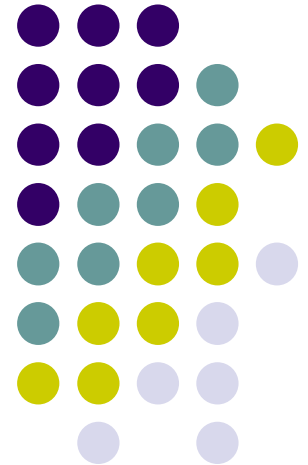
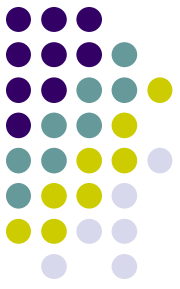


# BÖLÜM 1

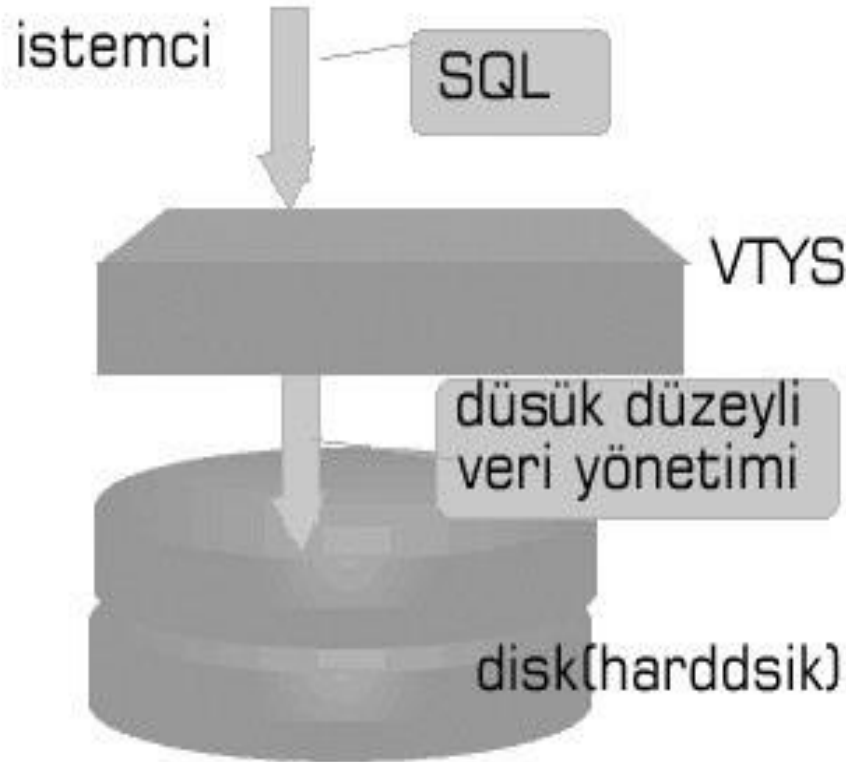
## VERİ TABANI YÖNETİM SİSTEMLERİ (VTYS)



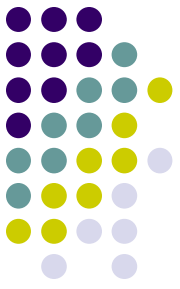


- Veritabanı yönetim sistemleri, fiziksel hafızada bilgileri çeşitli özelliklere göre gruplandırıp şekillendirdikten sonra saklayan programdır.
- Kısaca VTYS diye adlandırılır. VTYS, saklanan bu veriyi, SQL komutları ile insanların istekleri çerçevesinde işler, yeniden şekillendirir. Yani, Veritabanı Yönetim Sistemi'nin bir ucunda, bilgisayar disk(ler)inde saklanan düzenlenmiş veriler, diğer ucunda ise bir kullanıcı (genellikle insan) vardır.

- Veritabanı Yönetim Sistemleri, her zaman bir kullanıcıya, yönetim ekranlarından bilgi vermek için kullanılmaz. Bazen kullanıcı konumunda doğrudan bir insan yerine bir programda olabilir.



Şekil 1.1: VTYS Kalıbı



- Bilgi ve veri kavramları birbirleriyle karıştırılabilmektedir. Veri bilgiyi elde etmeye yönelik olan ve işlenmemiş ham bir malzeme, dağınık haldeki bir topluluktur.
- Bilgi ise verilerden anlamlı bir bütün teşkil edecek şekilde bir araya getirilmiş ve gelecekle ilgili olayları yada organizasyonları etkileyecek bir topluluktur.
- Bilgisayarda veri depolama işlemleri için çoğunlukla veritabanı yönetim sistemleri kullanılır.



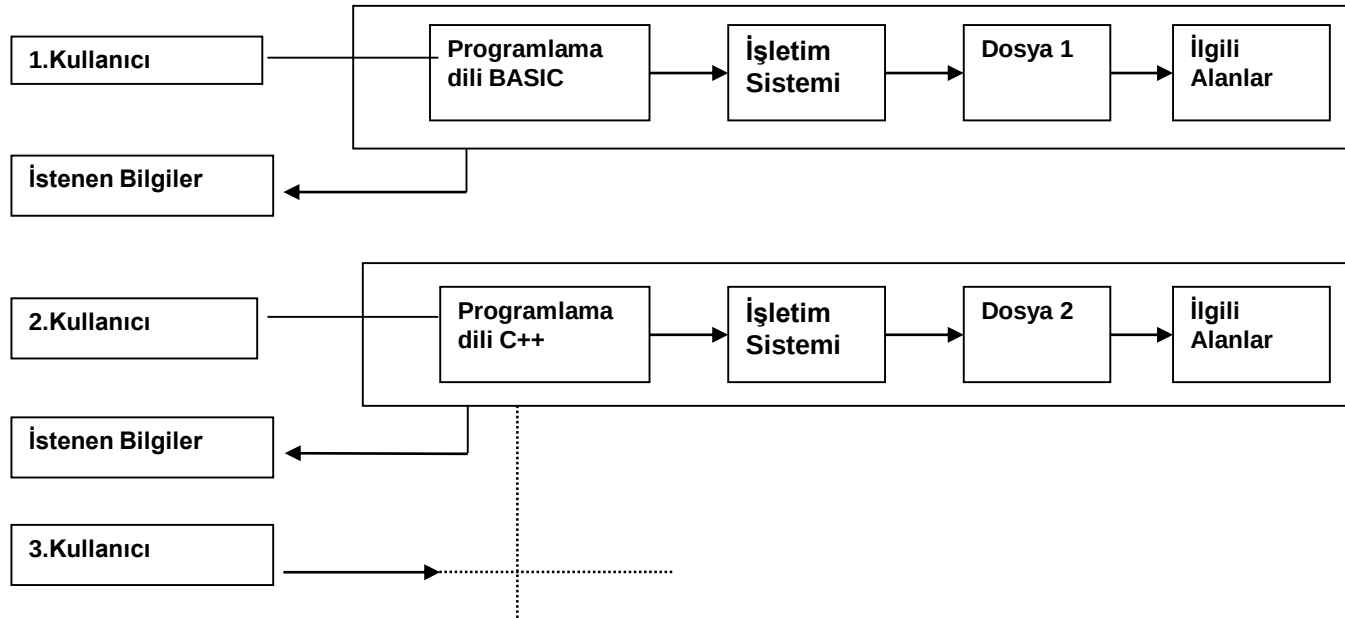


- Veritabanı yönetim sistemlerinde yanlış verinin depolanmasına ve/veya verinin istenmeyen kişilerin kullanımına sunulmasının engelleyen birtakım imkânlar bulunur. Çeşitli konularda birçok bilgi saklanması gerektiği zaman bu işlemler klasik dosya sistemi ve veritabanı sistemi ile yürütülebilir.

# 1.1.Klasik Dosya (File) Sistemi



- Burada n tane ayrı dosya n tane ayrı program tarafından işlenmektedir. Her program rapor üretme esnasında ayrı ayrı işletim sistemi ile yüz yüze (face to face) gelmektedir.

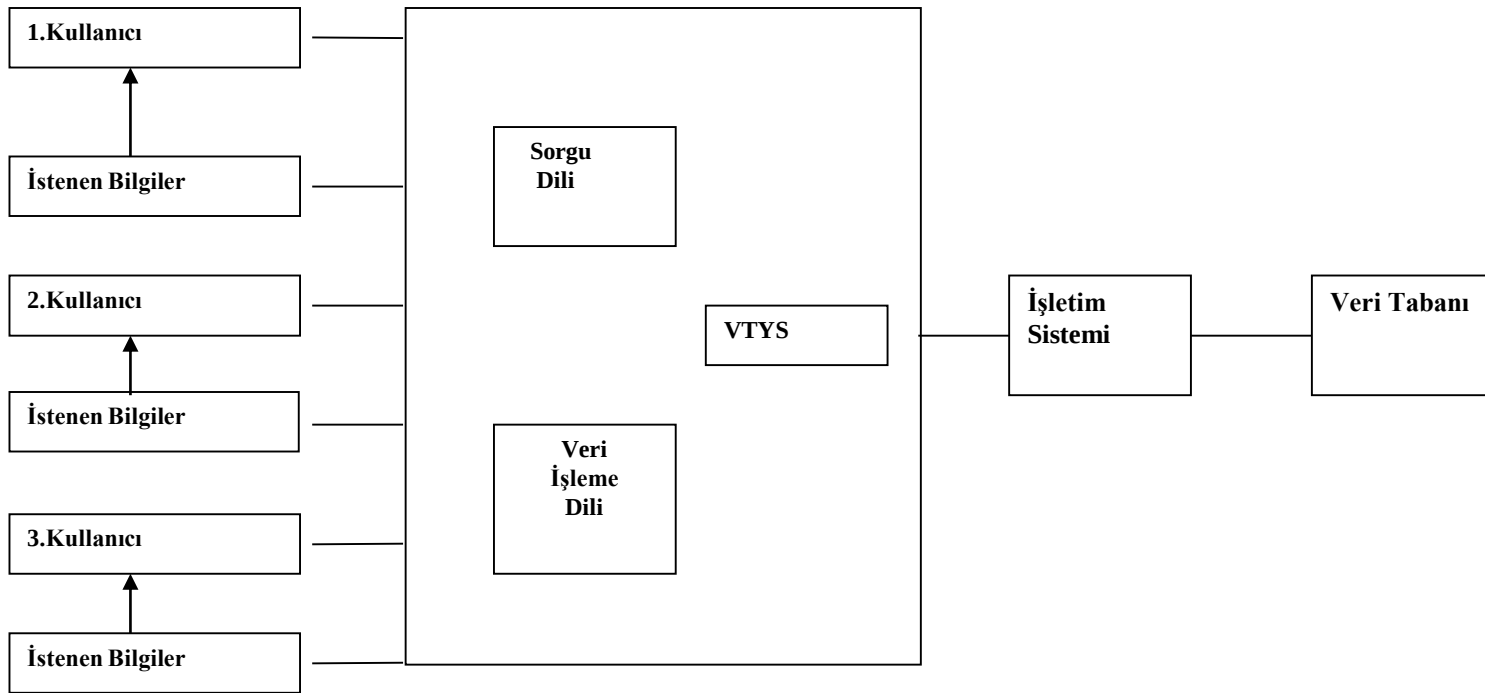


Şekil 1.2: Klasik Dosya Sistemi

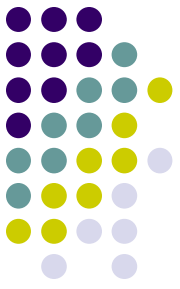
## 1.2.Veritabanı Sistemi



- N tane ayrı program tarafından ihtiyaç duyulan tüm veriler tek bir mantıksal dosya içerisinde yerleştirilmiştir. Böylece klasik dosya sisteminden farklı olarak her uygulama programı için ayrı bir dosya tutulması yerine bir veri tabanı dosyası kullanılmaktadır.



Şekil 1.3: Veri Tabanı Sistemi



## VYS'ler aşağıdaki bilgileri barındırmaktadır

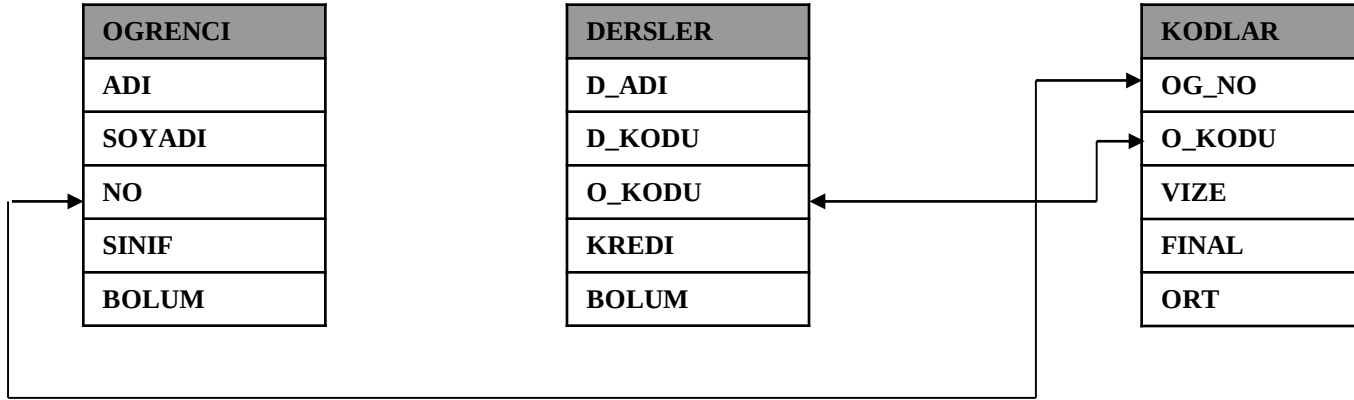
- İlişkili olan veriler (Collection of interrelated data)
- Veriye ulaşmak için gerekli olan yazılımlar kümesi

## Veritabanı Uygulamaları (Database Applications)

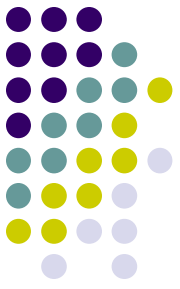
- Bankalar: tüm işlemler / hareketler
- Havayolları: rezervasyonlar, zaman programları
- Üniversiteler: Kayıt, notlar
- Satış: müşteriler, ürünler, alımlar
- Çevrimiçi Perakendicileri: Sipariş Kayıtlar, Kişiselleştirilmiş tavsiyeler.
- Üretim: imalat, stok, siparişler, tedarik ihtiyaçları
- İnsan Kaynakları: personel kayıtları, maaşlar, vergi kesintileri

**Veritabanları hayatımızın her alanında kullanılmaktadır.**

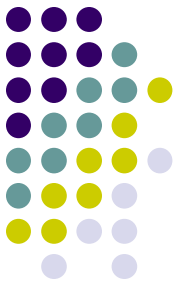
## ÖRNEK VERİ TABANI:



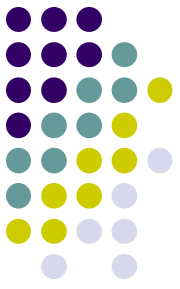
- Bu veri tabanının **tanımlanması** için her dosya içindeki **kayıt yapıları**, kayıtlar içindeki bulunan **alanlar** ve bu alanların **tipleri** belirlenmelidir. Farklı dosyalar içindeki alanlar birbirleri ile ilişkili olabilirler. Ayrıca veri tabanı üzerinde bir işlem yapıldığında birçok dosyaya birden erişim gerekebilir.
- ÖĞRENCİ dosyasından öğrencinin adı, soyadı bilgileri listelenmek istenirse bunun için gerekli komut yazılır.



- **Veritabanı ile ilgili bazı tanımlamalar aşağıda sıralanmıştır. Bunlar:**
- **Veritabanı;** birbirleriyle ilişkisi olan verilerin tutulduğu, kullanım amacına uygun olarak düzenlenmiş veriler topluluğunun mantıksal ve fiziksel olarak tanımlarının olduğu bilgi depolarıdır. Veritabanları gerçekte var olan ve birbirleriyle ilişkileri olan nesneleri ve ilişkilerini modeller.
- **Veritabanı sistemi;** veritabanlarını kurmayı, oluşturmayı, tanımlamayı, işletmeyi ve kullanmayı sağlayan programlar topluluğudur. Veritabanı yönetim sistemi (VTYS) veya database management system (DBMS) ismi ile de kullanılabilir.
- **Veritabanı yönetim sistemi;** yeni bir veritabanı oluşturmak, veritabanını düzenlemek, geliştirmek ve bakımını yaptırmak gibi çeşitli karmaşık işlemlerin gerçekleştirildiği birden fazla programdan oluşmuş bir yazılım sistemidir. Veritabanı yönetim sistemi, kullanıcı ile veritabanı arasında arabirim oluşturur ve veritabanına her türlü erişimi sağlar.

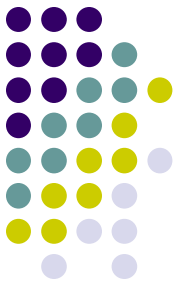


- **Veri tabanı yöneticisi (Database Administrator);** Bir veri tabanı üzerinde her türlü yetkiye sahip olan kişidir. Veri tabanının tasarımı, üzerinde değişiklikler yapma, kullanıcılara izinler verme gibi fonksiyonlara sahiptir. Veri tabanı yöneticisi, gerçek bir kişi olabileceği gibi, bir gruba da yetki verilmiş olabilir.
- **Veritabanının tanımlanması;** veritabanını oluşturan verilerin tip ve uzunluklarının belirlenmesidir.
- **Veritabanını oluşturulması;** veri için yer belirlemesi ve saklama ortamına verilerin yüklenmesini ifade eder.
- **Veritabanı üzerinde işlem yapmak;** belirli bir veri üzerinde sorgulama yapmak, meydana gelen değişiklikleri yansıtmak için veritabanının güncellenmesi ve rapor üretilmesi gibi işleri teslim eder.



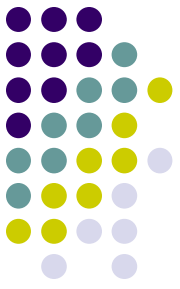
- **Verinin bakım ve sürekliliği;** veritabanına yeni kayıt eklemek, eskileri çağırmak ve gerekli düzenleme, düzeltme ve silme işlemlerini yapmak gibi işlemlerin gerçekleştirilmesini ifade eder. Veritabanı yönetim sistemi aynı zamanda verinin geri çağırılabilmesini de sağlar.
- **Veritabanını genişletme;** kayıtlara yeni veri eklemek ve yeni kayıtlar oluşturmak gibi işlemleri ifade eder.
- **Veritabanı yönetim sistemi programları;** fiziksel hafızaya ve veri tiplerini kullanıcılar adına şekillendirip denetleyen ve kullanıcılarına standart bir SQL ara yüzü sağlayarak onların dosya yapıları, veri yapısı, fiziksel hafıza gibi sorunlarla ilgilenmek yerine veri giriş – çıkışı için uygun ara yüzler geliştirmelerine olanak sağlayan yazılımlardır.



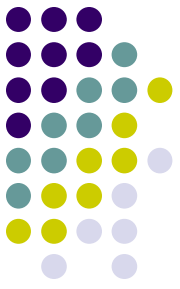


- Veritabanı yönetim sisteminde kullanıcılar, roller ve gruplar vardır ve bunlar verileri tutmak üzere birçok türde nesne ve bu nesnelere erişimlere düzenleme görevi yaparlar. Her bir kullanıcının veritabanı yöneticisi tarafından yapılan tanımlanmış belirli hakları vardır. Bu haklar verilebilir, verilmiş haklar artırılabilir, kısıtlanabilir veya silinebilir. **Örneğin;** bir tablo ya da programı bir kullanıcı kullanabiliyorken bir başkasının hakları kaldırılabilir ve kısıtlanabilir.
- Veritabanı yönetim sisteminde kullanıcılar, roller ve gruplar vardır ve bunlar verileri tutmak üzere birçok türde nesne ve bu nesnelere erişimlere düzenleme görevi yaparlar. Her bir kullanıcının veritabanı yöneticisi tarafından yapılan tanımlanmış belirli hakları vardır. Bu haklar verilebilir, verilmiş haklar artırılabilir, kısıtlanabilir veya silinebilir. **Örneğin** bir tablo ya da programı bir kullanıcı kullanabiliyorken bir başkasının hakları kaldırılabilir ve kısıtlanabilir.





- Bir veritabanından beklenen özellikler, verileri koruması, onlara erişilmesini sağlaması ve başka verilerle ilişkilendirilmesi gibi temel işlemleri yapabilmesidir.
- Veritabanı kullanılarak, verilerden daha kolay yararlanılabilir, istenilen veriye çok kolay erişilebilir, çeşitli sorunların çözümünde yardımcı olacak yeni bilgiler üretilebilir.
- En önemlisi veriler bir merkezde toplanabilir, herkesin bu verilere yetkileri ölçüsünde erişmesi, düzeltmesi, silmesi veya görebilmesi sağlanabilir. Böylece veri girişinde ve veriye erişimde etkinlik ve güvenilirlik sağlanır.



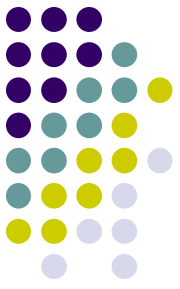
Veritabanı kullanıldığı zaman bir kuruluşa ait tüm operasyonel veriler merkezi bir yerde ve merkezi kontrol altında tutulmuş olur. Bir veritabanı üzerinde birden fazla veritabanı bileşeni vardır; bu bileşenler, saklanmak istenen ham bilginin, belli bir formatta alınarak, veri haline gelmesi işleminde etkin rol oynarlar. Şekil 1.4'te örnek bir veritabanı gösterilmiştir.

| dersler |             |                |
|---------|-------------|----------------|
|         | Column Name | Condensed Type |
|         | [d-adi]     | char(10)       |
| ▶       | [d-kodu]    | int            |
|         | [op-kod]    | int            |
|         | kredisi     | int            |
|         | bolumu      | char(10)       |

| notlar |             |                |
|--------|-------------|----------------|
|        | Column Name | Condensed Type |
| ▶      | [og-no]     | int            |
|        | [op-kodu]   | int            |
|        | vize        | int            |
|        | final       | int            |
|        | but         | int            |

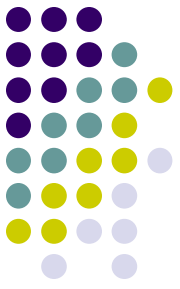
A line connects the [d-kodu] column in the 'dersler' table to the [op-kodu] column in the 'notlar' table. A yellow box labeled 'notlar' is placed over the [op-kodu] column in the 'notlar' table.

Şekil 1.4: Okul Veritabanı



## 1.3.Veritabanı Nerelerde Kullanılır?

- Veritabanı programlama ile birçok proje geliştirilebilir. Bir İngilizce-Türkçe sözlük bu yolla kolayca yazılabilir. Bir kütüphane takip otomasyonu, bir hastane otomasyonu, muhasebe programları ve daha birçok otomasyon programı temelde veritabanı projesidir.
- Günlük hayatta veritabanı programlama telefon şirketleri tarafından yoğun olara kullanılır. Konuşmaların süreleri ay boyunca veritabanlarında saklanır ve ay sonu geldiğinde istemci programlar tarafından her bir abonenin telefon faturası teker teker hesaplanır.



# 1.4. Veritabanı Yönetim Sistemlerinin Sınıflandırılması

- 1.4.1. Veri Modeline Göre Veritabanı Yönetim Sistemleri

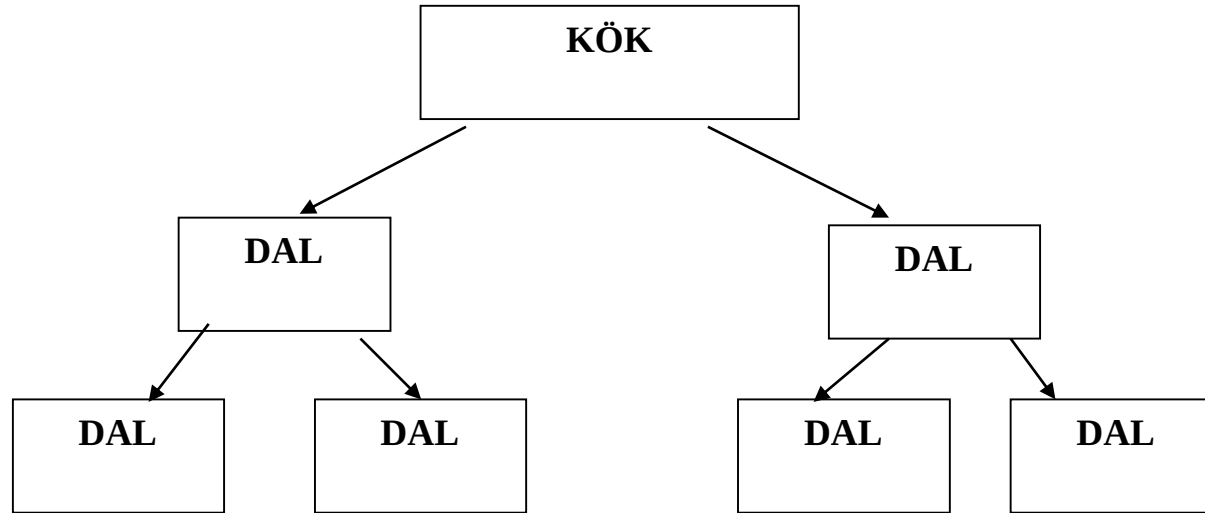
Yapısal olarak bütün veri tabanları bir değildir. Veritabanları verileri saklama ve onlara erişme bakımından farklı tiplere ayrılır.

- **Hiyerarşik Veritabanları**
- **Ağ Veritabanları**
- **İlişkisel Veritabanları**
- **Nesneye Yönelik Veritabanları**
- **Kullanıcı Sayısına Göre Veritabanı Yönetim Sistemleri**



## 1.4.1.1. Hiyerarşik Veritabanları

- Veritabanları için kullanılan ilk modeldir. Bu veritabanı tipi, ana bilgisayar ortamlarında çalışan yazılımlar tarafından kullanılmaktadır. IBM tarafından çıkarılan IMS bu türde en çok kullanılan yazılımdır. Hiyerarşik veritabanları uzun bir geçmişe sahip olmalarına rağmen, PC ortamına uyarlanmış olanı bulunmamaktadır. Hiyerarşik veritabanları, bilgileri bir ağaç (tree) yapısında saklarlar.
- Kök (root) olarak bir kayıt ve bu köke bağlı dal (branch) kayıtlar hiyerarşik veritabanının yapısı Şekil 1.5'te gösterildiği gibidir:

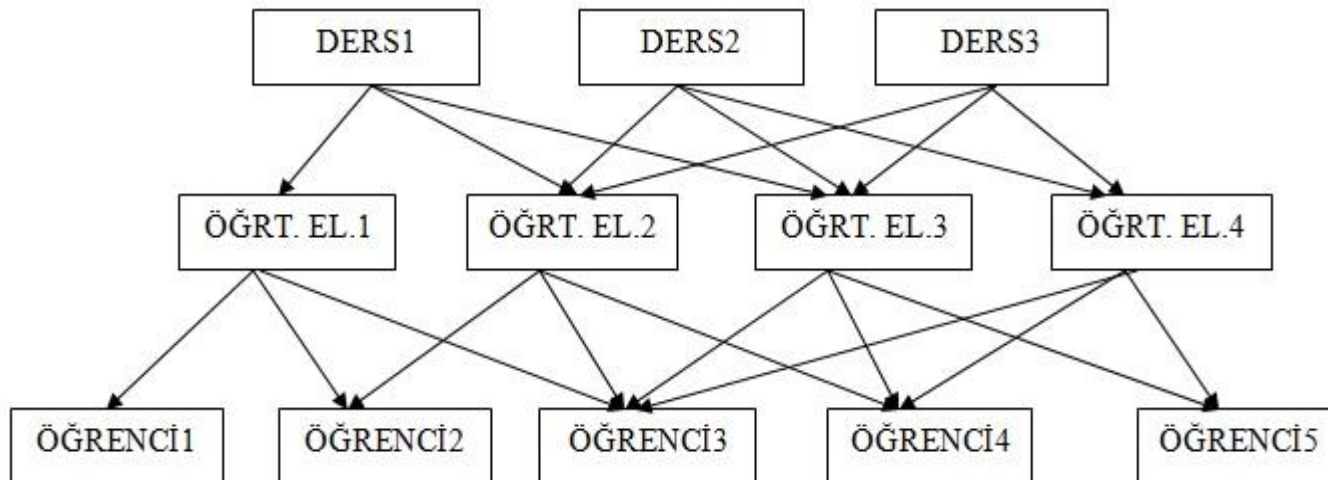


**Şekil 1.5:** Hiyerarşik Veritabanı Yapısı

## 1.4.1.2. Ağ Veritabanları



- Hiyerarşik veritabanlarının yetersiz kalmasından dolayı; 1960'ların sonunda toplanan Conference on Data System Languages (CODASYL) isimli bilimsel konferanstaki veritabanı çalışma grubu (Database Task Group – DBTG) bilim adamlarının ortak çalışması sonucu ortaya konulmuş bir veri tabanı türüdür.
- Ağ veritabanları verileri, ağaçların daha da gelişmiş hali olan graflar ( ağacın kendisi de özel bir graftır) şeklinde saklarlar ve bu yapı en karışık yapılardan biridir. Aşağıdaki diyagramda da görüldüğü gibi her öğrenciden birden fazla öğretim elemanına, her öğretim elemanı birden fazla derse bağlı olabilir.

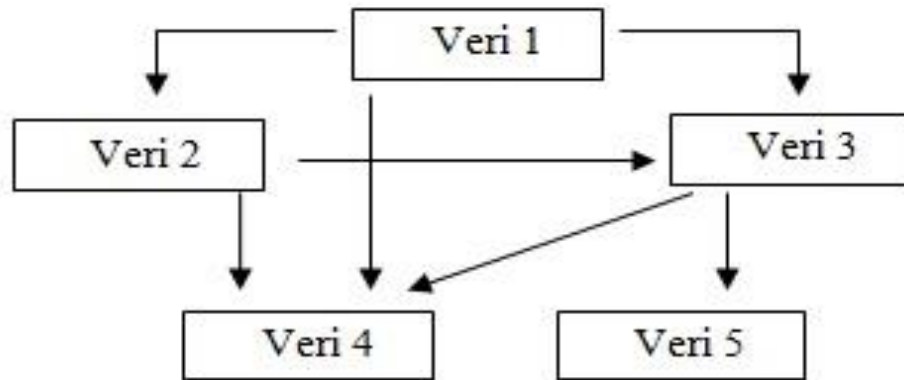




### 1.4.1.3.İlişkisel Veritabanları

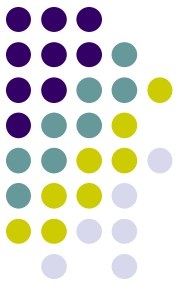


- 1970'lerin başında E.F.Codd tarafından geliştirilmiştir. Bu sistemde veriler tablo şeklinde saklanırlar. Bu veritabanı yönetim sisteminde; veri alışverişi için özel işlemler kullanılır. Bu işlemlerde tablolar operandlar olarak kullanılır. Tablolar arasındaki matematiksel bağıntılarla (ilişkilerle) temsil edilen ilişkiler belirtilir. Günümüzde hemen hemen tüm veri tabanı yönetim sistemleri ilişkisel veri modelini kullanırlar.
- Bu model, matematikteki ilişki teorisine (“the relational theory”) dayanır. İlişkisel veri modelinde (Relational Data Model) veriler basit tablolar halinde tutulur. Tablolar, satır ve sütunlardan oluşur. Sütunlar bilgi alanlarını, satırlar ise bilgilerin içeriğini belirlerler.

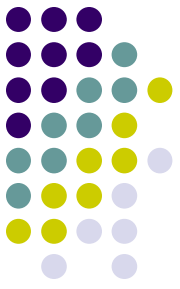


**Şekil 1.6:** İlişkisel Veritabanı Yapısı

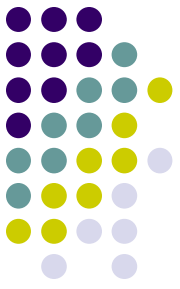
## 1.4.1.4. Nesneye Yönelik Veritabanları



- Günümüzde pek çok kelime işlemci ve hesap tablosu programlarında kullanmaya alışılan nesneler artık veritabanı yönetim sistemi yazılımlarında da kullanılmaktadır. Yüzde yüz nesneye yönelik bir yazılımın tamamen nesne temelli çalışması ve yazılımın mutlaka nesneye yönelik bir dilde yazılmış olması beklendiğinden; nesneye yönelik veritabanları gerçek anlamda bir nesneye yönelik yazılım değildirler.
- Nesneye yönelik veritabanı, C + + gibi nesneye dayalı bir dille (OOPL) yazılmış olan ve yine C + + gibi nesneye dayalı (OOPL) bir dille kullanılan veritabanı anlamına gelir.

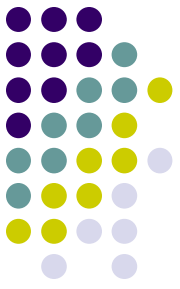


- Günümüz teknolojisinde yüzde yüz nesneye yönelik bir veritabanı yaygın olarak kullanıma sunulmamış olmasına rağmen nesneye yönelik veritabanlarının bazı üstünlükleri olacağından söz edilmektedir. Nesneye yönelik veritabanlarının ilişkisel veritabanlarına göre sahip olması gereken üstünlükleri şunlardır:
  1. Nesneler, bir tabloda yer alan bir kayıttan çok daha karmaşık bir yapıya sahiplerdir ve daha esnek bir yapıda çok daha kullanışlı düzenlenebilirler.



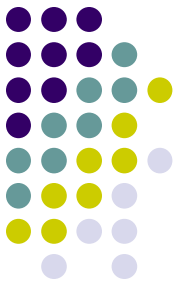
2. Nesneye dayalı bir veritabanında, yapısı gereği arama işlemleri çok hızlı yapılabilir. Özellikle büyük tablolarla uğraşırken ilişkisel veritabanlarından çok daha hızlı sonuca ulaşırlar. Ancak çalışma mantığı tümüyle değişir.
- Tüm bu özellikler tamamen nesneye yönelik olan veritabanları için geçerlidir. Bazı ilişkisel veritabanları ile çalışan yazılımlarda da nesnelerin bazı özellikleri kullanılırlar. Ama nesneye yönelik veritabanı bunu kendini ilişkisel veritabanı kurallarına uydurarak gerçekleştirebilir.

## 1.4.1.5. Kullanıcı Sayısına Göre Veritabanı Yönetim Sistemleri



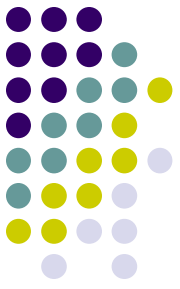
- Tek kullanıcı ve çok kullanıcı olmak üzere ikiye ayrılır. Günümüzde PC tabanlı VTYS'ler de dâhil olmak üzere tek kullanıcı veritabanı yönetim sistemi pek kalmamıştır. Çok kullanıcı ise sınırsız kullanıcı veya belirtilen sayı kullanıcı olarak iki kategoride satılır ve kullanılırlar.

# 1.5. Klasik Dosya Sisteminin Sakıncaları



- Veri tekrarları, klasik dosya sisteminde, ayrı veri çeşitli dosyalar içinde tekrar tekrar yer alabilmektedir.
- Çoklu güncelleme, aynı veri birden fazla dosyada tekrar ettiği için bir dosyada güncellenip, diğer dosyalarda güncellenmemesi veri bütünlüğünün kaybolmasına neden olur. Bunun sonucunda birbiri ile uyumsuz ve çelişen raporlar üretilmesi söz konusu olabilir.
- Bellek hacmi israfı, aynı verinin birden fazla dosyada tekrar tekrar kullanılması, kullanılan bellek alanın da israfa yol açacaktır.
- Erişim dili, klasik dosya sisteminde kullanılan programa göre dosyaya erişim farklılıklar gösterebilir. Standart bir dil kullanımı söz konusu değildir.

# 1.6. Veritabanı Sisteminin Yararları



## Verinin tekrarlanması önler.

- Veritabanı sistemleri alt sistemler arasında ilişki kurulması ve birçok uygulamada verilerin aynı veritabanı içerisinde ortak olarak tasarlanmasını öngörür.

## Verilerin tutarlı olmasını sağlar.

- Veri bütünlüğü(data integrity), verinin doğruluğunu ve tutarlılığını ifade etmektedir. Veri girişlerine kısıtlar konularak sadece istenilen aralıkta değer girişi sağlanabilir.

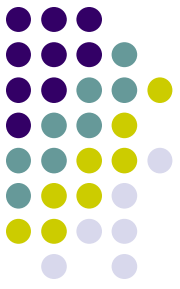
## Aynı andaki erişimlerde tutarsızlıkların ortaya çıkmasını önler.

- Veritabanı uygulamalarında, veritabanı nesneleri başka uygulamalar ve farklı kullanıcılar tarafından paylaşılabilir.

## Verilerin güvenliğini sağlar.

- Her kullanıcının erişeceği veriler ayrı ayrı tanımlanabilir. Yetkiler ve kısıtlamalar ile istenilen kullanıcı erişim ayarları gerçekleştirilir.

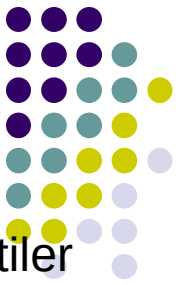
## 1.7 Veritabanı Sisteminin Sakıncaları



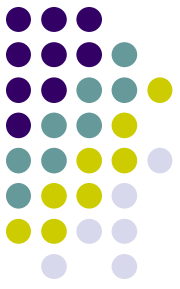
- Veri tabanı sisteminin kurulumu ve bakımı klasik dosya sisteminden daha pahalı ve zordur.
- Veri tabanı sistemi içinde, bazı bileşenler iyi tasarlanmadığı durumlarda, bir bütün olarak ciddi sistem başarısızlıklarına yol açabilir.



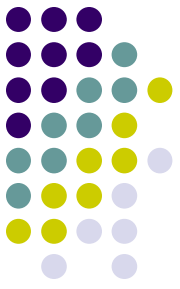
## 1.8.Bilinen VTYS Programları



- MS SQL Server, bir orta ve büyük ölçekli VTYS'dir. ANSI SQL'e eklentiler yazmak için T-SQL'i destekler.
- Oracle, daha çok yüksek ölçekli uygulamalarda tercih edilen bir VTYS'dir. ANSI SQL'e eklentiler yapmak için PL/SQL geliştirilmiştir.
- Sysbase, bir orta ve büyük ölçekli VTYS'dir. ANSI SQL'E eklentiler yazmak için T-SQL komutlarını destekler. Ülkemizde daha çok bankacılık ve kamusal alanlarda tercih edilmektedir.
- Informix, bir orta ve büyük ölçekli VTYS'dir.
- MySQL, genellikle Unix – Linux temelli Web uygulamalarında tercih edilen bir VTYS'dir. Açık kod bir yazılımdır. Küçük – orta ölçeklidir. Özellikle Web için geliştirilmiş bir VTYS'dir denilebilir.
- Postrage SQL, buda MySQL gibi açık kod bir VTYS'dir.



- MS Access, çoklu kullanıcı desteği yoktur. İşletim sisteminin sağladığı güvenlik seçeneklerini kullanır. Bunun yanında belli sayıda kayda kadar (1000000 civarı) ya da belli bir boyutun (yaklaşık 25 MB) altına kadar bir sorun çıkartmadan kullanılabilecek bir küçük ölçekli VTYS'dir.
- Advantage, Türk programcılar tarafından geliştirilen bir orta ve büyük ölçekli VTYS'dir.
- DB/2, IBM'in framework'lere yönelik büyük ölçekli VTYS'dir.
- Bunların dışında daha birçok VTYS programı mevcuttur. VTYS'lerin Avrupa genelindeki yaklaşık olarak pazar payları Tablo 1.1 'de gösterilmiştir. En büyük Pazar payı IBM (DB/2) ile Oracle arasındadır. Hemen arkasından MS SQL Server, Informix ve Sybase gelmektedir. Yeni başlayanlar için; hangi VTYS'yi öğrenmem en iyisi olur sorusunu yanıtlamak gerekebilir. Ülkemizde insan kaynakları açısından en çok kalifiye eleman aranan VTYS Oracle ve arkasından da MS SQL Server gelmektedir.

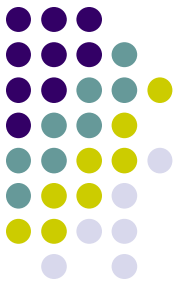


| VTYS      | Yüzdesi (%) |
|-----------|-------------|
| IBM       | 37.8        |
| Oracle    | %26.3       |
| Microsoft | 15.4        |
| Informix  | 3.2         |
| Sybase    | 3           |
| Diğerleri | 14.3        |

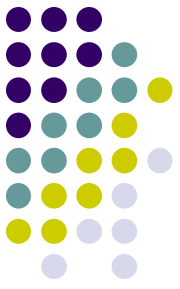
**Tablo 1.1:** VTYS'lerin Pazar Payları (Gartner, 2001)

- VTYS'lerin birçoğu ANSI SQL'in karşılayamadığı durumlarda kullanılmak üzere ek programlama komutları barındırırlar. Bu iş için MS SQL Server ve Sybase SQL Server Transact SQL (T-SQL) denilen komut takımlarını içerir. Oracle ise PL/SQL ile bu işe çözüm getirmiştir. Bu diller sayesinde, Stored (saklı prosedürler), Trigger, Fonksiyon gibi veritabanları için vazgeçilmez olan nesneler yazılabilmektedir.

## 1.9. Proje ve VTYS Arasındaki İlişki



- Herhangi bir veritabanı programında çalışmaya başlanılmadan önce, yapılacak işe uygun veritabanı tasarımı yapılmalıdır. Bu, işin en önemli aşamasıdır. Başlangıçta iyi tasarlanamayan bir veritabanı, ilerde geriye dönüşü olmayan verimsiz bir bilgi yığınınına dönüşebilir.
- En basit hali ile veritabanı tasarımında; hangi tabloların olacağı, bu tablolarda hangi alanların olacağı, tablolar arasındaki alan ilişkilerinin neler olacağı ve alanlara ait özelliklerin tanımlanması yapılır. Alan özelliklerinde alan adı, alan tipi, alanın uzunluğu, alanın varsayılan değeri, bu alana yazılacak verilerin geçerlilik koşullarının başlangıçta tasarlanması gerekir.



- Bir projede hangi veritabanının seçileceği, projenin büyüklüğü ile ilgili bir karardır. Aşağıdaki sorulara verilecek cevaplar projenin büyüklüğü konusunda karar vermede yardımcı olurlar.
- Projede kaç tablonun kullanılacağı,
- Her bir tabloda en fazla kaç satırın yer alabileceği (tablodaki bilgi sayısıdır),
- Projeye aynı anda en fazla kaç kullanıcının bağlanacağı,
- Projede günlük kaç transaction (insert-update-delete) gerçekleştirileceği,
- Proje en fazla ne kadar yer kaplayacak, ne kadar bir veritabanı dosyasına ihtiyaç duyulacağı,
- Proje için güvenliğin ne derece önemli olduğu.
- Ancak bir VTYS kullanılarak proje geliştirilecekse, hangi veritabanının seçileceği ile birlikte, veritabanının hangi sürümlerinin kullanılacağı ve hangi donanımlar üzerinde çalıştırılacağı da çok önemlidir.

# 2.VERİ ve VERİ MODELLERİ

BÖLÜM 2

## 2.1. Modelin Tanımı

- Model kelimesi; isim, sıfat ve fiil olarak kullanılmaktadır. İsim olarak “model”, bir mimarın, bir binanın küçük ölçekli modeli veya bir fizikçinin bir atomun büyük ölçekli modelini oluşturması gibi bir temsili ifade eder. Sıfat olarak “model” , “model uçak”, “model öğrenci”, “model ev” ifadelerinde olduğu gibi mükemmeliyetin veya idealin ölçüsünü ifade eder. Fiil olarak “model” ise, bir şeyin nasıl olduğunu ispat etmek, açıklamak, göstermek anlamındadır. Bilimde simgesel model, benzetim modeli ve sembolik model kavramları vardır.

- Simgesel modeller, durumların büyük veya küçük ölçekli temsilleridir. Gerçek şeylerin uygun özelliklerini temsil ederler. Şekilleri, temsil ettikleri şeyler benzerler. Yol haritaları, hava fotoğrafları bu tip modellere örnek verilebilir. Benzetim modelleri, bazı durumlarda ise; haritada yükselti, yol genişlikleri gibi özellikleri belirtmek gerekebilir. O zaman, renkler ve kontur çizgileri gibi bir takım açıklayıcı özelliklere ihtiyaç duyulur. Bu tip modeller Benzetim Modelleri olarak isimlendirilir. Sembolik modellerde, temsil edilen şeylerin özellikleri sembollerle ifade edilir.



- Böylece bir grafik ile gösterilen ilişki (benzetim modeli), bir eşitlik olarak da ifade edilebilir. Bu tip modellere Matematiksel Modeller de denilmektedir. Bu üç tip modelden benzetim modeli, soyut ve geneldir. Matematiksel model ise en soyut ve en genel modeldir. Üzerinde düzenleme yapılabilmesi daha kolaydır. Simgesel modellerin ise anlaşılması diğerlerine göre daha kolaydır. Bilişim sistemlerinin oluşturulması için kullanılan veri modelleri, benzetim modelleri ve sembolik modellerdir.

- Bir bilişim sistemi kullanıcısı, özellikle bir karar verici, kendisini sonsuz denebilecek kadar bilgi karşısında bulur. Bir bilişim sistemi modeli, gerçek bilgi kümesini oluşturur ve onun daha basit bir şeklidir. Bu şekil, işlenebilmeye imkân verir ve bunu kullanarak elde edilen çözüm veya cevap, gerçek hayatta uygulanmaya çalışılır. Model, var olan bilgi yığınınına bir düzen getirmeyi, hatta bir yapı oluşturmayı amaçlar. Tek bir model yoktur. Var olan bilgi yığınınına, uygulanan farklı modeller doğal olarak farklı modeller getirir.

## 2.2.Veritabanlarının Korunması

- Yanlış verinin iki türlü kaynağı olabilir: Programlama hataları, klavyeden hatalı giriş nedeniyle oluşan yanlışlıklar ve veritabanı programının kötü niyetli kullanımı.
- Veritabanlarının korunması iki başlık altında incelenebilir: Bunlar;
- veri güvenliği ve
- veri bütünlüğünün sağlanmasıdır.

## 2.2.1 Veri Güvenliği

- Veri güvenliği veri tabanını ve verilerini yetkisiz kullanımlara karşı korumadır. Bu verinin istenmeyen şekilde değiştirilmesine veya zarar görmesine ve yetkisiz kullanımlara engel olmaktadır. Bunu sağlamak için bazı genel teknikler geliştirilmiştir.

- Çok kullanıcılı ortamlarda farklı yetkilere sahip kullanıcılar vardır. Farklı yetkilere sahip kişilerin, veritabanında ulaşabilecekleri veriler farklıdır. Örneğin, bilgisayara veri girişi yapan bir memurun, kurumun muhasebe kayıtlarına, muhasebe müdürü kadar yetkiliymiş gibi girerek değişiklikler yapması engellenmelidir. Bu amaçla, hangi kullanıcıların hangi yetkilerinin olduğu ve bu yetkilerini kullanabilmek için gerekli şifreler daha önceden tespit edilmelidir.

- Şifre sisteminin yeterli olmadığı durumlarda, verinin fiziksel koruma altına alınması gerekir. (Yangın veya hırsızlığa karşı verinin yedeklenmesinin yapılması gibi.)
- Sistemde hangi kullanıcının hangi yetkilere ve haklara sahip olduğu ve neler yapabileceğinin önceden belirlenmiş olması gerekir.

- Özellikle veritabanının sorgulanmasında güvenlik problemleri ortaya çıkmaktadır. Hangi tür kullanıcının, hangi sorgu tiplerini sisteme yöneltebileceğinin daha önceden tespit edilmesi gerekmektedir. Fakat yukarıda bahsedilen önlemlerden hiçbiri tam bir koruma sağlamaz. Bu yüzden, birden fazla önlem kullanılarak güvenlik artırılabilir.

## 2.2.2. Veri Tekrarı ve Veri Bütünlüğü

- Bir veritabanı yönetim sisteminde farklı veri dosyalarında; isim, adres, numara gibi bilgilerin bulunması gerekebilir. Örneğin, hem personel bilgilerini içeren bir veritabanı dosyasında, hem de ödenecek maaşların yapılacağı ödemelerin bulunduğu başka bir veri dosyasında, ödenen maaş bilgilerinin yer alması gerekir. Bazen aynı verinin birden fazla veri dosyasında bulunması gerekebilir. Bu durum, veri tekrarı olarak ifade edilmektedir.



- Böyle bir durum, veri bütünlüğünün bozulmasına neden olur. Veri üzerinde yapılacak değişiklik, silme, ekleme gibi işlemlerin, o verinin bulunduğu bütün dosyalarda gerçekleştirilmesi gerekir. Bütün dosyalarda gerçekleşmezse, veritabanında uygun olmayan veri ile çalışılmış olunur. Özellikle çok kullanıcıli ortamlarda veri bütünlüğü çok önemlidir. Veri bütünlüğünün bozulmasının bir sebebi de veri tekrarıdır. Bir başka sebep de, verinin zayıf geçerlilik kontrolüdür. Bunun sebepleri de şu şekilde sıralanabilir:

- 
- Veri güvenliğinin yetersiz oluşu,
  - Veritabanının zarar görmesi durumunda kurtarma yöntemlerinin yetersiz oluşu,
  - Uzun kayıtların idaresinin zorluğu,
  - Değişikliklerin esnek olmaması,
  - Programlama ve bakım masraflarının yüksek olması,
  - İnsandan kaynaklanan hatalar.

- Günümüzde kullanılan çeşitli veritabanı yönetim sistemi programları, yukarıda sayılan bütün problemlerin pek çoğunu halledebilecek çözümler üretmişler ve bunları kolay kullanılabilir hale getirmişlerdir. Böylece kullanıcıların veritabanı oluştururken, ayrıca bu problemler için önlem almalarına gerek kalmamıştır.

## 2.3. Veri Modeli

- Bir veri modeli, verinin hangi kurallara göre yapılandırıldığını belirler.

### 2.3.1. Yapılar

- Soyutlama, küme ve ilişki veri yapılarının temel unsurlarıdır. Detayları gizleme ve genel üzerinde yoğunlaşma yeteneği olan soyutlama veriyi yapılandırma ve görüntüleme işlemini yapar ve veri kategorilerini elde etmek için kullanılır. Yapılar düzgün şekilde tanımlanmış veri gruplarıdır. Kendisi de aynı zamanda bir küme olan ilişki, iki nesne arasındaki ilişkiyi gösteren bir tip olarak kümelerin toplanmasını ifade eder. Örneğin, öğrenci ve okul arasında bir not ilişkisi vardır.

- Veri yapısı oluşturulurken, verideki nesneler ve onlar arasındaki ilişkiler tablo ile temsil edilir. Veritabanlarında uygulanabilecek genel kayıt ilişkilendirme tipleri şu şekilde sıralanabilir:
- **Birebir ilişkiler (one - to - one relationships)**, aralarında bir ilişki olan iki tablo arasında, tablolardan birindeki asıl anahtar alanın kayıt değerinin, diğer tablodaki sadece bir kayıta karşılığının olması durumunu gösteren ilişki tipi.

- **Tekil çoklu ilişkiler (one - to - many relationships)**, aralarında bir ilişki olan iki tablo arasında, asıl anahtar alanın kayır değerinin, diğer tablodaki birden fazla kayıta karşılığının olması durumunu gösteren ilişki tipi. Örneğin, bir öğrencinin birden fazla almış olduğu derse ve bu derse ait vize, final ve sınav sonuçları gibi. Bir öğrenciye karşılık gelen birden fazla ders notu.

- **Çoğul tekli ilişkiler (many - to - one relationships)**, aralarında bir ilişki olan iki tablo arasında, tablolardan birindeki bir kaydın değerinin, asıl anahtar alanın olduğu diğer tabloda, birden fazla kayıta karşılının olması durumunu gösteren ilişki tipi.
- **Çoklu ilişkiler (many - to - many relationships)**, aralarında bir ilişki olan iki tablo arasında, tablolardan herhangi birindeki herhangi bir kaydın, diğer tablodaki birden fazla kayıt ile ilişkilendirilebildiği ilişki tipi.

## 2.3.2. Kısıtlar

- Veriler üzerindeki sınırlamalara kısıt adı verilir. Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılır. Örneğin, “öğrenciye ait not bilgisinin 0 – 100 arasında olması gibi.

## 2.3.3. İşlemler

- İşlemler, bir veritabanı durumundan, bir başka veritabanı durumu elde etmek için yapılan işlemlerdir. Bunlar, verinin çağırılması, güncellenmesi, eklenmesi veya silinmesi ile ilgili işlemlerdir. Bütünlük mekanizması, toplam fonksiyonları (istatistiksel fonksiyonlar da bunlar arasındadır), veriye ulaşım kontrolleri gibi genel işlemler vardır. CODASYL tarafından yayılan bu mekanizmalara veritabanı yöntemleri denir.



## 2.4. Başlıca Veri Modelleri

Veri modellemesi yapmak amacıyla farklı durumlara uygun olan ve birbiriyle farklı özellikler taşıyan pek çok veri modeli vardır. Veri modelleri aşağıdaki gibi sınıflandırılabilir:

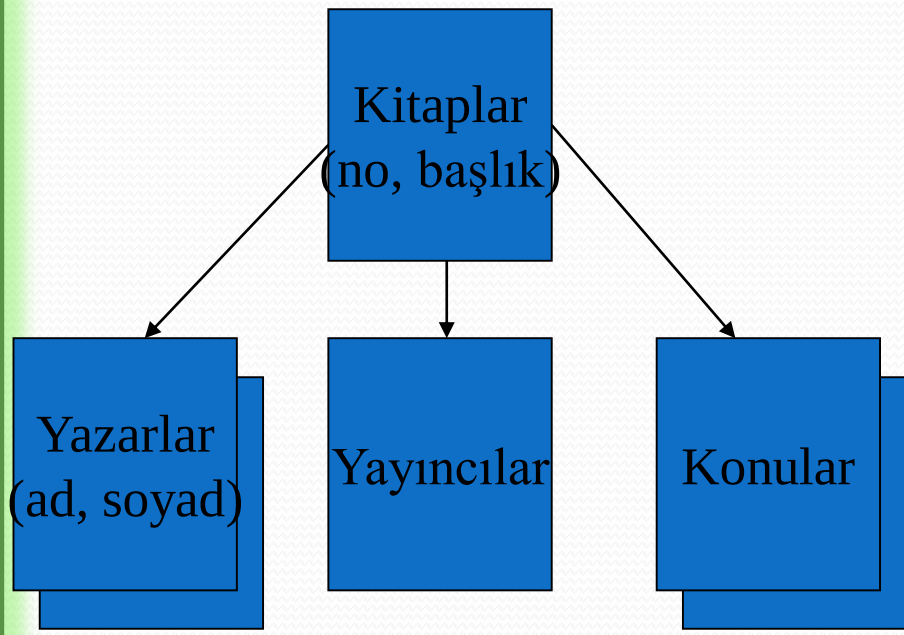
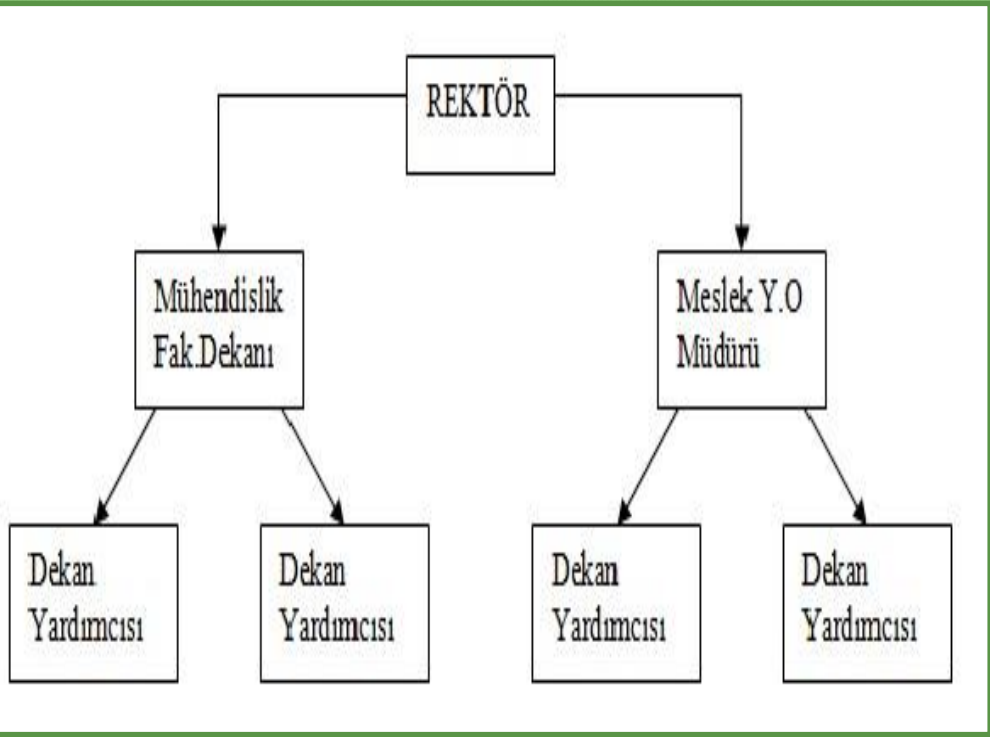
- Basit Veri Modelleri
- Hiyerarşik Veri Modelleri
- Şebeke Veri Modelleri
- Geliştirilmiş Veri Modelleri
- Varlık-İlişki Veri Modelleri ( Vİ Modeli)
- İlişkisel Veri Modelleri
- Nesne Yönelimli Veri Modelleri

## 2.4.1. Basit Veri Modelleri

- Bilgisayarlarda veri işleme ihtiyacının ortaya çıkmasıyla, dosyalama sistemleri oluşturmak amacıyla kullanılmaya başlanan Hiyerarşik ve Şebeke veri modelleridir.

## 2.4.2. Hiyerarşik Veri Modelleri

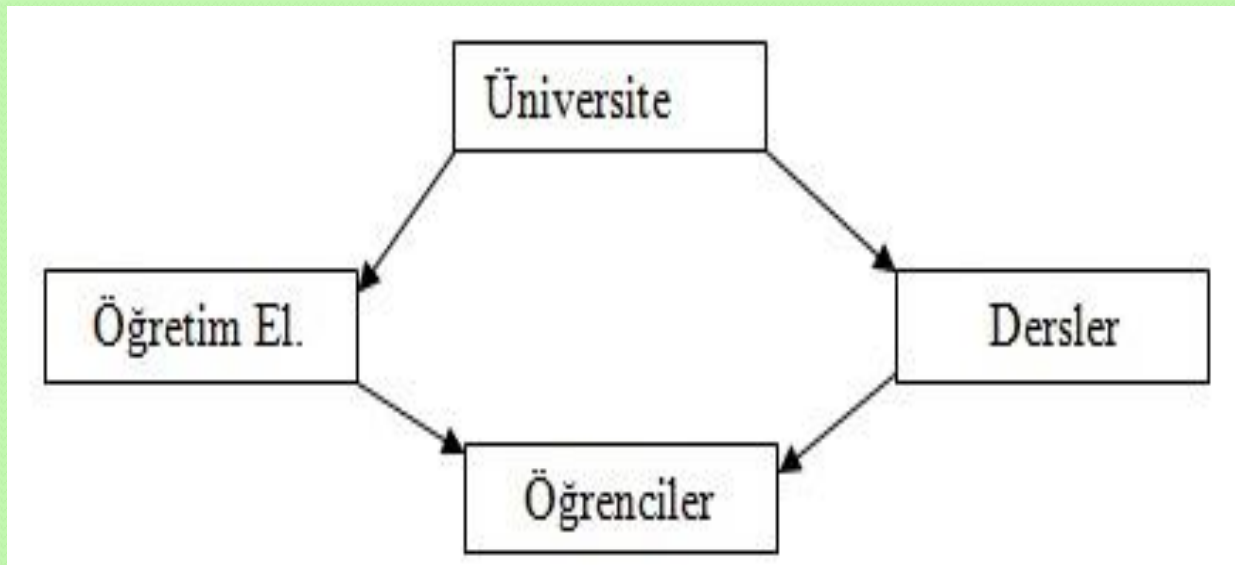
- Çoklu ilişkileri temsil edebilmek için, varlık tiplerinin gereksiz veri tekrarı yapmadan her ilişki için ayrı ayrı tanımlanmasına Hiyerarşik veri modeli denilir. Bu model, bir ağaç yapısına benzer. Model içerisindeki herhangi bir düğüm, altındaki  $n$  sayıda düğüme bağlanırken, kendisinin üstünde ancak bir düğüme bağlanabilir. Hiyerarşik yapının en tepesindeki düğüm noktasına kök denir ve bu düğümün sadece bağımlı düğümleri bulunur. Bu veri yapısını gösteren grafiğe de hiyerarşik tanım ağacı denir.(Şekil 2.1)



Şekil 2.1: Hiyerarşik Tanım Ağacı

## 2.4.3. Şebeke Veri Modelleri

- Tablo ve grafiklerden oluşan veri modeli Şebeke Veri Modelidir. Tablolar grafikteki düğümler olup varlık tiplerine karşılık gelirler. Grafiğin okları ise ilişkileri temsil eder. Özellikleri, 1971 yılında DBTG – CODASYL tarafından belirlenmiştir. Kayıt tipi ve bağlantı olmak üzere iki ayrı veri yapılandırma aracı vardır. Kayıt tipleri varlık tiplerini, bağlantılar ise ilişki tiplerini belirler. Bu yapıyı gösteren grafiğe de veri yapısı grafiği adı verilir.(Şekil 2.2)



**Şekil 2.2:** Şebeke Veri Yapısı Grafiği

- Şebeke içinde bir eleman, herhangi başka bir elemana bağlanabilir. Hiyerarşik yapılardan farklı olarak, şebeke yapılarında bağlantı açısından herhangi bir sınırlama yoktur. Şebeke veri modelleri, düğümler arasında çoklu ilişkiler kurulamadığı için, kısıtlı bir veri modeli olarak kabul edilir. Hiyerarşik veri modelleri ise, daha da kısıtlı bir veri modelidir. Şebeke veri modelinde kullanılan işlemler, ilişkisel veri modelinde kullanılan işlemlerin benzeridir. Fakat şebeke veri modellerinde bağlantılar tarafından belirlenmiş ilişkiler dışında, kayıt tipleri arasında ilişki belirlenemez.

## 2.4.4. Geliştirilmiş Veri Modelleri

- 1960 ve 1970’li yıllarda hiyerarşik veri modeli üzerine geliştirilmiş VTYS, sonra şebeke veri modeli ile çalışan VTYS yaygı kullanım alanı bulmuştur. 1970’li yıllarda gelişmesini tamamlamış olan ilişkisel veri modeline dayalı VTYS 1980’li yıllarda ticari kullanıma girerek çok hızla yaygınlaşmaya başlamıştır.
- Geliştirilmiş veri modelleri, Varlık – İlişki Veri Modelleri, İlişkisel Veri Modelleri ve Nesne Yönelimli Veri Modelleri olarak sıralanabilir.



## 2.4.5.Varlık-İlişki Veri Modelleri (E-R Modeli)

- Varlık-ilişki modeli, ya da kısaca E-R modeli (Entity-Relationship model) 1976 yılında P.P. Chen tarafından geliştirilen bir modeldir.
- Bugüne kadar varlık-ilişki modeline dayalı hiçbir VTYS geliştirilmemiştir.
- Buna karşılık varlık-ilişki modeli, VTYS'den bağımsız veri çözümlemede ve semantik veri modellemede en çok kullanılan modeldir.

- Bu model kullanılarak önce;
  - VTYS'den bağımsız olarak veriler çözümlenir,
  - Veri modellemesi yapılır,
  - Veriler ve veriler arası ilişkilerin anlamları ve özellikleri incelenerek E-R çizelgeleri oluşturulur;
  - Kllanılacak VTYS belirlenir
- sonra da E-R çizelgeleri bu sistemin veri modeline dönüştürülerek veri tabanı şemaları oluşturulur.

- Bir veritabanı uygulamasında hakkında tanımlayıcı bilgi saklanabilen her şey varlık olarak kabul edilir. Varlık, bağımsızdır ve tek başına tanımlanabilir. Bir varlık, ev, öğrenci, araba gibi bir nesne ya da futbol maçı, tatil, satış gibi olaylar olabilir. En anlamlı şekilde kendi öznitelikleri tarafından temsil edilir.
- Örneğin, bir EV; öznitelikleri olan ADRES, STİL, RENK ve MALZEME ile tanımlanabilir. Eğer bir özneliğin kendisi tanımlayıcı bilgi içeriyorsa, onu varlık olarak tanımlamak gerekir. Örneğin, eğer evin malzemesi hakkında ek bilgi depolamak gerekiyorsa MALZEME'Yİ de varlık olarak sınıflamak gerekir.

## 2.4.6. İlişkisel Veri Modelleri

- İlişkiler ve onların temsilleri olan tablolardan oluşan ilişkisel veri modeli ilk olarak 1970 yılında Codd tarafından ortaya atılmıştır. İlişkisel veri modellerinde kullanılan tek yapılandırma aracı ilişkidir. Tablo-2'de ki örneklerde büyük harflerle yazılan ifadeler ilişki isimlerini, parantez içindeki ifadeler de tanım kümesi isimlerini göstermektedir.

**Tablo 2.1:** İlişkisel veri Modeli Tanım Kümesi

**UNIVERSITE** (UniversiteKodu, UniversiteAdi, Adres)

**OGRETIMELEMANI** (OgrKodu, OgrAdi, OgrSoyadi, Bolum, Brans)

**OGRENCI** (OgrenciNo, OgrenciAdi, OgrenciSoyadi, Bolum, Adres)

**DERSLER** (DersKodu, DersinAdi, Kredisi, Yariyil)

**OGRENCINOTLAR** (OgrenciNo, DersKodu, Vize, Final)

Yukarıdaki satırlar, basit bir Üniversite veritabanının ilişkisel şemasını göstermektedir. İlişkisel şema, ilişki isimlerinin ve karşılık gelen tanım kümesi isimlerinin listesidir. Varlık tiplerini belirlemekte kullanılır.

**Tablo 2.2: İlişkisel Şema**

| UniversiteKodu | UniversiteAdi          | Adres    |
|----------------|------------------------|----------|
| 3300           | Fırat Üniversitesi     | Elazığ   |
| 0600           | İnönü Üniversitesi     | Malatya  |
| 3400           | Hacettepe Üniversitesi | İstanbul |

- İlişkisel şema listesini oluşturan her bir satır, bir tablo olarak temsil edilir. Tablonun sütunları öznitelik olarak isimlendirilir. Örneğin Tablo 2.2’te gösterildiği gibi Üniversite tablosunun öznitelikleri; ÜniversiteKodu, ÜniversiteAdi ve Adres bilgisidir. Tabloda veri tekrarı olmaması için her satır diğerlerinden farklıdır ve tabloda bütün özniteliklerin aynı değerleri aldığı ikinci bir satır bulunmaz. İlişki için bir anahtar kullanılması gerekir. Anahtar, bir satırı tek başına tanımlayabilen öznitelikler kümesidir. Anahtar kavramı, ilişkisel veri modelinde kullanılan önemli bir kısıttır. Bu kurallar kullanılarak hazırlanan bir ilişkisel modelde, yine de belirsizlikler ve uyumsuzluklar bulunabilir.

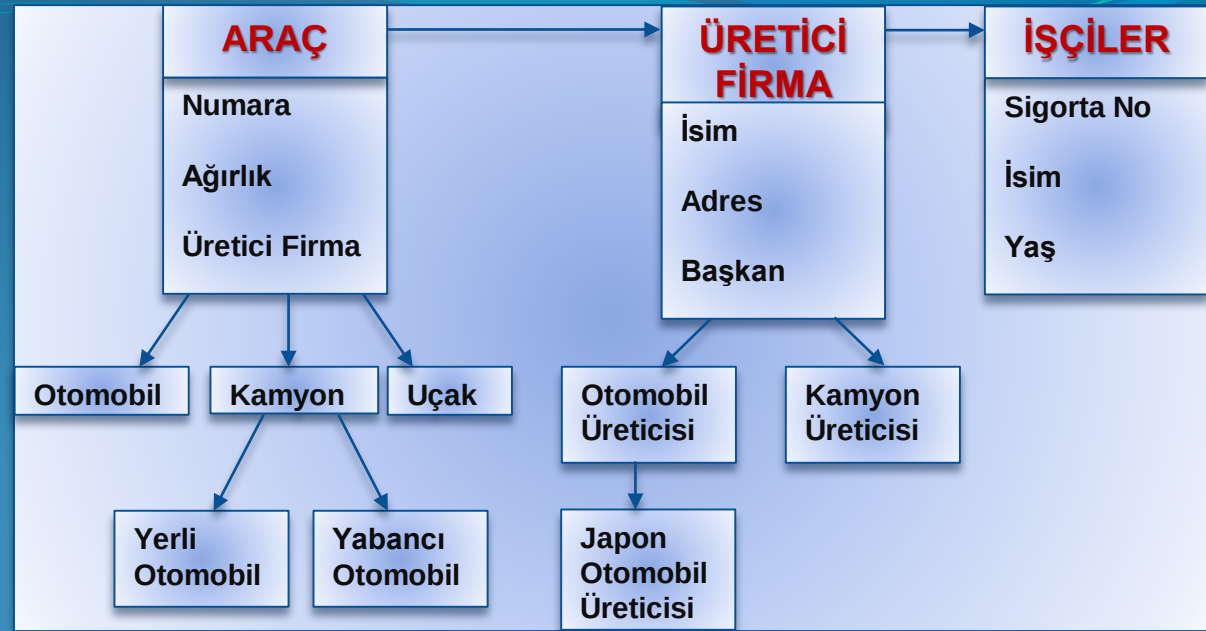
- Bunları gidermek için de bir dizi düzgüleme işlemine gerek duyulabilir. Düzgülemek, veritabanı tasarım prensiplerini yapısallaştırmayı amaçlar. İlişkiler ve öznitelikler arasındaki fonksiyonel bağımlılıkları düzenler. Birbirini takip eden beş işlemde oluşur. Fonksiyonel bağımlılık şu şekilde tarif edilebilir: “ $x$  ve  $y$  öznitelikleri arasındaki ilişki  $R$  ile gösterildiğinde, her bir  $x$  değerine bir tek  $y$  değeri karşılık geliyorsa,  $R$ ’nin  $y$  özniteliğinin,  $R$ ’nin  $x$  özniteliğine fonksiyonel olarak bağımlı olduğu söylenir.”

## 2.4.7. Nesne Yönelimli Veri Modelleri

- Nesne yönelimli programlamanın başlangıcı, 1960'ların sonu ve 1970'lerin başı arasında geliştirilen simülasyon dili Simula'ya kadar uzanır. Nesne yönelimli veri modelinde, bir sorgunun karşılığında mutlaka önceden tanımlanmış belirli bir nesne kümesi olması gerekir. Bir sorgunun sonucu olarak tesadüfî bir nesne kümesinin elde edilmesi mümkün değildir. Çünkü bütün nesnelerin, modelde önceden tanımlanmış olması gerekmektedir. Buna ilişkin bir örnek Şekil 2.3 'da verilmiştir. Genellikle soyutlama olarak anılan bu tip işlemler, şu başlıklar altında toplanabilir:



- Sınıflandırma ve elemanlarına ayırma, sınıflandırma, nesne yönelimli veri modeli yaklaşımının temelini oluşturmaktadır ve aynı özellik ve davranışlara sahip nesnelerin nesne sınıfları içinde gruplanması ile ilgilidir. Bir sınıftaki nesneler, o sınıfın tanımına göre tarif edilebilir. Böylece her nesneyi ayrı ayrı tarif etmeye gerek kalmaz. Elemanlarına ayırma ise sınıflandırma işleminin tersidir ve bir sınıf içinde farklı nesneler oluşturulması ile ilgilidir.



Şekil 2.3: Sınıf Hiyerarşisi

Tanımlama, bu işlem hem soyut kavramların (sınıf), hem de somut kavramların (elemanlar), teker teker tanımlanması ile ilgilidir ve anahtar değerler yardımıyla yapılır. Toplam, nesneler arasındaki ilişkilerin daha üst düzeyde, bir toplam nesne veya tip tarafından temsil edilmesiyle ilgili bir soyutlama yöntemidir. Bu toplam tipe genellikle anlamlı bir isim verilir ve bu isim modelin başka yerlerinde, ona ait özellikler referans olarak verilmeden kullanılabilir. Genelleştirme, aynı özelliklere sahip bir grup nesnenin, sosyal nesne olarak temsil edilmesiyle ilgili bir soyutlama yöntemidir.

# BÖLÜM 3

## VERİTABANI TEMEL KAVRAMLARI

## 3.1. Veritabanı (DataBase) :

En genel tanımıyla, kullanım amacına uygun olarak düzenlenmiş veriler topluluğudur. Müşteri adres defterleri, ürün satış bilgilerinin saklandığı dosyalar, öğrenciler ve öğrenciler ait harç ve not bilgileri gibi, personel bilgi dosyaları gibi bilgi düzenleri veritabanlarına örnek olarak verilebilir. Belirli bir konu hakkında toplanmış veriler; bir veritabanı programı altında toplanırlar.

-İstenildiğinde toplanan bilgilerin tümü veya istenilen özelliklere uyanları görüntülenebilir, yazdırılabilir hatta bilgilerinden yeni bilgiler üretilerek bunlar çeşitli amaçlarla kullanılabilirler.

-Veriler fiziksel hafızada Veri Dosyaları (DataFiles) halinde saklanırlar. Dosya, bilgisayarların bilgileri birbirinden ayırarak saklamak için kullandığı temel bilgi depolama yapısıdır. Bir dosyada, birçok veri yer alabilir. Bir personel otomasyonu ele alınacak olursa, personel ile ilgili bilgiler, personelin çalıştığı birimler, meslekleri, aldığı maaş ile ilgili bilgiler aynı veri dosyasında ama farklı tablolar içerisinde yer alabilirler.

## 3.2. Tablo ve Elemanları:

Tablo verilerin satırlar (row) ve sütunlar (column) halinde düzenlenmesiyle oluşan veri grubudur. Veritabanları bir veya daha fazla tablodan oluşurlar. Tablolar arasında ilişkiler düzenlenebilir. Tablonun satırlarındaki her bir bilgi kayıt (record), sütunlar ise alan (field) olarak isimlendirilir. Bir tabloda yer alan her bir kayıt bir satıra karşılık gelir.

**Örneğin** personel listesi (yani personel tablosunu) ele alınacak olursa, her bir satırda bir personele ait bilgiler yer alır. Sütunlardaki alanlar (Field) ise yapılandırılmış bilginin her bir kısmını saklamak üzere yapılan tanımlamadır. Bir personele ait bilgilerin her biri sütunlarda tutulur.

- Personelin sicil numarası, adı, soyadı, çalıştığı birim, doğum tarihi gibi bilgilerin her biri bir sütun alanıdır. Her bir alan, yapılandırılmış verinin bir birimini tutmak üzere tanımlanır. Her bir sütunun adı ile birlikte diğer bilgilerinin (en fazla kaç birimlik bilgi bu hücrede saklanabilecek, ne tür bilgi saklanacak vs.) ortaya koyduğu tanıma alan denir. Herhangi bir veritabanı programında çalışmaya başlamadan önce yapılacak işe uygun veri tabanı tasarımı yapılmalıdır. Bu işin en önemli aşamasıdır.

- Başlangıçta iyi tasarlanmayan bir veritabanı ileride geriye dönüşü olmayan verimsiz bir bilgi yığınınına dönüşebilir. En basit hali ile veritabanı tasarımında; hangi tabloların olacağı, bu tablolarda hangi alanların olacağı, tablolar arasındaki alan ilişkilerinin neler olacağı ve alanlara ait özelliklerin tanımlanması yapılır. Alan özelliklerinde alan adı, alan tipi, alanın uzunluğu, alanın varsayılan değeri, bu alana yazılacak verilerin geçerlilik koşulları başlangıçta tasarlanması gerekir.



- Veritabanının en önemli bileşeni tablodur. Her veritabanında en az bir tablo bulunur. Veritabanı işlemlerinde önce tablo/tablolara tanımlanır. Daha sonra tablolara kaydedilecek bilgilerin neler olacağı ve bu bilgilere ait özellikler tanımlanır. Personelin sicil numarası ve bunun sayılardan oluşması, personelin adı soyadı ve bunun harflerden oluşması gibi. Tanımlamalar bittikten sonra tablodaki bu alanlara ait gerçek bilgiler yazılır. Yazılan bu bilgiler tablolarda tutulur.

- Kayıt ile satır arasındaki temel fark, kayıt ile kastedilen yapının sütunlar hakkındaki bilgileri de içermesidir. Tablolara girilmiş bilgilerden belirli şartlara uyanların liste şeklinde alınmasına sorgu adı verilir. Tablolardan gerektiğinde sorgulamalar yapılabilir. Değişik amaçlara göre sorgular hazırlanarak tablodaki bilgilerin tümü, bir kısmı veya belirli şartı sağlayanların listesi alınabilir. **Örneğin;** muhasebe bölümünde çalışan personelin listesi gibi.

## 3.3. Veri Tipi (Data Type):

- Bilgisayar, kayıtları tablolarda yapısal olarak tutarken, onların yapıları hakkında fikir sahibi olabilmek için bazı özelliklerinin önceden tanımlanması gerekir. Örneğin, personel sicil numarası alanının mutlaka bir tam sayıdan oluşacağı, personel ad ve soyadının harflerden oluşacağı, personelin çalıştığı bölümün harf ya da rakamlardan oluşacağı, personelin doğum tarihinin tarih bilgilerinden oluşacağı gibi. Bir veritabanı oluşturulurken, önce tablolar ve sonrada bu tablodaki her bir alanın veri tiplerinin ne olacağı tanımlanmak zorundadır.

- Bir tablo alanına veri giriři yapılmadan önce o alanın tamsayı mı yoksa harf mi; tarih mi yoksa ondalıklı bir sayı mı olacađı tanımlanmalı ve veriler daha sonra tabloya yazılmalıdır. Ayrıca, “bir alanın uzunluđu ne kadar olacak, harf girilebiliyorsa en fazla kaç harf girilebilecek?”, “rakam ise en fazla kaç basamaklı olabilir?” türünden soruları yanıtlamak için de yine VTYS bir alan için veri tipi belirlenmesini ister. Her Veri Tabanı Programının veri tipleri farklıdır.

**Tablo 3.1: SOL Server Veri Tipleri ve Özellikleri (1)**

| VERİ TİPİ                   | SQL KOMUTU                                                                                                                                         | ÖZELLİĞİ                                                                                                                                                                               |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sabit uzunluklu karakter    | <b>CHAR (Uzunluk)</b><br>Örnek: CHAR(15)                                                                                                           | Sayısal işleme sokulmayacak veriler için kullanılır. Adres, isim, açıklama v.b. Uzunluk sabit olarak belirlenir. Maksimum uzunluk 254 karakterdir.                                     |
| Değişken uzunluklu karakter | <b>VARCHAR (Uzunluk)</b> Buradaki uzunluk, maksimum uzunluktur. Veri daha kısa ise, uzunluğun gerektiği kadarı kullanılır.ÖRNEK: VARCHAR(23) gibi. | Karakter türündeki veriler gibidir. Tek fark, uzunluk değişkendir; yani verinin gerektirdiği uzunluk kullanılır. Bu veri tipi standart değildir. Her derleyicide bulunmaz.             |
| Nümerik tam sayı            | <b>INTEGER</b>                                                                                                                                     | -2147483648 ile +2147483647 arasındaki tam sayılar için kullanılır. Ondalıklı nokta kullanılamaz. Bellekte 4 byte'lik yer kaplar.                                                      |
| Nümerik kısa tam sayı       | <b>SMALLINT</b>                                                                                                                                    | -32768 ile +32767 arasındaki tamsayılardır. Bu veri tipi standart değildir. Bazı derleyicilerde tanımlanmamıştır. Bu durumda veri tipi INTEGER'a çevrilir. Hafızada 2 byte yer kaplar. |

**Tablo 3.1: SOL Server Veri Tipleri ve Özellikleri (2)**

| VERİ TİPİ                    | SQL KOMUTU                                                                                       | ÖZELLİĞİ                                                                                                                                                                                                                                        |
|------------------------------|--------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nümerik kısa tam sayı</b> | <b>TINYINT</b>                                                                                   | 0 dan 255 e kadar olan sayıları tutarlar. Hafızada 1 byte yer kaplar.                                                                                                                                                                           |
| <b>Ondalık sayı</b>          | <b>DECIMAL(x,y)</b><br><b>REAL(x,y)</b> ya da<br><b>NUMERIC(x,y)</b><br>şeklinde tanımlanabilir. | x sayısı maksimum hane (digit) sayısı, y ondalık noktadan sonraki hane sayısıdır. x en fazla 20 olabilir. y 0-18 arasındadır.Örnek: -10.22 veya 1056.82 gibi sayılar.                                                                           |
| <b>Üstel sayı</b>            | <b>FLOAT(x,y)</b>                                                                                | x sayısı toplam hane (digit) miktarıdır. Maksimum değer 20'dir. y ondalık noktadan sonraki hane sayısıdır. 0-18 arasındadır.0.1 E-307 ile 0.9 E+308 arasındaki değerleri alabilir. Çok küçük ve çok büyük sayılar için uygun olan veri tipidir. |
| <b>Tarih türü veriler</b>    | <b>DATE</b>                                                                                      | Tarih türü bilgiler üzerinde işlem yapma imkânı sağlar. SQL için standart bir veri tipi değildir. Bir gerçekleştirimden diğerine farklı şekilde kullanımı mümkündür.                                                                            |
| <b>Mantıksal veri</b>        | <b>LOGICAL</b>                                                                                   | Doğru (True, .T.) ya da yanlış (False, .F.) şeklinde değer alabilen veriler için kullanılır.                                                                                                                                                    |

**Tablo 3.1: SOL Server Veri Tipleri ve Özellikleri (3)**

| VERİ TİPİ                                  | SQL KOMUTU           | ÖZELLİĞİ                                                                                                                                                              |
|--------------------------------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Zaman türü veriler</b>                  | <b>TIME</b>          | ss:dd:sn (saat, dakika, saniye) şeklinde veriler için kullanılır. Standart değildir.                                                                                  |
| <b>Tarih ve zaman türü veriler</b>         | <b>TIMESTAMP</b>     | Tarih ve zaman türü verilerin bir karışık şeklinde kullanılan veriler içindir. Standart değildir.                                                                     |
| <b>Grafik türü veriler</b>                 | <b>GRAPHIC(n)</b>    | n adet 16 bitlik karakterden oluşan bir sabit uzunluklu bilgi tanımlamak için kullanılır. (Karakter türü veri, 8 bitlik karakterden oluşmaktadır.) Standart değildir. |
| <b>Değişken uzunluklu grafik türü veri</b> | <b>VARGRAPHIC(n)</b> | n adet 16 bitlik karakterden oluşan değişken uzunluklu (n maksimum uzunluktur.) bir bilgi tanımlamak için kullanılır. Standart değildir.                              |

**Tablo 3.2: Tarih Formatları ve Örnekleri**

| STANDART ADI                                            | TARİH FORMATI                           | ÖRNEK      |
|---------------------------------------------------------|-----------------------------------------|------------|
| <b>Uluslararası Standartlar<br/>Organizasyonu (ISO)</b> | yyyy-aa-gg<br>y –yıl<br>a –ay<br>g –gün | 1992-11-29 |
| <b>IBM ABD Standardı USA</b>                            | aa/gg/yyyy                              | aa/gg/yyyy |
| <b>IBM Avrupa Standardı<br/>EUR</b>                     | gg.aa.yyyy                              | 29.11.1992 |
| <b>Japon Endüstri<br/>Standardı JIS</b>                 | yyyy-aa-gg                              | 1992-11-29 |



**Tablo 3.3: Zaman Formatları ve Örnekleri**

| STANDART ADI                                        | ZAMAN FORMATI                                 | ÖRNEK    |
|-----------------------------------------------------|-----------------------------------------------|----------|
| <b>Uluslararası Standartlar Organizasyonu (ISO)</b> | SS.dd.ss<br>S –saat<br>d -dakika<br>s –saniye | 16.25.07 |
| <b>IBM ABD Standardı USA</b>                        | SS:dd AM veya PM                              | 16:25 PM |
| <b>IBM Avrupa Standardı EUR</b>                     | SS.dd.ss                                      | 16.25.07 |
| <b>Japon Endüstri Standardı JIS</b>                 | SS:dd:ss                                      | 16:25:07 |

- Veri tipleri tablolarda tutacağımız verilerin yapısını belirlemeye yarar. Tabloları oluştururken doğru veri tiplerini kullanmak bize sonradan (hazırladığımız veri tabanını kullanırken) kaynak ve performans açısından büyük kazançlar sağlar. Aksi takdirde hazırladığımız veri tabanları gereğinden büyük boyutta veriler tutmamıza sebep olacağı için hazırladığımız veri tabanını kullanan programların performansını da olumsuz yönde etkileyecektir.

## 3.4. Zorlayıcı (Constraint)

Herhangi bir alan için girilebilecek verileri kısıtlayıcı kurallara zorlayıcılar denir. İlgili alana girilebilecek değerleri sınırlayan bir deyim yazılır. Kullanımı bazen çok faydalıdır ve özellikle yanlış bilgi girişini engeller ve verilerin doğru girilmesini zorunlu hale getirir.

- Kullanıcı, zorlayıcıda belirtilen kural dışında bir veriyi tabloya yazmaya çalıştığında, VTYS hata verir. Böylelikle veritabanına kullanıcının keyfi değerler girmesi önlenmiş olur ve veri tabanında tutarlılık sağlanmış olur. Örneğin, bir öğrencinin sınıf bilgisine ait değerler yazılırken bu alan için rakamsal 1 ile 6 arasında bir zorlayıcı değer tanımlanırsa; veri girişi sırasında 1 ile 6 arasındaki değer dışında bir değer sınıf bilgisi alanına yazılması engellenmiş olur. Dolayısı ile sınıf için yazılmaması gereken bir değer; bilgi girişi başlangıcında kontrol edilmiş olur.

## 3.5. Anahtar (Key)

Anahtar bir veya birden fazla alanın bir satır için niteleyici olarak girilmesi için tanımlanan özel bir çeşit zorlayıcıdır. Tekrarlamayacak bir anahtar alan tanımlandığında, bu anahtar alana birincil anahtar alan denir.

- 3 çeşit anahtar vardır:
- Primary Key (Birincil Anahtar)
- Unique Key (Tekil Anahtar)
- Foreign Key (Yabancı Anahtar)

(Bu konular ileride örneklerle tekrar anlatılacaktır.)

### 3.5.1.Primary Key (Birincil Anahtar) :

Bir tablodaki, her bir satırın yerine vekil olabilecek bir anahtar veridir. Tabloda bu alana ait bilginin tekrarlanmaması gerekir. Standart olarak bir tabloda verilerin, fiziksel hafıza üstünde de hangi alana göre dizileceğini de primary key belirler. Bu, bazen bir tek alan olabileceği gibi, bazen birden fazla alan da birleşerek bir birincil anahtar oluşturabilir.

**Örneğin** programda personelin sicil numarası alanına göre aramalar yapılacaksa Primary key personel sicil numarası olmalıdır. Personelin ad ve soyadına göre aramalar yapılacaksa ad ve soyad alanları birleştirilerek iki alandan tek anahtar alan tanımlaması yapılır.

| ÖĞRENCİ |       |          |
|---------|-------|----------|
| NO      | ADI   | BÖLÜM    |
| 55      | AHMET | BİL.TEK  |
| 34      | MURAT | ELEKTRİK |
| 12      | AYŞE  | KİMYA    |

| DERSLER |            |         |
|---------|------------|---------|
| KODU    | ADI        | KREDİSİ |
| 125     | İNGİLİZCE  | 2       |
| 224     | MATEMATİK  | 3       |
| 287     | BİLGİSAYAR | 3       |

| NOTLAR   |           |      |      |
|----------|-----------|------|------|
| ÖĞR. NO. | DERS KODU | NOT1 | NOT2 |
| 55       | 125       | 50   | 70   |
| 34       | 287       | 45   | 60   |
| 55       | 224       | 80   | 50   |
| 12       | 125       | 75   |      |
| 12       | 287       | 60   |      |

Primary Key

34 MURAT ELEKTRİK

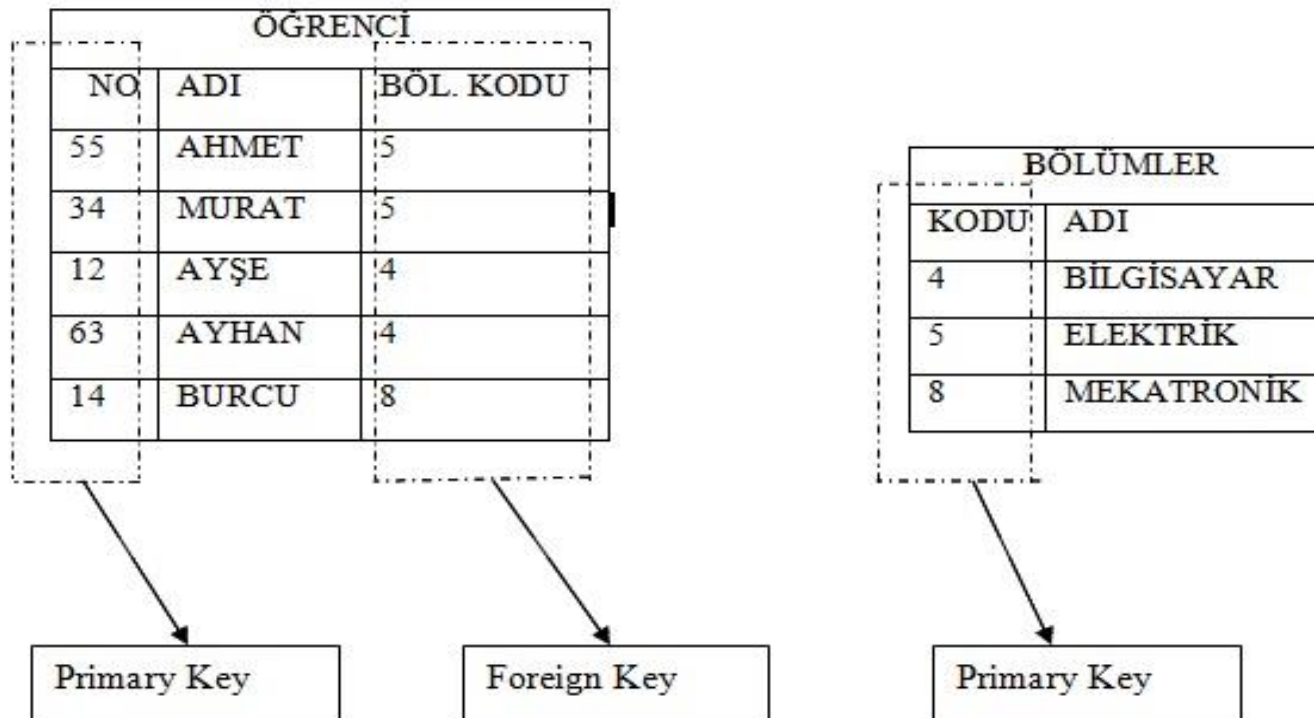
## 3.5.2.Unique Key(Tekil Anahtar):

Unique Key olarak tanımlanan alan için bir değer sadece bir kere girilebilir. Bir başka satıra daha aynı verinin girilmesine izin verilmez. Primary Key 'den farklı olarak Unique Key, NULL (boşluk) değerini alabilir. Örneğin programda her personele ait bir sicil numarası olacağı için bu alan Unique key olarak tanımlanabilir. Ama isim alanı birden fazla aynı isme sahip personel olabileceği için bir Unique key olarak tanımlanamaz. Ali isimli birden fazla personel olabileceği gibi.



### 3.5.3.Foreign Key (Yabancı Anahtar) :

Bir tabloya girilebilecek değerleri başka bir tablonun belli bir alanında yer alabilecek veri grubu ile sınırlandırmaya ve en önemlisi de ilişkilendirmeye yarar. Örneğin, olmayan bir kitabın ödünç tablosuna eklenememesi ve ödünç tablosuna eklenen bir kitabın numarası aracılığıyla detay bilgilerine erişilmesi gibi. Burada Kitap.KitapNo birincil anahtar alan; Odunc.kitapNo ise yabancı anahtardır.



## 3.6.Index (İndeks)

Kütüphanelerdeki kitapların raflardaki dizilişlerini ele alalım. Bir kitap arandığında, kitaplar bir kurala göre dizilmemişlerse, her bir kitaba teker teker bakılması gerekir. Kitaplar raflara alfabetik dizilirse, her bir kitap tek tek gözden geçirilmek zorunda kalınmaz. Aranılan kitap ile bakılmakta olunan kitabın isimleri karşılaştırılır, sağa ya da sola yönelip aramaya devam edilir. Aynı şekilde yazarlarına ya da kütüphane numarasına göre sıralanmış birer liste olursa, bu kriterlere göre de aranılan kitap kolayca bulunur. Veritabanlarında indeks oluşturularak, veriler veritabanındaki kayıtlı oldukları sıradan başka bir sırada gösterilebilir ve tıpkı kütüphanedeki bir kitaba ulaşmada olduğu gibi istenilen veriye daha kısa sürede ve kolayca ulaşılabilir.

## 3.7. View (Görüntü):

Bazen, tabloları olduklarından farklı gösterecek filtrelere ihtiyaç duyulur. Bu türden işlevler için VIEW kullanılır. VIEW 'ler, saklanmış sorgulardan ibarettirler. Aslında tablo gibi kullanılsa da hâlihazırda böyle bir tablo veritabanında bulunmaz, sadece view(görüntüsü) bulunur. VIEW 'ler şu görevler için kullanılır:

- Kullanıcıların bazı kritik tabloların sadece belli sütunlarını veya satırlarını görmesi istenildiğinde,
- Kullanıcıların, çeşitli birim dönüşümlerinden geçmiş değerler görmeleri gerektiğinde,
- Hâlihazırdaki tablolarda var olan verilerin başka bir tablo formatında sunulması gerektiğinde,
- Çok karmaşık sorguları basitleştirmek için,

**Örneğin** Kitap tablosunda sadece Bilgi Teknolojileri türündeki kitapların yer alacağı bir VIEW şu şekilde oluşturulabilir:

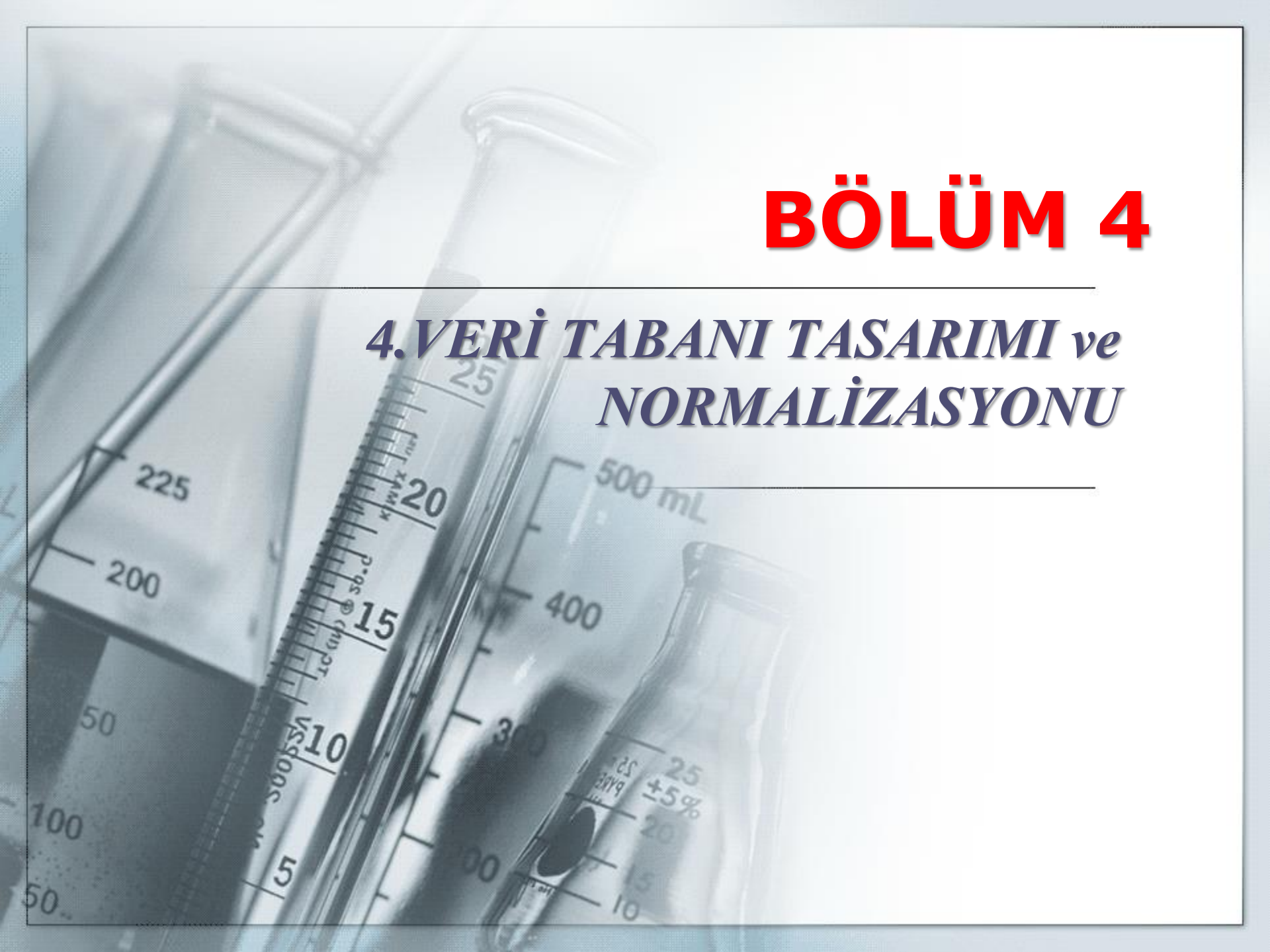
```
CREATE VIEW view_adi [(kolon1,kolon2...)] AS  
SELECT tablo1.kolon_adi_1, tablo2.kolon_adi_1  
FROM tablo_adi_1, tablo_adi_2;
```

**Not:** MSAccess 'de VIEW oluşturulamaz. SQLServer, SyBase, Oracle gibi orta ve büyük ölçekli VTYS 'lerde oluşturulabilir.

## 3.8. Joining (ilişkilendirme)

İki veya daha fazla tabloyu birlikte sorgulama işlemine join ismi verilir. İlişkisel veritabanının en temelinde birden fazla tablo üstünde birlikte işlem yapabilmek yatar. Bu sayede verilerin tekrarlanması önlenmiş olur ve sonuçta veri yönetimi kolaylaşır. Örneğin, Kitap tablosunda, Kitabın bir tekil numara ile listesini tutmak ve ödünç listesinde de bu Kitabın kim tarafından alındığının, geri getirilip getirilmediğinin kaydı tutulmaktadır. Bazen, bu iki tablodaki bilgilere de bir tek sorgu sonucu olarak ihtiyaç duyulabilir.

**Örneğin;** elimizde öyle bir sonuç olmalıdır ki, hangi kitabın kim tarafından ödünç alındığını bir listede görme ihtiyacı duyulsun. Bu iki tablo birbirine, kitapNo alanı ile bağlıdır. Çünkü ödünç verilen bir kitap hakkında detaylı bilgi edinilmek istenildiğinde, ödünç listesinden kitap numarasını alıp, daha sonra Kitap tablosundan aynı numarayı bulmak ve karşılığındaki kitap hakkındaki detayları görmek.



# BÖLÜM 4

---

## *4. VERİ TABANI TASARIMI ve NORMALİZASYONU*

---

## 4.1. Veri Tabanı Tasarımı

---

İyi bir veritabanı tasarımı yapabilmek için yetenek, bilgi ve tecrübe çok önemlidir. Öncelikle, ilişkisel veritabanının tanımını ve bununla ilgili 5 Normalizasyon kuralını çok iyi bilmek gerekir. 5N, tasarım aşamasında yol göstermek yerine hangi şartlara uygun tasarım yapılması gerektiğini anlatır. Bazen, bu kurallardan vazgeçmek durumunda olunabilir ancak, veritabanında saklanacak verilerin hacmi arttıkça yani veri tabanı büyüdükçe bu kuralların daha sıkı uygulanması gerekir.

Bir veri tabanı ile proje yapılırken işin en önemli aşaması veri tabanının tasarlanmasıdır. Başlangıçta yanlış tasarlanan bir veri tabanı ile yapılan projede sonradan yapılacak düzenlemelerle geri dönüş yapılamaz. O nedenle Veri tabanı tasarımı yapılırken aşağıdaki maddelere uyularak yapılması gerekir.



## 4.1.1. Nesnelerin Tanımlanması

Nesne, çeşitli özellikleri bulunan bir varlıktır. Herhangi bir proje de öncelikle nesneler tanımlanır. Birkaç proje için nesnelere örnek verilecek olunursa,

Kütüphane sistemi: Kitap, üyeler, türler, ödünç hareketleri,

E-ticaret sistemi: Ürünler, müşteriler, siparişler, teslimat, fatura bilgileri, üreticiler, tedarikçiler, dağıtıcılar,

Futbol Ligi: Takımlar, sahalar, oyuncular, fikstür, hakemler, antrenörler,

Okul Sistemi: Öğrenciler, öğretmenler, dersler, derslikler,

Personel Sistemi: Çalışanlar, meslekler, çalışılan birimler, maaşlar, izinler,

Sözlük: kelimeler, anlamlar, diller,

**Not:** Tablolara isim verilirken mümkünse tekil isimler kullanılmalıdır. Böyle yapılırsa; hem daha anlaşılır bir tasarım yapılmış olur hem de daha sonra kodlama aşamasında karışıklığın önüne geçilmiş olur. Örneğin içinde Kitap ile ilgili bilgiler bulunduran tablonun adını Kitap koymak oldukça mantıklıdır.

## 4.1.2. Her Nesne İçin Tablo Oluşturması

---

- Her nesne için bir tablo oluşturulur ve her bir tabloya içereceği veriyi en iyi anlatan bir isim verilir. Tablo oluşturma işi, bir kâğıt üstünde sembolik olarak gösterilebilir veya doğrudan MS Access, SQL Server, MySQL, Oracle... gibi kullanılmakta olunan VTYS üstünden de oluşturulabilir. Tüm proje bitirilinceye kadar bu tablolar üzerinde muhtemel değişiklikler yapılabilir.

## 4.1.3. Her Bir Tablo İçin Bir Anahtar Alan Seçilmesi

---

Veritabanındaki herhangi bir veriye erişilmeden önce tabloya erişilir. Bir veritabanında üzerinde en çok işlem yapılan nesne grubu genellikle tablolardır. Bu aşamaya kadar hangi tabloların oluşturulacağına karar verildi. Her bir tablonun içinde hangi bilgilerin saklanılacağı kabaca tasarlanır. Bu aşamada, tabloda yer alacak her bir kaydı bir diğerinden ayırabilecek bir sütuna ihtiyaç duyulur. Örneğin bir kitap seçilmek istenildiğinde, bu kitabın hangi kitap olacağı öyle bir anlatılabilmeli ki, başka hiçbir kitap ile karışmamalıdır. Bunu yapmanın tek yolu, bir alanı birincil anahtar alan olarak belirlemektir. Anahtar alan seçilirken, kısıtlamadığı sürece, doğal alanlar seçilmeye dikkat edilmelidir. Örneğin araçlar ile ilgili bir tablo yapılırken, plakalar anahtar alan olarak belirlenebilir. Çünkü her bir plakadan bir tek araç trafiğe çıkabilir ve plakalar kısıtlamaz.

---

Öğrenci tablosu için, öğrenci numarası doğal bir anahtar alandır çünkü aynı okulda, aynı numaradan bir öğrencinin daha bulunması söz konusu değildir. Personel tablosu için, personel sicil numarası doğal bir anahtar alandır çünkü aynı işyerinde, aynı numaradan bir personel daha bulunmaz. Kitap tablosu için ISBN numarası anahtar alan olarak tanımlanabilir ama aynı kitaptan iki adet olduğunda, ISBN numarası bizi kısıtlar. Elimizde iki adet “Önümüzdeki Yol” kitabı varsa, her iki kitabın da ISBN numarası aynıdır. Kitaplardan birisi eski diğeri yeni olabilir. Bu bir kargaşaya neden olabilir. Çünkü eski kitabı kime, yeni kitabı kime verdiğimizin takibini ISBN numarası ile yapmak mümkün değildir. Ancak bir E-Ticaret sitesi tasarlanırken, stoktaki tüm kitaplar birbiri ile eşdeğer olacağından ya da öyle olduğu varsayıldığından ISBN numarası birincil anahtar alan olabilir. Bu durumda, adet diye bir niteliğin aynı tabloda yer alması gerekecektir.

## 4.1.4. Nesnelerin Her Bir Özelliği İçin Tabloya Bir Sütun Eklenmesi

---

- Tablo adları tanımlandıktan ve anahtar adları belirlendikten sonra, tablolara sırasıyla adını veren nesnelerin her bir özelliği için bir alan (sütun) eklenir. Örneğin, kitap için;
- Kitap no, ISBN no, kitap adı, yazarı, türü, sayfa sayısı, özeti, fiyatı, baskı yılı...
- Üye için; UyeNo, adı, soyadı, e-mail adresi, ev telefonu, cep telefonu, iş telefonu....
- Personel için; Personel sicil No, adı, soyadı, e-mail adresi, mesleği, çalıştığı birim, maaş...
- Bu hazırlıklar yapılırken yapılması istenilen proje ile ilgili basılı formlar vs. varsa, onların incelenmesi tabloya eklenecek sütunların hangi özellikler olması gerektiği konusunda karar verilmesinde yardımcı olurlar.

- 
- **Not:** 1. En başa birincil anahtar olarak belirlenen alanı eklemek bir kural değildir, ancak tablonun anlaşılabilirliği ve göze hoş görünmesi açısından tercih edilmesi faydalı olacak bir tekniktir.
  - 2. Genellikle, yapay birincil anahtar alanlar tablo adı ile başlar ve sonunda ID vardır. Öğrenci tablosu için öğrenciID, Personel tablosu için personelID gibi.



## 4.1.5. Tekrarlayan Nesne Özellikleri İçin Ek Tablolar Oluşturması

---

- Akılda hep şu soru olmalıdır: veri tekrarı olacak mı? Veri tekrarı olacaksa bir yerlerde hata yapılıyor demektir. Bu durumda eldeki tablonun en az bir tabloya daha ayrılması gerekiyor demektir. Şu da unutulmamalıdır, her projeye uyacak evrensel bir veritabanı tasarım tekniği yoktur. Yani her şey belli kurallar çerçevesinde ne kadar detayıyla düşünülüp tasarlandığına bağlıdır. Örneğin, her bir kitap için tür belirledik ama bir kitap hem kişisel gelişim kategorisine hem de hikâye kategorisine girebilir. Ya da e-ticaret sisteminde bir ürünün birden fazla reyonda yer alması gerekli olabilir.

- 
- Veya bir kitap birden fazla kiři tarafından yazılmış olabilir. Bir kitap için birden fazla türü kaydedebilme ele alınsın: Bu türden bir sorunu çözmek için ilk akla gelen şey, Kitap tablosunda tür alanı için 2.sütun daha eklemek olabilir. Bu tabloya 2.Tür ve 3.Tür diye iki sütun alanı daha eklemek. Ama çoğu kitap bir tek türdendir ve bu kitap için eklenen 2 alan hep boş kalacaktır. Öte yandan, 4.türe birden giren bir kitap olduğunda 4.tür bilgisi nereye yazılacaktır? Aynı alana mı? Ya da dört adet bölüm mü açılacak? Bunlar, veritabanı tasarımının doğasına terstir. 2.Çözüm yolu ise, bir kitabı iki kere kaydedip, birincisini, 'Kişisel Gelişim' türü olarak; ikincisini de 'Hikaye' olarak girmektir. Bu durumda tabloda aynı kitaba ait iki kayıt olacaktır ve kitap türü dışındaki diğer tüm bilgiler tekrar edecektir.



- Ya da bir süre sonra, kitap hakkında girilen bilgilerin yanlış olduğu fark edildi. Hangi kayıt güncellenecektir? Ya biri düzeltip diğeri unutulursa? Sonuçta veri tekrarı ve veri bütünlüğünün bozulması söz konusudur. Bu da yine ilişkisel veritabanı tasarımının doğasına terstir. Bu durumda, türler diye bir yeni tablo oluşturup, bir de kitap\_turler diye 2.tablo'yu oluşturduktan sonra bu türden bilgileri burada tutmak gerekecektir. Böylelikle, hiçbir türde yer almayan kitaptan 10 ayrı türde yer alan kitaba kadar bütün olasılıklar için bir çözüm geliştirilmiş olur. Aynı işlem öğrenci ve öğrenciye ait ders notları için düşünülebilir. Öğrenciye ait ders not bilgilerinin yazıldığı tabloya ait sütunların aşağıdaki gibi olduğunu varsayılırsa;

| ÖğrenciNo | DersinAdı | VizeNotu | FinalNotu | Ortalama |
|-----------|-----------|----------|-----------|----------|
|-----------|-----------|----------|-----------|----------|

- Bir öğrenci aldığı dersten başarılı olursa vize ve final notu yazılarak ortalaması hesaplanır ve sorun yaşanmaz. Ama öğrenci bu dersten başarısız olursa bu dersi yeniden almak zorundadır. Yeniden aldığı bu derse ait ders notlarının nereye yazılacağına düşünülmesi gerekir. Eski notlarının da kalması gerektiği düşünüldüğü bu durumda tablo aşağıdaki gibi tasarlanabilir.

| ÖğrenciNo | DersinAdı  | VizeNotu | FinalNotu | Ortalama | VizeNotu | FinalNotu | Ortalama |
|-----------|------------|----------|-----------|----------|----------|-----------|----------|
| 03101001  | BILGISAYAR | 37       | 40        |          | 45       | 48        |          |

- Tabloda 2 adet not yazılabilecek alan vardır. Peki, öğrencinin dersi ikiden fazla kere tekrar etmesi gerekirse ne olacak? Bu durumda yeni sütun alanları mı eklemek gerekecek? Tabloya 3 tane not yazma alanı eklendiğinde dersi bir kere alan ve başarılı olan öğrenciler için 2. ve 3.alanlar boş kalacaktır. Bu her öğrenci için değişebilecek bir durum olduğu için tablo tasarımında bu mantıkla düşünmek doğru değildir. Yukarıda ki örnekte de açıklandığı gibi bu şekilde bir tasarım yapılmaz.

Ayrıca; tabloda tanımlanan her sütun alanı, bu alana hiçbir bilgi yazılmasa bile HD’de yer kaplayacağı için; diskte tanımlanan bu alanlar boşuna kullanılmış olacaktır. Dolayısı ile diskte de boş yere alan işgal edilmiş olacağından tabloda gereksiz sütun alanlarının tanımlanmaması gerekir. Örneğin, tabloda gereksiz tanımlanan bir sütun alanı diskte 4byte yer kaplıyor ise ve tabloda toplam 15 bin öğrenci var ise; gereksiz kullanılan toplam HD alanı  $4 * 15.000 = 60.000$  byte olacaktır. Sadece tek bir alan için bu kadar alanın boş yere kullanılmış olması hoş bir durum değildir. Bu durumda tablo tasarımında yapılması gereken düzenleme aşağıdaki gibi olmalıdır. Bir öğrenciye ait dersler yazılırken alt alta satırlar şeklinde kayıt (record) olarak yazılarak yapılmalıdır.

| ÖğrenciNo | DersinAdı  | VizeNotu | FinalNotu | Ortalama | Tekrar |
|-----------|------------|----------|-----------|----------|--------|
| 03531020  | BİLGİSAYAR | 37       | 40        | 38,5     |        |
| 03531037  | İNGİLİZCE  | 56       | 60        | 58       |        |
| 03531025  | BİLGİSAYAR | 45       | 48        | 46,5     |        |

Doğru tablo tasarımı ve kayıt girişi yukarıdaki tabloda olduğu gibi olmalıdır.

## 4.1.6. Anahtar Alana Bağlı Olmayan Alanların Belirlenmesi

---

- İlişkisel veritabanında, tablodan herhangi bir tek kayda erişmek için mutlaka bir farklı özellik sağlanmalıdır ve bu özellik de anahtar alan tarafından sağlanır. Ancak bazen, anahtar alan ile aynı satırda yer aldığı halde, anahtar alan ile birebir ilişkisi olmayan bir alan yer alabilir. Bu türden alanların elimine edilip ayrı tablolara ayrılması gerekir. Örneğin, ödünç tablosu ele alınacak olursa, ödünç verilen her kitap için ödünç alanın adresi de bilinmek istenirse, bu ödünç tablosuna yazılamaz. Çünkü ödünç tablosunun birincil anahtar alanı oduncNo 'dur ve bu alan, ödünç verme işlemi ile ilgilidir

---

. Oysa ödünç alanın adresi, ödünç alan kişinin kendisine bağlı bir özelliktir. Bu kişinin her aldığı kitap için adresini tekrar yazmaya gerek yoktur. Aynı şekilde otomasyon içerisinde başka yerlerde de bu kişinin adres bilgilerine muhtemelen ihtiyaç duyulabilir çünkü adres, üyenin bir özelliğidir. Ödünç verilen kitabın adresi öğrenilmek istenildiğinde, üyeler adında bir tablo daha açılıp, burada herkesin adres bilgisi tutulmak zorunda kalınır. Ödünç tablosunun ise, oduncAlan bilgisi olarak, Üyeler tablosunun birincil anahtar alanına bir bağlantı (yabancı anahtar) içermesi daha doğru olur.

## 4.1.7. Tablolar Arasındaki İlişkiler Tanımlanması

---

- Tanımlanan tablolardaki alanların birbirleriyle olan ilişkileri tanımlanır. Örneğin öğrenci bilgilerinin tanımlandığı tablo ile öğrenci notlarının tanımlandığı tablo arasında bir ilişki kurulur ve bu ilişki öğrenci numarası ile olur. Öğrencinin hangi dersine ait not bilgisi yazılacaksa ders kodları ve adlarının tanımlandığı tablo ile öğrenci notları tablosu arasında bir ilişki vardır. Bu ilişki notlar tablosundaki ders kodu ile ders kodları tablosundaki dersin kodu alanları arasında yapılır. İlişki her iki tablo bir birincil alan ve bir yabancı anahtar alan üstünden birbirine bağlanır. Bu ilişkiler Ms-SQL Server, Ms Access veya Oracle veritabanlarında şematik olarak hazırlanabilir.



- 
- Farklı tablolardaki iki alan aynı veriyi tutuyorsa iki alana da aynı ismi vermek karışıklığa yol açabilir gibi görünse de aslında daha düzgün bir yapının ortaya çıkmasını sağlar. ÖğrenciNo alanı öğrenci tablosunda da notlar tablosunda da öğrenci numarası olarak tanımlanabilir. Bu alanlardan birine ÖğrenciNo, diğerine NotlarÖğrenciNo ismi verilmesi sorgu ve program yazılımında isim karışıklıklarına neden olabilir. En önemlisi de her alan için her tabloda farklı isimlerin kullanılması değişkenlerin isimlerinin akılda tutulmasını ve daha sonraki tablolar üzerinde işlem yaparken işlemleri zorlaştırır.

- 
- Tablo ve alanları kullanılacağı zaman her seferinde ilgili alanın hangi isimle kaydedildiğine bir listeden bakmak zorunda kalınır. Çünkü büyük bir veritabanı projesinde 250'den fazla tablo bulunabilir. Her tabloda da birçok alanın bulunacağı dikkate alındığında her alana ait isimlerin akılda tutulması mümkün olmamaktadır. Birden fazla tabloda olan alanlar için; aynı ismi kullanmak bu zorluğu ortadan kaldıracaktır. En mantıklısı her ikisine de ÖğrenciNo ismi verilmesidir.



## 4.2 Veritabanı Normalizasyonu

---

- Bir tablo içerisinde yer alacak kaydın nelerden oluşmasına karar vermeye yarayan düzenlemelere normalizasyon kuralları denilir. Bu kurallar ilişkisel veritabanı tasarımından başlı başına bir işlemdir. Normalizasyon; veritabanı tasarım aşamasında çok önemli bir işlemdir. Normalizasyon arttıkça tablolar arasındaki ilişkiler artar. Bu nedenle tablodaki veriler erişmek için gerekli bağlantıların sayısı da artar.
- İlişkisel veritabanınının tanımı ile birlikte ortaya atılmış ve kabul görmüş 5 normalizasyon kuralı vardır ve aşağıda anlatıldığı gibidir.

## 4.2.1 Normalizasyon Kuralı

---

- Bir satırdaki bir alan yalnızca bir tek bilgi içerebilir. Birden fazla notu olan öğrenci için not1, not2 ve not3 diye alanların açılması ile bu kurala uyulmamış olunur. Böyle bir durumda, ayrıca notlar tablosu oluşturularak veritabanı tasarımı ve normalizasyon kuralına uyulmuş olur.

## 4.2.2 Normalizasyon Kuralı

---

- Bir tabloda, anahtar olmayan her alan, birincil anahtar olarak tanımlı tüm alanlara bağlı olmak zorundadır. Örneğin, Notlar tablosuna DersinAdı gibi bir alan eklenirse, bu sadece dersin adı ile ilgili bir bilgi olur ve DersinKoduna bağlı bir nitelik olmaz. Bu nedenle ders isimleri için ayrı bir tablo yapılarak sorun çözülebilir. Ya da anahtar alanın birden fazla alandan oluştuğu tablolarda, anahtar alanlardan sadece birine bağlı veriler tabloda yer almamalı, ayrı bir tabloya taşınmalıdır. Bunun tersi de geçerlidir. Yani iki ya da daha fazla tablonun birincil anahtarı aynı olamaz. Böyle bir durum söz konusu ise, bu iki tablo tek tabloya indirilmelidir.

## 4.2.3 Normalizasyon Kuralı

---

- Bir tablo için, anahtarı olmayan bir alan, anahtarı olmayan başka hiçbir alana bağlı olamaz. Örneğin, notlar tablosunda hangi not bilgisi için Sınav Tipi isimli bir alan eklenip burada Vize için V, Final için F ve Bütünleme için B kodlaması yapılırsa bu alan notlar tablosunun birincil anahtarı olan ÖğrenciNo alanına bağlı bir kodlama olmaz. Çünkü bu kodlama bir başka anahtarı olmayan alana bağlıdır. Bunun sonucunda da veritabanında, karşılığı olmayan bir kodlama yer almış olur. Sınav tipi bilgisini kodlu olarak tutan alan aslında Sınav tipi açıklaması olan başka bir alana bağlıdır. Bu ilişki başka bir tabloda tutulmalıdır.

- 
- Bu durumda, Sınav tipi bilgilerini tutan yeni bir tablo açılması daha doğru olur. Bu kodlamayı program içerisinde yapmak; ileride yeni oluşabilecek bir kodu tanımlama ve düzenleme açısından ve programın pek çok yerinde yeniden değişiklikler yapma nedeniyle birçok sorun meydana getirir. Bu tablonun alanları da SınavTipiKodu ve SınavSekli olarak tanımlanabilir ve Notlar tablosunda SınavTipi adında bir sütun eklenip buraya V,F,B gibi kodların yazılması gerekir.

## 4.2.4 Normalizasyon Kuralı

---

- Birincil anahtar alanlar ile anahtarı olmayan alanlar arasında, birden fazla bağımsız bire-çok ilişkisine izin verilmez. Örneğin, notlar tablosunda yer alan bir not hem vize, hem final, hem de bütünleme notu olamaz. Bu durumda notlar tablosu nasıl tasarlanabilir? Not ile birlikte bu notun hangi sınav tipine ait olduğu sütunu tanımlanır ve bilgiler ona göre yazılır. 4.Normal formu sağlamak için, her bağımsız bire çok ilişki için ayrı bir tablo oluşturulması gerekir.

## 4.2.5 Normalizasyon Kuralı

---

Tekrarlamaları ortadan kaldırmak için her bir tablonun mümkün olduğunca küçük parçalara bölünmesi gerekir. Aslında ilk 4 kural sonuçta bu işe yarar. Ancak bu kurallar kapsamında olmayan tekrarlamalar da 5 normalizasyon kuralı ile giderilebilir. Örneğin, öğrencilerin okula kayıt olma şekilleri olan OSYM sınavı ile Özel Yetenek Sınavı ile af ile gibi bilgileri içeren bir öğrenci okula kayıt şekli alanı olabilir. Bu kayıt şekilleri de KayıtSekliKodu ve KayıtSekliAdı alanını kapsayan başka bir tabloda tutulabilir. Bu tabloda KayıtSekliKodu için 1 ve KayıtSekliAdı için OSYM sınavı ile veya KayıtSekliKodu için 2 ve KayıtSekliAdı için Özel Yetenek Sınavı ile gibi bilgiler yazılabilir.



- 
- Böylelikle, kullanıcıların bu alana gelişi güzel bilgiler girmesi engellenmiş olur. Bu da sorgulama esnasında veriler arasında bir tutarlılık sağlar. Bu işlem sonucunda, tutarsızlıklara neden olabilecek ve sık tekrarlayan veriler başka bir tabloya taşınmış olur. Bu tablo için, veritabanı programlamada 'look-up table' terimi kullanılır. Ancak veritabanı normalizasyon kuralları, bir ilişkisel veritabanının tasarlanma aşamalarını değil de ilişkisel veritabanında yer alacak kayıtların ilişkisel veritabanı ile uyumlu olup olmadığını denetlemeye yöneliktir. İlişkisel bir veritabanı tasarımı aşağıdaki dört özelliği taşımalıdır.
  - Veri tekrarı yapılmamalıdır,
  - Boş yer mümkün olduğunca az olmalıdır,
  - Veri bütünlüğü sağlanmalıdır,
  - Veriler, aralarında bir ilişki tanımlamaya müsait olmalıdır.



---

Az kolonlu çok sayıda tablo kullanmanın faydaları şunlardır;

- Hızlı sıralama ve index oluşturma,
- Daha fazla clustered index,
- Kısa alanlarda oluşan indexler,
- Insert, update, delete işlemlerinde daha iyi performans,
- Daha az null bilgi ile database içerisinde tutarsızlığı engelleme.

Tablolar arasında çok sayıda bağlantı performansı düşürebilir. Bu nedenle ilişkisel database tasarımında belli kurallara uyulması gerekir. Bunlar:

- Her tablonun bir tanımlayıcısı (Primary Key) olması gerekir,
- Her tablo bir tek konuya ait bilgileri içermelidir,
- Tablolarda boş (null) kolon bulunmamalıdır,
- Tablolarda tekrar eden değer ya da kolon bulunmamalıdır.

## 4.3 İlişkisel Veritabanı Yönetim Sistemleri

---

- Günümüzde kullanımı en yaygın veritabanı ilişkisel veritabanı ve en yaygın veritabanı yönetim sistemleri ilişkisel Veritabanı Yönetim Sistemleri (VTYS) dır. İlişkisel veritabanının en önemli yanı, tablolardan oluşması ve bu tabloların birbiriyle ilişkilerinin olmasıdır. Bu nedenle VTYS'lere "ilişkisel" denilir.

---

Bir veritabanında ilişkiden söz edebilmek için veritabanında en az iki tablonun olması gerekir ve bu iki tablodaki verilerin birbiri ile bir şekilde ilişkilendirilebiliyor olması gerekir. Öğrenci Bilgileri ve Notlar tablosunda öğrenci numaraları arasında bir ilişki vardır ve öğrenci bilgisi tablosunda olmayan bir öğrencinin Notlar tablosunda not bilgisi olamaz. Aynı şekilde Notlar tablosunda kayıtlı bir öğrencinin numarası alınarak öğrenci bilgileri tablosundaki öğrenci hakkında ad, soyad gibi pek çok bilgi elde edilebilir.

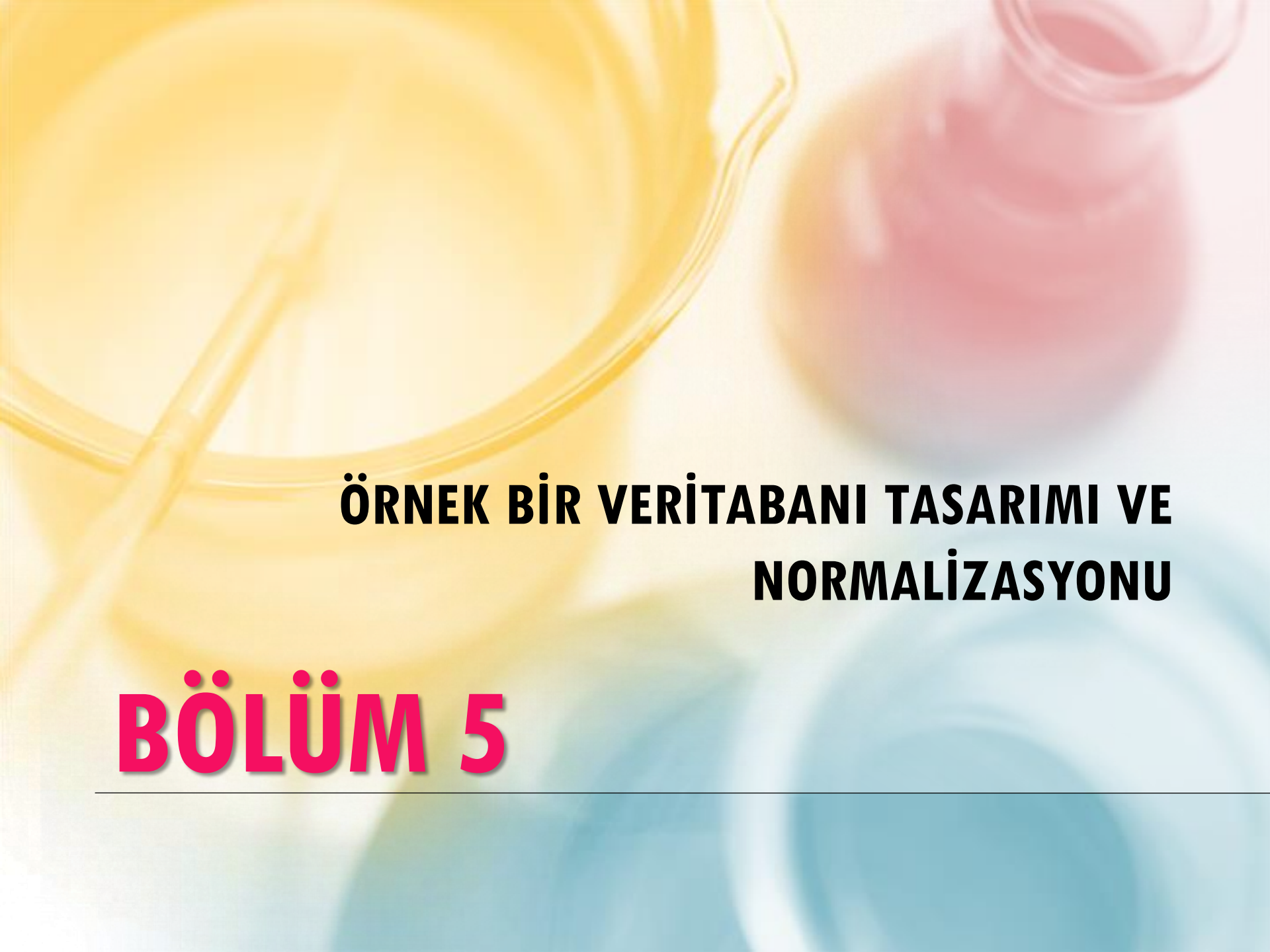
İlişkisel veritabanı yönetim sistemi tablolar ile ilgili aşağıdaki üç işlevi yerine getirmek zorundadır.

1- Seçme, herhangi bir tabloda (listede) yer alan tüm bilgileri gösterebilmelidir. Örneğin, öğrenci tablosundaki öğrencilerin hepsinin tüm bilgilerini listeleyebilmeli veya öğrenciler içerisinde sınıf, fakülte, bölüm, cinsiyet gibi istenilen kritere uyanları listeleyebilmeli.

- 
- **2-** İzdüşürme, herhangi bir tabloda sadece belli sütunların yer aldığı seçme işlevlerini yerine getirebilmelidir. Örneğin, öğrenci bilgileri tablosundan sadece öğrenciye ait numara, ad, soyad ve sınıf bilgisi seçilebilmelidir.
  - **3-** Birleştirme, birden fazla tabloda yer alan bilgiler, yeri geldiğinde tek bir tabloyu muş gibi sunulabilmelidir. Örneğin, tüm notlarının ortalaması 2.00'den küçük olan ve sınıf tekrarı yapacak öğrencilerin kişisel bilgileri, not bilgileri ve harç ödeme bilgileri tek bir tablo bilgisi gibi görüntülenebilir. Öğrencinin kişisel bilgileri bir tabloda, not bilgileri bir tabloda ve harç bilgileri de bir başka tabloda bulunmaktadır.

---

■ VTYS bu 3 temel işlevi yerine getirebilmelidir. Bunlardan üçü, ikisi veya biri aynı anda kullanılabilirdiği gibi teker teker veya ikili kombinasyon şeklinde de kullanılabilir. Örneğin, not ortalaması 3.00'den büyük öğrencilerin numara, ad ve soyad bilgileri listelenirse, hem izdüşürme hem de seçme işlemine ihtiyaç duyulur. Veriler ve depolanma şekilleri farklı olabilir. Önemli olan, VTYS'nin SQL ile yönetilebilir olmasıdır. Böylece kullanıcı verilerin bilgisayarda fiziksel olarak ne şekilde depolandığını bilmek zorunda değildir. SQL ile kullanıcı temel veri saklama işlem ve yöntemlerinden izole edilmiş olur. Kullanıcının verileri etkili olarak kullanması için iyi SQL biliyor olması gerekir.

The background of the slide is a blurred image of laboratory glassware. On the left, there is a large Erlenmeyer flask containing a yellow liquid. To its right, there is a smaller flask containing a pink liquid. In the foreground, there are several blue petri dishes. The overall image is out of focus, creating a soft, scientific atmosphere.

# **ÖRNEK BİR VERİTABANI TASARIMI VE NORMALİZASYONU**

## **BÖLÜM 5**

---



## 5.1. ÖRNEK BİR VERİTABANI TASARIMI

---

- Şimdi örnek bir veritabanı tasarlanacak ve veritabanı normalizasyon işlemleri yapılacaktır. Örnek proje bir üniversitede bulunan teknik eğitim fakültesindeki öğrencilerin, hocaların bilgilerini ve not işlemlerini kapsamaktadır. Burada okul projesindeki notlara ilişkin bir kısım işlemler örnek verilecektir. Unutulmamalıdır ki okulda öğrenci işlemleri çok daha fazla tablo, sütun v.b. işlemlerden meydana gelmektedir. Burada konunun anlaşılması açısından projenin bir kısmı üzerinde çalışılacaktır.

- Örnek projemizin adı okul veritabanı ve projemiz 7 adet tablodan meydana gelmektedir. Bunlar; bölüm, ünvan, ders, hocalar, memleket, not ve öğrenci bilgilerinin tutulduğu tablodur.

| Tablo Adı | Tablo Alanları                                                     |
|-----------|--------------------------------------------------------------------|
| Oğrenci   | no, adi, soyadi, bolkod, sınıf, h_ID, d_tarihi, memleket, cinsiyet |
| Hocalar   | h_ID, h_adi, h_soyadi, ünvan, top_ders_saati                       |
| Notlar    | no, op_kod, vize, final, büt                                       |
| Dersler   | op_kod, ders_kod, ders_adi, dönem, teori, pratik, hocası           |
| Bölüm     | bolkod, bol_adi                                                    |
| Memleket  | tr_kod, m_adi, tel_kod                                             |
| Unvan     | Un_kod, unvani                                                     |

**Tablo 5.1: Örnek Veritabanı Tasarımı**



- Bu tablolar ve bu tablolara ait alan (sütun) bilgileri aşağıda listelendiği gibi düzenlenmiştir. Veritabanlarında tablolar matris yapısında tutulurlar. Sütunlar tablo alanlarını, satırlar ise tablodaki kayıtları (record) teşkil ederler. Burada normalizasyon kurallarına göre bilgiler ayrı tablolarda tutulmuşlardır. Cinsiyet ve memleket için iki ayrı tablo tanımlanmıştır. Bunun nedeni öğrenci tablosunda cinsiyet ve memleket alanına bilgi yazılırken veri bütünlüğü ve standart sağlanabilmesi içindir. Bu tür bilgi girişlerinde ayrı tabloların tanımlanması gerektiği 3.normalizasyon kuralında belirtilmişti.

- Veritabanında bu kural çok önemlidir. Cinsiyet personel tablosuna yazılırken kod şeklinde tanımlanmayıp; öğrenci tablosunda isim olarak yazılsaydı bilgilerde sorun yaşanırdı. Örneğin; “Bayan” yazılırken kullanıcı bu bilgiyi kendine göre kısaltarak veya büyük küçük harfleri dikkate almayarak yazabilir. Bir kişi için “bayan” yazarken, bir diğeri için “Kız”, bir diğeri için “KIZ” veya “Bayan” yazabilir. Bu durumda veritabanındaki bu 4 bilgi birbirinden farklı bilgiler olur. Daha sonra “Bayan” olan kişilerin sorgusu yapılacak olursa, bu 4 bilgi birbirinden farklı olacağı için doğru sonuç elde edilemeyecektir. Cinsiyetler isim olarak yazılmayıp kodlanarak yazılmalıdır. Bu alana bayan için 0 gibi bir kodlama yapılır ve bu alana 0 bilgisi yazılırsa cinsiyeti 0 olanların sorgusu için elde edilecek veriler doğru olur ve 4 kişiye ait bilgi gelir. Bu nedenle cinsiyet tablosu ayrı oluşturulmalı ve kodlama işlemi yapılmalıdır.



- Öğrenci tablosuna cinsiyet yazılırken cinsiyet değil, cinsiyet tablosunda tanımlanmış cinsiyet yazılmalıdır. Aynı işlem memleket için yapılmış ve ayrı bir tablo oluşturularak memleket alanı kodlanmıştır. Örneğin; memleketi “Amasya” olan öğrenciye ait bilgiler tabloya yazılırken kullanıcı bunu da kendine göre kısaltarak veya büyük küçük harfleri dikkate almayarak yazabilir. Bir kişi için “AMASYA” yazarken, bir diğeri için “Amasya”, bir diğeri için “05” veya “Amasya” yazabilir. Veritabanında bu 4 bilgi birbirinden farklı bilgilerdir. Daha sonra “Amasya” memleketi olan öğrencilerin sorgusu yapılacak olursa bu 4 bilgi birbirinden farklı olacağı için doğru bir sonuç elde edilemeyecektir. Memleket isim olarak yazılıp bu alana Amasya için 5 gibi bir kodlama yapılır ve bu alana 5 bilgisi yazılırsa çalıştığı birim 5 olanların sorgusu için elde edilecek veriler doğru olacak ve ekrana 4 kişiye ait bilgi gelecektir.

- Veritabanında cinsiyet, memleket gibi birçok bilgi için standart olan bilgi girişı yapılacak alanlarda kodlama yapılmalıdır. Bu hem bilgilerin daha kısa sürede yazılmasını sağlayacak, hem de verilerde bütünlük ve tutarlılık sağlayacaktır. Bu kodlamalar numara veya harflerden veya her ikisinin karışımından oluşabilir. Daha önce anlatılan 3.normalizasyon kuralındaki Sınav Tipi alanı gibi. Sınav tipinde vize için V, Final için F ve Bütünleme için B kodlamasının kullanılması gibi. Dikkat edilmesi gereken nokta kodlamanın kısa karakterlerden oluşmasıdır.



- Bir diğer önemli özellik ise farklı tablolardaki aynı amaç için kullanılan alanların aynı isimle isimlendirilmesidir. Dersler tablosundaki op\_kod ile notlar tablosundaki op\_kod aynı isim ve aynı amaçla kullanılmıştır. Dersler tablosundaki op\_kod ile notlar tablosundaki op\_kod aynı isim ve aynı amaçla kullanılmıştır. Yine öğrenci tablosundaki bolkod ile bölüm tablosundaki bolkod aynı isim ve aynı amaçla kullanılmıştır.

- Aşağıda projemizdeki tabloların arasındaki ilişki (bağlantı) verilmiştir.

| Tablo Adı.Alanı | İlişkili TabloAdı.Alanı |
|-----------------|-------------------------|
| notlar.no       | ogrenci.no              |
| notlar.op_kod   | dersler.op_kod          |
| memleket.tr_kod | ogrenci.memleket        |
| ogrenci.h_ID    | hocalar.h_ID            |
| ogrenci.bolkod  | bolun.bolkod            |

**Tablo 5.2: Örnek Projedeki Tablo ilişkileri**

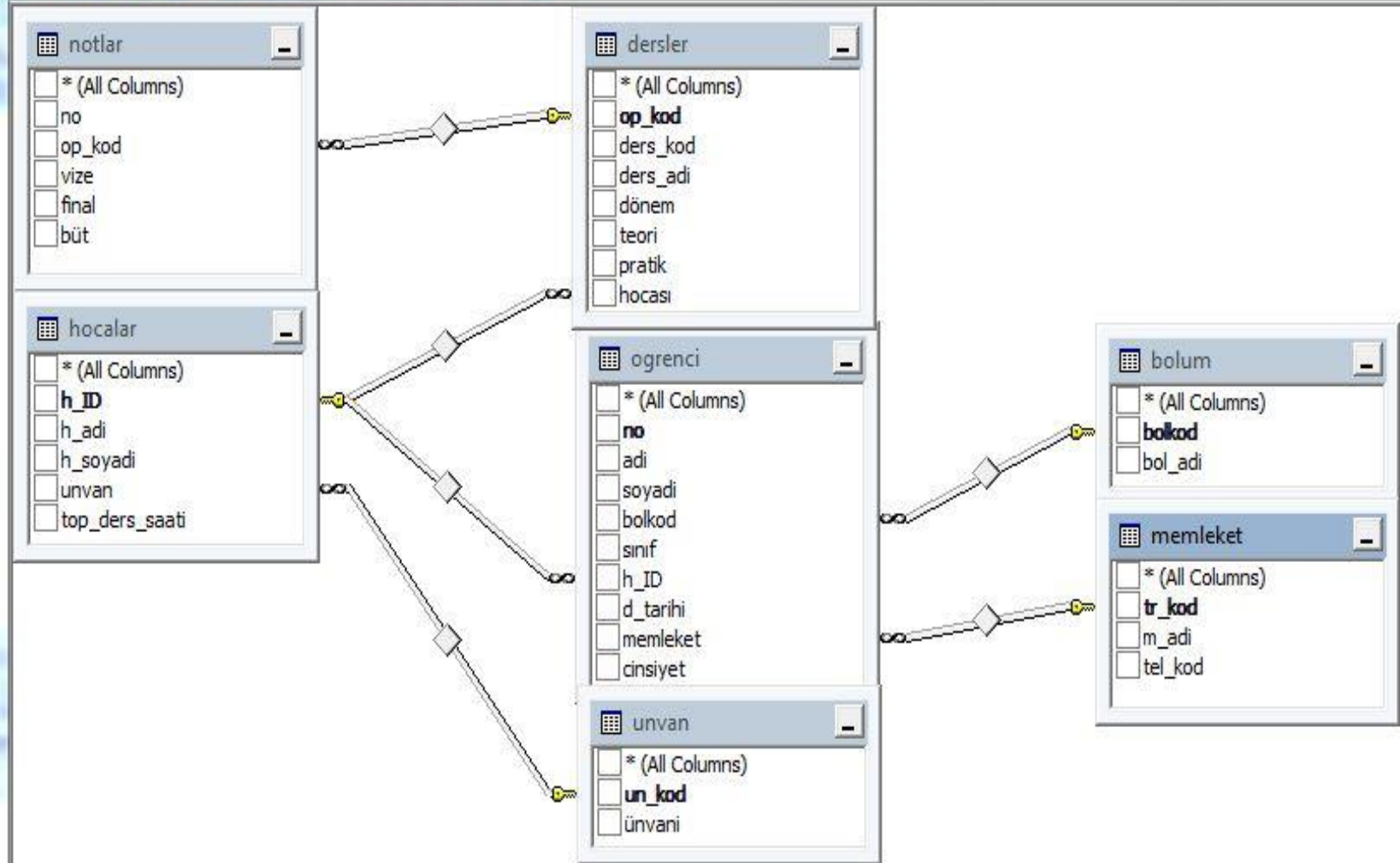
- Projede bir diğer tanımlanması gereken nokta da tablo için tanımlanan alanların (sütunların) hangi veri tipinden ve kaç karakterden oluşacağıdır.

| Tablo Adı | Tablo Alanları | Veri Tipi    |
|-----------|----------------|--------------|
| bolum     | bol_kod        | Sayı         |
|           | bol_adi        | Karakter(15) |
| memleket  | tr_kod         | Sayı(2)      |
|           | m_adi          | Karakter(15) |
|           | tel_kod        | Sayı(3)      |
| hocalar   | h_ID           | Sayı         |
|           | h_adi          | Karakter(20) |
|           | h_soyadi       | Karakter(20) |
|           | ünvan          | Sayı         |
|           | top_ders_saati | Sayı         |
| dersler   | op_kod         | Sayı         |
|           | ders_kod       | Karakter(10) |
|           | ders_adi       | Karakter(20) |
|           | dönem          | Sayı         |
|           | teori          | Sayı         |
|           | pratik         | Sayı         |
|           | hocası         | Sayı         |

|         |          |              |
|---------|----------|--------------|
| ogrenci | no       | Sayı         |
|         | adi      | Karakter(20) |
|         | soyadi   | Karakter(20) |
|         | bol_kod  | Sayı         |
|         | Sınıf    | Sayı         |
|         | h_ID     | Sayı         |
|         | d_tarihi | Tarih        |
|         | memleket | Sayı         |
|         | cinsiyet | Karakter(5)  |
| notlar  | no       | Sayı         |
|         | op_kod   | Sayı         |
|         | vize1    | Sayı         |
|         | fmal     | Sayı         |
|         | büt      | Sayı         |
| unvan   | Un_kod   | Sayı         |
|         | unvani   | Karakter(11) |

**Tablo 5.3:** Örnek Projedeki Veri Alanlarının Tipleri

## Query Designer



Şekil 5.1: Örnek Projenin SQL Server 2008’de ki ilişkilendirme Diyagramı



- Yedi tablo ve tablolar arasındaki ilişkiler bir önceki slayttaki şekilde gösterilmiştir. Dikdörtgen kutu içerisine alınan ve ok işareti ile gösterilen bölge tablolar arasındaki ilişkileri belirtmektedir. Aynı diyagramın bir benzeri SQL-Server veya Oracle veritabanlarında da hazırlanabilir. Bu veri tabanının tanımlanması için her dosya içindeki kayıt yapıları, kayıtlar içindeki bulunan alanlar ve bu alanların tipleri belirlenmelidir. Farklı dosyalar içindeki alanlar birbirleri ile ilişkili olabilirler. Ayrıca veri tabanı üzerinde bir işlem yapıldığında birçok dosyaya birden erişim gerekebilir.

- ÖĞRENCİ dosyasından öğrencinin adı, soyadı bilgileri listelenmek istenirse bunun için gerekli komut yazılır.

**SELECT** adi, soyadi **FROM** ogrenci çalıştırılrsa;

- VTYS bu talebi alır ve analiz eder.
- OGRENCI dosyası ilişkili erişim talebi ve iki alan talebi(gerektiği) tespit edilir.
- VTYS veri sözlüğünden gerekli kayıtları içeren disk dosyalarının yerini öğrenir.
- VTYS gerekli kayıtları diskten okumak üzere kontrolü işletim sistemine bırakır.
- İşletim sistemi gerekli bilgiyi alır. Buffer'a aktarır ve kontrolü VTYS'ye bırakır.
- VTYS veriyi buffer'dan alarak kullanıcının çalışma alanına gönderir.
- İstenilen liste görüntülenir.

# BÖLÜM 6

## SQL SERVER KOMUTLARI

# 6.1. CREATE (YARAT) KOMUTU

- Database, tablo, index, view vb. veri tabanı objelerini yaratmada kullanılan komuttur. SQL komutları ile veri tabanında işlem yapılabilmesi için önce veri tabanı sonra da veri tabanında kullanılacak tablolar tanımlanmalıdır. SQL Server'da veritabanı oluşturmak için CREATE DATABASE deyimi kullanılır.
- Kullanılışı: CREATE DATABASE isim;
- Bu komut isim ile belirtilen isimde bir veri tabanı oluşturur. Bir veri tabanı içerisinde çok sayıda veri tabanı kütüğü ya da tablo bulunabilir. Her tabloda saklanan verilerle ilgili alanlar bulunmalıdır.
- **Örnek**: Deneme, okul, sirket adında veritabanı oluşturan komutları yazınız.
- Create Database deneme;
- Create Database okul;
- Create Database sirket;

## 6.2. VERİ TABANINI AKTİF YAPMA

- USE komutu kullanılır. Önceden yapılmış veri tabanını aktif hale getirir.
- **Örnek:** Deneme, okul ve şirket adında oluşturduğunuz veritabanını aktif yapan komutları yazınız.
- Use deneme;
- Use okul;
- Use şirket;



## 6.3. TABLO OLUŞTURMA

CREATE TABLE deyimi kullanılır. Kullanılacak tüm alanlar bu deyim içerisinde belirtilmelidir.

**Örnek:** Yukarı da bahsedilen örnek okul veritabanı projesinin veritabanını oluşturup, aktif yapıp, bahsedilen tabloları oluşturan komutları yazınız.(Tablo 5.1, 5.2, 5.3'den yararlanarak)

Create Database proje

Use proje

Create table bolum(bolkod smallint primary key , bol\_adi char(15))

Create table unvan(un\_kod tinyint primary key , ünvanı char(11))

Create table hocalar(h\_ID tinyint primary key , h\_adi char(20) , h\_soyadı char(20) , unvan tinyint foreign key references unvan(un\_kod) , top\_ders\_saati tinyint)

Create table dersler(op\_kod smallint primary key , ders\_kod char(10) , ders\_adi char(20) , dönem tinyint , teori tinyint , pratik tinyint , hocası tinyint foreign key references hocalar(h\_ID))

Create table notlar(no int , op\_kod smallint foreign key references dersler(op\_kod) , vize tinyint , check(vize between 0 and 100) , final tinyint , check(final between 0 and 100) , but tinyint , check(but between 0 and 100) )

Create table memleket(tr\_kod tinyint primary key , m\_adi char(15) , tel\_kod smallint)

Create table ogrenci(no int primary key , adi char(20) , soyadı char(20) , bolkod smallint foreign key references bolum(bolkod) , sınıf tinyint , check( sınıf IN(1,2,3,4)) , h\_ID tinyint foreign key references hocalar(h\_ID) , d\_tarihi datetime , memleket tinyint foreign key references memleket(tr\_kod) , cinsiyet char(5) )

## 6.4. TABLOLARA VERİ YÜKLENMESİ

Bir tabloya veri girişi (ya da veri yüklenmesi) işlemi için, SQL'de mevcut komut INSERT INTO / VALUES komutudur.

**Örnek:** Yukarıdaki örnekte oluşturduğunuz tablolara birer kayıt giriniz.

`insert into bolum values(536,'Yapı-Resim');`

**Tablo 6.1:** Bölüm Tablosu

| bol_kod | bol_adi    |
|---------|------------|
| 530     | Elektronik |
| 531     | Bilgisayar |
| 532     | Metal      |
| 533     | Talaşlı    |
| 534     | Makine     |
| 535     | Yapı       |
| 536     | Yapı-Resim |

`insert into unvan values(1,'Prof.Dr.');`

**Tablo 6.2:** Ünvan Tablosu

| un_kod | ünvani      |
|--------|-------------|
| 1      | Prof.Dr.    |
| 2      | Doç.Dr.     |
| 3      | Yrd.Doç.Dr. |
| 4      | Öğr.Gör.    |
| 5      | Öğr.Gör.Dr. |
| 6      | Arş.Gör.    |

insert into dersler values(101,'tde 102','Türkdili',2,2,2,12)

**Tablo 6.3: Dersler Tablosu**

| op_kod | ders_kod | ders_adi             | dönem | teori | pratik | hocası |
|--------|----------|----------------------|-------|-------|--------|--------|
| 101    | tde 102  | Türkdili             | 2     | 2     | 2      | 12     |
| 104    | tdi 132  | İngilizce            | 2     | 2     | 0      | 16     |
| 114    | bil 166  | Nesne yön.prg.gr     | 2     | 2     | 2      | 14     |
| 115    | bil 168  | Bilgisayar donanımı  | 2     | 2     | 0      | 5      |
| 118    | egt 162  | Okul deneyimi        | 2     | 1     | 4      | 3      |
| 151    | tde 101  | Türk dili            | 1     | 2     | 0      | 12     |
| 155    | bil 161  | Temel bil.tek.kul.   | 1     | 2     | 2      | 11     |
| 164    | mat 167  | Matematik            | 1     | 3     | 0      | 12     |
| 167    | egt 171  | Öğrt.mesleğine giriş | 1     | 3     | 0      | 3      |
| 218    | elt 292  | Elk.devreler         | 2     | 3     | 2      | 6      |
| 219    | bil 282  | Meslek mat.          | 2     | 3     | 0      | 1      |
| 222    | bil 288  | Bil.prg.2            | 2     | 2     | 2      | 13     |
| 270    | bil 285  | Mantık devreler      | 1     | 3     | 0      | 4      |
| 271    | bil 287  | Bil.prg.             | 1     | 3     | 2      | 15     |
| 274    | elt 291  | Devre analizi        | 1     | 3     | 2      | 7      |
| 317    | egt 372  | Sınıf yönetimi       | 2     | 2     | 2      | 8      |
| 324    | bil 384  | Çoklu ortam          | 2     | 2     | 2      | 11     |
| 326    | bil 388  | SQL                  | 2     | 3     | 2      | 14     |
| 371    | bil 383  | Teknik iletişim      | 1     | 2     | 0      | 11     |
| 372    | bil 385  | Mikro. mimarisi      | 1     | 3     | 2      | 5      |
| 373    | bil 387  | Veri yapılan         | 1     | 3     | 0      | 13     |
| 374    | elt 395  | Ölçme                | 1     | 2     | 2      | 6      |
| 375    | bil 391  | İşletim sistemleri   | 1     | 3     | 2      | 10     |
| 421    | egt 472  | Rehberlik            | 2     | 3     | 0      | 9      |
| 433    | egt 484  | Bil.açılan           | 2     | 3     | 0      | 8      |
| 434    | egt 486  | Web Prog.            | 2     | 3     | 2      | 13     |
| 436    | egt 490  | Proje 2              | 2     | 0     | 2      | 8      |
| 467    | egt 471  | Okul deneyimi        | 1     | 1     | 4      | 9      |
| 481    | egt 483  | Bil.açılan2          | 1     | 3     | 2      | 1      |

|     |         |              |   |   |   |    |
|-----|---------|--------------|---|---|---|----|
| 482 | egt 485 | Web tasarımı | 1 | 3 | 2 | 14 |
| 483 | egt 487 | Yapay zeka   | 1 | 3 | 0 | 13 |
| 484 | egt 489 | Proje        | 1 | 0 | 2 | 8  |



insert into hocalar values(1,'Asaf','Varol',1,20)

**Tablo 6.4:** Hocalar Tablosu

| h_ID | h_adi      | h_soyadi   | unvan | top_ders_saati |
|------|------------|------------|-------|----------------|
| 1    | Asaf       | Varol      | 1     | 20             |
| 2    | Muammer    | Gökbulut   | 1     | 20             |
| 3    | Hakan      | Akpolat    | 1     | 20             |
| 4    | Hanifi     | Güldemir   | 1     | 20             |
| 5    | İbrahim    | Türkoğlu   | 2     | 18             |
| 6    | Mehmet     | Gedikpınar | 3     | 17             |
| 7    | Servet     | Tuncer     | 3     | 17             |
| 8    | Engin      | Avcı       | 3     | 17             |
| 9    | Davut      | Hanbay     | 3     | 17             |
| 10   | Abdülkadir | Şengür     | 3     | 17             |
| 11   | Nurhayat   | Varol      | 4     | 15             |
| 12   | Cafer      | Bal        | 5     | 16             |
| 13   | Erkan      | Tanyıldızı | 5     | 16             |
| 14   | Murat      | Karabatak  | 4     | 15             |
| 15   | Ferhat     | Bahçacı    | 6     | 12             |
| 16   | Korhan     | Kayışlı    | 6     | 12             |

insert into memleket values(23,'Elazığ',424)

**Tablo 6.5: Memleket Tablosu**

| tr_kod | m_adi      | tel_kod |    |            |     |    |           |     |
|--------|------------|---------|----|------------|-----|----|-----------|-----|
| 1      | Adana      | 322     | 28 | Giresun    | 454 | 55 | Samsun    | 362 |
| 2      | Adıyaman   | 416     | 29 | Gümüşh...  | 456 | 56 | Siirt     | 484 |
| 3      | Afyon      | 272     | 30 | Hakkari    | 438 | 57 | Sinop     | 368 |
| 4      | Ağrı       | 472     | 31 | Hatay      | 326 | 58 | Sivas     | 346 |
| 5      | Amasya     | 382     | 32 | Isparta    | 246 | 59 | Tekirdağ  | 282 |
| 6      | Ankara     | 312     | 33 | İçel       | 324 | 60 | Tokat     | 356 |
| 7      | Antalya    | 242     | 34 | İstanbul   | 216 | 61 | Trabzon   | 462 |
| 8      | Artvin     | 466     | 35 | İzmir      | 232 | 62 | Tunceli   | 428 |
| 9      | Aydın      | 256     | 36 | Kars       | 474 | 63 | Şanlıurfa | 414 |
| 10     | Balıkesir  | 266     | 37 | Kastamo... | 366 | 64 | Uşak      | 276 |
| 11     | Bilecik    | 228     | 38 | Kayseri    | 352 | 65 | Van       | 432 |
| 12     | Bingöl     | 426     | 39 | Kırklareli | 288 | 66 | Yozgat    | 354 |
| 13     | Bitlis     | 434     | 40 | Kırşehir   | 386 | 67 | Zonguldak | 372 |
| 14     | Bolu       | 374     | 41 | Koceli     | 262 | 68 | Aksaray   | 382 |
| 15     | Burdur     | 248     | 42 | Konya      | 332 | 69 | Bayburt   | 458 |
| 16     | Bursa      | 224     | 43 | Kütahya    | 247 | 70 | Karaman   | 338 |
| 17     | Çanakkale  | 286     | 44 | Malatya    | 422 | 71 | Kırıkkale | 318 |
| 18     | Çankırı    | 376     | 45 | Manisa     | 236 | 72 | Batman    | 488 |
| 19     | Çorum      | 364     | 46 | K.Maraş    | 344 | 73 | Şırnak    | 486 |
| 20     | Denizli    | 258     | 47 | Mardin     | 482 | 74 | Bartın    | 378 |
| 21     | Diyarbakır | 412     | 48 | Muğla      | 252 | 75 | Ardahan   | 478 |
| 22     | Edirne     | 284     | 49 | Muş        | 436 | 76 | İğdır     | 476 |
| 23     | Elazığ     | 424     | 50 | Nevşehir   | 384 | 77 | Yalova    | 226 |
| 24     | Erzincan   | 446     | 51 | Niğde      | 388 | 78 | Karabük   | 370 |
| 25     | Erzurum    | 442     | 52 | Ordu       | 452 | 79 | Kilis     | 348 |
| 26     | Eskişehir  | 222     | 53 | Rize       | 464 | 80 | Osmaniye  | 328 |
| 27     | Gaziantep  | 342     | 54 | Sakarya    | 264 | 81 | Düzce     | 111 |

insert notlar values(7536511,274,53,80,40)

**Tablo 6.6:** Notlar Tablosu

| no      | op_kod | vize | final | büt |
|---------|--------|------|-------|-----|
| 7536511 | 274    | 53   | 80    | 40  |
| 7536511 | 218    | 98   | 76    | 54  |
| 7536521 | 274    | 21   | 78    | 63  |
| 7536521 | 222    | 57   | 90    | 65  |
| 7536545 | 274    | 34   | 23    | 60  |
| 7536545 | 271    | 12   | 11    | 30  |
| 6532510 | 373    | 11   | 47    | 45  |
| 6532510 | 372    | 60   | 66    | 88  |
| 6532510 | 371    | 23   | 45    | 67  |
| 5531519 | 433    | 63   | 70    | 90  |
| 5531519 | 421    | 40   | 66    | 96  |
| 5530535 | 433    | 38   | 60    | 100 |
| 5530535 | 421    | 50   | 80    | 77  |

insert into ogrenci

values(7536545,'Cengiz','Güneş',536,2,13,'04.11.1988',66,'Erkek')

**Tablo 6.7:** Ogrerci Tablosu

| no      | adi    | soyadi    | bolkod | sınıf | h_ID | d_tarihi   | memleket | cinsiyet |
|---------|--------|-----------|--------|-------|------|------------|----------|----------|
| 5530535 | Feyza  | Uçan      | 530    | 4     | 7    | 1985-11-16 | 55       | Bayan    |
| 5531519 | Betül  | Cebe      | 531    | 4     | 5    | 1986-06-01 | 5        | Bayan    |
| 6532510 | Leyla  | Korkmaz   | 532    | 3     | 1    | 1986-09-11 | 1        | Bayan    |
| 6532518 | Ali    | Yılmaz    | 532    | 3     | 1    | 1987-08-13 | 40       | Erkek    |
| 6535501 | Bilal  | Baymaz    | 535    | 3     | 8    | 1987-07-21 | 28       | Erkek    |
| 7536511 | Aytül  | Çetinkaya | 536    | 2     | 13   | 1987-05-09 | 70       | Bayan    |
| 7536521 | Veli   | Uşak      | 536    | 2     | 13   | 1988-07-03 | 21       | Erkek    |
| 7536545 | Cengiz | Güneş     | 536    | 2     | 13   | 1988-04-11 | 66       | Bayan    |

## 6.5. TEK TABLO İÇERİSİNDE SORGULAMA YAPMA

- SQL Server'da, tek bir tablo içinde çeşitli kriterlere göre bilgi sorgulama, bilgiyi sıralı elde etme, bilgi özetleme, ortalama vb. gibi matematiksel işlemleri gerçekleştirmeyi sağlayan komut ve fonksiyonlar vardır. Ayrıca, birden çok tabloyu birlikte ele alarak sorgulama gerçekleştirmek de mümkündür.

## 6.5.1.SELECT KOMUTU

Tablo üzerinden gerekli bilgileri elde etmek ve sorgulama yapmak için SELECT komutu kullanılır.

Kullanılışı:

**SELECT** \* **FROM** tablo\_adi

Tablo adı ile belirtilen tablo içindeki tüm bilgiler koşulsuz olarak listelenecektir. SELECT sözcüğünü izleyen kısımda “ \* ” karakterinin bulunması, ilgili tablodaki bütün alan isimlerinin ve bu alanlardaki bilgilerin listelenmesini sağlayacaktır. FROM ifadesinden sonra ise kullanılacak tablo adı belirtilebilir.

SELECT komutunun genel kullanımı;

SELECT [ DISTINCT | ALL ] < sütun (lar) > FROM <tablo adı (lar)>

[ WHERE < şart (lar) > ]

[ GROUP BY < sütun (lar) > ]

[ HAVING < grup kısıtlaması > ]

[ ORDER BY <sütun (lar) [ASC | DESC >] ]

**Örnek:** Öğrenci tablosunun tüm verileri gösteren sorguyu yazınız.

```
SELECT * FROM ogrenci;
```

**Örnek:** Öğrenci tablosundaki sadece adı kolonundaki tüm verileri veren sorguyu yazınız.

```
SELECT adi FROM ogrenci;
```

**Örnek:** Öğrenci tablosundaki adı ve soyadı kolonundaki tüm verileri veren sorguyu yazınız.

```
SELECT adi, soyadi FROM ogrenci;
```

**Örnek:** Dersler tablosundaki op\_kod ve ders\_adi kolonundaki tüm verileri veren sorguyu yazınız.

```
SELECT op_kod, ders_adi FROM dersler;
```

## 6.5.2.DISTINCT İFADESİ

Birbirleriyle aynı olan satırların, listeleme esnasında bir kez yazılmasını sağlayan ifadedir.

**Örnek:** `SELECT DISTINCT` adi `FROM` ogrenci komutu ne işe yarar.

Öğrenci tablosunda adı aynı olan kişilerden sadece bir tanesi listelenir.

**Örnek:** `SELECT DISTINCT` yasi, bolumu `FROM` businif komutu ne işe yarar.

Bu sinif tablosunda yaşı aynı olan kişilerden sadece bir tanesini ve bölümünü listeler.



**Örnek:** Notlar tablosunda op\_kod aynı olanlardan sadece bir tanesini listeleyen sorgu ve çıktısını yazınız. (Okul projesine göre)

`select distinct op_kod from notlar`

|   | op_kod |
|---|--------|
| 1 | 218    |
| 2 | 222    |
| 3 | 271    |
| 4 | 274    |
| 5 | 371    |
| 6 | 372    |
| 7 | 373    |
| 8 | 421    |
| 9 | 433    |

**Örnek:**

| Adi    | Soyadi    |
|--------|-----------|
| Sibel  | Tanyol    |
| Gülşah | Yonar     |
| Demet  | Tanka     |
| Meral  | Tekinemre |
| Sibel  | Erdem     |

Yandaki tabloya göre `Select DISTINCT Adi from ogrenci` sorgusunun çıktısını yazınız.

**Çıktı:**

Adi

-----

Sibel

Gülşah

Demet

Meral

## 6.5.3.ORDER BY İFADESİ

- Listelenecek bilgilerin belirli bir alan adına göre sıralanmasını sağlamak için kullanılan komuttur.
- **ASC:** Sözcüğü nota göre sıralamanın artan olarak yapılmasını sağlar.
- **DESC:** Nota göre sıralamanın azalan olarak yapılmasını sağlar.
- Ayrıca tablo içindeki veriler sıralanırken aynı anda birden fazla alana göre sıralama yapmak mümkündür.

**Örnek:** Notlar tablosundaki kişileri aldıkları final notuna göre listelemek için(artan) gerekli sorguyu yazınız.(Okul projesine göre)

`select * from notlar order by final asc`

|    | no      | op_kod | vize | final | büt |
|----|---------|--------|------|-------|-----|
| 1  | 7536545 | 271    | 12   | 11    | 30  |
| 2  | 7536545 | 274    | 34   | 23    | 60  |
| 3  | 6532510 | 371    | 23   | 45    | 67  |
| 4  | 6532510 | 373    | 11   | 47    | 45  |
| 5  | 5530535 | 433    | 38   | 60    | 100 |
| 6  | 5531519 | 421    | 40   | 66    | 96  |
| 7  | 6532510 | 372    | 60   | 66    | 88  |
| 8  | 5531519 | 433    | 63   | 70    | 90  |
| 9  | 7536511 | 218    | 98   | 76    | 54  |
| 10 | 7536521 | 274    | 21   | 78    | 63  |
| 11 | 7536511 | 274    | 53   | 80    | 40  |
| 12 | 5530535 | 421    | 50   | 80    | 77  |
| 13 | 7536521 | 222    | 57   | 90    | 65  |

**Örnek:** Notlar tablosundaki kişileri aldıkları final notuna göre listelemek için(azalan) gerekli sorguyu yazınız.(okul projesine göre)

`select * from notlar order by final desc`

|    | no      | op_kod | vize | final | büt |
|----|---------|--------|------|-------|-----|
| 1  | 7536521 | 222    | 57   | 90    | 65  |
| 2  | 7536511 | 274    | 53   | 80    | 40  |
| 3  | 5530535 | 421    | 50   | 80    | 77  |
| 4  | 7536521 | 274    | 21   | 78    | 63  |
| 5  | 7536511 | 218    | 98   | 76    | 54  |
| 6  | 5531519 | 433    | 63   | 70    | 90  |
| 7  | 5531519 | 421    | 40   | 66    | 96  |
| 8  | 6532510 | 372    | 60   | 66    | 88  |
| 9  | 5530535 | 433    | 38   | 60    | 100 |
| 10 | 6532510 | 373    | 11   | 47    | 45  |
| 11 | 6532510 | 371    | 23   | 45    | 67  |
| 12 | 7536545 | 274    | 34   | 23    | 60  |
| 13 | 7536545 | 271    | 12   | 11    | 30  |

## Örnek:

| No | Adı     | Soyadı    |
|----|---------|-----------|
| 1  | Ali     | Gel       |
| 2  | Mehmet  | Kaya      |
| 3  | Murat   | Karabatak |
| 4  | Mustafa | Kara      |
| 5  | Mustafa | Kaya      |

Yandaki tabloya göre **order by** ifadesini kullanarak yapılan sorgular ve bu sorgular sonucunda elde edeceğimiz sıralamalar aşağıdaki gibi olur.

## Çıktı-1:

**SELECT \* FROM** ogrenci **ORDER BY** adi, soyadi

| no  | adi     | soyadi    |
|-----|---------|-----------|
| --- | -----   | -----     |
| 1   | Ali     | Gel       |
| 2   | Mehmet  | Kaya      |
| 3   | Murat   | Karabatak |
| 4   | Mustafa | Kara      |
| 5   | Mustafa | Kaya      |

- **Çıktı-2:**

- **SELECT** adi,soyadi **FROM** ogrenci **ORDER BY** adi, soyadi **DESC**

- adi                      soyadi

- ----                      -----

- Ali                      Gel

- Mehmet              Kaya

- Murat                Karabatak

- Mustafa             Kaya

- Mustafa             Kara

- **Çıktı-3:**

- **SELECT** \* **FROM** ogrenci **ORDER BY** adi **desc**, Soyadi **ASC**

- no      adi                      soyadi

- ---      -----      -----

- 4      Mustafa              Kara

- 5      MustafaKaya

- 3      Murat                Karabatak

- 2      Mehmet              Kaya

- 1      Ali                      Gel

## 6.5.4. KARŞILAŞTIRMA İFADELERİ

| İfade | Sembol     |
|-------|------------|
| >     | Büyük      |
| <     | Küçük      |
| =     | Eşit       |
| >=    | Büyük eşit |
| <=    | Küçük eşit |
| <>    | Eşit değil |

**Tablo 6.8:** Karşılaştırma İfadeleri

(NOT, OR, AND), birden fazla koşula göre sıralama işlemlerinde bu ifadeler kullanılır.

## 6.5.5. WHERE İFADESİ

Tablo içindeki bilgilerin bir koşula göre sıralanması için kullanılır. Where kullanılırken;

**Nümerik ifadelerde;**

**Örnek:** `select * from notlar where final >60` işlevini yazınız.(okul projesine göre)

Notlar tablosunda notu 60'den büyük olanların bilgilerini listeler.

|   | no      | op_kod | vize | final | büt |
|---|---------|--------|------|-------|-----|
| 1 | 7536511 | 274    | 53   | 80    | 40  |
| 2 | 7536511 | 218    | 98   | 76    | 54  |
| 3 | 7536521 | 274    | 21   | 78    | 63  |
| 4 | 7536521 | 222    | 57   | 90    | 65  |
| 5 | 6532510 | 372    | 60   | 66    | 88  |
| 6 | 5531519 | 433    | 63   | 70    | 90  |
| 7 | 5531519 | 421    | 40   | 66    | 96  |
| 8 | 5530535 | 421    | 50   | 80    | 77  |

**Örnek:** `SELECT * FROM personel WHERE brüt <= 8000000` işlevini yazınız.

Personel tablosunda brütü 8000000 eşit veya daha küçük olanların bilgilerini listeler.

### **Karakterlerde;**

**Örnek:** `SELECT * FROM ogrenci WHERE adi= 'murat'` işlevini yazınız.

Ogrenci tablosunda adı Murat olanların bilgisini listeler.

**Örnek:** `SELECT * FROM personel WHERE ad <> 'Ali'` işlevini yazınız.

### **Bitsel ifadelerde;**

**Örnek:** `SELECT * FROM ogrenci WHERE cinsiyet= 'Bayan'` işlevini yazınız.

Öğrenci tablosunda cinsiyeti bayan olanların bilgilerini listeler.

### **Tarih türü ifadelerde;**

**Örnek:** `SELECT * FROM ogrenci WHERE Dog_Tarihi>'1979/09/19'` işlevini yazınız.

Öğrenci tablosunda 19/09/1979 tarihinden daha sonra doğanların bilgilerini listeler.

**Örnek:** `SELECT * FROM personel WHERE dog_tar <= {12/31/59}` işlevini yazınız.

Personel tablosunda 31/12/1959 tarihi ve daha önce doğanların bilgilerini listeler.



**Örnek:** `SELECT * FROM businif WHERE yasi<19 and adi='mehmet'`  
Sorgusunun ne işlevini yazınız.

Bu sınıf tablosunda yaşı 19 küçük ve adı Mehmet olan öğrencilerin tüm bilgilerini listeler.

**Örnek:** `SELECT * FROM ogrenci WHERE YEAR (Tarih)=2000` işlevini yazınız.

Öğrenci tablosunda tarih 2000 yılına eşit olan kişilerin bilgilerini listeler.

**Örnek:** `SELECT * FROM ogrenci WHERE dog_tarihi BETWEEN '1979/12/31' AND '1979/01/01'` işlevini yazınız.

Öğrenci tablosunda doğum tarihi 31/12/1979 ve 01/01/1979 tarihleri arasında olan kişilerin bilgilerini listeler.

**Örnek:** `SELECT adi FROM ogrenci WHERE yasi BETWEEN 10 AND 15` komutunun işlevini yazınız.

Öğrenci tablosunda yaşı 10 ile 15 arasında bulunan kişilerin sadece adlarını listeler.

**Örnek:** `SELECT * FROM ogrenci WHERE yasi<12 AND adi= 'sibel'` komutunun işlevini yazınız.

Öğrenci tablosunda yaşı 12 den küçük adı Sibel olan kişilerin bilgilerini listeler.

**Örnek:** `SELECT * FROM ogrenci WHERE notu >50 AND cinsiyet= 'Erkek'` komutunun işlevini yazınız.

Öğrenci tablosunda notu 50'den fazla olan ve cinsiyeti erkek olan öğrencinin bilgilerini listeler.

**Örnek:** `SELECT * FROM ogrenci WHERE NOT cinsiyet= 'Erkek'` komutunun işlevini yazınız.

Öğrenci tablosunda cinsiyeti erkek olmayan kişilerin bilgilerini listeler.

**Örnek:** Doğum tarihi 1960'dan önce olan maaşı 6000000 – 10000000 arasındaki bayan personeli listeleyiniz.

`SELECT * FROM personel WHERE dog_tar < {01/01/60} AND brüt >= 6000000 AND brüt <= 10000000 AND cinsiyet = 'Bayan'`

**Örnek:** Bölümü Satış ya da Muhasebe olamayan 1960'dan sonra doğmuş bayan personeli listeleyiniz.

`SELECT * FROM personel WHERE NOT (böl_no =1 OR böl_no =2) AND dog_tar >={01/01/60} AND cinsiyet = 'Bayan'`

**Örnek:** Toplam ders saati 20'ye eşit olan hocaların adı ve soyadını bulduran sorguyu yazınız. (Okul projesi)

`select h_adi, h_soyadi from hocalar where top_ders_saati=20`

|   | h_adi   | h_soyadi |
|---|---------|----------|
| 1 | Asaf    | Varol    |
| 2 | Muammer | Gökbulut |
| 3 | Hakan   | Akpolat  |
| 4 | Hanifi  | Güldemir |

**Örnek:** Geçme notu 70'den yüksek olan öğrencilerin numaraları ve hangi dersten geçtiğini listeleyen sorguyu yazınız. (Okul projesi)

`select no,op_kod from notlar where ((vize*0.4)+(final*0.6))>70`

|   | no      | op_kod |
|---|---------|--------|
| 1 | 7536511 | 218    |
| 2 | 7536521 | 222    |

**Örnek:** 536 nolu bölümde okuyan öğrencilerin adını ve soyadını listeleyen ayrıca adını artana göre listeleyen komut satırını yazınız. (Okul projesi)

`select adi,soyadi from ogrenci where bolkod=536 order by adi asc`

|   | adi    | soyadi    |
|---|--------|-----------|
| 1 | Aytül  | Çetinkaya |
| 2 | Cengiz | Güneş     |
| 3 | Veli   | Uşak      |

**Örnek:** Dersler tablosunda 1. Dönem verilen dersleri sıralayan komut satırını bulunuz. (Okul projesi)

|    | ders_adi             |
|----|----------------------|
| 1  | Türk dili            |
| 2  | Temel bil.tek.kul.   |
| 3  | Matematik            |
| 4  | Öğrt.mesleğine giriş |
| 5  | Mantık devreler      |
| 6  | Bil.prg.             |
| 7  | Devre analizi        |
| 8  | Teknik iletişim      |
| 9  | Mikro. mimarisi      |
| 10 | Veri yapıları        |
| 11 | Ölçme                |
| 12 | İşletim sistemleri   |
| 13 | Okul deneyimi        |
| 14 | Bil.ağları2          |
| 15 | Web tasarımı         |
| 16 | Yapay zeka           |
| 17 | Proje                |

`select ders_adi from dersler where dönem=1`

**Örnek:** 1986 doğumlu öğrencilerin numaralarını listeleyen komut satırını yazınız. (Okul projesi)

- `select no from ogrenci where d_tarihi between '1986.01.01' and '1986.12.31'` veya
- `select no from ogrenci where YEAR(d_tarihi)=1986`

|   | no      |
|---|---------|
| 1 | 5531519 |
| 2 | 6532510 |

## 6.5.6.BETWEEN İFADESİ

Belirli bir aralık içindeki bilgileri listelemek için BETWEEN deyimi kullanılır.

**Örnek:** Notlar tablosunda final notu 40 ile 70 arasında olan kişilerin not bilgilerini veren sorguyu yazınız. (Okul projesi)

`select * from notlar where final between 40 and 70`

|   | no      | op_kod | vize | final | büt |
|---|---------|--------|------|-------|-----|
| 1 | 6532510 | 373    | 11   | 47    | 45  |
| 2 | 6532510 | 372    | 60   | 66    | 88  |
| 3 | 6532510 | 371    | 23   | 45    | 67  |
| 4 | 5531519 | 433    | 63   | 70    | 90  |
| 5 | 5531519 | 421    | 40   | 66    | 96  |
| 6 | 5530535 | 433    | 38   | 60    | 100 |

**Örnek:** `SELECT * FROM personel WHERE brüt BETWEEN 500000 AND 800000` sorgusu ne işe yarar.

Personel tablosunda brütü 500000 ve 800000 arasında olan kişilerin bilgilerini listeler.

**Örnek:** Hocalar içerisinde Ders saati 15 ve 17 arasında olan ve azalana göre sıralanarak bilgilerini listeleyen sorgu ve çıktısını yazınız. (Okul projesi)

`select * from hocalar where top_ders_saati between 15 and 17 order by top_ders_saati desc`

|   | h_ID | h_adi      | h_soyadi   | unvan | top_ders_saati |
|---|------|------------|------------|-------|----------------|
| 1 | 6    | Mehmet     | Gedikpınar | 3     | 17             |
| 2 | 7    | Servet     | Tuncer     | 3     | 17             |
| 3 | 8    | Engin      | Avcı       | 3     | 17             |
| 4 | 9    | Davut      | Hanbay     | 3     | 17             |
| 5 | 10   | Abdülkadir | Şengür     | 3     | 17             |
| 6 | 12   | Cafer      | Bal        | 5     | 16             |
| 7 | 13   | Erkan      | Tanyıldızı | 5     | 16             |
| 8 | 14   | Murat      | Karabatak  | 4     | 15             |
| 9 | 11   | Nurhayat   | Varol      | 4     | 15             |

## 6.5.7.LIKE İFADESİ

Karakter türü verilerde, veri içerisinde geçen belirli bir kelime veya verileri bulmak için kullanılır. % sembol A harfinin öncesi ve sonrasında herhangi bir bilgi alabileceğini gösterir. LIKE sözcüğünü, alt tire( \_ ) sembolü ile birlikte kullanmakta mümkündür.

**Örnek:** Adının içerisinde 'A' harfi geçenleri bulmak için kullanılan komutları yazınız.

```
SELECT * FROM ogrenci WHERE adi LIKE '%A%'
```

```
SELECT * FROM ogrenci WHERE adi LIKE '__A%'
```

```
SELECT * FROM ogrenci WHERE adi LIKE 'A_K_T'
```

**Örnek:** Öğrenci tablosunda adı a harfi ile başlayan öğrenci numaralarını listeleyen sorguyu yazınız. (Okul projesi)

```
select no from ogrenci where adi like 'a%'
```

|   | no      |
|---|---------|
| 1 | 6532518 |
| 2 | 7536511 |

**Örnek:** İsminde ikinci harfi A dördüncü harfi B olan öğrencilerin bilgilerini veren sorguyu yazınız.

**SELECT** \* **FROM** ogrenci **WHERE** adi LIKE '\_A\_B%'

**Örnek:** İsminde AL geçen öğrencilerin bilgilerini listeleyen sorguyu yazınız.

**SELECT** \* **FROM** ogrenci **WHERE** adi LIKE '%AL%'

**Örnek:** Adının başında Fırat geçen öğrencilerin bilgilerini veren sorguyu yazınız.

**SELECT** \* **FROM** ogrenci **WHERE** adi LIKE 'Fırat%'

**Örnek:** Adının sonunda Fırat geçen öğrencilerin bilgilerini bulan sorguyu yazınız.

**SELECT** \* **FROM** ogrenci **WHERE** adi LIKE '%Fırat'

**Örnek:** Adının ikinci harfi A olan ve uzunluğu belli olmayan öğrencinin bilgilerini veren sorguyu yazınız.

**SELECT** \* **FROM** businif **WHERE** adi LIKE '\_A%'

**Örnek:** **SELECT** \* **FROM** personel **WHERE** adres LIKE '%TAKSİM%' sorgusunun işlevini yazınız.

Adres LIKE '%TAKSİM%' ifadesi adres içinde her hangi bir yerde TAKSİM yazan yerde oturan personeli listeleyecektir.



## 6.5.8.IN İFADESİ

Belli bir kolonun kümesi verilerek işlerin daha kolay yapılmasını sağlar.

**Örnek:** İn komutunun kullanımını anlamak için aşağıdaki iki örneği inceleyin. Bu iki örneğin işlevleri aynıdır.

```
SELECT * FROM ogrenci WHERE no=7 or no=12
```

```
SELECT * FROM ogrenci WHERE no IN(7,12)
```

**Örnek:** Bolkod=531 olan öğrencilerin not bilgilerini veren sorguyu yazınız.

```
SELECT * FROM notlar WHERE numara IN ( SELECT numara  
FROM ogrenci WHERE bolkod=531)
```

**Örnek:** SELECT adi, soyadi, no, adres FROM ogrenci WHERE no IN (1,3,5,7) sorgusunun işlevini yazınız.

**Örnek:** 117 nolu öğrencinin 3 kredilik derslerden aldığı notları listeleyen sorguyu yazınız.

```
SELECT vize1, vize2, final FROM notlar WHERE og_no= 117 AND  
opkod IN( SELECT opkod FROM dersler WHERE kre=3)
```

**Örnek:** `SELECT` barkod, urunadi `FROM` urunler `WHERE` barkod NOT IN (`SELECT` barkod `FROM` urunsatis `WHERE` `DATEDIFF`(dd, tarih, `GETDATE`())<=30) sorgusunun işlevini yazınız.

1 ay içerisinde satılmayan ürünleri listeler.

**Örnek:** bölüm kodu 531 ve 532 olan öğrencilerin adını ve soyadını veren sorguyu yazınız. (Okul projesi)

`select` adi,soyadi `from` ogrenci `where` bolkod in(531,532)

|   | adi   | soyadi  |
|---|-------|---------|
| 1 | Betül | Cebe    |
| 2 | Leyla | Korkmaz |
| 3 | Ali   | Yılmaz  |

**Örnek:** Optik kodu 421 ve 433 olan derslerden notları açıklanan öğrencilerin numarasını bulan sorguyu yazınız. (Okul projesi)

`select distinct`(no) `from` notlar `where` op\_kod in(421,433)

|   | no      |
|---|---------|
| 1 | 5530535 |
| 2 | 5531519 |

## 6.5.9.ANY SÖZCÜĞÜ

Aşağıdaki örnek soru çerçevesinde bu sözcüğün SELECT komutu içindeki etkisi açıklanacaktır.

**Örnek:** Satış bölümünde çalışan personel her hangi birinden daha düşük maaş alan ve mühendislik bölümünde çalışan kişileri listeleyiniz.

```
SELECT * FROM Personel WHERE maas<ANY( SELECT maas  
FROM Personel WHERE bol_no=2 ) AND bol_no=1
```

Bu çözümün eşdeğeri olan ifade ise şöyledir.

```
SELECT * FROM Personel WHERE maas<(SELECT MAX  
(maas)FROM Personel WHERE bol_no=2) AND bol_no=1
```

- Burada satış bölümü kodu 2 ve mühendislik bölümü kodu ise 1 olarak kabul edilmiştir. İkinci çözüm ifadesinden de kolayca anlaşılabacağı gibi, iç içe SELECT ifadesinde, içteki SELECT sorgulaması sonucu, ikinci bölümde (Satış) çalışan personel içinde en yüksek maaş alan kişi maaşı bulunmakta, dıştaki SELECT ise, mühendislik bölümünde, bu maaştan düşük olan maaşa sahip kişileri listelemektedir. Buradaki düşünce tarzı şöyledir: Mühendislik bölümünde çalışan ve satış bölümündeki en yüksek maaştan düşük maaş alan kişi “SATIŞ BÖLÜMÜNDEKİ HERHANGİ BİR MAAŞ’TAN DÜŞÜK OLMA” koşulunu sağlayacaktır. Eğer mühendislik bölümündeki kişi maaşı, satış bölümündeki en yüksek maaştan daha yüksek olsa (veya ona eşit olsa) doğal olarak bu koşul sağlanmayacaktır.

Personel tablosu aşağıdaki verileri içeriyorsa,

| sicil | ad    | soyad  | bol_no | maas     | ..... |
|-------|-------|--------|--------|----------|-------|
| 117   | Ali   | Can    | 1      | 7000000  | ..... |
| 247   | Hasan | Okan   | 1      | 6000000  | ..... |
| 348   | Ayşe  | Pekcan | 2      | 50000000 | ..... |
| 548   | Ak    | Pekol  | 1      | 4000000  | ..... |
| 1148  | Mert  | Caner  | 2      | 12000000 | ..... |
| 1215  | Beril | Şen    | 2      | 5000000  | ..... |

Yukarıdaki SELECT komutları sonucu (ANY ile ya da eşdeğeri ile) tablosu elde edilecektir.

| sicil | ad    | soyad | bol_no | maas    | ..... |
|-------|-------|-------|--------|---------|-------|
| 117   | Ali   | Can   | 1      | 7000000 | ..... |
| 247   | Hasan | Okan  | 1      | 6000000 | ..... |
| 548   | Ak    | Pekol | 1      | 4000000 | ..... |

ANY (her hangi bir) sözcüğü yerine, tamamen eşdeğeri olan SOME sözcüğü de kullanılabilir.

**Örnek:** SELECT \* FROM notlar WHERE cinsiyet='Erkek' AND ort>ANY(SELECT ort FROM notlar WHERE cinsiyet='Bayan')  
sorgusunun işlevini yazınız.

**Örnek:** Optik kodu 421 olan dersin vize1 notundan herhangi birinden vize1 notu yüksek olan öğrencilerin bilgilerini veren sorguyu ve çıktısını yazınız. (Okul projesi)

`select * from notlar where vize>any(select vize from notlar where op_kod=421)`

|   | no      | op_kod | vize | final | büt |
|---|---------|--------|------|-------|-----|
| 1 | 7536511 | 274    | 53   | 80    | 40  |
| 2 | 7536511 | 218    | 98   | 76    | 54  |
| 3 | 7536521 | 222    | 57   | 90    | 65  |
| 4 | 6532510 | 372    | 60   | 66    | 88  |
| 5 | 5531519 | 433    | 63   | 70    | 90  |
| 6 | 5530535 | 421    | 50   | 80    | 77  |

**Örnek:** Final notu 65 ten küçük olan final notlarının herhangi birinden final notu düşük olan öğrencilerin numarasını veren sorguyu ve çıktısı yazınız. (Okul projesi)

`select no from notlar where final<any(select final from notlar where final<65 )`

|   | no      |
|---|---------|
| 1 | 7536545 |
| 2 | 7536545 |
| 3 | 6532510 |
| 4 | 6532510 |

## 6.5.10.ALL SÖZCÜĞÜ

“Hepsi, tamamı” anlamdaki bu sözcük, SELECT komutu içerisinde belirli bir koşulu sağlayan bir gruptan tamamı sağlanan koşullarla ilişkili olarak kullanılır. Aşağıdaki soru konuda açıklayıcı olacaktır:

**Örnek:** Satış bölümünde çalışan ve mühendislik bölümündeki personel hepsinden daha fazla maaş alan personeli listeleyiniz. Bu örnekte de satış bölümü kodu 2 ve mühendislik bölümü kodu 1 olarak alırsa aşağıdaki SELECT grupları çözüm teşkil edecektir.

### 1. Alternatif

```
SELECT *  
FROM Personel  
WHERE maas>ALL(SELECT maas  
FROM Personel  
WHERE bol_no=1)  
AND bol_no=2;
```

### 2. Alternatif

```
SELECT *  
FROM Personel  
WHERE maas>(SELECT MAX(maas)  
FROM Personel  
WHERE bol_no=1)  
AND bol_no=2;
```

- Personel tablosu aşağıdaki verileri içeriyorsa;

| sicil | ad    | soyad  | bol_no | maas     |
|-------|-------|--------|--------|----------|
| 117   | Ali   | Can    | 1      | 7000000  |
| 247   | Hasan | Okan   | 1      | 6000000  |
| 348   | Ayşe  | Pekcan | 2      | 50000000 |
| 548   | Ak    | Pekol  | 1      | 4000000  |
| 1148  | Mert  | Caner  | 2      | 12000000 |
| 1215  | Beril | Şen    | 2      | 5000000  |

Yukarıdaki 1. Alternatif ve 2. Alternatif olarak belirtilen, her iki SELECT komutları grubu da aşağıdaki sonucu verecektir.

| sicil | ad    | soyad  | bol_no | maas     |
|-------|-------|--------|--------|----------|
| ----- | ----- | -----  | -----  | -----    |
| 348   | Ayşe  | Pekcan | 2      | 5000000  |
| 1148  | Mert  | Caner  | 2      | 12000000 |



**Örnek:** Optik kodu 421 olan final notlarının hepsinden final notu büyük olan öğrencinin bilgilerini listeleyen sorguyu ve çıktısını yazınız. (Okul projesi)

```
select * from notlar where  
final>all(select final from notlar  
where op_kod=421)
```

|   | no      | op_kod | vize | final | büt |
|---|---------|--------|------|-------|-----|
| 1 | 7536521 | 222    | 57   | 90    | 65  |

## 6.6.İç İçe Select Komutları (Nested Selects)

Bazı sorgulamalar, özelliği itibarı ile iç içe SELECT komutlar kullanılması gerektirebilir. İçteki SELECT komutunun bulduğu sonuç, dıştaki SELECT komutunun işlevi yere getirmesi iç kullanılır.

**Örnek:** Parça numarası 24 olan parçayı kullanan projelerde çalışan personeli listeleyiniz. Örnek olarak aşağıdaki verileri göz önüne alalım.

**Tablo 6.9:** Parça Tablosu

| par_no | par_ad | pr_no | fiyat | ağırlık |
|--------|--------|-------|-------|---------|
| 24     | Vida   | 2     | 2000  | 500     |
| 24     | Vida   | 4     | 2000  | 500     |
| 37     | Civata | 2     | 6000  | 800     |
| 87     | Conta  | 2     | 7000  | 5000    |
| 112    | Pim    | 5     | 6000  | 70      |

- **Tablo 6.10:** Proje Tablosu

| proje_ad | proj_no | yer      | bl_no |
|----------|---------|----------|-------|
| 1        | 1       | İstanbul | 4     |
| 2        | 2       | İstanbul | 4     |
| 3        | 3       | Ankara   | 5     |
| 4        | 4       | Ankara   | 5     |
| 5        | 5       | İzmir    | 4     |

- **Tablo 6.11:** Personel Tablosu

| sicil | sosy_g_no | ad    | soyad  | dog_tar  | bol_no | adres              |
|-------|-----------|-------|--------|----------|--------|--------------------|
| 117   | 274251    | Ali   | Can    | 05/01/60 | 4      | Akar sok. 2 Fatih  |
| 247   | 527241    | Hasan | Okan   | 04/07/62 | 4      | Merk cad. 3 Pendik |
| 348   | 5276672   | Ayşe  | Pekcan | 04/08/65 | 5      | .....              |
| 548   | 443211    | Ak    | Pekol  | 07/02/70 | 4      | .....              |
| 1148  | 527625    | Mert  | Caner  | 04/08/70 | 5      | .....              |

**Tablo 6.12: Çalışma Tablosu**

| per_s_g_no | proje_no | saat |
|------------|----------|------|
| 274251     | 1        | 250  |
| 527241     | 2        | 350  |
| 527672     | 3        | 400  |
| 443211     | 5        | 300  |
| 527625     | 4        | 250  |

Yukarıdaki tablolardan yararlanarak aşağıdaki SELECT komutları ile arzu edilen işlem gerçekleştirilebilir:

```
SELECT * FROM Personel WHERE sosy_g_no in(SELECT  
Per_s_g_no FROM Calisma WHERE proje_no in(select  
proj_no from Proje where proj_no in(select pr_no from  
Parca where par_no=24))) ;
```

Buradaki içteki SELECT komutu parça, proje ve çalışma tabloları proje numaraları üzerinde (proje numaraları bu tablolarda sıra ile pr\_no, proj\_no ve proje\_no adı ile yer almaktadır) birleştirilerek elde edilen genişletilmiş tablodan sadece parça no'su 24 olan satırdaki personel sosyal güvenlik numaraları (pers\_s\_g\_no) çıkarmakta ve sonuçta yukarıdaki örnek data için;

**per\_s\_g\_no**

-----

527241

527625

değerleri elde etmektedir.

Dış SELECT komutu ise, personel tablosundan bu sosyal güvenlik numaralara sahip personel aşağıdaki tabloda görüldüğü gibi listelenecektir:

| sicil sosy_g_no | ad    | soyad | dog_tar  | bol_no |
|-----------------|-------|-------|----------|--------|
| -----           | ----- | ----- | -----    | -----  |
| 247             | Hasan | Okan  | 04/07/62 | 4      |
| 1148            | Mert  | Caner | 04/08/70 | 5      |

Benzer şekilde aşağıdaki sorunun çözümü de iç içe SELECT komutları ile gerçekleştirilebilir.

**Örnek:** Fatih'te oturan personel çalıştığı projeler adları ve yerleri listeleyin.

```
SELECT proje_ad, yer FROM Proje WHERE bl_no  
in(select bol_no from Personel where adres LIKE  
'%Fatih%');
```

| proj_ad | yer      |
|---------|----------|
| 1       | İstanbul |
| 3       | Ankara   |

çıktısı elde edilecektir.

**Örnek:** Adı Betül olan öğrencinin hangi derslerden sınava girdiğini bulan sorgu ve çıktısını yazınız. . (Okul projesi)

```
select op_kod from notlar where no in(select no  
from ogrenci where adi='betül')
```

|   | op_kod |
|---|--------|
| 1 | 433    |
| 2 | 421    |

**Örnek:** Finalden en yüksek notu alan öğrencinin adının bulalım (okul projesinden)

```
select adi from ogrenci where no in (select no from notlar where final=(select MAX(final) from notlar))
```

|   | adi  |
|---|------|
| 1 | Veli |

**Örnek:** Hangi derslerden büte kalınmış onu bulalım (okul projesinden)

```
select ders_adi from dersler where op_kod in (select op_kod from notlar where ((vize*0.4)+(final*0.6))<50)
```

|   | ders_adi        |
|---|-----------------|
| 1 | Bil.prg.        |
| 2 | Devre analizi   |
| 3 | Teknik iletişim |
| 4 | Veri yapıları   |

# 6.7.SQL SERVER'DA ARİTMETİKSEL İŞLEMLER VE FONKSİYONLAR

## 6.7.1. ARİTMETİKSEL İŞLEMLER VE FONKSİYONLAR

SELECT komutu ile veri tabanında mevcut tablolardan listeleme yaparken, tabloda ayrı bir sütun olarak yer almamış ve bir hesaplama sonucunda üretilebilecek bilgileri de liste içine katmak mümkündür.

**Örnek:** Aşağıdaki SELECT komutu ile bir personel tablosunda personelin şu anda geçerli olan maaşı ile bu maaşın %10 zamlı şekli listelenmektedir.

```
SELECT ad, soyad, maas, maas*1.1, maas*1.1-maas  
FROM personel;
```

| ad  | soyad | maas  |       |       |
|-----|-------|-------|-------|-------|
| --- | ----- | ----- | ----- | ----- |
| A   | B     | 10    | 11    | 1     |
| C   | D     | 90    | 99    | 9     |



**Tablo 6.13:** Aritmetiksel Semboller

| SQL'de Aritmetiksel Semboller |         |
|-------------------------------|---------|
| Operatör                      | İşlevi  |
| ** veya ^                     | Üs Alma |
| *                             | Çarpma  |
| /                             | Bölme   |
| +                             | Toplama |
| -                             | Çıkarma |

Öncelik sırası, matematikte ve diğer bilgisayar dillerinde olduğu gibidir. Üs alma hepsinden önceliklidir. Sonra \* ve / gelir. \* ve / aynı özelliğe sahiptir. + ve – en son önceliklidir. + ve – birbiri ile aynı önceliğe sahiptir. Parantezler kullanılarak, öncelikler değiştirilebilir.

|               |       |       |      |       |
|---------------|-------|-------|------|-------|
| <b>Örnek:</b> | Og_no | d_adi | vize | final |
|               | 1     | mat   | 60   | 70    |
|               | 2     | fizik | 70   | 70    |
|               | 3     | kimya | 55   | 65    |

Bu tabloya göre aşağıdaki sorgunun çıktısı ne olur, yazınız.

```
SELECT og_no, d_adi, vize*0.4 + final*0.6 FROM  
notlar WHERE og_no=1;
```

|       |       |       |
|-------|-------|-------|
| Og_no | d_adi |       |
| ----- | ----- | ----- |
| 1     | mat   | 66    |

**Örnek:** `SELECT adi, yasi*2 FROM businif WHERE yasi<12`  
 Komutunun işlevini yazınız.

Yaşı 12'den küçük olanların adını ve yaşının 2 ile çarpımını verir.

**Örnek:** Öğrencilerin notlarını bağıl sisteme göre hesaplayan sorguyu yazınız. (Okul Projesi)

`select no, op_kod, (vize*0.4)+(final*0.6) from notlar`

|    | no      | op_kod | (No column name) |
|----|---------|--------|------------------|
| 1  | 7536511 | 274    | 69.2             |
| 2  | 7536511 | 218    | 84.8             |
| 3  | 7536521 | 274    | 55.2             |
| 4  | 7536521 | 222    | 76.8             |
| 5  | 7536545 | 274    | 27.4             |
| 6  | 7536545 | 271    | 11.4             |
| 7  | 6532510 | 373    | 32.6             |
| 8  | 6532510 | 372    | 63.6             |
| 9  | 6532510 | 371    | 36.2             |
| 10 | 5531519 | 433    | 67.2             |
| 11 | 5531519 | 421    | 55.6             |
| 12 | 5530535 | 433    | 51.2             |
| 13 | 5530535 | 421    | 68.0             |

## 6.7.2. KÜMELEME FONKSİYONLARI

SQL tablo içerisindeki çeşitli matematiksel ifadelerin sonucunu otomatik olarak üretmeyi sağlayan fonksiyonlara sahiptir.

### 6.7.2.1. SUM FONKSİYONU:

Tablo içerisinde belirli bir sütuna göre toplama işlemi gerçekleştirir.

Kullanılışı: `Select SUM (sütun_adi ) FROM Tablo_adi;`

**Örnek:** `SELECT SUM(maas) FROM personel;`

Sorgusunun işlevini yazınız.

Tüm personelin maaşlarının toplamını yapar.

**Örnek:** `SELECT SUM(yas) FROM personel WHERE cinsiyet ='Erkek';`

Sorgusunun işlevini yazınız.

Cinsiyeti erkek olanların yaşları toplamını verir.

**Örnek:** `SELECT SUM(bmaas) - SUM(nmaas) FROM personel;`

Sorgusunun işlevini yazınız.

bmaasların toplamından nmaasların toplamını çıkarıp listeler.

**Örnek:** `SELECT SUM(maas*1.1-maas) FROM personel;`

Sorgusunun işlevini yazınız.

Maaşların %10'larını toplar ve listeler.

**Örnek:** Tüm hocaların girdiği toplam ders saatini bulan sorguyu yazınız.  
(Okul Projesi)

`select SUM(top_ders_saati) from hocalar`

|   | (No column name) |
|---|------------------|
| 1 | 269              |

## 6.7.2.2.AVG FONKSİYONU:

Belirli bir alan üzerinde aritmetik ortalama(average) hesaplamak için kullanılır.

**Kullanılışı:** `SELECT AVG (sütun_adi ) FROM Tablo_adi;`

**Örnek:** `SELECT AVG(notu) FROM ogrenci WHERE cinsiyet='Erkek';` Sorgusunun işlevini yazınız.

Erkeklerin notlarının ortalamasını bulur.

**Örnek:** `SELECT AVG(yasi) FROM businif;` Sorgusunun işlevini yazınız.

businifin yaş ortalamasını bulur.

**Örnek:** `SELECT AVG(maas) FROM personel;` Sorgusunun işlevini yazınız.

Personelin maaş ortalamasını bulur.

**Örnek:** Tüm sınavların ortalamasını veren sorguyu yazınız. (Okul Projesi)

`select AVG(vize), AVG(final), AVG(but) from notlar`

|   | (No column name) | (No column name) | (No column name) |
|---|------------------|------------------|------------------|
| 1 | 43               | 60               | 67               |

## 6.7.2.3. MAX FONKSİYONU:

İçinde, belirlenen sütun içindeki en büyük değeri bulmak için kullanılır.

Kullanılışı: `SELECT MAX (sütun_adı ) FROM Tablo_adı;`

**Örnek:** `SELECT MAX(notu) FROM ogrenci;` Sorgusunun işlevini yazınız.

Maksimum notu verir.

**Örnek:** `SELECT MIN (maas) FROM personel;` Sorgusunun işlevini yazınız.

En düşük maaşı verir.

**Örnek:** `SELECT MIN (yasi) FROM businif;` Sorgusunun işlevini yazınız.

Sınıftaki en küçük yaşı verir.

**Örnek:** Finalden alınan en düşük notu bulan sorguyu yazınız.  
(Okul Projesi)

`select MIN (final) from notlar`

| (No column name) |    |
|------------------|----|
| 1                | 11 |

## 6.7.2.4. COUNT (SAY) FONKSİYONU:

Tablo içerisinde herhangi bir alanda sayma işlemi gerçekleştirir.

**Kullanılışı:** `SELECT COUNT (sütun_adı ) FROM Tablo_adı;`

**Örnek:** `SELECT COUNT (*) FROM ogrenci;` Sorgusunun çıktısı ne olur. (Okul Projesi)

|   | (No column name) |
|---|------------------|
| 1 | 8                |

nı verir.

**Örnek:** `select COUNT(cinsiyet) from ogrenci where cinsiyet='bayan'` Çıktısını bulunuz. (Okul Projesi)

Öğrenci tablosunda bulunan toplam bayan sayısını verir.

|   | (No column name) |
|---|------------------|
| 1 | 5                |

- COUNT komutunda, “\*” argümanının kullanılması, bütün sütunların(alanların) işleme sokulmasını, alan adının belirtilmesi ise (COUNT(böl\_no) gibi), sadece, belirtilen sütunun işleme sokulmasını sağlar.

DISTINCT fonksiyonu COUNT fonksiyonu ile birlikte kullanılabilir.

**Kullanılışı:** `SELECT COUNT(DISTINCT sütun_adı) FROM tablo_adı;`

**Örnek:** Kaç farklı isme sahip öğrenci sayısını veren sorguyu bulunuz.

`SELECT COUNT(DISTINCT adi) FROM ogrenci;`

| (No column name) |   |
|------------------|---|
| 1                | 8 |

**Örnek:** Notlar tablosuna kaç farklı dersin not girildiğini bulan sorguyu ve çıktıyı yazınız. (Okul Projesi)

`select COUNT(distinct(op_kod)) from notlar`

| (No column name) |   |
|------------------|---|
| 1                | 9 |

**Örnek:** Final notu 40'ın altında olan öğrenci sayısı nedir? (okul projesinden)

`SELECT COUNT(final) FROM notlar WHERE final<40`

| (No column name) |   |
|------------------|---|
| 1                | 2 |



## 6.7.3. GRUPLANDIRARAK İŞLEM YAPMA

Kümeleme fonksiyonları, kullanılırken tablodaki bilgileri, bazı özelliklere göre gruplandırarak bu gruplandırılmış veri üzerinde sorgulama yapmak mümkündür. Bu işlem için, GROUP BY ifadesi kullanılır.

### 6.7.3.1.GROUP BY:

Tablodaki verileri belirli özelliklere göre gruplandırmak için kullanılır.

**Örnek:** `SELECT cinsiyet, AVG(ort) FROM ogrenci  
GROUP BY cinsiyet;` sorgusunun işlevini yazınız.

Erkeklerin ayrı, bayanların ayrı ortalamasını alarak ekrana yazar.

### Örnek:

| Sc no | Adi   | Soyadi | Maasi |
|-------|-------|--------|-------|
| 1     | Murat | Kara   | 20    |
| 2     | Murat | Kaya   | 15    |
| 3     | Ahmet | Tanyol | 20    |
| 4     | Ahmet | Ekinci | 25    |

Yandaki tabloya göre aşağıdaki sorgunun çıktısı ne olur.

```
SELECT adi, SUM(maasi) FROM personel GROUP BY adi;
```

Adi

----

Murat 35

Ahmet 45

**Örnek:** `SELECT adi, soyadi, dersi, MAX(ort) FROM ogrenci GROUP BY adi, soyadi, dersi;` sorgusunun işlevini yazınız.

Derslere göre en yüksek not kime ait ise, o kişiye ait verileri listeler.

| Bol no | Notu | Cinsiyeti |
|--------|------|-----------|
| 531    | 60   | 1         |
| 531    | 50   | 1         |
| 532    | 55   | 0         |
| 532    | 70   | 1         |
| 533    | 90   | 0         |
| 531    | 45   | 0         |
| 531    | 50   | 1         |

### Örnek:

Yandaki notlar tablosuna göre erkekler ve bayanların not ortalamasını veren sorgu ve çıktısı nasıl olur, yazınız.

```
SELECT cinsiyet, AVG(notu) FROM
notlar GROUP BY cinsiyet;
```

Cinsiyeti

-----

1

0

-----

57,5

63,33

Her bölümdeki erkek ve bayanların not ortalamalarını bulmak istersek sorgumuz aşağıdaki gibi olur.

```
SELECT bol_no, cinsiyet, AVG(notu) FROM notlar GROUP BY cinsiyet, bol_no;
```

Her bölümde notu en yüksek olan kişinin notunu veren sorgu ve çıktısı aşağıdaki gibi olur.

```
SELECT bol_no, MAX(notu) FROM notlar GROUP BY bol_no;
```

| bol_no |    |
|--------|----|
| 531    | 60 |
| 532    | 70 |
| 533    | 90 |

**Örnek:** Her bölümdeki bayanların notları ortalamasını bulan sorguyu ve çıktısı yazınız.(Okul Projesi)

```
select  
op_kod, AVG(vize), AVG(final),  
AVG(but) from notlar where no  
in(select no from ogrenci where  
cinsiyet='bayan') group by  
op_kod
```

|   | op_kod | (No column name) | (No column name) | (No column name) |
|---|--------|------------------|------------------|------------------|
| 1 | 218    | 98               | 76               | 54               |
| 2 | 271    | 12               | 11               | 30               |
| 3 | 274    | 43               | 51               | 50               |
| 4 | 371    | 23               | 45               | 67               |
| 5 | 372    | 60               | 66               | 88               |
| 6 | 373    | 11               | 47               | 45               |
| 7 | 421    | 45               | 73               | 86               |
| 8 | 433    | 50               | 65               | 95               |

## 6.7.3.1.1 HAVING

- Gruplandırarak, kümele fonksiyonlarını uygularken, koşul da verilebilir. Bu durumda, grup üzerindeki hesaplamalarla ilişkili koşul belirtirken, HAVING sözcüğünü kullanmak gerekir.
- **Örnek:** **SELECT** adi **FROM** ogrenci **GROUP BY** adi **HAVING** adi **LIKE** 'Ali'; sorgusunun işlevini yazınız.
  - Öğrencilerin adlarında Ali geçenleri gruplar.
- **Örnek:** Finali 50'den yüksek olup ortalama finalleri 60'a eşit ya da büyük olan öğrenci numaralarını ve final ortalamalarını listeleyin. (Okul Projesi)

```
select no,AVG(final) from notlar  
where final>50 group by no having AVG(final)>=60
```

|   | no      | (No column name) |
|---|---------|------------------|
| 1 | 5530535 | 70               |
| 2 | 5531519 | 68               |
| 3 | 6532510 | 66               |
| 4 | 7536511 | 78               |
| 5 | 7536521 | 84               |

- **Örnek:** Aşağıdaki çıktıyı veren sorguyu yazınız.

- 

| MAASAYI | COUNT (MAAS) | SUM (MAAS) | AVG (MAAS)     |
|---------|--------------|------------|----------------|
| -----   | -----        | -----      | -----          |
| 1       | 29           | 30935      | 1066.724137931 |
| 2       | 18           | 18585      | 1032.222       |
| 3       | 11           | 11950      | 1086.363653636 |

- `SELECT maasayi, COUNT (maas) ,SUM (maas) ,AVG (maas)`  
`FROM maaslar GROUP BY maasayi having maas<1200;`

- **Örnek:** Aşağıdaki çıktıyı veren sorguyu yazınız.

| • | MAASAYI | MAX (MAAS) | AVG (MAAS) |
|---|---------|------------|------------|
| • | -----   | -----      | -----      |
| • | 1       | 2400       | 1066.4589  |
| • | 2       | 1450       | 1032.25    |
| • | 3       | 2000       | 1098.50    |

- **SELECT** Maasay1, **MAX** (Maas) , **AVG** (Maas) , **FROM** Maaslar  
**GROUP BY** Maasay1 **HAVING** MAX (Maas)>1200;

- **Örnek:** Toplam Satış\_adedi 1000'in üzerinde olan ürünleri ve Satış toplamalarını listeleyin.

## SATIS

-----

Tarih

Urun\_kodu

Urun\_adi

Satis\_adedi

Urun\_Fiyatı

```
SELECT urun_adi, urun_kodu, SUM(satis_adedi) FROM SATIS  
GROUP BY urun_adi,urun_kodu HAVING SUM(satis_adedi)>1000;
```

- **Örnek:** Öğrencilerden toplam kredisi 50'den yukarı olanları listeleyin.
- **SELECT** og\_no, **SUM**(kredi) **FROM** notlar **GROUP BY** og\_no  
**HAVING SUM**(kredi)>50;



## 6.7.3.2 NULL DEĞER SORGULAMA

Null, içerisinde değer bulundurmeyen sütunlardır. Kayıt işlemi esnasında herhangi bir değer girilmezse o sütunun değeri NULL olarak atanır. Sütun içerisine bir boşluk dahi girilse o sütun artık NULL değildir. NULL değer içeren kayıtları sorgulamada karşılaştırma operatörleri kullanılmaz. Değerin NULL olup olmadığını bulmak için IS NULL ifadesi kullanılır. NULL değer içermeyen kayıtlar bulunmak isteniyorsa IS NOT NULL ifadesi kullanılır.

| Ogr_no | Ders_kodu | Sinav1 | Sinav2 |
|--------|-----------|--------|--------|
| 234    | BİL123    | 50     | 70     |
| 235    | BİL124    | 80     | NULL   |
| 236    | BİL125    | NULL   | 90     |

**Örnek:** Verilen notlar tablosunda 1.sınava veya 2.sınava girmeyen öğrencilerin numarasını ve sınava girmedği dersin kodunu bulmak için gerekli SQL ifadesini yazınız.

```
SELECT ogr_no, ders_kodu FROM NOTLAR WHERE sinav1 IS NULL OR Sinav2 IS NULL
```

| Ogr_no | Ders_kodu |
|--------|-----------|
| 235    | BİL124    |
| 236    | BİL125    |

**Örnek:** Bütünleme sınavına girmeyen öğrencilerin isimlerini bulan sql sorgusunu yazınız. (okul projesinden)

```
select adi from ogrenci where no in(select no from notlar where but is null)
```

| adi |
|-----|
|-----|

Bütünlemeye girmeyen öğrenci bulunmadığından sorgu sonucu boştur.

**Örnek:** Bütünleme sınavına giren öğrencilerin adlarını, hangi dersten bütünleme sınavına girdiklerini ve ders adlarına göre gruplandıran sorguyu yazınız. (okul projesinde)

```
select adi,ders_adi from ogrenci,dersler where no  
in(select no from notlar where but is not null and  
op_kod in(select op_kod from dersler)) group by  
ders_adi,adi
```

|    | adi    | ders_adi    |
|----|--------|-------------|
| 1  | Aytül  | Bil. ağlan  |
| 2  | Betül  | Bil. ağlan  |
| 3  | Cengiz | Bil. ağlan  |
| 4  | Feyza  | Bil. ağlan  |
| 5  | Leyla  | Bil. ağlan  |
| 6  | Veli   | Bil. ağlan  |
| 7  | Aytül  | Bil. ağlan2 |
| 8  | Betül  | Bil. ağlan2 |
| 9  | Cengiz | Bil. ağlan2 |
| 10 | Feyza  | Bil. ağlan2 |
| 11 | Leyla  | Bil. ağlan2 |

## 6.8.TABLOLARDA DEĞİŞİKLİK YAPMAK

### 6.8.1. TABLOYA VERİ EKLEMEK

**Kullanılışı:** **INSERT INTO** Tablo Adı(Sütun adı1,sütun adı2,.....sütun adın) **VALUES**  
(değer1,değer2,.....değer n);

**Örnek:** **INSERT INTO** ogrenci (adi soyadi, no)  
**VALUES** ('Firat','üniv',100);

## 6.8.2.TABLODAN VERİ SİLMEK

Bir tablonun verilerini silmek için DELETE komutu kullanılır.

**Kullanılışı:** `DELETE FROM` tablo\_adi;

**Örnek:** `DELETE FROM` ogrenci; sorgusunun işlevini yazınız.

Tablodaki tüm kayıtları siler.

**Örnek:** `DELETE FROM` ogrenci `WHERE` adi = 'Burcu';  
sorgusunun işlevini yazınız.

Adı Burcu olan tüm kayıtları siler.

**Örnek:** `DELETE FROM` ogrenci `WHERE` og\_no=172;  
sorgusunun işlevini yazınız.

Numarası 172 olan kaydı siler.

**Örnek:** DELETE FROM ogrenci WHERE notu<50; sorgusunun işlevini yazınız.

Notu 50'dan küçük olan kayıtları siler.

**Örnek:** DELETE FROM ogrenci WHERE adi like '\_a%'; sorgusunun işlevini yazınız.

İsminin ikinci harfi a olan kayıtları siler.

**Örnek:** Bilgisayar bölümü öğrencilerinin tüm not bilgilerini notlar tablosundan silen SQL komutunu yazınız.

```
DELETE FROM notlar WHERE no IN (SELECT no FROM  
ogrenci where bolum= '531')
```

## 6.8.3. VERİLERDE DEĞİŞİKLİK YAPMA-GÜNCELLEME(UPDATE) İŞLEMİ

Tablo üzerinde güncelleme yapmak için UPDATE komut kullanılır. DELETE komutunda olduğu gibi, UPDATE komutunu da, koşullu ya da koşulsuz olarak kullanmak mümkündür. Koşul belirtilmediği zaman değişiklik tüm tablo için geçerlidir.

**Kullanılışı:** UPDATE tabloadi SET kolon=deger1 WHERE koşul;

**Örnek:** UPDATE ogrenci SET donem=2 WHERE donem=1;  
sorgusunun işlevini yazınız.

1.dönemdeki tüm öğrencileri 2.döneme atadı.

**Örnek:** UPDATE ogrenci SET sinif=3 WHERE ort>49 AND sinif=2; sorgusunun işlevini yazınız.

Sınıf ortalaması 49 'dan büyük olan ve 2.sınıf öğrencilerini 3.sınıfa atadı.

**Örnek:** UPDATE ogrenci SET ort=50 WHERE ort<50 AND ort>45; sorgusunun işlevini yazınız.

Notu 45 ile 50 arasında olan öğrencilerin notlarını 50 yapar.

**Örnek:** Optik kodu 222 olan dersin final notlarını %10 artıran güncelleştirmeyi yapınız. (Okul Projesi)

update notlar set final=final\*1.1 where op\_kod=222



- select final from notlar where op\_kod=222

|   | final |
|---|-------|
| 1 | 99    |

**Örnek:** UPDATE ogrenci SET bolumu=531; sorgusunun işlevini yazınız.

Bolumu kolonunu 531 olarak değiştirir.



**Örnek:** UPDATE ogrenci SET bolumu=531; sorgusunun işlevini yazınız.

Bolumu kolonunu 531 olarak değiştirir.

**Örnek:** Adı Feyza olan öğrencinin aldığı rehberlik dersinin vizesine 10 puan ekleyen güncelleştirmeyi yapınız?(okul projesinden)

```
update notlar set vize=vize+10 where no in(select no
from ogrenci where adi='feyza') and op_kod in(select
op_kod from dersler ders_adi='rehberlik')
```

```
go
```

```
select adi,vize,ders_adi from
ogrenci,notlar,dersler where ogrenci.no=notlar.no and
notlar.op_kod=dersler.op_kod and adi='feyza' and
ders_adi='rehberlik'
```

|   | adi   | vize | ders_adi  |
|---|-------|------|-----------|
| 1 | Feyza | 60   | Rehberlik |

## 6.8.4.TABLO YAPISINDA DEĞİŞİKLİK YAPMA

ALTER TABLE komutu ile bir tablonun yapısında değişiklik yapmak mümkündür.

### 6.8.4.1. Mevcut Bir Tabloya Kolon Ekleme

Tabloya Alan(kolon) eklemek için ALTER TABLE komutuna ADD sözcüğü eklenmelidir.

**Örnek:** Öğrenci tablosunun içerisine öğrencinin yaş bilgilerini de eklemek istiyorsak;

```
ALTER TABLE ogrenci ADD yas INT NOT NULL;  
UPDATE ogrenci SET yas=10 Where Adi='MURAT'  
INTO ogrenci(yas) VALUES (10);
```

**Örnek:** `ALTER TABLE ogrenci ADD baba_adi CHAR(15);`  
Sorgusunun işlevini yazınız.

Öğrenci tablosuna baba\_adi diye char tipinde bir kolon ekler.

**Örnek:** Aşağıdaki tabloya kolon ekleyip değer giren komutları yazınız.

| Bol no | Notu | Cinsiyeti |
|--------|------|-----------|
| 531    | 60   | 1         |
| 531    | 50   | 1         |
| 532    | 55   | 0         |
| 532    | 70   | 1         |
| 533    | 90   | 0         |
| 531    | 45   | 0         |
| 531    | 50   | 1         |

AL numara CHAR(8);

| Bol-no | notu  | cins  | numara |
|--------|-------|-------|--------|
| -----  | ----- | ----- | -----  |
| 531    | 60    | 1     | NULL   |
| 531    | 50    | 1     | NULL   |
| 532    | 55    | 0     | NULL   |
| 532    | 70    | 1     | NULL   |
| 533    | 90    | 0     | NULL   |
| 531    | 45    | 0     | NULL   |
| 531    | 50    | 1     | NULL   |

**Değer Girmek İçin;**

UPDATE notlar SET numara=1 WHERE notu=90 AND bol-no=533;

## 6.8.4.2. Mevcut Alan Üzerinde Değişiklik Yapmak

Mevcut bir kolon üzerinde değişiklik yapma, değişken uzunluklu bir veri tipine sahip olan kolonun genişliğini arttırma ile sınırlıdır. Bu anlamda, kolon genişliğini azaltma ya da veri tipini değiştirme mümkündür. Bu işlem için ALTER TABLE komutuna MODIFY deyiminin eklenmesi ile gerçekleştirilir.

**Örnek:** ALTER TABLE ogrenci MODIFY adi CHAR(15); Sorgusunun işlevini yazınız.

Öğrenci tablosundaki adi alanının veri uzunluğunu 15 olarak değiştirir.

Alan tipini değiştirmek için, ALTER TABLE komutuna ALTER COLUMN deyiminin eklenmesi gerekir.

**Örnek:** `ALTER TABLE` ogrenci `ALTER COLUMN` adi `INT`;  
Sorgusunun işlevini yazınız.

Öğrenci tablosundaki adi alanının veri tipini int olarak değiştirir.

**Örnek:** `ALTER TABLE` ogrenci `ALTER COLUMN` notu `FLOAT`;  
Sorgusunun işlevini yazınız.

Öğrenci tablosundaki notu alanının veri tipini float olarak değiştirir.

**Örnek:** `ALTER TABLE` ogrenci `ALTER COLUMN` cins `BIT`;  
Sorgusunun işlevini yazınız.

Öğrenci tablosundaki cins kolonunun veri tipini bit olarak değiştirir.

## 6.8.4.3 Mevcut Bir Tablodan bir kolon Silme

Tablo üzerinden alan silmek için, ALTER TABLE komutuna DROP COLUMN deyimini eklenmesi gerekir.

**Örnek:** `ALTER TABLE` ogrenci `DROP COLUMN` yas;  
sorgusunun işlevini yazınız.

Öğrenci tablosundaki yaş kolonunu siler.

**Örnek:** `ALTER TABLE` personel `DROP COLUMN` cinsiyet;  
sorgusunun işlevini yazınız.

Personel tablosundaki cinsiyet kolonunu siler.

## 6.8.4.4. Mevcut Bir Tablonun Bir Kolonunun Adını Değiştirme

Tablonun adını değiştirmek için, ALTER TABLE komutuna RENAME TABLE deyiminin eklenmesi gerekir.

**Kullanılışı:** EXEC sp\_rename tablo\_adi. alan adi 'name' , 'column';

**Örnek:** ALTER TABLE ogrenci RENAME adi ogr\_adi; sorgusunun işlevini yazınız.

Öğrenci tablosundaki adi alanını ogr\_adi olarak değiştirir.

**Örnek:** EXEC SP\_RENAME 'ogrenci.adi','ogr\_adi', COLUMN; sorgusunun işlevini yazınız.

Öğrenci tablosundaki adi alanını ogr\_adi olarak değiştirir.

**Örnek:** EXEC SP\_RENAME 'personel.cins', 'cinsiyet', COLUMN; sorgusunun işlevini yazınız.

Personel tablosundaki cins kolonunun adını cinsiyet olarak değiştirir.

## 6.8.4.5. Mevcut Bir Tablonun Tümüyle Silinmesi

Bir tablonun tamamını silmek için DROP TABLE komutu kullanılır.

**Örnek:** DROP TABLE öğrenci;  
sorgusunun işlevini yazınız.

-Öğrenci tablosunu siler.

**Örnek:** DROP TABLE notlar;  
sorgusunun işlevini yazınız.

-Notlar tablosunu siler.



## 6.8.4.6. Mevcut Bir Tablonun Yapısının Görülmesi

Bir tablonun yapısını görmek için DESC[IRIBE] komutu kullanılır. Bu komut ile tanımlanan tabloların alan bilgileri alınır. Tablo alanları, alanların veri tipleri unutulduğunda veya kontrol edilmek istenildiğinde sıkça kullanılan bir komuttur.

**Kullanılışı:** DESC table\_name;

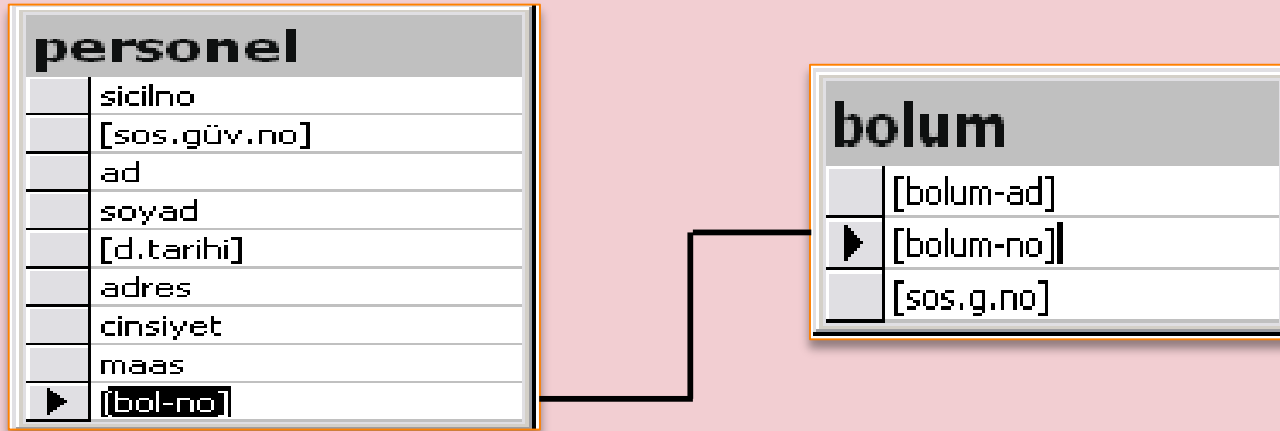
**Örnek:** Desc Personel;

| Name         | Type      |
|--------------|-----------|
| -----        | -----     |
| SİCİL NO     | NUMBER(4) |
| ADI          | CHAR(15)  |
| SOYADI       | CHAR(15)  |
| DOĞUM TARİHİ | DATE      |

**Not:** Desc komutu SQL Server 2008 de tanımlı değildir.

## 6.9.SQL DE JOIN (BİRLEŞTİRME) İŞLEMİ

Daha sık karşılaşılan ve daha zor olan sorgulamalar birden çok tablonun birbiriyle ilişkilendirilmesiyle gerçekleştirilir.



**Şekil 6.1:** SQL'de İki Tablo Arasındaki İlişkilendirme

Çalışan her personel ve bu personelin yöneticisiyle ilişkili bilgiler nelerdir? Bu sorunun cevabını tek tabloda bulamayız. Her iki tabloyu da kullanmak zorundayız. Bunu yapmak için de müşterek bir alan kullanmalıyız. Müşterek alan burada bölüm numarasıdır. Personel tablosunda böl-no, bölüm tablosunda bölüm-no adı ile yer almaktadır.

```
SELECT * FROM personel, bölüm WHERE  
personel.böl_no=bölüm.bölüm_no
```

Şeklinde birleştirme yapabiliriz.

**Örnek:** Bu konunun örneklerini aşağıdaki üç tabloya göre çözelim.

## Oğrenci Tablosu

| <u>no</u> | <u>adi</u> | <u>soyadi</u> | <u>sinif</u> | <u>yasi</u> |
|-----------|------------|---------------|--------------|-------------|
| 1         | A          | B             | 1            | 20          |
| 2         | C          | D             | 2            | 21          |

## Notlar Tablosu

| <u>og-no</u> | <u>op-kod</u> | <u>vize</u> | <u>final</u> | <u>but</u> | <u>ort</u> |
|--------------|---------------|-------------|--------------|------------|------------|
| 1            | 121           | 20          | 30           | 70         | 45         |
| 1            | 123           | 40          | 90           | null       | 65         |
| 2            | 121           | 70          | 70           | null       | 70         |
| 2            | 123           | 10          | 20           | 30         | 20         |

## Dersler Tablosu

| <u>op-kod</u> | <u>d-adi</u> | <u>teori</u> | <u>pratik</u> | <u>kredi</u> |
|---------------|--------------|--------------|---------------|--------------|
| 121           | Bilg         | 2            | 2             | 3            |
| 123           | Elk          | 3            | 0             | 3            |

**Örnek:** `SELECT *FROM ogrenci, notlar;` sorgusunun işlevini yazınız.

Öğrenci ve notlar tablosunu kartezyen olarak listeler.

**Örnek:** `SELECT *FROM ogrenci, notlar, dersler WHERE no=og_no AND dersler.op_kod=notlar.op_kod;` sorgusunun işlevini yazınız.

**Örnek:** Optik kodu 121 olan dersi alan öğrencilerin adlarını listeleyen sorguyu yazınız.

`SELECT adi FROM ogrenci, notlar WHERE ogrenci.no=notlar.no AND op_kod=121`

**Örnek:** Dersleri ve ortalamaları veren sorguyu yazınız.

`SELECT d_adi, AVG(ort) FROM dersler, notlar WHERE dersler.Op_kod=notlar.op_kod GROUP BY d_adi`

**Örnek:** Öğrencilerin maksimum ortalamalarını veren sorguyu yazınız.

`SELECT adi, MAX(ort) FROM ogrenci, notlar, WHERE ogrenci.no=notlar.no GROUP BY adi`

adi

--- ---

A 65

C 70

**Örnek:** Aldığı derslerin ortalaması 50'nin üzerinde olmasına rağmen finalde 50'nin altında not alan ve kalan öğrencilerin tüm bilgilerini veren sorguyu yazınız.

```
SELECT * FROM ogrenci, notlar, dersler WHERE no=og_no AND  
dersler.op_kod=notlar.op_kod AND final<50 AND (vize+final)/2>50
```

**Örnek:** Her öğrencinin finalde aldığı en yüksek notu bulan sorguyu yazınız.

```
SELECT adi, MAX(final) FROM ogrenci, notlar WHERE og_no=no  
GROUP BY adi;
```

**Örnek:** Bölüm kodu 531 olan öğrencinin memleket adlarını bulan sorguyu yazınız. (Okul Projesi)

```
select adi,soyadi,m_adi from ogrenci,memleket where  
memleket=tr_kod and bolkod=531
```

|   | adi   | soyadi | m_adi  |
|---|-------|--------|--------|
| 1 | Betül | Cebe   | Amasya |

**Örnek:** Finalden en yüksek notu alan öğrencinin adını, hangi dersten aldığını ve final notunu bulan sorguyu yazınız.

```
select adi,ders_adi,final from  
ogrenci,notlar,dersler where  
ogrenci.no=notlar.no and  
dersler.op_kod=notlar.op_kod and  
final=(select MAX(final) from notlar)
```

|   | adi  | ders_adi  | final |
|---|------|-----------|-------|
| 1 | Veli | Bil.prg.2 | 99    |

# **7.INDEKS OLUŞTURMA ve KULLANMA**

BÖLÜM 7

# 7.1 İNDEKS OLUŞTURMANIN AMACI

- ⌘ Bir indeks, veri tabanı ortamında bir tablo ya da view gibi bir nesnedir ve ilişkili olarak kullanıldığı tablo ya da view'deki satırların, indeksleme alanı (key field-anahtar alan) olarak kullanılan kolondaki verilere göre sıralanmasını sağlar. İndeksler tablodaki verilere daha hızlı erişim sağlayan bir mekanizmadır. Bir indeks bir tablonun bir ya da daha çok alanı (kolonu) üzerinde oluşturabilir. İki tür indeks vardır.



## 7.2 İNDEKS ÇEŞİTLERİ

- ⌘ Clustered İndeksler
- ⌘ Nonclustered İndeksler
- ⌘ **7.2.1 Clustered İndeksler:** Bu indeks tablo satırlarını kendi değerlerine göre sıralar ve saklar. Satırların bir sırası olduğundan bu tür indekste aranan satırların bulunmasını kolaydır. Tablonun fiziksel yapısını değiştirirler.
- ⌘ **7.2.2 NonClustered İndeksler:** Bu indeksler satırların düzeninden bağımsız olarak düzenlenirler. Nonclustered indeksler bir pointer olarak veri satırlarını gösterdiğinden arana satırların bulunması kolaydır. SQL Server'da iki türlü indeks oluşturulabilir.

⌘ Bunlardan

⌘ 1-CREATE INDEX deyimidir.

⌘ 2-CREATE TABLE deyimiyle indeks oluşturan kısıtlamalar yapılmasıdır.

⌘ Primary Key ve Unique kısıtlama terimleri otomatik olarak index oluştururlar. Clustered indexler null bilgilere izin vermezler. Unique indexler ise null değerinin girilmesine izin verirler. Clustered ve nonclustered indexlerin ikisi de unique olabilirler.

⌘ `CREATE TABLE ogrenci (no INT PRIMARY KEY,  
adi CHAR(10) ,soyad char(15) ,..... .) ;`

- ⌘ Tablolar üzerinde index düzenlemek genellikle yararlı bir iştir. Ancak index oluşturma zaman alması disk alan gereksinimini ve verinin girilmesi, değiştirilmesi, silinmesi durumunda güncellenmesi gereksinimi index kullanımında düşünülmesi gereken faktörlerdendir. Her tabloda bir tane clustered, 249 tane nonclustered index oluşturulabilir. Clustered index tablonun fiziksel yapısını değiştirir.

Kullanılışı: `CREATE CLUSTERED index index_adi ON tablo_adi(alan);`

`CREATE INDEX ogrindex ON ogrenci(no);`

`CREATE CLUSTERED INDEX ogrindex ON ogrenci(adi);`

- ⌘ **NOT:** Sadece create index dersek nonclustered olur.

## 7.3 İNDEKS YARATMA

SQL'de bir tablo ile ilişkili olarak indeks yaratmak için gerekli komut CREATE INDEX komutudur. Komutun yazılış biçimi aşağıdaki gibidir.

**Kullanılışı:** `CREATE INDEX` indeks adı

`ON` tablo adı (kolon adı 1, kolon adı 2, kolon adı 3, ..., kolon adı n);

İndeksleme artan ya da azalan şeklinde olabilir. Artan, alfabetik olarak A'dan Z'ye nümerik olarak küçükten büyüğe şeklindedir. Azalan ise bunun tersidir. Hiçbir özel sözcük kullanılmazsa indeksleme artan sayılır ya da alan adının yanında bir boşluktan sonra ASC sözcüğü kullanılırsa bu aklana göre artan sıralama yapılacak demektir. Her hangi bir alan adının yanında DESC sözcüğü kullanılması ise indekslemenin azalan olacağını gösterir. Komutun yazılış biçiminden anlaşılabacağı gibi, aynı anda, birden çok alana göre indeksleme de yapılabilir.

## 7.3.1 Tek Bir Alana Göre Artan Sırada İndeksleme

İşletmede çalışan personeli brüt maaşlarına göre artan sırada listelemek istersek, brüt alanına bir indeks oluşturmamız gerekir.

```
CREATE INDEX pers_maas ON personel (brut);  
INDEX CREATED 127 Rows
```

127 satırlık personel tablosu ile ilişkili olarak brüt kolonunu indeks anahtarı olarak kullanan pers-maas adlı indeks oluşturulmuştur.

Bu durumda

```
SELECT * FROM personel;
```

Şeklindeki listeleme komutu sonucunda, personel tablosundaki tüm personel, brüt maaşlarına göre sıralı olarak listelenecektir.

## 7.3.2 Tek Bir Alana Göre Azalan Sırada İndeksleme

- ⌘ İşletmede çalışan personeli brüt maaşlarına göre azalan sırada (yüksek maaştan düşük maaşa doğru) listelemek istersek, brüt alanına göre aşağıdaki gibi oluşturmak gerekir.

```
CREATE INDEX per_ymaas ON personel (brut DESC);
```

## 7.3.3 Birden Fazla Alana Göre İndeksleme

- ⌘ İşletmedeki personelin öncelikle adlarına göre, aynı adda olanların soyadlarına göre hem adı hem de soyadı aynı olanların maaşlarına göre sıralanmış olarak listelenmesi istenirse aşağıdaki komut kullanılır;

```
CREATE INDEX p_ad_soy_m ON personel  
(ad, soyad, brut) ;
```

Bu durumda `SELECT * FROM personel;`

- ⌘ Komut sonucunda, aşağıdaki şekilde sıralanmış tablo görüntülenecektir.

| ⌘ | <u>Sicil</u> | <u>Adı</u> | <u>Soyad</u> | <u>Brüt</u> |       |
|---|--------------|------------|--------------|-------------|-------|
| ⌘ | 11117        | Ahmet      | Caner        | 1500000     | ..... |
| ⌘ | 247          | Ahmet      | Deniz        | 2700000     | ..... |
| ⌘ | 645          | Ahmet      | Zoran        | 1200000     | ..... |
| ⌘ | 15372        | Ali        | Cenker       | 3400000     | ..... |
| ⌘ | 4246         | Ali        | Cenker       | 6500000     | ..... |
| ⌘ | 7216         | Beril      | Arkan        | 1800000     | ..... |
| ⌘ | 3871         | Burcu      | Günel        | 2600000     | ..... |
| ⌘ | 16656        | Fırat      | Günel        | 1200000     | ..... |
| ⌘ | .....        | .....      | .....        | .....       |       |



Burada kolayca görüleceği gibi personel öncelikle adı alanına göre sıralanmış(Ahmet, ali, beril) aynı ada sahip olanlar soyadlarına göre sıralanmış(Ahmetler, Caner, Deniz, Zoran şeklinde) hem ad hem de soyadı aynı olanların sıralanmasında ise brüt alanı dikkate alınmıştır.

İndeks komutu

```
CREATE INDEX p_ad_soy_m ON personel  
(ad, soyad, brut DESC) ;
```

şeklinde yazılsa idi, tablodaki değerler:

| ⌘ | sicil | adı      | soyad    | Brüt    |     |
|---|-------|----------|----------|---------|-----|
| ⌘ | _____ | _____    | _____    | _____   |     |
| ⌘ | 11117 | Burcu    | Aktürk   | 1500000 | ... |
| ⌘ | 7216  | Burcu    | Güenal   | 1800000 | ... |
| ⌘ | 645   | Can      | Gül      | 1200000 | ... |
| ⌘ | 247   | Fırat    | Güenal   | 2700000 | ... |
| ⌘ | 3871  | Meral    | Bozkurt  | 6500000 | ... |
| ⌘ | 16656 | Muhammed | Bakır    | 1200000 | ... |
| ⌘ | 15372 | Serap    | Bostan   | 3400000 | ... |
| ⌘ | 4246  | Seval    | Dağtekin | 2600000 | ... |

şeklinde sıralanırdı. Bu durumda farklı olan ad ve soyad alanı aynı olan kişilerin brüt maaşlarına göre, yüksek maaştan düşük maaşa göre sıralanmış olmasıdır.(brüt DESC ifadesinden dolayı).

## 7.3.4 Unique Sözcüğü

- ⌘ Bir tablo, seçilen bir sütuna göre indekslenirken indeksleme alanı olarak seçilen sütundaki verilerin tekrarlanmasının müsaade edilmesi istenmiyorsa, indeksleme yapılırken, CREATE INDEX komutu içinde UNIQUE sözcüğü kullanılmalıdır:

**Örnek:** `CREATE UNIQUE INDEX pers_sicil ON personel(sicil);`

UNIQUE sözcüğünün etkisi, bu komuttan sonra, tabloda, aynı sicilden birden fazla tekrar olmasını engellemesidir.

Personel tablosuna

```
INSERT INTO personel VALUES (53768, 2724168,
'ayşe', 'şen', {01/04/63}, 'Merkez Cad. 82-
Kadıköy', .F., 2700000, 2, 34261578)
```

Komutu ile sicil 53768 olan kişi eklenmek istendiği zaman, bu sicilden daha önce o tabloda mevcutsa, ekleme kabul edilmeyecek ve

- Error -data is not unique
- Hata - Veri tekrarsız(tek) değildir. Şeklinde bir hata mesajı alınacaktır.

## 7.3.5 Mevcut Bir İndeksin Silinmesi

- ⌘ Bir tablo üzerinde tanımlanmış herhangi bir indeks, o tablonun veri tabanından silinmesi ile otomatik olarak silinecektir. Tablo silinmeksizin, o tablo üzerinde oluşturulan indeksin silinmesi içinse, DROP INDEX komutu kullanılmalıdır.

**Örnek:** `DROP INDEX pers_in;`  
`INDEX DROPPED`  
(İndeks silindi)

mesajı alınacaktır. Böylece, personel tablosu üzerinde oluşturulmuş pers\_in adlı indeks, personel tablosu veri tabanında kaldığı halde silinecektir.

## 7.4 PRIMARY KEY

Oluşturulan tabloda primary key her satır için tek olan değerleri ifade eder. Örneğin ogrenci\_kodu, sicil\_no gibi alanlar primary key olabilir. Bir tabloda sadece bir primary key olabilir. Primary key olan bütün kolonlar not null olarak belirtilmelidir. Yani null bilgi olmamalıdır.

**Örnek:** `CREATE TABLE ogrenci (og_no INT  
PRIMARY KEY, adi CHAR(10), ...)`

**Örnek:** `CREATE TABLE bolum (bolkod SMALLINT  
PRIMARY KEY NOT NULL, bol_adi CHAR(15))`

## 7.5 UNIQUE KEY

Primary Key olmayan satırların tek değerler almasını sağlar. Unique her satırda farklı bir değer anlamına gelir. Mevcut değerler teklik bakımından kontrol edilir. Ayrıca yeni bir değer girilirken satırlar kontrol edilir. Bir unique index bir kolon ya da birçok kolon üzerinde oluşturulabilir.

**Örnek:** `CREATE TABLE ogrenci (no INT PRIMARY KEY, adi CHAR(10) UNIQUE, soyadi CHAR(15) UNIQUE);`

**Örnek:** `CREATE NONCLUSTERED INDEX ogr_index ON ogrenci (adi, soyadi);`

**Örnek:** `CREATE UNIQUE CLUSTERED INDEX deneme ON ogrenci (adi);`

## 7.6 DEFAULT

Tanımlanan bir alanın default değer almasını sağlar. Ayrıca default değerler için fonksiyonlar da kullanılabilir.

**Örnek:** `CREATE TABLE tabloadı (alanadı  
veritipi, ..... uyruğu CHAR(5) DEFAULT 'TC');`  
sorgusunun işlevini yazınız.

Uyruğu kolonu olan ve bu kolona değer girilmediğinde TC yazan öğrenci tablosun oluşturulur.

**Örnek:** `CREATE TABLE ogrenci (kay_tar  
DATETIME DEFAULT GETDATE()) ;`

## 7.7 FOREIGN KEY

Bu anahtar kolon içindeki değerin diğer tabloda olmasını zorunlu kılar. Primary Key ya da Unique değerlerine karşılık olarak bir değer tablodaki karşılık değerlerini ifade eder. Foreign Key kolonuna bir değer girildiğinde referans edilen değer diğer tablo kolonunda da değer olmalıdır. Aksi takdirde hata mesajı oluşur. Foreign Key kısıtlaması aynı tablo içindeki bir diğer kolonu da referans olarak kullanılabilir. Bir tabloda maksimum 252 tane Foreign Key kısıtlaması olabilir.

**Örnek:** `CREATE TABLE siparis(sip_no INT,  
per_no INT,mus_no FOREIGN KEY REFERENCES  
musteri(mus_kodu),sip_mik INT);`

**Örnek:** `CREATE TABLE notlar(numara INT FOREIGN  
KEY REFERENCES ogrenci(no),opkod INT FOREIGN KEY  
REFERENCES dersler(opkod));`



- ⌘ İlişki(relationstip) database içindeki tablolar arasında bir bağlantı kurarak bir bütünlük sağlamaktadır. Bütünlük, aralarında ilişki olan iki tablonun artık veri işlemlerinde birbirlerini kontrol etmesini ve herhangi bir tabloya girilen değerlerin diğer tablodaki verilerle kontrol edilmesini sağlar. Örneğin müşteri tablosunda yer alamayan bir müşterinin sipariş tablosunda yer alması gibi.

- ⌘ Tablolar arasında ilişki kurarken genellikle ana tablonun alan(primary key) ,diğer tablonun alanı(foreign key) ile bağlantı kurulmasını sağlar. Bütünlük kuralı ile ilişkili tablolar arasında bütünlüğün korunması (ilişkili kayıtların ana tablodan silinmesini, alt tablolara ilgisiz kayıt yapılmamasını)sağlar. Aralarında ilişki bulunan iki tabloda; bir taraftaki tabloda yer alamayan bir kayda diğer tabloda veri girilmesi mümkün olmaz. Böylece arasındaki verilerin uyumluluğu korunmuş olur.

## Ogrenci(Primary Key)

| No       | Adı Soyadı    | Bölümü     |
|----------|---------------|------------|
| 03531014 | Burcu Aktürk  | Bilgisayar |
| 03531037 | Fırat Günel   | Elektronik |
| 03531002 | Venhar Poyraz | Bilgisayar |

## Notlar(Foreign Key)

| No       | Dönemi | Ortalama |
|----------|--------|----------|
| 03531014 | 00(1)  | 70       |
| 03531037 | 00(2)  | 71       |
| 03531002 | 00(1)  | 82       |
| 03531010 | 00(1)  | 85       |

Şekil 7.1: Primary Key ve Foreign Key'in Tablolardaki İlişkileri

## 7.8 İLİŞKİSEL BÜTÜNLÜK KURALLARI

- ⌘ Ana tablonun alanının tek (Primary Key veya Unique) olmalı ve boş olmamalı.
- ⌘ Ana tabloda bir satır yoksa ilişkili alt tablolarda hiçbir satır olmaz.
- ⌘ Alt tablolara sahip olan bir ana tablodan bir satırın silinmesi ya da değiştirilmesi durumunda alt tablolara bakılır. Eğer alt tabloda ilişkili kayıtlar varsa ana tablodaki satırların silinmesine veya değiştirilmesine izin verilmez.
- ⌘ Alt tablolara girilen bilginin ana tabloya bağlı kontrol edilerek farklı bir bilginin girilmesine izin verilmez. Diğer deyişle olmayan bir mala ya da veriye değer girilemez.

## 7.8.1 İlişki Kurmada Olası Hatalar

⌘ İki tablo arasında ilişki kurulmasını engelleyen olası durumlar vardır. Bu durumlardan sık karşılaşılanları şunlardır:

1. Bir taraftaki tabloda anahtar alanın olmaması(Primary Key ve Unique).
2. Bu alanlara null değer girilmesi.
3. İlişki kurulacak alanların aynı tipte olmaması.
4. İlişki kurulacak alanlarda tutarsız bilgilerin mevcut olması.

## 7.9. CHECK

- ⌘ Kolon ya da kolonlara girilecek değerleri sınıflandırmaya yarayan bir kısıtlamadır. Bir kolonda birden fazla check kısıtlaması olabilir. Bir check kısıtlaması kolon düzeyinde ya da tablo düzeyinde olabilir. Kolon düzeyindeki check kısıtlaması sadece o kolonu ilgilendirirken, tablo düzeyindeki check kısıtlaması aynı tablodaki diğer kolonları da kapsar.

**Örnek:** CREATE TABLE siparis(sip\_no INT, per\_no INT, mus\_no FOREIGN KEY REFERENCES muster\_i(mus\_kodu), sip\_mik INT, CHECK(sip\_mik>10));

**Örnek:** CREATE TABLE siparis (sip\_no INT, per\_no INT, mus\_no FOREIGN KEY REFERENCES muster\_i(mus\_kodu), sip\_mik INT, CHECK(mus\_no IN(10,20,30,37,45)));

**Örnek:** CREATE TABLE siparis(sip\_no INT, per\_no INT, mus\_no FOREIGN KEY REFERENCES muster\_i(mus\_kodu), sip\_mik INT, CHECK mus\_no BETWEEN 10 AND 20));

**Örnek:** CREATE TABLE notlar(vize TINYINT, final TINYINT, but TINYINT, CHECK(vize<=100), CHECK(final BETWEEN 0 AND 100), CHECK(numara IN(10,11,12,13)));

## 7.10 IDENTITY

Tabloya yeni bir satır eklendiğinde SQL bu kolona otomatik olarak artan bir değer verebilir. Bu kolonlar Primary Key kolonları gibi tablonun satırlarına tek bir değer vermeyi sağlar. Bu özellik sadece int, tinyint, numeric, decimal ve smallint gibi alanlara atanır. Aşağıdaki kullanımda X başlangıç değeri, Y ise artış miktarıdır.

**Örnek:** CREATE TABLE musteriler (musteriler\_no INT IDENTITY (X, Y), mus\_adı CHAR (10));

**Örnek:** CREATE TABLE ogrenciler (no INT IDENTITY (x, y), adı CHAR (15));

Identity özelliğine sahip kolonlara kayıt girilemez, güncellenemez ve null değerine izin vermez.



## 7.11 SQL DEĞİŞKENLERİ

SQL değişkeni belirli bir türdeki değişkeni etkiler. SQL'de değişken tanımlamak için DECLARE ifadesi kullanılır. Değişkenlerin önünde @ işareti bulunur. Tanımlanan değişkenin ilk değeri null'dur. Değişkene bir değer atamak için SET ifadesi kullanılır.

**Örnek:** Numara adında bir değişken oluşturun, ona 10 değerini atayın ve buna uyan bilgilere sahip öğrenci bilgilerini görüntüleyin.

```
go
DECLARE @numara INT
SET @numara=10
SELECT * FROM ogrenci WHERE no=@numara
go
```

**Örnek:** İki Tarih arasını bulur.

```
DECLARE @Date1 smalldatetime
DECLARE @Date2 smalldatetime
SET @Date1 = '2011.05.01'
SET @Date2 = '2011.06.01'
SELECT DATEDIFF(d, @Date1, @Date2)
```

| (No column name) |    |
|------------------|----|
| 1                | 31 |

**Örnek:** Numarası 5531519 olan öğrencinin adı ve soyadını veren sorguyu değişken yardımıyla yapın. (Okul Projesi)

```
declare @numara int
set @numara=5531519
select adi,soyadi from ogrenci where no=@numara
```

|   | adi   | soyadi |
|---|-------|--------|
| 1 | Betül | Cebe   |

**Örnek:** Devre analinin dersinin vizesinden mi daha başarılılar yoksa finalinden mi , sorgusun gerçekleştiren sql sorgusunu yazınız?(okul projesi)

```
DECLARE @vize INT
DECLARE @final INT
SELECT @vize= AVG(vize),@final=AVG(final) FROM
notlar,dersler WHERE dersler.op_kod=notlar.op_kod AND
ders_adi='devre Analizi'
IF @final>@vize
PRINT'finalden daha başarılılar'
ELSE
PRINT'vizeden daha başarılılar'
```

```
finalden daha başarılılar
```

## 7.10.1 SQL'in Kendi Değişkenleri

@@servername

@@version

@@connection

@@language

⌘ **Örnek:** Veri tabanınızda dili İngilizce yapın. (Okul Projesi)

```
SELECT @@language english;
```

```
<|
english
-----
us_english
(1 row(s) affected)
```

⌘ **Örnek:** `SELECT getdate() AS 'Bu gunun tarih ve zamani', @@CONNECTIONS AS 'Baglanti Sayisi';` sorgusunun çıktısını yazınız.

|   | Bu gunun tarih ve zamani | baglanti sayisi |
|---|--------------------------|-----------------|
| 1 | 2011-05-31 17:44:35.973  | 51              |

⌘ **Örnek:** `SELECT 'Burcu' AS 'Adi';` sorgusunun işlevini ve çıktısını yazınız.

Bu sorgudaki “as” komutu kolonun adının “Adi” olmasını sağlıyor.

adi

-----

Burcu

## ***7.12. SQL'DE BAZI FONKSİYONLAR***

- ⌘ ABS : Mutlak değer hesaplar.
- ⌘ COS : Raydan olarak açının cosinüsünü hesaplar.
- ⌘ SIN : Raydan olarak açının sinüsünü hesaplar.
- ⌘ ACOS : Raydan olarak açının arccosinüsünü hesaplar.
- ⌘ ASIN : Raydan olarak açının arcsinüsünü hesaplar.
- ⌘ TAN : Raydan olarak açının tanjantını hesaplar.
- ⌘ ATAN : Raydan olarak açının arctanjantını hesaplar.
- ⌘ LOG : Belirtilen sayının doğal logaritmasını verir.
- ⌘ PI : 3.14 sayısını verir.
- ⌘ EXP : Üs alır.
- ⌘ POWER : Bir değerın kuvvetini alır.
- ⌘ RAND : 0 ile 1 arasında rasgele sayı üretir.
- ⌘ ROUND : Sayıları yuvarlar.
- ⌘ SQRT : Bir değerin karekökünü verir.
- ⌘ LOWER : Küçük harfe çevirir.

- ⌘ LTRIM : Önündeki boşlukları keser.
- ⌘ REVERSE : Bir karakter ifadesini ters çevirir.
- ⌘ RIGHT : Bir karakter ifadeyi sağ taraftan böler.
- ⌘ RTRIM : Sondaki boşlukları keser.
- ⌘ SPACE : Boşluk verir.
- ⌘ STR : Sayısal bilgiyi karaktere çevirir.
- ⌘ UPPER : Küçük karakterleri büyük harfe çevirir.
- ⌘ DATEDIFF : İki tarih arasındaki datepart sayısını verir.
- ⌘ GETDATE : Şu anki tarihi verir.
- ⌘ YEAR : Yılı verir.
- ⌘ MONTH : Ayı verir.
- ⌘ STDEV : Standart sapmayı hesaplar.

⌘ **Örnek:** `SELECT DATEPART (YEAR, GETDATE ()) AS 'BULUNDUGUMUZ YIL'`

|   | BULUNDUGUMUZ YIL |
|---|------------------|
| 1 | 2011             |

⌘ **Örnek:** `SELECT DATEPART (WEEK, GETDATE ()) AS 'BULUNDUGUMUZ HAFTA'`

|   | BULUNDUGUMUZ HAFTA |
|---|--------------------|
| 1 | 23                 |

⌘ **Örnek:** `SELECT DATENAME (MONTH, '04.15.2011') AS 'PARAMETRE OLARAK GIRILEN AYIN ISMI'`

|   | PARAMETRE OLARAK GIRILEN AYIN ISMI |
|---|------------------------------------|
| 1 | April                              |

⌘ **Örnek:** `SELECT DATEDIFF (MONTH, '2010.05.31', '2011.05.31') AS 'BELIRTILEN TARİHLERİN AY FARKI'`

|   | BELIRTILEN TARİHLERİN AY FARKI |
|---|--------------------------------|
| 1 | 12                             |

⌘ **Örnek:** Yazılan değerleri büyük harfe çevirir.

```
DECLARE @strBuyuk nvarchar(50)
SET @strBuyuk = 'Osman KALender'
SELECT UPPER (@strBuyuk)
```

|   | (No column name) |
|---|------------------|
| 1 | OSMAN KALENDER   |

⌘ **Örnek:** Yazılan değerleri küçük harfe çevirir.

```
DECLARE @strKucuk nvarchar(50)
SET @strKucuk = 'Osman KALender'
SELECT LOWER(@strKucuk)
```

|   | (No column name) |
|---|------------------|
| 1 | osman kalender   |

⌘ **Örnek:** Şu anki tarihe 7 gün ekleyelim

```
SELECT DATEADD(DD, 7, GETDATE()) AS '7 gün eklendi'
```

|   | 7 gün eklendi           |
|---|-------------------------|
| 1 | 2011-06-07 18:50:42.697 |

⌘ **Örnek:** SQL server tarihini gün ay yıl formatında yazdıralım.

```
SELECT CONVERT(VARCHAR, GETDATE(), 5) AS 'gün-ay-yıl formatında'
```

|   | gün-ay-yıl formatında |
|---|-----------------------|
| 1 | 31-05-11              |

**Örnek:** `select Reverse('ali') AS 'ismi tersten yazma'`

|   | ismi tersten yazma |
|---|--------------------|
| 1 | ila                |

⌘ **Örnek:** `select substring('mehmet Ali',1,4)`

|   | (No column name) |
|---|------------------|
| 1 | mehm             |

⌘ **Örnek:** Parametre olarak verilen sayı kadar boşluk oluşturmak için kullanılır.

`select 'SQL' + space(5) + 'SERVER'`

|   | (No column name) |
|---|------------------|
| 1 | SQL SERVER       |

⌘ **Örnek:** İfadenin sağındaki ve solundaki boşlukları temizler.

`Select Rtrim(Ltrim(' ali '))`

|   | (No column name) |
|---|------------------|
| 1 | ali              |



## 7.13 COMPUTE

- ⌘ Compute sözcüğü, toplama fonksiyonu kullanılarak Query sonucuna bir toplam satırı üretir.

### 7.13.1 COMPUTE BY

Ek toplama satırları üretir.

| Tarih | Mus_Kodu | Urun | Adet | Fiyat | Turu |
|-------|----------|------|------|-------|------|
|       |          | X_TV | 10   | 10000 | Ev   |
|       |          | X_TR | 12   | 15000 | Oto  |
|       |          | X_TR | 20   | 25000 | Oto  |
|       |          | X_TV | 15   | 20000 | Ev   |
|       |          | X_KR | 15   | 5000  | Is   |
|       |          | X_KR | 10   | 12000 | Is   |

**SELECT** urun, adet **FROM** siparis **ORDER BY** urun **COMPUTE**  
**SUM**(adet) ; çıktısını bulunuz.

| urun | adet |
|------|------|
| X_TV | 10   |
| X_TR | 12   |
| X_TR | 20   |
| X_TV | 15   |
| X_KR | 15   |
| X_KR | 10   |

Sum(adet)=82

**SELECT** urun, adet, turu **FROM** siparis **ORDER**  
**BY** urun **COMPUTE SUM**(adet)**BY** urun; çıktısını  
 bulunuz.

urun

turu

adet

-----  
X\_KR

Is

15

X\_KR

Is

10  
-----

Sum (adet) =25

X\_TR

Oto

12

X\_TR

Oto

20  
-----

Sum (adet) =32

X\_TV

Ev

10

X\_TV

Ev

15  
-----

Sum (adet) =25

## ⌘ Compute sözcüğünün kullanımında bazı kısıtlamalar vardır. Bunlar:

- ✖ DISTINCT komutu kullanılamaz.
- ✖ COMPUTE sözcüğünde bulunan kolon SELECT listesinde bulunmalıdır.
- ✖ COMPUTE BY sözcüğü ile beraber aynı şekilde ORDER BY kullanılmalıdır.

⌘ **Örnek:** `SELECT urun, turu, adet FROM siparis ORDER BY turu COMPUTE sum(adet) BY turu;` sorgusunun işlevini yazınız.

- × Sipariş tablosundaki ürün, türü, adeti listeler, türü artana göre sıralar, türe göre gruplar ve adetleri toplayarak altına yazar.

⌘ **Örnek:** Her bir öğrencinin numarasını, adını, ders adını ,ders ortalamasını ve aldığı bütün derslerin ortalamasını ek bir ortalama satırı üreten sql sorgusunu yazınız. (okul projesinden)

⌘ `Select ogrenci.no,adi,ders_adi, (vize*0.4+final*0.6) AS ortalama from dersler,notlar,ogrenci where ogrenci.no=notlar.no and dersler.op_kod=notlar.op_kod order by ogrenci.no compute avg(vize*0.4+final*0.6) by ogrenci.no`

|   | no        | adi    | ders_adi        | ortalama |
|---|-----------|--------|-----------------|----------|
| 1 | 5530535   | Feyza  | Bil.ağlan       | 51.2     |
| 2 | 5530535   | Feyza  | Rehberlik       | 72.0     |
|   | avg       |        |                 |          |
| 1 | 61.600000 |        |                 |          |
|   | no        | adi    | ders_adi        | ortalama |
| 1 | 5531519   | Betül  | Bil.ağlan       | 67.2     |
| 2 | 5531519   | Betül  | Rehberlik       | 55.6     |
|   | avg       |        |                 |          |
| 1 | 61.400000 |        |                 |          |
|   | no        | adi    | ders_adi        | ortalama |
| 1 | 6532510   | Leyla  | Veri yapılan    | 32.6     |
| 2 | 6532510   | Leyla  | Mikro. mimarisi | 63.6     |
| 3 | 6532510   | Leyla  | Teknik iletigim | 36.2     |
|   | avg       |        |                 |          |
| 1 | 44.133333 |        |                 |          |
|   | no        | adi    | ders_adi        | ortalama |
| 1 | 7536511   | Aytül  | Devre analizi   | 69.2     |
| 2 | 7536511   | Aytül  | Elk.devreler    | 84.8     |
|   | avg       |        |                 |          |
| 1 | 77.000000 |        |                 |          |
|   | no        | adi    | ders_adi        | ortalama |
| 1 | 7536521   | Veli   | Devre analizi   | 55.2     |
| 2 | 7536521   | Veli   | Bil.prg.2       | 76.8     |
|   | avg       |        |                 |          |
| 1 | 66.000000 |        |                 |          |
|   | no        | adi    | ders_adi        | ortalama |
| 1 | 7536545   | Cengiz | Devre analizi   | 27.4     |
| 2 | 7536545   | Cengiz | Bil.prg.        | 11.4     |
|   | avg       |        |                 |          |
| 1 | 19.400000 |        |                 |          |

## 7.14 AKIŞ KONTROLLERİ

1. Case
2. If-Else
3. Begin-End
4. While

## 7.14.1. CASE

- ⌘ Belirli bir ifadenin değerine göre bir değeri döndüren ve o değere göre yönlenen bir yapı sağlar.

### Kullanılışı:

CASE ifade

When ifade1 Then ifade2

When ifade3 Then ifade4

...

...

Else ifade n



## Örnek:

| Kodu | Grubu | Adi  | Soyadi | Bakiye      |
|------|-------|------|--------|-------------|
| 1    | A     | .... | ....   | 100.000.000 |
| 2    | F     | .... | ....   | 199.000.000 |
| 3    | B     | .... | ....   | 17.000.000  |
| 4    | C     | .... | ....   | 1.500 000   |

```
SELECT Adi, Soyadi, musteri_grubu= CASE Grubu
WHEN 'A' THEN 'Cirosu 100 milyonun üstünde'
WHEN 'B' THEN 'Cirosu 10 milyonun üstünde'
WHEN 'C' THEN 'Cirosu 1 milyonun üstünde'
ELSE 'Diğer sınıfta'
END FROM musteri ORDER BY adi;
```

Bu sorgu sonucunun çıktısı ne olur, yazınız.

| Adi  | Soyadi | musteri_grubu        |
|------|--------|----------------------|
| ---- | -----  | -----                |
| ...  | ...    | 100 milyonun üstünde |
| ...  | ...    | Diğer sınıfta        |
| ...  | ...    | 10 milyonun üstünde  |
| ...  | ...    | 1 milyonun üstünde   |

## Örnek:

```
SELECT Bolumu=CASE bolum
WHEN 'Elk' THEN 'elektronik'
WHEN 'Bil' THEN 'bilgisayar'
.
.
ELSE 'Bolum belli degil' END,
CAST (adres as varchar(10)) AS 'kisaca adresi',
Ort AS ortalama FROM ogrenci WHERE ort IS NOT
```

| No  | Adi | Soyadi | Bolum | Ort | Adres |
|-----|-----|--------|-------|-----|-------|
| ... | ... | ...    | Elk   | 20  | EL... |
| ... | ... | ...    | Elk   | 30  | ...   |
| ... | ... | ...    | Bil   | 40  | ...   |
| ... | ... | ...    | Oto   | 60  | ...   |
| ... | ... | ...    | Yap   | 50  | ...   |
| ... | ... | ...    | Bil   | 80  | ...   |

Sorgu sonucunda çıktı ne olur, yazınız.

| Bolumu            | kısaca adresi | ortalama |
|-------------------|---------------|----------|
| -----             | -----         | -----    |
| Bilgisayar        | ..... .       | 40       |
| Bilgisayar        | ..... .       | 80       |
|                   |               | Avg=60   |
| Elektronik        | .....         | 20       |
| Elektronik        | .....         | 30       |
|                   |               | Avg=25   |
| Bolum Belli Değil | .....         | 60       |
|                   |               | Avg=60   |
| Bolum Belli Değil | .....         | 50       |
|                   |               | Avg=50   |

**Örnek:** Çıktının aşağıdaki gibi olması için sorgu nasıl olur?

Ogrenci

no

adi

ort

| no  | adi | ort | durum |
|-----|-----|-----|-------|
| --- | --- | --- | ----- |
| 1   | a   | 70  | B     |
| 2   | b   | 89  | A     |
| 3   | c   | 30  | Kaldı |
| 4   | d   | 55  | C     |

```
SELECT no, adi, soyadi, ort, durum=CASE
WHEN ort<50 THEN 'Kaldı'
WHEN ort>84 THEN 'A'
WHEN ort>49 and ort<65 THEN 'C'
ELSE 'B'
END FROM ogrenci
```

## Örnek:

| <u>no</u> | <u>cinsiyet</u> | <u>ort</u> |
|-----------|-----------------|------------|
| 1         | 1               | 40         |
| 2         | 0               | 50         |
| 3         | 0               | 60         |

Yukarıdaki tabloda cinsiyet kısmına erkek ve bayan yazdırmak için aşağıdaki sorguyu yaparız.

```
SELECT no, cinsiyet=CASE cinsiyet  
WHEN 1 THEN 'Erkek'  
WHEN 0 THEN 'Bayan'  
END ort FROM ogrenci
```

**Örnek:** Aşağıdaki sorgu ne işe yarar, yazınız.

```
DECLARE @ a INT
```

```
SET @ a= (SELECT AVG(yasi) FROM ogrenci)
```

```
CASE @ a
```

```
WHEN 19 THEN PRINT 'sinif genç'
```

```
WHEN 20 THEN PRINT 'sinif yasli'
```

```
END
```

A değişkenine sınıfın ortalamasını atıp a eğer 19 ise “sınıf genç”, 20 ise “sınıf yaşlı” yazar.

## Örnek:

```
SELECT category= CASE type
WHEN 'pop_comp' THEN 'Populer Compatina'
WHEN 'mod_cook' THEN 'Modern Cooking'
...
...
ELSE 'not yet' END
CAST(TITLE AS VARCHAR(25)) AS 'Shortered
Title',
Price As Price FROM titles WHERE price is
NOT NULL ORDER BY type,
Price COMPUTE AVG(price) BY type;
```



| Category  | Shortered Title | Price |
|-----------|-----------------|-------|
| Popular   |                 |       |
| Compotina |                 |       |
| Cooking   |                 |       |
| ...       |                 |       |
| ...       |                 |       |
| not yet   |                 |       |

- ⌘ Title başlığını 'Shortered Title' olarak değiştirir. Uzunluğunu 25 karakter olarak alır. Price değeri null olmayanları yazar.
- ⌘ **SELECT @adi AS 'isim';**

|      |
|------|
| isim |
| ...  |

Adi değişkeninde bulunan değeri yazar.

## 7.14.2 IF-ELSE

### Kullanılışı:

**IF koşul**

**Sql Deyimleri**

**ELSE**

**Sql Deyimleri**

**Örnek:** Aşağıdaki sorgunun işlevi nedir, yazınız.

```
IF (SELECT AVG(adet) FROM siparis) >10
    SELECT mkodu, urun, adet FROM siparis
ELSE
    SELECT 'Ortalama adet 10 dan kucuk';
```

Ortalama adet 10'dan büyük ise "SELECT mkodu, urun, adet FROM siparis" bu komut satırını gerçekleştiren, değil ise "ortalama adet 10'dan küçük" yazar.

**NOT:**

```
IF @sayi >10
    PRINT '...'
...
ELSE
    ...
```

SQL'de ekrana yazı yazdırmak için 'print' komut da kullanılır.

**Örnek:** `Print 'Burcu' + ' ' + 'Günel'` çıktısı nasıl olur, yazınız.

Burcu Günel

## 7.14.3 BEGIN-END

- ⌘ Birden fazla SQL satırı varsa Begin-End kullanılabilir. BEGIN ve END deyimleri genellikle şu durumlarda kullanılır.
  - × WHILE ile döngü yapıldığında.
  - × Bir CASE fonksiyonunun blok deyimi içermesi durumunda.
  - × IF ve ELSE deyiminde

**Örnek:** Aşağıdaki sorgunun işlevini yazınız.

```
IF (SELECT AVG(adet) FROM siparis) > 10
    BEGIN
        PRINT 'Ortalama adet 10'dan büyük'
        SELECT mkodu, urun, adet FROM siparis
    END
ELSE
    SELECT 'Ortalama adet 10'dan küçük';
```

Ortalama adet 10'dan büyük ise 'Ortalama adet 10'dan büyük' yazar, SELECT mkodu, urun, adet FROM siparis uygular, değilse 'Ortalama adet 10'dan küçük' yazar.

## 7.14.4 WHILE

Kullanılışı: While koşul

Begin

Sql deyimleri

Break

Sql deyimleri

Continue

Sql deyimleri

End

Örnek: Aşağıdaki sorguyu yorumlayınız.

```
WHILE (SELECT AVG(adet) FROM siparis) <
    50
BEGIN
UPDATE siparis
SET adet= adet* 2
SELECT MAX(adet) FROM siparis
IF (SELECT MAX(adet) FROM siparis) > 50
BREAK
ELSE
CONTINUE
END
PRINT 'adet degeri büyük'
```

## BREAK ve CONTINUE kullanmadan:

```
WHILE (SELECT AVG(adet) FROM siparis)  
    < 50
```

```
BEGIN UPDATE siparis  
    SET adet= adet* 2
```

```
END
```

```
PRINT 'adet degeri büyük '
```

Satış adetleri 50 oluncaya kadar adet alanının artır.



## ÖDEV:

| No | Adi   | Soyadi | Ort | Harf  |
|----|-------|--------|-----|-------|
| 1  | Burcu | Aktürk | 70  | B     |
| 2  | ...   | ....   | 90  | A     |
| 3  |       |        | 50  | D     |
| 4  |       |        | 10  | Kaldı |

Yukarıdaki tabloya göre öğrencilerin ortalamalarını dikkate alarak harf alanını oluşturan sorguyu yazınız.

Kalanları bularak kalanlar kolonu altında listeleyen sorguyu oluşturun.

# UYGULAMALAR

1.Öğrencinin adını, soyadını, numarasını, danışmanın adını ve soyadını veren sorguyu yazınız.

```
select no,adi,soyadi,h_adi,h_soyadi from  
ogrenci,hocalar where  
ogrenci.h_ID=hocalar.h_ID
```

**ÇIKTI:**

| Results |         | Messages |           |         |            |
|---------|---------|----------|-----------|---------|------------|
|         | no      | adi      | soyadi    | h_adi   | h_soyadi   |
| 1       | 5530535 | Feyza    | Uçan      | Servet  | Tuncer     |
| 2       | 5531519 | Betül    | Cebe      | İbrahim | Türkoğlu   |
| 3       | 6532510 | Leyla    | Korkmaz   | Asaf    | Varol      |
| 4       | 6532518 | Ali      | Yılmaz    | Asaf    | Varol      |
| 5       | 6535501 | Bilal    | Baymaz    | Engin   | Avcı       |
| 6       | 7536511 | Aytül    | Çetinkaya | Erkan   | Tanyıldızı |
| 7       | 7536521 | Veli     | Uşak      | Erkan   | Tanyıldızı |
| 8       | 7536545 | Cengiz   | Güneş     | Erkan   | Tanyıldızı |

2.Cinsiyeti bayan erkek olarak gösteren sorguyu yazınız.

```
select adi,cinsiyet=CASE cinsiyet
when 1 then 'erkek'
when 0 then 'bayan'
end from ogrenci
```

3.Bilgisayar bölümü öğrencilerinin tüm vize ,final,büt ortalamalarını bulan sorguyu yazınız.

```
select Avg(vize),Avg(final),avg(but) from notlar,
ogrenci where notlar.no=ogrenci.no and bolkod='531'
veya
```

```
select Avg(vize),Avg(final),avg(but) from notlar,
ogrenci,bolum where notlar.no=ogrenci.no and
ogrenci.bolkod=bolum.bolkod and bol_adi='Bilgisayar'
```

**ÇIKTI:**

| Results |                  | Messages         |                  |
|---------|------------------|------------------|------------------|
|         | (No column name) | (No column name) | (No column name) |
| 1       | 51               | 68               | 93               |

4. Her Öğrencinin tüm ders ortalamalarını bulan sorguyu yazınız.

```
select adi, ders_adi, (vize*0.4+final*0.6) from  
dersler, notlar, ogrenci where  
notlar.op_kod=dersler.op_kod and  
ogrenci.no=notlar.no
```

**ÇIKTI:**

| Results |        | Messages        |                  |
|---------|--------|-----------------|------------------|
|         | adi    | ders_adi        | (No column name) |
| 1       | Feyza  | Rehberlik       | 68.0             |
| 2       | Feyza  | Bil. ağılan     | 51.2             |
| 3       | Betül  | Rehberlik       | 55.6             |
| 4       | Betül  | Bil. ağılan     | 67.2             |
| 5       | Leyla  | Teknik iletişim | 36.2             |
| 6       | Leyla  | Mikro. mimarisi | 63.6             |
| 7       | Leyla  | Veri yapılan    | 32.6             |
| 8       | Cengiz | Bil.prg.        | 11.4             |
| 9       | Cengiz | Devre analizi   | 27.4             |
| 10      | Veli   | Bil.prg. 2      | 76.8             |
| 11      | Veli   | Devre analizi   | 55.2             |
| 12      | Aytül  | Elk.Devreler    | 84.8             |
| 13      | Aytül  | Devre analizi   | 69.2             |

5. Bir öğrencinin tüm derslerden ortalamalarını bulan sorguyu yazınız.

```
select adi, soyadi, avg(vize*0.4+final*0.6)
from notlar, ogrenci where
ogrenci.no=notlar.no group by
ogrenci.no, adi, soyadi
```

**ÇIKTI:**

| Results |        |           |                  | Messages |  |  |  |
|---------|--------|-----------|------------------|----------|--|--|--|
|         | adi    | soyadi    | (No column name) |          |  |  |  |
| 1       | Feyza  | Uçan      | 59.600000        |          |  |  |  |
| 2       | Betül  | Cebe      | 61.400000        |          |  |  |  |
| 3       | Leyla  | Korkmaz   | 44.133333        |          |  |  |  |
| 4       | Aytül  | Çetinkaya | 77.000000        |          |  |  |  |
| 5       | Veli   | Uşak      | 66.000000        |          |  |  |  |
| 6       | Cengiz | Güneş     | 19.400000        |          |  |  |  |

6.

| <b>Bolkod</b> | <b>Soyadi</b> | <b>Adi</b> | <b>Ort</b> |
|---------------|---------------|------------|------------|
| 531           | Kaya          | Ali        | 70         |
| 531           | Gel           | Veli       | 80         |
| .             |               |            | +          |
| .             |               |            | <hr/>      |
| .             |               |            | 75         |
| 530           | Gel           | Veli       | 20         |
| 530           | Kaya          | Ali        | 10         |
|               |               |            | +          |
|               |               |            | <hr/>      |
|               |               |            | 15         |

Yukarıdaki formatta çıktıyı veren sql komutlarını yazınız.

```
select bolkod,adi,soyadi,avg(vize*0.4+final*0.6) from
notlar,ogrenci where ogrenci.no=notlar.no group by
adi,soyadi,bolkod order by bolkod compute
avg(vize*0.4+final*0.6) by bolkod
```

## 7. Tabloya harf alanını ekleyerek güncelleyen sorguyu yazınız.

```
Alter table tabloadi add harf char(2)
update tablo set harf=CASE
when ort<50 then 'Kaldı'
when ort>84 then 'A'
when ort>49 and ort<65 then 'C'
else 'B'
end from tabloadi
```





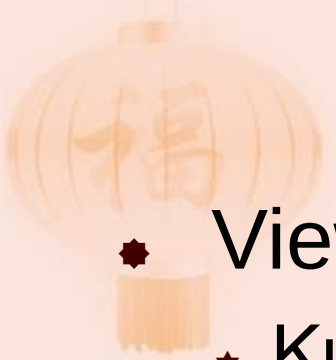
# VIEW (BAKIŞ) OLUŞTURMA

BÖLÜM 8



## 8.1 VIEW NEDİR?

- Bazen, tabloları olduklarından farklı gösterecek filtreleme ihtiyacı duyulur. Bu türden işlevler için VIEW kullanılır. View'ler, saklanmış sorgulardan ibarettirler. Veri tabanı içerisindeki farklı kullanıcıların veri tabanına nasıl baktıklarını ya da bakış açılarını anlatır. View'ler bir tablo gibi kullanılıyor olmalarına rağmen aslında böyle bir tablo veritabanında bulunmaz ve sadece görüntüsü bulunur. View'ler çok faydalı veritabanı elemanıdır



## ★ View'ler şu görevler için kullanılır:

- ★ Kullanıcıların bazı kritik tabloların sadece belli sütunlarını veya satırlarını görmesi istenildiğinde,
- ★ Kullanıcıların, çeşitli birim dönüşümlerinden geçmiş değer görmeleri gerektiğinde,
- ★ Hâlihazırdaki tablolarda var olan verilerin başka bir tablo formatında sunulması gerektiğinde,
- ★ Çok karmaşık sorguları basitleştirmek için

- **Örnek olarak;** bir kullanıcı örgenci tablosunda sadece 1.sınıf öğrencileri ile ilgilenecekse bu kullanıcıya sadece 1.sınıf öğrencilerini içeren bir View tablosu oluşturabilir.
- **NOT:** Ms Access'te View oluşturulamaz. SQL Server, SyBase, Oracle gibi orta ve büyük ölçekli VTYS'lerde oluşturulabilir.
- Persolene ait sicilno, ad ve soyad bilgisi Personel tablosunda, maaş bilgisi ise maslar tablosunda bulunmaktadır. Personele ait maaşlar listelenmek istenildiğinde; her seferinde sorgu yazılması gerekmektedir. Bunun yerine bir view yaratılarak bu view listelenirse her seferinde sorgu yazmaya gerek kalmaz. Bu view daha sonra form veya rapor alımında da kullanılabilir ve burada olduğu gibi büyük kolaylık sağlar.

✳ **Örnek:** CREATE VIEW personel\_maas AS SELECT  
a.sicilno, a.adi, a.soyadi, b.maas FROM  
personel a,maslar b WHERE  
a.sicilno=b.sicilno

✳ **Örnek:** Öğrencilerin adını, soyadını ve memleketini  
veren bir vie oluşturun.(okul projesinden)

```
create view ogrenci_mem as select  
adi,soyadi,m_adi from memleket,ogrenci where  
memleket=tr_kod
```

Command(s) completed successfully.

```
select * from ogrenci_mem where m_adi  
like '%a%'
```

|   | adi    | soyadi    | m_adi      |
|---|--------|-----------|------------|
| 1 | Feyza  | Uçan      | Samsun     |
| 2 | Betül  | Cebe      | Amasya     |
| 3 | Leyla  | Korkmaz   | Adana      |
| 4 | Aytül  | Çetinkaya | Karaman    |
| 5 | Veli   | Uşak      | Diyarbakır |
| 6 | Cengiz | Güneş     | Yozgat     |



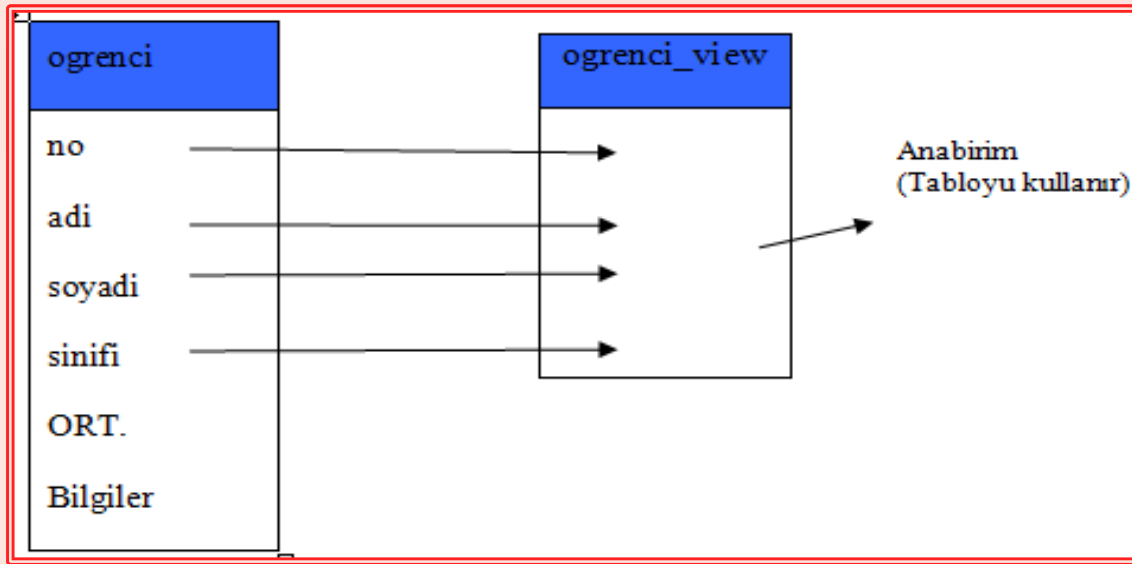
## 8.2 VIEW'İN FAYDALARI

### 8.2.1 Veri Güvenliği

Veritabanı içindeki bazı alanların sütunların herkes tarafından görünmesi istenmeyebilir. Bu durumda view kullanılmalıdır. Örneğin öğrencilerin bulunduğu bir tabloda notların öğrenciler tarafından görünmesi istenmiyorsa bu durumda view oluşturulabilir. Bunun için: CREATE VIEW komutu kullanılır.

Kullanılışı: `CREATE VIEW view_adi;`

Örnek: `CREATE VIEW ogrenci_view AS SELECT  
no, adi, soyadi, sinifi FROM ogrenci`



Şekil 8.1: Bir Örnekle View Görünümü

Bu view tüm öğrencilere açıldığı zaman bu kullanıcılar no, adi, soyadi, sinifi bilgilerine ulaşabilir. Bu view öğrenciler tarafından bir tablo gibi sorgulanarak kullanılabilir.

**Örnek:** Oluşturduğumuz viewi görüntüleyin.

```
SELECT * FROM ogrenci_view;
```

View üzerinde işlemler yapılabilir. View'de oluşturulan alan adları tablodaki alanlarla(adlarla) aynı olmak zorunda değildir.

**Örnek:** CREATE VIEW ogr\_view (numara, isim, soyisim, sinif) AS  
'view le yapilmis tablo' SELECT no, adi, soyadi,  
sinifi FROM ogrenci

**Örnek:** SELECT numara, isim FROM ogr\_view

**Örnek:** En başarılı öğrencinin il kodunu, il adını ve genel ortalamasını veren sorguyu yapınız?(okul projesinden)

```
Create view bas_il(il_kod,il_adi,genel_ort) AS  
Select memleket,m_adi,AVG(vize*0.4+final*0.6) from  
ogrenci,memleket,notlar where ogrenci.no=notlar.no  
and tr_kod=memleket group by memleket,m_adi  
go
```

```
Select il_kod,il_adi,genel_ort from  
memleket,bas_il where tr_kod=il_kod and  
genel_ort=(select MAX(genel_ort) from bas_il)
```

|   | il_kod | il_adi  | genel_ort |
|---|--------|---------|-----------|
| 1 | 70     | Karaman | 77.000000 |



## 8.2.2.Sorgulamaların Basitleştirilmesi

Karmaşık sorgulamalar, view özelliği kullanılarak daha basit hale getirilebilir. Örneğin karışık bir select sorgusu içerisinde başka bir select komutu kullanılıyorsa, bunun yerine view oluşturularak sorgulama basitleştirilebilir.

**Örnek:** Satış bölümünde çalışan personelin herhangi birinden daha düşük maaş alan ve mühendislik bölümünde çalışan kişileri listeleyiniz.

```
SELECT * FROM personel where brut<ANY (SELECT  
brut FROM personel WHERE bolum='satis') AND  
bolum='mühendis'
```

Şimdi bu sorunun cevabı olan tablo bir view olarak saklanırsa

```
CREATE VIEW deneme AS  
SELECT * FROM personel WHERE brut < ANY  
(SELECT brut FROM personel WHERE bolum='satis')  
AND bolum='mühendis'
```

**Örnek:** Öğrencilerden her desten en yüksek notu alan öğrencinin adı,dersin kodu ve aldığı notu veren sql sorgusunu yazınız? (okul projesinden)

```
Create View yuksek(ders_kodu,ynot) AS Select  
op_kod,MAX(final) from notlar group by op_kod  
go
```

```
select adi,ders_kodu,ynot from ogrenci,yuksek,notlar where  
ogrenci.no=notlar.no and yuksek.ders_kodu=notlar.op_kod  
and ynot=final
```

|   | adi    | ders_kodu | ynot |
|---|--------|-----------|------|
| 1 | Betül  | 433       | 70   |
| 2 | Feyza  | 421       | 80   |
| 3 | Leyla  | 373       | 47   |
| 4 | Leyla  | 372       | 66   |
| 5 | Leyla  | 371       | 45   |
| 6 | Aytül  | 274       | 80   |
| 7 | Cengiz | 271       | 11   |
| 8 | Veli   | 222       | 90   |
| 9 | Aytül  | 218       | 76   |

- Bundan sonra aynı tip sorgulama için sadece SELECT \* FROM deneme; yazmak yeterli olacaktır.

## 8.2.3.Sadece View'le Yapılan İşlemler

Bazı durumlarda sorgulama yapabilmek için view oluşturmak kaçınılmaz olur. Bu durumda view kullanmak zorundayız.

Örnek: Her sınıfta o sınıf ortalamasından daha yüksek not ortalaması olanları listeleyiniz

| <u>Ogrenci</u> | <u>Deneme</u> |
|----------------|---------------|
|----------------|---------------|

|    |          |
|----|----------|
| No | Siniflar |
|----|----------|

|     |          |
|-----|----------|
| Adi | Ortalama |
|-----|----------|

|        |  |
|--------|--|
| Soyadi |  |
|--------|--|

|       |  |
|-------|--|
| Sinif |  |
|-------|--|

|     |  |
|-----|--|
| Ort |  |
|-----|--|

|      |  |
|------|--|
| Yasi |  |
|------|--|

Bu sorunun cevaplandırılması için önce her sınıftaki ortalama notların bulunması gereklidir.

```
CREATE VIEW sinif_ort(sinif,ort) AS
SELECT sinifi, AVG(ort) FROM ogrenci GROUP
BY sinifi
```

Daha sonra oluşturulan sinif\_ort yardımı ile (bu view sınıf no'larını ve sınıf ortalama notlarını saklamaktadır.) sorulan sorunun cevabı elde edilebilir.

```
SELECT * FROM ogrenci, sinif_ort WHERE
sinif_ort. sinifi=ogrenci. sinifi AND
ogrenci. ort>sinif_ort;
```

**Örnek:** Her sınıfın yaş ortalamasından yaşı daha büyük olanları listleyen program

```
CREATE VIEW yasort(sinifi, yasort) AS
SELECT sinif, AVG(yasi) FROM ogrenci GROUP
BY sinif
```

```
SELECT * FROM ogrenci, yasort WHERE
sinif=sinifi AND yasi>yasort
```

**Örnek:** Öğrenciler arasından en çalışkan olanın yaşını ve genel ortalamasını veren sql sorgusunu bulunuz? (okul projesinden)

```
Create View encalis1(yas,y_ort) AS select  
DATEDIFF(yy,d_tarihi,getdate()),Avg(vize*0.4  
+final*0.6) from ogrenci,notlar  
where ogrenci.no=notlar.no group by d_tarihi  
go  
select yas,y_ort from encalis1 where  
y_ort=(select MAX(y_ort) from encalis1)
```

|   | yas | y_ort     |
|---|-----|-----------|
| 1 | 24  | 77.000000 |



## 8.2.4. Veri Bütünlüğü

View oluşturulurken “CHECK” sözcüğünün kullanılması ile o view'deki veri bütünlüğünde güvenlik sağlanmış ve bütünlük korunmuş olur.

**Örnek:** CREATE VIEW notlar AS SELECT \* FROM  
ogrenci WHERE ort>49 WITH CHECK OPTION

Bu view içerisinde notu 49'dan büyük olan öğrenciler bulunduğundan INSERT INTO ile notu 50'den küçük, bir öğrenci kaydedilmek istendiğinde hata mesajı verilerek kayıt engellenecektir.



## 8.3.VIEW ÜZERİNDE DEĞİŞİKLİK YAPMA

- ✱ View üzerindeki değişiklikler tablo üzerindeki değişikliklere benzemektedir. Ancak bazı şartları vardır.
  1. Birden fazla tablo ilişkilendirilmeden oluşturulmuş olmalı(ilişkilendirme olmamalı)
  2. İçerisinde MAX, MIN, SUM gibi fonksiyonlar olmamalı.
  3. DISTINCT, GROUP BY, HAVING gibi sözcükler bulunmamalı.

## 8.3.1 View İçine Satır Ekleme(Kayıt)

Daha önceden oluşturulmuş öğrenci adlı view, ad, soyad ve not alanlarını içermiş olsun. Bu view, güncellenebilir nitelikte ise, aşağıdaki INSERT komutu ile aynen tablolarda olduğu gibi kendisine bir satır eklemek mümkündür.

**Örnek:**      INSERT      INTO      öğrenci      VALUES  
( 'burcu' , 'aktürk' , 70 ) ;  
sorgusunun işlevini yazınız.

Öğrenci viewine kayıt girer.





## 8.3.2.View İçinden Satır Silme

Güncellenebilir bir view içinde satır silme işlemi, tablolarda satır silme işlemi ile aynı şekilde gerçekleştirilir.

**Örnek:** DELETE FROM notlar WHERE  
not<45; sorgusunun işlevini yazınız.

Notlar içinden, notu 45'den az olan kişileri silmektir.



### 8.3.3.View Satırları İçerisinde Güncelleme İşlemi:

Güncellenebilir view'lerde güncelleme işlemi tablolardakinin aynısıdır.

**Örnek:**    UPDATE            SIN\_OR\_VIEW            SET  
not=80        WHERE        sin\_no=42;        sorgusunun  
işlevini yazınız.

SIN\_OR\_VIEW adlı view'de sınıf numarası 42 olan örgencinin notunu 80 olarak değiştirmektir.



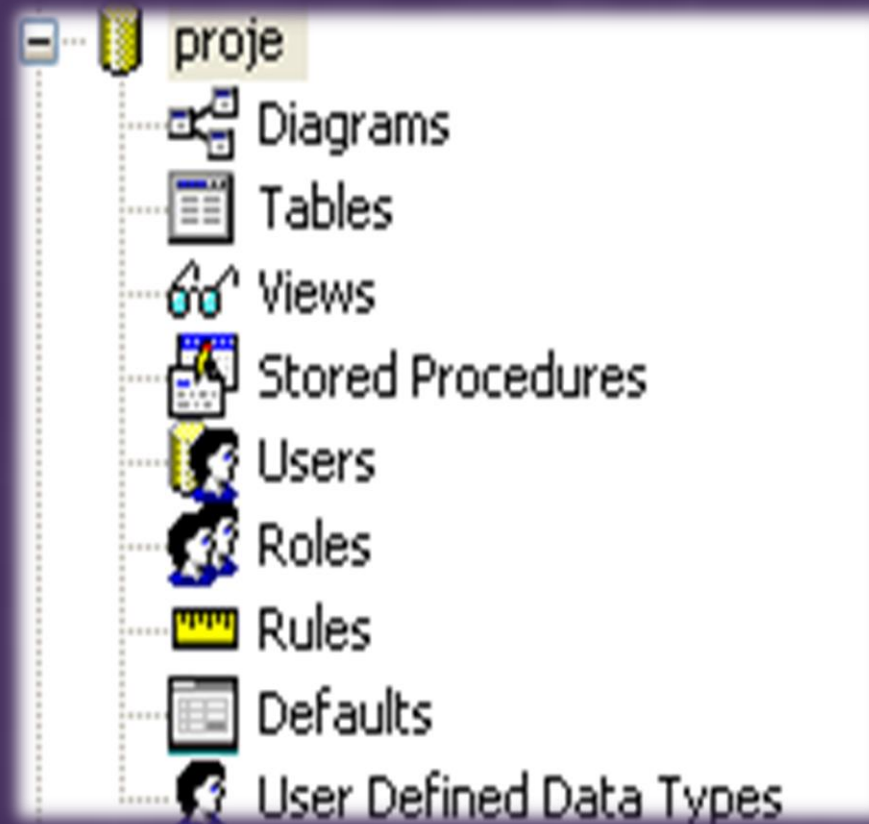
### 8.3.4. Bir View'i Silmek

- Tabloların silinmesine benzer şekilde, sistemde oluşturulabilen bir view, DROP VIEW komutu ile silinebilir.
- Örnek: `DROP VIEW SIN_OR_VIEW;`
- Bir view'in silinmesi ile o view'e bağlı olarak oluşturulmuş diğer bütün view'ler ve bu view ile ilişkili önceliklerin de tümü silinmiş olacaktır.

# BÖLÜM 9

## VERİ TABANI NESNELERİ

# 9.1 VERİTABANI NESNELERİ



Şekil 9.1: SQL Nesneleri



## 9.1.1 DIAGRAMS

✧ Veritabanı diyagramları veritabanı içindeki tabloları temsil ederler. Diyagramlar; tablo içindeki kolonları, tablolar arasındaki ilişkileri, tabloların indekslerini ve kısıtlamalarını gösterirler. Veritabanı diyagramları ile şu işlemler yapılabilir.

- ... Tablolar ve tablolar arası ilişkiler görüntülenebilir.
- ... Tabloların fiziksel yapısını değiştirmek gibi çeşitli işlemler yapılabilir.
- ... Yapılan değişiklikler tablolara kaydedilebilir.
- ... Tablolar arasında yeni ilişkiler oluşturulabilir.



## 9.1.2 TABLES

Ø Viewler tabloların belli kolonların listelendiği ayrıca hesaplama işlemlerinin yapıldığı bir veri erişim yöntemidir.



## 9.1.3.ROLES

- ✧ Veritabanı roles'leri izinlerin uygulanacağı birçok kullanıcının bir araya getirilmesidir. SQL Server, Server düzeyinde ve veritabanı düzeyinde roller oluşturabilir. Ayrıca SQL Server belirli yönetim fonksiyonlarına sahip hazır tanımlanmış birçok roll'ere sahiptir. Bu roll'er kolayca kullanıcılara atanabilir. Bunun dışında kullanıcı tarafından özel roll'er de oluşturulabilir.





Ø Tanımlı server roll'er yönetimsel düzeydeki dizinlerin gruplanmasını sağlar. Bu roll'er kullanıcı veritabanları üzerinde bağımsız olarak düzenlenebilir. Tanımlı roll'er dışında sistem yöneticileri kendi kullanıcıları içinde roll'er düzenleyebilirler. Böylece bir grup kullanıcının dizinlerinde toplu değişiklik yapılacağı için yönetim fonksiyonu kolaylaşacaktır.



## 9.1.4. RULES

Bir tabloya girilen verileri kısıtlamanın bir diğer yolu da rule oluşturmaktır. Bir rule bir koşul ile tanımlanır. Koşulu Like, Between gibi ifadelerle de oluşturabiliriz. Koşulun önünde @ işareti olması yerel değişken olmasını sağlar.

Örnek: CREATE RULE DENEME AS  
@DEGER<1000 AND @DEGER >500

Örnek: CREATE RULE LISTE AS NO  
IN ( '10' , '20' , '30' ...)

## 9.1.5. STORED PROCEDURE

- ✧ SQL Server içinde çok sayıda sistem stored procedure yer alır. Bunlar databaseler ve kullanıcılar hakkında bilgiler verirler ya da belli işlemleri yerine getiriler. SQL Server içindeki stored procedure ların kullanımı programları, performans, güvenlik gibi birçok konuda sistem yöneticisine kolaylık sağlar.(İleride bu konuya detaylı olarak değinilecektir.)



## 9.1.6. USER

✧ Bir database'i kullanmak için tanımlanmış adlardır. Kullanıcıların Windows NT ya da SQL Server kimlik denetimlerinin ardından databaseleri kullanmaları için bir kullanıcı adına gereksinim duyulur. Database kullanıcı adları ve rolleri kullanıcıların database üzerinde bir işlem yapmasını sağlayacak izinleri içerir. Bir database için geçerli olan ve güvenlik izinlerinin uygulandığı kişi database kullanıcılarıdır.



## 9.1.7. DEFAULT

Ø Veri bütünlüğünü zorlamak ve kolonları kısıtlamak için kullanılan iki yöntemden birisi defaultlar yaratmaktır. Bir default, bir kolona hazır bir değer girilmesini sağlar.



## 9.1.8. USER DEFINED VERİ TYPES (KULLANICI TANIMLI VERİ TİPLERİ)

- ✧ Kullanıcı tanımlı veri tipleri, SQL Server sistem veri tiplerine dayanırlar. Bu veri tipleri birçok tablonun aynı veri tipini kullanması gerektiğinde kullanılan bir düzenlemedir. Böylece kolonların aynı veri tipinde, aynı uzunlukta ve aynı null düzenlemesine sahip olduğundan emin olunur. Kullanıcı tanımlı veri tiplerine daha önceden oluşturulmuş bir rule veya default da bağlanabilir.



- **Örnek:** Aşağıdaki komut satırını yorumlayın.

```
Ogr_no_type (CHAR (8), NOT NULL, RULE LIKE '---531%', DEFAULT='0000000')
```

```
CREATE TABLE ogrenci (no ogr_no_type, adi CHAR);
```

Kullanıcı ogr\_no\_type isminde bir veri tipi tanımlamış ve özelliklerini belirterek daha sonra bu veri tipini kullanmıştır.

### Tablolarda Otomatik Değer Hesaplatma

- **Örnek:**

```
CREATE TABLE notlar (no INT FOREIGN KEY REFERENCES ogrenci (og_no), optik _kod INT FOREIGN KEY REFERENCES dersler (opkod), vize SMALLINT, final SMALLINT, ort AS (vize*0.4 + final*0.6))
```

```
- INSERT INTO notlar VALUES (10, 510, 90, 60,)
```

|    |     |    |    |
|----|-----|----|----|
| 10 | 510 | 90 | 60 |
|----|-----|----|----|

## 9.3 STORED PROCEDURE

✧ Belirli işlemleri gerçekleştiren kod parçalarıdır. SQL Server'daki Stored Procedure'ler diğer programlama dillerindeki alt programlara benzerler. Stored Procedure'lerle aşağıdaki işlemler yapılabilir.

- ... Input(giriş) parametrelerini kabul eder ve birçok değerin geri dönmesini sağlarlar.
- ... Veritabanı içindeki işlemleri yapmak için programlama deyimleri içerirler.
- ... Bazı işlemler hakkında durum bilgisi döndürürler.





✍ T-SQL deyimlerinden oluşan Stored Procedureler derlenmiş olarak veritabanında saklanırlar. Query Analizer veya diğer uygulamalar tarafından işletilebilirler. Stored Procedureler genellikle rutin hale gelmiş işleri kolayca yapmak için geliştirilmişlerdir. T-SQL deyimleri ile yazılan Stored Procedureler sadece ilk çalıştırıldıklarında derlenirler. Daha sonra çalıştırıldıklarında derlenmezler. Böylece hızlı bir şekilde çalışmış olurlar. Stored Procedure'lerin kullanılmasının yararları şunlardır.

- ... Modüler programlamayı desteklerler. Bir kez oluşturulan Stored Procedure'ler program içerisinde birçok kez çağrılabilir.
- ... Hızlı işletilebilir. Çok sayıda T-SQL deyimlerinin tekrarlanarak işletilmesi durumunda Stored Procedure'lerin kullanılması yararlı olur.
- ... Ağ(Network) trafiğini azaltırlar. T-SQL deyimlerinin ağ üzerinde tek tek işletilmesi yerine bir Stored Procedure' ün bir kez işletilmesi ağ trafiğini azaltır.



Stored Procedure oluşturmak için CREATE PROCEDURE deyimi kullanılır. İstenirse daha sonra da ALTER PROCEDURE deyimi ile Procedure'ün içeriği değiştirilebilir.

**Kullanılışı:**

```
CREATE PROCEDURE Procedure_ismi Değişken  
listesi AS SQL deyimleri
```

- Örnek:

```
CREATE PROCEDURE deneme
```

```
  @no INT
```

```
  @adi CHAR(10)
```

```
AS
```

```
SELECT * FROM ogrenci WHERE no=@no AND adi=@adi
```

Bu Procedure'ü çalıştırmak için Execute (Exec) deyimi kullanılır.

```
EXEC DENEME @no=510,@adi='burcu' ;
```

```
EXEC DENEME 510, 'burcu' ;
```

- **Örnek:** Parametre olarak girilen numaraya ait öğrencinin tüm bilgilerini 3 defa listeleyen bir Stored Procedure yazınız. (Okul Projesi)

```
Create proc tek3
```

```
@numara int AS
```

```
Declare @sayi smallint
```

```
Set @sayi=0
```

```
While (@sayi<3)
```

```
Begin
```

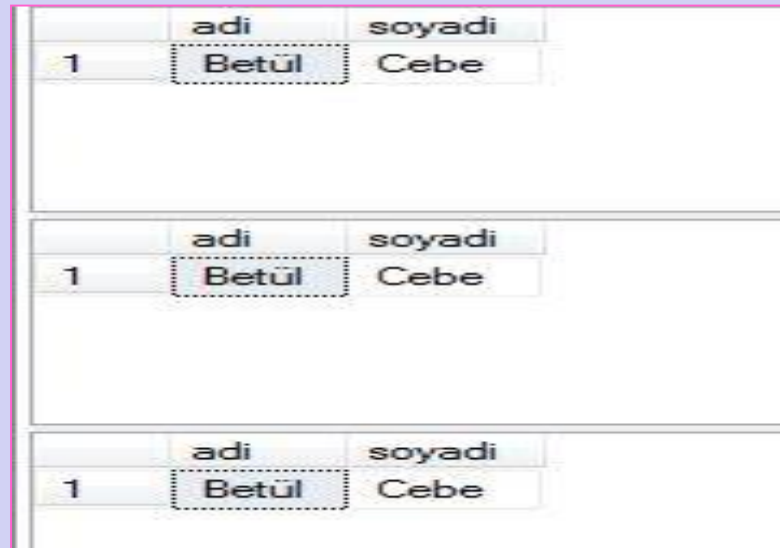
```
Select adi,soyadi from ogrenci where no=@numara
```

```
Set @sayi=@sayi+1
```

```
End
```

```
go
```

```
Exec tek3 @numara=5531519
```



The image displays three identical screenshots of a SQL query result, stacked vertically. Each screenshot shows a table with two columns: 'adi' (first name) and 'soyadi' (last name). The first row in each table has the value '1' in the first column and 'Betül Cebe' in the second column. The 'adi' cell is highlighted with a dashed border, indicating it is selected or focused.

|   | adi   | soyadi |
|---|-------|--------|
| 1 | Betül | Cebe   |

- **Örnek:** Ortalamaya göre sonuç bölümüne geçti veya kaldı yazan program yapınız.

| No       | Adi   | Soyadi | Ortalama |
|----------|-------|--------|----------|
| 03531037 | Fırat | Günel  | 60       |
| 03531014 | Burcu | Aktürk | 80       |
| 03531000 | Baran | Can    | 55       |

```
CREATE PROCEDURE durum
@no CHAR(8)
AS
DECLARE @durum char(5)
IF (SELECT ortalama FROM ogrenci WHERE no=@no)>50
SET @durum= 'Geçti'
ELSE SET @durum='Kaldı'
go
SELECT no, adi, soyadi, ortalama, @durum FROM ogrenci
WHERE no=@no
```

- **Örnek:** Sınıf bilgisi girildiğinde o sınıfa ait öğrencilerin harf alanına ait bilgileri güncellensin.

| Ogrenci |
|---------|
| No      |
| Adi     |
| Soyadi  |
| Sinifi  |
| Ort     |
| Harf    |

| No       | Adi   | Soyadi | Sinif | Ortalama | Harf |
|----------|-------|--------|-------|----------|------|
| 03531014 | Burcu | Aktürk | 4     | 60       | Null |
| 03531037 | Fırat | Günel  | 4     | 48       | Null |
| 03531000 | Baran | Can    | 4     | 55       | Null |

```
CREATE PROCEDURE ekle
```

```
@sinif INT
```

```
AS
```

```
UPDATE ogrenci SET harf= 'A' WHERE sinif=@sinif  
AND ort>85
```

```
UPDATE ogrenci SET harf= 'B' WHERE sinif=@sinif  
AND ort>64 AND ort<=85
```

```
UPDATE ogrenci SET harf= 'C' WHERE sinif=@sinif  
AND ort>49 AND ort<=64
```

```
UPDATE ogrenci SET harf= 'D' WHERE sinif=@sinif  
AND ort>34 AND ort<=49
```

```
UPDATE ogrenci SET harf= 'E' WHERE sinif=@sinif  
AND ort>=0 AND ort<=34
```

```
go
```

```
exec ekle 3
```

- Örnek: Girilen katsayılar tabloda yoksa 2. dereceden denklem köklerini hesaplayan Procedure yazınız.

|            |
|------------|
| Denklem    |
| Akatsayisi |
| Bkatsayisi |
| Ckatsayisi |
| Kok1       |
| Kok2       |

|      |
|------|
| Kok3 |
|------|



```
CREATE PROCEDURE hesapla
```

```
@a INT,
```

```
@b INT,
```

```
@c INT,
```

```
AS
```

```
DECLARE @kok1,@kok2,@delta FLOAT
```

```
IF (SELECT COUNT (*) FROM denklem WHERE Akatsayisi=@a AND Bkatsayisi=@b AND  
Ckatsayisi=@c)<1
```

```
BEGIN
```

```
SET @delta=@b*@b-4*@a*@c
```

```
IF @delta<0 PRINT 'kökler sanal'
```

```
IF @delta=0
```

```
BEGIN
```

```
SET @kok1=(-1)*@b/(2*@a)
```

```
SET @kok2=@kok1
```

```
END
```

```
IF @delta >0
```

```
BEGIN
```

```
SET @kok1=((-1)*@b+SQRT(@delta))/(2*@a)
```

```
SET @kok2=((-1)*@b-SQRT(@delta))/(2*@a)
```

```
END
```

```
INSERT INTO denklem VALUES (@a,@b,@c,@kok1,@kok2) END
```

```
Select * from denklem where Akatsayısı=@a and Bkatsayısı=@b and Ckatsayısı=@c
```

| Akatsayısı | Bkatsayısı | Ckatsayısı | Kok1 | Kok2 |
|------------|------------|------------|------|------|
| 1          | 2          | 3          | Null | Null |
| 4          | 5          | 2          | 4    | 3    |

- **Örnek:** Aşağıdaki tablolara göre ay ve yıl bilgisi parametre olarak girildiğinde o ay içerisinde toplam ne kadar ödeme yapıldığını bulan ve personel maaşlarından fazla mı yoksa eksik mi ödeme yapıldığı hakkında bilgi veren programı yazınız.

| <u>Personel</u> | <u>Birimler</u> | <u>Maas ode</u> |
|-----------------|-----------------|-----------------|
| sicilno         | birim_no        | sicil_no        |
| adi             | birim_adi       | tarih           |
| soyadi          | calisan_sayisi  | miktar          |
| cinsiyet        | maas_katsayisi  |                 |
| birimi          |                 |                 |
| maas            |                 |                 |

```
CREATE PROCEDURE hesapla
```

```
@ay TINYINT,
```

```
@yil SMALLINT
```

```
AS
```

```
DECLARE @topmaas MONEY
```

```
DECLARE @odenen MONEY
```

```
SET @topmaas= SELECT SUM(maas) FROM personel
```

```
SET @odenen= SELECT SUM(miktar) FROM maas_ode WHERE MONTH(tarih)=  
@ay and YEAR(tarih)= @yil
```

- `if(@odenen<@topmaas) printf 'fazla para ödenmiş '`
- `if(@odenen>@topmaas) printf 'eksik para ödenmiş '`
- `Else`
- `printf 'doğru para ödenmiş '`
- Örnek: Aşağıdaki prosedurenin işlevini yazınız.

```
CREATE PROCEDURE uygulama
@numara CHAR(8) AS
UPDATE ogrenci SET sonuc=CASE
WHEN ortalama>=50 THEN 'gecti'
ELSE 'kaldi'
END FROM ogrenci WHERE no=@numara
```

Parametre olarak numarası girilen kişinin ortalamaya göre sonuc bölümüne geçti veya kaldı yazan programdır.

- **Örnek:** Aşağıdaki programın işlevini yazınız.

**CREATE PROCEDURE uygulama**

**AS**

**DECLARE @bayan TINYINT**

**DECLARE @bay TINYINT**

**SET @bayan= (SELECT COUNT(DISTINCT urunsatis.musID) FROM  
urunsatis,musteri WHERE muster.musID=urunsatis.musID AND  
cinsiyet=0)**

**SET @bay= (SELECT COUNT(DISTINCT urunsatis.musID) FROM  
urunsatis,musteri WHERE muster.musID=urunsatis.musID AND  
cinsiyet=1)**

**IF(@bayan<@bay) PRINT 'erkekler daha fazla alis veris yapmistir'**

**IF (@bayan>@bay) PRINT 'bayanlar daha fazla alis veris yapmistir'**

**IF (@bayan=@bay) PRINT 'erkekler ve bayanlar esit alis veris yapmistir'**

- Bayanların mı erkeklerin mi daha fazla alışveriş yaptığını bulan procedure programıdır.

- **Örnek:** Parametre olarak numarası girildiğinde, numarası girilen öğrencinin almış olduğu dersleri ve o derslerdeki not ortalamasını gösteren procedure sorgusunu yazınız. (okul projesinden)

**Create Proc ortalama**

**@numara int AS**

**Select ders\_adi,(vize\*0.4+final\*0.6) AS ortalama**

**From dersler,notlar where**

**dersler.op\_kod=notlar.op\_kod and no=@numara**

**go**

**exec ortalama @numara=6532510**

|   | ders_adi        | ortalama1 |
|---|-----------------|-----------|
| 1 | Veri yapılan    | 32.6      |
| 2 | Mikro. mimarisi | 63.6      |
| 3 | Teknik iletişim | 36.2      |

# UYGULAMALAR

1. Create proc deneme

@a int,

@b int,

@c int OUTPUT

AS

Set @c= (@a\*0.4+@b\*0.6)

Go

DECLARE @c int

EXECUTE deneme 30,40,@c OUTPUT

```
2.Create proc harcodemebilgisi
OgrenciNo smallint
AS
Declare @bilgi int
if(select count(*) from ogrenci
where ogrenciNo=@ogrenciNo and
harcDurumu=0)>0
set @bilgi=0
Else set @bilgi=1
return @bilgi
```



### 3.Create proc Kayıtsil

@ogrenciNo int,

@harcsongun datetime='05.09.1991'

AS

Declare @bilgi smallint

EXEC @bilgi=harcodemebilgisi

@ogrenciNo=@ogrenciNo

if(@bilgi=0)

insert into ogrsil(ogrenciNo,HarcSongun,harcdurumu)

values (ogrenciNo,@harcsongun,0)

else print 'ogrenci harcını yatırmış'



```
4.Create proc ogrencibol
@ogrenciNo char(10)=NULL,
@ogrenciadi char(15)=NULL,
@ogrencisoyadı char(15)=NULL,
@Bolum char(15)=NULL AS
Declare @sorgu varchar(150)
set @sorgu='select * from ogrenci where 1=1'
if @ogrenciNo is NOT NULL
set @sorgu=@sorgu+'and ogrenciNo='+@ogrenciNo+' '
if @ogrenciadi is NOT NULL
set @sorgu=@sorgu+'and ogrenciadi like
'%'+@ogrenciadi+'%'
if @ogrencisoyadı is NOT NULL
set @sorgu=@sorgu+'and ogrencisoyadı like
'%'+@ogrencisoyadı+'%'
if (@bolum is NOT NULL)
set @sorgu=@sorgu+'and Bolum like '%'+@bolum+'%'
exec (@sorgu)
exec ogrenciadi='Yasemin',@bolum='Bilgisayar'
@sorgu='selct * from ogrenci where 1=1 and
ogrenciadi like '%Yasemin%' and Bolum='Bilgisayar'
```

**Soru:** Yıl bilgisi parametre olarak girildiğinde 1999 doğumluların not ortalamasını bulan procedure yazınız?

```
alter proc ortalama
@yil int,
@ort int output
AS
    set @ort=(select avg(vize*0.4+final*0.6)
from notlar,ogrenci where
@yil=year(d_tarihi) and
notlar.no=ogrenci.no)
declare @ort int
go
execute ortalama 1999,@ort output
select @ort
```

## 9.4 TRİGGER

- Trigger'lar Stored Procedure'lerin özel bir türüdür. Trigger'lar tablolara yapılan INSERT, UPDATE ya da DELETE işlemlerinin ardından otomatik olarak devreye girer ve çalışırlar. Trigger'lar özellikle veri işlemlerine karşı geliştirilmiş otomatik işlem mekanizmalarıdır ve doğrudan işletilemezler. Bir tabloda çok sayıda trigger olabilir. Örneğin öğrencilerin notları veri tabanına girildiğinde bir başka tablo öğrencilere ait istatistikleri tutuyorsa bu tablodaki veriler trigger'lar sayesinde otomatik olarak güncellenebilirler.

| Tablo_adı |
|-----------|
| Sinif_ort |
| Kız_ort   |
| Erkek_ort |
| Sınıf     |

- Trigger oluşturmak için CREATE TRIGGER deyimi kullanılır. Trigger'lar tabloya uygulanan herhangi bir INSERT, UPDATE ya da DELETE işlemlerinden hemen sonra çalışırlar. Bu olaylardan istenilen herhangi biri kullanılabilir. Check, Unique Key, Primary Key, Foreign Key vb. kısıtlamalar INSERT, UPDATE, DELETE işlemlerinden önce kontrol edilirler.
- Kullanılışı: **CREATE TRIGGER** trg\_adi **ON** tablo\_adi **FOR UPDATE, INSERT, DELETE AS** SQL deyimleri
- Trigger'lar genellikle programların gerektirdiği bütünlüğü sağlamak için kullanılır. Trigger'lar çalıştırılırken Primary Key veya Foreign Key gibi kısıtlamalar bozulursa bu trigger'lar işletilemezler.

- **Örnek:** Bir öğrenciye ait not değiştirildiğinde sınıf ortalamasını otomatik olarak güncelleyen bir trigger oluşturunuz.

|            |
|------------|
| İstatistik |
| sınıf_ort  |
| kız_ort    |
| erkek_ort  |
| stdsapma   |

```
CREATE TRIGGER guncelle ON ogrenci
FOR UPDATE, DELETE
AS
UPDATE istatistik SET sinif_ort=(SELECT
AVG(ort) FROM ogrenci)
SELECT * FROM istatistik
PRINT 'sınıf ortalamasi degisti'
GO
```

Aşağıdaki işlemde trigger'lar sonsuz döngüye girerler.

```
UPDATE ogrenci SET not=100 WHERE no=99
```

## 9.4.1. TRİGGER'LAR İÇİNDE KULLANILMAYACAK BAZI ÖZELLİKLER

**Alter Table**

**Create Table**

**Drop Table**

**Alter Veritabanı**

**Create Veritabanı**

**Drop Veritabanı**

**Alter Procedure**

**Create Procedure**

**Drop Procedure**

**Alter View**

**Create View**

**Drop Index**

**Create Index**

**Create Default**



- **Örnek:** Herhangi bir öğrencinin bir dersine not girildiğinde o öğrencinin tüm derslerini ve notlarını gösteren trigger hazırlayın.

**CREATE TRIGGER** deneme **ON** notlar  
**FOR INSERT AS**

**SELECT** ders\_adi **AS** 'Dersleri', (vize\*0.4+final\*0.6)  
**AS** 'Ortalama' **FROM** dersler, notlar **WHERE**  
dersler.op\_kod=notlar.op\_kod **AND**  
og\_no=(**SELECT** og\_no **FROM** INSERTED)

### Örnek:

```
CREATE TRIGGER guncelle ON mushar
FOR INSERT
AS
DECLARE @deger NUMERIC
SELECT @deger=tutar FROM INSERTED
UPDATE musteriy SET bakiye=bakiye-@deger FROM inserted
WHERE musteriy.kodu=inserted.kodu
```

Bir INSERT Trigger işlendiğinde değer trigger tablosuna eklendiği gibi inserted tablosuna da eklenir. Bu tablo, eklenen satırı tutar. Bu düzenleme DELETE Trigger için de geçerlidir. Bir satır silindiğinde deleted tabloları kullanılır.

```
CREATE Trigger deneme ON musteriy
FOR INSERT
AS
UPDATE siparis SET islem_tipi= 'G' FROM
siparis,inserted
WHERE siparis.kodu=Inserted.kodu
```



- **Örnek:**

| <u>Dersler</u> | <u>Notlar</u> | <u>Ortalama</u> |
|----------------|---------------|-----------------|
| D_adi          | Og_no         | Og_no           |
| Op_kod         | Op_kod        | Top_kredi       |
| Teori          | Vize          | Genel_ort       |
| Kredi          | Final         | Harf            |

Dersler tablosunda gerçekleşen herhangi bir olayda kredi bilgisi değişmişse tüm öğrencilerin toplam kredi alanını güncelleyen eğer kredi bilgisi değişmemişse kredi bilgisi güncellenmedi yazan bir trigger yazınız.

```
CREATE TRIGGER cevap ON dersler
FOR UPDATE
AS
DECLARE @fark INT
SET @fark=(SELECT kredi FROM INSERTED)-(SELECT kredi FROM DELETED)
IF @fark=0 print "kredi degişmedi"
ELSE
UPDATE ortalama SET top_kredi=top_kredi+@fark FROM notlar
WHERE notlar. og_no=ortalama. og_no
AND dersler. op_kod=(SELECT op_kod FROM inserted)
```

Trigger'ları silmek için DROP TRIGGER, değiştirmek için ALTER TRIGGER komutu kullanılır.

- **Örnek:** Aşağıdaki tablolara göre örnekleri inceleyin.

| mal_no | tanım        | gelis_fiyati | satis_fiyati |
|--------|--------------|--------------|--------------|
| 1      | davidoff     | 3000000      | 4500000      |
| 2      | Dove Sabun   | 1000000      | 1500000      |
| 3      | Tadelle 2000 | 400000       | 600000       |

**Tablo a:** Mal\_tanım Tablosu

| mal_no | miktar | siparis_bilgisi_no | barkod_no | ucret    |
|--------|--------|--------------------|-----------|----------|
| 2      | 10     | 2                  | 0000030   | 15000000 |
| 3      | 300    | 3                  | 0000040   | 15000000 |

**Tablo b:** Miktar Tablosu

| müsteri_no | adi   | soyadi | adres                             | telefon      | sehir   | posta_kodu |
|------------|-------|--------|-----------------------------------|--------------|---------|------------|
| 2          | ÖZGÜR | DÖNMEZ | Rizaiye mahallesi damla apartmanı | 0536 2568963 | MALATYA | 44100      |
| 3          | HAKAN | DİLEK  | Gazi Caddesi Onur Apartmanı       | 0535 2653971 | ELAZIĞ  | 23100      |

**Tablo c:** Müsteri Tablosu

| siparis_bilgisi_no | müsteri_no | gelis_tarihi        | cikis_tarihi |
|--------------------|------------|---------------------|--------------|
| 2                  | 2          | 04.01.2009 17:55:57 | 30.05.2005   |
| 3                  | 3          | 04.01.2009 17:55:57 | 27.05.2005   |

**Tablo d:** Siparis\_bilgisi Tablosu

## Uygulama 1:

```
CREATE TRIGGER liste ON siparis_bilgisi
FOR INSERT AS
DECLARE @gel_tar DATETIME
DECLARE @cik_tar DATETIME
SET @gel_tar=(SELECT gelis_tarihi FROM INSERTED)
SET @cik_tar=(SELECT cikis_tarihi FROM INSERTED)
SELECT adi, soyadi, tanim, miktar.miktar,
siparis_bilgisi.siparis_bilgisi_no, cikis_tarihi FROM
siparis_bilgisi,müsteri,miktar,mal_tanim where
cikis_tarihi>@gel_tar AND cikis_tarihi<@cik_tar AND
müsteri.müsteri_no=siparis_bilgisi.müsteri_no AND
siparis_bilgisi.siparis_bilgisi_no=miktar.siparis_bil
gisi_no AND miktar.mal_no=mal_tanim.mal_no
INSERT INTO
siparis_bilgisi(siparis_bilgisi_no,müsteri_no,cikis_
tarihi) VALUES (33,29,'2005-06-10')
```

## Uygulama 2:

```
CREATE TRIGGER hesap ON miktar
FOR INSERT AS
DECLARE @no INT
DECLARE @topmiktar INT
SET @no=(SELECT mal_no FROM INSERTED)
SET @topmiktar=(SELECT SUM(miktar.miktar) FROM
miktar WHERE miktar.mal_no=@no group by mal_no)
SELECT tanim, ((mal_tanim.satis_fiyati)-
(mal_tanim.gelis_fiyati)) AS 'birim
fiyati', @topmiktar AS 'miktar toplami',
((mal_tanim.satis_fiyati)-
(mal_tanim.gelis_fiyati))*@topmiktar from
mal_tanim where @no=mal_tanim.mal_no
```

### Uygulama 3:

```
CREATE TRIGGER musteri_sil ON miktar
FOR DELETE AS
DECLARE @sip_bil_no INT
DECLARE @mus_no INT
SET @sip_bil_no=(SELECT siparis_bilgisi_no FROM
DELETED)
delete from siparis_bilgisi where
siparis_bilgisi.siparis_bilgisi_no=@sip_bil_no
```

## 9.5 CURSOR

- İlişkisel veritabanlarında işlemler, satırlar üzerinde yapılır. Bu satırlar select deyimi içinde where sözcüğü ile seçilir. Böylece ortaya sonuç seti çıkar. Bununla birlikte uygulamaların sonuç seti üzerinde etkin bir biçimde çalışabilmesi için bir mekanizmaya sahip olması gerekir. Bu araca Cursor(gösterici) denir. Cursor'lar satır işlemlerini yönlendiren mekanizmalardır. Cursor'ler sayesinde bir sonraki, bir önceki, birinci satır, sonuncu satır ya da n. satıra gidilebilir. Aşağıdaki işlemler cursor işlemini gerekli kılar.

- Sonuç seti üzerinde belirli bir satıra gitmek
- Sonuç seti üzerinde bulunan konumdan bir ya da birçok satırı almak
- Mevcut kullanıcılar tarafından diğer kullanıcıların yaptığı değiştirme işlemlerini değişik düzeylerde desteklemek.
- Trigger ve Stored Procedure içindeki SQL deyimlerinin sonuç seti üzerinde veri değişikliği yapmasını sağlamak.

- SQL Cursor'lar genellikle Stored Procedure ve Trigger'larda kullanılırlar ve sonuç seti olarak oluşturulan listenin diğer SQL deyimleri tarafından da kullanılmasını sağlarlar. Bir cursor'un kullanım şekli şu şekildedir.
  - Cursor tarafından dönecek SQL değişkenleri tanımlanır.
  - Declare Cursor deyimi ile bir Select ifadesi ve SQL Cursor arasında ilişki kurulur. Declare Cursor deyimi Cursor'un özelliklerini tanımlar.
  - Select deyimini işletmek için OPEN komutu kullanılır.
  - FETCH INTO deyimi ile istenilen satırların elde edilmesi sağlanır.

Cursor'un sona erdirilmesinin ardından CLOSE deyimi komutu kullanılır.

**Kullanılışı: DECLARE cursor\_adi CURSOR FOR SELECT ... FROM ... OPEN cursor\_adi**

FETCH[NEXT|PRIOR|FIRST|LAST|ABSOLUTE N|RELATIVE N]

|            |                 |
|------------|-----------------|
| Next       | Sonraki         |
| Prior      | Önceki          |
| First      | İlk             |
| Last       | Son             |
| Absolute n | n.kayıt         |
| Relative n | n sonraki kayıt |

**Tablo 9.1:** Cursor'de Kullanılan Komutlar

**Not:** Relative n'de n yerine 1,2,-3,-0 rakamları girebiliriz. Mesela -7 girdiğimizde bulunduğu kayıttan 7 önceki kayda gider.

### **SELECT @@ Fetch\_Status**

**0** (Kayıtlar arasında olduğunu gösterir.)

**1** (Kayıt sonunda olduğunu ya da kayıt olmadığını gösterir.)

Eğer cursor FORWARD ONLY ise sadece next'i kullanabiliriz.

Eğer cursor SCROLL ise hepsini kullanabiliriz.



- **Örnek:** Aşağıdaki program kayıtlar arsında dolaşmaktadır. İnceleyiniz.

```
DECLARE deneme SCROLL CURSOR FOR
SELECT * FROM ogrenci
OPEN deneme
FETCH NEXT FROM deneme           //1.kayıt
FETCH ABSOLUTE 5 FROM deneme     //5.kayıt
FETCH RELATIVE 3 FROM deneme     //-3.kayıt
FETCH PRIOR FROM deneme         //1.kayıt
FETCH LAST FROM deneme          //10.kayıt
FETCH FIRST FROM deneme         //1.kayıt
CLOSE DENEME
```

- **Örnek:** Aşağıdaki programın işlevini yazınız.

DECLARE deneme SCROLL CURSOR FOR

SELECT soyadi, adi FROM ogrenci

OPEN deneme

FETCH NEXT FROM deneme

WHILE @@Fetch\_Status=0

BEGIN

FETCH NEXT FROM deneme

END

CLOSE deneme

- Tüm kayıtları sıra ile gösterir. Sonuncu kayda geldiğinde Fetch\_Status=-1 olur, while döngüsünden çıkar ve cursor kapanır.

| No | Adi   | Soyadi  | Not |
|----|-------|---------|-----|
| 1  | Burcu | Aktürk  | 100 |
| 2  | Fırat | Günel   | 54  |
| 5  | Serap | Bostan  | 65  |
| 8  | Meral | Bozkurt | 85  |
| 9  | Enes  | Can     | 64  |
| 11 | Bilge | Ak      | 90  |

- **Örnek:** Tüm kayıtlara uğrayarak istenen kriterleri yazdıran programı cursor yardımıyla yazınız.(Okul Projesi)

```
declare deneme scroll cursor for
select soyadi,adi from ogrenci
open deneme
fetch next from deneme
while @@FETCH_STATUS=0
begin
fetch next from deneme
end
```

|   |           |        |
|---|-----------|--------|
|   | soyadi    | adi    |
| 1 | Uçan      | Feyza  |
|   | soyadi    | adi    |
| 1 | Cebe      | Betül  |
|   | soyadi    | adi    |
| 1 | Korkmaz   | Leyla  |
|   | soyadi    | adi    |
| 1 | Yılmaz    | Ali    |
|   | soyadi    | adi    |
| 1 | Baymaz    | Bilal  |
|   | soyadi    | adi    |
| 1 | Çetinkaya | Aytül  |
|   | soyadi    | adi    |
| 1 | Uşak      | Veli   |
|   | soyadi    | adi    |
| 1 | Güneş     | Cengiz |
|   | soyadi    | adi    |

- Örnek: Select \* from ogrenci sorgusunun ilk kaydına giden cursoru bulunuz.

```
DECLARE cursor1 SCROLL CURSOR FOR  
SELECT * FROM ogrenci  
OPEN cursor1  
FETCH NEXT FROM cursor1
```

- Örnek: Select \* from ogrenci sorgusunun 3. kaydına giden cursoru bulunuz.

```
DECLARE cursor1 SCROLL CURSOR FOR  
SELECT * FROM ogrenci  
OPEN cursor1  
FETCH ABSOLUTE 3 NEXT FROM cursor1
```

- **Örnek:** `Select * from ogrenci` sorgusunun ilk kaydına giden cursoru bulunuz.

```
DECLARE cursor1 SCROLL CURSOR FOR  
SELECT * FROM ogrenci  
OPEN cursor1  
FETCH FIRST NEXT FROM cursor1
```

- **Örnek:** `Select * from ogrenci` sorgusunun son kaydına giden cursoru bulunuz.

```
DECLARE cursor1 SCROLL CURSOR FOR  
SELECT *FROM ogrenci  
OPEN cursor1  
FETCH LAST FROM cursor1
```

(Son kaydı gösterir)

- **Örnek:** `Select * from ogrenci` sorgusunun son kaydın önündeki kayda giden cursoru bulunuz.

```
DECLARE cursor1 SCROLL CURSOR FOR
```

```
SELECT *FROM ogrenci
```

```
OPEN cursor1
```

```
FETCH RELATIVE -1 FROM cursor1
```

- **ÖDEV:** Aşağıdaki programı kendi oluşturduğunuz bir veritabanına uyarlayarak ne işlevi olduğunu bulunuz.

```
DECLARE @no INT
DECLARE @adi CHAR(15)
DECLARE cursor_ Scroll CURSOR FOR
SELECT no, adi FROM ogrenci
OPEN cursor_
FETCH NEXT FROM cursor_
WHILE @@Fetch_Status=0
BEGIN
    FETCH NEXT FROM CURSOR INTO @no,@adi
    SELECT @no AS 'Numarası=', @adi As 'Adi'
END
CLOSE cursor1;
```

# BÖLÜM 10

## ÖRNEKLER



| musteri |
|---------|
| Id      |
| adi     |
| soyadi  |

| mus_satis |
|-----------|
| Id        |
| barkod    |
| adet      |

| Urunlar  |
|----------|
| barkod   |
| urun_adi |
| kodu     |
| fiyati   |

| Stok_hareket |
|--------------|
| tarih        |
| barkod       |
| gelen        |
| giden        |

1- mus\_satis tablosuna kayıt girildiğinde aynı kaydı stok\_hareket tablosuna ekleyen triggerı yazınız.

```
CREATE TRIGGER trg ON mus_satis
FOR INSERT AS
DECLARE @kod SMALLINT
DECLARE @adet SMALLINT
SET @kod=(SELECT barkod FROM INSERTED)
SET @adet=(SELECT adet FROM INSERTED)
INSERT INTO stok_hareket (tarih, barkod, giden)
VALUES (GETDATE(),@kod,@adet)
```

2- Aşağıdaki tabloyu veren sorguyu yazınız.

| Urun_adi | adet  |
|----------|-------|
| -----    | ----- |
| Gofret   | 53    |
| Sakız    | 44    |

```
SELECT urun_adi, SUM(gelen-giden) FROM urunler,  
stok_hareket  
WHERE urunler. barkod=stok_hareket. barkod GROUP  
BY barkod, urun_adi
```

3- Girilen yeni KDV değerini eski KDV değeri ile değiştiren ve yeni KDV'ye göre fiyatı güncelleyen programı yazınız.

```
EXEC kdvguncelle 18.20
```

```
CREATE kdvguncelle
```

```
@ekdv TINYINT
```

```
@ykdv TINYINT AS
```

```
DECLARE @fark TINYINT
```

```
SET @fark=(@ykdv-@ekdv)
```

```
UPDATE urunler SET fiyatı=fiyatı*(1+@fark/100)
```

```
WHERE kdv=@ekdv
```

```
UPDATE urunler SET kdv=@ykdv WHERE kdv=@ekdv
```



4- Mus\_satis tablosundan herhangi bir müşteriye ait tüm bilgileri sildiğimiz zaman stok\_hareket tablosundaki onunla ilgili bütün kayıtları silen triggerı yazınız.

```
CREATE TRIGGER trg ON mus_satis
FOR DELETED
DECLARE @no SMALLINT
DECLARE @adet SMALLINT
DECLARE cadi CURSOR FOR
SELECT barkod, adet FROM DELETED
Open cadi
Fetch next from cadi into @no, @adet
Delete from stok_hareket where barkod=@no and adet=@adet
WHILE @@fetch_status=0
BEGIN
FETCH NEXT FROM cadi INTO @no, @adet
DELETE FROM stok_hareket WHERE barkod=@no AND adet=@adet
END
CLOSE cadi
DEALLOCATE cadi
```

# İlişkisel Cebir

- İlişkisel cebir ,ilişkisel veritabanları üzerinde yapılan sorgulama işlemlerinin matematiksel olarak modellenmesi ve matematiksel işlemler ile çözülmesi için kullanılır.
- İlişkisel cebirde sorgulamalar teorik olarak gerçekleşir.
- Sorgular çeşitli operatörlerin birleşiminden meydana gelir.
- Her operatör geriye bir ilişki döndürür, dolayısıyla bir sorguda birden fazla operatör kullanılabilir.

# İlişkisel Cebir Operatörleri

- Seçim(Selection)
- Atma(Projection)
- Kartezyen çarpım(Cross-product)
- Birleşim(Union)
- Küme farkı(Set-difference)
- Kesişim(Intersection)
- Doğal birleştirme(Join)



# 1.Seçim(Selection)

- Bir ilişkiden ,belirlenen seçim kriterine göre satırlar içerisinde istenilenlerin seçilerek gösterilmesi işlemini gerçekleştirir.

$\sigma_{seçim\ kriteri}$ (Tablo Adı)

- Seçim kriteri yazılırken karşılaştırma operatörleri kullanılabilir.(=,>,<, ≤, ≥, ≠)
- Birden fazla seçim kriteri belirtilirken , kriterler
- «ve» operatörü ile bağlanacaksa: «/\»  
«veya» ile bağlanacaksa :«\»  
operatörleri kullanılır.

## Örnek:

Öğrenci(ogr\_no,ogr\_adi,bolum) tablosu için;  
Bölümü Bilgisayar olan öğrencileri bul.

$\sigma_{bolum="Bilgisayar"}(\text{Öğrenci})$

## Örnek:

Ürün(urun\_kod,urun\_adi,marka,fiyat) tablosu  
için;Markası ABCve fiyatı 500 den büyük olan ürünleri  
bul.

$\sigma_{marka=ABC/\backslash fiyat>500}(\text{Ürün})$

## 2.Atma(Projection)

- Bir ilişkide istenmeyen sütunları gizlemek için kullanılır.
- Seçim işlemiyle birlikte de kullanılabilir.
- Kullanımı;  $\pi_{sütun isimleri}(Tablo Adı)$

**Örnek:**Öğrenci(ogr\_no,ogr\_adi,bolum) tablosunda

1)Sadece ogr\_no ve bölüm bilgilerinin görüntülenmesi için gerekli ifade;

$$\pi_{ogr\_no,bolum}(\text{Öğrenci})$$

2) Bilgisayar bölümüne kayıtlı öğrencilerin o ve ad bilgilerinin ilişkisel cebir ifadesi;

$$\pi_{ogr\_no,ogr\_adi}(\sigma_{bolum="Bilgisayar"}(\text{Öğrenci}))$$

**Örnek:** Ürün (urun\_kod, urun\_adi, marka, fiyat) tablosunda;  
Markası ABC olan ve fiyatı 300 ve 450 arasında olan ürünlerin ad ve fiyat bilgilerini ilişkisel cebir ifadesiyle bulalım.

$$\pi_{urun\_adi, fiyat}(\sigma_{marka="ABC" \wedge (fiyat \geq 300 \wedge fiyat \leq 450)}(\text{Ürün}))$$

### 3.Kartezyen Çarpım(Cross-Product)

- İki ilişkiden elde edilebilecek tüm çiftlerin tek bir ilişki şeklinde gösterilmesidir.
- Örneğin öğrenci ve ders tablosundan elde edilebilecek tüm çiftlerin gösterilmesi.
- Sonuç kaynak olarak kullanılan ilişkilerin sütunlarının tamamını barındıracaktır.
- Kullanımı;  $\text{Tablo}_1 \times \text{Tablo}_2$

**Örnek:**

Öğrenci(Ogr\_no,Ogr\_adi,Bolum)

Ders(Ders\_id,Ders\_adi)

Öğrenci **X** Ders

(Ogr\_no,Ogr\_adi,Bolum,Ders\_id,Ders\_adi)



## Örnek:

Ürün (urun\_kod,urunAdi,marka,fiyat)

Marka(marka,üretici,depo)

Tabloları üzerinde;

1)Ankara'da bulunan depodaki markaları ve ürünleri listeleyen ilişki;

$$\sigma_{Marka.depo="Ankara"}(\text{Ürün } X \text{ Marka})$$

2)İzmir'de bulunan depodaki markalar ve tüm ürünler için ürün adi ve üretici bilgilerini listeleyen ilişki;

$$\pi_{\text{Ürün.urunAdi,Marka.üretici}}(\sigma_{Marka.depo="İzmir"}(\text{Ürün } X \text{ Marka}))$$

## 4) Birleşim(Union)

- İki ilişkide de bulunan tüm satırları almak için kullanılır.
- Belirtilen iki ilişki de aynı sayıda sütuna sahip olmalıdır ve karşılıklı sütunlar aynı tipte olmalıdır.
- İki ilişkide de aynı satırlar varsa bunlardan sadece bir tanesi alınır.
- Tablo<sub>1</sub> **U** Tablo<sub>2</sub>

## • Örnek:

| Tablo 1 |       |        |
|---------|-------|--------|
| Sira_no | Adi   | Soyadi |
| 1       | Ahmet | Demir  |
| 2       | Büşra | Uçar   |
| 3       | Fuat  | Cebe   |

| Tablo 2 |       |        |
|---------|-------|--------|
| Sira_no | Adi   | Soyadi |
| 1       | Murat | Barut  |
| 2       | Ayşe  | Kaya   |

| Tablo 1 U Tablo2 |       |        |
|------------------|-------|--------|
| Sira_no          | Adi   | Soyadi |
| 1                | Ahmet | Demir  |
| 2                | Büşra | Uçar   |
| 3                | Fuat  | Cebe   |
| 1                | Murat | Barut  |
| 2                | Ayşe  | Kaya   |

## Örnek:

Depo1(urunAdi,fiyatı)

Depo2(urunAdi,fiyatı)

Depo1 de fiyatı 400 ve üzeri olan ürünlerle Depo2 de bulunan ve fiyatı 500 ve üzeri olan ürünlerin adlarını listelemek için;

$$\pi_{urunAdi}(\sigma_{fiyatı \geq 400}(Depo1)) \cup \pi_{urunAdi}(\sigma_{fiyatı \geq 500}(Depo2))$$

## 5)Küme Farkı(Set-Difference)

- İki ilişkiden birisinde bulunup diğerinde bulunmayan satırları bulmak için kullanılır.
- Belirtilen iki ilişkide aynı sayıda sütuna sahip olmalıdır ve karşılıklı sütunlar aynı tipte olmalıdır.
- Örneğin , iki farklı depoda tutulan ürünlerden birinci depoda bulunan ve ikinci depoda bulunmayan ürünleri bulmak için kullanılabilir.
- Kullanımı; Tablo1 – Tablo2

- **Örnek:**

| Depo 1   |        |
|----------|--------|
| Ürün_Adi | Fiyatı |
| Ürün1    | 250    |
| Ürün2    | 500    |
| Ürün3    | 300    |

| Depo 2   |        |
|----------|--------|
| Ürün_Adi | Fiyatı |
| Ürün1    | 250    |
| Ürün4    | 700    |

| Depo 1 – Depo 2 |        |
|-----------------|--------|
| Ürün_Adi        | Fiyatı |
| Ürün2           | 500    |
| Ürün3           | 300    |

## 6)Kesişim( Intersection)

- İki ilişkiden her ikisinde de bulunan satırları almak için kullanılır.
- Belirtilen iki ilişki de aynı sayıda sütuna sahip olmalıdır ve karşılıklı sütunlar aynı tipte olmalıdır.
- Örneğin ,iki farklı depoda tutulan ürünlerden iki depoda da tutulan ürünleri bulmak için kullanılır.
- Kullanımı;  $\text{Tablo1} \cap \text{Tablo2}$

- Örnek:

| Tablo 1 |       |        |
|---------|-------|--------|
| Sira_no | Adi   | Soyadi |
| 1       | Ahmet | Demir  |
| 2       | Büşra | Uçar   |
| 3       | Aydın | Cebe   |

| Tablo 2 |       |        |
|---------|-------|--------|
| Sira_no | Adi   | Soyadi |
| 1       | Ahmet | Demir  |
| 2       | Büşra | Uçar   |
| 3       | Aydın | Kara   |

| Tablo 1 $\cap$ Tablo2 |       |        |
|-----------------------|-------|--------|
| Sira_no               | Adi   | Soyadi |
| 1                     | Ahmet | Demir  |
| 2                     | Büşra | Uçar   |



## 7) Bölme(Division)

- Elde edilen iki ilişkiyi karşılaştırarak ,birinci ilişkide ikinci ilişkinin tüm elemanlarını kapsayan satırları bulmak için kullanılır.
- Örneğin; ürün ve satış bilgilerinin tutulduğu iki tablo için satış tablosundaki müşterilerden tüm ürünlerden de alanı bulmak için kullanılabilir.
- Kullanımı; Tablo1 : Tablo2

| Personel |             |     |
|----------|-------------|-----|
| Sıra_no  | Ad-soyad    | Yaş |
| 1        | Ahmet Demir | 25  |
| 2        | Büşra Uçar  | 25  |
| 3        | Aydın Cebe  | 45  |

| Yaş |
|-----|
| 20  |
| 25  |

| Personel: Yaş |             |
|---------------|-------------|
| Sıra_no       | Ad-soyad    |
| 1             | Ahmet Demir |
| 2             | Büşra Uçar  |

## 8) Birleştirme (Join)

- Ortak sütunlara sahip ilişkilerin birleştirilerek tek bir ilişkiye dönüştürülmesine denir.
- Tablolarda kullanılan anahtar sütunlar birleştirmenin temelini oluşturmaktadır.
- Örneğin, öğrenci tablosunda bulunan bölüm kodu foreign key ,bölüm tablosunda bulunan bölüm kodu da primary key olarak tanımlanan iki tablo için birleştirme yapılarak öğrencilerin bölüm isimleri bulunabilir.

# Şartlı Birleştirme(Condition Join)

- Birleştirilen ilişkiler arasındaki ortak sütunlar için koşul ifadesi belirtilir.
- Elde edilen sonuç ilişkisi kartezyen çarpıma benzer ama koşul ifadesinden dolayı kartezyen çarpımdan daha az satır içerir.
- Gösterimi;
- Tablo<sub>1</sub> |X|<sub>c</sub> Tablo<sub>2</sub>
- c koşul ifadesi için kullanılır.

**Örnek:** Aşağıda ürünlerin alım ve satış fiyatlarını içeren tablolar verilmiştir. Alım fiyatı satış fiyatından yüksek ürünleri bulmak için aşağıda verilen ifade kullanılır.

| Satış |       |       |
|-------|-------|-------|
| Sno   | Ürün  | Fiyat |
| 1     | Ürün1 | 500   |
| 2     | Ürün2 | 300   |
| 3     | Ürün3 | 700   |

| Alım |       |       |
|------|-------|-------|
| Sno  | Ürün  | Fiyat |
| 1    | Ürün1 | 350   |
| 2    | Ürün2 | 350   |
| 3    | Ürün3 | 400   |

Satış  $|X|_{alim.fiyat > satiş.fiyat}$  Alım

| Sno | Ürün  | Fiyat | Sno | Ürün  | Fiyat |
|-----|-------|-------|-----|-------|-------|
| 1   | Ürün1 | 350   | 2   | Ürün2 | 300   |
| 2   | Ürün2 | 350   | 2   | Ürün2 | 300   |
| 3   | Ürün3 | 400   | 2   | Ürün2 | 300   |

# Eşit Birleştirme(Equijoin)

- Birleştirilen ilişkiler arasındaki ortak sütunlar için eşitlik ifadesi belirtilir.
- Elde edilen sonuç ifadesi kartezyen çarpıma benzer ancak ortak sütunlardan sadece bir tanesi vardır.
- $\text{Tablo1} \bowtie_X \text{Tablo2}$  ;e eşitlik ifadesi için kullanılır.

**Örnek:** Aşağıda ürünlerin alım ve satış fiyatlarını içeren tablolar verilmiştir . Ürünlerin alım ve satış fiyatını göstermek için aşağıdaki cebirsel ifade kullanılır.

| Satış |       |       |
|-------|-------|-------|
| Sno   | Ürün  | Fiyat |
| 1     | Ürün1 | 500   |
| 2     | Ürün2 | 300   |
| 3     | Ürün3 | 700   |

| Alım |       |       |
|------|-------|-------|
| Sno  | Ürün  | Fiyat |
| 1    | Ürün1 | 350   |
| 2    | Ürün2 | 350   |
| 3    | Ürün3 | 400   |

$$\text{Satış} \mid X \mid_{\text{alım.ürün}=\text{satış.ürün}} \text{Alım}$$

| Sno | Ürün  | Fiyat | Sno | Fiyat |
|-----|-------|-------|-----|-------|
| 1   | Ürün1 | 350   | 1   | 500   |
| 2   | Ürün2 | 350   | 2   | 300   |
| 3   | Ürün3 | 400   | 3   | 700   |

# Doğal Birleştirme(Natural Join)

- Birleştirilen ilişkiler arasında herhangi bir koşul ifadesi yazmaya gerek yoktur.
- Ortak sütunlar otomatik birleştirilir ve tek bir sütun olarak gösterilir
- Doğal birleştirme sayesinde diğer birleştirmelerde olan sütun tekrarı olmayacaktır.
- Tablo<sub>1</sub> |X| Tablo<sub>2</sub>



| Ogr_no | Ogr_adi      | Bolum_kod |
|--------|--------------|-----------|
| 1      | Ali Yılmaz   | BM        |
| 2      | Ayşe Fatma   | EEM       |
| 3      | Zehra Yıldız | EM        |
| 4      | Sevil Öz     | BM        |
| 5      | Fatma Su     | ÇM        |

| Bolum_kod | Bölüm               |
|-----------|---------------------|
| BM        | Bilgisayar          |
| EEM       | Elektrik-Elektronik |
| EM        | Endüstri            |
| ÇM        | Çevre               |

## Öğrenci |X| Bölüm

| Ogr_no | Ogr_adi      | Bolum_kod | Bölüm               |
|--------|--------------|-----------|---------------------|
| 1      | Ali Yılmaz   | BM        | Bilgisayar          |
| 2      | Ayşe Fatma   | EEM       | Elektrik-Elektronik |
| 3      | Zehra Yıldız | EM        | Endüstri            |
| 4      | Sevil Öz     | BM        | Bilgisayar          |
| 5      | Fatma Su     | ÇM        | Çevre               |

# Dışsal Birleştirme(Outer Join)

- Normal birleştirme işlemlerinde ilişkili olmayan satırlar gösterilmemektedir.
- Bunda ise ilişkili olmayan satırlar da gösterilecektir.
- 3 tür outer join vardır.Bunlar; left outer join,right outer join ve full outer join

# Sol Birleştirme(Left Outer Join)

- Birleştirme sembolünün solundaki ilişki belirleyicidir ve bunun diğer ilişkiyle bir ilişkisi bulunsun veya bulunmasın tüm satırları listelenecektir.
- Sağındaki ilişkide ise sadece ilişkili satırlar listelenecektir.
- Eğer bir ilişki bulunmuyorsa NULL değer içerecektir.
- Tablo1  $\sqcup$  Tablo2

| ogrNo | ogrAdi       | Bolum_kod |
|-------|--------------|-----------|
| 1     | Ali Yılmaz   | BM        |
| 2     | Ayşe Fatma   | EEM       |
| 3     | Zehra Yıldız | EM        |
| 4     | Sevil Öz     | MM        |
| 5     | Fatma Su     | ÇM        |

| BolumKod | Bölüm               |
|----------|---------------------|
| BM       | Bilgisayar          |
| EEM      | Elektrik-Elektronik |
| EM       | Endüstri            |
| ÇM       | Çevre               |

$\pi_{ogrNo,ogrAdi,bölüm}(\text{Öğrenci} \bowtie |X|_{\text{öğrenci.bolum}_{kod} = \text{bölüm.bölümKod}} \text{Bölüm})$

| Ogr_no | Ogr_adi      | Bölüm               |
|--------|--------------|---------------------|
| 1      | Ali Yılmaz   | Bilgisayar          |
| 2      | Ayşe Fatma   | Elektrik-Elektronik |
| 3      | Zehra Yıldız | Endüstri            |
| 4      | Sevil Öz     | NULL                |
| 5      | Fatma Su     | Çevre               |

# Sağ Birleştirme(Right Outer Join)

- Birleştirme sembolünün sağındaki ilişki belirleyicidir ve bunun diğer ilişkiyle bir ilişkisi bulunsun veya bulunmasın tüm satırları listelenecektir.
- Solundaki ilişkide ise sadece ilişkili satırlar listelenecektir.
- Eğer bir ilişki bulunmuyorsa NULL değer içerecektir.
- Tablo1 |X|⋈ Tablo2

| ogrNo | ogrAdi       | BolumKod |
|-------|--------------|----------|
| 1     | Ali Yılmaz   | BM       |
| 2     | Ayşe Fatma   | EEM      |
| 3     | Zehra Yıldız | EM       |

| BolumKod | Bölüm               |
|----------|---------------------|
| BM       | Bilgisayar          |
| EEM      | Elektrik-Elektronik |
| EM       | Endüstri            |
| ÇM       | Çevre               |

$\pi_{ogrNo,ogrAdi,bölüm}(\text{Öğrenci } |X| \leftarrow \text{öğrenci.bölümKod} = \text{bölüm.bölümKod})$  Bölüm

| OgrNo | OgrAdi       | Bölüm               |
|-------|--------------|---------------------|
| 1     | Ali Yılmaz   | Bilgisayar          |
| 2     | Ayşe Fatma   | Elektrik-Elektronik |
| 3     | Zehra Yıldız | Endüstri            |
| NULL  | NULL         | Çevre               |

# Yarı Birleştirme(Semi-Join)

- Doğal birleştirmeye benzer şekilde çalışır.
- Farklı olarak sadece belirtilen ilişkilerden bir tanesinin sütun bilgilerini içerir, diğer ilişkinin sütun bilgilerini içermez.
- Tablo1 |X Tablo2

# Anti Join

- İlişkilerden birisinde bulunan sütunları gösterir ama işlev olarak tam tersidir.
- İki ilişki arasında bağ bulunmayan satırları belirlemek için kullanılır.
- `Tablo1|> Tablo2`



## 9)Yeniden Adlandırma(Renaming)

- İlişkiler veya ilişkilerin içerdiği sütunlara takma isim vermek için kullanılır.
- Birleştirme veya kartezyen çarpım sonucu elde edilen aynı isimli sütunların isimlerinin karışıklığını önlemek için değiştirilebilir veya ilişkisel cebir sonucu elde edilen tabloya isim verilebilir.
- $\rho$  ile gösterilir.
- $\rho_S(b_1, b_2, \dots, b_n)(R)$  ; R ilişkisi için S ismini ,sütunlar içinde  $b_1, b_2 \dots b_n$  isimlerini verecektir.

- Öğrenci(ogrNo,ogrAd,BolumKod) tablosu için ;
- $\rho_{(no,ad,bkod)}(\text{Öğrenci})$  ifadesi öğrenci tablosundaki ogrNo sütununu no,ogrAd sütununu ad ,BolumKod sütununu bkod olarak isimlendirmektedir.

# 10) Özetleme ve Gruplama

- Özetleme işlemi bir tablo içeriğinin istenilen bir sütuna göre hesaplamalar yapılmasını içerir.
- Örneğin, personel tablosu için toplam personel sayısı, personellerin ortalama maaşlarının hesaplanması gibi işlemler özetleme olarak ifade edilir.

- Gruplama da özetlemeye çok benzer ama aralarında fark vardır.
- Gruplama , bir tablonun içeriğini istenilen bir sütun içeriğine göre gruplandırarak her bir grup için hesaplamalar yapılmasını sağlar.
- Örneğin, personel tablosu için her bir yöneticinin sorumlu olduğu personel sayısının bulunması , A departmanında çalışan personellerin ortalama maaşları gibi işlemler de gruplandırarak özetleme olarak ifade edilir.

- Bu tür hesaplamalarda kullanılan ve çoğu veritabanı tarafından desteklenen en temel 5 fonksiyon vardır.
- Bunlar; SUM,COUNT,AVERAGE,MAX,MIN fonksiyonlarıdır.
- Kullanım şekli;
- $[Gruplandırılarak\ fonksiyon]^T[fonksiyon\ adı][sütun\ adı]$

- **Örnek:**

- Personel(Sno,İsim,Yönetici,Maaş) tablosu için;

- Toplam personel sayısının bulunması ;

$$\tau_{COUNT\ Sno}(Personel)$$

- Personellerin ortalama maaşlarının bulunması;

$$\tau_{AVERAGE\ Maaş}(Personel)$$

En yüksek maaş miktarı;  $\tau_{MAX\ Maaş}(Personel)$

En düşük maaş miktarı;  $\tau_{MIN\ Maaş}(Personel)$