LAB-3

constraints
view
sequence
union – intersect - except

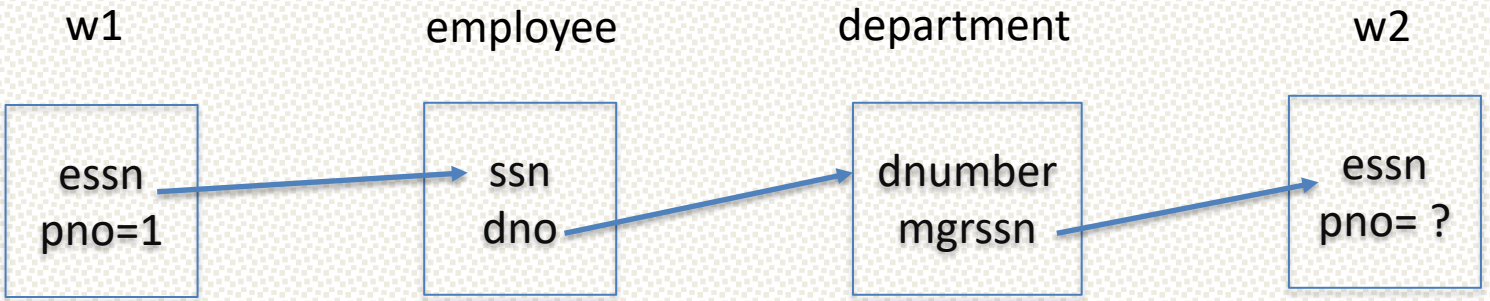
5 dk süre...

Aşağıdaki sorguyu ve çıktısını bulan chat'e yazsın ☺

> 1 no'lu projede çalışan kişinin departman yöneticisinin çalıştığı tüm projelerin numaralarını listeleyen sorguyu yazınız.

> 1 no'lu projede çalışan kişinin departman yöneticisinin çalıştığı tüm projelerin numaralarını listeleyen sorguyu yazınız.

```
select w2.pno
from works_on w1, works_on w2, employee, department
where w1.essn=ssn and dnumber=dno and mgrssn=w2.essn and w1.pno=1
```



Constraints – Kısıtlar

Verilerin doğruluk ve bütünlüklerinin devamı için tablolardaki sütunlar üzerinde çeşitli izinler-kısıtlar belirleriz.

Örneğin PROJECT tablosundaki ‘pnumber’ alanında yapacağımız bir değişiklik WORKS_ON tablosunu da etkileyeceği için kısıt koşulu gerektirir.

Örnek bir tablo oluşturarak anahtar kısıtları belirlemeyi görelim.

Şirket çalışanları kendi aralarında basketbol takımı kurmuşlardır.
"team" isimli tablo için primary key aşağıdaki gibi iki şekilde oluşturulabilir:

```
CREATE TABLE team (
     tnumber numeric(2),
     tname varchar(15)
     CONSTRAINT pk_team PRIMARY KEY(tnumber)
     );
```

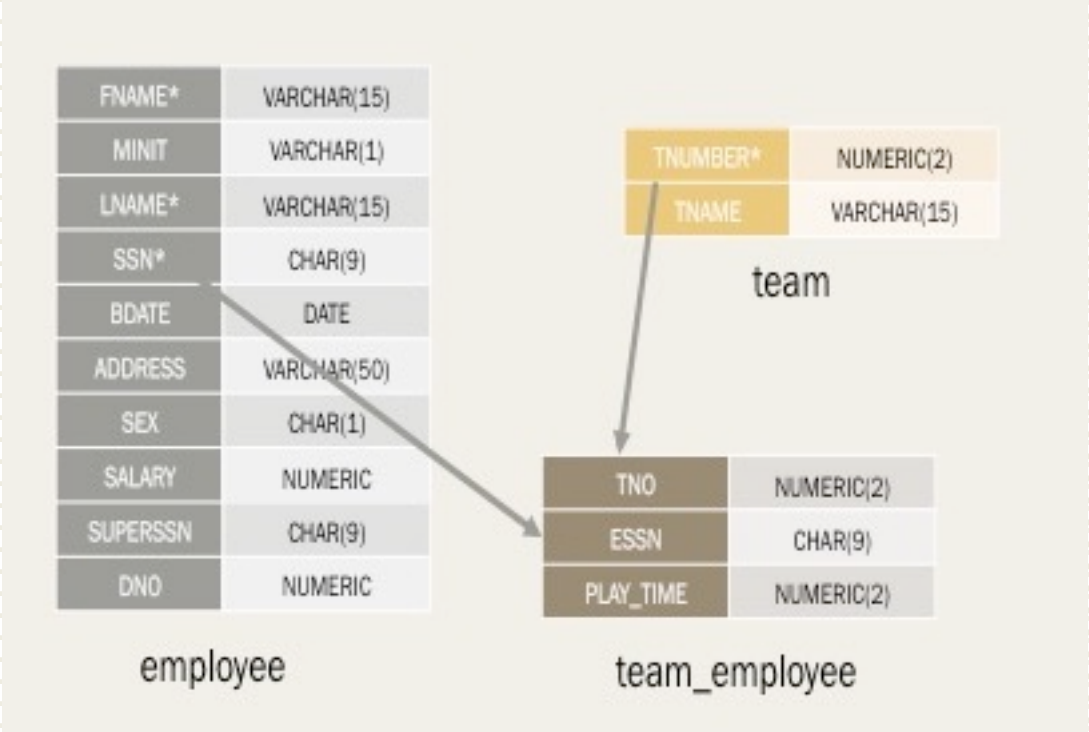
bu kısıtı silmek istersek: `ALTER TABLE team DROP CONSTRAINT pk_team ;`

```
CREATE TABLE team (
     tnumber numeric(2) PRIMARY KEY ,
     tname varchar(15)
     );
```

bu kısıtı silmek istersek: `ALTER TABLE team DROP CONSTRAINT team_pkey ;`

burada biz kısıta bir isim vermesek de primary key kısıtı için default isimlendirme: tabloadı_pkey

Oluşturmak istediğimiz team tablosu ve team_employee tablosu



"employee" ve "team" tabloları için ilişki tablosu olarak "team_employee" tablosu oluşturulmak isteniyor. Bu tabloya ait nitelikler şunlardır:

Sütunlar : `tno (numeric(2)), essn (char(9)), play_time (numeric(2))`

Kısıtlar:

1. **işçilerin tek bir takımda oynama zorunluluğu yoktur. tno ve essn birlikte primary key olmalıdır.**
2. **tno, "team" tablosundaki tnumber'ı referans almalıdır. "team" tablosundan bir satır silindiğinde, bu satıra ait tno'lu satırlar da silinmelidir.**
3. **essn, "employee" tablosundaki ssn'i referans almalıdır. "employee" tablosundan bir satır silindiğinde, bu satıra ait essn'li satırlar da silinmelidir.**
4. **bir çalışan, bir takımda 12 haftadan uzun süre oynayamaz.**

```
CREATE TABLE team_employee (
        tno numeric(2),
        essn char(9) ,
        play_time numeric(2) ,
        CONSTRAINT pk_team_employee
        PRIMARY KEY (tno, essn),

        CONSTRAINT fk_team FOREIGN KEY
        (tno) REFERENCES team (tnumber)
        ON DELETE CASCADE,

        CONSTRAINT fk_emp FOREIGN KEY
        (essn) REFERENCES employee (ssn)
        ON DELETE CASCADE,

        CONSTRAINT play_time_ck CHECK
        (play_time < 13)
        );
```

Foreign Key oluşturmada diğer yöntemler:

```
CREATE TABLE team_employee (
        tno numeric(2) REFERENCES team(number)
)
```

```
CREATE TABLE team_employee (
    tno numeric(2),
    FOREIGN KEY (tno) REFERENCES team(tnumber)
)
```

foreign key kısıtı için default isimlendirme: tabloadı_sütunadı_fkey
Silme yapmak istersek: `ALTER TABLE team_employee DROP CONSTRAINT team_employee_tno_fkey;`

ALTER TABLE ile Primary ve Foreign Key oluşturma:

```
ALTER TABLE team ADD CONSTRAINT pk_team PRIMARY KEY (tnumber)
```

silmek : `ALTER TABLE team DROP CONSTRAINT pk_team ;`

```
ALTER TABLE team_employee ADD CONSTRAINT te_fk FOREIGN KEY (tno)
REFERENCES team(tnumber)
```

silmek : `ALTER TABLE team_employee DROP CONSTRAINT te_fk;`

VIEW Oluşturma

Bir sorgu sonucu oluşan sanal tablo.
Sık kullanacağımız sorguyu bir kez yazıp, isimlendirme işlemi.

```
CREATE VIEW view_adı
AS
SELECT ...
FROM   ...
WHERE   ...
```

ÖRNEK: Maaşı 20000 ile 40000 arasında olan çalışanların isimlerini ve maaşlarını gösteren bir view yazınız.

```
CREATE VIEW maaslar
AS
SELECT fname, lname, salary
FROM   employee
WHERE  salary BETWEEN 20000 AND 40000
```

VIEW Çağırma: `SELECT * FROM maaslar`

NOT: employee tablosundan bu view'ı etkileyen bir bilgi sildiğimizde view otomatik olarak güncellenir.

SEQUENCE Oluşturma

Belli bir sırada sayısal değerler üretmemizi sağlar.

```
CREATE SEQUENCE sequence_adı
increment by ...
start with ...
maxvalue ...
nomaxvalue
minvalue ...
nominvalue
cycle  //son değere ulaşınca başa dön
no cycle
```

ÖRNEK: 9'dan başlayıp 99' a kadar birer bire artan bir sequence oluşturun.

```
CREATE SEQUENCE seq
MINVALUE 9
MAXVALUE 99
INCREMENT BY 1
```

Sequence hakkında bilgi almak için: `SELECT * FROM seq`

sequence_name name	last_value bigint	start_value bigint	increment_by bigint	max_value bigint	min_value bigint	cache_value bigint	log_cnt bigint	is_cycled boolean	is_called boolean
seq	9	9	1	99	9	1	0	false	false

Sequence'in sıradaki değeri:

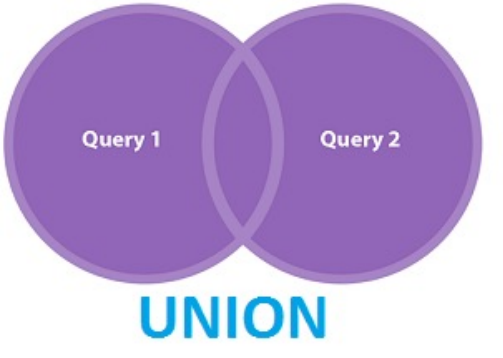
```
SELECT nextval('seq')
```

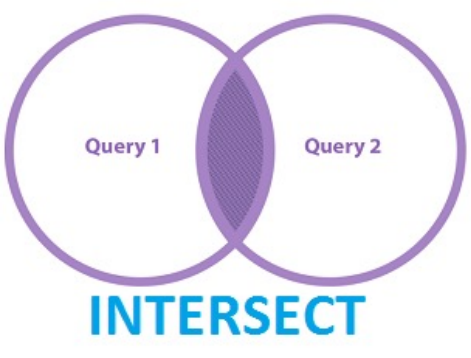
```
9,10,11,12 ...
```

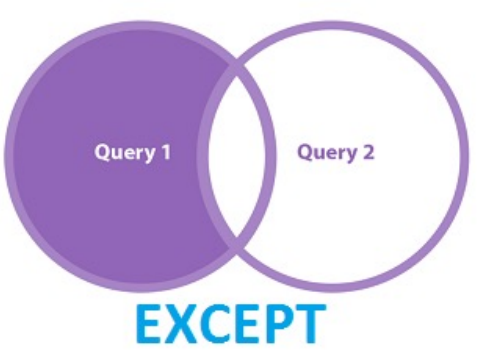
Sequence'deki değeri tabloya kayıt eklerken kullanma:

```
INSERT INTO employee(fname,ssn) VALUES ('John', nextval('seq'))
```

Sorgularda Kesişim – Birleşim – Fark İşlemleri







ÖRNEK-1: "OperatingSystems" isimli projede ve "Software" departmanında çalışanların ad, soyad bilgilerini bulunuz.

```
SELECT fname, lname
FROM project, works_on, employee
WHERE pname='OperatingSystems' and pnumber=pno and essn=ssn

INTERSECT

SELECT fname, lname
FROM department, employee
WHERE dname='Software' and dnumber=dno
```

ÖRNEK-2: **"OperatingSystems" isimli projede veya "Software" departmanında çalışanların ad, soyad bilgilerini bulunuz.**

```
SELECT fname, lname
FROM project, works_on, employee
WHERE pname='OperatingSystems' and pnumber=pno and essn=ssn

UNION

SELECT fname, lname
FROM department, employee
WHERE dname='Software' and dnumber=dno
```

ÖRNEK-3: **"OperatingSystems" isimli projede çalışıp, "Software" departmanında çalışmayanların ad, soyad bilgilerini bulunuz.**

```
SELECT fname, lname
FROM project, works_on, employee
WHERE pname='OperatingSystems' and pnumber=pno and essn=ssn

EXCEPT

SELECT fname, lname
FROM department, employee
WHERE dname='Software' and dnumber=dno
```

ÖRNEK-4: **Hiçbir departmanın veya hiçbir çalışanın yöneticisi olmayan çalışanların isimlerini bulunuz.**

```
SELECT fname, lname
FROM employee e
WHERE not exists (select null from department where mgrssn=e.ssn) and
      not exists (select null from employee e2 where e.ssn=e2.superssn)
```

ÖRNEK-5: İsmi 'John' olan işçilerin çalıştıkları departmanların isimlerini IN kullanarak bulunuz.

```
SELECT dname
FROM department
WHERE dnumber IN (SELECT dno FROM employee WHERE fname='John')
```

ÖRNEK-6: 'Sales' departmanında kaç kişinin çalıştığını, en düşük, en yüksek, ortalama ve toplam maaşı bulunuz.

```
SELECT  count(*),
        sum(salary),
        max(salary),
        min(salary),
        avg(salary)
FROM department, employee
WHERE dname='Sales' AND dnumber=dno
```

count bigint	sum bigint	max integer	min integer	avg numeric
14	571500	96000	29000	40821.428571428571

Yararlanabileceğiniz Kaynaklar:

https://www.postgresqltutorial.com/
https://www.w3schools.com/sql/

SON