** – *virtual cədvəl yaratmaq;*
- **CREATE INDEX** – *indeks yaratmaq;*

• ALTER DATABASE – verilənlər bazasını modifikasiya etmək;

• ALTER TABLE – cədvəli modifikasiya etmək;

• ALTER VIEW – virtual cədvəli modifikasiya etmək;

• ALTER INDEX – indeksi modifikasiya etmək;

• DROP DATABASE – verilənlər bazasını pozmaq;

• DROP TABLE – cədvəli pozmaq;

• DROP VIEW – virtual cədvəli pozmaq;

• DROP INDEX – indeksi pozmaq və s.

❖ Data Control Statements (DCS) – verilənlərə nəzarət operatorları. Bu operatorlar verilənlərə müdaxilə hüququna nəzarət etmək üçündür. Onlar aşağıdakılardır:

• GRANT – icazə vermək;

• REVOKE – icazə verməmək.

SQL dili əsasən serverlə işləmək üçün nəzərdə tutulmuşdur. Lokal verilənlər bazasına relyasiya üsulunun tətbiq edilməsi bir o qədər səmərəli deyil, lakin bununla yanaşı, SQL–sorgu vasitəsi ilə aşağıdakılara nail olmaq olar:

• əlavə yerinə yetirildikdə verilənlər yığımının sahələrini tərtib etmək;

• verilənlər yığımına bir neçə cədvəldən sahə və yazıları əlavə etmək;

• mürəkkəb kriterilər üzrə yazıları seçmək;

• istənilən sahələr üzrə, o cümlədən indeksli olmayan sahələr üzrə verilənləri nizamlamaq;

• qiymətlərin qismən uyğunluğu əsasında sahədə verilənlərin axtarışı və s.

Sadalanan bu əməliyyatların bir çoxunu Table verilənlər yığımı (VY) komponentinə tətbiq etmək mümkün deyildir.

SQL dilində tərtib edilmiş proqram *SQL–sorgu adlanır*. Delphi əlavələrində onun yerinə yetirilməsi üçün verilənlər yığımı komponenti kimi Query komponenti tətbiq edilir ki, o, SQL–sorgunu yerinə yetirməyə imkan verir. SQL–sorgunun mətni (proqram kodları) Query komponentinin SQL xassəsinə yazılır. Bu kodlar əlavənin layihələndirilməsi zamanı və ya əlavə icra edildikdə tərtib olunur. Query komponenti sorgunu yerinə yetirir

və onun nəticələrinə uyğun verilənlər yığımını alır. Open metodu ilə və ya Active xassəsinə True qiyməti verməklə Query komponenti aktivləşdikdə verilənlər yığını tərtib edilir.

SQL dili haqqında müəyyən tanışlıqdan sonra qeyd etmək istəyirik ki, bir fəsildə onu nəinki, tamamilə öyrənmək, hətta dolğun şərh vermək fikrindən çox uzağıq. Bu dili mükəmməl öyrənmək üçün digər mənbələrə müraciət etmək lazımdır.

3.2. SELECT operatoru

SELECT operatoru **SQL** dilinin ən çox istifadə edilən operatorudur. O, VB cədvəllərindən mürəkkəb axtarış kriterilərini ödəyən verilənləri seçməyə və alınmış nəticələri lazım olan şəkə çevirməyə imkan verir. Bu operatorun ümumi forması belədir:

```
SELECT [DISTINCT]
seçiləcək sahələrin siyahısı və ya *
FROM cədvəllərin siyahısı
[WHERE seçilmə şərti]
[ORDER BY nizamlanacaq sahələrin siyahısı]
[GROUP BY qruplaşdırılacaq sahələrin siyahısı]
[HAVING qrupların seçilmə şərti] ]
[UNION SELECT operatorunun başqa ifadəsi]
```

SELECT operatorunun təsvirində sahələrin siyahısı və FROM operandı hökmən göstərilməlidir. Kvadrat mötərizələrin daxilində yazılmış operandlar vacib deyildir. FROM operandında yazıların seçiləcəyi cədvəllərin adları sadalanmalıdır. Siyahıda ən azı bir cədvəl olmalıdır. Sahələrin siyahısına yekun verilənlər yığımına daxil ediləcək sahələrin adları yazılmalıdır. Siyahıda ən azı bir sahə olmalıdır. Əgər verilənlər yığımına cədvəlin (cədvəllərin) bütün sahələri daxil ediləcəksə, onda sahələrin siyahısı əvəzinə * simvolu yazmaq lazımdır. Əgər siyahı bir neçə cədvəldən ibarət olarsa, onda sahələrin hansı cədvələ aid olduğunu bilmək üçün tərkibli adlardan istifadə edilməlidir, yəni cədvəlin adı və ondan nöqtə ilə ayrılan sahə adı yazılmalıdır, məsələn, KAFEDRA . KAFA.

WHERE operandında yazıların seçilmə kriterisi yazılır. Seçmə şərti məntiqi ifadədir. Bu ifadədə hesabi və məntiqi (not, and, or və s.) əməliyyatlarla yanaşı aşağıdakı əməliyyatlar da istifadə edilə bilər:

❖ Müqayisə əməliyyatları:

- = – bərabərdir;
- < – kiçikdir;
- > – böyükdür;
- <= – kiçik və ya bərabərdir;
- >= – böyük və ya bərabərdir;
- !< – kiçik deyildir (yəni böyük və ya bərabərdir);
- !> – böyük deyildir (yəni kiçik və ya bərabərdir);
- < > – bərabər deyildir;
- != – bərabər deyildir;

❖ LIKE – şablona görə müqayisə;

❖ IS NULL – sıfır qiymətinin yoxlanması

❖ IN – mənsub olmanın yoxlanması;

❖ BETWEEN – diapazona daxil olmanın yoxlanması.

ORDER BY operandında nizamlanacaq sahələrin siyahısı yazılır. Adi halda hər bir sahə qiymətlərin artma siyahısına görə nizamlanır. Əgər sahəni azalma sırası ilə nizamlamaq tələb olunarsa, onda sahənin adından sonra DESC təsviredicisi yazmaq lazımdır.

GROUP BY operandı yazılar qrupunu seçib VY-də yerləşdirməyə imkan verir. Qrup dedikdə GROUP BY operandından sonra sadalanmış sahələrdə eyni qiymətli yazılar başa düşülür. Sahələrin qrupa ayrılması yazılar üzərində qrup əməliyyatlarının, məsələn, hər qrupda tələbələrin sayının tapılması, dərslərinin cəminin hesablanması və s. yerinə yetirilməsi üçün lazımdır.

HAVING operandı GROUP BY operandı ilə birlikdə qrup daxilində yazıları seçmək üçün istifadə edilir. Yazıların qruplaşdırılması qaydaları WHERE operandında seçmə şərtinin tərtib edilməsi qaydalarına analojidir.

DISTINCT operandı təkrarlanan qiymətlərin təsvir edilməsinin qarşısını alır.

SELECT operatoru mürəkkəb strukturlu və bir–birinin daxilində ola bilər. Operatorları birləşdirmək üçün UNION operandı istifadə edilir. UNION operandı daxilində SELECT alt sorğusu yerləşir.

Sadə halda SELECT operatoru belə formada yazıla bilər:

```
SELECT *  
FROM cədvəllərin siyahısı
```

Operatorun bu forması ilə bir və ya bir neçə cədvəldən bütün yazılar seçilərək VY yaradılır. Məsələn, TELEBE cədvəlinin bütün sahələrindən ibarət sorğu yaradaq. Bunu üçün forma üzərinə Query, DataSource və DBGrid komponentləri yerləşdirib onların arasında əlaqə yaradaq. Query komponentinin DatabaseName xassəsinə Dekan aliası seçib SQL xassəsinin qarşısındakı üç nöqtə təsvirli düyməni basaq. Bu zaman ekrana String List Editor redaktorunun pəncərəsi çıxacaqdır. Bu pəncərəyə aşağıdakı sorğu kodlarını yazaq (şəkil 3. 1):

```
SELECT *  
FROM KAFEDRA
```



Şəkil 3.1. SQL kodlarının daxil edilməsi

Redaktoru bağlayıb Query komponentinin Active xassəsinə True qiyməti verərək layihəni icra edin. Alınmış VY şəkil 3.2–də göstərilmişdir.



KAFK	KAFA	TEL	MUD	FOTO
01	İnformatika	555-44-30	Abbasov T.R.	Abbasov.bmp
02	Riyaziyyat	555-44-31	Babayev A.F.	Babayev.bmp
03	Fizika	555-44-32	İbrahimov K.K.	İbrahimov.bmp
04	Tarix	555-44-33	Mehdiyev S.E.	Mehdiyev.bmp

Şəkil 3.2. KAFEDRA cədvəlinin bütün sahələrindən ibarət sorğu

Biz bu sorğunu Obyektlər inspektorunun köməyi ilə yaratdıq. İndi isə həmin sorğunu proqram yolu ilə (dinamik sorğu) yaradaq.

Sorğuları proqram yolu ilə yaratdıqda əvvəlcə sorğunu bağlamaq, sonra isə SQL xassəsinin sətirlərini təmizləmək məsləhət görülür. Bu kodlar belə yazılır:

```
Query1.Close;
Query1.SQL.Clear;
```

Növbəti addımda Add metodu ilə sorğu kodlarını SQL xassəsinə əlavə etmək lazımdır:

```
Query1.SQL.Add('SELECT *');
Query1.SQL.Add('FROM KAFEDRA');
```

Sorğunun yerinə yetirilməsi və VY–nin alınması üçün Open metodu çağırılmalıdır:

```
Query1.Open;
```

Nəhayət, sonda bütün bu kodları hər hansı bir hadisə emaledicisinə (məsələn, `ButtonClick`) təhkim etmək lazımdır:

```
procedure TForm1.Button1Click(
    Sender: TObject);
begin
    Query1.Close;
    Query1.SQL.Clear;
    Query1.SQL.Add('SELECT *');
    Query1.SQL.Add('FROM KAFEDRA');
    Query1.Open;
end;
```

Layihəni icra edib `Button1` düyməsini basdıqda `KAFEDRA` cədvəlinin bütün sahələrindən ibarət `VY` alınacaqdır.

Yalnız kafedraların adları və onların müdirlərindən ibarət sorğu yaratmaq üçün `SELECT` operatoru belə yazılmalıdır:

```
SELECT KAFA, MUD
FROM KAFEDRA
```

3.2.1. WHERE operandı

İndi isə `WHERE` operandını tətbiq edək. `MUELLIM` cədvəli əsasında elmi rütbəsi professor olan müəllimlərin siyahısından ibarət sorğu yaradaq. Bunun üçün aşağıdakı proseduru yazmaq lazımdır:

```
procedure TForm1.Button1Click(
    Sender: TObject);
begin
    Query1.Close;
    Query1.SQL.Clear;
    Query1.SQL.Add('SELECT SAA, ER');
    Query1.SQL.Add('FROM MUELLIM');
    Query1.SQL.Add('WHERE
```



```

ER=' 'професор' ' ');
Query1.Open;
end;

```

TELEBE cədvəli əsasında üç əlamətə görə – soyadı A hərfi ilə başlayan, soyadında m hərfi olan və atasının adı L. ilə qurtaran – sorğu yaradaq. Burada şablon üzrə axtarış tələb olunduğu üçün LIKE əməliyyatından istifadə edəcəyik. Bu məqsədlə forma üzərinə Panel – panelini və onun üzərinə üç standart Button düymələrini yerləşdirək. Query, DBGrid və DataSource komponentlərini isə forma üzərində yerləşdirək. Düymələrin sərlövhələrini seçmə şərtinə uyğun olaraq **A...**, **...M...** və **L.** adlandıraraq. Bu düymələr üçün aşağıdakı kodları yazaq:

```

procedure TForm1.Button1Click(
    Sender: TObject);
begin
    Query1.Close;
    Query1.SQL.Clear;
    Query1.SQL.Add('SELECT QN,SAA,OBAL');
    Query1.SQL.Add('FROM TELEBE');
    Query1.SQL.Add('WHERE
        SAA Like ''A%''');
    Query1.Open;
end;

procedure TForm1.Button2Click(
    Sender: TObject);
begin
    Query1.Close;
    Query1.SQL.Clear;
    Query1.SQL.Add('SELECT QN,SAA,OBAL');
    Query1.SQL.Add('FROM TELEBE');
    Query1.SQL.Add('WHERE
        SAA Like ''%M%''');
    Query1.Open;
end;

```

```

procedure TForm1.Button3Click(
    Sender: TObject);
begin
    Query1.Close;
    Query1.SQL.Clear;
    Query1.SQL.Add( 'SELECT QN,SAA,OBAL' );
    Query1.SQL.Add( 'FROM TELEBE' );
    Query1.SQL.Add( 'WHERE
                        SAA Like ''%J.''' );
    Query1.Open;
end;

```

Hazır əlavədə (şəkil 3.3) **A....** düyməsini basdıqda soyadı A hərfi ilə başlayan, **...M.....** düyməsini basdıqda soyadında m hərfi olan və **....J.** düyməsini basdıqda isə atasının adı L. ilə qurtaran tələbələrin siyahısı təsvir olunacaqdır.



Şəkil 3.3. Üç əlamətə görə sorğu

Qeyd edək ki, sorğu kodlarını əvvəlcədən hər hansı bir faylda yadda saxlayıb oradan da yükləmək olar. Bunun üçün forma üzərinə `OpenDialog` komponenti yerləşdirib aşağıdakı proseduru yazmaq lazımdır:

```

procedure TForm1.Buutton1Click(

```

```

Sender: TObject);
begin
  if OpenFileDialog1.Execute then
    with Query1 do begin
      Close;
      SQL.LoadFromFile(
        OpenFileDialog1.FileName);
      Open;
    end;
end;

```

Bu prosedur icra olunan zaman Button1 düyməsini basdıqda OpenFileDialog1 komponenti istənilən faylı seçməyə imkan verir. SQL–kodlarının yerləşdiyi faylı seçdikdə onun məzmunu Query1 komponentinin SQL xassəsinə yüklənir.

TELEBE cədvəlində orta balı 48–dən yuxarı olan tələbələrin siyahısını almaq üçün sorğunun kodu belə olacaqdır:

```
SELECT * FROM TELEBE WHERE OBAL >= 48
```

TELEBE cədvəlində 649–cu qrupdan başqa yerdə qalan qrupların tələbələrinin siyahısını almaq üçün sorğunun kodunu belə yazı bilərik:

```
SELECT * FROM TELEBE WHERE QN <> '649'
```

Qrupun nömrəsi dırnaq işarəsi daxilində yazılmalıdır, çünki biz Paradox cədvəlinin strukturunda bu sahəni mətn tipli (Alpha) elan etmişdik.

BETWEEN əməliyyatını nümayiş etdirək. 651–655–ci qrupların tələbələrinin siyahısını təsvir etdirən sorğunun kodu belə olacaqdır:

```
SELECT * FROM TELEBE WHERE QN
      BETWEEN '651' and '655'
```

Bu kodu belə də yazmaq olar:

```
SELECT * FROM TELEBE WHERE  
QN >= '651' and QN <= '655'
```

651–655-ci qruplardan başqa yerdə qalan qrupların siyahısını aşağıdakı kodla göstərmək olar:

```
SELECT * FROM TELEBE WHERE  
QN NOT BETWEEN '651' and '655'
```

651–655-ci qrupların tələbələrinin siyahısını təsvir etdirən sorğunun kodunu IN əməliyyatından istifadə etməklə belə yazmaq olar:

```
SELECT * FROM TELEBE WHERE  
QN IN ('651', '652', '653', '654', '655')
```

IN əməliyyatından istifadə etdikdə 651–655-ci qruplardan başqa, yerdə qalan qrupların siyahısını aşağıdakı kodla göstərmək olar:

```
SELECT * FROM TELEBE WHERE QN  
NOT IN ('651', '652', '653', '654', '655')
```

3.2.2. ORDER BY operandı

ORDER BY operandı verilənlər yığımını nizamlamaq üçün tətbiq edilir. Bu operandın konstruksiyası belədir:

```
ORDER BY sütunların siyahısı
```

Burada, sütunların siyahısı–nda nizamlama aparılacaq sütunların adları sadalanır. Əgər siyahıda bir neçə sütun adı sadalanarsa, onda birinci sütun global nizamlama üçün, ikinci sütun qrup daxilində (birinci sütunun vahid qiyməti nəzərə alınmaqla) nizamlama üçün və s. istifadə ediləcəkdir.

Adi halda nizamlayma həmişə artma sırası ilə yerinə yetirilir. Zərurət yarandıqda nizamlayma qaydasını aşkar olaraq ASC (artma sırası ilə) və DESC (azalma sırası ilə) əməliyyatları ilə müəyyənləşdirmək olar.

TELEBE cədvəlində QN (*qrupun nömrəsi*) sahəsi üzrə artma siyahısı üzrə sorğu yaratmaq üçün aşkar kodu belə yazmaq olar:

```
SELECT * FROM TELEBE ORDER BY QN ASC
```

TELEBE cədvəlində SAA (*soyadı, adı, atasının adı*) sahəsi üzrə azalma siyahısı üzrə sorğu yaratmaq üçün kodu belə yazmaq olar:

```
SELECT * FROM TELEBE ORDER BY SAA DESC
```

İndi isə QN sahəsi üzrə artma, SAA sahəsi üzrə azalma sırası ilə sorğunun kodunu yazaq:

```
SELECT * FROM TELEBE ORDER  
BY QN ASC, SAA DESC
```

Bu sorğu şəkil 3.4–də göstərilmişdir.

3.2.3. Sorğularda hesablamalar

3.2.3.1. Hesablanan sahələr

Hesablanan sahələrdə qiymətləri hesablamaq üçün SELECT operatorunun qiymətləri VY–ə çıxarılacaq siyahılarında hesablama aparılacaq ifadələri yazmaq lazımdır. Bu ifadələrdə toplama, çıxma, bölmə və vurma əməliyyatları ilə yanaşı aqreqat funksiyalarını da istifadə etmək olar. Hesablanan sahəyə avtomatik olaraq onun ifadəsinə uyğun ad verilir. İstəsək bu adı özümüz verə bilərik.

FENN cədvəlində praktiki məşğələ saatlarını 2 dəfə artırıb müəhazirə saatları ilə cəmləmək. Bu hal üçün SQL kodunu belə yazmaq olar:



GN	TN	SAA	AIL	ADR	DEAL
649	03	Davudov E.A.	1998	Qazax	46
649	02	Arazov D.S.	1996	Quba	45
649	01	Abdullayev B.T.	1997	Bakı	48
651	01	Quliyev Q.T.	1997	Bakı	50
651	02	Babayev F.K.	1995	Astara	49
653	02	Teymurov R.V.	1997	Qusar	45
653	03	Ibrahimov S.Z.	1998	Tovuz	44
653	01	Qurbanov Z.L.	1996	Bakı	41
653	04	Aslanov M.C.	1999	Tovuz	49
655	02	Teymurov R.V.	1995	Yevlax	48
655	03	Ocaqov R.B.	1995	Zaqatala	43
655	04	Qafarov C.X.	1996	Salyan	50
655	05	Balayev F.F.	1998	Bakı	40
655	01	Abbasov H.O.	1997	Qax	46
657	01	Orucov B.F.	1996	Quba	42

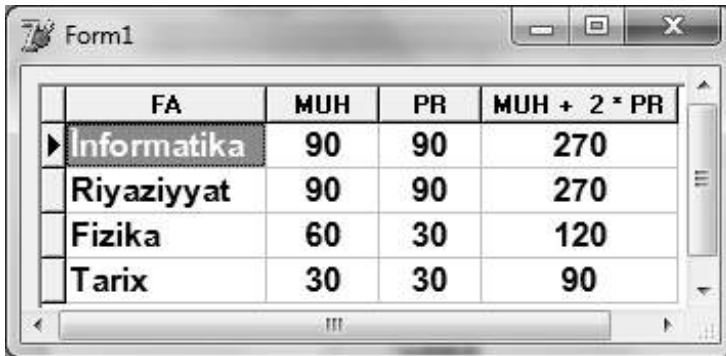
Şəkil 3.4. *Grup nömrəsi* üzrə – artma, *soyad* üzrə azalma sırasını əks etdirən sorğu

```
SELECT FA, MUH, PR, (MUH+2*PR) FROM FENN
```

Bu sorğunun nəticəsi şəkil 3.5–də göstərilmişdir.

İndi hesablanan sahəyə ad verməklə sorğu yaradaq. Bu məqsədlə AS operandı tətbiq edilir. Lakin, təəssüflər olsun ki, burada da latın hərflərindən başqa heç bir hərfdən istifadə etmək olmaz və ad bir neçə sözdən ibarət olduqda nəzərə çarpdırma simvolundan istifadə etmək lazımdır:

```
SELECT FA AS Fennin_adi, MUH AS  
Muhazire, PR AS Praktiki_Meshqele,  
(MUH+2*PR) AS Saatlarin_cemi FROM FENN
```



	FA	MUH	PR	MUH + 2 * PR
► Informatika		90	90	270
Riyaziyyat		90	90	270
Fizika		60	30	120
Tarix		30	30	90

Şəkil 3.5. Sorguda hesablanan sahənin yaradılması

Bu sorgunun nəticəsi isə şəkil 3.6–da göstərilmişdir.



Fənnin_adı	Muhazirə	Praktiki_Meshqele	Saatların_cəmi
► Informatika	90	90	270
Riyaziyyat	90	90	270
Fizika	60	30	120
Tarix	30	30	90

Şəkil 3.6. Sorguda sahələrə adların verilməsi

3.2.3.2. Aqreqat funksiyalar

Aqreqat funksiyalar yekun yazılar haqqında ümumi məlumatlar almaq üçün tətbiq edilir. Aqreqat funksiyaların arqumentləri kimi sahələrin adları göstərilir. Bu funksiyalar bütün cədvəl üçün yalnız bir qiymət hesablayır. Onlara aşağıdakılar aiddir:

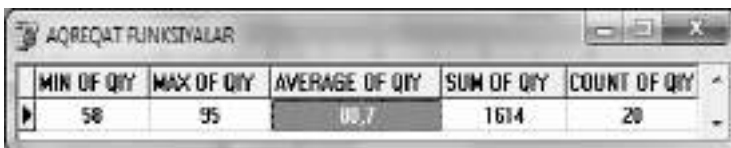
- SUM – sahənin qiymətlərinin cəmlənməsi;
- MAX – sahənin maksimal qiymətinin tapılması;
- MIN – sahənin minimal qiymətinin tapılması;

- AVG – sahənin orta qiymətinin hesablanması;
- COUNT – VY-də yazıların sayının təyini.

MUVEFFEQIYYET cədvəlində QIY (*tələbələrin balı*) sahəsi üzrə ən kiçik, ən böyük və orta balı və habelə balların cəmini hesablayaq:

```
SELECT MIN(QIY) , MAX(QIY) , AVG(QIY) ,  
SUM(QIY) , COUNT(QIY) FROM MUVEFFEQIYYET
```

Bu koda uyğun nəticələr şəkil 3.7-də göstərilmişdir.



MIN OF QIY	MAX OF QIY	AVERAGE OF QIY	SUM OF QIY	COUNT OF QIY
58	95	68.7	1614	20

Şəkil 3.7. Aqreqat funksiyalardan ibarət sorğu

COUNT funksiyası bütün yazıların sayını tapır. Təkrarlanan yazıları saymamaq üçün DISTINCT operandını tətbiq etmək lazımdır. Bu operand COUNT funksiyasının daxilində, sahənin qarşısında yazılır:

```
SELECT COUNT (DISTINCT QIY)  
FROM MUVEFFEQIYYET
```

Bu sorğunun nəticəsində 15 yazının olması haqqında məlumat verilir, halbuki, yuxarıdakı misalda 20 yazının olması bildirilmişdi.

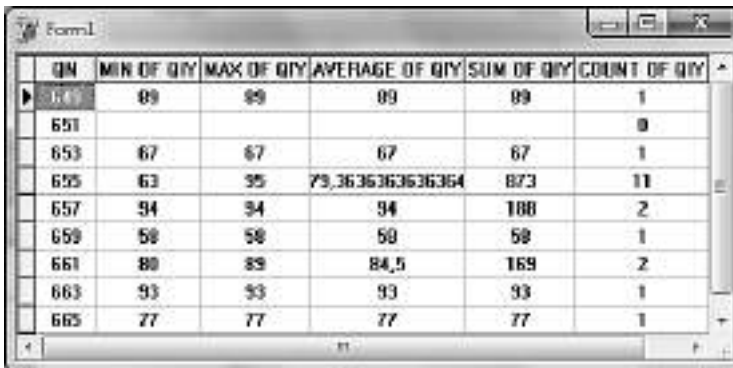
3.2.4. GROUP BY operandı

Bəzən aqreqat qiymətlərini (*minimum, maksimum, orta qiymət* və s.) bütünlükdə yekun VY üçün deyil, hər hansı bir

sütunda eyni qiymətlərlə xarakterizə olunan qruplar üzrə almaq tələb olunur. Məsələn, bizim sonuncu misalımızda bu qiymətləri biz ayrı-ayrı qruplar üzrə hesablaya bilərik. Bu halda GROUP BY operandı tətbiq edilir. GROUP BY operandından sonra qruplaşdırılacaq sahələrin adları sadalanmalıdır. O, hər bir ayrı qrup üçün yekun bir sətir yaradır. Bu operand SELECT operatorunun sonunda yazılır:

```
SELECT QN, MIN(QIY), MAX(QIY), AVG(QIY),  
SUM(QIY), COUNT(QIY) FROM MUVEFFEQIYYET  
GROUP BY QN
```

Sorğunun nəticəsi şəkil 3.8-də göstərilmişdir.



QN	MIN OF QIY	MAX OF QIY	AVERAGE OF QIY	SUM OF QIY	COUNT OF QIY
651	89	89	89	89	1
653	67	67	67	67	1
655	63	95	79,3636363636364	873	11
657	94	94	94	188	2
659	58	58	58	58	1
661	80	89	84,5	169	2
663	93	93	93	93	1
665	77	77	77	77	1

Şəkil 3.8. Sahələrin qruplaşdırılması

3.2.5. Cədvəllərin birləşdirilməsi

Əksər hallarda bir neçə cədvəllərdən verilənləri alıb bir yekun məntiqi cədvələ yığmaq tələb olunur. Bu əməliyyat cədvəllərin birləşdirilməsi adlanır. Birləşdirmə zamanı cədvəllərin müxtəlif sahələrinin əlaqələndirilməsi baş verir. Bir neçə birləşdirmə növü mövcuddur:

- CROSS JOIN – çarpaz birləşdirmə;

• INNER JOIN – daxili birləşdirmə, adi halda istifadə edilir;

- LEFT JOIN [OUTER] – sol xarici birləşmə;
- RIGHT JOIN [OUTER] – sağ xarici birləşmə;
- FULL JOIN [OUTER] – tam xarici birləşmə;

Xarici birləşmələr ANSI-92 standartlı **SQL** dili ilə yaradılır və SELECT operatorunda JOIN operandı tətbiq edilməklə yaradılır. Daxili birləşmələr (yaxud sadəcə birləşmələr) isə həm JOIN operandını tətbiq etməklə (ANSI-92 standartı), həm də ondan istifadə etməməklə (ANSI-89 standartı) yaradıla bilər.

ANSI-92 standartını tətbiq etdikdə cədvəllərin birləşdirilməsi üçün SELECT operatorunun ümumi forması belə olacaqdır:

```
SELECT sahələrin_siyahısı
FROM 1-ci_cədvəlin adı { INNER | LEFT | RIGHT }
JOIN 2-ci_cədvəlin adı
ON birləşdirmə_şərti
```

Sorğunu tərtib etdikdə aşağıdakı qaydalara riayət etmək lazımdır:

• JOIN operandından solda və sağda birləşdiriləcək cədvəllərin adları göstərilməlidir;

• ON operandından sonra birləşdirmə şərti yazılmalıdır;

• sağ cədvələ əsaslanan axtarış şərti ON operandında yerləşdirilməlidir;

• sol cədvələ əsaslanan axtarış şərti WHERE operandında yerləşdirilməlidir.

FROM operandından sonra bir-birindən vergüllə ayrılmaqla cədvəllərin adları sadalanır (JOIN operandını istifadə etmədikdə). Sahələrin adlarına isə tərkibli adlar kimi müraciət edilir. Yəni, əvvəlcə cədvəlin adı yazılır, sonra nöqtə işarəsi qoyulmaqla sahənin adı göstərilir. Məsələn, TELEBE cədvəlində QR (*qrupun nömrəsi*) sahəsinə müraciət etmək üçün TELEBE.QR yazmaq lazımdır.

Birləşdirmə zamanı adətən psevdonimlərdən istifadə edirlər. Psevdonimlər WHERE operandında cədvəlin adından sonra təyin edilir. Bizim bazamızda QRUP, TELEBE, MUVEFFEQIYYET və TEDRIS cədvəllərində eyni adlı QN (*qrupun nömrəsi*) sahələri vardır. Bu cədvəllər üçün aşağıdakı psevdonimləri təyin edə bilərik:

```
WHERE QRUP Q, TELEBE T,  
MUVEFFEQIYYET M, TEDRIS TS
```

Bu psevdonimlərə müraciət isə belə olacaqdır:

```
SELECT Q.QN, T.QN, M.QN, TS.QN  
WHERE QRUP Q, TELEBE T,  
MUVEFFEQIYYET M, TEDRIS TS
```

Göründüyü kimi, bu psevdonimlərin VB üçün yaradılan psevdonimlə (aliasla) heç bir əlaqəsi yoxdur. Psevdonimlər cədvəlin adlarına qoyulan tələblərə cavab verən istənilən identifikator ola bilər.

Daxili birləşdirmədə VY-ə elə yazılar daxil edilir ki, onlar üçün birləşdirmə şərti yerinə yetirilir. Belə bir məsələyə baxaq. Yalnız “*İnformatika*” fənni üzrə imtahan vermiş tələbələrdən ibarət cədvəl tərtib edək. Yaradacağımız cədvələ aid verilənlər MUVEFFEQIYYET və TELEBE cədvəllərində yerləşir. Qrupların nömrələri və ballar MUVEFFEQIYYET cədvəlindən, tələbələrin soyadları isə TELEBE cədvəlindən götürülməlidir. Bunun üçün SQL-kodlarını belə yaza bilərik:

```
SELECT M.QN, T.SAA, F.FK, M.QIY  
FROM MUVEFFEQIYYET M, TELEBE T  
WHERE M.FK = '01'
```

Bu halda kodu 01 olan “*İnformatika*” fənni üzrə hətta balı olmayan tələbələr də (boş xanalar) cədvələ daxil edilir. Bunun qarşısını almaq üçün sorğuya yeni kod əlavə edək:

```
SELECT M.QN, T.SAA, F.FK, M.QIY
```

```
FROM MUVEFFEQIYYET M, TELEBE T  
WHERE M.FK = '01' AND M.QIY IS NOT NULL
```

Artıq cədvəldə boş sahələr olmayacaqdır. Lakin cədvəldə fənnin kodunun yox, adının təsvir olunması daha yaxşı olardı. Ona görə də kodu təkmilləşdirək. Fənnin adı FENN cədvəlində yerləşdiyi üçün MUVEFFEQIYYET cədvəlindəki FK (*fənnin kodu*) sahəsi ilə FENN cədvəlindəki FK sahəsini əlaqələndirmək lazımdır. Bu məqsədlə biz FROM operandına FENN cədvəlini əlavə edib ona F psevdonimi verdikdən sonra, WHERE operandında FK sahələrini əlaqələndirməli və SELECT operatorunda FK əvəzinə FA sahəsini yazmalıyıq. Bundan başqa, qrupları artma sırası ilə yerləşdirək. Onda yekun SQL–kodlarını belə yaza bilərik:

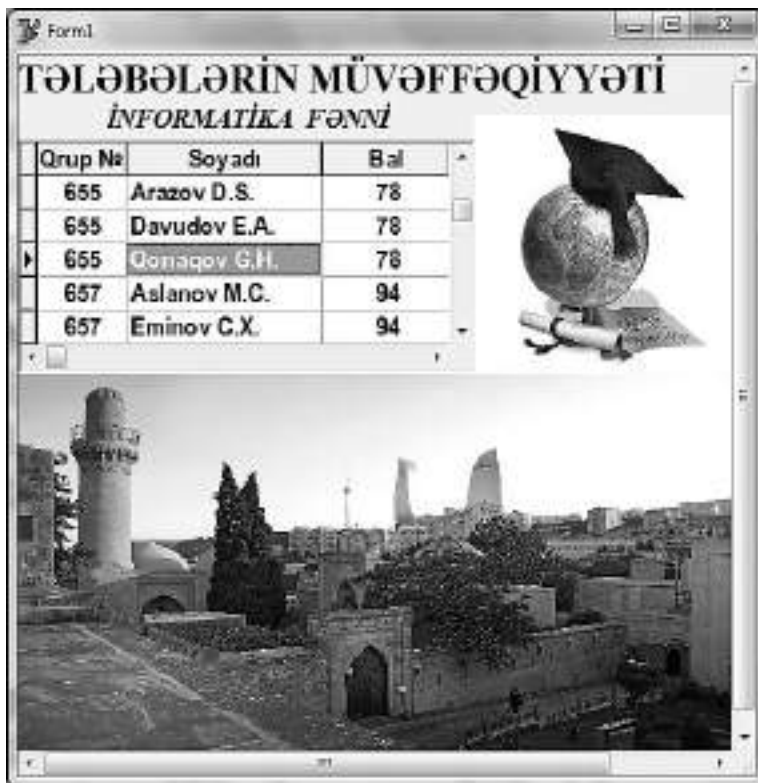
```
SELECT M.QN, T.SAA, F.FA, M.QIY  
FROM MUVEFFEQIYYET M, TELEBE T, FENN F  
WHERE M.FK = '01' AND M.QIY IS NOT NULL  
AND M.FK=F.FK  
ORDER BY M.QN
```

Beləliklə, bu sorğu aşağıdakı hissələrdən ibarət tərtib edildi: SELECT operatorundan sonra məntiqi cədvələ daxil ediləcək sahələrin adını sadaladıq. FROM operandı ilə verilənləri hansı cədvəllərdən götürəcəyimizi bildirdik. Burada həm də cədvəllərə psevdonimlər təyin etdik. WHERE operandında isə biz üç şərt göstərdik: kodu 01 olan fənnin seçilməsi; boş sahələrin yekun cədvələ daxil edilməməsi; MUVEFFEQIYYET cədvəlindəki FK (*fənnin kodu*) sahəsi ilə FENN cədvəlindəki FK sahəsi arasında əlaqə. Sonda qrupların nömrəsini artma sırası ilə düzdük.

Sorğu yalnız “*İnformatika*” fənni üzrə yerinə yetirildiyi üçün DBGrid komponentində FA sahəsini gizlədib (Visible – False) fənnin adını Label komponentində yazaq. Müəyyən tərtibat işləri aparıldıqdan sonra alınmış yekun cədvəl şəkil 3.9–da göstərilmişdir.

Daxili birləşdirmədə axtarış şərti WHERE operandında yazılır. Xarici birləşdirmədə isə şərt ON operandında yazılır və belə birləşdirmənin SQL–kodlarının ümumi formatı belədir:

```
SELECT sahələrin siyahısı
FROM 1-ci_cədvəl [ birləşdirmə növü ] JOIN 2-ci_cədvəl
ON birləşdirmə_şərti
```



Şəkil 3.9. Daxili və xarici birləşdirmə ilə yaradılan sorğunun nəticəsi

Xarici birləşdirmə daxili birləşdirmədən onunla fərqlənir ki, yekun VY—ə *aparıcı* birləşdirmə cədvəlinin sahələri də daxil edilir. Hansı cədvəlin aparıcı olmasını *birləşdirmə növü* müəyyən edir:

- LEFT – (sol xarici birləşdirmə) 1-ci cədvəl aparıcı olur (birləşdirmə növü–ndən solda yerləşən cədvəl);

- RIGHT – (sağ xarici birləşdirmə) 2-ci cədvəl aparıcı olur (birləşdirmə növü–ndən sağda yerləşən cədvəl);

- FULL – (tam xarici birləşdirmə) hər iki cədvəl aparıcı olur.

Sol birləşdirmədə (LEFT JOIN) yekun VY–ə sol cədvəlin bütün yazıları, sağ cədvəlin isə yalnız birləşmə şərtini ödəyən yazıları daxil edilir. Əgər sağ cədvəldə verilən şərti ödəməyən yazılar olarsa, ikinci cədvəlin qiyməti kimi Null daxil edilir.

Sağ birləşdirmədə (RIGHT JOIN) yekun VY–ə sağ cədvəlin bütün yazıları, sol cədvəlin isə yalnız birləşmə şərtini ödəyən yazıları daxil edilir.

Tam birləşdirmədə (FULL JOIN) yekun VY–ə həm sol, həm də sağ cədvəlin bütün yazıları daxil edilir. Birləşdirmə şərti ödəndikdə hər iki cədvəlin yazıları seçilir, şərt ödənmədikdə isə sağ və sol cədvəllərin olmayan qiymətlərinin yerinə Null qiyməti daxil edilir.

Yuxarıda göstərdiyimiz misalı xarici birləşdirmə kimi həll etsək, həmin sorğuya uyğun SQL– kodlarını belə yazı bilərik:

```
SELECT M.QN, T.SAA, F.FA, M.QIY  
FROM MUVEFFEQIYYET M JOIN TELEBE T  
ON M.FK = '01' AND M.QIY IS NOT NULL  
JOIN FENN F  
ON M.FK = F.FK  
ORDER BY M.QN
```

Bu kodlarda Siz özünüz JOIN operandını LEFT JOIN (və ya RIGHT JOIN) operandı ilə əvəz etməklə müvafiq dəyişikliyi izləyə bilərsiniz.

3.2.6. Parametrlı sorğular

Delphi parametrlı sorğu adlanan daha çevik sorğu forması yaratmağa imkan verir. Belə sorğularda WHERE və INSERT ifadələrində ayrı–ayrı sözlərin yerinə dəyişənin (parametrin)

qiymətini qoymaq olar. Bu parametrin qiyməti istənilən zaman dəyişə bilər.

Parametrlı sorğuları öyrənmək üçün SQL operatorlarından birinə təkrarən diqqət yetirək:

```
SELECT QN,SAA,OBAL  
FROM TELEBE WHERE OBAL = 48
```

Bu kodun sağ tərəfini BAL dəyişəni ilə əvəz etməklə onu parametrlı sorğuya çevirmək olar:

```
SELECT QN,SAA,OBAL  
FROM TELEBE WHERE OBAL = :BAL
```

Bu operatorla BAL dəyişəninin qiyməti əvvəlcədən təyin edilməmişdir və o, istənilən zaman istənilən qiyməti ala bilər. BAL dəyişəni parametrdir. *Parametr – qarşısında : simvolu yazılmış addır.* Parametrin adı sahənin adı ilə eyni olmaya bilər.

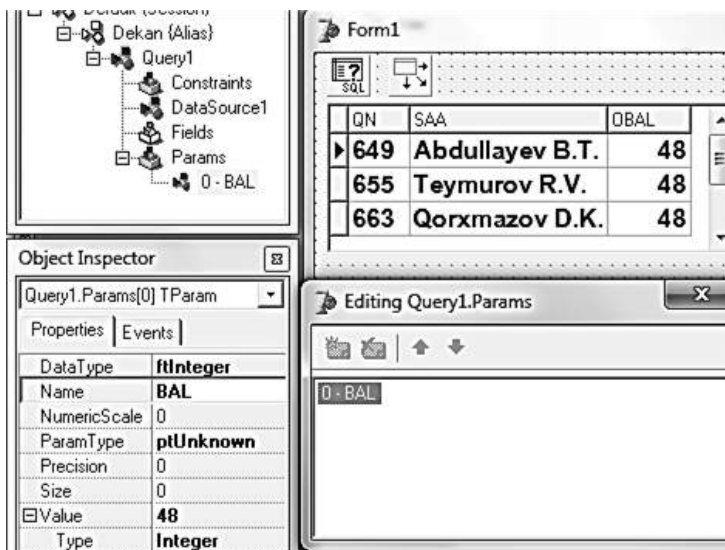
Sorğu mətnini araşdıran xüsusi **SQL parser** proqramı anlayır ki, o, sabitlə deyil, parametrlə işləyir, çünki, parametrin qarşısında : simvolu vardır. : simvolu parametr əlamətidir. Bu simvol proqramı xəbərdar edir ki, BAL dəyişəni parametrdir və bir azdan ona qiymət veriləcəkdir.

Parametrə qiymətlər bir neçə üsul ilə verilə bilər. Obyektlər inspektorunda bu parametrə qiymət vermək üçün əvvəlcə SQL kodlarını Query komponentinin SQL xassəsinə daxil edib Params xassəsi qarşısında mausun düyməsini basmaq lazımdır. Açılan Editing Query1.Params dialoq pəncərəsində (şəkil 3.10). Siz artıq daxil etdiyiniz parametrin adını və onun nömrəsini (0) görə bilərsiniz. Ümumiyyətlə, bu xassə SQL xassəsini avtomatik olaraq izləyir və hər yeni əlavə edilən parametr Params xassəsində qeyd olunur. Həmin pəncərədə parametrin adını seçdikdə Obyektlər inspektorunda həmin parametrə aid xassələr təsvir olunacaqdır. Value xassəsinə parametrin ala biləcəyi qiymətlərdən birini, məsələn, 48 daxil edib, Query komponentini aktivləşdirdikdə sorğu icra olunacaqdır.

Bundan başqa, parametərə daha iki yolla qiymət mənimsətmək olar:

- Query komponentinin Params xassəsi vasitəsi ilə və ya
- digər verilənlər yığımından informasiya almaq üçün DataSource xassəsi vasitəsi ilə.

Sorğunun özünü isə ya Obyektlər inspektorunda, ya da proqram yolu ilə tərtib etmək olar.



Şəkil 3.10. Obyektlər inspektoru vasitəsi ilə parametərə qiymətin verilməsi

3.2.6.1. Parametrlərə qiymətlərin Params xassəsi vasitəsi ilə proqram yolu ilə verilməsi

Proqram yolu ilə parametrlərə qiymətlər bir neçə üsulla verilə bilər:

• İndeks vasitəsi ilə. Məsələn, SQL–sorguda birinci parametərə `Params[0]`, ikinci parametərə `Params[1]` və s. kimi müraciət olunur;

• `ParamByName` metodunda parametrin adını göstərməklə. Məsələn, `BAL` parametrinə müraciət etmək üçün `ParamByName(BAL)` yazmaq lazımdır;

• `TParams` sinfinin `ParamValues` xassəsi vasitəsi ilə;

• `TParams` sinfinin `Values` xassəsi vasitəsi ilə.

Parametərə konkret qiymət verdikdə onun tipini göstərmək üçün `TParams` sinfinin xassələrindən biri (məsələn, `asString`, `asInteger` və s.) və ya `Variant` tipli `Value` xassəsi istifadə edilir. Məsələn,

```
SELECT QN,SAA,OBAL FROM TELEBE WHERE
OBAL=:BAL
```

sorgusu üçün `BAL` parametrinə qiymət aşağıdakı üsullarla verilə bilər:

```
Params[0].asInteger:= 48;
ParamByName('BAL').asInteger:= 45;
Params[0].Value:= 50;
Params[0].asInteger:=
StrToInt(Edit1.text);
Params.ParamValues['BAL']:=
StrToInt(Edit1.text);
```

Daha bir misal:

```
SELECT QN,SAA,OBAL FROM TELEBE
WHERE SAA Like :AD
```

sorgusunda sətir tipli `AD` parametri istifadə edilir. Onun üçün qiymət aşağıdakı üsullarla verilə bilər:

```
Params[0].asString:= 'A%';
ParamByName('AD').asString:= '%M%';
Params[0].Value:= '%J.';
Params[0].asString:= Edit1.text;
Params.ParamValues['AD']:= Edit1.text;
```

Parametrin ala biləcəyi qiymətlər əvvəlcədən məlum olduqda ComboBox komponentindən istifadə etmək olar ki, parametrin qiymətlərini bu siyahıdan seçmək mümkün olsun.

Parametrə qiymətləri Params xassəsi vasitəsi ilə mənimsətdikdə aşağıdakı dörd addımı icra etmək lazımdır:

- Query komponentini bağlamaq;
- Prepare metodunu çağırmaqla *sorğunu icra olunmaq üçün hazırlamaq*;
- Params xassəsinə parametrin aldığı qiymətləri mənimsətmək;
- Query komponentini açmaq.

Parametrə qiymət verənə qədər SQL–sorğu tərtib edilməlidir. Prepare metodu məhz bu məqsədə xidmət edir. O, sorğunun sintaksisini yoxlayır, onu kompilyasiya edir və sorğunu BDE–nin bufer yaddaşında saxlayır. Prepare metodu sorğu yalnız birinci dəfə icra olunduqda işləyir, növbəti dəfələrdə o icra olunmur və ona görə də sintaksisin yoxlanılmasına əlavə vaxt sərf edilmir. Parametrlərə qiymətləri bu methodsuz da mənimsətmək olar, bu halda sorğu ləng icra olunacaqdır. Əgər bu metodu aşkar göstərməsək, onda sorğu icra edilməzdən əvvəl Prepare metodu hər dəfə çağrılacaqdır. Sorğu başa çatdıqda isə UnPrepare metodu çağrılacaqdır. UnPrepare metodu resursları təmizləmək üçündür. Beləliklə, ikinci addım yalnız sorğu ilk dəfə yerinə yetirildikdə lazımdır, sonradan onu istifadə etməmək olar.

Yuxarıda sadaladığımız addımları kodlar vasitəsi ilə belə təsvir etmək olar:

```
Query1.Close;  
Query1.Prepare;  
Query1.Params[0].asString := 'A%';  
Query1.Open;
```

Bu kodları izah edək. İzahata üçüncü sətirdən başlayaq, çünki Params metodu bu prosesin "canıdır", həm də Prepare metodu ilə artıq müəyyən dərəcədə tanış olmuşuq.

Params – indeksli xassədir. SQL–sorguda birinci parametərə müraciət etmək üçün Params massivinin sıfırıncı elementinə müraciət etmək lazımdır:

```
Params[0].asString:= 'A%';
```

Əgər parametrlı sorğu

```
SELECT QN,SAA,OBAL FROM TELEBE  
WHERE SAA Like :AD
```

kimi olarsa, onda sorğunun faktiki nəticəsi

```
SELECT QN,SAA,OBAL FROM TELEBE  
WHERE SAA Like 'A%'
```

olacaqdır. Yəni burada sadəcə olaraq AD dəyişəninə A% qiyməti mənimsədilir.

Əgər sorguda bir neçə parametr olarsa, onda onlara Params massivinin növbəti indeksləri ilə, məsələn,

```
Params[1].asInteger:= 50;
```

və ya

```
ParamByName('BAL').asInteger:= 50;
```

kimi müraciət etmək olar.

Beləliklə, Prepare metodu ilə sorğunu hazırladıqdan və Params metodu ilə parametrlərə qiymətlər mənimsətdikdən sonra Open metodu çağrılmalıdır ki, dəyişənlərin əlaqələndirilməsi başa çatsın və tələb olunan verilənlər yığını alınmış olsun.

Bütün bu mülahizələri konkret sorğu yaratmaqla yekunlaşdıraq. MUELLIM cədvəli əsasında elə bir sorğu yaradaq ki, baş hərfləri seçdikdə soyadları bu hərflə başlayan müəllimlərin siyahısından ibarət cədvəl alınsın. Bu məqsədlə forma üzərinə ənənəvi Query, DataSource və DBGrid komponentlərindən başqa, Win 3.1 səhifəsindən TabSet komponenti də

yerləşdirək. Birinci üç komponenti bir–biri ilə əlaqələndirin. TabSet komponentini DBGrid komponentinin alt hissəsində yerləşdirin. Bu komponent çoxmövqeli idarəedici elementdir (onun əvəzinə TabControl komponentini də istifadə etmək olar). Biz burada baş hərflərdən ibarət əlfəcin yaradacağıq. TabSet komponentinin əsas xassəsi olan Tabs xassəsinə kiril əlifbasının baş hərflərini yerləşdirəcəyik. Bu prosedur belədir:

```
procedure TForm1.FormCreate(  
    Sender: TObject);  
var  
    i : Byte;  
begin  
    Query1.Prepare;  
    for i:=0 to 25 do  
        TabSet1.Tabs.Add(Chr(Byte('À')+i));  
end;
```

Bu prosedurla həm də sorğu hazırlanır (Query1.Prepare;)

Növbəti addımda TabSet1 komponenti üçün OnChange hadisə emaledicisi yaradıb onun vasitəsi ilə parametrin qiymətini Params xassəsinə ötürəcəyik, yəni:

```
procedure TForm1.TabSet1Change(  
    Sender: TObject; NewTab: Integer;  
    var AllowChange: Boolean);  
begin  
    with Query1 do begin  
        Close;  
        Params[0].AsString:=  
            TabSet1.Tabs.Strings[NewTab]+'%';  
        Open;  
    end;
```

Bu prosedurla TabSet1 komponentinin Tabs xassəsindən baş hərflər Params xassəsinin sıfırıncı elementinə ötürülür və axtarış baş hərfləri ilə aparıldığı üçün hərflərə % simvolu da əlavə

edilir. Bu sorğuya uyğun Delphi əlavəsinin nəticəsi şəkil 3.11-də göstərilmişdir. Əlavədə seçilən hərflərə uyğun olaraq müxtəlif müəllimlər haqqında məlumat təsvir olunur.



TABN	KAFK	SAA	ED	ER
102	01	Qaralov D.D.	T.e.n.	dosent
104	01	Qurbanov S.D.	F.r.e.n.	dosent
203	02	Qurbanov R.R.	F.r.e.d.	professor
304	03	Qarayev T.A.		
401	04	Quliyev H.P.	Tar.e.d.	professor

Şəkil 3.11. Baş hərflər əsasında seçmə aparən sorğu əlavəsi

3.2.7. Verilənlərin modifikasiyası

VB-də verilənlərin dəyişdirilməsi DML dilinin aşağıdakı üç operatoru vasitəsi ilə həyata keçirilir:

- INSERT – yeni yazıların əlavə edilməsi;
- UPDATE – yazıların dəyişdirilməsi (yeniləşdirilməsi);
- DELETE – yazıların pozulması.

Bu operatorlar vasitəsi ilə yalnız bir yazı və ya yazılar qrupu dəyişdirilə bilər.

Verilənləri modifikasiya etdikdə Open metodunu deyil, ExecSQL metodunu çağırmaq lazımdır. Ümumiyyətlə, Open metodu o zaman tətbiq edilir ki, SELECT operatoru ilə verilənlər yığımları tərtib etmək lazım gəlir, əgər belə VY yaratmağa ehtiyac yoxdursa, onda ExecSQL metodu tətbiq edilir. Sadaladığımız bu üç operator heç bir VY yaratmadığı üçün onları tətbiq etdikdə biz ExecSQL metodunu çağıracağıq.

3.2.7.1. INSERT operatoru

SQL dilinin sadə variantında bu operatorun ümumi forması belədir:

```
INSERT INTO cədvəlin adı (<1-ci sahə>, <2-ci sahə>, ...)
VALUES (<1-ci qiymət>, <2-ci qiymət>, ...)
```

Operator icra olunduqda *cədvəlin adı* ilə göstərilmiş *cədvəl*in *1-ci sahəsinə 1-ci qiymət*, *2-ci sahəsinə 2-ci qiymət* və s. mənimsədir. Məsələn, aşağıdakı operatorla QRUP *cədvəlinə* yeni yazı əlavə ediləcəkdir:

```
INSERT INTO QRUP (QN, SAY, OBAL)
VALUES ('999', 25, 50)
```

Bu operatorla QRUP *cədvəlinin* bütün sahələri sadalanır və onlar məhz *cədvəl* yaradıldıqda göstərilmiş ardıcılıqla yazıldığı üçün onu daha sadə yazmaq olar:

```
INSERT INTO QRUP
VALUES ('999', 25, 50)
```

3.2.7.2. UPDATE operatoru

Bu operator *cədvəl*in bir yazısının və ya yazılar qrupunun qiymətlərini dəyişdirmək üçün istifadə edilir. Operatorun ümumi forması belədir:

```
UPDATE cədvəlin adı
SET 1-ci sahə = <1-ci qiymət>, 2-ci sahə = <2-ci qiymət>, ...
[ WHERE axtarış şərti ]
```

Göründüyü kimi, UPDATE operatorundan sonra birbaşa *cədvəl*in adı yazılır, sonra isə SET operandında *cədvəl* sahələrinin adları və onlara mənimsədiləcək qiymətlər yazılır. Diqqət yetirin

ki, əgər WHERE *axtarış şərti* operandını yazmasanız cədvəlin bütün yazıları dəyişdiriləcəkdir.

QRUP cədvəlinin SAY sahəsində (*tələbələrin sayı*) sayı 1–ə bərabər olan qiymətləri 10 vahid artırmaq:

```
UPDATE QRUP
SET SAY=SAY+10
WHERE SAY=1
```

3.2.7.3. DELETE operatoru

Bu operator cədvəlin bir yazısını və ya yazılar qrupunu pozmaq üçün istifadə edilir. Operatorun ümumi forması belədir:

```
DELETE FROM cədvəlin adı
[ WHERE axtarış şərti ]
```

Operator icra olunduqda *axtarış şərti*-ni ödəyən bütün yazılar cədvəldən pozulur. UPDATE operatorunda olduğu kimi, əgər WHERE *axtarış şərti* operandını yazmasanız cədvəlin bütün yazıları pozulacaqdır.

QRUP cədvəlindən qrup nömrəsi 901 olan (QN sahəsi) olan yazıları pozaq:

```
DELETE FROM QRUP
WHERE QN='901'
```

Hər hansı qrupu pozmaq üçün parametrlı sorğu yaratmaq daha məqsəduyğun ola bilər:

```
DELETE FROM QRUP
WHERE QN= :qrN
```

qrN parametrinə qiyməti isə yuxarıda öyrəndiyimiz qaydalarla vermək olar, məsələn, proqramın yerinə yetirilməsi zamanı bu parametərə 901 qiyməti vermək üçün

```
Query1.Prepare;  
Query1.Params[0]:='901';  
Query1.ExecSQL;
```

yaza bilərik. Burada, SQL – kodlarını araşdırmaq və Params xassəsini hazırlamaq üçün əvvəlcə Prepare metodu çağırılır. Növbəti addımda Params xassəsinə 901 qiyməti mənimsədir və hazırlanmış SQL–sorgu yerinə yetirilir. Sorgunu yerinə yetirmək üçün ExecSQL metodu istifadə edilir.

D ö r d ü n c ü f ə s i l

DELPHI–nin DİGƏR VERİLƏNLƏR BAZASI CƏDVƏLLƏRİ İLƏ ƏLAQƏSİ

Obyektyönlü proqramlaşdırmanın əsas prinsiplərindən biri odur ki, haçansa, kim tərəfindənsə yaradılmış proqram məhsulu itməməli, batmamalıdır. Əgər Ms Excel elektron cədvəlində, Ms Access VBİS–də və s. hazır cədvəllərimiz varsa, onda bizim həmin cədvəlləri Paradox–da, dBase və s. sistemlərdə təkrarən yenidən yaratmağımıza ehtiyac yoxdur. Xüsusən də nəzərə alsaq ki, bu gün Access–lə işləmək Paradox və s. ilə işləməkdən daha asandır, onda baxdığımız hal daha aktuallıq kəsb edir. Biz bu fəsilə Ms Access və Ms Excel elektron cədvəllərinin ADO texnologiyası ilə Delphi–də istifadə edilməsindən, onların əsasında Delphi–də sorğu, hesabat və s. yaradılmasından bəhs edəcəyik. Verilənlərə müdaxilə mexanizmləri Delphi ilə inteqrasiya kontekstində şərh ediləcəkdir

4.1. Verilənlərə müdaxilə mexanizmləri

Verilənlər bazası texnologiyalarının əmələ gəldiyi andan proqramçılar bu verilənləri əldə etmək zərurəti ilə üzləşmişlər. Proqramçıların bu tələbatını ödəmək üçün müxtəlif şirkətlər özünəməxsus şəkildə bu məsələni həll etməyə cəhd etmişlər. Məsələn, *dBase* tipli cədvəllərlə işləmək üçün *Clipper* adlanan VBİS yaradılmışdı. **Ms DOS** əməliyyat sisteminin dövrü üçün bu ən yaxşı vasitə idi. Lakin *Clipper* digər cədvəllərlə işləyə bilmirdi və belə cədvəl tiplərinin sayı artdıqca daha universal müdaxilə

vasitələrinin yaranması zərurəti ortaya çıxırdı ki, o, bütün cədvəl tipləri ilə işləyə bilsin.

Həm lokal, həm də kliyent əlavələrindən verilənlərə müdaxilə (və ya verilənləri əldə etmək) üçün bir neçə üsul mövcuddur. Hər bir verilənlər bazasını idarəetmə sisteminin özünün tərkibinə xüsusi tətbiqi proqram interfeysi – *Application Programming Interface (API)* və **COM**–obyektlər (*Component Object Model*) daxildir ki, onlar, yalnız özünə aid verilənlərə müdaxilə edə bilər. *API* – funksiyalar yığımından ibarətdir. Əlavələrdə *API* və *COM*–obyektlərin istifadə edilməsi təbii və səmərəlidir, lakin onlar yalnız özlərinin məxsus olduqları konkret VBİS–lərdə verilənlər üzərində manipulyasiyalar etməyə imkan verir, belə ki, həm *API*, həm də *COM* obyekt modeli heç bir standartla təbə deyil və müxtəlif VBİS–lər üçün müxtəlifdir. Əlavələrdə verilənlərlə işləmək üçün mövcud olan digər üsul isə verilənlərə universal müdaxilə üsuluna əsaslanır.

Verilənlərə müdaxilə mexanizmi verilənlər bazasına və onun cədvəllərinə müdaxilə etməyə imkan verən proqram vasitəsidir. Adətən, bu proqramlar **.dll** genişlənmiş hissəli drayverlərdir. Bu drayverlər istifadəçinin fərdi kompüterinə (həm də kliyentin) yüklənməlidir və onlar VB ilə əlaqə yaradan proqramlar tərəfindən istifadə olunur. Əgər VBİS–i dəyişdirmək zərurəti ortaya çıxarsa, onda universal müdaxilə mexanizmlərindən istifadə edən əlavələrdə cüzi dəyişiklər etməklə onları asanlıqla modifikasiya etmək olar. Bütün üstünlüklərinə baxmayaraq universal müdaxilə mexanizmləri nöqsansız deyildir. Konkret VBİS–ə aid olan funksionallığın tam istifadə edilə bilməməsi, məhsuldarlığın aşağı düşməsi və prosedurların nisbətən mürəkkəbləşməsi verilənlərə universal müdaxilə mexanizmlərinin çatışmaz cəhətləridir. Sonuncu nöqsan onunla əlaqədardır ki, əlavənin tərkibinə universal mexanizmlərin icrası üçün məsul olan və onların düzgün fəaliyyət göstərməsini təmin edən kitabxanalar və drayverlər qoşmaq lazımdır.

Verilənlərə universal müdaxilə mexanizmlərinə aşağıdakılar aiddir:

- **ODBC** – *Open Database Connectivity*;
- **OLE DB** – *Object Linking and Embedding Database*;
- **ADO** – *ActiveX Data Objects*;

- **BDE** – *Borland Database Engine*;
- **DBExpress**.

OLE DB və *ADO* Microsoft firmasının verilənlərə universal müdaxilə mexanizminin (*Microsoft Universal Data Access*) tərkib hissələrindən biridir. Onlar həm relyasiya, həm də fayl sistemi, elektron–poçt verilənləri və s. kimi qeyri–rəlyasiya verilənlərini əldə etməyə imkan verir.

Beləliklə, verilənlər bazası ilə işləyən əlavələrdə verilənlərə aşağıdakı müdaxilə üsulları tətbiq oluna bilər:

- *API* funksiyalara və ya *COM*–obyektlərə birbaşa müraciət etməklə;
- *ODBC API* funksiyalarını çağırmaqla;
- *OLE DB* interfeysinə birbaşa müraciət etməklə;
- *ADO* texnologiyasını tətbiq etməklə;
- *ADO+OLE DB+ODBC* texnologiyalarını birgə tətbiq etməklə;
- *BDE+SQL Links* texnologiyalarını birgə tətbiq etməklə;
- *BDE+ODBC Link+ODBC* texnologiyalarını birgə tətbiq etməklə.

Verilənlərə universal müdaxilə mexanizmlərini izah edək.

BDE fiziki olaraq verilənləri əldə etmək üçün kitabxanalar yığımından ibarətdir. *BDE* vasitəsi ilə verilənlərə müdaxilə etmək üçün kompüterə ümumi təyinatlı *BDE* kitabxanası və konkret VBİS üçün drayverlər yüklənməlidir. Server VBİS–ləri üçün belə drayverlər *SQL Links* adlanır. *BDE* drayverləri içərisində *ODBC API* funksiyalarından istifadə etməklə yaradılmış *ODBC Link* adlanan drayver də var ki, *ODBC* drayveri ilə birlikdə seçilmiş VBİS üçün tətbiq edilir. *Paradox*, *dBase* və mətn fayllarının verilənlərinə müdaxilə üçün *BDE*–nin birbaşa müraciət drayverləri, *Oracle*, *Sybase*, *IBM DB2*, *Informix*, *InterBase* server VBİS–ləri üçün isə *SQL Links* drayverləri mövcuddur. Onlara *ODBC Link* və *ODBC* drayverləri ilə də müraciət etmək olar. Digər VBİS–lərin verilənlərinə isə yalnız *ODBC Link* və müvafiq *ODBC* drayverləri ilə müraciət etmək olar.

BDE Borland firmasının məhsuludur. Bu mexanizm həm lokal, həm də fayl–server formatlı *dBase*, *FoxPro* və *Paradox* verilənlər bazasına, müxtəlif *SQL* serverlərinə və *ODBC* drayverlərinin köməyi ilə müdaxilə oluna biləcək bir sıra verilənlər

mənbəyinə müdaxilə etməyə imkan verir. Məsələn, *BDE* vasitəsi ilə *Ms Excel* cədvəlləri ilə birbaşa işləmək olar. Təəssüflər olsun ki, *BDE* mexanizmi artıq köhnəlmişdir. Bunu Borland firması özü də nəinki təsdiq edir, hətta onu progressiv olmadığı üçün inkişaf da etdirmir. Hal-hazırda Delphi-nin əksər vasitələri kros-platforma xarakterlidir, yəni cüzi dəyişikliklər etməklə onları digər əməliyyat sistemlərinə köçürmək olar. Borland firması yeni – *Kylix* – proqramları cəld işləmə – mühiti yaratmışdır, onun köməyi ilə *Linux* əməliyyat sistemi üçün əlavələr yaratmaq olur. Əslində *Kylix* mühiti *Linux* üçün Delphi variantıdır – Delphi-ni bilənlər *Kylix*-də də proqramlar tərtib edə bilirlər.

Bütün bunlara baxmayaraq, proqramçılar çox da böyük olmayan şirkətlər üçün VB əlavələri işlədikdə *BDE* vasitələrindən istifadə edirlər. Məsələn, həm şəhər, həm də mobil telefonlar üçün müasir elektron ATS-lər hazırlayan *Huawei* Çin şirkəti bu ATS-lərin statistik verilənlərinə müraciət üçün hələ də *BDE* vasitələrindən istifadə edir. Bunun səbəbi ondadır ki, *BDE* proqramçılar üçün proqram yolu ilə cədvəllərin hazırlanması kimi çoxlu sayda çox sadə və rahat imkanlara malikdir.

BDE-nin əsas çatışmaz cəhəti əlavələrin geniş yayılmasına imkan verə bilməməsidir. Bir kompüterdə yaradılmış əlavəni digər kompüterdə istifadə etmək üçün həmin kompüterə də *BDE* komponentləri yüklənməlidir. Aliasdan istifadə etdikdə isə Sizin proqramınızı istifadə edən kliyent də öz əlavəsini həmin aliasa kökləməlidir. Doğrudur, bu vəziyyəti **InstallShield Express** utilitlərindən istifadə edərək qurma paketi yaratmaqla aradan qaldırmaq olar, lakin bu özü də vaxt və yaddaş həcmi tələb edir. *BDE*-nin digər çatışmazlığı bütün müdaxilə mexanizmlərinə xas olan çatışmazlıqdır, yəni prosedurlar “ağırlaşır”.

ODBC (*Open Database Connectivity*) – Microsoft firmasının geniş yayılmış proqram interfeysidir. *ODBC* vasitəsi ilə konkret VBİS-in verilənlərinə müraciət üçün bu VBİS-in məxsusi kliyent hissəsindən başqa, ona müraciət etmək üçün *ODBC Administrator* və *ODBC* drayverləri lazımdır. *ODBC Administrator* verilmiş kompüter üçün *ODBC*-in köməyi ilə hansı verilənlər mənbəyinin əlçatan olduğunu müəyyən etməyə və yeni verilənlər mənbəyini təsvir etməyə imkan verən əlavədir. *ODBC* drayverləri dinamik yüklənən kitabxanadan ibarətdir. Kliyent

əlavəsi bu kitabxanayı özünün ünvan sahəsinə yükləyə bilər və verilənlər mənbəyinə müraciət etmək üçün istifadə edə bilər. Verilənlərlə işləmək üçün həm birbaşa *ODBC API* funksiyaları, həm də universal müdaxilə mexanizmləri, məsələn, *OLE DB*, *ADO*, *BDE* istifadə oluna bilər. İndiki anda *ODBC* daha çox istifadə olunan universal müdaxilə mexanizmi olduğundan, belə oxşar komponentlərin gələcəkdə daha çox istifadə olunması şübhə doğurmur. *ODBC*–mənbələrinə müdaxilə üçün komponentləri istifadə edən əlavələr, *BDE* və *ODBC Link* mexanizmlərini istifadə edən əlavələrdən daha məhsuldar olur, çünki onlar *BDE* əlavə kitabxanalarından imtina edir.

OLE DB (*Object Linking and Embedding Database*) – Microsoft firmasının bu texnologiyası *COM* (*Component Object Model – Komponentlərin obyekt modeli*) interfeyslər yığımından ibarətdir. *OLE DB* – standart *COM*–interfeysləri vasitəsi ilə tipi və yerləşdiyi yerdən asılı olmayaraq istənilən verilənlərə müdaxilə mexanizmidir. Belə verilənlər kimi verilənlər bazası, *Ms Excel* cədvəlləri, mətn sənədləri və digər istənilən verilənlər mənbəyini göstərmək olar. *ODBC* drayverləri ilə müdaxilədən fərqli olaraq, *OLE DB* texnologiyası *SQL* dilini tətbiq etməklə həm *SQL*–serverlərə, həm də digər istənilən verilənlər mənbəyinə müdaxilə edə bilər. *OLE DB* texnologiyası ilə verilənlərə müdaxilə mexanizmini təmin edən vasitələrə *OLE DB – provayderlər* deyilir. *OLE DB* texnologiyası ilə verilənlərə müdaxilə üçün kliyent əlavəsinin istifadə edildiyi kompüterdə verilmiş VBİS üçün *OLE DB*–provayder qurulmalıdır. Verilənlər mənbəyinə müraciət üçün istifadə edilən *OLE DB*–provayder dinamik kitabxanadan ibarətdir və kliyent əlavəsinin ünvan sahəsinə yüklənir. Hər bir VBİS üçün onun özünəməxsus provayderi olmalıdır, çünki bu provayderlər müxtəlif VBİS–lər üçün fərqli olan müxtəlif *API* funksiyalarına əsaslanır. Əgər konkret verilənlər mənbəyinə müdaxilə üçün yalnız *ODBC*–drayverlər mövcuddursa, onda *OLE DB* texnologiyasını tətbiq etmək üçün, *ODBC*–mənbələrə müraciət üçün nəzərdə tutulmuş *OLE DB*–provayderdən istifadə etmək olar. Bu provayder hər hansı VBİS–in kliyent hissəsinin *API* funksiyalarını deyil, *ODBC API* interfeysini istifadə edir, ona görə də o, seçilmiş VBİS üçün *ODBC*–drayveri ilə birlikdə tətbiq olunur. *OLE DB* interfeyslər yığını *OLE DB* obyektləri (*COM*) ilə reallaşır. *OLE*

DB-nin baza modellərinə *DataSource*, *Session*, *Rowset* obyektləri daxildir.

DataSource (*verilənlər mənbəyi*) obyektı verilənlər mənbəyi ilə əlaqə yaratmaq və bir neçə seans yaratmaq üçün nəzərdə tutulmuşdur. Bu obyekt birləşməni idarə edir.

Session (*seans*) obyektı verilənlər mənbəyi ilə qarşılıqlı əlaqəni idarə edir, sorğuları yerinə yetirir. Seans zamanı bir neçə əmərlər yaradıla bilər.

Rowset (*nəticə yığımı*) obyektı əmrin yerinə yetirilməsi nəticəsində əldə edilən və ya seansda yaradılan verilənlərdən ibarətdir. Burada əsas rolu mətn formatlı əmr icra edən ***Command*** (əmr) obyektı oynayır. Belə əmərlər *SQL*-əmərləri ola bilər. Tranzaksiyaları idarə etmək üçün isə ***Transaction*** (tranzaksiya) obyektı nəzərdə tutulmuşdur.

dbExpress– Borland firmasının özünün *BDE* mexanizminə alternativ olaraq yaratdığı mexanizmdir. *dbExpress* verilənlər mənbəyi ilə birləşmə, tranzaksiya və sorğularla işləyən drayverlər və komponentlər yığımıdır. *dbExpress* verilənləri əldə etmək üçün *SQL* dilindən istifadə etməklə drayverlər vasitəsi ilə VBİS-lə ünsiyyət qurur. Bu zaman kliyent əlavəsi tərəfində verilənlər keşləşdirilmir, bir istiqamətli axın istifadə olunur və cədvəllərə birbaşa düzəlişlər etmək mümkün olmur. Ümumiyyətlə, istənilən halda, *dbExpress* lokal VB-lərlə iş üçün nəzərdə tutulmamışdır. *dbExpress* tərəfindən dəstəklənən VBİS-lərə *DB2*, *Oracle*, *Ms SQL* və *MySQL* aid etmək olar. *Interbase* VBİS də *dbExpress* tərəfindən dəstəklənir. Ancaq, *Interbase* VBİS üçün *dbExpress* mexanizminin istifadə edilməsi o qədər də yaxşı qərar deyildir. Çünki, Delphi-də daha bir texnologiya, daha doğrusu komponentlər yığımı – *IBExpress* mövcuddur ki, o, *Interbase* VBİS-lə birbaşa qarşılıqlı əlaqə yarada bilər. Bu komponentlər Delphi-nin Komponentlər palitrasının *InterBase* səhifəsində yerləşir. Bu komponentlər *BDE*-nin bütün imkanlarını və habelə həm də *InterBase* VBİS üçün xarakterik olan imkanları istifadə etməyə imkan verir. Bundan başqa, *InterBaseAdmin* komponentlər yığımı da mövcuddur ki, onların köməyi ilə *InterBase* VBİS-in özünün üzərində manipulyasiyalar etmək mümkün olur.

4.2. ADO ilə iş

Kitabımızın bu fəslə məhz *ADO* ilə işə həsr olunduğu üçün verilənlərə bu müdaxilə üsulunu digərlərindən fərqləndirərək ayrı bir bölmədə izah etməyə üstünlük verdik. Beləliklə, **ADO** (*ActiveX Data Objects*) *OLE DB* interfeyslərindən istifadə edilməklə Microsoft firması tərəfindən yaradılmışdır. 1990-cı illərin ortalarından *COM* texnologiya sürətlə inkişaf etməyə başladı və bununla əlaqədar olaraq Microsoft firması köhnə *ODBC* texnologiyasından yeni *OLE DB* texnologiyasına keçdiyini elan etdi. Bununla bərabər, *OLE DB* texnologiyası kifayət qədər mürəkkəb texnologiyadır, bu texnologiyadan istifadə sistem səviyyəsində həyata keçir və proqramçıdan xeyli bilik və zəhmət tələb edir. Bundan başqa, *OLE DB* texnologiyası səhvlərə çox həssas olduğu üçün ilk fürsətdə tez “sıradan çıxır”. Ona görə də, proqramçıların işini asanlaşdırmaq üçün, Microsoft firması *ADO* texnologiyasını yaratdı. *ADO* – *COM*–obyektlərdən ibarət kitabxanalar yığımıdır ki, onlar verilənləri əldə etmək üçün tətbiqi proqram interfeyslərini realizasiya edir və kliyent əlavələrində istifadə olunur. *ADO* texnologiyası verilənlərə müdaxilə üçün aşağı səviyyəli interfeyslərdən ibarət *OLE DB* kitabxanalarından istifadə edir.

Öz imkanlarına görə *ADO* daha güclü texnologiya olmasına baxmayaraq, o, *BDE*–yə çox oxşayır. Borland firması *ADO* ilə işləmək üçün komponentlər yığını işləyib hazırladı və ilkin olaraq onu *ADOExpress* adlandırdı. Lakin, Microsoft firması ona aid olan şərti–işarələrin kənar firmaların məhsullarında istifadə edilməsinə qəti etiraz etdiyi üçün, Delphi 6 versiyasından başlayaraq bu komponentlər yığını *dbGo* adlandırılmağa başlandı.

BDE kimi *ADO* texnologiyası da konkret VB serverindən asılı deyil, həm müxtəlif tipli lokal verilənlər bazalarını, həm də bəzi kliyent–server tipli verilənlər bazalarını dəstəkləyir. *ADO* getdikcə daha çox populyarlaşır, çünki, o, bütün Windows ailəsi əməliyyat sistemlərinin nüvəsinə və çox geniş yayılmış *Ms Office* və *Ms Internet Explorer* proqram məhsullarının tərkibinə daxil edilmişdir. Bu isə o deməkdir ki, hansı kompüterdə Windows sistemi qurulmuşdursa, orada Sizin proqramınız işləyəcəkdir. Microsoft firmasının öz tələblərini ödəməsi üçün yaratdığı

drayverlər digər firmaların yaratdıqları drayverlərdən daha etibarlı olur. Ona görə də *Ms Acces* və ya *Ms SQL* kliyent–server arxitekturalı verilənlər bazası ilə işləmək zərurəti ortaya çıxdıqda, *ADO* texnologiyasına üstünlük vermək lazımdır. Bütün bunlar *ADO* texnologiyasının üstün cəhətləridir.

Nə qədər qərribə də olsa, *ADO* texnologiyasının çatışmaz cəhətlərindən biri elə həm də Windows sisteminin geniş yayılmasındadır. Microsoft firması kifayət qədər hiyləgərcəsinə hərəkət edir. Hər üç–beş ildən bir Windows sisteminin yeni versiyası peyda olur. Sırası istifadəçinin yeni əməliyyat sisteminə keçməsinə heç bir ehtiyacı olmadığı halda (xüsusən də nəzərə alsaq ki, bunun arxasında yeni kompüter resursları tələb olunur), istifadəçini yeni versiyaya keçməyə məcbur etmək məqsədi ilə, Microsoft firması hökmən köhnə versiyalarla uzlaşmayan bir neçə yeni standart və ya texnologiyalar daxil edir. Köhnə versiyaları isə yeni versiya dəstəkləmir. Yazıq istifadəçi isə əməliyyat sisteminin və *Ms Office* paketinin yeni versiyasını almaq məcburiyyətində qalır. Ona görə də *ADO* texnologiyasını istifadə etdikdə qabaqcadan nəzərə almalısınız ki, sonuncu istifadəçinin kompüterində Windows sisteminin hansı versiyası quraşdırılmışdır və Sizin proqramınız həmin kompüterdə işləyəcəkmi?

Əslində, *ADO* texnologiyası daha iri miqyaslı **Microsoft Data Access Components (MDAC)** texnologiyasının tərkib hissəsidir. *MDAC* anlayışı Microsoft firmasının VB ilə əlaqədar bütün proqram məhsullarının ümumi adıdır. *MDAC* texnologiyasının tərkibinə *ADO*, *OLE DB*, *ODBC* və **RDS** (*Remote Data Services*) daxildir. Bəzən insanlar *MDAC* və *ADO* anlayışlarını sinonim kimi eyniləşdirirlər, lakin bu düzgün deyildir, belə ki, *ADO* *MDAC*–ın tərkib hissələrindən biridir. *MDAC*–ın əsas versiyalarına 2.5, 2.6, 2.7 və 2.8 versiyaları aiddir. Microsoft onu ayrı bir proqram məhsulu kimi yayır və firmanın rəsmi saytından onu pulsuz yükləmək olar. Delphi 7–də *MDAC* 2.6 versiyasından istifadə olunur. Tam əmin ola bilərsiniz ki, *MDAC* artıq Sizin kompüterinizə yüklənmişdir, çünki, o, *Internet Explorer* brauzerinin tərkibində yüklənir və yeniləşir. Onu da əlavə edək ki, Microsoft firması *MDAC*–ın yalnız sonuncu və ondan əvvəlki versiyalarını dəstəkləyir. Ona görə də əmin ola bilərsiniz ki, Sizin əlavəniz ya ən sonuncu, ya da sonuncudan əvvəlki versiya ilə

işləyir. Windows 2000, Windows XP və daha yuxarı versiyalarda MDAC 2.6 versiyası istifadə olunduğu halda, Windows 7 və Windows 8 əməliyyat sistemlərində MDAC 2.8 versiyası tətbiq olunmuşdur. Ümumiyyətlə, ADO istifadəçiləri, MDAC-a aid suallara cavab almaq üçün, Microsoft firmasının rəsmi sahifəsinə (www.microsoft.com/data) müntəzəm olaraq müraciət etməlidirlər.

ADO-nun daha bir ciddi çatışmazlığı var: VB-yə qoşulmaq üçün o, kifayət qədər ləng işləyən COM texnologiyanı istifadə edir. Əgər baza bir neçə min yazılardan ibarət olarsa, onda hesab edin ki, sürət BDE-yə nisbətən yüz dəfələrlə az olacaqdır. Lakin 2 QHs və daha çox tezliyə malik kompüterlərdə belə ləngimə o qədər də hiss olunmayacaqdır.

4.3. Delphi-də dəstəklənən müdaxilə mexanizmləri və ADO ilə iş üçün komponentlər

Verilənlər bazası əlavələrini yaratmaq üçün Delphi-də mövcud olan sinif və komponentlər aşağıdakı qruplara bölünür:

1. ODBC drayverlərindən və ya BDE daxili drayverlərindən istifadə etməklə BDE verilənlər prosessoru vasitəsi ilə verilənlərə müdaxilə edən komponentlər:

- əsasına OLE DB texnologiyasının tətbiq edildiyi ADO-obyektlərlə müdaxilə;

- lokal və ya məsafədə yerləşən InterBase SQL-serverinə müdaxilə;

- dbExpress drayverləri vasitəsi ilə müdaxilə;

- çoxmənqəli arxitektura VB-ə müdaxilə (DataSnap sahifəsinin komponentləri);

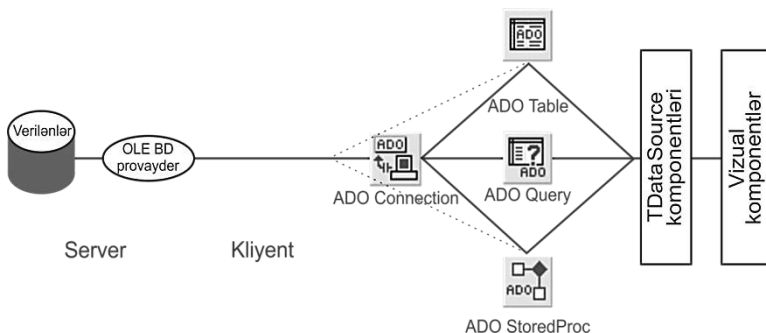
2. Verilənlər mənbəyi ilə vizual komponentlər arasında əlaqə yaratmaq üçün istifadəçinin interfeysini təşkil edən komponentlər;

3. İstifadəçi interfeysini icra edən vizual komponentlər;

4. Hesabatı vizual layihələndirmək üçün komponentlər.

Bu mexanizmlərin verilənlərlə iş sxemləri bir-birinə oxşardır. Biz BDE-nin daxili drayverlərindən (Paradox formatlı) istifadə etməklə iş sxemini öyrəndik. ODBC-drayverlərini istifadə etməklə BDE ilə iş analoji qaydada yerinə yetirilir.

ADO ilə işlədikdə Delphi–nin VB əlavələri yaratmaq arxitekturası şəkil 4.1–də göstərilmişdir.



Şəkil 4.1. Delphi–də ADO texnologiyasının arxitekturası

Komponentlərlə işlədikdə verilənlər moduluna (və ya formaya) verilənlər yığımı komponenti əlavə edilir (TDataSet sinfinin obyektləri) və DatabaseName xassəsinə qiymət verməklə verilənlər mənbəyi ilə əlaqə yaradılır. Əlaqə VB–nin adı, qovluq və ya psevdonim (alias) vasitəsi ilə yaradılır (bu verilənlər mənbəyinin tipindən asılıdır). Verilənlər moduluna (və ya formaya) verilənlər mənbəyi komponenti (TDataSource) əlavə edilir ki, bu komponent verilənlər yığımı ilə verilənləri təsvir etdirən idarəedici elementlər arasında (məsələn, TDBGrid) əlaqələndirici rol oynayır. TDataSource komponentinin DataSet xassəsi TTable və ya TQuery kimi siniflərin komponentləri ilə yaradılan verilənlər yığımını müəyyən edir. Sonra isə bu verilənlər yığımı ilə işləmək üçün formaya TDBGrid, TDBEdit, TDBCheckbox və s. komponentlər yerləşdirilir.

Verilənlərə ADO, dbExpress mexanizmləri ilə müdaxilə sxemində isə yanaşma olduğu kimi qalır, lakin digər komponentlər istifadə olunur.

Bütün verilənlər yığımı sinfinin əcdadı `TDataSet` sinfidir. Əlavədə istifadə olunan müdaxilə mexanizmindən asılı olaraq verilənlər yığımının baza sinfi aşağıdakılar ola bilər:

- `TTable`, `TQuery`, `TStoredProc` – *BDE* istifadə edən bir və ya iki mənbəli əlavələr üçün;

- `TClientDataSet` – kliyent verilənlər yığımını və paylanmış müdaxilə istifadə edən çoxmənbəli arxitekturanı realizasiya etmək üçün;

- `TADODataSet` – *ADO*–obyektlər istifadə edən əlavələr üçün;

- `TSQLDataSet` – `dbExpress` vasitəsi ilə verilənlər bazasına müdaxilə üçün.

ADO texnologiyasını tətbiq etmək üçün Delphi 7-də Komponentlər palitrasının *ADO* səhifəsində 7 komponent nəzərdə tutulmuşdur.

4.3.1. ADO komponentləri

4.3.1.1. TADOConnection komponenti

`TADOConnection` komponenti verilənlər mənbəyinə qoşulmaq üçün nəzərdə tutulmuşdur. O, Access, Excel və s. faylları ilə əlaqə yaradır. Belə əlaqə onun əsas xassəsi olan `ConnectionString` sətri vasitəsi ilə yaradılır. Verilənlərlə əlaqə yaradıldıqdan sonra bir `TADOConnection` komponenti vasitəsi ilə bir neçə `TADOTable` və `TADOQuery` komponentləri bazanın müxtəlif cədvəlləri ilə əlaqələndirilə bilər. Bu komponentin bəzi xassələrini nəzərdən keçirək.

`Connected` xassəsi verilənlər mənbəyi ilə əlaqəni açmaq (`True`) və ya bağlamaq (`False`) üçün istifadə edilir. Bu əməliyyatı, koddan istifadə etdikdə, müvafiq olaraq `Open` və `Close` metodları ilə də icra etmək olar.

`ConnectOptions` xassəsi əlaqənin tipini müəyyən edir. Bu xassəyə qiymət seçməklə sinxron və asinxron əlaqə yaratmaq mümkündür. Qeyd edək ki, *ADO*–nun belə əlaqə formasının *BDE* və *InterBase*–də analoqu yoxdur. Adi halda sinxron əlaqə

yaradılır. Server ləng işlədikdə asinxron əlaqə ilə işləmək daha sərfəlidir. Serverlə aşağıdakı əməliyyatlar asinxron yerinə yetirilə bilər:

- serverlə birləşmə (Connection);
- əməllərin yerinə yetirilməsi (Execute);
- verilənlərin seçilməsi (Fetch).

Serverlə birləşmək üçün `ConnectOptions` xassəsinə `coAsyncConnect` qiyməti vermək lazımdır.

Qeyd etmək lazımdır ki, ADO–nun əksər komponentləri aktivləşdirildikdə və ya icra olunduqda **SQL** dilində əməlləri emal edirlər. Bunlar `ADOCommand`, `ADODataset`, `ADOTable`, `ADOQuery` və `ADOSToredProc` kimi komponentlərdir. Əməlləri asinxron icra etmək üçün bu komponentlərin `ExecuteOptions` xassəsinin `coAsyncExecute` alt xassəsinə `True` qiyməti seçmək lazımdır.

Verilənlərin asinxron seçilməsi `ADOCommand`, `ADODataset`, `ADOTable`, `ADOQuery` və `ADOSToredProc` komponentləri tərəfindən dəstəklənir. Bu rejimi qoşmaq üçün həmin komponentlərin `ExecuteOptions` xassəsinin `coAsyncFetch` alt xassəsinə `True` qiyməti seçmək lazımdır.

`Provider` xassəsi `Connection` obyektinin hal–hazırda hansı provayderi istifadə etməsini göstərir. Yaradılmış əlaqə ilə müdaxilə hüququ `Mode` xassəsi ilə müəyyənləşdirilir. Bu xassə `ADOConnection` obyektinin `ConnectModeEnum` xassəsi ilə birbaşa əlaqədərdir. Bu xassə bir sıra qiymətlər ala bilər ki, onlardan bəziləri aşağıda sadalanmışdır:

- `cmUnknown` – heç bir məhdudiyyət qoyulmamışdır və ya təyin edilə bilmir;
- `cmRead` – yaradılmış əlaqə verilənləri yalnız oxuya bilər, onları dəyişdirə bilməz;
- `cmWrite` – yaradılmış əlaqə verilənləri yalnız dəyişdirə bilər, onları oxuya bilməz;
- `cmReadWrite` – yaradılmış əlaqə verilənləri həm oxuya, həm də dəyişdirə bilər;
- `cmShareExclusive` – digər istifadəçilərin əlaqəni açmasını qadağan edir.

`KeepConnection` xassəsi açıq verilənlər yığımı olmadıqda, yaratdığımız əlavənin verilənlər bazası ilə əlaqəsini dəstəkləyə bilməsini müəyyən edir. Onun qiyməti `True` olduqda əlaqə həmişə açıq vəziyyətdə olur; bu halda məsafədə yerləşən VBİS-lər və verilənlər yığımını tez-tez açıb–bağlayan əlavələr üçün şəbəkə trafikə əhəmiyyətli dərəcədə kiçilir və əlavənin işləmə sürəti artır.

`IsolationLevel` xassəsi tranzaksiyaların izolyasiya səviyyəsini müəyyən edir. Bu səviyyə cədvələ eyni zamanda müraciət edən müxtəlif birləşmələrlə tranzaksiyaların qarşılıqlı əlaqəsini və tranzaksiyaların görünmə oblastını təyin edir.

`Errors` xassəsi ADO səhvlər obyektinə birbaşa müraciət üçün nəzərdə tutulmuşdur. `DataSets` xassəsində komponentlə əlaqələndirilmiş aktiv verilənlər yığımı massivləri saxlanır. `DataSetCount` xassəsindən isə komponentlə əlaqələndirilmiş aktiv verilənlər yığımının sayını tapmaq olar.

4.3.1.2. TADODataset komponenti

Bu komponentin vəzifəsi ADO–nun əlaqə yaratdığı VB–nin cədvəllərindən verilənlər yığımını almaqdır. O, `SELECT` tipli sorğunun köməyi ilə bir və ya bir neçə cədvəllərdən yekun verilənlər yığımı almağa imkan verir. Bundan başqa, o, verilənləri təsvir etmək üçün vizual komponentlərlə də işləyə bilər. O, `ADOTable`, `ADOQuery` və ya `ADOSToredProc` komponentlərinin əvəzinə istifadə oluna biləcək daha ümumi verilənlər yığımı komponentidir. Bu komponentin `CommandText` xassəsi vardır ki, orada əmrələr mətnini yazmaqla verilənlər yığımını əldə etmək olar. Belə əmr kimi `SQL`–sorgu, cədvəlin adı və ya saxlanılan prosedur yazıla bilər. `SQL`–sorgu kimi isə yalnız `SELECT` operatoru yazıla bilər (`DELETE`, `UPDATE`, `INSERT` operatorları yazıla bilməz). Komponentin `CommandType` xassəsində isə `SQL`–sorgunun tipi göstərilir. Bu tiplər aşağıdakılardan biri ola bilər:

- `cmdUnknown`–məlum olmayan tipli əmr;

- cmdText–mətn tipli əmr və ya saxlanılan prosedur;
- cmdTable– cədvəlin adı;
- cmdStoredProc– saxlanılan proseduran adı;
- cmdFile– verilənlər yığımı faylının adı.

Verilənlər bazası ilə əlaqə Connection və ya ConnectionString xassələri vasitəsi ilə yerinə yetirilir.

4.3.1.3. TADOCommand komponenti

Bu komponent yekun verilənlər yığımı qaytarmadan SQL–əmərlərini yerinə yetirmək üçün nəzərdə tutulmuşdur. Yerinə yetirilməsi zəruri olan SQL–əmərləri birbaşa bu komponentin CommandText xassəsinə yazılır. CommandType xassəsində SQL–sorgunun tipi müəyyənləşdirilir. Bu xassənin ala biləcəyi qiymətlər TADODataSet komponentinin CommandType xassəsinin aldığı qiymətlərlə tamamilə eynidir. Bu komponentlə də işlədikdə verilənlər bazası ilə əlaqə Connection və ya ConnectionString xassələri vasitəsi ilə yerinə yetirilir.

4.3.1.4. TADOTable komponenti

TADOTable komponenti ADO vasitəsi ilə verilənlərə müdaxilə etmək və onları cədvəl şəklində təsvir etmək üçündür, BDE səhifəsindəki Table komponentinin birbaşa analoqudur. Verilənlər bazası ilə əlaqə Connection və ya ConnectionString xassələri vasitəsi ilə yaradılır. Connection xassəsində TADOConnection komponenti, ConnectionString xassəsində isə verilənlər mənbəyinin ünvanı göstərilir. Əgər əlaqə TADOConnection komponenti ilə yaradılmışdırsa, onda TADOTable komponentinin Active xassəsinə True qiyməti verdikdə də baza ilə əlaqə yaradılır. Bazanın konkret cədvəlini seçmək üçün TableName xassəsi nəzərdə tutulmuşdur. Əgər cədvəli yalnız oxumaq üçün açmaq

lazımdırsa, onda `ReadOnly` xassəsinə `True` qiyməti vermək lazımdır, bu halda verilənləri dəyişmək mümkün olmayacaqdır. Əsas və tabe olan cədvəllər arasında istinad tamlığı münasibəti `MasterSource` xassəsi ilə təyin olunur. Bu xassənin qiyməti kimi əsas cədvəllə əlaqə yaradan `DataSource` komponenti göstərilir. `TableDirect` xassəsi ilə cədvələ müraciət üsulu müəyyənləşdirilir. Əgər bu xassəyə `True` qiyməti mənimsədilsə, onda adı vasitəsi ilə cədvələ birbaşa müraciət olunur. Bəzi VB-lər birbaşa müraciət üsulu ilə işləməyə imkan vermədiyi üçün, adi halda bu xassəyə `False` qiyməti verilmişdir. Keş yaddaşın həcmi `CacheSize` xassəsi ilə təyin olunur. Əgər bu xassəyə 50 qiyməti versək, onda cədvələ birinci dəfə qoşulduqda komponent birinci 50 sətiri seçir və onları lokal yaddaşa yükləyir ki, bu da verilənlərə müraciəti sürətləndirir. Yerdə qalan sətirlər zərurət yarandıqca serverdən yüklənəcəkdir. `CommandTimeout` xassəsində əmrin icrasını gözləmə vaxtı göstərilir. Komponent VB-yə əmr göndərdikdə gözləmə taymerini qoşur, əmr icra olunmadıqda və vaxt başa çatdıqda səhv haqqında məlumat verilir.

4.3.1.5. TADOQuery komponenti

`TADOCommand` komponentindən fərqli olaraq, `TADOQuery` komponenti VB-nin bir və ya bir neçə cədvəllərindən verilənlər yığımını almaq üçün tətbiq edilir. Faktiki olaraq o, `TQuery` komponentinin bütün funksiyalarını təkrarlayır. Verilənlər bazasına sorğu əmrləri çoxsətirli SQL xassəsində yazılır. Sorğuda `CREATE TABLE` operatorunu istifadə etmək olar. Verilənlər yığımı ilə birbaşa və ya `TADOConnection` komponenti vasitəsi ilə əlaqə yarada bilər. Sorğu parametrləri `Parameters` xassəsində yerləşir. Əgər komponent verilənlər yığımını qaytararsa, onda onu ya `Open` metodu ilə, ya da `Active` xassəsinə `True` qiyməti verməklə açmaq lazımdır. `TQuery` komponentində olduğu kimi, `TADOQuery` komponentinin də `DataSource` xassəsi mövcuddur ki, onun vasitəsi ilə sorğu parametrləri bir komponentdən digərinə ötürülə bilər.

4.3.1.6. TADOStoredProc komponenti

Bu komponent serverdə saxlanılan proseduru yerinə yetirmək etmək üçün istifadə edilir. Verilənlər yığımı ilə birbaşa və ya TADOConnection komponenti vasitəsi ilə əlaqə yarada bilər. BDE və InterBase-dən fərqli olaraq ADO-da prosedurlar verilənlər yığımını qaytara bilər. Saxlanılan prosedurun adı ProcedureName xassəsi ilə müəyyən olunur. Prosedurun adını artıq mövcud olan siyahıdan da seçmək olar. Prosedurun giriş-çıxış parametrlərini müəyyən etmək üçün Parameters xassəsindən istifadə olunur. Parametrlər vasitəsi ilə prosedur özünün arqumentlərini alır və işinin nəticəsini qaytarır. TADOStoredProc komponenti özünün funksional imkanlarına görə BDE yönümlü TStoredProc komponenti ilə tamamilə eynidir, lakin ondan daha çox imkanlara malik olması ilə fərqlənir.

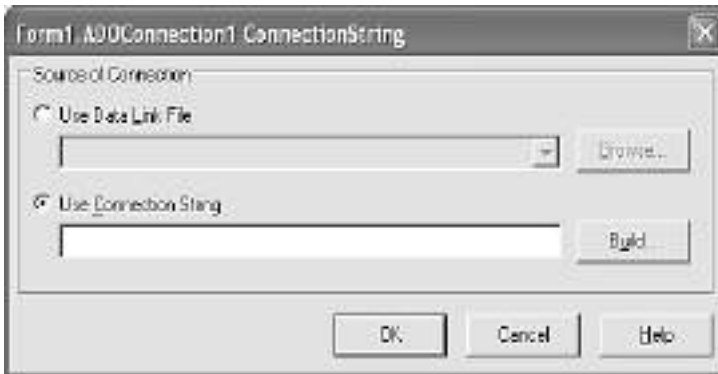
4.3.1.7. TRDSConnection komponenti

ADO-nun sonuncu komponenti olan bu komponent çoxmənqəli arxitekturanı (bir və ya bir neçə əlavələrdən ibarət) dəstəkləmək üçün yaradılmışdır. Bu komponent kliyent proqramı ilə əlavə serveri arasında əlaqə yaradır. O, yazılar yığımını bir kompüterdən digərinə köçürdükdə verilənlər marşallinqini idarə edir. *Verilənlər marşallinqi* kliyentə digər ünvan fəzasında və ya digər kompüterdə yerləşən obyektlərə müraciət etməyə imkan verən mexanizmdir. TRDSConnection komponentinin əsas xassəsi ServerName xassəsidir. Onun vasitəsi ilə lazım olan əlavə serverinə qoşulmaq mümkün olur.

4.4. Delphi ilə Ms Access cədvəlləri arasında əlaqə

Access cədvəlləri ilə əlaqə yaratmaq üçün Delphi-ni yükləyib, forma üzərinə Komponentlər palitrasının ADO səhifəsindən ADOConnection və ADOTable komponentləri yerləşdirin. ADOConnection komponenti ADO səhifəsində

yerləşən digər komponentlərlə VB arasında əlaqə yaradır. Belə əlaqə onun `ConnectionString` xassəsi vasitəsi ilə yaradılır. Qeyd edək ki, `ADOTable` komponentinin də `ConnectionString` xassəsi mövcuddur və Access cədvəlləri ilə bu komponent vasitəsi ilə də əlaqə yaratmaq olar, lakin formada bir neçə `ADOTable` komponenti olduqda onların hər biri üçün bu əlaqə yaradılmalıdır. Ona görə də `ADOConnection` komponenti vasitəsi ilə bir dəfə əlaqə yaradıb onu digər komponentlər üçün istifadə etmək daha rahatdır. Obyektlər inspektorunda `ConnectionString` xassəsi qarşısında mausun düyməsini iki dəfə basın. Komponentin ADO–ya qoşulma pəncərəsi açılacaqdır (şəkil 4.2).

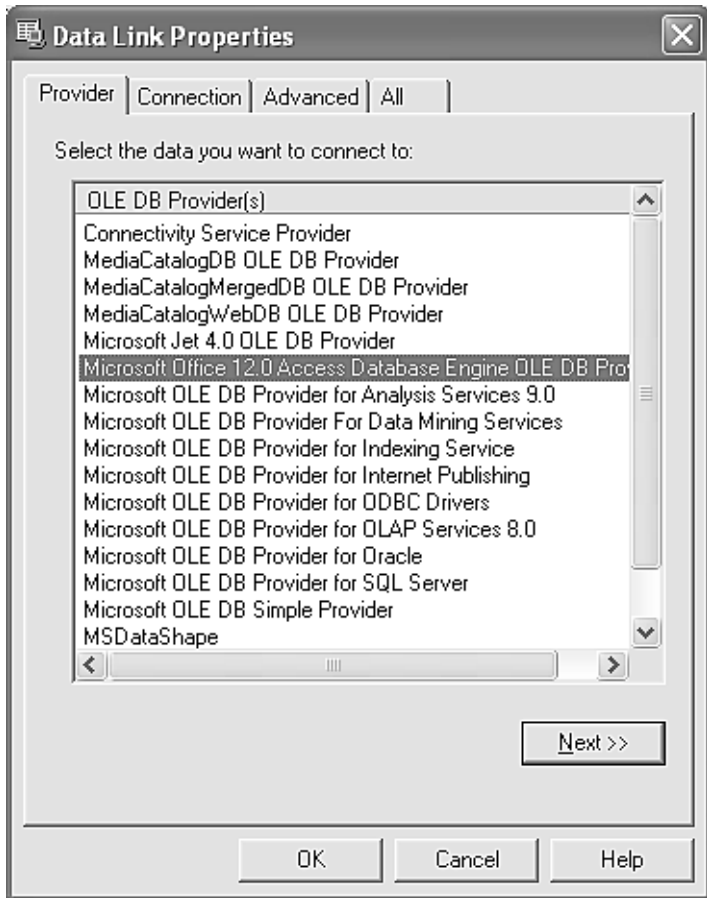


Şəkil 4.2. Komponentin ADO–ya qoşulma pəncərəsi

Bu pəncərədə biz Access cədvəli ilə üç üsulla əlaqə yarada bilirik:

- əvvəlcədən yaradılmış link–faylını istifadə etməklə;
- `Use Connection String` sətrinə faylın yerləşdiyi ünvanı yazmaqla;
- `Build...` düyməsindən istifadə edərək ünvanı sətər yazmaqla.

Build... düyməsini basmaqla əlaqə yaradaq. Şəkil 4.3–də göstərilən qoşulmanın sazlanması pəncərəsi açılacaqdır.



Şəkil 4.3. Qoşulmanın sazlanması pəncərəsi (Provider səhifəsi)

Bu pəncərədə biz *OLE DB*–provayderini seçməliyik. Ms Access cədvəlləri ilə işləmək üçün daha çox **Microsoft Jet 4.0 OLE DB Provider** uyğun gəlir. **JET**– Ms Access–ə quraşdırılmış VBİS–lərlə iş mexanizminin adıdır (əslində *JET* mexanizmi digər

çoxlu sayda lokal verilənlər mənbələri ilə də işləməyə imkan verir). *JET* adı həmişə Access–lə əlaqələndirilir, həqiqətən də Access *JET*–lə qarşılıqlı əlaqədə olan əsas sistemdir. Hazırda üzərində dayandığımız **Microsoft Jet 4.0 OLE DB Provider** yalnız **.mdb** formatlı Access cədvəlləri ilə əlaqə yarada bilir. Əgər VB cədvəlləri **Access 2007** və daha yüksək versiyalarda yaradılmışdırsa, onda **Microsoft Office 4.0Access Database Engine OLE DB Provider** seçilməlidir. Yeri gəlmişkən digər VBİS–lərlə yaradılmış verilənlər bazaları ilə işləmək üçün bəzi provayderləri sadalayaq:

- *dBase* cədvəlləri ilə işlədikdə – **Microsoft OLE DB Provider for ODBC Drivers**;

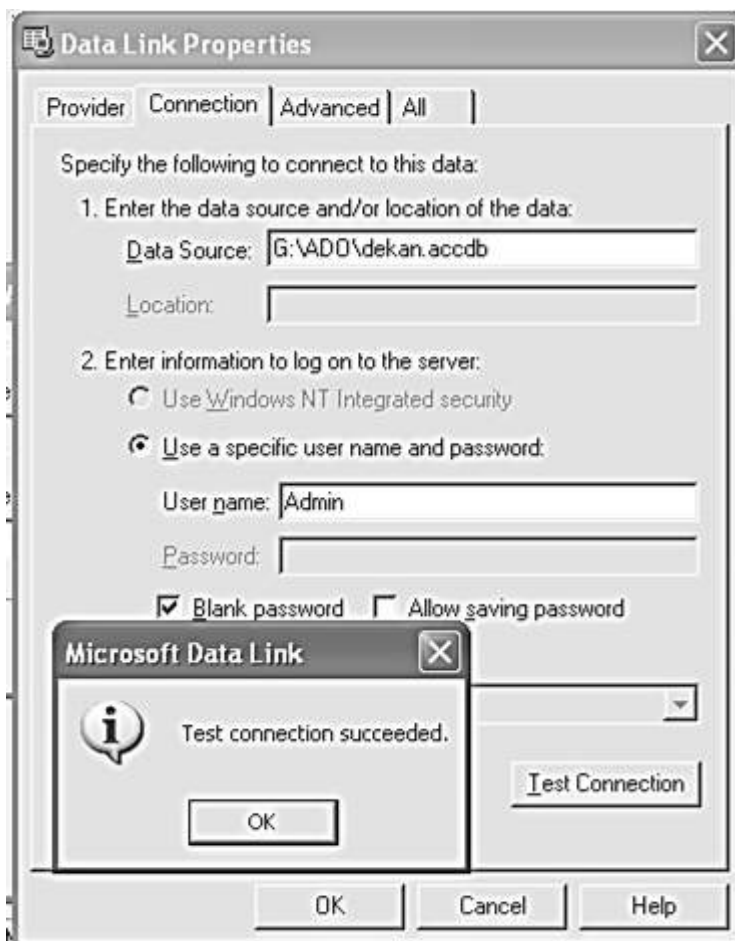
- *Oracle* serveri ilə işlədikdə – **Microsoft OLE DB Provider Oracle**;

- *Microsoft SQL Server* serveri ilə işlədikdə – **Microsoft OLE DB Provider SQL Server**.

Bizim bazamız **Access 2010**–da yaradıldığı üçün biz **Microsoft Office 12.0Access Database Engine OLE DB Provider** provayderi seçirik (şəkil 4.3). Qoşulmanı davam etdirmək üçün ya *Next* düyməsini basırıq, ya da pəncərənin *Connection* səhifəsinə keçirik (şəkil 4.4). *DataSource* sahəsinə Access 2010–da yaratdığımız bazanın ünvanını və adını yazırıq. Bazanın ünvanı sətirdə qeyd olunduqda *Test Connection* düyməsini basaraq baza ilə əlaqənin yaranıb–yaranmamasını yoxlayırıq, əgər əlaqə yaranmışdırsa (*Test connection succeeded*), *OK* düyməsini basmaqla qoşulma əməliyyatını başa çatdırırıq. Obyektlər inspektorunda *ADOConnection* komponentinin *LoginPrompt* xassəsinə *False* qiyməti seçirik, əks halda hər dəfə bazaya müraciət olunduqda Delphi bizdən parolun daxil edilməsini tələb edəcəkdir. Biz isə bazaya parol verməmişik.

ADOTable komponentini seçib, onun *Connection* xassəsinə *ADOConnection1* qiyməti təyin edirik, *TableName* xassəsinə isə VB–nin hər hansı bir cədvəlinin adını, məsələn, *KAFEDRA* seçirik. Cədvəli forma üzərində əks etdirmək üçün

Data Access səhifəsindən DataSource komponentini, DataControls səhifəsindən isə DBGrid cədvəl torunu forma üzərində yerləşdiririk.



Şəkil 4.4. Qoşulmanın sazlanması pəncərəsi (Connection səhifəsi)

Sonra, BDE ilə işlədiyimiz əməliyyatları təkrar edirik, yəni, DataSource1 komponentinin DataSet xassəsinə – ADOTable1, DBGrid komponentinin DataSource xassəsinə– DataSource1 qiymətləri verib, ADOTable1 cədvəlini aktivləşdiririk (Active=True). KAFEDRA cədvəli forma üzərində təsvir olunacaqdır. Bundan sonra isə BDE-də Paradox cədvəlləri üzərində aparıla biləcək bütün əməliyyatları, analoji qaydalarla, burada da aparmaq olar. Lakin SQL Builder sorğu qurucusu ilə iş müstəsnaqlıq təşkil edir.

4.5. Access cədvəlləri üzərində filtrləmə əməliyyatı və sorğuların yaradılması

Access cədvəlləri üzərində işimizi davam etdirək. Bu bölmədə biz cədvəldə filtrləmə əməliyyatını və sorğuların yaradılmasını öyrənəcəyik.

Forma üzərinə ADOConnection, ADOQuery, DBGrid, DataSource və Button komponentləri yerləşdirib, yuxarıda öyrəndiyimiz qayda ilə dekan.accdb faylı ilə əlaqə yaradaq. Komponentlər arasında əlaqə yaradaq: ADOQuery1-in Connection xassəsinə ADOConnection1, DataSource1-in DataSet xassəsinə ADOQuery1 və DBGrid1-in DataSource xassəsinə DataSource1 qiymətləri təyin edək. Button1 düyməsini basdıqda təyin edəcəyimiz filtrə uyğun cədvəl verilənləri DBGrid torunda təsvir olunacaqdır. Bildiyimiz kimi, ADOQuery komponentinin TableName xassəsi yoxdur, ona görə də hansı cədvəldə filtrləmə aparacağımızı ADOQuery komponentinə bildirməliyik. Bunun üçün ADOQuery komponentinin SQL xassəsi qarşısındakı üç nöqtə təsvirli düyməni basıb, açılan redaktorda

```
SELECT *FROM MUELLIM
```

SQL kodunu yazıb (MUELLIM cədvəlini seçirik) aşağıdakı proseduru tərtib edirik:

```
procedure TForm1.Button1Click(
```

```
Sender: TObject);  
begin  
    ADOQuery1.Filtered:=true;  
    ADOQuery1.Filter:='ER=' 'professor' ' ' ;  
end;
```

Bu filtrləmə nəticəsində cədvəldə yalnız professorların siyahısı əks olunacaqdır. Əslində filtr birqat dırnaq işarələri daxilində yazılmalıdır. Lakin, bilirik ki, Delphi-də birqat dırnaq işarələri daxilində sətir ifadələri yazılır, ona görə də burada üçqat dırnaq işarələri yazmışıq (iki dırnaq işarəsi filtrin qiyməti üçün, biri isə sətirin sonu üçün). Bu dırnaq işarələrinə vaxt itirməmək üçün QuotedStr funksiyasından istifadə etmək olar. Belə ki, bu funksiyaya sətir ötürülür, əvəzində o, həmin sətiri bizə dırnaq işarəsi daxilində qaytarır. Onda həmin sətiri biz aşağıdakı kodla əvəz edə bilərik:

```
ADOQuery1.Filter:='ER='+  
    QuotedStr('professor');
```

Qeyd edək ki, bu filtrləmə əməliyyatını SQL kodlarına və ADOQuery komponentinə müraciət etmədən, ADOTable komponenti ilə daha asanlıqla yarada bilərik. Belə ki, Access faylı ilə əlaqə yaradıldıqdan sonra ADOTable komponentinin TableName xassəsində MUELLIM cədvəlini seçib, yaratdığımız prosedurun sətirlərini aşağıdakı kodlarla əvəz etmək lazımdır:

```
ADOTable1.Filtered:=true;  
ADOTable1.Filter:='ER='+  
    QuotedStr('professor');
```

Biz cədvəli hər hansı bir sahə üzrə nizamlaya da bilərik. Məsələn, MUELLIM cədvəlində yazıları müəllimlərin soyadına görə nizamlamaq üçün prosedur yaradıb oraya aşağıdakı kodu yazmaq lazımdır:

```
ADOTable1.IndexFieldNames:='SAA' ;
```

İndi isə sorğu yaradaq. Forma üzərinə ADOConnection, ADOQuery, DBGrid və DataSource komponentləri yerləşdirib onlar arasında əlaqə yaradaraq ADOQuery komponentinin SQL xassəsinə

```
SELECT MUELLIM.SAA, MUELLIM.ER  
FROM MUELLIM  
WHERE ER='professor'
```

kodu yazıb, onun Active xassəsinə True qiyməti versək, DBGrid torunda yalnız iki sütun – *müəllimlərin soyadı və elmi rütbələri* (yalnız professorlar) təsvir olunacaqdır.

4.5.1. SQL Builder ilə sorğuların yaradılması

ADO səhifəsindəki ADOQuery komponenti SQL Builder ilə ümumiyyətlə işləmir. Əgər biz BDE səhifəsindən Query1 komponentini forma üzərində yerləşdirsək görürük ki, kontekst menyu vasitəsi ilə SQL Builder sorğu qurucusu pəncərəsi açılır, lakin baza ilə əlaqə üçün verilənlər mənbəyinin adı tələb olunur. Delphi əlavəsinin verilənlər mənbəyinə müraciət edə bilməsi üçün müvafiq *verilənlər mənbəyinin adını* – DSN (Data Source Name) göstərmək lazımdır. İstifadəçi verilənlər mənbəyinin üç tipi vardır:

- **User DSN (istifadəçi).** Bu tip drayver yalnız onu təyin edən istifadəçi tərəfindən istifadə görünür. Uyğun olaraq, əlavə indiki anda sistemdə qeydiyyatdan keçən və məhz bu drayveri sazlayan istifadəçi üçün işləyəcəkdir;

- **System DSN (sistem).** Bu tip drayver bu kompüterdə işləyən bütün əlavələr tərəfindən görünür;

- **File DSN (fayl).** Bu tip drayver, hansı istifadəçilərdə ki belə drayver quraşdırılmışdır və onların müvafiq müdaxilə hüququ vardır, onlar üçün ümumi resurs (birgə istifadə) edilə bilər.

İndi Access cədvəlləri ilə SQL Builder-də sorğular yaratmaq qaydalarını öyrənək.

Qarşımıza qoyduğumuz məsələni həll etmək üçün verilənlər mənbəyini yaratmaq lazımdır. Bu mənbə ODBC Administrator utiliti vasitəsi ilə yaradılır. Bu utilit iki üsulla çağrıla bilər:

- *Delphi/Database/Explore* əmərlər ardıcılığı icra edildikdən sonra açılan SQL Explorer pəncərəsindən Object/ODBC Administrator... əmərlər ardıcılığını icra etmək;

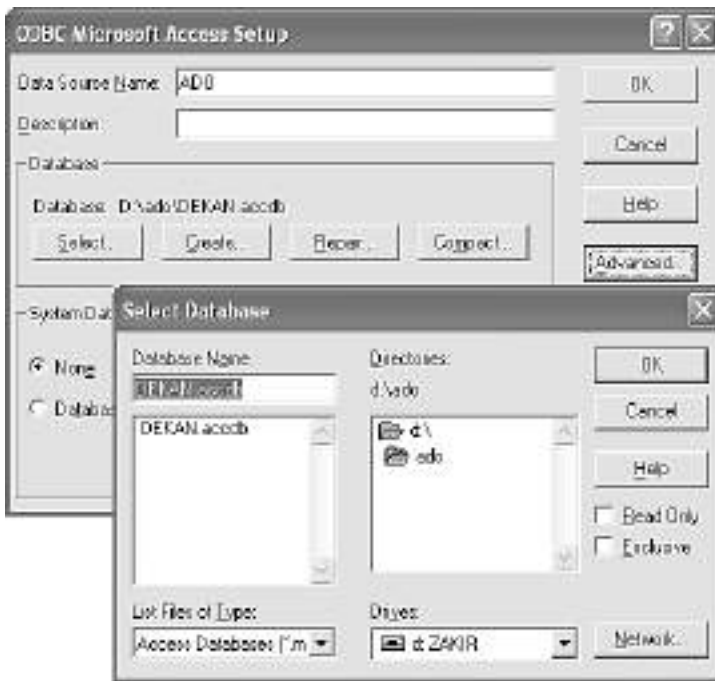
- *Start/Control Panel/Administrative Tools/Data Source (ODBC) (Пуск/Панель управления/Администрирование / Источники данных (ODBC))* əmərlər ardıcılığını icra etmək.

Hər iki halda ODBC Data Sources Administrator (Администратор источников данных ODBC) pəncərəsi açılacaqdır (şəkil 4.5).



Şəkil 4.5. ODBC DataSource Administrator pəncərəsində Access drayverinin seçilməsi

Bu pəncərədə Add... (Добавить...) düyməsini basıb, açılan pəncərənin Select a driver for which you want to set up a data source (Выберите драйвер, для которого задается источник) sahəsindən **Microsoft Access Driver(*.mdb, *.accdb)** seçib Finish (Готово) düyməsini basın. Yeni ODBC Microsoft Access Setup pəncərəsi açılacaqdır (şəkil 4.6).



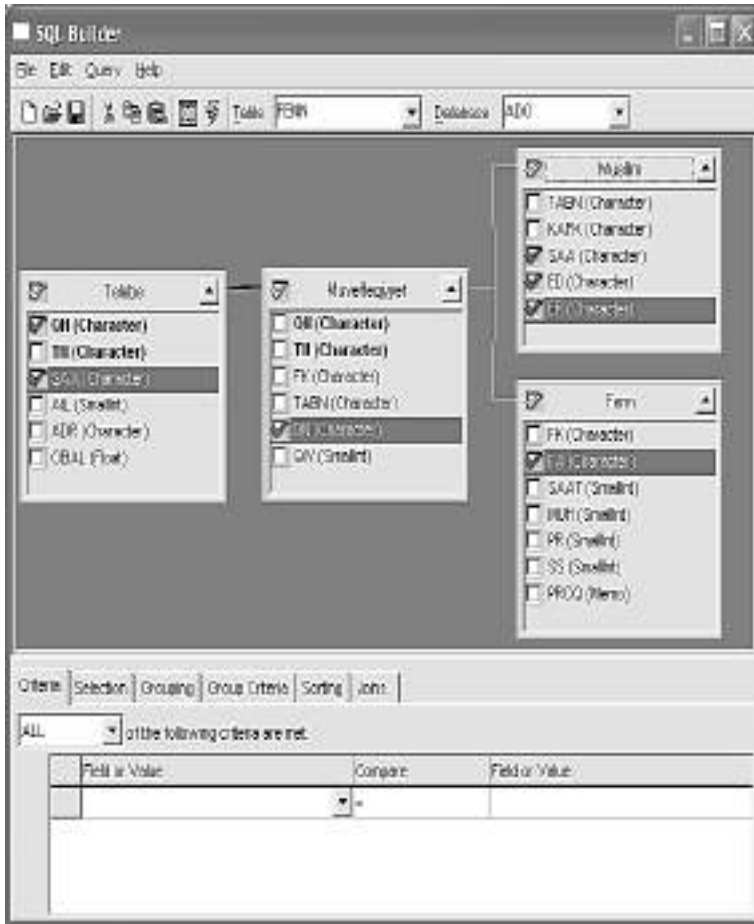
Şəkil 4.6. Verilənlər mənbəyinə adın verilməsi və Access faylının seçilməsi

Data Source Name (Имя источника данных) sahəsinə yaradacağınız verilənlər mənbəyinin adını daxil edin, məsələn, ADO. Access-də yaratdığımız baza ilə əlaqə yaratmaq


üçün Select... düyməsini basıb açılan pəncərədə faylın yerləşdiyi diski, qovluğu və faylı seçib OK düyməsini basmaq lazımdır. Create... düyməsini basaraq məxsusi verilənlər bazası yaratmaq olar. Repair... düyməsi zədələnmiş faylı bərpa etməyə imkan verir. Faylı “zibildən” təmizləmək üçün Compact... düyməsindən istifadə edilir. Page Timeout: sahəsində (Options düyməsini basanda açılır) vaxt intervalı göstərilir. Bu vaxt ərzində istifadə olunmayan verilənlər səhifəsi buferdə saxlanacaqdır. Buffer Size: xassəsində buferin ölçüsü göstərilir ki, drayver bu həcmdə verilənləri əlavədən diskə və əksinə ötürür. Buferin ölçüsü 256–a bölünən ədəd olmalıdır. Set Advanced Options pəncərəsində (Advanced düyməsini basmaqla açılır) login, şifrə daxil etmək və digər sazlamalar aparmaq olar. Bütün bu parametrləri müəyyənləşdirdikdən sonra geriye qayıdış baş verəcək və Siz Access faylının adını pəncərədə görəcəksiniz (şəkil 4.6). Bu pəncərədə OK düyməsini basdıqda şəkil 4.5–də göstərilən pəncərədə verilənlər mənbəyinin adı əlavə olunacaqdır. OK düyməsini basıb, pəncərəni bağlayın. Bununla da verilənlər mənbəyinin yaradılma prosesi başa çatdı.

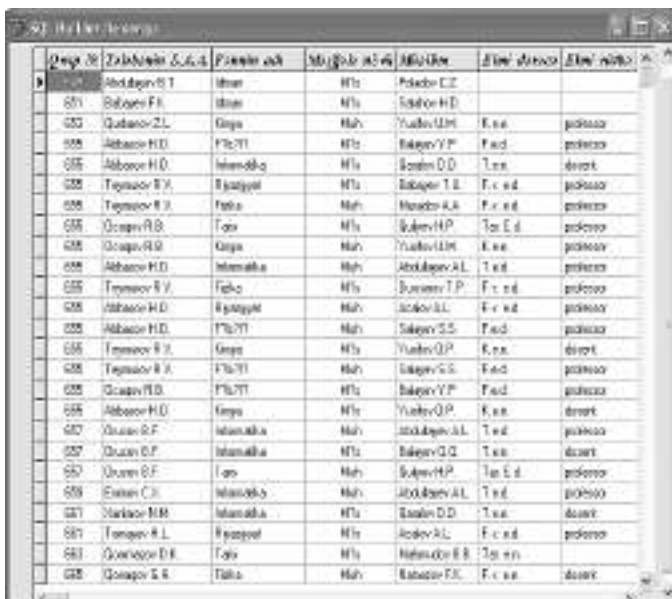
İndi isə Delphi–ni yükləyib forma üzərinə BDE səhifəsindən Query komponenti yerləşdirib, onun Database xassəsinə yaratdığımız mənbənin adını – ADO seçin. Sorğunun nəticəsini görmək üçün formaya əlavə olaraq DataSource və DBGrid komponentləri də yerləşdirək. DataSource1 komponentinin DataSet xassəsinə Query1, DBGrid1 komponentinin DataSource xassəsinə isə DataSource1 qiyməti təyin edək. Query1 komponenti üzərində mausun sağ düyməsini basaraq kontekst menyudan SQL Builder... əmrini icra edin. Delphi istifadəçi adının və şifrənin daxil edilməsini tələb edən dialog pəncərəsi açacaqdır; Siz heç nə daxil etmədən OK düyməsini basın (biz istifadəçi adı və şifrə verməmişik), SQL Builder pəncərəsi açılacaqdır. Bu pəncərənin Database siyahısından ADO seçdikdən sonra Table siyahısında yaratdığımız bütün Access cədvəlləri görünəcəkdir (şəkil 4.7). Bu siyahıdan hansı cədvəli seçsəniz, pəncərədə onun strukturu təsvir olunacaqdır. Nümunə üçün bir sorğu yaradaq. Table siyahısından TELEBE,

MUVEFFEQIYYET, MUELLIM və FENN cədvəllərini seçək və bu cədvəllər arasında əlaqə yaradaq. Əlaqə yaratmaq üçün sadəcə olaraq sahəni əlaqə yaradacağımız cədvəlinin sahəsinə dartmaq lazımdır.



Şəkil 4.7. Cədvəllər arasında əlaqələrin yaradılması və sorğu üçün sahələrin seçilməsi

Bu əlaqələri belə yaradaq: TELEBE və MUVEFFEQIYYET cədvəlləri QN və TN sahələri üzrə; MUVEFFEQIYYET və MUELLIM cədvəlləri TABN sahəsi üzrə; MUVEFFEQIYYET və FENN cədvəlləri FK sahəsi üzrə. Sorğuda hansı sahələri görmək istəyiriksə, onların qarşısında bayraq işarəsi qoymaq lazımdır. Sahələr hansı ardıcılıqla seçiləcəksə, yekun cədvəldə də həmin ardıcılıqla təsvir olunacaqdır.  düyməsini basdıqda sorğu icra olunacaq və cədvəl şəklində təsvir olunacaqdır. Bu sorğunu DBGrid cədvəlində göstərək. Bunun üçün sorğunu yadda saxlayıb, Query1 komponentinin Active xassəsinə True qiyməti verib, layihəni icra etmək lazımdır (F9 klavişini və ya yaşıl düyməni basmaqla). Sorğunun nəticəsi şəkil 4.8–də göstərilmişdir.



Dərs №	Tədrisin adı	Fənnin adı	Müəllimin adı	Məktəb	Əlavə sahə	Əlavə rəhbər
651	Abdullayev S.T.	İbtisam	Mf	Şadlıq C.Ş.		
652	Abdullayev F.H.	İbtisam	Mf	Şadlıq H.D.		
653	Qurbanov Z.L.	Qəna	Mf	Yusifov L.H.	F. c. ad	professor
654	Abdullayev H.D.	Fizika	Mf	Bakayev Y.P.	F. ad	professor
655	Abdullayev H.D.	İntematika	Mf	Səlim D.D.	T. ad	dozent
656	Təmmərov S.Y.	Şəxsiyyət	Mf	Bakayev T.B.	F. c. ad	professor
657	Təmmərov S.Y.	Fizika	Mf	Məmmədov A.A.	F. c. ad	professor
658	Qurbanov R.B.	Qəna	Mf	Bakayev H.P.	T. c. ad	professor
659	Qurbanov R.B.	Qəna	Mf	Yusifov L.H.	F. c. ad	professor
660	Abdullayev H.D.	İntematika	Mf	Abdullayev A.L.	T. ad	professor
661	Təmmərov S.Y.	Fizika	Mf	Burayev T.P.	F. c. ad	professor
662	Abdullayev H.D.	Şəxsiyyət	Mf	Şadlıq S.L.	F. c. ad	professor
663	Abdullayev H.D.	Fizika	Mf	Bakayev S.S.	F. ad	professor
664	Təmmərov S.Y.	Qəna	Mf	Yusifov Q.P.	F. c. ad	dozent
665	Təmmərov S.Y.	Fizika	Mf	Bakayev S.S.	F. ad	professor
666	Qurbanov R.B.	Fizika	Mf	Bakayev Y.P.	F. ad	professor
667	Abdullayev H.D.	Qəna	Mf	Yusifov Q.P.	F. c. ad	dozent
668	Qurbanov R.F.	İntematika	Mf	Abdullayev A.L.	T. ad	professor
669	Qurbanov R.F.	İntematika	Mf	Bakayev Q.Q.	T. ad	dozent
670	Qurbanov R.F.	Qəna	Mf	Bakayev H.P.	T. c. ad	professor
671	Qurbanov C.L.	İntematika	Mf	Abdullayev A.L.	T. ad	professor
672	Abdullayev M.B.	İntematika	Mf	Səlim D.D.	T. ad	dozent
673	Təmmərov S.L.	Şəxsiyyət	Mf	Şadlıq S.L.	F. c. ad	professor
674	Qurbanov R.F.	Qəna	Mf	Məmmədov B.B.	T. c. ad	dozent
675	Qurbanov S.B.	Fizika	Mf	Abdullayev F.C.	F. c. ad	dozent

Şəkil 4.8. SQL Builder–də yaradılmış sorğunun nəticəsi

Əgər bizi bu sorğuya uyğun SQL kodları maraqlandırarsa, onda SOL Builder pəncərəsinin Query menyusundan Show SQL əmrini icra etmək (və ya F7 klavişini basmaq) lazımdır (şəkil 4.9).



Şəkil 4.9. SQL Builder–də yaradılmış sorğunun SQL kodları

Ümumiyyətlə, ADO texnologiyasında SQL Builder–lə işlərin BDE texnologiyasındakı işlərdən heç bir fərqi olmadığı üçün, hesab edirik ki, əlavə şərtlərə ehtiyac yoxdur və izahatlarımızı burada dayandırırıq.

4.6. Delphi ilə Ms Excel cədvəlləri arasında əlaqə

ADO texnologiyası **Ms Excel** cədvəlləri ilə də işləyə bilər. Ms Excel cədvəli ilə işləmək üçün ya sadə bir Excel cədvəli yaratmaq, ya da mövcud Access cədvəlini Ms Excel –ə ixrac etmək lazımdır. Biz ikinci üsulla belə cədvəl yaradaq. Bunun üçün MS Access VBİS–i yükləyib QURUP cədvəlini seçib *Lent*–in *External Data* (Внешние данные) səhifəsinin *Export* (Экспорт)

qrupundakı Excel əmrini icra etməklə **qrup.xlsx** faylını yaradaq. Biz bu faylı DBGrid cədvəlində təsvir etdirəcəyik.

Forma üzərinə ADOConnection, ADOQuery, DBGrid, DataSource və Button komponentləri yerləşdirək. ADOConnection1 komponentini *qrup.xlsx* faylı ilə əlaqələndirək. Bu məqsədlə onun ConnectionString xassəsi qarşısında **Ms Excel 2007** və **Ms Excel 2010** faylları üçün aşağıdakı sətiri yazaq:

```
Provider =Microsoft.ACE.OLEDB.12.0;  
Data Source = M:\ADO\QRUP.xlsx;  
Extended Properties ="Excel 12.0 Xml;  
HDR=YES";
```

MS Excel 2003 faylları ilə əlaqə yaratdıqda isə bu sətiri belə yazmaq lazımdır:

```
Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source= M:\ADO\QRUP.xls;  
Extended Properties=Excel 8.0;
```

Əgər Siz Excel faylını yaradacağınız Delphi layihəsinin yerləşdiyi qovluqda yerləşdirsəniz, onda fayla yolu göstərmək lazım deyil, yəni yuxarıdakı sətirlərdə

```
Data Source= M:\ADO\QRUP.xlsx;
```

əvəzinə

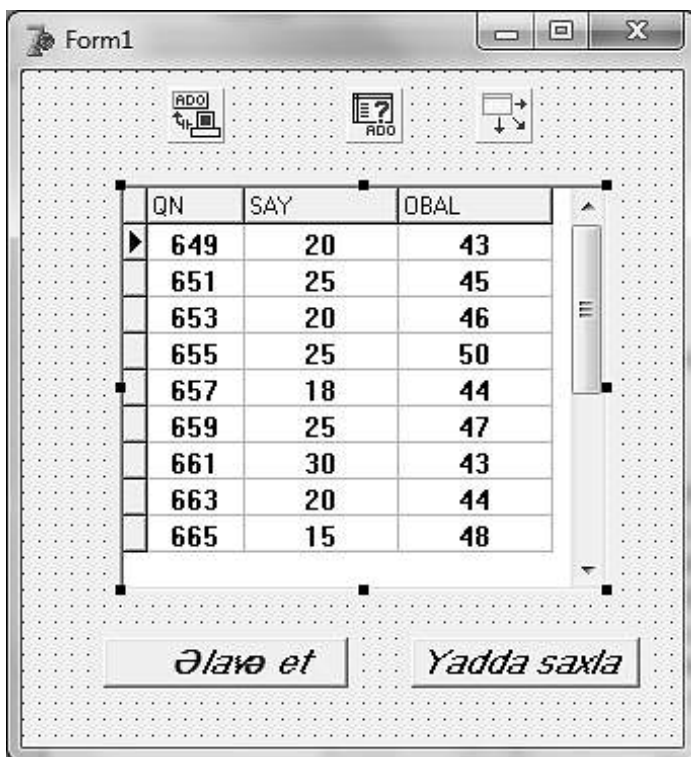
```
Data Source=QRUP.xlsx;
```

yazmaq kifayətdir. Həmişə olduğu kimi, LoginPrompt xassəsinə False qiyməti təyin edin.

Komponentlər arasında yuxarıda öyrəndiyimiz qayda ilə əlaqə yaradaq. ADOConnection1 komponentinin Connected xassəsinə True qiyməti verək. ADOQuery1 komponentinin xassələrini belə müəyyənləşdirək:

```
Connection – ADOConnection1
```

SQL – SELECT *FROM[QRUP\$]; (burada diqqətli olun: bizim misalımızda *grup.xlsx* faylı Excel kitabının *QRUP* adlandırılmış vərəqində yerləşmişdir, əgər Sizin kompüterinizdə bu fayl *Sheet1 (Jucm1)* vərəqində yerləşərsə, müvafiq dəyişiklik edin) və Active – True. DataSource1-in DataSet xassəsinə ADOQuery1 və DBGrid1-in DataSource xassəsinə DataSource1 qiymətləri təyin edək. Əgər bütün əməliyyatları düzgün yerinə yetirmişsinizsə, onda DBGrid-də Excel cədvəli təsvir olunacaqdır (şəkil 4.10).



Şəkil 4.10. DBGrid komponentində Excel cədvəli

Cədvəlin tərtibatını yaxşılaşdırmaq üçün Object TreeView pəncərəsində ADOQuery1/Fields seçib hər bir sahə üçün DisplayWidth (sütunun eni) xassəsinə müvafiq qiymətlər, şrift və s. təyin edin. Bu əməliyyatları DBGrid üzərində mausun düyməsini iki dəfə basaraq açılan Editing DBGrid1.Columns pəncərəsi vasitəsi ilə də icra etmək olar.

İndi isə düymələr üçün funksiyalar müəyyənləşdirək. Button1 düyməsinin sərlovhəsini *Yadda saxla*, Button2 düyməsinin sərlovhəsini isə *Əlavə et* adlandıraraq. Button1 düyməsi üçün aşağıdakı proseduru yazaq:

```
procedure TForm1.TntButton1Click(  
    Sender: TObject);  
begin  
    ADOQuery1.Edit;  
    ADOQuery1.Post;  
end;
```

Button2 düyməsi üçün isə aşağıdakı proseduru yazaq:

```
procedure TForm1.TntButton2Click(  
    Sender: TObject);  
begin  
    ADOQuery1.Append;  
end;
```

Diqqətli olun! Layihəni işə buraxmazdan əvvəl ADOConnection1 komponentinin Connected və ADOQuery1 komponentinin Active xassələrinə False qiyməti verin. Əks halda Delphi səhv haqqında məlumat verəcəkdir ki, “qoşula bilmirəm”, çünki, bu qoşulma artıq istifadə edilir. Bütün bu qaydalara əməl etdikdən sonra layihəni işə buraxa bilərsiniz. Siz *Əlavə et* düyməsini basdıqda yeni sətir əlavə edə biləcəksiniz, *Yadda saxla* düyməsini basdıqda isə cədvəl əlavə olunmuş sətirlərlə birlikdə yadda saxlanacaqdır. Buna tam əmin olmaq üçün həmin cədvəli Ms Excel-də açıb baxa bilərsiniz. Lakin, təəssüflər olsun ki, bu provayder cədvəldən yazıları pozmağa icazə vermir.

4.7. Paradox cədvəllərinin ADO vasitəsi ilə Delphi əlavələrinə qoşulması

Paradox cədvəllərini yalnız *BDE* ilə deyil, ADO vasitəsi ilə də Delphi əlavələrinə qoşmaq mümkündür. Bunun üçün *Start/Control Panel/Administrative Tools/Data Source (ODBC) (Пуск/Панель управления/Администрирование/Источники данных (ODBC))* əmrilər ardıcılığını icra edərək ODBC Data Source Administrator (Администратор источников данных ODBC) pəncərəsini açıb Add (Добавить) düyməsini basdıqdan sonra, **Microsoft Paradox Driver (*.db)** drayverini seçib Finish (Готово) düyməsini basırıq (ADO vasitəsi ilə SQL Builder-lə işləri izah etdikdə bu pəncərələrin şəkillərini göstərdiyimiz üçün onları təkrarlamağa ehtiyac görmürük). Sonra Data Source Name (Имя источника данных) sahəsinə *My PARADOX* yazın, biz bu mənbəni yalnız öz əlavəmiz üçün istifadə edəcəyik. Növbəti addımda Use current directory (Использовать текущий каталог) yazısının qarşısından bayrağı götürüb Select directory (Выбор каталога) düyməsini basıb qovluğu seçin (məsələn, *M:\ADO*), OK düyməsini basıb bütün pəncərələri bağlayın.

İndi isə Delphi-ni yükləyib forma üzərinə ADOConnection, ADOTable, DataSource və DBGrid komponentləri yerləşdirin. ADOConnection1 komponentinin ConnectionString xassəsi qarşısında mausun düyməsini basıb, açılan redaktorda Build.. düyməsini basıb, Provider (Поставщик данных) səhifəsindən Paradox-la işləmək üçün zəruri olan **Microsoft OLE DB Provider for ODBC Drivers** drayverini seçin. Sonra Connection (Соединение) səhifəsinə keçin (bunu Next (Далее) düyməsini basmaqla da etmək olar). Həmin səhifədə User Data Sources: (Использовать имя источника данных:) açılan siyahısından bizim təyin etdiyimiz mənbənin adını – *My PARADOX* seçin. Test Connection (Проверить подключение) düyməsini basın, əgər ekrana Test connection succeeded (Проверка соединения выполнена) məlumatı çıxarsa, deməli, qoşulma

baş tutmuşdur. Bundan sonrakı əməliyyatları isə biz artıq çox yaxşı bilirik: LoginPrompt xassəsinə False, Connection xassəsinə True qiymətləri verib, ADOTable, DataSource və DBGrid komponentləri arasında əlaqə yaradıırıq, ADOTable–nın TableName xassəsinə QRUP seçib Active xassəsinə True qiyməti veririk və bütün bu əməliyyatların nəticəsi olaraq DBGrid–də Paradox cədvəlinin təsvirini görəcəyik.

4.8. ADO və VB haqqında bəzi məlumatların alınması

ADO ilə işlədikdə ADO–nun versiyası, cədvəllər, onların sahələri, cari qoşulmanın vəziyyəti və s. haqqında zəruri məlumatlar almaq olar. ADO–nun versiyası haqqında məlumat almaq üçün sadəcə olaraq formaya ADOConnection komponenti yerləşdirib, onu heç nə ilə əlaqələndirmədən hər hansı bir hadisə emaledicisinin proseduranda, məsələn düymə üçün OnClick proseduranda, ADOConnection1.Version; kodu yazmaq kifayətdir. VB–də olan cədvəllər haqqında məlumat almaq üçün

```
procedure GetTableNames(List: TStrings;  
    SystemTables: Boolean = False);
```

prosedurundan istifadə etmək olar. Bu proseduru tətbiq etmək üçün forma üzərinə ADOConnection komponenti endirib onu dekan.accdb faylı ilə əlaqələndirək. Formaya Standard səhifəsindən ListBox və Button komponentləri yerləşdirib aşağıdakı proseduru yazaq:

```
procedure TForm1.Button1Click(  
    Sender: TObject);  
begin  
    ADOConnection1.GetTableNames(  
        ListBox1.Items, system.True);  
    Form1.Caption:='Versiya ' +  
        ADOConnection1.Version;  
end;
```

Beləliklə, `Button1` düyməsini basdıqda formanın sərlövhəsində ADO-nun versiyası (bizim halda – **6.1**, lakin Sizdə bu versiya fərqli ola bilər), `ListBox1` komponentində isə Access-də yaratdığımız bazanın cədvəllərinin və sistem cədvəllərinin adları təsvir olunacaqdır (şəkil 4.11). Əgər `GetTableNames` prosedurunda vacib olmayan `system.True` parametrini yazmasaq və ya onun əvəzinə `system.False` yazsaq, onda yalnız Access cədvəlləri görünəcəkdir.



Şəkil 4.11. Versiya və cədvəllər haqqında məlumat

Beşinci fəsil

MİCROSOFT SQL SERVER-də VERİLƏNLƏR BAZASININ YARADILMASI

Bizim məqsədimiz mövcud verilənlər bazasına Visual Studio mühitindən müdaxilə etmək və layihələr yaratmaqdır. Biz bu mühitdə Microsoft SQL Server-də yaradılmış verilənlər bazası cədvəllərinə müdaxilə edəcəyik. Ona görə də əvvəlcə, bütün təfərrüatlarına varmadan, Microsoft SQL Server-də VB-nin yaradılması qaydasını öyrənək. Biz bunu pulsuz yayılan *Microsoft SQL Server 2005* versiyasında yerinə yetirəcəyik. Bu versiyanı www.microsoft.com/express/ ünvanından yükləyə bilərsiniz.

5.1. Serverə qoşulma

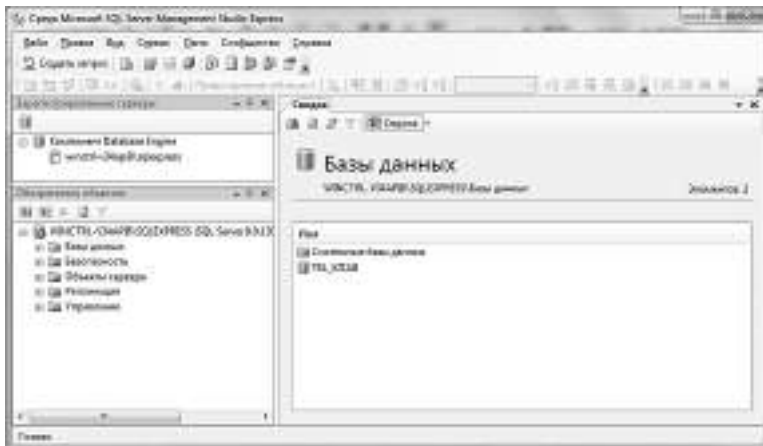
Beləliklə, *Microsoft SQL Server*-in tərkibinə daxil olan *Среды Microsoft SQL Server Management Studio Express* proqramını işə salın (şəkil 5.1).

Bu pəncərədə Проверка подлинности parametri serverə qoşulma zamanı autentifikasiyanı bildirir. Əgər Проверка подлинности Windows (Windows authentication) seçilərsə (bizim halda bu belədir), onda istifadəçi adı və şifr kimi sistem parametrləri istifadə edilir.

Bu pəncərənin Соединение с сервером dialog pəncərəsində serverin adını seçib Соединить düyməsini basın. Əgər kompüterinizdə server yoxdursa (bizdə olduğu kimi) onda həmin sahəyə kompüterinizin adını daxil edin. Bundan sonra açılacaq pəncərədə verilənlər bazasından ibarət Databases (База данных) qovluğunu açın (şəkil 5.2).



Şəkil 5.1. *Среда Microsoft SQL Server Management Studio Express* pəncərəsi



Şəkil 5.2. База данных qovluğu açıldıqdan sonra *Microsoft SQL Server Management Studio Express* proqramının görünüşü

5.2.1. Verilənlərin tipləri

Verilənlər bazasında verilənlər cədvəllərdə saxlanılır və hər bir verilənin konkret tipi təyin edilməlidir. Bu bölmədə Microsoft SQL Server-də sistem tərəfindən dəstəklənən verilənlərin tiplərini təsvir edəcəyik.

Cədvəli yaratmadan əvvəl, orada saxlanılacaq verilənlərin tiplərini müəyyən etmək lazımdır. Verilənlərin tipi informasiyanın tipini (simvol, ədəd və ya tarix), bu verilənlərin saxlanması üsulunu müəyyən edir. *Microsoft SQL Server 2005*-in təqdim etdiyi tiplər cədvəl 5.1-də göstərilmişdir.

Cədvəl 5.1. *Microsoft SQL Server*-də verilənlərin tipləri

Kateqoriya		Verilənlərin tipləri
Rəqəmsal	Tam ədəd	int, bigint, smallint, tinyint
	Dəqiq	decimal numeric
	Təqribi	float, real
	Pul	money, smallmoney
Tarix və vaxt		datetime, smalldatetime
Simvol	Unicode simvollarını dəstəkləməyən	char, varchar, varchar(max), text
	Unicode simvollarını dəstəkləyən	nchar, nvarchar, nvarchar(max), ntext
İkilik		binary, varbinary, varbinary(max)
Təsvir		image
Global identifikator		uniqueidentifier
XML		xml
Xüsusi		bit, cursor, timestamp, sysname, table, sql_variant

5.2.2. Cədvəllərin yaradılması

Cədvəli yaratdıqda əvvəlcədən yaradılacaq cədvəlin adını, sütunların adlarını və sütun verilənlərinin tiplərini bilmək lazımdır.

Konkret cədvəl üçün sütunların adları unikal olmalıdır, lakin, eyni bir bazanın müxtəlif cədvəllərində eyniadlı istənilən qədər sütunlar ola bilər. Hər sütun üçün verilənlərin tipi təyin edilməlidir. VB ilə işlədikdə aşağıdakılara yol verilir:

- verilənlər bazası, cədvəllər də daxil olmaqla, 2 milyarddan çox elementə malik ola bilər;

- bir cədvəldə 1024-ə qədər sütun ola bilər;

- sətir 8060 bayta qədər informasiyadan ibarət ola bilər

TEL_KITAB adlandırdığımız verilənlər bazasının qovluğunu şəkil 5.4-də göstərildiyi kimi açın.

Biz telefon məlumat kitabçası rolunu oynayan VB yaradacağıq. O, iki cədvəldən ibarət olacaqdır. Birinci – Kontaktlar cədvəli id, ad və soyad kimi sütunlardan ibarət olacaqdır.

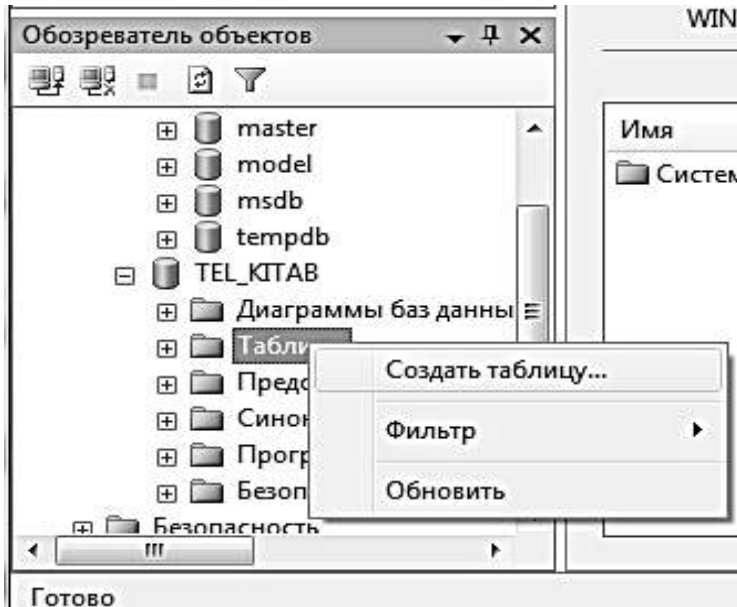
id sütunu cədvəlin sətirlərinin tanınması üçün unikal tam ədədlərdən ibarət olacaqdır. Bu sütun açar sahəsi kimi nəzərdə tutulmuşdur. ad və soyad sütunlarında isə insanların adı və soyadı yazılacaqdır.

Kontaktlar cədvəlini yaratmaq üçün TEL_KITAB qovluğuna daxil olan Таблицы qovluğu üzərində mausun sağ düyməsini basıb kontekst menyudan New Table... (Создать таблицу...) əmrini icra etmək lazımdır (şəkil 5.4).

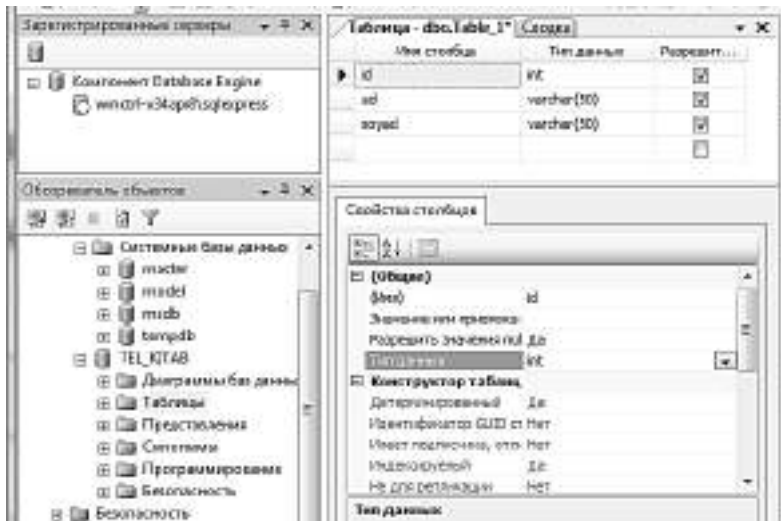
Bu əmr icra edildikdən sonra cədvəlin sahələrinin yaradılması redaktoru açılacaqdır (şəkil 5.5).

Cədvəl redaktorunun birinci sütununa Column Name (Имя столбца) sahənin adı yazılır, ikinci sütununa Data Type (Тип данных) sahənin tipi seçilir. id sahəsi üçün int, qalan sahələr üçün isə dəyişən uzunluqlu varchar mətn tipi seçirik. Üçüncü, Allow Nulls (Разрешить значения null) sahəsində bütün sahələr üçün bayraqcıq nişanını götürmək lazımdır, çünki, bizim bazada boş sütun olmamalıdır.

Növbəti addımda açar sahəsi müəyyən edilməlidir. Bizim misalda bu id sahəsidir. Onu açar sahəsi etmək üçün onun üzərində kontekst menyunun Set Primary Key (Задать первичный ключ) əmrini icra etmək lazımdır.



Şəkil 5.4. Cədvəlin yaradılması



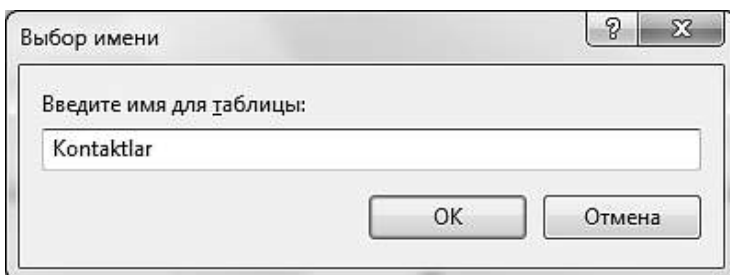
Şəkil 5.5. Cədvəlin sahələrinin yaradılması

Bundan sonra sahənin adının əvvəlində açar nişanı əmələ gələcəkdir (şəkil 5.6).



Şəkil 5.6. İlk açar sahəsinin müəyyən edilməsi

Sonuncu mərhələdə cədvəlin strukturunu yadda saxlamaq lazımdır. Bunun üçün *Cpəda Microsoft SQL Server Management Studio Express* pəncərəsinin alətlər panelində yerləşən disket nişanının üzərində mausun düyməsini basmaq lazımdır. Bu zaman açılan dialoq pəncərəsində cədvəl ad verib Ok düyməsini basmaq (şəkil 5.7) və cədvəlin strukturunu yaradan redaktoru bağlamaq lazımdır.



Şəkil 5.7. Cədvəl adın verilməsi

Biz Kontaktlar cədvəlini əyani olaraq yaratdıq. Bu zaman avtomatik olaraq aşağıdakı SQL əmrləri tərtib edilmişdir (əslində bu kodları biz özümüz tərtib etməklə cədvəli yarada bilərdik):

```
USE [TEL_KITAB]
GO
/***** Объект: Table
[dbo].[Kontaktlar]      Дата
сценария: 04/26/2019 14:14:13 *****/

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[Kontaktlar](
    [id] [int] NOT NULL,
    [ad] [varchar](50) COLLATE
Azeri_Latin_90_CI_AS NOT NULL,
    [soyad] [varchar](50) COLLATE
Azeri_Latin_90_CI_AS NOT NULL,
    CONSTRAINT [PK_Kontaktlar] PRIMARY KEY
CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF,
IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
SET ANSI_PADDING OFF
```

İndi isə analogi qayda ilə Telefonlar cədvəlini yaradaq. Bu cədvəl də id, idKontakt və Telefon sahələrindən ibarət olacaqdır. Burada da id sahəsi ilkin açar sahəsi olacaq, Telefon sahəsi isə mətn tipli olmaqla, telefon nömrələrini özündə saxlayacaqdır.

idKontakt sahəsi isə Kontaktlar cədvəli üçün xarici açar sahəsidir. Bu sahədə Kontaktlar cədvəlinin açar sahəsindəki qiymətlər yerləşəcək, çünki, elə ola bilər ki, eyni bir kontakt bir neçə telefona malik olsun. Onda bu sahədə eyni qiymətli bir neçə sətir yazılacaqdır. Ona görə də bu sahənin verilənlərinin tipi int tipli olmalıdır. Şəkil 5.8-də bu cədvəlin strukturu göstərilmişdir.



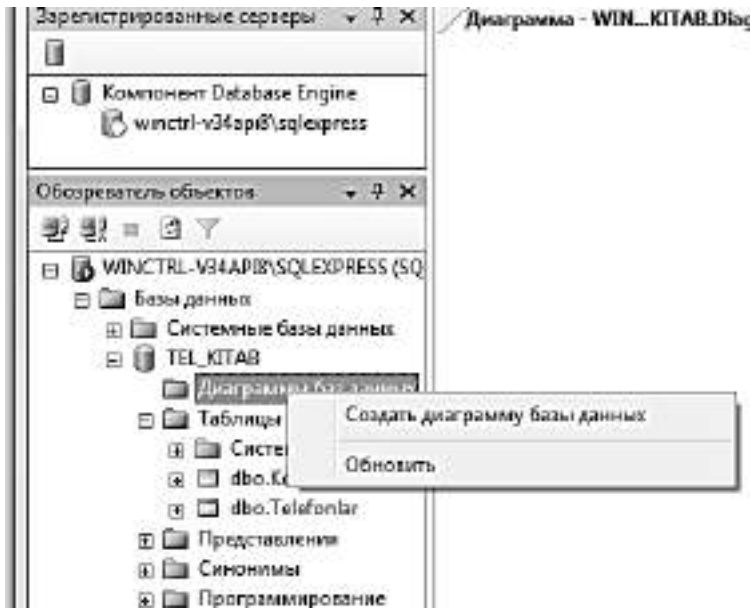
Имя столбца	Тип данных	Разрешить значения null
id	int	<input type="checkbox"/>
idKontakt	int	<input type="checkbox"/>
Telefon	varchar(50)	<input type="checkbox"/>

Şəkil 5.8. Kontaktlar cədvəlinin strukturu

5.2.3. Cədvəllərin əlaqələndirilməsi

VB cədvəllərinin arasında əlaqə və verilənlər bazasının sxemi yaratmaq üçün yeni verilənlər bazası diaqramı yaratmaq lazımdır. Bunun üçün Database Diagram (Диаграммы баз данных) qovluğunun kontekst menyusundan Add New Diagram (Создать диаграмму базы данных) əmrini icra etmək lazımdır (şəkil 5.9).

Bu əmrin icrasından sonra açılacaq pəncərədə aralarında əlaqə yaradılacaq cədvəlləri bir-bir seçib Add (Добавить) düyməsini basmaq lazımdır. *Management Studio* pəncərəsi şəkil 5.10-da göstəriləyi kimi olacaqdır.

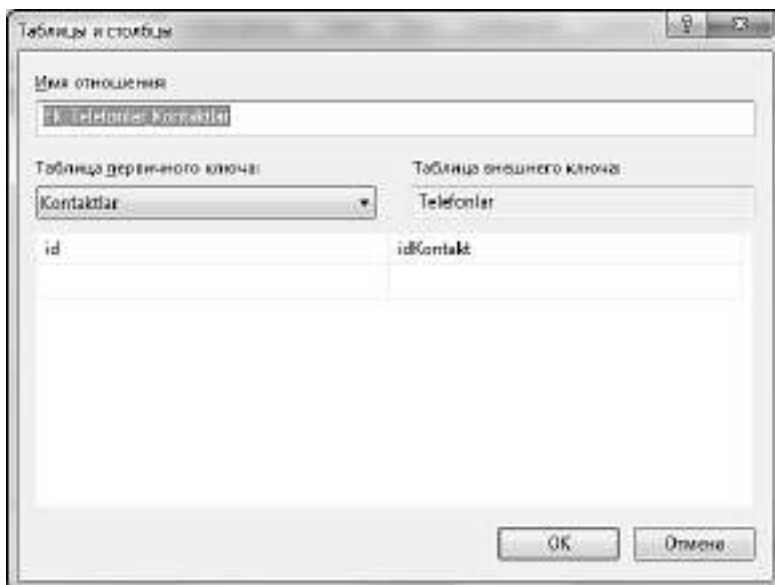


Şəkil 5.9. Cədvəllər arasında əlaqələrin yaradılması



Şəkil 5.10. Əlaqələndiriləcək cədvəllərin əlavə edilməsi

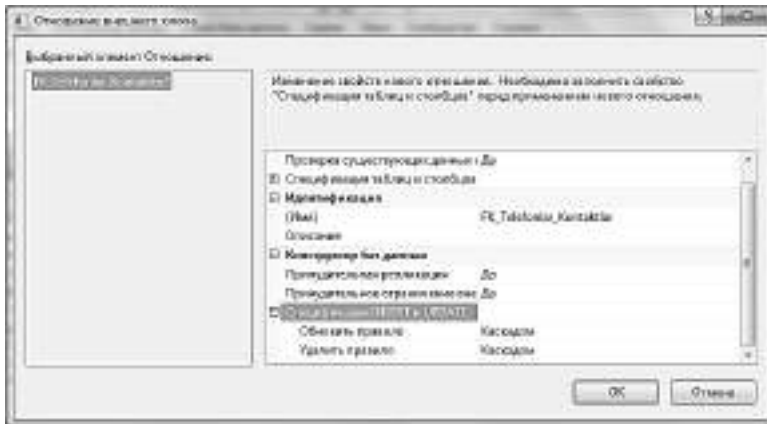
Əlaqənin yaradılması müvafiq sahələri mausla dartıb birləşdirməklə yerinə yetirilir. Biz Kontaktlar cədvəlinin id sahəsi ilə Telefonlar cədvəlinin idKontakt sahəsini əlaqələndirəcəyik. Ona görə də mausun düyməsini id sahəsində basıb, mausu idKontakt sahəsinə dartırıq. Bu zaman eyni zamanda iki pəncərə əmələ gələcəkdir: Tables and Columns (Таблицы и столбцы) və Foreign Key Relationship (Отношение внешнего ключа). Таблицы и столбцы pəncərəsində (şəkil 5.11) cədvəllərarası münasibətə ad verilir və əsas və təbə olan cədvəllərin ilkin və xarici açar sahələri göstərilir.



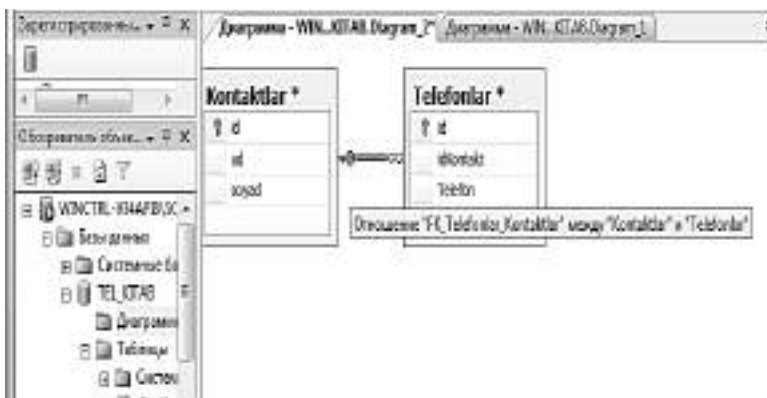
Şəkil 5.11. Таблицы и столбцы pəncərəsi

Bu pəncərədə Ok düyməsini basıb, Отношение внешнего ключа pəncərəsinə keçdikdə (şəkil 5.12) INSERT and UPDATE Specifical (Спецификация INSERT и

UPDATE) bəndini açıb, cədvəldə əlaqələndirilmiş verilənlərin yenilənməsi və pozulması qaydalarını каскадом seçməliyik. Bundan sonra Ok düyməsini basdıqda cədvəllər arasında şəkil 5.13-də göstərilmiş əlaqə yaradılacaqdır.



Şəkil 5.12. Отношение внешнего ключа *pəncərəsi*



Şəkil 5.13. Cədvəllər arasında əlaqənin yaradılması

Əlaqə yaradıldıqdan sonra, disket nişanını basaraq sxemi yadda saxlamaq lazımdır. Qeyd edək ki, Ms Access verilənlərinin sxemindən fərqli olaraq, Microsoft SQL Server-də bir neçə diaqram yaradıla bilər.

5.2.4. Cədvəlin açılması və verilənlərin daxil edilməsi

Cədvələ verilənləri daxil etmək və ya verilənlərə düzəlişlər etmək üçün müvafiq cədvəlin adı üzərində mausun düyməsini basıb, kontekst menyudan Open (Открыть) əmrini icra etmək lazımdır (şəkil 5.14).



Şəkil 5.14. Cədvələ verilənlərin daxil edilməsi

Bununla da biz Microsoft SQL Server verilənlər bazasının yaradılmasını başa çatdırırıq, və onun bütün imkanlarını öyrənməyiniz üçün digər mənbələrdən istifadə etməyinizi tövsiyə edirik.

Altıncı fəsil

VISUAL STUDIO MÜHİTİNDƏ ADO İLƏ VERİLƏNLƏR BAZALARINA İNTEQRASIYA

6.1. ADO.NET texnologiyası

Bu gün verilənlərlə iş böyük əhəmiyyətə malikdir. Verilənlərin saxlanması üçün müxtəlif verilənlər bazasını idarə sistemləri (VBİS) istifadə olunur: MS SQL Server, Oracle, MySQL və s. Verilənləri saxlamaq üçün böyük əlavələrin əksəriyyəti bu və ya digər VBİS-lərdən istifadə edir. Ancaq verilənlər bazası ilə C# arasında əlaqə qurmaq üçün vasitəçi lazımdır. Məhz ADO.NET (Active Data Object .NET) texnologiyası belə vasitəçidir.

ADO.NET verilənlərlə işləmək üçün nəzərdə tutulmuş, .NET Framework platformasına əsaslanmış texnologiyadır. Bu texnologiya bizə siniflərin yığımını təqdim edir ki, biz onların vasitəsi ilə verilənlər bazalarına sorğular göndərə bilərik, onlara qoşula bilərik, verilənlər bazasından cavab ala bilərik və bir sıra başqa əməliyyatları yerinə yetirə bilərik.

Həm də qeyd etmək vacibdir ki, VBİS-lər çox ola bilər. Öz mahiyyətlərinə görə onlar bir-birindən kəskin fərqlənə bilər. MS SQL Server, məsələn, sorğuların yaradılması üçün T-SQL dilini istifadə edir, amma MySQL və Oracle VBİS-ləri isə PL-SQL dilini tətbiq edir. Müxtəlif verilənlər bazaları sistemləri müxtəlif tipli verilənlərə malik ola bilər. Həmçinin hansı isə başqa fərqli cəhətləri ola bilər. Ancaq ADO.NET funksionalı elə qurulmuşdur ki, istifadəçilərə ən müxtəlif VBİS-lərlə işləmək üçün unifikasiya edilmiş interfeys təqdim edir.

ADO.NET-də verilənlər bazaları ilə qarşılıqlı təsir interfeysinin əsasını məhdud obyektlər dairəsi təşkil edir. Həmin

obyektlər bunlardır: Connection, Command, DataReader, DataSet və DataAdapter. Connection obyektinin köməyi ilə verilənlər mənbəyinə qoşulma yaradılır. Command obyektini VB verilənləri ilə əməliyyatları yerinə yetirməyə imkan verir. DataReader obyektini sorğu nəticəsində alınmış verilənləri oxuyur. DataSet obyektini VB-də verilənlərin saxlanması üçün nəzərdə tutulmuşdur və VB-dən asılı olmayaraq onlarla işləməyə icazə verir. DataAdapter obyektini DataSet ilə verilənlər mənbəyinin arasında vasitəçidir. Verilənlər bazası ilə işlər, əsasən, bu obyektlər vasitəsilə yerinə yetirilir.

Ancaq müxtəlif verilənlər mənbələri üçün eyni obyektlər yığımından istifadə etmək üçün, müvafiq verilənlər provayderi lazımdır. Məhz verilənlər provayderinin köməyi ilə ADO.NET-də verilənlər bazası ilə qarşılıqlı təsir həyata keçirilir. Həm də hər verilənlər mənbəyi üçün ADO.NET-in öz provayderi ola bilər ki, məhz yuxarıda göstərilən sinifləri konkret reallaşdırın.

Susmaya görə ADO.NET-də aşağıdakı təməl provayderləri mövcuddur:

- MS SQL Server üçün provayder;
- OLE DB provayderi (MS SQL Server-in bəzi köhnə versiyalarına, həmçinin Access, DB2, MySQL və Oracle VB-ina girişi təmin edir);
- ODBC üçün provayder (öz provayderi olmayan verilənlər mənbələri üçün provayder);
- Oracle üçün provayder;
- EntityClient provayderi. ORM Entity Framework texnologiyası üçün verilənlər provayderi;
- SQL Server Compact 4.0 serveri üçün provayder.

Bu təməl provayderlərindən başqa, müxtəlif verilənlər bazaları üçün, məsələn, MySQL üçün nəzərdə tutulmuş çoxlu provayderlər vardır.

ADO.NET kitabxanası - müxtəlif verilənlər bazaları ilə qarşılıqlı təsir üçün nəzərdə tutulmuş siniflər yığımıdır. Kitabxana verilənlərlə işləmək üçün lazım olan siniflərdən ibarətdir. Onların köməyi ilə Siz serverə qoşula bilərsiniz, sorğu tərtib edib serverə göndərə bilərsiniz və nəticəni alıb onu emal edə bilərsiniz.

Verilənlər bazaları ilə işləmək üçün ADO.NET kitabxanası birinci və yeganə kitabxana deyil. Təkcə Microsoft firması

tərəfindən DAO (Data Access Objects), RDO (Remote Data Objects), ODBC (Open DataBase Connectivity), ADO (Active Data Objects) kimi texnologiyalar hazırlanmışdır. Digər şirkətlər də müxtəlif texnologiyalar hazırlamışdır ki, onların da özünəməxsus üstünlükləri və çatışmazlıqları vardır.

İndi .NET-də verilənlər bazaları ilə işləmək üçün əsas vasitə ADO.NET texnologiyasıdır.

ADO.NET-in ADO-nun sələflərindən iki əsas üstünlüyü vardır:

- çox geniş yayılmış məşhur XML-in tam dəstəklənməsi;
- serverdə verilənlərin yenilənməsi məntiqinə tam nəzarətin mümkünlüyü.

İkinci bənd kodun yazılmasının çevikliyi nöqtəyi-nəzərindən və təhlükəsizlik nöqtəyi-nəzərindən çox əhəmiyyətlidir. Məsələn, əgər Sizin bazanızda verilənlərin SQL dilinin UPDATE operatorunun köməyi ilə birbaşa dəyişdirilməsi qadağandırsa, onda Siz saxlanılan prosedurlar vasitəsi ilə verilənlərin yenilənməsi üçün özünüz kod yaza bilərsiniz. Təkcə bu yenilik ADO.NET-ə keçmək haqqında fikirləşməyə dəyər.

ADO.NET-in ibarət olduğu sinifləri şərti olaraq iki kateqoriyaya bölmək olar: qoşulma tələb edənlər (bəzən *qoşulmuşlar* və ya *connected* ifadələri ilə qarşılaşırıq) və qoşulma tələb etməyənlər (və ya, başqa cür desək, *əlaqəsi kəsilmiş*, çünki, termin ingiliscə *disconnected* kimi səslənir). Adından göründüyü kimi, siniflərin birinci kateqoriyası verilənlər bazasına qoşulmanın mövcudluğunu tələb edir. İkinci kateqoriya isə qoşulmanın mövcudluğunu tələb etmir, çünki artıq kliyent maşınına yüklənmiş verilənlərlə işləyir və qoşulma bağlı ola bilər.

Qoşulmanın mövcudluğunu tələb edən siniflərə aşağıdakılar aiddir: Connection, Transaction, DataAdapter, Command, Parameter, DataReader. Qoşulmanın mövcudluğunu tələb etməyən ikinci kateqoriyaya isə aşağıdakıları aid etmək olar: DataSet, DataTable, DataRow, DataColumn, Constraint, DataView

Verilənlər bazalarına qoşulma və onlarla bilavasitə işləmək üçün verilənləri tədarük edənlər cavabdehirlər. .NET-də verilənlərin iki tədarükçüsü var: SQL Client .NET Data Provider və OLE DB .NET Data Provider. Onlardan birincisi yalnız

Microsoft SQL Server 7 və daha yuxarı versiyasının verilənlər bazaları ilə işləmək üçün nəzərdə tutulmuşdur. Dar çərçivə istiqamətində, yəni, yalnız bir tədarükçüdən bir verilənlər bazasına istifadəyə hesablanmış serverlə işin maksimal effektivliyi sinif və provayderlərin kodu ilə optimallaşdırıla bilər.

Microsoft şirkəti hər bir verilənlər bazası üçün, məsələn, Microsoft SQL Server kimi, provayderlər yaratmadı. Verilənlər bazalarını istehsal edənlərin özləri verilənlərə optimallaşdırılmış girişi təmin edən kitabxanalar yazdı bilər, amma bu o qədər də asan məsələ deyil. Bunun əvəzinə, Microsoft şirkəti istənilən verilənlər bazasına qoşulmağa imkan verən universal OLE DB .NET Data Provider provayderini yaratdı ki, onun OLE DB kimi verilənlər tədarükçüsü var. Hazırda verilənlər bazalarının əksəriyyəti üçün belə tədarükçülər var, buna görə də ikinci provayderin köməyi ilə biz, praktiki olaraq, istənilən verilənlər bazasına qoşula bilərik.

Microsoft, SQL Server OLE DB drayverinə malik olduğundan, bu bazaya yuxarıda adlarını sadaladığımız iki provayderdən istənilən birinin köməyi ilə qoşulmaq olar. Əlbəttə, SQL Client .NET Data Provider daha yaxşı işləyir, amma OLE DB .NET Data Provider istənilən verilənlər bazaları ilə işləyə biləcək universal kod yaratmağa imkan verir. Sinifləri bir-birindən ayırmaq üçün, onlar müxtəlif adlar sahəsində yerləşdirilmişdir. SQL Client .NET Data Provider ilə işləmək üçün siniflər `System.Data.SqlClient` və OLE DB .NET Data Provider ilə işləmək üçün siniflər `System.Data.OleDb` ad sahələrində yerləşdirilmişdir. Biz yalnız universal OLE DB drayverinin siniflərinə baxacağıq.

Əgər Siz provayderlərdən hansını seçəcəyinizi bilmirsinizsə, onda sadəcə olaraq, layihəinizdə hansı verilənlər bazasının istifadə olunacağını müəyyən etməlisiniz. Əgər Microsoft SQL Server-lə işləyəcəksinizsə, onda, əlbəttə, SQL Client .NET Data Provider provayderini seçmək lazımdır. Digər verilənlər bazası ilə işlədikdə və ya verilənlər mənbəyini müəyyən edə bilmədikdə, OLE DB .NET Data Provider provayderini seçmək daha yaxşıdır. Ümumiyyətlə isə, variant seçimində qeyri-müəyyənlik olmamalıdır. Siz qabaqcadan proqramın tələblərini bilməlisiniz və istifadə edilən verilənlər mənbəyi haqqında qərar qəbul etməlisiniz.

6.2. Ms Access-də yaradılmış verilənlər bazasına qoşulma

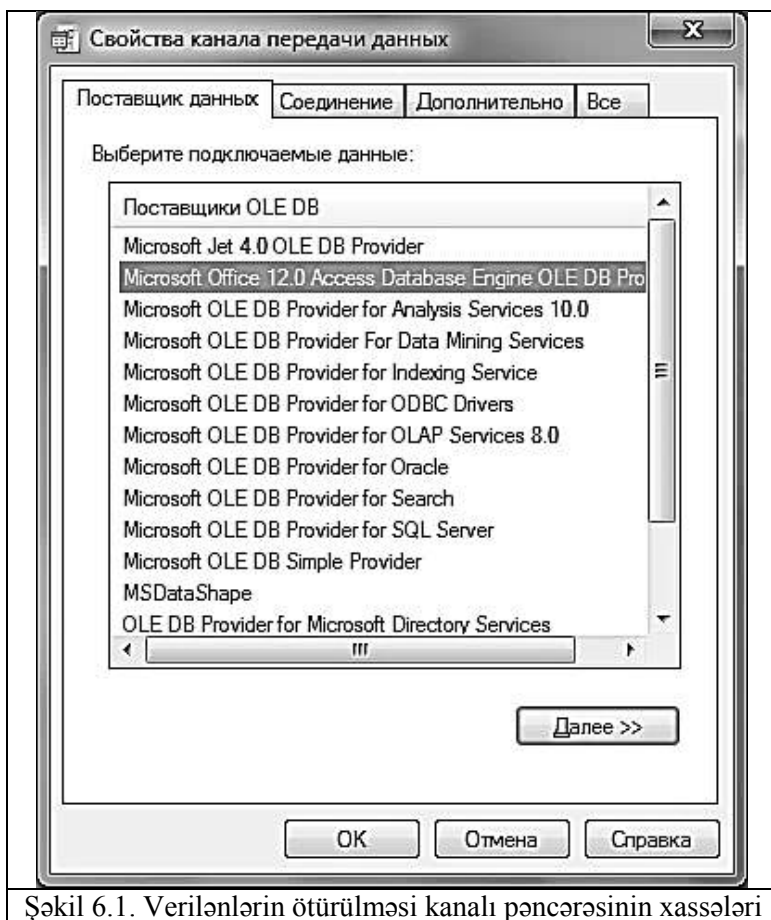
Verilənlər bazasına qoşulmaq üçün `OleDbConnection` sinfindən istifadə edilir. Qoşulma sətirinin (`ConnectionString`) köməyi ilə verilənlər bazasına qoşulma parametrlərini bu sinfə göstərmək lazımdır. Bu sətir vasitəsi ilə komponent verilənlər bazasının harada yerləşdiyini, qoşulma və avtorizasiya üçün hansı parametrlərdən istifadə edilməsini müəyyən edir. Qoşulma sətirini müstəqil olaraq əl ilə yazmaq və ya ADO-nun təməlinə mövcud olan qoşulma sətirinin yaradılması pəncərəsindən istifadə etmək olar.

Qoşulma sətirinin yaradılması pəncərəsindən istifadə etmək daha rahatdır. Bunun üçün, diskin istənilən yerində udl genişlənməsinə malik olan istənilən adlı fayl yaradın. Bunu bələdçidə (`Explorer-Проводник`) və ya istənilən başqa fayl menecerində etmək olar. Faylın adı və onun yeri əhəmiyyət kəsb etmir, əsas odur ki, onun genişlənməsi hissəsi udl olsun. Həmin fayl üzərində mausun düyməsini iki dəfə basdıqda şəkil 6.1-də göstərilmiş Verilənlərin ötürülməsi kanalının xassələri `Data Link Properties` (`Свойства канала передачи данных`) pəncərəsi ekranda təsvir ediləcəkdir.

Əvvəlcə bu pəncərənin `Provider` (`Поставщик данных`) səhifəsini açaq. Burada `.NET` verilənlər tədarükçülərinin istifadə edilə biləcəkləri drayverlərin siyahısı göstərilmişdir. Bu drayverlərin əksəriyyətlərinin adlarında `OLE DB` yazısı iştirak edir. Bu o deməkdir ki, tədarükçü məhz `OLE DB` drayverindən istifadə edəcəkdir.

Bu drayverlərin əksəriyyətlərinin adlarında həmçinin verilənlər bazalarının adları iştirak edir ki, onların köməyi ilə həmin bazalara qoşulmaq olar. Bunların içərisində ən məşhur drayver geniş kütlələr tərəfindən istifadə edilən `Microsoft Access` verilənlər bazalarına qoşulma drayveridir. Bu drayver `Microsoft Jet 4.0 OLE DB Provider` adlanır və o bütün kompüterlərdə olur. **JET**– `Ms Access`–ə quraşdırılmış `VBIS`–lərlə iş mexanizminin adıdır (əslində **JET** mexanizmi digər çoxlu sayda lokal verilənlər mənbələri ilə də işləməyə imkan verir). *JET* adı həmişə `Access`–lə əlaqələndirilir, həqiqətən də `Access`

JET–lə qarşılıqlı əlaqədə olan əsas sistemdir. Hazırda üzərində dayandığımız **Microsoft Jet 4.0 OLE DB Provider** yalnız **.mdb** formatlı Access cədvəlləri ilə əlaqə yarada bilər. Əgər VB cədvəlləri **Access 2007** və daha yüksək versiyalarda yaradılmışdırsa, onda **Microsoft Office 12.0 Access Database Engine OLE DB Provider** seçilməlidir.

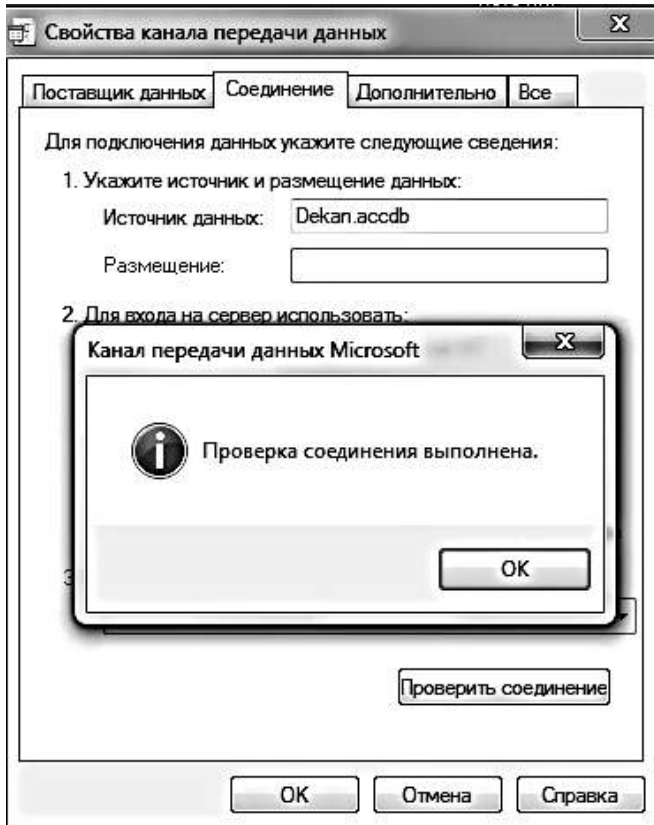


Şəkil 6.1. Verilənlərin ötürülməsi kanalı pəncərəsinin xassələri

İndi qoşulma sətrinin yaradılması pəncərəsinin ikinci səhifəsinə keçək. Onun məzmunu istifadə edilən verilənlər

bazasından asılıdır. Microsoft Access verilənlər bazası üçün bu səhifə şəkil 6.2-də göstərildiyi kimi təsvir ediləcəkdir.

Bazaya qoşulmaq üçün birinci boş sahəyə Access faylının adını yazmaq kifayətdir. İstifadəçini və şifri göstərmək vacib deyil.



Şəkil 6.2. Ms Access bazasına qoşulmanın sazlanması

Test Connection (Проверить соединение) düyməsini basaraq birləşmənin yaranmasını yoxlamaq olar: əgər birləşmə baş vermişdirsə, onda Test Connection Succeeded (Проверка соединения выполнена) məlumatından ibarət

dialog pəncərəsi peyda olacaqdır. Bununla da Ms Access verilənlər bazasına qoşulma sona çatır.

Genişlənmiş hissəsi udl olan fayla, qoşulma ilə əlaqədar dəyişikliklər etmək üçün, onu kontekst menyunun *Open with/OLE DB Core Services (Открыть с помощью/OLE DB Core Services)* əmri ilə açmaq olar. Əgər bu faylı kontekst menyunun *Open with/Notepad (Открыть с помощью/Блокнот)* əmri ilə açsaq, orada aşağıdakı mətni görəəcəyik:

```
[oledb]
; Everything after this line is an OLE
DB initstring
Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=Dekan.accdb;Persist Security
Info=False
```

Birinci sətir OLE DB qoşulmasının başlanğıcını bildirən bölmənin başlanğıcını bildirir. İkinci sətir nöqtəli vergül simvolu ilə başlayır ki, bu da şərhə bildirir. Üçüncü sətir isə qoşulma sətridir ki, bu sətri birin birə proqram koduna köçürmək və proqramda istifadə etmək olar.

Verilənlər bazasına qoşulma sətri bir-birindən nöqtəli vergüllə ayrılmış `ad=qiymət` şəklində parametrlərdən ibarətdir. Bizim sətrimizdə üç parametr görünür:

`Provider` - bazaya qoşulmaq üçün istifadə edilən provayderin adıdır;

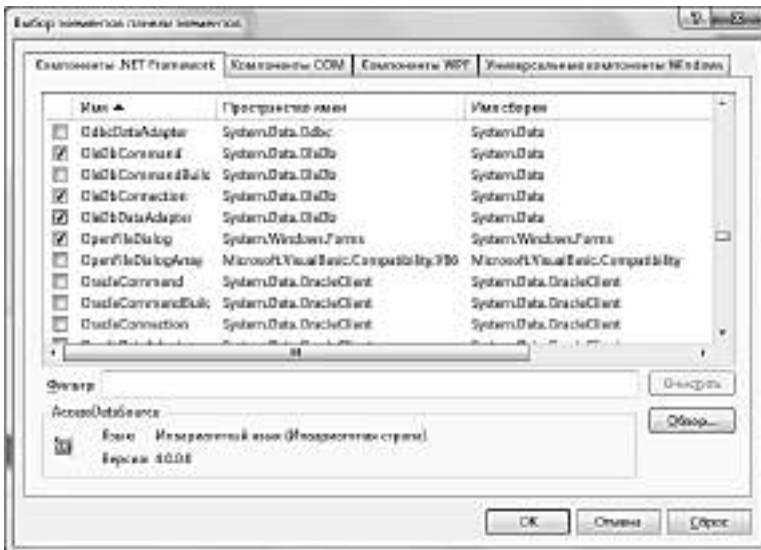
`Data Source` – verilənlər bazasına yoldan ibarət verilənlər mənbəyidir;

`Persist security Info` – onu müəyyən edir ki, qoşulma sətrində müəyyənləşdirilmə üçün lazım olan şifrlər kimi informasiya saxlana bilər.

İndi isə ADO texnologiyasının vasitəsi ilə Ms Access 2010-da yaradılmış `Dekan.accdb` adlı verilənlər bazasına qoşulmaq üçün C# əlavəsi yaradaq.

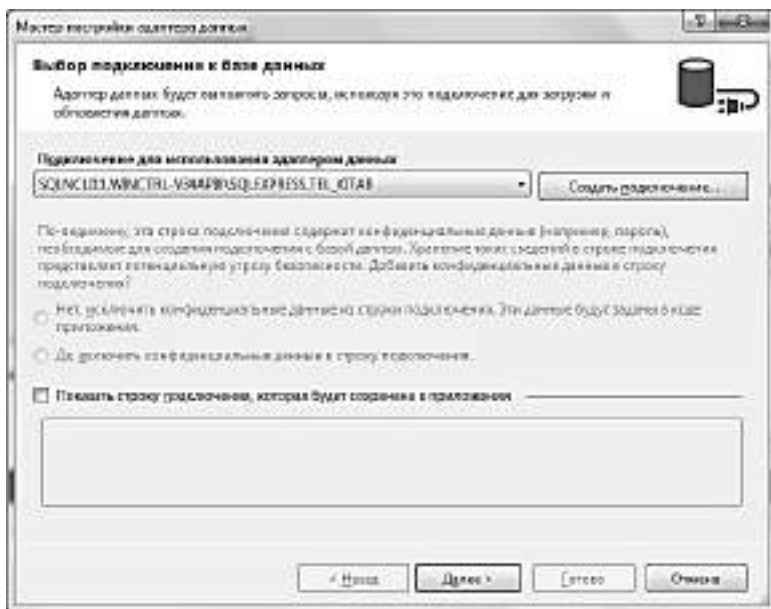
Visual Studio mühitini işə salıb Windows Forms App (.NET Framework) (Приложение Windows Forms (.NET Framework)) əlavəsi yaradaq.

Toolbox (Панель элементов) panelində Data (Данные) səhifəsini açıb forma üzərinə OleDbDataAdapter komponentini yerləşdirək. Əgər həmin səhifədə bu komponent görünməzsə, onda Toolbox (Панель элементов) panelinin Data (Данные) səhifəsində mausun düyməsini basıb Choose Items... (Выбрать элементы...) əmrini icra etmək lazımdır. Bu zaman açılan Choose Toolbox Items (Выбор элементов панели элементов) pəncərəsində lazım olan komponenti seçib Ok düyməsini basmaq lazımdır (şəkil 6.3).



Şəkil 6.3. Data (Данные) səhifəsinə elementlərin əlavə edilməsi

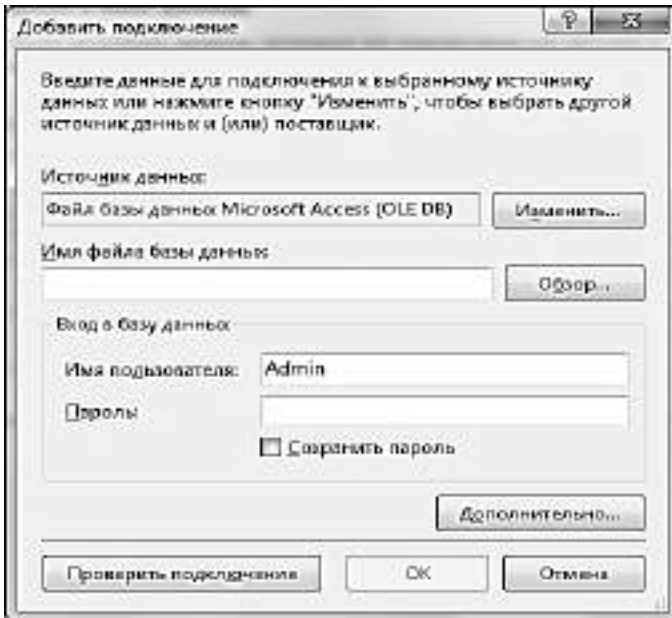
Forma üzərində OleDbDataAdapter komponentini yerləşdirdikdə (əslində o forma üzərində deyil, formanın altındakı boş sahədə yerləşəcək) dərhal Data Adapter Configuration Wizard (Мастер настройки адаптера данных) pəncərəsi ekranda təsvir ediləcəkdir (şəkil 6.4).



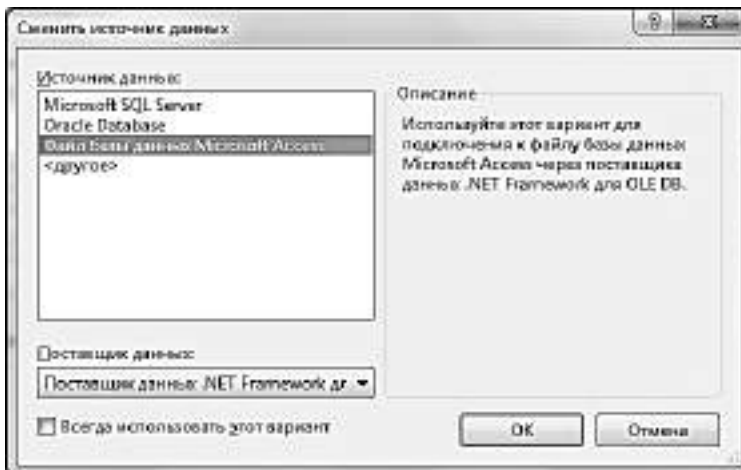
Şəkil 6.4. Verilənlər adapterinin sazlanması

Bu pəncərənin Which data connection should the data adapter use (Подключение для использования адаптером данных) açılan siyahısında əvvəllər istifadə edilmiş qoşulmaların adları yerləşir. Əgər bu siyahı boşdursa və ya Dekan.accdb faylının adı yoxdursa, onda yeni qoşulma yaratmaq lazımdır. Bunun üçün New Connection... (Создать подключение...) düyməsini basmaq lazımdır.

Yeni açılan pəncərənin (şəkil 6.5) Data Source: (Источник данных:) sahəsində Change... (Изменить...) düyməsinin köməyi ilə Microsoft Access Database File (Файл базы данных Microsoft Access) və Data Provider: (Поставщик данных:) sahəsində isə .NET Framework Data Provider for OLE DB (Поставщик данных .NET Framework для OLE DB) seçmək lazımdır (şəkil 6.6).

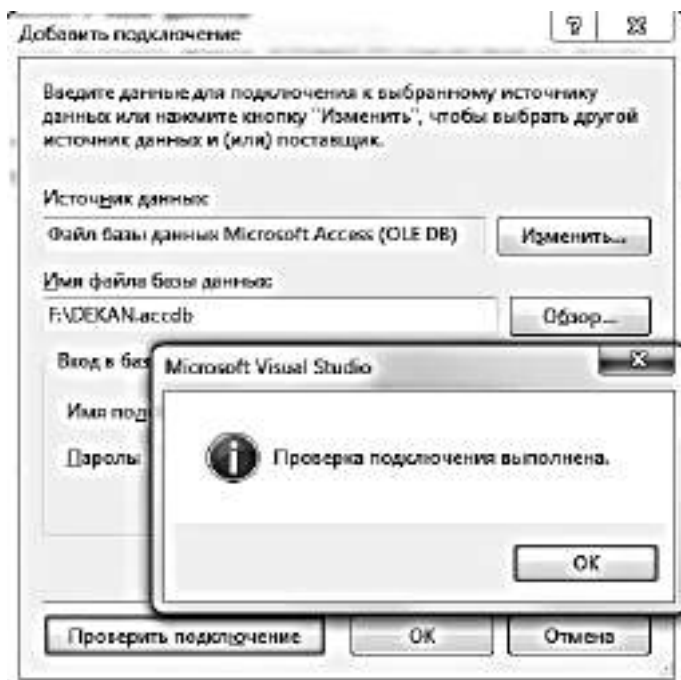


Şəkil 6.5. Qoşulmanın əlavə edilməsi



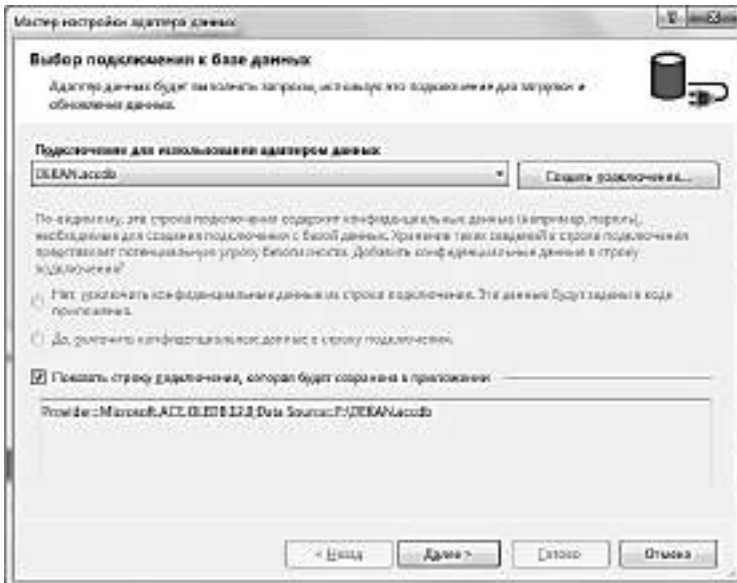
Şəkil 6.6. Verilənlər mənbəyinin seçilməsi

Add Connection (Добавить подключение) pəncərəsinə qayıtdıqda Browse... (Обзор...) düyməsini basmaqla Database File Name: (Имя файла базы данных:) sahəsində Dekan.accdb faylını göstərmək və Test Connection (Проверить подключение) (şəkil 6.7) düyməsini basıb qoşulmanın yerinə yetirilməsini yoxlamaq lazımdır. Əgər qoşulma alınmışdırsa, onda ekranda Test connection succeeded (Проверка подключения выполнена) məlumatından ibarət dialoq pəncərəsi təsvir olunacaqdır. Bu pəncərədəki Advanced... (Дополнительно...) düyməsi ADO mexanizminin çoxlu sayda müxtəlif parametrlərini seçməyə imkan verir.



Şəkil 6.7. Qoşulmanın yoxlanması

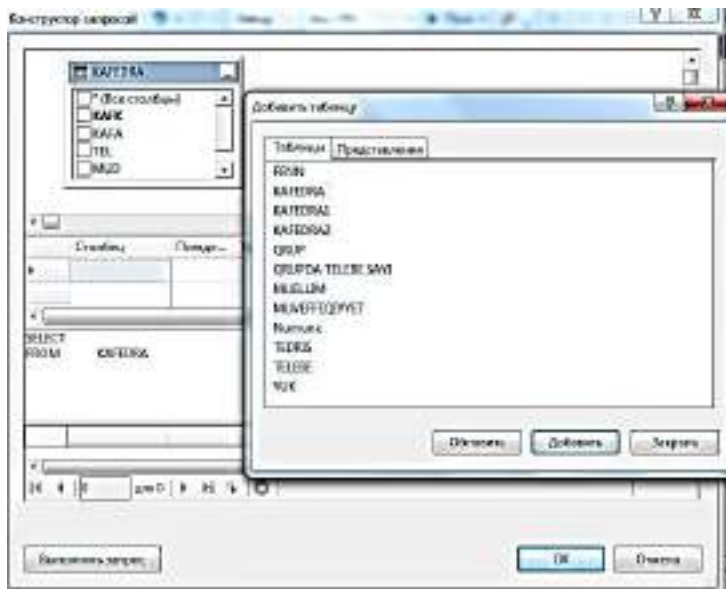
Ok düyməsini basdıqda Data Adapter Configuration Wizard (Мастер настройки адаптера данных) pəncərəsinə qayıdacağıq (şəkil 6.8).



Şəkil 6.8. Yaradılmış qoşulma

Bu pəncərədə Next> (Далее>) düyməsini basıb açılan pəncərədə Access üçün mümkün olan yeganə variantı, yəni, Use SQL statements (Использовать инструкции SQL) dəyişdiricisini qoşub yenidən Next> (Далее>) düyməsini basmaq lazımdır. Növbəti pəncərədə SQL sorğusunun yaradılması tələb olunur. Bu pəncərənin mətn sahəsində əl ilə SQL kodlarını yazmaq olar. Lakin, sorğunu Query Builder... (Конструктор запросов...) düyməsini basmaqla yaratmaq daha rahatdır. Bu düyməni basdıqda ekranda dərhal iki pəncərə təsvir olunacaqdır: Query Builder (Конструктор запросов) və Add Table (Добавить таблицу) (şəkil 6.9). Kafedra cədvəlini seçib Add (Добавить) düyməsini basdıqda

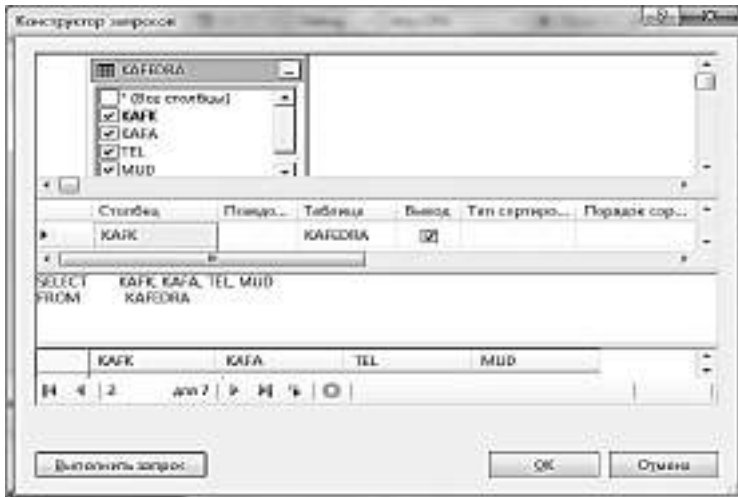
həmin cədvəl Query Builder (Конструктор запросов) pəncərəsində təsvir ediləcəkdir.



Şəkil 6.9. VB cədvəllərinin əlavə edilməsi və sorğu pəncərələri

Cədvəlin sahələrini seçib Execute Query (Выполнить запрос) düyməsini basdıqda sorğu yaradılacaqdır (şəkil 6.10).

Ok düyməsini basmaqla Data Adapter Configuration Wizard (Мастер настройки адаптера данных) pəncərəsinə qayıtdıqda (şəkil 6.11) Advanced Options... (Дополнительно...) düyməsini basıb açılan pəncərədə yalnız Generate Insert, Update and Delete Statements (Создать инструкции Insert, Update и Delete) dəyişdiricisini qoşaq.

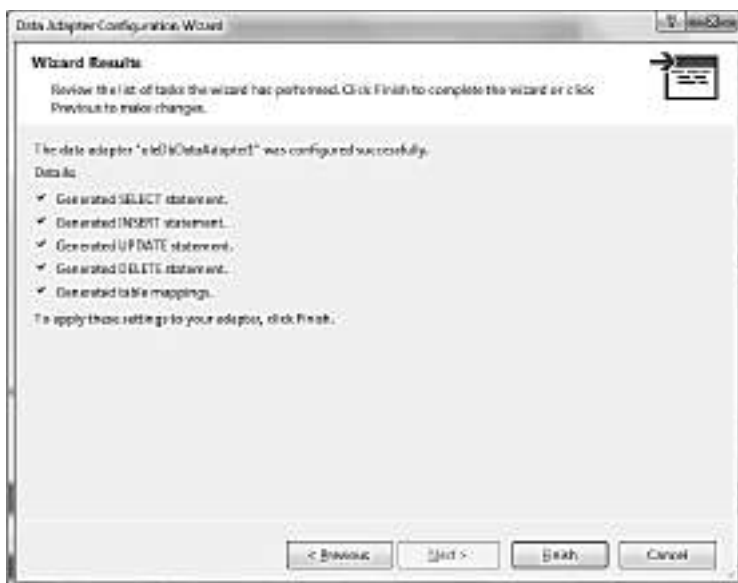


Şəkil 6.10. Sorğunun yaradılması



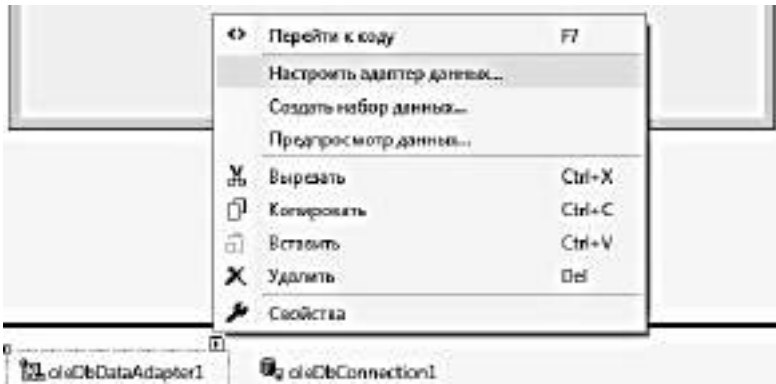
Şəkil 6.11. Data Adapter Configuration Wizard pəncərəsi

Bu, bizə cədvəl verilənlərinə düzəlişlər etməyə imkan verəcəkdir. Next> (Далее>) düyməsini basaraq Data Adapter Configuration Wizard (Мастер настройки адаптера данных) pəncərəsinə keçib, Finish (Готово) düyməsini basmaq lazımdır (şəkil 6.12). Bununla da baza ilə əlaqə yaranacaqdır.



Şəkil 6.12. Qoşulmanın yekunlaşması pəncərəsi

Diqqət yetirin ki, formadan aşağıdakı boş sahədə `oleDbDataAdapter1` komponenti ilə yanaşı yeni `oleDbConnection1` komponenti meydana oldu. `oleDbDataAdapter1` komponentinin kontekst menyusunun `Configure Data Adapter...` (Настроить адаптер данных...) əmri ilə (şəkil 6.13) qoşulma parametrlərinə dəyişikliklər etmək və ya yarımçıq qalmış sazlamaları başa çatdırmaq olar.

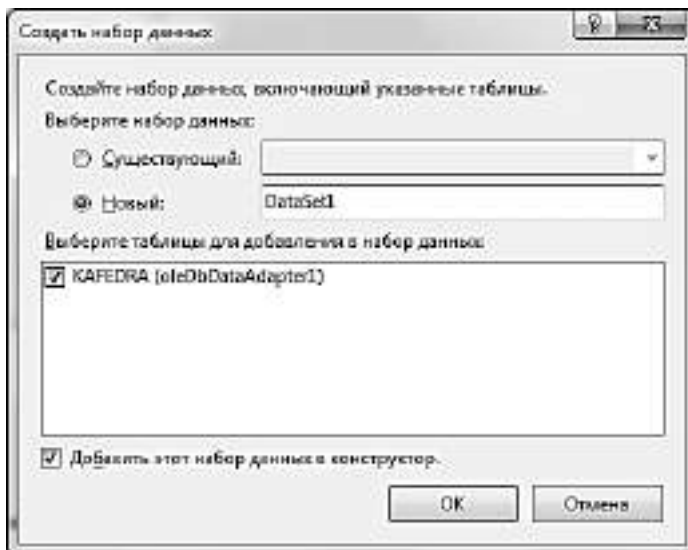


Şəkil 6.13. Verilənlər adapterinin sazlanması

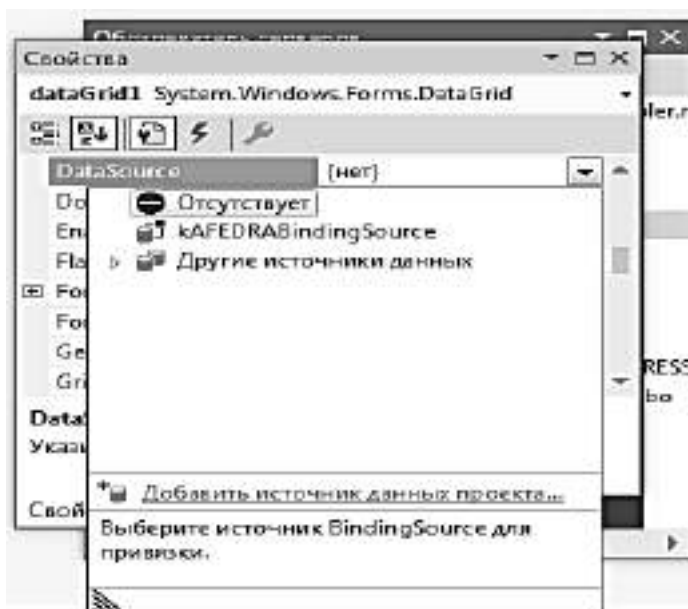
Solution Explorer (Обозреватель решений) pəncərəsində Form1.Designer.cs faylını açıb OleDbCommand, OleDbDataAdapter və OleDbConnection siniflərinin nüsxələrinin yaradılmasına əmin olun.

oleDbDataAdapter komponentinin kontekst menyusundan Generate DataSet... (Создать набор данных...) əmrini icra edək. Ekranda təsvir edilən pəncərədə (şəkil 6.14) New (Новый) dəyişdiricisini qoşub, Ok düyməsini basmaq lazımdır. Bununla da formanın aşağı hissəsindəki boş sahədə dataSet11 verilənlər yığımı əmələ gələcəkdir.

Cədvəlin verilənlərini təsvir etdirmək üçün forma üzərinə dataGrid komponenti yerləşdirək. Bu komponenti verilənlər yığımı ilə əlaqələndirmək lazımdır. Ona görə onu seçib Properties (Свойства) pəncərəsində onun DataSource xassəsinin siyahısını açıb, oradan Add Project Data Source... (Добавить источник данных проекта...) əmrini icra edək (şəkil 6.15).



Şəkil 6.14. Verilənlər yığımının yaradılması



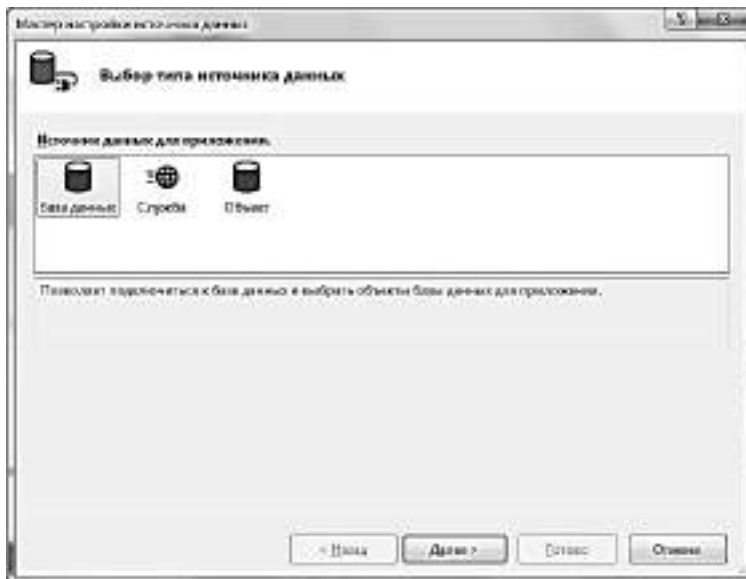
Şəkil 6.15. Add Project Data Source... dialogu

Bu zaman açılan Data Source Configuration Wizard (Мастер настройки источника данных) pəncərəsi (şəkil 6.16) aşağıdakı verilənlər mənbəyindən birini seçməyə imkan verir:

- Database (База данных) – verilənlər bazası;
- Service (Служба) – xidmət təklif edən servisdır.

Əksər hallarda bu, Web-servisdır;

– Object (Объект) – verilənləri yaradan obyektləri seçmək üçün obyekt və onlarla işləmək üçün obyektlərdir.



Şəkil 6.16. Data Source Configuration Wizard pəncərəsi

Bizə Database (База данных) verilənlər mənbəyini seçmək lazımdır və Next> (Далее>) düyməsini basırıq. Növbəti açılan pəncərədə Dataset (Набор данных) seçib, yenidən Next> (Далее>) düyməsini basırıq. Şəkil 6.17-də göstərilmiş pəncərə açılacaqdır. Bu dialoqun məqsədi qoşulma sətirinin yaradılmasıdır ki, burada ADO mexanizmi üçün verilənlər

bazasının tipi, onun yerləşdiyi yer, istifadəçilərin adları, təhlükəsizlik vasitələri və s. kimi qoşulma parametrləri təsvir edilir.



Şəkil 6.17. Qoşulmanın seçilməsi

Bu pəncərədə Which data connection should your application use to connect to the database? (Какое подключение ваше приложение должно использовать для работы с базой данных?) siyahısını açdıqda əvvəllər yaradılmış bütün qoşulmalar təsvir ediləcəkdir. Əgər lazım olan qoşulma yoxdursa, onda New connection... (Создать подключение...) düyməsindən istifadə etmək lazımdır.

Yenidən Next> (Далее>) düyməsini basaraq yeni pəncərəyə keçdikdə qoşulmanın təklif edilən adını (DEKANConnectionString) qəbul edib (şəkil 6.18), yenidən Next> (Далее>) düyməsini basıqda açılan pəncərədən (şəkil) Kafedra cədvəlinin adını seçib Finish (Готово) düyməsini

basırıq. İndi formanın aşağı hissəsində dEKANDataSet, kAFEDRABindingSource və kAFEDRATableAdapter komponentləri əmələ gələcəkdir.



Şəkil 6.18. Yaradılmış qoşulmanın adı

İndi layihəni işə saldıqda forma üzərində Kafedra cədvəlinin sütunları təsvir ediləcəkdir (şəkil 6.19). Göründüyü kimi, bu mərhələdə cədvəl verilənləri hələlik təsvir edilmir.

Cədvələ yeni verilənlər əlavə etmək üçün forma üzərinə button1 düyməsi yerləşdirib onun Text xassəsinə əlavə et mətni yazaq. Biz bu düyməni basdıqda cədvələ yeni sətir əlavə ediləcəkdir. Bu düymə üçün Click hadisə emaledicisini yaradaq:

```
private void button1_Click(object sender,
                        EventArgs e)
{
    OleDbDataAdapter1.Fill(dEKANDataSet);
}
```

	KAFK	KAFA	TEL	MUD	FOTO
*					

Şəkil 6.19. Kafedra cədvəli

Cədvələ edilmiş əlavələrin ilkin bazada yadda saxlanması üçün isə forma üzərinə ikinci düymə yerləşdirib, onun Text xassəsinə Yadda saxla mətni yazıb Click hadisə ernaledicisini yaradaq:

```
private void button2_Click(object sender,
                           EventArgs e)
{
    OleDbDataAdapter1.Update(dEKANDataSet);
}
```

Proqramın nəticəsi şəkil 6.20-də göstərilmişdir.

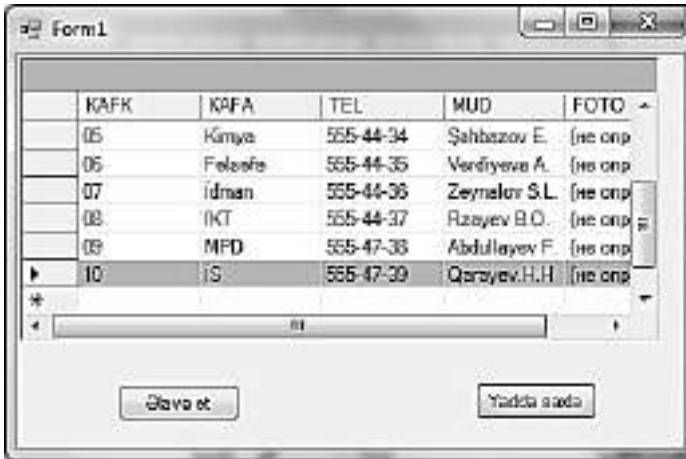
6.3. Visual Studio mühitində ADO ilə iş üçün komponentlər

6.3.1. VBİS-lərə müdaxilə üsulları

Əvvəlki bölmələrdə qeyd etdik ki, VBİS-lərə müdaxilə üsullarını iki böyük qrupa bölmək olar: proqram interfeysləri və obyekt interfeysləri.

İndi obyekt interfeysləri ilə tanış olaq. Belə interfeyslər COM (Component Object Model) komponent obyektlərinin

modellərinin istifadə edilməsinə əsaslanır. COM obyektlərini obyektin xassə və metodlarına müdaxilə etməyə imkan verən interfeyslər yığımı kimi təsəvvür etmək olar.



Şəkil 6.20. Cədvələ yeni verilənlərin daxil edilməsi və yadda saxlanması

Geniş tətbiq edilən obyekt interfeysləri aşağıdakılardır:

- ❖ Remote Data Object (RDO);
- ❖ Object Linking and Embedding, Database (OLE DB);
- ❖ ActiveX Data Objects (ADO).

RDO obyekt interfeysi Microsoft Visual Basic və Visual Basic for Application əlavələrindən Microsoft SQL Server VBİS-in serverinə qoşulmanı asanlaşdırmaq üçün yaradılmışdır. O, ODBC interfeysinin bütün imkanlarını reallaşdırır, lakin, onu istifadə etmək üçün VBİS və ya ODBC-nin proqram interfeysini birbaşa çağırmaq tələb olunmur.

OLE DB obyekt interfeysi əlavələrin verilənlər bazasına universal müdaxiləsi üçün açıq standartdan ibarətdir. Bu interfeys ODBC və RDO interfeyslərindən fərqli olaraq, əlavələrə yalnız relyasiya bazalarına yox, həm də poçt serverlərinə və s. müdaxilə etməyə imkan verir.

OLE DB obyekt interfeysi üç komponentdən ibarətdir: *provayder* (provider), *istehlakçı* (consumer) və verilənlərin emalı və ötürülməsini yerinə yetirən *xidməti* komponent.

İstehlakçı rolunda əlavə çıxış edir. OLE DB provayderinin funksiyası OLE DB interfeysini yerinə yetirməkdir.

OLE DB obyekt interfeysi avtomatlaşdırma mexanizmini icra etmir ki, bunun da nəticəsində bu üsul JavaScript və VB Script server ssenarilərinə əsaslanmış Web-əlavələrinin layihələndirilməsi üçün yaramır.

ADO obyekt interfeysi OLE DB obyekt interfeysi əsasında qurulmuşdur. Bununla yanaşı, OLE DB obyekt interfeysi ODBC üçün Microsoft OLE DB Provider (MSDASQL) və ya SQL Server üçün Microsoft OLE DB Provider (SQLOLEDB) kimi provayderlərin köməyi ilə verilənlərə universal müdaxiləni təmin edir.

ADO obyektləri avtomatlaşdırma vasitələrini reallaşdırdığı üçün ASP, C++, Visual Basic, Visual Basic for Application, Java və s. server ssenariləri kimi əlavələrdən ADO interfeysi əlçatandır.

ADO interfeysinin köməyi ilə əlavələr verilənlər bazası üzərində aşağıdakı əməliyyatları yerinə yetirməyə imkan verir:

- qoşulmanın yaradılması;
- əmr və parametrlərin hazırlanması;
- əmrlərin icrası;
- əmrlərin yerinə yetirilməsinin nəticələrinin emalı;
- qoşulmanın bağlanması;
- səhvlərin emalı.

Verilənlər bazasına müraciət etməzdən əvvəl, əlavə verilənlər bazası serverinə qoşulmalıdır. Bu zaman verilənlər mənbəyinin adı (Data Source Name-DSN) və ya verilənlər mənbəyi haqqında drayverin adı, serverin adı, şifrə və s. kimi informasiya göstərilməlidir.

6.3.1.1. ADO.NET müdaxilə metodu

Yuxarıda baxdığımız proqram və obyekt interfeysli müdaxilə üsulları daha çox *kliyənt-server* əlavələri üçün yararlıdır. Belə əlavələr, adətən, işin başlanğıcında verilənlər bazasına

qoşulmanı açır, iş başa çatdıqda isə qoşulmanı bağlayır. Əgər istifadəçilər çox olarsa, onda onlardan hər biri öz iş müddəti ərzində, VBİS-lə ən azı bir qoşulma saxlayacaqdır. Bu isə çoxlu miqdarda server resursları və server lisenziyaları tələb edir.

6.3.1.1. 1. Çox səviyyəli sistemlər

Verilənlər bazaları ilə inteqrasiyada olan Web-əlavələrin meydana gəlməsi ilə çox səviyyəli sistemlər inkişaf etməyə başladı. Belə sistemlərdə kliyent (brauzer bu rolda çıxış edir) VBİS-ə birbaşa deyil, Web-server vasitəsi ilə müraciət edir.

Belə müraciət o zaman başlayır ki, brauzer Web-serverə sorğu göndərir. Bundan sonra, Web-server aşağıdakı kimi işləyir:

- VBİS-ə qoşulmanı açır;
- verilənlər bazasına müraciət etməklə sorğunu yerinə yetirir;
- verilənlər bazasına qoşulmanı bağlayır;
- sorğunun nəticəsini brauzerə göndərir.

Brauzer sorğunun nəticəsini HTML mətn sənədi formatında alır və onu öz pəncərəsində təsvir etdirir.

Göründüyü kimi, verilənlər bazasına qoşulma yalnız iş prosesi zamanı baş verir, bu isə server resurslarına və kliyent lisenziyalarına qənaət etməyə imkan verir. VBİS-in faktiki kliyenti Web-server olur və lisenziya yalnız onun üçün lazımdır.

6.3.1.1. 2. Əlaqəsi kəsilmiş sistemlər

ADO.NET texnologiyası əlaqəsi kəsilmiş (disconnected) sistemlərə də müdaxilə etməyə imkan verir.

Əlaqəsi kəsilmiş sistemlər verilənlər bazalarından çıxarılmış verilənləri lokal almağa, onları lokal emal etməyə, sonra isə bu emalətmənin nəticələrinə görə serverdə verilənlər bazasını yeniləməyə imkan verir.

ADO.NET metodu ilə VBİS serverindən çıxarılmış verilənlər dataSet sinfinin obyektlərində saxlanır. Bu obyekt özündə eyni zamanda bir neçə verilənlər cədvəli, o cümlədən, bir-

biri ilə əlaqələndirilmiş cədvəlləri və habelə məhdudiyyətləri (constraints) saxlaya bilir. Xüsusi halda, lokal yaradılmış dataSet obyektinə serverdə yerləşən verilənlər bazasının bütün məzmununu yazmaq olar.

ADO.NET müdaxilə metodundan istifadə etməklə əlaqəsi kəsilmiş sistemlərdə kliyentlə serverin qarşılıqlı təsir belə baş verir:

- VBİS serverinə qoşulma açılır;
- verilənlər bazasına sorğu göndərilir;
- qoşulma bağlanır;
- dataSet sinfinin obyektini kimi alınmış verilənlərin emalı;
- VBİS serverinə qoşulma açılır;
- dataSet sinfinin obyektinin məzmunundan istifadə etməklə verilənlər bazası yeniləşdirilir.
- qoşulma bağlanır.

6.3.1.1. 3. Verilənlərin paylanmış emalı

Verilənlərin paylanmış emallı İS-ni yaratmaq tələb olunduqda serverlə kliyentin qarşılıqlı təsiri müəyyənləşdirilməlidir. ADO müdaxilə metodu bu məsələni COM vasitəsi ilə yerinə yetirir, lakin bu üsulun müəyyən çatışmazlıqları vardır. Bu çatışmazlıqlar sistemin qovşaqlarını İnternet kanallarının köməyi ilə birləşdirdikdə özünü göstərir.

Məsələ ondadır ki, İnternetə qoşulmuş korporativ intra şəbəkələr, adətən, brandmauerlə mühafizə olunur. Bu brandmauerlər yalnız müəyyən TCP/IP portları və müəyyən verilənləri ötürmə protokolları üçün müdaxiləni açır. Adətən, yalnız HTTP protokolu vasitəsi ilə Web-serverlə iş üçün nəzərdə tutulmuş 80-ci port və elektron poçtun ötürüldüyü SMTP, POP3 və IMAP protokollarının portları açılır. Bu məhdudiyyətlər COM modelini istifadəsi ilə reallaşdırılan məsafədən emal sistemləri ilə uzlaşmır.

ADO.NET müdaxilə metodu isə verilənləri XML formatı şəklində təsvir edir. Bu halda verilənlər HTTP protokolu ilə ötürülə bilər ki, bu da İS-ni İnternet kanalları ilə birləşdirməyə imkan verir. Bu, hətta sistem brandmauerlə mühafizə olunsay belə, mümkün olur.

6.3.1.1. 4. Verilənlər provayderi

Verilənlər provayderi (data provider) adlanan program komponenti əlavə ilə verilənlər mənbəyi arasında körpü rolunu oynayır. Onun əsas işi mənbədən verilənləri çıxarmaq və verilənlər mənbəyini yeniləşdirməkdir.

.NET platforması üçün Microsoft şirkəti 3 verilənlər provayderi hazırlamışdır. Bunlar SQL Server .NET Data Provider, OLE DB .NET Data Provider və ODBC .NET Data Provider provayderləridir. Bunlardan birinci ikisi Microsoft .NET Framework tərkibinə daxildir, üçüncünü isə Microsoft şirkətinin saytıdan, yəni <http://msdn.microsoft.com/downloads> ünvanından yükləmək olar.

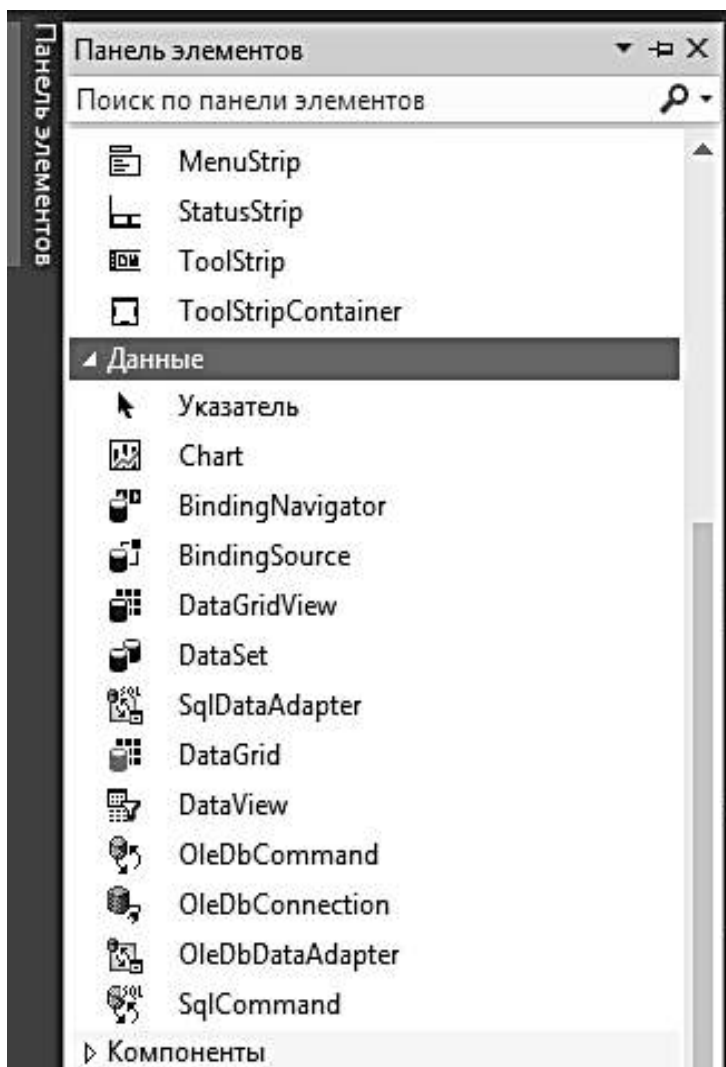
Əgər C# dilində yaradılmış əlavə Microsoft SQL Server VBİS-nin 7.0 və daha yuxarı versiyaları ilə işləyəcəksə, onda SQL Server .NET Data Provider drayverini istifadə etmək lazımdır.

OLE DB .NET Data Provider provayderi Ms Access və digər VB-ina girişi təmin edir.

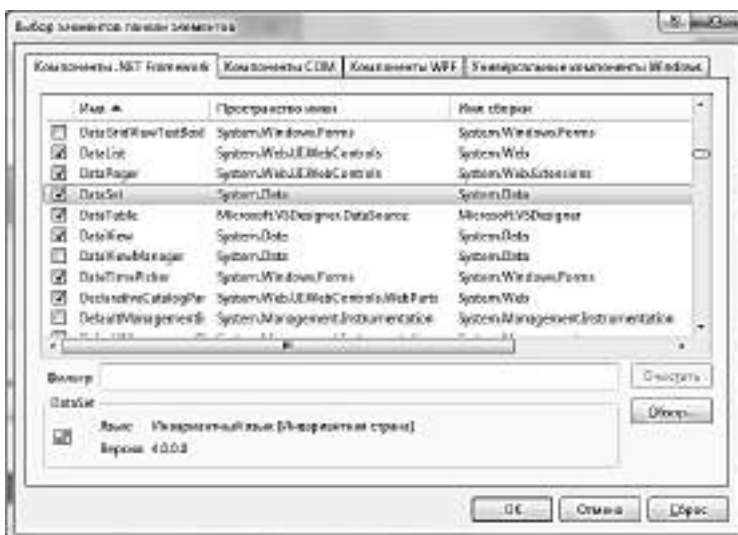
6.3.2. ADO komponentləri

ADO komponentləri ToolBox (Панель элементов) panelinin Data (Данные) səhifəsində yerləşir (şəkil 6.21). Bu panelə birinci dəfə müraciət etdikdə bir çox komponentlər görünür. Həmin komponentləri göstərmək üçün ToolBox (Панель элементов) panelinin Data (Данные) səhifəsində mausun sağ düyməsini basıb, kontekst menyunun Choose Items... (Выбрать элементы...) əmrini icra etmək lazımdır.

Bu zaman ekranda təsvir edilən pəncərədə (şəkil 6.22.) görmək istədiyiniz komponentləri qoşub (bayraqcığ qoymaqla) Ok düyməsini basmaq lazımdır.



Şəkil 6.21. Toolbox (Панель элементов) panelinin Data (Данные) səhifəsi



Şəkil 6.22. Data (Данные) səhifəsinə elementlərin əlavə edilməsi

6.3.2.1. dataSet verilənlər mənbəyi

ADO-nun əsas komponenti dataSet verilənlər mənbəyidir. Yuxarıda qeyd etdiyimiz kimi, dataSet obyektı ayrı-ayrı və bir-biri ilə əlaqəli cədvəllər və məhdudiyyətlərdən ibarət ola bilər. Bu obyekt və VB ilə işləmək üçün zəruri olan digər obyektlər System.Data adlar fəzasında təyin olunmuşdur.

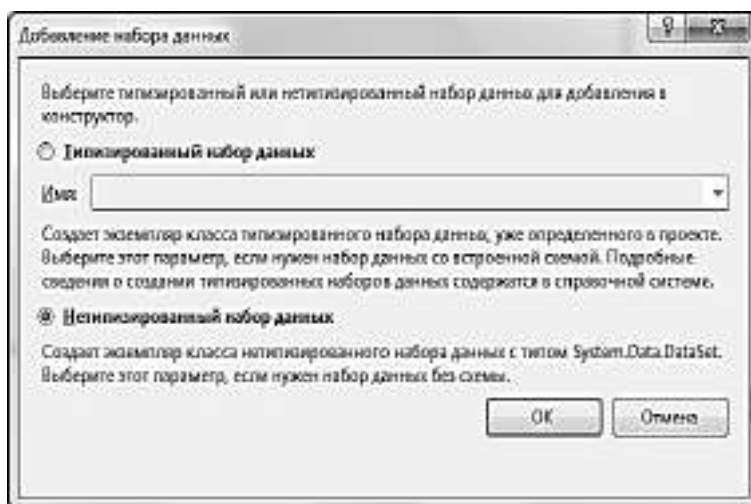
Bu komponenti praktiki olaraq öyrənmək üçün onun tətbiqi ilə əlavə yaradaq. Beləliklə, forma üzərinə dataSet komponenti yerləşdirək. Bu komponenti forma üzərinə atan kimi şəkil 6.23-də göstərilmiş dialoq pəncərəsi təsvir ediləcəkdir.

Bu pəncərədə tipləşdirilmiş və ya tipləşdirilməmiş verilənlər yığımını seçmək lazımdır. Bizim halda Untyped dataset (Нетипизированный набор данных) seçib Ok düyməsini

basırıq. Bunun nəticəsində layihəyə dataSet tipli dataSet1 komponenti əlavə ediləcək (dataSet komponenti vizual olmadığı üçün formanın altındakı boş sahədə yerləşir). Bu mərhələdə forma dizayneri faylına baxsaq, orada

```
private System.Data.DataSet dataSet1;
```

kodunu görə bilərik. Bu o deməkdir ki, dataSet obyektı sinfin parametri olmayan System.Data.DataSet konstrukturu ilə yaradılmışdır.



Şəkil 6.23. Verilənlər yığımının yaradılması

İndi dataSet komponentinin xassələrini öyrənək. Bunun üçün onu seçib Xassələr pəncərəsinin Properties (Свойства) səhifəsinə keçək.

DataSetName xassəsi komponentin adını bildirir. Susmaya görə bu xassəyə NewDataSet adı verilir, istəsəniz bu adı dəyişə bilərsiniz.

Locale xassəsi regional sazlama haqqında informasiyanı (dil, klaviaturla düzülüş, tarix formatı və s.) müəyyən edir.

Tables xassəsi cədvəllər yığımını, Relations isə cədvəllər arasında münasibəti müəyyən edir (əgər belə əlaqə yaradılmışdırsa).

Biz bu mərhələdə yuxarıda yaratdığımız Kontaktlar və Telefonlar cədvəllərini yenidən yarada bilərik. Bunun üçün Tables xassəsinin qarşısındakı üç nöqtə təsvirli düyməni basıb açılan pəncərədə Add (Добавить) düyməsini basmaqla cədvəl əlavə edirik. Sonra bu pəncərədə TableName xassəsinə Kontaktlar (və ya Telefonlar) adı veririk və cədvəl sahələrini yaradıırıq. Biz bu şərhə əlavə informasiya kimi nəzərinizə çatdırdıq, bizim yaratdığımız cədvəli yenidən yaratmaq fikrimiz yoxdur.

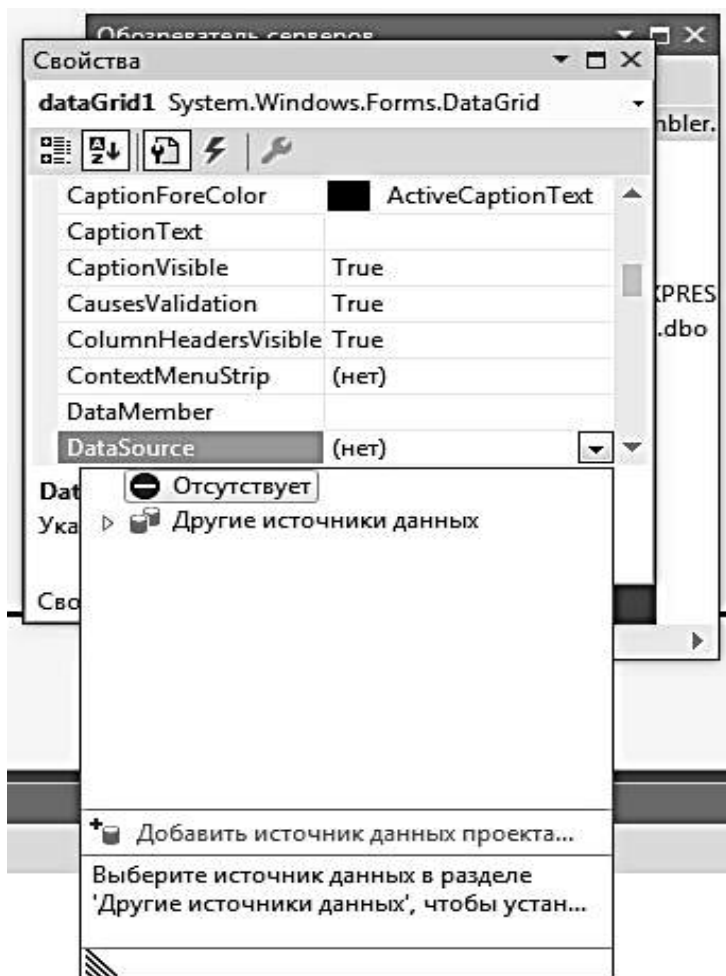
İndi isə həmin cədvəlləri formada əks etdirək. Bunun üçün dataGrid komponentindən istifadə edəcəyik.

6.3.2.2. dataGrid komponenti

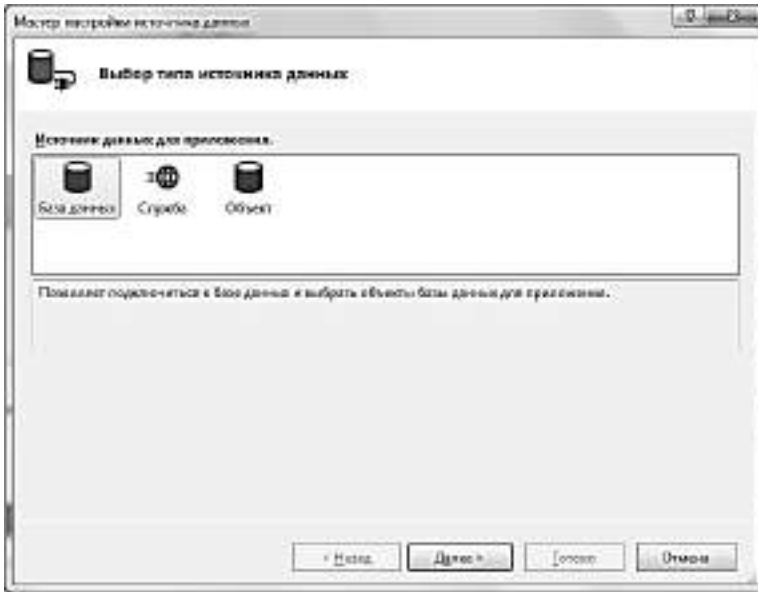
Verilənlər mənbəyindəki verilənləri forma üzərində təsvir etdirmək üçün dataGrid adlı çox güclü bir komponent nəzərdə tutulmuşdur. Bu komponenti də öyrənmək üçün onu forma üzərində yerləşdirək. O vizual komponentdir, ona görə də formada yerləşir. Bu komponentdə VB cədvəlləri təsvir edilir.

dataGrid komponentinin əsas xassəsi DataSource xassəsidir. Bu xassə vasitəsi ilə dataGrid komponenti dataSet1 cədvəlləri yığımına qoşulur. Beləliklə, dataGrid komponentini seçib, DataSource xassəsinin qarşısındakı siyahını açıırıq (şəkil 6.24). Açılan siyahıdan Add Project Data Source... (Добавить источник данных проекта...) əmrini icra edirik.

Bu zaman Ms Access verilənlər bazası ilə işlədikdə qarşılaşdığımız pəncərə açılacaqdır (şəkil 6.25).



Şəkil 6.24. Verilənlər mənbəyinin seçilməsi



Şəkil 6.25. Verilənlər mənbəyinin tipinin seçilməsi

Bu pəncərədə Database (База данных) seçib, Next> (Далее>) düyməsini basıb, yeni pəncərəyə keçdikdə yeganə element olan DataSet (Набор данных) seçib, yenidən Next> (Далее>) düyməsini basırıq. Yenidən əmələ gələn pəncərədə (şəkil 6.26) Which data connection should your application use to connect to the database? (Какое подключение ваше приложение должно использовать для работы с базой данных?) siyahısından

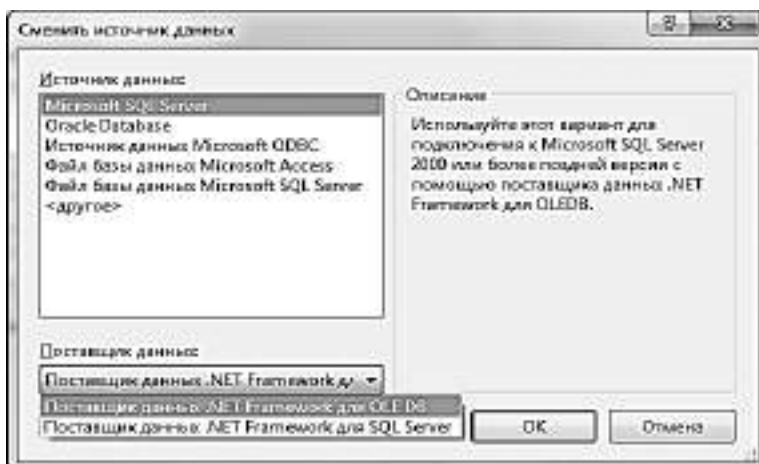
wincntrl-v34api8\sqlcxpress.TEL_KITAB.dbo
seçirik.

Əgər belə qoşulma olmazsa, onda New Connection... (Создать подключение...) düyməsini basaraq Add Connection (Добавить подключение) pəncərəsində (şəkil) Change... (Изменить...) düyməsini basıb, yeni əmələ gələn pəncərədə (şəkil 6.27) Data source: (Источник данных:) siyahısından Microsoft SQL Server, Data Provider

(Поставщик данных) siyahısından isə .NET Framework Data Provider for OLE DB (Поставщик данных .NET Framework для OLE DB) seçib Ok düyməsini basırıq.



Şəkil 6.26. Qoşulmanın seçilməsi



Şəkil 6.27. Verilənlər mənbəyinin seçilməsi

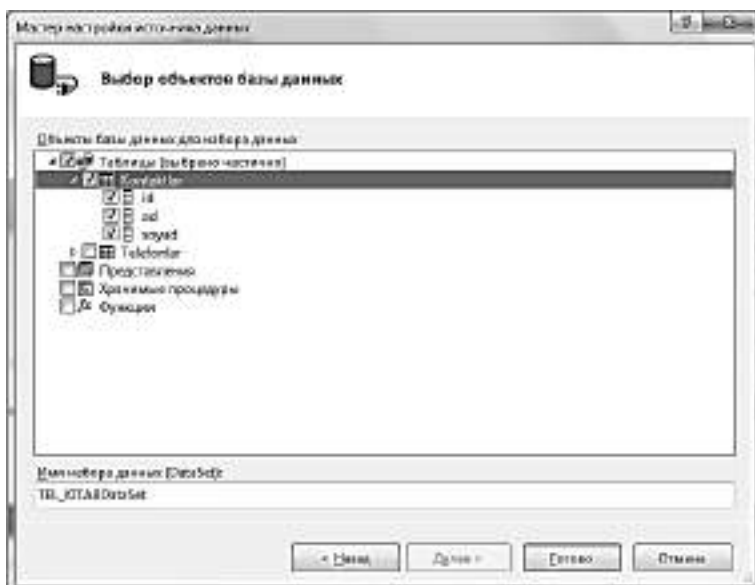
Növbəti mərhələdə açılacaq pəncərələr bizə artıq yaxşı tanışdır və yuxarıda öyrəndiyimiz qaydalarla zəruri parametrləri müəyyənləşdirmək lazımdır. Bu parametrlər müəyyən edildikdən sonra Data Source Configuration Wizard (Мастер настройки источника данных) pəncərəsində `TEL_KITABConnectionString` qoşulma sətri yaradılacaq (şəkil 6.28).



Şəkil 6.28. Qoşulma sətri

Next> (Далее>) düyməsini basıb yeni pəncərəyə keçdikdə bizə lazım olan Kontaktlar cədvəlini seçək və Finish (Готово) düyməsini basaq (şəkil 6.29.).

Bütün bu əməliyyatlar düzgün yerinə yetirilsə, formanın aşağı hissəsində daha üç komponent əmələ gələcəkdir: `tEL_KITABDataSet`, `kontaktlarBindingSource` və `kontaktlarTableAdapter`. Bununla yanaşı, Kontaktlar cədvəlinin sahələri `dataGrid` cədvəlində təsvir ediləcəkdir. Lakin bu mərhələdə cədvəl yalnız bir sətri təsvir edilir, verilənlər isə təsvir edilmir. Layihəni icra etdikdən sonra Kontaktlar cədvəli formada tam təsvir ediləcəkdir (şəkil 6.30).



Şəkil 6.29. Kontaktlar cədvəlinin seçilməsi

	id	ad	soyad
▶	1	Zakir	Məhərrəmov
	2	Rauf	Abdullayev
	3	AZAD	QURBANOV
	4	Əhməd	Veysov
	5	Şəbnəm	Məmmədova
	6	Dövlətxan	Balcanov

Şəkil 6.30. Kontaktlar cədvəlinin formada təsviri

Diqqət yetirin ki, dataGrid cədvəl komponenti üçün verilənlər mənbəyi, yəni, DataSource xassəsinin qiyməti kontaktlarBindingSource komponentidir. kontaktlarBindingSource komponentinin DataMember xassəsinin qiyməti isə Kontaktlar olmuşdur.

dataGrid komponentinin bəzi xassələri ilə tanış olaq. Onların əksəriyyəti bu komponentin xarici görünüşünü dəyişmək üçündür. Bu xassələr aşağıdakılardır:

- DataMember xassəsi DataSource verilənlər mənbəyinin alt siyahısında yerləşən cədvəlləri dataGrid komponentində təsvir etdirmək üçündür.

- BorderStyle xassəsi komponentin haşiyəsinin formasını idarə edir.

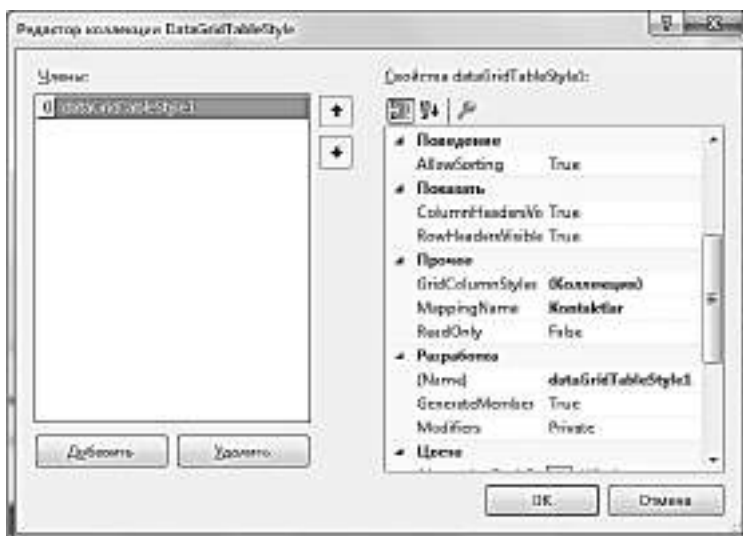
- CaptionText xassəsi cədvəlin başlığını müəyyən edir.

- CaptionFont xassəsi cədvəlin başlığının şriftini müəyyən edir.

- FlatMode xassəsinə True qiyməti verdikdə cədvəl müstəvi formasında təsvir edilir.

- TableStyle xassəsi cədvəlin müxtəlif tərtibat üslublarını təyin edir.

TableStyle xassəsinə qiymət vermək üçün onun qarşısındakı üç nöqtə təsvirli düyməni basıb açılan boş pəncərədə Add (Добавить) düyməsini basmaq lazımdır (şəkil 6.31). Bu zaman cədvəl dataGridTableStyle1 üslubu əlavə ediləcəkdir. Delete (Удалить) düyməsini basdıqda üslub silinir. dataGridTableStyle1 üslubu üçün MappingName xassəsinə cədvəlin adını – Kontaktlar yazın. Ümumiyyətlə, TableStyle xassəsi ilə cədvəl verilənlərini, xətlərini, cədvəlin başlıqlarını və s. müxtəlif qaydada tərtib etmək olar.



Şəkil 6.31. TableStyle xassəsinə qiymətlərin müəyyən edilməsi

6.3.2.3. sqlDataAdapter verilənlər adapteri

Verilənlər mənbəyi ilə dataSet obyektı arasında əlaqəni təmin edir. Bir tərəfdə verilənlər mənbəyi, digər tərəfdə dataSet dayanır. Verilənləri çıxarmaq və dataSet obyektini verilənlərlə doldurmaq SqlDataAdapter komponentinin əsas təyinatıdır.

Bütün bu funksiyaları yerinə yetirmək üçün onun iki əsas metodu vardır:

Fill metodu dataSet verilənlərini dəyişdirir. Fill metodu yerinə yetirildikdə dataAdapter obyektı DataTable və ya dataSet obyektini VB-dən alınmış verilənlərlə doldurur. Yaddaşa yüklənmiş verilənləri emal etdikdən sonra, modifikasiya edilmiş verilənlər Update metodu ilə VB-ə yazıla bilər.

Update metodu verilənlərə edilmiş dəyişiklikləri VB-də yadda saxlamaq üçün istifadə edilir.

Aşağıdakı xassələrlə (əmlrlərlə) SqlDataAdapter komponenti VB-nin dataSet obyektinə təsir edir:

– SelectCommand – sql sətirindən ibarətdir, VB-dən verilənlərin seçilməsini təmin edir. Fill metodunu çağırıqda bu əmr DataTable və ya dataSet obyektini doldurur;

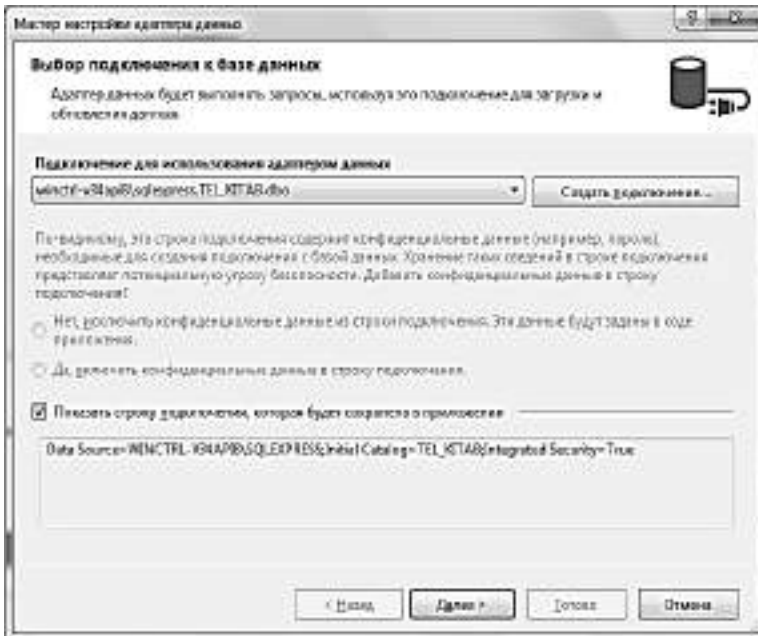
– InsetCommand – sql sətirindən ibarətdir, cədvələ sətir əlavə edir;

– DeleteCommand – sql sətirindən ibarətdir, cədvəldən sətiri silir;

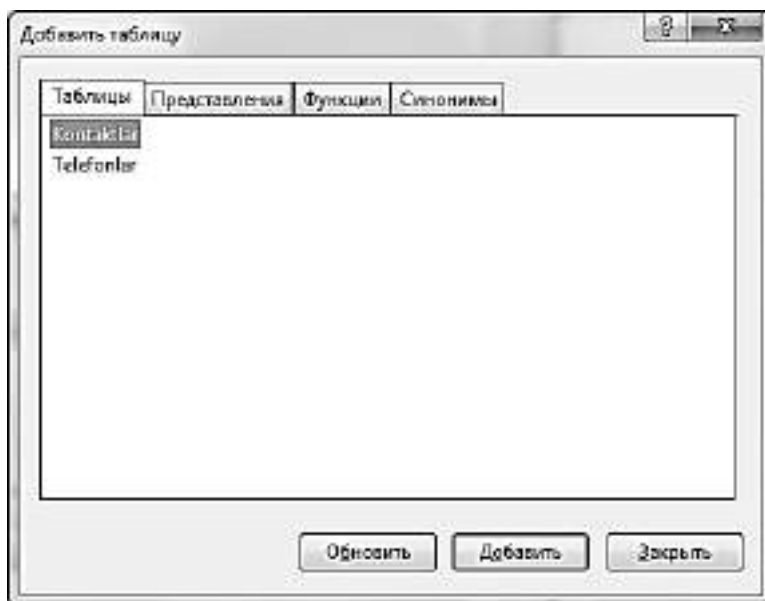
– UpdateCommand – sql sətirindən ibarətdir, VB-də verilənləri yeniləşdirir.

sqlDataAdapter komponentinin tətbiqi ilə C# əlavəsi layihələndirək.

Forma üzərinə sqlDataAdapter komponenti yerləşdirək. Ekranda dərhal şəkil 6.32-də göstərilmiş pəncərə əmələ gələcəkdir.

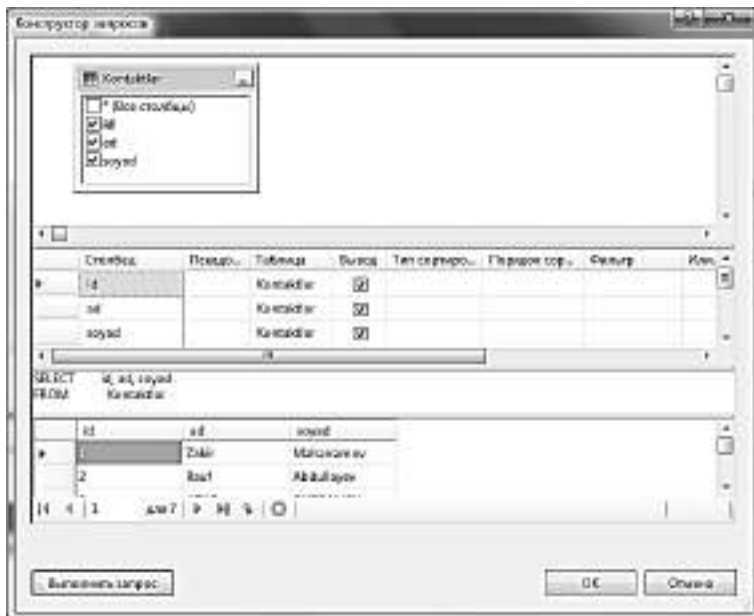


Şəkil 6.32. sqlDataAdapter komponentinin bazaya qoşulması



Şəkil 6.33. Bazanın cədvəlinin əlavə edilməsi

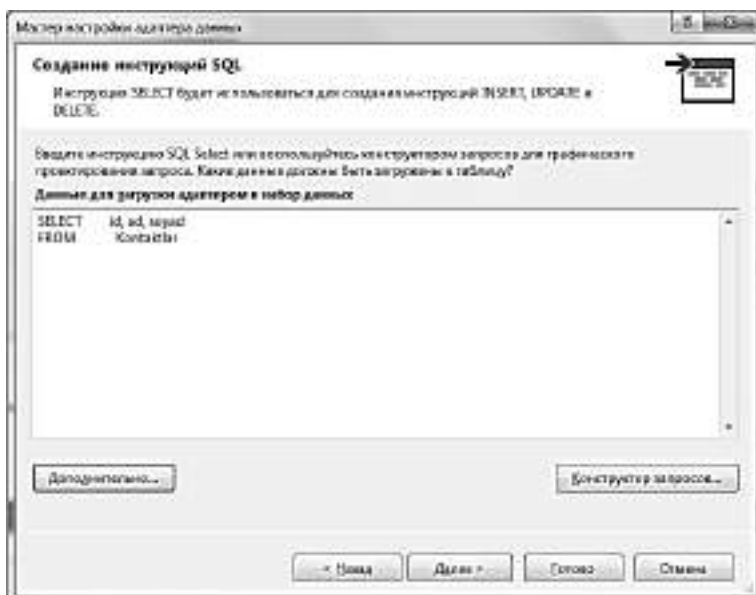
Bu pəncərədə Next> (Далее>) düyməsini basıb, növbəti keçəcəyimiz pəncərədə Use SQL statements (Использовать инструкции SQL) dəyişdiricisini qoşub, yenidən Next> (Далее>) düyməsini basırıq. Yeni pəncərədə Query Builder... (Конструктор запросов...) düyməsin basdıqda daha iki pəncərə əmələ gələcəkdir. Add table (Добавить таблицу) pəncərəsində (şəkil 6.33) Kontaktlar cədvəlini seçib, Add (Добавить) və Close (Закрыть) düymələrini basıb, Query Builder (Конструктор запросов) pəncərəsinə keçdikdə bu pəncərədə Kontaktlar cədvəli təsvir ediləcəkdir (şəkil 6.34).



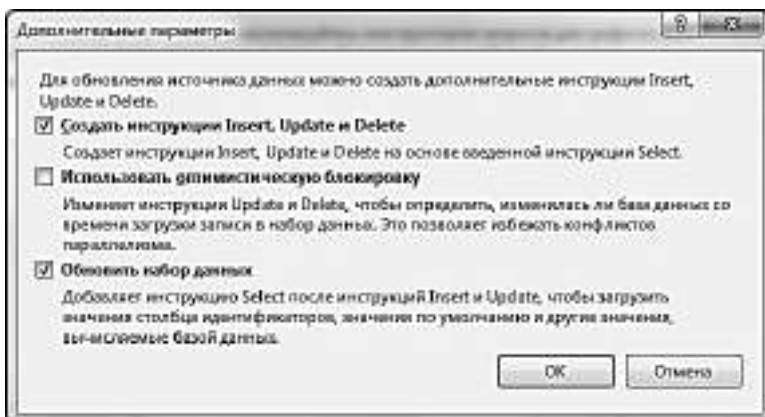
Şəkil 6.34. Sorğunun yaradılması

Cədvəlin bütün sahələrini seçib Execute Query (Выполнить запрос) düyməsini basdıqda cədvəlin verilənləri təsvir ediləcəkdir. Ok düyməsini basdıqda Data Adapter Configuration Wizard (Мастер настройки адаптера данных) pəncərəsinə qayıdacağıq (şəkil 6.35).

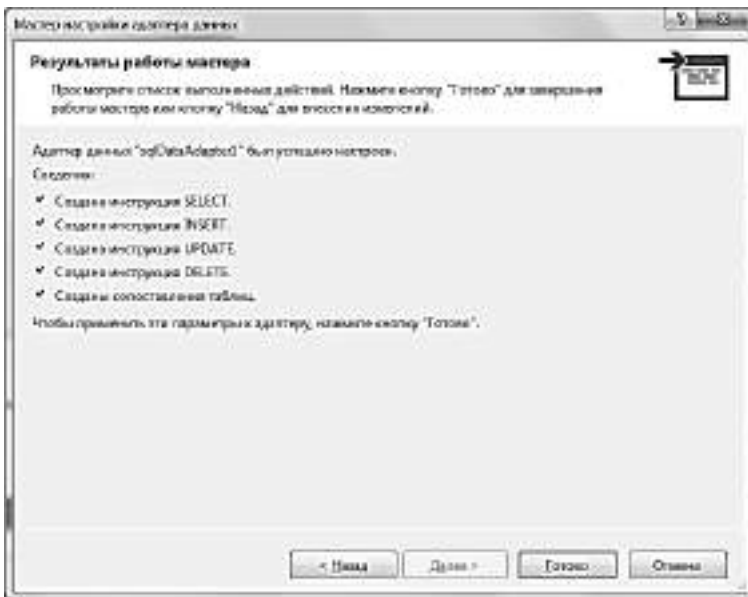
Həmin pəncərədə Advanced Options... (Дополнительно...) düyməsini basıb növbəti pəncərədə (şəkil 6.36) Generate Insert, Update and Delete statements (Создать инструкции Insert, Update и Delete) və Refresh the DataSet (Обновить набор данных) dəyişdiricilərini qoşuruq və Ok düyməsini basırıq. Geriyə qayıtdıqda Next> (Далее>) düyməsini basıb, növbəti açılan pəncərədə (şəkil 6.37) heç nəyi dəyişdirmədən Finish (Готово) düyməsini basırıq.



Şəkil 6.35. SQL təlimatlarının yaradılması



Şəkil 6.36. Sorğu parametrlərinin seçilməsi



Şəkil 6.37. Sazlamanın nəticəsi

İndi formadan aşağıda, avtomatik olaraq, yeni bir `sqlConnection1` adlı program komponenti əmələ gələcəkdir. `sqlConnection1` komponentinin `ConnectionString` xassəsinə birləşmə parametrləri yazılır:

```
Data Source=WINCTRL-V34API8\SQLEXPRESS;  
Initial Catalog=TEL_KITAB;Integrated  
Security=True
```

Bütün bu əməliyyatlardan sonra layihəyə aşağıdakı kodlar əlavə edilir:

```
private System.Data.SqlClient.SqlCommand  
    sqlSelectCommand1;  
  
private System.Data.SqlClient.SqlConnection  
    sqlConnection1;
```

```
private System.Data.SqlClient.SqlCommand  
    sqlInsertCommand1;  
  
private System.Data.SqlClient.SqlCommand  
    sqlUpdateCommand1;  
  
private System.Data.SqlClient.SqlCommand  
    sqlDeleteCommand1;  
  
private System.Data.SqlClient.SqlDataAdapter  
    sqlDataAdapter1;
```

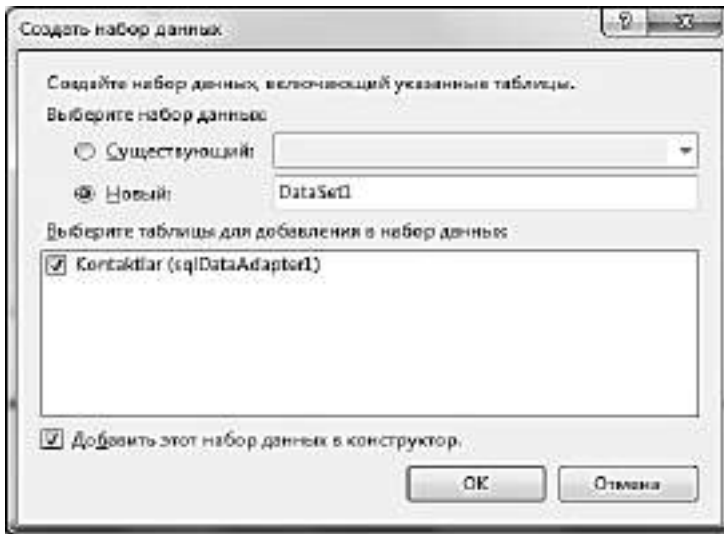
Bütün bu işlərin nəticəsində biz layihəyə sqlDataAdapter komponentini əlavə etdik və verilənlər mənbəyinə qoşulma yarardıq. İndi bizə dataSet lokal verilənlər mənbəyini yaratmaq lazımdır ki, o SQL serverində yerləşən bazada əks olunacaqdır.

Bunun üçün forma üzərində mausun düyməsini basıb, kontekst menyunun Generate DataSet... (Создать набор данных...) əmrini icra etmək lazımdır. Açılan eyniadlı pəncərədə (şəkil 6.38) New (Новый) və Add this dataset to the designer (Добавить этот набор данных в конструктор) dəyişdiricilərini qoşub, Ok düyməsini basmaq lazımdır.

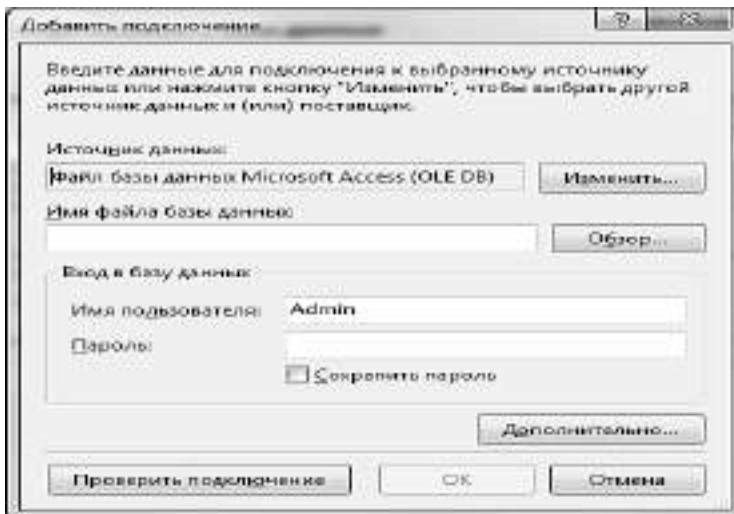
Bundan sonra formanın altında dataSet11 komponenti, Solution Explorer pəncərəsində isə dataSet1.xsd faylı əmələ gələcəkdir. Bu fayl cədvəlin XML dilində təsvirindən ibarətdir.

İndi dataSet verilənlər yığımını təsvir etdirmək üçün forma üzərinə dataGrid komponenti yerləşdirək. Bu komponenti verilənlər yığımı ilə əlaqələndirmək üçün onun DataSource xassəsinə qiymət verilməlidir. Həmin xassənin açılan siyahısından Add Project Data Source... (Добавить источник данных проекта...) əmrini icra etmək lazımdır. Açılan pəncərədən Database (База данных) seçib Next> (Далее>) düyməsini basırıq və yeni pəncərədə DataSet

(Набор данных) seçib yenidən Next> (Далее>) düyməsini basdıqda şəkil 6.39-da göstərilən pəncərə açılacaqdır.

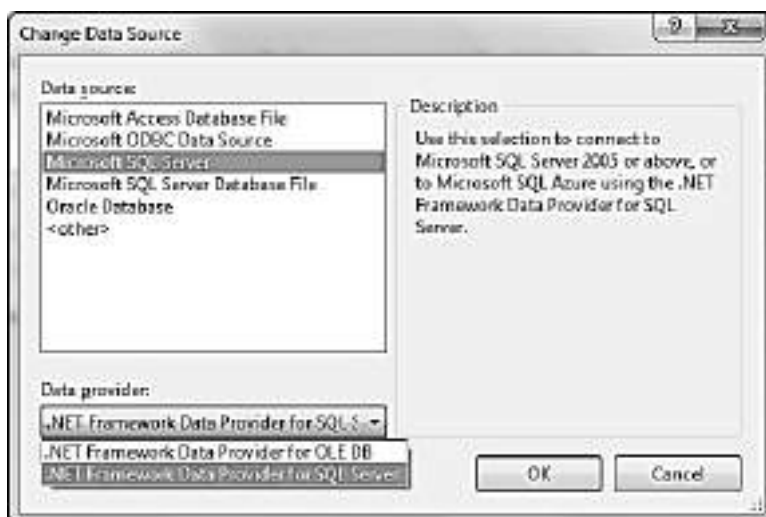


Şəkil 6.38. VY-nin yaradılması



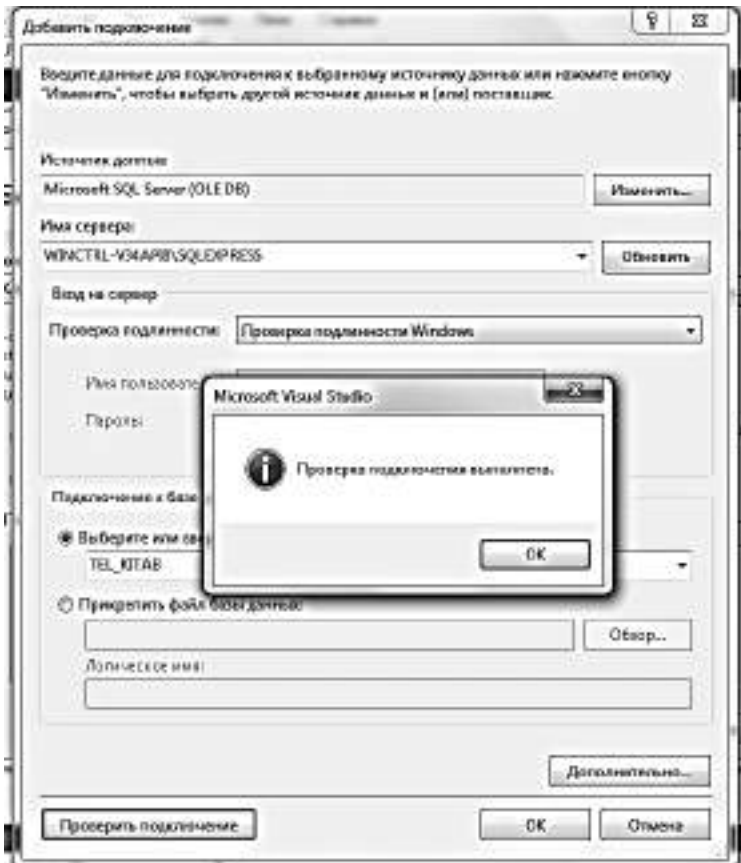
Şəkil 6.39. dataSet verilənlər yığımının parametrləri

Change... (Изменить...) düyməsini basdıqda açılan pəncərədə Data source: (Источник данных:) sahəsindən Microsoft SQL Server və Data provider: (Поставщик данных:) siyahısından .NET Framework Data Provider for SQL Server (Поставщик данных для .NET Framework SQL Server) seçmək lazımdır (şəkil 6.40).



Şəkil 6.40. Verilənlər mənbəyinin seçilməsi

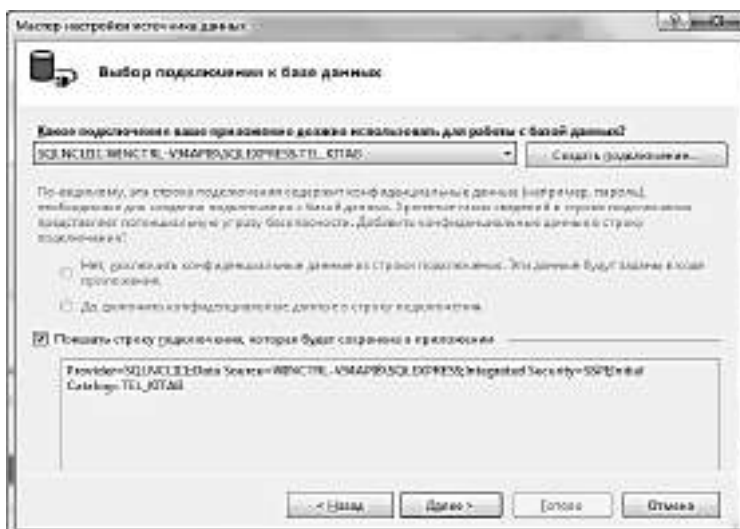
Ok düyməsini basıb əvvəlki pəncərəyə qayıtdıqda Server name: (Имя сервера:) sahəsinə serverin adını (WINCTRL-V34API8\SQLEXPRESS) daxil edib, Select or enter a database name: (Выберите или введите имя базы данных:) sahəsindən bazanın adını (TEL_KITAB) seçin və Test Connection (Проверить подключение) düyməsini basaraq bazaya qoşulmanı yoxlayın (şəkil 6.41).



Şəkil 6.41. Qoşulmanın yoxlanması

Ok düyməsini basdıqda şəkil 6.42-də göstərilən pəncərəyə qayıdış baş verəcək.

Bu pəncərədə və yenidən açılacaq pəncərədə Next> (Далее>) düyməsini basıb, növbəti açılan pəncərədə Kontaktlar cədvəlini seçib (şəkil 6.43), Finish (Готово) düyməsini basmaq lazımdır.

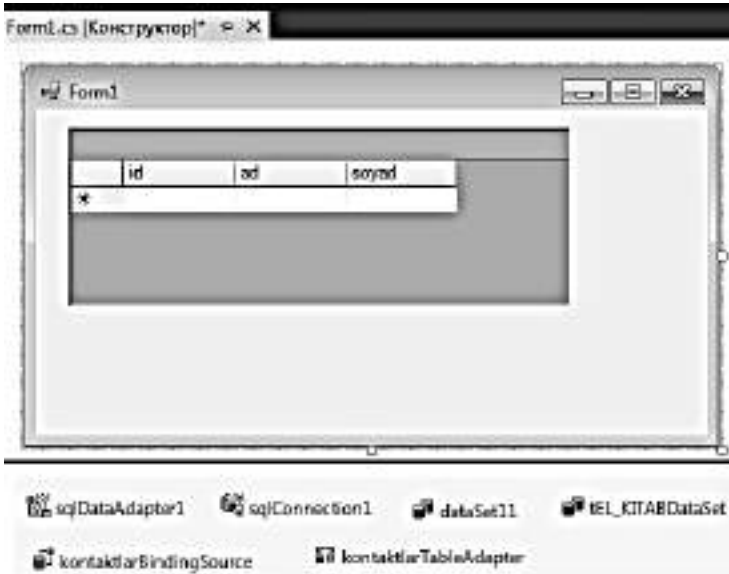


Şəkil 6.42. Qoşulmanın seçilməsi



Şəkil 6.43. Cədvəlin seçilməsi

Bu əməliyyatların icrasından sonra formanın altında `tEL_KITABDataSet1`, `kontaktlarBindingSource` və `kontaktlarTableAdapter` komponentləri peyda olacaq və `Kontaktlar` cədvəlinin sahələri `dataGrid` cədvəlində təsvir ediləcəkdir (şəkil 6.44).



Şəkil 6.44. Cədvəl `dataGrid` komponentində təsviri

Lakin, bu mərhələdə cədvəl yalnız bir sətiri təsvir edilir, verilənlər isə təsvir edilmir. Layihəni icra etdikdən sonra `Kontaktlar` cədvəli formada tam təsvir ediləcəkdir.

İndi `Kontaktlar` cədvəlinə verilənlərin yüklənməsi və serverdə VB-ə dəyişikliklərin edilməsinə baxaq.

Bunun üçün forma üzərinə iki `button` komponenti yükləyib, onların üzərində uyğun olaraq `Əlavə et` və `Yenilə` mətnləri yazaq. `Əlavə et` düyməsini basdıqda `Kontaktlar` cədvəlinə yeni verilənlər daxil edilməlidir. Bunun üçün `sqlDataAdapter` komponentinin `Fill` metodunu tətbiq edəcəyik. Bu məqsədlə aşağıdakı emaledicini yaradaq:

```
private void button1_Click(object sender,
                        EventArgs e)
{
    sqlDataAdapter1.Fill(dataSet11);
}
```

Cədvəl edilmiş əlavələrin serverdəki VB-də yadda saxlanması üçün isə sqlDataAdapter komponentinin Update metodunu tətbiq edəcəyik:

```
private void button2_Click(object sender,
                        EventArgs e)
{
    sqlDataAdapter1.Update(dataSet11);
}
```

Əlavənin nəticəsi şəkil 6.45-də göstərilmişdir.



	id	ad	soyad	
	6	Dövlətxan	Balcanov	
▶	7	Nurlan	Babayev	
	8	Cabir	Məmmədov	
	9	Hafiz	Rəcəbov	
*				

Form1

Əlavə et Yenilə

Şəkil 6.45. Əlavənin nəticəsi

6.3.2.4. bindingSource komponenti

Bizi maraqlandıran növbəti komponent bindingSource komponentidir. Bu idarəetmə elementi layihəyə daxil olan bütün idarəetmə elementləri ilə, və müvafiq olaraq, verilənlərlə işləmək üçün elementlərlə əlaqə yaradır. bindingSource komponenti layihədə yazılması tələb olunan hadisə emaleddicilərinin sayını və kodun həcmi kifayət qədər azaltmağa imkan verir.

bindingSource komponentinin əsas xassələri DataSource və DataMember xassələridir. DataSource xassəsinin açılan siyahısı əlçatan verilənlər mənbəyindən ibarətdir. DataMember xassəsində dataBinding komponentində istifadə ediləcək cədvəlin adı göstərilir. Bu xassə vasitəsi ilə xarici açarın müəyyən edildiyi münasibət verilə bilər ki, bu da onunla əlaqələndirilmiş verilənləri görməyə imkan verir.

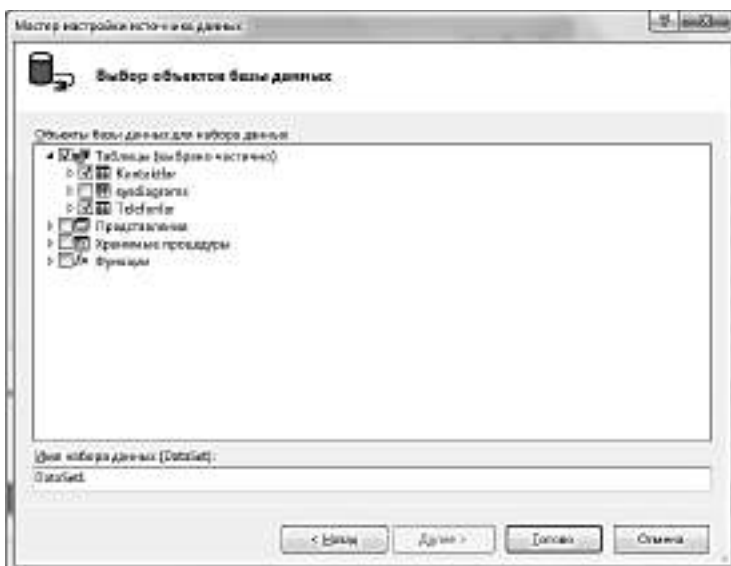
bindingSource komponenti ilə yanaşı, adətən, bindingNavigator komponenti də istifadə edilir. Bu komponent vizual olmaqla yanaşı, həmişə formanın yuxarı hissəsinə bərkidilir. Mahiyyət etibarı ilə bindingNavigator komponenti özünü düymə və digər elementlərdən ibarət alətlər paneli kimi aparır. Onu birinci dəfə forma üzərinə yerləşdirdikdə, o, verilənlərlə işləmək üçün standart funksional imkanlara malik olan bir sıra düymələrdən, məsələn, birinci və ya sonuncu elementə keçid, həmçinin elementin əlavə edilməsi, silinməsi, saxlanması kimi düymələrdən ibarət olur. bindingNavigator komponenti nəinki, bu elementləri yaradır, həm də onları bir-biri ilə əlaqələndirir.

İndi C# əlavəsi layihələndirib, hər iki komponenti tətbiq edək.

Layihənin məqsədi əlaqəli Kontaktlar və Telefonlar cədvəlinin forma üzərində təsvirindən ibarət olacaqdır.

Beləliklə, forma üzərinə bindingSource komponenti yerləşdirək. O, vizual komponent olmadığı üçün formanın alt hissəsində yerləşəcək. Onu verilənlər mənbəyi ilə əlaqələndirək. Bu məqsədlə onun DataSource xassəsinin siyahısını açıb, Add Project Data Source... (Добавить источник данных проекта...) əmrini icra edək. Bu əmrdən sonra bizə çox yaxşı

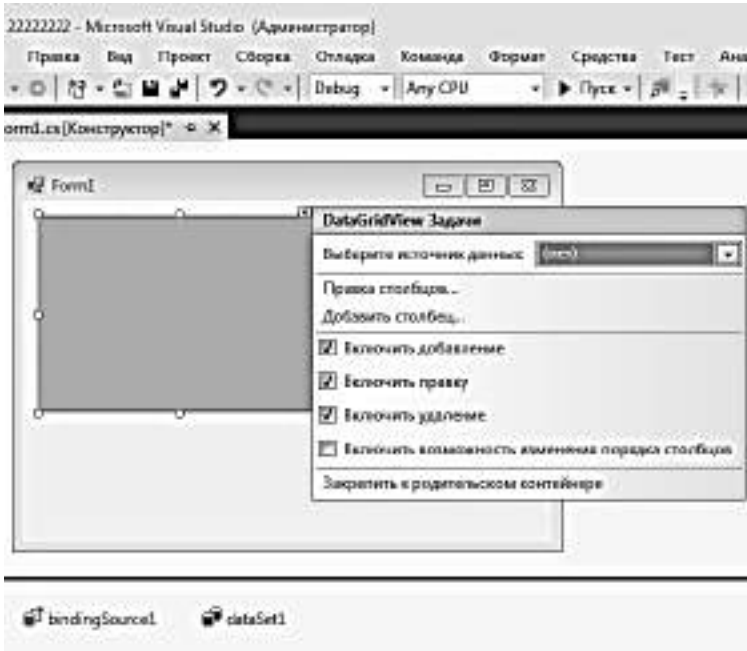
tanış olan pəncərələr ardıcılığında artıq bildiyimiz əməliyyatları təkrar etmək lazımdır. Yeganə fərq ondan ibarət olacaqdır ki, Data Source Configuration Wizard (Мастер настройки источника данных) pəncərəsində Kontaktlar və Telefonlar cədvəlinin hər ikisini seçmək lazımdır (şəkil 6.46).



Şəkil 6.46. Kontaktlar və Telefonlar cədvəlinin hər ikisini seçilməsi

Finish (Готово) düyməsini basdıqda bindingSource komponentinin yanında dataSet1 komponenti təsvir olunacaqdır.

İndi əlaqələndirdiyimiz verilənləri formada təsvir edək. Bunun üçün forma üzərinə Data (Данные) elementlər qrupundan dataGridView komponenti yerləşdirək. Bu komponent verilənləri cədvəl şəklində təsvir etdirir. Bu vizual komponentdir, forma üzərində şəkil 6.47-də göstərildiyi kimi təsvir olunur.



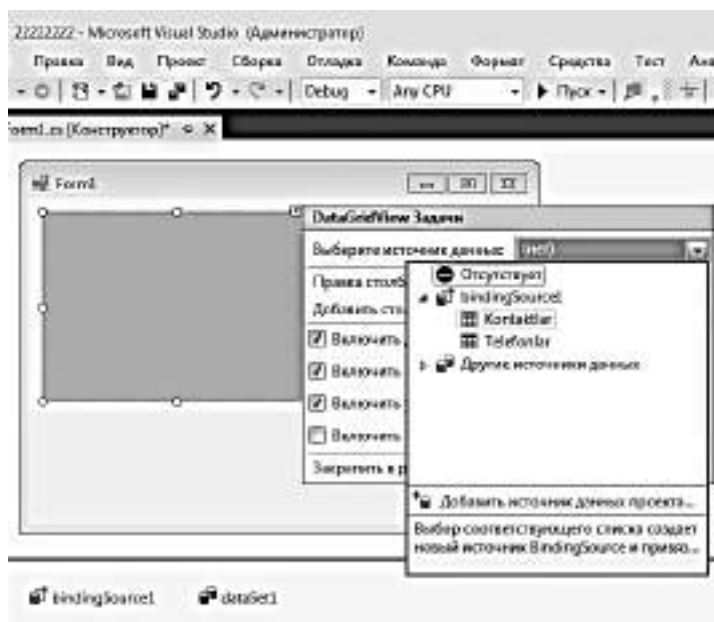
Şəkil 6.47. dataGridView komponenti

Komponenti forma üzərinə atan kim, ekranda dataGridView TaskBar (DataGridView Задачи) menyusu görünür. Bu menyuda onun aşağıdakı əsas menyuları təsvir edilmişdir:

- Enable Adding (Включить редактирование),
- Enable Editing (Включить правку),
- Enable Deleting (Включить удаление);
- Enable Column Reordering (Включить возможность изменения порядка столбцов);
- Dock in Parent Container (Закрепить в родительском контейнере).

Komponentin verilənləri təsvir etdirməsi üçün Choose Data Source: (Выберите источник данных:)

siyahısını açıb, oradan verilənlər mənbəyini seçirik (şəkil 6.48). Bunun üçün `bindingSource1` əlaqəsinin siyahısını açıb Kontaktlar cədvəlini seçirik.



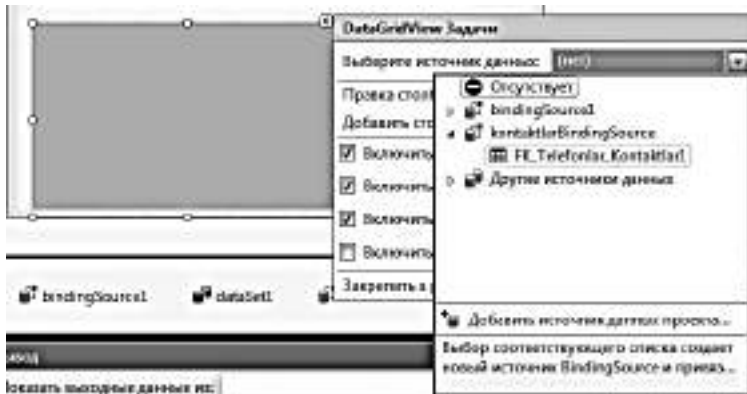
Şəkil 6.48. `dataGridView` komponenti üçün verilənlər mənbəyinin seçilməsi

Belə seçmədən sonra `dataGridView` komponenti boş xanalardan ibarət cədvəlin bir sətirini (strukturunu) təsvir etdirir (şəkil 6.49).

İndi Telefonlar əlaqəli cədvəlinin verilənlərini təsvir etdirmək lazımdır. Bunun üçün formaya ikinci `dataGridView` komponenti yerləşdirək. `dataGridView` komponentinin verilənlər mənbəyini seçək (şəkil 6.50).



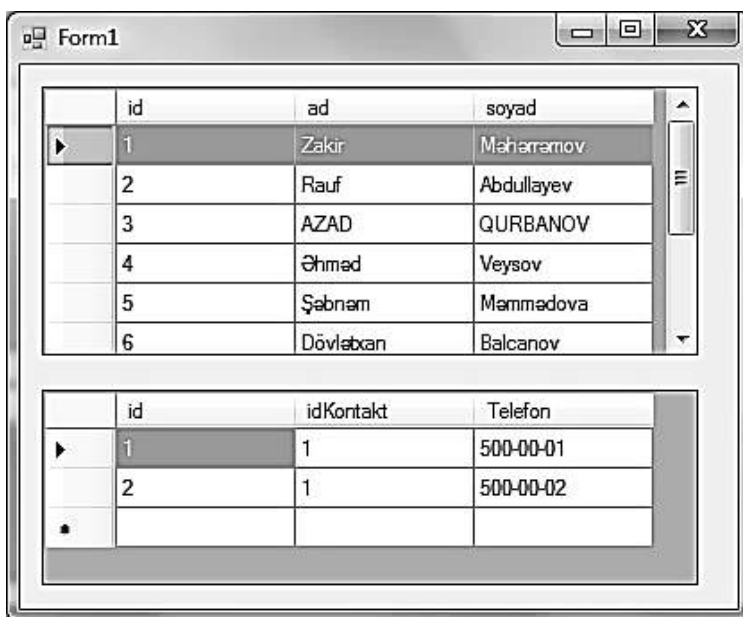
Şəkil 6.49. dataGridView komponenti boş xanalardan ibarət cədvəlin bir sətrini (strukturunu) təsvir etdirir



Şəkil 6.50. dataGridView komponentinin verilənlər mənbəyinin seçilməsi

Bu dəfə Telefonlar cədvəlinin özünü yox, kontaktlarBindingSource əlaqəsinin siyahısını açıb FK_Telefonlar_Kontaktlar1 əlaqəsini seçirik. Belə seçim verilənlərin düzgün yenilənməsinə və silinməsinə zəmanət verir.

Layihəni işə saldıqdan sonra əlavənin nəticəsi şəkil 6.51-də göstərilmişdir.



	id	ad	soyad
▶	1	Zakir	Mahərrəmov
	2	Rauf	Abdullayev
	3	AZAD	QURBANOV
	4	Əhməd	Veysoy
	5	Şəbnəm	Məmmədova
	6	Dövlətxan	Balcanov

	id	idKontakt	Telefon
▶	1	1	500-00-01
	2	1	500-00-02
■			

Şəkil 6.51. Hazır əlavə

Cədvəl verilənləri üzərində cədvəl kursoru və ya idarəetmə klavişləri ilə yerdəyişmə narahatçılıq yaradır. Bunu asanlaşdırmaq üçün formaya `bindingNavigator` komponenti yerləşdirək. O, formanın yuxarı hissəsində yerləşəcəkdir. Bu komponent yazılar arasında yerdəyişmələri asanlaşdırmaqla yanaşı, cədvələ sətirlər əlavə etməyə və ya onları silməyə imkan verir. Komponentin imkanlarını və ya xarici görünüşünü sazlamaq olar, belə ki, o, `toolStripContainer` menyu zolağından ibarətdir.

Bu komponentin əsas xassəsi `BindingSource` xassəsidir. Bu xassəyə qiymət vermək üçün `bindingNavigator` komponentini seçib, `BindingSource` xassəsinin açılan siyahısından `kontaktlarBindingSource` əlaqəsini seçirik. Layihəni icra etdikdə şəkil 6.52-də göstərilmiş nəticə alınacaqdır.

Form1

5 для 9

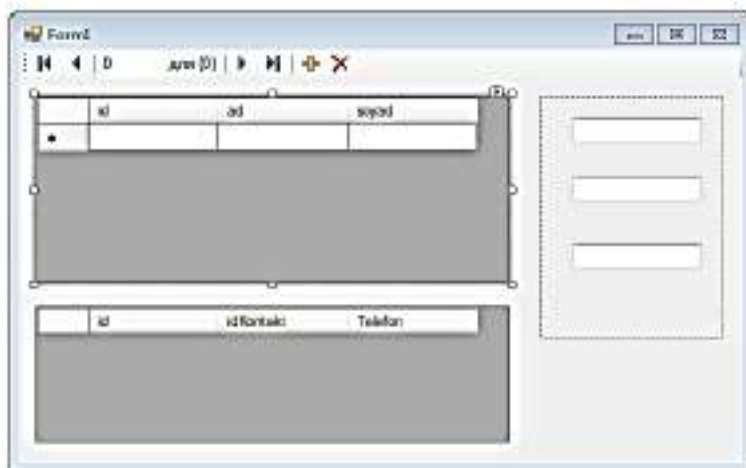
	id	ad	soyad
	1	Zakir	Məhərrəmov
	2	Rauf	Abdullayev
	3	AZAD	QURBANOV
	4	Əhməd	Veysoy
▶	5	Şəbnəm	Məmmədova
	6	Dövlətxan	Balcanov

	id	idKontakt	Telefon
▶	5	5	500-00-05
•			

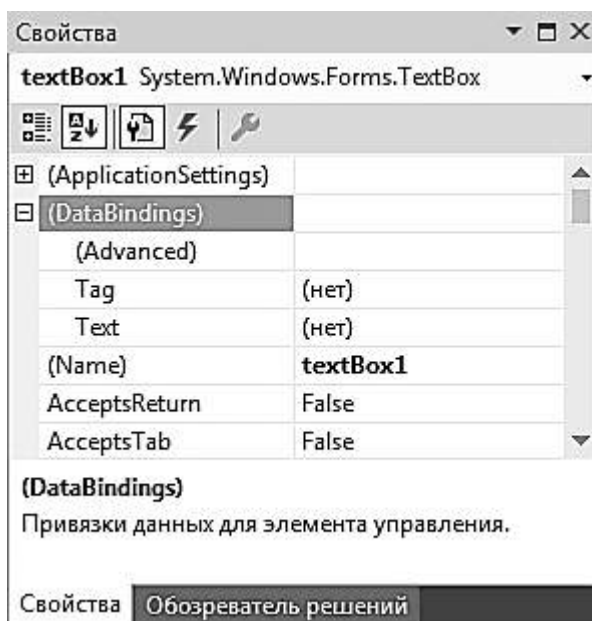
Şəkil 6.52. bindingNavigator komponentinin əlavə edilməsi

Cədvəlin xanalarını mətn redaktorlarında da təsvir etdirmək olar. Bunun üçün panel komponenti, onun üzərinə isə üç ədəd textBox komponentlərini forma üzərinə şəkil 6.53-də göstərildiyi kimi yerləşdirək.

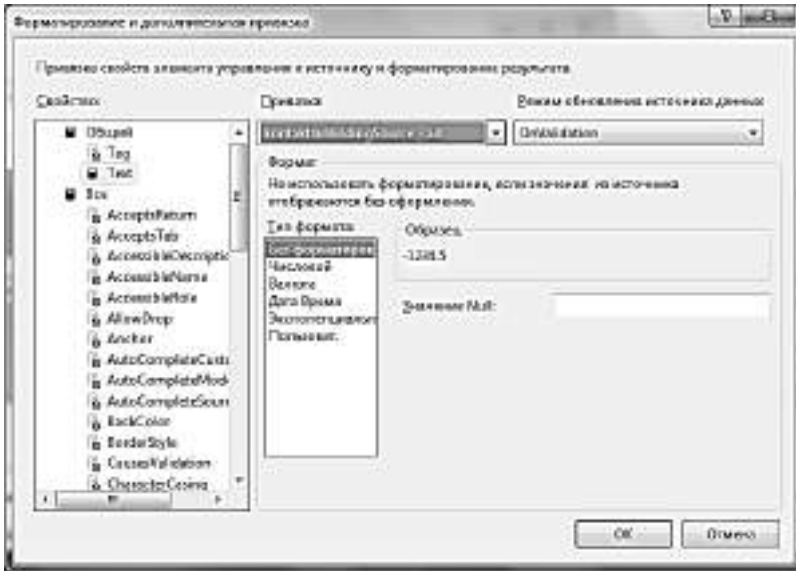
İndi textBox komponentlərini cədvəl sahələri ilə əlaqələndirək. textBox1 komponentini seçib, xassələr pəncərəsində DataBindings xassələr redaktorunu açaraq Advanced alt xassəsinin qarşısındakı üç nöqtə təsvirli düyməni basaq (şəkil 6.54). Bu zaman şəkil 6.55-də göstərilmiş Formatting and Advanced Didding (Форматирования и дополнительная привязка) dialog pəncərəsi əmələ gələcəkdir.

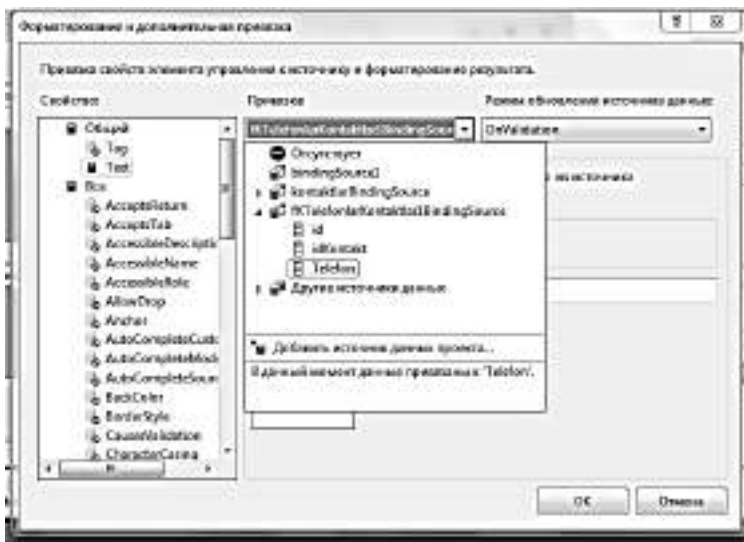


Şəkil 6.53. textBox komponentlərini forma üzərinə əlavə edilməsi

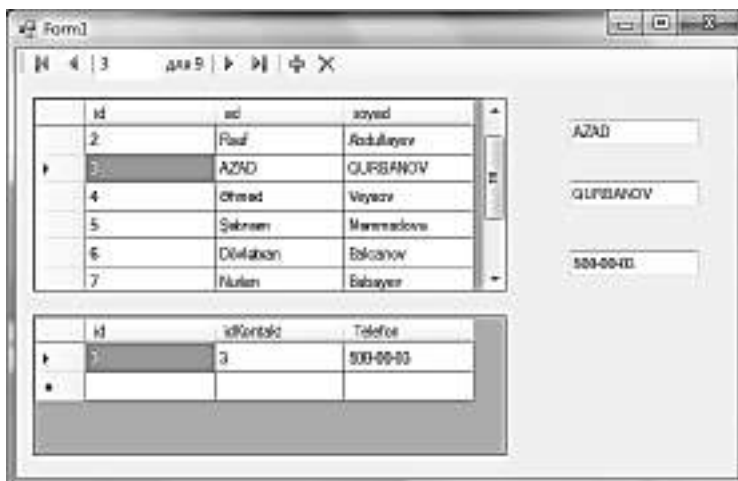


Şəkil 6.54. DataBindings xassələr redaktoru





Şəkil 6.56. textBox3 komponentinin Telefonlar cədvəlinin Telefon sahəsi ilə əlaqələndirilməsi



Şəkil 6.57. Yekun əlavə

ƏDƏBİYYAT

1. Məhərrəmov Z.T. Verilənlər bazası. Ali məktəb tələbələri üçün dərs vəsaiti. – Bakı: 2015. – 436 s.
<https://www.kitabyurdu.org/kitab/it/2138-verilenler-bazasi-delphi.html>
2. Məhərrəmov Z.T. Pascal–dan Delphi–yə. Ali məktəb tələbələri üçün dərs vəsaiti. – Bakı: “Təhsil” NPM, 2014. – 412 s.
<https://www.kitabyurdu.org/kitab/it/2139-pascal-dan-delphi-ye.html>
3. Zakir Məhərrəmov, Zəfər Cəfərov. Visual C#. LAP LAMBERT Academic Publishing, 2019. -472 pagine.
<https://www.amazon.it/Visual-C-Zakir-M%C9%99h%C9%99r%C9%99mov/dp/6139482925>
4. Фленов М.Е. Библия Delphi.– СПб.: БХВ-Петербург, 2008.–800 с.
5. Фленов М. Е. Библия C#. 3- е изд., перераб. и доп. - СПб.: БХВ - Петербург, 2016. -544 с.
6. Ахмадулин Р.К. Пример разработки приложений для баз данных. Часть 1. Тюмень: ТюмГНГУ, 2015. – 22 с.
7. Евсеева О. Н., Шамшев А. Б. Работа с базами данных на языке C#. Технология ADO .NET. Учебное пособие. Ульяновск: УлГТУ, 2009. – 170 с.
8. Осетрова И.С., Разработка баз данных в MS SQL Server 2014. - СПб: Университет ИТМО, 2016. – 114 с.
9. Александр Бондарь. Microsoft SQL Server 2014. — СПб.: БХВ-Петербург, 2015. — 592 с.

Çapa imzalanmışdır: 04.06.2019

Format: 60-84 1/16. Həcmi: 14.5, Tirajı: 200 nüsxə

“Elm” nəşriyyatının mətbəəsində çap edilmişdir.

Ünvan: Bakı ş., İstiqlaliyyət küç., 28