

SQL

Structured Query Language
Yapılandırılmış Sorgu Dili

DATA-DATABASE

```
public class facebook {  
  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        System.out.println("Enter your name");  
        String name = scan.nextLine();  
  
        System.out.println("Enter your surname");  
        String surname = scan.nextLine();  
  
        System.out.println("Enter your surname");  
        String email = scan.nextLine();  
  
        System.out.println("Enter your password");  
        String password = scan.nextLine();  
  
        scan.close();  
    }  
}
```

Kaydol

Hızlı ve kolaydır.

Doğum Tarihi ?

1

Eki

2020

Cinsiyet ?

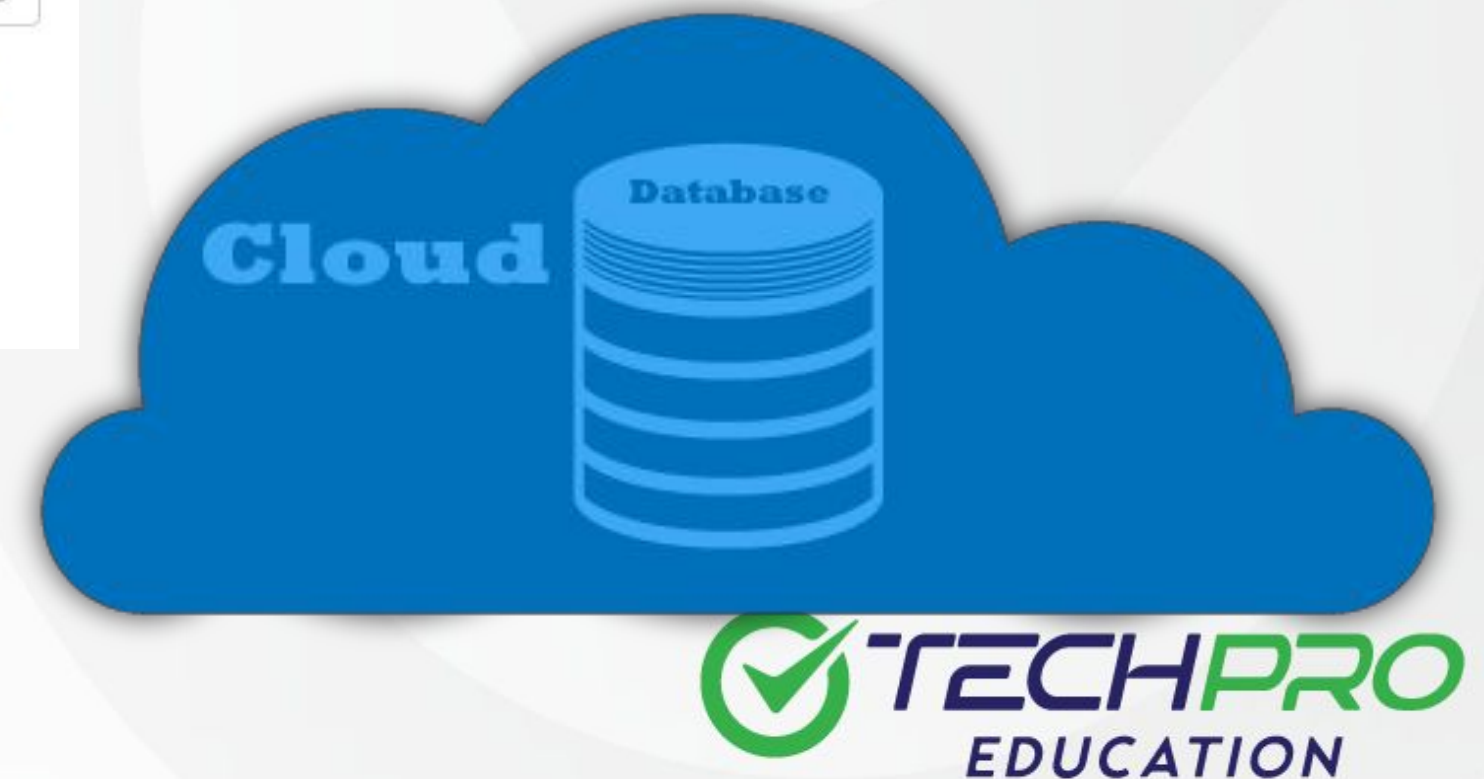
Kadın

Erkek

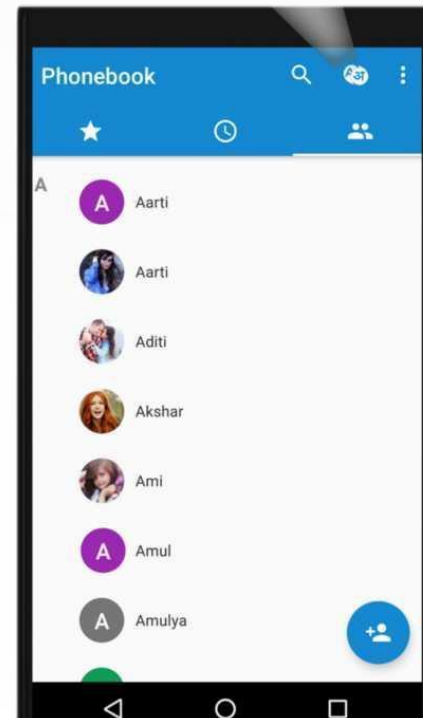
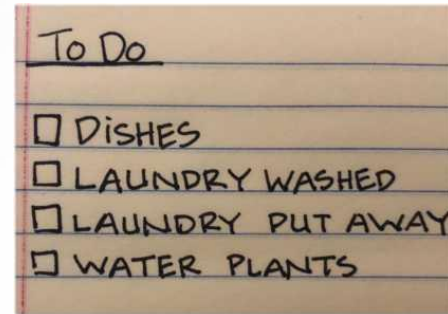
Özel

Kaydol düğmesine tıklayarak, Koşullarımızı, Veri İlkemizi ve Çerezler İlkemizi kabul etmiş olursun. Bizden SMS Bildirimleri alabilir ve bu bildirimleri istediğin zaman durdurabilirsin.

Kaydol



Database Management System (DBMS)



Veritabanı genellikle elektronik olarak bir bilgisayar sisteminde depolanan yapılandırılmış(**Structured**) bilgi veya veriden oluşan düzenli bir koleksiyondur.

Veritabanı genellikle bir Veritabanı Yönetim Sistemi **DBMS** (**D**ata**B**ase**M**anagement**S**ystem) ile kontrol edilir.

Çoğu veritabanında veri yazma ve sorgulama için yapılandırılmış sorgu dili SQL (**S**tructured **Q**uery **L**anguage) kullanılır.

DATABASE'IN FAYDALARI

- 1) Yüksek miktarda bilgi depolanabilir
- 2) Oluşturma, Okuma,Değiştirme ve Silme kolaylığı
Create,Read,Update,Delete (CRUD)
- 3) Girişin kolay ve kontrollü olması
- 4) Dataya ulaşım kolaylığı
- 5) Güvenlik

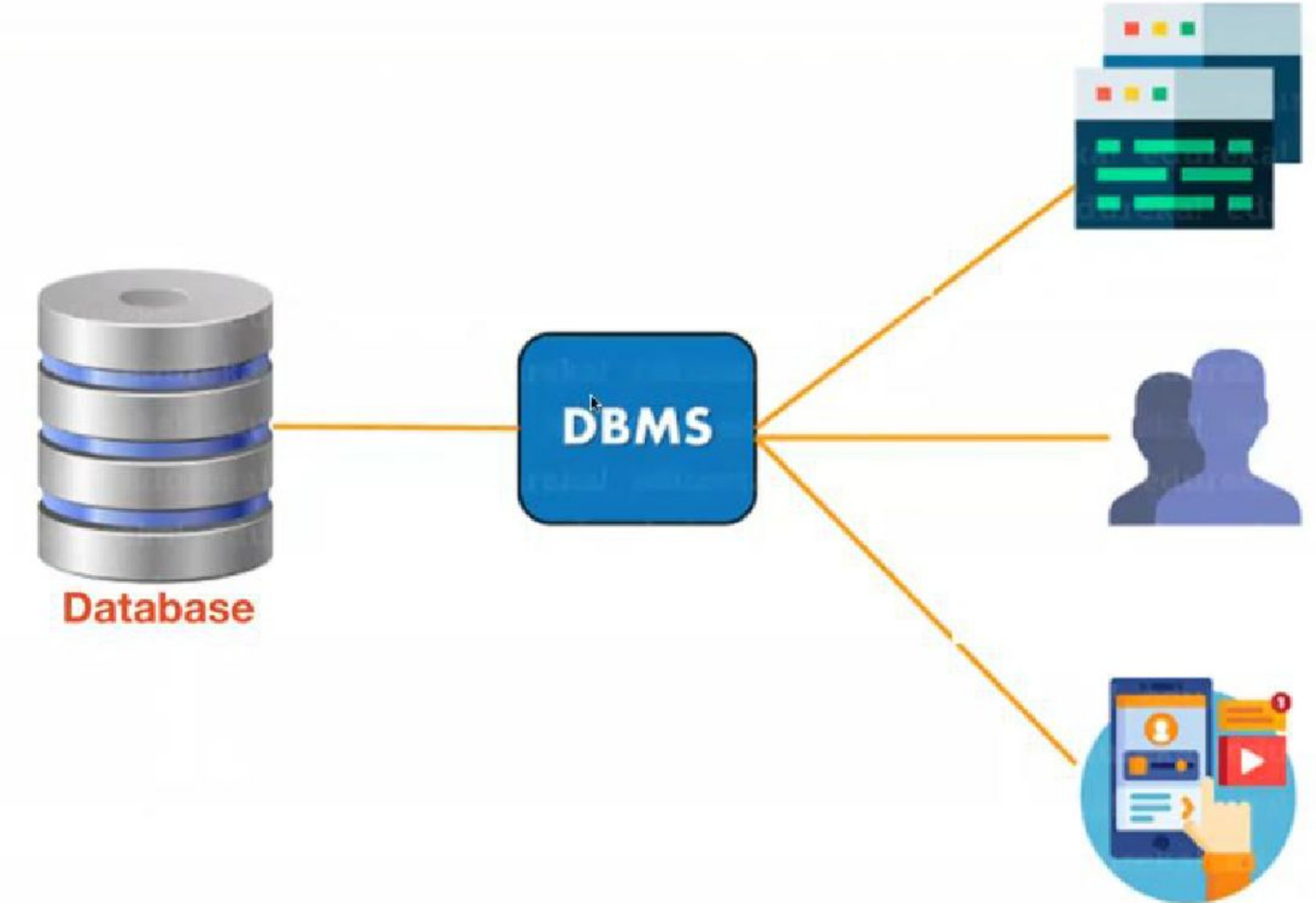
ono	adi	soyadi	dyeri	bid
1	Ali	Turan	İstanbul	1
2	Ahmet	Büyük	Ankara	1
3	Leyla	Şahin	İzmir	1
4	Can	Türkoğlu	Manisa	2
5	Aziz	Keskin	İstanbul	2
6	Talat	Şanlı	İzmir	3
7	Kamuran	Kece	Adana	3
8	Turgut	Cemal	Bursa	4



Database Management System (DBMS)

- Veritabanlarını yönetmek, kullanmak, geliştirmek ve bakımını yapmak için kullanılan yazılımlara denir.

- ❑ Database'e erişimi düzenler
- ❑ **C**reate, **R**ead, **U**ppdate, **D**eleate işlemleri düzenler
- ❑ Data güvenliğini sağlar
- ❑ Sorgular oluşturur ve sorgular iletilir,
- ❑ Raporlar oluşturur ve raporları işletir,
- ❑ Uygulamayı kontrol eder
- ❑ Diğer uygulamalarla (Application) iletişimi sağlar.



Database Management System (DBMS)



SQL Server : Microsoft tarafından geliştirilmiştir

Negatif: Pahalı – Kurumsal Kullanıcılar için binlerce dolar ödenmesi gereklidir

Pozitif : Zengin bir **user interface**'e sahip ve çok büyük data'ların kullanılmasında sorunsuz çalışır.

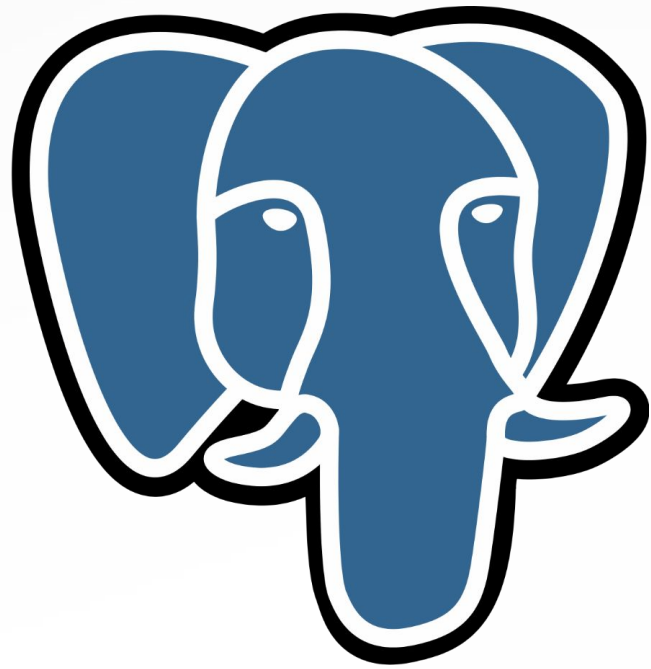


MySQL Server : İsveçli MySQL firması tarafından geliştirildi.
2010'da Oracle satın aldı

Negatif: Es zamanlı çok fazla işlem girildiğinde çalışmayı durdurabilir.

Pozitif: Açık kaynak. Online destek ve ücretsiz çok fazla doküman var

Database Management System (DBMS)



PostgreSQL Server : Created by a computer science professor Michael Stonebraker.

Pozitif:Yeni nesil olarak ortaya cikti. Kişiselleştirme mumkundur, zor gorevler icin ideal olabilir.

ORACLE®
DATABASE



PL/SQL Oracle database sunuculari icinde depolanir

PL/SQL SQL komutlarini özellikle karsilamak uzere dizayn edilmistir.

Pozitif: PL/SQL yüksek güvenlik seviyesi saglar ve Object-Oriented Programming ile uyumludur

TABLULAR (TABLES)

Headers====>

Row (Record)====>

Row (Record)====>

Row (Record)====>

Row (Record)====>

contactID	name	company	email
1	Bill Gates	Microsoft	bill@XBoxOneRocks.com
2	Steve Jobs	Apple	steve@rememberNewton.com
3	Linus Torvalds	Linux Foundation	linus@gnuWho.org
4	Andy Harris	Wiley Press	andy@aharrisBooks.net

Column (Field) ^^^^>

Column (Field) ^^^^>

Column (Field) ^^^^>

Column (Field) ^^^^>

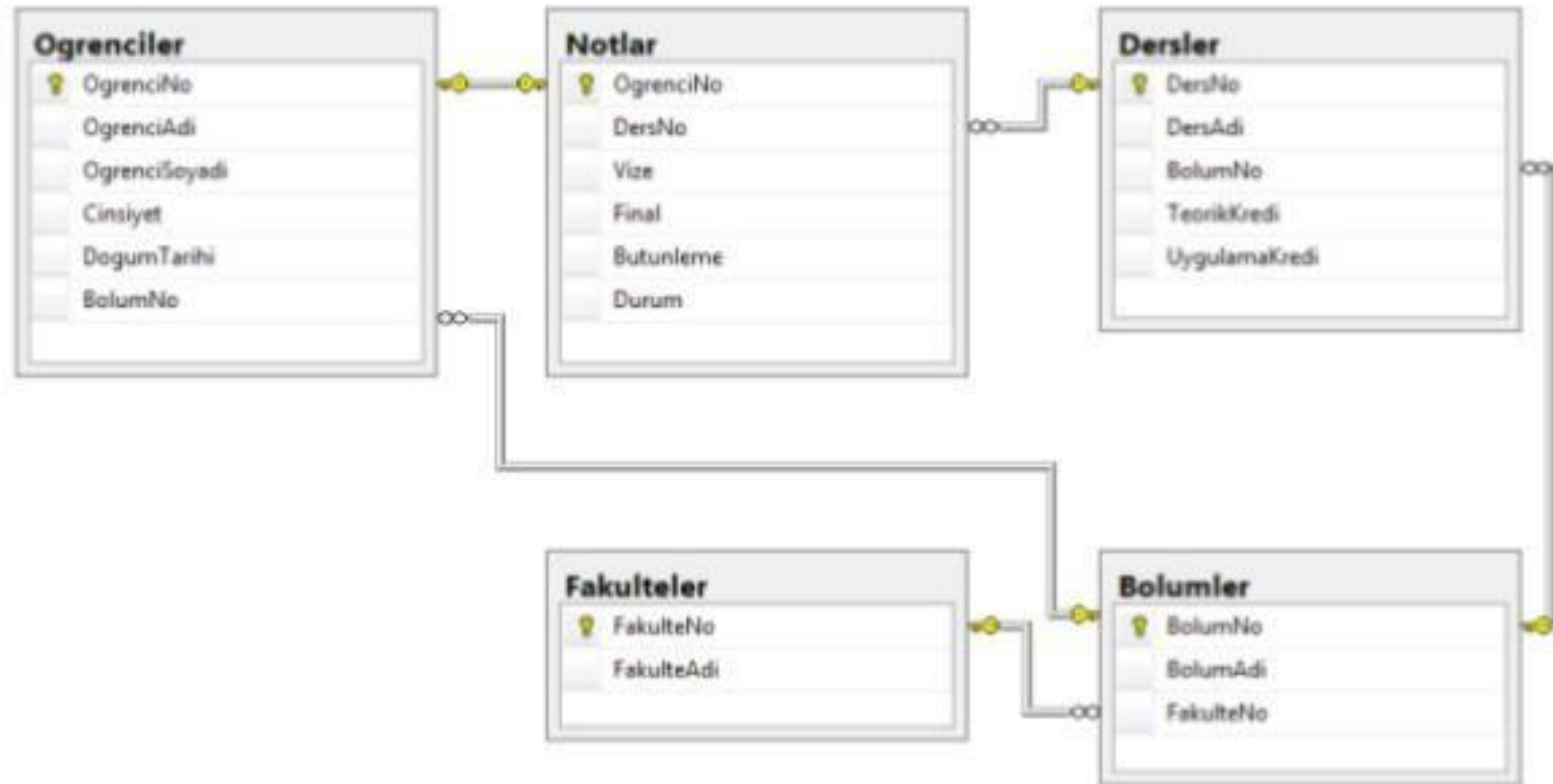
RELATIOANAL DATABASES (ILISKILI TABLOLAR)

Student						
id	isim	soyisim	cinsiyet	ders	kredi	derskodu
1	Ali	Eren	E	Matematik	6	MAT101
2	Veli	Arslan	E	Türkçe	5	TURK102
3	Ayşe	Koç	K	Fizik	3	FIZ103
4	Fatma	Tek	K	Matematik	6	MAT101
5	Betül	Şen	K	Kimya	2	KIM104
6	Nisa	Can	K	Türkçe	5	TURK102

Student				
id	isim	soyisim	cinsiyet	ders_id
1	Ali	Eren	E	101
2	Veli	Arslan	E	102
3	Ayşe	Koç	K	103
4	Fatma	Tek	K	101
5	Betül	Şen	K	104
6	Nisa	Can	K	102

Course			
ders_id	ders	kredi	derskodu
101	Matematik	6	MAT101
102	Türkçe	5	TURK102
103	Fizik	3	FIZ103
104	Kimya	6	KIM104

RELATIONANAL DATABASES (ILISKILI TABLOLAR)



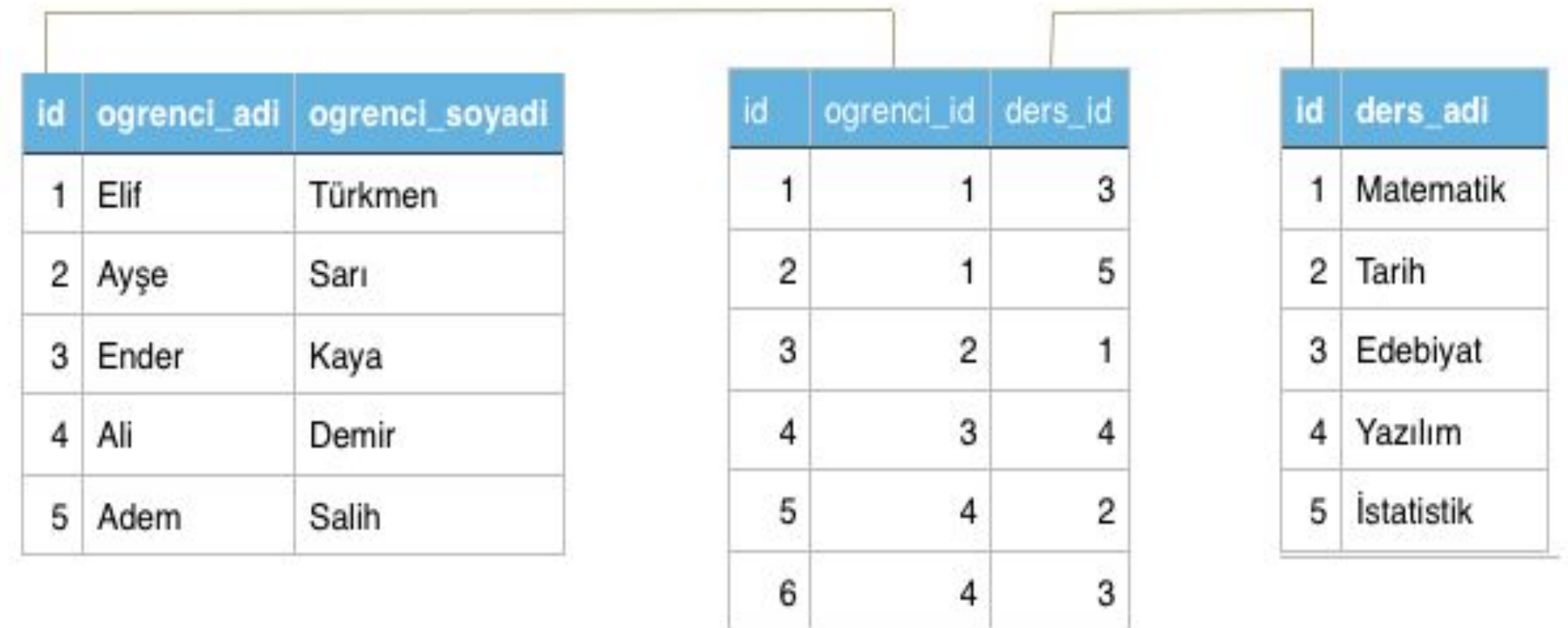
RELATIONANAL DATABASES (ILISKILI TABLOLAR)

❑ **SQL tablolar** dataları ilişkili tablolarda depolar.

❑ Tablolar arası ilişkiler net olmalıdır.

❑ Tablolar arası geçiş kolay olmalıdır

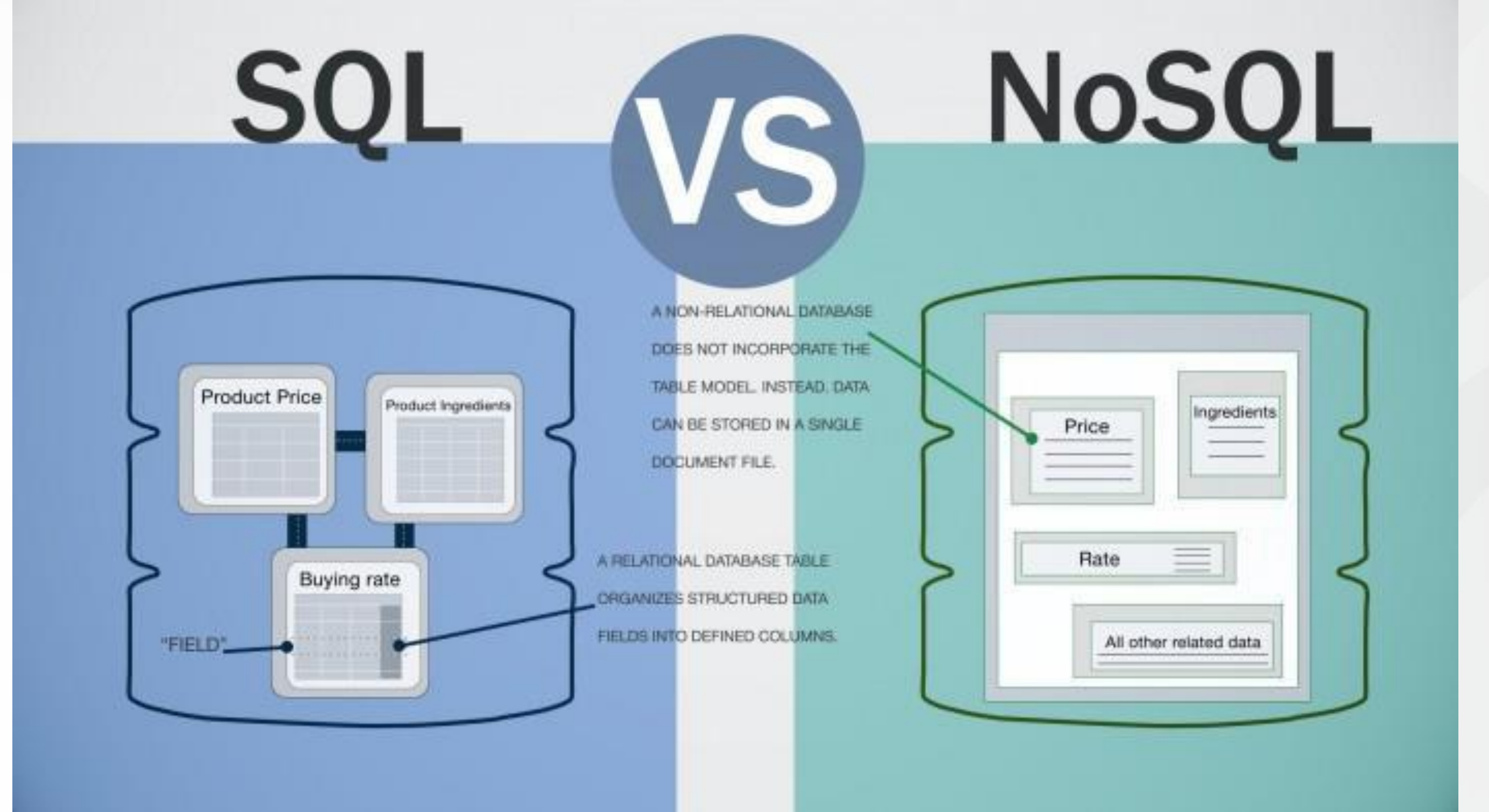
❑ Tabloların ve ilişkilerin bütününe
SCHEMA denir



❑ Relational Databases, **SQL Databases** (Structured Query Language) olarak da adlandırılır

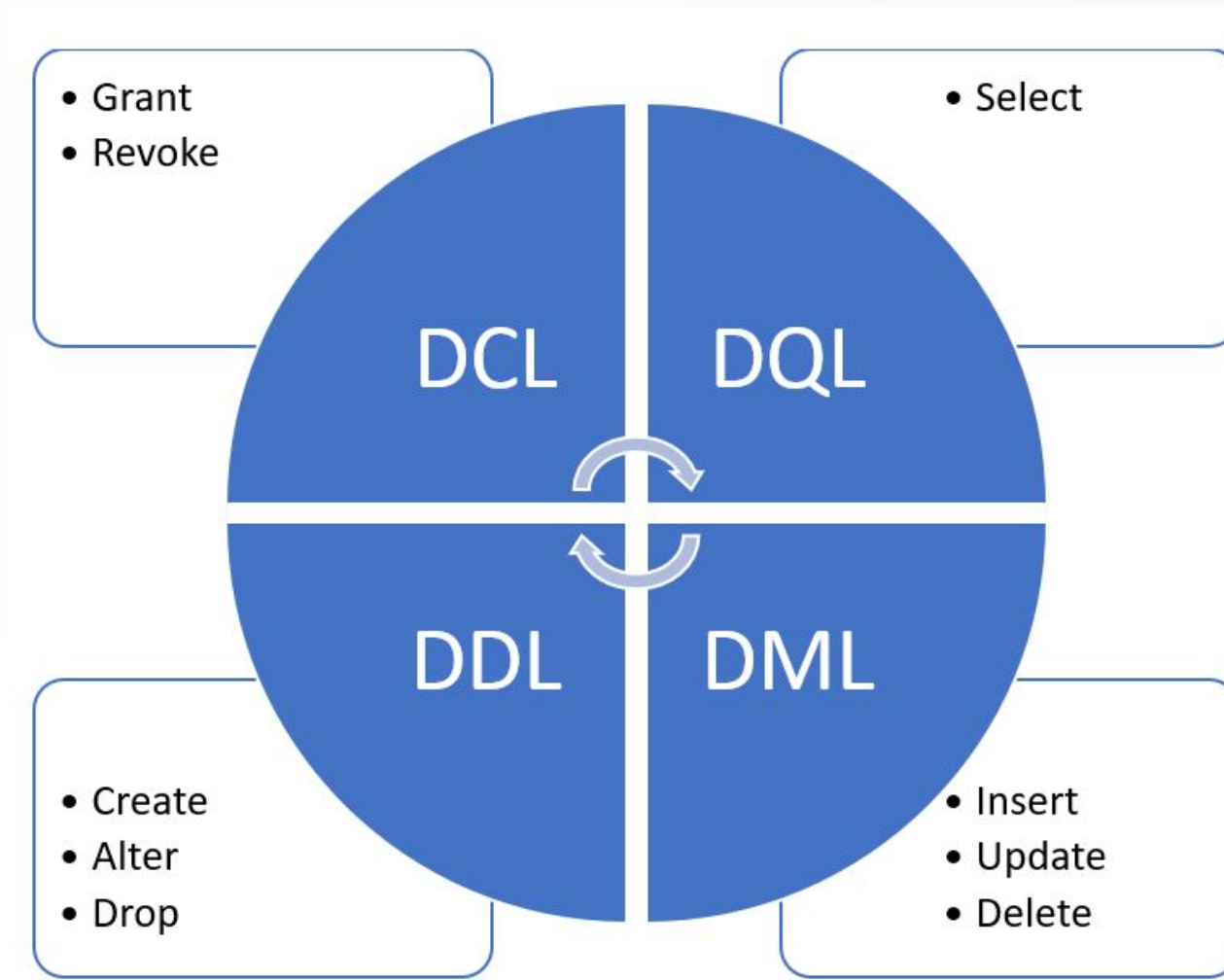
NON-RELATIOANAL DATABASES (non-SQL Database)

SQL veritabanı verilerle ilgilenirken Yapısal Sorgu Dili kullanır. Veri yapısını belirlemek için önceden tanımlanmış şemalar gerektirir.



NoSQL veritabanı verilerle çalışırken Yapılandırılmamış Sorgu Dili kullanır.

SQL Komutlari



SQL Komutları

SQL komutları 4 ana gruba ayrılır:

1. Veri Sorgulama Dili (Data Query Language - DQL)

DQL içindeki SELECT komutu ile veri tabanında yer alan **mevcut kayıtların** bir kısmını veya tamamını tanımlanan koşullara bağlı olarak alır.

SELECT : Veritabanındaki verileri alır.

2. Veri Kullanma Dili (Data Manipulation Language - DML)

DML komutları ile veri tabanlarında bulunan verilere işlem yapılır. DML ile veritabanına yeni kayıt ekleme, mevcut kayıtları güncelleme ve silme işlemleri yapılır.

INSERT : Veritabanına yeni veri ekler.

UPDATE : Veritabanındaki verileri günceller.

DELETE : Veritabanındaki verileri siler.

SQL Komutları

3. Veri Tanımlama Dili (Data Definition Language - DDL)

DDL komutları ile veritabanı ve tabloları oluşturma, değiştirme ve silme işlemleri yapılır:

CREATE : Bir veritabanı veya veritabanı içinde tablo oluşturur.

ALTER : Bir veritabanı veya veritabanı içindeki tabloyu günceller.

DROP : Bir veritabanını veya veritabanı içindeki tabloyu siler.

4. Veri Kontrol Dili (Data Control Language - DCL)

DCL komutları ile kullanıcılara veritabanı ve tablolar için yetki verilir veya geri alınır:

GRANT : Bir kullanıcıya yetki vermek için kullanılır.

REVOKE : Bir kullanıcıya verilen yetkiyi geri almak için kullanılır.

SQL Data Types

String Data Types

<i>Data Type</i>	<i>Aciklama</i>
char(size)	Maximum boyutu 2000 byte olur. 1 karakter 1 byte kullanir. “ size ” database’e eklenecek karakter sayisidir. “char” data tipinden uzunlugu sabit datalari depolar. (Strings) “char” SSN, zip kodu gibi uzunlugu sabit datalari depolamak icin idealdir.
nchar(size)	Maximum boyutu 2000 byte olur. 1 karakter 2 byte kullanir “ size ” depolanacak karakter sayisi ’dir. “ nchar ” Unicode datalari depolamak icin kullanilir. Genellikle farkli dillerdeki karakterler icin kullanilir Uzunlugu belli Stringler icin kullanilir.
varchar(size)	Maximum boyutu 4000 byte olur. 1 karakter 1 byte kullanir. “ size ” database’e eklenecek max. karakter sayisidir. Degisken uzunluktaki stringler icin kullanilir.
nvarchar(size)	Maximum boyutu 8000 byte olur. 1 karakter 2 byte kullanir “ size ” depolanacak karakter sayisi ’dir. Degisken uzunluktaki stringlerin Unicode degerleri icin kullanilir.

SQL Data Types

Numeric Data Types

Data Type	Aciklama
numeric(p, s) (yüksek doğruluk)	<p>“Precision” (p) sayidaki rakam sayisidir “Scale” (s) virgulden sonra kac rakam oldugunu belirler</p> <p>1) “numeric(5, 2)” virgulden once 3,virgulden sonra 2 rakam olan sayi ==> 123,45</p> <p>2) “numeric(7)” ondalik kısmi olmayan 7 basamakli sayi demektir ==> 12345,67’i kabul eder ama 12345 olarak depolar</p> <p>Note: “numeric(7)” ve “numeric(7, 0)” aynı şeydir</p>
real (yeterli doğruluk)	Gerçek sayılar(ondalıklı) için kullanılır.Hafızada kapladığı alan: 4 byte

SQL Data Types

Numeric DataTypes

<i>Data Type</i>	<i>Aciklama</i>
SMALLINT	-32.768 ile 32.767 arasında deęer alır. Hafızada kapladığı alan: 2 byte
INTEGER	Alabileceęi deęerler –2147483648 ile 2147483647 arasındadır. Hafızada kapladığı alan: 4 byte.
BIGINT	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arasında deęer alır. “Boyut” ile alabileceęi sınırı belirtebiliriz. Hafızada kapladığı alan: 8 byte.
SERIAL	0’dan başlar. Tabloya bir veri girildiğinde bir artırılır, 1 olur. Dięer databaseselerde bulunan Auto Increment özellięinin karřılıęıdır. Serial ifadeler eřsiz olacak diye bir kural yoktur.

SQL Data Types

Date Data Types

<i>Data Type</i>	<i>Aciklama</i>
DATE	<p>“DATE” data tipi tarih ve zamani depolamak icin kullanilir. Saniyenin virgüllü kısmını da alır.</p> <p>“DATE” yil, ay, gun, saat, dakika, ve saniye içerir.</p> <p>Standart “Date Format” , “YYYY - MM - dd”.Örneğin 1982 - 07- 01</p>

SQL Data Types

BLOB Data Types

<i>Data Type</i>	<i>Aciklama</i>
BYTEA	“ BYTEA ” resim,video,ses gibi datalari binary formatina cevirerek depolar.

Database Oluşturma/Silme-DDL

Database Oluşturma:

Syntax: CREATE DATABASE db_name;

Database Silme:

Syntax: DROP DATABASE db_name;

Tablo Oluşturma/Silme-DDL

1-Tablo Oluşturma:

Syntax: CREATE TABLE (IF NOT EXISTS)

```
table_name(  
column_name_1 data_type,  
column_name_1 data_type,  
column_name_1 data_type,  
.....  
column_name_n data_type,  
);
```

3-Tablo Silme:

Syntax: DROP TABLE table_name;

2-Var olan tablodan yeni tablo oluşturma:

Syntax: CREATE TABLE table_name
AS SELECT columns
FROM old_table

Tabloya Data Ekleme

□ **INSERT INTO** komutu, tabloya bir veya birden fazla kayıt eklemek için kullanılır.

Tüm Field'lara data eklemek için:

Syntax: INSERT INTO table_name **VALUES** (value1,value2,...);

Bazı Field'lara data eklemek için:

Syntax:INSERT INTO table_name (column1,column2,...) **VALUES** (value1,value2,...);

Tablodaki Tüm Kayıtları Görüntüleme-DQL

- **SELECT** komutu, tablodaki kayıtları sorgulamak(görüntülemek) için kullanılır.

Tablodaki Tüm Kayıtları **Tüm** Field'leri ile **Çağırma**

Syntax: SELECT * FROM table_name;

Tablodaki Tüm Kayıtları **Belirli** Field'leri ile **Çağırma**

Syntax: SELECT col1,col2,... FROM table_name;

CONSTRAINT

UNIQUE - Bir sütundaki tüm değerlerin BENZERSİZ/TEKRARSIZ yani tek olmasını sağlar.

NOT NULL - Bir Sütunun NULL içermemesini sağlar.

NOT NULL kısıtlaması için constraint ismi tanımlanmaz. Bu kısıtlama veri türünden hemen sonra yerleştirilir

PRIMARY KEY - Bir sütünün NULL içermemesini ve sütundaki verilerin BENZERSİZ olmasını sağlar. (NOT NULL ve UNIQUE)

FOREIGN KEY - Başka bir tablodaki Primary Key'i referans göstermek için kullanılır. Böylelikle, tablolar arasında ilişki kurulmuş olur.

CHECK - Bir sütuna yerleştirilebilecek değer aralığını sınırlamak için kullanılır .

CONSTRAINT:unique ve not null

- Bir sütuna tekrarlı verilerin eklenememesi için tablo ve sütunları tanımlanırken **UNIQUE** kısıtlaması eklenir.

Syntax: CREATE TABLE table_name(
col1 data_type UNIQUE,
col2
)

- Bir sütuna NULL değerlerin eklenememesi için tablo ve sütunları tanımlanırken **NOT NULL** kısıtlaması eklenir.

Syntax: CREATE TABLE table_name(
col1 data_type NOT NULL,
col2
)

CONSTRAINT:primary key

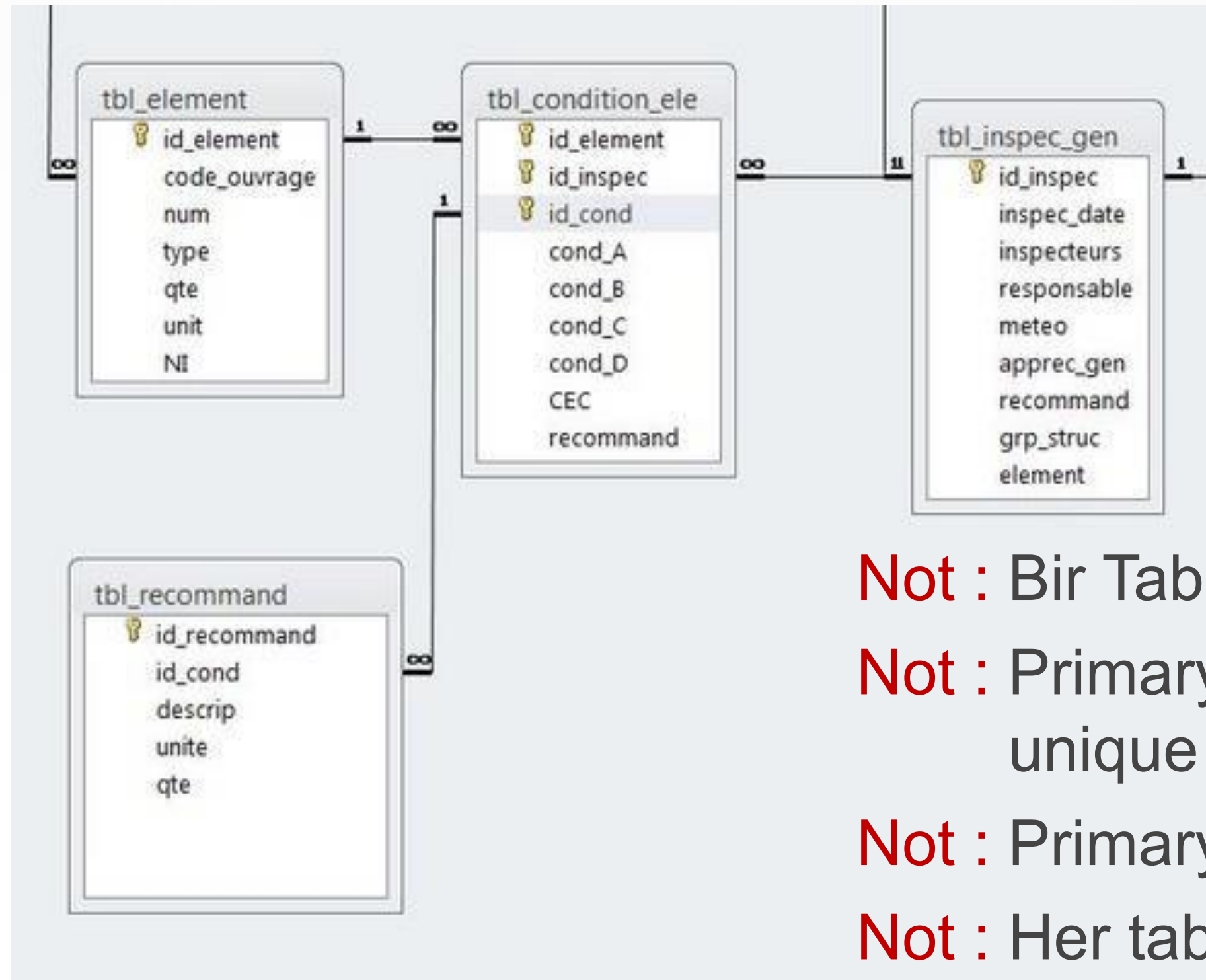
Primary Key (birincil anahtar), bir veri tablosunda yer alan her satır için bir vekil / tanımlayıcı (identify) görevi görür, kısıtlamadır (constraint) ve eşsizdir (Unique).

Primary Keys



<u>StudentId</u>	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

CONSTRAINT: primary key*



Primary Key ilişkisel(relational) veri tabanlarında ilişkisel bağlantıları kurmak için kullanılır. Başka bir tabloyla ilişkilendirilecek olan tablo mutlaka Primary Key içermelidir.

Not : Bir Tabloda 1 tane Primary Key olur.

Not : Primary Key benzersiz (Unique) olmalıdır ama her unique data Primary Key değildir

Not : Primary key her türlü datayı içerebilir. Sayı, String..

Not : Her tabloda Primary Key olması zorunlu değildir

CONSTRAINT: primary key

StudentID	FirstName	LastName
10 ←	John	Walker
11	Tom	Hanks
12	Kevin	Star
13 ←	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18 ←	John	Walker
19	Pamela	Star
20 ←	Carl	Wall

Genel olarak kayıt eklenmeden önce üretilen sıra numarası gibi sayısal değerlere **Surrogate Key** denir.

Primary Key, dış dünyadaki gerçek verileri temsil ediyorsa, örneğin; TC kimlik numarası, bir kitabın ISBN numarası, bir ürünün ismi, email hesabi gibi buna **Natural key** denir

Email	FirstName	LastName
JWalker@gmail.com	John	Walker
THanks@gmail.com	Tom	Hanks
KStar@gmail.com	Kevin	Star
CWall@gmail.com	Carl	Wall
AApazniak@gmail.com	Andrei	Apazniak
MHigh@gmail.com	Mark	High
CStar@gmail.com	Clara	Star
JOcean@gmail.com	John	Ocean
JWalker01@gmail.com	John	Walker
PStar@gmail.com	Pamela	Star
CWall01@gmail.com	Carl	Wall

CONSTRAINT: foreign key

Foreign Key iki tablo arasında ilişki kurmak için kullanılır

Foreign Key başka bir tablodaki Primary Key ile ilişkilendirilmiş olmalıdır.

Foreign Key

StudentID	FirstName	LastName	CourseID
10	John	Walker	200
11	Tom	Hanks	400
12	Kevin	Star	400
13	Carl	Wall	200
14	Andrei	Apazniak	300
15	Mark	High	400
16	Clara	Star	100
17	John	Ocean	100
18	John	Walker	200
19	Pamela	Star	300
20	Carl	Wall	NULL

Child Table

Primary Key

CourseID	CourseName	CourseCredit	CourseFee
100	Biology	3	1200
200	Math	3	1200
300	English	2	600
400	Selective	1	200

Parent Table

Bir tabloda birden fazla Foreign Key olabilir

Foreign Key **NULL** değeri kabul eder

Foreign Key olarak tanımlanan field'da **tekrarlar** olabilir

Foreign Key, değerleri farklı bir tablodaki Primary Key ile eşleşen bir sütun veya sütunların birleşimidir.

CONSTRAINT:foreign key-primary key

Note: Foreign key Tablonun kendi icinde bir relation olusturabilir.

Emp_ID	first_name	last_name	birth_date	Gender	salary	Job_ID	Manager_ID
100	Jan	Levinson	1961-05-11	F	110,000	1	NULL
101	Michael	Scott	1964-03-15	M	75,000	2	100
102	Josh	Porter	1969-09-05	M	78,000	3	100
103	Angela	Martin	1971-06-25	F	63,000	2	101
104	Andy	Bernard	1973-07-22	M	65,000	3	101

Job_ID	Job_Name
2	SDET
3	Manual Tester
1	QE Lead

- 1) Michael Scott'un manager'i kimdir?
- 2) Angela Martin'in Job_Name'i nedir ?
- 3) Manual Tester'lerin ortalama Salary'si ne kadardir ?
- 4) En yuksek Salary'yi alan kisinin Job_Name'i nedir?

CONSTRAINT:composite key

Job_ID	Job_Name
2	SDET
3	Manuel Tester
1	QA Led
Job Table	

Recruiter	NumberOfClient
Mark Eye	121
John Ted	283
Angela Star	301
Cory Al	67
Recruiter Table	

Job_Id	Name	Company
2	Mark Eye	RCG
3	John Ted	RCG
1	Mark Eye	Signature
1	John Ted	Info Log
1	Cory Al	Info Log
2	Angela Star	Signature
Company Table		

Composite Key birden fazla field(kolon)'in kombinasyonu ile oluşturulur.

Tek basına bir kolon **Primary Key** olma özelliklerini taşıyamıyorsa, bu özellikleri elde etmek için birden fazla kolon birleştirilerek Primary oluşturulur

“UNIQUE KEY” & “PRIMARY KEY”

“UNIQUE KEY” ve “PRIMARY KEY” arasındaki farklar

Primary Key

Bir Tabloda 1 tane olur

NULL değer Kabul etmez

Unique Key

Bir tabloda birden fazla olabilir

NULL değeri Kabul eder

“UNIQUE KEY” ve “PRIMARY KEY” ortak özellikleri

Duplication(Cift Kullanım)’a izin vermez

Related Tablolarla Çalışma : One to One Relation

StudentID	FirstName	LastName
10	John	Walker
11	Tom	Hanks
12	Kevin	Star
13	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18	John	Walker
19	Pamela	Star
20	Carl	Wall

StudentID	Street	ZipCode	City	State
10	1234 W 23th Street	33018	Hialeah	Florida
11	1235 N 3th Street	22145	Austwell	Texas
12	1236 SE 12th Street	54234	Orange	California
13	1237 N 5th Street	33018	Hialeah	Florida
14	1238 SW 53th Street	33026	Miami	Florida
15	1239 S 123th Street	22314	Avery	Texas
16	1240 N 1 st Street	12345	Arlington	Virginia
17	1241 NW 2nd Street	65432	Pittsburgh	Pennsylvania
18	1242 W 5th Street	22133	Baytown	Texas
19	1243 SE 55th Street	74352	Beachwood	Ohio
20	1244 SW 17th Street	22314	Avery	Texas

- 1) Tom Hanks'in adresi nedir?
- 2) Kevin Star'in eyaleti nedir?
- 3) ID'si 17 olan kisinin sehri nedir?

Related Tablolarla Çalışma : One to Many Relation

CourseID	CourseName	CourseCredit	CourseFee	InstructorID
100	Biology	3	1200	1
200	Math	3	1200	2
300	English	2	600	3
400	Selective	1	200	1

- 1) Biology dersi alan ogrenciler kimler?
- 2) Selective ders alan ogrencilerin isimleri ?
- 3) CourseFee 600 olan ogrencilerin isimleri ?

StudentID	FirstName	LastName	CourseID
10	John	Walker	200
11	Tom	Hanks	400
12	Kevin	Star	400
13	Carl	Wall	200
14	Andrei	Apazniak	300
15	Mark	High	400
16	Clara	Star	100
17	John	Ocean	100
18	John	Walker	200
19	Pamela	Star	300
20	Carl	Wall	400

Related Tablolarla Çalışma : Many to Many Relation

StudentID	FirstName	LastName
10	John	Walker
11	Tom	Hanks
12	Kevin	Star
13	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18	John	Walker
19	Pamela	Star
20	Carl	Wall

StudentID	InstructorID
12	1
11	2
12	2
13	1
15	1
17	3
15	4

InstructorID	FirstName	LastName	Phone	Department
1	Mark	Adam	1234567891	Science
2	Eve	Sky	1239876543	Engineering
3	Leo	Ocean	1237845691	Language
4	Andy	Mark	1232134567	Health

- 1) Öğretmeni Mark Adam olan öğrencilerin isimleri nedir?
- 2) Kevin Star'ın öğretmenlerinin isimleri nedir?

Bir Tabloya “Primary Key” Nasıl Eklenir?

- 1) Primary Key bir record'u tanımlayan bir field veya birden fazla field'in kombinasyonudur.
- 2) Primary Key Unique'dir
- 3) Bir tabloda en fazla bir Primary Key olabilir
- 4) Primary Key field'ında hiç bir data NULL olamaz

“**id**” field'ini “**primary key**” yapmak için 2 yol var

1) Data Type'dan sonra “**PRIMARY KEY**” yazarak.

2) **CONSTRAINT** Keyword Kullanılarak Primary Key Tanımlanabilir

“**CONSTRAINT** constraintName **PRIMARY KEY**(column1, column2, ... column_n)”

Bir Tabloya “Foreign Key” Nasıl Eklenir

- ❑ **Foreign Key** iki tablo arasında Relation oluşturmak için kullanılır.
- ❑ **Foreign Key** başka bir tablonun Primary Key'ine bağlıdır.
- ❑ **Referenced table** (baglanılan tablo, Primary Key'in olduğu Tablo) *parent table* olarak adlandırılır. Foreign Key'in olduğu tablo ise *child table* olarak adlandırılır.
- ❑ Bir Tabloda birden fazla **Foreign Key** olabilir
- ❑ **Foreign Key** NULL değeri alabilir

Note 1: “Parent Table”da olmayan bir id'ye sahip datayı “Child Table”a ekleyemezsiniz

Note 2: Child Table'i silmeden Parent Table'i silemezsiniz. Once “Child Table” silinir, sonra “Parent Table” silinir.

Bir Tabloya “Foreign Key” Nasıl Eklenir

SIRKETLER Tablosu

Primary Key

SIRKET_ID	SIRKET	PERSONEL_SAYISI
100	Honda	12000
101	Ford	18000
102	Hyundai	10000
103	Toyota	21000

Parent Table

PERSONEL Tablosu

Foreign Key

ID	ISIM	SEHIR	MAAS	SIRKET
123456789	Ali Seker	Istanbul	2500	Honda
234567890	Ayşe Gul	Istanbul	1500	Toyota
345678901	Veli Yilmaz	Ankara	3000	Honda
456789012	Veli Yilmaz	Izmir	1000	Ford
567890123	Veli Yilmaz	Ankara	7000	Hyundai
456789012	Ayşe Gul	Ankara	1500	Ford
123456710	Fatma Yasa	Bursa	2500	Honda

Child Table

Bir Tabloya “Check Constraint” Nasıl Eklenir

CHECK ile bir alana girilebilecek değerleri sınırlandırabiliriz. Mesela tablomuzda YAŞ bir alanı **number** data tipinde yani sayısal alan olarak belirlemiş olabiliriz. Ancak bu alan negatif sayı girilmesi anlamsız olacağı için CHECK yapısını kullanarak negatif giriş yapılmasını engelleyebiliriz.

```
CREATE TABLE sehirler2 (  
  alan_kodu int PRIMARY KEY,  
  isim varchar(20) NOT NULL,  
  nufus int CHECK (nufus>0)  
);
```

DQL - SELECT KOMUTU ve WHERE KOŞULU

–Tablodaki **B**elli **B**ir **F**ield'i **C**agirma

```
SELECT isim  
FROM calisanlar;
```

–Tablodaki Koşulu Sağlayan **B**elli **B**ir **F**ield'i **C**agirma

```
SELECT isim  
FROM calisanlar  
WHERE maas>5000;
```

Tablodan sadece maas'ı 5000 den büyük olanların isim field'indeki dataları getirir

WHERE ile Kullanilan Mantiksal Operatorler

- = ==> Equal to sign
- > ==> Greater than sign
- < ==> Less than sign
- >= ==> Greater than or equal to sign
- <= ==> Less than or equal to sign
- <> ==> Not Equal to sign
- AND ==> And operator
- OR ==> Or operator

DQL - SELECT KOMUTU

Tablodan **B**irden **F**azla **F**ield'i **C**agirma

```
SELECT column1,column2 FROM table_name;
```

```
SELECT column1,column2 FROM table_name WHERE column_name=condition;
```

DML - DELETE KOMUTU

* **DELETE** FROM tablo_adi; ---> Tablonun tüm içeriğini siler.

* **DELETE** FROM tablo_adi WHERE sutun_adi = veri; —>Tabloda istediğiniz veriyi siler.

NOT:“**DELETE** FROM tablo_adi” ->tablodaki tüm dataları siler, fakat tabloyu silmez.

DML - DELETE - TRUNCATE

TRUNCATE komutu DELETE komutu gibi bir tablodaki verilerin tamamını siler. Ancak, seçmeli silme yapamaz.

!!! TRUNCATE TABLE where OLMAZ

TRUNCATE TABLE table_name; -- tablodaki verileri siler

ON DELETE CASCADE

- Her defasında önce child tablodaki verileri silmek yerine ON DELETE CASCADE silme özelliğini aktif hale getirebiliriz.
Bunun için FK olan satırın en sonuna ON DELETE CASCADE komutunu yazmak yeterli

- **on delete cascade** sayesinde parenttaki silinen bir kayıt ile ilişkili olan tüm child kayıtlarını silebiliriz
- cascade yoksa önce child silinir sonra parent

Drop Table Komutu

-Tabloyu SCHEMA'dan kaldırma:
Oluşturduğumuz tabloları silmek için DROP anahtar kelimesi kullanılır.

DROP TABLE (IF EXISTS) tablo_adi;

Burada IF EXISTS yapısını kullanarak yanlış tablo ismi yazımı durumunda hata mesajı almayı önleriz.

IN CONDITION

IN Condition birden fazla mantıksal ifade ile tanımlayabileceğimiz durumları (**Condition**) tek komutla yazabilme imkanı verir

AND (ve): Belirtilen şartların her ikisinde gerçekleşiyorsa o kayıt listelenir.

OR (veya): Belirtilen şartlardan biri gerçekleşirse, kayıt listelenir.

BETWEEN CONDITION

BETWEEN Condition iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkanı verir.

Yazdığımız 2 sınır da aralığa dahildir (**INCLUSIVE**)

Örnek:

--Müşteriler tablosunda urun_id 20 ile 40 arasında olan ürünlerin tüm bilgilerini listeleyniz.

NOT BETWEEN CONDITION

NOT BETWEEN Condition iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkanı verir. Yazdığımız 2 sınırdaki aralığa harictir (**EXCLUSIVE**)

1) Urun_id 20 ile 40 arasında olmayan ürünlerin tüm bilgilerini listeleymek

```
SELECT *  
FROM musteriler  
WHERE urun_id < 20 OR urun_id > 40;
```

ALIASES

Aliases(takma adları) bir tabloya veya sütuna geçici bir ad vermek için kullanılır. Bir takma ad sadece sorgu süresi boyunca mevcuttur.

```
CREATE TABLE workers (  
  calisan_id char(9),  
  calisan_isim varchar(50),  
  calisan_dogdugu_sehir varchar(50)  
);
```

calisan_id character (9)	calisan_isim character varying (50)	calisan_dogdugu_sehir character varying (50)
123456789	Ali Can	Istanbul
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir

```
INSERT INTO workers VALUES(123456789, 'Ali Can', 'Istanbul');  
INSERT INTO workers VALUES(234567890, 'Veli Cem', 'Ankara');  
INSERT INTO workers VALUES(345678901, 'Mine Bulut', 'Izmir');
```

ID	ISIM	DOGUM_YERI
123456789	Ali Can	Istanbul
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir

ID	ISIM_VE_DOGUM_YERI
123456789	Ali CanIstanbul
234567890	Veli CemAnkara
345678901	Mine BulutIzmir

AGGREGATE FONKSİYON KULLANIMI

AGGREGATE FONKSİYON birden çok değer üzerinde bir hesaplama gerçekleştirir ve tek bir değer döndürür.

SQL; avg, count, sum, min, max, vb. içeren birçok toplama işlevi sağlar.

SUBQUERIES

- SUBQUERY** başka bir SQL sorgusunun içinde bulunan ve onun sonucundan yararlanarak çalışan bir sorgudur. Bu sayede, veritabanında bulunan verileri daha karmaşık ve detaylı bir şekilde sorgulama imkanı sağlar.
- **Aggregate Metotlari** (SUM,COUNT, MIN, MAX, AVG) Subquery içinde kullanılabilir.

id integer	isim character varying (50)	sehir character varying (50)	maas integer	isyeri character varying (20)
123456789	Ali Seker	Istanbul	2500	Vakko
234567890	Ayse Gul	Istanbul	1500	LCWaikiki
345678901	Veli Yilmaz	Ankara	3000	Vakko
456789012	Veli Yilmaz	Izmir	1000	Pierre Cardin
567890123	Veli Yilmaz	Ankara	7000	Adidas
456789012	Ayse Gul	Ankara	1500	Pierre Cardin
123456710	Fatma Yasa	Bursa	2500	Vakko

marka_id integer	marka_isim character varying (20)	calisan_sayisi integer
100	Vakko	12000
101	Pierre Cardin	18000
102	Adidas	10000
103	LCWaikiki	21000

marka_id integer	marka_isim character varying (20)	sehir_sayisi bigint
100	Vakko	3
101	Pierre Cardin	2
102	Adidas	1
103	LCWaikiki	1

EXISTS CONDITION

EXISTS Condition subquery'ler ile kullanılır. IN ifadesinin kullanımına benzer olarak, EXISTS ve NOT EXISTS ifadeleri de alt sorgudan getirilen değerlerin içerisinde bir değer olması veya olmaması durumunda işlem yapılmasını sağlar.

MART		
urun_id	musteri_isim	urun_isim
10	Mark	Honda
20	John	Toyota
30	Amy	Ford
20	Mark	Toyota
10	Adam	Honda
40	John	Hyundai
20	Eddie	Toyota

NİSAN						
urun_id	musteri_isim	urun_isim		urun_id	musteri_isim	
10	Hasan	Honda		10	Mark	
10	Kemal	Honda		20	John	
20	Ayşe	Toyota		20	Mark	
50	Yasar	Volvo		10	Adam	
20	Mine	Toyota		20	Eddie	

Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

CREATE TABLE tedarikciler -- parent

```
(  
vergi_no int PRIMARY KEY,  
firma_ismi VARCHAR(50),  
irtibat_ismi VARCHAR(50)  
);
```

CREATE TABLE urunler -- child

```
(  
ted_vergino int,  
urun_id int,  
urun_isim VARCHAR(50),  
musteri_isim VARCHAR(50),  
CONSTRAINT fk_urunler FOREIGN KEY(ted_vergino)  
REFERENCES tedarikciler(vergi_no)  
on delete cascade  
);
```

```
INSERT INTO tedarikciler VALUES (101, 'IBM', 'Kim Yon');  
INSERT INTO tedarikciler VALUES (102, 'Huawei', 'Çin Li');  
INSERT INTO tedarikciler VALUES (103, 'Erikson', 'Maki  
Tammen');  
INSERT INTO tedarikciler VALUES (104, 'Apple', 'Adam Eve');
```

```
INSERT INTO urunler VALUES(101, 1001,'Laptop', 'Ayşe Can');  
INSERT INTO urunler VALUES(102, 1002,'Phone', 'Fatma Aka');  
INSERT INTO urunler VALUES(102, 1003,'TV', 'Ramazan Öz');  
INSERT INTO urunler VALUES(102, 1004,'Laptop', 'Veli Han');  
INSERT INTO urunler VALUES(103, 1005,'Phone', 'Canan Ak');  
INSERT INTO urunler VALUES(104, 1006,'TV', 'Ali Bak');  
INSERT INTO urunler VALUES(104, 1007,'Phone', 'Aslan Yılmaz');
```

Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

- vergi_no'su 102 olan tedarikcinin firma ismini 'Vestel' olarak güncelleyiniz.
- vergi_no'su 101 olan tedarikçinin firma ismini 'casper' ve irtibat_ismi'ni 'Ali Veli' olarak güncelleyiniz.
- urunler tablosundaki 'Phone' değerlerini 'Telefon' olarak güncelleyiniz.
- urunler tablosundaki urun_id değeri 1004'ten büyük olanların urun_id'sini 1 arttırın.
- urunler tablosundaki tüm ürünlerin urun_id değerini ted_vergino sutun değerleri ile toplayarak güncelleyiniz.
- urunler tablosundan Ali Bak'in aldığı urunun ismini, tedarikci tablosunda irtibat_ismi 'Adam Eve' olan firmanın ismi (firma_ismi) ile degistiriniz.
- Urunler tablosunda laptop satin alan musterilerin ismini, firma_ismi Apple'in irtibat_isim'i ile degistirin.

IS NULL CONDITION

Arama yapılan field'da NULL degeri almıs kayıtları getirir.

```
CREATE TABLE insanlar  
(  
  ssn char(9),  
  name varchar(50),  
  address varchar(50)  
);
```

```
INSERT INTO insanlar VALUES(123456789, 'Ali Can', 'Istanbul');  
INSERT INTO insanlar VALUES(234567890, 'Veli Cem', 'Ankara');  
INSERT INTO insanlar VALUES(345678901, 'Mine Bulut', 'Izmir');  
INSERT INTO insanlar (ssn, address) VALUES(456789012, 'Bursa');  
INSERT INTO insanlar (ssn, address) VALUES(567890123, 'Denizli');  
INSERT INTO insanlar (ssn, name) VALUES(567890123, 'Veli Han');
```

ORDER BY CLAUSE

ORDER BY komutu belli bir field'a gore NATURAL ORDER olarak siralama yapmak icin kullanilir

ORDER BY komutu sadece **SELECT** komutu ile kullanilir

```
CREATE TABLE people  
(  
  ssn char(9),  
  isim varchar(50),  
  soyisim varchar(50),  
  adres varchar(50)  
);
```

```
INSERT INTO people VALUES(123456789, 'Ali','Can', 'Istanbul');  
INSERT INTO people VALUES(234567890, 'Veli','Cem', 'Ankara');  
INSERT INTO people VALUES(345678901, 'Mine','Bulut', 'Ankara');  
INSERT INTO people VALUES(256789012, 'Mahmut','Bulut', 'Istanbul');  
INSERT INTO people VALUES (344678901, 'Mine','Yasa', 'Ankara');  
INSERT INTO people VALUES (345678901, 'Veli','Yilmaz', 'Istanbul');
```

Insanlar tablosundaki datalari adres'e gore siralayin

SSN	ISIM	SOYISIM	ADRES	SSN	ISIM	SOYISIM	ADRES
123456789	Ali	Can	Istanbul	345678901	Mine	Bulut	Ankara
234567890	Veli	Cem	Ankara	344678901	Mine	Yasa	Ankara
345678901	Mine	Bulut	Ankara	234567890	Veli	Cem	Ankara
256789012	Mahmut	Bulut	Istanbul	123456789	Ali	Can	Istanbul
344678901	Mine	Yasa	Ankara	345678901	Veli	Yilmaz	Istanbul
345678901	Veli	Yilmaz	Istanbul	256789012	Mahmut	Bulut	Istanbul

GROUP BY CLAUSE

Group By komutu sonuçları bir veya daha fazla sütuna göre gruplamak için **SELECT** komutuyla birlikte kullanılır

ISIM	URUN_ADI	URUN_MIKTAR
Ali	Elma	5
Ayşe	Armut	3
Veli	Elma	2
Hasan	Uzum	4
Ali	Armut	2
Ayşe	Elma	3
Veli	Uzum	5
Ali	Armut	2
Veli	Elma	3
Ayşe	Uzum	2

1) İsme göre alınan toplam ürünleri bulun

ISIM	ALINAN_TOPLAM_MEYVE
Veli	10
Ayşe	8
Ali	9
Hasan	4

HAVING CLAUSE

HAVING, AGGREGATE FUNCTION'lar ile birlikte kullanılan FİLTRELEME komutudur.

UNION OPERATOR

İki farklı sorgulamanın sonucunu birleştiren işlemidir. Seçilen **Field SAYISI** ve **DATA TYPE**'i aynı olmalıdır.

NOT : Union işlemi yaparken

- 1) Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı
- 2) Alt alta gelecek sütunların data type'leri aynı olmalı

UNION ALL OPERATOR

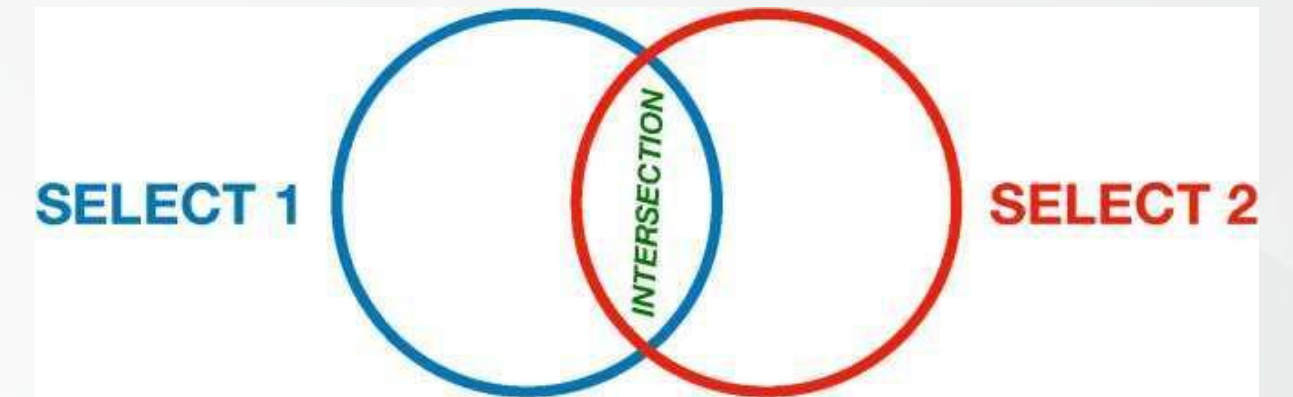
UNION islemi 2 veya daha çok **SELECT** isleminin sonuc **KUMELERINI** birleştirmek için kullanılır, Aynı kayıt birden fazla olursa, sadece bir tanesini alır.

UNION ALL ise tekrarlı elemanları, tekrar sayısınca yazar.

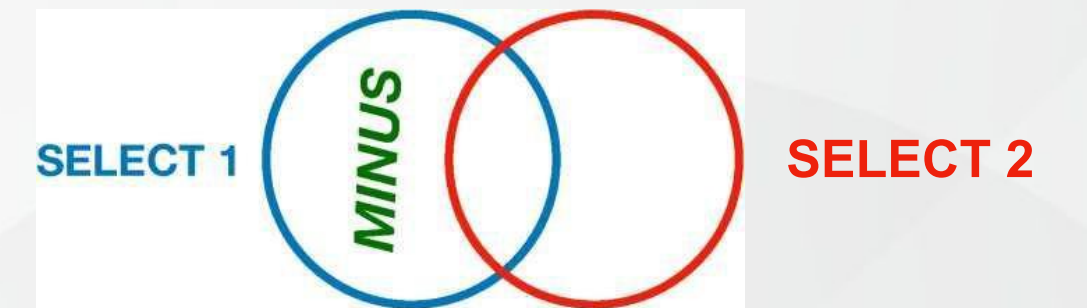
NOT : UNION ALL ile birleştirmelerde de

- 1) Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı
- 2) Alt alta gelecek sütunların data type'leri aynı olmalı

INTERSECT OPERATOR



EXCEPT OPERATOR



TEKRAR

- 1-**HAVING**, AGGREGATE FUNCTION'lar ile birlikte kullanılan FİLTRELEME komutudur.
- 2- UNION : İki farklı sorgulamanın sonucunu birleştiren işlemidir. İkinci sorgudan gelen sonuç ilk sorgudan gelmişse tekrar yazılmaz
 - Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı
 - Alt alta gelecek sütunların data type'leri aynı olmalı
- 3-**UNION ALL**, UNION işlemi ile aynı işleve sahiptir , farkı ise tekrarlı elemanları, tekrar sayısınıca yazar.
- 4-INTERSECT : Her iki sorgu sonucunu karşılaştırır ve ortak olanları listeler
- 5- EXCEPT : Birinci sorguda olup, ikinci sorguda olmayan sonuçları listeler

JOINS

2 veya daha fazla **tablo**daki datalari Birleştirmek için kullanılır.
Su ana kadar gördüğümüz Union, Intersect ve Except (Minus) sorgu sonuçları için kullanılır. Tablolar için ise **JOIN** kullanılır

5 Cesi Join vardır

- 1) INNER JOIN iki Tablodaki ortak datalari gösterir
- 2) LEFT JOIN İlk datada olan tüm recordları gösterir
- 3) RIGHT JOIN İkinci tabloda olan tüm recordları gösterir
- 4) FULL JOIN İki tablodaki tüm recordları gösterir
- 5) SELF JOIN Bir tablonun kendi içinde Join edilmesi ile oluşur.

INNER JOINS

TABLE 1

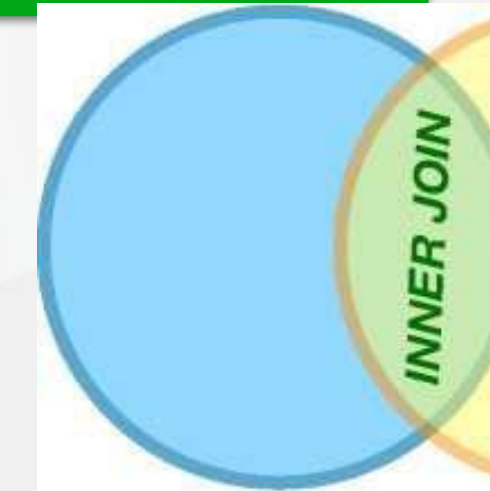


TABLE 2

SORU) İki Tabloda `sirket_id`'si aynı olanların `sirket_ismi`, `siparis_id` ve `siparis_tarihleri` ile yeni bir tablo oluşturun

```
SELECT sirketler.sirket_isim, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler INNER JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

SIRKET_ISIM	SIPARIS_ID	SIPARIS_TARIHI
Honda	11	17-APR-20
Ford	22	18-APR-20
Hyundai	33	19-APR-20

NOT :

- 1) Select'ten sonra tabloda görmek istediğiniz sütunları yazarken **Tablo_adi.field_adi** şeklinde yazın
- 2) From'dan sonra tablo ismi yazarken **1.Tablo ismi + INNER JOIN + 2.Tablo ismi** yazmalıyız
- 3) Join'i hangi kurala göre yapacağınızı belirtmelisiniz. Bunun için **ON+ kuralımız** yazılmalı

LEFT JOINS

TABLE 1

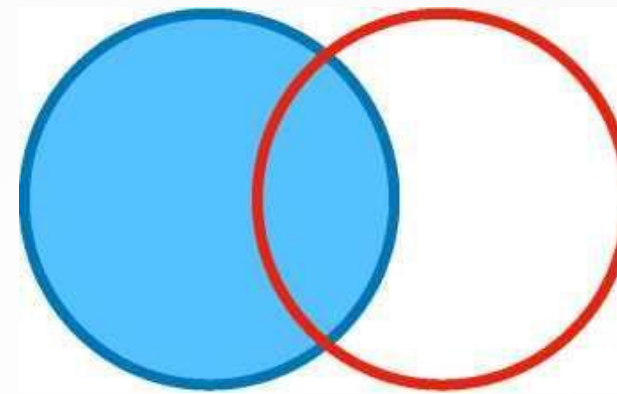


TABLE 2

```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler LEFT JOIN siparisler  
ON sirketler.sirket_id = siparisler.sirket_id;
```

SIRKET_ISIM	SIPARIS_ID	SIPARIS_TARIHI
Honda	11	17-APR-20
Ford	22	18-APR-20
Hyundai	33	19-APR-20
Toyota	-	-

NOT :

- 1) Left Join'de ilk tablodaki tum record'lar gosterilir.
- 2) İlk tablodaki datalara 2.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gosterilir ancak ortak olmayan datalar icin o kisimler bos kalir
- 3) İlk yazdiginiz Tablonun tamamini aldigi icin hangi tabloyu istedigimize karar verip once onu yazmaliyiz

RIGHT JOINS

TABLE 1

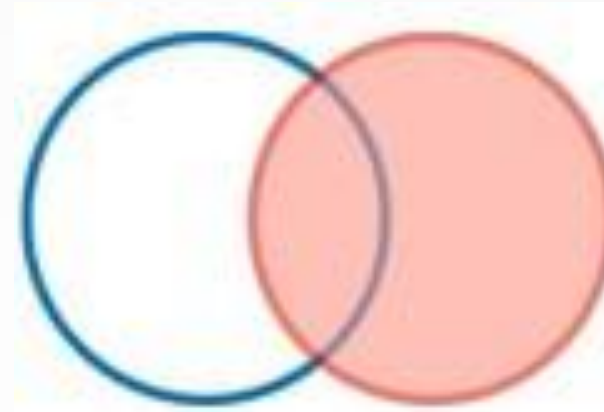


TABLE 2

```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler RIGHT JOIN siparisler  
ON sirketler.sirket_id = siparisler.sirket_id;
```

SIRKET_ISIM	SIPARIS_ID	SIPARIS_TARIHI
Honda	11	17-APR-20
Ford	22	18-APR-20
Hyundai	33	19-APR-20
-	55	21-APR-20
-	44	20-APR-20

NOT :

- 1) Right Join'de ikinci tablodaki tum record'lar gosterilir.
- 2) Ikinci tablodaki datalara 1.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gosterilir ancak ortak olmayan datalar icin o kisimler bos kalir

FULL JOINS

```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler FULL JOIN siparisler  
ON sirketler.sirket_id = siparisler.sirket_id;
```

NOT :

- 1) FULL Join'de iki tabloda var olan tum record'lar gosterilir.
- 2) Bir tabloda olup otekinde olmayan data'lar bos kalir

SIRKET_ISIM	SIPARIS_ID	SIPARIS_TARIHI
Honda	11	17-APR-20
Ford	22	18-APR-20
Hyundai	33	19-APR-20
-	44	20-APR-20
-	55	21-APR-20
Toyota	-	-

SELF JOINS

LIKE Condition

LIKE condition **WHERE** ile kullanılarak **SELECT**, **INSERT**, **UPDATE**, veya **DELETE** statement ile çalışan **wildcards**'a(özel sembol) izin verir.. Ve bize **pattern matching** yapma imkani verir.

LIKE Condition

3) **REGEXP_LIKE** => Daha karmaşık sorgular için herhangi bir kod, metin içerisinde istenilen yazı veya kod parçasının aranıp bulunmasını sağlayan kendine ait söz dizimi olan bir yapıdır.
(REGEXP_LIKE) PostgreSQL'de " ~ " karakteri ile kullanılır

NOT LIKE Condition

LIKE Condition

LIKE: Sorgulama yaparken belirli patternleri(kalıp ifadelerle sorgu) kullanabilmezi sağlar

ILIKE: Sorgulama yaparken büyük/küçük harfe duyarsız olarak eşleştirir.

LIKE = ~~

ILIKE = ~~*

NOT LIKE = !~~

NOT ILIKE = !~~*

NOT REGEXP_LIKE = !~*

NOT REGEXP_LIKE = !~

Tekrar

JOIN : Bir tablonun kendisinden veya iki farklı tablodan istediğimiz field'leri kullanarak yeni tablo elde etmek için kullanılır.

- Join işlemi yeni tablo oluşturmaz fakat yazdığımız sorgu sonuçlarını istediğimiz şekilde bir tabloda görmemizi sağlar
- Daha önce gördüğümüz union,intersect,minus komutları tabloları birleştirmiyor, ayrı ayrı sorgular yapıp sorgu sonuçlarını birleştiriyordu. Hatta farklı tablolardan gelen ism ve şehir gibi değerleri tablo olarak yansıtamadığından alt alta yazmamız gerekiyordu.
- INNER : her iki tabloda ortak olan recordlara ait her iki tablodaki bilgileri gösterir
- LEFT JOIN: Birinci tablodaki tüm recordları ve ortak recordlara ait ikinci tablodaki bilgileri gösterir
- RIGHT JOIN : İkinci tablodaki tüm recordları ve ortak recordlara ait birinci tablodaki bilgileri gösterir
- FULL JOIN : Her iki tablodaki tüm record'ları gösterir, bir tabloda olup diğeri tabloda karşılığı olmayan bilgiler için null yazar

Tekrar

- SELF JOIN : Bir tablonun kendi içerisinde farklı bir kurala göre INNER JOIN yapılmasıdır.

Aynı tablo iki kere kullanılacağı için geçici olarak tabloya iki farklı isim verilir ve bu genelde tablo adının ilk harfi ile 1 ve 2 rakamlarının kullanılmasıyla olur (p1,p2 gibi)

FROM satırında tablo adı ile kısaltmalar birlikte yazılarak tanımlanır

FROM personel p1 INNER JOIN personel p2

ON komutundan sonra oluşturulan iki sanal tablo arasındaki kural tanımlanır

- LIKE : WHERE komutundan sonra kullanılır, wildcards kullanımina izin vererek

PATTERN MATCHING (şekil benzetme) özelliğini kullanır

- (%) : yazdığımız yerden sonra 0 veya daha fazla karakter olabilir

- (__) : sadece 1 tane karakteri gösterir

String Functions

Tabloları yazdırırken büyük harf(UPPER), küçük harf(LOWER) veya ilk harfleri büyük diğerleri küçük harf(INITCAP) yazdırmak için kullanırız

DISTINCT: DISTINCT clause, çağrılan terimlerden tekrarlı olanların sadece birincisini alır.

FETCH NEXT (SAYI) ROW ONLY- OFFSET

ALTER TABLE STATEMENT

ALTER TABLE statement tabloda **add**, Type(**modify**)/Set, Rename veya **drop columns** islemleri için kullanılır.

ALTER TABLE statement tabloları yeniden isimlendirmek için de kullanılır.

ALTER TABLE STATEMENT

1) ADD default deger ile tabloya bir field ekleme

ALTER TABLE personel

ADD ulke_isim varchar(20) DEFAULT 'Turkiye';

ID	ISIM	SEHIR	MAAS	SIRKET	ULKE_ISIM
123456789	Ali Yilmaz	Istanbul	5500	Honda	Turkiye
234567890	Veli Sahin	Istanbul	4500	Toyota	Turkiye
345678901	Mehmet Ozturk	Ankara	3500	Honda	Turkiye
456789012	Mehmet Ozturk	Izmir	6000	Ford	Turkiye
567890123	Mehmet Ozturk	Ankara	7000	Tofas	Turkiye
456715012	Veli Sahin	Ankara	4500	Ford	Turkiye
123456710	Hatice Sahin	Bursa	4500	Honda	Turkiye

2) Tabloya birden fazla field ekleme

ALTER TABLE personel

ADD cinsiyet varchar(20) , ADD yas int;

ID	ISIM	SEHIR	MAAS	SIRKET	ULKE_ISIM	CINSIYET	YAS
123456789	Ali Yilmaz	Istanbul	5500	Honda	Turkiye	-	-
234567890	Veli Sahin	Istanbul	4500	Toyota	Turkiye	-	-
345678901	Mehmet Ozturk	Ankara	3500	Honda	Turkiye	-	-
456789012	Mehmet Ozturk	Izmir	6000	Ford	Turkiye	-	-
567890123	Mehmet Ozturk	Ankara	7000	Tofas	Turkiye	-	-
456715012	Veli Sahin	Ankara	4500	Ford	Turkiye	-	-
123456710	Hatice Sahin	Bursa	4500	Honda	Turkiye	-	-

ALTER TABLE STATEMENT

3) DROP tablodan sutun silme

ALTER TABLE personel
DROP COLUMN yas;

ID	ISIM	SEHIR	MAAS	SIRKET	ULKE_ISIM	CINSIYET
123456789	Ali Yilmaz	Istanbul	5500	Honda	Turkiye	-
234567890	Veli Sahin	Istanbul	4500	Toyota	Turkiye	-
345678901	Mehmet Ozturk	Ankara	3500	Honda	Turkiye	-
456789012	Mehmet Ozturk	Izmir	6000	Ford	Turkiye	-
567890123	Mehmet Ozturk	Ankara	7000	Tofas	Turkiye	-
456715012	Veli Sahin	Ankara	4500	Ford	Turkiye	-
123456710	Hatice Sahin	Bursa	4500	Honda	Turkiye	-

4) RENAME COLUMN sutun adi degistirme

ALTER TABLE personel
RENAME COLUMN ulke_isim TO ulke_adi;

ID	ISIM	SEHIR	MAAS	SIRKET	ULKE_ADİ	CINSIYET
123456789	Ali Yilmaz	Istanbul	5500	Honda	Turkiye	-
234567890	Veli Sahin	Istanbul	4500	Toyota	Turkiye	-
345678901	Mehmet Ozturk	Ankara	3500	Honda	Turkiye	-
456789012	Mehmet Ozturk	Izmir	6000	Ford	Turkiye	-
567890123	Mehmet Ozturk	Ankara	7000	Tofas	Turkiye	-
456715012	Veli Sahin	Ankara	4500	Ford	Turkiye	-
123456710	Hatice Sahin	Bursa	4500	Honda	Turkiye	-

ALTER TABLE STATEMENT

5) RENAME tablonun ismini degistirme

ALTER TABLE personel
RENAME TO isciler;

6) TYPE/SET sutunlarin ozelliklerini degistirme

ALTER TABLE isciler















ALTER COLUMN ulke_adi TYPE varchar(30),

ALTER COLUMN ulke_adi SET NOT NULL;

Not: String data türünü numerik bir data türüne dönüştürmek istersek;

ALTER COLUMN fieldname

TYPE int USING(fieldname::int) şeklinde yaparız.

Columns						
		Name	Data type	Length/Precision	Scale	Not NULL?
		id	integer v			<input checked="" type="checkbox"/>
		isim	character varying v	50		<input type="checkbox"/>
		sehir	character varying v	50		<input type="checkbox"/>
		maas	integer v			<input type="checkbox"/>
		sirket	character varying v	20		<input type="checkbox"/>
		ulke_adi	character varying v	30		<input checked="" type="checkbox"/>
		cinsiyet	character varying v	20		<input type="checkbox"/>

TRANSACTION (Begin – Savepoint – rollback - commit)

- Transaction veritabanı sistemlerinde bir işlem başladığında başlar ve işlem bitince sona erer. Bu işlemler veritabanı oluşturma , veri silme , veri güncelleme, veriyi gerigetirme gibi işlemler olabilir .

```
CREATE TABLE ogrenciler2  
(  
  id serial,  
  isim VARCHAR(50),  
  veli_isim VARCHAR(50),  
  yazili_notu real  
);
```

```
BEGIN;  
INSERT INTO ogrenciler2 VALUES(default, 'Ali Can', 'Hasan',75.5);  
INSERT INTO ogrenciler2 VALUES(default, 'Merve Gul', 'Ayse',85.3);  
savepoint x;  
INSERT INTO ogrenciler2 VALUES(default, 'Kemal Yasa', 'Hasan',85.6);  
INSERT INTO ogrenciler2 VALUES(default, 'Nesibe Yilmaz', 'Ayse',95.3);  
savepoint y;  
INSERT INTO ogrenciler2 VALUES(default, 'Mustafa Bak', 'Can',99);  
INSERT INTO ogrenciler2 VALUES(default, 'Can Bak', 'Ali', 67.5);  
ROLLBACK to y;  
COMMIT;
```

TRANSACTION (Begin – Savepoint – rollback - commit)

--Transaction kullanımında SERIAL data türü kullanımı tercih edilmez. Savepointten sonra eklediğimiz veride sayaç mantığı ile çalıştığı için sayakta en son hangi sayıda kaldıysa oradan devam eder

NOT :PostgreSQL de Transaction kullanımı için «Begin;» komutuyla başlarız sonrasında tekrar yanlış bir veriyi düzeltmek veya bizim için önemli olan verilerden sonra ekleme yapabilmek için "SAVEPOINT savepointismi" komutunu kullanırız ve bu savepointe dönebilmek için "ROLLBACK TO savepointismi" komutunu kullanırız ve rollback çalıştırıldığında savepoint yazdığımız satırın üstündeki verileri tabloda bize verir ve son olarak Transaction'ı sonlandırmak için mutlaka "COMMIT" komutu kullanılır.

INTERVIEW QUESTION

CREATE TABLE personel

```
(  
  id number(9),  
  isim varchar2(50),  
  sehir varchar2(50),  
  maas number(20),  
  sirket varchar2(20)  
);  
  
INSERT INTO personel VALUES(123456789, 'Johnny Walk', 'New Hampshire', 2500, 'IBM');  
INSERT INTO personel VALUES(234567891, 'Brian Pitt', 'Florida', 1500, 'LINUX');  
INSERT INTO personel VALUES(245678901, 'Eddie Murphy', 'Texas', 3000, 'WELLS FARGO');  
INSERT INTO personel VALUES(456789012, 'Teddy Murphy', 'Virginia', 1000, 'GOOGLE');  
INSERT INTO personel VALUES(567890124, 'Eddie Murphy', 'Massachuset', 7000, 'MICROSOFT');  
INSERT INTO personel VALUES(456789012, 'Brad Pitt', 'Texas', 1500, 'TD BANK');  
INSERT INTO personel VALUES(123456719, 'Adem Stone', 'New Jersey', 2500, 'IBM');
```

CREATE TABLE isciler

```
(  
  id number(9),  
  isim varchar2(50),  
  sehir varchar2(50),  
  maas number(20),  
  sirket varchar2(20)  
);  
  
INSERT INTO isciler VALUES(123456789, 'John Walker', 'Florida', 2500, 'IBM');  
INSERT INTO isciler VALUES(234567890, 'Brad Pitt', 'Florida', 1500, 'APPLE');  
INSERT INTO isciler VALUES(345678901, 'Eddie Murphy', 'Texas', 3000, 'IBM');  
INSERT INTO isciler VALUES(456789012, 'Eddie Murphy', 'Virginia', 1000, 'GOOGLE');  
INSERT INTO isciler VALUES(567890123, 'Eddie Murphy', 'Texas', 7000, 'MICROSOFT');  
INSERT INTO isciler VALUES(456789012, 'Brad Pitt', 'Texas', 1500, 'GOOGLE');  
INSERT INTO isciler VALUES(123456710, 'Mark Stone', 'Pennsylvania', 2500, 'IBM');
```

INTERVIEW QUESTION

1) Her iki tablodaki ortak id'leri ve personel tablosunda bu id'ye sahip isimleri listeleyen query yazınız

```
SELECT isim,id
FROM personel
WHERE id IN (SELECT id
             FROM isciler
             WHERE isciler.id=personel.id);
```

ISIM	ID
Johnny Walk	123456789
Teddy Murphy	456789012
Brad Pitt	456789012

2) Her iki tablodaki ortak id ve isme sahip kayitlari listeleyen query yaziniz

```
SELECT isim,id
FROM personel
```

```
INTERSECT
```

```
SELECT isim,id
FROM personel;
```

ISIM	ID
Brad Pitt	456789012

INTERVIEW QUESTION

3) Personel tablosunda kac farkli sehirden personel var?

```
SELECT COUNT (DISTINCT sehir) AS sehir_sayisi  
FROM personel;
```

SEHIR_SAYISI
5

4) Personel tablosunda id'si cift sayi olan personel'in tum bilgilerini listeleyen Query yaziniz

```
SELECT *  
FROM personel  
WHERE MOD (id,2)=0;
```

ID	ISIM	SEHIR	MAAS	SIRKET
456789012	Teddy Murphy	Virginia	1000	GOOGLE
456789012	Brad Pitt	Texas	1500	TD BANK

INTERVIEW QUESTION

5) Personel tablosunda kac tane kayıt oldugunu gosteren query yazın

```
SELECT COUNT(*)  
FROM personel;
```

COUNT(*)
6

```
SELECT COUNT(id) AS kayıt_sayisi  
FROM personel;
```

KAYIT_SAYISI
6

6) Isciler tablosunda en yuksek maasi alan kisinin tum bilgilerini gosteren query yazın

Max Maas

```
SELECT MAX(maas) AS max_maas  
FROM isciler;
```

```
SELECT *  
FROM isciler  
WHERE maas IN (SELECT  
MAX(maas)  
FROM isciler);
```

ID	ISIM	SEHIR	MAAS	SIRKET
567890123	Eddie Murphy	Texas	7000	MICROSOFT

INTERVIEW QUESTION

7) Personel tablosunda en dusuk maasi alan kisinin tum bilgilerini gosteren query yazin

```
SELECT *  
FROM personel  
ORDER BY maas  
FETCH NEXT 1 ROW ONLY;
```

ID	ISIM	SEHIR	MAAS	SIRKET
456789012	Teddy Murphy	Virginia	1000	GOOGLE

8) Isciler tablosunda ikinci en yuksek maasi maasi gosteren query yazin

```
SELECT MAX(maas)  
FROM personel  
WHERE maas<>(SELECT MAX(maas)  
FROM personel);
```

MAX(MAAS)
2500

INTERVIEW QUESTION

9) Isciler tablosunda ikinci en dusuk maasi alan iscinin tum bilgilerini gosteren query yazin

```
SELECT *  
FROM isciler  
ORDER BY maas  
OFFSET 1 ROW  
FETCH NEXT 1 ROW ONLY;
```

ID	ISIM	SEHIR	MAAS	SIRKET
234567890	Brad Pitt	Florida	1500	APPLE

ID	ISIM	SEHIR	MAAS	SIRKET
123456789	John Walker	Florida	2500	IBM
234567890	Brad Pitt	Florida	1500	APPLE
345678901	Eddie Murphy	Texas	3000	IBM
456789012	Eddie Murphy	Virginia	1000	GOOGLE
567890123	Eddie Murphy	Texas	7000	MICROSOFT
456789012	Brad Pitt	Texas	1500	GOOGLE
123456710	Mark Stone	Pennsylvania	2500	IBM

ID	ISIM	SEHIR	MAAS	SIRKET
456789012	Eddie Murphy	Virginia	1000	GOOGLE
234567890	Brad Pitt	Florida	1500	APPLE
456789012	Brad Pitt	Texas	1500	GOOGLE
123456710	Mark Stone	Pennsylvania	2500	IBM
123456789	John Walker	Florida	2500	IBM
345678901	Eddie Murphy	Texas	3000	IBM
567890123	Eddie Murphy	Texas	7000	MICROSOFT

INTERVIEW QUESTION

10) Isciler tablosunda en yuksek maasi alan iscinin disindaki tum iscilerin, tum bilgilerini gosteren query yazin

```
SELECT *  
FROM isciler  
WHERE maas<>( SELECT MAX(maas)  
               FROM isciler)  
ORDER BY maas DESC;
```

ID	ISIM	SEHIR	MAAS	SIRKET
345678901	Eddie Murphy	Texas	3000	IBM
123456710	Mark Stone	Pennsylvania	2500	IBM
123456789	John Walker	Florida	2500	IBM
234567890	Brad Pitt	Florida	1500	APPLE
456789012	Brad Pitt	Texas	1500	GOOGLE
456789012	Eddie Murphy	Virginia	1000	GOOGLE

Tekrar Sorulari

- 1) DELETE ile TRUNCATE arasindaki fark nedir ?
- 2) DELETE ile DROP arasindaki fark nedir?
- 3) DROP ile DROP PURGE arasindaki fark nedir?
- 4) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz.

```
SELECT *  
FROM ogrenciler  
WHERE yas >= 8 AND yas <= 17;
```

- 5) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
  
FROM ogrenciler  
WHERE yas < 8 OR yas > 17;
```

- 6) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
  
FROM ogrenciler  
WHERE yas = 6 OR yas = 7 OR yas = 8 OR yas = 9;
```

Tekrar Sorularinin Cevaplari

- 1) DELETE ile TRUNCATE arasindaki fark nedir ?
 - A) TRUNCATE tum kayitlari siler, DELETE istersek tum kayitlari, istersek belirli kayitlari siler
 - B) DELETE ile sildigimiz datalari ROLLBACK yapabiliriz, TRUNCATE ile silinenler geri getirilemez
 - C) DELETE ile WHERE komutunu kullanabiliriz ama TRUNCATE ile kullanamayiz
- 2) DELETE ile DROP arasindaki fark nedir?

DELETE kayitlari siler, DROP ise tab;olari.
- 3) DROP ile DROP PURGE arasindaki fark nedir?

DROP ile sildigimiz dosyalar RECYCLEBIN'e gider. PURGE RECYCLEBIN'deki dosyalari geri getirilmeyecek sekilde siler. DROP PURGE beraber kullanilrsa geri getirilmeyecek sekilde silinir.

- 4) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz.

```
SELECT *  
FROM ogrenciler  
WHERE yas >= 8 AND yas <= 17;
```

- 5) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
FROM students  
WHERE age < 8 OR yas > 17;
```

- 6) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
FROM students  
WHERE yas = 6 OR yas = 7 OR yas = 8 OR yas = 9;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas BETWEEN 8 AND 17;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas NOT BETWEEN 8 AND 17;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas IN (6,7,8,9);
```

KISA TEKRAR

Personel isminde bir tablo olusturun.Icinde id,isim,sehir,maas ve sirket field'lari olsun. Id'yi 2.yontemle PK yapin

```
CREATE TABLE personel
```

```
(  
  id number(9),  
  isim varchar2(50),  
  sehir varchar2(50),  
  maas number(20),  
  sirket varchar2(20),  
  CONSTRAINT personel_pk PRIMARY KEY (id)  
);
```

```
INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'Istanbul', 5500, 'Honda');  
INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'Istanbul', 4500, 'Toyota');  
INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');  
INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'Izmir', 6000, 'Ford');  
INSERT INTO personel VALUES(567890123, 'Mehmet Ozturk', 'Ankara', 7000, 'Tofas');  
INSERT INTO personel VALUES(456715012, 'Veli Sahin', 'Ankara', 4500, 'Ford');  
INSERT INTO personel VALUES(123456710, 'Hatice Sahin', 'Bursa', 4500, 'Honda');
```

Personel_bilgi isminde bir tablo olusturun.Icinde id,tel ve cocuk sayisi field'lari olsun. Id'yi FK yapin ve personel tablosu ile relation kurun

```
CREATE TABLE personel_bilgi
```

```
(  
  id number(9),  
  tel char(10) UNIQUE ,  
  cocuk_sayisi number(2),  
  CONSTRAINT personel_bilgi_fk FOREIGN KEY (id) REFERENCES personel(id)  
);
```

```
INSERT INTO personel_bilgi VALUES(123456789, '5302345678' , 5);  
INSERT INTO personel_bilgi VALUES(234567890, '5422345678', , 4);  
INSERT INTO personel_bilgi VALUES(345678901, '5354561245', 3);  
INSERT INTO personel_bilgi VALUES(456789012, '5411452659', 3);  
INSERT INTO personel_bilgi VALUES(567890123, '5551253698', 2);  
INSERT INTO personel_bilgi VALUES(456789012, '5524578574', 2);  
INSERT INTO personel_bilgi VALUES(123456710, '5537488585', 1);
```

KISA TEKRAR

ID	ISIM	SEHIR	MAAS	SIRKET
123456789	Ali Yilmaz	Istanbul	5500	Honda
234567890	Veli Sahin	Istanbul	4500	Toyota
345678901	Mehmet Ozturk	Ankara	3500	Honda
456789012	Mehmet Ozturk	Izmir	6000	Ford
567890123	Mehmet Ozturk	Ankara	7000	Tofas
456789152	Veli Sahin	Ankara	4500	Ford
123456710	Hatice Sahin	Bursa	4500	Honda

ID	TEL	COCUK_SAYISI
123456789	5302345678	5
234567890	5422345678	4
345678901	5354561245	3
456789012	5411452659	3
567890123	5551253698	2
456789012	5524578574	2
123456710	5537488585	1

SORU 1) Personel_bilgi tablosundan 5 cocugu olan kisinin cocuk sayisini 2 yapin

```
UPDATE personel_bilgi  
SET cocuk_sayisi=2  
WHERE cocuk_sayisi=5;
```

Tekrar Sorulari

SORU 2) Persone tablosundan ucreti 4500 veya 5000 olanlarin maaslarini %10 artirin

```
UPDATE personel  
SET maas=maas*1.1  
WHERE maas IN (4500,5000);
```

SORU 3) Persone tablosundan maasi 4950 olanlari silin

```
DELETE FROM personel  
WHERE maas =4900;
```

```
ORA-02292: integrity constraint (SQL_AHZDXVGZXDBVBVLYOPIBABNDG.PERSONEL_BILGI_FK) violated - child record found  
ORA-06512: at  
"SYS.DBMS_SQL", line 1721
```

Tekrar Sorulari

SORU 4) Personel_bilgi tablosundan 3 veya 4 cocugu olanlari silin

```
DELETE FROM personel_bilgi  
WHERE cocuk_sayisi IN (3,4);
```

SORU 5) Persone tablosundan HONDA'da calisip maasi 3500 olanlari silin

```
DELETE FROM personel  
WHERE maas =3500 AND sirket='Honda';
```

Tekrar Sorulari

SORU 6) Personel_bilgi tablosundan datalari geri getirilemeyecek sekilde silin

```
TRUNCATE TABLE personel_bilgi;
```

SORU 7) Persone tablosundan maasi 4000 ile 5000 arasinda olanlari silin

```
DELETE FROM personel  
WHERE maas BETWEEN 4000 AND 5000;
```

Tekrar Sorulari

SORU 8) Personel tablosundan maasi 5000 ile 6000 arasinda olmayanlari silin

```
DELETE FROM personel  
WHERE maas NOT BETWEEN 5000 AND 6000;
```

SORU 9) Persone tablosunu geri getirilemeyecek sekilde silin

```
DROP TABLE personel PURGE; HATA VERIR  
Once personel_bilgi tablosunu silin
```

```
DROP TABLE personel_bilgi;
```

Persone tablosunu geri getirilemeyecek sekilde silin
