# Sıra Bağıntıları

Ders 5



# Sıra Bağıntıları

- Birçok küme doğal olarak sıralanmış elemanlara sahiptir. Örneğin büyüklüğe göre sıralanmış reel sayılar kümesi. Benzer şekilde bir küme topluluğu eleman sayısına göre sıralanabilir.
- Örneğin, $A \subseteq B$ ise A, B' den küçüktür deriz.
- Eşdeğerlik bağıntılarından farklı olarak birçok farklı tip sıra bağıntısı vardır. En genel sıra bağıntısı 'parçalı sıra' bağıntısıdır.
- **Tanım:** Bir kümedeki **parçalı sıra**, yansıyan, ters simetrik ve geçişli olan bir bağıntıdır.
- Bir kümede parçalı sıra varsa bu kümeye **parçalı sıralı küme** denir.

# Sıra Bağıntıları - Örnek

- **Örnek:** Reel sayılar kümesi üzerinde x ***R*** y sadece ve sadece $x \leq y$ ise şeklinde tanımlanan ***R*** bağıntısı parçalı sıradır.
- Öte yandan, x ***S*** y sadece ve sadece $x<y$ ise şeklinde tanımlanan ***S*** bağıntısı parçalı sıra değildir çünkü yansıyan değildir.
- **Teorem:** ***R***, A kümesi üzerinde parçalı bir sıra ve B de A'nın herhangi bir alt kümesi olsun. Bu durumda, $S = R \cap (B \times B)$ B üzerinde bir parçalı sıradır.



# En büyük ve en küçük eleman

- Teoreme göre reel sayıların her hangi bir alt kümesi $\leq$ bağıntısı ile parçalı sıralıdır.
- Bu şekilde sıralanmış bazı reel sayı kümeleri en büyük veya en küçük elemana sahip olabilir, bazıları da olmayabilir.
- Örneğin, tam sayılar kümesinin en büyük veya en küçük elemanı yokken, pozitif tamsayıların en küçük elemanı 1' dir fakat en büyük elemanı yoktur.
- En büyük veya en küçük eleman bir tane olmayabilir.
- Örneğin, *{a,b,c}* kümesinin öz alt kümelerini eleman sayısına göre sıralarsak, en küçük eleman $\emptyset$ iken en büyük eleman üç tanedir çünkü üç tane iki elemanlı alt küme vardır.

# En büyük ve en küçük eleman

**Tanım:** ***R***, *A* kümesi üzerinde bir parçalı sıra olsun. *A*' nın **en büyük elemanı,** tüm a $\in$ *A* için a ***R*** α olmak üzere α elemanıdır.

**Tanım:** ***R***, *A* kümesi üzerinde bir parçalı sıra olsun. *A*' nın **en küçük elemanı,** tüm a$\in$*A* için β ***R*** a olmak üzere β elemanıdır.

- Yeniden {a,b,c}' nin öz alt kümeleri örneğine dönersek iki elemanlı her bir alt küme en büyük eleman olacaktır. O halde bu düşünceyi maksimal eleman tanımı ile formülize edebiliriz.

**Tanım:** *A*, ***R*** sıra bağıntılı bir parçalı sıralı küme olsun. Tüm a $\in$ *A* için x ***R*** a, x=a anlamına geliyorsa *A*' daki x elemanı **maksimal**dır.

**Tanım:** *A*, ***R*** sıra bağıntılı bir parçalı sıralı küme olsun. tüm a $\in$ *A* için a ***R*** y, a=y anlamına geliyorsa y elemanı **minimal**dir.



# Uygulama: İlişkisel Veritabanları

- Bilgiyi saklamak için tasarlanmış bilgisayar yazılıma veritabanı sistemi denir. Saklanmış verilerin işlenmesinin kontrol eden yazılıma da veritabanı yönetim sistemi (database management system) veya DBMS denir.
- Tüm veritabanı yönetim sistemleri, verinin özel bir tip yapıya sahip olduğunu ve DBMS' in saklı veriyi, verinin kendi teorik modeline göre işlediğini varsayar.
- Bu yüzden bir çok değişik tip DBMS bulunur: ilişkisel, ağ ve hiyerarşik. Bu bölümde matematiksel bağıntıları esas alan ilişkisel veritabanı sistemlerinden bahsedilecektir.

- 1970'de Matematikçi Edward Fredrik Codd IBM San Jose Laboratory da ilişkisel veri modelini tanımladı.
- 1976'da Peter Chen Varlık İlişkisel (ER) modeli tanımladı.

- Bir veri birçok kısımdan oluşur. Örneğin, adres defterindeki bir kayıt isime, adrese, telefon numarasına göre sınıflandırılabilir. Verinin her bir parçası 'attribute (özellik, nitelik)' olarak adlandırılır.
- Verilerin her zaman belli bir nitelik kümesine sahip olduğunu varsayarız ve bu nitelik kümesine kayıt tipi (record type) adı verilir.

- Tüm verilerin aynı tip olduğu ve veri tekrarının olmadığı tablolara **birincil normal form**dadır **(first normal form, 1NF**) denir.
- İlişkisel veritabanlarının temel kuralı tüm kayıt dosyalarının birincil normal formda olmasıdır.

# Normal Formlar Arası Geçiş

Normal Olmayan Form (UNF)

- Satır içerisinde tekrar eden grupları kaldırın

Birinci Normal Form (1NF)

- Anahtar olmayan her bir özellik tamamıyla birincil anahtara bağımlı olmalıdır.

İkinci Normal Form (2NF)

- İlişki içerisinde geçişli bağımlılıklar olmamalıdır.

Üçüncü Normal Form (3NF)

- Her bir bağımlı olunan özellik aday anahtar olmalıdır.

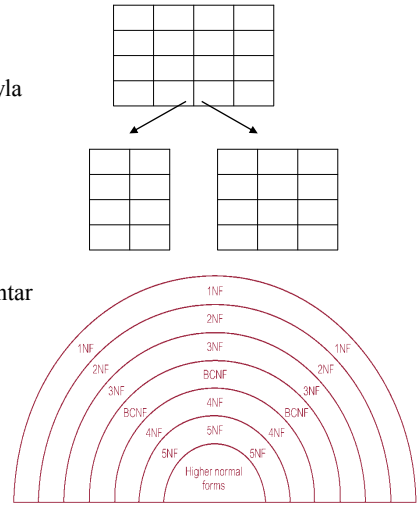
Boyce-Codd Normal Form (BCNF)

- Çok değerli bağımlılıkları kaldırın

Dördüncü Normal Form (4NF)

- Geri kalan anormallikleri kaldırın

Beşinci Normal Form (4NF)






# Uygulama: İlişkisel Veritabanları

**Tanım:** Veri attribute (özellik) adı denilen bileşenlerine ayrılır. Bir kayıt tipi bir attributelar (veya fieldlar) kümesidir.

- Bir kayıt örneği (record instance), belli bir kayıt tipinin gerçek verisidir ve kayıt dosyası aynı kayıt tipinden olan kayıt örneklerinin kümesidir.

**Örnek:** GYTE isimli bir yardım derneği kendisine yapılan bağışları yapan kişileri, isimlerini, adreslerini, telefon numaralarını ve bağışla ilgili diğer detayların bilgilerini tutmak istediğini varsayalım.

- Öncelikle bu dernek; bağışlayanın_adı, bağışlayanın_adresi, bağışlayanın_telefonu, bağış_miktarı ve bağış_tarihi şeklinde adlandırabileceğimiz attribute' ları belirler.
- Bu beş attribute kayıt tipini tanımlar. Tablo 3.1' de bazı kayıt örnekleri gösterilmiştir.

| *bağnın_adı* | *bağnın_adresi* | *bağnın_telefonu* | *bağış_miktarı* | *bağış_tarihi* |
|---|---|---|---|---|
| Giggs, R | 33 New Street, Manchester | 4614-3939 | £100 | Ocak 1997 |
| Giggs, R | 33 New Street, Manchester | 4614-3939 | £150 | Mart 1999 |
| Beatie, J | 24 Oaks Road, Southampton | 6578-4108 | £300 | Ekim 1998 |
| Veron,S | 2A Great Oldtown, London | 2467-1297 | £250 | Kasım 2000 |
| Veron,S | 2A Great Oldtown, London | 2467-1297 | £500 | Aralık 1999 |

# Uygulama: İlişkisel Veritabanları

- Bağış yapanın açık adresi sadece bir attribute ile etiketlendiğinden bu kayıt dosyasından coğrafik bilgiyi elde etmek kolay olmayabilir.
- Örneğin dernek, Manchester' dan bağış yapanları bulmak isterse şehir adı tek başına bir attribute olarak istenmediğinden çok zor olacaktır. bağışlayanın_adresi isimli tek bir attribute cadde ve şehir olarak ikiye ayrılsaydı şirketin işi çok daha kolay olurdu.
- Bu örnek, attribute tanımlamak için önemli bir noktayı göstermiştir. Bir kayıt örneğindeki potansiyel yararlı bilgi parçalarının her biri bir attribute ile belirtilmelidir. Bu mecburi bir kural değildir zira 'potansiyel yararlı bilgi parçası' verinin kullanıldığı yere göre değişir. Yukarıdaki örnekte eğer coğrafik konumun bir önemi yoksa adresleri tek bir attribute olarak belirtmek daha mantıklıdır.
- İlişkisel veritabanı modelinde kayıt dosyası bir tablo olarak gösterilir. Tablonun sütunları attribute isimlerini, satırları ise her bir kayıt örneğini oluşturur.



# Uygulama: İlişkisel Veritabanları

- Bir kayıt tipinin $A_1$, $A_2$, …, $A_n$ şeklinde n tane attribute' ten oluştuğunu düşünelim. Bu durumda herhangi bir $A_i$ attribute' u için bir veri girişleri kümesi olacaktır.
- $X_i$' ye de $A_i$ attribute' u ile elde edilen değerler kümesi diyelim. $X_i$ kümeleri zamana bağımlıdır ve kayıt dosyasına yeni girişler oldukça veya kayıt silindikçe değişir.
- Bu notasyona göre; verilen bir kayıt örneği her bir $x_i$, $X_i$ kümesine ait olmak üzere n-tuple (sıralı n-li, $(x_1, x_2,\ldots x_n)$) dır. Bunun anlamı tüm kayıt örnekleri n tane aynı tip bilgi parçasından oluşur.
- $x_i \in X_i$ olmak üzere tüm n-tupple' ların $(x_1, x_2, \ldots, x_n)$ kümesi $X_1 \times X_2 \times \cdots \times X_n$ kartezyen çarpımıdır.
- Bu yüzden R kayıt dosyası kartezyen çarpımın alt kümesidir

$$R \subseteq ( X_1 \times X_2 \times \cdots \times X_n).$$

# Uygulama: İlişkisel Veritabanları

**Örnek:** $A_1$, $A_2$, …, $A_5$ sırasıyla bağışlayanın_adı, bağışlayanın_adresi, bağışlayanın_telefonu, bağış_miktarı ve bağış_tarihi olsun.

- Her bir $A_i$ attribute' u için bu attribute'a karşılık gelen $X_i$ kümesi olduğunu varsayarız.
- O halde, bir kayıt örneği $x_i \in X_i$ olmak üzere 5-tuple' dır.
- Bu kayıt tipine göre önemli sayıda bilgi yinelemesi olur.
- Örneğin, bağış yapanın ismi, adresi ve telefonu her bağış yaptığında tekrar kaydedilir.
- Bu bilginin tutulduğu yerden kayıplara yol açacağı gibi kayıt dosyasının güncellenmesini de zorlaştırır.
- Mesela, iki bağış yapmış Mr. Giggs adres değiştirdi diyelim. Bu durumda, kayıt dosyasını güncelleştirmek için iki kayıtta da adresi değiştirmek gerekecektir.



# Uygulama: İlişkisel Veritabanları

- Bu sebeplerle veriyi aşağıdaki gibi iki ayrı kayıt dosyasına bölmek daha mantıklıdır.
- $A_1$, $A_2$, $A_3$: bağışlayanın_adı, bağışlayanın_adresi, bağışlayanın_telefonu
- $A_1$, $A_4$, $A_5$: bağışlayanın_adı, bağış_miktarı, bağış_tarihi
- Bu durumda orijinal veritabanındaki yineleme probleminden kurtulmuş oluruz ve daha kolay güncelleme yapabiliriz.
- Mevcut durumda veritabanı iki ilişkili kayıt dosyası içerir; birisi $X_1 \times X_2 \times X_3$ 'ün, diğeri $X_1 \times X_4 \times X_5$ 'ün alt kümesidir.
- Tabii ki, iki kayıt dosyasını bağışlayanın_adı attribute' u birbirine bağlar.

| ***bağnın_adı*** | ***bağnın_adresi*** | ***bağnın_telefonu*** |
|---|---|---|
| Giggs, R | 33 New Street, Manchester | 4614-3939 |
| Beatie, J | 24 Oaks Road, Southampton | 6578-4108 |
| Veron,S | 2A Great Oldtown, London | 2467-1297 |

| ***bağnın_adı*** | ***bağış_miktarı*** | ***bağış_tarihi*** |
|---|---|---|
| Giggs, R | £100 | Ocak 1997 |
| Giggs, R | £150 | Mart 1999 |
| Beatie, J | £300 | Ekim 1998 |
| Veron,S | £250 | Kasım 2000 |
| Veron,S | £500 | Aralık 1999 |

# Uygulama: İlişkisel Veritabanları

**Tanım:** $A_1, A_2, \ldots, A_n$ attribute' ler topluluğu olsun ve her bir $A_i$' ye ilişkin bir $X_i$ veri kümesi olduğunu düşünelim. **İlişkisel veritabanı** her biri bazı $X_i$ kümeleri arasındaki bağıntılar topluluğudur. Her bir bağıntı bir **kayıt dosyası**dır.

- Kayıt dosyasındaki kayıt örneklerine anahtar (key) ile erişilir. Key, tek bir kayıt örneğini belirten attribute' lar kümesidir, fakat bu kümenin hiçbir öz alt kümesi tek bir kayıt örneğini belirtme özelliğine sahip değildir.
- Pratikte bir çok olası anahtar seçme imkanı vardır. Key olarak kullanılabilecek attribute'lar kümesine **candidate key (aday anahtar)** denir. Bunlardan biri gerçek key olarak seçilir ve buna **primary (birincil) key** denir.
- Örneğimizde, {bağışlayanın_adı} herhangi iki bağış yapanın adının aynı olmaması durumunda Tablo 1 için bir candidate keydir. Bu durumda her bir kayıt örneği bağış yapanın adı ile belirtilebilir. Öte yandan iki farklı bağış yapan kişinin aynı adı taşıması durumunda {bağışlayanın_adı} key olmaz bunun yerine {bağışlayanın_adı, bağışlayanın_telefonu} attribute kümesi key olarak kullanılabilir.
- İlişkisel veritabanları üzerinde beş çeşit işlem yapılabilir.



# Uygulama: İlişkisel Veritabanları

## Selection (Seçme) $\sigma_c(R)$

- Selection işlemi kayıt dosyasından verilen kriter kümesini sağlayan kayıt örneklerini listeler.
- Örneğin, X şehrinde yaşayan müşterilerin tüm isim ve adres kayıtlarını listelemek bir selection örneğidir.
- Selection işlemini yeni kayıt dosyaları tanımlamak yani veritabanındaki kayıt dosyalarının alt kümeleri şeklinde düşünebiliriz.
- Bu yeni kayıt dosyaları muhtemelen geçicidir ve veritabanını oluşturan kayıt dosyaları kümesine eklenmezler.
- Aynı zamanda selection kayıt dosyasının tablo gösterimi şeklinde de tanımlanabilir. Bu yeni kayıt dosyaları gerekli attribute'lara sahip satırları çekerek elde edilir.
- Örneğin; GYTE veritabanında 'Ocak 1999'dan sonraki tüm bağışları seçmek' istediğimizde Tablo 3.3 'te gösterilen kayıt dosyasından ikinci, dördüncü ve beşinci satırlar elde edilecektir.
- Maaşı 40000 TL'den fazla olan çalışanların hepsini bulunuz:
  - $\sigma_{Maaş > 40000}$(Çalışanlar)

## Uygulama: İlişkisel Veritabanları

**İzdüşüm (Projection)** $\Pi_{A1,\ldots,An}(R)$

- Selection tablodaki belli satırları geri döndürürken projection işlemi sütunları döndürür.
- Sütunlar attribute' lara karşılık geldiğinden sonuçta ortaya çıkan kayıt dosyası orijinalden daha az sayıda attributelu kayıt tipine sahiptir.
- Projection işleminin resmi tanımı şöyledir: R, ($A_1$, …, $A_p$) tipinde bir kayıt dosyası ve q ≤ p ve her bir $B_i$ aynı zamanda R' nin attribute' u olmak üzere ($B_1$,…, $B_q$) kayıt tipi olsun.
- Yani, her bir B bir j için $A_j$' ye eşit olsun. Projection, kayıt örnekleri R' nin her bir kayıt örneklerinin $B_i$ attribute' larından oluşan ($B_1$,…, $B_q$) tipinde yeni kayıt dosyası tanımlar.
- Örnek: Çalışanlar tablosundaki TCKN ve Adı özelliklerinin izdüşümünü bulunuz:
  - $\Pi_{TCKN, Adı}$ (Çalışanlar)



## İzdüşüm Örneği

**Çalışanlar**

| TCKN | Adı | BölümNo | Maaş |
|---|---|---|---|
| 21343344 | Ahmet Kısa | 1 | 30000 |
| 21343354 | Ali Uzun | 1 | 32000 |
| 21343346 | Mehmet Kara | 2 | 45000 |

- Çalışanlar tablosundaki TCKN ve Adı özelliklerinin izdüşümünü bulunuz :

$$P_{TCKN, Adı}(Çalışanlar)$$

$P_{TCKN, Adı}$ (Çalışanlar)

| TCKN | Adı |
|---|---|
| 21343344 | Ahmet Kısa |
| 21343354 | Ali Uzun |
| 21343346 | Mehmet Kara |

# Örnek

Satma ilişkisi:

| Üretici | Araba | fiyat |
|---|---|---|
| Toyota | Sedan | 33000 |
| Toyota | Ticari | 32000 |
| Honda | Sedan | 30000 |
| Honda | Ticari | 32000 |

Fiyat := $PROJ_{Araba,fiyat}$(Satma):

| Araba | fiyat |
|---|---|
| Sedan | 33000 |
| Ticari | 32000 |
| Sedan | 30000 |



## Uygulama: İlişkisel Veritabanları

### Doğal Birleşim (Natural Join)

- GYTE veritabanının örnekteki gibi ikiye ayrıldığını düşünelim.
- Bu durumda bağış yapanların isimlerini, telefon numaralarının ve bağış miktarlarını nasıl alabiliriz?
- Buradaki problem bağış yapanın telefon numarası ile bağış miktarlarının farklı kayıt dosyalarında olmalarıdır.
- O halde kayıt dosyalarını birleştirerek üç attribute 'u da içeren yeni bir kayıt dosyası üretmemiz gerekir. İki dosyada ayrıca bağışlayanın_adresi ve bağış_tarihi de bulunur ve sonuçta oluşacak birleşmiş tabloda bu attributeler de bulunacaktır. Ancak bu bir sorun değildir zira projection ile bu dosyadan gerekli kayıt tipleri çekilebilir.
- Natural join işleminin matematiksel temeli şöyledir: R ve S, ($A_1$, …, $A_p$, $B_1$,…, $B_q$) ve ($A_1$, …, $A_p$, $C_1$,…, $C_r$) tipinde kayıt dosyaları olsun. R ve S' nin doğal birleşimi ($A_1$, …, $A_p$, $B_1$,…, $B_q$, $C_1$,…, $C_r$) tipinde yeni bir kayıt dosyasıdır. Doğal birleşimim oluşturan kayıt örneklerinin hepsi ($x_1$,…, $x_p$, $y_1$,…, $y_q$) ∈ R ve ($x_1$,…, $x_p$, $z_1$,…, $z_r$) ∈ S özelliğine sahip (p+q+r)-tuple ($x_1$,…, $x_p$, $y_1$,…, $y_q$, $z_1$,…, $z_r$) 'dır.

# Natural Join Örnek

Çalışanlar

| TCKN | Adı |
|---|---|
| 21343344 | Ahmet Kısa |
| 21343345 | Ali Uzun |

Bağımlılar

| TCKN | Çocuk_Adı |
|---|---|
| 21343344 | Ahmet |
| 21343344 | Mehmet |
| 21343345 | Mustafa |

**Çalışanlar** ⋈ **Bağımlılar** =

$P_{TCKN,\ Adı,\ Çocuk\_Adı}(s_{\ TCKN=TCKN2}(Çalışanlar \times P_{TCKN2,Çocuk\_Adı}(Bağımlılar))$

| TCKN | Adı | Çocuk_Adı |
|---|---|---|
| 21343344 | Ahmet Kısa | Ahmet |
| 21343344 | Ahmet Kısa | Mehmet |
| 21343345 | Ali Uzun | Mustafa |

```
SELECT A.TCKN, A.Adı, B.B_adı FROM Çalışanlar A
INNER JOIN Bağımlılar B on A.TCKN = B.TCKN;
```



# Uygulama: İlişkisel Veritabanları

## Birleşim ve Fark (Union and Difference)

- Verilen iki aynı kayıt tipinde R ve S kayıt dosyasının birleşimi ve farkı, bildiğimiz küme teorisindeki birleşim ve fark işlemlerine karşılık gelir.
- Bu yüzden R ∈ S, ve R ve S 'deki kayıt örneklerinin tamamını (listeyi tekrarlamadan) içeren kayıt dosyasıdır. R-S ise R de bulunan fakat S' de bulunmayan kayıt örneklerini içeren kayıt dosyasıdır.