

***DƏRS VƏSAİTİ***

## Mündəricat

[Cədvəllərin yaradılması və pozulması 2](#_bookmark0)

[SELECT – sadə şərtlər 5](#_bookmark1)

[SELECT – mürəkəb şərtlər 7](#_bookmark2)

[SELECT - müqayisə, məntiqi və xüsusi operatorlar 10](#_bookmark3)

[IN, BETWEEN, LIKE, IS NULL 13](#_bookmark4)

[Aqreqat funksiyalar 14](#_bookmark5)

[Bir necə cədvəldən verilənlərin seçilməsi 17](#_bookmark6)

[Altsorğular 20](#_bookmark7)

[Bağlanmış altsorğular 22](#_bookmark8)

[EXISTS operatoru 25](#_bookmark9)

[ANY, ALL operatorları 28](#_bookmark10)

[UNION əmri 29](#_bookmark11)

[INSERT - sətirlərin daxil olunması 32](#_bookmark12)

[DELETE - sətirlərin cədvəldən silinməsi 34](#_bookmark13)

[UPDATE - sətirin qiymətlərinin dəyişməsi 34](#_bookmark14)

# Cədvəllərin yaradılması və pozulması

SQL Serverin VB-sinin bütün məlumatları cədvəllərdə qorunur. Cədvəllər eyni tipi birləşdirən sütunlardan və cədvəl yazılarını saxlayan sətirlərdən ibarətdir. Hər bir bazada 2 milyarda yaxın cədvəl ola bilər, cədvəldə 1024 sütun, sətirdə isə 8060 bayt məlumat ola bilər.

SQL Server növbəti tip verilənləri saxlayır.

|  |  |  |
| --- | --- | --- |
| Verilənlərin tipi | Mənası | Ölçü, baytla |
| **Binar verilənlər** | binary varbinary[(n)] | 1-8000 |
| **Simvollar** | char[(n)]varchar[(n)] | 1-8000  (8000-ə qədər) |
| **Unicode Simvollar** | nchar[(n)]  nvarchar[(n)] | 1-8000  (4000 simvol) |
| **Vaxt və tarix** | datetime smalldatetime | 8  4 |
| **Dəqiq qiymətlər** | decimal[(p[,s])]  numeric[(p[,s])] | 5-17 |
| **Təqribi qiymətlər** | float[(n)] real | 4-8  4 |
| **Qlobal dəyişənlər** | uniqueidentifier | 16 |
| **Tam tiplər** | int smallint, tinyint | 4  2, 1 |
| **Pullar !!!** | money, smallmoney | 8, 4 |
| **Xüsusi** | bit, cursor, sysname, timestamp | 1, 0-8 |
| **Mətn və şəkil** | text, image | 0-2 Qb |
| **Unicode mətn** | ntext | 0-2 Qb |

Cədvəlləri **Transact-SQL** dilinin operatoru olan **CREATE TABLE** -ın köməkliyi ilə yaradırlar, həmçinin **Enterprise Manager** köməkliyi ilə ə yaratmaq olar. Əvvəlcə bunun **Transact-SQL** köməkliyi ilə necə yaranmasına baxaq.

Cədvəllərin CREATE TABLE-ın köməkliyi ilə yaradılması.

Cədvəllərin yaradılması üçün **CREATE TABLE** operatoru istifadə olunur. Bu operatorun sintaksisi belə təyin olunur.

*CREATE TABLE table\_name (column\_name data\_type [NULL | NOT NULL]*

*[,...n])*

Nümunə:

CREATE TABLE member (

member\_no int NOT NULL, lastname char(50) NOT NULL, firstname char(50) NOT NULL, photo image NULL

)

Bu operatorun köməkliyi ilə dörd sütundan ibarət olan **members** cədvəli yaradılır.

* **member\_no** - int tipindədir, NULL qiymıti buraxılmır
* **lastname** - char(50) tipindədir - 50 simvoldur, NULL buraxılmır
* **firstname** - lastname oxşardır
* **photo** - image(şəkil) tipindəndir, NULL qiyməti buraxılır

**Qeyd. NULL** - elementin heç bir qiymət almadığını göstərən dəyişəndir. Sütuna bu tipi mənimsədəndə qeyd olunur ki bu elementlər inisial ola bilməzlər. NOT NULL

* yazılanda "boş" qiymətlərə icazə verilmir. Əgər yazı daxil olunarkən bu cür sütun üçün xana boş olarsa onda daxil olma olmayacaq, və SQL Server səhv haqqında məlumat verəcək.

Bu əmri realizə etməyə çalışın. **Query Analyzer**-i işə salın. Öz serveriniz ilə qoşulun. VB siyahısından **sqlStep** cədvəlini seçin. Əmr pəncərəsinə cədvəli yaratmaq üçün istifadə olunan əmri yazın və onu realizə edin. (Əgər yaddan çıxmibsa **F5**və ya **Ctrl-E**-ni sıxmaq lazımdır).

Cədvəlin yarandığına dəqiq əmin olmaq üçün aşağıdakı əmri yığın: sp\_help member

Onu sadə redaktorda yığın və sonda **F5** düyməsini sıxın. Nəticə pəncərəsinə **members** cədvəli barədə məlumat çıxacaq.

### Məlumat üçün!!!

**sp\_help** - VB-nin verilənləri haqqında məlumatları verən (cədvəllər, saxlanmış proseduralar və s.) sistem funksiyadır.

Çağırma funksiyası belədir:

*sp\_help <cədvəlin adı>*

Cadvəli pozmaq çox asandır. Elə orda, sorğuçuda (bizdə **Query Analyzer**-i belə adlandırırlar) aşağıdakını yığın:

*drop table member*

Bu sətiri seçin və **F5**-i sıxın. Cədvəl pozulacaq və bunun haqda sizə məlumat veriləcək. Bizim halda bu prosedura sadədir. Əslində isə ri VB-də cədvəli silmək o qədər də asan deyil, o digər cədvəllərlə bağlı ola bilər və əvvəlcə həmin əlaqələri silmək lazımdır. Bunun necə edilməsini öyrənmək üçün növbəti addımlara bax.

SQL Server Enterprise Manaher vastəsi ilə cədvəli necə yaratmaq olar Ardıcıllıqla açın: **SQL Server Group**, <sizin server>, **Databases**. VB-ni seçin

(mən fikirləşirəm ki **SqlStepByStep**), sağ düyməni sıxın və açılmış menyudan "**New**" bölməsini, sonra isə "**Table...**" bölməsini seçin. İlk növbədə sizdən cədvəlin adını soruşacaq. Onu daxil edin və Enter düyməsini sıxın. Ekranda pəncərə çıxacaq və orda: sütunların adlarını, tipini, uzunluğunu, ölçüsünü, dəqiqliyini (bu üç sütun tipdən aslı olaraq bloklanır), NULL icazə verən bayraq, susmaya görə qiymət. Son üç sütun hələlik maraq mərkəsində deyil.

Sütunların adlarını, tipini və uzunluğunu yuxarıdakı nümunədəki kimi göstərin. Disket işarəsi olan ikonkaya sıxın və yadda saxlanandan sonra pəncərəni bağlaya bilərsən. Öz Bazanızı yenidən açın və "**Tables**" hissəsində öz bazanızı yenıcə yaratdığınız cədvəli cədvəllər hissəsində ğörəcəksiniz. Onun pozulması üçün onu seçin, mausun sağ düyməsini sıxın və açılan menyuda "**Delete**" hissəsini sıxın. Cədvəl pozulacaq.

Biz cədvəlin yaranıb pozulmasını öyrəndik. Cox funksiyalı, nümunədə öyrənəcəyimiz VB-nin yaradılmasına aid olan növbəti funksiyaya keçək.

* + VB-də qiymətli məlumatlar nə deməkdir və onlar **SQL Server** vastəsi ilə necə realizə olunurlar.
  + Cədvəllərdə məlumatları mecə yeniləmək olar (**INSERT, UPDATE, DELETE, SELECT** operatorları).
  + yadda saxlanmış triqqer və proseduraları necə istifadə etməli.

# SELECT – sadə şərtlər

Nəzəriyyədən praktikaya keçmək vaxtı gəlib çatdı. SQL Serverin gələcəkdə də öyrənilməsi üçün balaca bir VB yaradaq. VB-nin yaradılması üçün proektdən bu addım üçün olan **SQLByStep.sql** faylını yükləmək lazımdır ([yüklə](http://www.firststeps.az/derslikler/sql/sqlbystep.zip)).

Bu skript **SQLByStep** VB-sinin yaranması üçün özündə **Transact SQL** əmrlərini saxlayır. Burada üç cədvəl və onların içində məlumatlar var. VB-ni yaradılması üçün siz aşağıdakıları etməlisiniz:

* + 1. Diskdə boş yer tapmalı - VB təqribi 2Mb tələb edir.
    2. VB-nin yerləşməsi üçün qovluq yaratmalı - susmaya görə **C:\SQLByStep\data** istifadə olunur. Əgər başqa yol seçmək istəyirsinizsə onda faylda dəyişiklik edin (**CREATE DATABASE** əmrində faylın və loqun verilənlərini dəyişin).
    3. **Query Analyzer**-i işə salın
    4. **File->Open** menyusunu seçin və SQLByStep.sql faylını yükləyin.
    5. Onu realizə edin (**F5**-i sıxın).

OK, VB yaradıldı və məlumatlar ona yerləşdirildi. Bundan sonra müxtəlif sualları realizə etmək üçün **Query Analyzer**-dən istifadə olunacaq. **Query Analyzer** - əsl proqramistlərin alətidir :-) VB özendə satışa nəzarət etmək üçün SQL- i istifadə edən fraqmenti taşkil edir. VB-də **Salespeople** (Ticarət agentləri), **Customers** (Sifarişçilər) və **Orders** (Sifarişlər) adlı üç cədvəl var.

**Salespeople** (Ticarət agentləri) cədvəli:

|  |  |
| --- | --- |
| **Sütun** | **Məzmunu** |
| SNUM | Hər bir ticarət agentinə verilmiş unikal kod. |
| SNAME | Ticarət agentinin adı |
| CITY | Agentin yerləşdiyi yer (şəhər) |
| COMM | Komissyon satıcılar |

**Customers** (Sifarişçilər) cədvəli:

|  |  |
| --- | --- |
| **Sütun** | **Məzmunu** |
| CNUM | Hər bir sifarişçiyə verilmiş unikal kod |
| CNAME | Sifarişçinin adı |
| CITY | Sifarişçinin yerləşdiyi yer (şəhər) |
| RATING | Verilmiş sifarişçinin üstünlüyünü göstərən kod, daha yüksək kod daha da üstünlüyü dildirir |
| SNUM | Verilmiş sifarçi üçün təyin olunmuş agentin kodu (**Salespeople** cədvəlindən) |

**Orders** (Sifarişlər) cədvəli:

|  |  |
| --- | --- |
| **Sütun** | **Məzmunu** |
| ONUM | Hər bir sifarişin unikal kodu |
| AMT | Sifarişin məbləği (təbii ki şətri qiymətlərlə :-) |
| ODATE | Sifarişin tarixi |
| CNUM | Sifariş edən sifarişçinin nömrəsi (**Customers** cədvəlindən) |
| SNUM | Sifarişi qəbul edən agentin nömrəsi (**Salespeople** cədvəlindən) |

İndi **Transact-SQL**-in köməkliyi ilə cədvəldən məlumatların çıxarılmasını nümayiş etdirmək olar.

Bütün suallar yeganə **SELECT** əmri ilə realizə olunurlar. Sadə formada **SELECT** əmri VB-dən məlumatların çıxarılması üçün təlimat verir. Məs: agentlər (**Salespeople**) cədvəlindən məlumatları çıxartmaq üçün növbətiləri daxil etmək lazımdır:

SELECT snum, sname, city, comm FROM Salespeople

Bu əmr sadəcə olaraq cədvəldən bütün məlumatları çıxardır. Onun çıxarışı aşağıdakı kimi ğörünəcək:

snum sname city comm

----- -------- ----------- -----

|  |  |  |
| --- | --- | --- |
| 1001 | Иванов | Москва 12 |
| 1002 | Петров | Хабаровск 13 |
| 1003 | Егоров | Караганда 10 |

1004 Сидоров Сочи 11

1007 Шилин Красноярск 15 Bu əmrdə:

* + **SELECT** - açar sözüdür.
  + **snum, sname** - cədvəldən sual ilə seçiləcək sütunların siyahısıdır. Bu siyahıda qeyd olunmamış digər sütunlar sayılmırlar.
  + **FROM** - açar sözüdür, ondan sonra məlumat toplusu olan cədvəllərin adları yazılır.

Əgər siz cədvəldəki bütün sütunları görmək istəyirsinizsə onda xüsusi ixtisardan istifadə edin:

SELECT \*

FROM Salespeople

Çıxarılacaq məlumat əvvəlki haldakı kimi olacaq.

Ümumi halda **SELECT** əmri açar sözü olan **SELECT** ilə başlayır, ondan sonra ekrana çıxartmaq istıdiyimiz sütunların adları, əgər hamı sütunları istəyirsinizsə onda

* (ulduz) işarəsi yazılır. Daha sonra isə **FROM** açar sözü və sual olunan cədvəlin adı verilir.

Daha nə olacaq? Növbəti addımlarda **SELECT** əmrinin dərinlikləri təqdim olunacaq: verilənlərin nizamlanması, çətin məntiqi verilənlər və xüsusi operatorlar.

# SELECT – mürəkəb şərtlər

**Transact-SQL**-in əsasları olan **SELECT** əmrinin imkanlarını öyrənməyi davam edək. O ciddi müəyyən edilmiş məlumatların cədvəldən çıxartmaq xassəsinə malikdir. Başlanğıc üçün cədvəldən müəyyən edilmiş sütunların seçilməsinə baxaq. Bu **SELECT** əmrindən sütunların çıxarılması ilə əldə edilir.

Sual:

SELECT sname, comm FROM Salespeople

Nəticə aşağıdakı kimi olacaq: sname comm

-------- -----------

Иванов 12

Петров 13

Егоров 10

Сидоров 11

Шилин 15

**DISTINCT** - sizin **SELECT** əmrində olan təkrar olunan qiymətlərin aradan qaldırılmasını təmin edir. Tutaq ki siz bilmək istəyirsiniz ki hal hazırda hansı agentlərin sizin sifariş cədvəlinizdə sifarişçiləri var. Sizə agentlərin neçə sifarişi olduğu bilmək lazım deyil, sizə sadəcə olaraq agentlərin kodları (**snum**) lazımdır. Bunun üçün siz növbətiləri daxil edəcəksiniz:

SELECT snum FROM Orders

və belə nəticə alacaqsınız:

snum

----------- 1007

1004

1001

1002

1007

1002

1001

1003

1002

1001

Gördüyünüz kimi kodlar təkrar olunurlar. Təkrarsız siyahının alınması üçün növbətiləri daxil etmək lazımdır:

SELECT DISTINCT snum

FROM Orders

Artıq nəticə ayrı cürdür:

snum

----------- 1001

1002

1003

1004

1007

**DISTINCT**-i **SELECT** əmrində yalnız bir dəfə istifadə etmək olar. Əgər bir neçə sütun seçilibsə onda **DISTINCT** tam eyni olan sətirləri çıxardır. Bir neçə qiymətləri eyni olan sətirlər qalacaqlar.

**DISTINCT** yerinə siz **ALL** göstərə bilərsiniz. Bu əks təsir edəcək, sətirlərin oxşarlığı yadda qalacaq. Bu heç bir arqumentin yazılmamasına oxşayır. Ona görə də **ALL** sadəcə olaraq başa salıcı arqumentdir.

Cədvəllər çox böyük olmaq meylinə malikdirlər. Bir halda ki sizi onlardan bəziləri maraqlandırır onda bu sətirləri seçmək üçün qayda təyin etmək imkanı var.

**WHERE** - cədvəlin ixtiyari sətiri üçün qayda təyin edən və doğru və ya yalnış olan **SELECT** əmrinin təklifidir. Əmrə yalnız cədvəlin verilmiş sətirləri yerinə yetirən sətirlərini yetirən sətirləri verir. Tutaq ki siz Moskvada olan agentlərin ad və komissiyonnularının adlarını görmək istəyirsiniz. Bundan sadə heç nə yoxdur:

SELECT sname, city FROM Salespeople WHERE city = 'Москва'

Sualda **WHERE** təklifi olanda **SQL Server** bütün cədvələ sətir-sətir baxır və hər bir sətirdə verilmiş təklifin doğruluğunu yoxlayır.

**Qeyd.** 'Москва' tipli sətir konstantları **Transact-SQL**-də apostrof ', və ya dırnaq" ilə məhdudlaşırlar.

İndi isə **WHERE** təklifini rəqəm tipli ədədlər üçün qurmağa çalışaq. Sifarişçilər (Customers) cədvəlinin rating sütununun sifarişçiləri bu ədədlərin köməkliyi ilə bir neşə kriteriyaya əsaslanmış qruplara bölmək üçün təyin olunub. Məsələn bu əvvəlki sifarişlərin ölçüsündən aslı olan kreditin qiymətlənməsi və qiyməti forması ola bilər. Reytinqi 100 olan bütün sifarişçiləri seçək:

SELECT \*

FROM Customers WHERE rating = 100

Burda dirnaq işarəsindən istifadə olunmur, yəni ki - bunlar ədəd tiplidirlər.

Sorğunun nəticəsi:

CNUM CNAME CITY RATING SNUM

----- ------------------ ------- ----------- -----------

2001 ТОО Рога и копыта Москва 100 1001

2006 Clemens Лондон 100 1001

2007 ОАО "ООО" ТОМСК 100 1004

İndi siz cədvələ sizə lazım olan informasiyanın sizə verilməsi üçün onu necə məcbur etmək üsullarını bilirsiniz. Əsas olur ki siz cədvəldən min sətir içərisindən birini seçmək üsulunu bilirsiniz. Şərtlər çətin ola bilər, lakin elə bu SQL dilini daha güclü edir.

# SELECT - müqayisə, məntiqi və xüsusi operatorlar

Əvvəlki addımda biz cədvəldən məlumatın çəkilməsi üçün **WHERE** təklifində bərabərliklik = işarəsindən istifadə etdik. **Transact-SQl** yazılardan çətin şərtli məlumatların çıxarilmasını təmin edir və bunun ümün yanaşma operatorları, məntiqi operatorlar və xüsusi operatorlar istifadə olunur. Yanaşma operatorları aşağıdakılardır:

 = Bərabər

* + > Böyük
  + < Kiçik
  + >= Böyük bərabər
  + <= Kiçik bərabər
  + <> Fərqli (bərabər deyil)

Onlar simvol və tarix tipli dəyişənlər üçün eyni qiymət alırlar. Simvol tipli dəyişənlət onların kodlarındakı terminlərlə müqayisə olunurlar.

Tutaq ki bizə reytinqi 200-dən çox olan sifarişçiləri görmək lazımdır: SELECT \*

FROM Customers WHERE rating > 200 Nəticə belədir:

CNUM CNAME CITY RATING SNUM

----- --------------------------------- ---------- ------- -----

2004 Концерн "Дети лейтенанта Шмидта" Бобруйск 300 1002

2008 ОАО "Валют-транзит" Караганда 300 1007 Məntiqi operatorlar:

* **AND** məntiqi "və"
* **OR** məntiqi "va ya"
* **NOT** məntiqi yox (əks fikir)

**AND** operatoru iki məntiqi qiyməti müqaisə edir və əgər hər iki qiymət doğrudursa TRUE (doğru), digər hallarda isə FALSE (yanlış) qiymətini qaytarır. **OR** operatoru hər hansı bir qiymət TRUE olarsa TRUE qaytarır. **NOT** operatoru isə əgər qiymət FALSE-dirsə TRUE və əksinə qiymət qaytarır. Məntiqi operatorların istifadə olunması **SELECT** əmrinin imkanlarını artırır. Məs: 'Караганда'-dan olan və reytinqi 200-dən artıq olan sifarişçilərə baxmaq üçün sadəcə olaraq belə əmr lazımdır:

SELECT \*

FROM Customers

WHERE city = 'Караганда' AND rating > 200

Bizim VB-də bu şərti ödəyən yalnız bir sifarişçimiz var. CNUM CNAME CITY RATING SNUM

----- --------------------------------- ---------- ------- -----

2008 ОАО "Валют-транзит" Караганда 300 1007

Əgər **OR**-dan istifadə etsək, onda 'Караганда'-da olan və ya reytinqi 200-dən şox olanların siyahısını alacağıq:

SELECT \*

FROM Customers

WHERE city = 'Караганда' OR rating > 200

Sualın nəticəsi:

CNUM CNAME CITY RATING SNUM

|  |  |  |
| --- | --- | --- |
| ----- --------------------------------- ---------- ------- ----- |  | |
| 2004 Концерн "Дети лейтенанта Шмидта" Бобруйск | 300 | 1002 |
| 2008 ОАО "Валют-транзит" Караганда 300 | 1007 |  |

NOT qiyməti dəyişmək üçün istifadə olunur. Sual:

SELECT \*

FROM Customers

WHERE city = "Караганда" OR NOT rating > 200

Nəticə:

CNUM CNAME CITY RATING SNUM

----- -------------------- --------- ------- ------

2001 ТОО Рога и копыта Москва 100 1001

2002 AО Бендер и К Одесса 200 1003

2003 Фирма ХХХ Рязань 200 1002

2006 Clemens Лондон 100 1001

2007 ОАО "ООО" ТОМСК 100 1004

2008 ОАО "Валют-транзит" Караганда 300 1007

Məs:

İfadəni qruplaşdırmaq üçün **Transact-SQL**-də mötərizədən **()** istifadə olunur.

SELECT \*

FROM Customers

WHERE NOT (city = 'Караганда' OR rating > 200)

Bu sual 'Караганда'-da olmayan və reytinqi 200-dən çox olmayanları seçəcək.

Nəticə:

CNUM CNAME CITY RATING SNUM

----- ------------------ ------- ------- -----

2001 ТОО Рога и копыта Москва 100 1001

2002 AО Бендер и К Одесса 200 1003

2003 Фирма ХХХ Рязань 200 1002

2006 Clemens Лондон 100 1001

2007 ОАО "ООО" ТОМСК 100 1004

# IN, BETWEEN, LIKE, IS NULL

**IN** operatoru qiymətin verilmiş sətir qiymətinə daxil olduğunu təyin edir. Məs: əgər siz Москва və Хабаровск-da yerləşən bütün satıcıları tapmaq üçün belə bir sorğu verməlisiniz:

SELECT \*

FROM Salespeople WHERE city = 'Москва' OR

city = 'Хабаровск' Lakin daha sadə üsul var:

SELECT \*

FROM Salespeople

WHERE city IN ( 'Москва', 'Хабаровск' ) Bu sualın nəticəsi:

SNUM SNAME CITY COMM

----- ------- ---------- -----

1001 Иванов Москва 12.0

1002 Петров Хабаровск 13.0

**IN** operatoru üçün qiymətlər yumru mötərizə daxilində yazılır və qiymətlər vergül ilə ayrılırlar:

**BETWEEN** operatoru **IN** operatoruna oxşayır. Siyahıdan fərqli olaraq **BETWEEN** operatoru qiymətlər diapazonunu təyin edir. Sorğuda siz əvvəlcə **BETWEEN** sözünü yazmalı sonra başlanğıc qiymət, açar spzü olan **AND** və son qiymət. Birinci qiymət ikincidən kiçik olmalıdır. Növbıti sual komissiyonnuları 10 və 12 arasında olan agentləri seçəcək:

SELECT \*

FROM Salespeople

WHERE comm BETWEEN 10 AND 12 SNUM SNAME CITY COMM

----- -------- ---------- -----

1001 Иванов Москва 12.0

1003 Егоров Караганда 10.0

1004 Сидоров Сочи 11.0

**LIKE** operatoru yalnız simvol tipli sıtirlərə tətbiq olunur və o alt sətirləri axtarır. Yəni ki o həmin sətirin müəyyən hissəsini uyğunlaşdığını axtarır. Şərti olaraq o xüsusi simvolları istifadə edir:

* Altı xətli simvol \_ - İxtiyari bir hərifli simvolu əvəz edir. Məs: 'к\_т' 'кот' və 'кит'-ə uyğun gələcək, amma 'крот' olmayacaq.
* Faiz % simvolu - İxtiyari simvollar ardıcıllığını əvəz edir. Məs: '%м%р' 'компьютер' və 'омар' sözlərinə uyğun gələcək:

Gəlin adları 'О' hərfi ilə başlayan sifarişçiləri seçək:

SELECT \*

FROM Customers WHERE cname LIKE 'О%'

CNUM CNAME CITY RATING SNUM

----- -------------------- ---------- ------- -----

2008 ОАО "Валют-транзит" Караганда 300 1007

2007 ОАО "ООО" ТОМСК 100 1004

**LIKE**-ın istifadəsi sizin qiyməti tam bilməyəndə axtarışı üçün lazımdır. Siz yalnız bildiyiniz hissəni istifadə edə bilərsiniz.

NULL qiymətin olmamasını bildirdiyi kimi siz NULL ilə ixtiyari nəticəsinin necə olduğunu bilməyəcəksiniz. Adətən sizə NULL olan sətirləri ayırmaq lazım gələcək. Bunun üçün xüsusi operator olan **IS NULL** istifadə olunur. VB- də *city* sütunları NULL olan sifarışçıləri seçək:

SELECT \*

FROM Customers WHERE city IS NULL

VB-də belə yazı yoxdur, lakin bu cür yazı ona sifarişçini agentsiz daxil edərkən baş verə bilər.

# Aqreqat funksiyalar

Bu addımda biz qrup qiymətləri götürüb bir qiyətə gətirən aqreqat funksiyaların istifadəsinə keçəcəyik.

SQL Server bir neçə aqreqat funksiya təklif edir.

* **COUNT** - Verilmiş suala uyğun gələn sətirlərin sayını verir
* **SUM** - Sütunda olan bütün qiymətləri cəmləyir.
* **AVG** - Bütün qiymətlərin riyazi ortasını tapır.
* **MAX** - Seçilmiş qiymətlərdən ən böyüyünü tapır
* **MIN** - Seçilmiş qiymətlərdən ən kiçiyini tapır

**SUM** və **AVG** funksiyaları yalnız ədədi sətirlər üçün tətbiq olunandırlar. **COUNT, MAX, MIN** ilə ədədi və simvol sətirlər istifadə oluna bilər.

Simvol sətirlərlə istifadə olunanda **MIN, MAX** dəyişənləri alfavit sırası üzrə müqayisə edir. Aqreqat funksiyalar öz işlərində **NULL** qiymətini nəzərə almırlar. **Orders** cədvəlindəki bütün sifarişçilərin cəmini tapmaq üçün aşağıdakı sualı vermək lazımdır:

SELECT SUM( amt )

FROM Orders Nəticə:

--------------------- 26658.4000

**COUNT** funksiyası digərlərindən bir az fərqlənir. O verilmiş sütunun və ya cədvəlin sətirlərinin sayını hesablayır. Məs: **Orders** cədvəlinə yazılmış ticarət agentlərinin sayını hesablayaq.

SELECT COUNT( DISTINCT snum )

FROM Orders

Nəticə:

----------- 5

Fikir verin ki sorğuda **DISTINCT** açar sözündən istifadə olunmuşdur, bu vaxtı agentlərin yalnız unikal nömrələri hesablanır. Əgər bu açar sözü yazılmasaydı onda nəticə aşağıdakı kimi olacaqdı:

----------- 10

Əgər cədvəldə bütün sətirlərin sayını hesablamaq lazım gələrsə, onda sorğuda COUNT funkiyasında sütun yernə ulduz işarəsi qoymaq lazımdır:

SELECT COUNT(\*)

FROM Customers Nəticə:

----------- 7

**GROUP BY** təklifi sizə aqreqat funksiyanın dəyişməsi üçün lazım olan çoxluq verir. O **SELECT** ifadəsinin daxilində sətir və aqreqat funksiyaları cəmləndirmək imkanı verir. Tutaq ki sizə hər bir ticarət agenti tərəfindən alınmış ən böyük sifarişi tapmaq lazımdır:

SELECT snum, MAX( amt ) FROM Orders

GROUP BY snum Bu sualın nəticəsi:

snum

----- ----------

|  |  |
| --- | --- |
| 1001 | 9891.8800 |
| 1002 | 5160.4500 |
| 1003 | 1713.2300 |
| 1004 | 1900.1000 |
| 1007 | 1098.1600 |

**GROUP BY** aqreqat funksiyaları yazı qruplarından aslı olmayaraq tətbiq edilir. Qrupların formalaşması qaydası - sətirlərin eyni qiymətləridir (verilmiş halda **snum**). Verilmiş sualı realizə edərkən **MAX** funksiyası hər bir **snum** qiyməti üçün hesablanır.

**GROUP BY**-ı bir neçə sətir ilə hesablamaq olar. Əvvəlki sualı çətinləşdirək: SELECT snum, odate, MAX( amt )

FROM Orders

GROUP BY snum, odate

Yəni biz hər bir tarix ilə qəbul olunmuş agentlərin kod və maksimal qiymətlərini hesablayırıq.

snum odate

----- ------------------------ ----------

|  |  |  |
| --- | --- | --- |
| 1001 | 1999-10-03 00:00:00.000 | 767.1900 |
| 1001 | 1999-10-05 00:00:00.000 | 4723.0000 |
| 1001 | 1999-10-06 00:00:00.000 | 9891.8800 |
| 1002 | 1999-10-03 00:00:00.000 | 5160.4500 |
| 1002 | 1999-10-04 00:00:00.000 | 75.7500 |
| 1002 | 1999-10-06 00:00:00.000 | 1309.9500 |
| 1003 | 1999-10-04 00:00:00.000 | 1713.2300 |
| 1004 | 1999-10-03 00:00:00.000 | 1900.1000 |
| 1007 | 1999-10-03 00:00:00.000 | 1098.1600 |

Təbii ki sifariş olmayan günlər görünməyəcək.

Tapşırığı çətinləşdirək: İndi 3000-dən böyük olan hər bir agentin maksimal cəmini tapaq. Bu cür effektə nail olmaq üçün **HAVING** təklifindən istifadə olunur, hansı ki meyar təyin edir, **WHERE** təklifinin ayrıca nəticələr üçün etdiyi nəticə qrupundan pozulması üçün. Bu bu cür edilir:

SELECT snum, odate, MAX( amt ) FROM ORDERS

GROUP BY snum, odate HAVING MAX( amt ) > 3000

snum odate

----- ------------------------ ----------

|  |  |  |
| --- | --- | --- |
| 1002 | 1999-10-03 00:00:00.000 | 5160.4500 |
| 1001 | 1999-10-05 00:00:00.000 | 4723.0000 |
| 1001 | 1999-10-06 00:00:00.000 | 9891.8800 |

Aqreqat funksiyaları təkcə VB-dən təyin olunmuş məlumatları seçmək üçün deyil, həm də onların ümumiləşdirilməsi və analizi üçün də istifadə olunurlar.

# Bir necə cədvəldən verilənlərin seçilməsi

İndiyədək bizim bütün suallarımız yalnız bir cədvələ aid idi. Lakin SQL bir sualda bir neçə cədvələ müraciət etmək imkanı verir. Elə bu xassə SQL dilini məşhur edir. Cədvəldəki sütunun tam adı faktiki olaraq cədvəlin adı sonra nöqtə və sütunun

adından ibarət olur. (Əslində desək əvvəldə istifadəçi adı da istifadə oluur, lakin buna gələcəkdə qayıdacağıq). Adlara nümunə:

Salespeople.snum Salespeople.city Orders.odate

İndiyədək biz suallarda cədvəllərin adlarını buraxırdıq, çünki yalnız bir cədvəldən sorğu edirdik. Əgər biz müxtəlif cədvəllərin sütunlarını birləşdirmək istəyiriksə, onda serverin onları ayırması üçün

### Selespeople.city və Customers.city

yazmalıyıq.

Tutaq ki ticarət agentləri və sifarişçilərin şəhərlər üzrə kombinasiyasını görmək istəyirsiniz. Bu aşağıdakı kimi edilir.

SELECT Customers.cname, Salespeople.sname, Salespeople.city FROM Salespeople, Customers

WHERE Salespeople.city = Customers.city

Sualın nəticəsi:

cname sname city

-------------------- ------- ----------

ТОО Рога и копыта Иванов Москва ОАО "Валют-транзит" Егоров Караганда

Yəni ki city sütunu nticarət agentləri və sifarişçilər cədvəlində var, cədvəlin adları prefiks kimi istifadə olunmalıdırlar.

Bu sual necə işləyir? **SQL Server** hər iki cədvəlin sütunlarının kombinasiyasını yoxlayır və onların **WHERE** şərtinə olan şərtlərini yoxlayır.Əgər bu kombinasiya şərtləri ödayirsə onda o nəticə verir. Cədvəllərin birləşməsi üçün bərabərliklərdən başqa digər müqayisə şərtlərini də istifadə etmək olar. Məs:

SELECT Salespeople.sname, Customers.cname FROM Salespeople, Customers

WHERE Salespeople.sname < Customers.cname AND Customers.rating < 200

Nəticə:

-------- ------------------

Егоров ТОО Рога и копыта Иванов ТОО Рога и копыта Петров ТОО Рога и копыта Сидоров ТОО Рога и копыта Егоров ОАО "ООО" Иванов ОАО "ООО"

Prinsipcə bu heç də xeyirli sorğu deyil. O satıcının adı və sifarişçinin adı arasındakı kombinasiyanı elə edir ki birinci sonuncudan alfavit sırada əvvəl gəlsin, sifarişçinin isə reytinqi 200-dən az olsun.

Tutaq ki bizə agent ilə bir şəhərdə olan bütün sifarişçiləri tapmaq lazımdır.

Bunun üçün üç cədvəli bağlamaq lazımdır.

SELECT Orders.onum, Customers.cname, Orders.cnum, Orders.snum FROM Salespeople, Customers, Orders

WHERE Customers.city <> Salespeople.city AND Orders.cnum = Customers.cnum AND Orders.snum = Salespeople.snum

Nəticə:

onum cname cnum snum

----- ---------------------------------- ----- -----

3001 ОАО "Валют-транзит" 2008 1007

3002 ОАО "ООО" 2007 1004

3005 Фирма ХХХ 2003 1002

3006 AО Бендер и К 2002 1007

3007 Концерн "Дети лейтенанта Шмидта" 2004 1002

3008 Clemens 2006 1001

3009 AО Бендер и К 2002 1003

3010 Концерн "Дети лейтенанта Шмидта" 2004 1002

3011 Clemens 2006 1001

İndi diz bir neçə cədvələ eyni vaxtda sorğu göndərə bilərsiniz. Siz cədvəl ilə bağlı olan ixtiyari qaydaları təyin edə bilərsiniz. əslində elə buna görə də SQL yaradılıb.

# Altsorğular

Sualları digər sualların köməkliyi ilə idarə etmək olar. Bu biz sorğunun şərti yerinə digər sorğunun yerləşdirilməsinə və doğru və ya yalnış şərtlər vastəsi ilə olunur.

Adətən daxili sualın qiyməti, xarici sorğunun doğru olub-olmadığını yoxlayıb realizə edir. Məs: biz ticarət agentinin adını - 'Сидоров' bilirik, lakin onun kodunu bilmirik (snum), və onun Sifarişçilər (Orders) cədvəlindən olan bütün sifarişlərinin almaq istəyiri.

SELECT \*

FROM Orders WHERE snum = (

SELECT snum FROM Salespeople

WHERE sname = 'Сидоров'

)

Xarici sorğunu (əsas) realizə etmək üçüç, əvvəlcə **WHERE** təklifinin daxilində olan daxili (alt sorğu) sorğu realizə olunur. Alt sual realizə olunarkən **sname** sətiri 'Сидоров' alan **Salespeople** cədvəlinə baxılır və sonra **snum** -in qiyməti hesablanır. Yeganə sətir **snum** = 1004 olacaq. Sonra alınmış nəticə əsas sorğunun şərtində yerləşdirilir, belə ki çərtdə belə olacaq.

WHERE snum = 1004

Sonra əsas sorğunun nəticəsi açağıdakı kimi olacaq:

ONUM ODATE AMT CNUM SNUM

----- ------------------------ ---------- ------ -----

3002 1999-10-03 00:00:00.000 1900.1000 2007 1004

Alt sorğularda müqayisə əməliyyatlarını (kiçik, böyük, bərabər, fərqli və s.) apararkən siz əmin olmalısınız ki nəticə bir də yalnız bir cavab qaytaracaq. Əgər sizin alt sual heç bir nəticə verməzsə, onda əsas sorğuda heç bir nəticə verməyəcək.

Əgər siz bir neçə cavab qaytaran alt suallardan istifadə etmək istəyirsinizsə onda **IN** operatorunu istifadə etmək lazımdır. Yadınızdadırsa bu operator mümkün qiymətlər çoxluğunu təyin edir, o alt sorğularda istifadə edilərkən alt sorğunu qaytaran qiymət realizə olunur. Moskvadan olan agentin bütün sifarişçilərini tapaq.

SELECT \*

FROM Orders WHERE snum IN (

SELECT snum FROM Salespeople

WHERE city = 'Москва'

)

Nəticə:

ONUM ODATE AMT CNUM SNUM

----- ------------------------ ---------- ----- -----

|  |  |  |
| --- | --- | --- |
| 3003 | 1999-10-03 00:00:00.000 | 767.1900 2001 1001 |
| 3008 | 1999-10-05 00:00:00.000 | 4723.0000 2006 1001 |
| 3011 | 1999-10-06 00:00:00.000 | 9891.8800 2006 1001 |

Verilmiş halda alt sualın istifadə olunması, uyğunlaşma ilə müqayisədə oxunma və realizənin asanlaşdırır:

SELECT Orders.\*

FROM Orders, Salespeople

WHERE Orders.snum = Salespeople.snum AND Salespeople.city = 'Москва'

Bu sorğunun əvvəlkinə ekvivalent olmasına baxmayaraq, **SQL Server** hər bir iki cədvəldək ibarət olan sətirlərin mümkün kombinasiyasına baxacaq və onlardan uyğunlarını yoxlayacaq. Ən sadəsi cədvəldən ticarət agentləri və onların kodlarını çıxartmalı və onların sifarişçilər cədvəlində axtarmalı. Əslində icrada uduş yoxdur, belə ki qurulmuş **SQL Server**-də sualların optimizatoru son sorğunu alt sorğu ilə olan formaya ötürəcək.

Bütün yuxarıda qeyd olunmuş alt sorğuları onların bir sütunu seşmələri birləşdirir. Bu vacibdir, belə ki onların nəticəsi eyni qiymətlə müqayisə edilir. **SELECT \*** tipli əmrlərin alt sorğularda istifadəsi qadağandır.

Alt sorğuları həmçinin **HAVING** təkliflərində də isdifadə etmək olar. Bu alt sorğular özlərinin xüsusi təklifləri olan **GROUP BY** və **HAVING** təkliflərini istifadə edə bilərlər. Növbəti sorğu buna nümunədir:

SELECT rating, COUNT( DISTINCT cnum ) FROM Customers

GROUP BY rating HAVING rating > (

SELECT AVG( rating ) FROM Customers WHERE city = 'Москва'

)

Bu əmr Moskvada olan və orta reytinqdən yüksək olan sifarişçiləri sayır. Nəticə: rating

------- --

200 2

300 2

İndi siz düzəlmiş alt sorğulardan bilirsiniz. Bu mexanizm verilənlərin seçilməsi imkanını artırır.

# Bağlanmış altsorğular

Siz alt sorğuları istifadə edərkən qurulmuş alt sorğuya xarici alt sorğudan müraciət edə bilərsiniz. Məs, 3 oktyabra olan sifarişçiləri necə tapmalı:

SELECT \*

FROM Customers C WHERE '1999-10-03' IN (

SELECT odate FROM Orders O

WHERE O.cnum = C.cnum

)

Nəticə:

CNUM CNAME CITY RATING SNUM

----- -------------------- ---------- ------- -----

|  |  |  |  |
| --- | --- | --- | --- |
| 2001 | ТОО Рога и копыта | Москва 100 | 1001 |
| 2002 | AО Бендер и К | Одесса 200 | 1003 |
| 2003 | Фирма ХХХ | Рязань 200 | 1002 |
| 2007 | ОАО "ООО" | ТОМСК 100 | 1004 |

2008 ОАО "Валют-транзит" Караганда 300 1007 Bunlar hamsı necə işləyir?

Yuxarıda istifadə olunan **C** və **O** cədvəllərin təxəllüsləridir. Be lə ki sorğunun cnum sətirindəki qiymət dəyişir, və xarici sorğu hər bir daxili sorğunun nəricəsi üçün ayrıca realizə olunmalıdır. Onun üçün daxili realizə olunan xarici sorğu sətiri namizəd sətir adlanır.

Bağlı sorğu ilə realizə olunan sətirin realizə prosesi:

1. Daxili sorğuda qeyd olunmuş sətiri cədvəldən seçmək. Bu cari namizəd sətir olacaq.
2. Bu namizəd sətirin qiymətini hələlik buferə salmaq.
3. Alt sualı realizə etmək. Yazıların seçilməsi üçün namizəd sətirdən istifadə etmək.
4. III bölmədə realizə olunan daxili alt sorğuların nəticəsi əsasında xarici sorğunun nəticəsini hesablamalı. Namizəd sətirin seçdiyi təuin edilir.
5. Digər sətirlər üçün əməliyyatı təkrar etməli.

Prinsipcə aşağıdakı kimi birləşmədən də istifadə etmək olar: SELECT C.\*

FROM Customers C, Orders O WHERE C.cnum = O.cnum AND

O.odate = '1999.10.03'

Lakin, əgər eyni istifadəci iki və daha artıq sifariş edibsə onda onun adı bir neçə dəfə təkrar olunacaq. Bunu **DISTINCT**-dən istifadə etməklə dəf etmək olar, lakin bu

effektli nəticə deyil. **IN** operatoru alt sorğu ilə olan variantda alt sorğu olə bir dəfə və təkrarən seçilən dəyişənlər arasında fərq qoymur. Ona görə də **DISTINCT** lazım deyil.

Tutaq ki biz heç olmasa bir sifarişçisi olan satıcıların ad və nömrələrini bilmək istəyirik:

SELECT snum, sname FROM Salespeople S WHERE 1 < (

SELECT COUNT(\*)

FROM Customers c WHERE c.snum = s.snum

)

Nəticə:

snum sname

----- -------

1001 Иванов

1002 Петров

Bağlı sualları özləri ilə müqayisə üçün də istifadə etmək olar. Məs, sifarişçilər üçün orta balın cəmindən yuxarı olan sifarişləri tapmaq olar.

SELECT \*

FROM Orders O WHERE amt > (

SELECT AVG( amt )

FROM Orders O1

WHERE O1.cnum = O.cnum

)

Nəticə:

ONUM ODATE AMT CNUM SNUM

----- ------------------------ ---------- ----- -----

3009 1999-10-04 00:00:00.000 1713.2300 2002 1003

3010 1999-10-06 00:00:00.000 1309.9500 2004 1002

3011 1999-10-06 00:00:00.000 9891.8800 2006 1001

Təbii ki bizim balaca VB-də aksər sifarişçilərin yalnız bir sifarişləri var, əksər qiymətlər eyni vaxtda orta olmurlar və ona görə də seçilmirlər.

# EXISTS operatoru

Alt sorğularla işi bitirdikdən sonra indi yalnız alt sorğuların arqumentlər kimi istifadə etdiyi operatorların öyrənilməsinə keçmək vaxtıdır və **EXISTS** operatorundan başlayaq.

**EXISTS** operatoru alt sorğunu arqument kimi götürür və əgər alt sorğu hanısa bir sətiri qaytarırsa - doğru, əks halda yanlış qiymət alır. Məs, biz heç olmazsa bir cədvəldən sifarişçisi Moskvadan olan məlumatları seçə bilərik.

SELECT cnum, cname, city FROM Customers

WHERE EXISTS ( SELECT \*

FROM Customers WHERE city = 'Москва'

)

Nəticə:

cnum cname city

----- --------------------------------- ---------

2001 ТОО Рога и копыта Москва 2002 AО Бендер и К Одесса

2003 Фирма ХХХ Рязань

2004 Концерн "Дети лейтенанта Шмидта" Бобруйск 2006 Clemens Лондон

2007 ОАО "ООО" ТОМСК

2008 ОАО "Валют-транзит" Караганда

Daxili sorğu Moskvadan olan sifarişçilərin bütün məlumatlarını seçir. **EXISTS** operatoru yoxlayır ki daxili sorğu heç ilmasa bir nəticə verdi və şərt

doğrudur. Alt sorğu xarici sorğu üçün yalnız bir dəfə realizə olunuz və bütün hallarda eyni qiymət alır Ona görə də **EXISTS** bu cür istifadə edilərkən eyni vaxtda şərti sətirlər üçün doğru və ya yalnış edir.

Bağlanmış alt sorğularda **EXISTS** təklifi xarici sorğuda adları qeyd olunmuş cədvəlin hər bir sətiri ümün qiymətləndirilir. Bu **EXISTS**-ə əsas sorğuda qeyd olunmuş nəticələri cədvəlin hər bir sətiri üçün müxtəlif cavablar kimi istifadə etmək imkanı verir. Məs, biz bir necə sifarişçisi olan ticarət agentlərini çıxara bilərik.

SELECT DISTINCT snum

FROM Customers couter WHERE EXISTS (

SELECT \*

FROM Customers cinner

WHERE cinner.snum = couter.snum AND cinner.cnum <> couter.cnum

)

Nəticə:

snum

----------- 1001

1002

Xarici sorğunun hər bir namizəd satiri üçün (Cari vaxtda sifarişçi göstərməni yoxlayan), daxili sorğu **snum** (agentin malik olduğu) sətiri ilə üst-üstə düşən, lakin **cnum** (digər sifarişçiyə aid olan) ilə düçməyən sətirlər tapır. Əgər belə sətirlər daxili sorğu vastəsi ilə tapılıbsa, onda bu o deməkdir ki, bir satıcı tərəfindən xidmət edilən iki sifarişçi var. Elə buna görə **EXISTS** cari sətir və satıcı (**snum**) üçün doğru cavab verəcək. Əgər **DISTINCT** istifadə olunmasaydı, onda bu satıcılardan hər biri onların sifarişçilərinin sayı qədər seçiləcək.

Bu agentlər haqqında yalnız onların adları yox, həmçinin hər tərəfli məlumat verilsə idi yaxşı olardı. Bunu sifarişçilər cədvəlini agentlər cədvəli ilə birləşdirməklə etmək olar:

SELECT DISTINCT first.snum, first.sname, first.city FROM Salespeople first, Customers second

WHERE EXISTS (

SELECT \*

FROM Customers third

WHERE second.snum = third.snum AND second.cnum <> third.cnum)

AND first.snum = second.snum

Nəticə:

snum sname city

----- ------- ----------

1001 Иванов Москва

1002 Петров Хабаровск

Daxili sorğu əvvəlki üsuldakı kimidir - xarici sorğu agentlər cədvəli ilə sifarişçilər cədvəlinin birləşməsidir.

Əvvəlki nümunə göstərdi ki, **EXISTS** operatorunu məntiqi operatorlar ilə də istifadə etmək olar. Bu cür istifadələrdən ən sadəsi **NOT** operatorunun istifadəsidir - görünür **EXISTS** operatorunun ən qalın üsuludur. Bir sifarişçi ilə olan satıcıların tapılmasının bir üsulu da əvvəlki sorğunun invertisiyasından ibarətdir:

SELECT DISTINCT snum

FROM Customers couter WHERE NOT EXISTS (

SELECT \*

FROM Customers cinner

WHERE cinner.snum = couter.snum AND cinner.cnum <> couter.cnum

)

Nəticə:

snum

----------- 1003

1004

1007

# ANY, ALL operatorları

Şəhərlərində yerləşmiş sifarişçiləri ilə olan agentlərin tapılmasının yeni üsuluna baxaq:

SELECT \*

FROM Salespeople WHERE city = ANY (

SELECT city FROM Customers

)

Nəticə:

SNUM SNAME CITY COMM

----- ------- ---------- -----

1001 Иванов Москва 12

1003 Егоров Караганда 10

**ANY** operatoru alt sorğuda yazılmış bütün sorğunu götürür və əgər onlardan xarici sorğunun ixtiyari biri cari sətirdəki şəhərə bərabərdirsə onda onu doğru kimi qiymətləndirir. Bu o deməkdir ki alt sorğu, əsas şərtdəki qiymətə uyğun olan sorğunu seçməlidir.

Yuxarıda qeyd olunmuş sorğuda **IN** operatorundan da istifadə etmək olar. Amma **ANY** operatorunu yalnız bərabərlik operatoru ilə istifadə etmək olzaz. Məs, sifarişçiləri əlifba sırasında olan bütün agentləri tapmaq olar:

SELECT \*

FROM Salespeople WHERE sname < ANY (

SELECT cname FROM Customers

)

Nəticə:

SNUM SNAME CITY COMM

----- -------- ---------- -----

1001 Иванов Москва 12

1002 Петров Хабаровск 13

1003 Егоров Караганда 10

1004 Сидоров Сочи 11

**ALL** operatoru, əgər alt sorğu ilə seçilmiş xarici sorğunun hər biri şərti ödəyirsə onda o şərt doğru hesab edir. Reytinqi Moskvadakı sifarişçilərdən yüksək olan sifarişçiləri seçək:

SELECT \*

FROM Customers WHERE rating > ALL(

SELECT rating FROM Customers

WHERE city = 'Москва'

)

Nəticə:

CNUM CNAME CITY RATING SNUM

----- --------------------------------- ---------- ------- -----

|  |  |  |
| --- | --- | --- |
| 2002 | AО Бендер и К Одесса 200 1003 |  |
| 2003 | Фирма ХХХ Рязань 200 1002 |
| 2004 | Концерн "Дети лейтенанта Шмидта" Бобруйск | 300 1002 |
| 2008 | ОАО "Валют-транзит" Караганда 300 | 1007 |

# UNION əmri

**UNION** əmri sadəcə olaraq bir neçə sorğunu bir sorğu kimi birləşdirir. Məs, aşağıdakı sorğu sifarişçi və agentləri birləşdirir:

SELECT snum, sname FROM Salespeople WHERE city = 'Москва'

UNION

SELECT cnum, cname FROM Customers WHERE city = 'Москва'

Nəticə:

snum sname

----- ------------------

2001 ТОО Рога и копыта 1001 Иванов

**UNION** əmrinin istifadəsi üçün iki qaydadan istifadə edilir.

* + Bütün sorğularda sütunların ardıcıllığı eyni olmalıdır.
  + Verilənlərin tipləri uyğun gəlməlidirlər. Tiplərin uyğunluğu sadə təyin olunur.

|  |  |
| --- | --- |
| Sütundakı verilənin tipi | Nəticənin tipi |
| **Hər iki sütun qeyd olunmuş L1 və L2 uzunluqlu** char **tiplidirlər.** | L1 və L2-dən ən böyük olan uzunluğa malik **char** |
| **Hər iki sütun qeyd olunmuş L1 və L2 uzunluqlu** binary **tiplidirlər.** | L1 və L2 dən ən böyük olan uzunluğa malik **binary** |
| **Bir və ya bir neçə** varchar **tipi** | L1 və L2 dən ən böyük olan uzunluğa malik **varchar** |
| **Bir və ya bir neçə** varbinary **tipi** | L1 və L2 dən ən böyük olan uzunluğa malik **varbinary** |
| **Hər iksi say tiplidir (**smallint**,**money**,** float**)** | Verilənlərin tipi ən böyük dəqiqliklə (**int**=>**float**) |

**UNION** avtomatik olaraq çıxışa verilən aşkar sətirləri silir. Əgər siz istəyirsiniz ki bütün sətirlər nəticəyə verilsin onda **UNION ALL**-dan istifadə edin.

SELECT snum, city FROM Customers

UNION ALL

SELECT snum, city FROM Salespeople

Nəticə:

snum city

----- -----------

1001 Москва

1003 Одесса

1002 Рязань

1002 Бобруйск

1001 Лондон

1004 ТОМСК

1007 Караганда

1001 Москва

1002 Хабаровск

1003 Караганда

1004 Сочи

1007 Красноярск

**UNION** ilə nəticəni sıralamaq ümün **ORDER BY**-ı istifadə etmək olar. Bu vaxt **ORDER BY, UNION**-dan çıxan yalnız axırıncıdan sonrakı sorğunu göstərir.

SELECT a.snum, sname, onum, 'Наибольший на ',odate FROM Salespeople a, Orders b

WHERE a.snum = b.snum AND b.amt = (

SELECT MAX(amt)

FROM Orders c

WHERE c.odate = b.odate

) UNION

SELECT a.snum, sname, onum, 'Наименьший на ', odate FROM Salespeople a, Orders b

WHERE a.snum = b.snum AND b.amt = (

SELECT MIN(amt)

FROM Orders c

WHERE c.odate = b.odate

)

ORDER BY 3

Nəticə:

snum sname onum odate

----- ------- ----- -------------- -----------

1007 Шилин 3001 Наименьший на 1999-10-03

1002 Петров 3005 Наибольший на 1999-10-03

1002 Петров 3007 Наименьший на 1999-10-04

1001 Иванов 3008 Наименьший на 1999-10-05

1001 Иванов 3008 Наибольший на 1999-10-05

1003 Егоров 3009 Наибольший на 1999-10-04

1002 Петров 3010 Наименьший на 1999-10-06

1001 Иванов 3011 Наибольший на 1999-10-06

3 sadəcə olaraq çıxış sütununun nömrəsidir. Yazıları bu cür sadə sıralamaq olar, yəni ki **UNION**-u istifadə edərkən sütunların adları ixtiyari cür görünür.

İndi siz ixtiyari sayda soröunu birləşdirməyi bacarırsınız. Əgər sizdə müxtəlif istifadəçilərə aid olan bir-neçə cədvəl varsa, uyğunlaşma eyni cür qarışdırılır və sıralama üsuluna malikdir.

# INSERT - sətirlərin daxil olunması

İndiyədək biz müxtəlif üsullarla cədvəllərdən məlumatları seçirdik. Onların ora yerləşməsini öyrənmək vaxtı gəlib çatıb.

Dəyişənlar üç əmr vastəsi ilə pozula və yerləşdirilə bilər:

* **INSERT** - verilənlərin yerləşməsi
* **UPDATE** - verilənlərin dəyişməsi
* **DELETE** - pozulma Verilənlərin daxil olunması

Bütün sətirlər yalnız **INSERT** əmri vastəsi ilə daxil olunur. Ən sadə formada aşağıdakı sintaksisdən istifadə olunur:

*INSERT INTO table\_name VALUES ( value, value, ... )*

Belə ki ticarət agentləri cədvəlinə yazı daxil etmək üçün aşağıdakı əmri istifadə etmək olar:

INSERT INTO Salespeople

VALUES( 1008, 'Johnson', 'London', 12 )

Dəyişiklik əmri heç bir çıxış vermir. Lakin **Query Analyzer** sizə bir yazının daxil olunması barədə məlumat verəcək. Əmrin istifadəsindən qabaq cədvəl mövcud olmalıdır, və **VALUES** sətirindən sonra mptarozə daxilindıki yazıların hər biri verilmiş sütunlar ilə üst-üstə düşmalidir. Birinci dəyişən birinci sətirə, ikinci-ikinciyə və s.

Əgər sizə boş sətir (**NULL**) daxil etmək lazımdırsa, onda onu siyahıda qeyd edin. Məs:

INSERT INTO Salespeople VALUES ( 1009, 'Peel', NULL, 12 )

Siz dəqiq daxil edəcəyiniz sütunları da qeyd edə bilərsiniz. Bu dəyişənləri ixtiyari ardıcıllıqda daxil etmək imkanı verir.

INSERT INTO Customers( city, cname, cnum ) VALUES( 'Новосибирск', 'Петров', 2010 )

Fikir versəz görərsiniz ki *reytinq* və *snum* sütunları yoxdurlar. Bu o deməkdir ki yazı daxil edilərkən onlara qiymət susmaya görə veriləcək. Adətən bu **NULL** və ya cədvəl yaradılarkən göstərilən qiymət olur. Buna daha ətraflı irəlidə baxacağıq. **INSERT** əmrini sorğunun nəticələrini yerləşdirmək üçün də istifadə etmək olar. Bunu etmək üçün sadəcə olaraq **VALUES**-də dəyişəni uyğun sorğu ilə dayişirik:

INSERT INTO MoscowStaff SELECT \*

FROM Salespeople WHERE city = 'Москва'

Burada "Ticarət agentləri" cədvəlindən city="Москва" qiyməti olan bütün məlumatlar **MoscowStaff** cədvəlinə yerləşdirilir. Bunların işləməsi üçün **MoscowStaff** cədvəli aşağıdakı şərtləri ödəməlidir:

* O **CREATE TABLE** əmri vastəsi ilə yaradılmalıdır
* O ticarət agentləri cədvəlinin sütunlarının şərtlərini ödəyən dörd sütundan ibarət olmalıdır.

# DELETE - sətirlərin cədvəldən silinməsi

Sətirlərin cədvəldən silinməsi üçün DELETE əmrindən istifadə olunur. O ayrı dəyişənləri pozmur, bütün sətiri pozur. Agentlər cədvəlinin bütün tərkibini pozmaq üçün siz aşağıdakı əmri istifadə etməlisiniz:

*DELETE FROM Salespeople*

Lakin mən sizə yalnız bunu etməyi məsləhət görmürəm.

Adətən sizə cədvəldən bəzi təyin olunmuş sətirləri silmək lazım olur. Bu cür pozulacaq sətirləri təyin etmək üçün sorğularda etdiyimiz kimi şərtləri istifadə edin. Məs, Şilin agentini silmək üçün aşağıdakıları daxil etmək lazımdır:

DELETE FROM Salespeople WHERE snum = 1007

Təbii ki əgər şərtə bir neçə sətir uyğun gələrsə onda onların hahsı silinəcək.

**DBASE** kimi VBİS (ODBC)-lərdən fərqli olaraq **SQL Serve**r yazıları pozuluş kimi qeyd etmir, onları fiziki pozur, yəni onları qaytarmaq olmur. **DELETE** əmri ilə ehtiyatlı olun!

# UPDATE - sətirin qiymətlərinin dəyişməsi

**UPDATE** əmri cədvəldə bir neçə və ya bütün yazıları dəyişmək imkanı verir.Bu əmr özündə arxasınca cədvəlin adını göstərən və lazım olacaq dəyişikliyi

göstərən **SET** təklifi olan **UPDATE** təklifini saxlayır. Məs, bütün sifarişçilərin reytinqini 200-ə dəyişmək üçün aşağıdakı əmri daxil etmək lazımdır:

*UPDATE Customers SET rating = 200*

**UPDATE** və **DELETE** əmrlərinə oxşar olaraq burda da dəyişiklikliyə lazım olacaq sətirləri seçmək olar. İvanov adlı bütün agentinin reytinqini belə dəyişmək olat (kod 1001):

UPDATE Customers SET rating = 300 WHERE snum = 1001

**SET** təklifində vergül ilə bir neçə cədvəl adı vermək olar.