BMB202. Veritabanı Yönetimi

Ders 5.
İlişkisel Cebir ve SQL

Erdinç Uzun
NKÜ Çorlu Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Dersin Planı

- İlişkisel Cebir
- SQL'e Giriş

İlişkisel Cebir (Relational Algebra)

- İlişkisel cebir, bir ilişkisel sorgulama dilidir.
- Bir veya iki ilişkiyi girdi olarak alıp sonuç olarak yeni bir ilişki üreten bir dizi işlemden oluşur.
- Örneğin, belirli kayıtların seçilmesi, belirli sütunların sorgulama sonucunda elde edilmesi gibi daha birçok işlem ilişkisel cebir ifadeleri kullanılarak kolayca ortaya konabilmektedir.
- İlişkisel cebir bir veritabanı sorgulama dilidir.
- Ancak bu sorgulamalar sadece biçimsel olarak yapılır. İlişkisel Cebrin sorgulama dili için bir yorumlayıcı ya da bir derleyici yoktur.
- İlişkisel cebir ifadeleri daha sonraki aşamada bir sorgu dili ile (örneğin SQL) komutlara dönüştürülmektedir.

İlişkisel Cebir

İlişkisel cebrin temel işlemleri:

- Kayıtlar arasında seçim,
- Sütunların seçimi,
- Kartezyen çarpım,
- Birleştirme işlemi,
- Kesişme işlemi,
- Fark,
- Doğal birleştirme,
- Bölme işlemi

Seçme İşlemi

- Belirli bir ilişkiden bazı kayıtların seçilerek ortaya konulması işlemidir.
- σ işareti ile gösterilmektedir. Seçme işlemi şu şeklide tanımlanmaktadır.

$$\sigma_{Seçim\ Kriteri}(TABLO)$$

- Seçim işleminde karşılaştırma işleçleri olan =, >, <, ≠, ≤, ≥ ifadeleri de kullanılmaktadır. Ayrıca mantıksal operatörler olan “ve” için ^, “veya” için v işaretleri de kullanılmaktadır.

Seçme Örneği 1

İlçesi “Beşiktaş” olan Müşteriler

MÜŞTERİ

Müşteri adı	No	İlçe	Şehir	Bakiye
Ahmet	2520	Beşiktaş	İstanbul	100
Faruk	6345	Beşiktaş	İstanbul	150
Ali	6755	Şişli	İstanbul	250
Veli	3000	Kızılay	Ankara	100
Zeki	2521	Ulus	Ankara	500

$$\sigma_{\text{ilçe}=\text{"Beşiktaş"}}(\text{MÜŞTERİ})$$

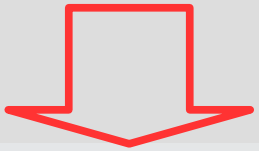
Müşteri adı	No	İlçe	Şehir	Bakiye
Ahmet	2520	Beşiktaş	İstanbul	100
Faruk	6345	Beşiktaş	İstanbul	150

Seçme Örneği 2

- MÜŞTERİ tablosunu yeniden göz önüne alalım. İlçesi "Beşiktaş" ve bakiye miktarı 100'den büyük olan müşterileri seçmek istiyoruz.

$$\sigma_{ilçe = "Beşiktaş" \wedge Bakiye > 100}(MÜŞTERİ)$$

Müşteri adı	No	İlçe	Şehir	Bakiye
Ahmet	2520	Besiktas	İstanbul	100
Faruk	6345	Beşiktaş	İstanbul	150
Ali	6755	Şişli	İstanbul	250
Veli	3000	Kızılay	Ankara	100
Zeki	2521	Ulus	Ankara	500



Müşteri adı	No	İlçe	Şehir	Bakiye
Faruk	6345	Beşiktaş	İstanbul	150

Projeksiyon / Atma İşlemi

- Belirli bir ilişkiden sadece bazı sütunları atmak için yapılan bir işlemdir.
- Bu işlem Π sembolü ile gösterilmektedir. Atma işlemi aşağıdaki şekilde tanımlanmaktadır

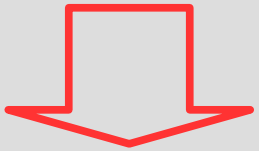
$$\Pi_{\text{Sütun İsimleri}}(TABLO)$$

Atma Örneği

- Müşteri Adı ve Şehrini görüntüleyen işlem.

$$\Pi_{\text{müşteri adı, şehir}}(\text{MÜŞTERİ})$$

Müşteri adı	No	İlçe	Şehir	Bakiye
Ahmet	2520	Beşiktaş	İstanbul	100
Faruk	6345	Beşiktaş	İstanbul	150
Ali	6755	Şişli	İstanbul	250
Veli	3000	Kızılay	Ankara	100
Zeki	2521	Ulus	Ankara	500



Müşteri adı	Şehir
Ahmet	İstanbul
Faruk	İstanbul
Ali	İstanbul
Veli	Ankara
Zeki	Ankara

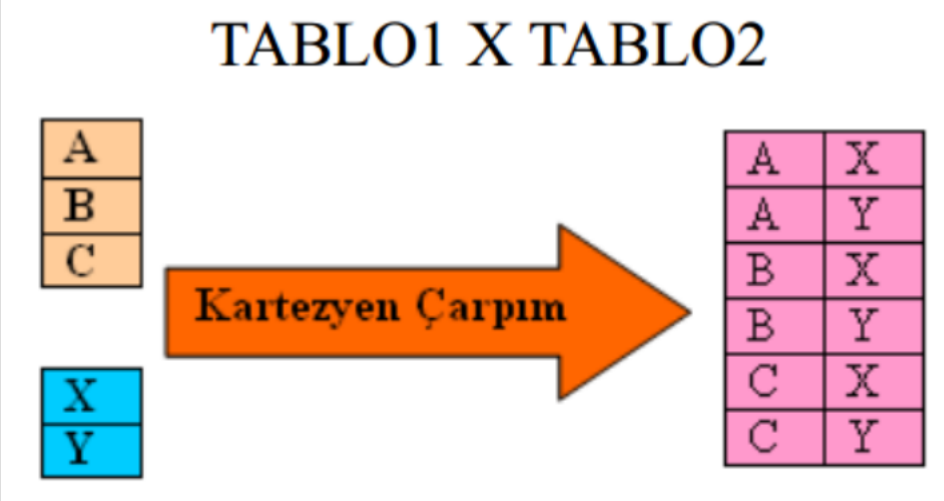
Atma Örneği

- Müşteri tablosunda bakiyesi 200 den büyük olan müşterilerin adı ve şehir bilgisi nedir?

$$\Pi_{Müsteriadı,Şehir}\left(\sigma_{Bakiye>200}\ (MÜŞTERİ)\right)$$

Çarpma

- Belirli bir ilişkiden mümkün olabilecek tüm ilişki çiftlerinin elde edilmesi ve tek bir ilişki biçiminde gösterilmesi için kartezyen çarpım kullanılmaktadır.
- Kartezyen çarpım X sembolü ile gösterilmektedir. Kartezyen çarpımın genel kullanım biçimi şöyledir.



Çarpma Örneği
ÖĞRENCİ ve DERSLER

ÖĞRENCİ

Öğrenci	Bölüm
Burak	Bilgisayar
Begüm	İktisat
Dilay	İktisat
Selin	İşletme
Seray	Hukuk

DERSLER

Ders	Saat
Matematik	3
İstatistik	2

ÖĞRENCİ × DERSLER

Öğrenci	Bölüm	Ders	Saat
Burak	Bilgisayar	Matematik	3
Begüm	İktisat	Matematik	3
Dilay	İktisat	Matematik	3
Selin	İşletme	Matematik	3
Seray	Hukuk	Matematik	3
Burak	Bilgisayar	İstatistik	2
Begüm	İktisat	İstatistik	2
Dilay	İktisat	İstatistik	2
Selin	İşletme	İstatistik	2
Seray	Hukuk	İstatistik	2

Çarpma Örneği

İktisat Bölümünde okuyan Öğrenci X Dersler

ÖĞRENCİ

Öğrenci	Bölüm
Burak	Bilgisayar
Begüm	İktisat
Dilay	İktisat
Selin	İşletme
Seray	Hukuk

DERSLER

Ders	Saat
Matematik	3
İstatistik	2

$$\sigma_{Bölüm="İktisat"}(ÖĞRENCİ \times DERSLER)$$

Öğrenci	Bölüm	Ders	Saat
Begüm	İktisat	Matematik	3
Dilay	İktisat	Matematik	3
Begüm	İktisat	İstatistik	2
Dilay	İktisat	İstatistik	2

Çarpma Örneği

"İktisat" bölümünde okuyan ve hem "Matematik" hem de "İstatistik" dersi

ÖĞRENCİ

Öğrenci	Bölüm
Burak	Bilgisayar
Begüm	İktisat
Dilay	İktisat
Selin	İşletme
Seray	Hukuk

DERSLER

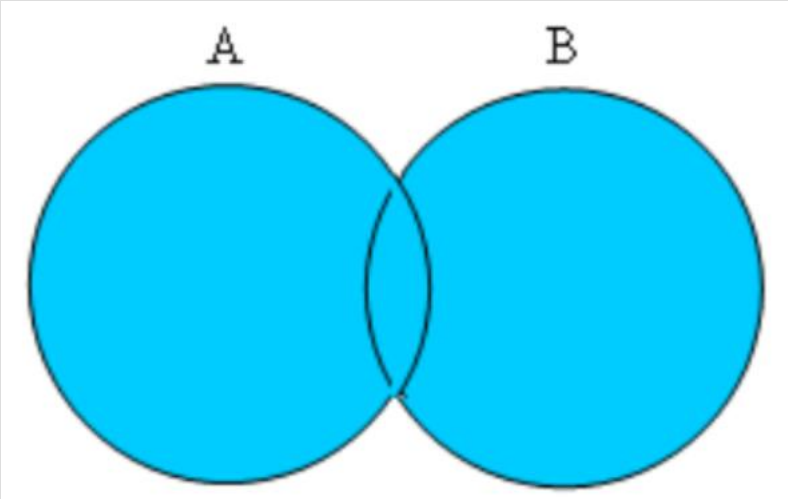
Ders	Saat
Matematik	3
İstatistik	2

$$\prod_{\text{Öğrenci, Ders}} \left(\sigma_{\text{Bölüm} = \text{"İktisat"}} \left(\text{ÖĞRENCİ} \times \text{DERSLER} \right) \right)$$

Öğrenci	Ders
Begüm	Matematik
Dilay	Matematik
Begüm	İstatistik
Dilay	İstatistik

Birleştirme

- Kurulan iki ilişkiden birinde veya her ikisinde birden bulunan kayıtların seçilmesi için yapılan bir işlem türüdür.
- Bu işlem Y veya U simgesi ile gösterilmektedir.



Birleştirme Örneği

- Bankanın "Beşiktaş" şubesinde mevduat ve/veya kredi hesabı bulunan kişileri seçmek ve sadece isimlerini sunmak için.

Müşteri	Bakiye	İlçe
Burak	50	Beşiktaş
Selin	125	Beşiktaş
Sezin	300	Ulus

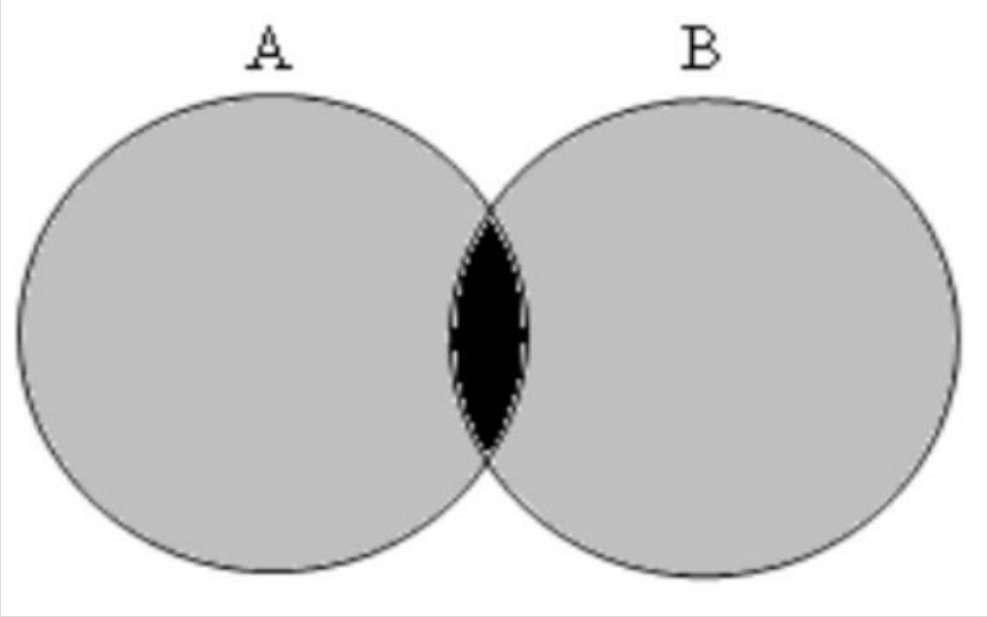
Müşteri	Bakiye	İlçe
Burak	100	Beşiktaş
Begüm	200	Şişli

$$\prod_{kredi.müşteri}\left(\sigma_{ilçe="Beşiktaş"}(KREDİ)\right) \cap \prod_{mevduat.müşteri}\left(\sigma_{ilçe="Beşiktaş"}(MEVDUAT)\right)$$

Müşteri
Burak
Selin

Kesişme

- Belirlenen iki ilişkiden birinde bulanan kayıtların belirlenmesi için kullanılan bir işlemdir.
- Bu işlem I veya ∩ simgesi ile gösterilmektedir.



Kesişme Örneği

Bankanın "Beşiktaş" şubesinde hem mevduat hem de kredi hesabı olan müşteriler

Müşteri	Bakiye	İlçe
Burak	50	Beşiktaş
Selin	125	Beşiktaş
Sezin	300	Ulus

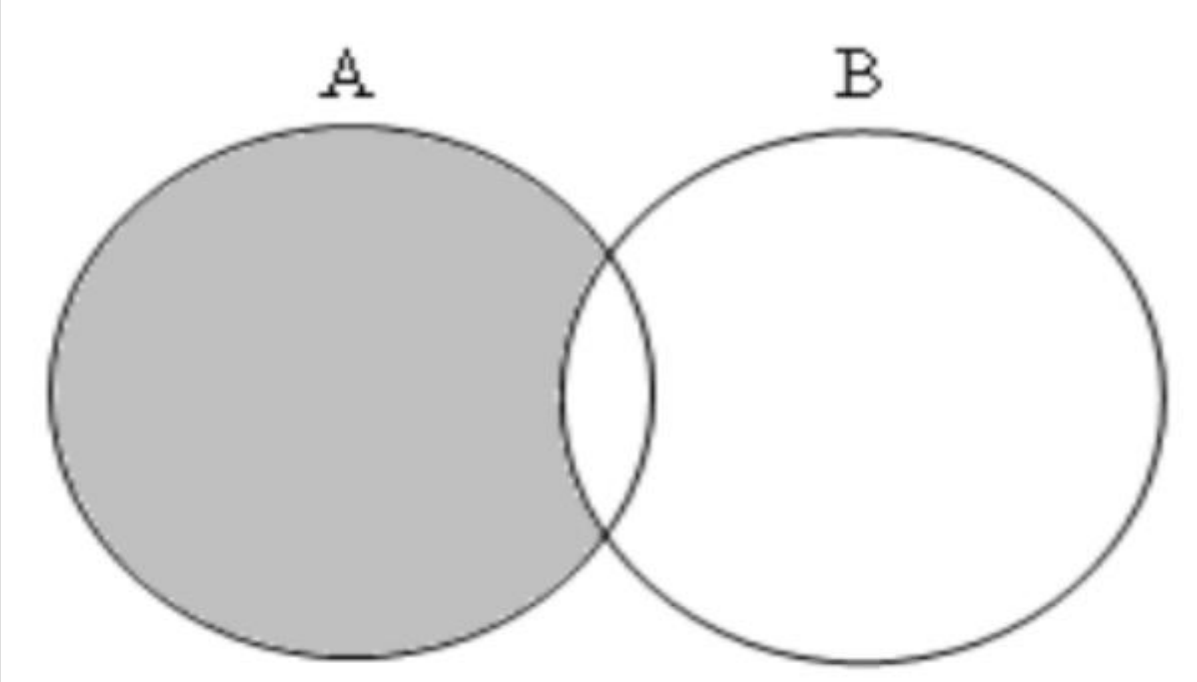
Müşteri	Bakiye	İlçe
Burak	100	Beşiktaş
Begüm	200	Şişli

$$\prod_{kredi.müşteri}\left(\sigma_{ilçe="Beşiktaş"}(KREDİ)\right) \cap \prod_{mevduat.müşteri}\left(\sigma_{ilçe="Beşiktaş"}(MEVDUAT)\right)$$

Müşteri
Burak

Fark

- Verilen iki ilişkide birinde bulunup diğerinde bulunmayan kayıtların gösterilmesi için kullanılan bir işlemdir.
- (-) işareti ile gösterilmektedir.



Fark Örneği

Bankanın "**Ulus**" şubesinde mevduatı olup kredisi olmayan müşterilerin isimleri

MEVDUAT

Müşteri	Bakiye	İlçe
Burak	50	Beşiktaş
Selin	125	Beşiktaş
Sezin	300	Ulus

KREDİ

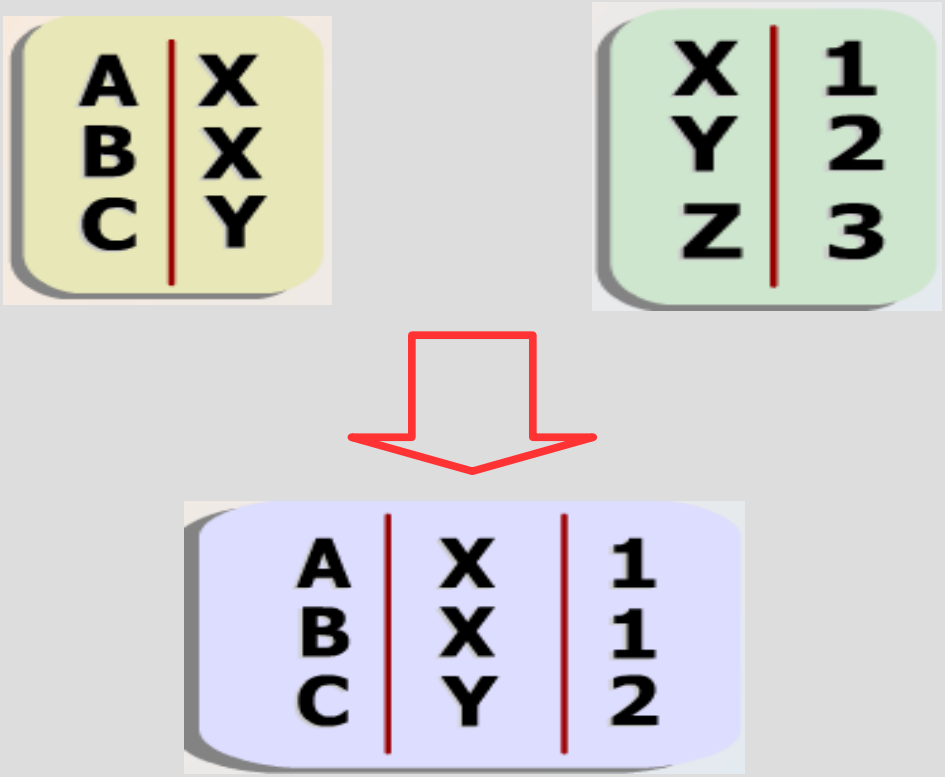
Müşteri	Bakiye	İlçe
Burak	100	Beşiktaş
Begüm	200	Şişli

$$\prod_{mevduat.müşteri}\left(\sigma_{ilçe="Ulus"}(MEVDUAT)\right)-\prod_{kredi.müşteri}\left(\sigma_{ilçe="Ulus"}(KREDİ)\right)$$

Müşteri
Sezin

Doğal Birleştirme

- (A,B) ve (B,C) niteliklerine sahip iki ayrı ilişkinin, (A,B,C) niteliklerine sahip tek bir ilişki haline dönüştürülmesine Doğal Birleştirme denir.
- Bu ⋈ işlem işareti ile gösterilir.



Doğal Birleştirme Örneği

Kredi hesabı olan ve aynı ilçede oturan tüm müşterilerin isimlerini

KREDİ

Müşteri	İlçe	İl
Burak	Beşiktaş	İstanbul
Begüm	Şişli	İstanbul
Selin	Bakırköy	İstanbul
Sezin	Şişli	İstanbul
Dilay	Kızılay	Ankara

MÜŞTERİ

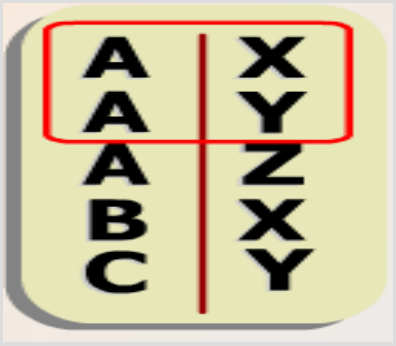
Müşteri	Bakiye	İlçe
Burak	100	Beşiktaş
Begüm	200	Şişli

$$\Pi_{kredi.adı,\ kredi.ilçe}\left(KREDİ \bowtie MÜŞTERİ\right)$$

Müşteri	İlçe
Burak	Beşiktaş
Begüm	Şişli

Bölme

- İkili ve birli; iki ayrı ilişkiyi karşılaştırarak, birli olan ilişkiye eş olan ikinci ilişkinin değerlerinden oluşan, yeni bir ilişki oluşturulabilir.
- Bu tür bir işlem bölme işlemidir ve (:) işareti ile gösterilir.



Hem X hemde Y'de eşleşen

Bölme Örneği

İSTANBUL ilindeki bütün şubelerinde mevduat hesabı olan müşteriler

Adım 1

ŞUBE

Şube	İl
Beşiktaş	İstanbul
Şişli	İstanbul
Yayla	İstanbul
Kızılay	Ankara
Ulus	Ankara

$$r_1 = \prod_{\text{Şube adı}} \left(\sigma_{\text{il}=\text{"İSTANBUL"}} (\text{ŞUBE}) \right)$$

Şube adı
Beşiktaş
Şişli
Yayla

Bölme Örneği

İSTANBUL ilindeki bütün şubelerinde mevduat hesabı olan müşteriler

Adım 2

MEVDUAT

Müşteri	Bakiye	Şube
Burak	50	Beşiktaş
Selin	125	Şişli
Sezin	300	Ulus
Dilay	200	Kızılay

$$r_2 = \prod_{\text{Müşteri, Şube}} (MEVDUAT)$$

Müşteri	Şube
Burak	Beşiktaş
Selin	Şişli
Sezin	Ulus
Dilay	Kızılay

Bölme Örneği

İSTANBUL ilindeki bütün şubelerinde mevduat hesabı olan müşteriler

Adım 3

Şube adı
Beşiktaş
Şişli
Yayla

Müşteri	Şube
Burak	Beşiktaş
Selin	Şişli
Sezin	Ulus
Dilay	Kızılay

$$r_2 : r_1$$

Müşteri
Burak
Selin

SQL (Structured Query Language)

- SQL çok yüksek seviyeli bir dildir.
- İngilizce bilen herkes bu dili kolayca öğrenebilir. Programlama dillerine göre öğrenilmesi çok daha kolaydır. Çünkü programlama dillerindeki gibi işlemin “nasıl yapılacağı” değil, işlemde “ne yapılacağı” yazılır.
- Birçok VTYS, yazılan sorguları en iyi şekilde işlemek için sorgu en-iyileştirme (query optimization) mekanizmaları kullanır.

DML, DDL, DCL

- Sorgulama için sadece SELECT komutu kullanılsa da; SQL içinde başka komutlar da yer alır. Bu komutlar işlevlerine göre sınıflandırılmış ve başka alt-diller oluşturulmuştur:
  - DML (Data Manipulation Language): Tablolara veri girme, var olan veriyi güncelleme ve veri silme işlemleri için kullanılan komutları içerir.
  - DDL (Data Definition Language): Tablo, Kullanıcı gibi nesneleri yaratmak için kullanılan komutları içerir.
  - DCL (Data Control Language): Kullanıcılara çeşitli yetkiler verme, yetkileri geri alma gibi işlemleri gerçekleştirmek için kullanılan komutları içerir.

Sorgulama İşlemleri

- SQL'de sorgulama işlemleri, SELECT deyimi yardımıyla yerine getirilir.
- SELECT deyimi temel olarak üç farklı işlemi yerine getirmek için kullanılır:
  - Seçme İşlemi
  - Atma İşlemi
  - Birleştirme İşlemi

Select

- SELECT deyimi en basit biçimde şu şekilde ifade edilmektedir
  - SELECT [DISTINCT] { * | sütun, ...} FROM tablo;
- Tanım içinde bazı SQL anahtar kelimelerine yer verilmektedir. Bu anahtar kelimeler, SQL 'in kendi özel kelimeleridir ve aynen bu şekilde ifade edilmelidir.
  - SELECT SQL'in sorgulama deyimidir.
  - FROM Hangi tablonun sorgulanacağını ifade eder.
  - DISTINCT Çift kayıtları önleyen anahtar kelimedir.

Select

- [ ] Kullanılması zorunlu olmayan SQL sözcükleri, bu işaretler arasında tanımlanır.
- Altı çizili ve italik olan bu ifadeler, kullanıcı tarafından verilen isimleri ifade etmektedir. Bunlar SQL sözcüğü değildir.
- {..|.. } Bu biçimde gösterilen ifadeler, birden fazla seçeneğin varlığını ve bu seçeneklerden birinin mutlaka seçilmesi gerektiğini ifade eder. Seçenekler birbirlerinden | işareti ile ayrılmaktadır.
- * Tek bir sütunu değil, tüm sütunları ifade eder.

Örnek

- Access üzerinde Personel adında bir tablo oluşturalım. Aşağıdaki özellikleri içersin.
  - PERSONEL_NO (PK)
  - AD
  - SOYAD
  - GOREV
  - MAAS
  - GIRIS_TARIHI
- Tabloya 10 civarında kayıt ekleyelim.

Select Örneği

- Sorgu bölümüne aşağıdaki SQL ifadesini yazalım.
  - SELECT * FROM PERSONEL
- Sonuçları İncele...

Where

SELECT sütun [yada sütunlar]

FROM tablo [yada tablolar]

WHERE seçim kriteri

- SELECT ifadesinden sonra * kullanılırsa tüm nitelikler (sütunlar) seçilir.
- Kriter verilmezse “WHERE” sözcüğü de yazılmaz. Bu durumda tüm kayıtlar (satırlar) seçilir.
- Personel tablosundaki tüm kayıtların tüm nitelikleri aşağıdaki SQL cümlesi ile gösterilir:

Where kriter verme

- Eğer tüm kayıtların değil de sadece belirli kayıtların görüntülenmesi istenirse WHERE ile kriter verilir.
- Aşağıdaki SQL cümlesi maaşı 1800 TL'nin üzerinde olan personelin adı ve soyadını ekranda gösterir:
  - SELECT AD, SOYAD

    FROM PERSONEL

    WHERE MAAS > 1800

Kriterlerde kullanılan işleçler

- Programlama dillerinde kullanılan aritmetiksel karşılaştırma işleçleri (<, <=, >, >=, =, <>) ve mantıksal işleçler (AND, OR, NOT) SQL dilinde de kriter verirken kullanılır.
- Aşağıdaki SQL cümlesi görevi müdür olan ve maaşı 5000 TL'den fazla olan personeli gösterir:

```
SELECT * FROM PERSONEL
WHERE GOREV = 'Müdür' AND MAAS > 5000
```

Programlama dillerinde olduğu gibi SQL'de de karakter türü veriler ile işlem yapılacaksa tek tırnak yada çift tırnak kullanılır.

Karakter türü verilerin karşılaştırılması

- Karakter türü veriler ile de büyüklük-küçüklük kıyaslamaları yapılabilir.
- Aşağıdaki SQL cümlesi adı N-Z arasında bir harf ile başlayan personeli gösterir:
  - SELECT * FROM PERSONEL WHERE AD > ‘N’
- Aşağıdaki SQL cümlesi adı E harfi ile başlayan personeli gösterir:
  - SELECT * FROM PERSONEL
  - WHERE AD > ‘E’ AND AD < ‘F’

Like İşleci

- Belirli bir karakter katarını barındıran verileri aramak için LIKE kullanılır.
- Önceki slaytta yer alan, adı E harfi ile başlayan personeli gösteren sorgu LIKE ile de yazılabilir:
  - SELECT * FROM PERSONEL WHERE AD LIKE ‘E*’
- Adresler şehir adı ile bitiyorsa, Edirne ilinde ikamet eden öğrencileri görmek için aşağıdaki sorgu kullanılabilir:
  - SELECT * FROM ÖĞRENCİLER

    WHERE ADRES LIKE ‘*Edirne’
- * yerine bazı VTYS’lerde % kullanılabilir.

Between İşleci

- İki değer arasında karşılaştırma yapmak için Between … And … işleci (… ile … arasında) kullanılabilir.
- Aşağıdaki SQL cümlesi maaşı 1000 ile 2000 TL arasında olan işçileri görüntüler:

```
SELECT * FROM PERSONEL
WHERE MAAS BETWEEN 1000 AND 2000
AND GOREVI = 'İşçi'
```

- Bu sorgu Between işleci kullanılmadan da yazılabilirdi:
  - SELECT * FROM PERSONEL WHERE MAAS >= 1000 AND MAAS <= 2000 AND GOREVI = 'İşçi'

In İşleci

- Bir listedeki değerler ile karşılaştırma yapmak için IN işleci kullanılır.
- 1000 ile 2000 TL arasında değil de sadece 1000, 1500 ve 2000 TL maaş alanları listelemek için aşağıdaki SQL cümlesi kullanılabilir:

```
SELECT * FROM PERSONEL WHERE MAAS IN
  (1000, 1500, 2000)
```

Tarihsel karşılaştırma

- Belirli bir tarihe eşit olan veya o tarihten büyük yada küçük olan verilerin aranması istenirse, tarih ay/gün/yıl biçiminde ve # karakterleri arasında yazılmalıdır:
  - SELECT * FROM OGRENCILER

    WHERE DOGUM_TARIHI BETWEEN

    #1/1/1989# AND #12/31/1989#
- VTYS'ye göre farklılık gösterebilir.

Distinct İfadesi

- Eğer tablonun bir alanında yer alan veriler içinde aynı olan veriler varsa SELECT ifadesinden sonra kullanılan DISTINCT ile bu tekrar eden verilerin sadece 1 defa görüntülenmesi sağlanabilir.
- Aşağıdaki SQL cümlesi farklı kayıtlardaki aynı adları her kayıt için tekrar göstermek yerine 1 defa gösterilmesini sağlar:

```
SELECT DISTINCT AD FROM PERSONEL
```

As ifadesi ve Sütun içeriği birleştirme

- Sütunların kendi ismi yerine AS ifadesi ile takma isim almaları sağlanabilir.
- İki yada daha fazla sayıda sütunun içeriğini birleştirmek için sütun isimleri arasında Access'te &, Oracle'da ise || işleçleri kullanılır.
- Aşağıdaki SQL cümlesi AD ve SOYAD sütunlarının içeriklerini araya bir boşluk karakteri ekleyerek birleştirir ve ISIM adlı bir sütun şeklinde gösterir.

```
SELECT AD & ' ' & SOYAD AS ISIM FROM
  OGRENCILER
```

Matematiksel İşlemler

- SELECT ifadesinden sonra bir sütunun bir matematiksel işleme tabi tutulması ve bu işlemin sonucunun gösterilmesi sağlanabilir.
- Tabloda aylık maaşları saklanan personelin yıllık maaşlarının görüntülenmesi istenirse, aşağıdaki SQL cümlesi kullanılabilir:

```
SELECT AD, SOYAD, MAAS * 12 AS YıllıkÜcret
FROM PERSONEL
```

NULL (boş) değerler ile ilgili işlemler

- Eğer bir kayıt, bazı alanları boş bırakılarak eklendiyse, matematiksel işlemlerde sorun çıkabilir (NULL, sıfır değeri ile aynı değildir).
  - 1500 * 12 + 0 işleminin sonucu 1800 iken,
  - 1500 * 12 + NULL işleminin sonucu NULL olacaktır.
- Karşılaştırma işlemlerinde de NULL ile = işleci kullanılmaz, IS kullanılır.
  - SELECT * FROM PERSONEL WHERE TC_NO IS NULL

Türkçe karakter kullanma

- Birçok VTYS, tablo ve nitelik isimlerinde Türkçe karakter kullanımına izin verir. Fakat sorgularda bazı sıkıntılara neden olabileceği için kullanılması tavsiye edilmez.
- Örneğin PERSONEL tablosundaki nitelik isimleri ADI ve SOYADI şeklinde büyük harfler ile verildiyse, bazı VTYS'ler "SELECT Adi, Soyadi FROM Personel" ifadesini, bazıları ise "SELECT Adı, Soyadı FROM Personel" ifadesini doğru kabul eder.

Alan ismi olarak kullanılmaması gereken kelimeler

- Bir öğrencinin not bilgilerini saklamak için “NOT” isminde bir alan yaratılırsa sorgularda bu alana göre kriter verilmek istendiğinde, İngilizcede “değil” anlamına geldiği için hata verecektir. “NOT” yerine “NOTU”, “VIZE”, “FINAL” gibi ifadeler tercih edilmelidir:
  - SELECT * FROM NOTLAR WHERE NOT >= 60 (Hatalı)
  - SELECT * FROM NOTLAR WHERE NOTU >= 60 (Doğru)
- Benzer şekilde SQL’e ait olan “SELECT”, “FROM”, “WHERE”, “ORDER”, … gibi ifadeler de alan ismi olarak kullanılmamalıdır.

ORDER BY

Sıralama

- Eğer görüntülenecek olan kayıtların belirli bir sütuna göre sıralı olarak görüntülenmesi isteniyorsa ORDER BY kullanılır.
- Sıralama yukarıdan aşağıya doğru artan sırada olacaksa ASC, azalan sırada olacaksa DESC kullanılır. Varsayılan sıralama şekli artan olduğu için ASC yazılmasa bile artan sıralama kullanılmış olur.
- Aşağıdaki SQL cümlesi PERSONEL tablosundaki kayıtları maaşa göre azalan sırada gösterir:
  - SELECT * FROM PERSONEL ORDER BY MAAS DESC