VERİ TABANI YÖNETİM SİSTEMLERİ

Temel Veri Tabanı Kavramları

Veri Nedir?

Bilgi ve veri kavramı bazen birbiri ile karıştırılmaktadır. Bilgi (information) ve veri ifadeleri çeşitli açılardan açıklanabilir. Genellikle veri ile bilgi arasında farklılık olduğu ve bilgiyi elde etmeye yarayan işlenmemiş ham malzemenin veri olduğu kabul edilir. Öğrenilmek istenilen şeyleri bildikten ve veriyi kullanmaya başladıktan sonra bilgi ifadesi ortaya çıkar. Bilgi şimdi bilinen ve gelecek zamanda verilecek olan kararlar için var olan gerçek bir değerdir ve anlamlı biçimde derlenen ve birleştirilen verilerden oluşur. Bir kaynaktan, bir alıcıya iletilen mesajın içeriğidir ve bu anlamda bilgi, karar verme ile bağlantılıdır ve veriye göre daha etkin bir kavramdır.

Kısaca 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.

Veri Tabanı Nedir?

Veri tabanı (Database), birbirleriyle ilişkisi olan verilerin tutulduğu, kullanım amacına uygun olarak düzenlenmiş veriler topluluğunun mantıksal ve fiziksel olarak tanımlarının olduğu ve bunların sayısal ortamlarda saklandığı ve gerektiğinde tekrar bir erişime olanak sağlayan, büyük boyutlarda veriler barındıran bilgi depolarıdır. Veri tabanları gerçekte var olan ve birbirleriyle ilişkileri olan nesneleri ve ilişkilerini modeller. Veri tabanı; banka, üniversite, okul, seyahat şirketi, hastane vb kuruluşların çalışıp işleyebilmesi için gereken uygulama programlarının kullandığı operasyonel çok çeşitli verilerin toplamıdır. Ticari bir şirket için müşteri bilgileri, satış bilgileri, ürün bilgileri, ödeme bilgileri, vb., okul için öğrenci bilgileri, açılan dersler, okula kaydedilmiş öğrenciler, öğretmen bilgileri, sınav tarihleri, sınav sonuçları vb., hastane için hasta bilgileri, doktor bilgileri, teşhis-tedavi bilgileri, mali bilgiler vb. kullanılan çok çeşitli operasyonel verilere örnek olarak verilebilir.

Belirli bir konu hakkında toplanmış veriler bir veritabanı programı altında toplanırlar. Bu verilerden istenildiğinde; toplanan bilgilerin tümü veya istenilen özelliklere uyanları görüntülenebilir, yazdırılabilir ve hatta bu bilgilerden yeni bilgiler üretilerek bunlar çeşitli amaçla kullanılabilir.

Veri tabanı yönetim sistemi(VTYS), yeni veritabanları oluşturmak, veri tabanını düzenlemek, geliştirmek ve bakımını yapmak gibi çeşitli karmaşık işlemlerin gerçekleştirildiği birden fazla programdan oluşmuş bir yazılım sistemidir. Veri tabanı yönetim sistemi, kullanıcı ile veri tabanı arasında bir arabirim oluşturur ve veri tabanına her türlü erişimi sağlar. Veri tabanı ile ilgili bazı tanımlar aşağıda listelenmiştir. Bunlar:

Veri Tabanı Sistemi: veri tabanlarını kurmayı, oluşturmayı, tanımlamayı, işletmeyi ve kullanmayı sağlayan programlar topluluğudur. Veri tabanı yönetim sistemi(VTYS) veya Database Management System(DBMS) olarak ta bilinir. Eğer söz konusu veri tabanları yapısı ilişkisel yapıda ise, veri tabanı sistemi, ilişkisel veri tabanı sistemi olarak adlandırılır. (Relational Database Management System).

Veri tabanının tanımlanması: Veri tabanını oluşturan verilerin tip ve uzunluklarının belirlenmesidir.

Veri tabanının oluşturulması: Veri için yer belirlenmesi ve saklama ortamına verilerin yüklenmesini ifade eder.

Veri tabanı üzerinde işlem yapmak: Belirli bir veri üzerinde sorgulama yapma, meydana gelen değişiklikleri yansıtmak için veri tabanının güncellenmesi ve rapor üretilmesi gibi işlemleri temsil eder.

Verinin bakım ve sürekliliği: Veri tabanına yeni kayıt eklemek, eskileri çağırmak ve gerekli düzenleme, düzeltme ve silme işlemlerini yapmak gibi işlemlerin gerçekleştirilmesini ifade eder. Veri tabanı yönetim sistemi aynı zamanda verinin geri çağrılabilmesini de sağlar.

Veri tabanını genişletme: Kayıtlara yeni veri eklemek ve yeni kayıtlar oluşturmak gibi işlemleri ifade eder.

Bir veri tabanından beklenen özellikler, verileri koruması, onlara erişilmesini sağlaması ve başka verilerle ilişkilendirilmesi gibi temel işlemleri yapabilmesidir. Veri tabanı kullanarak verilerden daha kolay yararlanılabilir, istenilen verilere 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 görmesi sağlanabilir. Böylece veri girişinde ve veriye erişimde etkinlik ve güvenirlilik sağlanır.

Veri tabanı kullanıldığı zaman, bir kurulaşa ait tüm operasyonel veriler merkezi bir yerde ve merkezi kontrol altında tutulmuş olur.

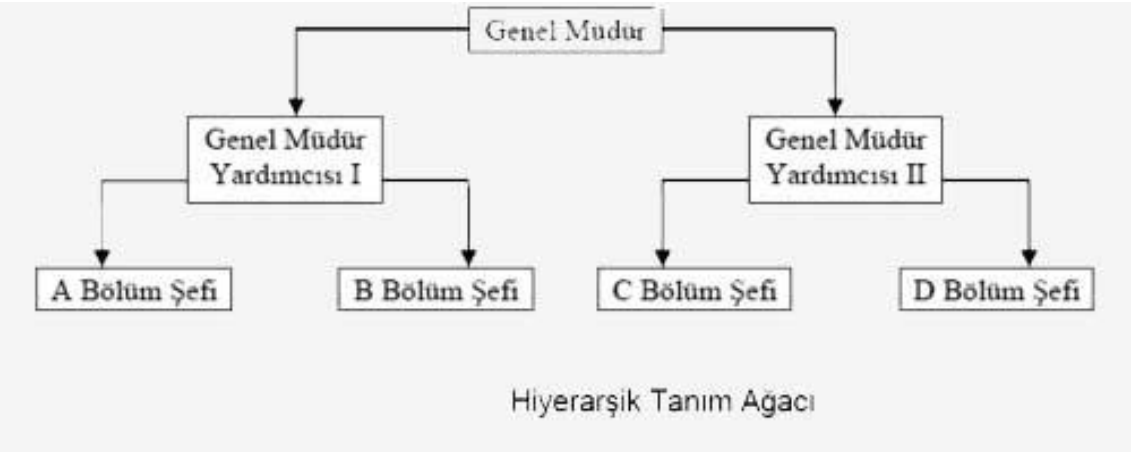
Bir veritabanı üzerinde birden fazla veri tabanı bileşeni vardır; bu bileşenler, saklanmak istenen ham bilginin belli bir formatta alınarak, veri haline gelmesi işleminde etkin rol oynarlar.

Veri tabanı yönetim sistemi programları, fiziksel hafızayı ve veri tiplerini kullanıcılar adına şekillendirip denetleyen ve kullanıcılarına standart bir SQL arayüzü sağlayarak onların dosya yapıları, veri yapısı, fiziksel hafıza gibi sorunlarla ilgilenmek yerine veri giriş-çıkışı için uygun arayüzler geliştirmelerine olanak sağlayan yazılımlardır. VTYS'de kullanıcılar, roller ve gruplar vardır ve bunlar verileri tutmak üzere bir çok türde nesne ve bu nesnelere erişimleri düzenleme görevi yaparlar. Her bir kullanıcının veri tabanı 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 veya kısıtlanabilir.

Veri tabanı yönetim sistemlerinin sınıflandırılması

Veri modeline göre sınıflandırma:

Hiyerarşik Veri Tabanları : veri tabanları için kullanılan ilk modeldir. Bu veri tabanı tipi, ana bilgisayar ortamında çalışan yazılımlar tarafından kullanılmaktadır. IBM tarafından çıkarılan IMS bu türde en çok kullanılan yazılımdır. Hiyerarşik model, bir ağaç yapısına benzer. Model dahilindeki 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. Hiyerarşik bir veri yapısı ve tanım ağacı aşağıda verildiği gibi bir veri modeli oluşturur. Aşağıdaki yapı bir şirketin yönetim kademesinin hiyerarşik yapısını içermektedir.

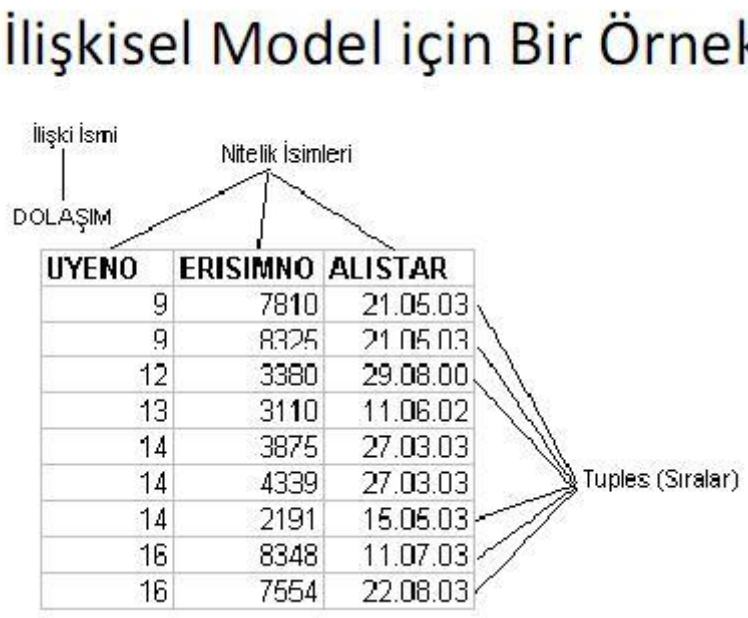


Hiyerarşik Tanım Ağacı

Hiyerarşik modelin PC ortamına uyarlanmış bir veritabanı yönetim sistemi şekli bulunmamaktadır. Bu kısımda ilişkisel veri modelini esas alan ilişkisel veritabanları önemli bir yer tutmaktadır.

İlişkisel Veri Tabanları:

1970'lerin başında E.F.Codd tarafından geliştirilmiş bir veri modeli şeklini esas alır. Bu sistemde veriler tablolar şeklinde saklanır. Bu veri tabanı yönetim sisteminde;veri alışverişi için özel işlemler kullanılır. Bu işlemlerde tablolar operandlar olarak kullanılır. Tablolar arasındaki matematiksel bağlantılarla(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 belirler.



Neden veri tabanları kullanılır:

Veri Tabanlarının Avantajları:

Bilgisayar ortamında verilerin tutulması, saklanması ve erişilmesinde günümüze kadar değişik yöntem ve yaklaşımlar kullanılmıştır. Bu yaklaşımlardan biri olan Geleneksel Yaklaşım verilerin ayrı ayrı dosyalarda gruplanma yaklaşımını kullanmaktadır. Veri tabanı programlarından önce verileri saklamak için programlama dillerinde sıralı(sequential)ve rastgele(random) dosyalama sistemi kullanılırdı. Bu sistem; birbiriyle ilgili olan ve aynı gruba

dahil olan verilerin bir dosyada, bir başka gruba dahil olan verilerinde bir başka dosyada tutulması yöntemine dayanmaktadır. Verilerin artması, verilere aynı anda erişilmesi ve aynı anda erişilen verilerin erişenlere göre düzenlenmesi gibi ihtiyaçlar arttıkça geleneksel yaklaşım yetersiz kalmıştır. Geleneksek Yaklaşımın aşağıda listelenmiş bu ve buna benzer bir çok yetersizlik ve beraberinde getirdiği sorunlara alternatif olarak Veri Tabanı Yaklaşımı zamanla Geleneksel Yaklaşımın yerini almıştır. Günümüzde veriler Veri Tabanı Yaklaşımı ilkesine göre tutulmakta ve işlenmektedir.

Geleneksel Yaklaşımın(Dosya-İşletim sistemi) Sakıncaları

- Veri tekrarı ve veri tutarsızlığı
- Verinin paylaşılamaması
- Uygulamalardaki her yeni gereksinim ve değişikliğin yalnız uzman kişiler tarafından karşılanabilmesi
- Veriye erişim ve istenen veriyi elde etme güçlükleri
- Karmaşık veri saklama yapıları ve erişim yöntemlerini bilme zorunluluğu
- Bütünlük(integrity) sorunları
- Güvenlik, gizlilik sorunları
- Tasarım farklılıkları ve standart eksiklikleri
- Yedekleme, yeniden başlatma, onarma gibi işletim sorunları.

Veri Tabanı Yaklaşımının Yararları

- Ortak verilerin tekrarının önlenmesi; verilerin merkezi denetiminin ve tutarlılığının sağlanması
- Veri paylaşımının sağlanması
- Fiziksel yapı ve erişim yöntemleri yaklaşımlarının, çok katmanlı mimarilerle kullanıcılardan gizlenmesi
- Her kullanıcıya yalnız ilgilendiği verilerin, alışık olduğu kolay, anlaşılır yapılarda sunulması
- Sunulan çözümleme, tasarım ve geliştirme araçları ile uygulama yazılımı geliştirmenin kolaylaşması
- Veri bütünlüğü için gerekli olanakların sağlanması, mekanizmaların kurulması
- Güvenlik ve gizliliğin istenilen düzeyde sağlanması
- Yedekleme, yeniden başlatma, onarma gibi işletim sorunlarına çözüm getirilmesi.

- Veriler tek bir merkezde tutulur ve aynı veri her kullanılan değişik bilgisayarlarda tekrar tekrar tutulmaz. Programcı, kullandığı verilerin yapısı, organizasyonu ve yönetimi ile ilgilenmeden veri tabanının bunları kendinin koordine etmesi ve yönetmesidir. Veri bağımsızlığı, veri tabanı yönetim sistemi programlarının en temel amaç ve özelliklerindendir.

Bilinen VTYS programları:

MS SQL Server: Microsoft firması tarafından geliştirilen, 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 adlı dil geliştirilmiştir.

Sybase: 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 (open source) bir yazılımdır. Küçük-orta ölçeklidir.

Postrage SQL: Bu da 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.(1 milyon civarı) yada belli bir boyutun (yaklaşık 25 MB) altına kadar bir sorun çıkarmadan kullanılabilecek bir küçük ölçekli VTYS'dir.

VERİ MODELİ

Bir veri modeli, verinin hangi kurallara göre yapılandırıldığını belirler.

Varlık İlişki Veri Modeli(Entity-Relationship model):

Vİ yada ER modeli olarak isimlendirilen bu model 1976 yılında geliştirilmiştir.

Bugüne kadar varlık-ilişki modeline dayalı bir VTYS geliştirilmemiştir. Buna rağmen burada varlık ilişki modelinin verilmesinin temel sebebi, VTYS'den bağımsız veri çözümlemesinde ve semantik veri modellemede en çok kullanılan model olmasıdır.

Bu model kullanılarak önce;

- VTYS'den bağımsız olarak veriler çözümlenir,
- Veri modellemesi yapılır,
- Veriler ve veriler arasındaki ilişkilerin anlamları ve özellikleri incelenerek E-R çizelgeleri oluşturulur,
- Kullanılacak VTYS belirlenir ve daha sonra E-R çizelgeleri bu sistemin veri modellerine dönüştürülerek veri tabanı şemaları oluşturulur.

Varlık ve Varlık Kümesi

Varlık:

Var olan ve benzerlerinden ayırt edilebilen her nesneye **varlık (entity)** adı verilir. Varlık bağımsızdır ve tek başına tanımlanabilir. Bir veri tabanı uygulaması hakkında tanımlayıcı bilgi saklanabilen her şey varlık olarak kabul edilir. Bir varlık, ev, araba gibi bir nesne yada futbol maçı, tatil, satış gibi olaylar vb olabilir.

Örnek: Bir öğrenci, bir kitap, Veri Tabanı Yönetim Sistemleri Dersi, Burak birer varlıktır.

Varlık Kümesi:

Aynı türden benzer varlıkların oluşturduğu kümeye ise **varlık kümesi (entity set)** adı verilmektedir. Varlık kümeleri iç içe, kesişen yada ayrık kümeler olabilir.

Örnek: Öğrenciler, kız öğrenciler, Bilgisayar Programlama Bölümü Öğrencileri, yurtta kalan öğrenciler, renkler, dersler, yıllar, tarihler, satış miktarlar vb… varlık kümesi örnekleri olarak sayılabilir.

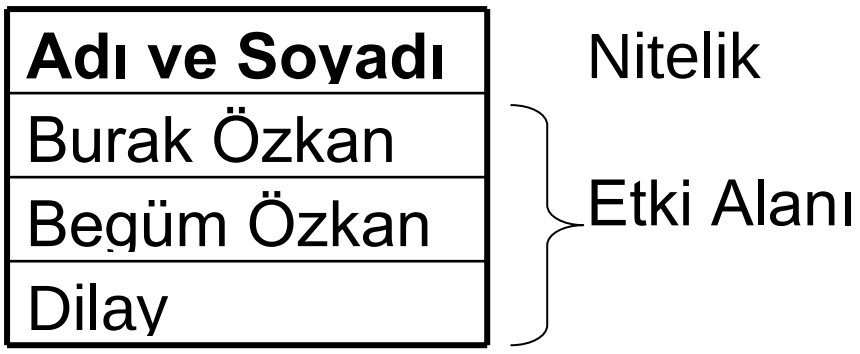
Nitelik:

Bir varlık kümesinde varlıkların özelliklerini göstermek ve varlıkları birbirinden ayırt etmek için **nitelikler(attributes)** kullanılır. Gerçek dünyada varlıkların çok sayıda özellikleri olabilir. Ancak veri modellemede, bu özelliklerin uygulamalarda gerekli olan küçük bir kısmı alınarak soyut bir modelleme yapılır.(Sicil no, adı, soyadı, adres vb..)

Etki Alanı:

Her niteliğin bir etki alanı (domain) vardır.Etki alanı ilgili niteliğin olabilecek tüm değerlerini içeren bir kümedir.

Örnek:



Türetilen Nitelik:

Bir nitelik kullanılarak bir başka varlık niteliği elde edilebiliyorsa, bu yeni niteliğe **türetilen nitelik** adı verilir.

Örneğin, "personel" varlığının "doğum tarihi" niteliğinden yararlanılarak "yaş" niteliği elde edilebilir. Bu örnekte "yaş" niteliği türetilen niteliktir, tasarımda ayrıca tanımlanmasına gerek yoktur.

Birleşik Nitelik:

Birden fazla nitelik birleştirilerek yeni bir nitelik oluşturulabilir. Bu tür niteliklere birleşik nitelik adı verilir.

Örneğin, "mahalle", "cadde", "sokak", "apartman", "posta kodu" ve "şehir" gibi nitelikler birleştirilerek "adres" isimli yeni bir nitelik oluşturulabilir.

İlişki:

Varlıklar arasındaki bağıntıya **ilişki (relationship)** adı verilir.

- İkili ilişki:
  - bir öğrenci ile bir ders (öğrenci dersi **alır**)
  - bir firma ile bir malzeme (firma malzemeyi **üretir**)
- Üçlü ilişki:
  - Bir işçi, bir ürün ve bir makine (işçi bu ürünü üretirken bu makineyi kullandığı için)

İlişki Kümesi:

Aynı türden ilişkilerin oluşturduğu kümeye ilişki kümesi adı verilir.

- Matematiksel olarak E1, E2, E3, ..., En varlık kümeleri arasındaki bir R ilişkisi aşağıdaki gibi tanımlanır:

$R = \{(e1, e2, e3, ..., en) : e1 \in E1, e2 \in E2, e3 \in E3, ..., en \in En\}$

Örnek:

Aşağıdaki iki varlık kümesini göz önüne alalım:

E1 = { Ali, Ayşe }

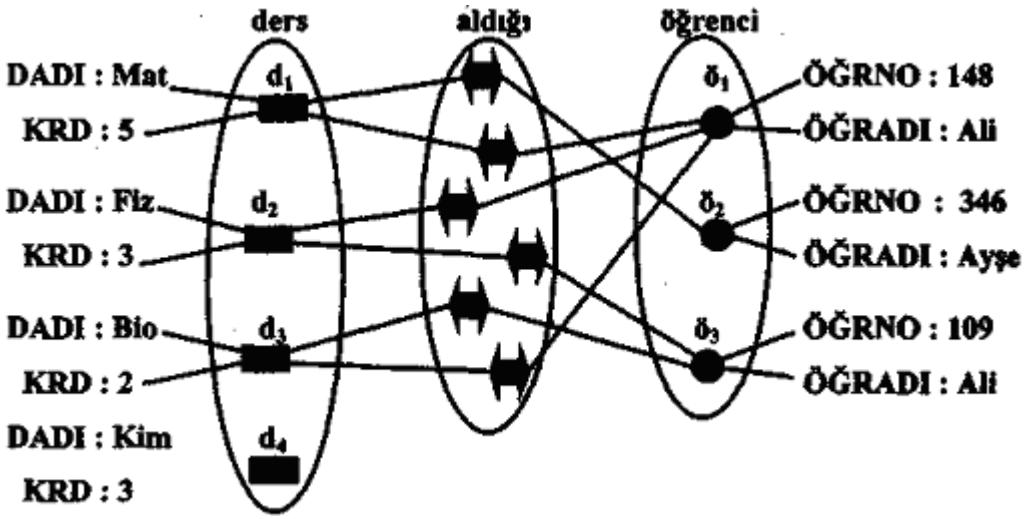
E2 = { Matematik, Fizik }

Bu varlık kümeleri için öğrenci ve aldığı ders ilişkileri aşağıdaki gibi ifade edilebilir:

R1 = { (Ali, Matematik) } R2 = { (Ali, Fizik) }

R3 = { (Ayşe, Matematik) } R4= { (Ayşe, Fizik) }

Aşağıdaki şekilde "öğrenci" ve "ders" varlık kümeleri arasında "aldığı" ilişkisi ile bir ilişki örneği görülmektedir:



Rol:

Aralarında ilişki kurulan varlıklardan her birinin ilişkideki işlevine varlığın rolü denir.

Farklı varlık kümeleri arasındaki ilişkilerde roller dolaylı yoldan anlaşılabildiği için çoğunlukla açıkça belirtilmez

- Örneğin, öğrenci ve ders arasında kurulan "aldığı" ilişkisinde varlıkların rolleri bellidir: öğrenci dersi alan, ders ise öğrenci tarafından alınandır.

Örnek:

İlişkilerdeki roller belirlenirken, başka niteliklere de bakmak gerekebilir.

- Örneğin "Personel" varlığında "ast-üst" ilişkisini belirlemek için "Görevi" niteliği dışında "Bölümü" niteliğine de bakmak gerekebilir.

Yönetici (üst, ast) ilişkileri:
(Begüm, Burak)
(Begüm, Dilay)
(Selin, Sezin)

Personel varlığı:

Adı	Bölümü	Görevi
Burak	Muhasebe	İsci
Begüm	Muhasebe	Yönetici
Dilay	Muhasebe	İsci
Selin	Satış	Yönetici
Sezin	Satış	İsci

- **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, ÖĞRENCİ ve OKUL arasında bir NOT ilişkisi vardır.

Veri yapısı oluşturulurken, verideki nesneler ve onlar arasındaki ilişkiler tablo ile temsil edilir. Veri tabanında uygulanabilecek genel kayıt ilişkilendirme tipleri şu şekilde sıralanabilir:

Bire bir ilişkiler (One-to-one relationships): Aralarında ilişki olan iki tablo arasında, tablolardan birindeki asıl anahtar alanın kayıt değerinin, diğer tablodaki sadece bir kayıtta karşılığının olması durumunu gösteren ilişki tipidir.

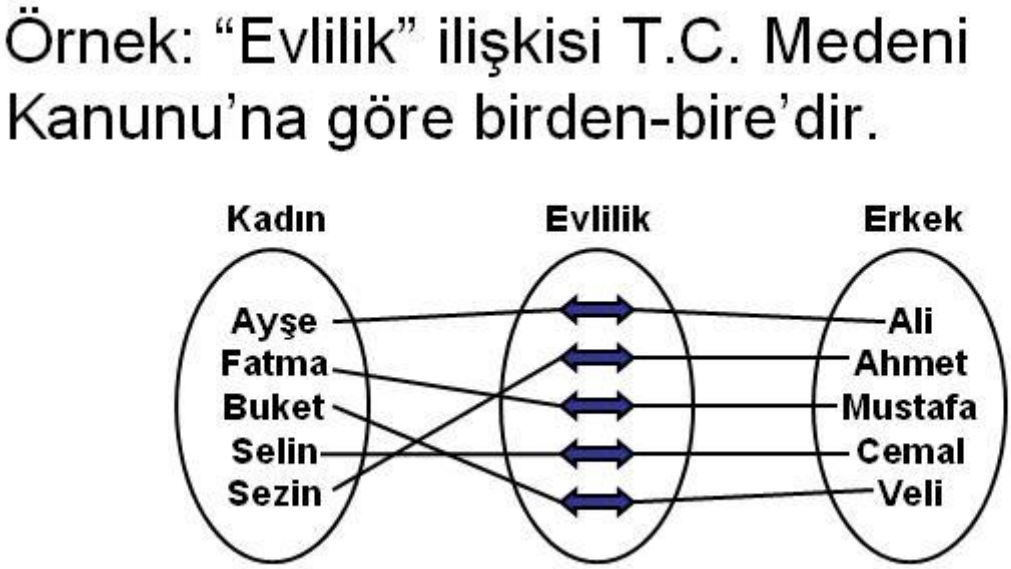
Örnek: Bir öğrencinin doğum yeri bilgisinin doğum yerleri tablosunda bir şehre karşılık gelmesi gibi.

Aşağıdaki şekilde bir bire bir ilişki örneği görülmektedir.

1 Çalışanlar Tablosunda her Futbolcuyla eşleşen BİR kayıt vardır.

2 Bu değer kümesi, ÇalışanNo alanının ve Çalışanlar Tablosunun bir alt kümesidir.

Bir başka bire bir ilişki örneği evlilik ilişkisi üzerinden verilebilir. Aşağıdaki şekilde kümeler ve ilişki yönü ile birebir bir ilişki olarak gösterilmiştir.

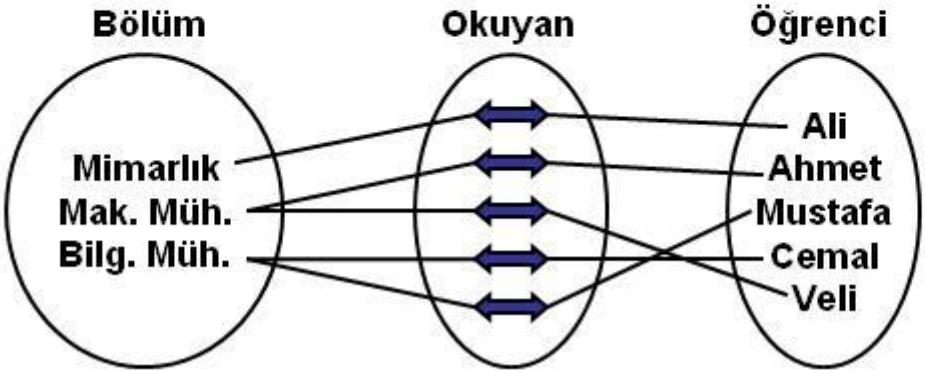


Tekil çoklu ilişkiler (One-to-many relationships): Aralarında bir ilişki bulunan iki tablo arasında, anahtar alanının kayıt değerinin, diğer tablodaki birden fazla kayıtta karşılığının olması durumunu gösteren ilişki tipidir.

Örnek: Bir öğrencinin birden fazla almış olduğu derse ve bu derse ait vize, final ve sınav sonuçları gibi. Bir öğrenciye karşılık birden fazla ders notu. Aşağıda birden çoğa bir ilişki türü olarak kitaplar ile yazar arasındaki ilişki örneği verilmiştir. Bir yazar birden çok kitap yazmış olabilir. Ancak her kitap mutlaka bir yazar tarafından yazılmıştır.

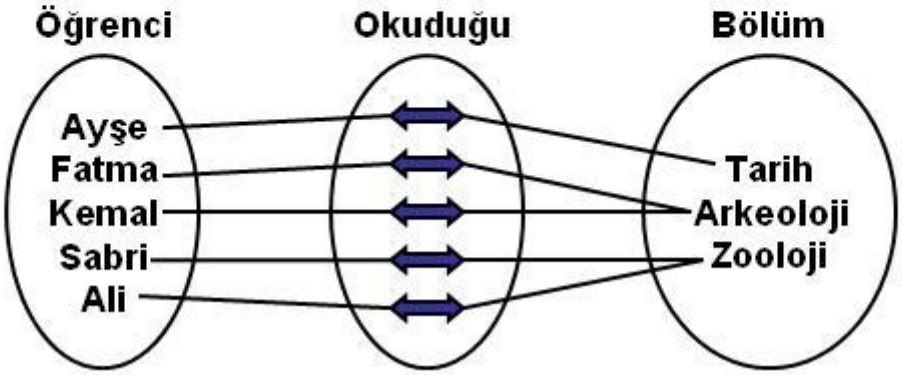
Birden çoğa bir başka ilişki örneği aşağıdaki şekilde gösterilmiştir. Bölümle öğrenci arasında bulunan ”okuyan” ilişkisi örneği:

Ornek: “Bölüm” ve “Oğrenci” varlık kümeleri arasındaki “Okuyan” ilişkisi, bölümden öğrenciye doğru birden-çoğa şeklindedir.



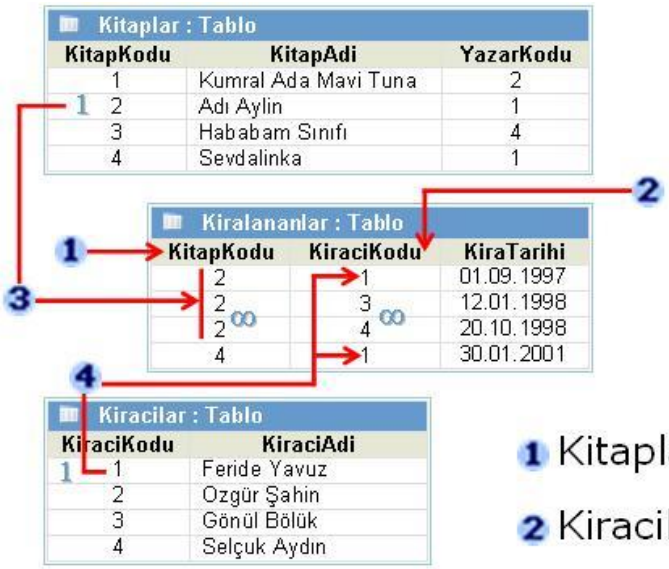
Çoğul tekli ilişkiler (Many-to-one relationships): Aralarında bir ilişki olan iki tablo arasında, tablolardan birindeki bir kaydın değerinin, asıl anahtar alanının olduğu diğer tabloda, birden fazla kayıtta karşılığının bulunması durumunu gösteren ilişki tipidir. Çoktan bire bir ilişki şekli aşağıda verilmiştir. Bölümle öğrenci arasında “okuduğu” ilişkisi ile belirtilen bir ilişki şu şekilde olmaktadır:

Örnek: “Öğrenci” ve “Bölüm” varlık kümeleri arasındaki “Okuduğu” ilişkisi, öğrenciden bölüme doğru çoktan-bire şeklindedir.



Çoklu ilişkiler(Many-to-many relationships): Aralarında bir ilişki bulunan iki tablo arasında, tablolardan herhangi birindeki herhangi bir kaydın, diğer tablodaki birden fazla kayıtla ilişkilendirilebildiği ilişki tipidir.

Aşağıdaki şekilde çoklu ilişkilerin örneği görülmektedir.



1 Kitaplar Tablosundaki Birincil Anahtar

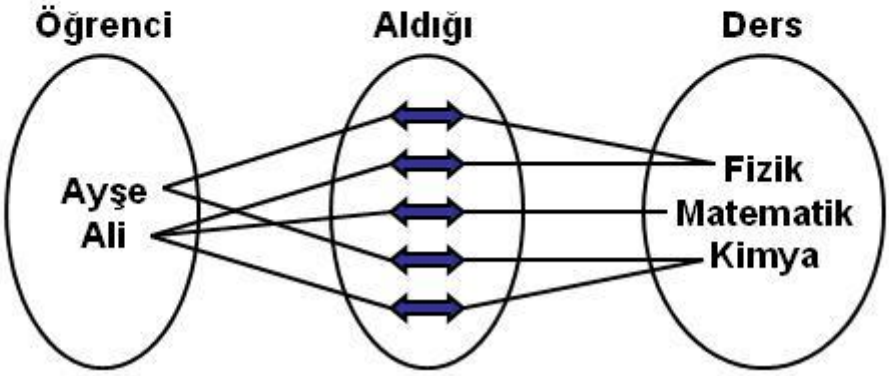
2 Kiracilar Tablosundaki Birincil Anahtar

3 Bir Kitap Bir-Çok kiralamada yer alabilir ...

4 ... her Kiracı Bir-Çok Kiralamada görülebilir.

Aşağıdaki resimde bir çoktan çoğa ilişki örneği öğrenci ve ders varlıkları arasında verilmiştir. Birden fazla öğrenci birden fazla ders alabilmektedir.

Örnek: "Öğrenci" ve "Ders" varlık kümeleri arasındaki "Aldığı" ilişkisi, çoktan-çoğa şeklinde bir ilişkidir.



- **Kısıtlar:**

Veriler üzerindeki sınırlamalara kısıt(kısıtlama) adı verilir. Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılırlar. Örneğin "öğrenciye ait not bilgisinin 0-100 arasında olması " gibi.

- **İşlemler:**

İşlemler bir veri tabanı durumundan bir başka veri tabanı durumuna geçmek için kullanılan işlemlerdir. Bunlar verinin çağrılması, güncellenmesi, eklenmesi veya silinmesi ile ilgili işlemlerdir. Bütünlük mekanizması, toplam fonksiyonları (istatistiksel fonksiyonlarda bunlar arasındadır), veriye ulaşım kontrolleri gibi genel işlemler vardır. CODASYL tarafından yayınlanan bu mekanizmalara veri tabanı yöntemleri denir.

Ödev: 1-CODASYL nedir araştırınız.

2-Verilen ilişki türlerine(birden bire, birden çoğa, çoktan bire ve çoktan çoğa ilişkileri) **en az** birer örnek geliştiriniz. Örneklerinizi varlık kümelerini ve ilişki adını belirterek, derste verildiği gibi şekille gösteriniz.

3- Veri tabanı kısıtlamalarına yeni örnekler üretiniz. Hangi alanları nasıl bir kısıtlamaya tabi tutabilirsiniz. Örnekler üzerinden açıklayınız.

Anahtarlar:

Bir varlık kümesi içindeki varlıkları ya da bir ilişki kümesi içindeki ilişkileri birbirinden ayırt etmek için kullanılan nitelik ya da nitelik grubuna bu varlık ya da ilişki kümesinin anahtarı denir.

Anahtar, hem varlık kümeleri hem de ilişki kümeleri için geçerli bir kavram olsa da, daha çok varlık kümeleri için kullanılır.

Anahtar Türleri:

- **Süper anahtar (superkey)**
  - Değerleri ile bir kümedeki varlıkları (veya ilişkileri) ayırt etmeyi sağlayan niteliğe (veya nitelik grubuna) bu varlık / ilişki kümesinin süper anahtarı denir. Ayırt etme özelliğine sahip olmak için gereğinden fazla nitelik içerebilir.
- **Aday anahtar (candidate key)**
  - Eğer bir varlık / ilişki kümesinin süper anahtarının bir altkümesi de bu varlık / ilişki kümesini ayırt edebiliyorsa, bu altküme aday anahtardır (ya da kısaca anahtardır).

Örnek:

Eğer bir üniversitede tüm öğrencilerin numaraları birbirinden farklı ise, öğrencileri ayırt etmek için öğrenci numarası yeterlidir.

Bu durumda "öğrenci numarası", "öğrenci" varlık kümesi için aday anahtar ya da kısaca anahtardır, içinde öğrenci numarası bulunan her nitelik grubu (örneğin "öğrenci numarası", "adı" ve "soyadı") ise bu varlık kümesinin süper anahtarıdır.

Güçlü ve Zayıf Varlık Kümeleri:

Her varlık kümesi için bir anahtar bulmak mümkün olmayabilir.

Eğer bir varlık kümesinin niteliklerinden en az bir anahtar oluşturulabiliyorsa, bu varlık kümesine **güçlü (strong**) varlık kümesi denir.

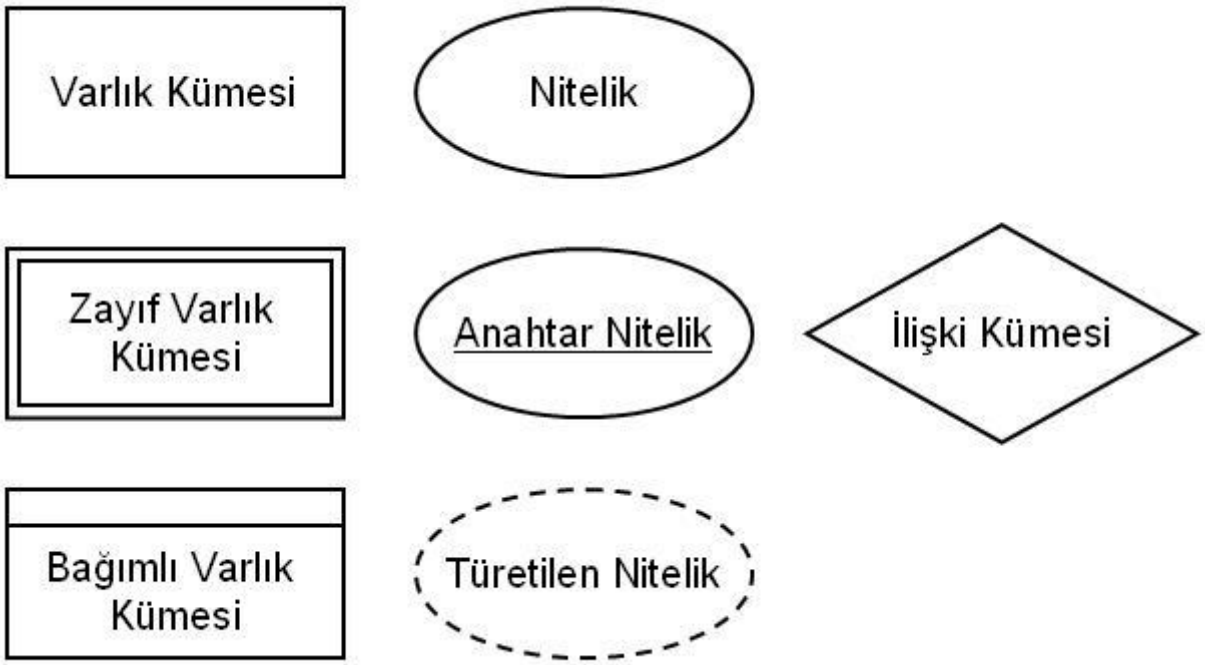
Eğer bir varlık kümesinin niteliklerinin tümü alınsa bile bir anahtar oluşturmuyorsa bu varlık kümesine **zayıf (weak)** varlık kümesi denir

Zayıf Varlığı Güçlendirme:

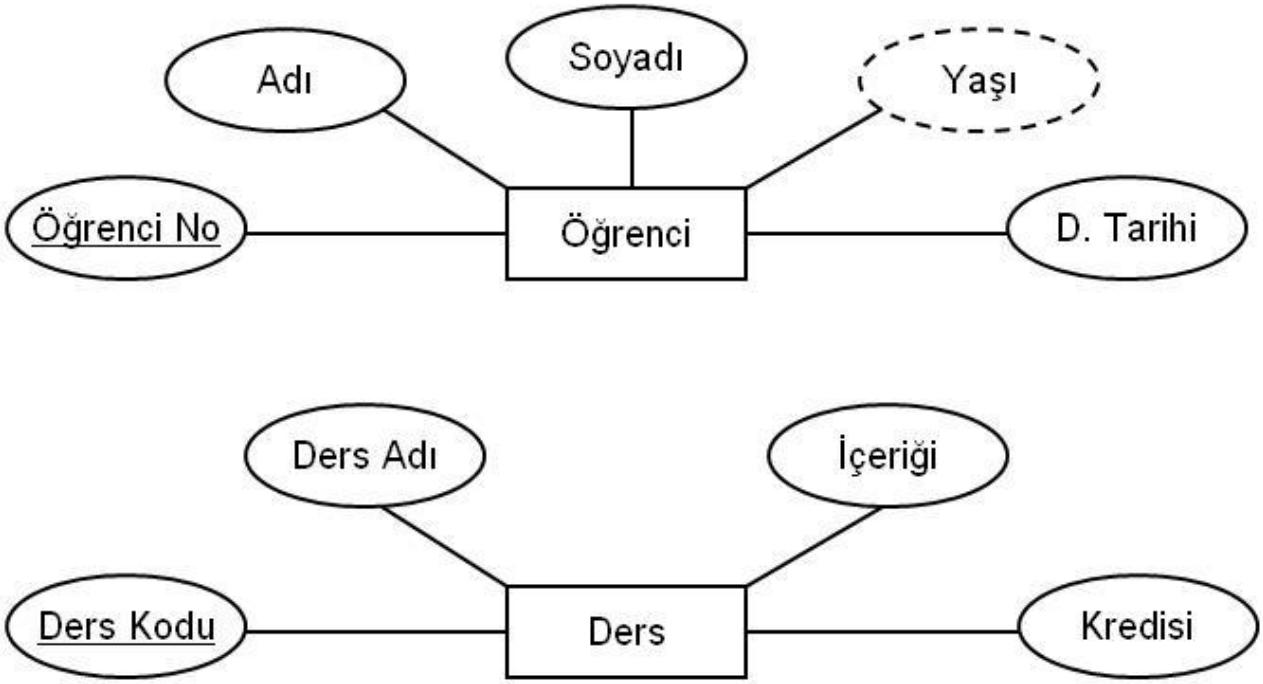
Türkiye'deki tüm lise öğrencilerinin bilgilerini içeren ÖĞRENCİ varlık kümesi zayıf bir varlık kümesidir. Çünkü farklı liselerde öğrenci numarası, adı ve soyadı aynı olan öğrenciler bulunabilir.

ÖĞRENCİ varlık kümesi ile LİSE varlık kümesi arasında bir OKUYAN ilişkisi kurulursa, öğrencileri birbirinden ayırdetmek için kullanılan ÖĞR_NO niteliğine, LİSE varlık kümesinin anahtarı olan LİSE_KODU eklenir. Bu durumda ÖĞRENCİ varlık kümesinin anahtarı (LİSE_KODU, ÖĞR_NO) ikilisi olur.

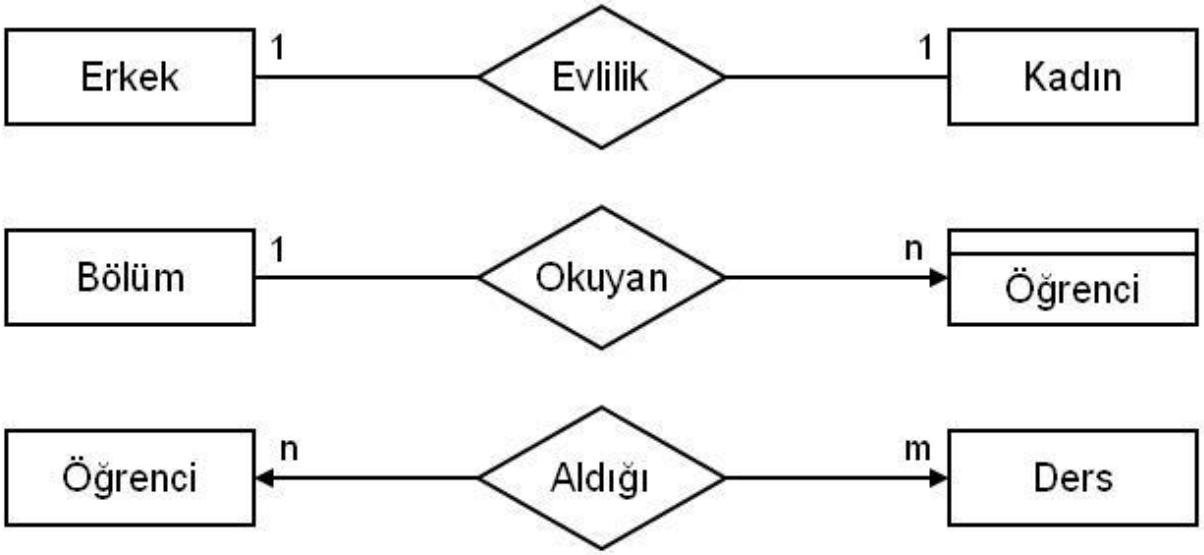
Varlık İlişki Çizelgeleri:



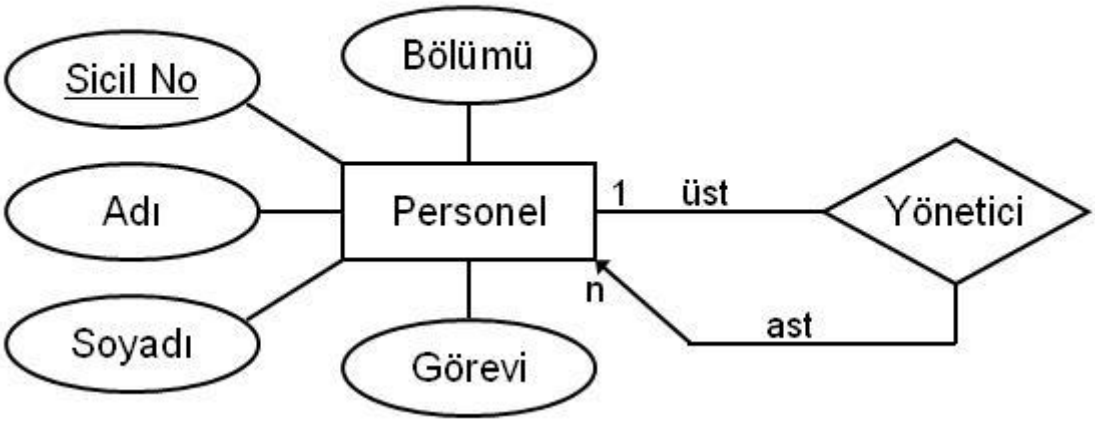
Varlık Kümesi ve Nitelik Örnekleri:



İlişki Kümesi Örnekleri:

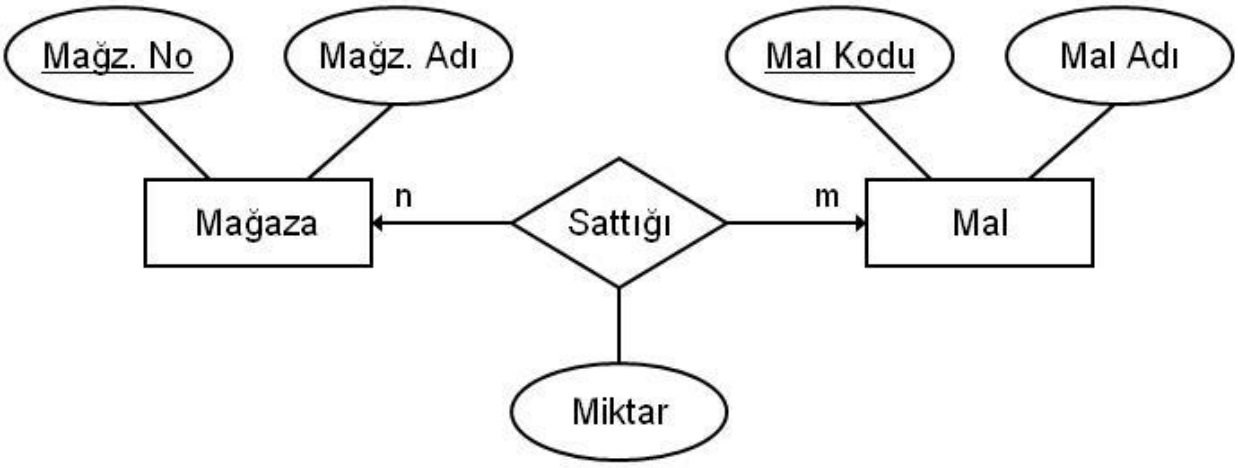


Rol Örnekleri:



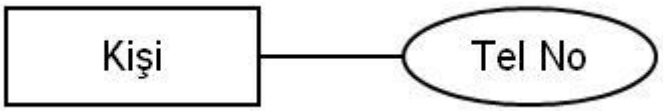
İlişkilerde Nitelik:

- İlişkilerde de tanımlayıcı nitelikler bulunabilir.
- Aşağıdaki şekilde "Miktar" niteliği "Sattığı" ilişkisi için tanımlayıcı niteliktir.



Örnek Veri Modelleri:
Örnek: Telefon Numarası:

- "telefon numarası" **kişi** varlık kümesinin bir niteliği olarak düşünülürse;
  - "telefon numarası" kişilerden bağımsız olarak var olamaz.
  - bir kişinin sadece bir telefon numarası bulunabilir.
  - birden çok kişinin telefon numarası aynı olabilir (telefon numarası kişi varlık kümesinin anahtarlarından biri olarak tanımlanmadığı sürece).

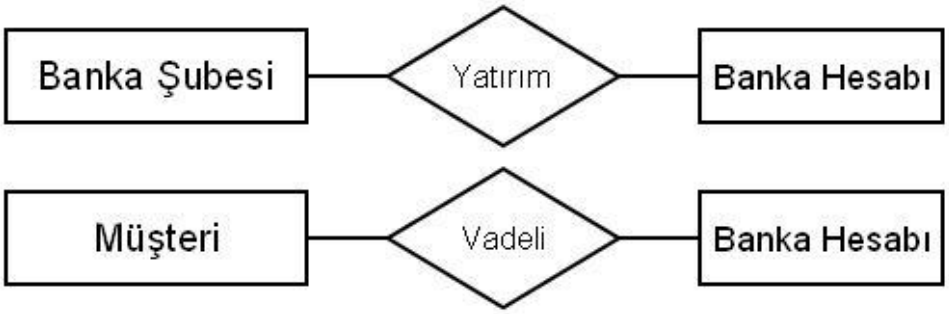


- "telefon numarası" ayrı bir varlık kümesi olarak düşünülüp, bu varlık kümesi ile kişi varlık kümesi arasında ilişki kurulursa;
  - telefonun numarası dışında nitelikleri de bulunabilir.
  - kişi ve telefon varlık kümeleri arasındaki ilişkinin türüne göre her kişinin bir ya da birçok telefonu olabilir.
  - bir telefon numarası bir ya da birçok kişiye verilebilir.

Örnek : Banka Hesabı

- "Banka hesabı" banka şubesi ve müşteri varlık kümeleri arasında bir ilişki olarak düşünülebilir.

- "Banka hesabı" ayrı bir varlık kümesi olarak düşünülüp bu varlık kümesi ile banka şubesi ve müşteri varlık kümeleri arasında birer ilişki de kurulabilir.

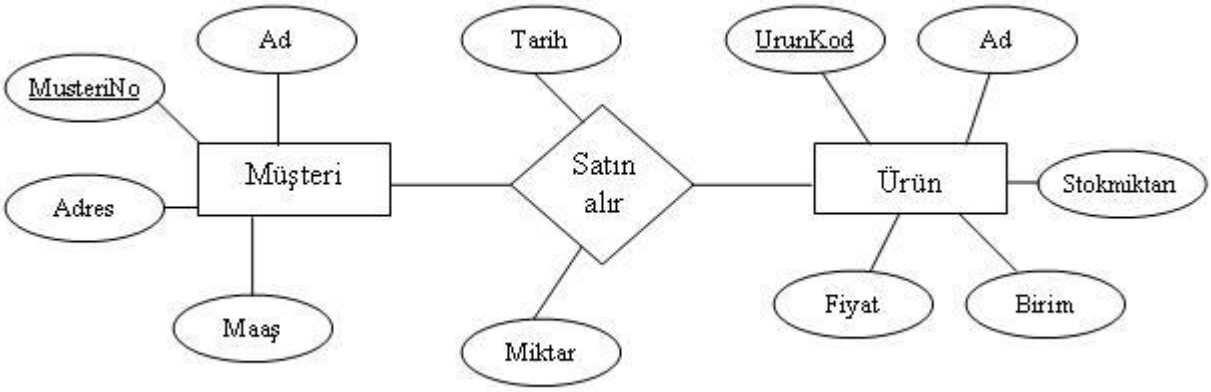


Veri Modellemede Seçim:

- Veri modellemede varlık kümelerinin, niteliklerin ve ilişki kümelerinin seçimi çok önemlidir, Ancak bunların nasıl seçileceğine ilişkin kesin kurallar da yoktur.
- Kuruluşun öncelikleri ve uygulamaların özellikleri yanında veri modellemeyi gerçekleştiren bilişim teknik personelinin anlayışı da düzenlemede etkili olmaktadır.

Ödev:

4-Aşağıdaki diyagramı derste anlatılanlara göre detaylı bir şekilde açıklayınız.



5- Varlık ilişki diyagramlarından tablolamaya geçiş hakkında bilgi edininiz.

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. Aşağıdaki örneklerde büyük harflerle yazılan ifadeler ilişki isimlerini, parantez içindeki ifadeler de tanım kümesini göstermektedir.

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 veri tabanının ilişkisel şemasını göstermektedir. İlişkisel şema, ilişki isimlerinin ve karşılık gelen tanım kümesi isimlerinin listesidir. Varlık tiplerini belirlemekte kullanılır.

Universite

UniversiteKodu	UniversiteAdi	Adres
2300	Fırat Üniversitesi	Elazığ
0600	ODTU	Ankara
3400	Fatih Üniversitesi	İstanbul

Şekil:İlişkisel Tablo

Veri Tipleri(Data Types)

MS Access İçin Veri Tipleri:

Veri Türü	Anlamı
Text (Metin)	Alfabetik yada alfasayısal verileri tanımlar. En fazla 255 karakterlik veriler içerebilir. Bu veriler üzerinde matematiksel işlemler yapılamaz.
Memo (Büyük metin)	Tablo içindeki uzun açıklamalar veya notların tutulması için kullanılır.Bu alanlara 32000 karakter uzunluğundaki metinler kaydedilebilir.
Number (Sayı)	Sayısal alanların tanımlanması amacıyla kullanılır. Bu tür verileri içeren alanlar üzerinde matematiksel işlemler yapılabilir.
Date/Time (Tarih/Saat)	Tarih veya saat bilgilerini içeren alanların tanımlanması amacıyla kullanılır.Bu alanlar baytlık yer işgal ederler.
Currency	Özellikle büyük rakamsal değerlerin kullanıldığı alanlardır.
AutoNumber (Otomatik Sayı)	Bu veri türüne sahip alanlar, tabloya yeni bir kayıt eklendiğinde Access tarafından otomatik olarak üretilen sıralı yada raslantısal bir sayısal değer içerirler. Dolayısıyla bu alanda belirtilen sayısal değerler tektir ve ayrı kayıtlarda birbirinin aynısı olamaz. Bu tür alanlar kullanıcı tarafından güncelleştirilemez.
Yes/No	Bir baytlık bir uzunluğa sahip olan bu alanlar Yes/No biçimindeki verilerin saklanması amacıyla kullanılır.
Ole Object	Eğer tablonun bir alanında resim, ses veya grafik gibi OLE nesnelerinin saklanması söz konusu ise böyle bir tanım yapılır. Bu alanın büyüklüğü 1 GB kadar olabilir.
Hyperlink	Bu veri türü ile herhangi bir Web sitesini bu alandaki veri olarak tanımlayabiliriz. WWW'de bulunan bilgilerle direkt bağlantı sağlanabilir.
Lookup	Bir başka tablo ile bağlantı kurularak açılan listeden bilgi

Wizard (AramaSihirbazı)	seçilmesini ve tanımlanan alana taşınmasmını sağlar.

MYSQL İçin Veri Tipleri:

TEXT - YAZI
CHAR( ) : Sabit 0 - 255 karakter.
VARCHAR( ) : Değişken 0 - 255 karakter.
TINYTEXT : En Fazla 255 karakter.
TEXT : En Fazla 65.535 karakter.
BLOB : En Fazla 65.535 karakter.
MEDIUMTEXT : En Fazla 16.777.215 karakter.
MEDIUMBLOB : En Fazla 16.777.215 karakter.
LONGTEXT : En Fazla 4.294.967.295 karakter.
LONGBLOB : En Fazla 4.294.967.295 karakter.

SAYILAR
TINYINT( ) : -128 ,127 yada 0-255 UNSIGNED.
SMALLINT( ) : -32.768 ,32.767 yada 0 – 65.535 UNSIGNED.
MEDIUMINT( ) : -8.388.608 , 8.388.607 yada 0 – 16.777.215 UNSIGNED.
INT( ) : -2.147.483.648 , 2.147.483.647 yada 0 – 4.294.967.295 UNSIGNED.
BIGINT( ) : -9.223.372.036.854.775.808 , 9.223.372.036.854.775.807 yada 0 – 18.446.744.073.709.551.615 UNSIGNED.
FLOAT : Küçük Noktalı sayı.
DOUBLE(,) : Büyük Noktalı sayı.
DECIMAL(,) : DOUBLE tipte string şeklinde saklanır.

TARİH , SAAT
DATE : YYYY-MM-DD
DATETIME : YYYY-MM-DD HH:MM:SS
TIMESTAMP : YYYYMMDDHHMMSS
TIME : HH:MM:SS

DİĞER
ENUM () : Kullanıcı tanımlı liste tipi. Ör; ENUM('e','h')
SET : Küme Tipi. ENUM benzeri. Aynı anda birden fazla kayıt tutabilir.

ORACLE VERİ TİPLERİ :

CHAR(karakterSayisi) : Maximum 255 karakterlik sabit uzunluktaki alfanümerik verilerin tutulabileceği alandır.
DATE : tarih ve saat tutan alandır. Ülke kodu desteği vardır. Standart olan veri tipi DD-MON-YY (31-JUL-05)dir.
MSLABEL : Trusted Oracle?da kullanılan işletim sistemine ait binary dosyadır.
NUMBER (toplam,ondalık) : Sayısal verilerin tutulduğu alanlar için kullanılır. İlk hane toplam karakter sayısını(ondalık dahil), ondalık bölümü ise ondalık kısmın uzunluğunu belirtir.
NUMBER(hane) : Ondalık içermeyen tam sayılar için kullanılan veri tipidir.
NUMBER : herhangi bir sayı girilmeden belirtilen sayısal alan tipidir. Tavsiye edilmemekle birlikte, oracle tarafından desteklenen maximum sayısal değere kadar veri girilebilir.
VARCHAR2 (sayı): Maximum 4000 karakterli değişken uzunluktaki alfanümerik dataların tutulabildiği alanlar için kullanılır.
LONG : 2 GB? a kadar karakter bilgi tutabilen bir alan türüdür. Bir tabloda birden fazla long veri tipine sahip alan olamaz. Bu alan üzerinde indeks oluşturulamaz. (*WHERE *GROUP BY * ORDER BY * DISTINC * CREATE CLUSTER *CREATE TABLE AS SELECT *SUBSTR, INSTR gibi SQL cümlelerinde kullanılamaz.)
LONG RAW : 2 GB? a kadar binary bilgi tutabilen alanlar için kullanılır.
RAW (sayı) : Maksimum 255 byte?a kadar bilgi tutabilen binary alanlar için kullnılır.
ROWID : Oracle´ın, tablodaki her bir satır için oluşturduğu sıra numarasıdır. Oracle tarafından otomatik oluşturulur.

MS SQL SERVER Veri Tipleri:

SQL Server 2005 için veri tipleri aşağıdaki gibidir.

1. **Exact Numeric**: Sayıları eğer varsa ondalık kısımlarıyla birlikte depolayan veri tipleri
2. **Approximate Numeric** : Exact Numeric tipinin tutamadığı uzunluktaki ondalık sayıları tutan veri tipleri
3. **Monetary** : Parasal birimleri tutmak için kullanılan veri tipleri
4. **Date and Time** : Tarihsel değerlerin tutulduğu veri tipleri
5. **Cahracter** : Uzunluğuna göre karakterleri depolayan veri tipleri (kelimeler… gibi)
6. **Binary** : Binary tipli verileri tutmak için kullanılan veri tipleri (resim, mp3…gibi)
7. **Special Purpose** : Özelleştirilmesi gereken karmaşık veri tiplerini tutmak için tasarlanmış veri tipleridirler.

1. Exact Numeric Data Tipleri

Bu veri tipleriyle tam sayılar veya ondalık sayılar tutlabilirler. Bu veri tipleriyle matematik işlemleride yapılabilir. Kapladıkları alana göre ve kapasitelerine gore 6 ya ayrılırlar:

Bigint = 8 bayt yer kaplar. (int: Integer(tamsayının kısaltması) Adından belli olduğu üzere tam sayıları tutar. Tutabileceği aralıkda -2^{63} ile $2^{63}-1$ arasıdır. (Pek kullanılmaz)

Int= 4 bayt yer kaplar. Yine belirli aralıktaki tam sayıları tutarlar. Aralıkda -2^{31} ile $2^{31}-1$ dır. En çok kullanılan veri tiplerinden biridir. Bizim için sadece bir tane olması gereken değerlere (ID lere) verilebilir.

Smallint=2 bayt yer kaplar. -2^{15} ile $2^{15} - 1$ arasındaki değerleri tutar. (-32768 ile 32767 arası)

Tinyint=1 bayt yer kaplar. 0 'dan 255 'e kadar olan sayıları tutarlar.

Decimal(p,s)=Decimal ondalık veya tam butun sayıları tutar ama ondalık sayı için kullanılır. Neden? Çünkü kapladığı alan bakımından. Decimal içine parametre alır. Yani şöyle ki mesela decimal(4,2) diye bie değişken tanımladım, bu demek oluyorki önce 4 basamak yazıyorum ondan sonra sağdan 2 basamak sayıp virgülü koyuyorum.

Örnek: decimal(5,3) önce rastgele beş basamak sayı yazıyorum "12345" , şimdide 3 basamak ayırıyorum "12,345". Yani decimal(5,3) sayısı 2 basamaklıdır ve virgülden sonra 3 basamak bulundurur.

Numeric(p,s) = İşlev ve özellik bakımından decimal ile aynıdır.(Pek kullanılmaz)

2. Aproximate Numeric Data Tipleri

Bu veri tiplerinde (zaten 2 tane var) amaç decimal in tutamadığı büyüklükteki değerleri tutmaktır.

Float(n) = -1,79E + 308 'den 1,79E + 308 'e kadar olan değerleri alabilir. Kapladığı alan içine aldığı "n" değerine göre değişir. (1<= n <= 53 olmalı)

Real = eski versiyonlarda kullanıldığı için bu versiyonda da kullanılmış. Bu da -3,40 + 38 'den 3,40 + 38 'ekadar olan küsürlü verileri tutar. 4 bayt yer tutar.

3. Monetary Data Tipleri

Söyleyecek fazla bişey sanırım, adları üstünde; para yerine kullanacağımız değişkenler

Money= -922 337 203 685 47,.5808 'den 922 337 203 685 477,5807 'ye kadar paraları tutabilir. 8 bayt yer kaplar.

Smallmoney= 214 748,3648 'den 214 748,3647 'kadar olan sayıları tutabilir. 4 bayt yer kaplar.

4. Date and Time Data tipleri

Tarih ve zamanı tutmak için kullanılan veri tipleri

Datetime= 1 Ocak 1753 'ten 31 Aralık 9999 'a kadar olan tarih aralığıı tutabilir. 8 bayt yer kaplar.

Smalldatetime= 1 Ocak 1900 'den 6 Haziran 2079 'a kadar olan tarih aralığını tutar. 4 bayt yer kaplar.

5. Character Data Tipleri

Karakter data tipleri en çok kullanılan data tipleridir. Kelimelerin yerine daha doğrusu harflerin yerine kullanılır. Parametre alırlar. Aldıkları parametreler içerdikleri karakter sayısıdır. İsimlerinde genelde "char " kelimesi vardır. Bunun başına "n" veya " var" önekleri getirilerek diğer data tipleri oluşturulmuştur. Kısaca bunlar ne demek bi bakalım.

Önce "n" varsa veri tipinin başında bu unicode karakterleride içeriyor demektir. Yani her dilde görüntülenebilir. Başında " n " olanlar olmayanlarına göre 2 kat daha fazla yer tutarlar (açıklayınca daha rahat görebilirsiniz).

Başında "var" olanlar ise içine aldıkları parametreye kadar genişletilebilirler. Örneğin varchar(5) direk olarak 5 baytlık yer ayırmaz, içine 3 harf yazarsak 3 bayt, 2 harf yazarsak 2 bayt yer ayırır. Bu neden boyle diyecek olursanız veri tipi varchar(5) değilde char(5) olsaydı ben içine tek bi karakter bile yazsam direk 5 baytlık alnı işgal edecekti. Şimdi diceksinizki o zaman ben hepsini varchar la yaparım. Ne kadar girersem o kadar yer kaplasın. Ama bu sizin yapacağınız sorgularda performans kaybetmenize sebep olabilir. Veri tiplerine geçelim..

Char(n) = Enfazla 8000 karakter alabilir. Dolayısıyla maksimum 8000 bayt yer kaplar. Bu veri tipi içine girilecek olan karakter sayısı kesin belli olduğu zaman kullanılır. Örnek: Northwind tablosundaki CustomerID ler (hepsi 5 karakterli)

Nchar(n) = Enfazla 4000 karakter alır. 2 ile 8000 bayt arası yer tutar. (unicode karakterleri yüzünden)

Varchar(n) = Enfazla 8000 karakter içerir. 8000 bayta kadar yer tutabilir. Bu veri tipini girceğimiz değerin uzunluğu kesin belli olmadığında kullanabiliriz. Örnek: şehirlerin isimleri…

Nvarchar(n) = Enfazla 4000 karakter alır. 2 ile 8000 bayt arası yer tutar. (unicode karakterleri yüzünden)

Varchar(max) = 2Gb 'a kadar karakter alabilir. 1 073 741 824 karakter alabilir.

Nvarchar(max) = 2Gb 'a kadar karakter alabilir. 536 870 912 karakter alabilir. (unicode karakterleri yüzünden)

Text = varchar(max) ın aynısı.

Ntext = nvarchar(max) ın aynısı. Bu text ve ntext çok fazla kullanılmıyor. Onların yerine varchar, char … kullanılıyor.

6. Binary Data Tipleri

Bu data tiplerini anlatmanın bence en klay yolu bunları depoya benzetmektir. İçine tutabileceği kadar ne koysanız tutar.Örneğin mp3, resim, word belgesi…gibi. Bunlarda parametre alıyor ve yukardaki "var" olayı bunlarda da var. (alanı direk ayırmayıp, biz değer girdikçe artan olay) Zaten 4 tane var bunlardan.

Binary(n) = 8000 bayta kadar veri tutabilir.

Varbinary(n) = 8000 bayta kadar veri tutabilir.

Varbinary(max) = 2Gb 'a kadar veri tutabilir.

Image = 2Gb 'a kadar veri tutabilir.(image=varbinary(max))

7. Specialized Data Tipleri

Hepsinin adını yazayım ama 3 tanesine değineceğim sadece. Bit, timestamp, uniqueidentifier, sql_variant, cursor, table, Xml.

Bit= 0 veya 1 tam sayı değeri alan değişkenlerdir. Yani geriye true veya false bi değer döndürür. Örneğin cinsiyette, evli-bekar, evet-hayır, var-yok gibi şekillerde kullanılabilir.

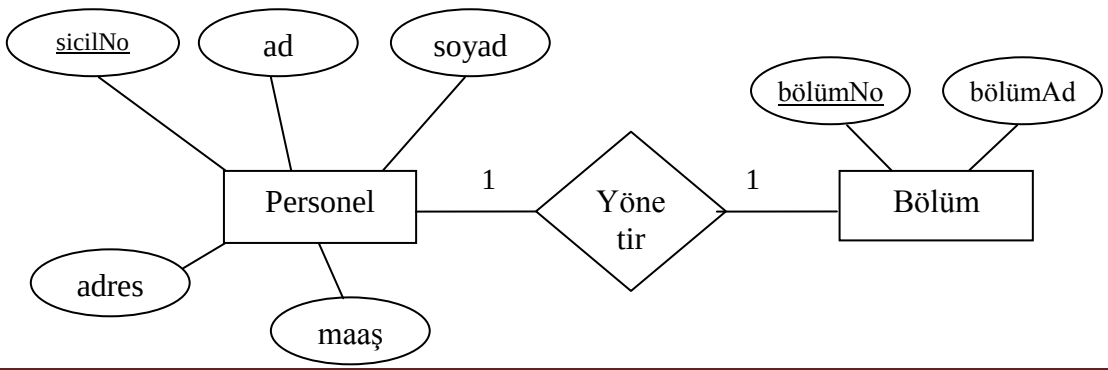
Sql_variant= Bu gireceğimiz değerin çeşidini bilmediğimiz sütunlarda kullanabileceğimiz bir veri tipidir. Yani resim, string, sayı, table ne olduğunu bilmediğim şeyler yerine kullanabilirim. Tabiki 8000 bayta kadar.

Uniqueidentifier= 16 bayt yer kaplar. Global tek değişkenlerdir (GUID, Globally Unique Identifier).

ER şemasının tablolara dönüştürülmesi

Bire-bir ilişkilerin dönüştürülmesi

Birebir ilişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. Nitelikleri tabloların alanlarına dönüşür. Uygun olan varlık kümesinin anahtar alanı diğer varlık kümesine yabancı anahtar olarak eklenir. Bire-bir ilişkide belirtilen tanımlayıcı nitelikler, yabancı anahtar eklenen tabloya alan olarak eklenir. Örneğin bölüm ve personel varlık kümeleri arasındaki yönetir birebir ilişkisinde iki şekilde dönüştürme yapılır. Birincisi, her iki varlık kümesi bölüm ve personel tablolarına dönüştürülür. Bölüm tablosuna personel tablosunun anahtar alanı olan sicilno alanı yabancı anahtar olarak eklenir. İkincisi, benzer şekilde her iki varlık kümesi bölüm ve öğrenci tablolarına dönüştürülür. Öğrenci tablosuna bölüm tablosunun anahtar alanı olan bölümno alanı yabancı anahtar olarak eklenir.



Şekil: Personel-Bölüm yönetir bire-bir ilişkisi ER şeması

ER şemasının tablolara dönüştürme

1. Yol

Bölüm (bölümNo, bölümAd, sicilNo)

Personel (sicilNo, ad, soyad, adres, maaş)

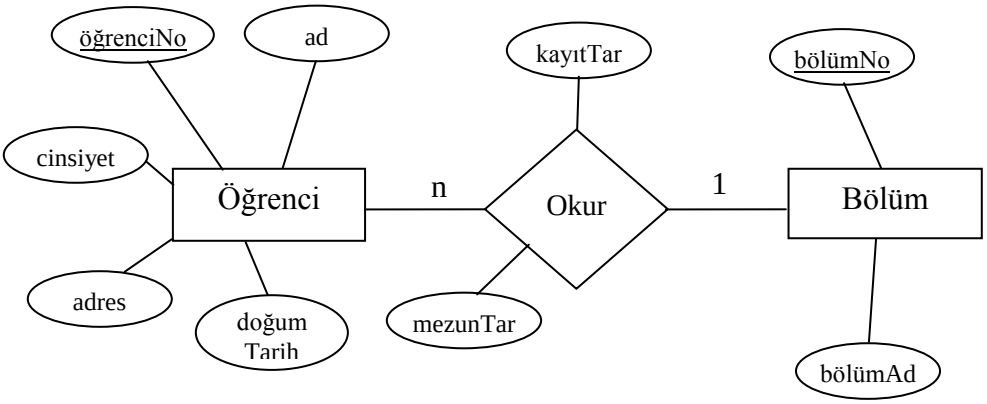
2. Yol

Bölüm (bölümNo, bölümAd)

Personel (sicilNo, ad, soyad, adres, maaş, bölümNo)

Bire-birçok(1-n) ilişkilerin tablolara dönüştürülmesi

İlişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. İlişkinin n tarafındaki tabloya 1 tarafındaki tablonun anahtar alanı yabancı anahtar olarak eklenir. İlişkide belirtilen tanımlayıcı nitelikler n tarafına alan olarak eklenirler. Örneğin öğrenci bölüm varlık kümeleri arasındaki 1-n okur ilişkisinde öğrenci ve bölüm varlık kümeleri tablolara dönüştürülür . İlişkinin n tarafındaki tablo olan öğrenci tablosuna bölüm tablosunun anahtar alanı olan bölümNo alanı yabancı anahtar olarak eklenir. İlişkiyi tanımlayan kayıtTar ve mezunTar nitelikleri de öğrenci tablosuna alan olarak eklenirler.



Şekil: Öğrenci ve Bölüm 1-n okur ilişkisi ER şeması

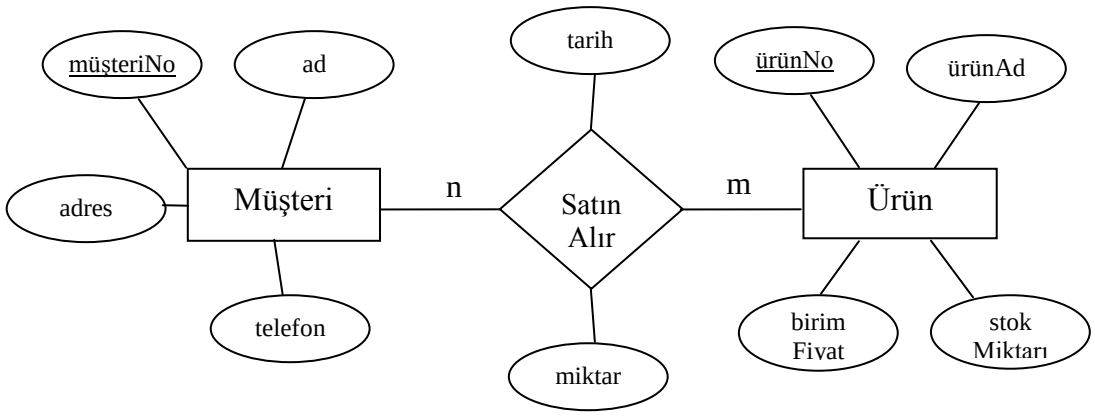
Yukarıdaki ER şeması tablolara dönüştürülürse;

Öğrenci(öğrenciNo, ad, cinsiyet, adres,doğumTarih, bölümNo, kayıtTar, mezunTar)

Bölüm(bölümNo, bölümAd)

Birçoğa-birçok(n-m) ilişkilerin tablolara dönüştürülmesi

İlişkiyi oluşturan varlık kümeleri tablolara dönüştürülür. Ancak ilişki isminde yeni bir tablo oluşturulur. İlişkiyi oluşturan tabloların anahtar alanları yeni tabloya yabancı anahtar olarak eklenir. İlişkide belirtilen tanımlayıcı nitelikler varsa yeni tabloya alan olarak eklenir. Yeni tablonun anahtar alanı ilişkiyi oluşturan tabloların yabancı anahtarlarından oluşan ikili veya daha fazla alandan oluşur. Ancak bazı durumlarda bu anahtar tanımı yeterli olmaz. Yeni tabloya tabloya uygun şekilde yeni bir anahtar alan eklenir. Örneğin aşağıdaki şekildeki müşteri-ürün n-m ilişkisi tabloya dönüştürülürken, müşteri ve ürün adında iki tablo oluşturulur. Ayrıca satınalır isminde yeni bir tablo oluşturulur. Satınalır tablosun müşteri tablosunun anahtar alanı olan müşteriNo ve ürün tablosunun anahtar alanı olan ürünNo yabancı anahtar olarak eklenirler. Ayrıca satın alır ilişkisinde tanımlanan tarih ve miktar nitelikleri de satınalır tablosuna alan olarak eklenirler. Satınalır tablosunun anahtar alanı olarakta satınalırNo isminde yeni bir alan eklenir.



Şekil: Müşteri ürün n-m satın alır ilişkisi ER şeması

Yukarıdaki ER şeması tablolara dönüştürülürse;

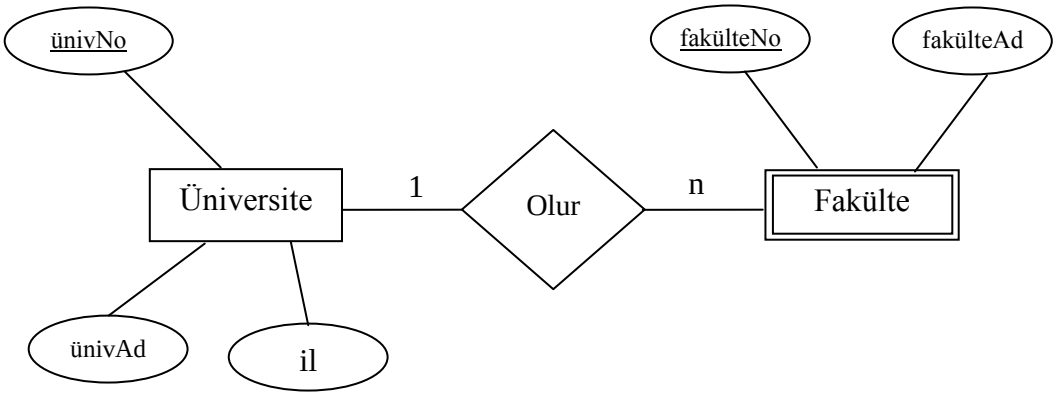
Müşteri (müşteriNo, ad, adres, telefon)

Ürün (ürünNo, ürünAd, birimFiyat, stokMiktarı)

SatınAlır (satınAlırNo, müşteriNo, ürünNo, tarih, miktar)

Zayıf varlık kümelerinin tablolara dönüştürülmesi

Zayıf varlık kümeleri tablolara dönüştürülür. Ancak zayıf varlık kümesinin anahtar alanı kendi anahtar alanı ile birlikte ilişki kurduğu güçlü varlık kümesinin yabancı anahtarının birleşimidir. Örneğin, aşağıdaki şekildeki üniversite-fakülte 1-n ilişkisinde, üniversite varlık kümesi güçlü, fakülte varlık kümesi ise zayıf varlık kümesidir. Üniversite ve fakülte varlık kümeleri, ayrı ayrı üniversite ve fakülte varlık kümelerine dönüştürülür. Ancak, fakülte varlık kümesinin anahtar alanı, fakülteNo ile birlikte üniversiteNo alanlarından oluşturulur.



Şekil : Üniversite-Fakülte olur 1-n güçlü-zayıf ilişkisi

Yukarıdaki ER şeması tablolara dönüştürülürse;

Üniversite (ünivNo, ünivad,il)

Fakülte (fakülteNo, ünivNo, fakülteAd)

ÖDEV:

Muhtarlık, Eczane, Mağaza, Hastane, Okul, Personel, Stok Takip vb konularda geliştirilecek bir otomasyon için ER diyagramlarının detaylı bir şekilde oluşturulması(Varlıkların ve niteliklerinin ve varlık kümeleri arasında ilişkilerin detaylıca tanımlanması vb…), ER diyagramlarından veritabanı tablolarının oluşturulması(Tabloların oluşturulması, niteliklerinin belirlenmesi, tabloların anahtarlarının oluşturulması, tablolar arasındaki ilişkilerin tespiti)…(Teslim Tarihi:24 nisan 2009 Cuma)

SQL
(Structed Query Language)

1. SQL nedir?

İlişkisel Veritabanı Yönetim Sistemleri (Relational Database Management Systems- RDBMS) modeli ilk önce 1970 yılında Dr. E.F. Codd tarafından tarif edilmiştir. SQL veya Structured English Query Language (SEQUEL), IBM firması tarafından Codd'un modelini kullanmak için geliştirilmiştir. SEQUEL daha sonra SQL olmuştur. 1979 yılında, Relational Software (şu an Oracle Corporation), SQL' in ilk ticari uygulamasını geliştirmiştir. Bugün SQL, ilişkisel veritabanı yönetim sistemleri standardı olarak kabul edilmektedir.

SQL (Structured Query Language) kendisi bir programlama dili olmamasına rağmen bir çok kişi tarafından programlama dili olarak bilinir. SQL herhangi bir veri tabanı ortamında kullanılan bir alt dildir (sub language). SQL ile yalnızca veri tabanı üzerinde işlem yapabiliriz. SQL cümlecikleri kullanarak veri tabanına kayıt ekleyebilir, olan kayıtları değiştirebilir silebilir ve bu kayıtlardan listeler oluşturabiliriz.

Veritabanı içindeki verileri elde etmek ve değiştirmekle ilgili SQL deyimleridir.

- SELECT → Verileri istediğimiz şekilde seçebilmemizi sağlar.
- INSERT → Tablo içine yeni kayıt eklememizi sağlar.
- UPDATE → Tablodaki bilgileri değiştirmemizi sağlar
- DELETE → Kayıt yada kayıtları silmemizi sağlar.

SQL'in en önemli ve en çok kullanılan deyimi SELECT deyimidir.

2. Verilere Erişmek (SELECT)

Veritabanındaki verilere erişmenin, diğer bir deyişle onları görmenin en sık kullanılan yöntemidir. Veritabanımızdaki hangi tablo yada tablolardaki alanları (bilgileri) görmek istiyorsak SQL cümlemizin başında mutlaka SELECT deyimi yer almalıdır.

Bu komut ile veritabanı üzerindeki tablonun hangi kolonları alacağımız veritabanına söyleriz. Tablonun bütün kolonlarını görmek istiyorsak '*' karakterini kullanırız. Sadece belli kolonları görmek istiyorsak kolon isimlerini aralarına virgül koyarak yan yana yazmalıyız

SELECT deyiminin Kullanım şekli şöyledir:

SELECT Alan_Adı FROM Tablo_adı;

Aşağıdaki örnekleri açıklayacak olursak birinci örnek personel tablosundaki bütün kayıtları getirecektir.

Örnek 1 :

```
SELECT * FROM personel;
```

Örnek 2 :

```
SELECT * FROM meslekler ;
```

Bütün meslek bilgilerini almak istersek yukarıdaki gibi bir query (sorgu cümlesi) yazmalıyız.

Örnek 3 :

```
SELECT ad, soyad FROM personel ;
```

Bu query ise bize firmamızda çalışan bütün personeli listeler.

FROM

Bu komut bütün SQL cümleciklerinde bulunması gerekli bir komuttur. Bu komut ile hangi tablolar üzerinde çalışacağımızı veri tabanına söylüyoruz. Eğer aynı sql cümleciği ile bir kaç tablo üzerinde işlem yapmak istersek tablo isimleri arasına virgül koymalıyız.

SELECT * FROM personel;
Personel tablosundaki bütün kayıtları seç(göster).

WHERE

Gerçek hayat'ta bu kayıtların sadece bir kısmına ihtiyaç duyarız. Bize gerekli olan dataları diğerlerinden ayıran bazı özellikleri vardır. İşte bu özellikleri Where komutu yardımı ile kullanarak gerekli datalara ulaşabiliriz. Where komutu ile select sorgu cümleciğimize şart ifadesi eklemiş oluyoruz. Şu özelliğe sahip olanları getir şeklinde.

Örnek 4 :

Adı Ahmet olan personeli listelemek istersek ne yapacağız. Aşağıdaki gibi bir sorgulama yapacağız.

```
SELECT * FROM personel WHERE ad='Ahmet';
```

Örnek 5 :

Yaşı 40'dan büyük personeli listeleme istersek;

```
SELECT * FROM personel WHERE Dogum_tarihi < '13.04.1959'
```

sorgulamasını kullanmalıyız. Elemanın 40 yaşında büyük olması için 1959 yılından önce doğmuş olması gerekmektedir. O halde dogum_tarihi 1959 yılından küçük olmalıdır.

Örnek 6 :

Adana'da doğmuş personeli listelemek istersek

```
SELECT * FROM personel WHERE Dogum_yeri = 'Adana'
```

Örnek 7 :

300 milyondan fazla maaş alan kişileri işe maaşa göre sıralamak istersek;

```
SELECT * FROM ucretler WHERE  aylik_ucret >= 300000000 ORDER BY
aylik_ucret;
```

BETWEEN (ARASINDA)

Aralıklı sorgulama yapmak istersek kullanabileceğimiz bir operatördür.

örnek 1 : Öğrenci numarası 240 ile 400 arasında olan öğrenciler kimlerdir?

```
SELECT *
FROM öğrenciler
WHERE öğrenci_no> =240 AND öğrenci_no < = 400;
```

BETWEEN komutu ile daha kısa olacaktır.

```
SELECT *
FROM öğrenciler
WHERE öğrenci_no BETWEEN 240 AND 400;
```

Örnek 2 :

```
SELECT * FROM ucretler WHERE aylik_ucret BETWEEN 200000000 AND
300000000
```

Bu cümlecik ile 200 ile 300 milyon arasında maaş alanlar listelenecektir.

ORDER BY (SIRALA)

Bu komut ile belirtilen kolona göre artan veya azalan bir sıralama ile sorgulama yapabiliriz.

ASC : kullanarak küçükten büyüğe doğru artan sıralama yapabiliriz.

DESC : kullanarak büyükten küçüğe doğru azalan sıralama yapabiliriz.

Ancak ASC kullanmak zorunlu değildir. Çünkü default sıralama tipi ASC'dir. Aynı anda birkaç kolon üzerinden de sıralama yapabiliriz.

Örnek 1 : Öğrenciler tablosundan ,öğrenci_no,adı,soyadı sütunlarını seç ve öğrenci numarasına göre büyükten küçüğe sırala.

```
SELECT öğrenci_no, adı, soyadı
FROM öğrenciler
ORDER BY öğrenci_no DESC;
```

ASC Küçükten büyüğe sıralar, DESC Büyükten küçüğe sıralar.

Bir tablo içinde , birden fazla sütundan aynı anda sıralamak için ;

örnek 2 : Öğrenciler tablosundan seçili sütunları öncelik adda olmak üzere büyükten küçüğe,soyadı büyükten küçüğe ve öğrenci numarasını küçükten büyüğe sıralayalım.

```
SELECT öğrenci_no,adı,soyadı
FROM öğrenciler
ORDER BY adı DESC, soyadı DESC, öğrenci_no ASC;
```

veya;

```
SELECT öğrenci_no,adı,soyadı
FROM öğrenciler
ORDER BY adı, soyadı DESC,öğrenci_no;
```

DESC ' li durumda yanına yazıp belirtilir, yazılmazsa ASC direk kabul edilir.

Örnek 3 :

```
SELECT * FROM personel ORDER BY ad ASC;
```

Bu query ile personel tablosundaki bütün kayıtları ad'a göre küçükten büyüğe doğru sıralarız.

Örnek 4 :

```
SELECT * FROM personel ORDER BY soyad DESC;
```

Bu query da yukarıdakinin tersine kayıtları büyükten küçüğe doğru sıralar.

Örnek 5 :

```
SELECT * FROM personel ORDER BY ad, soyad;
```

Bu query kayıtları ad göre artan bir sıralama yapar. Ancak aynı ad ile yaratılmış birden fazla kayıt varsa ise bunları da soyad sırasına göre artan bir şekilde sıralar. Eğer her iki kolonda aynı ise o zaman okuduğu sırada sıralar.

Örnek 6 :

```
SELECT * FROM personel ORDER BY dogum_tarihi DESC,ad,soyad ;
```

Bu query'de ise personel kayıtları büyükten küçüğe doğru sıralanıyor. Yani en genç eleman'dan başlanarak en yaşlı elemana doğru bir liste yapılıyor. Doğum tarihleri aynı olanlarda ise ad ve soyad'a göre bir sıralama yapılmaktadır.

LIKE

İçinde belli bir karakter dizisi bulunan datalara (verilere) ulaşmak istersek kullanabileceğimiz bir operatördür.

Örnek 1 :

```
SELECT * FROM personel WHERE adres LIKE '%İstanbul%'
```

Bu sorgulama ile adres alanında İstanbul geçen kayıtları listelemiş oluruz.

Örnek 2 :

```
SELECT * FROM personel WHERE adres LIKE '%İstanbul'
```

Bu sorgulama ile adres alanının sonunda İstanbul geçen kayıtları listelemiş oluruz.

Örnek 3 :

```
SELECT * FROM personel WHERE adres LIKE 'İstanbul%'
```

Bu sorgulama ile adres alanının başında İstanbul geçen kayıtları listelemiş oluruz

MAX

Tablo içinde, verilen kolondaki en büyük değeri geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select MAX(kolon_adı) FROM tablo;

Örnek 1:

En fazla aylık ücret alan personel ne kadar maaş alıyor?

```
Select MAX(aylik_ucret) From ucretler;
```

Örnek 2:

En fazla aile yardımı alan personelin maaşını ve sicil no' sunu öğrenmek istersek ;

```
Select per_sicil_no,MAX(aylik_ucret) From ucretler ;
```

örnek 3: Okul içerisinde en fazla devamsızlığı bulunan öğrenciyi görmek istersek;

```
SELECT MAX (devamsızlık)
FROM öğrenciler;
```

MIN

Tablo içinde, verilen kolondaki en küçük değeri geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select MIN(kolon_adı) FROM tablo;

Örnek :

En düşük aylık ücret alan personel ne kadar maaş alıyor ?

```
Select MIN(aylik_ucret) From ucretler;
```

Örnek :

En az aile yardımı alan personelin maaşını ve sicil no'sunu öğrenmek istersek ;

```
Select per_sicil_no,MIN(aylik_ucret) From ucretler ;
```

SUM (TOPLA)

Verilen kolondaki bütün değerleri toplayarak geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select SUM(kolon_adı) FROM tablo;

Örnek 1:

Personele ödenen toplam ücret nedir ?

```
Select SUM(aylik_ucret) From ucretler;
```

örnek 2: Okuldaki öğrencilerin devam etmedikleri günlerin toplamı ne kadardır? dersek;

SELECT SUM (devamsızlık)
FROM öğrenciler;

AVG (ORTALAMA)

Verilen kolondaki değerlerin aritmetiksel ortalamasını geri döndürür. Genel yazım biçimi aşağıdaki gibidir;

Select AVG(kolon_adı) FROM tablo;

Örnek :

Aylık ödenen ücret ortalamasını bulmak istersek ;

```
Select AVG(aylik_ucret) From ucretler;
```

SQL DEYİMLERİ VE ÖRNEKLERİ:

CREATE DEYİMİ

CREATE deyimi tablo ve view gibi bir veritabanı nesnesi yaratmayı sağlar. Veritabanı üzerinde bir tablo veya bir veritabanı nesnesi yaratmak için CREATE deyimi kullanılır.

Yapısı:
CREATE <tablo adı>
Örnek:

```
CREATE TABLE Musteri
(
  mus_id      char(4)       NOT NULL
  mus_ad      varchar(40)   NULL,
  ili         varchar(20)   NULL,
  ulke        char(2)       NULL,
  adres       varchar(30)   NULL
)
```

NOT: Char, varchar, integer, numeric gibi sözcükler tablo alanlarındaki temsil edilecek verinin türünü belirtir. SQL'de SMALLINT, VARCHAR, DECIMAL(x,y), FLOAT(x;y), DATE, LOGICAL, TIME, TIMESTAMP, GRAPHIC(n) gibi alan veri türleri vardır.

Örnek:

```
CREATE TABLE  Personel
(
```

```
Sskno Integer,
Adi Varchar(20) not null,
Soyadi Varchar(20) not null,
Departman integer
)
```

Örnek:

CREATE DATABASE Person

Person adında bir veri tabanı oluşturulur.

Örnek:

```
CREATE TABLE PERSONEL(
PERSONEL_ID int,
AD varchar(10),
SOYAD varchar(10)
)
```

Bu şekilde bir yazımla PERSONEL adında bir tablo oluşturulur. Tablo sütunları da PERSONEL_ID, AD, SOYAD'dır.

Örnek:

```
CREATE TABLE PERSONELYAKIN(
PERSONEL_ID int,
YAKIN_ID int,
YAKIN_AD varchar(10),
YAKIN_SOYAD varchar(10)
)
```

Bu örnekte de PERSONELYAKIN adında bir tablo oluşturulmuştur. Tablo sütunları da PERSONEL_ID, YAKIN_ID, YAKIN_AD, YAKIN_SOYAD' dır.

ALTER DEYİMİ

Daha önce yaratılmış nesnenin değiştirilmesini sağlar. Örneğin bir tablonun tasarımını değiştirmek gibi.

Örnek:

```
ALTER TABLE Musteri
ADD tel varchar(20) NOT NULL
```

Yukarıdaki deyimde musteri tablosunun alanlarına tel adlı bir alan daha eklenmiştir.

DROP DEYİMİ

Bir nesnesin silinmesini sağlar.

Örnek:

```
DROP TABLE MUSTERI
```

Müşteri tablosunun verilerini ve tabloyu siler.

SELECT DEYİMİ

Veritabanındaki verilere erişmenin, diğer bir deyişle onları görmenin ya da onları elde etmenin en sık kullanılan yöntemidir. Genellikle bir ya da daha çok tablonun bütün alanları ya da belli alanları için SELECT deyimi yazılır.

Temel Yapısı:

SELECT [ALL] [DISTINCT] liste [INTO yeni tablo] FROM [tablo]
[WHERE ifade]
[GROUP BY ifade]
[HAVING ifade]
[ORDER BY ifade]
[COMPUTE ifade]

Seçeneklerin Anlamları:

ALL sözcüğü bütün satırların sonuç listesinde görünmesini sağlar.
DISTINCT sözcüğü sadece tek olan (unique) kayıtların sonuç listesinde yer almasını sağlar.
liste parametresi veriden seçilecek kolonu (sütunu) belirtir.
INTO sözcüğü yeni bir tablo yaratmayı sağlar.
yeni tablo parametresi sorgu sonucu yaratılacak tabloyu belirtir.
FROM sözcüğü belli bir tablonun seçilmesini sağlar.
tablo parametresi ise sorgulanacak olan tablo ya da tabloları, görünümleri belirtir.
WHERE bir koşulu belirterek sadece o kuşula uyan kayıtların seçilmesini sağlar.
GROUP BY Kayıtların gruplanmasını sağlar. HAVING deyimiyle de ara toplamların alınmasını sağlar.
HAVING sözcüğü de kayıtlarda kısıtlama yapar ancak hesaplamayı etkilemez.
ORDER BY sözcüğü ise belirtilen kolona göre listelenen kayıtları sıralamayı sağlar. Sıralama artan (ASC) ya da azalan (DESC) olabilir
COMPUTE sözcüğü ise hesaplama yapar. Tipik olarak SUM, AVG, MIN, MAX, COUNT gibi fonksiyonları kullanarak hesaplama yapar.

Örnek:

SELECT * FROM musteri

Yukarıdaki deyim ile musteri tablosundaki bütün bilgiler elde edilir. SELECT deyiminin ardından kullanılan * (asterisk) işareti bütün kayıtlar anlamına gelir. Bu deyimin aynısı (aynı sonucu vereni) şu şekilde de yapılabilir:

Örnek:

SELECT kodu, ad, soyad, grup, il, bakiye FROM musteri

Müşteri tablosu:

kodu	Ad	Soyad	grup	İl	bakiye
1	Ahmet	Uzun	ithal	İZMİR	300000
2	Ayşe	Yılmaz	ithal	ANKARA	400000

3	Mehmet	Yılmaz	ihraç	ANKARA	100000
4	Hüseyin	Uzun	İhraç	İZMİR	600000
5	Nuri	Gezer	İthal	İZMİR	900000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

Sorgunun Sonucu:

1	Ahmet	Uzun	İthal	İZMİR	300000
2	Ayşe	Yılmaz	İthal	ANKARA	400000
3	Mehmet	Yılmaz	İhraç	ANKARA	100000
4	Hüseyin	Uzun	İhraç	İZMİR	600000
5	Nuri	Gezer	İthal	İZMİR	900000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

SELECT deyimi ile sadece belli kolonlar (alanlar) da seçilebilir:
Arama Kriterleri:

Karşılaştırma operatörleri	(=, >, <, >=, <>, !=, !<, !>
Aralık belirtme	BETWEEN ve NOT BETWEEN
Liste	IN ve NOT IN
String karşılaştırma	LIKE ve NOT LIKE
Bilinmeyen değerler	IS NULL ve IS NOT NULL
Koşulların birleştirilmesi	AND, OR
Olumsuzlaştırma	NOT

ÖRNEK:

- SELECT * from musteri WHERE bakiye BETWEEN 100000 AND 3000000

- SELECT * from musteri WHERE bakiye <= 100000 AND >= 3000000

1	Ahmet	Uzun	İthal	İZMİR	300000
3	Mehmet	Yılmaz	İhraç	ANKARA	100000
6	Fatma	Örnek	İhraç	İSTANBUL	300000

ÖRNEK:

```
SELECT * FROM MUSTERI

WHERE bakiye BETWEEN 100000 AND 500000

AND grup='ithal'
```

ARİTMETİK İŞLEMLER

Aritmetik işlemleri gerçekleştirmek için belli operatörler kullanılır:

OPERATÖRLER AÇIKLAMALARI

+	Toplama
-	Çıkarma
/	Bölme
*	Çarpma

ÖRNEK:

SELECT bakiye, bakiye*2 FROM MUSTERI

VERİLERİ SIRALAMAK

SELECT deyimi ile elde edilen veriler istenirse sıralanabilir. Sıralama belirtilen bir ya da daha fazla kolona göre yapılır. Bunun dışında sıralama ASC (ascending-artan) ya da DESC (descending-azalan) olarak belirtilebilir.

Kullanım Biçimi:

SELECT kolon_listesi

ORDER BY kolon adı **ASC** ya da **DESC**

ORDER BY sözcüğü ise verilerin istenilen alan göre sıralı olarak listelenmesini sağlar.

ÖRNEK:

SELECT * FROM musteri

ORDER BY ad

Yukarıdaki deyim ile müşteri tablosundaki bütün kayıtlar ad alanına göre sıralı olarak listelenirler.

INSERT DEYİMİ

Tabloya veri girmek için kullanılır.

INSERT INTO <tablo adı>
(sütunlar listesi**) VALUES (**değerler listesi**)**

ÖRNEK:

INSERT INTO CARIANA

(kodu, adi, grubu, adresi)

VALUES ('600', 'FARUK', 'A', '76 sokak no 5')

ÖRNEK:

INSERT INTO MUSTERI

(kodu,ad,soyad,grup,il,bakiye) VALUES

(‘05’,’Muhammet’,’BAYKARA’,’ihraç’,’Elazığ’,’500000’)

UPDATE DEYİMİ:

Tablodaki verileri güncellemek için kullanılır. Genellikle güncelleştirilecek satırı belirtmek için WHERE sözcüğüyle kullanılır.

Mevcut bir tablodaki satırları değiştirmek için UPDATE deyimi kullanılır. UPDATE deyimi sadece bir tablo üzerinde kullanılmalıdır. UPDATE deyimi ile SET ve WHERE sözcüğü kullanılır.

SET sözcüğü değiştirilecek kolonları ve değerleri belirtir. WHERE sözcüğü ise değiştirilecek satırı belirtir.

Kullanım biçimi:

UPDATE *tablo*

SET *kolon = ifade*

WHERE *arama_koşulu*

Örnek: Aşağıdaki örnekte fiyat değerini %10 artırır.

UPDATE siparis

SET fiyatı= fiyatı * 1.1

Örnek:

UPDATE MUSTERI

SET bakiye=bakiye *1.5

Where ad=’Muhammet’

Örnek:

UPDATE Musteri

SET Ad = 'Nuri Yılmaz'

WHERE kod='1';

DELETE DEYİMİ

Bir tablodaki verileri silmek için DELETE komutu kullanılır. Örneğin Öğrenci tablosundaki tüm verileri silmek için;

DELETE * from musteri;

Tabloda, bakiyesi 1000'den küçük olan müşterilerin satırlarını silmek için:

DELETE * FROM musteri WHERE bakiye <=1000

Kullanım biçimi:

DELETE tablo

WHERE arama_koşulu

Örnek: Tablodan satır silmek

Aşağıdaki örnekte müşteri tablosundan ithal grubuna sahip olan müşteriler silinir.

DELETE musteri

WHERE grup = 'ithal'

“

Örnek:

DELETE MUSTERI

WHERE ad=’Muhammet’

Örnek:

DELETE MUSTERI

WHERE bakiye<100000