GİRİŞ

SQL (Structural Query Language)-insanların Verilənlər Bazası sistemləri ilə əlaqəsini təmin edən bir dildir. Bu dil sayəsində verilənlər Bazasından məlumatlari ala bilir, dəyişdirə bilir və ya yeni məlumatlar daxil edə bilərik. SQL bir dildir, ancaq proqramlaşdirma dili deyildir. Proqramı inkişaf etdirən zaman SQL-dən istifadə olunur, ancaq təkbaşına bu iş üçün yetərli deyildir.

Verilənlərin məlum xassələrinə görə qruplaşdırılıb diskdə yaddaşda saxlanmasına Verilənlər Bazası İdarəetmə Sistemi deyilir. Verilənlər Bazasından ən məşhur olanı Əlaqəli verilənlər bazasıdır. Əlaqəli Verilənlər Bazasının əsası 1970-ci illərdə İBM firmasında həll edilən məsələlərdə qoyulmuşdur. Ardıcıl məsələlərlə, 1983-cü ildə SQL standartlari tanınmış və ardınca 1987-ci ildə əvvəl İSO, sonra da ANSİ tərəfindən qəbul edilmişdir. Daha sonra bu standartlar çərçivəsində bir çox Verilənlər Bazası İdarəetmə sistemləri yaradılmışdır. Bunlardan ən çox istifadə olunanları Oracle, Sybase, MySql, Server, İnformix və MySql-dir. Bu Verilənlər Bazası İdarəetmə Sistemlərinin asanliq üçün standart dildən uzaqlaşan tərəfləri var, amma ümumi istifadə edilən dil hamisi üçün ortaqdir və SQL-dir.

Verilənlər Bazası İdarəetmə Sistemləri:

Verilənlər Bazası İdarəetmə sistemləri fiziki yaddaşda məlumatları müxtəlif xassələrinə görə qruplaşdirib saxlayan proqramlardir. Verilənlər Bazası İdarəetmə sistemləri saxlanan verilənləri SQl əmrləri ilə insanların istəkləri çərçivəsində hazirlayir, yenidən şəkilləndirirlər.

SQL verilənləri tanitma dili

SQl verilənləri tanitma dili (Data Definition Language) verilənin nə olduğundan əlavə verilənin tipi ilə əlaqəli məlumatlari təyin etmək üçün istifadə edilir. Bir Verilənlər Bazasında hansı cədvəllər olacaq, bu cədvəllərdə hansı sahələr olacaq və sahələrin tipləri nə olacaq, indeks və ya açar olacaqmi?-bunların hamısı DDL ilə müəyyənləşdirilir, dəyişdirilir və olan bir təyin etmədən imtina edib, silinə bilər. Verilənlər Bazası üzərində hər hansı bir təyin daxil ediləcəksə bu, CREATE ifadəsi ilə yaradılır. DROP ifadəsi ilə təyindən imtina edilib, silinir. ALTER ifadəsi isə təyindən üzərində dəyişikliklər edilir.

Əsas verilən tipləri:

SQL-də yeni bir veriləni təyin etmək üçün sadəcə var olan tiplərdən istifadə edə bilərik.

İNTEGER(ölçü**)-**Tam say;

İNT(ölçü**)-**Tam say

Smallint(ölçü**)-**Kiçik tam say

Tinnyint(ölçü**)-**rəqəm tam ölçü

Decimal(ölçü, d**)-**onluq

Float(ölçü, d**)-**kəsir say

Char(ölçü**)-**daha çox uzunluğu sabit olan (telefon nömrəsi kimi) simvol verilənləri üçün istifadə edilir.

Ölçü:bu sahədə ən çox neçə simvol ola biləcəyini verir.

Datetime:Tarix

Logical(bit**)-**bit true/false və ya yes/no ola bilər. Ən az yer tutan verilən tipidir. 1 və 0 olmaqla iki qiymət ala bilər.

Verilənlər Bazası:Verilənlər Bazası cədvəlləri saxlayan Verilənlər Bazası yaradıcısınn ən böyük üzvüdür. Bu Verilənlər Bazası aşağıdakı şəkildə açılır:

Create DataBase database_name

Misal1.

Create DataBase db kitabxana;

Və aşağıdakı şəkildə silinir:

Drop Database_name

Misal2.

Drop Database db kitabxana;

Cədvəllər:

Cədvəl yaratmaq-

Create Table cədvəl_adı (sütun_adi1 verilən tipi[Not Null], sütun_adi2 verilən tipi[Not Null], …] ilə yeni bir cədvəl daxil edilir.

Misal 3.

Create Table (kitab(kitabNo İnteger Not Null constraint cnskitabNo Primary Key, kitabadı Varchar (63) Not Null, İSBNo Varchar(15), Tipi Varchar(20), səhifəsayi İnteger, kitabxülasəsi Varchar(255))

Ilə cədvəl yarada bilərik.

Məhdudiyyətlər(Constraint) yaratma:

Məhdudiyyətlər cədvəllərin bir hissəsi olaraq təyin olunur, MsAccess və SQL Server-də istifadə edilir.

Aşağidakı əməliyyatlari yerinə yetirə bilir:

1. İNDEX kimi ilkin açar sahə təyin edə bilirlər (Primary Key): Bu halda standart olaraq Clustred İndex kimi hərəkət edirlər.
2. İndex kimi ilkin sahə təyin edə bilirlər.
3. Cədvəl yaratma zamani da edilə bilən (Not Null) sahə təyin edə bilirlər.
4. Xarici açar təyin edə bilirlər(Foreign Key).

PRİMARY KEY:Bir cədvəldəki hər bir sətrin yerinə onu qoruya biləcək bir açar verilməli-dir. Standart olaraq bir cədvəldə verilənlərin fiziki yaddaşda da hansi sahəyə görə düzüləcəyəni də Primary Key təyin edir. Bu, bəzən tək bir sahə ola biləcəyi kimi, bəzən də birdən çox sahə də birləşərək ilkin açar yaradila bilər.

UNİQUE KEY:Unique key olaraq təyin olunan açaq üçün bir qiymət sadəcə bir dəfə qeyd oluna bilər. Bir başqa sətrə daha eyni verilənlərin daxil olunmasina icazə verilmir. Primary key-dən fərqli olaraq Unique Key Null qiymət ala bilir.

FOREİGN KEY:Bir cədvələ daxil edilə biləcək qiymətləri başqa bir cədvəlin məlum sahəsində olan verilən qrupu ilə sərhədləndirməyə və ən əsasi da əlaqələndirməyə yarayır.

Məsələn, olmayan bir kitabin qiymət cədvəlinə daxil edilməməsi və qiymət cədvələ daxil olunan bir kitabın nömrəsi vasitəsi ilə məlumatların əldə olunması. Burada kitab. kitabNo ilkin açar sahə, Qiymət. kitabNo isə xarici açardir.

Ümumi şəkli aşağıdaki kimidir:

CONSTRAİNT constraint_name PRİMARY KEY\UNİQUE KEY\NOT NULL\REFERENCES foreign_table[foreign_field1], foreign_field2, …])]

Cədvəllər yaradilarkən bəzi sahələrə daxil olunacaq qiymətlər ilə əlaqəli məhdudiyyətlər qoymaq məcburiyyətində qala bilərik. Belə hallarda Constraintlər istifadə edilir.

Constraintlər əslində İNDEX-lərə oxşayirlar, amma indexlərdən fərqli olaraq 1 cədvəl üstündə təsirli olmaya bilər. Xüsusilə xarici açar məcburedici əlaqəli verilən daxili üçün olduqca təsirli

bir məcburedicidir. Ancaq bir Foreign Key təyini edə bilmək Foreign Key xarici açarin əsas cədvəlində ilkin açar olması lazımdır.
Misal4.

Create Table Kitab (kitabNo intNot Null, kitab adı Varchar(63), İSBNNO Varchar(15), səhifəsayi İnt, kitabxülasəsi Varchar(255))
Və ya
Create Table qiymət(qiymətNo İntNotNull, KitabNo İntNotNull, ÜzvNo İntNotNull, vermətarixi DateTime NotNUll, verməmüddəti İntNotNull, gəldimi Bit)

CƏDVƏL SİLMƏ

Bir cədvəli daxil edərkən imtina etsək:
DROP Table cədvəl_adi
Cədvəldə dəyişiklik etmə, bir cədvələ sütun əlavə etmək və ya silmək üçün
ALTER TABLE istifadə edilir.
ALTER TABLE cədvəl_adı
{Add{Column sahə sahə_tipi[(ölçü)[Not Null][Constraint index] constraint çoxlu index}
Drop{Column sahə/constraint constraint_adı}
}
Misal 5.
Kitab cədvəlinə kitab əvəzi adında yeni bir tam tipdən sütun əlavə edək.
Alter Table Kitab Add Kitabevezi İnteger;
Misal6. Bu sütunun boş olmamasini istəsək,
Alter Table kitab Alter Column Kitabevezi İnteger NotNull;yazmalıyıq.
Misal7.
Bu sütunu silmək üçün Alter Table Kitab Drop Column Kitabevezi yazılır.

İNDEKSLƏR

Kitabxanada kitablar müəyyən bir qaydayla (məsələn, əlifba sırasına görə) düzülsə axtarış asanlaşar. Verilənlər Bazasında da indexlərdən istifadə edərək, kitabları daxil olunduqları ardıcıllıqla yox, müəyyən bir qaydayla düzmək olar.
İndekslərin 3 funksiyası var:
1. Tək indekslər verilən əlaqələrini və verilən tamlığını təmin edən ilkin açar sahələr yaratmaqda istifadə edilir.
2. İndeks olan sahənin qiymətinə görə bir qeydin qeydlər arasındaki yerini göstərir.
3. Sorğularin yerinə yetirilmə müddətini qisaldırlar.
Create [Unique] index index_adi On cədvəl_adi(sütun_adi1[, sütun_adi2, …][Desc])
Misal8. Unique İndex indKitabNo On kitab(kitabNo)
Kimi yazdiqdan sonra iki fərqlim kitaba eyni nömrə vermək olmayacaq. Bir indeksi silmək üçün DROP İndex cədvəl_adi. index_adı istifadə edilir. Bir cədvəl və ya index sahə silindiyində indeks də avtomatik silinmiş olur.
Misal 9. İndeksi silək:Drop İNDEX Kitab. indKitabNo

SQL Server-də İndex sırasına görə verilənlərin fiziki olaraq yenidən sıralanmasını istəsək CREATE UNİQUE clustered İndex indKitabNo On kitablar(kitabNo) yazılır. Cədvəllər bir-biri ilə əlaqələndirilərkən index sahələr üzərində əlaqə qurulursa, daha sürətli sorğular əldə edilir.
View: Bəzən cədvəlləri olduqlarından fərqli göstərən filtrlərə ehtiyac duyulur. Bu cür funksi-yalar üçün ANSİ SQL VİEW istifadə etməyi məsləhət görülür. View-lar saxlanmiş sorğulardan ibarətdir. Əslində cədvəl kimi istifadə edilsələr belə cədvəl hal-hazirda yoxdur. View-lar bu vəzifələr üçün istifadə edilir:

*İstifadəçilərin bəzi kritik cədvəllərin sadəcə məlum sütunlarının və ya sətirlərinin görünməsi istəndikdə.
*İstifadəçilərin müxtəlif vahid çevrilmələrindən keçmiş dəyərləri görmək istədikdə.
*Hal-hazırda cədvəllərdə olan verilənlərin başqa bir cədvəl formatında verilməsini istədikdə.
*Çox kompleks sorğularin sadələşdirilməsi üçün.
Məsələn, kitab cədvəlindəki yalnız Texnologiyalar növündəki kitabların yer alacağı bir View bu şəkildə yaradilir:

Create View View_adi [(sütun1, sütun2, . . .)]Select cədvəl1. sütun_adi1 From cədvəl_adi;

Misal 10. Create View T kitabi (kitabNo, Kitabadi, İSBNNO, Səhifəsayi, xülasə) AsSelect kitabNo, kitabadi, İSBNNO, səhifəsayi, xülasə from kitab Where növü='Texnologiya';

SQL - VERİLƏN ƏMƏLİYYATLARI DİLİ (DATA MANİPLATİON LANGUAGE)

Verilən əməliyyatlari dili verilənin şablonu üzərində dəyişiklik etməz, sadəcə mövcud cədvəllərdəki məlumatları uyğun şəkildə seçmək(Select), yeni məlumatlar daxil etmək(İnsert), məlumatlar üzərində yeniləmə etmək (Update) və məlumatlarl silmək (Delete) üçün istifadə edilir. Verilən əməliyyat dili yalniz Verilənlər Bazasında olan məlumatlardan istifadə edir.

1. RESULT SET(RECORD SET, DATA SET)

Verilənlər Bazası İdarəetmə sistemlərində bir sorğu yerinə yetirildikdə, bir nəticə verir. Bir ResultSet birdən çox cədvəldən məlumat daxil edə bilir. Bir ResultSet-n daxili Verilənlər Bazası proqramlari ilə inkişaf etdirilərkən, ADO obyektlərindən RecordSet daxilinə köçürülür və verilənlər proqram içərisində bu obyekt vasitəsi ilə idarə edilir.

2. SELECT

Ən sadə olaraq cədvəldəki bütün qeydləri seçmək üçün:
Select sahə1[, sahə2, sahə3, . . . /*] From cədvəl_adi;yazisi istifadə edilir.
Misal1. Kitab cədvəlindəki bütün məlumatlari seçək:
Select * From kitab;
SELECT VƏ WHERE-bəzən müəyyən şərtləri ödəyən məlumatlar lazim olur. Bu zaman Where-dən istifadə olunur:
SELECT sahə1[sahə2, sahə3, . . . /*] From cədvəl_adi Where şərt1[AND şərt2[OR şərt3[NOT şərt4]]];
Misal1. kitabNo 12-dən böyük olan kitablari seçək:
Select * From kitab Where kitabNo>12;

RİYAZİ MÜQAYİSƏ İŞARƏLƏRİ:

Şərtlərdə verilən hal riyazi müqayisə işarələrindən istifadə edilərək daxil olunur:
=-bərabərdir.
>-böyükdür.
<-kiçikdir.
>=-böyük bərabərdir.
<=-kiçik bərabərdir.
<>-fərqlidir.
!=-bərabər deyildir.
Like-oxşar.

MƏNTİQİ İŞARƏLƏR.

Birdən çox seşmə əməliyyati aparilarkən AND, OR, NOT məntiqi operatorlardan istifadə edilir. Məntiqi funksiya cədvəlləri aşağidaki şəkildədir:
AND-şərtlərdən hər ikisini də təmin edir.

OR-şərtlərdən ən az birini təmin edir.
NOT-şərti təmin etməyən məlumatlari verir.
Bu funksiyalardan başqa XOR və XNOR fiunksiyalari da bəzi VBİS-lər tərəfindən dəstəklənir. Ancaq bu iki əmr nadir istifadə olunur.
X(şərt)Not X(şərtin tərsi)
0 1(seçilir)
1 0(seçilməz)

Kitab No	Kitab adı	İSBNNO	Səhifə sayı	Kitab xülasəsi
1	Visual Basic. Net	0-672-32203-X	204	Visual Basic mövzusunda ilkin məlumatlar
2	Təqlid və Hipnoz ilə öyrənmə texnologiyasi	975-6700-15-7	274	Öyrənmə və imtahanda hipnoz texnologiyasi
3	Yatirim plani Yapma	975-381-263-9	74	Yatirim üçün hardan başlamali
4	İş başinda duygusal zeka	975-434-224-5	447	Duygusal zekanin üstünlükləri
5	Həyat yolunda çətin-liklərlə mübarizə	975-362-135-3	119	Həyat yolunda çətinliklərlə mü-barizə

Select *From kitab Where kitabNo<3 AND səhifəsayi>200 OR kitabadi='Visual Basic. Net';
DİSTİNCT-birdən çox təkrarlanan məlumat çevirən Select funksiyası da hər bir məlumat tək olaraq yer almasına istəyiriksə, Distinct istifadə edilir.
Misal 2. Üzv cədvəlindəki adlarin tək bir siyahısını almaq istəyirik:
Select Distinct adi From üzv;
Üzvlərimizdən eyni olanların adı sadəcə bir dəfə gələcək.
İN-bir verilənin çoxluğunu məlum bir sahə ehtiva edən məlumatları tapmaq istədikdə, İN ifadəsindən istifadə olunur.
Misal 8. 1, 5, 6 nömrəli kitablarin borclarını görmək üçün
SELECT *From borc Where kitabNo=1 or kitabNo=5 or kitabNo=6; yerinə
Select * From borc Where kitabNo İN(1, 5, 6); istifadəsi daha asandır.
ANY, SOME, ALL-bəzi iç-içə sorğularda SOME, ANY və ya ALL ifadələri ilə kənardakı Select ifadəsinin seçəcəyi qeydlər uyğun kriterilərinə görə istifadə edilə bilər.
EXİSTS, NOT EXİSTS
EXİSTS istifadə olunduqda, kənardakı sorğuda bir və ya daha çox qeyd olar-sa, kənardakı sorğu tətbiq olunur. Heç bir qeyd olmazsa, kənardakı sorğu tətbiq oluna bilməz. NOT EXİSTS isə içəridəki sorğunun nəticəsində 0 qeyd olarsa, kənardakı sorğunun tətbiq olunması üçün istifadə olunur.
Misal3.
5 nömrəli kitab borc verilmişsə kitab nömrəsinin və kitab məlumatlarını seçək:
Select kitabNo, kitabAdi From kitab Where EXİSTS (select * from borc Where kitabNo=5) AND kitabNo=5;
UNİON(birləşdirmə):UNİON əmri , iki Select sorğusunun nəticəsini və ya iki cədvəli yalniz bir nəticə halinda ala bilmək üçün istifadə edilir. Bunun üçün iki select ifadəsinin bərabər sayda və verilən tipində sütundan yaranan (bərabər dəyər) nəticələr verməsi lazimdir.
İstifadəsi aşağidaki şəkildədir.
1. Select ifadəsi. . . .
UNİON

Misal 4. Sistemə daxil olanlarin bir qismini (müəlliflər və üzvlərin adlarini) bir siyahida görmək üçün yazaq.
Müəlliflərin ilk ikisini seçmək üçün:
Select adi, soyadi
From müəllif Where müəllifnNo<3;
Üzvlərin ilk ikisini seçək:
Select adi, soyadi
From üzv Where üzvNo<3;
Select adi, soyadi
From müəllif Where müəllifNo<3
UNİON select adi, soyadi
From üzv Where üzvNo<3;

KƏSİŞMƏ TAPMA:

İki cədvəlin və ya iki select sorğusunun kəsişməsini tapmaya imkan verir.
Misal 5. Hansi müəllif adlari eyni anda üzv adı olaraq da var? Select adl
From müəllif Where exists (Select * from üzv Where müəllif_adi=üzv adi)

EXPECT(FƏRQ TAPMA)

İki select sorğusu nəticəsi və ya iki cədvəl arasindaki fərqi tapmağa deyilir. Bu funksiya üçün də bir çox üsul istifadə edilə bilər. NOT İN, NOT EXİSTS bunlardan ikisidir.
Misal6. Hansi kitablar borc verilməyib?
Select *From kitab
Where kitabNo NotİIn(
Selcet kitabNo From borc Where gəldimi=0
);

BETWEEN. . . . AND. . . .

Bir araliq içində sorğu əldə etmək üçün betWeen-aşaği sərhəd, AND-yuxarı sərhəd şəklindəki yazıdan istifadə edilir.
Misal7. Səhifə sayi 300 ilə 500 arasındakı kitabların siyahısını almaq istədikdə
Select* From kitab Where səhifəsayı>300 and səhifəsayı<500;
yerinə
Select * From
Where səhifə sayı betWeen 300 and 500 ;
də yaza bilərik.

SİMVOL UYĞUNLUĞU

Simvol ifadələrin (mətn və tarix tipli verilənlər) araşdırılması da rəqəmlərlə eynidir.
Misal 8.
Select * From kitab
Where kitabadi='Önümüzdəki yol';
istifadə edilir.

LİKE:

Simvol ifadələrin necə yazildiğini və daxilinin tam olaraq nə olduğunu bilmək çətin olduğu üçün bəzən seçimi genişləndirmək üçün = yerinə LİKE ifadəsindən istifadə olunur.
Misal9.
Adi 'Önümüzdəki Yol' olan kitabi LİKE əmri ilə seçək:
Select *
From Kitab Where kitabadi LİKE 'Önümüzdəki Yol';

JOKER SİMVOLLAR

Seçimi bir az genişləndirək və adinda 'Yol' olan kitablarin siyahısını əldə edək:Bu halda Joker simvollardan istifadə etmək lazimdir. '%', '-', və ya '?'simvollarina joker simvollar deyilir.

Misal 10. Adinda yol olan kitablarin siyahısını əldə edək:

```
Select *
From kitab Where kitabadi     LİKE '%Yol%';
```

NULL QARŞİLAŞDİRMA

Bəzən bir sahəyə aid olan məlumatin daxil edilib-edilməməsi məcburiyyətində qala bilərik. Null qarşilaşdirma zamani FSNULL ifadəsi istifadə edilir. Boşluq olmayan sahələr üçün qarşilaşdirma zamani İS NOT NULL ifadəsi istifadə edilir.

```
Select sahələr
From cədvəl_adi
Where sahə1 is [Not]Null;
```

Misal11.

Tsb nömrəsi olmayan kitablarin siyahısı:

```
Select   *
From kitab  Where  İSBNNO    İSNULL;
```

3. ORDER BY

Select funksiyasi ilə əldə edilən nəticənin bir sahəyə görə siralanmasi üçün order by-dan istifadə olunur. Ümumi şəkli aşağidaki kimidir:

```
Select sahələr
From cədvəl_adi
Where şərtlər
Order  By sütun1[Desc|ASC], sütun2[Desc|ASC], . . . ];
```

ASC-artan siralamaq üçün, Desc-azalan siralamaq üçün istifadə olunur.

4. GROUP BY

Bəzən verilənləri qruplaşdiraraq əməliyyat aparmaq lazim olur. Çoxluq funksi-yalari:SUM(sütun_adi):İstənilən bir sütundaki dəyərlərin cəmini verir. Say tipində olan verilənlər üçün ödənilir.

Sahəyə qoşma ad(ALİAS)

Bəzi hallarda sorğu nəticəsində yer alan bir sahənin adini fərqli bir ad olaraq istifadə edə bilərik. Bu cür hallarda sahə adindan sonra As ifadəsi gəlir və qoşma ifadə verilir.

```
Select Sum(səhifəsayi) AS Cəm
From kitab;
```

AVG(sütun_adi)-istənilən bir sütun üçün qiyməti hesablamaq üçün istifadə edilir. Say tipli verilənlər üçün ödənilir.

Max(sütun_adi)-Referans sütuna görə ən böyük dəyəri tapmaq üçün istifadə edilir. Bit tipdən sahələrlə birlikdə istifadə edilməz.

Min(sütun_adi)-Referans sütuna görə ən kiçik qiyməti tapmaq üçün istifadə edilir.

Bit tipdən sahələrlə birlikdə istifadə edilə bilməz.

VARİANCE(sütun_adi)-Qrup variasiyasini tapir.

Count(*)-məlumatlarin hamisini saymaq üçün istifadə edilir. Say tipində olmasina ehtiyac yoxdur.

Count(sütun_adi)-Referans sütuna görə cəmi neçə məlumat olduğunun tapir.

Count Distinct(sütun_1):sütun 1-də neçə fərqli dəyər təkrarlandiğini sayir.

5. JOİNİNG

Əlaqəli Verilənlər Bazasının əsasinda birdən çox cədvəllər üzərində birlikdə əməliyyat apara bilmək durur. Bunun sayəsində verilənlərin təkrarlanmasinin qarşisi alinir və nəticədə verilənlərin idarəsi asanlaşir. İki cədvəli birlikdə tətbiq etməyin ən asan yolu, əsas cədvəldəki ilkin açar ilə ikinci cədvəldəki xarici açari bir-birinə bağlamaqdir.

Select cədvəl1. sahə1|cədvəl2. sahə1. [cədvəl1. sahə2. , . . .]
From cədvəl1, cədvəl2
Where cədvəl1. ilkin açar=cədvəl2. xarici açar[AND digər şərtlər]

CƏDVƏLƏ QOŞMA AD

Birdən çox cədvəl üzərində əməliyyat aparilarkən tez-tez cədvəl adlarindan istifadə edilir. Bu da bir adlar çoxluğu olaraq qarşimiza çixir. Bunun qarşisini almaq üçün cədvəllərin yerinə qoşma ad verib o adlari istifadə edə bilərik.

BİR CƏDVƏLİ ÖZÜ İLƏ ƏLAQƏLƏNDİRMƏ(SELF-JOİN)

Bir cədvəli özü ilə əlaqələndirərkən sorğulamaq lazim gəldiyində eyni cədvələ fərqli iki qoşma ad verib, fərqli iki cədvəl üzərində əlaqəli cədvəl üzərində sorğu edirmiş kimi əlaqəli aparila bilər. Bu, verilənlərin doğruluğunu yoxlamaq üçün istifadə olunur.

LEFT[OUTER]JOİN

İki cədvəl arasinda əlaqəli sorğu aparilarkən, Left Outer Join istifadə edildikdə 1 cədvəldəki bütün qeydlər gətirilir. 2-ci cədvəldə isə sadəcə bir əlaqəyə görə uyğun qeydlər sag tərəfə əlavə olunur. Ümumi şəkli:

. . . . From cədvəl1
Left JOİN cədvəl2
ON cədvəl1. sahə1
Qarşilaşdirma cədvəl2. sahə2;
Kimidir.

RİGHT(OUTER)JOİN

İki cədvəl arasinda əlaqəli sorğu aparilarkən Right outer join istifadə edildiyində ikinci cədvəldəki bütün qeydlər gətirilir. Birinci cədvəldə isə sadəcə əlaqəyə görə uyğun qeydlər sağ tərəfə əlavə olunur. Ümumi şəkli:

. . . From cədvəl1. Right Join cədvəl2 ON
cədvəl1. sahə1 qarşilaşdirma cədvəl2. sahə2;

İNNER JOİN

İNNER JOİN ən çox istifadə edilən cədvəl birləşdirmə üsuludur.

From cədvəl1 İNNER JOİN cədvəl2 ON cədvəl1. sahə1 qarşilaşdirma cədvəl. sahə2;
Join əməliyyati birdən çox cədvəl üzərində də aparila bilər.

6. İNSERT

Bir cədvələ SQL ilə məlumat daxil etmək üçün İNSERT ifadəsi istifadə olunur.

İNSERT İNTO cədvəl_adi(sahə1[, sahə2, . . .])
Values (dəyər1[, dəyər2, . . .])
Misal 12.
Qaytarilmamiş kitabimizi almaq üçün bir cədvəl yaratmaq lazimdirsa, bunlari seçmə əməliyyatinin nəticəsini yeni cədvələ daxil etmə şəklindədə edə bilərik:

İNSERT İNTO qaytarilmamiş kitab(kitabNo, kitabAdi, əvəzi)
Select kitabNo, kitabAdi, əvəzi From kitab
İNTER JOİN əvəz ON
əvəz. kitabNo=kitab. KitabNo

Where əvəz. gəldimi=0;

7. UPDATE

Bir qeyd üzərində dəyişiklik etmək lazim gəldiyində UPDATE ifadəsi istifadə edilir. Ümumi şəkli aşağidaki kimidir:

16

Update cədvəl_adi
Set sahə1=təzə qiymət1[, sahə2=təzə qiymət1, . . .]
[Where şərt]
Misal 13.
UpDate kitab
Setkitabəvəzi=kitabəvəzi*1, 18;

8. DELETE

Cədvəldə olan qeydlərin hamisini və ya bir qismini silmək üçün DELETE ifadəsi istifadə edilir. Ümumi şəkli aşağidaki kimidir:

Delete
From cədvəl_adi
[Where şərt];
Misal 14.
Soyadi 'Zamanov' olan üzvləri silmək
DELETE
From üzv
Where soyad LİKE '%Zamanov%'

9. ÜMUMİ SQL FUNKSİYALARİNİN İSTİFADƏSİ

SQL-də çox funksiya yoxdur. Bununla əlaqədar bəzi Verilənlər Bazası İdarəetmə Sistemləri tərəfindən əlavə funksiyalar da əlavə edilə bilir.

Tarix-zaman funksiyalari:

Getdate():Microsoft və Sybase əsasli sistemlərdə sistem tarixini tapir.

Date(): Getdate kimi istifadə olunur.

Sysdate():eyni istifadə olunur.

NoW():Ms əsasli sistemlərdə anliq tam zamani (gün, ay, il, saat, dəq, san) verir.

CƏBRİ FUNKSİYALAR

Mod(say, mod**)**-Sayin mod dəyərinə bölünməsindən alinan qaliği verir. Oracle-də istifadə olunur.

ABS(say**)**-Sayin mütləq qiymətini verir.

Cos(), Sin(), Tan(), CosH(), SinH(), TanH()-triqonometrik funksiyalarin eynisidir, verilən dəyər RADİAN tipində olmalidir.

LN(say)-sayin e tərtibli loqarifmini verir.

Log(say, tərtib)-İstənilən sayin verilmiş tərtibdən loqarifmasini verir.

PoWer(say, üstü)-Bir sayin qüvvətini verir.

Sign(say)-Sayin işarəsini verir. Mənfi üçün -1, müsbət üçün +1, sifir üçün 0 verilir.

Sqrt(say)-Bir sayin kökaltini verir.

SİMVOL ƏMƏLİYYAT FUNKSİYALARİ

CHR()-Bir ASCİİ kodunun qarşiliğini verir.

CONCAT(mətn1, mətn2)-İki və ya daha çox simvol ifadəni uc-uca əlavə etmək üçün istifadə edilir.

İNİTCAP(simvol)-İlk hərfi böyük edir.

REPLACE(simvol, ifadə, yerinə yazilacaq ifadə)
İfadələri tapib dəyişdirir.
LOWER(mətn), UPPER(mətn)-uyğun olaraq bütün hərfləri böyük və ya kiçik etmək üçün istifadə olunur(Oracle)
LCASE(mətn) və UCASE(mətn)-bütün hərfləri kiçik və ya böyük etmək üçün istifadə olunur(MS).
LTRİM(mətn), RTRİM(mətn), TRİM(mətn)-soldaki, sağdaki və ya hər iki tərəfdəki boşluqlari atmaq üçün istifadə olunur.
LENGHT(mətn)-bir mətnin uzunluğunu verir(Oracle)
LEN(mətn)-Mətnin uzunluğunu verir(MS)
ÇEVİRMƏ FUNKSİYALARİ
CONVERT(verilən tipi[(uzunluğu)], sütun_adi, format);
Bir verilənin bir formatdan başqa bir formata çevrilməsi üçün istifadə edilir.
Misal 11.
Üzvlük nömrəsinin sonu 1 olan üzvlərin siyahısını əldə edək:
Select üzvNo, adi, soyadi
From üzv
Where Convert (varchar(4), üzvNo) Like '%1'
Cast (sütun_adi AS dəyişdiriləcək verilən tipi (N));
Bir sütunun ya da verilənin tipini başqa bir tipə dəyişdirmək üçün də istifadə edilir.
Misal 15.
Select Cast (müəllifNo As varchar(4))+' '+''+soyadi
From müəllif
CƏBRİ İŞARƏLƏR SQL-də hər yerdə olduğu kimi istifadə olunur:
+-toplama
--cixma
*-vurma
/-bölmə
%-mod