Veritabanı Yönetim Sistemleri
1906003022015

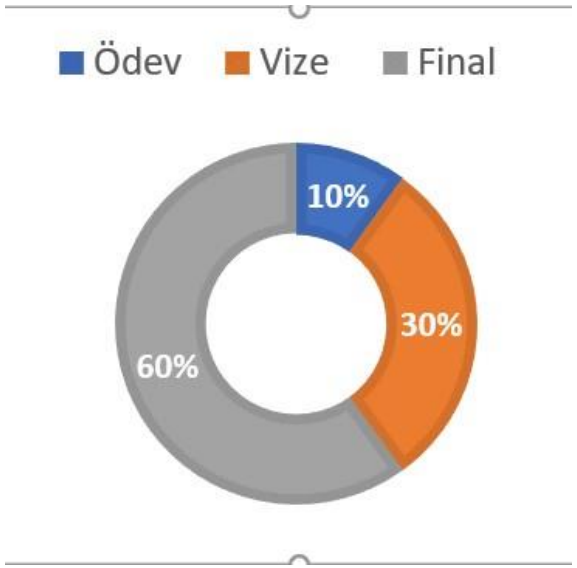
Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

Giriş

Ders Günü ve Saati:

Pazartesi: 09:15-12:45

- Devam zorunluluğu %70
- Uygulamalar MS SQL ve MongoDB üzerinde gerçekleştirilecektir.



HAFTA	KONULAR
Hafta 1	VT ve VTYS'ne giriş
Hafta 2	ER Veri Modeli
Hafta 3	İlişkisel Modeller, İlişkisel model tasarımı
Hafta 4	İlişkisel Cebir ve Hesaplamalar
Hafta 5	İlişkisel Sorgular, SQL giriş
Hafta 6	SQL ile veri tabanı programlama
Hafta 7	SQL-Kısıtlar:Veri-tipi,birincil-anahtar,ikinci-anahtar,
Hafta 8	Vize
Hafta 9	İlişkisel Veri Tabanı Tasarımı ve Normalizasyon
Hafta 10	yarı-yapısal veri modelleri, XML
Hafta 11	JSON
Hafta 12	İlişkisel olmayan DB, NoSQL
Hafta 13	NoSQL
Hafta 14	DBMS -Eşzamanlılık (Concurrency) Kontrolü

İlişkisel Cebir

İlişkisel veritabanı sistemlerinin, kullanıcılarının veritabanı örneklerini sorgulamasına yardımcı olabilecek bir sorgu dili ile donatılması beklenmektedir. İki tür sorgu dili vardır - ilişkisel cebir ve ilişkisel hesap.

Her veritabanı yönetim sistemi, kullanıcıların veritabanında depolanan verilere erişmesine izin vermek için bir sorgu dili tanımlamalıdır. İlişkisel Cebir , verilere farklı yollarla erişmek için veritabanı tablolarını sorgulamak için kullanılan prosedürel bir sorgu dilidir.

İlişkisel cebir, ilişkisel model üzerinde çalışan bir prosedürel sorgu dilidir. Bir sorgu dilinin amacı, veri tabanından veri almak veya veri üzerinde ekleme, güncelleme, silme gibi çeşitli işlemleri gerçekleştirmektir. İlişkisel cebirin yordamsal bir sorgu dili olduğunu söylediğimde, hangi verinin alınacağını ve nasıl alınacağını söylediği anlamına gelir.

Öte yandan ilişkisel hesaplama, prosedürel olmayan bir sorgu dilidir, yani hangi verilerin alınacağını söyler ancak nasıl alınacağını söylemez.

Giriş

Bir sorgunun girdileri ve çıktıları ilişkilerdir. Bir sorgu, her girdi ilişkisinin örnekleri kullanılarak değerlendirilir ve çıktı ilişkisinin bir örneğini üretir.

Sorgu dili, kullanıcının veritabanından bilgi talep ettiği bir dildir.

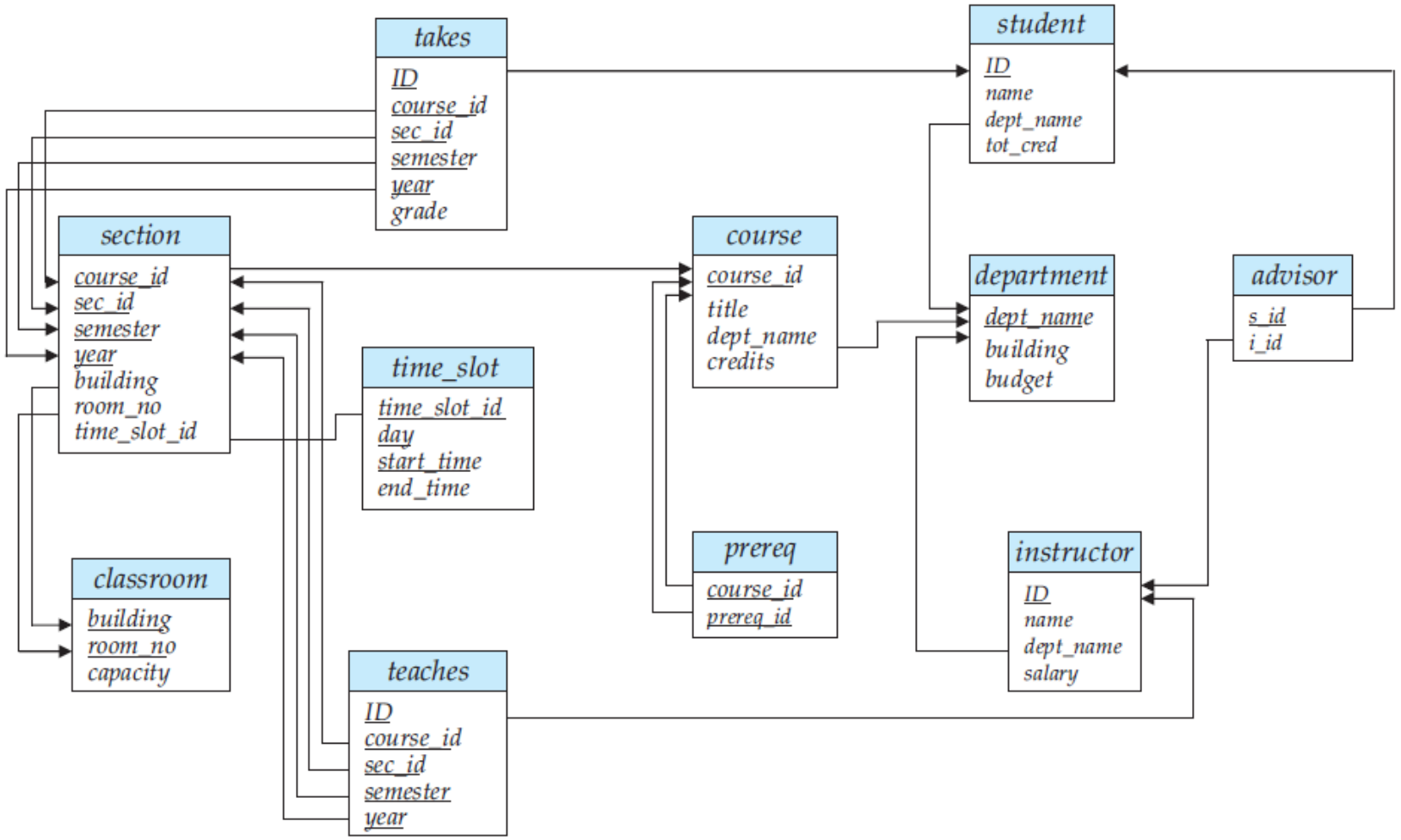
İlişkisel veritabanı sistemlerinin, kullanıcılarının veritabanı örneklerini sorgulamasına yardımcı olabilecek bir sorgu dili ile donatılması beklenmektedir. İki tür sorgu dili vardır –

- **ilişkisel cebir**
- **ilişkisel hesap.**

Giriş

- **ilişkisel cebir**
- **ilişkisel hesap.**
- İlişkisel cebir, girdi olarak bir veya iki ilişkiyi alan ve sonucu olarak yeni bir ilişki üreten bir dizi işlemden oluşur.
- İlişkisel hesap, bu sonucu elde etmek için herhangi bir özel cebirsel prosedür vermeden istenen sonucu tanımlamak için yüklem mantığını kullanır.

Giriş


takes
ID
course_id
sec_id
semester
year
grade
student
ID
name
dept_name
tot_cred
section
course_id
sec_id
semester
year
building
room_no
time_slot_id
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
time_slot
time_slot_id
day
start_time
end_time
prereq
course_id
prereq_id
instructor
ID
name
dept_name
salary
classroom
building
room_no
capacity
teaches
ID
course_id
sec_id
semester
year

Giriş

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Figure 2.9 Schema of the university database.

İlişkisel Cebir

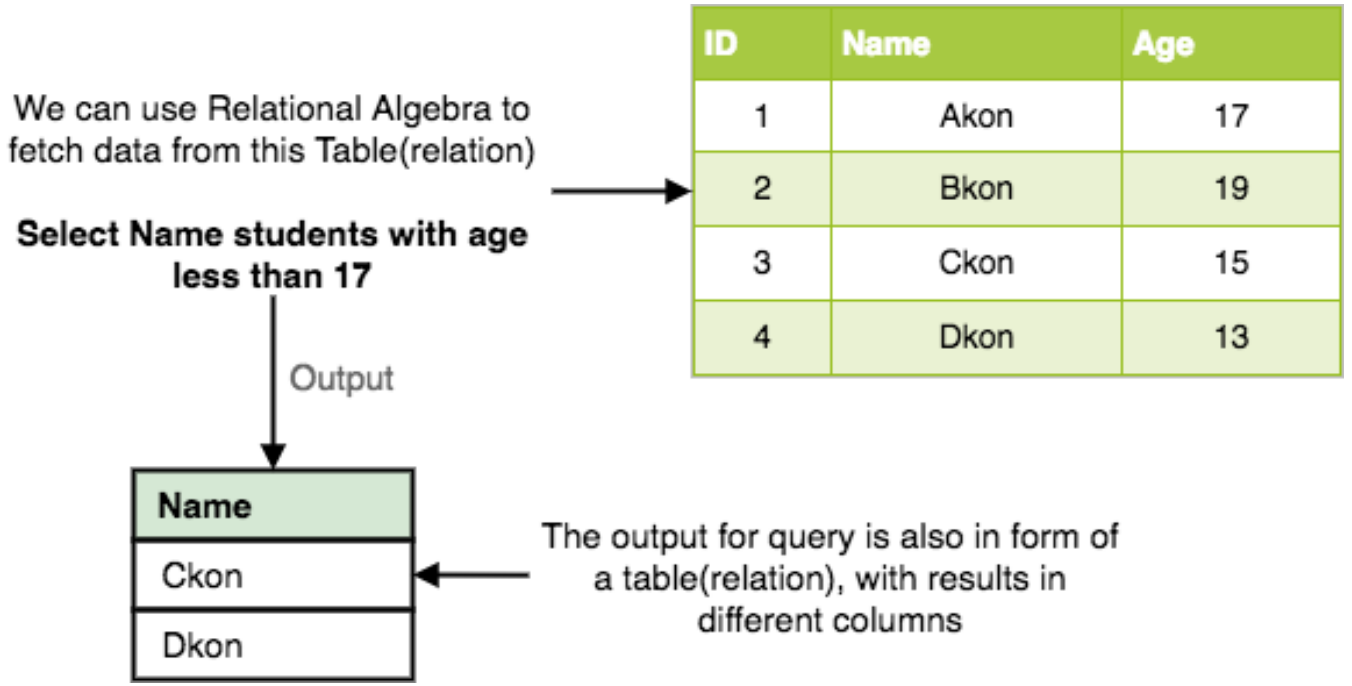
İlişkisel Cebir

- İlişkisel cebir, ilişkilerin örneklerini girdi olarak alan ve ilişkilerin örneklerini çıktı olarak veren prosedürel bir sorgulama dilidir.
- Cebirdeki sorgular, bir operatör koleksiyonu kullanılarak oluşturulur. Bir operatör tekli veya ikili olabilir .
- Temel bir özellik, cebirdeki her operatörün (bir veya iki) ilişki örneğini argüman olarak kabul etmesi ve sonuç olarak bir ilişki örneği döndürmesidir.
- Bu özellik, karmaşık bir sorgu (ilişkisel cebir ifadesi) oluşturmak için operatörler oluşturmayı kolaylaştırır; yinelemeli olarak bir ilişki, tek bir ifadeye uygulanan birli cebir operatörü veya iki ifadeye uygulanan bir ikili cebir operatörü olarak tanımlanır..

İlişkisel Cebir

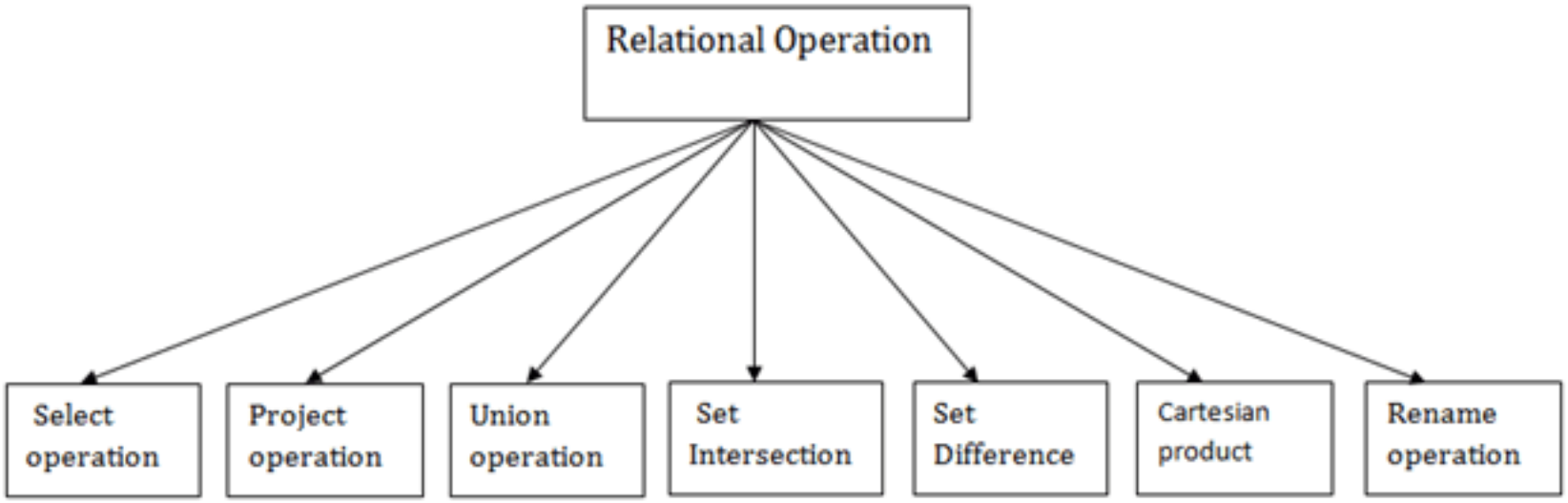
İlişkisel cebirde, girdi bir ilişkidir (verilere erişilmesi gereken tablo) ve çıktı da bir ilişkidir (kullanıcı tarafından istenen verileri tutan geçici bir tablo).

İlişkisel Cebir aynı anda tüm tabloda çalışır, bu nedenle tek tek tüm veri satırlarını (demetleri) yinelemek için döngüler vb. Kullanmak zorunda değiliz. Tek yapmamız gereken, verilere ihtiyacımız olan tablo adını belirtmek ve tek bir komut satırında, ilişkisel cebir sizin için veri almak için verilen tablonun tamamını geçecektir.



İlişkisel Cebir

İlişkisel cebir, ilişki örneklerini girdi olarak alan ve çıktı olarak ilişki örneklerini veren bir prosedürel sorgu dilidir. Sorgu yapmak için operatörleri kullanır. Bir operatör tekli veya ikili olabilir . İlişkileri girdi, verim ilişkilerini çıktı olarak kabul ederler. İlişkisel cebir, bir ilişki üzerinde yinelemeli olarak gerçekleştirilir ve ara sonuçlar da ilişkiler olarak kabul edilir.



İlişkisel Cebir

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural join)	$instructor \bowtie department$
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian product)	$instructor \times department$
	Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the two input relations.

İlişkisel cebirde işlem türleri

1. Temel İşlemler

2. Türetilmiş İşlemler

Temel / Temel İşlemler:
1. Operatör (σ) seçim
2. Projeksiyon (∏)
3. Birleşim (∪)
4. Farkı Ayarlama (-)
5. Kartezyen çarpım (X)
6. Yeniden adlandırma (ρ)

Türetilmiş İşlemler:
1. Doğal Birleşim (⋈)
2. Sol, Sağ, Tam dış birleşim (⟕, ⟖, ⟗)
3. Kesişim (∩)
4. Bölme (÷)

SET Operatörleri

Örnek

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

Figure 4.1 Instance $S1$ of Sailors

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
31	Lubber	8	55.5
44	guppy	5	35.0
58	Rusty	10	35.0

Figure 4.2 Instance $S2$ of Sailors

sid	*bid*	*day*
22	101	10/10/96
58	103	11/12/96

Figure 4.3 Instance $R1$ of Reserves

1. Seçim (SELECT) Operatör(σ)

Seçim (SELECT) Operatörü sigma (σ) ile gösterilir ve verilen koşulu karşılayan bir ilişkideki (veya tablodaki) tuple'ları (veya satırları) bulmak için kullanılır.

Sözdizimi: $\sigma_p(r)$

Burada, σ Select Predicate'i temsil eder, r ilişkinin adıdır (veri aramak istediğiniz tablo adı) ve p veriler tarafından karşılanması gereken koşulları belirttiğimiz edat mantığıdır. -Edat matıkları **tekli** ve **ikili** gibi operatörler =, <, >

1. Seçim (SELECT) Operatör(σ)

Sözdizimi: $\sigma_p(r)$

$\sigma_{age > 17}$ (Student)

Bu, tuple'ları (satırları) **Öğrenci** tablosundan getirecek ve bu **yaş** için **17'den** büyük olacaktır .

İki koşulu belirtmek için and, orvb. İşleçlerini de kullanabilirsiniz, örneğin,

$\sigma_{age > 17 \text{ and } gender = 'Male'}$ (Student)

Bu tablo gelen dizilerini (satırlar) dönecektir **Öğrenci** daha 17'ye göre yaş, erkek öğrencilerin bilgilerle

σ seçim tahmini için kullanılır
r , ilişki için kullanılır
p , AND OR ve NOT gibi bağlayıcıları kullanabilen bir önermesel mantık formülü olarak kullanılır. Bu ilişkisel, =, ≠, ≥, <,>, ≤ gibi ilişkisel operatörler olarak kullanılabilir.

SQL'de aynı amaç için kullanılan bir where cümlesi olarak düşünebilirsiniz

1. Seçim (SELECT) Operatör(σ)

Sözdizimi: $\sigma_p(r)$

Örneğin: KREDİ İlişkisi

BRANCH_NAME	LOAN_NO	MİKTAR
Şehir merkezi	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Roundhill	L-11	900
Perryride	L-16	1300

$$\sigma \text{ BRANCH\_NAME} = \text{"perryride"} \ (\text{KREDİ})$$

BRANCH_NAME	LOAN_NO	MİKTAR
Perryride	L-15	1500
Perryride	L-16	1300

1. Seçim (SELECT) Operatör(σ)

$$\sigma_{subject = "database"}(Books)$$

Çıktı - Konunun 'veritabanı' olduğu kitaplardan demetleri seçer.

$$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$$

Çıktı - Konunun 'veritabanı' ve 'fiyat'ın 450 olduğu kitaplardan dizileri seçer.

$$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$$

Çıktı - Konunun 'veritabanı' ve 'fiyat'ın 450 olduğu kitaplardan veya 2010'dan sonra yayınlanan kitaplardan dizileri seçer.

1. Seçim (SELECT) Operatör(σ)

sid	*sname*	*rating*	*age*
28	yuppy	9	35.0
58	Rusty	10	35.0

Figure 4.4 $\sigma_{rating>8}(S2)$

2. Projeksiyon (Project) Operatörü (π)

Proje işlemi, bir ilişkinin yalnızca belirli bir nitelik kümesini yansıtmak için kullanılır. Basit bir deyişle, **Öğrenci** tablosundaki tüm öğrencilerin sadece **isimlerini** görmek istiyorsanız, Proje İşlemini kullanabilirsiniz.
Yalnızca istenen sütunları veya öznitelikleri yansıtır veya gösterir ve ayrıca yinelenen verileri sütunlardan kaldırır.

Sözdizimi: $\prod_{A1, A2...}(r)$
burada A1, A2 vb. öznitelik isimleridir (sütun adları).
Örneğin,

$$\prod_{Name, Age}(Student)$$

Yukarıdaki ifade, **Öğrenci** tablosundaki tüm veri satırları için bize yalnızca **Ad** ve **Yaş** sütunlarını gösterecektir .

2. Projeksiyon (Project) Operatörü (π)

Tablo : MÜŞTERİ

CUSTOMER_ID	CUSTOMER_NAME	Customer_City
C10100	Steve	Agra
C10111	Raghu	Agra
C10115	Chaitanya	Noida
C10117	Ajeet	Delhi
C10118	Carl	Delhi

$$\prod \text{Customer\_Name} , \text{Customer\_City} (\text{MÜŞTERİ})$$

Customer_Name	Customer_City
Steve	Agra
Raghu	Agra
Chaitanya	Noida
Ajeet	De

2. Projeksiyon (Project) Operatörü (π)

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

> ∏ NAME, CITY (CUSTOMER)

2. Projeksiyon (Project) Operatörü (π)

∏ NAME, CITY (CUSTOMER)

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

2. Projeksiyon (Project) Operatörü (π)

sname	*rating*
yuppy	9
Lubber	8
guppy	5
Rusty	10

Figure 4.5 $\pi_{sname,rating}(S2)$

sname	*rating*
yuppy	9
Rusty	10

Figure 4.7 $\pi_{sname,rating}(\sigma_{rating>8}(S2))$

2. Birleştirme (Union) Operasyonu (U)

- Bu işlem, iki ilişkiden (tablolardan) veya geçici ilişkiden (başka bir işlemin sonucu) veri almak için kullanılır.
- Bu işlemin çalışması için, belirtilen ilişkiler (tablolar) aynı sayıda özniteliğe (sütun) ve aynı öznitelik alanına sahip olmalıdır. Ayrıca yinelenen demetler otomatik olarak sonuçtan çıkarılır.

- Birleşim operatörü U sembolü ile gösterilir ve iki tablodan (ilişkiler) tüm satırları (tuplelar) seçmek için kullanılır.
- Diyelim ki iki ilişkimiz var R1 ve R2'nin her ikisi de aynı sütunlara sahip ve bu ilişkilerden tüm tuple'ları (satırları) seçmek istiyoruz, o zaman bu ilişkilere birleşim operatörünü uygulayabiliriz.

- Not: Her iki tabloda da bulunan satırlar (demetler) birleşim kümesinde yalnızca bir kez görünecektir. Kısaca sendika operasyonundan sonra mükerrer kayıt bulunmadığını söyleyebilirsiniz.

2. Birleştirme (Union) Operasyonu (∪)

Sözdizimi: $A \cup B$

burada A ve B ilişkilerdir.

Örneğin, **RegularClass** ve **ExtraClass** adlı iki **tablomuz varsa** , her ikisinde de **öğrencinin** adını kaydetmek için bir sütun **öğrencisi** olur, o zaman,

$$\prod_{Student}(RegularClass) \cup \prod_{Student}(ExtraClass)$$

2. Birleştirme (Union) Operasyonu (∪)

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s \}$$

Burada **r** ve **s** veritabanı ilişkileri veya ilişki sonuç kümesi (geçici ilişki) alınarak belirlenir.
Bir birleşim işleminin geçerli olması için aşağıdaki koşulların geçerli olması gerekir

- **r** ve **s** aynı sayıda özniteliğe sahip olmalıdır.
- Öznitelik alanları uyumlu olmalıdır.
- Yinelenen demetler otomatik olarak ortadan kaldırılır.

$$\prod_{author}(Books) \cup \prod_{author}(Articles)$$

Çıktı - Bir kitap veya makale veya her ikisini birden yazan yazarların adlarını yansıtır.

2. Birleştirme (Union) Operasyonu (∪)

DEPOSITOR

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

$$\prod \text{CUSTOMER\_NAME}(\text{BORROW}) \cup \prod \text{CUSTOMER\_NAME}(\text{DEPOSITOR})$$

2. Birleştirme (Union) Operasyonu (∪)

$\prod$ CUSTOMER_NAME (BORROW) ∪ $\prod$ CUSTOMER_NAME (DEPOSITOR)

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

2. Birleştirme (Union) Operasyonu (∪)

Table R1 is as follows –

Regno	Branch	Section
1	CSE	A
2	ECE	B
3	MECH	B
4	CIVIL	A
5	CSE	B

Table R2 is as follows –

Regno	Branch	Section
1	CIVIL	A
2	CSE	A
3	ECE	B

```
∏ branch, section (R1) ∪ ∏ branch, section (R2)
```

2. Birleştirme (Union) Operasyonu (U)

```
∏ branch, section (R1) ∪ ∏ branch, section (R2)
```

Output

Branch	Section
CSE	A
ECE	B
MECH	B
CIVIL	A
CSE	B

2. Birleştirme (Union) Operasyonu (∪)

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0
28	yuppy	9	35.0
44	guppy	5	35.0

Figure 4.8 $S1 \cup S2$

3. Fark (set-difference) Opeatörü (-)

Bu işlem, bir ilişkide bulunan ve ikinci ilişkide bulunmayan verileri bulmak için kullanılır. Bu operasyon, tıpkı Birlik operasyonu gibi iki ilişkide de geçerlidir.

Sözdizimi: A - B

burada A ve B ilişkilerdir.

Örneğin, normal sınıfa devam eden ancak fazladan sınıfa gitmeyen öğrencilerin adını bulmak istiyorsak, aşağıdaki işlemi kullanabiliriz:

$\prod_{Student}(RegularClass) - \prod_{Student}(ExtraClass)$

$$\prod_{author}(Books) - \prod_{author}(Articles)$$

3. Fark (set-difference) Opeatörü (-)

DEPOSITOR

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

3. Fark (set-difference) Opeatörü (-)

Input:

> ∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

3. Fark (set-difference) Opeatörü (-)

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0

Figure 4.10 $S1 - S2$

4. Kartezyen Çarpım (X)

Kartezyen çarpım, **X** sembolü ile gösterilir. Diyelim ki, R1 ve R2 olmak üzere iki ilişkimiz var, o zaman bu iki ilişkinin kartezyen çarpımı (R1 X R2), ilk R1 ilişkisinin her bir demetini, ikinci R2 ilişkisinin her bir demeti ile birleştirecektir.

Bu, iki farklı ilişkiden (tablolardan) gelen verileri bir araya getirmek ve birleşik ilişkiden verileri almak için kullanılır.

Sözdizimi: A X B

Örneğin sabah saatlerinde yapılan Normal ve Ekstra Sınıf bilgilerini bulmak istiyorsak aşağıdaki işlemi kullanabiliriz:

$$\sigma_{time = 'morning'} (RegularClass \ X \ ExtraClass)$$

İşe Yukarıdaki sorgu için hem **RegularClass** ve **ExtraClass** niteliği olmalıdır .

Gösterim **- r X s**

Burada **r** ve **s** ilişkiler ve bunların çıkış olarak tanımlanır -

$$r \ X \ s = \{qt \mid q \in r \text{ ve } t \in s\}$$

4. Kartezyen Çarpım (X)

ÇALIŞAN

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	Bir
2	Harry	C
3	John	B

BÖLÜM

DEPT_NO	DEPT_NAME
Bir	Pazarlama
B	Satış
C	Yasal

ÇALIŞAN X BÖLÜMÜ

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	Bir	Bir	Pazarlama
1	Smith	Bir	B	Satış
1	Smith	Bir	C	Yasal
2	Harry	C	Bir	Pazarlama
2	Harry	C	B	Satış
2	Harry	C	C	Yasal
3	John	B	Bir	Pazarlama
3	John	B	B	Satış
3	John	B	C	Yasal

4. Kartezyen Çarpım (X)

$R1$ ve $S1$'in her ikisinin de ***sid*** adında bir alanı olduğundan, alan adları konusundaki konvansiyonumuz gereği, $S1\ X\ R1$'deki karşılık gelen iki alan adsızdır ve yalnızca Şekil 4.11'de göründükleri konumla belirtilir. $S1\ X\ R1$ 'deki alanlar, $R1$ ve $S1$'deki karşılık gelen alanlarla aynı etki alanlarına sahiptir. Şekil 4.11'de *sid*, kalıtsal bir alan adı olmadığını vurgulamak için parantez içinde listelenmiştir; yalnızca karşılık gelen etki alanı devralınır.

(*sid*)	*sname*	*rating*	*age*	(*sid*)	*bid*	*day*
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

Figure 4.11 $S1 \times R1$

5. Yeniden Adlandırma Operatörü (ρ)

Bu işlem, «Select, Project» vb. gibi sonuçları döndüren herhangi bir sorgu işlemi için çıktı ilişkisini yeniden adlandırmak veya bir ilişkiyi (tablo) yeniden adlandırmak için kullanılır.

Sözdizimi: ρ(RelationNew, RelationOld)

Birleştirme (join) işlemleri için de kullanılır:

- Doğal Birleştirme
- Dış Birleştirme
- Teta birleştirme vb.

5. Yeniden Adlandırma Operatörü (ρ)

Bir ilişkisel cebir ifadesinin sonucunun, mümkün olan her durumda, argüman (girdi) ilişkisi örneklerinden alan adlarını doğal bir şekilde miras almasını sağlayan alan adı kurallarını benimsemeye özen gösterdik.

Ancak bazı durumlarda isim çakışmaları ortaya çıkabilir; (S1x R1'de sid). Bu nedenle, bir ilişkisel cebir ifadesi tarafından tanımlanan bir ilişki örneğinin alanlarına açıkça adlar verebilmek uygundur. Aslında, alt ifadelerin sonuçlarına isimler vererek büyük bir cebir ifadesini daha küçük parçalara ayırabilmemiz için örneğe bir isim vermek genellikle uygundur.

5. Yeniden Adlandırma Operatörü (ρ)

- Bu amaçla bir yeniden adlandırma operatörü $\boldsymbol{\rho}$ kullanılır.
- $\rho(R(F), E)$ ifadesi keyfi bir ilişkisel cebir ifadesi E yi alır ve R adlı bir (new) ilişkinin bir örneğini (instance) döndürür. R, E'nin sonucuyla aynı satırları (tuple) içerir ve E ile aynı şemaya sahiptir, ama bazı alanlar yeniden adlandırılır. (F, yeniden adlandırma listesi)
- R ilişkisindeki alan adları, "eski_ad -> yeni_ad" veya "konum -> yeni_ad" biçimindeki terimlerin bir listesi olan F yeniden adlandırma listesinde yeniden adlandırılan alanlar dışında E'deki ile aynıdır.
- $\boldsymbol{\rho}$ 'nin iyi tanımlanmış olması için, alanlara yapılan referanslar belirli (unambiguous) olabilir ve sonuçtaki hiçbir iki alan aynı ada sahip olmamalıdır.

5. Yeniden Adlandırma Operatörü (ρ)

For example, the expression $\rho(C(1 \rightarrow sid1, 5 \rightarrow sid2), S1 \times R1)$ returns a relation that contains the tuples shown in Figure 4.11 and has the following schema: C(*sid1:* `integer`, *sname:* `string`, *rating:* `integer`, *age:* `real`, *sid2:* `integer`, *bid:* `integer`, *day:* `dates`).

(*sid*)	*sname*	*rating*	*age*	(*sid*)	*bid*	*day*
22	Dustin	7	45.0	22	101	10/10/96
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	22	101	10/10/96
31	Lubber	8	55.5	58	103	11/12/96
58	Rusty	10	35.0	22	101	10/10/96
58	Rusty	10	35.0	58	103	11/12/96

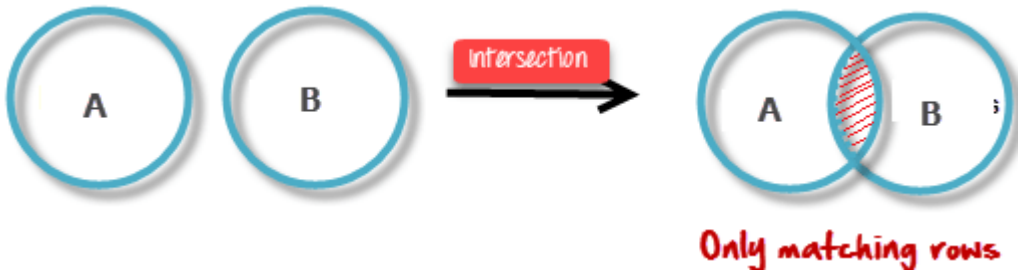
Figure 4.11 $S1 \times R1$

Kesişim (ıntersection) Operatörü (∩)

Kesişim operatörü ∩ sembolü ile gösterilir ve iki tablodan (ilişkiler) ortak satırları (tuplelar) seçmek için kullanılır.

Diyelim ki iki ilişkimiz var R1 ve R2 her ikisinin de aynı sütuna sahip ve her iki ilişkide de bulunan tüm bu tuplelarları (satırları) seçmek istiyoruz, bu durumda bu iki ilişki R1 ∩ R2 üzerinde kesişim işlemi uygulayabiliriz.
Not: Sonuç kümesinde yalnızca her iki tabloda da bulunan satırlar görünecektir.
Kesişim Operatörünün Sözdizimi (∩)

```
tablo_adı1 ∩ tablo_adı2
```



Kesişim (ıntersection) Operatörü (∩)

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

Kesişim (ıntersection) Operatörü (∩)

KURS

Ders_Kimliği	Öğrenci_Adı	Öğrenci_Kimliği
C101	Aditya	S901
C104	Aditya	S901
C106	Steve	S911
C109	Paul	S921
C115	Lucy	S931

ÖĞRENCİ

Student_Id	Student_Name	Student_Age
S901	Aditya	19
S911	Steve	18
S921	Paul	19
S931	Lucy	17
S941	Carl	16
S951	Rick	18

Kesişim (ıntersection) Operatörü (∩)

Sorgu:

```
∏ Öğrenci_Adı ( DERS ) ∩ ∏ Öğrenci_Adı ( ÖĞRENCİ )
```

Çıktı:

```
Öğrenci_Adı ------------ Aditya Steve Paul Lucy
```

Join (⋈) Operasyonları

Birleştirme (join) işlemi, iki veya daha fazla ilişkiden gelen bilgileri birleştirmenin en yaygın kullanılan yoludur.

Types of JOIN:

Inner Joins:
- Theta join
- EQUI join
- Natural join

Outer join:
- Left Outer Join
- Right Outer Join
- Full Outer Join

Join (⋈) Operasyonları

Theta (θ) Join

Teta birleşimi, teta koşulunu sağlamaları koşuluyla farklı ilişkilerden demetleri birleştirir. **Birleştirme koşulu θ** sembolü ile gösterilir

```
R1 ⋈θ R2
```

$$R \bowtie_c S = \sigma_c(R \times S)$$

Join (⋈) Operasyonları

Theta (θ) Join

Student		
SID	**Name**	**Std**
101	Alex	10
102	Maria	11

Subjects	
Class	**Subject**
10	Math
10	English
11	Music
11	Sports

Join (⋈) Operasyonları

Theta (θ) Join

$$STUDENT \bowtie_{Student.Std = Subject.Class} SUBJECT$$

Student_detail				
SID	**Name**	**Std**	**Class**	**Subject**
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

Join (⋈) Operasyonları

Theta (θ) Join

(*sid*)	*sname*	*rating*	*age*	(*sid*)	*bid*	*day*
22	Dustin	7	45.0	58	103	11/12/96
31	Lubber	8	55.5	58	103	11/12/96

Figure 4.12 $S1 \bowtie_{S1.sid<R1.sid} R1$

Join (⋈) Operasyonları

Equijoin Join

Theta birleştirmesi yalnızca eşitlik (=) karşılaştırma operatörünü kullandığında, bunun equijoin olduğu söylenir. Yukarıdaki örnek, equijoin'e karşılık gelir.

$$STUDENT \bowtie_{Student.Std = Subject.Class} SUBJECT$$

Student_detail

SID	Name	Std	Class	Subject
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

sid	*sname*	*rating*	*age*	*bid*	*day*
22	Dustin	7	45.0	101	10/10/96
58	Rusty	10	35.0	103	11/12/96

Figure 4.13 $S1 \bowtie_{R.sid=S.sid} R1$

Join (⋈) Operasyonları

Natural Join (⋈)

Doğal birleştirme herhangi bir karşılaştırma operatörü kullanmaz. Kartezyen bir ürünün yaptığı gibi birleştirmez. Natural Join işlemini ancak iki ilişki arasında en az bir ortak öznitelik varsa gerçekleştirebiliriz. Ayrıca, öznitelikler aynı ada ve etki alanına sahip olmalıdır.

Doğal birleştirme, her iki ilişkideki niteliklerin değerlerinin aynı olduğu eşleşen nitelikler üzerinde hareket eder.

Join (⋈) Operasyonları

Natural Join (⋈)

EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

Join (⋈) Operasyonları

Natural Join (⋈)

Operation: (EMPLOYEE ⋈ SALARY)

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Join (⋈) Operasyonları

Outer Joins:

Theta Join, Equijoin ve Natural Join, iç birleşimler olarak adlandırılır. Bir iç birleşim, yalnızca eşleşen niteliklere sahip olan demetleri içerir ve geri kalanlar, ortaya çıkan ilişkide atılır. Bu nedenle, katılan ilişkilerdeki tüm demetleri ortaya çıkan ilişkiye dahil etmek için dış birleşimleri kullanmamız gerekir. Üç tür dış birleşim vardır - sol dış birleşim, sağ dış birleşim ve tam dış birleşim.

Join (⋈) Operasyonları

Outer Joins: Left Outer Join(A ⟕ B)

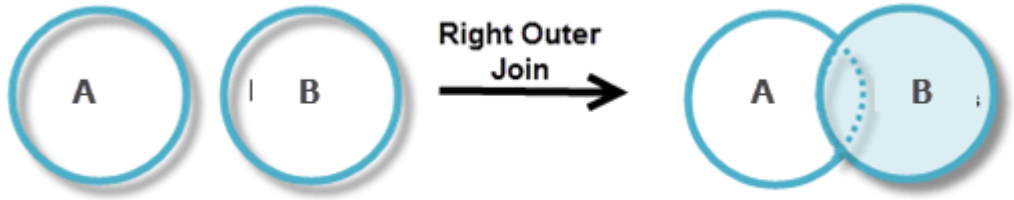
Sol ilişkideki tüm demetler, R, sonuçtaki ilişkiye dahil edilir. R'de Sağ ilişki S'de eşleşen herhangi bir demet olmadan demetler varsa, sonuçta ortaya çıkan ilişkinin S-öznitelikleri NULL yapılır.

Courses ⟕ HoD			
A	B	C	D
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya

Join (⋈) Operasyonları

Outer Joins: Right Outer Join: (R ⟖ S)

Sağ dış birleşimde, işlem, tüm demetlerin doğru ilişkide tutulmasına izin verir. Ancak, sol ilişkide eşleşen bir demet bulunmazsa, birleştirme sonucundaki sol ilişkinin nitelikleri boş değerlerle doldurulur.



All rows from Right Table.

Courses ⟖ HoD			
A	**B**	**C**	**D**
100	Database	100	Alex
102	Electronics	102	Maya
---	---	104	Mira

Join (⋈) Operasyonları

Outer Joins: Full Outer Join: (R ⟗ S)

Tam bir dış birleşimde, eşleşme koşulundan bağımsız olarak her iki ilişkideki tüm demetler sonuca dahil edilir.

Courses ⟗ HoD			
A	B	C	D
100	Database	100	Alex
101	Mechanics	---	---
102	Electronics	102	Maya
---	---	104	Mira

İlişkisel Hesap

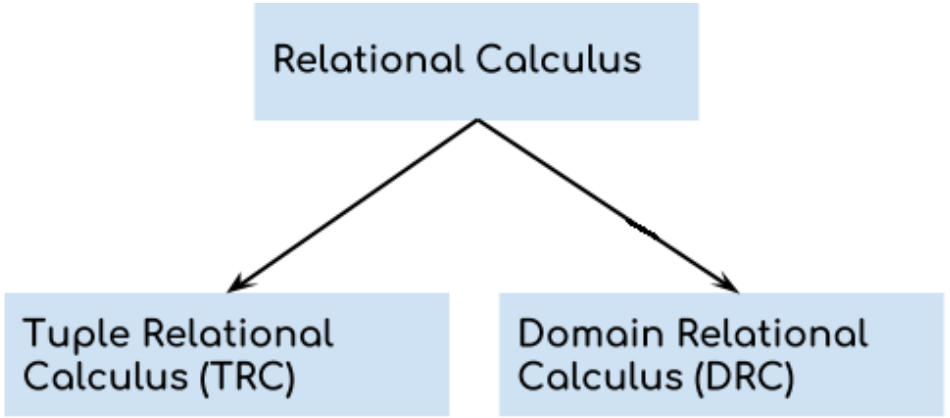
İlişkisel Hesap

Verileri almak için yordamsal bir sorgu dili olan ve nasıl yapıldığını da açıklayan İlişkisel Cebir'in aksine, yordamsal olmayan sorgu dilinde **İlişkisel Hesaplama** ve sorgunun nasıl çalışacağı veya verilerin nasıl alınacağı hakkında hiçbir açıklama yoktur. Sadece ne yapılacağına odaklanır, nasıl yapılacağına değil.

İlişkisel Cebirin aksine, İlişkisel Hesaplama prosedürel olmayan bir sorgu dilidir, yani ne yapılacağını söyler ama nasıl yapılacağını asla açıklamaz.

İlişkisel Hesap Türleri

1. Tuple İlişkisel Hesap (TRC)
2. Etki Alanı İlişkisel Hesabı (DRC)



1. Tuple İlişkisel Hesap (Tuple Relational Calculus- TRC)

Tuple ilişkisel hesaplamada, verilen koşula göre demetleri filtrelemeye çalışılır.

Sözdizimi: { T | Condition }

Bu ilişkisel hesap formunda, bir tuple değişkeni tanımlıyoruz, bir koşulla birlikte başlığın aranacağı tablo (ilişki) adını belirtiyoruz.

Ayrıca , sonuçta yalnızca belirli bir özniteliği (sütun) elde etmek için tuple değişkeni ile bir nokta operatörü (.) kullanarak sütun adını da belirtebiliriz .

Pek çok bilgi, doğru! İçine girmesi için biraz zaman tanıyın.

Bir tuple değişkeni bir isimden başka bir şey değildir, herhangi bir şey olabilir, genellikle bunun için tek bir alfabe kullanıyoruz, bu yüzden diyelim ki T, bir tuple değişkenidir.

```
{T | P (T)} veya {T | Koşul (T)}
```

1. Tuple İlişkisel Hesap (Tuple Relational Calculus- TRC)

Yaşı 30'dan büyük olan öğrencilerin soyadını görüntülemek için sorgu

```
{ t . Soyadı | Öğrenci ( t ) VE t . yaş > 30 }
```

Yukarıdaki sorguda | ile ayrılmış iki bölüm görebilirsiniz. sembol. İkinci bölüm koşulu tanımladığımız yerdir ve ilk bölümde seçilen tuplelar için görüntülemek istediğimiz alanları belirtiyoruz.

Yukarıdaki sorgunun sonucu şöyle olacaktır:

```
Soyadı --------- Singh
```

Soyadının 'Singh' olduğu öğrencilerin tüm ayrıntılarını görüntülemek için sorgu

```
{ t | Öğrenci ( t ) VE t . Last_Name = 'Singh' }
```

```
FIRST_NAME'in last_name Yaş ---------- --------- ---- Ajeet Singh 30 Chaitanya
Singh 31
```

2. Etki Alanı İlişkisel Hesabı (Domain Relational Calculus- DRC)

Alan ilişkisel analizinde filtreleme, tuple değerlerine göre değil, özniteliklerin alanına göre yapılır.

Sözdizimi: { c1, c2, c3, ..., cn | F(c1, c2, c3, ... ,cn)}

burada, c1, c2 ... vb özniteliklerin alanını (sütunlar) temsil eder Fve verilerin getirilmesi için koşulu içeren formülü tanımlar.

Örneğin,

{< name, age > | ∈ Student ∧ age > 17}

Yine yukarıdaki sorgu **Öğrenci** tablosundaki 17 yaşından büyük öğrencilerin isimlerini ve yaşlarını döndürecektir .

2. Etki Alanı İlişkisel Hesabı (Domain Relational Calculus- DRC)

Öğrenci yaşının 27'den büyük olduğu öğrencilerin adını ve yaşını bulma sorgusu

```
{< Ad , Yaş > | ∈ Öğrenci ∧ Yaş > 27 }
```

```
FIRST_NAME'in Yaş ---------- ---- Ajeet 30 Chaitanya 31 Carl 28
```

Mantıksal operatörler için kullanılan semboller şunlardır: VE için ∧, VEYA için ∨ ve DEĞİL için ¬ .

2. Etki Alanı İlişkisel Hesabı (Domain Relational Calculus- DRC)

Sembol	Adı	Yorumlama
$\emptyset$	Boş Küme	
$\varnothing$		$\{n \in \mathbb{N} : 1 < n^2 < 4\} = \varnothing$
$\{\}$		
$\in$	Eleman	
$\notin$	Eleman değil	
$\subseteq$	Alt küme	
$\subset$		
$\supseteq$	Kapsar	
$\supset$		
$\cup$	Birleşim	
$\cap$	Kesişim	

2. Etki Alanı İlişkisel Hesabı (Domain Relational Calculus- DRC)

Sembol	Adı	Yorumlama
A^c	A'nın tümleyeni	
$\setminus$	Fark	$A \setminus B$ (A kümesinin B kümesinden farkı)
$/$		A/B : A kümesinin B bağıntısına bölüm kümesi
A_X^c	A kümesinin X evreni içerisindeki tümleyeni	
$\prod$	Çoklu kartezyen çarpım	
$\bigcup$	Çoklu birleşim	
$\bigcap$	Çoklu kesişim	
$\biguplus$	Ayrık birleşim	birleşen kümeler birbirinden ayrık, ikişerli kesişimleri boş

2. Etki Alanı İlişkisel Hesabı (Domain Relational Calculus- DRC)

Sembol	Adı	Yorumlama
$=$	Eşittir	$x = y$ (x eşittir y'ye)
$\neq$	Eşit değildir	$x \neq y$ (x eşit değildir y'ye). **2 + 2 ≠ 5**
$\wedge$	Ve	
$\vee$	Veya	
$\Rightarrow$	İse, Gerektirir, İçin gerek şart	
$\Leftrightarrow$	Ancak ve ancak, İçin gerek ve yeter şart	
$\Leftarrow$	Ancak, İçin yeter şart	
$\forall$	Her, Bütün	
$\exists$	Vardır, En az bir	
$\exists!$	Yalnızca bir tane vardır, Vardır tek türlü	