

*Karadeniz Teknik Üniversitesi*

*Bilgisayar Mühendisliği*

**VERİ TABANI**



*Burkay Durdu*



Veritabanı nedir?

⇒ Verilerin saklandığı bir platformdur.

DBMS → Data Base Management System (Veritabanı yönetim sistemi)

→ Oracle, MySQL, mssql, DB2



→ Verilerin tutulduğu diskteki yer.

→ Sorgular var. Eczanede ilaç stoku, bilet satış sistemleri bunlar veritabanı kullanır.

DBMS ⇒ Senkroniasyonu ortadan kaldırıyor olayın senkroniasyonunu sağlıyor.

→ Elektrik kesimlerinde "recover" yapıyor önceden

→ Yeterlilik sağlıyor hizlendirme

→ Sorguların olurda optimum şekilde hızlı bir yol izliyor.

Ogrenci

(Entity) → varlık (Kendisine ait özellikleri olan şeylere)

Ders

Ono

Ad

Soyad

Did

İsmi

Kredi

Entityler database'de tablolara karşılık düşer.  
tablolardı içerisinde verilerin tutulduğu yer  
tablo varlığındır.

tablolardı isimleri olan içerisinde verileri olan veriyapılardır.  
tablolardı veri tabanında bulunan veriyapılardır.

tablolardı içerisinde özellikleri göre veriler açılın yapılardır.

field → alan

attribute → özellik  
tablo içi

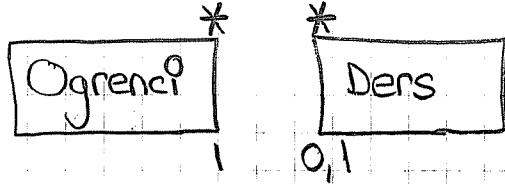
bunlar tablolara karşılık düşer.

Yukarıdakı tablodı Öğrencilerin aldığı dersleri tutucuk bir  
tablo yapacağız. Yukarıda öğrenci tablosuna DersId acıksık  
tablodı büyüktek kişi 20 dersde alabilir. bu tablo çok büyük olacaktır.

Ono

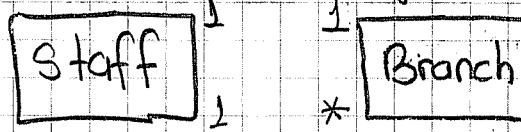
Did

bir tablo oluşturuyoruz.



Cogun, coga ilişkisi var ise  
 farklı bir table olmaz gerekecektir.

Personel



1 staff min 1 haq subbede olsisir  
 1 " " max " "

1 bir subbede min 1 haq staff olsisir,  
 (sol) \* " " " max " " " "

burda  $1 \rightarrow *$  ilişkisi vardır. Bir tarafın primary onchtar değeri  
 diğer onchtar olur.

Primary key foreign key

Staff

Sno	Sname	Ssurname	branchno
Branch			foreign key
bno	city	address	
Primary Key			

Primary key güncellendig sonan foreign key olan yerdeki  
 beside güncellenmektedir, bu gibi halde vardır.

Primary key  $\Rightarrow$  her bir alt ornegi tek basina tonimlaya bilen  
 Öğrenci numarası, tc kimlik numarası

bir tabloda birden fazla primary key olabilir.

Alternative key  $\rightarrow$  alternatif primary key

Not: Her database primary key numeric bir değer verilmemesi gereklidir.

foreign key = Bir tablouun primary key baska bir tablouun attribute yeklestiriliyorsa k buna yobanlı orchtor diyez.

consistency  $\Rightarrow$  tutarlılığı saglamaktır.

bu keyler tutarsızlığı engelliyor update edince değerler ikisi tablolarda değişiyor ve tutarlılığı korunuyor.

composite key = Birden fazla attribute olusun primary orchtor değerler.

(Kiralikeyler)

Property For Rent

Staff (calisan)

Client (lokci müsteri)

Viewing  $\rightarrow$  göstermek

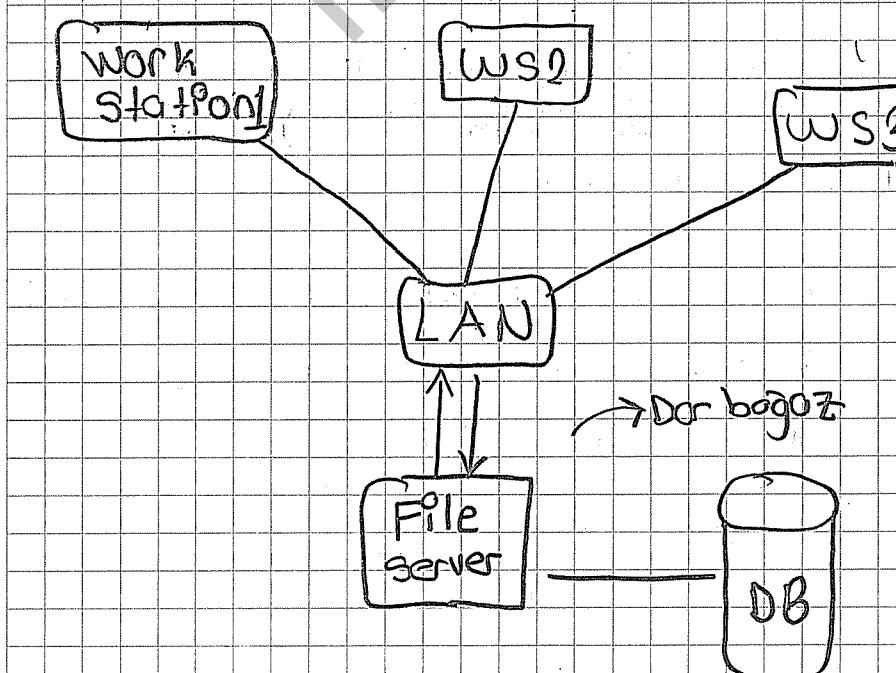


Bir client bir evi belirli staff ile beraber bu tarihte ziyaret etti.

Pno Cno data Primary key olur  
sadece 1 alici 1 evi içinde 1 kez  
ziyaret edecektir. bu şart yok ise herşef  
primary key olması gereklidir.

Composite key dir bu.

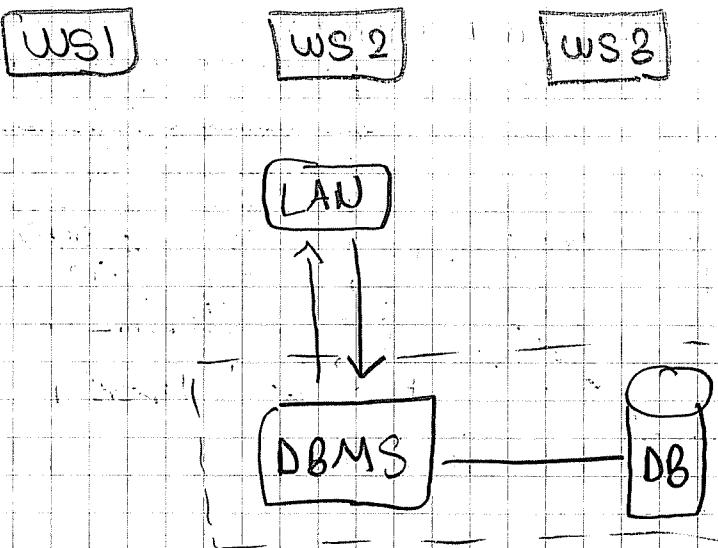
Kullanicı grubu



cografî konum  
olarak farklı yerlerde  
istek oluyor. Onçlik  
dosya sunucusuna  
istek talep ediliyor.  
dosya sunucusu db  
erisip veriyi alıp  
gönderiyor. DBMS  
yokken, bu verdi

de avantajı yok olmasi

Update ederken ortak bir dosyayı  
sıkıntı olur haberisiz olarak birbirinden.



► Bu sunucu mackinesi  
bunlar bu mckinede bulunur.

DBMS bütün veriyi döndürmez diğerinde "Öğrenci dosyası" döndürdü simdi istenilen veri döndürür. Haberleşmeyi minimize ediliyor. Güvenlik problemleri ortadan kaldırıyor.

↑ Two Tier Architecture Server.

↓ Three Thier Architecture

Yeri mimaride

Application

Server

↓  
uygulama yazılımı tutuluyor.

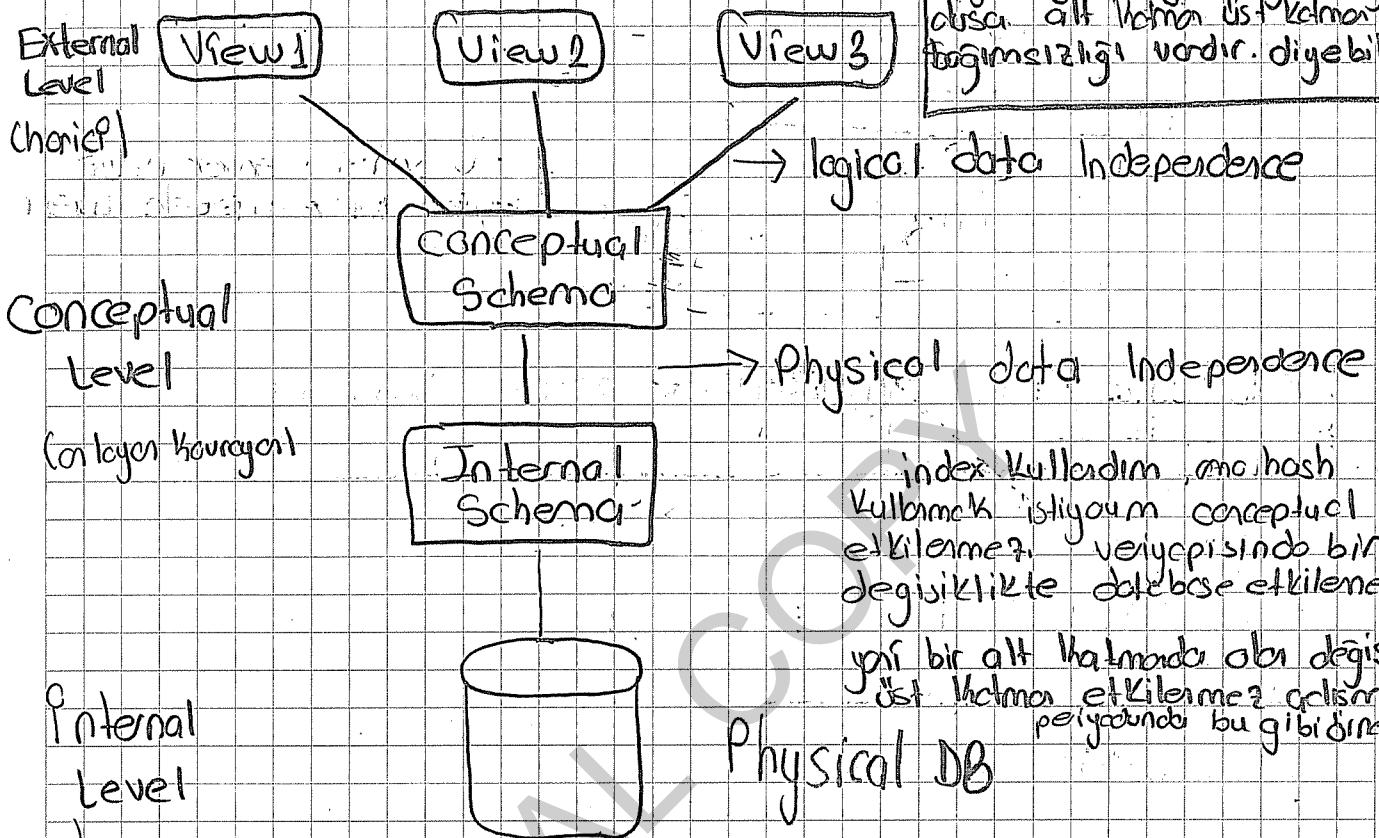
DBMS → sorgulanan sonucu döndürür.

Client baki oralar bilimler var. Application clientlardaki güncelliyor olsalar update ile tüm clientları güncelliyor aynı anda.

## Three level Anti/Sparc Architecture

→ DBMS bunu kullanıyor.

Vari tablo seklini degistirdim tablo gormenin degistirmeyecektir bir yuvarlak tabloda hisseli mayen gorulemeyecek degisiklik olusur. Alt tablon ust tablon tanisizligi vardir. digebiliriz



→ logical data independence

index kullandim, mo hash  
kullanicik istiyorum conceptual  
etkilemez, veriyapiinda bir  
degisiklikte database etkilemez

yani bir alt tabloda obi degisim  
ust tablon etkilemez calisma  
polyglot bu gibi orneklerde

Physical DB

Ssn	frame	name	DOB	salary
-----	-------	------	-----	--------

Veritabanı yönünden bunun göründüğü seviye conceptual seviyedir.

struct staff {

```

int Ssn;
char frame[15];
char name[15];
:;
```

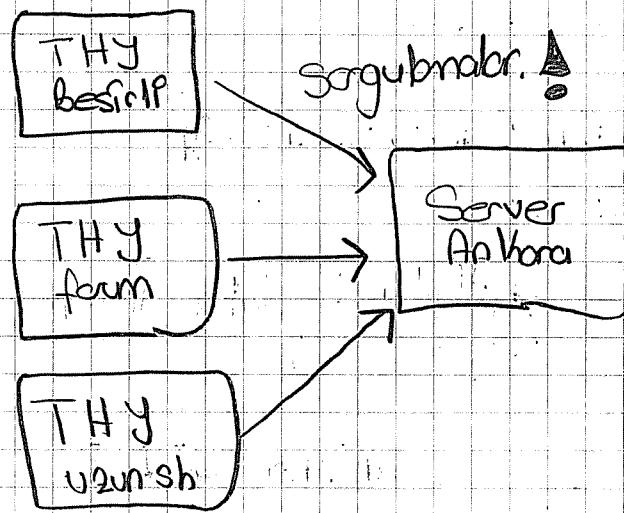
External seviye yazilim  
muhendisine belli tablon  
belitti ozelliklere erismesini  
istiyorum bugoruntuyu "View" de  
veriyorum "DOB", "salary" gelmesini  
istemiyorsak burda gorsterniyiz  
istesek bu seviyede bu var.

bağımsızlık var salary ben sildim view etkilemez  
zaten görmediğim, bunlar bağımsızlığın olduğu söyledir.

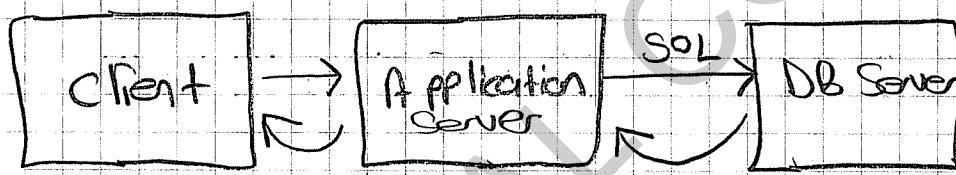
## Structured Query Language

Veritabanı yönetim sisteminde bu dili Kullanarak sorgu yapabiliyoruz.

Cilent



- DBMS server üzerinde
- Clientlar istemciler
- ile veri gönderir
- uygulandı client bil.



- Three Tier Client Server Systems

THY yer bir programı içinde: sadece A.S değişimek yeterlidir.

Hash fonksiyonu: Bir tablo üzerinde eşitlik sorgularını optimize etmek için.

md5 gibi uygulamalarda bir fonksiyon igerisi.

=> logical data independence: (External - conceptual)  
concept değiştiğinde → kullanıcı bundan etkilenemedi

=> physical data independence: (conceptual - internal)  
Hash ti oğaq gevirdik. Altılık bilgi etkilenemedi.

atomic  $\rightarrow$  bülteneger

Deletion  $\rightarrow$  Tabloya Korsilik düşüyor.

bno	Street	City	Postcode	{ attribute, field }
record				
row				
tuple				

$\rightarrow$  ilişkide tuple sayisi = cardinality  
1 satır versa 1 cardinality

$\rightarrow$  number of attribute = ilişkinin derecesi.

binary relation = 2 tane attribute vardır.

bno → attribute if  $\rightarrow$  Primary key olduğunu gösterir.

## Properties of Relations (İşki Özellikleri)

1-) The relation has a distinct name (her bir tablonun ilişkinin) kendine gerek unix bir ismi olmasıdır.

2-) Each cell must contain an atomic value (her bir hücrenin atomic değerler olacakları)  
Her bir ilişkide tek bir telefon numarası olabilir hücrenin içine birden çok numarayı virgül ile ayırarak yazarsak SQL komutunda sorguda tek bir değer döndürmez. Sorgu açısından sora bu atomic olmaz tek bir numara olacaktır.  
telefon numarası için herko bir tablo olmasının istenmesi.  
Verimli bir sorgu olmaz çok numara olursa optimum tassimler hizasına izin vermez.

3-) Her bir "attribute" un ismi farklıdır. Tekrarlanmaz, farklı tablolarda kullanılabilir.

4-) Her bir kayıt unique dir. primary key versi diyebiliriz.  
her bir ilişkinin primary key olmaz zaten olsatır.

5-) attribute sırasının önemini vermidir? sorgular sütunlarından gerekkestiriliyor.

6-) tuple sırasının bir önemini vermidir. not tablosu var büyükten küçüğe ekleset sırda şekilde buyordan sıralanması bir avantaj sağlar bize.

# Tabel Tonimba cause→neden olmak

Branch	bno	Street	city	Postcode
--------	-----	--------	------	----------

Staff	sno	financ	Income	position	sex	dob	salary	bno
-------	-----	--------	--------	----------	-----	-----	--------	-----

PropertyForRent	pno	street	city	postcode	type	rooms	rent	ono	sno	bno
-----------------	-----	--------	------	----------	------	-------	------	-----	-----	-----

Client	cno	fname	lname	tel no	pref type	max rent
--------	-----	-------	-------	--------	-----------	----------

Owner	ono	fname	lname	address	Phone number
-------	-----	-------	-------	---------	--------------

Viewing	cno	pno	date	comment
---------	-----	-----	------	---------

Registration	cno	bno	sno	date joined
--------------	-----	-----	-----	-------------

schibi  
mülk

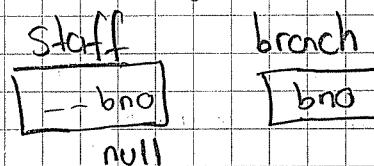
av  
widome  
client

## Integrity (Bütünlük sınırlançları)

Constraints (Sınırlançlar) = tutarlılığı sağlamak için DBMS yapan sınırlançlar.

\* Entity Integrity = Primary key değer asla null olmaz  
DBMS: "A primary key cannot be null"

\* Referential integrity = Eğer bir tabloda F.K varsa  
ve içerdigىن değer null olabilir yada değer versa base relationda olmalıdır



"başlık bir bilgidir"  $\Rightarrow$  "Null farklı başlık değil yok demekdir"

\* General Constraints = Bir tabloda bir attribute da bir sınırlama getiriyosun pulon > 100 üstündede ise veiyi eklemeye kes diyez. bunu user yapıyor.

## Relational Algebra ("İllüksisel Cebir")

$\rightarrow$  Soruyu en etkin şekilde nasıl gerçekleştirir.

$\rightarrow$  Optimal modül Query Optimizer = soruyu "DBMS" süzüyor

$\rightarrow$  gelen SQL metni matematiksel ifadeye dönüştürüp optimun yapmayı hazırlıyor.

"Özel modül Query Optimizer = gelen ifadeyi optimun elmek"

\* Unary Operations : Tek bir ifadeyle gerçekleştirilen

toplama, çarpma = binary

1. 1 -> Selection Operation

Maasi 10.000 \$ fazla olan staffları listele

(Staff)

$\Rightarrow 6 \text{ salary} > 10000$

Selection

6 (R)

predicate !  
(Kuralı) İllüksel

Salary				
1000	2000	3000	4000	5000
1000	2000	3000	4000	5000
1000	2000	3000	4000	5000
1000	2000	3000	4000	5000

segim işlemini satırda gerçekleştir.

"Select \* from Staff where salary > 1000"

1.2) Projection Operation  $\rightarrow$  Unary tek parametreli işlendir  
yine tablo

$\Pi_{a_1, a_2, a_3 - a_r}$  (R)  
2 olsat attribute

$\Pi_{SNO, fname, lname}$  (Staff)

"Select SNO, fname, lname from Staff"

1/1	1/4	1/1		
1/1	1/4	1/1		
1/1	1/4	1/1		

$\rightarrow$  Sütun Seçiyoruz

## 2-binary Operations

### 2.1-) SET OPERATIONS

\*Union (birleşme)

(R ∪ S)  $\rightarrow$  2 tane ilişki

R kayıtları hepsiyle S kayıtları alttaki sıralı aynı tupleler  
elimine edilirler.

İçerisinde Kiralanan ev olan şehiller  $\rightarrow$  içерisinde sube olan cityleri listele

$\prod_{city} (.PFR) \cup \prod_{city} (\text{Branch})$

union

$\rightarrow$  ilişkide attribute aynı sayıda domain(string,int) aynı  
dimalı

$\rightarrow$  tekrarları消除 eder.

n(INTERSECT)

$\rightarrow$  ilişkide aynı şehri gelen şehilleri listele

/ (EXCEPT)

$\rightarrow$  içерisinde Kiralanan ev var ancak Sube yok  
o şeherde

$\rightarrow$  select olun ilişkisinde olmayan  
değerin gösterisi.

## Cartesian Product (R x S)

→ R ıñisinde her bir tuple S ıñisinde her bir tuple ile  
birlesir.

$$R \times S \\ (4) \quad (2) \\ 4 \times 2 = 8 \text{ kayit olur.}$$

\* list the newest comments of all clients who have viewed a PER

Tüm clientler isimleri biriktirmis oldugu yorumlari liste  
hangi evi gordu

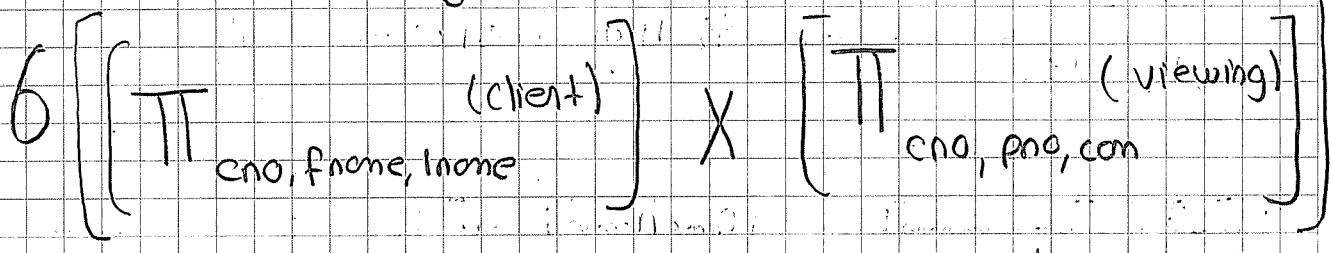
Viewing (cno, pno, date, comment)

Client (cno, fname, lname, phonenum, preftype, maxload)

Client	II	cno, fname, lname	*	Viewing	II	cno., pno, comment	(Viewing)
			*				
		c01   G   U		c01   p1   berbat			
		c02   M   U		c01   p2   Super			
				c02   p1   eh			

	client.cno	client.fname	client.lname	viewing.cno	viewing.pno	v.comment
→	c01	G	U	c01	p1	berbat
*	c01	G	U	c01	p2	Super
→	c01	G	U	c02	p1	eh
→	c02	M	U	c01	p1	berbat
→	c02	M	U	c01	p2	Super
*	c02	M	U	c02	p1	eh

- tek basına karakterler doğrudır, onlara değildir.
- $\text{C}_1$  gibi gözüken yorum  $\text{C}_2$  birleşmiş tuple var bunlar onlarsız 3 kavşat göstermesi gerekli olursa sonuçlar var.
- Bunu filtrelemem gereklidir.



$\text{Client.cno} = \text{Viewing.cno}$

"Select \* from Viewing, PFL } Where Client.cno=Viewing.cno"

Karakterler gorpimdir.

Filtreleme yesidir.

Join Operation (Tablo Birlesimi) Selection  $\times$  Karakterler

→ bir tabloda beliri veride birden çok özellik gösterileceksen bunu kullanırsınız.

1-) Theta Join ( $\Theta$ )

$$R \bowtie S = \bigcup_F (R \times S)$$

$\downarrow$   $\rightarrow$  Kural  
İşki  $\downarrow$  İşki

$\sqsubseteq$   $\rightarrow$  equijoin  
 $\neq$   $\rightarrow$  esitlikle ise

"belirli Kuralı kullanarak  
iki tane ilişkili isteniyor."

" $\leftrightarrow$  oddırırtır.

bu Kuralı kullanarak birleştirilir.

## 2 -> Natural Join

$R \bowtie S$

- Ortak olan bütün özellikleri "łączenie" = "kontrolü yapar"
- hicabisey yazılmaz ise direkt = "ortak olur."
- equijoin için her ortak özelligi kontrol edilir. equijoin de ise bu belirtilir.

R		
fname	lname	commrce
o	o	
o	o	

S		
fname	lname	rest
o	o	
o	o	

equijoin belirtiriz

$R \bowtie S$

$R.fname = S.fname$

yazmazsa hepsi kontrol eden  
bu da natural join olur.

## 3 -> Outer Join

$\rightarrow$  Right  $R \bowtie S$

$\rightarrow$  Left  $R \bowtie S$

$\rightarrow$  full  $R \bowtie S$

\* Produce a status report on properties

evlerin hakkında özellikleri ve yorumları göster.

→ outer özellikleri bu buncu -bu client gitti su yorumu bıraktı

$(\prod PFD)$   
pno, street, city

$\bowtie$  Viewing

$PFD, pno = Viewing.pno$

$\bowtie$   $\rightarrow$  SQL taroficki bir keydin  
sog tarofic koydu yaktı.  
0 ye null ip doldurur.

PFR

Viewing

PO1	US	TRB
PO2	AMB	Samsun

PO1	CO1	güzel

N

PO1	US	TRB	PO1	CO1	güzel
-----	----	-----	-----	-----	-------

Y

PO2	AMB	Samsun	Null	Null	Null
-----	-----	--------	------	------	------

Aşağıda sağ yolk ise silinen ve gösterilen önemli olan bölge

B) Her ikiinde kontrol eder sağda sadecemayın, sağda sağda mayın

İmportan olan bölge join işlemin yönünü gösteriyor. full her ikiinde göster diyeği konusunu ölsün olmasın.

Division Operation

→ Bölme işlemi

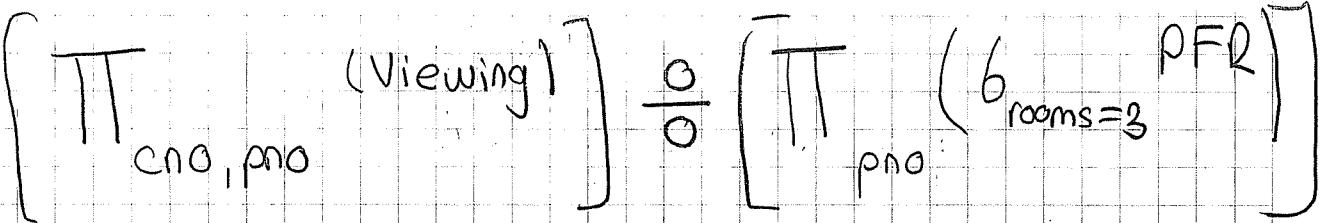
R  $\frac{O}{S}$  S

\* Identify all clients who have viewed all properties with brows

3 adlı evlerin ziynet eden clientleri listele

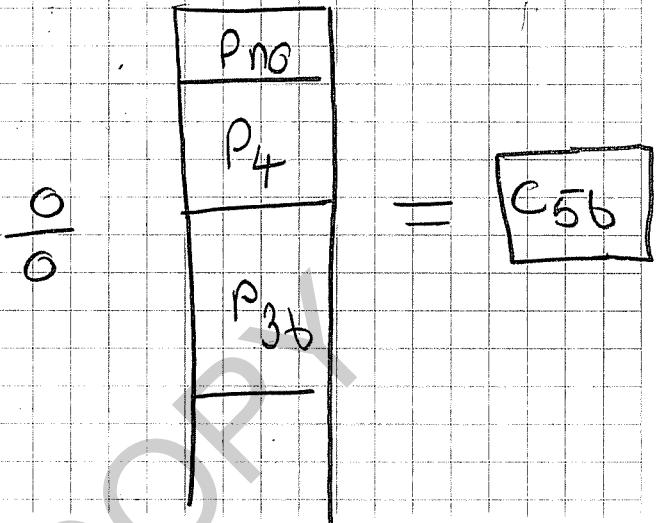
clientlerin içinde bu evleri ziynet eden kimdir.

Zadoli tüm evler



\* \*

CNO	PNO
C56	P14
C76	P4
C56	P4
C62	P14
C56	P36



Aggregations and Grouping Operations

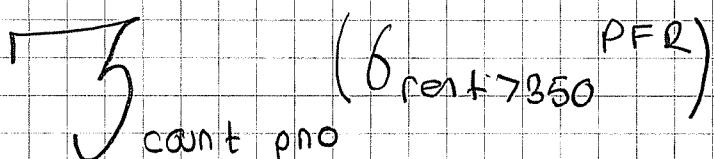
(İliskisel cebir)

↳ Block ieri birinden islem yapma.

- COUNT
- SUM
- AVG
- MIN
- MAX

↳ (R) → iliski

(ALL) → yuzelk oldigum fonksiyonlar



Kiral 350 den büyük olan evlerin listesi bu verilerin pnosunu ergistir! buluyuz pno ları say kac tane pno var.

P01	250
P02	400
P03	500

= 3 doldurdu bize

Sorgular matemtiksel cebir dñustürüyor sorgular.

Σ

(Staff)

min (salary), max (salary), avg (salary)

SNO	salary
S01	300
S02	100
S03	200

	min	max	avg
	100	300	200

\* Tek kayıtta bir ilişkî döndür

Aggregations'lar tek bir değer döner tek bir satır ilişkî döndür

P → simleme operatör  
P : (Mymin, Mymax, Myavg) → attribute isimleri  
Q → geridönen ilişkisi isimleri

P<sub>R</sub> (a<sub>1</sub>, a<sub>2</sub>, a<sub>3</sub>) → R ilişkisi ve a attribute isimleri

Mycount
3

Select min(salary) From Staff

Grouping Operations

Σ<sub>AL</sub> (R)

↳ grublarda kılınan attribute özellik

→ Grouping belki attribute ları kullanarak belirli gruplara ayırmaya yarıyor.

→ Find the number of Staff working each branch and and the sum of their salaries.

Ogrenci tablosunu sehir bilgisine göre grupta

Trab201  
İşyerler  
Zararlılık gibi gruplar olusuyor bu grubları ayrı ayrı aggregations uyguluyoruz.

→ Her bir branch ta çalışan staffların sayısını ve bu staffların toplam ne kadar maaş oluyor bunları listeleyelim.

branchno, staffcount, sumSalary



(Staff)

Count(sno), sum(salary)

bno

↳ göre grupta ayrıca hantane sno var ve maaşların toplarını

sno	bno	salary	group var.
S01	100	801	1
S02	200	801	
S03	150	801	
S04	300	802	2
S05	400	802	
B01	3	450	
B02	2	700	

## Fleksisel Çeşit BİLF

3 (Staff)

P<sub>R</sub> (bno, mycount, mysum)  
count (sno), sum (salary)

bno	my count	mysum
B01	3	450
B02	2	700

SQL (Sorgulama Dil'i) (Structur Query Language)

Baska bir dil üzrenen cogiriln dil

\* Select — From

1- List full details of STAFF

\* Select \* from Staff  
→ bütün sutunları listele

\* Select fname, lname from Staff

\* Select pno From Viewing

burda pno'yu tekrarlayabilir  
ayni vorsa...

\* Select Distinct pno From Viewing

→ ay尼 isimlerini minimize eder.

calculated fields → Hesaplama fieldler

\*) Select fname, lname, salary / 12 from Staff

→ salary sütunundaki bilgileri her bir row'da 12'le bölerek gösterir.

fname	lname	(mont salary) COL 3	attribute ismi olmuyor.
S	U	4500	

⇒ Row Selection (Where)

→ belirli bir koşul doğrultusunda  
kayıtları geri döndürür.

\*) Select \* from Staff Where salary > 10000

→ fazla koşul varsa

\*) Select \* from branch Where city = 'Zonguldak'

Or city = 'Trabzon'

→ Trabzon veya Zonguldak şehirlerini listeleyin

$\Rightarrow$  Range Search : not between arasında olmayan

Ⓐ Select \* from Staff Where salary Between  
20000 and 30000

$\Rightarrow$  20000 ile 30000 arasındaki verileri döndür.

Ⓑ  $\Rightarrow$  Salary > 20000 and Salary < 30000  
between yerine bu olur.

$\Rightarrow$  Set membership Search condition  
"bir kümeye bağlılığı var mı"

Ⓐ Select \* from Staff Where position In ('Manager', 'Supervisor')

in('1','2') ikiinden biri ise disarı oktarır.

not in  $\rightarrow$  dahil olmayan

"Where position = 'manager' or position = 'supervisor'"

$\Rightarrow$  Pattern match Search condition

(like, Not like)

0/0

- single character

pattern

orsatırımsız

yapıyoruz.

## attribute

address like 'H%' → bas harfi 'H' ile başlayan  
 || || 'H--' → 4 harften ve bas harfi it olmamalı  
 || || '0%e' → e ile biten  
 || || '0%Glasgow%' içinde geçen.

Like '15%' 15 ile başlayan diye onlar.

Like '15#0%' Escape '#' diye tanımlar.

'15##0%' → 0' onluk 1 tane oluyor,

150% → bunu oncağı degerde.

→ glasgow yer alan bütün müşteriler adresi

⊗ Select \* from client where address like

'0% glasgow%'

⇒ Null Search

(IS NULL, IS NOT NULL)

Select cno, date from Viewing where  
 pno = 'P04' and comment is null

|| comment null olan ve pno P04 olanların  
 cno ve date yazdır.

⇒ Order by sıralı bilgileri echeriz

\* Select \* from Staff order by salary desc

"Ücret azları sıradı verilenin bütün özellik" (azalan) terini göster.

\* Select pno, type, rooms, rent from PFR  
order by type, rent Desc

P01	Flat	4GG
P02	Flat	300
P03	House	1100
P04	House	1000

\* → önce type göre  
sonra rent sıralı  
diyez.

\* → default order by sıralanmış  
oynat → doğrudır.

⇒ Using The SQL Aggregate Functions

\* Select sno, Count(salary) from Staff  
"Yanlış Kullanım"

\* Select count(\*) as mycount from PFR  
where rent > 350

\* Select count(Distinct pno) as mycount  
from Viewing where date between '11-May-14'  
and '31-May -14'

→ tarihler arasındakilerin sayılarını yazdır.

Ⓐ Select count(sno) as mycount, sum(salary) as mysum from Staff where position = 'manager'

1 tek bir satır döner

⇒ Group by

Select bno, count(sno), sum(salary) from Staff group by bno

Aynı branch no ya ait sno sayısını ve tıpkı maaşlarını  
sayısını listeley.

b01	4	1121
b02	5	1342

Ⓐ Select bno, count(sno), sum(salary) from Staff group by bno having count(sno) > 1

↳ grubu ayırmış olduğumuz  
alt koşullar üzerinde işlem  
yapıyor.

bno | sno | salary |

B01	S01	100
B01	S02	200
B02	S03	400
B03	S05	100

B01	2	400
B02	1	600
B03	2	200

⇒ Subqueries (Alt sorular)

→ '163 codde' kıl branch'ta çalışan staffların bilgilerini  
bina listeley

⇒ "Bak dersde bno = yoptik"

Select \* from Staff  
Where bno in

(Select bno from branch Where  
street = '163 codde st')

⇒ Using a Subquery with an aggregated function

→ Staff maaşı ortalaması maksın yükseliş olsun aynı zamanda ortalamadan ne kadar yükseliş olduğunu göster.

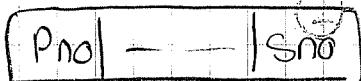
Select sno, fname, lname, salary - (Select avg(salary) from Staff)  
from Staff Where salary > (Select avg(salary) from Staff)

→ 163 num street subede çalışan stafflar tarafindan yönetilen eulerin listesi?:

(Select sno from Staff ) (Select bno from branch Where street =  
... where bno in ) '163 num street'

Select \* from PFL where sno in (

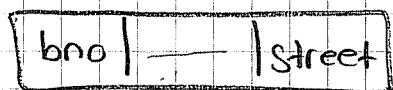
PFL



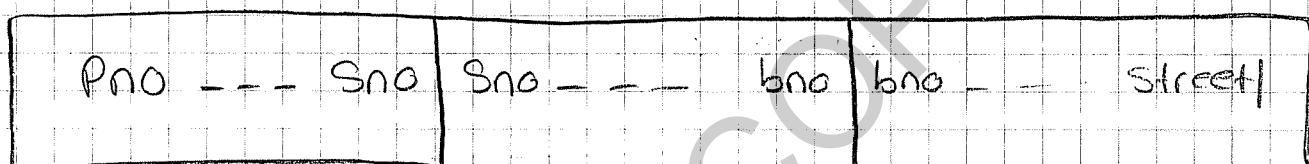
Staff



Branch



Select \* from PFL, Staff, branch where  
PFL.Sno = Staff.Sno and Staff.bno = branch.bno and  
branch.street = '1163 man st'



Any / All

B subesinde calisan staffin en azindan bir tanesinin  
maasinden daha yüksek olsun.

Select \* From Staff  
Where salary >

ANY (Select salary from Staff  
Where bno = 'B003')

All → olusdu herkesin maasinden yüksek olacak (max büyük esit olur)

Any → en azindan birinden daha buyuk olacak (min daha büyük)

\* Select min(salary) from staff

## Simple Join (Equijoin)

⇒ Eve gidip yorum bıraktı client'ın eki bilgileri o gidip girdükler evlerde yorumları listeleyecek.

Client \* Viewing



↳ eno, pno, date, command

CNO, fname, lname

\*  
SELECT \* From Client, Viewing (Kortezyen by)  
Where Client.CNO = Viewing.CNO

\*  
From Client C Join Viewing V On  
C.CNO = V.CNO

⇒ Staff ismini ve yarattığı olduğu evin idlerini listeleyin.

bno, sno, pno

Staff → bno, sno  
PFL → pno, sno

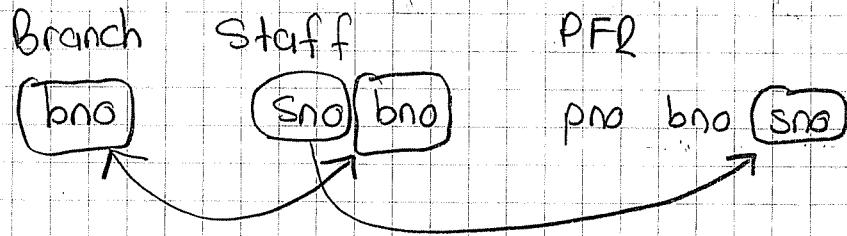
SELECT S.bno, S.sno, fname, lname, pno  
FROM Staff S, PFL P

B2	S01	g	M	P01
B3	S01	g	M	P01
B3	S03	M	M	P04
B3	S03	M	U	P05
B4	S02	A	M	P11

Where S.sno = P.sno

order by bno, sno, pno

→ Her bir şube için şube yer oldığı Şehir, çalışan stafların numarası, şube numarası, staf ilk ismi son ismi, yönettiği eylem numarası...



$\text{SELECT } b.bno, b.city, s.sno, s.fname, s.lname \text{ p.pno}$   
 $\text{FROM staff s, branch b, PFL p}$   
 $\text{where } (b.bno = s.bno) \text{ and}$   
 $(b.bno = p.bno)$

Alternatif \* FROM Staff S Natural Join Branch b Natural Join PFL p

Left outer join:

Aynı şekilde yer almış subelerin ve eylem numaralarını listeleyin

$\text{SELECT } b.* , p.* \text{ from } (\text{Branch}) b$   
 $\text{LEFT JOIN } (\text{PFL}) p \text{ ON}$   
 $b.city = p.city$

Branch	
bno	city

PFL	
pno	city

Branch 1	
B3	Glasgow
B4	Bristol
B2	London

PFL 1	
P14	Aberdeen
P94	London
P4	Glasgow

b.bno	b.city	p.pno	p.city
B3	Glasgow	P4	Glasgow
B4	Bristol	null	null
B2	London	P94	London

left join

null	null	P14	Aberdeen
B2	London	P94	London
B3	glasgow	P4	Glasgow

right join

EXISTS / NOT EXISTS

\* London'daki subelek koliso staff larin listesi.

SELECT \* FROM Staff S

Where EXISTS (

Alternatif:

Select \* from Staff where

bno in (Select bno from Branch

where city = 'London')

SELECT \* FROM Branch b

Where S.bno = b.bno) AND

city = 'London' )

\* city sadece branch dava buyulunden berilirmeye gerek yoktur.

\* EXISTS true / false dondurulur. Tek bir kayıt donese TRUE donmezse FALSE döner. TRUE donuyse o anki kayıt listelenir.

sno	bno
S1	B7
S4	B3

FALSE  
TRUE

NOT EXISTS → Tom testi oluyor.

B3 London

$\Rightarrow$  2. işaret edilmiş ve yorum birakılmış evlerin listesinin

```
SELECT * FROM PFR p  
WHERE EXISTS (  
    SELECT * FROM Viewing v  
    WHERE (p.pno=v.pno)  
        AND command IS NULL)
```

Union ifadeleri  $\Rightarrow$

İçerisinde 1. sube olan veya Kiralanan evlerin bütün  
şehirlerin listesinin.

```
(SELECT City FROM Branch)  
Union  
(SELECT City FROM PFR)
```

$\rightarrow$  her sube içinde Kiralanan sube (Intersect)

$\rightarrow$  sube olup ev olmayan şehir (Except)

$\rightarrow$  Birleşim (Union)

Sube olan Kiralanan ev olmayan.

```
SELECT city FROM Branch b  
WHERE NOT EXISTS ( SELECT * FROM  
PFR p WHERE  
p.city = b.city )
```

Code =>

```
SELECT city FROM Branch b  
WHERE city NOT IN (SELECT  
city FROM PFL)
```

## INSERT

→ INSERT INTO Staff VALUES ('S07', 'Alan', 'brown',  
'Assistant', 'M', DATE '1987-05-28'  
8.300, 'B03')

→ INSERT INTO Staff (sno, fname, lname)  
VALUES ('S04', 'Anne', 'Jones')

→ Yeni tablo oluşturdu (sno, fname, lname, propcount)

→ StaffPropCount tablosu ←  
Bölük verinin tabloya aktarımı:

p. pno

Insert into StaffPropCount (Select sno, fname, lname, COUNT(\*)

"Count kisinin kaçıncı  
ev yonetigini veriyor" //  
FROM Staff S, PFL P  
Where (S.sno = P.sno) AND  
GROUP BY sno, fname, lname  
Union

p. pno

→ Burda sno, fname, lname olde grup ledik  
ve grupta kaçıncı ev varsa etmede birini  
veriyor. Asagidaki billesim Staffin varlığı  
ev yonetise count() atamış. yani bu bilgi  
bölük evi etmeyen staff'yi veriyor bilmesi  
yaptığı için yukarıda sadece evi olanlar  
gelicek. bu sekilde bütünlüğü sağlanır.

Select sno, fname, lname, 0  
FROM Staff S

Where Not EXISTS (

Select \* From PFL P

Where S.sno = P.sno

Update:

Staff maasların, arttirıcom

\* Update Staff SET salary = salary \* 1.03

"bütün çalışanların update edildi salary'lerin"

\* Update Staff SET salary = salary \* 1.05

where position = 'manager'

\* Update Staff SET position = 'manager', salary = 18000

Where sno = 'S03'

Delete:

\* Delete from Viewing where pno = 'P04'

\* Delete From Viewing // bütün kayıtlar silinir.

CREATE

bno : CHAR(4) → 4 desen 2 kullanıcı bile 4 ayrırlır,

bno : VarChar(4) → değişen karakterler 2 karakter kayıtları 2 harcanır 2 kullanılır

bitString : BIT(4) → 4 bitlik değer ayrıldı.

rooms : smallInt

salary : Decimal(7,2) naktadan sonra 2 tane önce 7 tane  
99999.99 olur.

FLOAT  
REAL

Viewdate : Date

Operatorler: → || concat = iki tane stringi birlestirmek.  
→ Current\_user = tabloyu olusturan user ismini string donduru  
→ LOWER = A → a ALP → ali, ALL → alp  
Lower (select fname, lname  
where sno = 'S41')  
→ UPPER = a → A uppär  
Hello → TRIM (BOTH '\*' FROM '\*\*Hello\*\*')  
→ position ('ee', IN, 'Beech') → 2 dördüncü  
→ Substring ('Beech' From 1 to 3) → Bee  
→ Extract (YEAR From dateJoined)  
→ Yıllı bilgisini çek

not null field attribute değeri null olmasın hicbir zaman  
position varchar(50) not null

SEX char not null Check (SEX IN ('M', 'F'))  
" girilen değer her zaman M ya da F olsun"

CREATE Domain SexType As Char Default 'F'  
Check (Value in ('M', 'F')) Deger girilmez ise 'F' koy

SEX SexType not null Kullanılabilir. (Denemisit domain icinden)  
Domain

Create Domain BranchNumber As Char(4)

Check (value IN (Select bno From Branch))

Branch teklisun olmayan bir değer  
girilemeyecektir

Tüm bunları öğrenmek Domain silmek

DROP DOMAIN Domain Name [RESTRICT (CASCADE)]

On tabloda obr bir tabb  
varsa silme islemi engellenir.

Drop Domain Branch Number RESTRICT

aktif olmak tabloda kuyubulmasa  
silme islemi engellenir

→ CASCADE → Domain kullanmasın ya da kullanmasın silinir.  
tabbya ne olur. BranchNumber tabloda ne var (null) ise  
char(4) kullanılır. artik.

PRIMARY KEY (pno) ← Primary Key Tanimlandi

PRIMARY KEY (pno,chn)

staff tablosunda bunu yaziyorum.

FOREIGN KEY (bno) ← Foreign Key Tanimlama

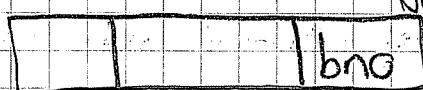
REFERENCES Branch

ON DELETE CASCADE → Parent silinince child da silinir.

SET NULL → Parent silindi child olsun null olur.

NO ACTION → silme güncellene yormaz.

Staff



Child table

SET DEFAULT

→ nullen default  
deger nosa.

bunun keyin  
sillinenine one  
verilmesisse

default dardur.

Branch



Parent table

→ PFR'de bunu yöneten staff kactane eyle ilgileniyor bunu kontrol edicez. 99-100-101> olmasın diyez bunu 100'ye kadar 1'in ve 2'yi kontrol edicez programla. Programlama olmada da kod ile mümkün

PFR → sno

sno

S1 → kactane evi yönetiyor > 100 gecese girisine engel ol diyeceğiz.

SELECT count(\*) FROM PFR WHERE sno = 'S1'

CREATE ASSERTION StaffHandlingTooMuch

CHECK (NOT EXISTS (SELECT sno  
FROM PFR GROUP BY sno  
Having (count(sno) > 100)))

Fsim bu  
visitbma ismi

Tabb Pönümleme =>

CREATE Domain OwnerNumber AS VARCHAR(5)  
CHECK (VALUE IN (SELECT ono FROM PrivateOwner))

CREATE Domain StaffNumber AS VARCHAR(5)  
CHECK (VALUE IN (SELECT sno FROM staff))

CREATE Domain BranchNumber AS CHAR(4)  
CHECK (VALUE IN (SELECT bno FROM Branch))

CREATE Domain PropertyNumber AS VARCHAR(5)

"	"	Street	"	"	(25)
"	"	City	"	"	(15)
"	"	Postcode	"	"	(8)

CHECK (Value IN ('B', 'C', 'D', 'E', 'F', 'M'))

CREATE Domain PropertyRooms AS SMALLINT

CHECK (Value BETWEEN 1 AND 15)

CREATE Domain PropertyRENT AS Decimal(6,2)  
CHECK VALUE BETWEEN 0 AND 9999.99

CREATE TABLE PropertyForRent(

Pno	PropertyNumber	NOT NULL
Street	Street	11 11 9
city	City	11 11 9
postcode	Postcode	11 11 9
type	PropertyType	11 11 Default 'F' 9
rooms	PropertyRooms	11 11 11 4,
rent	PropertyRent	11 11 11 600,
ono	OwnerNumber	11 11

SNO StaffNumber CONSTRAINT StaffHandlingTooMuch  
CHECK (NOT EXISTS(---) NOT NULL,

bno BranchNumber NOT NULL,

PRIMARY KEY (pno),

FOREIGN KEY (sno) References Staff ON DELETE SET NULL

ON UPDATE CASCADE

11 11 (ono) References PrivateOwner ON DELETE NO ACTION  
ON UPDATE CASCADE

11 11 (bno) References Branch ON DELETE NO ACTION  
ON UPDATE CASCADE

→ Tablo içinde Değiştirme.

Alter Table Staff

Alter position DROP DEFAULT

→ position'un default değerini siliyoruz.

Alter Table Staff

Alter sex SET Default 'F'

→ önceden 'M' idi şimdi 'F' oluyor default değeri..

Drop TABLE PFL [RESTRICT | CASCADE]

→ Drop edilebilir direkt.

↳ Engellenir başka tablo  
kullanılıyorsa

- Görüntü oluşturuz Kullanıcının gideceği View

→ Bir tablo gerçekle dönen son tabloya  
AS → sonra view içini oluşturuyoruz.

CREATE VIEW Manager3Staff

AS SELECT \* From Staff

Where bno = 'B03' → B03 de collision staff listeliyor  
Kendi collisionini gösteriyoruz.

CREATE VIEW StaffPropnt (bno, sno, cnt)

AS [SELECT s.bno, s.sno, count(\*)

FROM Staff S, PFL p

Where s.sno = p.sno

GROUP By s.bno, s.sno

SELECT \* FROM StaffPropnt

View → hit için yapılmazırome gerek yoktur.  
ancı null iniçliye gelsemeli.  
View dist üzeinde değil dinamiktir.

→ View drop edebiliriz.

Dinamik oluşturular bellekte  
Bu view cascade gibi sayesinde verilebiliriz.

DROP VIEW manager3Staff

```
SELECT sno, count * FROM StaffPropCount  
Where bno = 'B03'  
Order by sno
```

Veritabanı Y.S View üzerinde sorulur  
View resolution ile güncellestiriliyor.

→ View segulook attribute ların orjinallerine yerine getiriyor.

1-) SELECT S.sno AS sno, COUNT(\*) AS cnt

2-) View Uzeltmek için hangi tablodan Kullandık  
FROM Staff S, PFL p

3-) WHERE S.sno = p.sno AND bno = 'B03'

4-) View oluşturur cumlede group by kullanırsınız

GROUP BY S.bn, S.sno

5-) ORDER BY sno

View İYİDİR BİLE

1-) Select ifadesinde DISTINCT kullanırsan update edilmez

2-) Select listesinde her bir elemen bir satır olmalıdır.  
Aggregate olunuz 0 olmaz

Update ettiğiniz i takip edebilmek için gereligidir.

3-) From cümlesi tek bir table içersin

4-) Where cümlesi herhangi bir select ifadesi içermemeli

5-) Group by, Olmucak Having de olmaz.

CREATE VIEW Manager3Staff

AS SELECT \* FROM Staff  
Where bno = 'B03'

WITH CHECK OPTION → View update edilmemesi  
için kural sağlanması gereklidir.

Update Manager3Staff Set bno = 'B05'

Where sno = 'S37'

→ Yanlış.

2. InnoDB subede gollerde update ederken with check 0. dedigimiz  
bu kural uyması gereklidir.

→ Viewlerin amacı nedir.

→ Security amaçlı Kullanıcıya belirli bir gölgenliği veriyorum.  
→ Yetkilendirme yapığının bu iş yapısını sunu yapmasını好吧...

→ Gök fazla sorgu varsa view de performans düşer.

→ Gerçek tabloda hard disk erişmedi. Bu da iyi değil.

CREATE view StaffPPrfent(sno)

AS SELECT DISTINCT sno  
FROM PFR

Where bno = 'B03' And rent > 600

b=B03 ve rent>600 olsun staffların numaraları

Kılları 600 den küçük olan yani bir  
ev ekliyorum.

View Materialization:

SNO
S37
S14

Neyi bir kez diskten okup etm kaydını ve  
kayının üzerinde sorgular yapalmayı diskten değişimde güncellenebilir view.

1- 'P24' — 550, 'C040', 'SIG', 'B08'

bu gösterilmesi gereklir viewde

2- 'P54' — 450, 'C089', 'S37', 'B03'

↳ bu var olan güncellemeye  
gerektir viewe

3- ' ' — 300, —

↳ bu eklenmez 400<unki

Bu Stone kaydını view update edilmesinde gerektir  
diskten yapılmıyor.

delete → SIG satır silindi bu silmede view diskleri arasında  
bir değişim olacak mı.

Diskteki değişim view sürekli update delete yapılmıyor bu iyi  
değil. tam tersinde ise bu

bu metter. Verilerin bolleğe alıyor DBM ise bolleğin içindeki  
bunun genelligini kontrol ediliyor.

Tablo diskde genelliyor ise bu view güncellenmesi gereklidir.

resolution ise diskten verileri getirip regülüyoruz. bu yarastır.

Kullanıcıya yetki verme işbirlikçi → SELECT, DELETE, INSERT, UPDATE

GRANT [Privilege list] ALL privilege

↳ 16 türli yetki

ON ObjectName - hangi tabloda

TO [AuthorizationIDList | Public] skime

[With Grant Option] → Bir更大的 beskoğanlık  
verilebilir bu herki içinini

GRANT SELECT, UPDATE (salary)

ON Staff

TO Director

= Director Staff tablosunda select ve salary update edebilir.

GRANT ALLPRIVILEGES

ON Staff

TO Manager WITH Grant Option

=> bütün sorguları staff üzerinde Manager yapabılır ve bu türki manager oluşturabilir.

GRANT SELECT

ON Branch

TO Public

=> Branch Tablosu üzerinde herkes select yapabılır.

Entity Relationship Diagram

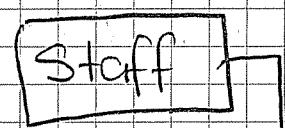
ERD

Vorlikler arsında ilişkilerin ilişkilendirerek gösterildiği bu diagrame gösteriyorum.

Entity → var olan nesneyi temsil etmek için kullanıldığımiza vorluktur.

→ Staff bir entity'di Branch entity'di

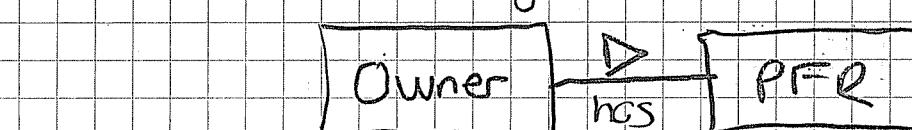
ERD vorlikler dikdörtgeninde gösteriyoruz



△ has (şahip olma ilişkisi)

"Branch şahiptir Staff'a"

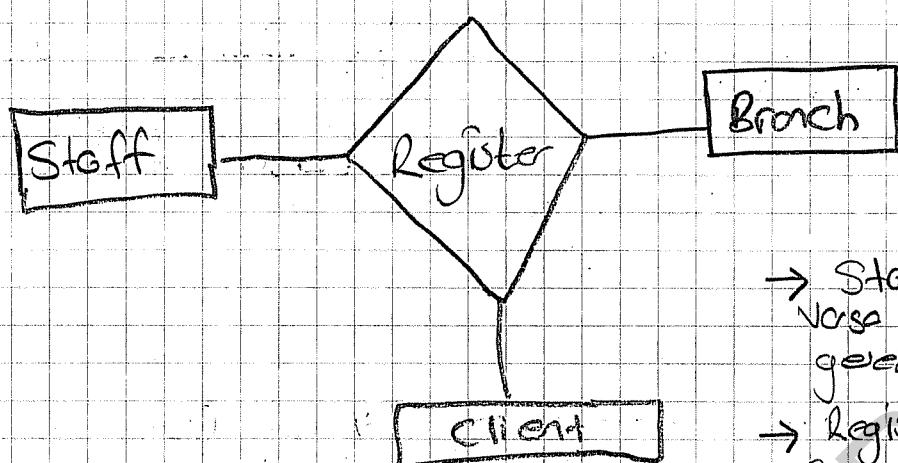
Binary relationship iki entity arasında ilişkidir.



Eş şahipleri evlere eşleştir.

## Ternary relationship =

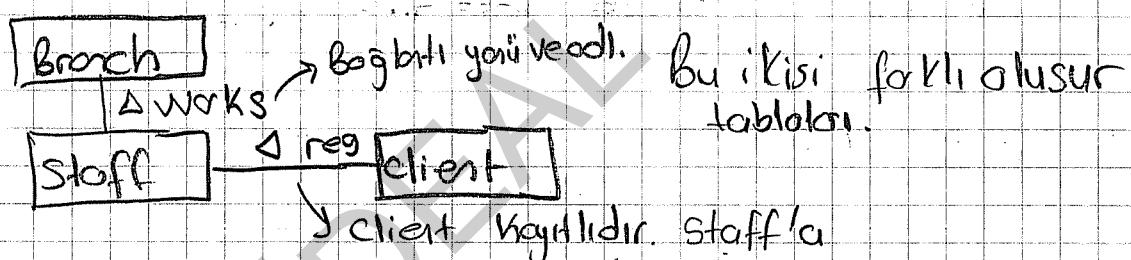
Üçüncü ilişkideki ilişkileri nasıl göstericez



→ Staff ile Branch ilişkisi  
ve ya böyle ayrı göstermek  
gerekli değildir.

→ Register üzerinden çok  
sayı yapayosak bu gerekli  
olabilir.

→ Bir müşteri belili bir subede çalışan belili bir staffa ait  
omurtayle o staff ilişkisidir.



client

cnr | ename - lsnr

sno | sname - lno

branch

bno

→ elinizdeki problemede staff  
branch ilişkisi ve ya bu ayrı  
iliski oluncu olt toplu toplu matirili  
mer eniselli yason yapmaya  
gerek yoktur.

Staff sno | sname | --

sno | bno | lno

Register

Branch bno | bname | --

client cnr | ename | --

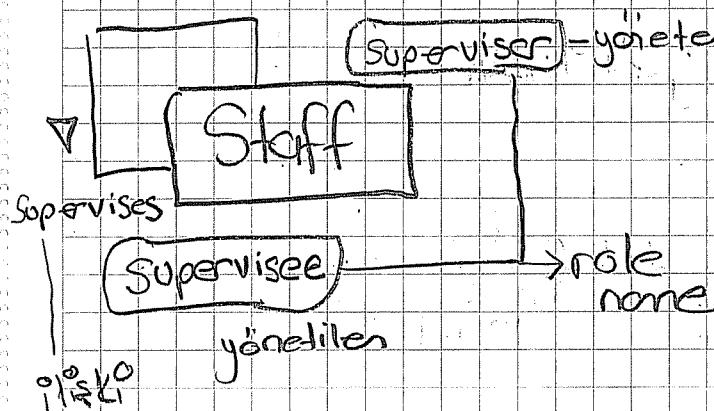
Bir branch helyisine kayit olan  
musteriyi içinde direkt gorur ikinci  
dolaylı olarak görür.

"Burda olt bir alımı açıcaz?"

02.11.2016

Supervise = Yönetmek

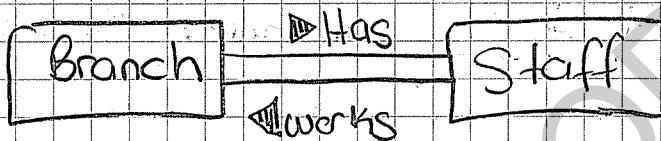
## Recursive Relationshipler (İç içe gelen ilişkiler)



→ Staff, Kendi içinde birini supervis eder.

Öğrenci tablosunda bir kişi yine öğrenci tablosunda bir kişiyi yönetir burda bir ilişki var.

→ Bir öğrenci kimler tarafından yönetiliyor olsun recursive ilişkiye ihtiyaç var.



→ İkili ilişkisi ona aynı sayf tensileder.

İki ayrı şey için göstermek gerekmektedir.

Entity'de var olacak özelliklerin belirlenmesi --

Simple Attribute

→ Tek bir değerden oluşuraktır.

→ Staff position tek bir bilgi vardır. (Bir staff bir pozisyonu var)

→ Cinsiyet tek bir alt obası yoktur.

Composite Attribute

→ Kendi içinde Alt obaların oluşturacağı

Adress gibi codde gibi

Soyad Kızlık soyad gibi

address → codde Sok, il İlçesi, postodu binaları ayırt etmemiz gerekmektedir çünkü bir degee cismen gerek

Select \* FROM Satis

Addressin spesifik obasına erişmem.

"Optimum Değil"

Where address

Likhe 10% 02%

## Single valued :

bir özellik tir (bno sno) gibi unique degerler.  
çoklu degeri gibi tek degerle özellikler.

## Multi valued :

Birden çok deger alır. telefon numarası nobeler.

0542 0541 bir kullanicının iki telefonu olabilir.

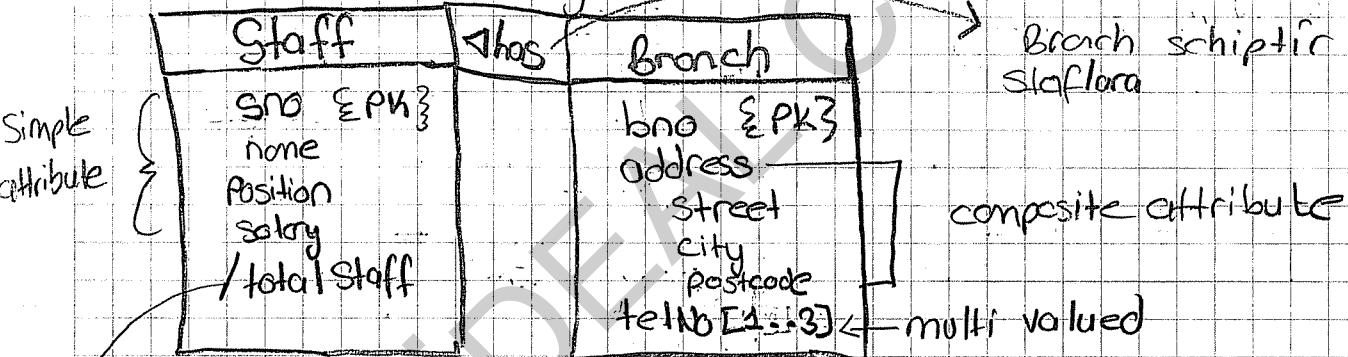
Birden çok nobelerde olabilir.

3 tane tel numarası nesil yopicaz 3 tane olağanlık bellegin içinde sitemli  
tablo segülenceleri bellegedir. ona göre tablo olağanlı

## Derived attribute :

Bu özellikler fiziksel olarak bulunmaz Dogum tarifi  
den sonra yes bilgisini olabilir.

Manager Staff branchi yönetir.



Füretilen bir attribute

→ Derived attribute

Haklı staf hangi staffları yönetir olabilir.

Hangi staf ordanada ne kadar yönetici olabilir.

fazla masclar olabilir.

Entity

Strong

- tek basına var olabilir
- tek basına çoklu gibi olabilir
- önceliği olabilir

- Dersler baska tabloya gerek yok olusmustur.

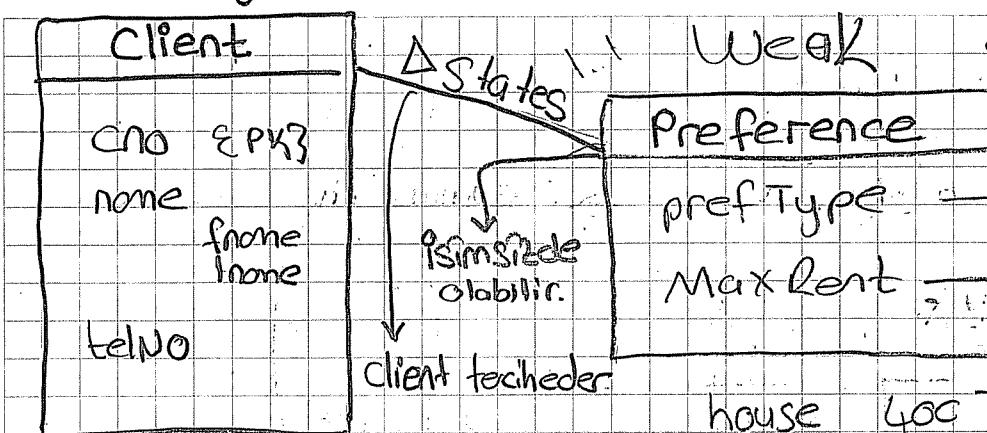
- Baska tabloya ihtiyac olmadan olabilir.

Weak

- Baska tabloya ihtiyac duyarlar

strong

→ Bir varlık olusumu basılı varlık sayılır. weak varlıktır.



Müşterinin Tercih ettigii  
ev türü vermek ileyikli tutor  
Tercih varlığı.

→ Tercih edilen tür.

→ Max. İddanecek tutor.

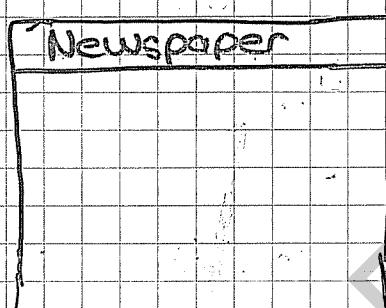
house 400 → Bu tür bir türün

flat 600, ifade etmeli  
kendi başına

→ Client olt öznelleri kendisinde bir bütünü var

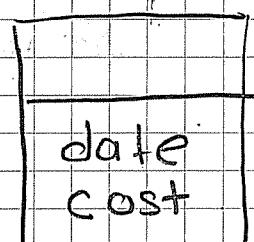
→ Preference tek başına bir aranın içermiyor.

Pişkilerin "Özellikleri" vardır.



→ Gazete ekstra bilgiler tutarock  
bu varlıktır.

Gazete ibarede  
evi.



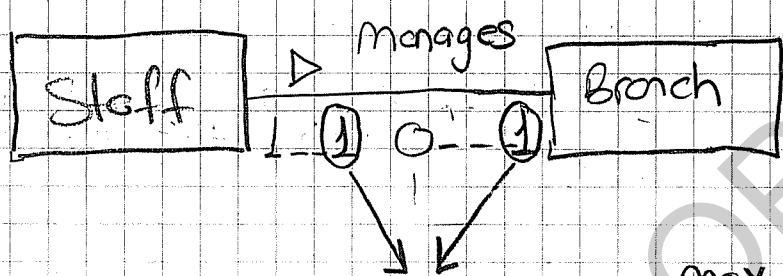
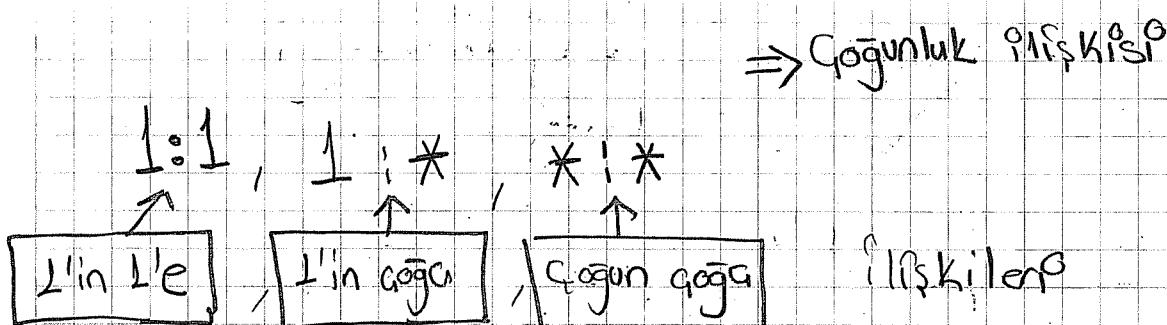
→ İlhiliğe ait bir özellikdir.

İlhiliğini ilhili ne zaman ve kim  
Maliyeti maliyeti nedir.

Bu bir entity değil bir özellikdir ilhiliğe

→ attribute'ler entity'nin özellikleri  
→ ilhiliğin özellikleri

## Relationship Types



⇒ max aldiğimiz  
⇒ max değerler ilişkili türünü belirtir.

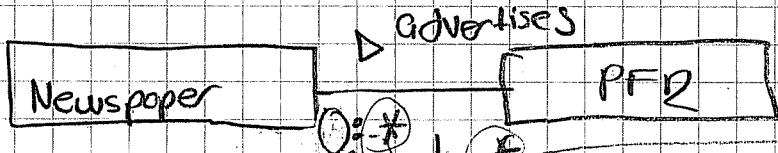


→ J. sitaf min. kag evi yönetir.  
" " max    " "    " "      Foreign key olur.

L oldugu taraf önemlididir.

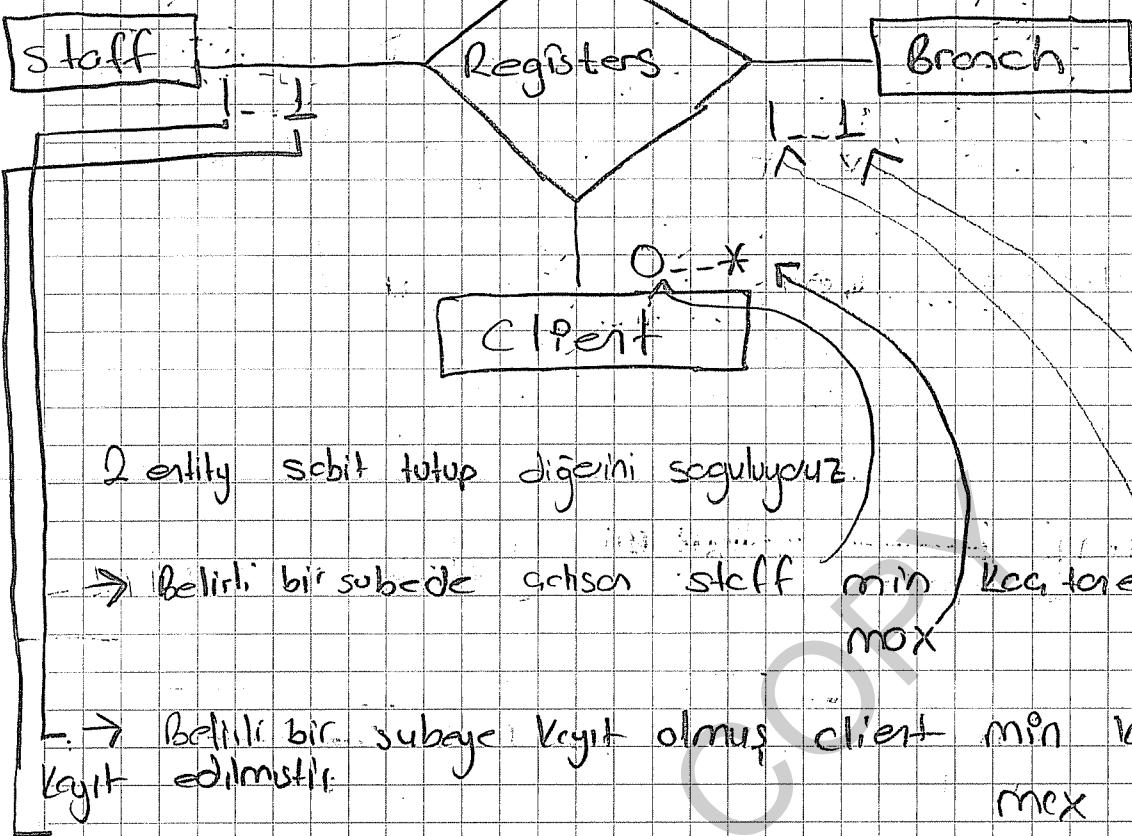
→ Lev. min kag Staff yönetilir.

→ " " max " "    " "



L gozete min kag evi ilan bilgisini verir)  $\Rightarrow n:m$

Lev min kag gozete ilan olusturur.



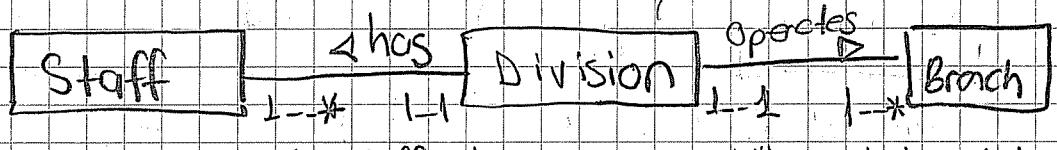
→ Belirli bir sube de action staff min kac tane client kaydetti max

↳ Belirli bir sube de Kayit olmus client min kac staff toplamda kayit edilmistir.

→ Belirli Bir Staff toplamda Kayit edilmiş client min kac subeye Kayit edilmiştir.

1-1 Fan Traps Problemı =>

↳ Bölgeler



Kac staff silip

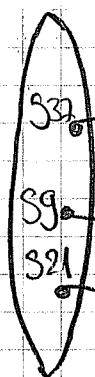
bölge subeleri gelistirir.

→ Ayni entity iken 2 den fazla 1..\* varsa Fan Traps olusur.

## II. chapter

Burda 1--\* var mı?

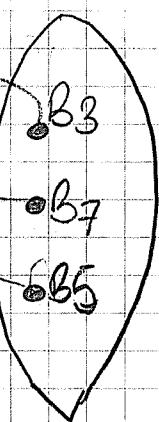
Staff



Division



Branch

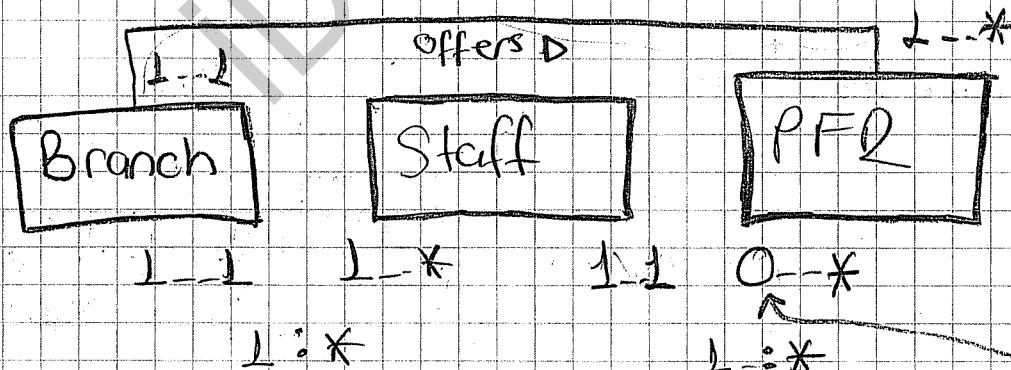


→ Staffın hangi Branch'la ilişkisi var, S37'nin hangi branch'la ilişkisi var.



Bu çözüm sıklıkla --\*

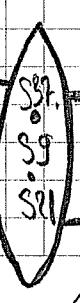
2-) Chashm traps Problemı =>



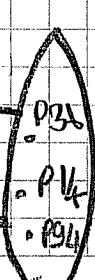
Branch



Staff



PFQ



Bir ev varigelince  
staff oturması 200lu değil  
ma bir branchde karsılık  
göreniyruz.

P14 numaralı EV hangi  
branch yönetir,

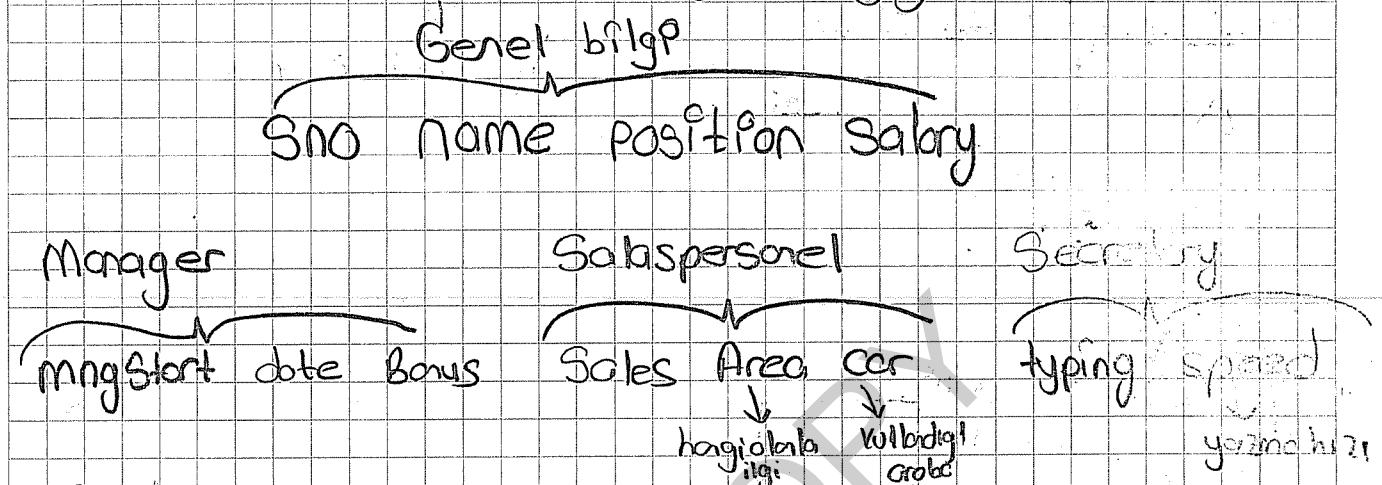
dogrudan birbirine bağlıysa 2

minimum kesit oluşturur

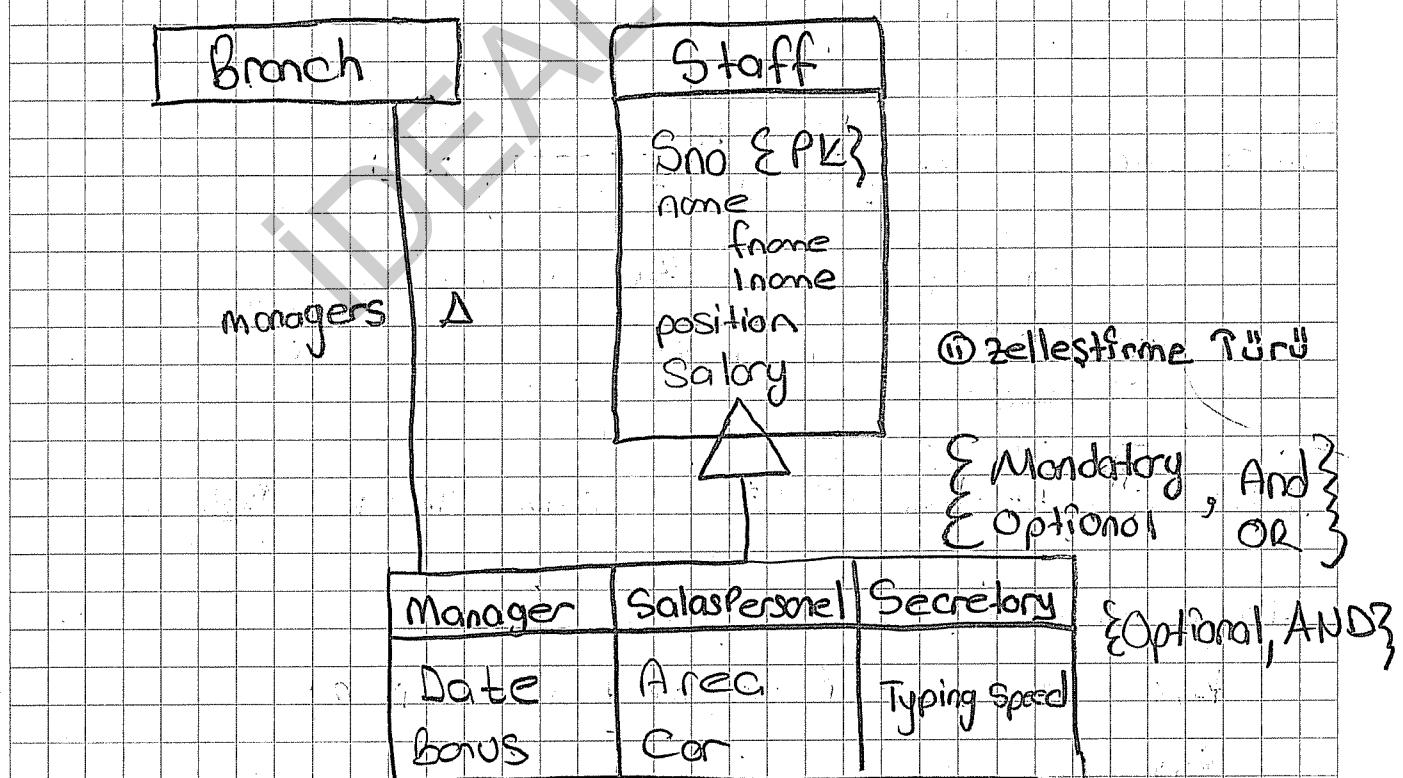
FABER CASTELL

## Enhanced ER Diagram

- Enhanced, ER gibidir sadece extra özellik gibi düşünülebilir.
- Özelleşme geliyorsa buna ihtiyac duyuyoruz.

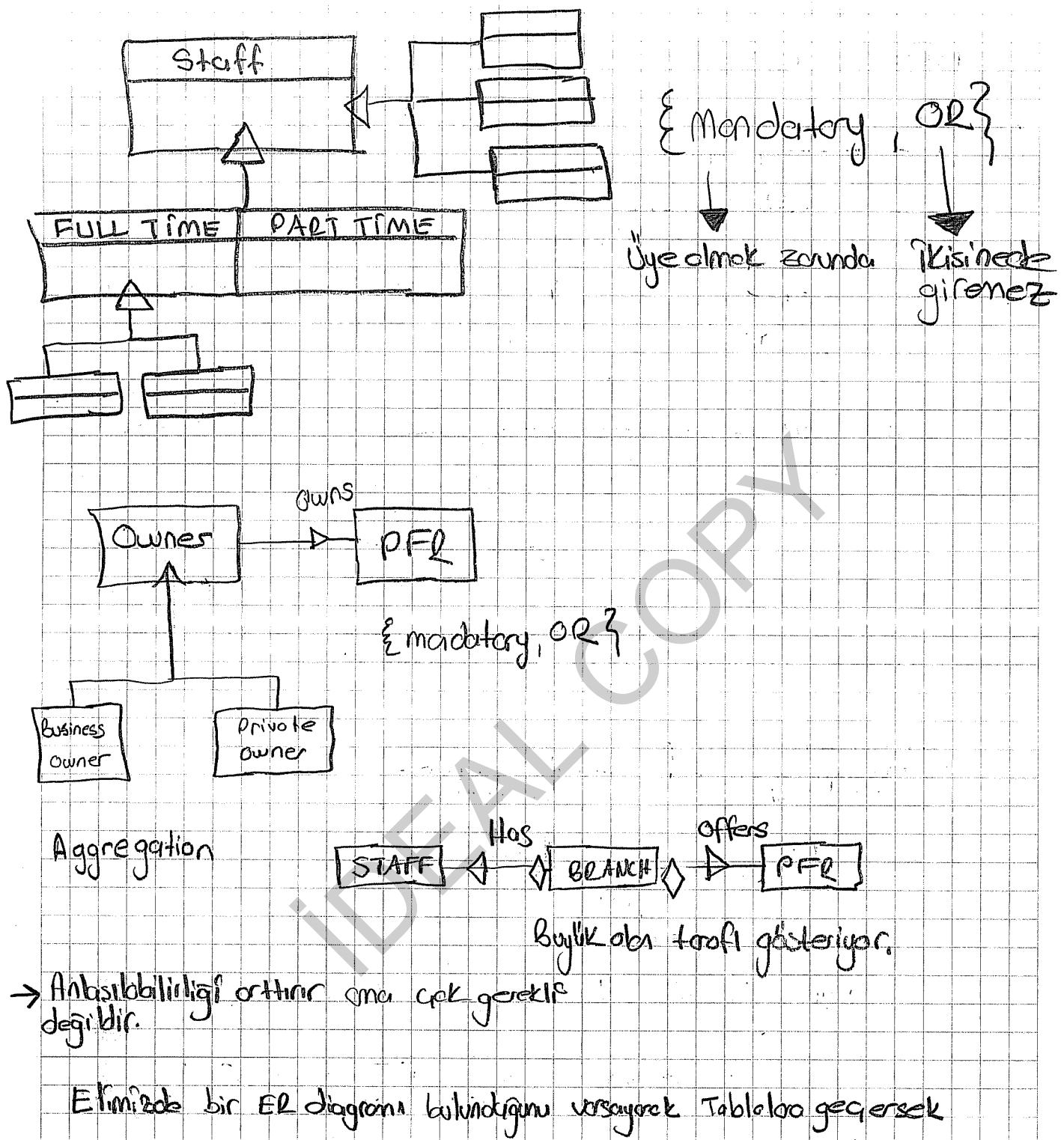


Bir Üniversitede Colson Kodlu tc Kimlik adı bulunan ortak eno akademik personel, idari personel diye ayrılır. Akademikte idarede kütüye kütünlük hızı gibi ayrılıyor.



Staff tablosunda her bir kişi alt sınıfların en azından 1 tanesine sahip olmak zorundaysa Mandatory yoksa Optional dur.

Super class içindeki tüm alt sınıfların yalnız birine sahip olabilirse OR birden fazla sahip olabiliyorsa AND dir.



1-) Strong Entity ( $\gamma \rho \epsilon =$ ) Strong entity tabloları için PK değerini ver.

Staff (sno, fname, lname, position, sex, DOB)

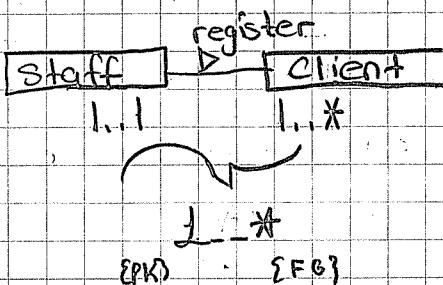
Primary Key (sno)

preference  $\Rightarrow$  tercih

2-) Weak Entity Type: weak entityler için yeni bir tablo

Preference (prefType, maxRent)

3-) 1:\*



Staff Client'i kaydeder

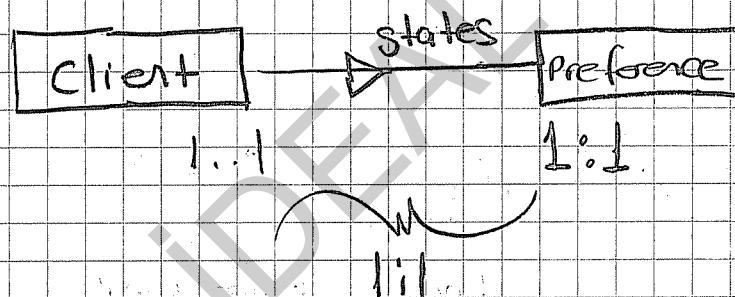
client (cno, fname, lname, telno)

PK (cno)

Bir tarafın (parent: staff) PK'sı child tarafına  $\rightarrow$  client  
Foreign key olarak telimlendir.

Foreign key sno References Staff (sno)

4-) 1:1 Relationships

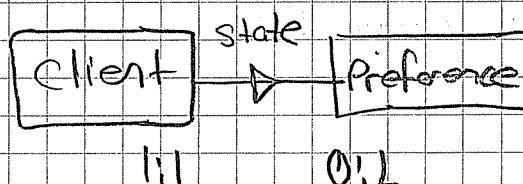


Teklilerin birbirine bağlıyoruz

client (cno, fname, lname, tel, prefType, maxRent, sno).

PK(cno)

FK sno preference Staff-(sno)



client max kira ve ev tipinin belirtmek zorunda degildir.

Preference (cno, prefType, maxRent)

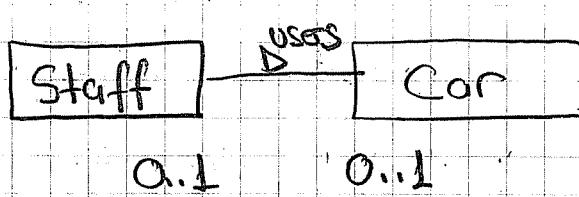
cno; FK cno; PK

FABER-CASTELL

} weak entity old için  
PK yoktu bu yüzden

} cno aynı zamanda PK  
oluyor.

Optional Part



Staff (sno, ... , carno)

Car (cno, ... )

$\Rightarrow$  Stafftan carno'yu null olur.

Staff (Sno, ... , 1)

car (cno, ... , sno)

$\Rightarrow$  Stafftan carno'yu null olur.

## (6) Super Class / Subclass

a-1 Mandatory AND

Herhangi birine  
üye olmak zorundır.

Herhangi ikisine dahil  
olabilir. Bu da null  
değer alma ihtimalleri  
üye olabilir. Düşürür.

(ono, address, telno, fname, lname,  
bname, btype, contactname,  
pflag, bflag)

Private  
owner

Bus owner



b-1 Mandatory OR

Private owner (ono, fname, lname, address, telno)

Business owner (ono, bname, btype, contactname, address, telno)

## C) Optional AND

Owner (ono, address, telno) → super sınıf için tablo oluştur  
owner details (ono, fname, lname, bname, btype, cname, pflag, bflag)

Alt sınıfların ikinci bir tablo ve bunun alt sınıf değerlerin bir tabloda topluyorum. Çünkü bir alt sınıf üye olmalı zannedilmez. Private veya Business'a ait olmaya bilir.

## D) Optional OR

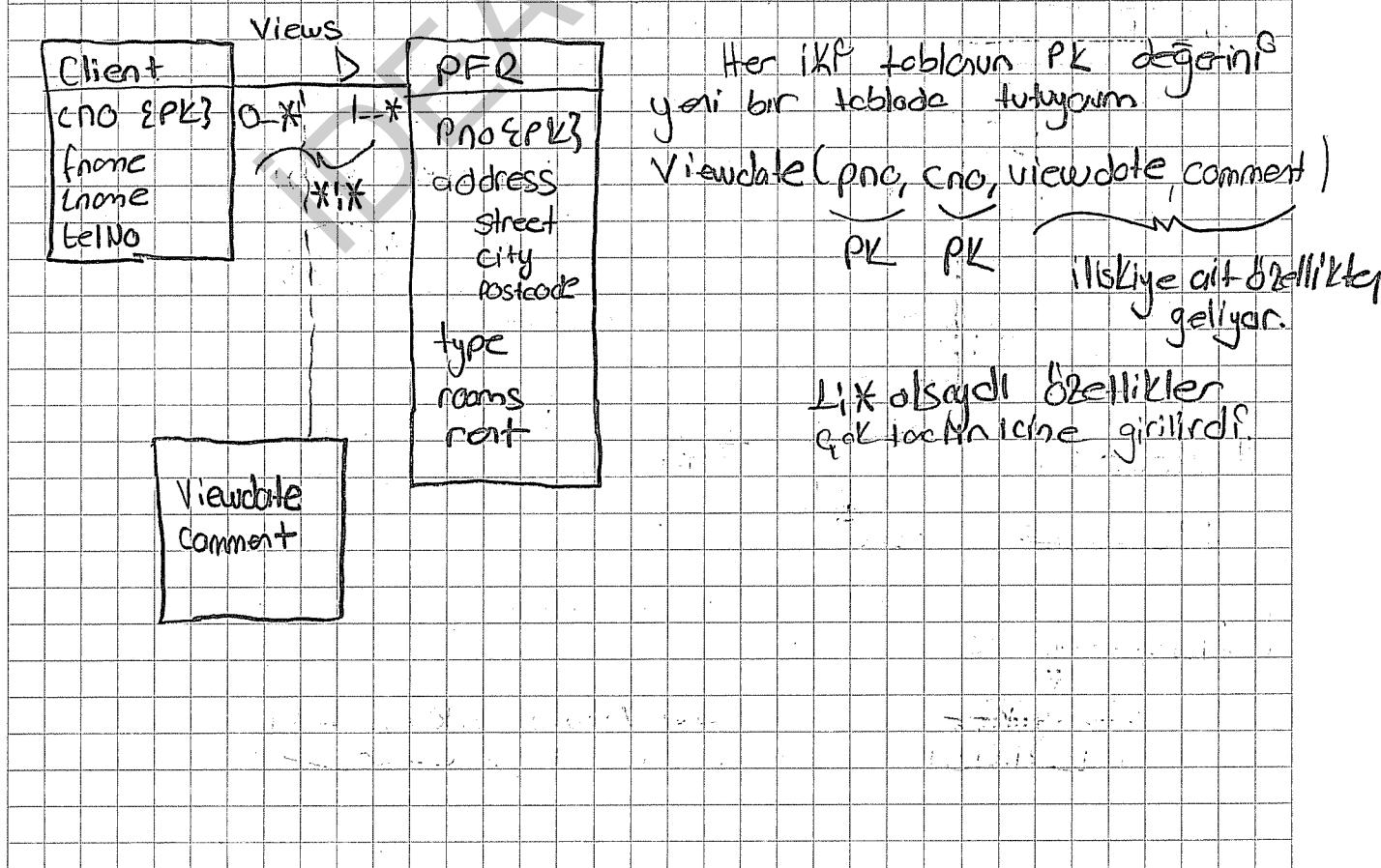
Aynı anda ikisine birden üye olmakla super sınıf ve alt sınıfları gösteren tablolar olacak 3 tane.

Owner (ono, adr, telNo)

Power (ono, fname, lname)

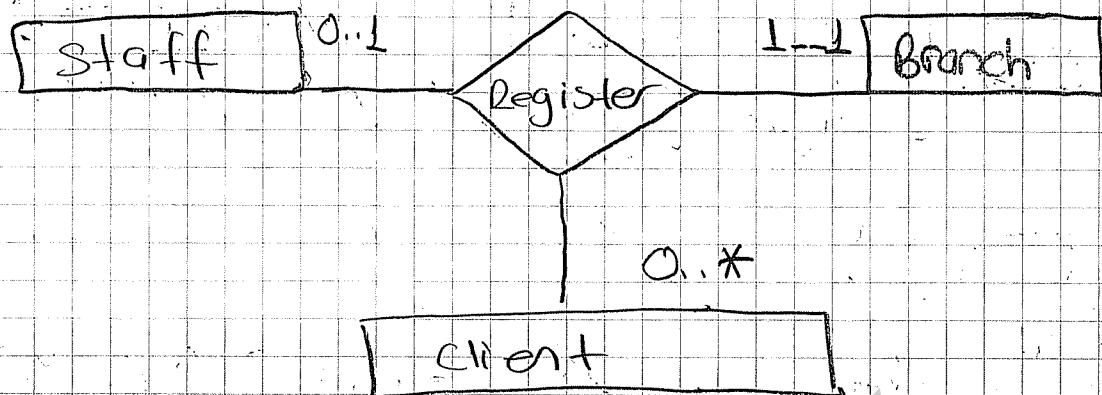
bowner (ono, bname, btype, contactname)

## E) \* : \* Relationship



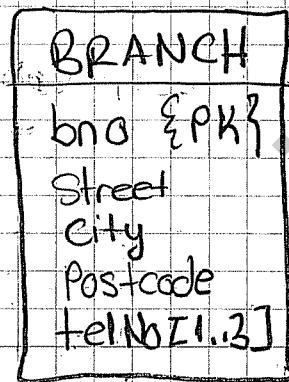
## ⑧ Complex Relationship

İki den fazla entitetenin bir araya geldiği ilişkiler.



Registration (lno, bno, sno, dateJoined)  
 ~~~~~  
 F.G F.G

## ⑨ Multi valued attribute



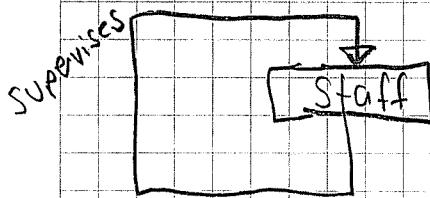
Branch (bno, street, city, postcode)

Telephone (telNo, bno)

birer PK

Yeni tablo oluştur. esas entry  
PK'sini de buraya yaz.

## ⑩ One to One (1:1) recursive relationship

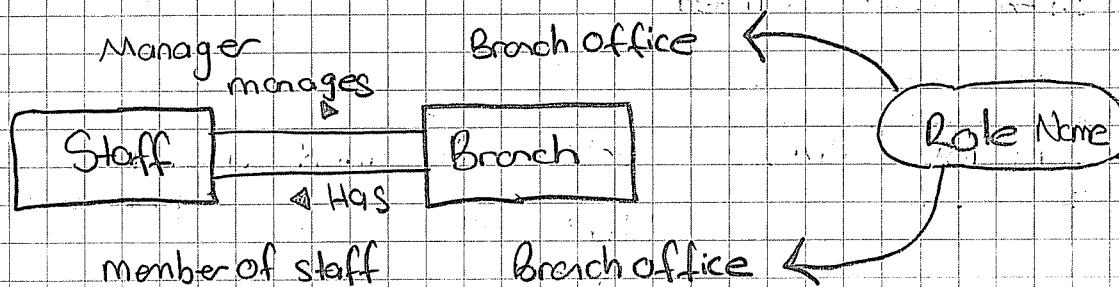


Supervises (sno1, sno2)

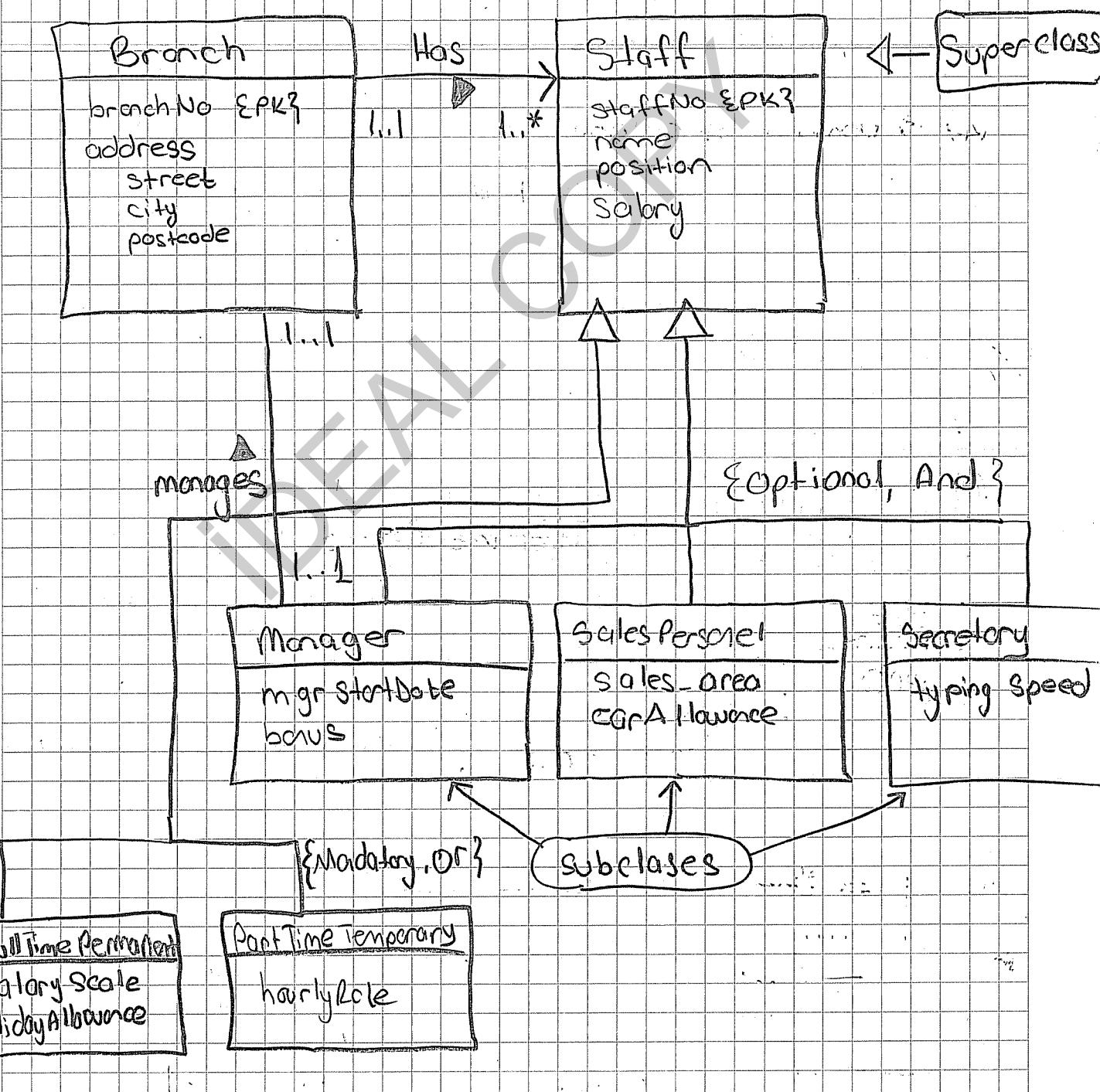
s687 s664

Primary key'in iki hanesi olacak kim  
Kim? Yönetici?

## < Recursive Relationship >



Example





- ① → Entity'lerin belirlemesi.
- ② → Entity'lerin arasındaki ilişkileri belirleme
  - er diagramı oluşturuyoruz
  - multiplicity belirlemeye gerek yok (İşkilerin tiplerini buluyoruz.)
  - ER problem vermi bunu belirliyoruz  $1..*$ ,  $*..*$ ,  $1..1$
- ③ → Attribut belirliyoruz bu entity attribute mu yoksa ilişkisel attribute mu diye...
  - simple, composite attribute
  - single, multi valued attribute
  - derived attribute

④ → Domain of attribute

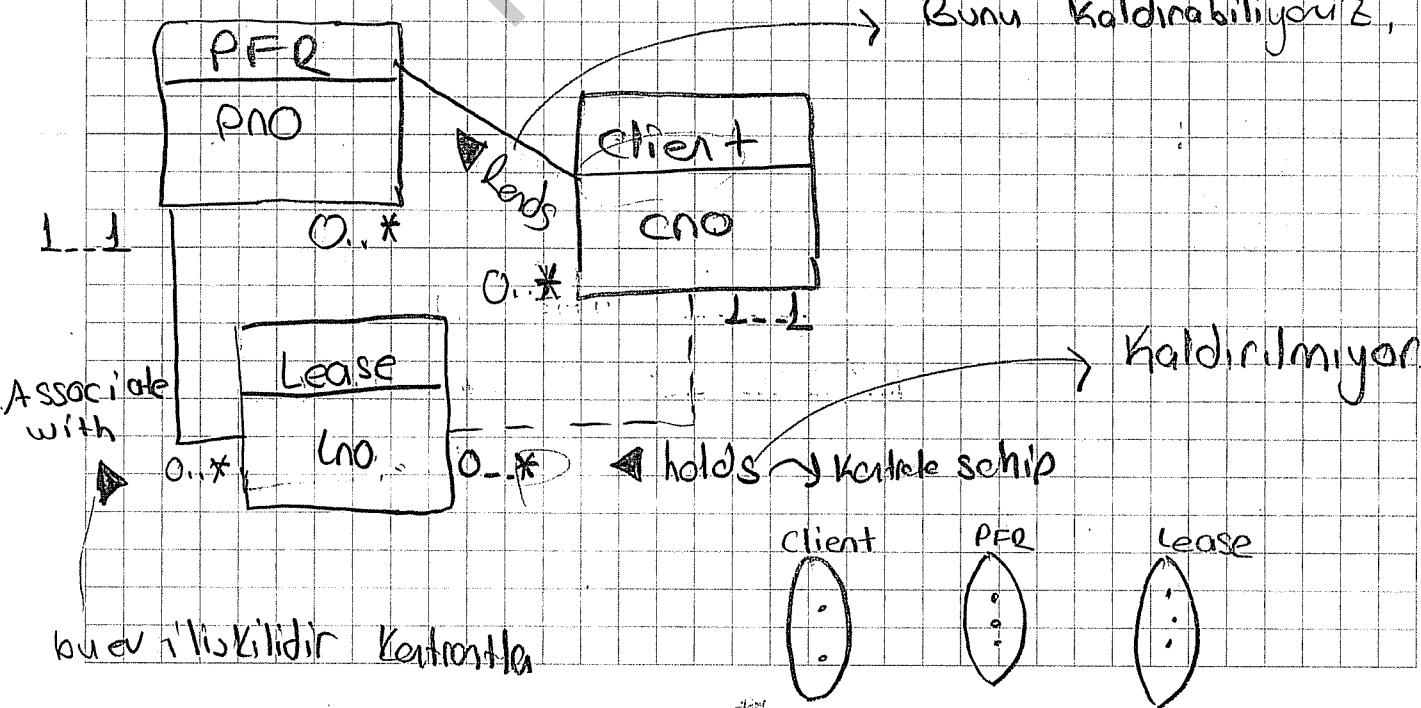
⑤ → candidate, primary key and foreign key

⑥ → Check model for
 

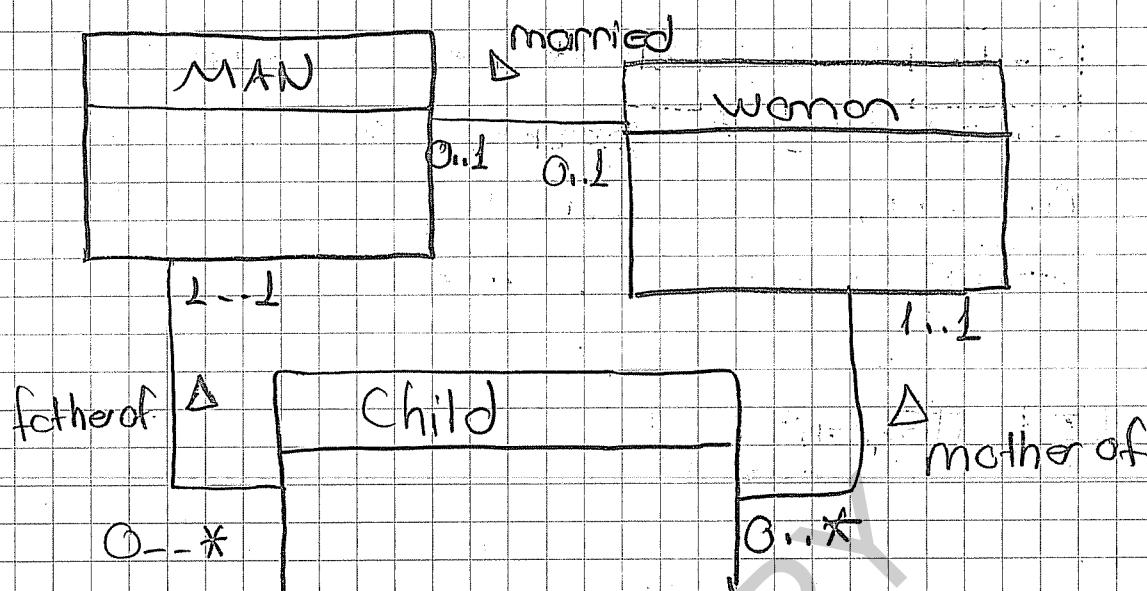
- veri tekrarı var mı fazlaların bilgi vermi diye test edicez...

Büyük test tespit edicez...

0-) Remove redundant relationships  
(fazlalar ilişkileri kaldırma)



b) Consider time dimension

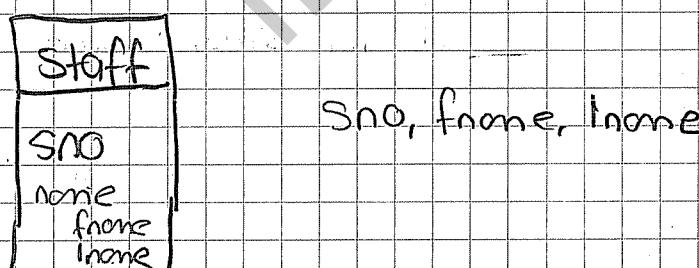


Burda Koldırma normu z gerekir

Logical DB Design

er tablolara dönüştürme hali

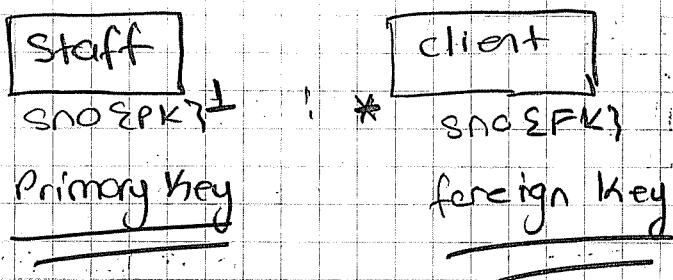
(1-) Strong entity type : elimizde strong entity iken he birine tablo clusturn içinde simple attribute oldum. Composite attribute varsa egezgik yopin.



(2-) Weak entity type = he bir weak entity iken yani bir tablo clusturn simple attribute oldum yazın. (weak entity pm key ensen belileyecek)

Preference (protoype, maxLen)

(3-) Eliminade  $L^*$  iliskisi varsa.



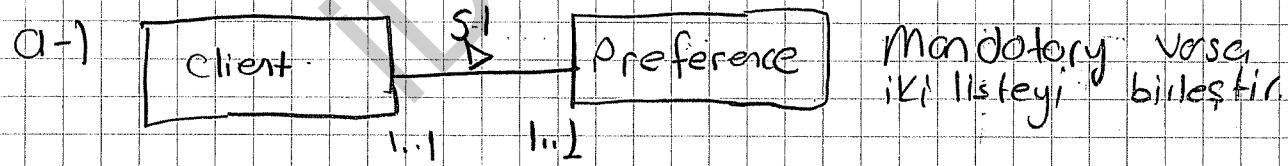
(4-) 1. 1. miskilenic

a-) her ikisi toplamının mindeki min olacak.

1..1 → 1-1 (mandatory) partie -

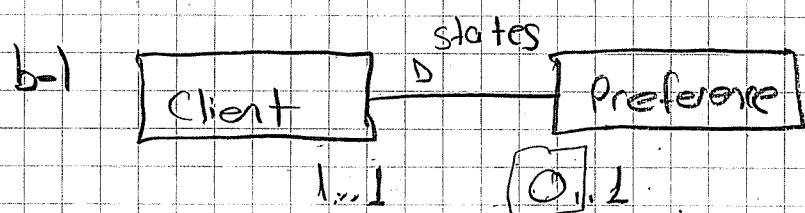
$1-1 \rightarrow 0-1$  (      )  
obviously  
dimensional.

①-1 → ①-1 (optional) in in both side



min Lise ikkjhdede  
tonnlar

{`CNG`.clone - :prototype, mendirif  
göruntü okulu teçhizatçılar}

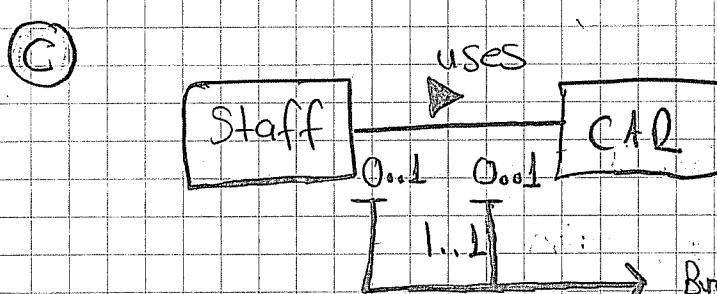


## Client (cnc, cnone)

Preference( $c_n$ , pretype, maxrent)

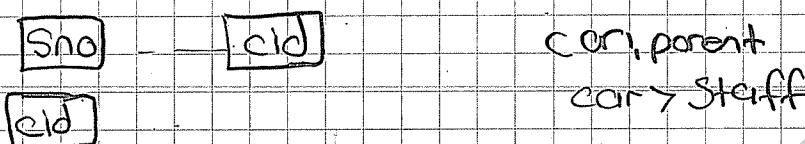
Terikh yomnaya geek yok.

S.F.K {PK} → P.Kolorok olusturmonitz FÄDER-CASTELL gerçkiyer olustururken



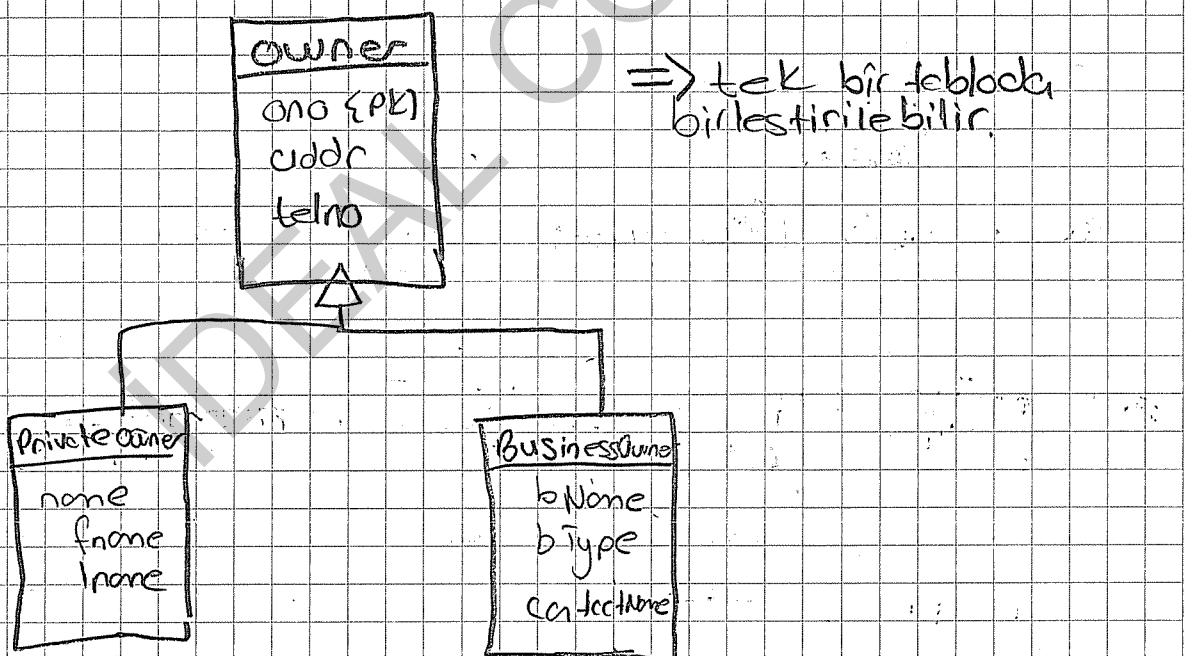
Birden birine parent, child Kuyruk  
hangisi geksə parent oluyor.

Parent hangisi ise child parentin pm Kini foreign key afor.



## 5-1 Sub / base Class

## (-) Mandatory. And



$\Rightarrow$  tek bir feblock  
bileştirilebilir.

Allowner(ono,addr,feno,fname,iname,bname,btype,conctNone,  
P P P P P P Powneflag, bOwneflag)  
b b b null null

## 2-1 Mandatory or

Private owner (ono, fname, lname, addr, telno)

Business owner (ono, addr, telno, bname, btype, concname)

## 3- Optional and

Owner (ong, addr, telno)

Allow owner (ong, fname, lname, btype, bname, concname, bflkg, pfkg)

## 4- Optional or

Owner (ong, addr, telno)

Private owner (ono) fname, lname)

Business owner (ono, btype, bname, concname)

6 Multif valued Att... nesn et me gib'i yon' bir  
tbls oluşturuyoruz...

→ PK olur

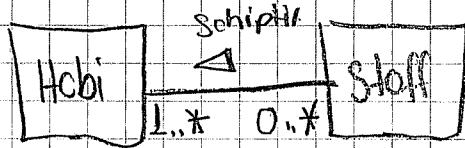
| bno | telno |
|-----|-------|
| FK  |       |
|     |       |

{Hobi}

Composite PK

{Staff}

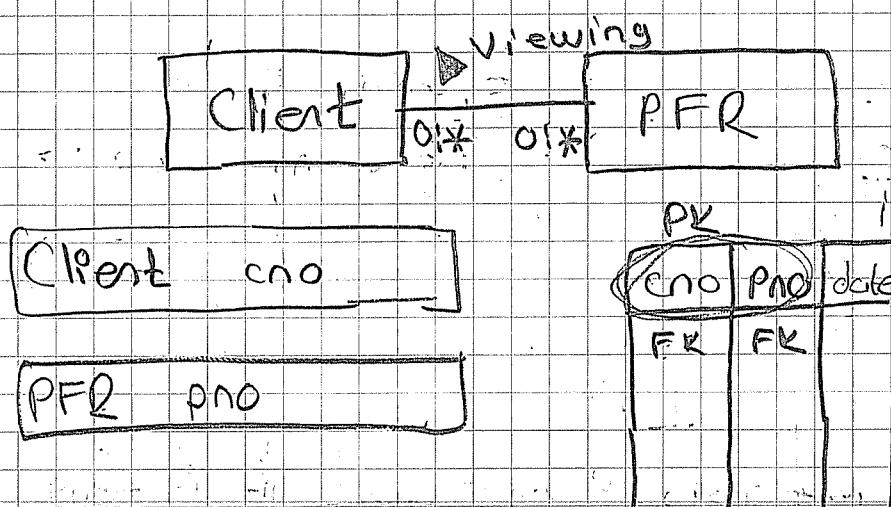
| sno | hobi |
|-----|------|
| S01 | H1   |



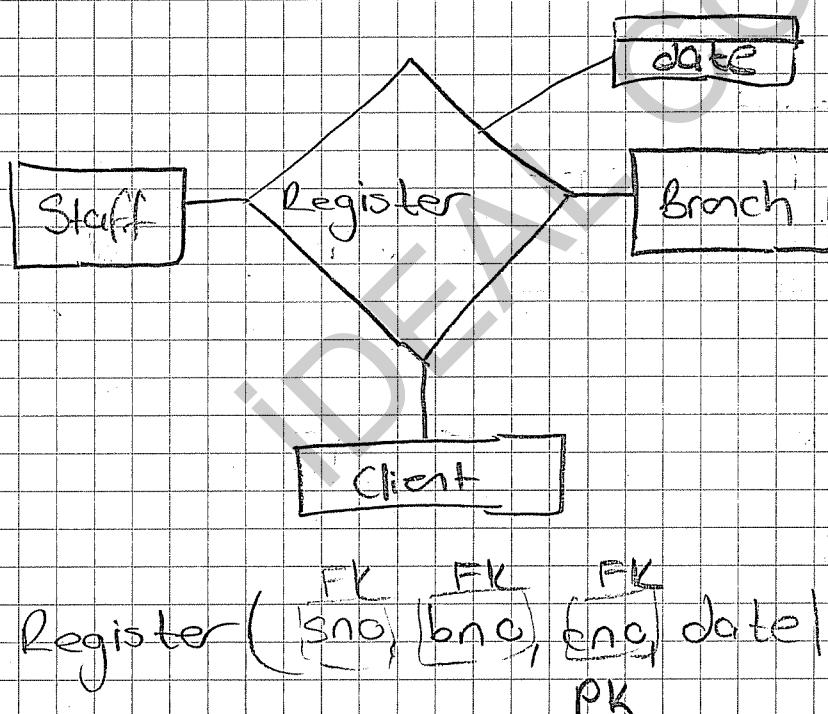
| sno | hobi |
|-----|------|
| S01 | H1   |

} composite PK

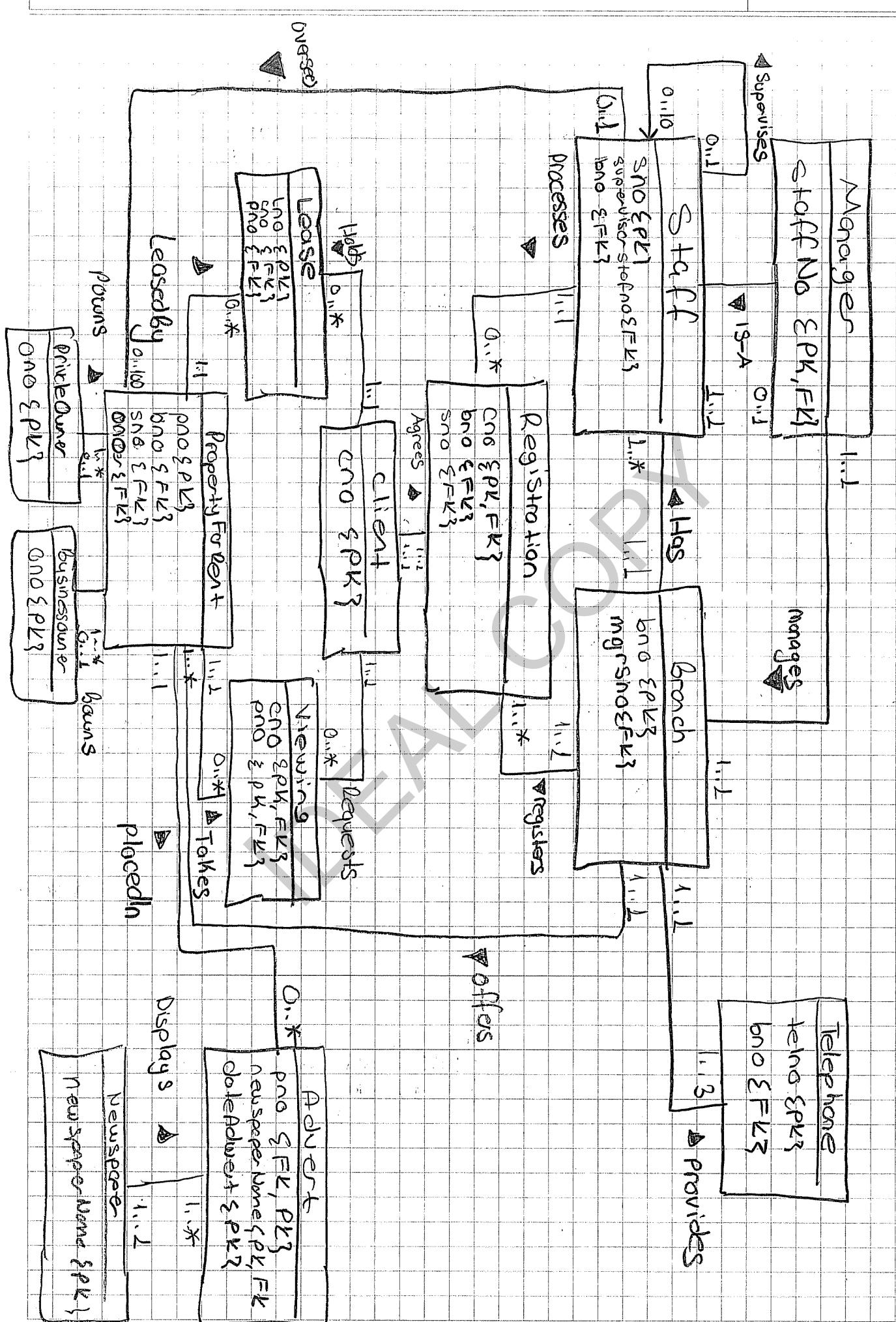
7-1 \*-\* ilişkisi

ilklerde ilk gelince  
enude buru kaydediz.

## 8- Complex Relationship



Göktuk ilişkisinin olduğu taraf PK cluyor.



## Normalization

ER

Tables

→ Sonucunda tablolar oluşuyor.

→ ER'den yada normalizationden doğru tablolara geçebiliriz.

## Data redundancy and update anomalies

### StaffBranch

| sno | sname | position | salary | bno | baddr   |
|-----|-------|----------|--------|-----|---------|
| S21 |       |          |        | B05 | London  |
| S37 |       |          |        | B03 | Glasgow |
| S14 |       |          |        | B03 | Glagow  |
| S9  |       |          |        | B07 | Aberdon |

→ Veri tekrarı varsa bu anomaliler oluşabilir.

### Update anomalies

1-) Insertion anomalies : Yeni bir staff ekleniyse B07 gelisiyor ve bu branchin adresi önceden tabloya girilmiş. Benim o yeri adres giverek yukarıdaki değerin aynı adresini girmek gerekiyor bude insertion anomalies olur buna.

Branch (bno,adr) → adresi burda tutucuz bu update olmaz  
Staff (sno, bno) <sup>staff tablosundan eklenince zaten bitti</sup> FK  
igretediyor.

→ burda branch to hiç staff olmese sno null olurdu bu da olmaz. bu tabloda ayrılmadığı için.

## 2-1 Deletion anomalies =>

Bu tabloda bir staff çalışırı branch o staff silinince bu gider sonki branch silinmiş gibi oluyor.

## 3-1 Modification anomalies =

Bir branch adresi update olunca tabloda aynı branchta olunan satırın güncellenmelidir. Güncellemesse hatalı olur.

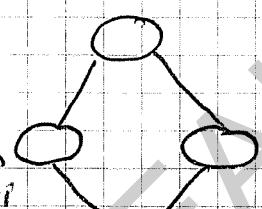
→ Bu anomalies çözümlemek için tabloları normalizasyon gereklidir.

Bir tabloyu normaliyorsak sulanız dikkat et.

## 1-Losles Join

→ Alt tablolara eklediğimiz bilgiler üst tabloda olmalıdır.

Alt tablolara eklediğimiz bilgiler üst tabloda birlesince silgiyi alt tablodan bu bilgiyi vermemesi gereklidir.



Kayıpsız olmalıdır.

Bunlar birlesince boş vermesi gereklidir.

Kayıpsız olmalıdır.

→ Join işlemini kayipsız olmalıdır.

## 2-1 dependency preservation =

→ one tabloda var olsalar kısıtlama alt tablolarda sağlanmalıdır. bağımlılıklar alt tablolarda sağlanmalıdır.

position → salary belirler.

deniz porulandığında bu bağımlılık sağlanmalıdır.

## Normalization

### Functional

1) Dependency (Fonksiyonal bağımlılıklar)

$$R = (A, B, C, St, Z)$$

↓  
ilişki

↓  
attribute

$$A \rightarrow B$$

B fonksiyonel olarak A bağlıdır.

→ anın her bir değeri için B tek bir değer alır.

determinant

$$\rightarrow AC \rightarrow B$$

AC nin her bir değeri için B tek bir değer alır.

$$SNO \rightarrow position$$

→ SNO her bir değeri için position tek bir değeri olur.

→ Bir kişiin tek bir pozisyonu vardır. diyeceğiz...

~~position → SNO~~

bir pozisyon için tek staff yoktur.

$$SNO \rightarrow Sname$$

bir SNO bir isme karşı düşer

~~Sname → SNO~~

• Aynı isimden kimse yoksa olur olsa ve olsa bu bağımlılık geçerli değildir.

## Full functional Dependency

A ve B bir ilişkinin attributeleri ise

$$\begin{array}{l} AC \rightarrow B \\ C \rightarrow B \end{array}$$

full functional olmasız

Determinantın alt kümeleri<sup>o</sup> segdini<sup>o</sup> taşımamalıdır.

$\rightarrow sno \rightarrow bno$  (böyle yaparsak full functional dir)

sno studente  $\rightarrow$  bno

farklıysa bağlılıdır.

Full farklıysa bağlı değil  $sno \rightarrow bno$  taşımaz büyükler,

$$AB \rightarrow C$$

$$A \nrightarrow C$$

$$B \nrightarrow C$$

full functional dependency

1  $\rightarrow$  1:L ilişkisi vardır.

2  $\rightarrow$  Butun Zorunlar ian geceli (Orki veri ile değil tümü ile korunur)

3  $\rightarrow$  Determinant toplı minimum sayıda attribute olmalıdır.

Yani composite keyler<sup>o</sup> en az ifade edebilecek duruma getirmeliyiz.  
primary key'in en az attribute ile ifade etmek gereklidir.

Partial Dependency =

$$A \rightarrow B$$

bu ilişkide A ya ait alt özellik cıkarıldığı halde bu devam ediyorsa.

İlk

$$SNO \text{ sname} \rightarrow bno$$

sname  $\rightarrow$  bno  $\rightarrow$  partial dependency (kismen bağımlı)

$$AB \rightarrow C$$

$$A \rightarrow C \text{ (PD)}$$

Transitive Dependency =

$$A \rightarrow B$$

$$B \rightarrow C$$

A  $\rightarrow$  C (Transitive dep.) (Geçişli bağımlılık)

SNO

$\rightarrow$  sname, position, salary, bno, baddr

$$bno \rightarrow baddr$$

$$SNO \rightarrow baddr \text{ formular}$$

Sağ tarafta birbirini işaret eden  
kızılık yaşa bu olur.

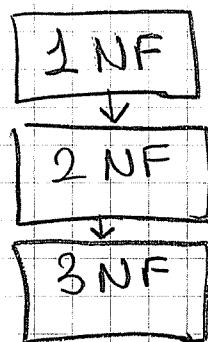
$$AB \rightarrow CDEF$$

$$E \rightarrow F \text{ (TD)}$$

$$A \rightarrow F \text{ (PD)}$$

Fonksiyonel bağımlılığı oluşturduğumuzda  
sonra bunlar belirlenir.

→ Fonksiyonel bağımlılıklar Primary Key'i verir bize



Partial dep ortodoks Kaldıracaz  
Transitive dep // //  
(Veri tekrarı ortodoks Kaldırmak)

Olmaması gereken bağımsızlıklar,  
bu lar neden olmazsa Veri tekrarı  
ortodoks Kaldırmak

1. Aşama

Client Rental

(PK) fd1 CNO PRO CHNAME PADDR STAT RFINISH RENT ONO ONAME  
| | ↑ | | ↑ | | ↑ | | ↑ | | ↑ | | ↑ | | ↑ |

fonsiyonel bağımlılıklar PM key bulmak denektir.

1 ve 2 kere kırılabilir diyince CNO PRO PM key  
burası fonsiyonel bağımlılığıdır. Diğer attributeleri tanımalar,  
full functional olmak zorundadır.

fd1 CNO, PRO → RSTART, RFINISH

(FD) fd2 CNO → CHNAME (tek basına tanımlayabileceğim)

(FD) fd3 PRO → PADDR, RENT, ONO, ONAME

(TD) fd4 ONO → ONAME bunu alkışmaya gerek yok.

(CK) fd5 ONO, RSTART → PRO, CHNAME, PADDR, RFINISH, RENT, ONO, ONAME

(CK) fd6 PRO, RSTART → CNO, CHNAME, PADDR, RFINISH, RENT, ONO, ONAME

adag  
cuchlar.

2. asma (PD) ortadan kaldırıcaz...

Her bir (PD) iain 2 tablo oluştur.

Client (cno, cname)

PropertyOwner (pno, addr, rent, ono, cname)

Rental (cno, pno, rStart, rFinish)

fd1 => cno, pno → rStart, rFinish (PK)

fd2 => cno → cname (PK)

fd3 => pno → paddr, rent, ono, cname (PK)

fd4 => ono → cname (TD)

(K) fd5' => cno, rstart → pno, rfinish, paddr, rent, ono, cname  
Lematikleden

(K) fd6' => pno, rstart → ono, cname, rfinish

3. asma

(TD) ortadan kaldırıcaz

→ Owner (ono, cname)

PFD (pno, addr, rent, ono)

Client (cno, cname)

Rental (cno, pno, rStart, rFinish)

bileşir kayıpları  
olu.

Ait tablolarda bir kayıt sagusunu ist tablolarının sagusunun  
verdiği sonucu vermesi gereklidir.

3NF

Her bir determinant  
sadece  
bir yerde  
bulunur.

Boyce Codd Normal Form

BCNF

Stafflig

Clientlerin görüşmelerin yapıldığı bilgileri tutulması isteniyor.

Client Interview

| cno | idate | itime | sno | roomno |
|-----|-------|-------|-----|--------|
|-----|-------|-------|-----|--------|

→ Her bir staff belirli bir günde sadece 1 odayı kullanıyor.



→ bir oda aynı günde birden çok staff tarafından kullanılabilir.  
roomno, idate  $\rightarrow$  sno  
 $\downarrow$   
sno

→ Bir müşteri bir saatte 1 kez görüşme yapıyor.

cno, idate

| cno | idate | itime | sno | roomno |
|-----|-------|-------|-----|--------|
|-----|-------|-------|-----|--------|

(PK) fd1

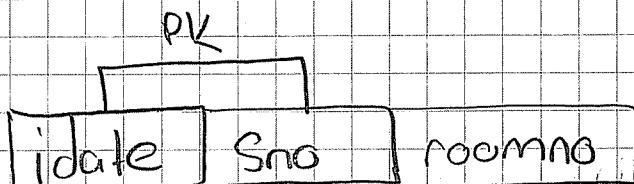
(CK) fd2

(PD) fd1

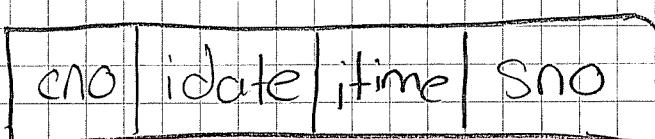
(CK) fd2

→ bunlar PK olması gereklidir.

Bir ilişkinin BCNF olması için her bir determinant adayı enktarasa olur.



→ ONE tablodan roomno  
çıktı...



BCNF tasimatı için FD4 kaldırımiyor tasıyor sonunda kaldırılacak

## Transaction Management Transaction Support

Vaibonindaki verileri tek bir kulanıcı tarafından güncellemesine  
bu işlemi Transaction diyoruz.

→ Staff denilen tabloya veri ekleme

→ Silme

→ update buton transaction demek

→ Staff numarasını değiştirdik evdeki yine de staffin  
numarası PFQ var bunabda numacyada değiştirmek  
gerektir.

→ Tek bir ifade değil bir çok ifade transaction olabilir.

read (StaffNo = x, salary) → x numaralı staffın salary'sini oku

salary = salary \* 1.1 → maasını 1.1 ile çarp

write (StaffNo = x, salary) → ve bunu ilgili staff'in salary'sine kay (güncelle)

a

1 transaction

delete (StaffNo = x)

for all PropertyForRent records, pno

begin

read (propertyNo = pno, staffNo)

if (staffNo = x) then

begin

staffNo = newStaffNo

write (propertyNo = pno, staffNo)

end

end

x numaralı staff'i sil

for durumu Prop.FD tablosundan kayit

pno su = pno olsun staff no'nu oku

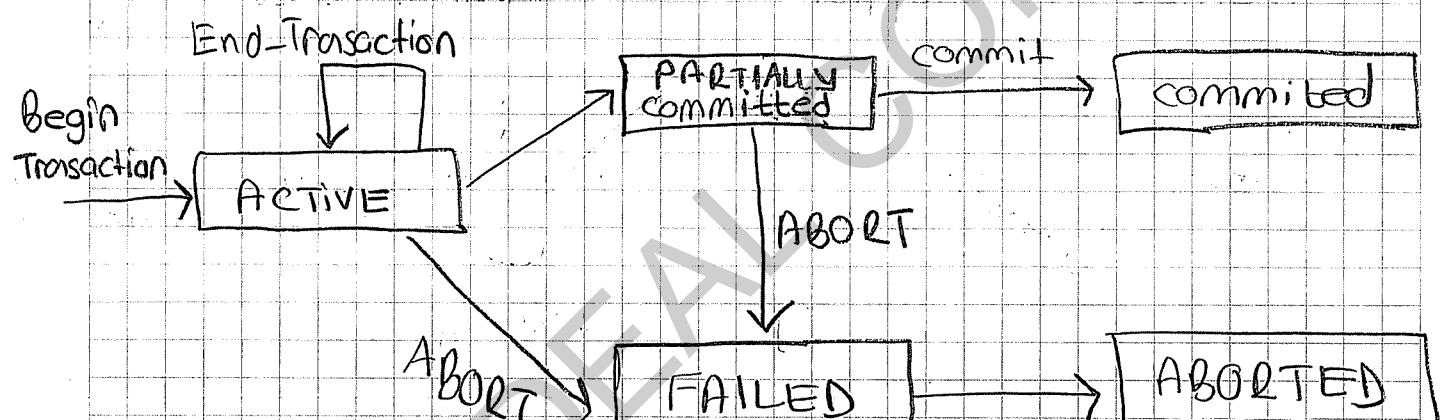
o kayit silinen kayitsa

buna yenisir staff No ver  
bunabda update et

2 transaction

- 2 farklı transaction aynı anda çalışması! bu iki transaction birbirine bağlı değildir.
- Birbirine bağlı işlemler varsa ozonun tutarsızlık olma ihtiyaci vardır. bunları doğru konföderenek gereklidir.
- Bu tablonuzda maşalılığı diğer sınıf ilgili.
- Ancak iki transaction aynı zamanın bir erişim veya ozonun tutarsızlık olabilir. bunları doğru konföderenek gereklidir.
- İki client NOR ve bunlar işlem yapıyor yani 2 transaction NOR bunları tutarı olacak şekilde bu istekler yapılmışlardır.

Transaction içinde bulunabileceği 5 durumu vardır.



→ İlkten transaction aktif

→ Son bir hatayı yaşadı PM key null atandı ozonun abort olur. Failed olur.

→ PFL yeni bir kayıt eklandı kişiye atandı bu ev bu işlem doğrulanmış olsun mu  $> 100$  se kayıt engel olunmalıdır. 100 fazla ekle ilgilenmez bu işten geri çekmen gerek cevap eklense ise (Partially committed) den abort olacak. cevap ekleyse commit durumuna geçer ve değişiklikler "daimi" olur.

- Transaction olması istenilen durumda salvamamışsa <sup>say transaction</sup> yaptığı tüm işlemler geri alınır.
- Öğrencilerin Findleri update ettiğ <sup>9/10</sup> oranda bulusun bulun ulandır. Elektrikler yesildi <sup>0/25 de</sup> o zaman veritabanı bekley er son transaction doğru biçimde salvadımi salvamamışsa yaptığı işlemler geri alınır. <sup>11</sup> ROLL BACK

## Properties of Transactions

Atomicity  $\Rightarrow$  Ya hep ya hiç <sup>özellik</sup>: Bir transaction ya hep olursa yada hiç çalışmaz yarida kalmaz <sup>genç</sup> alınır. Atamadan Recovery Management, Transaction Management  $\downarrow$  Scrumludur.

Consistency  $\Rightarrow$  bir transaction gerçekleşmeden önceki tutarlı durumun transaction gerçekleştiğinden sonra yine tutarlı olmalıdır. (tutarlılık) tutarsız bir duruma sekmeliydi.

Tutarlı olması kontrolü = DBMS, geliştiriciler.

$\downarrow$   
PK null alamaz  
Aynı değerden girilmez  
bunlar DBMS sağlı tutarlılığı sağlar.

FK cascade bunları önce bireye yerleştir sonra DBMS de buna gerçekleştiriyor.

Isolation  $\Rightarrow$  Eğer transaction gölistığında bunun uredeceği circa sınaqlar diğer bir transaction (Aynı anda çalışın) bunu negatif yönde etkilemeyecektir aynı anda gölistirilmelidir.

"Once T1 Sonra T2" gölistirilmelidir.

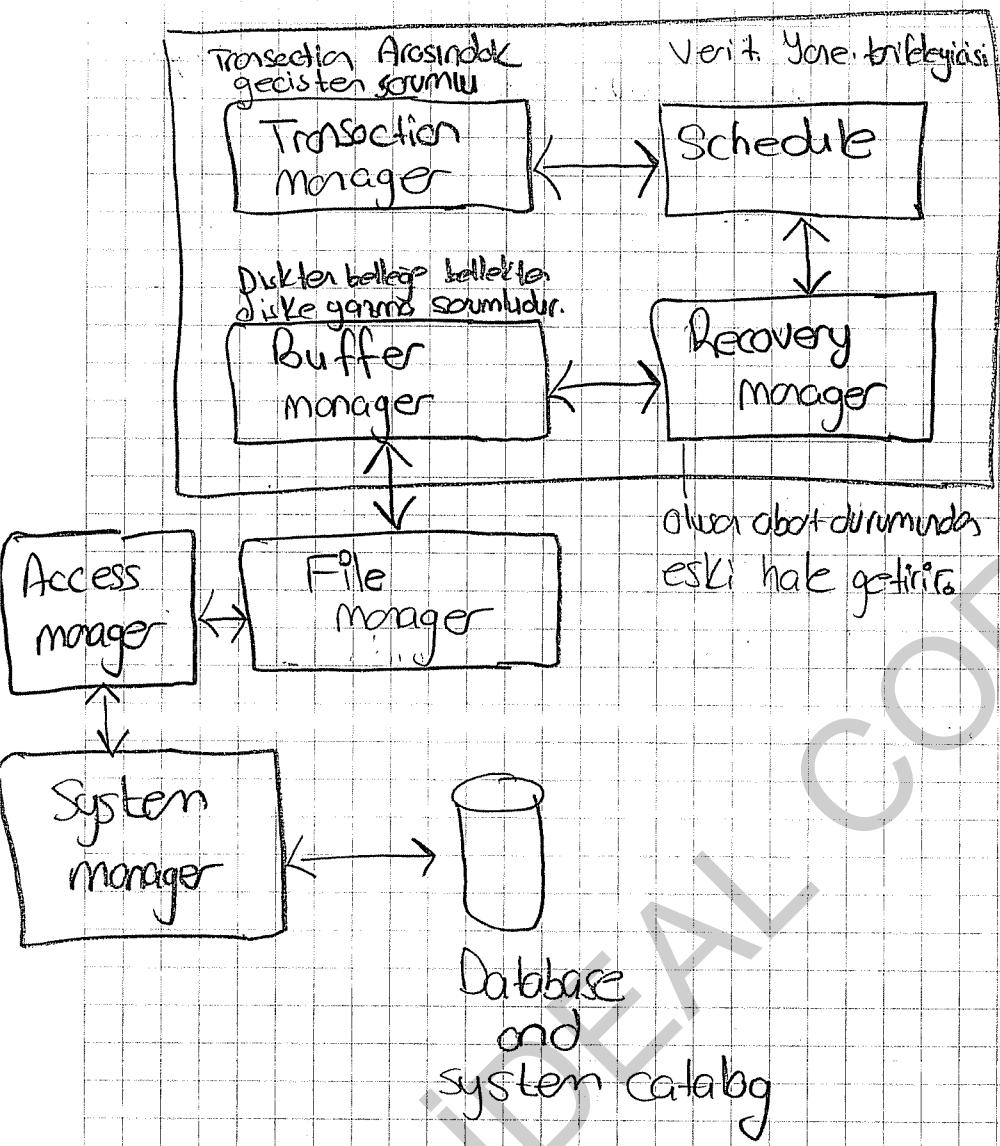
Scrumlu = Schedule scrumlu olduğudur.

Durability  $\Rightarrow$  Commit edilmiş transaction dahi olmuş veri <sup>özellik</sup> doğru biçimde salvamışsa, bunun yaptığı değişiklikler sürdürülür dması gerekir. Elektrik kesimelerinden sonra aktiflenmesi o değişiklik orda kalmalıdır.

FABER CASTELL = Recovery Management System  
Scrumlu = Recovery Management System

yapılan dan değişiklikler daimi olarak kaydedilir. Ancak her zaman bu elmaز.

## Database Architecture



## Concurrency Control

- es zamanlı olarak takip edileceğine gönüm
- Amerikan Throughput'yu test etmektedir. (Birim zamanda yapan çok iş miktarını iyileştirmeye)

## Lost update Problemı

rootback nedir yapar?

## Lost update Problem:

T<sub>1</sub>

begin - transaction

read (bal<sub>x</sub>) -

$$bal_x = bal_x - 10 = 90$$

write (bal<sub>x</sub>) - 90

commit

T<sub>2</sub>

begin - transaction

read (bal<sub>x</sub>)

$$bal_x = bal_x + 100 = 200$$

write (bal<sub>x</sub>) - 200

commit

bal<sub>x</sub>

100

106

100

200

90

90

→ Concurrent & broke collision transactionların ürettileri genug ile  
Seri拙k collision transactionları aynı olmalıdır.

burda 90

190

concurrent

Serial

Update Kayboldu...

## Uncommitted dependency problem:

T<sub>3</sub>

begin - transaction

read (bal<sub>x</sub>)

$$bal_x = bal_x - 10$$

write (bal<sub>x</sub>)

commit

T<sub>4</sub>

begin - transaction

read (bal<sub>x</sub>)

$$bal_x = bal_x + 100$$

write (bal<sub>x</sub>)

rootback

bal<sub>x</sub>

100

100

100

200

200

190

190

yapmış olduğu  
istende hala  
oluşturanı.

200 yanlış oluşturulan değer

şuan ki değer  
kırıldı degerdir.

## Inconsistent Problem

Transactionsın sadece okuma bile problemü üretebilir. Yüzne olmadan okunabildikçe kaynakları problem olabilir.

| T <sub>5</sub>                           | T <sub>b</sub>                              | bal <sub>x</sub> | bal <sub>y</sub> | bal <sub>z</sub> | SUM    |
|------------------------------------------|---------------------------------------------|------------------|------------------|------------------|--------|
|                                          | begin-transaction                           | 100              | 50               | 25               |        |
| begin-transaction                        | sum = 0                                     | 100              | 50               | 25               | 0      |
| read(bal <sub>x</sub> )                  | read(bal <sub>x</sub> )                     | 100              | 50               | 25               | 0      |
| bal <sub>x</sub> = bal <sub>x</sub> - 10 | sum = sum + <sup>100</sup> bal <sub>x</sub> | 100              | 50               | 25               | 100    |
| write(bal <sub>x</sub> )                 | read(bal <sub>y</sub> )                     | 90               | 50               | 25               | 100    |
| read(bal <sub>z</sub> )                  | sum = sum + bal <sub>y</sub>                | 90               | 50               | 25               | 150    |
| bal <sub>z</sub> = bal <sub>z</sub> + 10 |                                             | 90               | 50               | 25               |        |
| write(bal <sub>z</sub> )                 |                                             | 1                | 1                | 1                | 35     |
| commit                                   | read(bal <sub>z</sub> )                     |                  |                  |                  | 35 150 |
|                                          | sum = sum + bal <sub>z</sub>                |                  |                  |                  | 35 185 |
|                                          | commit                                      |                  |                  |                  |        |

175 olmamıştı sum

185 oldu olsunlar  
yanlış sıralanıyla

terif edidiği için  
uygun bir scheduler  
değildir.

## Serializability and Recoverability

(Serializability) (Geçici alınırlılık)

→ Transactionlar var bunları concurrent block called serializability bunları seri şekilde çalıştırıldığında üretilen sonuçlar aynılsa sonuçtur. Tutarlılık yok bu da serializable deniyor.

Schedule  $\Rightarrow$  concurrent block koşuluyla böyle bir torfeliyorum k' normaliyle tutarlılık gösteriyor.

insert update veya bu sırada korunmalıdır. Önce silip sonra güncelleme yine aynı olmalıdır tersi olmucaktır.

Serial Schedule  $\Rightarrow$  Transactionların sıralı gerçekleştirilmesi  $T_1$  başlıyor.  $T_2$  bitti.

Nonserial Schedule  $\Rightarrow$  Sıralı değil operasyonlar  $T_1$  sonra  $T_2$  gerçekleştirilemeyeceğine  $T_1$  oldu bunu non serial schedule denir.

→ concurrentla üretilen sonuçla seriallock üretilen sonuç aynıysa bu schedule'in serializable olduğunu söyleyebilir. Yani serializable.

→ serializablede read ve write işlemlerin sırası önemlidir. İki transaction X değerini okusasa aynı zamanın içindeki birincisi birinci okuyur.

→ İki transaction biri read biri write ve bular farklı değişikliklerde bulunduğunda sırasının önemini yekter.

→  $T_1$  X değiştirdiyse diğer okuyor veya yazıyor o zamanı bunun sırasının önemini verdir.

Tanı felâme Hari tasla  $\Rightarrow$

T<sub>7</sub>

begin-tran

read(balx)

write(balx)

read(baly)

write(baly)

commit

T<sub>8</sub>

begin-transaction

read(balx)

write(balx)

read(baly)

write(baly)

commit

T<sub>7</sub>

begin-tran

read(balx)

write(balx)

read(baly)

write(baly)

commit

T<sub>8</sub>

begin-tran

read(balx)

write(balx)

read(baly)

write(baly)

commit

T<sub>7</sub>

begin-transac

read(balx)

write(balx)

read(baly)

write(baly)

T<sub>8</sub>

begin-tran

read(balx)

write(balx)

read(baly)

write(baly)

commit

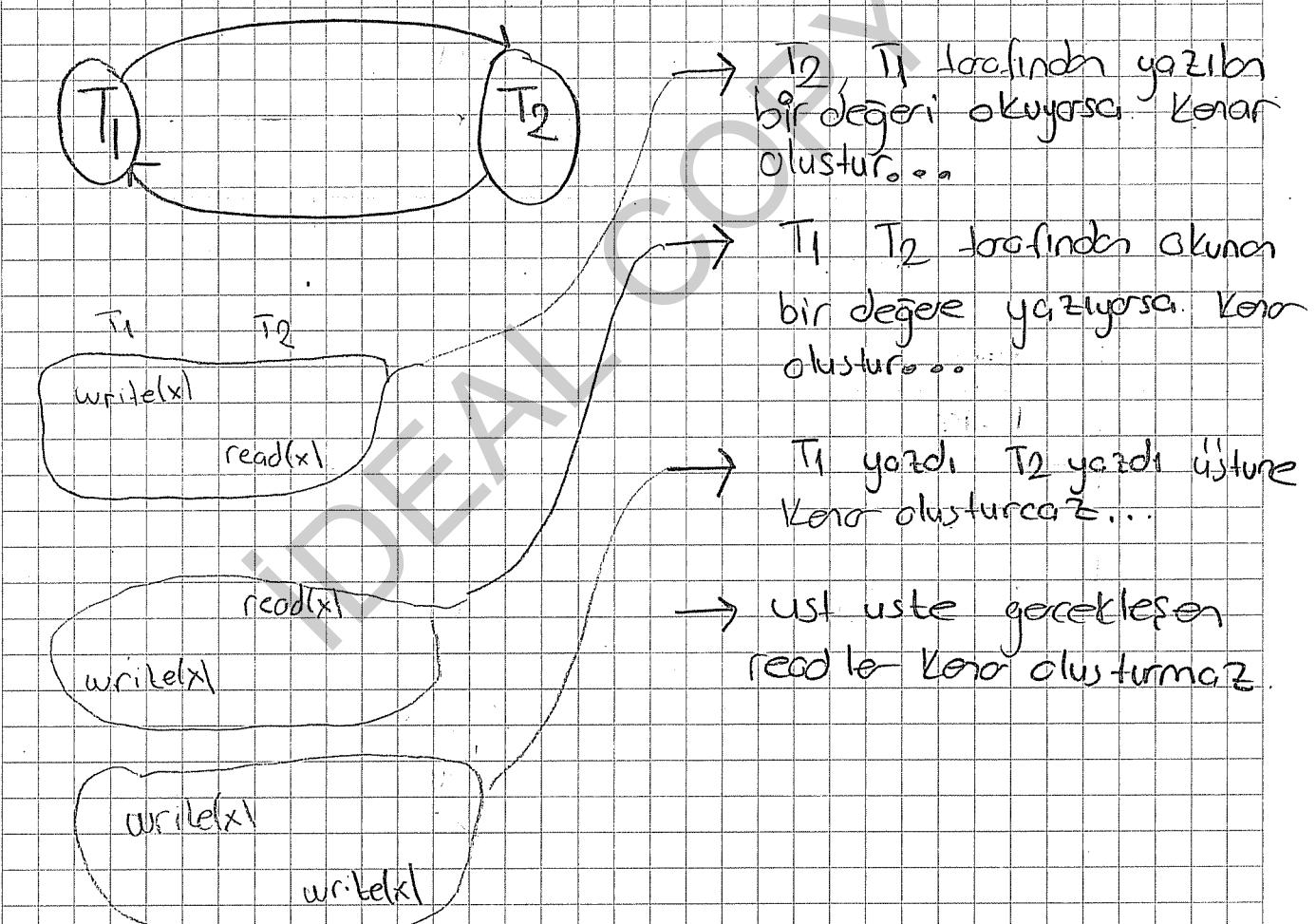
Serileştilebilir...

conflict Serializability:

- Bir transaction'ın aynı anda farklı operationların yerini değiştirebilir.
- Öyle değiştirmek için istemleri birbirini izleyici gibi işlemeyebilir hale getirir.
- Bir transaction'ın串行izable olduğunu nasıl test ederiz.

Test =>

I-) Her bir Transaction için bir node oluşturun...



T<sub>9</sub>

begin-transaction

read(bal<sub>x</sub>)

bal<sub>x</sub> = bal<sub>x</sub> + 100

write(bal<sub>x</sub>)

T<sub>10</sub>

begin-transaction

read(bal<sub>x</sub>)

bal<sub>x</sub> = bal<sub>x</sub> \* 1.1

write(bal<sub>x</sub>)

read(bal<sub>y</sub>)

bal<sub>y</sub> = bal<sub>y</sub> \* 1.1

write(bal<sub>y</sub>)

commit

read(bal<sub>y</sub>)

bal<sub>y</sub> = bal<sub>y</sub> - 100

write(bal<sub>y</sub>)

commit

w(x)

r(x)

T<sub>9</sub>

T<sub>10</sub>

r(y)

w(y)

→ oluşturulan schedule cycle içerisinde bu birfelenecez serisi obrak yani aynı sırada üretmez...

cycle vase, bu doğrudır  
birfelene degildir.

elimde schedule var bu seileştirilebilir mi diye soruyuz -  
açıklaması bekarak.

diger Graf teknikleriyle hizli bir sekilde çözülebilir.

# View Serializability (Görünüm Sıralayılabilirliği)

İkisinde hedefi Schedule'in serileştirilebilir olup olmadığına  
Korur vermektedir.

2. října 2019 v 14:00 hodin na křižovatce ulic

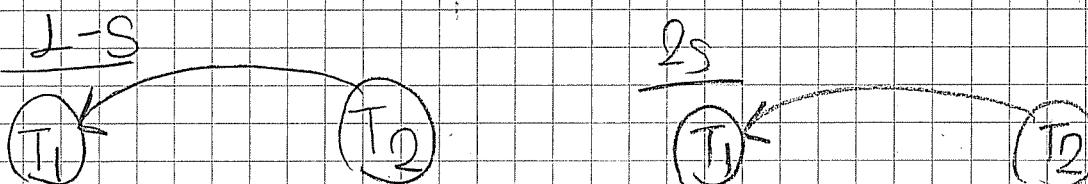
→ İki farklı tarafın ekonomik alısun  $S_1$  ve  $S_2$  ve her bir tarafın aynı sayıda iskeleye ve n tane transaction dan olusun...

bu iki kriterlerin eşit olduğunu söyleye bilmek için  
, grundte eşit olduğunu söyleye bilmek için B koşul sağlanmalıdır.

→ Neden bu iyi görüntünen esitt oldugunu soruyorum bir tariheksenin ser bir tarihekseneye Kosilik dusup dusundenesi  $\pi \approx 3.14$

1-1 Her bir veri için Transaction Ti LSchedule'da x'in ilk değeri 0 oluyorsa 2. koşulun eklendiği x'in ilk değeri 0 oluyor transaction o olmalıdır.

2-1 Her bir read işlemi igin 1 transaction topluca okunur  
X degse 2 transaction topluca yozulur bir degse.



1. trilemede 2 transaction hocinda yezilir  
x degerini okuyorsa 2. trilemede bunu qynisi  
geçerlidir.

B-1

Si Scheduler'la  $\times$  deðeri en son hangi transaction  
terafindeñ terafindeñ yazıldıysa diger Scheduler'da yine aynı transaction  
yazılmalıdır.

1-S1

L tran  $\times$  deðeri en son update eden

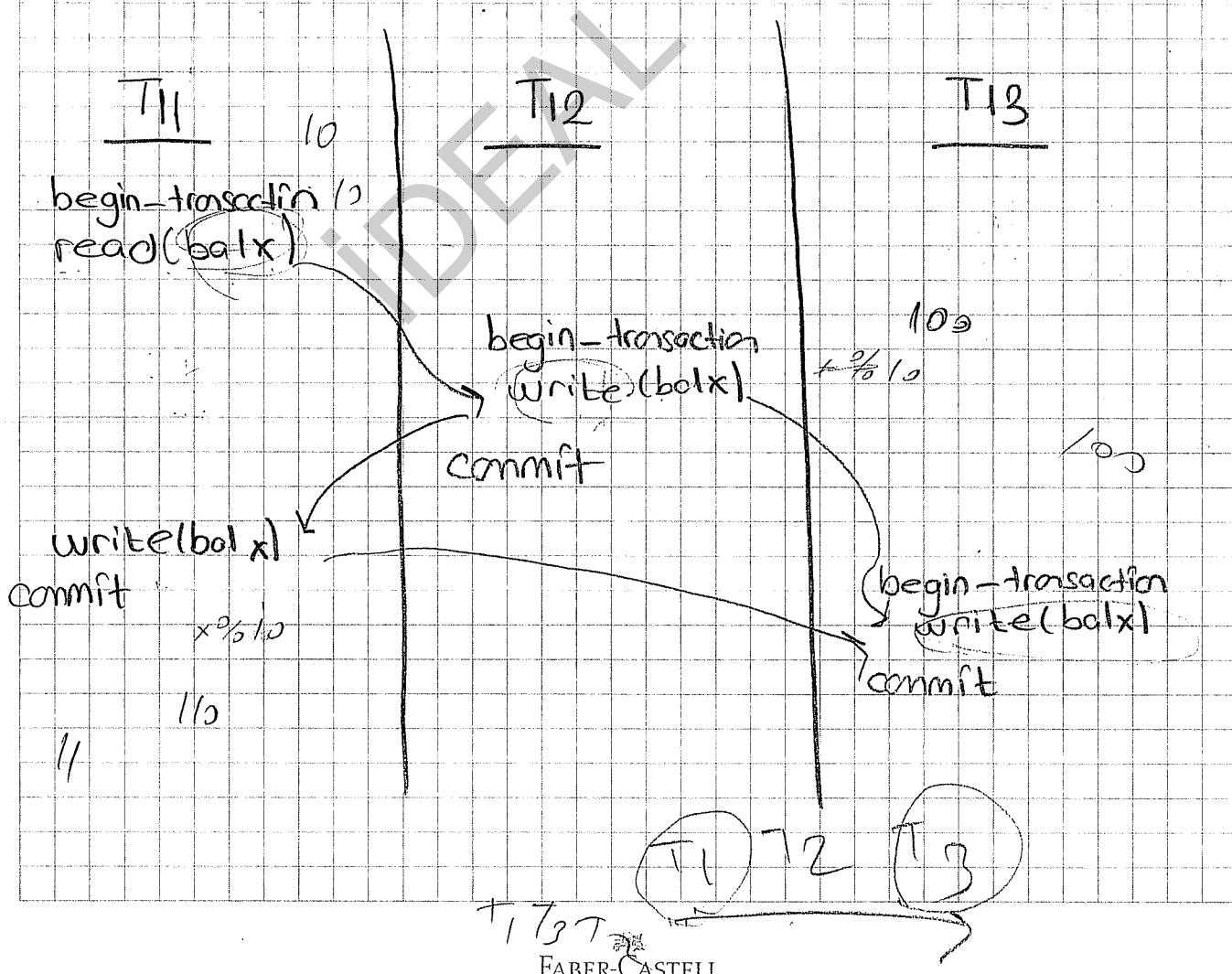
2-S2

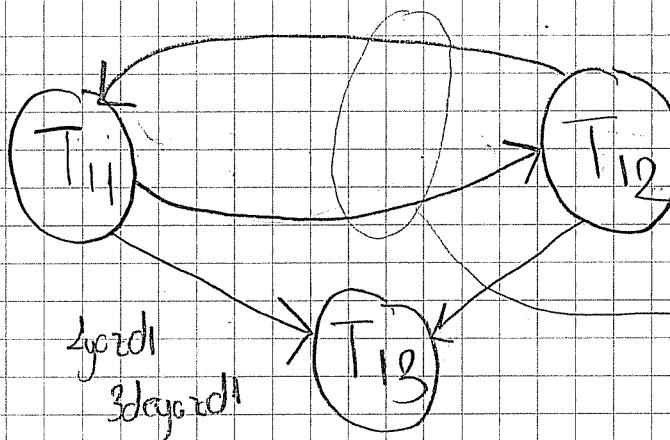
L tran  $\times$  deðeri en son update eden



2 Konsistansı aslinda biri seri olarak bunu kiyaslıyoruz.

L Scheduler conflict set. aynı zamanda view seri'dir.  
on taması doğru değildir.





View midir?

Seri hali nedir.

```
begin transaction
read(balx)
write(balx)
commit
```

```
begin transaction
write(balx)
commit
```

```
begin transaction
write(balx)
commit
```

1-) Yalnız degeri 11 okuyor  
(kisindek...)

2-) Yazma ve okuma yoktur  
burda. yazip → okuma

3-) Son yazan gitti kişi olmazdır.

Bu view dir. Scheduler

$T_p \quad T_j \quad T_k$

$T_{bw}$   
 $w(A)$   
 $w(A)$   
 $w(A)$

$T_3 \quad T_4 \quad T_7 \quad T_8 \quad T_9 \quad T_{10} \quad T_f$

$r(Q)$

$w(Q)$

$r(Q)$

$w(A)$

$r(A)$

$w(Q)$

$w(Q)$

$w(B)$

$r(A)$

$w(A)$

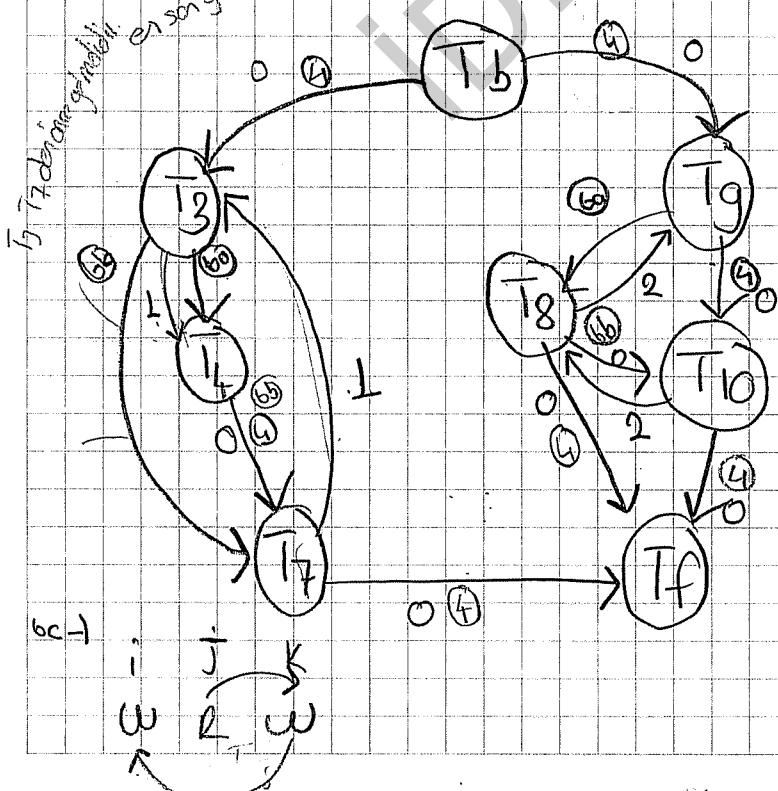
$\text{read}(A)$   
 $\text{read}(A)$   
 $\text{read}(B)$

→ Bir degisken ilk kim okuması alternatif teyine okunmalıdır

yazılım L ve 2 de sadece  
ini kullanıcaz...

Ortada gereklesen write, read  
write bırakıcaz

f → k demek okuma sonraki  
yazma



- 1-)  $T_0$  bir transaction'ının node oluşturur  
 2-)  $T_{bw}$  diye verilenin transaction basına ekle versaydı bu transaction  
 her ifade eden butun değişikliklere yazma yetisini veririz.  
 3-)  $T_{fr}$  verilenin transaction dir ve sonuna eklenir butun veilenin okunmasında  
 yararlıdır.  
 4-)  $T_i \rightarrow T_j$  Eger  $T_j$   $T_i$  tarafından yazılmış bir değerini okuyarsa

$T_{bw} \text{ } w(Q) \rightarrow T_3 \text{ } r(Q)$

- 5-) Bir veri  $T_j$  tarafından okunmuş, ona öteki önceden  $T_i$  tarafından yazılmış  
 $T_k$  de bir yerde yazılır bu değişiklik  
 ( $T_{bw}$ ) ( $T_{fr}$ )

a-) Eger yazan kişi ilk transaction'a okumayı yapmışsa son transaction değişilse  
 önceki okuma yapmışsa yazma yapma eğrisi K'ye doğru oku ve left olur.

Eger bir transaction bir transactionde Q'nun ilk değerini okuyasın denk olan  
 transactionde onun ilk değerini okuyan o olmalıdır.  
 $T_3$  de sonra ilk yazan  $T_4$

$T_{bw}$  yazıcak okuyucu son değişilse yazan transaction okuyucu sonra gelmesini  
 sağlayın önce  $T_3$  dusun sonra  $T_4$  yazın (ilk okuya kişiye yine ilk  
 okuyan dusun. yazma işlemi devreye girmeden okusun.)

Q'nun değerini ilk okuyan  $T_3$  ondan önce yazan 'bw' transaction  $T_2$  den sonra  
 yazın vero mi?  $T_4 - T_3 \rightarrow T_4$  3 önce dusun sonra yazilsın

b-) Yazan ilk transaction değişikten ( $T_{bw}$ ) önce okuyan son transaction sen  
 yazma işlemi önce okunmadan önce gelmelidir.

→ K değerini sen yazan kimse alternatif transactionde yazan o olmalıdır.

K-i den önce gelmek

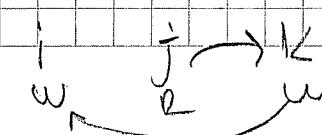
Aynı en son yazan  $T_{10}$  sonda 1 önce yazan  $T_2$

$T_2$  ve  $T_8$   $T_{10}$  den önce yazmalı en son yazan  $T_{10}$  olmalıdır.

Q'ları en son yazan 7 yazmış ondan önce 8 ve 4 var  
 en son yazanı geçiti ediyoruz.

c-)urbada gerçekleşen write, read, write birlikte

(Kendi uretinde olmucuk  
istende)



$T_2$  okucak  $T_K$  yazıcak sonra  $T_1$  yazıcak

三

```
begin-transaction  
read(balx)
```

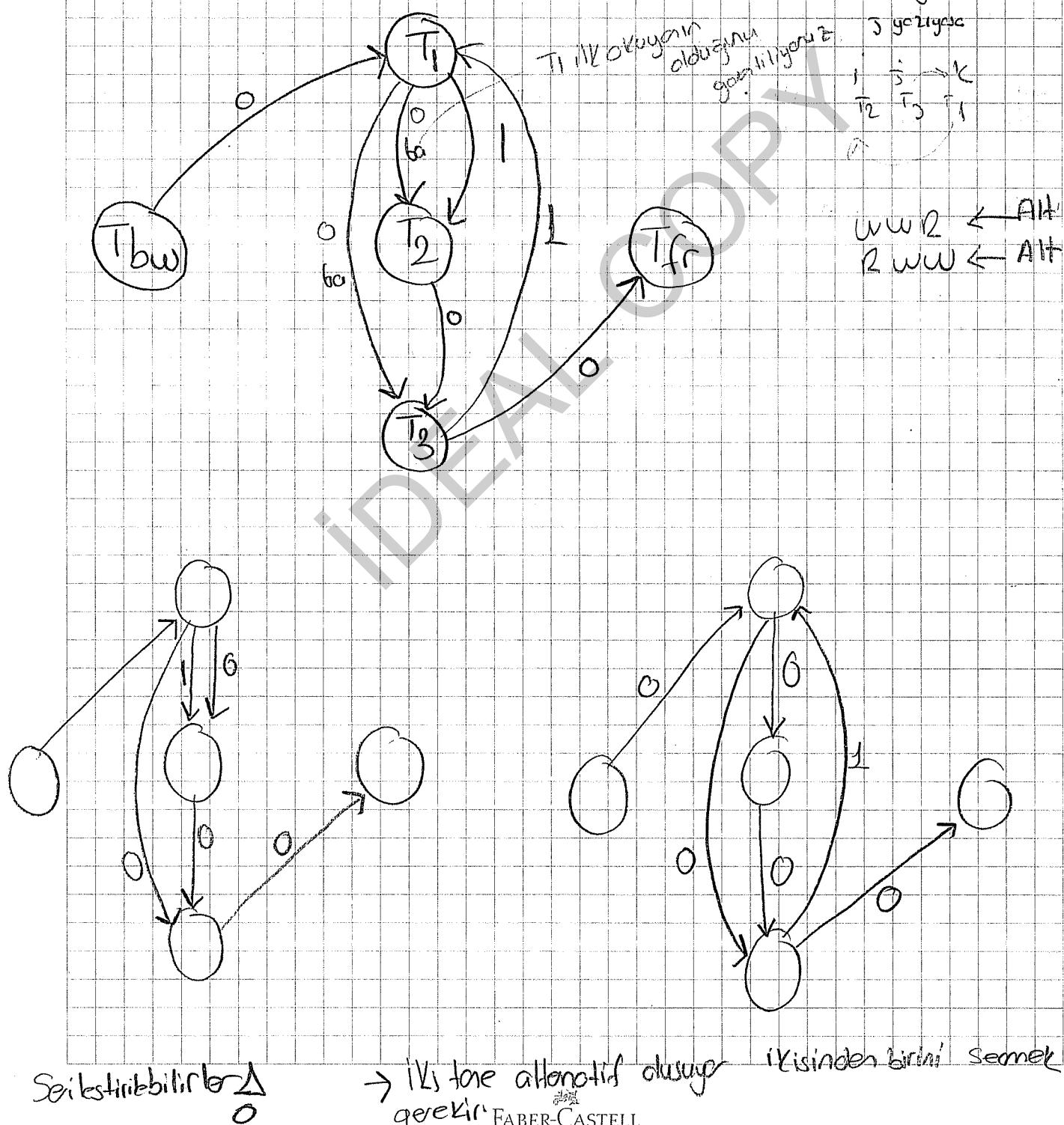
· Writeback  
commit

TQ

```
begin-transaction  
write(balx);  
commit
```

T3

begin-transaction  
record(bal x)



## Locking Methods

Veriye erişmede kullanılır bir prosedürdür.

→ Bir transac. veritabanında bir değişkene eriştiğinde Lockı erişmesini diye biri  
kullanırız. Sebeb yarlıs sonuçlar engel almak için.  
(Sadece okuma)

(read) → Aynı veri üzerinde birden fazla shared lock kullanılabilir.

Shared lock okunmak için

Exclusive lock  
(write)

update etmek için

(Yazma yapmak ) lone olur

Koşulları yapıyoruz --

→ Herhangi bir üzerinde exclusive lock yapmışsa başkasının bu okumasını engellerim  
(Hem okuma hem yazma)

→ Herhangi bir değişkenin verisin üzerinde kilit yoksa bu kilit size verecektir elimek  
istedığınız veride.

→ Lock close bir datayı elimek istiyorsanız bir diğerin üzerinde shared lock var  
başka okunmak için kullanırız DBMS de bu da kilit ve bu okunu killidi  
başka okunmak isteyen 2ndn bu ona verebilirim des.  
full durumlarda siz bekletecektir yazmak için elimek isterseniz siz bekletir.  
okunu yapan okuma kilitini bırakıcaz kader.

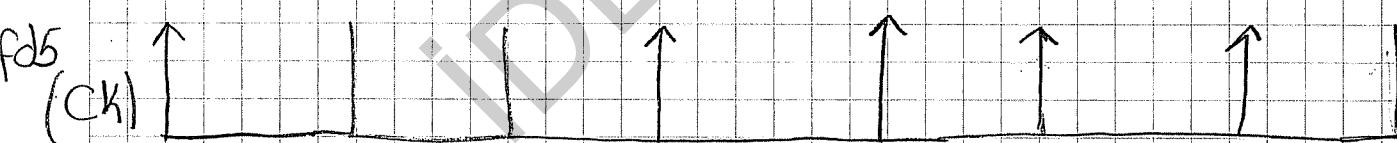
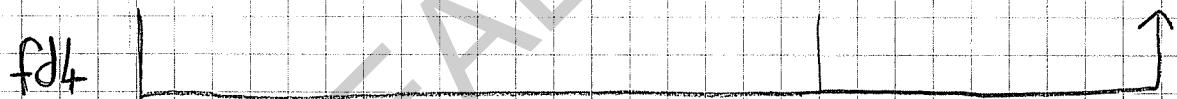
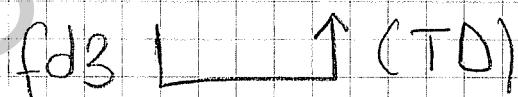
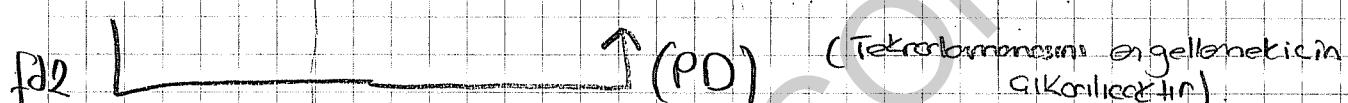
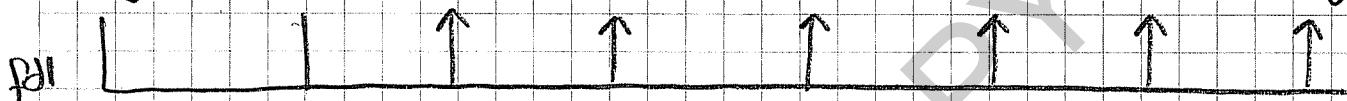
→ Bir lock tutulmayc devam eder Transaction bu birikine kader yada transaction  
commit oluncaya kader. Sonra birakılır.

→ exclusive lock close ettim modify ettim ben bu lock bıraklığım zaman  
diğerlerin bu veriyi görebilir.

- Bir staff evi gösterdiğinde o tarihe Kullanicak olduğu crabo belliidir.
- Bir crabo aynı gün birden çok staff Kullanicabilir.
- Belirli bir tarihte birden çok evi gösteren Staff
- Bir ev belili bir tarihte sadece bir hizmete gösterilir.

INF

| PropertyNo | iDate | iTime | pAddress | comments | staffNo | sName | carReg |
|------------|-------|-------|----------|----------|---------|-------|--------|
|------------|-------|-------|----------|----------|---------|-------|--------|



staffNo iDate → carReg  
bunucukatır.  
(TD)

2NF

Property (propertyNo, pAddress)

PropertyInspection(propertyNo, iDate, iTime, comments, staffNo, sName, carReg)

3NF

Staff (StaffNo, sName)

PropertyInspect (propertyNo, iDate, iTime, comments, staffNo, carReg)

Property (propertyNo, pAddress)

fd2 property No → pAddress PM

fd3 StaffNo → sName PM

fd1' property No, iDate → iTime, comments, staffNo, carReg PM

fd4 staffNo, iDate → carReg

fd5 carReg, iDate, iTime → propertyNo, comment, staffNo PM

fd6' staffNo, iDate, iTime → propertyNo, comments PM

BCNF 'degil olmasi iacin Determination criteri PM olmali gerek.

StaffCar (staffNo, iDate, carReg)

Inspection (propertyNo, iDate, iTime, comments, staffNo)

Staff Property Inspection

1NF

PropertyInspection

1NF

propertyInspect

3NF

Staff

StaffCar

inspection

Property

BCNF

T<sub>g</sub>

begin-transaction

read(bal<sub>x</sub>)

$$\text{bal}_x = \text{bal}_x + 100$$

write(bal<sub>x</sub>)

read(bal<sub>y</sub>)

$$\text{bal}_y = \text{bal}_y - 100$$

write(bal<sub>y</sub>)

commit

T<sub>10</sub>

begin-transaction

read(bal<sub>x</sub>)

$$\text{bal}_x = \text{bal}_x * 1.1$$

write(bal<sub>x</sub>)

read(bal<sub>y</sub>)

$$\text{bal}_y = \text{bal}_y * 1.1$$

write(bal<sub>y</sub>)

commit

S = (write-lock(T<sub>g</sub>, bal<sub>x</sub>), read(T<sub>g</sub>, bal<sub>x</sub>), write(T<sub>g</sub>, bal<sub>x</sub>), unlock(T<sub>g</sub>, bal<sub>x</sub>),  
write-lock(T<sub>10</sub>, bal<sub>x</sub>), read(T<sub>10</sub>, bal<sub>x</sub>), write(T<sub>10</sub>, bal<sub>x</sub>), unlock(T<sub>10</sub>, bal<sub>x</sub>),  
write(T<sub>10</sub>, bal<sub>y</sub>), read(T<sub>10</sub>, bal<sub>y</sub>), write(T<sub>10</sub>, bal<sub>y</sub>), unlock(T<sub>10</sub>, bal<sub>y</sub>),  
commit(T<sub>10</sub>), write-lock(T<sub>g</sub>, bal<sub>y</sub>), read(T<sub>g</sub>, bal<sub>y</sub>), write(T<sub>g</sub>, bal<sub>y</sub>),  
unlock(T<sub>g</sub>, bal<sub>y</sub>), commit(T<sub>g</sub>))

Scrij op schr. T<sub>g</sub> → T<sub>10</sub> also synchroneert banken lock released yalliseeldu

$$\text{bal}_x = 100 \quad \text{bal}_y = 400$$

## Bunun için denilen Two phase locking (2PL)

→ Hilit elde edileksen transaction bosinda yer alır. birbirinden hep birbirinle  
zorunlu olur.

< Elde et işlem ve blok > bu sırası dursak.

Nehanı bir obje üzerinde hilit birincia hep birbirinden gerek  
elde et istedigin kader elde et işlem yap elde et işlem yapanaq,  
birbirin son sureklili birbirin ve dahi hiliti olmıcaksin.

T<sub>1</sub>

begin-transaction

write-lock (balx)

wait

wait

wait

read(balx)

balx = balx + 100

write(balx)

commit/unlock (balx)

T<sub>2</sub>

begin-transaction

write-lock (balx)

read(balx)

balx = balx + 100

write(balx)

commit/unlock (balx)

(two phase locking), evet □  
○

T<sub>1</sub>

WL(A)

R(A)

W(A)  
UL(A)

commit

T<sub>2</sub>

A attempt to  
RL(A) fails

DL(A)  
R(A)  
UL(A)

commit

T<sub>1</sub>

R(A)  
W(A)

R(B)  
W(B)

commit

T<sub>2</sub>

R(A)  
W(A)  
R(B)  
W(B)

commit

T<sub>1</sub>

WL(A)  
W(A)

WL(B)  
R(B)  
W(B)  
UL(A)  
UL(B)

commit

T<sub>2</sub>

WL(A) fail

wait  
wait  
wait  
wait  
wait

WL(A)  
R(A)  
W(A)  
WL(B)  
R(B)  
W(B)  
UL(A)  
UL(B)

commit

DPL Seçileştiğimiz bir  
terifelene sunuyor. Yüllüler  
başka elbret şahı bir okurken  
hepsińi bırak...

Bunun deadlock olma riski var

T1

T2

WL(A)

WL(B)

R(A)

R(B)

WL(B) fail

WL(A) fail

Bunun oluşması için ilk olarak bütün kilitler boşa elde ettiğimizde bir rokirkası hepsi bir rok buna karşılık için.

( hold & wait ) kuralıdır...

IDEAL COPY