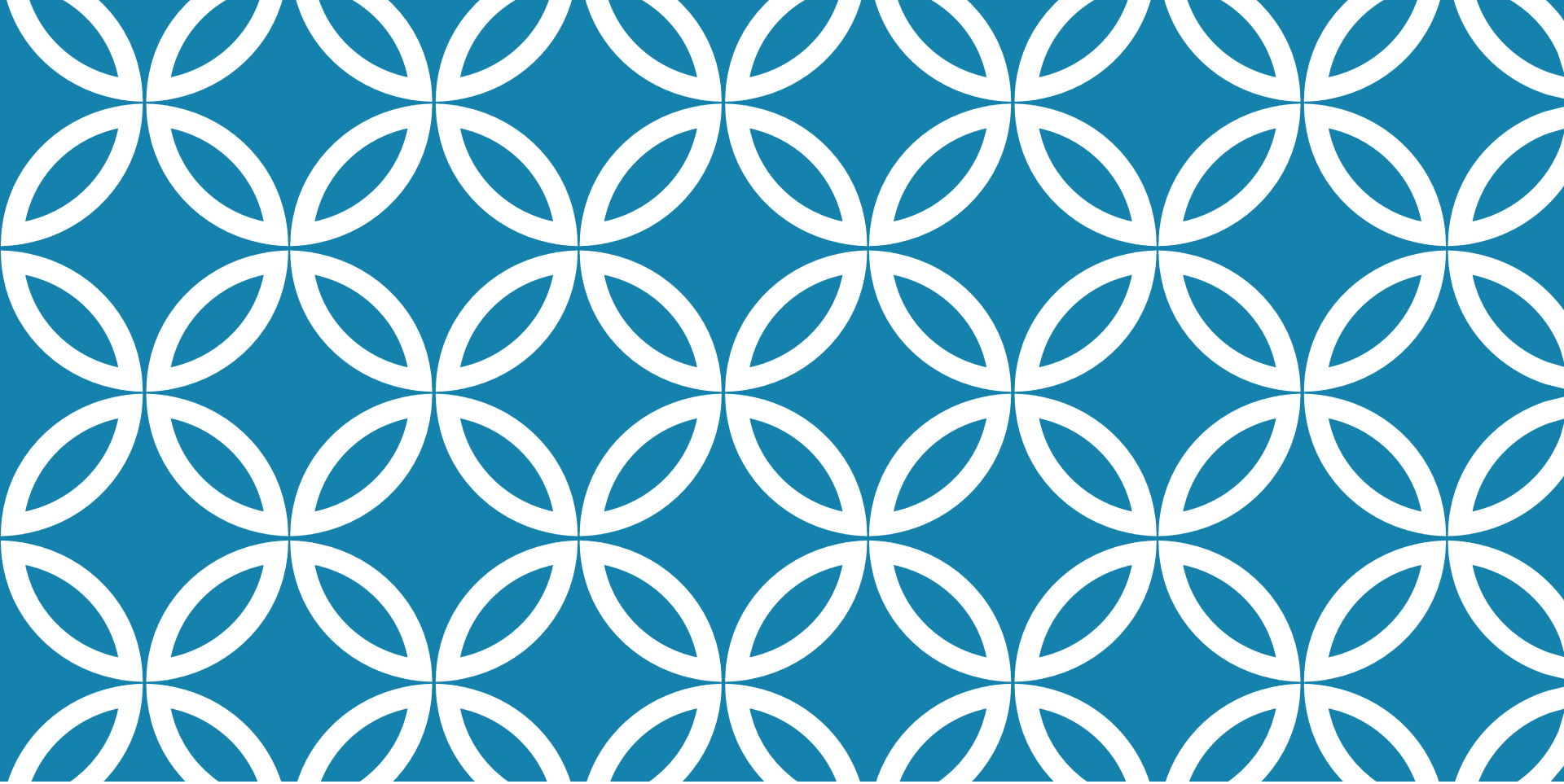
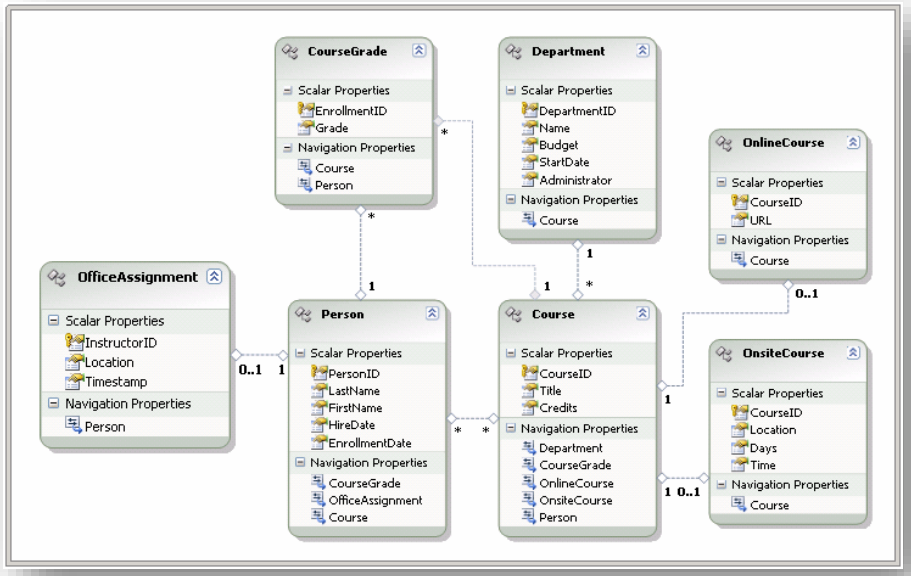


VERİTABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. Üyesi Ender Şahinaslan

BÖLÜM -3-

İLİŞKİSEL VERİ MODELİ & İLİŞKİSEL CEBİR İFADELERİ

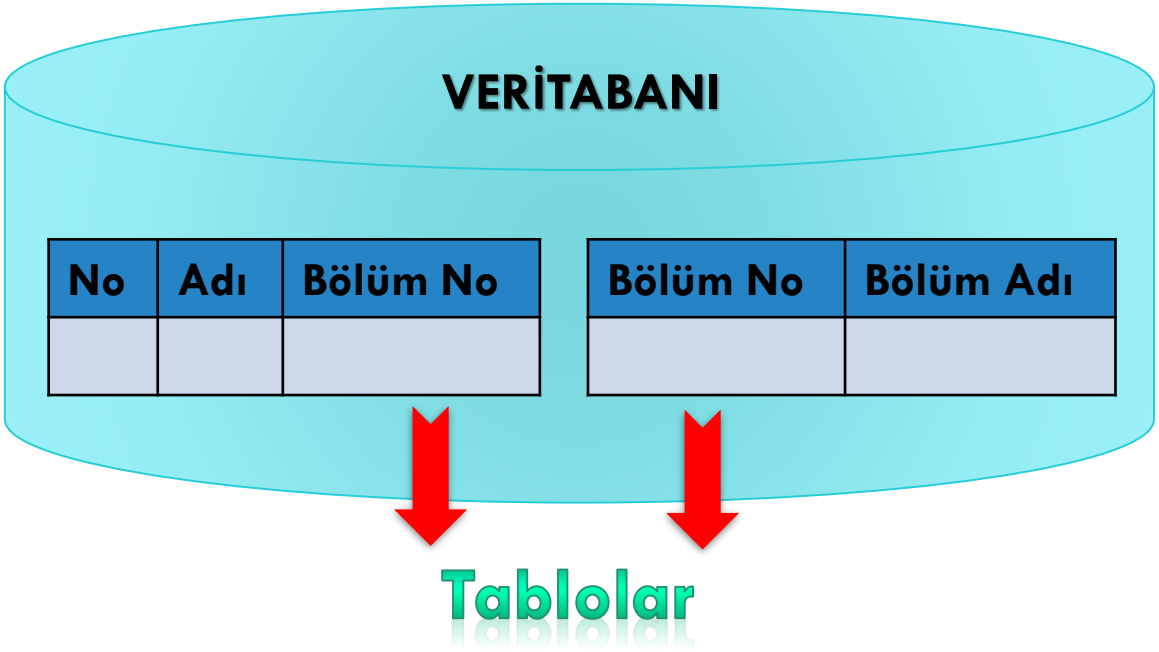


GENEL BAKIŞ…

- İlişkisel veritabanı kavramı ve ana bileşeni olan tabloların özellikleri,
- Veritabanı şeması,
- Veritabanı bütünlüğü,
- Anahtarların bütünlük sınırlamalarındaki önemi,
- Veritabanı sınırlaması türleri,
- İlişkisel cebir,
- İlişkisel cebir temel işlemleri.

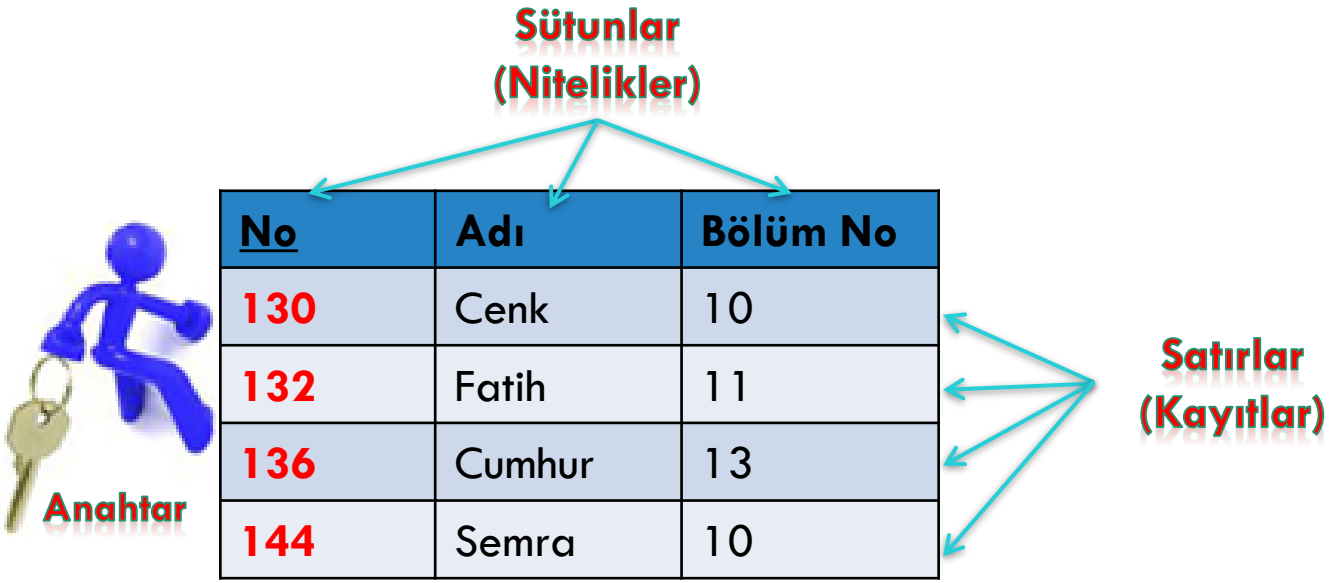
3.1. İLİŞKİSEL MODEL

İlişkisel model, varlıklar arasındaki bağlantının, içerdiği değerlere göre sağlanması esasına dayanır. İlişkisel model, varlıklar arasında oluşan karmaşık ilişkileri basite indirgemek amacıyla geliştirilmiştir.



3.2. İLİŞKİSEL VERİTABANI

- İlişkisel veritabanı, her biri özel isimlere sahip tablolardan oluşur. İlişkisel veri tabanında her bir tablo bir varlığa veya bir ilişkiye karşılık gelmektedir.



3.2.1. TABLOLARIN ÖZELLİKLERİ

İlişkisel veritabanı içinde yer alan her bir tablo;

Sütunlardan oluşur ve her bir sütunun ayrı bir adı vardır.

Her bir sütun, aynı niteliğin tanımlandığı aynı etki alanının (domain) belirlediği değerleri içerir.

Her bir satır birbirinden farklıdır.

Satırların ve sütunların sırası önemsizdir.

Sütunların veya satırların yer değiştirmesi tabloyu değiştirmez. Bu iki tablo birbirinin aynıdır.

No	Adı	Bölüm No
130	Cenk	10
132	Fatih	11
136	Cumhur	13

Adı	No	Bölüm No
Cenk	130	10
Fatih	132	11
Cumhur	136	13

3.2.2. VERİTABANI ŞEMASI

Veritabanının mantıksal tasarımına **veritabanı şeması** adı verilir. Tablolar ve onların nitelikleri; veritabanı şemasını oluşturur.

Veritabanı şemalarını iki ana grup altında ortaya koymak mümkündür.

- ✓ Fiziksel Şema
- ✓ Kavramsal Şema

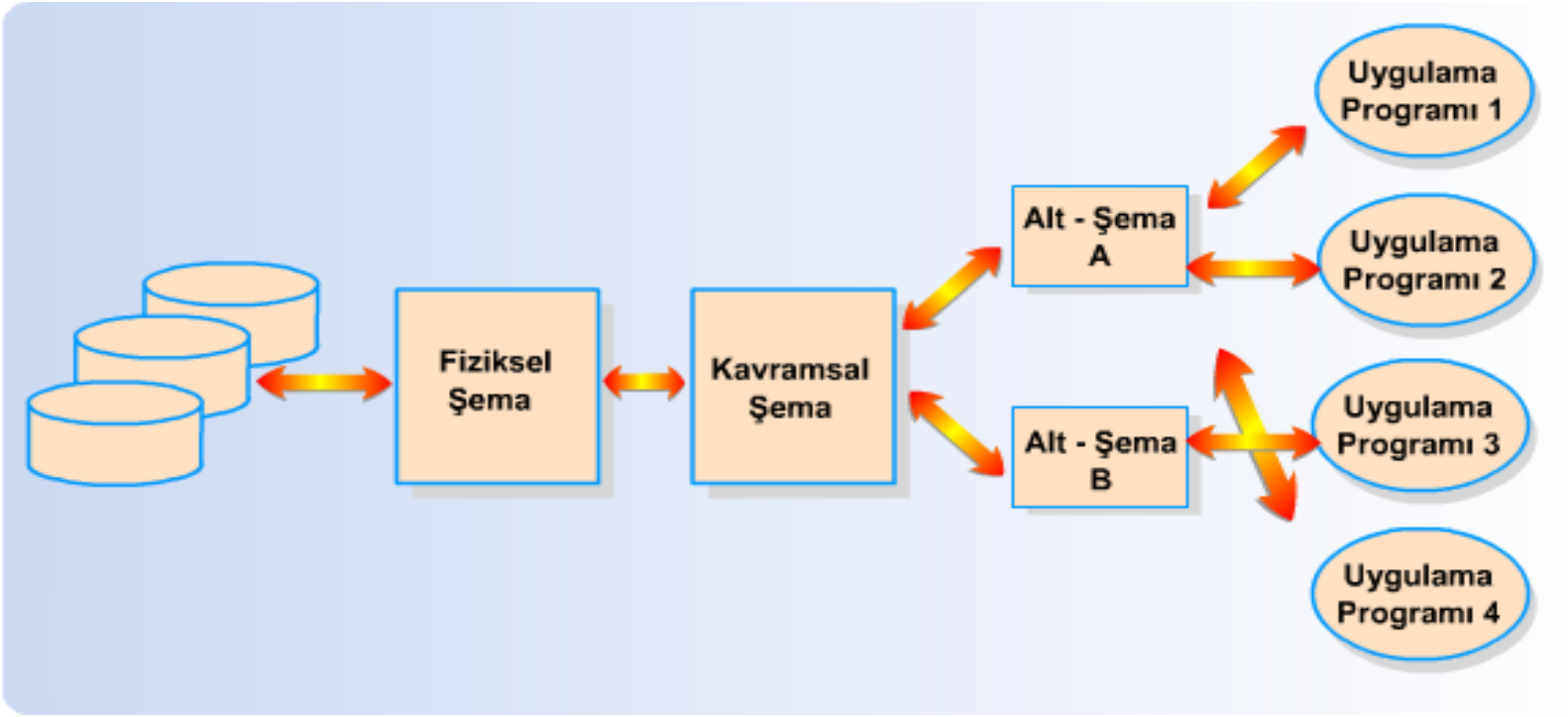
3.2.2. VERİTABANI ŞEMASI (DEVAM...)

Fiziksel şema, veritabanının fiziksel çevresi ile ilgili tanımları içerir. Örneğin, veritabanı bilgisayarda bir disk dosyası biçiminde yer alacaktır. Bu dosyanın disk üzerindeki adresi ve özellikleri ile ilgili tanımlar fiziksel şemayı oluştur.

Kavramsal şema ise tüm veritabanının mantıksal tasarımıdır. Veritabanına konulmasına karar verilen veriler arasındaki mantıksal ilişkilerin yapısının saptanması için veritabanı şeması oluşturulur. **Bu şemada**, veritabanında veri alanları, kayıtlar, dosyalar vb gibi ne tür veri elemanlarının bulunacağı, veri elemanları arasındaki ilişkiler ve veritabanının yapısı hakkında bilgiler yer alır.

3.2.2. VERİTABANI ŞEMASI (DEVAM...)

Veritabanı şeması veya bir başka deyişle kavramsal şema tasarlandıktan sonra, her bir uygulama için **alt-şemalar** hazırlanır. Örneğin, muhasebe uygulaması bir alt şemadır. Çünkü bu uygulama veritabanının tümü ile ilgilenmez.



3.2.3. VERİTABANI BÜTÜNLÜĞÜ

Verinin doğru ve tutarlı olmasına **"veri bütünlüğü (data integrity)"** denir.

Veritabanının doğru ve tutarlı biçimde çalışması ve işlemleri yerine getirmesi gerekir.

Veri bütünlüğünün sağlanması sonucunda, veritabanının eksik, yanlış, tutarsız ve çelişkili **olmam**ası sağlanır.

3.2.3. VERİTABANI BÜTÜNLÜĞÜ (DEVAM...)

Bir veritabanında belirli kurallar ve kısıtlar altında veri ekleme, silme ve güncelleme işlemlerinin yapılmasının ardından, o veritabanının **fiziksel ve mantıksal yapısında** bir bozulma olmaz ise **veri bütünlüğü** sağlanmış olur.

Veritabanında, **veri bütünlüğünü sağlamak için** birçok yol bulunmaktadır. Bunlar;

- Constraint'ler (kısıtlayıcılar),
- Rule'lar (kurallar) ve
- Default'lar (varsayılanlar) olmak üzere üç çeşidi vardır.

3.2.3.1. CONSTRAINTS (KISITLAMALAR)

Veri üzerindeki mantıksal sınırlamalara **kısıt** adı verilir.

Kısıtların genel olması tercih edilen bir durumdur.

Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılır.

Kısıtlamalar, tabloların tanımlanmasıyla beraber oluşan öğelerdir.

Constraintler **tablo oluştururken** yani ***CREATE TABLE*** komutuyla tanımlanabilir. Tablo **daha önce oluşturulmuşsa** ***ALTER TABLE*** komutuyla bu işlem gerçekleşir.

- ALTER TABLE komutuyla kullanıldığında sütunlara girilen bilgilerin dikkate alınması gerekir.

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Kısıtlamalar, nesnelerdeki alanlara girilen veriyi kontrol ederek **verinin güvenilirliğini artırırlar** ve **veri girişini daha kolay** hale getirirler.

Kısıtlama Türleri:

- Primary Key Constraint
- Unique Constraint
- Foreign Key Constraint
- Default Constraint
- Check Constraint

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Primary Key Constraint

- **Birincil anahtar kısıtlayıcı**, her kaydın birbirinden farklı olmasını yani aynı olmayan değerlerin girilmesini sağlar.

- Her tablonun **en fazla 1 adet** *Primary Key Constraint'i* olabilir.

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Primary Key Constraint (Örneği)

```
CREATE TABLE Okul(
    OgrNo int NOT NULL,
    Adi varchar(15),
    Soyadi varchar(20),
    Sinif  varchar(10),
    TCKimlikNo varchar(11),
    BolumID int NOT NULL,
    CONSTRAINT PKC_OgrNo PRIMARY KEY (OgrNo)
    )
```

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Unique Constraint

- **Tekil alan kısıtlayıcı** anlamındadır. Birincil anahtar olan ve tablodaki diğer alanlar içinde aynı içeriğe sahip verilerin olmaması için *Unique Constraint* tanımlanır.

- TC Kimlik Numarası «Unique» şeklinde bir tanımlama Unique Constraint'e bir örnektir.

```
ALTER TABLE Okul
ADD CONSTRAINT UC_TCKimlikNo UNIQUE (TCKimlikNo)
```

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Foreign Key Constraint

- Dış anahtar kısıtlayıcı anlamındadır. Bir tablodaki bir sütuna ait verilerin başka bir tablonun belirli bir sütunundan gelmesini denetler.

```
ALTER TABLE Okul
ADD CONSTRAINT FKC_bolumID
FOREIGN KEY (bolumID) REFERENCES bolum (bolumID)
```

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Default Constraint

- Bir tabloya veri girişi esnasında o alanın alacağı varsayılan bir değerin tanımlanması için kullanılan kısıtlayıcıdır.

```
ALTER TABLE Okul
ADD CONSTRAINT DC_sinif DEFAULT 10 FOR sinif
```

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Check Constraint

- Tabloda belirtilen **bir sütuna**(alan/varlık nitelik) **istenilen şartlara göre değer girilebilmesini** ve **bunların kontrolünü** sağlayan kısıtlayıcıdır.

- Aynı sütun için birden fazla Check Constraint kullanılabilir.

```
ALTER TABLE Okul

ADD CONSTRAINT CC_ogrNo CHECK (LEN(ogrNo)=9)
```

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

Constraint Örneği:

- Bir stok programı yazıldığını düşünelim.
- "Urunler" adlı tabloda ürünlerle ilgili bilgilerin olduğunu varsayalım.
- UrunGirisTarihi, ürünün depoya giriş tarihinin, UrunCikisTarihi de ürünün depodan çıkış tarihinin girildiği sütunlar olarak belirlensin.
- Bu verilenleri dikkate alarak **ürünün depodan çıkış tarihinin her zaman boş veya giriş tarihinden büyük olduğunu** garanti edecek bir Check Constraint yazınız.

3.2.3.1. CONSTRAINTS (KISITLAMALAR) (DEVAM...)

```
CREATE TABLE Urunler(
        UrunNo varchar(10),
        UrunAd varchar(200),
        UrunGirisTarihi datetime,
        UrunCikisTarihi datetime NULL,
        CONSTRAINT CC_UrunCikisTarihi
                CHECK (UrunCikisTarihi IS NULL OR
                        UrunCikisTarihi >= UrunGirisTarihi)
                FOR UrunCikisTarihi
)
```

3.2.3.2. RULES (KURALLAR)

Veri bütünlüğünü sağlamak üzere "**rule**" kavramı da kullanılmaktadır. Bir **kural (rule)** bir tablodaki **belirli bir sütuna** veri girişi sırasında entegre edilen **kısıtı ifade** eder.

Örneğin bir banka tarafından müşterilerine verdiği kredi kartının PIN kodunun 4 haneli olması gibi. Bu gibi durumların tümü **SQL Server'in** basitçe "**rule**" nesneleri ile gerçekleşir.

Bunun yanında SQL Server **check constraints**'ler ile de kendiliğinden destek sağlanmaktadır. **İkisi arasındaki tek fark**, **rule**'larin ayrı bir nesne olarak, check **constraint**'lerin ise tablo içerisinde yer alan bir tablo kısıtı olarak saklanmasıdır.

3.2.3.2. RULES (KURALLAR) (DEVAM...)

Bir **rule** tanımlamak için aşağıdaki T-SQL ifadesi kullanılır.

```
CREATE RULE rule_name AS statement
sp_bindrule 'rule_name' 'table_name.column_name'
```

Rule kullanımına örnek olarak bir bankanın müşterilerine verdiği 4 basamaklı PIN kodu kullanımını verebiliriz.

```
CREATE RULE rule_pin AS LEN(@value)=4
sp_bindrule 'rule_pin' 'info.pin'
```

- Bu örnekte ***rule_pin*** isimli kural ***info*** tablosunun ***pin*** sütununa uygulanmaktadır.

3.2.3.3. DEFAULTS (VARSAYILANLAR)

"**Default**", veritabanında bulunan bir tablodaki **bir sütunun veri girildiği anda boş kalmaması** için daha önceden tanımlanan **varsayılan bir değer** kısıtı girilmesini tanımlamaktadır.

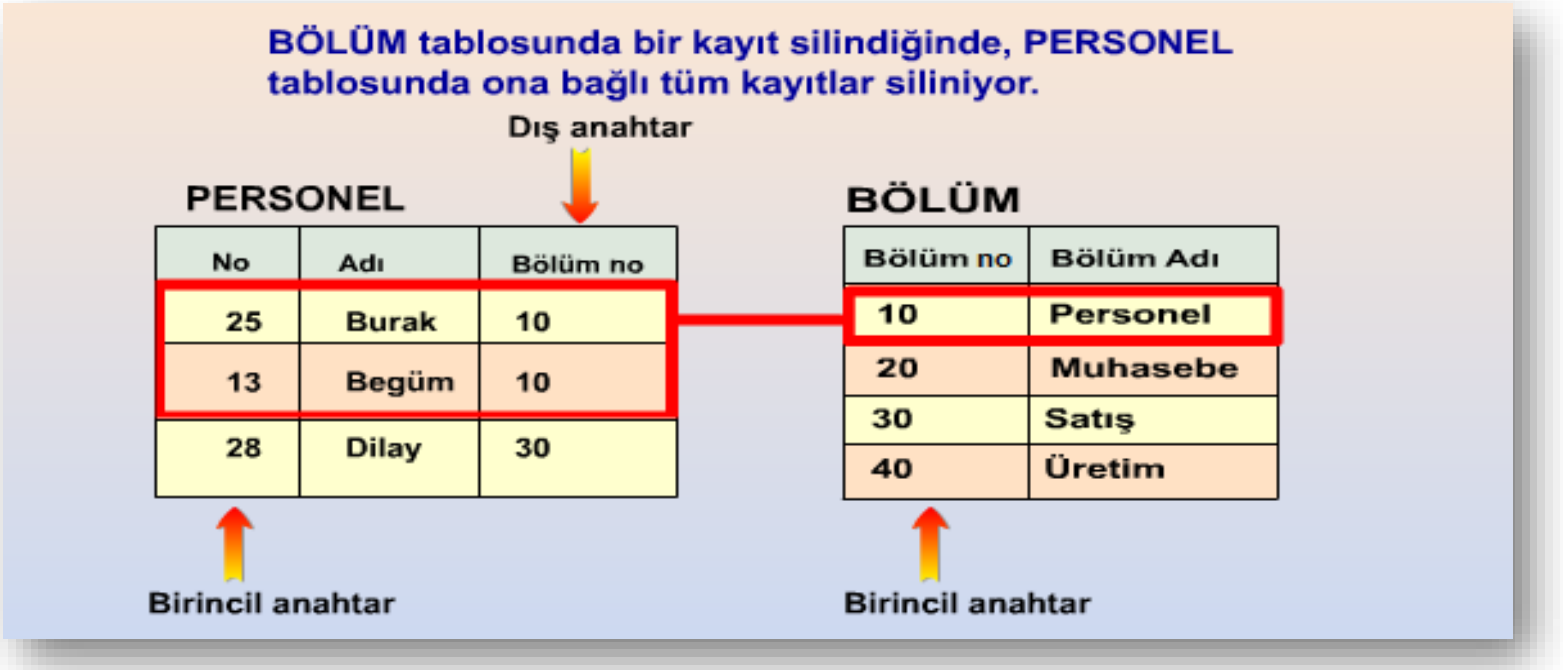
SQL Server'in bu anlamda sunduğu **default constraint'in** programatik olarak **karşılığı** "**default**" nesnesidir.

Aşağıda bir default nesne oluşumunun genel yapısı görülmektedir.

CREATE DEFAULT default_name AS value

3.2.3.4. CASCADING DATA INTEGRITY (BASAMAKLI VERİ BÜTÜNLÜĞÜ)

İki tablo birbirleriyle dış anahtar kullanılarak ilişkilendirildikten sonra, bu sınırlamalara dayanarak, iki tablo arasında silme ve güncelleştirme işlemleri otomatik olarak yerine getirilebilir.



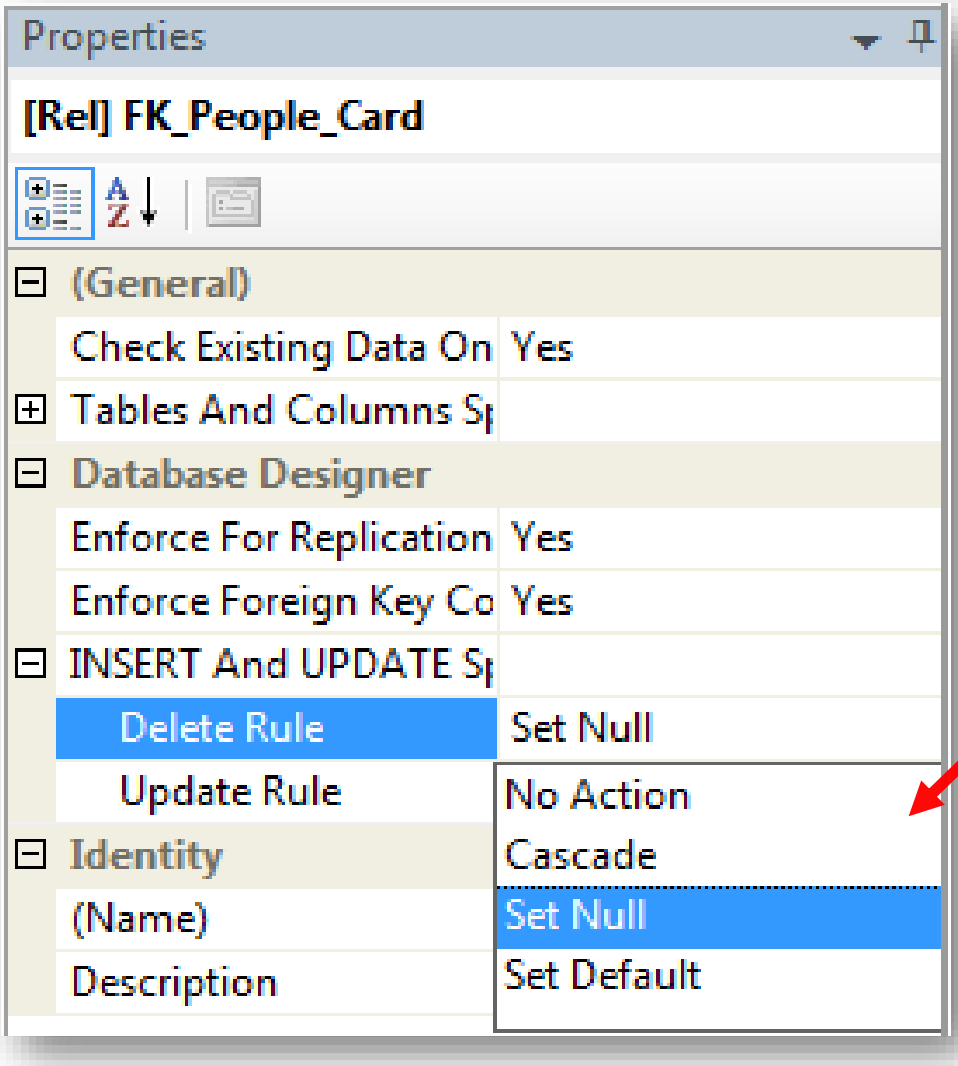
3.2.3.4. Cascading Data Integrity (Basamaklı Veri Bütünlüğü)

Örneğin; PERSONEL ve BÖLÜM isimli iki tabloyu göz önüne alalım.

- PERSONEL tablosunun ***Bölüm No*** isimli sütunu, BÖLÜM tablosuna ilişkin bir **dış anahtar** olarak tanımlanmıştır. Bu tür bir tanım beraberinde ilişkisel bütünlük özelliklerini getirecektir.

- **BÖLÜM isimli tablodan bir satırı,** örneğin 10 numaralı bölümü silmek istiyoruz. Bu satır silindiğinde, PERSONEL tablosunda da **aynı satır ile ilgili tüm kayıtlar** otomatik olarak yok olacaktır.

3.2.3.4. Cascading Data Integrity *(Basamaklı Veri Bütünlüğü)*



SQL Server **4** adet cascading parametresi sağlamaktadır.

3.2.3.4. Cascading Data Integrity *(Basamaklı Veri Bütünlüğü)*

- **No Action (Default)** - Primary key silindiği zaman, bu key'e bağlı olan **foreign key**'in olduğu **tabloda kayıt varsa** bu **silme işlemi hata verir.**

- **Cascade** - Primary key silindiği zaman, bu key'e bağlı olan **foreign key'in olduğu tabloda da ilgili kayıtlar silinir.**

- **Set Null** - Primary key silindiği zaman, bu key'e bağlı olan **foreign key'in olduğu tabloda ilgili kayıtlar silinmez,** ilgili **Foreign key kolonuna "null" atılır.**

- **Set Default** - Primary key silindiği zaman, bu key'e bağlı olan foreign key'in olduğu tabloda ilgili kayıtlar silinmez, ilgili **foreign key kolonuna "default" değer atılır.**

3.2.3.4. Cascading Data Integrity (Basamaklı Veri Bütünlüğü)

Aşağıdaki UPDATE ve DELETE işlemleri için foreign key kullanımına dair bir örnek mevcuttur:

```
CREATE TABLE faculties(
        facultyID int NOT NULL,
        facultyName varchar(20),
        CONSTRAINT pk_faculties PRIMARY KEY (facultyID)
)
CREATE TABLE departments(
        departmentID int NOT NULL PRIMARY KEY,
        departmentName varchar(20),
        facultyID int,
        CONSTRAINT fk_faculties_ departments
        FOREIGN KEY (facultyID) REFERENCES faculties (facultyID)
)
```

3.2.3.4. Cascading Data Integrity (Basamaklı Veri Bütünlüğü)

```
ALTER TABLE departments
ADD CONSTRAINT fk_faculties_departments
FOREIGN KEY (facultyID)
REFERENCES faculties(facultyID)
ON DELETE [NO ACTION | CASCADE | SET NULL | SET DEFAULT]
ON UPDATE [NO ACTION | CASCADE | SET NULL | SET DEFAULT]
```

- **Var olan bir kısıtı silmek için** aşağıdaki komutları kullanırız:

```
ALTER TABLE departments
DROP CONSTRAINT fk_faculties_departments
```

3.3. İLİŞKİSEL MODEL ÖRNEĞİ

Örnek olarak basit bir market veritabanı tasarlayalım. Bir markette olabilecek tablolar aşağıda gösterilmiştir.

Reyonlar

Reyon No	Reyon Adı
23	Manav
34	Şarküteri
42	Deterjan

Ürünler

Ürün No	Reyon No	Ürün Ad	Alış Fiyatı	Satış Fiyatı	Geliş Tarihi	Miktar
4	23	Elma	100	130	10.10.2009	50
10	42	Sabun	20	50	12.10.2009	100

Personel

Personel No	Adı	Soyadı	Maaş	Reyon No
25	Fatih	Güven	1500	23
55	Ali	Yakar	1000	42

3.3. İLİŞKİSEL MODEL ÖRNEĞİ (DEVAM...)

Tablolar arasında çeşitli ilişkiler bulunmaktadır.

Örneğin, Personel tablosunda yer alan **Reyon No,** Reyonlar tablosundaki **Reyon No** ile ilişkilidir. Burada **Reyon No = 23** olan personel **"Fatih Güven"** ın **Manav** reyonunda çalıştığı bu ilişkiye bakılarak söylenebilmektedir.

Reyonlar tablosunda yer alan her ürün grubuna bir numara verilerek birbirlerine karışmaları engellenmiştir. Ayrıca Ürünler tablosundaki **Reyon No** alanı Reyonlar tablosundaki **Reyon No** ile ilişkilidir.

3.3. İLİŞKİSEL MODEL ÖRNEĞİ (DEVAM…)

Burada her ürün grubuna bir numara verilerek sistematik bir şekilde düzenlenmesi sağlanmıştır.

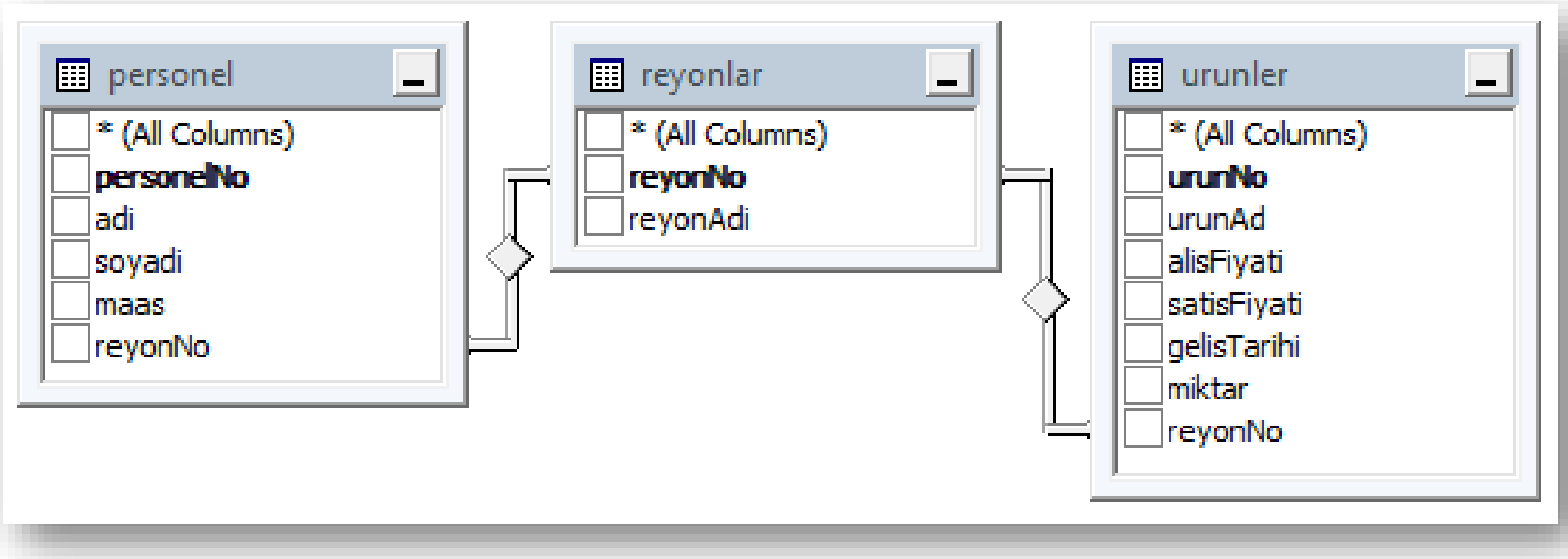
Daha sonra Reyonlar ve Ürünler arasındaki ilişki kullanılarak ürün bilgilerine erişim sağlanmıştır.

Burada ayrıca ürün adına bakılarak, ürünün hangi reyonda yer aldığı hemen tespit edilmektedir.

- √ Örneğin; "Elma" ürünü "23" numaralı reyonda yer almaktadır.
- √ 23 numaralı reyonda çalışan personelin ise "Fatih Yücalar" olduğu aradaki ilişkilere bakılarak tespit edilebilmektedir.

3.3. İLİŞKİSEL MODEL ÖRNEĞİ (DEVAM...)

SQL Server' da bu tablolar arasındaki ilişki aşağıdaki gibi olacaktır.



3.4. İLİŞKİSEL CEBİR

İlişkisel cebir, bir ilişkisel sorgulama dilidir. Bu ifadeler yardımı ile birçok sorgulama işlemi kodlama aşamasına geçmeden önce tanımlanabilmektedir.

Örneğin, belirli kayıtların seçilmesi, belirli sütunların sorgulama sonucunda elde edilmesi gibi daha birçok işlem ilişkisel cebir ifadeleri kullanılarak kolayca ortaya konabilmektedir.

İlişkisel cebir, bir veya iki ilişkiyi girdi olarak alıp, sonuçta yeni bir ilişki üreten bir dizi işlemden oluşmaktadır. Ayrıca ilişkisel cebir bir veritabanı sorgulama dilidir.

3.4. İLİŞKİSEL CEBİR (DEVAM...)

İlişkisel cebir bir sorgulama dilidir fakat bu sorgulamalar sadece biçimsel olarak yapılmaktadır.

Yani sorgulama işleminin yapılması için bir yorumlayıcı yada derleyiciye ihtiyaç yoktur. Bu bakımdan ele alındığında SQL' den farklıdır.

Burada yazılan cebir ifadeleri daha sonraki aşamada bir sorgu dili ile (örneğin SQL) komutlara dönüştürülmektedir.

3.4. İLİŞKİSEL CEBİR (DEVAM...)

Bir ilişkisel veritabanında kullanılan temel ilişkisel cebir ifadeleri şunlardır.

- ✓ Seçme (select) işlemi
- ✓ Projeksiyon / Atma (project) işlemi
- ✓ Kartezyen çarpım (cartesian product) işlemi
- ✓ Birleştirme (join) işlemi
- ✓ Kesiştirme (intersect) işlemi
- ✓ Fark (difference) işlemi

3.4. İLİŞKİSEL CEBİR (DEVAM...)

Operation	My HTML	Symbol
Projection	PROJECT	π
Selection	SELECT	σ
Renaming	RENAME	ρ
Union	UNION	$\cup$
Intersection	INTERSECTION	$\cap$
Assignment	<-	$\leftarrow$

Operation	My HTML	Symbol
Cartesian product	X	$\times$
Join	JOIN	$\bowtie$
Left outer join	LEFT OUTER JOIN	$⟕$
Right outer join	RIGHT OUTER JOIN	$⟖$
Full outer join	FULL OUTER JOIN	$⟗$
Semijoin	SEMIJOIN	$\ltimes$

3.4.1. SEÇME İŞLEMİ

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.

3.4.1. SEÇME İŞLEMİ (DEVAM...)

Örnek: Müşteri tablosu aşağıda verilmiştir. Bu tabloyu kullanarak ilçesi "Maltepe" olan müşterileri listeleyiniz.

Müşteri

Müşteri No	Adı	İl	İlçe	Bakiye
25	Fatih	İstanbul	Maltepe	1250
55	Ali	İstanbul	Beşiktaş	3524
32	Serap	Ankara	Kızılay	2642

$$\sigma_{İlçe="Maltepe"}(MÜŞTERİ)$$

3.4.2. PROJEKSİYON / ATMA İŞLEMİ

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

$$\prod_{Sütun\ İsimleri} (TABLO)$$

Yine burada da mantıksal operatörler ve karşılaştırma işlemleri kullanılabilmektedir.

3.4.2. PROJEKSİYON / ATMA İŞLEMİ (DEVAM...)

Örnek:Müşteri tablosunda müşteri adı ve bakiye bilgilerini listeleyiniz.

$$\Pi_{\text{Adı, Bakiye}}(\text{MÜŞTERİ})$$

Müşteri tablosunda bakiyesi 2000 den büyük olan müşterilerin adı ve şehir bilgisi nedir?

$$\Pi_{\text{Adı, İl}}(\sigma_{\text{Bakiye}>2000}(\text{MÜŞTERİ}))$$

3.4.3. KARTEZYEN ÇARPIM İŞLEMİ

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.

TABLO1 X TABLO2

3.4.3. KARTEZYEN ÇARPIM İŞLEMİ (DEVAM…)

Örnek: Aşağıda verilen Öğrenci ve Dersler tablolarını göz önüne alarak Bilgisayar bölümünde okuyan ve tüm dersleri alan öğrencileri listeleyen ilişkisel cebir ifadesi nedir?

Öğrenci

Öğrenci	Bölüm
Fatih Güven	Yazılım
Veli Sakin	Bilgisayar
Erdal Aydın	Elektronik
Raif Aslan	Bilgisayar

Dersler

Ders	Kredi
Veritabanı ve Yönetimi	3
Matematik	4

$$\sigma_{Bölüm="Bilgisayar"}(ÖĞRENCİ\ X\ DERSLER)$$

3.4.3. KARTEZYEN ÇARPIM İŞLEMİ (DEVAM...)

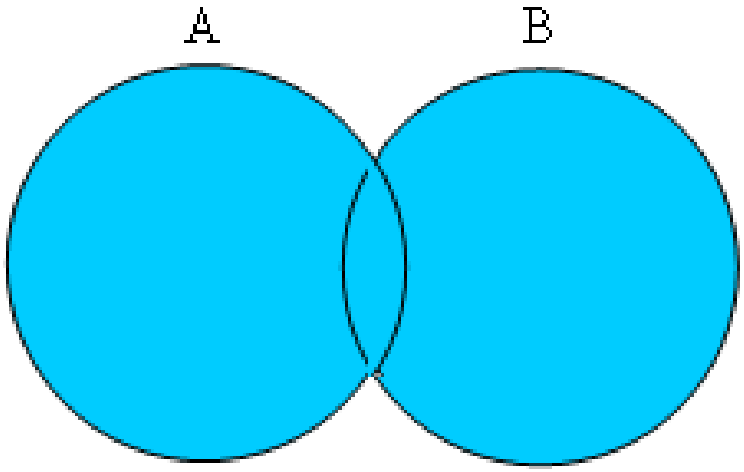
Sonuç olarak elde edilecek olan kartezyen çarpım tablosu aşağıdaki gibi olacaktır.

Öğrenci X Dersler

Öğrenci	Bölüm	Ders	Kredi
Veli Sakin	Bilgisayar	Veritabanı Yönetimi	3
Veli Sakin	Bilgisayar	Matematik	4
Raif Aslan	Bilgisayar	Veritabanı Yönetimi	3
Raif Aslan	Bilgisayar	Matematik	4

3.4.4. BİRLEŞTİRME İŞLEMİ

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



3.4.4. BİRLEŞTİRME İŞLEMİ (DEVAM...)

Örnek: Aşağıda verilen Kredi ve Mevduat tablolarını göz önüne alarak, bankanın Maltepe şubesinde mevduat **ve/veya** kredi hesabı bulunan müşterilerin isimlerini listeleyiniz?

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Veli	2000	Maltepe
Raif	1600	Konak

$$\Pi_{Kredi.Müşteri}(\sigma_{İlçe="Maltepe"}(KREDİ)) \cup$$

$$\Pi_{Mevduat.Müşteri}(\sigma_{İlçe="Maltepe"}(MEVDUAT))$$

3.4.4. BİRLEŞTİRME İŞLEMİ (DEVAM...)

Bu sorgu neticesinde elde edilen sonuç aşağıdaki gibi olacaktır.

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

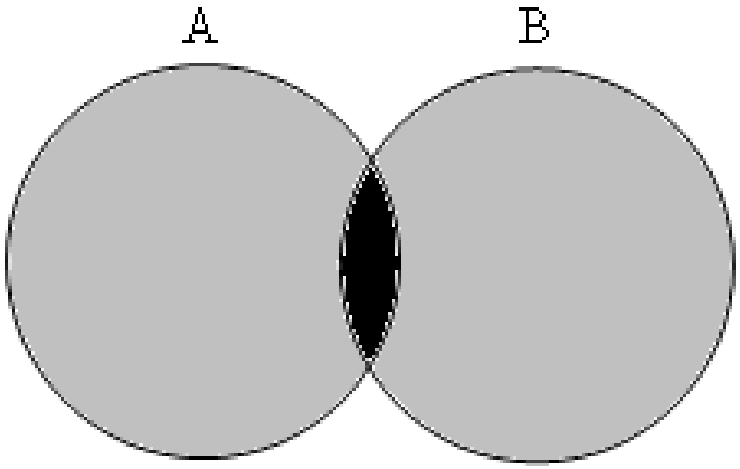
Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Veli	2000	Maltepe
Raif	1600	Konak

Müşteri
Fatih
Veli

3.4.5. KESİŞME İŞLEMİ

Belirlenen iki ilişkiden birinde bulanan kayıtların belirlenmesi için kullanılan bir işlemdir. Bu işlem ∩ simgesi ile gösterilmektedir. Genel gösterimi aşağıdaki gibidir.



3.4.5. KESİŞME İŞLEMİ (DEVAM…)

Örnek: Aşağıda verilen Kredi ve Mevduat tablolarını göz önüne alarak, bankanın Maltepe şubesinde **hem mevduat hem de kredi** hesabı bulunan müşterilerin isimlerini listeleyiniz?

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Erdoğan	2000	Maltepe
Raif	1600	Konak

$$\prod_{Kredi.Müşteri}(\sigma_{İlçe="Maltepe"}(KREDİ)) \cap$$

$$\prod_{Mevduat.Müşteri}(\sigma_{İlçe="Maltepe"}(MEVDUAT))$$

3.4.5. KESİŞME İŞLEMİ (DEVAM...)

Bu sorgu neticesinde elde edilen sonuç aşağıdaki gibi olacaktır.

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

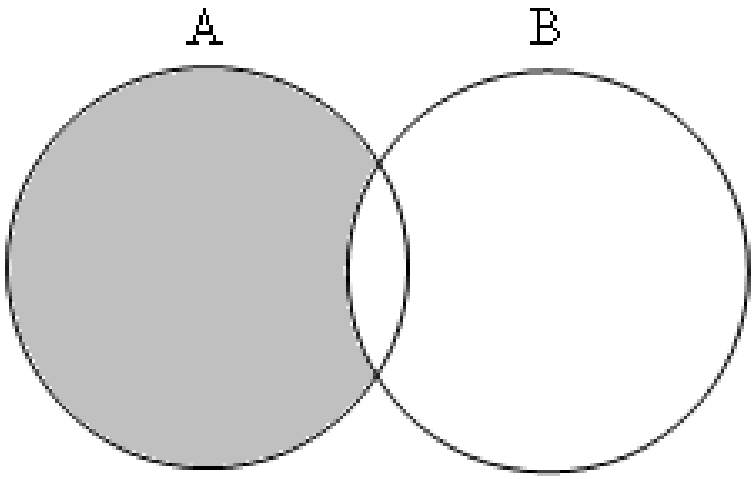
Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Veli	2000	Maltepe
Raif	1600	Konak

Müşteri
Fatih

3.4.6. FARK İŞLEMİ

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



3.4.6. FARK İŞLEMİ (DEVAM...)

Örnek: Aşağıda verilen Kredi ve Mevduat tablolarını göz önüne alarak, bankanın Konak şubesinde mevduat hesabı olup kredi hesabı olmayan müşterilerin isimlerini listeleyiniz?

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Veli	2000	Maltepe
Raif	1600	Konak

$$\prod_{Mevduat.Müşteri}(\sigma_{İlçe="Konak"}(MEVDUAT)) -$$

$$\prod_{Kredi.Müşteri}(\sigma_{İlçe="Konak"}(KREDİ))$$

3.4.6. FARK İŞLEMİ (DEVAM...)

Bu sorgu neticesinde elde edilen sonuç aşağıdaki gibi olacaktır.

Kredi

Müşteri	Bakiye	İlçe
Erdal	1500	Kartal
Fatih	1750	Maltepe

Mevduat

Müşteri	Bakiye	İlçe
Fatih	1500	Maltepe
Erdal	1750	Kartal
Veli	2000	Maltepe
Raif	1600	Konak

Müşteri
Raif

TEŞEKKÜRLER…

SORULARINIZ ?