

VERİ TABANI YÖNETİM SİSTEMLERİ - I



YRD. DOÇ. DR. ZEHRA ALAKOÇ BURMA

İÇİNDEKİLER

VERİ TABANI YÖNETİM SİSTEMLERİ	2
1. TEMEL VERİ TABANI KAVRAMLARI	2
1.1. Veri Nedir?	2
1.2. Veri Tabanı Nedir ?	3
1.3. Veri Modeline Göre Veritabanı Yönetim Sistemleri	4
1.4. Neden Veritabanı Kullanılır?	6
1.5. Veri Tabanı Yönetim Sistemlerinin Sağladığı Yararlar	7
1.6. Bilinen VTYS Programları	7
1.7. Proje ve VTYS arasındaki ilişki.....	8
2. VERİ ve VERİ MODELLERİ	10
2.1. Model Nedir?.....	10
2.2. Veri Kavramı	11
2.2.1. Veri Güvenliği.....	12
2.2.2. Veri Tekrarı ve Veri Bütünlüğü	12
2.3. Veri Modeli	13
2.3.1. Yapılar.....	13
2.3.2. Kısıtlar	14
2.3.3. İşlemler	14
2.4. Başlıca Veri Modelleri.....	15
2.4.1. Basit Veri Modelleri	15
2.4.2. Geliştirilmiş Veri Modelleri	16
3. VERİ TABANI TEMEL KAVRAMLARI	22
3.1. Tablo ve Elemanları	22
3.2. Veri Tipi (Data Type)	23
3.2.1. Access Veri Tabanı Veri Tipleri	23
3.2.2. MySQL Veri Tabanı Veri Tipleri.....	24
3.2.3. Oracle Veri Tabanı Veri Tipleri	24
3.3. Zorlayıcı (Constraint).....	25
3.4. Anahtar (Key)	26
3.5. Index (İndeks)	27
3.6. View (Görüntü)	27
3.7. Joining (ilişkilendirme).....	28
4. VERİ TABANI TASARIMI ve NORMALİZASYONU	29
4.1. Veri Tabanı Tasarımı.....	29
4.2. Veri Tabanı Normalizasyonu	33
4.3. İlişkisel Veri Tabanı Yönetim Sistemleri	35
5. ÖRNEK BİR VERİ TABANI TASARIMI ve NORMALİZASYONU	37
5.1. Örnek Personel Projesi Veri Tabanı Tasarımı	37
6. SQL VERİ İŞLEME DİLİ	41
6.1. SQL Nedir?	41
6.2. Veri Tabanı Programlarında ve SQL de Değişken Tanımlama	42
6.3. SQL Programı Çalıştırılması ve Yazım Kuralları	43

7.	TEMEL SQL KOMUTLARI-I	48
7.1.	CREATE (Yarat) Komutu	48
7.2.	ALTER (Düzenle) TABLE Komutu.....	51
7.3.	DROP (Sil) TABLE Komutu.....	52
7.4.	DESCRIBE Komutu.....	52
7.5.	INSERT (Ekle) Komutu	53
8.	TEMEL SQL KOMUTLARI-II	60
8.1.	SELECT (Seç) Komutu	60
8.1.1.	SQL Operatörleri	62
8.1.2.	Order By.....	65
8.1.3.	Distinct	67
8.2.	UPDATE (Güncelle) Komutu	68
8.3.	DELETE (Sil) Komutu.....	70
9.	SQL 'de FONKSİYONLAR	71
9.1.	SUM (Topla) Fonksiyonu.....	71
9.2.	AVG (Ortalama) Fonksiyonu	71
9.3.	MAX (En Büyük) Fonksiyonu	72
9.4.	MIN (En Küçük) Fonksiyonu.....	72
9.5.	COUNT (Say) Fonksiyonu.....	73
10.	SQL 'de GRUPLANDIRMA	75
10.1.	GROUP BY (Gruplandır) Deyimi.....	75
10.2.	HAVING (Sahip) Deyimi	77
11.	SQL 'de JOIN (BİRLEŞTİRME) İŞLEMİ	79
11.1.	JOIN (Birleştirme) İşlemi	79
11.2.	JOIN İşlemine Ait Örnekler	83
12.	SQL 'de YÖNETİMSEL FONKSİYONLAR.....	87
12.1.	VIEWS (Tablo Görünümü).....	87
12.2.	CREATE TABLESPACE (Tablo Uzayı = Veri Alanı).....	89
12.3.	CREATE USER (Kullanıcı)	89
12.4.	CREATE ROLE	90
12.5.	CREATE INDEX	90
12.6.	CREATE SEQUENCE	91
12.7.	GRANT	91
12.8.	REVOKE.....	91
	KAYNAKLAR.....	92

VERİ TABANI YÖNETİM SİSTEMLERİ

Veritabanı kavramı ilk olarak 1980’li yıllarda ortaya atılmış olmasına rağmen; günümüzde hemen hemen tüm veri kullanılan alanlarda Veritabanı Yönetim Sistemleri (VTYS) olmadan hiçbir şey yapılamaz hale gelmiştir. Basit bir Web uygulamasından, devasa kuruluşların ağır verilerine kadar, günümüzde bir çok alanda veritabanı uygulamalarına ihtiyaç duyulmaktadır. İşletim sistemlerinden sonra en popüler ve en çok gelir getiren yazılımlar Veritabanı Yönetim Sistemi Yazılımlarıdır.

Günümüzde, bir çok alandaki veri işlemlerinde pek çok Veri Tabanı Yönetim Sistemleri programları yaygın olarak kullanılmaktadır. Birbirinden farklı isimler adı altında anılan bu programlar için bir çok nesne birbiri ile aynı temel işlevi yerine getirmekte olup, yaklaşık olarak aynı teorilere dayanarak çalışırlar.

Veri tabanı, bir kuruluşun uygulama programlarının kullandığı operasyonel verilerin bütünüdür. Veritabanı Yönetim Sistemleri, verilerin fiziksel hafızadaki durumlarını, kullanıcıların erişimlerini düzenleyen sistemlerdir. İlişkisel VTYS’ler günümüzde yaygın olarak kullanılmaktadır.

1. TEMEL VERİ TABANI KAVRAMLARI

1.1. Veri Nedir?

Bilgi (information) kavramı yeni biçimlenmeye başlayan bir kavramdır; üzerindeki düşünce ekolleri henüz yeterince gelişmediğinden, İngilizce’de bile, bir çok anlama çekilmektedir. Bu kavram, daha önce de değinildiği gibi, çeşitli açılardan başlıca şu şekillerde açıklanmaktadır:

- Bilgi, bir nesne veya olayda veya bunlara ilişkin raporlarda ortaya çıkan mesaj ile ilgilidir. Bu açıdan ele alındığında, sadece kaynağın bir fonksiyonu olma özelliği taşır ve bazen veri olarak da ifade edilir.
- Diğer bir açıdan bilgi, mesajın iletilmesini açıklayan bir kavramdır. Bu açıdan ele alındığında bilgi, mesajın iletilmesi ile ilgili ihtimal hesaplarına dayanan, belirsizliğin azaltılması için gerekli olan bir kavramdır. Bir başka ifadeyle, bu anlamda bilgi iletişim kanalının da bir fonksiyonudur.
- Diğer bir açıdan ise, bilgi bir alıcı tarafından kazanılan anlam ile ilgilidir. Bu anlamdaki bilgi, hem iletişim kaynağının hem de alıcının bir fonksiyonudur.

Bu görüşlerin birincisi, literatürde fazla benimsenmeyen bir görüştür. Genellikle veri ile bilgi arasında farklılık olduğu ve verinin bilgi elde etmeye yarayan işlenmemiş ham malzeme olduğu kabul edilir. Kişi; bilmek, öğrenmek istedikten ve veriyi kullanmaya başladıktan sonra bilgi ortaya çıkar.

İkinci görüş en yaygın olanıdır. Buna göre bilgi anlamlı biçimde derlenen ve birleştirilen veridir ve şimdiki zamanda ve gelecekte verilecek kararlar için varolan gerçek bir değerdir. Bir başka ifade ile, bir kaynaktan, bir alıcıya iletilen mesajın içeriğidir. Bu anlamda bilgi, karar verme ile bağlantılıdır ve dolayısıyla veriye göre daha etkin bir kavramdır.

Veri kelimesinin tekil hali (datum) Latince'den gelmektedir. Sözlük anlamı “gerçek” tir. Fakat, veri her zaman somut gerçekleri göstermez. Bazen, kesin değildirler veya hiç olmamış şeyleri, örneğin bir fikri tarif etmek için kullanılırlar. Burada bahsedildiği şekliyle veri, bir kişinin formülleştirmeye veya kayıt etmeye değer bulduğu her türlü olay ve fikir anlamındadır. Bilgisayarda veri depolanacağı zaman, çoğunlukla veri tabanı yönetim sistemleri kullanılarak gerçekleştirilir. Çünkü bu tip sistemlerde yanlış verinin depolanmasını ve/veya verinin istenmeyen kişilerin kullanımına sunulmasını engelleyen bir takım imkân bulunmaktadır.

1.2. Veri Tabanı Nedir ?

Veritabanı en genel tanımıyla, kullanım amacına uygun olarak düzenlenmiş veriler topluluğudur. Birbirleriyle ilişkileri olan verilerin tutulduğu, mantıksal ve fiziksel olarak tanımlarının olduğu bilgi depolarıdır. Veritabanları gerçekte var olan ve birbirleriyle ilişkisi olan nesneleri ve ilişkileri modeller.

Veri tabanı, bir kuruluşun uygulama programlarının kullandığı operasyonel verilerin bütünüdür (toplamıdır). Burada; “kuruluş”, bir okul, üniversite, banka, bir üretim şirketi, hastane, devlet kuruluşu, vb. olabilir. “Operasyonel veri” bir kuruluşun çalışabilmesi, işleyebilmesi için kullanılan çok çeşitli verilerdir. Ticari bir şirket için müşteri bilgileri, satış bilgileri, ürün bilgileri, ödeme bilgileri, vb., okul için öğrenci bilgileri, açılan dersler, kimlerin kaydolduğu, öğretmen bilgileri, boş ve dolu derslikler, sınav tarihleri, vb., hastane için hasta bilgileri, doktor bilgileri, yatakların doluluk boşluğu, teşhis-tedavi bilgileri, mali bilgileri, vb ...

Belirli bir konu hakkında toplanmış veriler bir veritabanı programı altında toplanır. İstenildiğinde toplanan bilgilerin tümü veya istenilen özelliklere uyanları görüntülenebilir, yazdırılabilir hatta bilgilerden yeni bilgiler üretilerek bunlar çeşitli amaçlarla kullanılabilir.

Veri tabanı yönetim sistemi(VTYS), yeni bir veritabanı 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 oluşturmaktadır ve veri tabanına her türlü erişimi sağlar. Veri tabanının tanımlanması: veri tabanını oluşturan verilerin tip ve uzunluklarının belirlenmesidir. Veri tabanını oluşturulması ise veri için yer belirlemesi ve saklama ortamına verilerin yüklenmesini ifade eder. Veri tabanı üzerinde işlem yapmak; belirli bir veri üzerinde sorgulama yapmak, meydana gelen değişiklikleri yansıtmak için veri tabanının güncellenmesi ve rapor üretilmesi gibi işleri temsil eder. Ayrıca veri tabanı yönetim sistemi, verinin geri çağırılmasını sağlar. Veri tabanına yeni kayıt eklemek, eskileri çağırarak ve gerekli düzeltmeleri yapmak yoluyla, verinin bakımını ve sürekliliğini gerçekleştirir, kayıtlara yeni veri eklemek ve yeni kayıtlar oluşturmakla, veri tabanını genişletir.

Bir veritabanından beklenen özellikler, verileri koruması, onlara erişilmesini sağlaması ve başka verilerle ilişkilendirilmesi gibi 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, düzeltilmesi, silmesi veya görebilmesi sağlanabilir. Böylece veri girişinde ve veriye erişimde etkinlik ve güvenilirlik sağlanır.

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

Veri tabanlarını kurmayı, yaratmayı, tanımlamayı, işletmeyi ve kullanmayı sağlayan programlar topluluğuna “**veri tabanı sistemi**” ya da “**veri tabanı yönetim sistemi (VTYS) – data base management system (DBMS)**” denir.

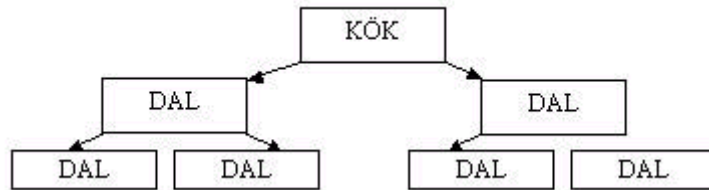
Bir veritabanı üzerinde birden fazla veritabanı bileşeni vardır; bu bileşenler, saklanmak istenen ham bilginin, belli bir formatta alınarak, veri haline gelmesi işleminde etkin rol oynarlar.

VTYS’ler 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 verileri tutmak üzere bir çok türde nesne ve bu nesnelere erişimleri düzenlemek üzere kullanıcılar, roller ve gruplar yer alır. Her bir kullanıcının belli hakları vardır. Bu haklar, kısıtlanabilir. Örneğin bir tablo ya da programcıyı bir kullanıcı kullanabilirken bir başkasının hakları veritabanı yöneticisi tarafından kısıtlanmış olabilir.

1.3. Veri Modeline Göre Veritabanı Yönetim Sistemleri

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

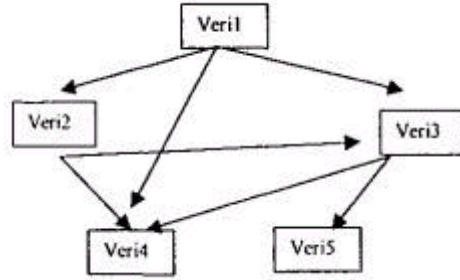
Hiyerarşik Veri Tabanları : Bu veritabanı tipi, ana bilgisayar ortamlarında çalışan yazılımlar tarafından kullanılmaktadır. Bu türde en çok kullanılan yazılım, IBM tarafından çıkarılan IMS’ dir. Uzun bir geçmişe sahip olmasına rağmen, PC ortamına uyarlanan hiyerarşik veri tabanları yoktur. Hiyerarşik veri tabanları, bilgileri bir ağaç (tree) yapısında saklar. Kök (Root) olarak bir kayıt ve bu köke bağlı dal (Branch) kayıtlar bu tip veritabanının yapısını oluşturur. Aşağıda böyle bir veri tabanının yapısı gösterilmektedir.



Şekil 1.3.1. Hiyerarşik Veri Tabanı Yapısı

Ağ Veri Tabanları : Hiyerarşik veri tabanlarının yetersiz kalmasından dolayı bilim adamlarının ortak çalışması sonucu ortaya konulmuş bir veri tabanı türüdür. Ağ veri tabanları verileri ağaçların daha da gelişmiş hali olan graflar (ağacın kendisinde özel bir graftır.) şeklinde saklarlar. Bu yapı en karışık yapılardan biridir.

İlişkisel Veri Tabanları : E.F. Codd Tarafından Geliştirilmiştir. Bu sistemde veriler tablo şeklinde saklanır. Bu veri tabanı yönetim sisteminde; veri alış verişi için özel işlemler kullanılır. Bu işlemlerde tablolar operandlar olarak kullanılır. Tablolar arasında ilişkiler belirtilir. Bu ilişkiler matematiksel bağıntılarla (ilişkilerle) temsil edilir. Günümüzde hemen hemen tüm veri tabanı yönetim sistemleri ilişkisel veri modelini kullanırlar. İlişkisel modeli 1970 yılında Codd önermiştir. 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



Şekil 1.3.2. İlişkisel Veri Tabanı Yapısı

Nesneye Yönelik Veri Tabanları : Günümüzde nesne kavramı her yerde kullanılmaktadır. Pek çok kelime işlemci ve hesap tablosu programlarının alıştığımız görünümüne artık bir de nesneler eklenmiştir. Ancak bu gerçek anlamda bir nesneye yönelik yazılım demek değildir. Yüzde yüz nesneye yönelik bir yazılımın tamamen nesneye temelli çalışması gerekir. Yazılımın mutlaka nesneye yönelik bir dilde yazılmış olması beklenir. Fakat Windows gibi işletim sistemi üzerinde çalışan yazılımlar bu özelliklere tümüyle sahip değildir. Sadece nesne kavramını kullanarak bazı ek özellikler sunarlar. Nesneye yönelik veri tabanı da , C++ gibi nesneye dayalı bir dille (OOPL) yazılmış olan ve yine C++ gibi nesneye dayalı (OOPL) bir dille kullanılan veri tabanı anlamına gelir. Günümüz teknolojisinde yüzde yüz nesneye yönelik bir veri tabanı yaygın olarak kullanıma sunulmuş değildir. Ancak nesneye yönelik veri tabanlarının bazı üstünlükleri olacağından söz ediliyor. İlişkisel veri tabanları ile karşılaştırıldığında; nesneye yönelik veri tabanlarının sahip olması gereken üstünlükler şunlardır:

1. Nesneler, bir tabloda yer alan bir kayıttan çok daha karmaşık yapıya sahiplerdir ve daha esnek bir yapıda çok daha kullanışlı düzenlenebilirler.
2. Nesneye dayalı bir veri tabanında, yapısı gereği arama işlemleri çok hızlı yapılabilir. Özellikle büyük tablolarla uğraşırken ilişkisel veri tabanlarından çok daha hızlı sonuca ulaşırlar. Ancak çalışma mantığı tümüyle değişir

Tüm bu özellikler tamamen nesneye yönelik olan veri tabanları için geçerlidir. Bazı ilişkisel veri tabanları ile çalışan yazılımlarda da nesnelerin bazı özellikleri ni kullanırlar, ama nesneye yönelik veri tabanı bunu kendini ilişkisel veri tabanı kurallarına uydurarak gerçekleştirebilir

1.4. Neden Veritabanı Kullanılır?

Bilgisayar ortamında veri saklama ve erişiminde geçmişten günümüze değişik yöntemler ve yaklaşımlar kullanılmıştır. Bunlardan Geleneksel Yaklaşım (Dosya - İşlem Sistemi) verileri ayrı ayrı dosyalarda gruplamaya dayanır. Verileri saklamak için programlama dillerinde kullanılan sıralı (Sequential) ve rastgele (Random) dosyalama sistemleri gibi. Birbiriyle ilgili olan ve aynı gruba dahil olan veriler bir dosyada, bir başka gruba dahil olan veriler de başka bir dosyada tutulurdu. Geleneksel Yaklaşımın birçok sakıncası vardır ve bu sakıncaların beraberinde getirdiği sorunların üstesinden gelebilmek için de Veri Tabanı Yaklaşımı zamanla Geleneksel Yaklaşımın yerini almıştır. Günümüzde veriler artık Veri Tabanı Yaklaşımı ilkesine göre VTYS' lerde tutulmakta ve işlenmektedir.

Geleneksel Yaklaşımın (Dosya - İşlem Sistemi) Sakıncaları

- Veri tekrarı ve veri tutarsızlığı
- Verinin paylaşılabilmesi
- Uygulamalardaki her yeni gereksinimin 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, standart eksikliği
- 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 yöntemi karmaşıklıklarının, çok katmanlı mimarilerle kullanıcılardan gizlenmesi
- Her kullanıcıya yalnız ilgilendiği verilerin, alışıktığı 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

1.5. Veri Tabanı Yönetim Sistemlerinin Sağladığı Yararlar

- Aynı veri değişik kişilerin PC'lerinde veya değişik bilgisayarlarda tekrar tekrar tutulmaz; **veri tekrarı ("data redundancy")** azaltılır ya da yok edilir.
- **Veri tutarlılığı ("data consistency")** : Aynı verinin değişik yerlerde birkaç kopyasının bulunması "bakım" zorluğu getirir: bir yerde güncellenen bir adres bilgisi başka yerde güncellenmeden kalabilir ve bu durum veri tutarsızlığına ("data inconsistency") yol açar.
- **Veri paylaşımı / Eşzamanlılık ("concurrency")** : Veri tabanı yönetim sistemi (VTYS) kullanılmadığı durumlarda veriye sıralı erişim yapılır. Yani birden çok kullanıcı aynı anda aynı veriye erişemez. Bir VTYS'de ise verinin tutarlılığını ve bütünlüğünü bozmadan aynı veritabanlarına saniyede yüzlerce, binlerce erişim yapılabilir.
- **Veri bütünlüğü ("data integrity")**: Bir tablodan bir öğrenci kaydı silinirse, öğrenci var olduğu diğer tüm tablolardan silinmelidir.
- **Veri güvenliği ("data security")** : Verinin isteyerek ya da yanlış kullanım sonucu bozulmasını önlemek için çok sıkı mekanizmalar mevcuttur. Veri tabanına girmek için kullanıcı adı ve şifreyle korumanın yanı sıra kişiler sadece kendilerini ilgilendiren tabloları ya da tablo içinde belirli kolonları görebilirler.
- **Veri Bağımsızlığı ("data independence")** : Programcı, kullandığı verilerin yapısı ve organizasyonu ile ilgilenmek durumunda değildir. VERİ BAĞIMSIZLIĞI, VTYS'lerinin en temel amaçlarındandır.

1.6. 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 yazmak için PL/SQL 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 bir yazılımdır. Küçük-orta ölçeklidir. Özellikle Web için geliştirilmiş bir VTYS'dir denilebilir.

PostgreSQL: 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 (1000000 civarı) ya da belli bir boyutun (yaklaşık 25MB) 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.

Bu ders notunda popüler olan VTYS programlarının isimleri yazılmıştır. Bunların dışında daha bir çok VTYS programı mevcuttur. VTYS'lerin Avrupa genelindeki pazar payları yaklaşık olarak aşağıda listelenmiştir. 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 elaman aranan VTYS Oracle ve arkasından da MS SQL Server gelmektedir.

IBM	%37.8
Oracle	%26.3
Microsoft	%15.4
Informix	%3.2
Sybase	%3
Digerleri	%14.3

2001 yılında bir araştırmaya göre Avrupa çapında VTYS'lerinin pazar payları [kaynak: Gartner]

VTYS'lerin bir çoğu 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, bu konu içerisinde öğrenmeyeceğimiz Stored Procedure (saklı prosedürler), Trigger, Fonksiyon gibi veritabanları için vazgeçilmez olan nesneler yazılabilmektedir.

1.7. Proje ve VTYS arasındaki ilişki

Herhangi bir veritabanı programında çalışmaya başlanılmadan önce, yapılacak işe uygun veri tabanı tasarımı yapılmalıdır. Bu işin en önemli aşamasıdır. Başlangıçta iyi tasarlanamayan 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 koşullarının başlangıçta tasarlanması gerekir.

Bir projede hangi veritabanının seçileceği, projenin çapı ile ilgili bir karardır. Aşağıdaki sorulara verilecek cevaplar projenin çapı konusunda karar vermede yardımcı olurlar.

- Projede kaç tablo kullanılacak?
- Her bir tabloda en fazla kaç satır yer alabilir? (tablodaki bilgi sayısıdır)
- Projeye aynı anda en fazla kaç kullanıcı bağlanacak?
- Proje günlük kaç transaction (INSERT-DELETE-UPDATE) gerçekleştirecek?
- Proje en fazla ne kadarlık yer kaplayacak ne kadarlık bir veritabanı dosyasına ihtiyaç duyulacak?
- Proje için güvenlik ne derece önemli? Ancak bir VTYS kullanılarak proje geliştirilecekse, hangisinin seçilmesi gerektiğinin dışında, hangi sürümlerinin kullanılacağı ya da hangi donanımlar üstünde çalıştırılacağı da önemlidir.

2. VERİ ve VERİ MODELLERİ

2.1. Model Nedir?

Model kelimesi; isim, sıfat ve fiil olarak ve her birinde oldukça farklı çağrışımlar yapacak şekilde kullanılmaktadır. İsim olarak “model”, bir temsili ifade eder. Bu temsil; bir mimarın, bir binanın küçük ölçekli modeli veya bir fizikçinin bir atomun büyük ölçekli modelini oluşturması anlamındadır. Sıfat olarak “model”, mükemmeliyetin veya idealin ölçüsünü ifade eder. “Model ev”, “model öğrenci” ve “model eş” ifadelerinde olduğu gibi. Fiil olarak “model” ise, bir şeyin nasıl olduğunu ispat etmek, açıklamak, göstermek anlamındadır.

Bilimsel modeller bütün bu çağrışımları bünyelerinde bulundurlar. Onlar; durumların, nesnelerin ve olayların temsilleridir. Gerçeklerden daha az karmaşık ve böylece araştırma amacıyla kullanılmaları daha kolay olduğundan, bu anlamda ideal hale getirilmişlerdir. Gerçek durumlarla karşılaştırıldıklarında, modellerin basitliğinin sebebi, gerçeklerin sadece uygun özelliklerini temsil etmelerinden kaynaklanmaktadır. Örneğin, yeryüzünün bir kısmının modeli olan bir yol haritasında, bitki örtüsü gösterilmez. Çünkü bu durum, o haritanın bir yol haritası olarak kullanımı açısından uygun değildir. Güneş sisteminin bir modelinde, gezegenleri temsil eden toparların, gezegenlerle aynı maddeden yapılmış olmaları veya aynı sıcaklığa sahip olmalarına ihtiyaç yoktur.

Bilimsel modellerden, gerçeklerin farklı boyutları hakkındaki bilgiyi artırmak ve birbirleri ile ilişkilendirmek için faydalanılır. Modeller, gerçeği ortaya çıkarmak ve bundan daha fazla olarak, geçmiş ve şimdiki durumu açıklamak ve geleceği tahmin ve kontrol etmek için kullanılır. Modeller uygulanarak, gerçekler üzerinde bilimin kontrolü sağlanır. Modeller gerçeğin tarifi ve açıklamasıdır. Bir bilimsel model, aslında, gerçek hakkında bir veya bir dizi ifadelerdir. Bu ifadeler olaylara dayanan, kanun benzeri ya da teorik olabilir.

Bilimde, sıradan işlemlerde olduğu gibi, değişik tipte modeller kullanılır: Simgesel Model, Benzetim Modeli ve Sembolik Model.

Simgesel Modeller, durumların büyük veya küçük ölçekli temsilleridir. Gerçek şeylerin uygun özelliklerini temsil ederler. Şekilleri, temsil ettikleri şeylere benzer. Yol haritaları, hava fotoğrafları bu tip modellere örnek verilebilir.

Benzetim Modelleri, bazı durumlarda ise; haritada yükseltiler, 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 sisteminin kullanıcısı, özellikle bir karar verici, kendisini sonsuz denebilecek boyutta bilgi karşısında bulur. Bir bilişim sistemi modeli, gerçek bilgi kümesinin alt 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 yığınının bir düzen getirmeyi, hatta bir yapı oluşturmayı amaçlar. Tek bir model yoktur. Var olan bilgi yığınının, uygulanan farklı modeller doğal olarak farklı yorumlar getirir.

Gerçek hayattan alınan bir olayın modelinin iki tip özelliğinin olması gerekir. Birincisi, statik özellikler, ikincisi de dinamik özelliklerdir. Statik özellikler zamana göre değişiklik göstermez yada çok az gösterir. Dinamik özellikler ise bunun tam tersi olarak devamlı değişkendirler. Bu durumda, herhangi bir model (M), o modeli oluşturan kurallar kümesi (K) ve işlemler kümesinin (İ) bir fonksiyonu olarak tanımlanabilir.

$$M = f(K, İ)$$

Modeli oluşturan kurallar kümesi (K), veri modelinin statik özelliklerini temsil eder ve Veri Tanımlama Dili'ne (VTD) karşılık gelir. (M) veri modeli içinde, veri için izin verilen yapıların tanımlanması için kullanılır. Mümkün olan yapılar, birbirini tamamlayan iki şekilde belirlenir. Nesneler ve ilişkiler, kategorilerinin belirlenebilmesi için genel kurallar kullanılarak tespit edilir. Modelde bulunmasına izin verilmeyecek olan nesneler veya ilişkiler, sınırlar tespit edilerek hariç tutulur. Örneğin, bir işçi veri tabanında, her işçinin bir sigorta numarasının olması ve yöneticisinden fazla kazanmaması gibi sınırlar tespit edilebilir.

Gerçek hayattaki dinamik özelliklerin modelde kullanılabilmesini işlemler kümesi sağlar ve Veri Yönlendirme Dili'ne (VYD) karşılık gelir. Di gibi bir veri tabanı oluşumundan Dk gibi başka bir veri tabanı oluşumu elde etmek için yapılmasına izin verilen işlemleri tanımlar.

2.2. Veri Kavramı

Birinci bölümde veri kelimesi tanımlanmış ve yanlış verinin depolanmasını ve/veya verinin istenmeyen kişilerin kullanımına sunulmasını engelleyen bir takım imkânların olması gerektiği belirtilmişti.

Yanlış verinin iki türlü kaynağı olabilir: Programlama hataları, klavyeden hatalı giriş nedeniyle oluşan yanlışlıklar ve veri tabanı programının kötü niyetli kullanımı. Veri tabanlarının korunması iki başlık altında incelenebilir

1. Veri güvenliği,
2. Veri bütünlüğünün sağlanması.

2.2.1. Veri Güvenliği

Veri güvenliğinin konusu, veri tabanını, dolayısıyla veriyi yetkisiz kullanımlara karşı korumaktır. Bu konuda çok çeşitli yaklaşımlar vardır. Hem verinin istenmeyen şekilde değiştirilmesine veya zarar görmesine hem de yetkisiz kullanımlara engel olmak gerekir. Bunu sağlamak için bazı genel teknikler geliştirilmiştir.

- **Kullanıcıların tanımlanması:** Çok kullanıcıli ortamlarda farklı yetkilere sahip kullanıcılar vardır. Farklı yetkilere sahip kişilerin, veri tabanında ulaşabilecekleri veri farklıdır. Örneğin, bilgisayara veri girişi yapan bir işletimcinin, 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.

- **Fiziksel koruma:** Şifre sisteminin yeterli olmadığı durumlarda, verinin fiziksel koruma altına alınması gerekir. Yangın veya hırsızlığa karşı verinin yedeklenmesinin yapılması gibi.

- **Kullanıcı haklarının temin edilmesi:** Sistemde hangi kullanıcının hangi yetkilere ve haklara sahip olduğu ve neler yapabileceğinin önceden belirlenmiş olması gerekir. Bir kişinin yetkisini veya hakkını başka bir kişiye vermesi ise, sistemin müdahalesi dışında gerçekleşen bir durumdur.

Özellikle veri tabanının sorgulanmasında güvenlik problemleri ortaya çıkmaktadır. Hangi tür kullanıcının, hangi sorgu tiplerini sisteme yönltebileceğinin daha önceden tespit edilmesi gerekmektedir. Fakat, yukarıda bahsedilen önlemlerden hiç biri tam bir koruma sağlamaz. Bu yüzden, birden fazla önlem kullanarak güvenlik artırılabilir.

2.2.2. Veri Tekrarı ve Veri Bütünlüğü

Bir veri tabanı yönetim sisteminde farklı veri dosyalarında; isim, adres, numara gibi bilgilerin bulunması gerekebilir. Örneğin, hem müşteri bilgilerini içeren bir veri tabanı dosyasında, hem de satılan malların seviyatının yapılacağı adreslerin bulunduğu başka bir veri dosyasında, müşteri adresi bilgilerinin yer alması gerekebilir. Yani, pek çok durumda, 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. Özellikle çok kullanıcıli ortamlarda bu işlem oldukça önemlidir. Aksi taktirde, veri tabanında uygun olmayan veri ile çalışılmış olur. Veri bütünlüğünün bozulmasının bir sebebinin, veri tekrarı olduğu söylenebilir. Bir başka sebep de, verinin zayıf geçerlilik kontrolüdür. Bunun sebepleri de şu şekilde sıralanabilir:

- Veri güvenliğinin yetersiz oluşu,
- Veri tabanının 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 veri tabanı yönetim sistemi programları, yukarıda sayılan bütün problemlerin üstesinden hemen hemen gelebilecek çözümler üretmişler ve bunları kolay kullanılabilir hale getirmişlerdir. Kullanıcıların, bir veri tabanı oluştururken, ayrıca bu problemler için önlem almalarına gerek kalmamaktadır.

2.3. Veri Modeli

Bir veri modeli, verinin hangi kurallara göre yapılandırıldığını belirler. Fakat yapılar, verinin anlamı ve nasıl kullanılacakları hakkında tam bir açıklama vermezler. Veri üzerinde yapılmasına izin verilen işlemlerin belirlenmesi de gerekir. İşlemler, yapının sunduğu çerçeve içinde çalıştırılırlar.

2.3.1. Yapılar

Veriyi yapılandırma ve görüntüleme mekanizmalarından biri soyutlamadır. Soyutlama, detayları gizleme ve genel üzerinde yoğunlaşma yeteneğidir. Veri modellemesinde soyutlama, veri kategorilerini elde etmek için kullanılır.

Veri yapılarının oluşturulmasında kullanılan kavramlardan biri de kümelerdir. Bir küme, düzgün bir şekilde tanımlanmış ve bir üyelik koşulu tarafından temsil edilen nesneler topluluğudur. Üyeleri az ya da çok homojen olan kümeler vardır. Örneğin, 10 ile 20 arasındaki tam sayılar, uzunluğu 20 karaktere kadar olabilen alfanümerik değerler gibi. Bu homojen kümeler, tanım kümesi olarak isimlendirilirler. Semantik açıdan bir anlam taşıyan nesneyi temsil eden, isimlendirilmiş bir tanım kümesi (örneğin, MAAŞLAR), öznitelik olarak isimlendirilir.

Veri yapılarının unsurlarından biri de ilişkilerdir. İlişki, kümelerin toplanmasını ifade eder. Aynı zamanda kendisi de bir kümedir ve semantik olarak belirli bir karşılığı yoktur. Fakat, veri modellemesinde ilişki, iki nesne arasındaki ilişkiyi gösteren bir tip olarak tanımlanabilir. Örneğin, İŞÇİ ve İŞYERİ arasında bir İŞ ilişkisi vardır. Bir ilişkiye uygulanabilecek semantik bir tercüme, her satırı bir varlığa karşılık gelecek şekilde belirlemektir. Varlığın tam bir tanımı olmamasına rağmen, objektif bir gerçekliği olan veya olduğu düşünülen şey olarak tarif edilmektedir. Örneğin, İŞÇİ bir varlık tipi olarak belirlenebilir. Bu varlık tipinin özellikleri de, İŞVEREN, İSİM, ADRES, YAŞ, BÖLÜM, TECRÜBE ve MAAŞ olabilir.

Bir veri yapısı oluşturulurken, verinin bir şekilde bilgisayara yerleştirilmesi söz konusu olduğu için, nesneler ve onlar arasındaki ilişkilerin temsil edilmesi gerekir. Bu tür bir temsil tablolarla yapılabilir. Bir tabloda sütun başlıkları olarak öznitelikler ve satırlarda da bu özniteliklerin aldığı değerler (kayıt birimleri) yer alır. Tablodaki her bir sütun, bir veri birimidir.

Düz bir dosyadan oluşan veri tabanları olabileceği gibi (örneğin isim ve adres alanlarından oluşan adres veri tabanları), birden fazla dosyadan oluşan veri tabanları da vardır ve daha yaygın bir şekilde kullanılmaktadır. Bir veri tabanında temsil edilebilecek genel kayıt ilişkilendirme tipleri vardır. Bunlar şu şekilde sıralanabilir:

- **Bire bir 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. **Örnek** : bir işçinin doğum yeri bilgisinin doğum yerleri tablosunda bir şehre karşılık gelmesi gibi.

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

Veri üzerindeki mantıksal sınırlamalara kısıt adı verilir. Kısıtların genel olması tercih edilen bir durumdur. Örneğin, “Tüm yöneticilerin maaşları, işçilerinden daha fazladır” ifadesi, “Ali Bey’in maaşı Veli Bey’in maaşından daha fazladır” ifadesinden daha geneldir ve dolayısıyla daha kullanışlıdır. Kısıtlar, veri modellerinde bütünlük sağlamak ve semantik nedenlerle kullanılır. Kümeler üzerinde kullanılabilir. Örneğin, “İŞÇİ varlık tipinin YAŞ özniteliği 15 ve 65 arasında değer alabilir” şeklinde bir kısıtlama veri modelinde uygulanabilir. Bu sayede gerçek dünyada karşılaşılan bir özellik, oluşturulacak veri tabanına yansıtılabilir.

Bir ilişki, iki veya daha fazla kümenin elemanları (nesneler) arasında mümkün olabilecek tüm kombinasyonları içerir. Bu işleme haritalandırma denir. Kısıtlamalar ilişkiler üzerinde belirlendiği zaman, nesneler arasındaki bir takım anlam ifade etmeyen, fakat teorik olarak mümkün olabilen ilişkilerin, gereksiz yere modele yerleştirilmeye çalışılması önlenmiş olur.

Tablolarda kısıtların kullanılması, fonksiyonel bağımlılıkların belirlenmesi amacını taşımaktadır. Örneğin, İŞÇİ varlık tipinde İŞÇİ_NO özniteliği bir aday anahtar olabilir. Çünkü, diğer öznitelikler, bu özniteliğe bağlı olabilir. Başka aday anahtarlar da olabilir (İSİM, ADRES gibi). Bir tabloda, aday özniteliklerinden biri asıl anahtar olarak belirlenir. İki ayrı ilişki tipi arasında bir bağlantı kurabilmek, o ilişkinin bir tablosundaki bir anahtarın, diğer tabloya eklenmesi ile mümkün olur. Buna türetme, ikinci tabloya eklenen anahtara da yabancı anahtar denir.

2.3.3. İşlemler

İşlemler, bir veri tabanı durumundan, bir başka veri tabanı durumu elde etmek için yapılan işlemlerdir. Bunlar, verinin çağırılması, güncellenmesi, eklenmesi veya silinmesi ile ilgili işlemlerdir. Çok kesin seçimler üzerinde yapılır. Bunların yanında, daha genel işlemler de vardır. Örneğin; bütünlük mekanizması, toplam fonksiyonları (istatistiksel fonksiyonlar da bunlar arasındadır), veriye ulaşım kontrolleri gibi. Bu mekanizmalara veri tabanı yöntemleri denir. Bu mekanizmalar, CODASYL tarafından yayınlanmıştır.

2.4. Başlıca Veri Modelleri

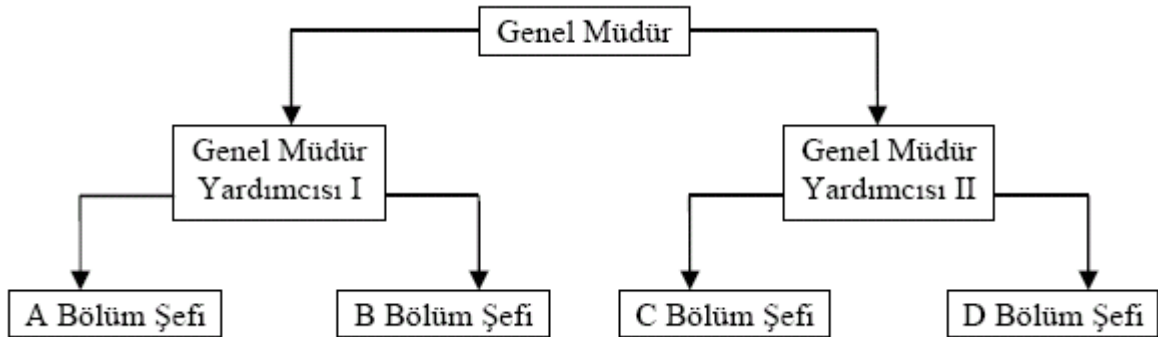
Veri modellemesi yapmak amacıyla pek çok veri modeli geliştirilmiştir. Fakat, bütün modeller aynı özellikleri taşımaz. Her modelin farklı durumlara uygun olan çeşitli özellikleri vardır. O yüzden, modeller arasında tam bir sıralama yapmak mümkün değildir. Bununla birlikte, yetersiz de olsa bir sınıflama yapılabilir.

2.4.1. Basit Veri Modelleri

Basit veri modelleri olarak ayrılan ilk grup veri modelleri, bilgisayarlarda veri işleme ihtiyacının ortaya çıkmasıyla, dosyalama sistemleri oluşturmak amacıyla kullanılmaya başlanan Hiyerarşik ve Şebeke veri modelleridir.

2.4.1.1. Hiyerarşik Veri Modelleri

Hiyerarşik veri modellerinde çoklu ilişkileri temsil edebilmek için, varlık tiplerinin her ilişki için ayrı ayrı tanımlanması gerekir. Bu da gereksiz veri tekrarına sebep olur. 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.

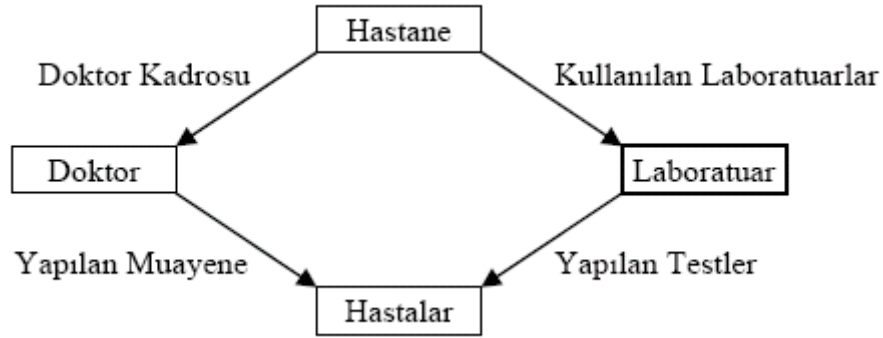


Şekil 2.4.1.1.1. Hiyerarşik Tanım Ağacı

2.4.1.2. Şebeke Veri Modelleri

Şebeke veri modelleri, tablo ve grafik temellidir. Grafikteki düğümler varlık tiplerine karşılık gelir ve tablolar şeklinde temsil edilir. Grafiğin okları, ilişkileri temsil eder ve tabloda bağlantılar olarak temsil edilir. Spesifikasyonu, 1971 yılında DBTG-CODASYL tarafından belirlenmiştir.

İki ayrı veri yapılandırma aracı vardır: Kayıt tipi ve bağlantı. Kayıt tipleri varlık tiplerini belirler. Bağlantılar ise, ilişki tiplerini belirler. Bu yapıyı gösteren grafiğe de veri yapısı grafiği adı verilir.



Şekil 2.4.1.2.1. Şebeke Veri Yapısı Grafiği

Şebeke veri modeli, veri modelleri içinde en genel olanlarından biridir. Şebeke içinde bir eleman, herhangi bir başka 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.

2.4.2. Geliştirilmiş Veri Modelleri

1960 ve 1970'li yıllarda hiyerarşik veri modeli üzerine geliştirilmiş veri tabanı yönetim sistemleri ile, daha sonra, şebeke veri modeli ile çalışan VTYS yaygın kullanımda iken, teorik temelleri ve deneysel uygulama ve geliştirme aşamaları, 1970'li yıllarda tamamlanmış olan ilişkisel veri modeline dayalı VTYS, 1980'li yıllarda ticari kullanıma girerek çok hızla yaygınlaşmışlardır.

Geliştirilmiş veri modelleri, Varlık-İlişki Veri Modelleri, İlişkisel Veri Modelleri ve Nesne Yönelimli Veri Modelleri olarak sıralanabilir.

2.4.2.1. Varlık-İlişki Veri Modelleri (Vİ Modeli)

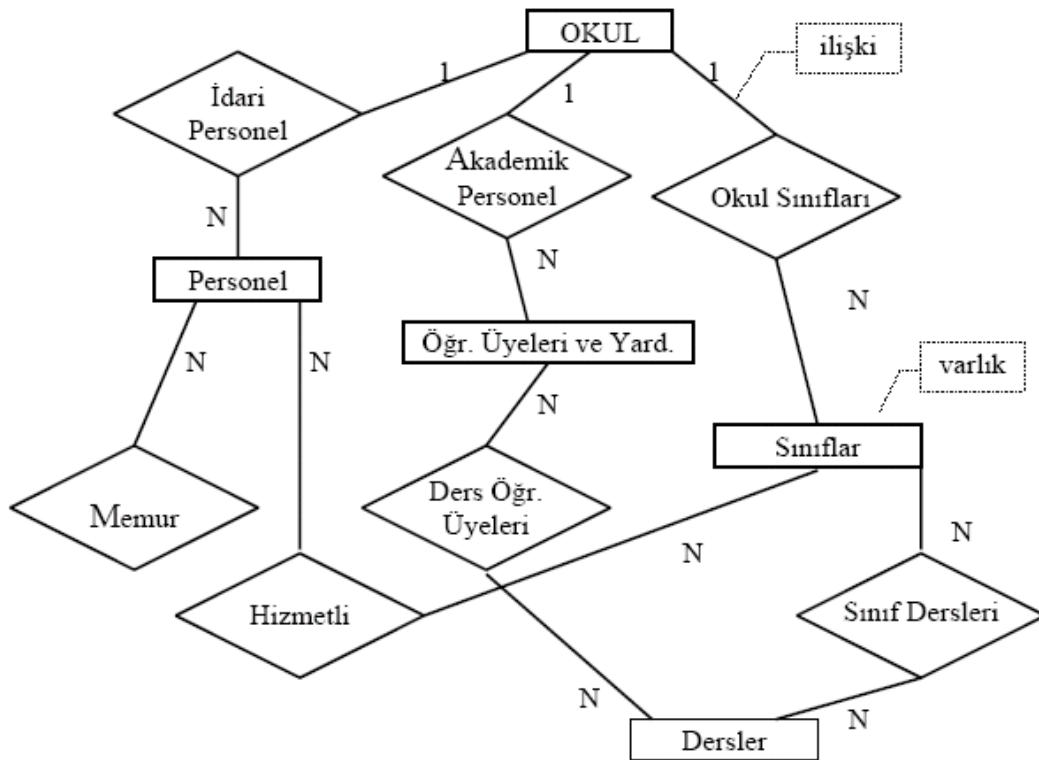
Bir veri tabanı uygulaması için varlık, hakkında tanımlayıcı bilgi saklanabilen herşey olarak kabul edilmektedir. Varlık, bağımsızdır ve tek başına tanımlanabilir. Bir varlık, ev, öğrenci, araba gibi bir nesne ya da futbol maçı, tatil, satış gibi olaylar olabilir. En anlamlı şekilde kendi öznitelikleri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

Varlık-İlişki veri modelleri (Vİ), sütunlarında, öznitelikleri temsil eden değişkenlerin yer aldığı ve satırlarında da enstantanelerin temsil edildiği tablolar, varlıklar ve aralarındaki ilişkileri oklarla göstermek için kullanılan grafikler üzerine kurulmuş veri modelleridir.

Ticari veri tabanlarında yaygın olarak kullanılan veri modellerinden biridir. Şebeke ve hiyerarşik veri modelleri ile ortak noktaları vardır. Fakat, veri tabanı tasarım süreçleri için kullanılmak maksadıyla geliştirildiklerinden bu iki modelin genelleştirilmiş şeklidir. Çoklu ilişki tiplerinin doğrudan modelde kullanılmasına izin verir. Bu modelde, kurum şeması kavramı söz konusudur. Bu şema, kurumun tüm verisinin görünümünü temsil eder ve fiziksel sınırlamalardan bağımsızdır. Aynı zamanda, bu şema ANSI/X3/SPARC kavramsal şemasına çok benzemektedir. Aralarındaki temel fark, kavramsal şemanın, dahili şema ve harici şema arasında haritalandırma yapabmesidir. Temelde, Vİ veri modeli, veri tabanının mantıksal özelliklerinin bir dokümantasyonudur. Vİ modeline göre düzenlenen veri tabanının yapısı, **Varlık-İlişki Diyagramı** ile gösterilir.

Şebeke ve hiyerarşik veri modellerinde, sadece ikili fonksiyonel bağlantılara izin verilmektedir. Vİ veri modelinde ise, varlıklar arasında n adet ilişki tanımlanabilir. Bu ilişkiler, bire bir, fonksiyonel veya çoklu olabilir. Tekrar eden bağlantılar da kullanılabilir.

Öznitelik, varlık veya ilişki ile bunların aldığı değerler arasındaki haritalandırmayı temsil eder. Bazı özniteliklerin birden fazla değeri olabilir. Örneğin, telefon numarasını bir öznitelik olarak kabul edersek, bir şirketin birden fazla numarası olabilir. Fakat, doğum günü özniteliği ele alındığında, her bir kişinin bir doğum günü olduğundan, bu öznitelik, çok değere sahip değildir.



Şekil 2.4.2.1.1. Varlık – İlişki Diyagramı

Vİ modeli ilk olarak ortaya konulduğunda (1976) bir veri dili geliştirilmemişti. Bunun anlamı, bilgi sorgulamalarının küme işlemleri ile yapılması demektir. Daha sonra, veri modeli için CABLE (ChAin-Based LanguagE) dili geliştirildi. Vİ modellerinin en büyük avantajlarından biri, uzman olmayan kişiler tarafından da anlaşılabilir yapıda olmasıdır. Üzerinde düzeltme işlemleri kolayca yapılabilir. Bu açıdan belirli bir veri tabanı yönetim sistemine bağlı değildir.

2.4.2.2. İlişkisel Veri Modelleri

İlişkiler ve onların temsilleri olan tablolardan oluşan veri modelleri ilk olarak 1970 yılında Codd tarafından ortaya atılmıştır. İlişkisel veri modelleri formüle edilirken, veri yönetimi ihtiyaçlarını karşılayabilmek için ilişkinin matematiksel teorisi, mantıksal olarak genişletilmiştir. İlişkisel veri modellerinde kullanılan tek yapılandırma aracı ilişkidir. İlişkinin tanımı, veri tabanı ilişkilerinin zamana bağlı olması dışında, matematiksel tanımı ile aynıdır. Yani, bir veri tabanı ilişkisinde satırlar, eklenebilir, değiştirilebilir ya da düzeltilebilir. Aşağıdaki örneklerde büyük harflerle yazılan ifadeler ilişki isimlerini, parantez içindeki ifadeler de tanım kümesi isimlerini göstermektedir.

```
HASTANE(Hastane_Kodu, Hastane_Adı, Adres, Tel_No, Yatak_Sayısı)
DOKTOR(Hastane_Kodu, Diploma_No, Adı, Uzmanlığı)
İŞÇİ(Sigorta_No, Adı, Adres, Kıdem, Maaş, Yaşı)
```

Şekil 2.4.2.2.1. Tanım Kümesi

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

HASTANE				
Hastane_Kodu	Hastane_Adı	Adres	Tel_No	Yatak_Sayısı
1	Marmara Üniv.	Altunizade	216 333 33 33	150
2	HP Numune	Haydarpaşa	216 333 33 34	150
3	Kartal Devlet	Kartal	216 333 33 35	120
4	Haseki	Fındıkzade	212 555 55 55	100

Şekil 2.4.2.2.2. İlişkisel Tablo

İlişkisel şema listesini oluşturan her bir satır, bir tablo olarak temsil edilir. Tablonun sütunları öznitelik olarak isimlendirilir. Örneğin, HASTANE tablosunun öznitelikleri; Hastane_Kodu, Hastane_Adı, Adres, Tel_No ve Yatak_Sayısı'dır.

Tablonun satırlarında bütün özniteliklerin aynı değerler aldığı iki satır olamaz. Her satır diğerinden mutlaka farklıdır. Aksi halde veri tekrarı söz konusu olur. Veri tabanlarındaki ilişki kavramı, matematikteki küme kavramını esas aldığı için aynı satırın bir tabloda birden fazla yer alması mümkün değildir. Bu nedenle, ilişki için bir anahtar kullanmak gerekir. Anahtar, bir satırı tek başına tanımlayabilen öznitelikler kümesidir. Anahtar kavramı, ilişkisel veri modelinde kullanılan önemli bir kısıttır.

Bu kurallar kullanılarak hazırlanan bir ilişkiyel modelde, yine de belirsizlikler ve uyumsuzluklar bulunabilir. Bunları gidermek için de bir dizi düzgüleme işlemine gerek duyulabilir. Düzgülemek, veri tabanı tasarım prensiplerini yapısalılaştırmayı amaçlar. İlişkiler ve öznitelikler arasındaki fonksiyonel bağımlılıkları düzenler. Birbirini takip eden beş işlemden 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.”

Veri üzerinde yapılacak işlemler için, ilişkiyel veri modellerinde üç tip dil kullanılır. Birincisi, matematikteki ilişkiyel işlemlere dayanır. Bu tip dillere örnek olarak INGRES ve QUEL verilebilir.

İkinci tip dil, görüntü yönelimlidir. Boşluk doldurma yöntemiyle çalışır. Örneğin, QBE (Query By Example) ve CUPID bu tür dillerdendir.

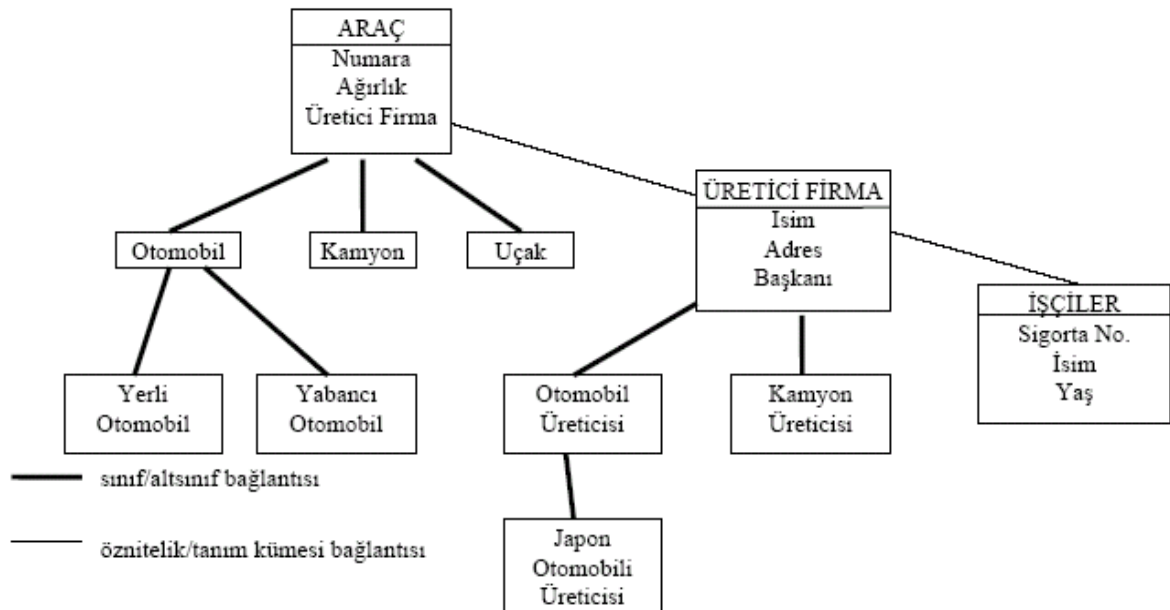
Üçüncü tip dil, haritalandırma yönelimli dildir. Bu tip diller, bilinen bir özniteliğin ya da öznitelik kümesinin, aranan bir özniteliğin ya da öznitelik kümesinin üzerinde, bir ilişki yoluyla haritalandırılması prensibiyle çalışır. Örneğin, yapısal sorgulama dili (SQL) bu tip bir veri dilidir. SQL ilerleyen bölümlerde detayları ile anlatılacaktır

2.4.2.3. Nesne Yönelimli Veri Modelleri

Nesne yönelimli sistemler, bir istatistiksel sistem içinde, esnek veri yapılarının geliştirilmesi ve istatistiksel modellerin sunumunda da kullanılmaktadır. 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 tesadüfi bir nesne kümesinin elde edilmesi mümkün değildir. Çünkü bütün nesnelerin, modelde önceden tanımlanmış olması gerekmektedir. İlişkiyel modeldeki ilişki kavramı, nesne yönelimli modelde sınıf kavramına karşılık gelmektedir.

Nesne yönelimli modellemenin en önemli faydalarından bir tanesi de, modeldeki nesneleri tanımlarken, ortak öznitelik ve metotlara sahip nesnelerin kullanıldıkları her farklı ortamda, tekrar tanımlanmalarına gerek duyulmamasıdır. Aksi taktirde bu durum, hem tekrardan dolayı yer kaybına, hem de modeldeki dinamik değişikliklerin pratik olmamasına sebep olacaktır. Nesne yönelimli veri modelindeki sınıf hiyerarşisi ve kalıtım özelliği, bu olumsuz durumu ortadan kaldırarak, nesnelerin özniteliklerinin ve metotlarının yeniden kullanımına imkân vermektedir. Çünkü bir sınıf, ait olduğu üst sınıfın tüm özelliklerini taşır ve o sınıftaki nesneler, modelin başka bir yerinde kullanılacağı zaman yeniden tanımlanmaya gerek kalmadan tekrar kullanılabilir.



Şekil 2.4.2.3.1. Sınıf Hiyerarşisi

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. Aşağıdaki nesne yönelimli veri modeli buna bir örnektir.

```

CLASS Otel
    PROPERTIES
        isim : string;
        adres : string;
        sahibi : kurum;
        yönetci : kişi;
        servis : (yüzme_havuzu, sauna, tenis, bar, lokanta...)
    ...
    OPERATIONS
        Create (işemler)
        Rezervasyon (oda_no: integer; müşteri: kişi; geliş_tar, ayrılış_tar: Date
Type)
    ...
END Otel

```

Bu örnekteki sahibi, adres, servis gibi nesneler, Otel sınıfının elemanlarıdır ve tanımları da birbirinden farklıdır. Örneğin, Otel sınıfının bir elemanı,

İsim : İstanbul Oteli Adres : Beşiktaş, İstanbul Sahibi : (kurum elemanı) Yönetici : (yönetici elemanı) Servisler : yüzme_havuzu, sauna, tenis, bar, lokanta
--

şeklinde tanımlanırken, kurum nesnesinin bir elemanı, aşağıdaki gibi,

İsim : İstanbul Otelcilik A.Ş. İdari Yeri : İstanbul Telefon : 0212-555 5555
--

yönetici nesnesinin bir elemanı da, aşağıdaki gibi tanımlanabilir.

İsim : Ahmet Gel Adres : Fenerbahçe, İstanbul Doğum_Tar : 1943
--

- **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 edilmesi ile 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, soysal nesne olarak temsil edilmesi ile ilgili bir soyutlama yöntemidir. Örneğin, bir kurumda çalışan personel şu şekilde düzenlenebilir:

Bilgisayar Ekibi (Analizci, Programcı, İşletimci) Çalışanlar (Bakım Ekibi, Bilgisayar Ekibi, Yönetici)

“Bilgisayar Ekibi” nesnesi; Analizci, Programcı ve İşletimci nesneleri için bir soysal nesnedir. Aynı şekilde Çalışanlar nesnesi de Bakım Ekibi, Bilgisayar Ekibi ve Yönetici nesneleri için soysal nesne durumundadır.

Nesne yönelimli veri modellerinin, ilişkisel veri modellerine karşı üstünlükleri vardır. Bunlar; NYVM’nde veri tiplerinin (tamsayı, gerçek sayı, alfanümerik değer, tarih vb.) İLVM’e göre daha esnek olması, nesne tanımlarında soyutlama yapılabilmesine imkân vermesi ve bu tanımların semantik içeriklerinin de olması sayesinde, veri bütünlüğünün daha kolay sağlanabilmesi ve ilişkisel veri modellerine göre, mevcut veri yapısında daha fazla genişleme ve yeniden düzenleme imkânlarına sahip olması sayılabilir.

Not : Bu bölüm

[http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri\(csutcu%20dr%20tezinin%20bir%20bolumu\).pdf](http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri(csutcu%20dr%20tezinin%20bir%20bolumu).pdf)
adresindeki dosyadan alınmıştır.

3. VERİ TABANI TEMEL KAVRAMLARI

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, bir çok 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.

Bu bölümde veri tabanı ile ilgili temel kavramlar üzerinde durulacaktır. Öncelikle bu kavramlar tanımlanacak ve hangi amaçla kullanıldığı anlatılacaktır. Bu kavramlarla ilgili detaylı örnek ve açıklamalar sonraki bölümlerde yapılacaktır.

3.1. 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 yığınına dönüşebilir. En basit hali ile veritabanı tasarımı; 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. Sorgular SQL; ilerleyen bölümlerde detayları ile anlatılacaktır.

3.2. 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 girişi yapılmadan önce o alanın tamsayı mı yoksa harf mi; tarih mi yoksa ondalıklı bir sayı mı olacağı tanımlanmalı ve veriler daha sonra tabloya yazılmalıdır. Ayrıca, “bir alanın uzunluğu 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. Aşağıda Ms Access, MySQL ve Oracle veri tabanı programı örnek veri tipleri verilmiştir.

3.2.1. Access Veri Tabanı Veri Tipleri

Metin : Yazılacak bilgiler harflerden veya hem harf hem de sayılardan meydana geliyorsa kullanılacak veri türüdür. Bu alana boşlukta dahil olmak üzere en fazla 255 karakter bilgi yazılabilir. Bu alana yazılan bilgiler sadece sayılardan da oluşabilir, ama yazılan sayılar hesaplama işlemlerinde kullanılamazlar.

Not : Uzun metin yada metin ve sayı bileşimi kullanılabilir. Genelde açıklama ya da uzun bir not yazılacaksa bu alan kullanılır. 64.000 karakterle sınırlıdır.

Sayı : Öğrenci numarası, öğrencinin sınıfı gibi sayısal bilgiler için kullanılır. Sayısal alanlar matematiksel hesaplamalarda kullanılabilir. Borç, alacak, öğrenci harcı gibi.

Tarih/Saat : 100 ile 9999 arasındaki yıllar için tarih ve saat değerleridir.

Para Birimi : Bir ile dört arasındaki ondalık basamağı olan, matematik hesaplamalarında kullanılan para birimi değerleri ve sayısal veriler.

Otomatik Sayı : Tabloya yeni bir kayıt eklendiğinde, Access tarafından atanan benzersiz ardışık (biri birer artan) ya da rasgele sayılar. Otomatik sayı alanları değiştirilemez.

Evet/Hayır : Yalnızca iki değerden birini içeren alanlar Evet / Hayır, Doğru / Yanlış, Açık / Kapalı gibi alanlar gibi.

OLE Nesnesi : Access tablosuna bağlanmış ya da katıştırılmış bir nesne. (Microsoft Word veya Excel çalışma sayfası gibi)

Köprü : Tıklandığında kullanıcıyı başka bir dosyaya, dosyadaki bir konuma veya Internet'teki (www) bir bölgeye yönlendiren bağlantı.

Arama Sihirbazı : Değerleri başka tablo, sorgu ya da değerler listesindeki değerlerden seçilen bir alan yaratmamıza yardımcı olan sihirbaz.

3.2.2. MySQL Veri Tabanı Veri Tipleri

MySQL'de bir çok veri türü oluşturulabilir. Ancak Web programları açısından önemli olan bir kaç ve özellikleri şöyle sıralanabilir:

INT : Tamsayı: -2147483648'den 2147483647 kadar değişen diziye "signed" (işaretli), 0'dan 4294967295'e kadar değişenine "unsigned" (işaretsiz) denir.

VARCHAR(n) : n sayısını geçmemek şartıyla değişen boyutta karakter olabilir

CHAR(n) : Kesinlikle n sayısı kadar karakter olabilir.

TEXT : En fazla 65535(2¹⁶-1) karakter alabilen metin alanı.

MEDIUMTEXT : En fazla 16777215(2²⁴-1) karakter alabilen metin alanı.

DATE : 1000-01-01'den 9999-12-31'e kadar değişebilen tarih alanı.

TIMESTAMP : 1 Ocak 1970'den 18 Ocak 2038'e kadar olan ve Yıl+Ay+Gün+Saat+Dakika+Saniye biçimindeki zaman bilgisi.

3.2.3. Oracle Veri Tabanı Veri Tipleri

CHAR(sayı): Sabit uzunluktaki alfasayısal verilerin tutulabildiği alanlar için kullanılır. Oracle 7 ve daha önceki sürümler için bu alanın uzunluğu en fazla 255 karakter olabilir. Oracle 8 ve sonrasında 2000 karakter uzunluğundadır. Eğer, sayı ile ifade edilen numaradan daha kısa uzunlukta veriler girilirse Oracle kaydın sonuna boşluk ekleyerek sabit uzunluğa kadar getirir. Örnek char(20).

VARCHAR2(sayı): Değişken uzunluklu alfasayısal verilerin tutulduğu alanlar için kullanılır. Oracle 7 ve önceki sürümlerinde 2000 karakter, Oracle 8 ve sonraki sürümlerinde 4000 karakter uzunluğunda bilgi girilebilir. Örnek varchar2(30).

NUMBER(n,p): Tamsayı ve Gerçel sayılar için kullanılan sayısal veri tipidir. Tam kısım en fazla 38 basamak olabilir. Ondalık kısmın basamak sayısı da -84 ile 127 arasında değişmektedir. Number veri tipinden türetilmiş int[eger], dec[imal], smallint ve real veri tipleri de kullanılabilir.

DATE: Tarih tutan alanlar için kullanılır. Bu tip alanlarda, tarih bilgileri ve saat bilgileri tutulabilir. Tarih formatları Oracle yüklenilirken seçilen dile göre değişir. Amerikan standardı için 'DD-MON-YY' dir. Yani bir tarih '03-MAY-01' şeklinde görünür. NLS_DATE_FORMAT parametresi ile tarih formatı değiştirilebilir. Tarihsel alanlar üzerinde aritmetiksel işlemler yapılabilir. Sistem tarihi SYSDATE fonksiyonu kullanılarak öğrenilebilir. Sayısal veya karakter olarak tanımlı bir alandaki veriler TO_DATE fonksiyonu ile tarih tipine çevrilebilir.

LONG: 2 GB 'a kadar bilgi tutabilen karakter alanlar için kullanılır. Bir tabloda bu tipten ancak bir adet alan tanımlanabilir. Long veri tipine sahip alanlar için index oluşturulamaz.

Not: Oracle'da boolean veri tipi yoktur. Bunun için char(1) ya da number(1) şeklinde tanımlama yapıp kullanılabilir.

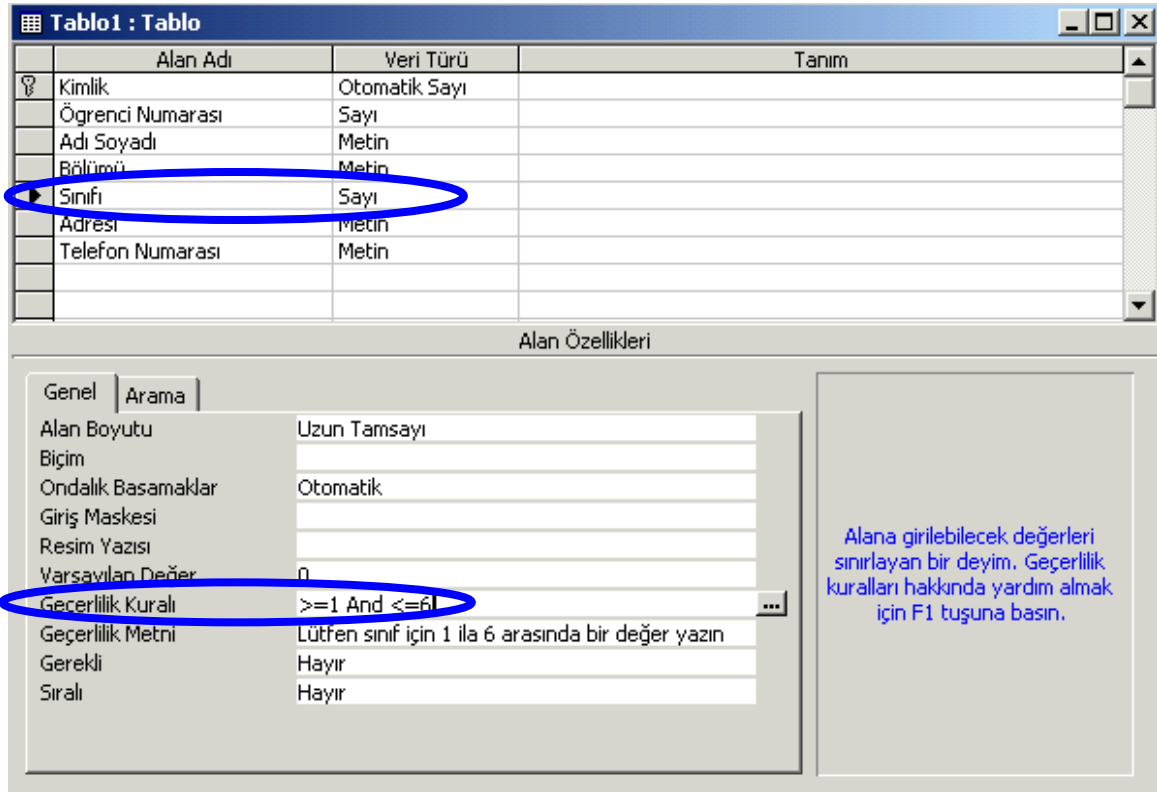
Not: Bir tablonun alanları kendi veri tipine uygun değerler alabildiği gibi bir de NULL değer alabilirler. NULL değeri sayısal olarak 0'dan ve karakter olarak ta boş karakterden(' ') farklıdır.

3.3. Zorlayıcı (Constraint)

Herhangi bir alan için girilebilecek verileri kısıtlayıcı kurallara zorlayıcılar denir. İlgili alana girilebilecek değerleri sınırlandıran 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.

Şekil 3.3.1. de mavi çizgilerle çevrelenmiş alanda Ms Access veri tabanı programında bir kısıtlayıcının tanımlanması yapılmış ve özellikle kullanıcı tarafından yanlış bilgi girişi engellenmiş ve verilerin doğru yazılması zorunlu hale getirilmiştir.



Alan Adı	Veri Türü	Tanım
Kimlik	Otomatik Sayı	
Öğrenci Numarası	Sayı	
Adı Soyadı	Metin	
Bölümü	Metin	
Sınıfı	Sayı	
Adresi	Metin	
Telefon Numarası	Metin	

Alan Özellikleri

Genel Arama

Alan Boyutu: Uzun Tamsayı

Biçim:

Ondalık Basamaklar: Otomatik

Giriş Maskesi:

Resim Yazısı:

Varsayılan Değer: 0

Geçerlilik Kuralı: >=1 And <=6

Geçerlilik Metni: Lütfen sınıf için 1 ile 6 arasında bir değer yazın

Gerekli: Hayır

Sıralı: Hayır

Alana girilebilecek değerleri sınırlandıran bir deyim. Geçerlilik kuralları hakkında yardım almak için F1 tuşuna basın.

Şekil 3.3.1. MS Access de tanımlanmış zorlayıcı ekranı

3.4. 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. Primary Key, Unique Key ve Foreign Key olmak üzere 3 çeşit anahtar vardır.

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.

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.

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.

3.5. 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.

Temelde indekslerin ilişkisel veritabanında şu üç işlevi vardır:

1. Tekil indeksler, veri ilişkilerini ve veri bütünlüğünü sağlayan birincil anahtar alanlar oluşturmada kullanılır.
2. İndeks olan alanın değerine göre bir kaydın kayıtlar arasındaki sırasını gösterirler.
3. Sorguların neticelenme sürelerini kısaltırlar.

Constraint'ler (zorlayıcılar) aslında index'lere benzerler ama indekslerden farklı olarak bir tek tablo üstünde etkili olmayabilirler. Özellikle yabancı anahtar zorlayıcısı ilişkisel veri girişi için oldukça etkili bir zorlayıcıdır. Ancak bir Foreign Key tanımlı yapabilmek için, FOREIGN KEY yabancı anahtarının asıl tablosunda birincil anahtar olması gerekir.

3.6. View (Görüntü)

Bazen, tabloları olduklarından farklı gösterecek filtreleme ihtiyacı duyulur. Bu türden işlevler için VIEW kullanılır. VIEW 'ler, saklanmış sorgulardan ibarettirler. Aslında tablo gibi kullanılsa da halihazırda 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,
- * Halihazırdaki tablolarda var olan verilerin başka bir tablo formatında sunulması gerektiğinde
 - Çok kompleks 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.

3.7. 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.

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.

1. Nesneler Tanımlanır: Nesne, çeşitli özellikleri bulunan bir varlıktır. Herhangi bir proje de öncelikle nesneler tanımlanır. Birkaç proje için nesnelere örnek verilecek olursa,

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.

2. Her nesne için bir tablo oluşturulur: Her nesne için bir tablo oluşturulur ve her bir tabloya içereceği veriyi en iyi anlatan bir isim verilir. Tablo oluşturma işi, bir kağıt 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.

3. Her bir tablo için bir anahtar alan seçilir: 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 anlatılabilirdi 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 verdiğimizizin takibini ISBN numarası ile yapmak mümkün değildir. Ancak bir E-Ticaret sitesi tasarlanırken, stoktaki tüm kitaplar birbiri ile eşdeğer olduğundan ya da öyle olduğu varsayıldığından ISBN numarası birincil anahtar alan olabilir. Bu durumda, adet diye bir niteliğin aynı tabloda yer alması gerekecektir.

4. Nesnelerin gerekli her bir özelliği için tabloya bir sütun eklenir: 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.

İPUCU : 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.

5. Tekrarlayan nesne özellikleri için ek tablolar oluşturulur : 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 hikaye kategorisine girebilir. Ya da e-ticaret sisteminde bir ürünün birden fazla reyonda yer alması gerekli olabilir. Veya bir kitap birden fazla kişi 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 düşündüğü 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 ama 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
03101001	BILGISAYAR	37	40	
03101002	INGILIZCE	56	58	
03101001	BILGISAYAR	45	48	
03101001	BILGISAYAR	69	78	

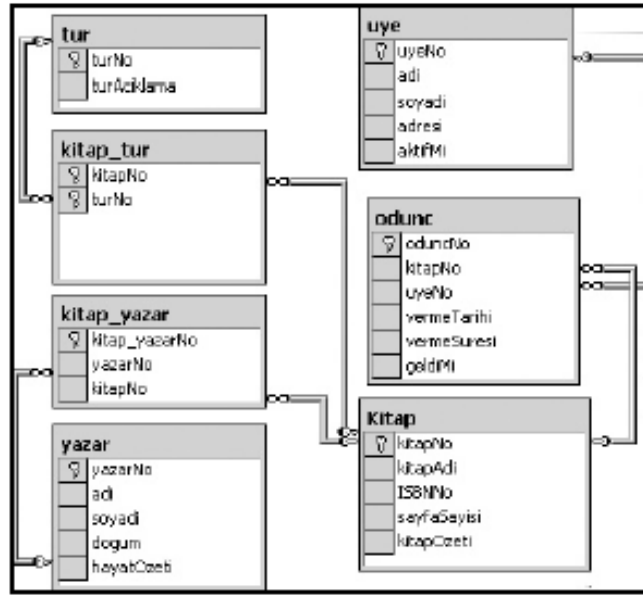
Doğru tablo tasarımı ve kayıt girişi yukarıdaki tabloda olduğu gibi olmalıdır. Burada 03101001 numaralı öğrencinin BILGISAYAR dersine ait notları bu tablodan ilerleyen bölümde anlatılan SQL cümlecği ile seçilerek bulunabilir.

6. Anahtar Alana Bağlı Olmayan Alanlar Belirlenir : 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.

7.Tablolar arasındaki ilişkiler tanımlanır : Her biri bir nesneye dair özellikleri barındıran tabloların tümü göz önüne alınır ve birbirleri ile olan ilişkileri tanımlanmaya çalışılır. Örneğin kitabı ödünç verebiliriz. Bu durumda, ödünç tablosu ile Kitap tablosu ilişkili olacaktır. Kitap üyelere ödünç verilir. Bu durumda, ödünç ile üyeler arasında da bir ilişki vardır. Türler ile Kitap arasında bir ilişki vardır, bir kitabın en az bir türe dahil olması gerekir.

Bu projedeki nesneler (tablolar) arasında ilişkiler aşağıda yer almaktadır:



Şekil 4.1.1. Örnek Projenin SQL Server 2000 deki diyagramı

İlişkili her iki tablo bir birincil alan ve bir yabancı anahtar alan üstünden birbirine bağlanır. Aynı diyagramın bir benzeri Ms Access veya Oracle veri tabanlarında da hazırlanabilir.

Farklı tablolardaki iki alan aynı veriyi tutuyorsa, iki alana da aynı adı vermek, karışıklığa yol açabilir gibi görünse de aslında daha düzgün bir yapı ortaya çıkar. KitapNo alanı kitap tablosunda da ödünç tablosunda da kitap numarasını tutmaktadır. Bu alanlardan birine KitapNo, diğerine ödünçGidenKitapNo demek, kafa karışıklığına neden olabilir. En önemlisi de her alan için her tabloda farklı isimler kullanmak değişkenlerin isminin akılda tutulmasını zorlaştıracak ve daha sonraki tablolar üzerinde işlem yaparken işlemleri zorlaştıracaktır. Her seferinde ilgili alanın hangi isimle kaydedildiğine bir listeden bakmak zorunda kalınacaktır. Çünkü büyük bir veritabanı projesinde 250 den fazla tablo bulunabilir. Her tabloda da bir çok 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 KitapNo demektir.

4.2. Veri Tabanı Normalizasyonu

Aslında ilişkisel veri tabanı tasarımından ziyade, bir tablo içerisinde yer alacak kaydın nelerden oluşmasına karar vermeye yarayan normalizasyon kuralları başlı başına bir işlemdir. Normalizasyon; veritabanı tasarım aşamasında gerekli bir işlem olduğundan bu bölümde incelenecektir. Genel kabul görmüş 5 normalizasyon kuralı vardır. Burada her bir kuralı tam olarak anlatmak mümkün değildir. Ancak bu kurallar, ilişkisel veritabanının tanımı ile birlikte ortaya konulmuştur. Özet olarak fikri vermesi açısından normalizasyon kurallarına aşağıda yer verilmiştir.

1. Normalizasyon Kuralı :

Bir satırdaki bir alan yalnızca bir tek bilgi içerebilir. Birden fazla yazarı olan kitap için yazar1, yazar2 ve yazar3 diye alanların açılması ile bu kurala uyulmamış olunur. Böyle bir durumda, ayrıca yazarlar tablosu da oluşturularak kural çiğnenmemiş olur.

Veri tabanı tasarımında; verileri virgül veya bir başka karakter ile ayrılıp aynı alana girilmesi ve daha sonra program içerisinde split ile bu değerlerin ayrılması genellikle sık yapılan hatalardan birisidir. Ancak bu ilişkisel veritabanının doğasına terstir. Bunun yapılmaması gerekir.

2. Normalizasyon Kuralı:

Bir tablo için, anahtar olmayan her alan, birincil anahtar olarak tanımlı tüm alanlara bağlı olmak zorundadır. Örneğin, Ödünç tablosuna KitapAdı diye bir alan eklense idi, bu sadece ödünç verilen kitap ile ilgili bir bilgi olacaktı ve oduncNo 'na bağlı bir nitelik olmayacaktı. Bunu çözmek için, kitap adları ayrı bir tabloda tutularak 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.

3. Normalizasyon Kuralı:

Bir tablo için, anahtarı olmayan bir alan, anahtarı olmayan başka hiç bir alana bağlı olamaz. Örneğin, kitaplar için cilt tipi adında bir alan eklenip burada da karton kapak için K, deri cilt için D, spiral cilt için S yazılıyorsa, bu kodlama, kitap tablosunun birincil anahtarı olan kitapNo alanına bağlı bir kodlama olamazdı. Çünkü bu kodlama bir başka anahtarı olmayan alana bağlıdır. Bunun sonucunda da veritabanında, karşılığı olmayan bir kodlama yer almış olurdu. Cilt tipi bilgisini kodlu olarak tutan alan aslında cilt tipi açıklaması olan başka bir alana bağlıdır. Bu ilişki başka bir tabloda tutulmalıdır. Bu durumda, cilt şekillerini tutan bir tablo açılması gerekir. Bu tablonun alanları da ciltTipKodu ve ciltSekli olabilir. Ancak bundan sonra, kitaplar tablosunda ciltTipi adında bir sütun açıp buraya da D,S,K gibi kodlar yazılabilir.

4. Normalizasyon Kuralı:

Birincil anahtar alanlar ile anahtarı olmayan alanlar arasında, birden fazla bağımsız bire-çok ilişkisine izin verilmez. Örneğin, tabloda yer alan bir kitap, hem hikaye kitabı hem de kişisel gelişim kitabı olabilir. (Bu durumda kitabın adı, kişisel gelişim hikayeleri olurdu her halde) Bu durum Kitap tablosunda nasıl ifade edilebilir?

4.Normal formu sağlamak için, her bağımsız bire çok ilişki için ayrı bir tablo oluşturulması gerekir. Bu örnekte, türler için yeni bir tablo açılması gerekir. Tablonun adına türler denilebilir. Daha sonra kitapTurleri diye bir başka tablo daha açılması gerekir. 'Kişisel Gelişim Hikayeleri' adlı kitap için, öncelikle kitap numarası, Hikaye bölümünün kodunun yer aldığı bir satır; ardından da yine kitap numarası, ardından da kişisel gelişim türünün kodunun aldığı yeni bir satırın daha eklenmesi gerekir.

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, kitaplar için bir edinme şekli bilgisi girilecek sütun eklenmek istenebilir: Bu bölüme girilebilecek bilgiler bellidir: Bağış veya satın alma.

Bu bilgiler başka bir tabloda tutulabilir. Böylelikle, kullanıcıların bu alana geliş 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. Özetle ilişkisel bir veritabanı tasarımı şu dört ögeyi barındırmalıdır.

1. Veri tekrarı yapılmamalıdır.
2. Boş yer mümkün olduğunca az olmalıdır.
3. Veri bütünlüğü sağlanmalıdır.
4. Veriler, aralarında bir ilişki tanımlanmaya müsait olmalıdır.

4.3. İlişkisel Veri Tabanı Yönetim Sistemleri

Veritabanı Yönetim sistemlerinden günümüzde kullanımı en yaygın olan ilişkisel veritabanıdır ve en yaygın veritabanı yönetim sistemleri, ilişkisel Veritabanı Yönetim Sistemleri ' (VTYS) dir. İlişkisel veritabanının en önemli yanı, tablolardan oluşmasıdır. Daha önemli yanı da bu tabloların birbiri ile ilişkilerinin olmasıdır. VTYS 'lere "ilişkisel" denmesinin anlamı budur.

Bir veritabanında ilişkiden söz edebilmek için en az iki tablonun yer alması gerekir ve bu iki tablodaki verilerin birbiri ile bir şekilde ilişkilendiriliyor olması gerekir. Yine bir önceki örnek olaya dönecek olursak, Kitap listesi ile ödünçler listesi arasında bir ilişki vardır. Çünkü Kitap listesinde olmayan bir kitap bizde yoktur ve ödünç verilemez. Haliyle de mantık olarak bu türden bir ödünç bilgisi ödünç listesinde yer almamalıdır. Olaya tersten bakılacak olursa, geri dönmeyen bir kitap hakkındaki detaylar öğrenilmek istenildiğinde ödünç listesindeki kitap numarası alınır. Daha sonra aynı numaraya karşılık gelen kitap, Kitap tablosundaki satırda bulunur. Bu satırdaki bilgiler, bize kitap hakkındaki tüm detayları verir.

Kitap tablosundaki kitapNo alanı aday anahtar (indeks)'tir. Odunc tablosundaki KitapNo alanı, 'yabancı anahtar ' (foreign key) alanıdır, çünkü Kitap tablosundaki bir kaydı sembolize etmektedir. Tüm bunların ardından VTYS 'leri hakkında özet olarak diyebiliriz ki;

Bir İlişkisel Veritabanı Yönetim Sistemi tablolar üstünde şu üç işlevi yerine getirmek zorundadır.

1.Seçme : Herhangi bir tabloda (listede) yer alan tüm bilgileri gösterebilmelidir. Örneğin, Kitap tablosunun bir dökümünü verebilmelidir ya da kitap listesinden bazı kitapların bilgilerini getirip diğer bir kısmını getirmeyebilmelidir.

2.İzdüşürme : Herhangi bir tablodan sadece belli sütunların yer aldığı seçme işlevlerini yerine getirebilmelidir. Örneğin, canı isteyen bir kullanıcı kitabın sadece adını ve kaç sayfa olduğunu seçebilmelidir.

3.Birleştirme : Birden fazla tabloda yer alan bilgiler, yeri geldiğinde tek bir tabloymuş gibi sunulabilmelidir. Örneğin, ödünç alınıp da geri getirilmeyen kitapların adları ve kimler tarafından alındığı bir tek tabloymuş gibi gösterilebilmelidir.

VTYS bu 3 temel işlevi yerine getirebilmelidir. Bunlardan üçü, ikisi veya biri aynı anda yerine getirilmek durumunda kalınabilir. Örneğin, sayfa sayısı 200 'den büyük kitapların sadece ismi görülmek istenirse, hem izdüşürme hem de seçme işlemine ihtiyaç duyulur. Veriler ve depolanma şekilleri farklı olabilir. Önemli olan, VTYS'nin SQL ile yönetilebilir olmasıdır. Böylelikle, verilerin bilgisayarda fiziksel olarak ne şekilde depolandığı, kullanıcı bilmek zorunda değildir.

Yani, kullanıcı temel veri saklama işlem ve yöntemlerinden izole edilmiş olur. Kullanıcının verileri etkili olarak kullanması için bilmesi gereken tek şey SQL olmalıdır. SQL ile ilgili bölümlerde anlatılmıştır.

5. ÖRNEK BİR VERİ TABANI TASARIMI ve NORMALİZASYONU

Bir veri tabanı ile proje yapılırken işin en önemli aşamasının veri tabanının tasarlanması olduğundan ve bunu yaparken hangi kurallara göre yapılacağından bahsedilmişti. 4.bölümde veri tabanı tasarımında tablolar ve sütunlardan bahsedilmiş, normalizasyon ile de tablolara kaydedilecek bilgilerin özelliklerinden ve hazırlanma kurallarından bahsedildi.

Bu ders notu kapsamında veri tabanı yönetim sistemleri ve SQL komutları anlatılmaktadır. Bu bölümde örnek bir veri tabanı tasarlanacak ve veri tabanı normalizasyon işlemi yapılacaktır. Tanımlanan ve veri girişi yapılan bu veri tabanı ilerleyen bölümlerde kullanılacak ve örneklerle konuların daha iyi anlaşılması sağlanacaktır.

Örnek proje bir işyerinde çalışan personel ve personele ait meslek, çalışılan birim ve maaş işlemlerini kapsamaktadır. Burada personel projesindeki işlere ait bir kısım işlemler örnek olarak verilecektir. Unutulmamalıdır ki bir işyerindeki personel işlemleri çok daha fazla tablo, sütun ve işlemlerden meydana gelmektedir. Burada konunun anlaşılabilmesi açısından projenin bir kısmı üzerinde çalışılacaktır.

5.1. Örnek Personel Projesi Veri Tabanı Tasarımı

Örnek projemizin adı personel işleri ve projemiz 4 adet tablodan meydana gelmektedir. Bunlar; Personel Bilgileri, Personelin Çalıştığı Birimler, Personele Ait Meslekler ve Personele Ait Maaş bilgilerinin tutulduğu tablolar. Bu tablolar ve bu tablolara ait alan (sütun) bilgileri aşağıda listelendiği gibi düzenlenmiştir.

TabloAdı	TabloAlanları
Personel	SicilNo,Adı,Soyadı,MeslekKodu, BirimKodu, DoğumTarihi, Adres
Meslekler	MeslekKodu, MeslekAdı
ÇalışılanBirim	BirimKodu, BirimAdı
Maaslar	SicilNo, MaasAyı, Maas

Veritabanlarında tablolar matris yapısında tutulurlar. Sütunlar tablo alanlarını, satırlar ise tablodaki kayıtları (record) teşkil ederler. Ders notlarının ilerleyen bölümlerinde tablolar matris yapısında görüntüleneceklerdir.

Burada normalizasyon kurallarına göre bilgiler ayrı tablolarda tutulmuşlardır. Meslekler ve ÇalışılanBirim için iki ayrı tablo tanımlanmıştır. Bunun nedeni personel tablosunda meslek ve çalışılan birim alanına bilgi yazılırken veri bütünlüğü ve standart sağlanabilmesi içindir. Bu tür bilgi girişlerinde ayrı tabloların tanımlanması gerektiği 3.normalizasyon kuralında söylenmişti. Personel tablosunda kişiye ait meslek bilgisi de veri bütünlüğü ve standardın sağlanması açısından kodlanarak yazılacaktır. Veri tabanında bu kural çok önemlidir.

Meslekler personel tablosuna yazılırken kod şeklinde tanımlanmayıp; personel tablosunda isim olarak yazılıyorsa bilgilerde sorun yaşanır. Örneğin : “Müdür yardımcısı” mesleği yazılırken kullanıcı bu bilgiyi kendine göre kısaltarak veya büyük küçük harf dikkate almayarak yazabilir. Bir kişi için “Müd.Yar”, yazarken bir diğeri için “M.Yard.” bir diğeri için “Müdür Yardımcısı” veya “MÜDÜR YARDIMCISI” yazabilir. Veri tabanında bu 4 bilgi birbirinden farklı bilgilerdir. Daha sonra “Müdür Yardımcısı” olan kişilerin sorgusu yapılacak olursa bu 4 bilgi birbirinden farklı olacağı için doğru bir sonuç elde edilemeyecektir. Halbuki bu alana müdür yardımcısı için 2 gibi bir kodlama yapılır ve bu alana 2 bilgisi yazılırsa mesleği 2 olanların sorgusu için elde edilecek veriler doğru olacak ve ekrana 4 kişiye ait bilgi gelecektir. Bu nedenle Meslekler tablosu ayrı yaratılmalıdır.

Aynı şey çalışanBirim içinde yapılmış ve bu alan da kodlanmıştır. Örneğin : çalıştığı birim “Bilgi İşlem” olan personele ait bilgiler tabloya yazılırken kullanıcı bunu da kendine göre kısaltarak veya büyük küçük harf dikkate almayarak yazabilir. Bir kişi için “Bilgi İşlem”, yazarken bir diğeri için “B.İşlem” bir diğeri için “BİLGİ İŞLEM” veya “Bil.İşl.” yazabilir. Veri tabanında bu 4 bilgi birbirinden farklı bilgilerdir. Daha sonra “Bilgi İşlem” de çalışan kişilerin sorgusu yapılacak olursa bu 4 bilgi birbirinden farklı olacağı için doğru bir sonuç elde edilemeyecektir. Halbuki bu alana Bilgi İşlem için 3 gibi bir kodlama yapılır ve bu alana 3 bilgisi yazılırsa çalıştığı birim 3 olanların sorgusu için elde edilecek veriler doğru olacak ve ekrana 4 kişiye ait bilgi gelecektir.

Meslek ve çalışan birim için ayrı tablo tanımlanmasa ve bu alanlar kodlanmadan personel tablosu alanı içerisine dahil edilmiş olsaydı tablo yapısı aşağıdaki gibi olurdu.

Personel : Tablo							
	SicilNo	Adı	Soyadı	MeslekAdı	BirimAdı	DogumTarihi	Adres
	876	Ali	ÖZDEMİR	Müdür Yardımcısı	Bilgi İşlem	23.03.1968	Mersin
	765	Neşe	KÖKSAL	Müd.Yard.	Bil.İşl.	12.12.1972	Mersin
	231	Ahmet	KILIÇ	MÜDÜR YARDIMCISI	BİLGİ İŞLEM	03.10.1965	Mersin
	128	Handan	BİLİR	müdür yar.	B.İşlem	04.08.1958	Mersin

Şekil 5.1.1. Personel tablosu için ayrı tablo tanımlanmamış hali

Personel : Tablo							
	SicilNo	Adı	Soyadı	MeslekKodu	BirimKodu	DogumTarihi	Adres
	876	Ali	ÖZDEMİR	2	3	23.03.1968	Mersin
	765	Neşe	KÖKSAL	2	3	12.12.1972	Mersin
	231	Ahmet	KILIÇ	2	3	03.10.1965	Mersin
	128	Handan	BİLİR	2	3	04.08.1958	Mersin

Şekil 5.1.2. Personel tablosu için ayrı tablo tanımlanmış hali

Bir diğer özellikte maaş tablosunda maaşAyı bilgisinin olmasıdır. Bu bilgi maaşın hangi ay için ödendiği bilgisidir. Bir personel bir yılda birden fazla maaş alacağına göre ilgili maaşının hangi aya ait olduğu bilgisi bu alanda tutulacaktır. Bu alandan; istenilen personele ait birden fazla ay maaş bilgisi elde edilebilir. Örneğin; personelin ilk 6 ay aldığı toplam maaş miktarı bulunabilir. Ya da bir yılda 10 milyardan fazla maaş alan personel listesi elde edilebilir. Bu sorgular SQL dili ile yapılacaktır.

Buradan şu sonuç çıkarılmalıdır : Veritabanında meslek, çalışılan birim gibi birçok bilgi için standart olan bilgi girişi 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. Bir önceki bölümde anlatılan 3.Normalizasyon kuralındaki kitap için cilt tipi alanı gibi. Cilt tipinde kitap için K, deri cilt için D ve spiral cilt için S kodlamasının kullanılması gibi. Dikkat edilmesi gereken nokta kodlamanın kısa karakterlerden oluşmasıdır.

Ders notlarında; tablolar ve tablo alanları **tabloAdı.TabloAlanı** şeklinde ifade edilecektir. Örnek; personel tablosundaki SicilNo alanı **Personel.SicilNo** gibi.

Buradaki bir diğer noktada farklı tablolardaki aynı amaç için kullanılan alanların aynı isimle isimlendirilmesidir. Bu konu ile ilgili bilgi “Farklı tablolardaki iki alan aynı veriyi tutuyorsa, iki alana da aynı adı vermek, karışıklığa yol açabilir gibi görünse de aslında daha düzgün bir yapı ortaya çıkar.” İfadesiyle daha önce açıklanmıştır. Personel tablosundaki SicilNo ile Maaslar tablosundaki SicilNo aynı isim ve aynı amaçla kullanılmıştır. Personel tablosundaki MeslekKodu ile Meslekler tablosundaki MeslekKodu aynı isim ve aynı amaçla kullanılmıştır. Yine Personel tablosundaki BirimKodu ile ÇalışılanBirim tablosundaki BirimKodu aynı isim ve aynı amaçla kullanılmıştır.

Aşağıda projemizdeki tablolar arasındaki ilişki (bağlantı) verilmiştir.

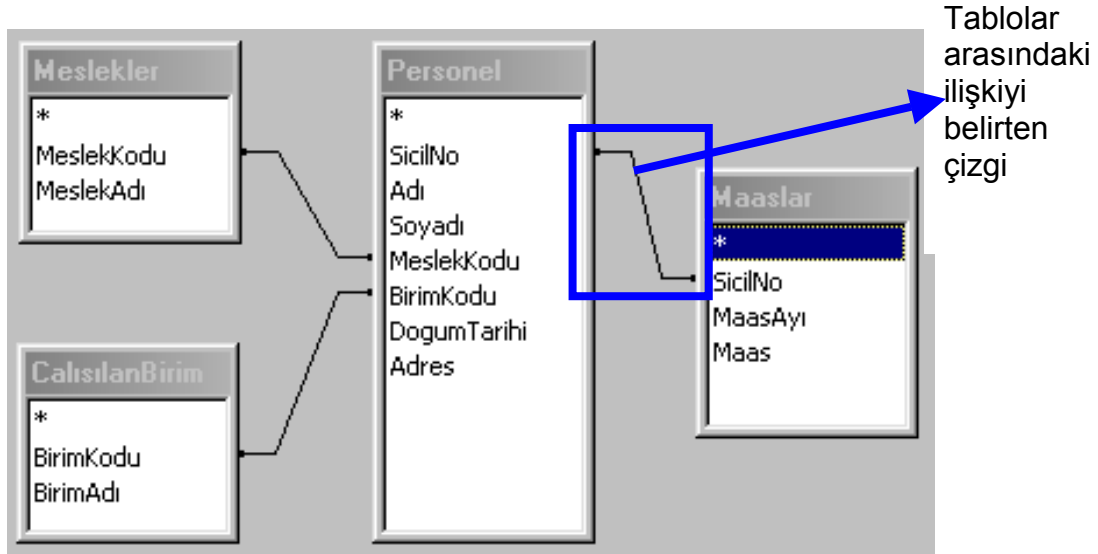
<u>Tablo Adı.Alanı</u>	<u>İlişkili TabloAdı.Alanı</u>
Personel.SicilNo	Maaslar.SicilNo
Personel.MeslekKodu	Meslekler.MeslekKodu
Personel.BirimKodu	CalisilanBirim.BirimKodu

Projedeki bir diğer tanımlanması gereken nokta da tablo için tanımlanan alanların (sütunların) hangi veri tipinden ve kaç karakterden oluşacağıdır. Sicil Numarasının rakamlardan, ad ve soyadın harflerden, doğum tarihinin tarih bilgisinden, adresin hem rakam hem de harflerden oluşacağının tanımlanması gibi.

Bu tanımlamaların kullanılan veri tabanına göre yapılması gerektiği ve her veritabanında bu tanımlamaların farklı olduğu 3.bölümde anlatılmış ve MS Access, MySql ve Oracle veri tabanı veri tiplerinden örnekler verilerek anlatılmıştır.

Örnek veri tabanındaki tablolar, tablo alanları ve veri tipleri aşağıdaki gibidir. Bu tabloların nasıl yaratıldığı ve tablolar üzerinde hangi işlemlerin ve nasıl yapılacağı sonraki bölümlerde anlatılmıştır.

TabloAdı	TabloAlanları	Veri Tipi
Personel	SicilNo, Adı, Soyadı, MeslekKodu, BirimKodu, DoğumTarihi, Adres	Sayı(4) Karakter(15) Karakter(15) Sayı(3) Sayı(3) Tarih Karakter(25)
Meslekler	MeslekKodu, MeslekAdı	Sayı(3) Karakter(20)
ÇalışılanBirim	BirimKodu, BirimAdı	Sayı(3) Karakter(20)
Maaslar	SicilNo, MaasAyı, Maas	Sayı(4) Sayı(2) Karakter(12)



Şekil 5.1.3. Örnek Personel Projenin MS Access deki diyagramı

Dört tablo ve tablolar arasındaki ilişkiler yukarıdaki şekilde gösterilmiştir. Mavi çizgi ile tablolar arasındaki ilişkiler gösterilmiştir. Aynı diyagramın bir benzeri SQL-Server veya Oracle veri tabanlarında da hazırlanabilir.

Bundan sonraki bölümlerde bu tablolar, tablo alanları, veri tipleri ve ilişkileri üzerinde işlem yapılacaktır.

6. SQL VERİ İŞLEME DİLİ

SQL insanların veritabanı sistemleri ile konuşmasını sağlayan popüler bir dildir. Bu dil tüm veritabanı programlarında kullanılabilir. Bu dil sayesinde, bir veritabanından kayıtlar alınabilir, değiştirilebilir ya da yeni kayıtlar eklenebilir. SQL bir dildir; ancak bir programlama dili değildir. Program geliştirme aşamasında SQL 'den faydalanılır, ancak tek başına bu iş için yeterli değildir. PHP, Asp, Visual Basic, Delphi, C, c++ gibi bir çok programlama dili SQL komutlarını desteklemektedirler. Yani bir program geliştirme aşamasında; SQL komutlarını bilmek gerekmektedir ama SQL tek başına bir programlama dili olmadığı için ayrıca bir programlama diline de ihtiyaç duyulmaktadır.

1983 'lü yıllarda IBM laboratuvarlarında yapılan çalışmalarda SQL (Structural Query Language) standartları tanımlanmış ve ardından 1987 de ISO ardından da ANSI tarafından bir standart olarak kabul edilmiştir. Daha sonra, bu standartlar çerçevesinde bir çok veritabanı yönetim sistemleri geliştirilmiştir. Bunlardan belli başlıları, Oracle, Sybase, MS SQL Server, Informix ve MySQL 'dir. Bu VTYS 'lerin işlerin daha kolay yürümesi için kendi adlarına standart dilden uzaklaşan tarafları vardır. Ancak genel işlemlerde kullanılan dil tüm veri tabanı programları için de ortaktır ve bu SQL 'dir. VTYS'de saklanan veriler SQL komutları ile insanların istekleri çerçevesinde işler ve yeniden şekillendirilir.

6.1. SQL Nedir?

Açılımı “Structured Query Language” yani “Yapısal Sorgulama Dili” olan SQL, veritabanı işlemleri ile ilgili komutlardan oluşan bir dildir. Bu dil ile veri tabanı üzerinde; veritabanının kendisini oluşturmak, tablo, indeks, kullanıcı oluşturmak gibi komutlar ve kayıt ekleme, silme, düzeltme gibi işlemler yapılabilir. SQL dilindeki komutlar Pascal, C, Visual Basic, Delphi ve benzeri dillerdeki fonksiyon ve prosedür oluşturarak bir program yazmaktan biraz farklıdır. Yani kullanıcı SQL kullanırken fonksiyon ve prosedür yazamaz. Yine SQL kullanımında şartlı ifadeler ve dallanmalar bulunmaz. Yani kullanıcı diğer programlama dillerindeki İf, Case, next, do gibi ifadeler kullanamaz. Şartlı ifadeler, döngüler, karşılaştırmalar SQL 'de bulunmaz.

SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmaların eksikliğini giderebilmek için; Oracle PL/SQL (Programming Language/SQL), MS SQLServer ve Sybase 'de T-SQL dilini geliştirilmiştir. Oracle tarafından kullanılan PL/SQL komutları ile T-SQL bir çok noktada hemen hemen aynıdır ve SQL'de kullanılamayan procedür, fonksiyon, şartlı ifadeler, döngüler ve karşılaştırmalar kullanılabilir, PL/SQL ve T-SQL if,case,for..next gibi programlama için gereken işlemleri kullanmayı olanaklı kılar.

Görüldüğü gibi SQL karar yapıları, döngüler ve benzeri gibi bir programlama diline özgü yetilerden yoksundur. Ancak bir çok VTYS'de bu yetiler Transact-SQL veya PL/SQL gibi dil tanımları ile bir noktaya kadar desteklenmiştir. Verilerin hacmi arttıkça daha gelişmiş VTYS'lere ihtiyaç duyulur.

PL/SQL ve T-SQL sadece içerisinde SQL komutları kullanılabilen bir dildir. Yani SQL'in yapısını değiştirmemiştir. Komut modunda yazılan bir SQL cümlesi alınıp PL/SQL ve T-SQL blokları arasına yazılabilir. Veri tabanı programları tüm uygulamalarda SQL kullanmayı esas almıştır. Kullanıcı veri tabanı ürünlerini kullanarak yaptığı tüm işlemlerin arkasında SQL komutlarını çalıştırır. SQL'in veritabanı işlemleri için kullanılan komutları 5 kategoride toplanabilir:

- Veri sorgulama komutları
- Tabloya veri ekleme,değiştirme ve silme komutları
- Veritabanı nesneleri oluşturma,değiştirme ve silme komutları
- Veritabanına ve veritabanı nesnelere erişimi kontrol etme komutları
- Veritabanının tutarlılığını ve bütünlüğünü koruma komutları

SQL(Structred Query Language) ile bir çok yerde kastedilen şey, ANSI 'nin 1992 yılında yayınladığı standarttır. Ancak SQL bir programlama dili değildir. Bir kullanıcı arayüzü tanımlayamaz ya da bir dosya yönetimi yapamaz. Temelde 3 alt ifade grubundan oluşur.

1. Veri Tanımlama Dili : (Data Defination Language = DDL) : Bu gruptaki komutlar kullanılarak, tablo, trigger, view gibi veritabanı nesneleri tanımlanır. Üç temel ifade, CREATE ile bir nesne tanımlanır, ALTER ile nesne üstünde değişiklik yapılır ve DROP ifadesi ile bir nesne silinebilir.

2. Veri İşleme Dili : (Data Manuplation Language = DML) : Veri üstünde düzenlemeler yapılır. Bu alt dil bir tabloya veri ekleme(INSERT), silme(DELETE) ve güncelleme(UPDATE) yapmanın yanı sıra verileri seçmek ve raporlamak için SELECT ifadesi ve SELECT ifadesi ile birlikte kullanılan, INTO, FROM, WHERE, LIKE, GROUP BY, ORDER BY, HAVING... gibi çok yan ifade vardır. Bu ifadelere ait örnekleri bir çok yerde bulmak mümkündür.

3. Veri Kontrol Dili : Temel 2 ifadeden oluşur. VTYS'de tanımlı Roller ve kullanıcılar için ifade ve nesne kullanma izni tanımlar. Erişim(GRANT) ve erişim kaldırma(REVOKE) ifadeleri ile bu haklar ayarlanır. Oracle, SQL Server 2000 gibi VTYS'lerinde bunlara ek olarak erişim engelleme(DENY) ifadesi de yer almaktadır.

6.2. Veri Tabanı Programlarında ve SQL de Değişken Tanımlama

SQL 'de tablo adları, alan(field), veritabanı dosyası, indeks vb. isimler değişken isimleridir. Genel değişken isimlendirme kuralları burada da geçerlidir.

1. Değişken isimleri,harf ile başlamak zorundadır.
2. Değişken isimleri,harf,rakamlar ve '_' dan oluşmak zorundadır.

3. Değişken isimlerinde Türkçe noktalı harflerin (İ,ı,Ğ,ğ,Ü,ü,Ş,ş,Ç,ç,Ö,ö) ilerde dil problemi yaşanmaması açısından kullanılmaması gerekir.
4. Komut olarak ayrılmış kelimeler değişken adı olamazlar (select, like, not, or, delete, update vs.)
5. SQL büyük-küçük harf duyarlı değildir.
6. Değişken isimlerinde boşluk yer alamaz.

Değişken isimlendirme notasyonları:

1. **Deve notasyonu** : degiskenAdi şeklinde yazılır.
2. **Alt çizgi notasyonu** : degisken_adi şeklinde yazılır.

Veritabanı programlamada, büyük-küçük harf duyarlılığı olmadığından genellikle alt çizgi notasyonu kullanılır ve değişken adları küçük harf olarak verilir. Ancak bu bir kural olmayıp sadece okunurluğu artırmak için programcıların bir çoğu tarafından tercih edilen bir yoldur.

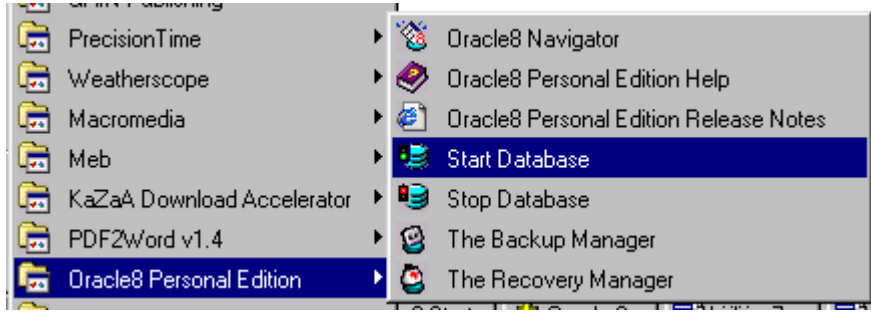
NULL mu, boşluk mu? Bir kayıt için, alanlardan biri hiç girilmediği için boş olabilir veya bilgisayardaki space tuşunun karşılığı ASCII değeri girilmiş olabilir. Space (ASCII-32 karakteri) tuşuna basılarak elde edilmiş boşluk ile daha hiçbir bilgi girilmemiş olan boşluk bilgisayar dilinde birbirinden farklıdır. Daha önce hiçbir şey girilmemiş alan için NULL terimi kullanılır.

6.3. SQL Programı Çalıştırılması ve Yazım Kuralları

SQL programını çalıştırabilmek için bilgisayara veri tabanı programı ve daha sonra bu veritabanı ile bağlantı sağlayacak özel olarak bir SQL programı yüklenmelidir. Oracle, Sybase gibi büyük veri tabanı programları client / Server (istemci / Sunucu) yapıda çalıştıkları için server üzerine yüklenen veritabanı programlarından başka kişisel çalışmalar için özel PC programları da vardır. Veritabanı programı yüklenirken SQL 'i de yükleme seçenekleri vardır. Bilgisayara buradan veri tabanı programı ile SQL programı aynı anda yüklenebilir. Genelde izlenen metot budur.

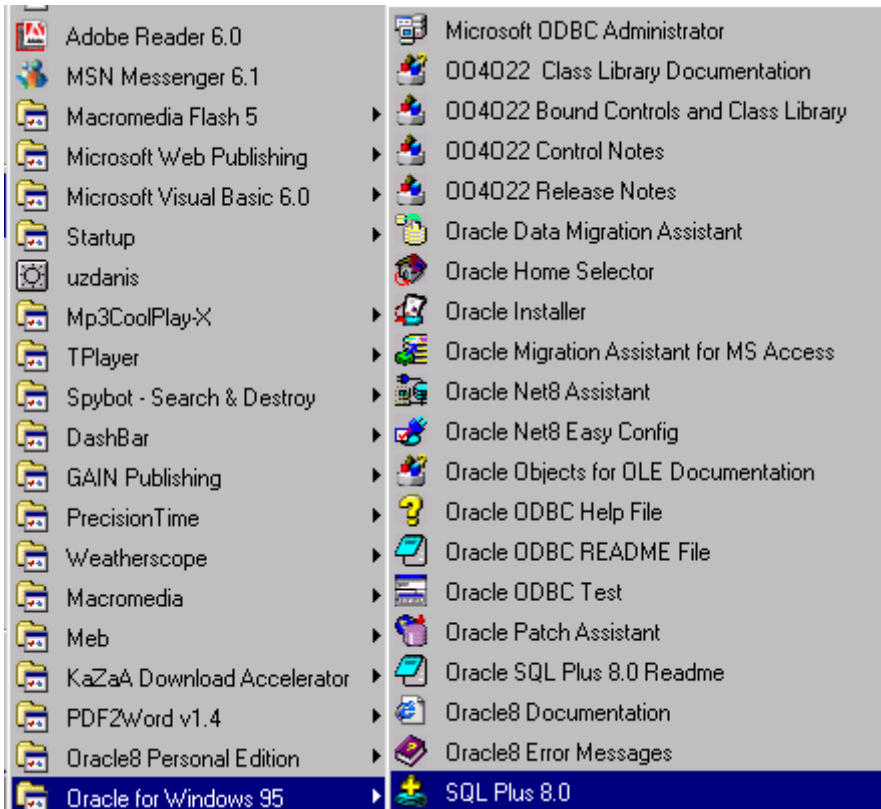
Bu ders notu kapsamında örnekler ORACLE veri tabanında hazırlanmış ve SQL Oracle veri tabanı ile kullanılmıştır. Bilgisayara Personel Oracle 8 ve SQL yüklenmiştir. Buradaki 8 rakamı veri tabanının versiyon numarasıdır. Oracle şu anda 11. versiyonunu piyasaya sürmüştür. Ders notları kapsamında Oracle 8 ile çalışacağız. Veritabanının başlatılması ve SQL 'i öğrenmek açısından pek bir değişiklik olmayan bu versiyon kullanılmıştır. Oracle programının bilgisayara yüklenme işleminden sonra programlar kısmında iki adet alt seçenek oluşur. Bunlar :

1. **“Oracle8 Personel Edition”** ve onun alt seçenekleri : Burada veritabanı ve yönetimi işlemleri yapılabilir. Veri tabanının başlatılması, durdurulması, yedeğinin alınması, veri tabanındaki bozulmaların düzeltilmesi, veri tabanı yönetim ekranı gibi.



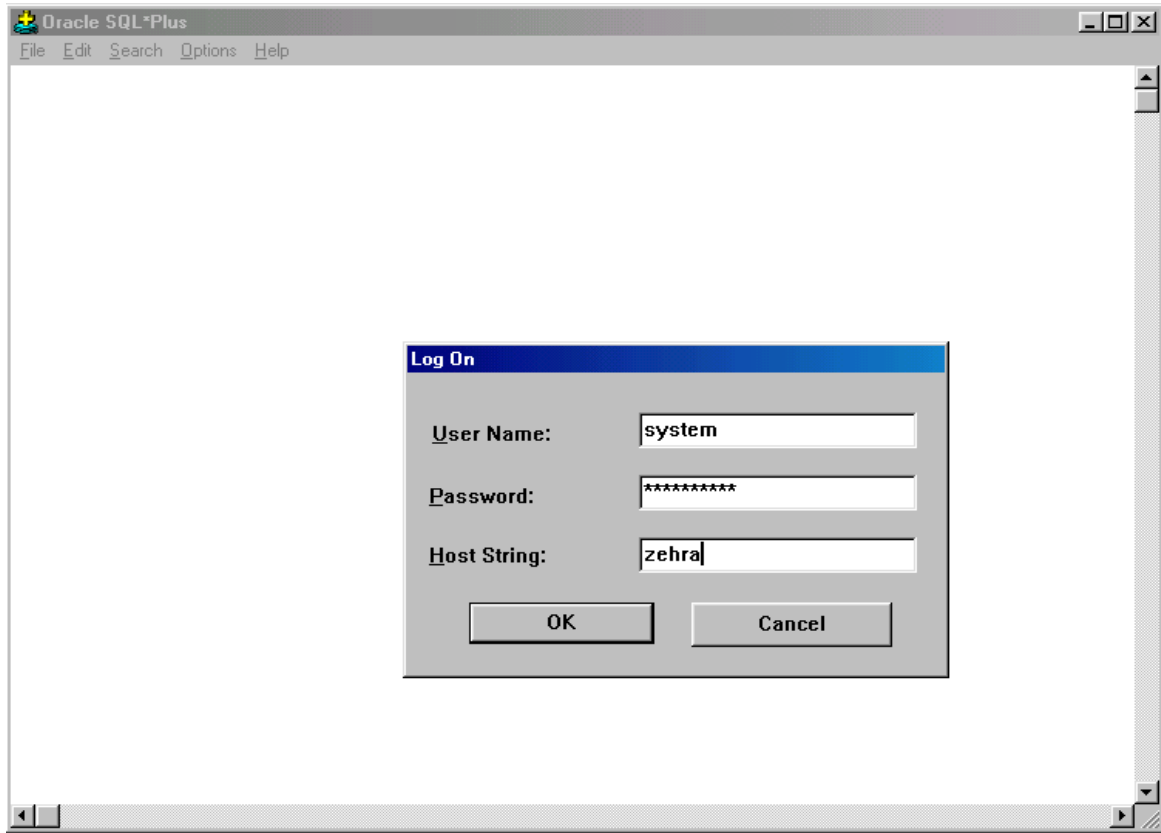
Şekil 6.3.1. Oracle8 Personal Edition Ekranı

2. **“Oracle for Windows95”** ve onun alt seçenekleri : Burada veritabanına farklı metotlarla bağlanmak için gereken ayarlamaların yapılabileceği programlar, yardım dosyaları ve SQL ile ilgili programlar bulunur. SQL Plus 8.0 programı SQL yazmak için kullanacağımız program olacaktır. 8.0 programın versiyonudur. SQL ile çalışmaya başlanmadan önce veritabanı 1.seçenekten başlatılmalıdır.

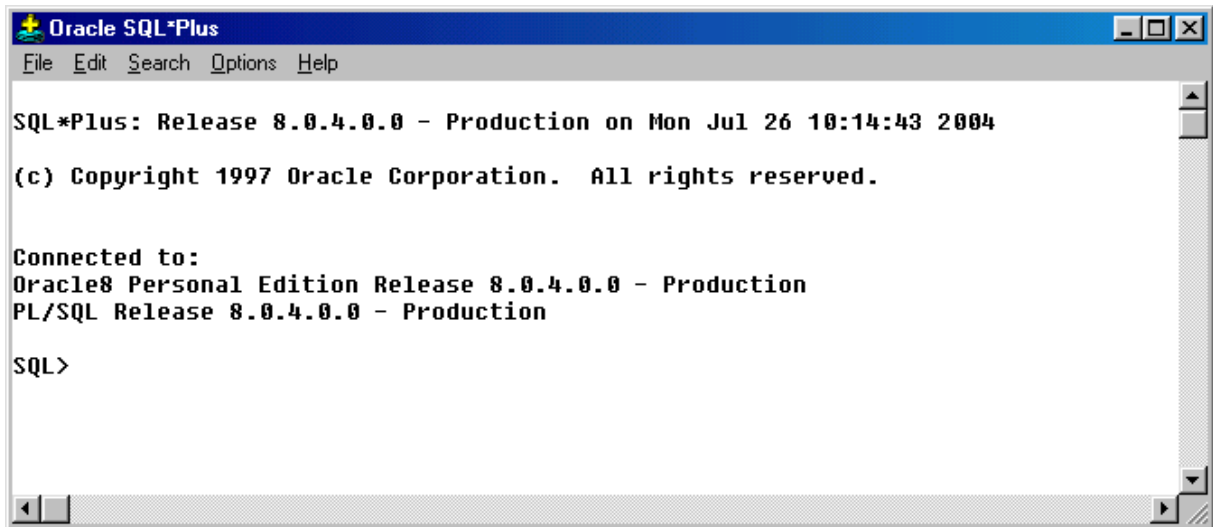


Şekil 6.3.2. Oracle for Windows95 Ekranı

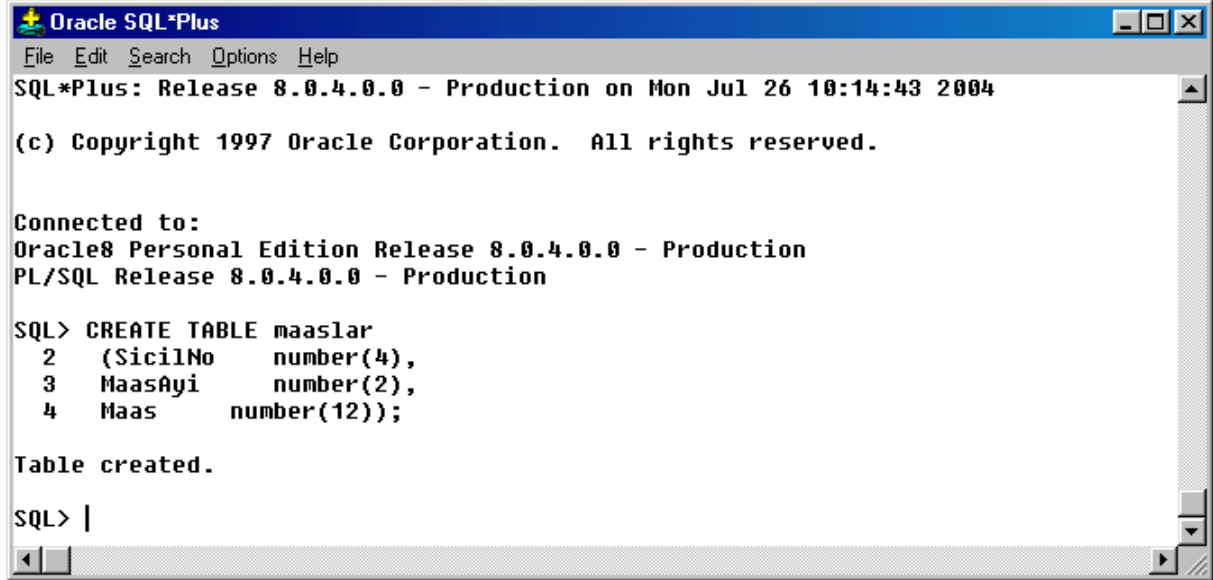
SQL çalışmak için Bu ekrandan SQL Plus 8.0 çalıştırıldığında ekrana

**Şekil 6.3.3.** SQL Plus 8.0 Başlangıç Ekranı

Bu ekranda ilgili alanlara ait bilgiler yazıldıktan sonra OK tuşuna basıldığında SQL çalışma ekranı karşımıza gelir. Artık Veri tabanına bağlantı sağlanmıştır ve SQL cümlecikleri yazılarak çalıştırılabilir. Bu alanlara nelerin yazılacağı Veri Tabanı Yöneticisi tarafından ayarlanmaktadır.

**Şekil 6.3.4.** SQL Plus 8.0 Çalışma Ekranı

Bu ekranda SQL> yazısının hemen yanına komutlar yazılır.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL*Plus: Release 8.0.4.0.0 - Production on Mon Jul 26 10:14:43 2004

(c) Copyright 1997 Oracle Corporation. All rights reserved.

Connected to:
Oracle8 Personal Edition Release 8.0.4.0.0 - Production
PL/SQL Release 8.0.4.0.0 - Production

SQL> CREATE TABLE maaslar
2   (SicilNo      number(4),
3   MaasAyı      number(2),
4   Maas         number(12));

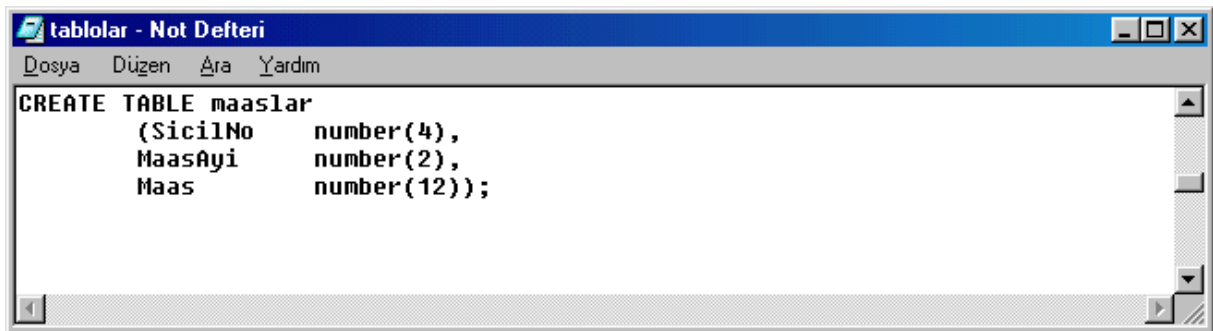
Table created.

SQL> |
```

Şekil 6.3.5. SQL Plus 8.0 Komut Yazılmış Ekran

Bir komut yazılırken birden fazla satıra komutun istenilen bir yerinden bölünerek yazılabilir. Yukarıdaki ekranda olduğu gibi birinci satıra bir şeyler yazdıktan sonra ENTER tuşuna basılarak istenirse alt satıra geçilir. Alt satıra geçerken satır numarasını belirten 2 rakamı kendiliğinden ekrana gelir ve 2.satır komutları yazılmaya devam edilir. Tekrar alt satıra geçilmek istenildiğinde ENTER tuşuna basılır, ekrana 3 yazısı gelir ve yazma işlemine bu şekilde devam edilir. Komut yazım işlemi bittikten sonra en sona noktalı virgül (;) yazılarak ENTER tuşuna basılır ve komut çalıştırılır. SQL yazma işleminin bitip komutun çalıştırılmaya başlayacağını noktalı virgül işaretini gördükten sonra anlar ve bu işaretten sonra yazılan hiç bir şeyi dikkate almadan komutu çalıştırır. Komutu çalıştırdıktan sonra yaptığı ile ilgili ekrana bilgi mesajı verir ve tekrar SQL> konumuna gelir. SQL komutları bu şekilde yazılmaya devam edilerek işlemler yapılır.

SQL komut satırında komutları tek tek yazmak yerine bu komutlar herhangi bir editörde yazılıp kopyala-yapıştır ile veya dosya adı yazılarak da SQL 'de çalıştırılabilir.



```
tablolar - Not Defteri
Dosya Düzen Ara Yardım

CREATE TABLE maaslar
(SicilNo      number(4),
MaasAyı      number(2),
Maas         number(12));
```

Şekil 6.3.6. NOTEPAD ile yazılmış SQL Komut Ekranı

Yukarıdaki örnekte olduğu gibi NOTEPAD editör programına yazılan SQL komut satırlarının hepsi kopyala komutu ile alınarak SQL programına yapıştırılabilir ve komutların direk çalışması sağlanabilir. NOTEPAD ile yazılarak yapılan kopyalama işlemi özellikle uzun komutların yazılması ve işletilmesinde büyük pratiklik sağlar. Çünkü SQL ortamında yanlış yazılmış satırlar için ENTER tuşuna bastıktan sonra yukarı veya aşağı doğru olan satırlara gidiş veya geri dönüş mümkün değildir. SQL 'de bu yanlışlıkları düzenlemenin farklı yöntemleri vardır. Bu da biraz zaman alıcıdır ve SQL'in bazı kullanım özelliklerini bilmek gerektirir. NOTEPAD ile yazmak ve daha sonra bunu kopyalayarak kullanmak daha pratik bir yöntemdir. Hangi metodun kullanımı tercihi kullanıcının isteğine bağlıdır.

Not : Oracle ve Sybase gibi bazı veritabanı programlarında her bir SQL cümleciğinin sonuna noktalı virgül (;) konulması gerekir. Microsoft tabanlı sistemlerde bu türden bir zorunluluk bulunmamaktadır.

7. TEMEL SQL KOMUTLARI-I

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. SQL cümlecikleri genellikle aynı olmakla birlikte farklı veri tabanı ortamlarında değişebilmektedir. Ayrıca veri tabanlarının kendilerine özgü sql komutları da vardır. Bu bölümde her ortamda geçerli olan temel sql komutları öğretilecektir. Şimdi 5. bölümde tasarladığımız örnek personel veritabanını SQL ile kullanalım. Örnek Veri Tabanımızdaki tablolar ve tablolara ait alan ile alanların veri tipleri aşağıda listelenmiştir.

TabloAdı	TabloAlanları	Veri Tipi
Personel	SicilNo, Adı, Soyadı, MeslekKodu, BirimKodu, DoğumTarihi, Adres	Sayı(4) Karakter(15) Karakter(15) Sayı(3) Sayı(3) Tarih Karakter(25)
Meslekler	MeslekKodu, MeslekAdı	Sayı(3) Karakter(20)
ÇalışılanBirim	BirimKodu, BirimAdı	Sayı(3) Karakter(20)
Maaslar	SicilNo, MaasAyı, Maas	Sayı(4) Sayı(2) Karakter(12)

7.1. 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. Veri tabanının yaratılması aşağıda verilmiştir.

```
CREATE DATABASE database_name  
[ON {DEFAULT | database_device} [= size]  
[, database_device [= size]]...]  
[LOG ON database_device [= size]  
[, database_device [= size]]...][FOR LOAD]
```


database_name : Bu yaratılacak olan veri tabanının ismidir.

ON : Yaratılacak olan veri tabanının hangi device üzerinde yer alacağını belirten bir parametredir. Burada aynı zamanda bu device üzerinde **size (alan, ölçü)** parametresi ile database'in ne kadar yer kaplayacağını belirtilmiş olur. Eğer device tanımlanmazsa SQL server default device üzerinde 5 mb bir veri tabanı yaratır. Bu parametre içerisinde birkaç device ismi kullanılarak veri tabanının bir kaç device üzerinde yer alması sağlanabilir.

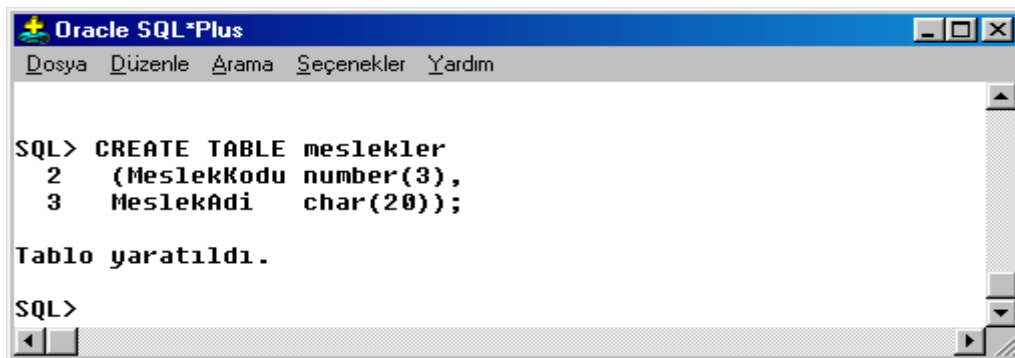
LOG ON : Yaratılacak olan veri tabanının log'unun hangi device üzerinde yer alacağını belirten bir parametredir. Burada aynı zamanda bu device üzerinde **size (alan, ölçü)** parametresi ile database'in log'unun ne kadar yer kaplayacağı belirtilir. Eğer device tanımlanmazsa SQL server default device üzerinde bir log tutar. Bu parametre içinde birkaç device ismi kullanılarak veri tabanı log'unun bir kaç device üzerinde yer almasını sağlanabilir.

Örnek 1 : `CREATE DATABASE personel_isleri`

Bu komut ile SQL Server üzerinde personel_isleri isimli boş bir database yaratılmış olur. Bu disk üzerinde fiziksel bir isimdir ve yaratılacak tablolar bu isimli veritabanının altında yaratılacaktır. Daha sonra bu veritabanı üzerinde verilerin yazılacağı tablolar tanımlanır. Tablo yaratmak için de create komutu kullanılır. Yazılışı aşağıdaki gibidir.

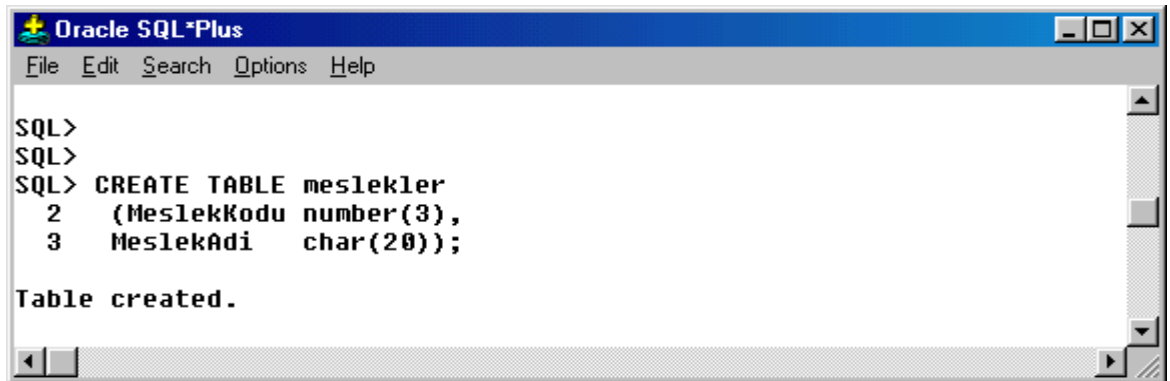
```
CREATE TABLE tablo_adı
(sütun_adı_1 veri_tipi,
sütun_adı_2 veri_tipi,
.....);
```

SQL programında Create komutu ile meslekler tablosu aşağıdaki gibi yaratılır.



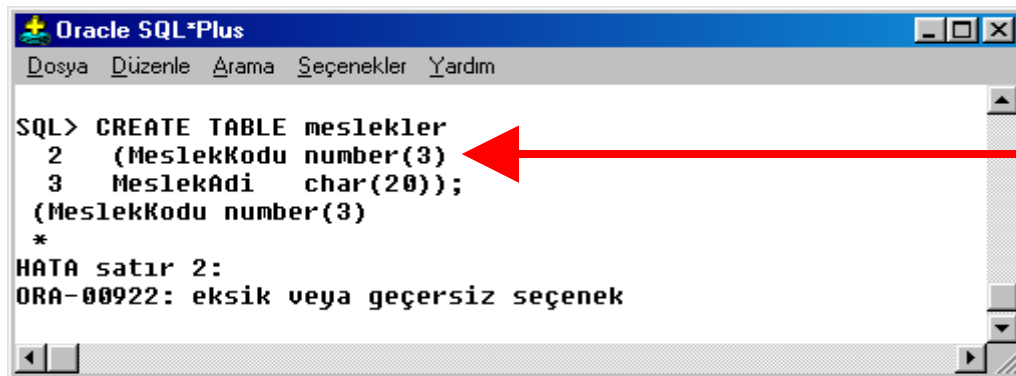
Şekil 7.1.1. Tablo Yaratılması Ekranı-1

Tablo yaratılmış ve tablo yaratıldı mesajı alınmıştır. SQL bilgisayara kurulurken dil seçeneği olarak Türkçe seçildiği için ekrandaki program menü seçenekleri ve “Tablo Yaratıldı” bilgi mesajı ekrana Türkçe olarak gelmiştir. Dil olarak İngilizce seçilmiş olsaydı ekran görüntüsü aşağıdaki gibi olurdu. Burada tablo yaratıldı değil **“Table created”** bilgi mesajı ve İngilizce menü seçenekleri bulunmaktadır.



Şekil 7.1.2. Tablo Yaratılması Ekranı-2

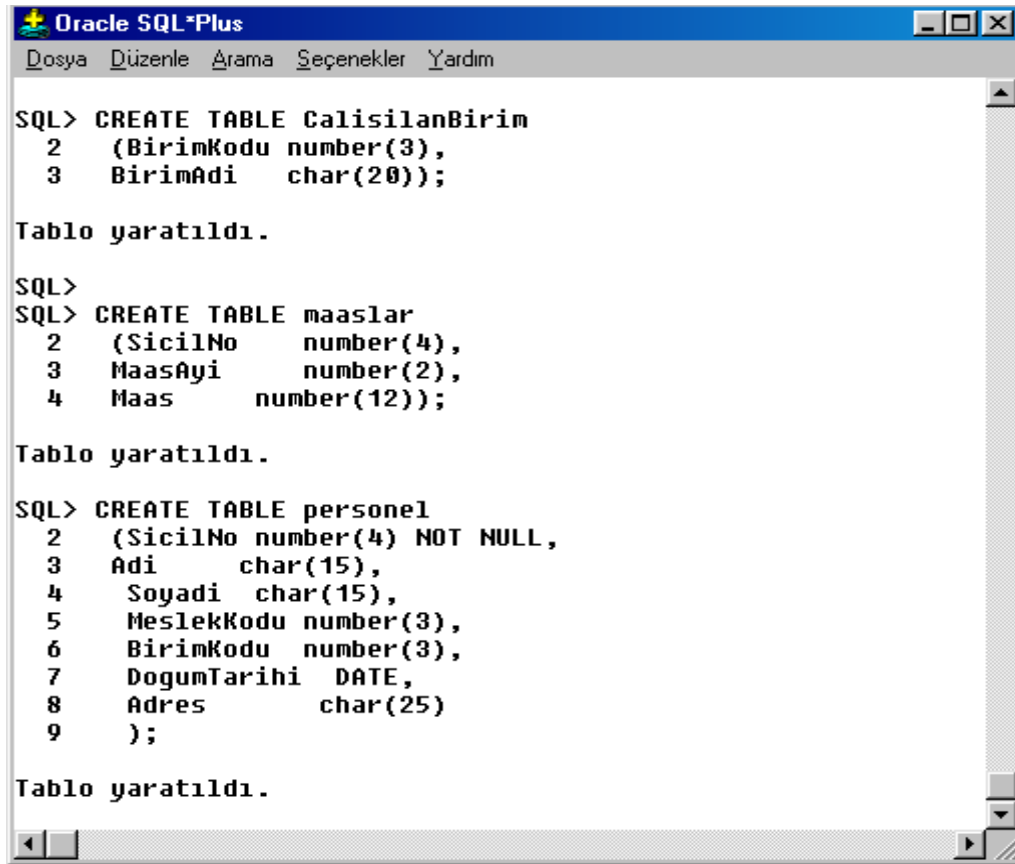
Tablo yaratılırken yazılan cümlelerde yazılım yanlış yapılsa aşağıdaki gibi hata mesajı alınır ve komut çalışmaz. Buradaki hata create komutunda her bir tanımlanan alanlar arasına virgül konulması gerektiği halde 2.satır sonuna virgül işareti yazılması unutulmuştur.



Şekil 7.1.3. Tablo Yaratılması Yanlış Yazım Ekranı

Aşağıdaki ekranda CalisilanBirim, maaslar ve personel tabloları SQL komutları ile yaratılmıştır. Personel tablosu yaratılırken **(SicilNo number(4) NOT NULL,** ifadesi kullanılmıştır. NOT NULL ifadesi tabloya bilgi yazılırken bu alanın boş bırakılamayacağı, muhakkak bilgi yazılması gerektiği durumlarda kullanılır. Kullanıcı bilgi girişi yaparken Personel SicilNo alanını boş bırakamaz ve muhakkak bir değer yazmak zorundadır. Bu tanımlama ile veri tabanındaki bilgilerin tabloyu tanımlama aşamasında doğru yazılması sağlanabilir. Burada NOT NULL ifadesi kullanılmamış olsaydı; kullanıcı personel sicil no alanını boş bırakabilirdi. Sicil nosu olmayan bir personel olamayacağı için kullanıcıya hata yapma imkanı verilmiş olurdu. Bu durumda kullanıcıya boş geçme izni vermediğimiz için oluşabilecek hatayı baştan engellemiş oluruz.

Not : SQL komutları için oracle veritabanı kullanıldığından veri tipleri Oracle veri tipleridir. Char yerine varchar2 veri tipi kullanılabilir. Oracle veri tabanına has olan bu değişken tipinde ayrılan yer kadar değil bu alana yazılan karakter sayısı kadarlık alan veri tabanında yer tutar. Dolayısı ile diskten yer kazanılacaktır.



```
Oracle SQL*Plus
Dosya Düzenle Arama Seçenekler Yardım

SQL> CREATE TABLE CalisilanBirim
2   (BirimKodu number(3),
3   BirimAdi   char(20));

Tablo yaratıldı.

SQL>
SQL> CREATE TABLE maaslar
2   (SicilNo   number(4),
3   MaasAyı    number(2),
4   Maas       number(12));

Tablo yaratıldı.

SQL> CREATE TABLE personel
2   (SicilNo number(4) NOT NULL,
3   Adi      char(15),
4   Soyadi   char(15),
5   MeslekKodu number(3),
6   BirimKodu number(3),
7   DogumTarihi DATE,
8   Adres     char(25)
9   );

Tablo yaratıldı.
```

Şekil 7.1.4. Tablo Yaratılması Ekranı-3

7.2. ALTER (Düzenle) TABLE Komutu

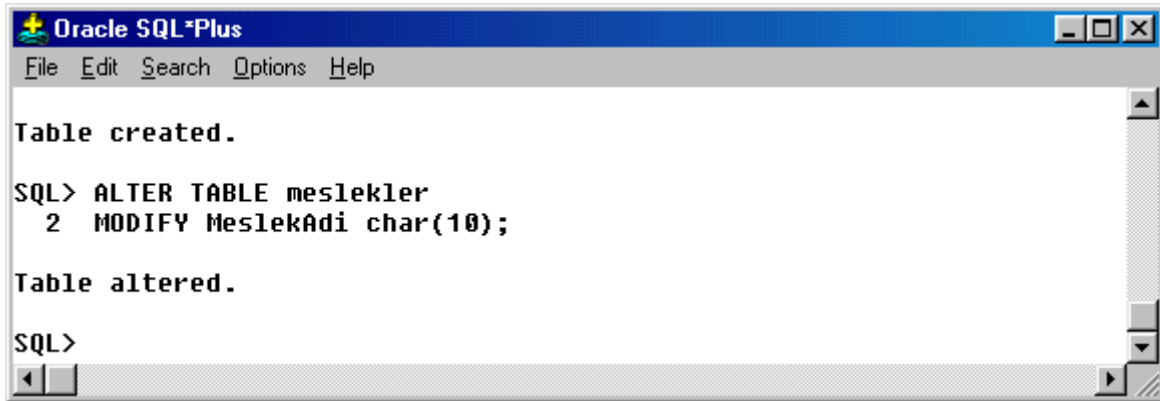
Daha önceden oluşturulmuş bir tablonun yapısını değiştirmek için ALTER TABLE komutu kullanılır.

```
ALTER TABLE tablo_adı
ADD | MODIFY | DROP (<sütun adı> veri tipi <sütun kısıtlaması>)
ENABLE ifade1
DISABLE ifade2
```

ALTER komutuyla tablolara yeni bir alan ve kısıtlama eklenebilir, var olan alan ve kısıtlamaların durumu değiştirilebilir veya tablodan kısıtlamalar silinebilir.

```
ALTER TABLE tbl_ogr
ADD CONSTRAINT cst_Bolum
FOREIGN KEY(Bolum)
REFERENCES usr_gazi.tbl_bol(BolKod);
```

Yukarıdaki örnekte tbl_ogr adlı tablonun yapısı değiştiriliyor. Bir başka kullanıcının bir tablosu ile ilişki kuruluyor.



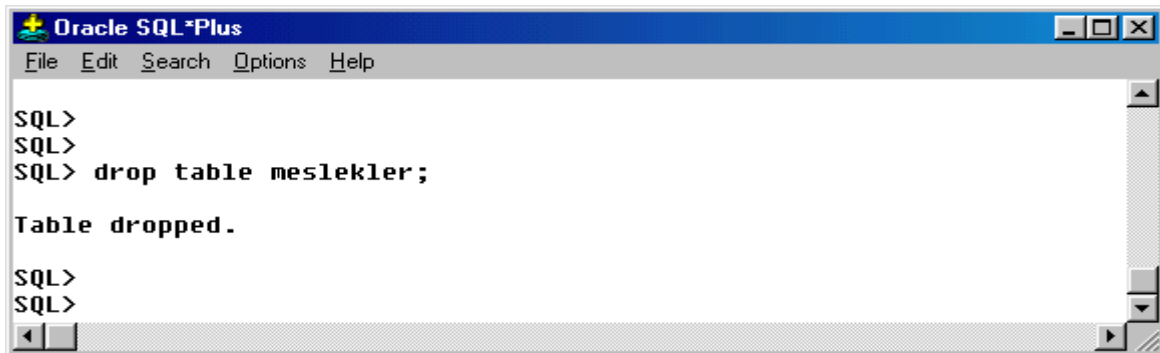
Şekil 7.3.1. Tablo Yapısının Değiştirilmesi Ekranı

Bu örnekte meslekler tablosunun MeslekAdi alanının uzunluğu 20 den 10 karaktere düşürülmüştür. Kullanıcı bu alan için 10 karakterden fazla bilgi yazamaz.

7.3. DROP (Sil) TABLE Komutu

`DROP TABLE table_name [CASCADE CONSTRAINTS]`

Bir Tabloyu silmek için DROP deyimi kullanılır. Köşeli parantez içerisindeki tanım kullanılırsa mater-detay ilişkili tablolarda master tablo silinince detay tablolarında otomatik olarak silinmesi sağlanır. Eğer bu seçenek kullanılmazsa diğer tablolarla ilişkisi bulunan bir tablo silinemez. Tablo ancak ilişkiler kaldırıldıktan sonra silinebilir.

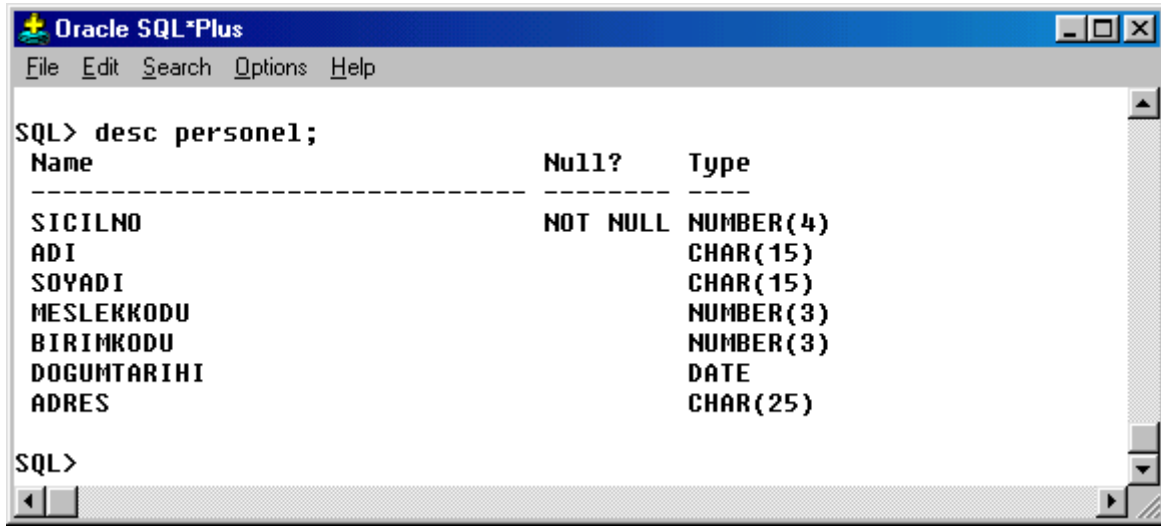


Şekil 7.3.1. Tablo Silinmesi Ekranı

7.4. DESCRIBE Komutu

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.

`DESC table_name`



```
SQL> desc personel;
```

Name	Null?	Type
SICILNO	NOT NULL	NUMBER(4)
ADI		CHAR(15)
SOYADI		CHAR(15)
MESLEKKODU		NUMBER(3)
BIRIMKODU		NUMBER(3)
DOGUMTARIHI		DATE
ADRES		CHAR(25)

```
SQL>
```

Şekil 7.4.1. Tablo Yapısını Görüntüleme Ekranı

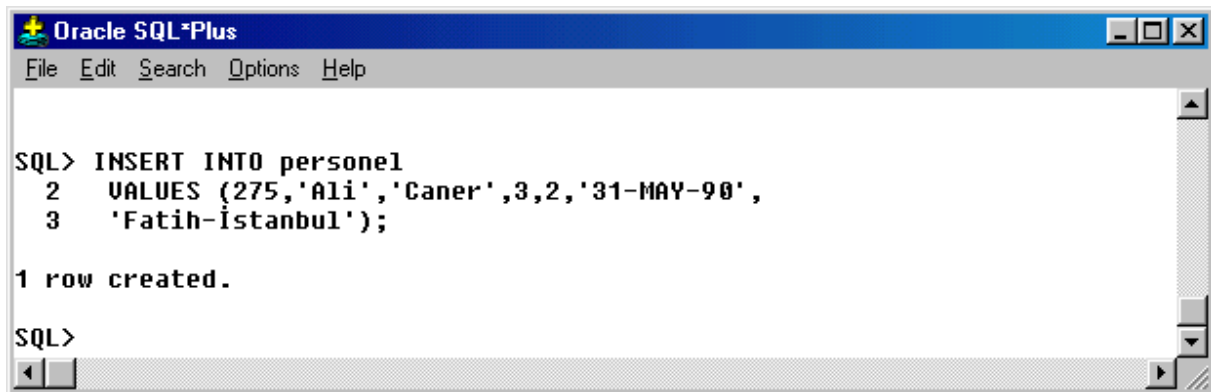
7.5. INSERT (Ekle) Komutu

Bir tabloya bilgi eklemek için kullanılan komuttur. Genel kullanım şekli aşağıdaki gibidir ;

`INSERT INTO table_name VALUES (deger1, deger2, deger3,...)`

Eğer sadece belirli alanlara (sütunlara) değer yazılması isteniyorsa INSERT INTO komutu aşağıdaki gibi de kullanılabilir. Sütunlara yazılan alanların değerleri sayılardan oluşuyor ise, sayı direk olarak yazılmalı, karakterlerden oluşuyor ise ifade tek tırnak işareti arasına alınarak yazılmalıdır. Her alan için yazılacak değerler tablodaki sütun sırasına göre ve her bir değer yazıldıktan sonra aralarına virgül işareti konularak yazılmalıdır.

`INSERT INTO table_name (sütun_adı_1, sütun_adı_2,)
VALUES(değer1, değer2,);`



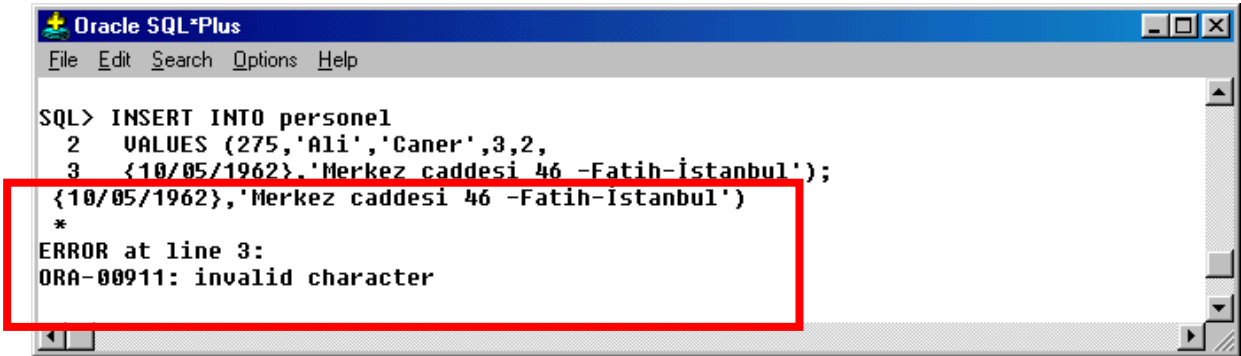
```
SQL> INSERT INTO personel  
2   VALUES (275,'Ali','Caner',3,2,'31-MAY-90',  
3   'Fatih-İstanbul');
```

1 row created.

```
SQL>
```

Şekil 7.5.1. Personel Tablosuna Kayıt Ekleme Ekranı

Aşağıda personel tablosuna INSERT komutu ile kayıt eklerken yapılan hatalar ve ekrana gelen bilgi mesajlarından örnekler verilmiştir. Hatanın olduğu yere * işareti otomatik olarak gelir ve hatanın hangi satırda olduğu ve hatanın ne olduğuna dair bir açıklama satırı ekrana gelir. Aşağıda hata mesajı dikdörtgen kutu içerisine alınmıştır.

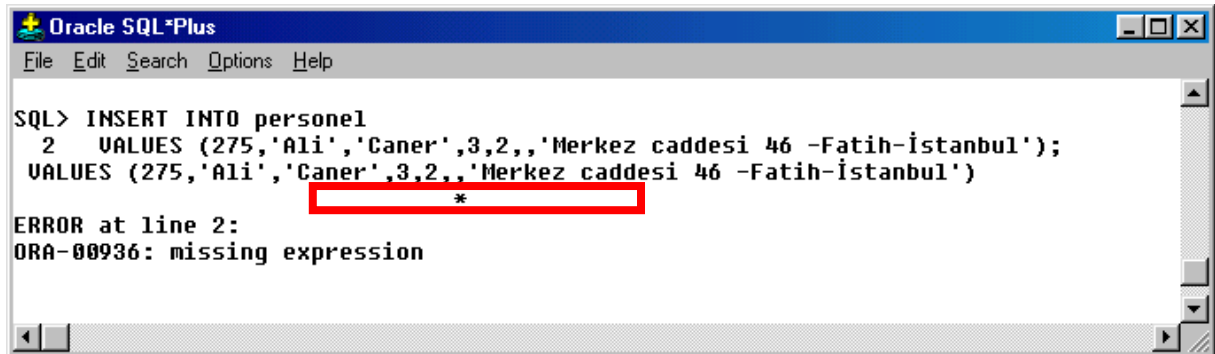


```
Oracle SQL*Plus
File Edit Search Options Help

SQL> INSERT INTO personel
  2   VALUES (275,'Ali','Caner',3,2,
  3   {10/05/1962},'Merkez caddesi 46 -Fatih-İstanbul');
  {10/05/1962},'Merkez caddesi 46 -Fatih-İstanbul')
  *
ERROR at line 3:
ORA-00911: invalid character
```

Şekil 7.5.2. Personel Tablosuna Kayıt Ekleme Hata Ekranı-1

Burada personelin doğum tarihi için yazılan değer yanlış formatta yazılmıştır.

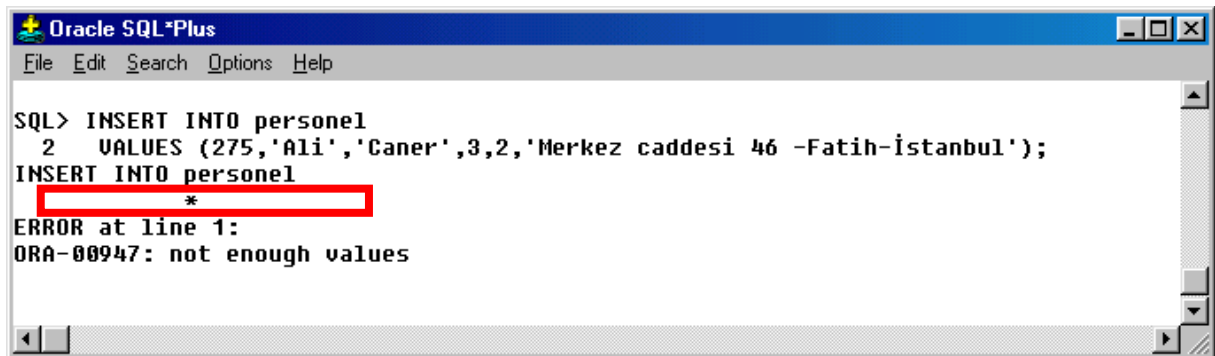


```
Oracle SQL*Plus
File Edit Search Options Help

SQL> INSERT INTO personel
  2   VALUES (275,'Ali','Caner',3,2,,'Merkez caddesi 46 -Fatih-İstanbul');
  VALUES (275,'Ali','Caner',3,2,,'Merkez caddesi 46 -Fatih-İstanbul')
  *
ERROR at line 2:
ORA-00936: missing expression
```

Şekil 7.5.3. Personel Tablosuna Kayıt Ekleme Hata Ekranı-2

* olduğu noktada bir adet virgül fazla yazılarak yazılım yanlış yapılmıştır.



```
Oracle SQL*Plus
File Edit Search Options Help

SQL> INSERT INTO personel
  2   VALUES (275,'Ali','Caner',3,2,'Merkez caddesi 46 -Fatih-İstanbul');
INSERT INTO personel
  *
ERROR at line 1:
ORA-00947: not enough values
```

Şekil 7.5.4. Personel Tablosuna Kayıt Ekleme Hata Ekranı-3

Personel tablosunda 7 tane sütun(alan) olmasına rağmen burada 6 tane değer yazılmıştır. Bu nedenle yeterli değer yazılmamış olduğundan hata vermiştir.

```

Oracle SQL*Plus
File Edit Search Options Help

SQL> INSERT INTO personel
  2   VALUES (275,'Ali','Caner',3,2,'31-MAY-90',
  3   'Merkez caddesi 46 -Fatih-İstanbul');
  *
ERROR at line 3:
ORA-01401: inserted value too large for column

```

Şekil 7.5.5. Personel Tablosuna Kayıt Ekleme Hata Ekranı-4

Burada da adres alanı için 20 karakterlik yer ayrılmasına rağmen daha uzun karakter sayısında bir adres değeri yazılmıştır. Sütun için uzun bir değer hata mesajı alınmıştır.

Tabloya eklenecek kayıtlar bu şekilde tek tek INSERT komutu ile yazılarak yapılır. Şimdi sırasıyla tablolarımıza değerlerini yazalım. Burada her kayıt için INSERT komutu tek tek yazılabileceği gibi daha önceden de belirtildiği gibi bunlar bir editörde yazılarak kopyala-yapıştır yöntemi de kullanılabilir. Örnek aşağıdaki gibidir.

```

personel - Not Defteri
Dosya Düzen Ara Yardım

INSERT INTO personel VALUES (1240,'Durmus','Bozkara',004,101,'10-Jan-68','Fatih-istanbul');
INSERT INTO personel VALUES (648,'Ali','Kantar',002,101,'01-Feb-72','Taksim-istanbul');
INSERT INTO personel VALUES (117,'Fikret','Tatar',005,101,'12-Mar-80','Konak-izmir');
INSERT INTO personel VALUES (326,'Halil','Kangal',005,101,'23-Dec-58','Konak-izmir');
INSERT INTO personel VALUES (890,'Hüseyin','Kantar',005,101,'01-Jan-55','Alsancak-izmir');
INSERT INTO personel VALUES (1002,'Zehra','Yüksek',005,101,'30-May-78','Fatih-istanbul');
INSERT INTO personel VALUES (1290,'Kaan','Yalçın',005,101,'31-Mar-79','Mezitli-Mersin');
INSERT INTO personel VALUES (835,'Kazim','Akbeke',003,201,'20-May-66','Mezitli-Mersin');
INSERT INTO personel VALUES (680,'Hülya','Bakar',002,201,'25-Dec-76','Taksim-istanbul');
INSERT INTO personel VALUES (186,'Ali','Sümbül',006,401,'14-Nov-88','Çankaya-Ankara');
INSERT INTO personel VALUES (153,'Orhan','Duru',007,201,'10-Feb-83','Çankaya-Ankara');
INSERT INTO personel VALUES (245,'Fatma','Papatya',007,201,'15-Sep-66','Alsancak-izmir');
INSERT INTO personel VALUES (345,'Serpil','Yücel',005,201,'18-Jun-70','Taksim-istanbul');
INSERT INTO personel VALUES (654,'Refik','Erel',005,201,'20-Apr-71','Beykoz-istanbul');
INSERT INTO personel VALUES (522,'Selim','Arman',007,201,'23-Jan-77','Beykoz-istanbul');
INSERT INTO personel VALUES (1110,'Salih','Bayar',006,301,'11-Dec-78','Ulus-Ankara');
INSERT INTO personel VALUES (178,'Fatma','Yalçın',006,301,'13-Dec-80','Ulus-Ankara');
INSERT INTO personel VALUES (898,'Mehmet','Gezgin',008,301,'27-May-76','Ulus-Ankara');
INSERT INTO personel VALUES (1077,'Orhan','Sezgin',008,301,'30-Mar-79','Ulus-Ankara');
INSERT INTO personel VALUES (1121,'Nazan','Ulusoy',005,301,'18-Feb-71','Çankaya-Ankara');
INSERT INTO personel VALUES (960,'Özgen','Üney',002,301,'16-Jan-70','Taksim-istanbul');
INSERT INTO personel VALUES (590,'Özgür','Irmak',003,401,'14-Oct-59','Sincan-Ankara');
INSERT INTO personel VALUES (772,'Nihat','Ünalp',008,401,'12-Nov-87','Sincan-Ankara');
INSERT INTO personel VALUES (393,'Salih','Talay',008,401,'10-Sep-68','Seyhan-Adana');
INSERT INTO personel VALUES (849,'Necila','Alakoç',005,401,'05-Apr-83','Merkez-Van');
INSERT INTO personel VALUES (967,'Fatma','Erdem',005,401,'02-Mar-69','Konak-izmir');
INSERT INTO personel VALUES (267,'Cengiz','Burma',001,101,'15-Feb-58','Mezitli-Mersin');
INSERT INTO personel VALUES (520,'Nuran','Gölbakay',007,201,'22-Feb-68','Gölbasi-Ankara');
INSERT INTO personel VALUES (456,'Galip','Aslan',008,101,'08-Jun-70','Tece-Mersin');
INSERT INTO personel VALUES (997,'Zehra','Burma',006,301,'23-Dec-68','Mezitli-Mersin');

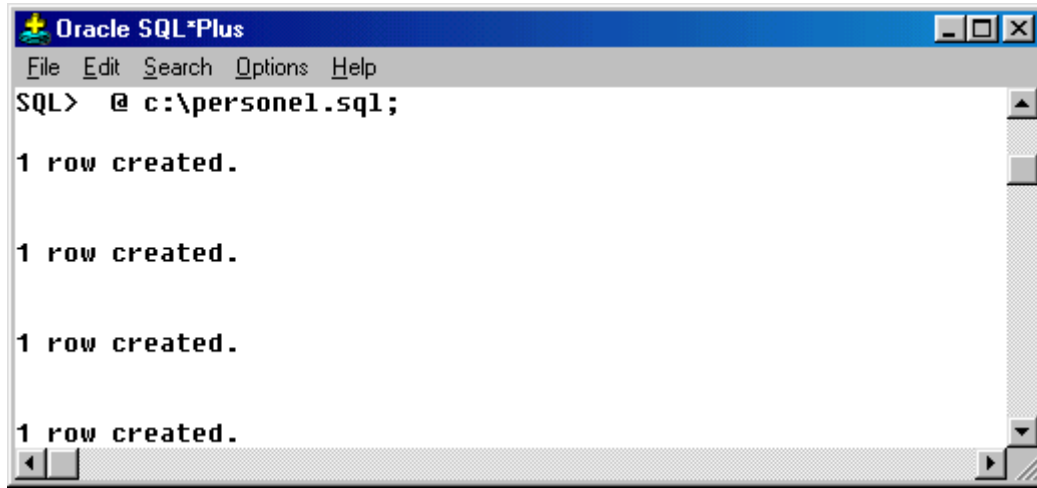
```

Şekil 7.5.6. Personel Tablosuna Kayıt Ekleme için Yazılan Personel.sql dosyası

Bir diğer metot ise bilgileri bir editörde yazıp bunu dosya uzantısı .SQL olarak dosyaya kaydetmek ve daha sonra

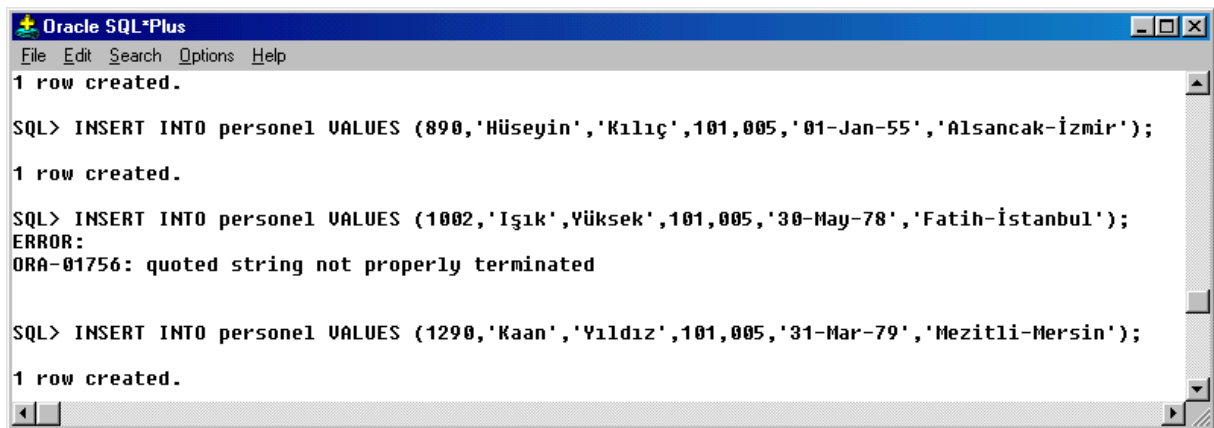
```
SQL> @ c:\tablolar.sql      veya
SQL> START c:\tablolar.sql
```

Komutları ile çalıştırmaktır. Bu örnekte INSERT komutu için tüm tablo kayıtları yukarıda ekran görüntüsü olan personel.sql isimli bir dosyaya yazılıp, daha sonra bu dosya yukarıda anlatılan şekilde çalıştırılarak kayıtların daha kolay yazılması sağlanmıştır.



Şekil 7.5.7. Personel.sql dosyasının çalıştırılarak Tabloya Kayıt Eklenmesi

“1 row created” mesajı her bir insert komutundan sonra tabloya eklenen kayıt için verdiği mesajdır. Kayıtlar sorunsuz eklenmiştir.

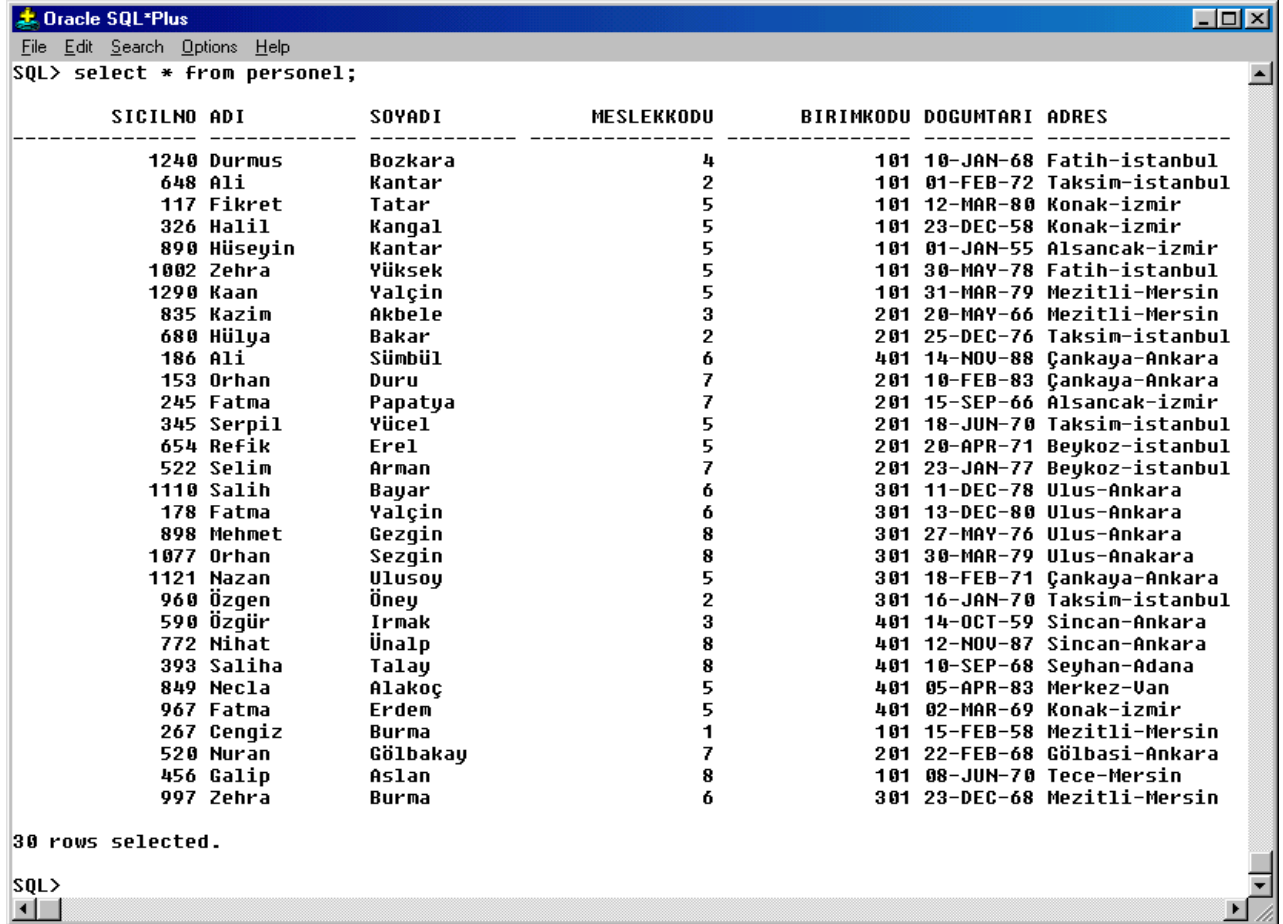


Şekil 7.5.8. Personel.sql dosyasından Kopyala-Yapıştır ile Kayıt Eklenmesi

Yukarıdaki ekranda personel.sql dosyasında yazılan komutlar kopyala-yapıştır metodu ile kullanılmıştır.

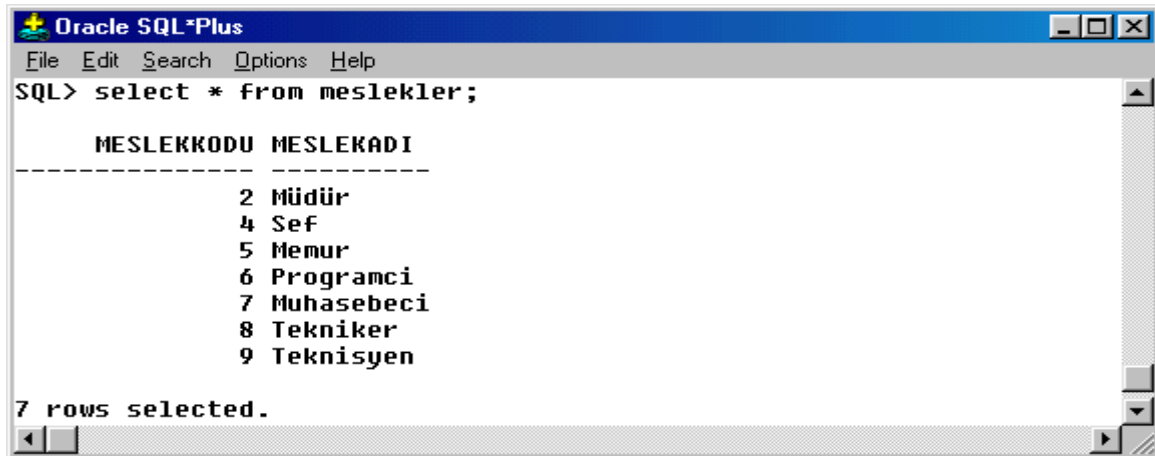
Kopyala-Yapıştır yöntemindeki en büyük farkın; her bir komut satırının ekranda görüntüleniyor olmasıdır. Bu ekranda birde **ERROR** satırı bulunmaktadır. Bu satır; soyadı alanı için **Yüksek** değeri yazılırken başlangıçtaki tırnak işaretinin unutulmuş olmasıdır.

Personel tablosuna ait 30 adet personelin bilgilerinin girişi tamamlanmıştır.



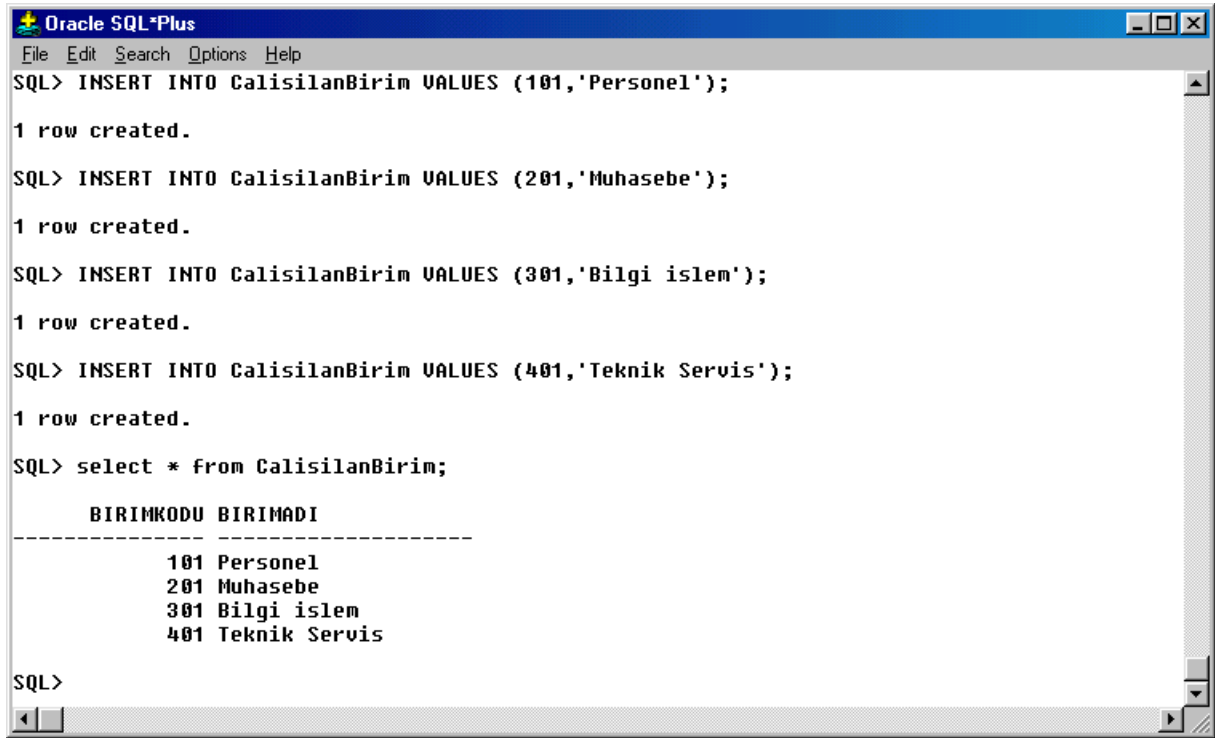
SICILNO	ADI	SOYADI	MESLEKKODU	BIRIMKODU	DOGUMTARI	ADRES
1240	Durmus	Bozkara	4	101	10-JAN-68	Fatih-istanbul
648	Ali	Kantar	2	101	01-FEB-72	Taksim-istanbul
117	Fikret	Tatar	5	101	12-MAR-80	Konak-izmir
326	Halil	Kangal	5	101	23-DEC-58	Konak-izmir
890	Hüseyin	Kantar	5	101	01-JAN-55	Alsancak-izmir
1002	Zehra	Yüksek	5	101	30-MAY-78	Fatih-istanbul
1290	Kaan	Valçin	5	101	31-MAR-79	Mezitli-Mersin
835	Kazim	Akbele	3	201	20-MAY-66	Mezitli-Mersin
680	Hülya	Bakar	2	201	25-DEC-76	Taksim-istanbul
186	Ali	Sümbül	6	401	14-NOV-88	Çankaya-Ankara
153	Orhan	Duru	7	201	10-FEB-83	Çankaya-Ankara
245	Fatma	Papatya	7	201	15-SEP-66	Alsancak-izmir
345	Serpil	Yücel	5	201	18-JUN-70	Taksim-istanbul
654	Refik	Erel	5	201	20-APR-71	Beykoz-istanbul
522	Selim	Arman	7	201	23-JAN-77	Beykoz-istanbul
1110	Salih	Bayar	6	301	11-DEC-78	Ulus-Ankara
178	Fatma	Valçin	6	301	13-DEC-80	Ulus-Ankara
898	Mehmet	Gezgin	8	301	27-MAY-76	Ulus-Ankara
1077	Orhan	Sezgin	8	301	30-MAR-79	Ulus-Ankara
1121	Nazan	Ulusoy	5	301	18-FEB-71	Çankaya-Ankara
960	Özgen	Öney	2	301	16-JAN-70	Taksim-istanbul
590	Özgür	Irmak	3	401	14-OCT-59	Sincan-Ankara
772	Nihat	Ünalp	8	401	12-NOV-87	Sincan-Ankara
393	Saliha	Talay	8	401	10-SEP-68	Seyhan-Adana
849	Necle	Alakoç	5	401	05-APR-83	Merkez-Van
967	Fatma	Erdem	5	401	02-MAR-69	Konak-izmir
267	Cengiz	Burma	1	101	15-FEB-58	Mezitli-Mersin
520	Nuran	Gölbakay	7	201	22-FEB-68	Gölbasi-Ankara
456	Galip	Aslan	8	101	08-JUN-70	Tece-Mersin
997	Zehra	Burma	6	301	23-DEC-68	Mezitli-Mersin

Şekil 7.5.9. Personel Tablosuna Ait Kayıtlar.



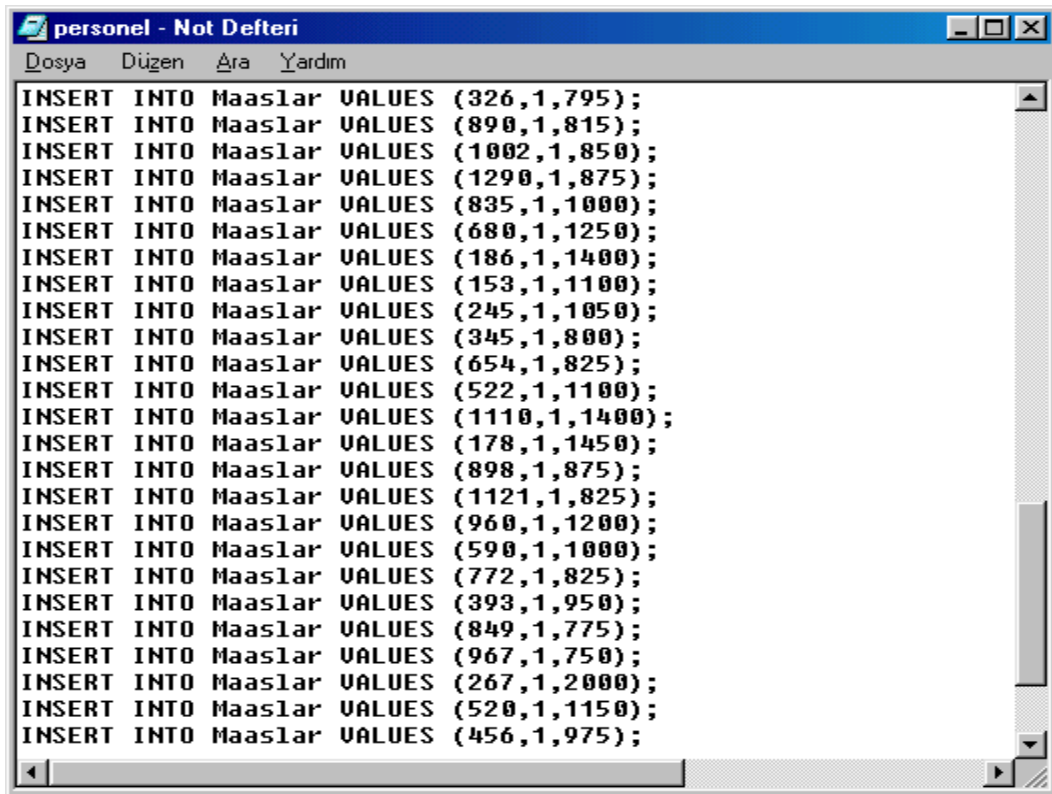
MESLEKKODU	MESLEKADI
2	Müdür
4	Sef
5	Memur
6	Programci
7	Muhasebeci
8	Tekniker
9	Teknisyen

Şekil 7.5.10. Meslekler Tablosuna Ait Kayıtlar.



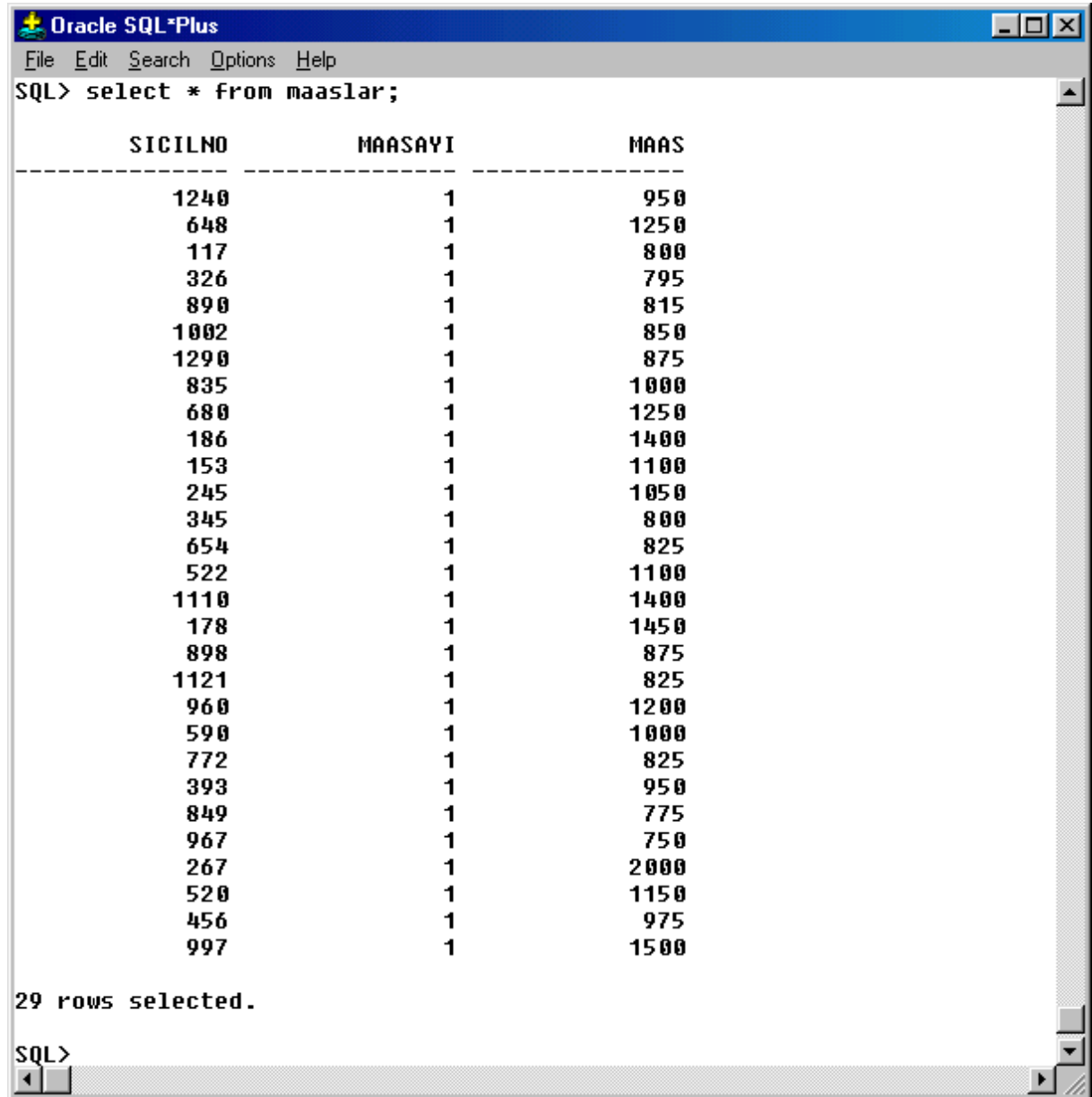
```
Oracle SQL*Plus
File Edit Search Options Help
SQL> INSERT INTO CalisilanBirim VALUES (101,'Personel');
1 row created.
SQL> INSERT INTO CalisilanBirim VALUES (201,'Muhasebe');
1 row created.
SQL> INSERT INTO CalisilanBirim VALUES (301,'Bilgi islem');
1 row created.
SQL> INSERT INTO CalisilanBirim VALUES (401,'Teknik Servis');
1 row created.
SQL> select * from CalisilanBirim;
  BIRIMKODU BIRIMADI
  -----
      101 Personel
      201 Muhasebe
      301 Bilgi islem
      401 Teknik Servis
SQL>
```

Şekil 7.5.11. CalisilanBirim Tablosu Kayıtları ve Insert komutu ile kayıt eklenmesi.



```
personel - Not Defteri
Dosya Düzen Ara Yardım
INSERT INTO Maaslar VALUES (326,1,795);
INSERT INTO Maaslar VALUES (890,1,815);
INSERT INTO Maaslar VALUES (1002,1,850);
INSERT INTO Maaslar VALUES (1290,1,875);
INSERT INTO Maaslar VALUES (835,1,1000);
INSERT INTO Maaslar VALUES (680,1,1250);
INSERT INTO Maaslar VALUES (186,1,1400);
INSERT INTO Maaslar VALUES (153,1,1100);
INSERT INTO Maaslar VALUES (245,1,1050);
INSERT INTO Maaslar VALUES (345,1,800);
INSERT INTO Maaslar VALUES (654,1,825);
INSERT INTO Maaslar VALUES (522,1,1100);
INSERT INTO Maaslar VALUES (1110,1,1400);
INSERT INTO Maaslar VALUES (178,1,1450);
INSERT INTO Maaslar VALUES (898,1,875);
INSERT INTO Maaslar VALUES (1121,1,825);
INSERT INTO Maaslar VALUES (960,1,1200);
INSERT INTO Maaslar VALUES (590,1,1000);
INSERT INTO Maaslar VALUES (772,1,825);
INSERT INTO Maaslar VALUES (393,1,950);
INSERT INTO Maaslar VALUES (849,1,775);
INSERT INTO Maaslar VALUES (967,1,750);
INSERT INTO Maaslar VALUES (267,1,2000);
INSERT INTO Maaslar VALUES (520,1,1150);
INSERT INTO Maaslar VALUES (456,1,975);
```

Şekil 7.5.12. Maaslar Tablosuna Insert komutu ile kayıt eklenmesi.



Oracle SQL*Plus

File Edit Search Options Help

SQL> select * from maaslar;

SICILNO	MAASAYI	MAAS
1240	1	950
648	1	1250
117	1	800
326	1	795
890	1	815
1002	1	850
1290	1	875
835	1	1000
680	1	1250
186	1	1400
153	1	1100
245	1	1050
345	1	800
654	1	825
522	1	1100
1110	1	1400
178	1	1450
898	1	875
1121	1	825
960	1	1200
590	1	1000
772	1	825
393	1	950
849	1	775
967	1	750
267	1	2000
520	1	1150
456	1	975
997	1	1500

29 rows selected.

SQL>

Şekil 7.5.13. Personele ait 1. ay Maaslar Tablosu Kayıtları.

Yukarıda daha önce tanımlanan personele ait Maaslar tablosunun kayıtları bulunmaktadır. (Maaslar $maas \times 100.000$ olarak kısaltılmıştır). Personel tablosunda 30 kişi bulunmakta ama buna karşılık 29 adet personel maaşı bulunmaktadır. 1 personelin maas bilgisi özellikle eksik girilmiştir. Daha sonraki bölümlerde buna ait düzenleme yapılacaktır.

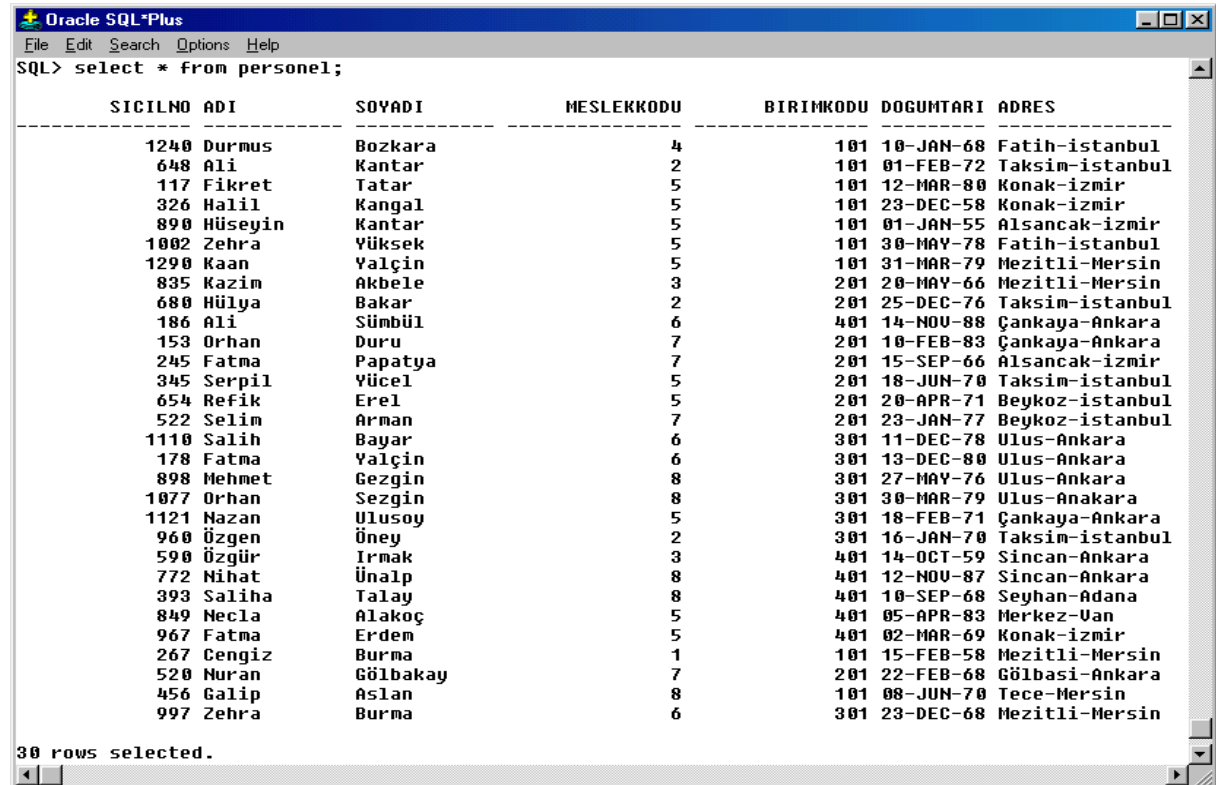
8. TEMEL SQL KOMUTLARI-II

8.1. SELECT (Seç) Komutu

Veri tabanındaki tablo veya tablolardan istenilen özellikteki verileri seçip listeleme için kullanılan komuttur. Genel kullanımı aşağıdaki gibidir.

```
SELECT [ DISTINCT | ALL ] <sütun(lar)> FROM <tablo adı (lar)>
[ WHERE <şart (lar)> ]
[ GROUP BY <sütunlar> ]
[ HAVING < grup kısıtlaması> ]
[ ORDER BY <sütun(lar)> [ ASC | DESC ]> ]
```

Select komutundan sonra listelenmesi istenilen sütunların isimleri aralarına virgül işareti konularak yan yana yazılır. Tablonun bütün sütunlarının listelenmesinde “*” karakteri kullanılır. FROM dan sonra listeleme yapılacak tablo / tabloların isimleri yazılır. Eğer aynı sql cümlecigi ile bir kaç tablo üzerinde işlem yapılmak istenirse tablo isimleri arasına virgül konulmalıdır. Personel tablosundaki tüm kayıtların listelenmesi istenirse ;



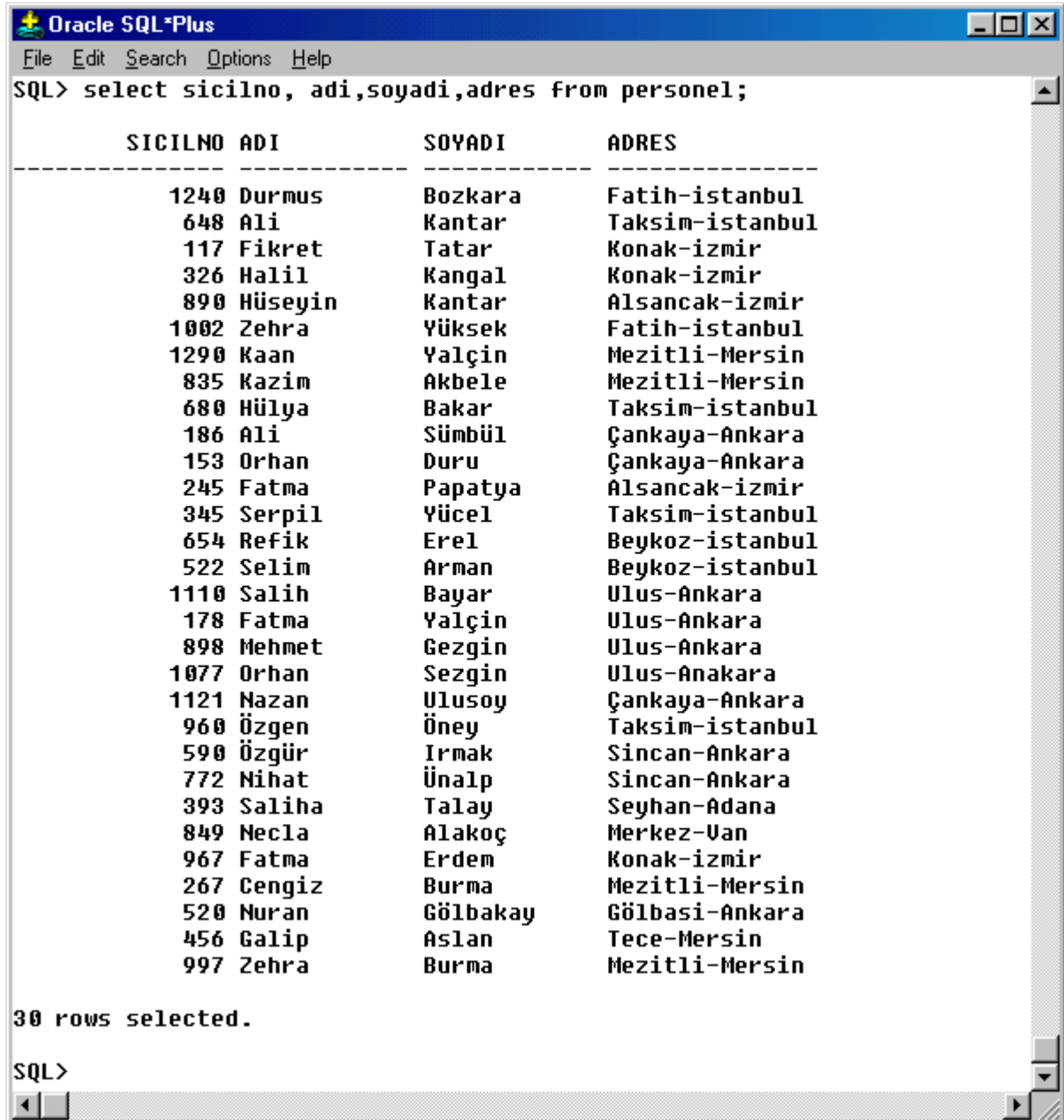
SICILNO	ADI	SOYADI	MESLEKKODU	BIRIMKODU	DOGUNTARI	ADRES
1240	Durmus	Bozkara	4	101	10-JAN-68	Fatih-istanbul
648	Ali	Kantar	2	101	01-FEB-72	Taksim-istanbul
117	Fikret	Tatar	5	101	12-MAR-80	Konak-izmir
326	Halil	Kangal	5	101	23-DEC-58	Konak-izmir
890	Hüseyin	Kantar	5	101	01-JAN-55	Alsancak-izmir
1002	Zehra	Yüksek	5	101	30-MAY-78	Fatih-istanbul
1290	Kaan	Yalçın	5	101	31-MAR-79	Mezitli-Mersin
835	Kazim	Akbele	3	201	20-MAY-66	Mezitli-Mersin
680	Hülya	Bakar	2	201	25-DEC-76	Taksim-istanbul
186	Ali	Sümbül	6	401	14-NOV-88	Çankaya-Ankara
153	Orhan	Duru	7	201	10-FEB-83	Çankaya-Ankara
245	Fatma	Papatya	7	201	15-SEP-66	Alsancak-izmir
345	Serpil	Yücel	5	201	18-JUN-70	Taksim-istanbul
654	Refik	Erel	5	201	20-APR-71	Beykoz-istanbul
522	Selim	Arman	7	201	23-JAN-77	Beykoz-istanbul
1110	Salih	Bayar	6	301	11-DEC-78	Ulus-Ankara
178	Fatma	Yalçın	6	301	13-DEC-80	Ulus-Ankara
898	Mehmet	Gezgin	8	301	27-MAY-76	Ulus-Ankara
1077	Orhan	Sezgin	8	301	30-MAR-79	Ulus-Ankara
1121	Nazan	Ulusoy	5	301	18-FEB-71	Çankaya-Ankara
960	Üzgen	Öney	2	301	16-JAN-70	Taksim-istanbul
590	Üzgür	İrmak	3	401	14-OCT-59	Sincan-Ankara
772	Nihat	Ünalp	8	401	12-NOV-87	Sincan-Ankara
393	Saliha	Talay	8	401	10-SEP-68	Seyhan-Adana
849	Necla	Alakoç	5	401	05-APR-83	Merkez-Uan
967	Fatma	Erden	5	401	02-MAR-69	Konak-izmir
267	Cengiz	Burma	1	101	15-FEB-58	Mezitli-Mersin
520	Nuran	Gölbakay	7	201	22-FEB-68	Gölbasi-Ankara
456	Galip	Aslan	8	101	08-JUN-70	Tece-Mersin
997	Zehra	Burma	6	301	23-DEC-68	Mezitli-Mersin

30 rows selected.

Şekil 8.1.1. Personel Tablosundaki Tüm Kayıtların Listelenmesi

Tablo içerisinde istenilen bir şarta uygun kayıtların listelenmesi istenirse “WHERE” yardımcı sözcüğü kullanılır ve sözcüğün yanına gerekli şart yazılır. Ayrıca select sözcüğünün yanına sadece listelenmesi istenilen alanların adları yazılabilir.

SQL dilinde where ifadesinden sonra yazılan şartta karşılaştırılan alan bilgisi rakamsal ifade ise aynen yazılarak, karakter ise alan bilgisi tek tırnak işareti arasına alınarak yazılmalıdır.



Oracle SQL*Plus

File Edit Search Options Help

SQL> select sicilno, adi, soyadi, adres from personel;

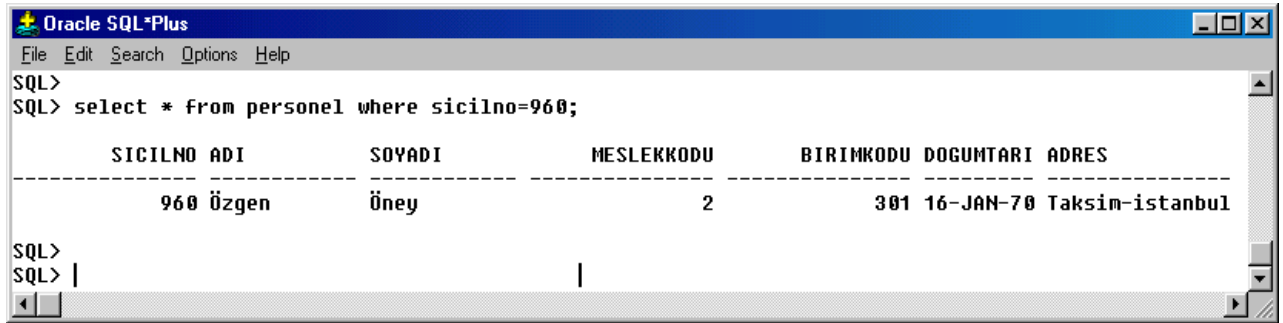
SICILNO	ADI	SOYADI	ADRES
1240	Durmus	Bozkara	Fatih-istanbul
648	Ali	Kantar	Taksim-istanbul
117	Fikret	Tatar	Konak-izmir
326	Halil	Kangal	Konak-izmir
890	Hüseyin	Kantar	Alsancak-izmir
1002	Zehra	Yüksek	Fatih-istanbul
1290	Kaan	Yalçın	Mezitli-Mersin
835	Kazim	Akbele	Mezitli-Mersin
680	Hülya	Bakar	Taksim-istanbul
186	Ali	Sümbül	Çankaya-Ankara
153	Orhan	Duru	Çankaya-Ankara
245	Fatma	Papatya	Alsancak-izmir
345	Serpil	Yücel	Taksim-istanbul
654	Refik	Erel	Beykoz-istanbul
522	Selim	Arman	Beykoz-istanbul
1110	Salih	Bayar	Ulus-Ankara
178	Fatma	Yalçın	Ulus-Ankara
898	Mehmet	Gezgin	Ulus-Ankara
1077	Orhan	Sezgin	Ulus-Ankara
1121	Nazan	Ulusoy	Çankaya-Ankara
960	Özgen	Öney	Taksim-istanbul
590	Özgür	Irmak	Sincan-Ankara
772	Nihat	Ünalp	Sincan-Ankara
393	Saliha	Talay	Seyhan-Adana
849	Necle	Alakoç	Merkez-Van
967	Fatma	Erdem	Konak-izmir
267	Cengiz	Burma	Mezitli-Mersin
520	Nuran	Gölbakay	Gölbasi-Ankara
456	Galip	Aslan	Tece-Mersin
997	Zehra	Burma	Mezitli-Mersin

30 rows selected.

SQL>

Şekil 8.1.2. Personel Tablosundan İstenilen Kayıtların Listelenmesi

Yukarıdaki örnekte personel tablosundan sadece sicilno, ad, soyad ve adres alanları listelenmiştir.



Şekil 8.1.3. Personel Bilgilerinin Seçilerek Listelenmesi-1

Sicil numarası 960 olan personel ait tüm bilgiler listelenmiştir.

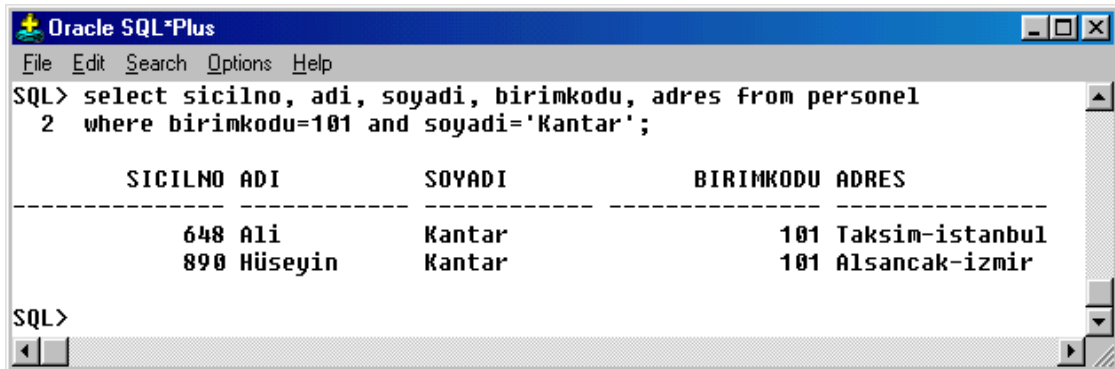
8.1.1. SQL Operatörleri

Her programlama dilinde olduğu gibi SQL'de de operatörler bulunur. Üç çeşit operatör mevcuttur. Karşılaştırma operatörleri, mantıksal operatörler ve kümeleme operatörleri.

Karşılaştırma operatörleri aşağıdaki gibidir;

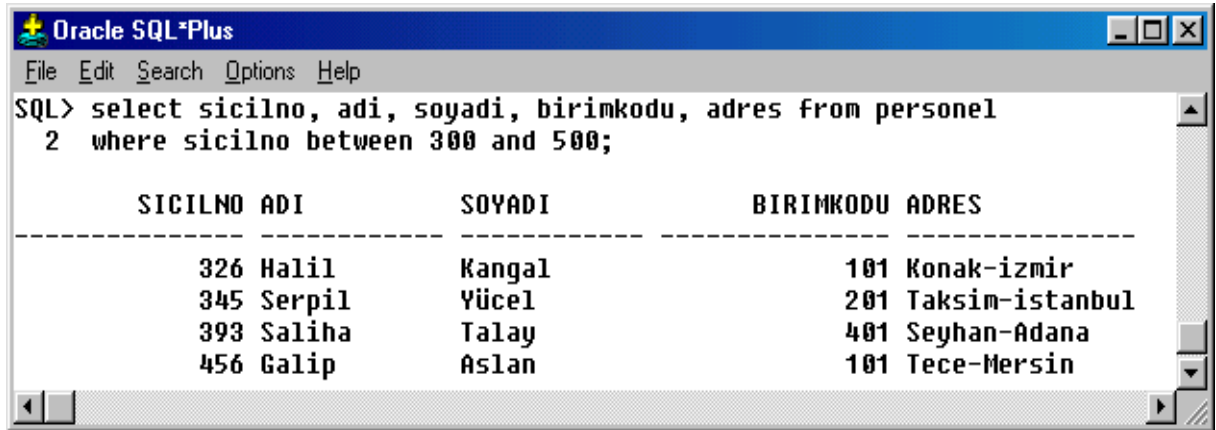
- a>X ... X... a'dan küçük
- a<X ... X... a'dan büyük
- a=X ... X... a'ya eşit
- a=>X ... X... a'dan küçük eşit
- a<=X ... X... a'dan büyük eşit
- a<>X ... X... a'ya eşit değil

Mantıksal operatörler ise **AND, OR, NOT** olarak verilebilir. Bu operatörler standart bütün dillerde aynı olan operatörler olduğu için ayrıca anlatılmayacaktır. Kümeleme operatörleri ise datalar üzerinde grupta yapmayı sağlayan operatörlerdir. Bunlar **Between, In, Like** operatörleridir. Bu operatörlerin hepsi where ile birlikte kullanılmalıdır.



Şekil 8.1.1.1. Personel Bilgilerinin Seçilerek Listelenmesi-2

Burada birim kodu 101 ve soyadı Kantar olan personele ait sicilno, ad, soyad, birim kodu ve adres bilgileri seçilerek listelenmiştir.



```
SQL> select sicilno, adi, soyadi, birimkodu, adres from personel
2  where sicilno between 300 and 500;
```

SICILNO	ADI	SOYADI	BIRIMKODU	ADRES
326	Halil	Kangal	101	Konak-izmir
345	Serpil	Yücel	201	Taksim-istanbul
393	Saliha	Talay	401	Seyhan-Adana
456	Galip	Aslan	101	Tece-Mersin

Şekil 8.1.1.2. Personel Bilgilerinin Seçilerek Listelenmesi-3

Between Aralıklı sorgulama yapılmak istenildiğinde kullanılan bir operatördür. Burada sicil numarası 300 ile 500 arasında olan personele ait sicilno, ad, soyad, birimkodu ve adres bilgileri **between and** deyimleri ile seçilerek listelenmiştir.



```
SQL> select sicilno, adi, soyadi, meslekkodu, adres from personel
2  where Meslekkodu IN (1,2,3,7);
```

SICILNO	ADI	SOYADI	MESLEKKODU	ADRES
648	Ali	Kantar	2	Taksim-istanbul
835	Kazim	Akbele	3	Mezitli-Mersin
680	Hülya	Bakar	2	Taksim-istanbul
153	Orhan	Duru	7	Çankaya-Ankara
245	Fatma	Papatya	7	Alsancak-izmir
522	Selim	Arman	7	Beykoz-istanbul
960	Özgen	Öney	2	Taksim-istanbul
590	Özgür	Irmak	3	Sincan-Ankara
267	Cengiz	Burma	1	Mezitli-Mersin
520	Nuran	Gölbakay	7	Gölbasi-Ankara

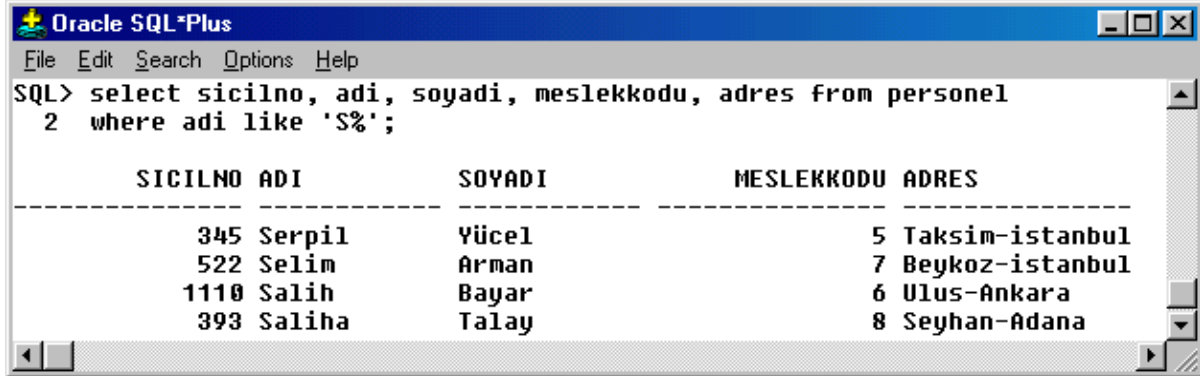
10 rows selected.

Şekil 8.1.1.3. Personel Bilgilerinin Seçilerek Listelenmesi-4

IN operatörü ile belli bir kolonun kümesi verilerek işlerin daha kolay yapılması sağlanır. Burada meslekkodu 1,2,3 ve 7 olan personele ait sicilno, ad, soyad, meslekkodu ve adres bilgileri **IN** operatörü ile seçilerek listelenmiştir.

```
SQL> select sicilno, adi, soyadi, birimkodu, adres from personel
2  where meslekkodu=1 or meslekkodu=2 or meslekkodu=3 or meslekkodu=7;
```

IN operatörü kullanılmadan bu komut yukarıdaki gibi de yazılabilirdi. Her iki komutta aynı işi yapar. Ama **IN** ile sorgunun yazılması daha kolaydır.



```

Oracle SQL*Plus
File Edit Search Options Help
SQL> select sicilno, adi, soyadi, meslekkodu, adres from personel
2 where adi like 'S%';

```

SICILNO	ADI	SOYADI	MESLEKKODU	ADRES
345	Serpil	Yücel	5	Taksim-istanbul
522	Selim	Arman	7	Beykoz-istanbul
1110	Salih	Bayar	6	Ulus-Ankara
393	Saliha	Talay	8	Seyhan-Adana

Şekil 8.1.1.4. Personel Bilgilerinin Seçilerek Listelenmesi-5

Like operatörü içinde belli bir karakter dizisi bulunan kayıtlara ulaşmak için kullanılır. Like operatöründe % işareti kullanılır. Burada personelin adı içerisinde S harfi geçen personele ait sicilno, ad, soyad, meslekkodu ve adres bilgileri Like deyimi ile seçilerek listelenmiştir. Eğer şart **'where adi like 'Se%'** olarak yazılmış olsaydı adı sadece Serpil ve Selim olan personel listelenirdi.

Örnek-1) **SELECT * FROM personel WHERE adres LIKE '%İstanbul%';**

Bu sorgulama ile adres alanında İstanbul geçen kayıtlar listelenir.

Örnek-2) **SELECT * FROM personel WHERE adres LIKE '%İstanbul';**

Bu sorgulama ile adres alanının sonunda İstanbul geçen kayıtlar listelenir.

Örnek-3) **SELECT * FROM personel WHERE adres LIKE 'İstanbul%';**

Bu sorgulama ile adres alanının başında İstanbul geçen kayıtları listelenir.

Örnek-4) **SELECT * FROM isci WHERE isciadi LIKE 'mu__%';**

İşçi tablosunda adı 'mu' ile başlayan kayıtların listesi.

Örnek-5) **SELECT * FROM isci WHERE isciadi LIKE 'mu__%';**

İşçi tablosunda adı 'mu' ile başlayan kayıtların listesi.

```

SQL> select * from isci where isciadi like 'mu__%';

```

ISCINO	ISCIADI	ISI	MUDURU	ISBASTAR	UCRET	BOLUM
7566	mustafa	yönetici	7839	02/04/0081	2975	20
7876	musa	tezgahtar	7788	12/01/0083	1100	20
7934	murat	tezgahtar	7782	23/01/0082	1300	10

```

SQL> select * from isci where isciadi like 'mu__%';

```

ISCINO	ISCIADI	ISI	MUDURU	ISBASTAR	UCRET	BOLUM
7566	mustafa	yönetici	7839	02/04/0081	2975	20
7934	murat	tezgahtar	7782	23/01/0082	1300	10

Şekil 8.1.1.5. Like Operatörü ile Tablodaki Bilgilerin Seçilerek Listelenmesi

Bu örnekte “_” alt tire işareti tek bir karaktere ve ‘%’ işareti birden fazla karaktere karşılık gelir. Birinci sorguda iki alt tire işareti kullanıldığında 3 adet kayıt listelenmiş, ikinci sorguda alt tire işareti üçe çıkarıldığında iki kayıt listelenmiştir. Çünkü ikinci sorguda istenen kayıtların isciadi alanının uzunluğu en az 5 karaktere çıkarılmış oluyor.

8.1.2. Order By

Bu komut ile belirtilen sütuna göre artan veya azalan bir sıralama ile sorgulama yapılabilir. Sıralama yapılacak olan alan sayılardan oluşuyorsa rakamsal değerler üzerinden artan veya azalan sıralama yapılır. Eğer alan karakterlerden oluşuyorsa sıralama A ‘dan – Z ‘ye veya Z ‘den – A ‘ya göre bir sıralama olacaktır.

ASC : kullanarak küçükten büyüğe doğru artan sıralama yapılabilir.

DESC : kullanarak büyükten küçüğe doğru azalan sıralama yapılabilir.

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

Örnek-1) *SQL> SELECT * FROM personel ORDER BY sicilno;*

Tüm personel sicil numarası alanına göre küçükten büyüğe doğru sıralanır.

Örnek-2) *SQL> SELECT * FROM personel ORDER BY sicilno ASC;*

Tüm personel sicil numarası alanına göre küçükten büyüğe doğru sıralanır.

Örnek-3) *SQL> SELECT * FROM personel ORDER BY sicilno DESC;*

Tüm personel sicil numarası alanına göre büyükten küçüğe doğru sıralanır.

Örnek-4) *SQL> SELECT * FROM işçi WHERE
2 soundex(isciadi)=soundex('ercin');*

SQL> select * from isci where soundex(isciadi)=soundex('ercin');						
ISCINO	ISCADI	ISI	MUDURU	ISBASTAR	UCRET	BOLUM
7900	ergün	tezgahtar	7698	03/12/0081	950	30

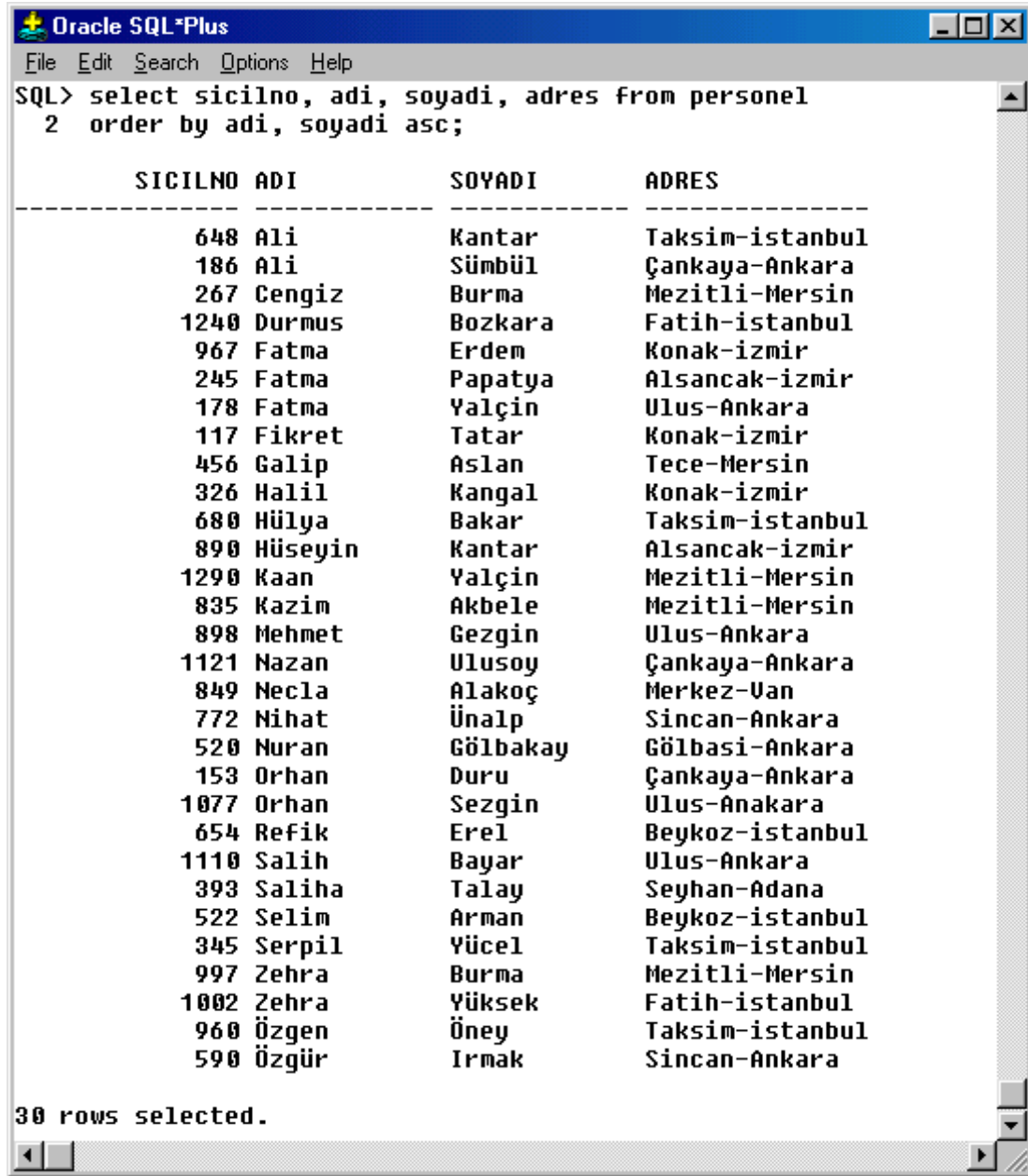
Şekil 8.1.2.1. Soundex Operatörü ile Tablodaki Bilgilerin Seçilerek Listelenmesi

Eğer tablodan listelenecek kaydın bir alanının içeriği tam olarak bilinmiyorsa ilgili kelimenin yakın telaffuzu yazılarak “soundex” fonksiyonu ile listeleme yapılabilir.

Örnek-5) *SQL> SELECT sicilno, adi, soyadi, adres FROM personel
2 ORDER BY adi, soyadi ASC;*

Tüm personele ait sicilno, ad, soyad ve adres bilgisi ad ve soyad alanına göre; büyükten küçüğe doğru sıralanır. Burada birden fazla alana göre sıralama yapılmıştır. Birden fazla alana göre sıralama yapıldığında önce birinci alan üzerinde sıralama yapılır. Daha sonra belirtilen ikinci alan da birinci alanlar içinde sıralanır.

Bu örnekte önce isimler sıralanmış sonra aynı isimli olan personel kendi içerisinde soyadlarına göre sıralanmıştır. Örneğin 3 tane Fatma isimli personel vardır. Bunlar alt alta sıralanmış olup soyadlarına göre de bir alfabetik sıralama yapılmıştır.



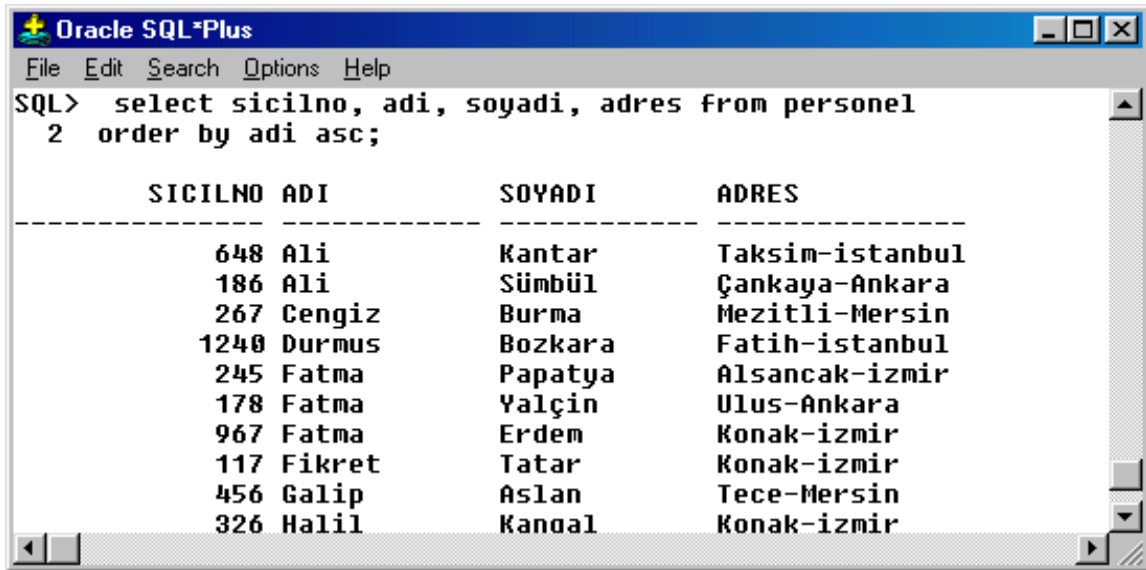
Oracle SQL*Plus window showing a query result. The query is: `SQL> select sicilno, adi, soyadi, adres from personel
2 order by adi, soyadi asc;`

SICILNO	ADI	SOYADI	ADRES
648	Ali	Kantar	Taksim-istanbul
186	Ali	Sümbül	Çankaya-Ankara
267	Cengiz	Burma	Mezitli-Mersin
1240	Durmus	Bozkara	Fatih-istanbul
967	Fatma	Erdem	Konak-izmir
245	Fatma	Papatya	Alsancak-izmir
178	Fatma	Yalçın	Ulus-Ankara
117	Fikret	Tatar	Konak-izmir
456	Galip	Aslan	Tece-Mersin
326	Halil	Kangal	Konak-izmir
680	Hülya	Bakar	Taksim-istanbul
890	Hüseyin	Kantar	Alsancak-izmir
1290	Kaan	Yalçın	Mezitli-Mersin
835	Kazim	Akbele	Mezitli-Mersin
898	Mehmet	Gezgin	Ulus-Ankara
1121	Nazan	Ulusoy	Çankaya-Ankara
849	Necle	Alakoç	Merkez-Van
772	Nihat	Ünalp	Sincan-Ankara
520	Nuran	Gölbakay	Gölbasi-Ankara
153	Orhan	Duru	Çankaya-Ankara
1077	Orhan	Sezgin	Ulus-Ankara
654	Refik	Erel	Beykoz-istanbul
1110	Salih	Bayar	Ulus-Ankara
393	Saliha	Talay	Seyhan-Adana
522	Selim	Arman	Beykoz-istanbul
345	Serpil	Yücel	Taksim-istanbul
997	Zehra	Burma	Mezitli-Mersin
1002	Zehra	Yüksek	Fatih-istanbul
960	Özgen	Öney	Taksim-istanbul
590	Özgür	Irmak	Sincan-Ankara

30 rows selected.

Şekil 8.1.2.2. Personel Bilgilerinin Seçilerek Listelenmesi-6

Sorgu sadece personelin adı alanına göre yapılmış olsaydı soyadlar isimlere göre alfabetik sırada olmayacaklardı. Daha uzun listelerde birden fazla alana göre sıralama yapmak listenin kontrolü ve kullanışlılığı açısından her zaman daha avantajlıdır. Bu duruma fakülte, bölümler, sınıf ve sınıfta okuyan öğrencilerin numaralarına göre sıralanması örneği verilebilir. Bir üniversitede okuyan 15 bin öğrenci bu kritere göre sıralanmadan; düzensiz bir sırada liste alınırsa aranılan kişinin nasıl bulunacağını düşünün.



Oracle SQL*Plus

File Edit Search Options Help

SQL> select sicilno, adi, soyadi, adres from personel
2 order by adi asc;

SICILNO	ADI	SOYADI	ADRES
648	Ali	Kantar	Taksim-istanbul
186	Ali	Sümbül	Çankaya-Ankara
267	Cengiz	Burma	Mezitli-Mersin
1240	Durmus	Bozkara	Fatih-istanbul
245	Fatma	Papatya	Alsancak-izmir
178	Fatma	Yalçın	Ulus-Ankara
967	Fatma	Erdem	Konak-izmir
117	Fikret	Tatar	Konak-izmir
456	Galip	Aslan	Tece-Mersin
326	Halil	Kanoal	Konak-izmir

Şekil 8.1.2.3. Personel Bilgilerinin Seçilerek Listelenmesi-7

Sorgu sadece personelin adı alanına göre yapılmıştır. Soyadlar isimlere göre alfabetik sırada değildir. Örneğin Fatma isimli personel isime göre sıralanmış ama Fatmalar kendi içerisinde soyadına göre alfabetik sıralanmamıştır.

Örnek-6) SQL> *SELECT * FROM personel
2 ORDER BY dogumtarihi DESC, adi, soyadi;*

Bu sorguda da birden fazla alana göre yani 3 alana göre sıralama yapılmıştır. Birinci alan azalan sıralama iken diğer iki alan artan sıralamadır. Burada personel kayıtları doğum tarihine göre büyükten küçüğe doğru sıralanıyor. Yani önce en genç eleman'dan başlanarak en yaşlı elemana doğru bir sıralama yapılıyor. Doğum tarihleri aynı olanlarda ise ad ve soyada göre A'dan Z'ye artan bir sıralama yapılmaktadır. Doğum tarihi aynı olanlar önce isimlere göre sıralanır. Burada aynı isimli olanlar alt alta gelecek şekilde sıralanır. Daha sonra da aynı isimli olanlar kendi içerisinde soyadlarına göre sıralanırlar.

8.1.3. Distinct

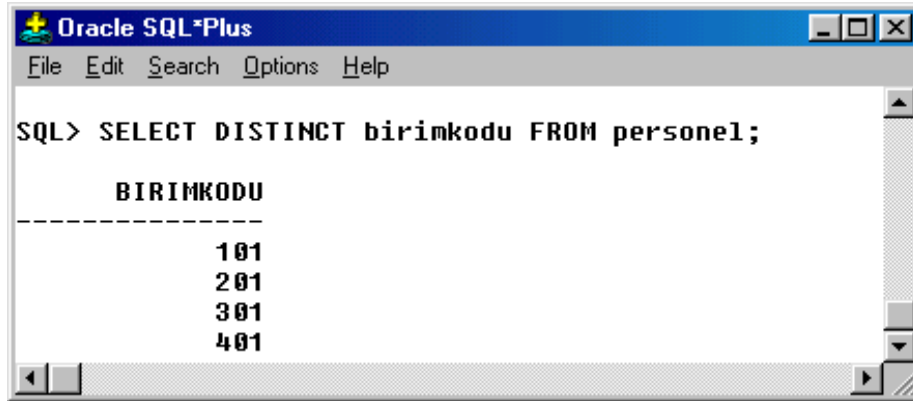
SQL'de bir tablo içinde bazı alanlarda birbirinin aynı olan kayıtlar bulunabilir. Aynı satırların listeleme esnasında bir kez yazılması için DISTINCT kullanılır.

Örnek-1) SQL> *SELECT DISTINCT soyad FROM personel ;*

Personel tablosundan soyad'lar tekrarsız olarak listelenecektir. Yani aynı soyada sahip iki kişi varsa sadece ilk kişi listelenir. Tekrarlar engellenmiş olur.

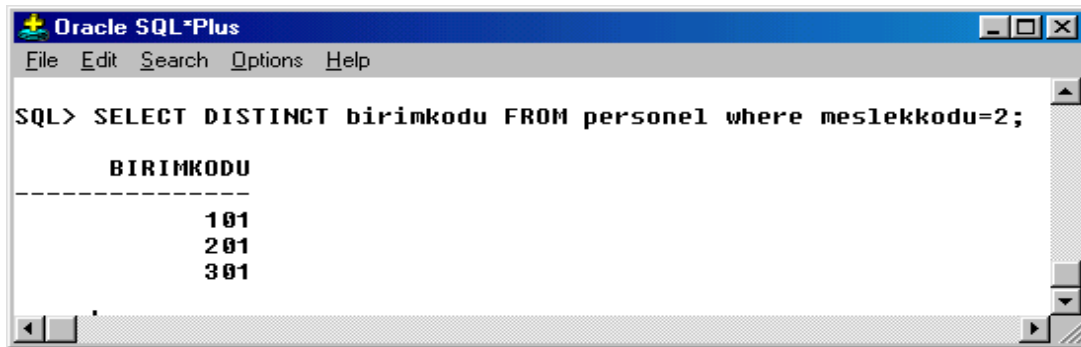
Örnek-2) SQL> *SELECT DISTINCT meslekKodu FROM personel ;*

Bu sorgu sonucunda Personel tablosunda farklı olan meslek grupları listelenir. Bu tür durumlarda DISTINCT çok kullanışlı bir operatördür.



Şekil 8.1.3.1. Personel Bilgilerinin DISTINCT ile Seçilerek Listelenmesi-1

Burada personel tablosunda olan birim kodları tekrarsız olarak listelenmiştir. Distinct operatörü kullanılmazaydı; 30 adet personelimiz olduğu için birim kodları tekrarlanarak ekrana 30 adet birim kodu bilgisi gelirdi.



Şekil 8.1.3.2. Personel Bilgilerinin DISTINCT ile Seçilerek Listelenmesi-2

Burada da personel tablosunda çalışan ve meslek kodu 2 olan mesleklerin birimleri tekrarsız olarak listelenmiştir. Bu sorgu sonucunda 2 nolu meslekten 3 farklı birimde çalışan personel olduğunu öğreniyoruz. 401 nolu birimde meslekkodu 2 olarak çalışan personel bulunmamaktadır.

8.2. UPDATE (Güncelle) Komutu

Tablodaki bir kayıt veya kayıtlar değiştirilmek istenirse UPDATE komutu kullanılır. Genel kullanım şekli aşağıdaki gibidir ;

```
UPDATE tablo_adı  
SET sütun_adı_1=deger1, sütun_adı_2=deger2,...  
WHERE koşul;
```

Örnek-1) `SQL> UPDATE personel`

`2 SET adres='Mezitli-Mersin' WHERE sicilno=849 ;`

Personel tablosunda sicil numarası 849 olan personelin adresi değiştirilmiştir.

Örnek-2) **SQL> UPDATE maaslar SET maasayi=2 ;**

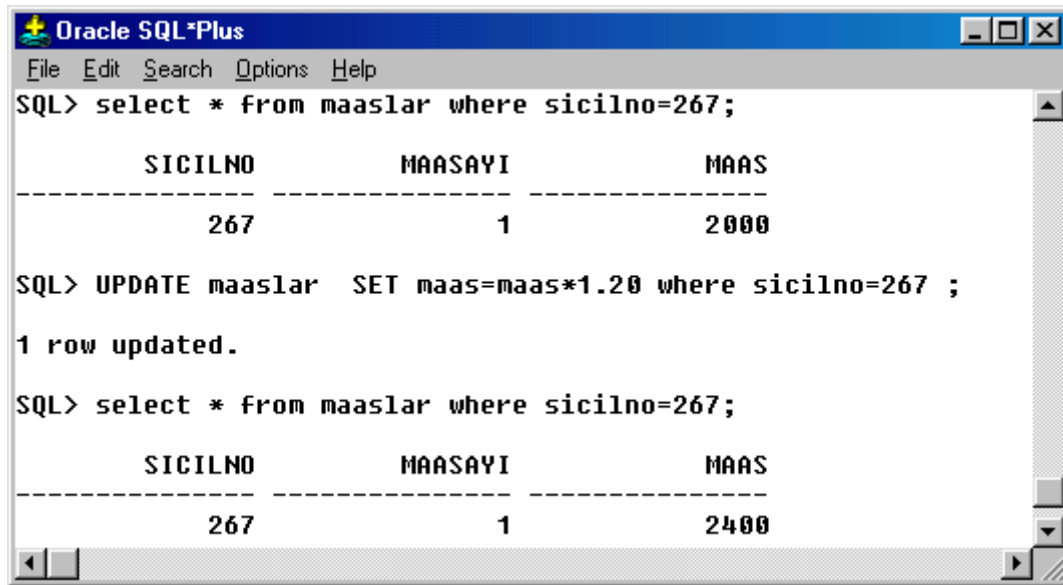
Maaşlar tablosunda bütün personelin maaş ayı 2 olarak değiştirilmiştir.

Örnek-3) **SQL> UPDATE maaslar SET maas=maas*1.20 ;**

Maaşlar tablosunda bütün personelin maaşına %20 zam yapılmıştır.

Örnek-4) **SQL> UPDATE maaslar SET maas=maas*1.20 where sicilno=267 ;**

Maaşlar tablosunda 267 sicil numaralı personelin maaşına %20 zam yapılmıştır. Diğer personelin maaşlarında bir değişiklik yapılmamıştır.



Şekil 8.2.1. Update Komutunun Kullanım Ekranı

Burada önce 267 sicil numaralı personele ait 2000 olan maaş listelenmiş ve update komutu ile kişiye %20 zam yapılmıştır. Daha sonra da zam aldıktan sonraki maaşı olan 2400 ekranda listelenmiştir.

Örnek-5) **SQL> UPDATE personel SET**

2 soyadi='Bakay', birimkodu=101, adres='Mezitli-Mersin'

3 WHERE sicilno=849 ;

Personel tablosunda sicil numarası 849 olan personelin soyadı Bakay, birimKodu 101 ve adresi Mezitli-Mersin olarak değiştirilmiştir.

Örnek-6) **SQL> UPDATE personel SET soyadi='Bakay';**

Personel tablosundaki tüm personelin soyadını Bakay olarak değiştirir.

NOT : Eğer yukarıdaki gibi bir SQL sorgu cümlesi yazılıp çalıştırılırsa tablodaki bütün personelin soyadı Bakay olarak değişir. Eğer bir yerde yedek bilginiz yoksa bilgilere yeniden ulaşılması söz konusu değildir. **Bu nedenle UPDATE komutu kullanılırken çok dikkat edilmelidir.**

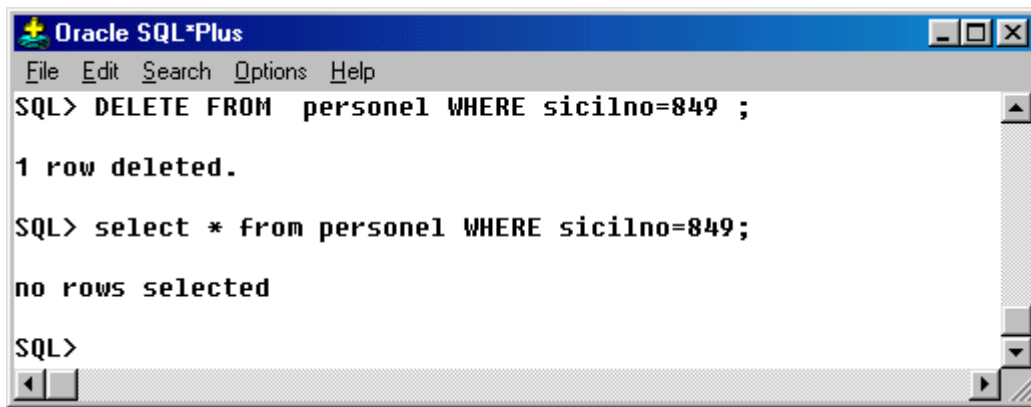
8.3. DELETE (Sil) Komutu

Tablodaki bir kayıt veya kayıtlar silinmek istenirse DELETE komutu kullanılır. Genel kullanım şekli aşağıdaki gibidir ;

DELETE FROM tablo_adı WHERE koşul;

Örnek-1) SQL> DELETE FROM personel WHERE sicilno=849 ;

Personel tablosunda sicil numarası 849 olan personeli siler.



Şekil 8.3.1. Delete Komutunun Kullanım Ekranı

Burada sicil numarası 849 olan personeli personel tablosundan silinmiştir. Daha sonra bu personel listelenmeye çalışıldığında böyle bir kayıt bulunamadığına dair “No rows selected” bilgisi ekrana gelmiştir.

Örnek-2) SQL> DELETE FROM personel WHERE birimkodu=101 ;

Personel tablosunda birimkodu 101 olan personelleri siler. Burada 101 nolu Birimde çalışan 9 personel olduğu için tablodan 9 adet personelin bilgisi silinecek ve geriye 21 adet personel bilgisi kalacaktır.

Örnek-3) SQL> DELETE FROM personel WHERE adi IS NULL ;

Personel tablosundan adı alanı boş olan personeli siler. IS NOT ile istenilen alanı boş olan bilgiler silinebilir. IS NOT NULL ise boş olmayan alanlar için kullanılır.

Örnek-4) SQL> DELETE FROM personel;

Personel tablosundaki tüm bilgileri siler.

NOT : Eğer yukarıdaki gibi bir SQL sorgu cümlesi yazılıp çalıştırılırsa tablodaki bütün kayıtlar silinir. Eğer bir yerde yedek bilginiz yoksa bilgilere yeniden ulaşılması söz konusu değildir. **Bu nedenle DELETE komutu kullanılırken çok dikkat edilmelidir.**

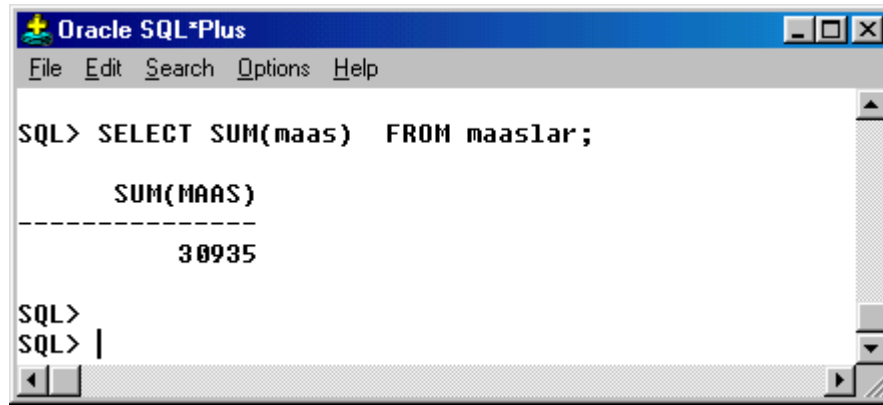
9. SQL 'de FONKSİYONLAR

9.1. SUM (Topla) Fonksiyonu

Belirtilen sütundaki bütün değerlerin toplamını alır. Genel kullanımı aşağıdaki gibidir.

Select SUM(sütun_adi) FROM tablo_adi;

Örnek-1) *SQL> SELECT SUM(maas) FROM maaslar;*
Tüm personele ödenen toplam maaş miktarını verir.



Şekil 9.1.1. Sum Fonksiyonu Ekranı

Örnek-2) *SQL> SELECT SUM(maas) FROM maaslar WHERE maasayi=1;*
Personele ait 1. ay için ödenen toplam maaş miktarını verir.

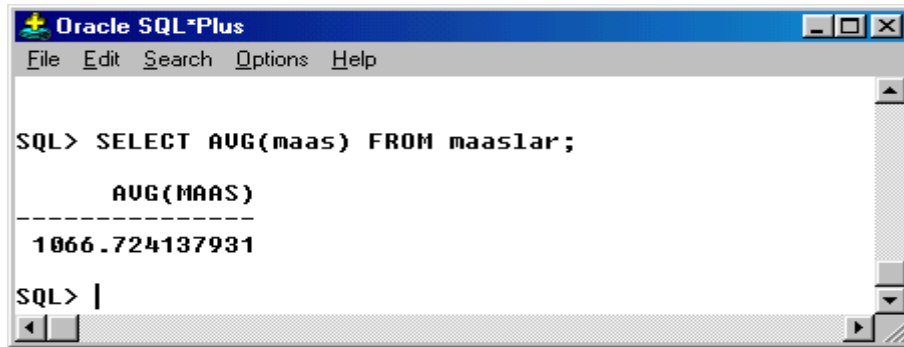
9.2. AVG (Ortalama) Fonksiyonu

Belirtilen sütundaki bütün değerlerin ortalamasını alır. Genel kullanımı aşağıdaki gibidir.

Select AVG(sütun_adi) FROM tablo_adi;

Örnek-1) *SQL> SELECT AVG(maas) FROM maaslar;*
Tüm personele ödenen maaşların ortalamasını verir.

Örnek-2) *SQL> SELECT AVG(maas) FROM maaslar WHERE maasayi=1;*
Personele ait 1. ay için ödenen maaşların ortalamasını verir.



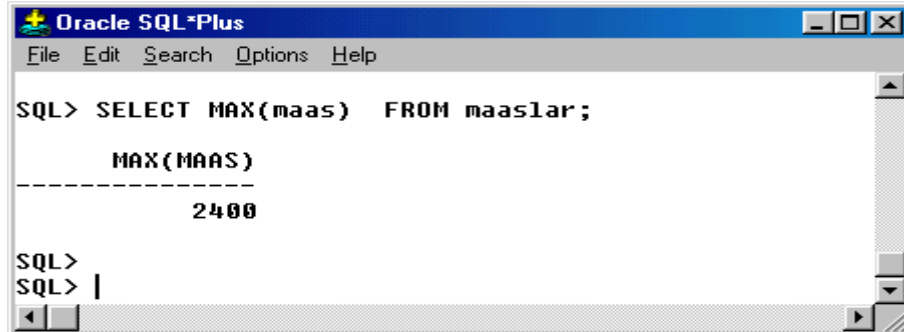
Şekil 9.2.1. Avg Fonksiyonu Ekranı

9.3. MAX (En Büyük) Fonksiyonu

Belirtilen sütundaki en büyük değeri verir. Genel kullanımı aşağıdaki gibidir.

`Select MAX(sütun_adı) FROM tablo_adı;`

Örnek-1) `SQL> SELECT MAX(maas) FROM maaslar;`
En yüksek maaş bilgisini verir.



Şekil 9.3.1. Max Fonksiyonu Ekranı

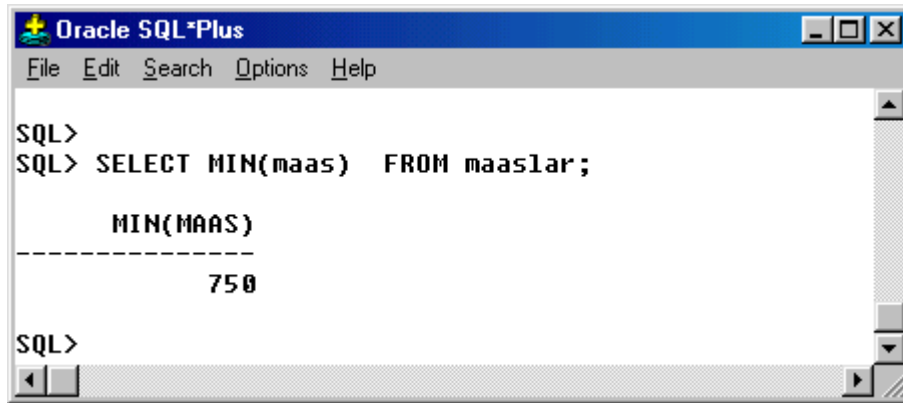
Örnek-2) `SQL> SELECT MAX(maas) FROM maaslar WHERE maasayi=1;`
1. ay en yüksek maaş bilgisini verir.

9.4. MIN (Em Küçük) Fonksiyonu

Belirtilen sütundaki en küçük değeri verir. Genel kullanımı aşağıdaki gibidir.

`Select MIN(sütun_adı) FROM tablo_adı;`

Örnek-1) `SQL> SELECT MIN(maas) FROM maaslar;`
En düşük maaş bilgisini verir.



Şekil 9.4.1. Min Fonksiyonu Ekranı

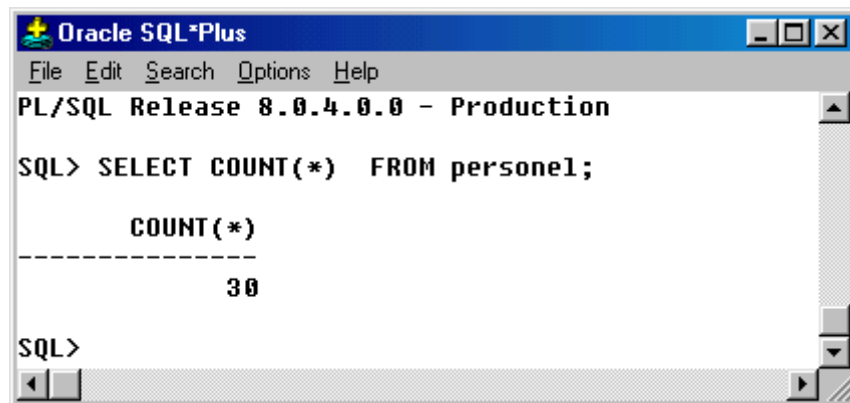
Örnek-2) `SQL> SELECT MIN(maas) FROM maaslar WHERE maasayi=1;`
1. ay en düşük maaş bilgisini verir.

9.5. COUNT (Say) Fonksiyonu

Tablo içinde, her hangi bir sayma işlemi gerçekleştirmek için kullanılır. COUNT fonksiyonu sorgu sonucu dönen kayıt sayısını verir. Eğer parametre olarak "*" girilirse tablodaki tüm kayıt sayısını verir. Parametre olarak bir sütun adı verilirse, o sütundaki içeriği NULL (boşluk) olmayan tüm kayıt sayısını verir. Genel kullanımı aşağıdaki gibidir.;

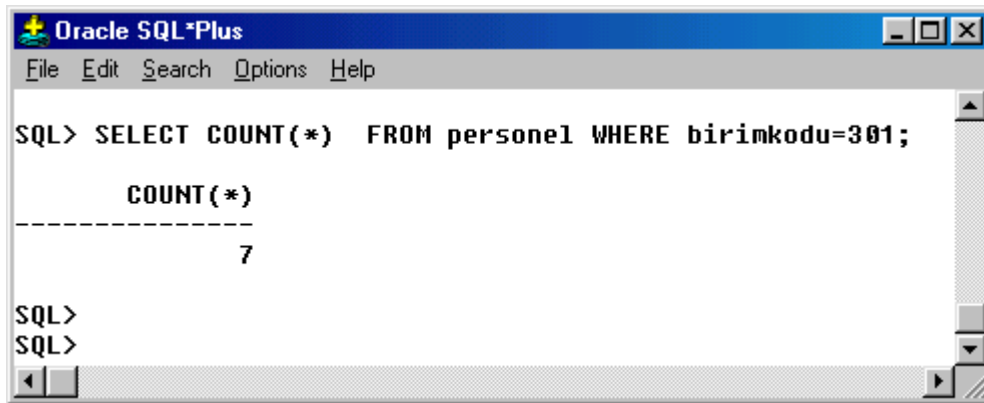
`Select COUNT(sütun_adı) FROM tablo_adı;`

Örnek-1) `SQL> SELECT COUNT(*) FROM personel;`
Kaç tane personelimiz olduğunu listeler.



Şekil 9.5.1. Count Fonksiyonu Ekranı-1

Örnek-2) `SQL> SELECT COUNT(*) FROM personel WHERE birimkodu=301;`
Birim kodu 301 olan Bilgi İşlem bölümünde çalışan personel sayısını verir.



Şekil 9.5.2. Count Fonksiyonu Ekranı-2

Örnek-3) *SQL> SELECT COUNT(*) FROM maaslar WHERE maas>=1300;*
Maaşı 1300 'e eşit ve daha fazla olan personel sayısını verir.

DISTINCT fonksiyonu COUNT fonksiyonu ile birlikte kullanılabilir. Verilen kolondaki unique record sayısını geri döndürür. Genel yazım biçimi aşağıdaki gibidir ;

Select COUNT(DISTINCT sütun_adı) FROM tablo_adı;

Örnek-4) *SQL> SELECT COUNT(DISTINCT adi) FROM personel;*
Kaç farklı isme sahip personel sayısını verir.

Örnek-5) *SQL> SELECT COUNT(DISTINCT maas) FROM maaslar;*
Çalışan personele ödenen kaç farklı maaş olduğunun sayısını verir.

Örnek-6) *SQL> SELECT COUNT(DISTINCT birimKodu) FROM personel;*
Kaç farklı birimde çalışan olduğunun sayısını verir.

Örnek-7) *SQL> SELECT COUNT(DISTINCT meslekKodu) FROM personel;*
Şirkette çalışan kaç farklı meslek grubu olduğunun sayısını verir.

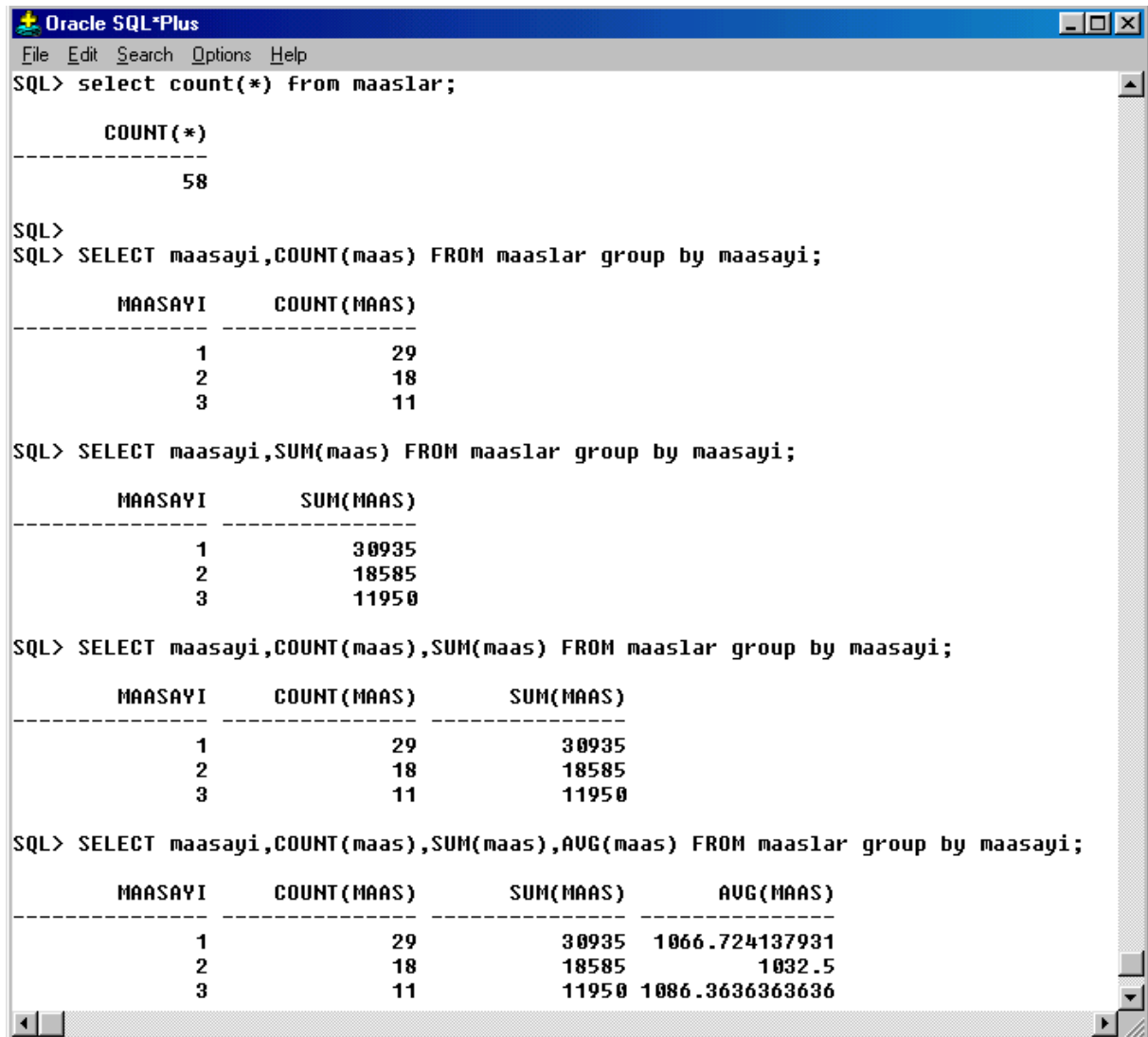
Not : SQL'de bu bölümde anlatılan fonksiyonların dışında pek çok fonksiyon bulunmaktadır. Örnek olarak; SUBSTR, LENGTH, LTRIM, RTRIM, to_char, to_number, to_date gibi. Bu bölümde en çok kullanılan fonksiyonlar anlatılmıştır.

10. SQL 'de GRUPLANDIRMA

10.1. GROUP BY (Gruplandır) Deyimi

“GROUP BY” yardımcı sözcüğü bir alana göre kayıtları gruplamak için kullanılır. Genel kullanımı aşağıdaki gibidir.

```
SELECT [ DISTINCT | ALL ] <sütun(lar)> FROM <tablo adı (lar)>
[ WHERE <şart (lar)> ]
[ GROUP BY <sütunlar>]
```



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> select count(*) from maaslar;

COUNT(*)
-----
58

SQL>
SQL> SELECT maasayi,COUNT(maas) FROM maaslar group by maasayi;

MAASAYI      COUNT(MAAS)
-----
1              29
2              18
3              11

SQL> SELECT maasayi,SUM(maas) FROM maaslar group by maasayi;

MAASAYI      SUM(MAAS)
-----
1          30935
2          18585
3          11950

SQL> SELECT maasayi,COUNT(maas),SUM(maas) FROM maaslar group by maasayi;

MAASAYI      COUNT(MAAS)      SUM(MAAS)
-----
1              29          30935
2              18          18585
3              11          11950

SQL> SELECT maasayi,COUNT(maas),SUM(maas),AVG(maas) FROM maaslar group by maasayi;

MAASAYI      COUNT(MAAS)      SUM(MAAS)      AVG(MAAS)
-----
1              29          30935      1066.724137931
2              18          18585          1032.5
3              11          11950      1086.3636363636
```

Şekil 10.1.1. Group By Ekranı-1

Maaşlar tablosunda bulunan personelin bazılarına group by ifadesini açıklamak üzere 2. ve 3.aya ait maaş bilgileri eklenmiştir. Yukarıdaki ekranda yapılan eklemeler sonunda 5 adet sorgu yapılmıştır. Bunlar ve açıklamaları aşağıdaki örnekte açıklanmıştır. Buradaki örneklerden de görüldüğü gibi bir sorguda birden fazla fonksiyon bir arada kullanılabilir.

Örnek-1) SQL> select count(*) from maaslar;

Maaslar tablosunda bulunan toplam kayıt sayısı 58 olarak bulunmuştur. Tabloda 58 personeler ait 1., 2. ve 3. ay maaş bilgileri bulunmaktadır.

COUNT(*)

```
-----  
58
```

Örnek-2)SQL>SELECT maasayi,COUNT(maas) from maaslar group by maasayi;

Her bir ay için kaç personele maaş ödendiğini listeleyen sorgu.

MAASAYI COUNT(MAAS)

```
-----  
1      29  
2      18  
3      11
```

Örnek-3) SELECT maasayi,SUM(maas) FROM maaslar group by maasayi;

Her bir ay için personele ödenen toplam maaş miktarını hesaplayan sorgu.

MAASAYI SUM(MAAS)

```
-----  
1      30935  
2      18585  
3      11950
```

Örnek-4) SQL> SELECT maasayi, COUNT(maas), SUM(maas) FROM maaslar group by maasayi;

Her bir ay için kaç personele, ne kadar toplam maaş ödendiğini hesaplayan sorgu.

MAASAYI COUNT(MAAS) SUM(MAAS)

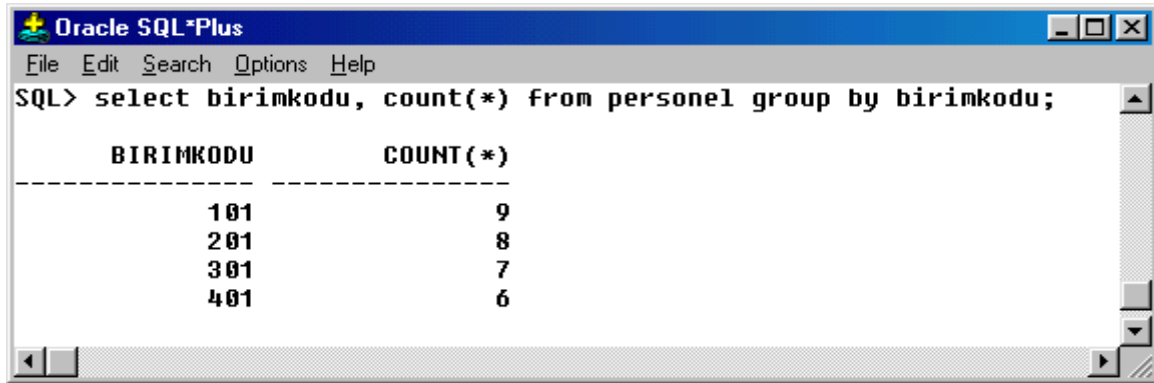
```
-----  
1      29      30935  
2      18      18585  
3      11      11950
```

Örnek-5) SQL> SELECT maasayi, COUNT(maas), SUM(maas), AVG(maas) FROM maaslar group by maasayi;

Her bir ay için kaç personele, ne kadar toplam maaş ödendiği ve her ayın maaş ortalamasını hesaplayan sorgu.

MAASAYI COUNT(MAAS) SUM(MAAS) AVG(MAAS)

```
-----  
1      29      30935      1066.724137931  
2      18      18585      1032.5  
3      11      11950      1086.3636363636
```

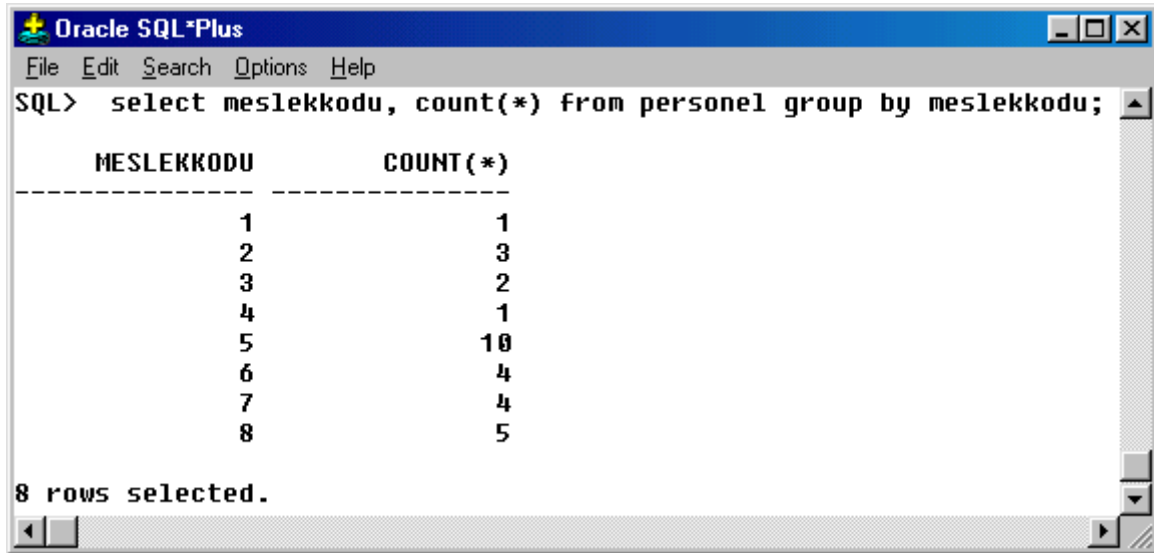



```
SQL> select birim kodu, count(*) from personel group by birim kodu;
```

BIRIMKODU	COUNT(*)
101	9
201	8
301	7
401	6

Şekil 10.1.2. Group By Ekranı-2

Yukarıdaki örnekte; personel tablosu birim koduna göre gruplandırma yapılarak her bir birimde çalışan personel sayısı listelenmiştir.



```
SQL> select meslekkodu, count(*) from personel group by meslekkodu;
```

MESLEKKODU	COUNT(*)
1	1
2	3
3	2
4	1
5	10
6	4
7	4
8	5

8 rows selected.

Şekil 10.1.3. Group By Ekranı-3

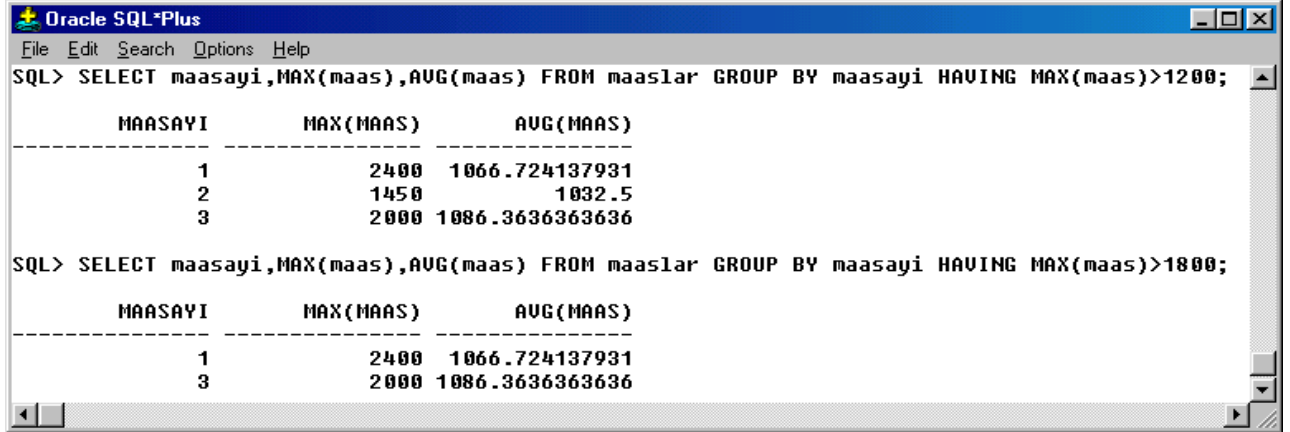
Yukarıdaki örnekte; personel tablosu meslek koduna göre gruplandırma yapılarak her bir meslek grubunda çalışan personel sayısı listelenmiştir.

10.2.HAVING (Sahip) Deyimi

Gruplandırarak kümeleme fonksiyonu kullanılırken, sorguda bir koşulunda verilmesi gerekiyorsa kullanılır. Bu durumda grup üzerindeki hesaplamalarla ilgili koşul belirtilirken HAVING (SAHİP) sözcüğü kullanılır. “HAVING” yardımcı sözcüğü “GROUP BY” yardımcı sözcüğü ile gruplanan kayıtlar üzerinde kısıtlama yapma işine yarar. Genel kullanımı aşağıdaki gibidir.

```
SELECT [ DISTINCT | ALL ] <sütun(lar)> FROM <tablo adı (lar)>  
[ WHERE <şart (lar)> ]  
[ GROUP BY <sütunlar> ]  
[ HAVING < grup kısıtlaması> ]
```

HAVING sözcüğü **SELECT** komutunda **GROUP BY** bulunmadığı zaman geçersizdir. **HAVING** sözcüğünü izleyen ifade içinde **SUM**, **COUNT(*)**, **AVG**, **MAX** yada **MIN** fonksiyonlarından en az biri bulunmalıdır. **HAVING** sözcüğü sadece gruplanmış veriler üzerindeki işlemlerde geçerlidir. **WHERE** sözcüğü bir tablonun tek satırları üzerinde işlem yapan koşullar içinde geçerlidir. Bazı durumlarda **HAVING** ve **WHERE** sözcükleri ile birlikte **SELECT** komutu içinde kullanılabilir.



Oracle SQL*Plus

File Edit Search Options Help

SQL> SELECT maasayi,MAX(maas),AVG(maas) FROM maaslar GROUP BY maasayi HAVING MAX(maas)>1200;

MAASAYI	MAX(MAAS)	AVG(MAAS)
1	2400	1066.724137931
2	1450	1032.5
3	2000	1086.3636363636

SQL> SELECT maasayi,MAX(maas),AVG(maas) FROM maaslar GROUP BY maasayi HAVING MAX(maas)>1800;

MAASAYI	MAX(MAAS)	AVG(MAAS)
1	2400	1066.724137931
3	2000	1086.3636363636

Şekil 10.2.1. Having Ekranı-1

Örnek-1) SQL> SELECT maasayi, MAX(maas), AVG(maas) FROM maaslar GROUP BY maasayi HAVING MAX(maas)>1200 ;

En yüksek maaşın 1200 den fazla olduğu aylardaki personele ait en yüksek maaş ve ortalama maaşı listeler. Maaş aylara göre gruplandırılmıştır.

Örnek-2) SQL> SELECT maasayi, MAX(maas), AVG(maas) FROM maaslar GROUP BY maasayi HAVING MAX(maas)>1000 ;

En yüksek maaşın 1000 den fazla olduğu aylardaki personele ait en yüksek maaş ve ortalama maaşı listeler. Maaş aylara göre gruplandırılmıştır.

Örnek-3) SQL> SELECT böl_no, AVG (brüt) FROM isciler WHERE cins= .T. GROUP BY böl_no HAVING AVG (brüt) > 9000000;

İsci tablosu içinde her bölümde erkek personele ait maaşlar için ortalamanın 9000000'dan fazla olduğu bölümleri listeler. Bu tabloda cins isimli bir tablo alanı olduğu ve onunda mantıksal bir alan olarak tanımlandığı varsaymıştır. Mantıksal alanın true olması durumunda cinsiyetin erkek, false olması durumunda cinsiyetin bayan olduğu varsayılarak sorgu yazılmıştır. Mantıksal veriler için mümkün olabilen sadece iki değer söz konusudur. DOĞRU D(TRUE T), YANLIŞ Y (FALSE F) ile simgelenir. Mantıksal alanlar .T. veya .F. olarak karşılaştırılırlar.

SQL> select isi,count(*) from isci group by isi having count(*)>2;

ISI	COUNT(*)
satış gör	3
tezgahtar	4

Şekil 10.2.1. Having Ekranı-2

Örnek; isci tablosunda ikiden fazla kişi tarafından yapılan işlerin listesini verir.

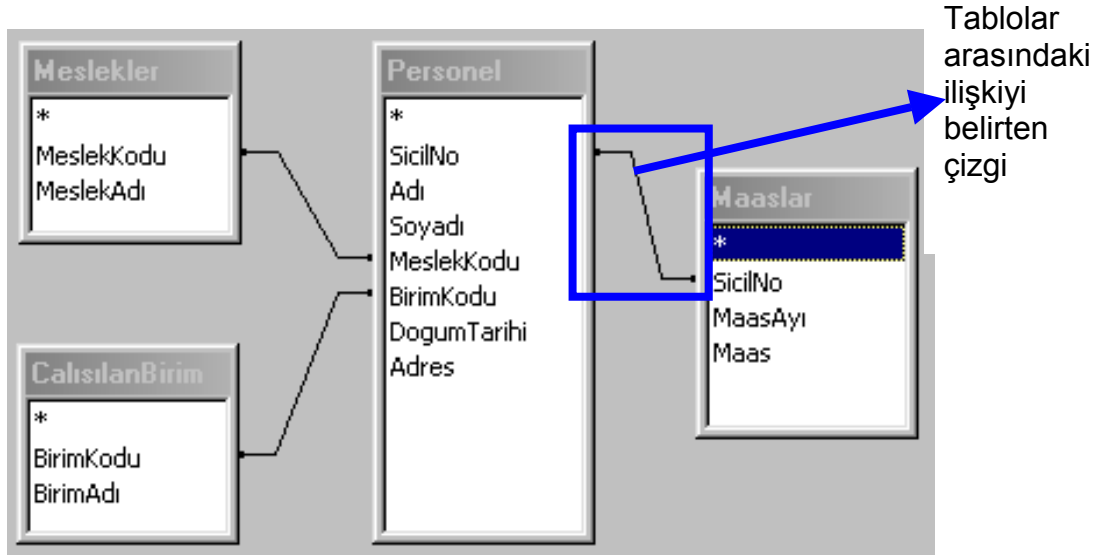
11. SQL 'de JOIN (BİRLEŞTİRME) İŞLEMİ

11.1. JOIN (Birleştirme) İşlemi

Veri tabanı kayıtları oluşturulurken bütün bilgiler bir tabloda değil de, birkaç tablo üzerinde tutulur. Bu dataların daha düzenli olmasını, gereksiz veri tekrarlarının engellenmesini ve veri yönetimini kolaylaştırır. Daha önce bu konu ile ilgili 4. ve 5.Bölümde veri tabanı ve normalizasyon konusundan detayları ile bahsedilmiş ve örnekler verilmiştir.

Bu bölüme kadar yazılan tüm sorgularda tek tablo üzerinde işlem yapıldı. Bu bölümde, ilişkisel veri tabanlarının ve SQL'in çok önemli ve yararlı bir özelliği olan birleştirme ("join") işlemi anlatılacaktır. İlişkisel veri tabanlarına "ilişkisel" denmesinin nedeni olan Birleştirme işlemi iki veya daha çok tablo arasında bağlantı ("link") kurarak tek bir tablo oluşturur ve sorgu bu tablo üzerinde çalışır .

Birden fazla tablo üzerinde işlem yapılacağı zaman; select cümlecikinden sonra tabloadı.tablo_alanları aralarına virgül konularak yazılır ve kullanılacak tablolar from cümlecikinden sonra aralarına virgül konularak yazılır. Daha sonra da WHERE şartı ile tablolar arasındaki ilişkili ortak alanlar eşitlenir. Buradaki en önemli özelliğin iki tablo arasında ilişkili ortak bir alanın olması gerektiğidir. Aşağıda projemize ait tablolar ve ilişkili alanlar belirtilmiştir.



Şekil 11.1.1. Örnek Personel Projenin MS Access deki diyagramı

<u>Tablo İsimleri</u>	<u>Ortak Alanlar</u>
Personel-Maaş tablosu	SicilNo
Personel-Meslekler tablosu	MeslekKodu
Personel-CalıslanBirim tablosu	BirimKodu

Projedeki her tablonun bir alanı ana tablo olan personel tablosunun bir alanı ile ortaktır ve tanımlanan alanlar aynı özelliktedir.

Örnek-1) *SQL> select * from personel where birimkodu=301;*
BirimKodu 301 olan personelin listesini verir.



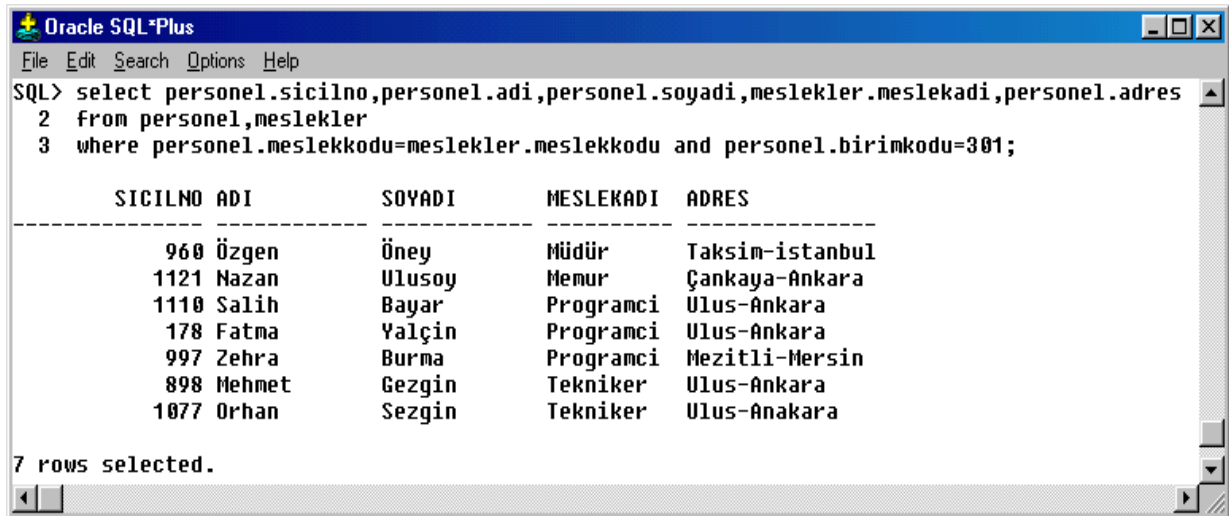
```
SQL> select * from personel where birimkodu=301;
```

SICILNO	ADI	SOYADI	MESLEKKODU	BIRIMKODU	DOGUNTARI	ADRES
1110	Salih	Bayar	6	301	11-DEC-78	Ulus-Ankara
178	Fatma	Yalçın	6	301	13-DEC-80	Ulus-Ankara
898	Mehmet	Gezgin	8	301	27-MAY-76	Ulus-Ankara
1077	Orhan	Sezgin	8	301	30-MAR-79	Ulus-Anakara
1121	Nazan	Ulusoy	5	301	18-FEB-71	Çankaya-Ankara
960	Özgen	Öney	2	301	16-JAN-70	Taksim-istanbul
997	Zehra	Burma	6	301	23-DEC-68	Mezitli-Mersin

7 rows selected.

Şekil 11.1.2. Join Sorgu Ekranı-1

Bu sorgu sonucunda personele ait birimler birimKodu 301 olarak ve mesleklerde meslekkodu olarak listelenmektedir. Kod olarak listelenen bilgiler çok açıklayıcı değildir. Örneğin Salih Başar isimli personelin meslekkodu 6 olarak görüntülenmekte ama 6 kodunun hangi meslek olduğu anlaşılamamaktadır. Personel tablosundaki meslekkodunun karşılığı olan değer meslekler tablosundan bulunarak getirilmelidir. Bu meslekkodlarının hangi mesleğe denk geldiğini bulmak için personel ve meslekler tablosu arasında join yapılmalıdır. Şöyleki; personel tablosunda Salih Bayar isimli personele ait 6 olan meslek kodu meslekler tablosunda aranacak ve meslekler tablosunda meslekkodu 6 olan kayıta bulunan programcı mesleği personelin mesleği olarak yazılacaktır. Bu işlemden sonra meslek alanına programcı yazılacak ve veriler daha anlamlı hale gelecektir. Bu karşılaştırma işlemini yapan ifade WHERE karşılaştırmasından sonra kullanılacak olan ifadedir.



```
SQL> select personel.sicilno,personel.adi,personel.soyadi,meslekler.meslekadi,personel.adres
2 from personel,meslekler
3 where personel.meslekkodu=meslekler.meslekkodu and personel.birimkodu=301;
```

SICILNO	ADI	SOYADI	MESLEKADI	ADRES
960	Özgen	Öney	Müdür	Taksim-istanbul
1121	Nazan	Ulusoy	Memur	Çankaya-Ankara
1110	Salih	Bayar	Programci	Ulus-Ankara
178	Fatma	Yalçın	Programci	Ulus-Ankara
997	Zehra	Burma	Programci	Mezitli-Mersin
898	Mehmet	Gezgin	Tekniker	Ulus-Ankara
1077	Orhan	Sezgin	Tekniker	Ulus-Anakara

7 rows selected.

Şekil 11.1.2. Join Sorgu Ekranı-2

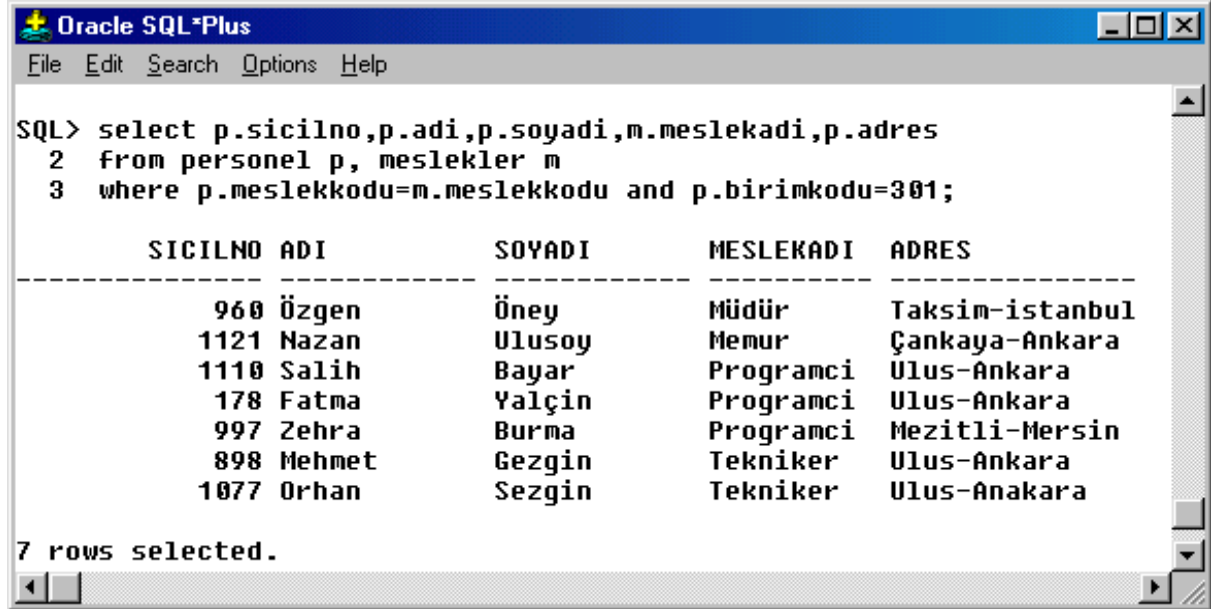
Sorgu çok uzun ve karışık gibi görünmesine rağmen aslında çok basittir. Select komutunun ardından birden fazla tablo kullanıldığı için alanın hangi tabloya ait olduğunu bildirmek için listelenecek her alan için **tabloadı.tablo_alanı** kullanılmıştır. **Personel.sicilno** veya **meslekler.meslekkodu** gibi. From komutundan sonra kullanılan tablolar virgül ile ayrılarak yazılmıştır. Daha sonra personel tablosundaki meslekkodunun meslekler tablosundaki karşılığını bulmak için

where personel.meslekkodu=meslekler.meslekkodu karşılaştırması yapılmıştır.

and personel.birimkodu=301 ifadesi tüm personel yerine 301 nolu birimde çalışan personelin listelenmesi içindir.

and And ile iki şart birbiri ile birleştirilmiştir.

Yukarıdaki sorgu aşağıdaki gibi de yazılabilir.



Oracle SQL*Plus

File Edit Search Options Help

SQL> select p.sicilno,p.adi,p.soyadi,m.meslekadi,p.adres
2 from personel p, meslekler m
3 where p.meslekkodu=m.meslekkodu and p.birimkodu=301;

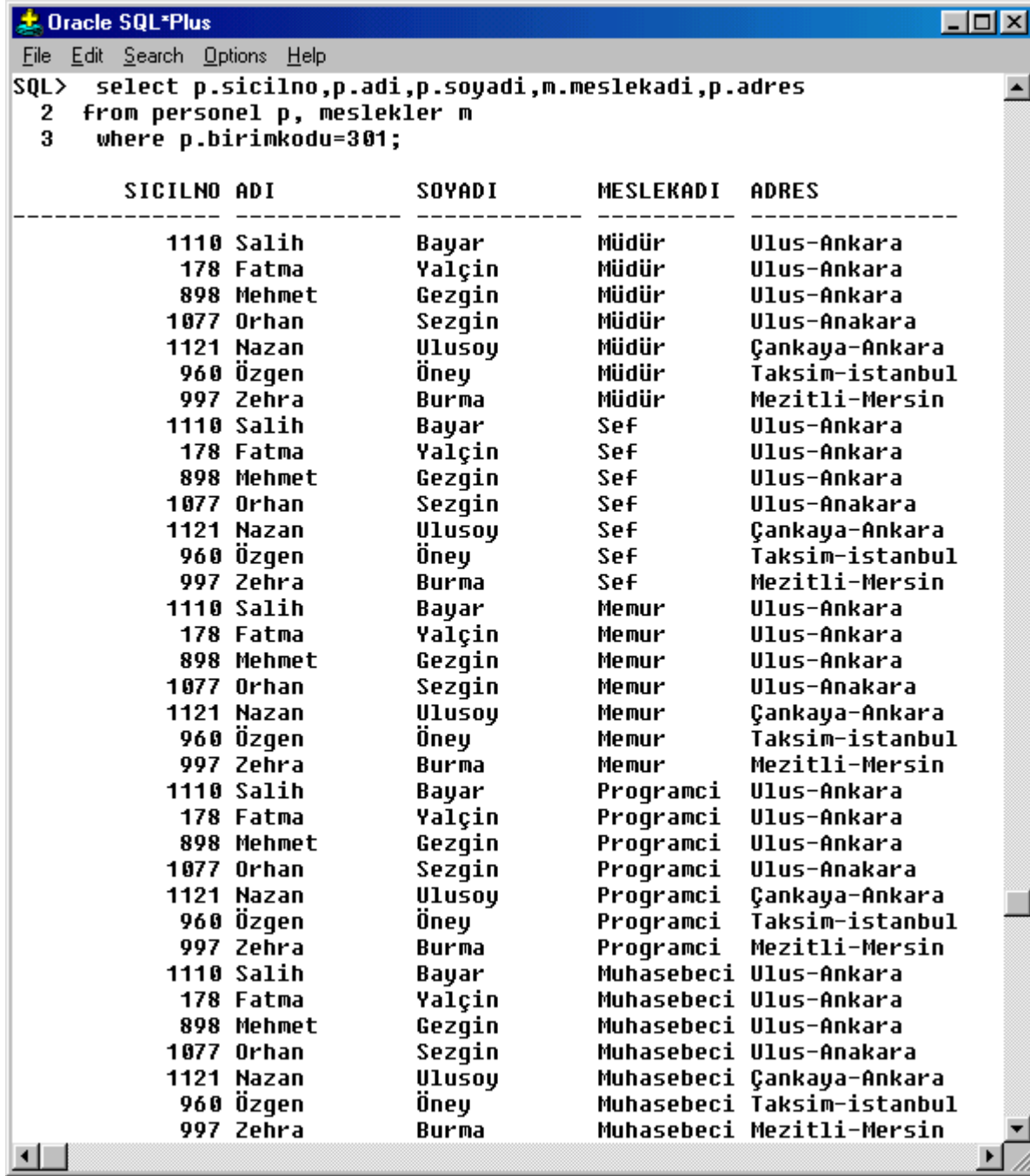
SICILNO	ADI	SOYADI	MESLEKADI	ADRES
960	Özgen	Öney	Müdür	Taksim-istanbul
1121	Nazan	Ulusoy	Memur	Çankaya-Ankara
1110	Salih	Bayar	Programci	Ulus-Ankara
178	Fatma	Yalçın	Programci	Ulus-Ankara
997	Zehra	Burma	Programci	Mezitli-Mersin
898	Mehmet	Gezgin	Tekniker	Ulus-Ankara
1077	Orhan	Sezgin	Tekniker	Ulus-Anakara

7 rows selected.

Şekil 11.1.3. Join Sorgu Ekranı-3

İki sorguda aynıdır. Tek fark yazılım farkıdır. Dikkat edilirse p.sicilno gibi ifadeler vardır. Burada olduğu gibi "from" sözcüğünden sonra yazılan tablo adları için bir boşluk bıraktıktan sonra kısa bir isim verilebilir. Böylece uzun tablo adını sürekli yazmaktansa o tablo adı için "alias" olarak adlandırılan kısa isim kullanılabilir. Alias olarak verilen tablo ismi alanlar listelenirken yazım olarak büyük kolaylık sağlamaktadır. Burada personel tablosu için p, meslekler tablosu için m kullanılmıştır. En çok kullanılan yöntem ilk tabloya a harfinden başlamak, diğerlerine de b,c,... gibi alias isimler vermektir.

NOT : where personel.meslekkodu=meslekler.meslekkodu Bu ifade kodun karşılığını bulmak açısından en önemli olan ifadedir. kullanılmaz ise sorgu sonucu yanlış çıkacaktır. Şöyleki; her bir personel meslek tablosundaki tüm mesleklere karşılık gelecek şekilde listelenecektir.



```

SQL> select p.sicilno,p.adi,p.soyadi,m.meslekadi,p.adres
2  from personel p, meslekler m
3  where p.birimkodu=301;

```

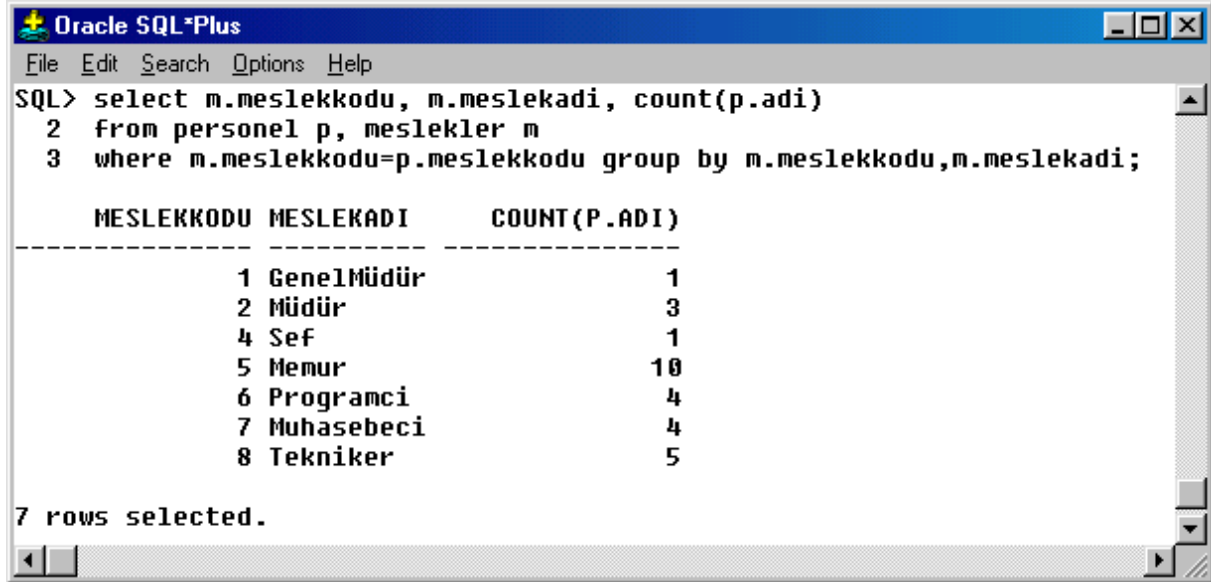
SICILNO	ADI	SOYADI	MESLEKADI	ADRES
1110	Salih	Bayar	Müdür	Ulus-Ankara
178	Fatma	Yalçın	Müdür	Ulus-Ankara
898	Mehmet	Gezgin	Müdür	Ulus-Ankara
1077	Orhan	Sezgin	Müdür	Ulus-Anakara
1121	Nazan	Ulusoy	Müdür	Çankaya-Ankara
960	Özgen	Öney	Müdür	Taksim-istanbul
997	Zehra	Burma	Müdür	Mezitli-Mersin
1110	Salih	Bayar	Sef	Ulus-Ankara
178	Fatma	Yalçın	Sef	Ulus-Ankara
898	Mehmet	Gezgin	Sef	Ulus-Ankara
1077	Orhan	Sezgin	Sef	Ulus-Anakara
1121	Nazan	Ulusoy	Sef	Çankaya-Ankara
960	Özgen	Öney	Sef	Taksim-istanbul
997	Zehra	Burma	Sef	Mezitli-Mersin
1110	Salih	Bayar	Memur	Ulus-Ankara
178	Fatma	Yalçın	Memur	Ulus-Ankara
898	Mehmet	Gezgin	Memur	Ulus-Ankara
1077	Orhan	Sezgin	Memur	Ulus-Anakara
1121	Nazan	Ulusoy	Memur	Çankaya-Ankara
960	Özgen	Öney	Memur	Taksim-istanbul
997	Zehra	Burma	Memur	Mezitli-Mersin
1110	Salih	Bayar	Programci	Ulus-Ankara
178	Fatma	Yalçın	Programci	Ulus-Ankara
898	Mehmet	Gezgin	Programci	Ulus-Ankara
1077	Orhan	Sezgin	Programci	Ulus-Anakara
1121	Nazan	Ulusoy	Programci	Çankaya-Ankara
960	Özgen	Öney	Programci	Taksim-istanbul
997	Zehra	Burma	Programci	Mezitli-Mersin
1110	Salih	Bayar	Muhasebeci	Ulus-Ankara
178	Fatma	Yalçın	Muhasebeci	Ulus-Ankara
898	Mehmet	Gezgin	Muhasebeci	Ulus-Ankara
1077	Orhan	Sezgin	Muhasebeci	Ulus-Anakara
1121	Nazan	Ulusoy	Muhasebeci	Çankaya-Ankara
960	Özgen	Öney	Muhasebeci	Taksim-istanbul
997	Zehra	Burma	Muhasebeci	Mezitli-Mersin

Şekil 11.1.4. Join Sorgu Ekranı-4

Yukarıda where şartında eksik yazılan sorgu ekranının bir kısmı bulunmaktadır. Listelenen kayıt sayısı 56 dır. Bu sayı 7 (meslek sayısı) * 8 (301 de bulunan personel sayısı) çarpımıdır. Bu sorgu çalışan 30 personel ait bir liste olsaydı ve **where personel.meslekkodu=meslekler.meslekkodu** şartı yazılmasaydı liste 7*30 = 210 tane olurdu. Bu nedenle Where koşulunda ilişkili ortak alanların karşılaştırılmasına çok dikkat edilmelidir.

11.2. JOIN İşlemine Ait Örnekler

Örnek-1) Personel tablosundaki her meslek grubunda kaç personel olduğunun listesini veren sorgu.



Oracle SQL*Plus

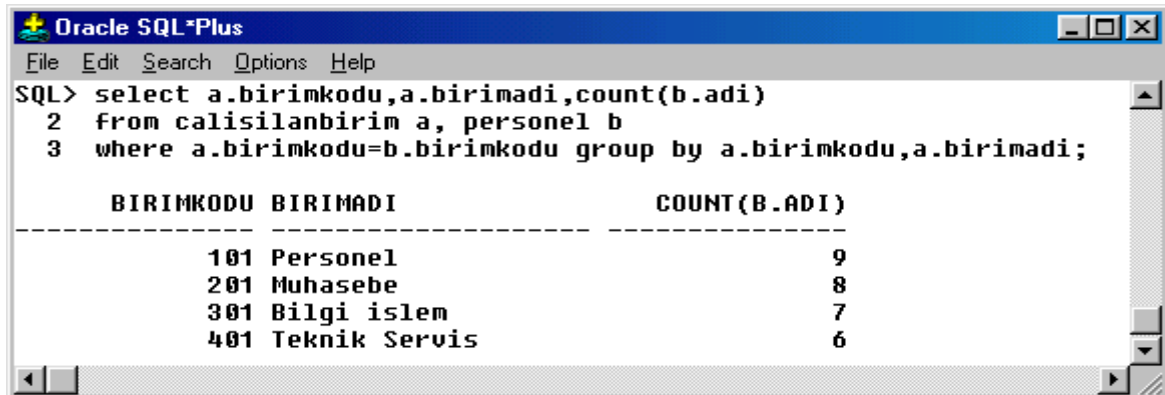
```
File Edit Search Options Help
SQL> select m.meslekkodu, m.meslekadi, count(p.adi)
2   from personel p, meslekler m
3   where m.meslekkodu=p.meslekkodu group by m.meslekkodu,m.meslekadi;
```

MESLEKKODU	MESLEKADI	COUNT(P.ADI)
1	GenelMüdür	1
2	Müdür	3
4	Sef	1
5	Memur	10
6	Programci	4
7	Muhasebeci	4
8	Tekniker	5

7 rows selected.

Şekil 11.2.1. Örnek Join Sorgu Ekranı-1

Örnek-2) Personel tablosundaki her birimde çalışan kaç personel olduğunun listesini veren sorgu.



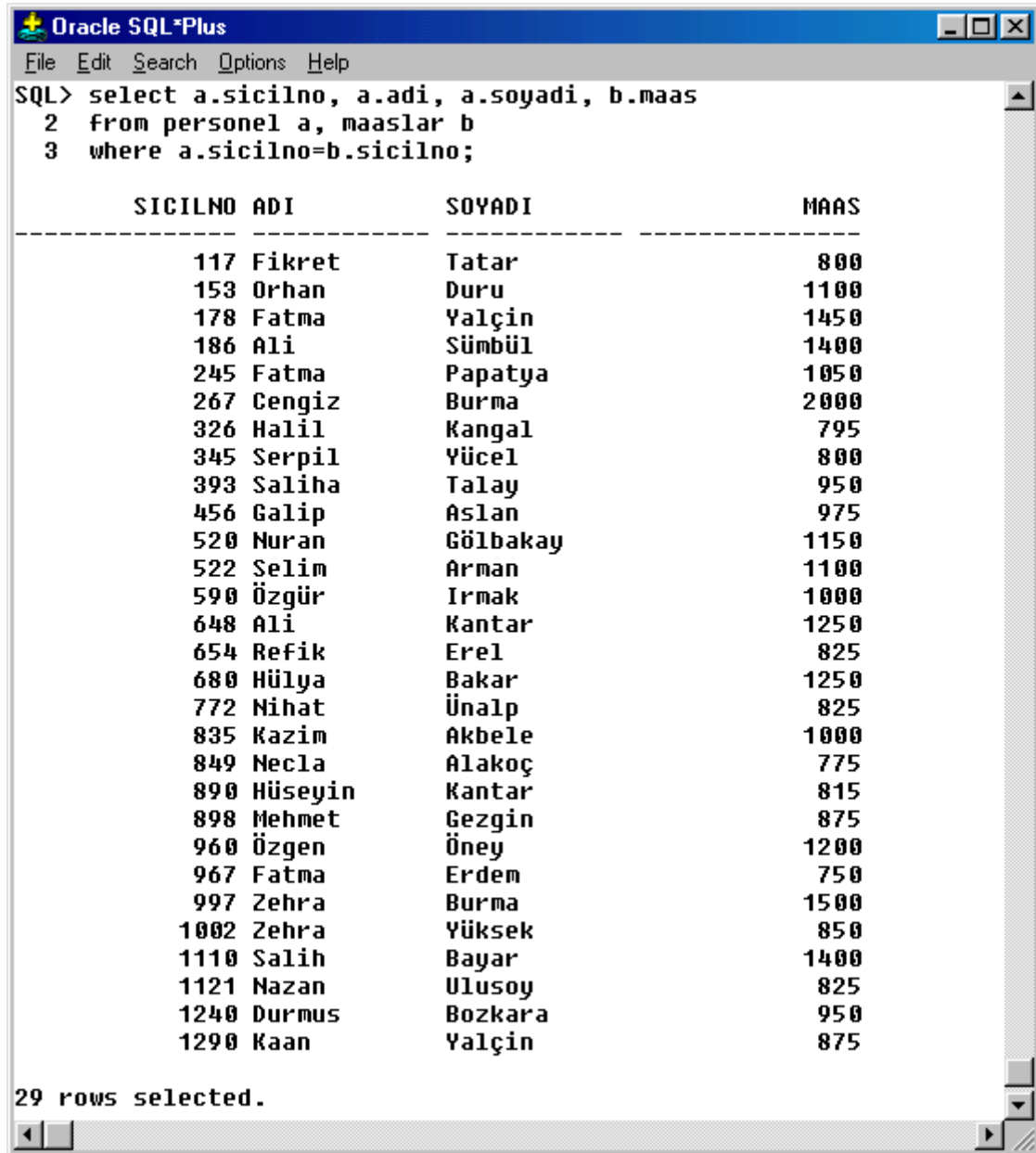
Oracle SQL*Plus

```
File Edit Search Options Help
SQL> select a.birimkodu,a.birimadi,count(b.adi)
2   from calisilanbirim a, personel b
3   where a.birimkodu=b.birimkodu group by a.birimkodu,a.birimadi;
```

BIRIMKODU	BIRIMADI	COUNT(B.ADI)
101	Personel	9
201	Muhasebe	8
301	Bilgi islem	7
401	Teknik Servis	6

Şekil 11.2.2. Örnek Join Sorgu Ekranı-2

Örnek-3) Tüm personelin aldığı maaşların listesini veren sorgu. Bu sorguda 30 personel olmasına rağmen 29 personele ait maaş bilgisi listelenmiştir. Daha önce de bunun özellikle yapıldığı belirtilmişti. Birleştirme işlemleri yapılırken karşımıza şöyle bir problem çıkmaktadır. Birleştirme yapılan tablolardan ikinci tabloda birinci tablodaki her kaydın karşılığı olmazsa, karşılığı olmayan kayıtlar sorgu sonucunda sadece olmayan kayıtlar değil bilakis hiç kayıt gelmez. Bunun için “dış birleştirme” kullanılır. Dış birleştirme işlemi, kayıtları eksik olan tablonun şart tarafına “(+)” işareti konularak yapılır. Örnek 4 de buna ait uygulama yapılmıştır.



Oracle SQL*Plus

File Edit Search Options Help

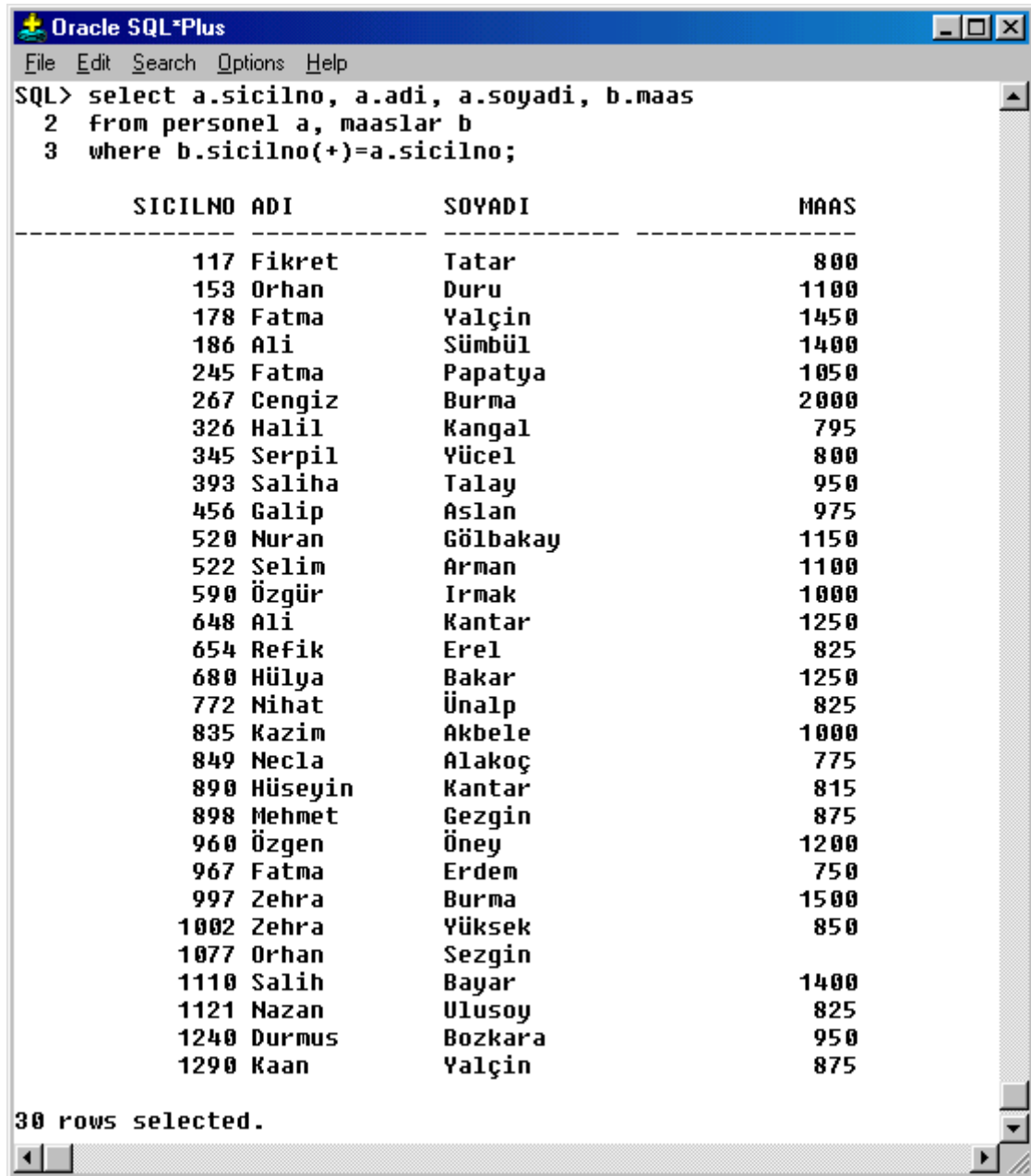
```
SQL> select a.sicilno, a.adi, a.soyadi, b.maas
2 from personel a, maaslar b
3 where a.sicilno=b.sicilno;
```

SICILNO	ADI	SOYADI	MAAS
117	Fikret	Tatar	800
153	Orhan	Duru	1100
178	Fatma	Yalçın	1450
186	Ali	Sümbül	1400
245	Fatma	Papatya	1050
267	Cengiz	Burma	2000
326	Halil	Kangal	795
345	Serpil	Yücel	800
393	Saliha	Talay	950
456	Galip	Aslan	975
520	Nuran	Gölbakay	1150
522	Selim	Arman	1100
590	Özgür	Irmak	1000
648	Ali	Kantar	1250
654	Refik	Erel	825
680	Hülya	Bakar	1250
772	Nihat	Ünalp	825
835	Kazim	Akbele	1000
849	Necia	Alakoç	775
890	Hüseyin	Kantar	815
898	Mehmet	Gezgin	875
960	Özgen	Öney	1200
967	Fatma	Erdem	750
997	Zehra	Burma	1500
1002	Zehra	Yüksek	850
1110	Salih	Bayar	1400
1121	Nazan	Ulusoy	825
1240	Durmus	Bozkara	950
1290	Kaan	Yalçın	875

29 rows selected.

Şekil 11.2.3. Örnek Join Sorgu Ekranı-3

Örnek-4) Tüm personelin aldığı maaşların listesini veren sorgu. Maaşlar tablosunda 1077 sicil numaralı Orhan Sezgin isimli personele maaş bilgisi yazılması unutulmuştur. Buradan SQL'in aynı zamanda bir kontrol mekanizması olarak da kullanılabileceğini çıkarabiliriz.



Oracle SQL*Plus

File Edit Search Options Help

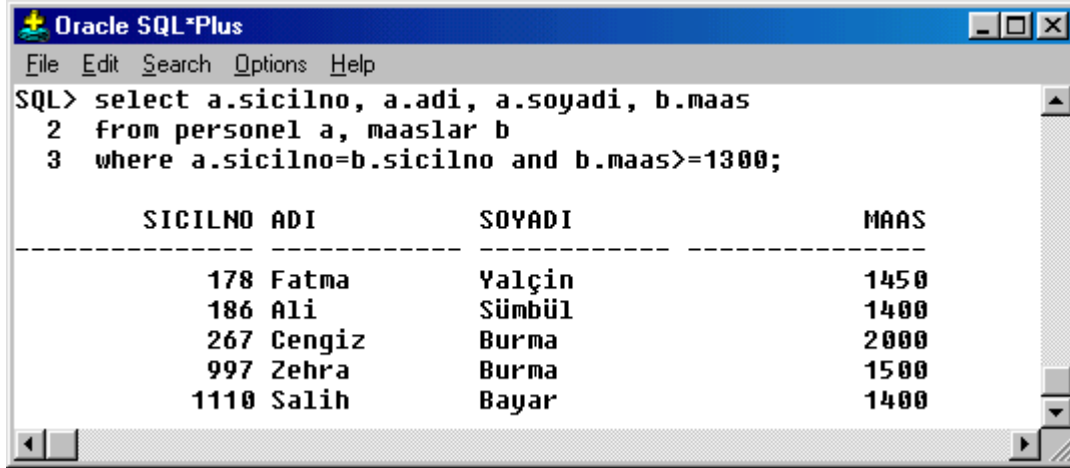
SQL> select a.sicilno, a.adi, a.soyadi, b.maas
2 from personel a, maaslar b
3 where b.sicilno(+)=a.sicilno;

SICILNO	ADI	SOYADI	MAAS
117	Fikret	Tatar	800
153	Orhan	Duru	1100
178	Fatma	Yalçın	1450
186	Ali	Sümbül	1400
245	Fatma	Papatya	1050
267	Cengiz	Burma	2000
326	Halil	Kangal	795
345	Serpil	Yücel	800
393	Saliha	Talay	950
456	Galip	Aslan	975
520	Nuran	Gölbakay	1150
522	Selim	Arman	1100
590	Özgür	Irmak	1000
648	Ali	Kantar	1250
654	Refik	Erel	825
680	Hülya	Bakar	1250
772	Nihat	Ünalp	825
835	Kazım	Akbele	1000
849	Necia	Alakoç	775
890	Hüseyin	Kantar	815
898	Mehmet	Gezgin	875
960	Özgen	Öney	1200
967	Fatma	Erdem	750
997	Zehra	Burma	1500
1002	Zehra	Yüksek	850
1077	Orhan	Sezgin	
1110	Salih	Bayar	1400
1121	Nazan	Ulusoy	825
1240	Durmus	Bozkara	950
1290	Kaan	Yalçın	875

30 rows selected.

Şekil 11.2.4. Örnek Join Sorgu Ekranı-4

Örnek-5) Aylık ücreti 1300 den büyük olan personelin listesini veren sorgu.



The screenshot shows the Oracle SQL*Plus interface. The command window contains the following SQL query:

```
SQL> select a.sicilno, a.adi, a.soyadi, b.maas
2   from personel a, maaslar b
3   where a.sicilno=b.sicilno and b.maas>=1300;
```

The results are displayed in a table with the following columns: SICILNO, ADI, SOYADI, and MAAS.

SICILNO	ADI	SOYADI	MAAS
178	Fatma	Yalçın	1450
186	Ali	Sümbül	1400
267	Cengiz	Burma	2000
997	Zehra	Burma	1500
1110	Salih	Bayar	1400

Şekil 11.2.5. Örnek Join Sorgu Ekranı-5

NOT : Bu bölüm sonuna kadar temel SQL komutları anlatılmıştır. SQL komutları çok kullanışlı ve yararlıdır. Örneğin: personel bilgileri içerisinde maaşı 1500 den büyük olanların listesi ve sayısı alınırken SQL dili değil de klasik programlama dili kullanılmış olsaydı böyle bir liste hangi programlama mantığı ile elde edilebilirdi. Sonuca ulaşabilmek için birçok satırdan meydana gelen kod yazılması gerekirdi. Programlama dilleri ile elde edilecek sonuçta izlenebilecek yöntemlerden biri şu olabilirdi : Bilgilerin yazıldığı dosyanın ilk kaydına gidilir, sonra dosyanın sonuna gelinene kadar her bir personel maaşının 1500’den büyük olup olmadığı şartı sorulur ve bu şarta uyan elemanlar listelenir. Kaç kişi olduğunun bilgisini bulmak içinde bu bilgi bir değişkende tutularak değişkenin değeri 1’er artırılarak şarta uyan kaç eleman olduğu bilgisine ulaşılır.

Aynı şekilde; her bölümde çalışan personel sayısı nedir veya kaç farklı bölüm vardır? SQL sorgusu için düşünülecek olursa bu tip sonuçların klasik programlama mantığı ile yapılabilmesinin ne kadar zor ve vakit alıcı olduğunu düşünün.

COUNT, AVG, SUM, DISTINCT, GROUP BY ve HAVING ile elde edilen sorgu sonuçlarının klasik programlama mantığı ile nasıl yapılabileceğini düşünün.

Bu örneklerden sonra SQL ile işlem yapmanın ne kadar kolay olduğu, SQL’in ne kadar yetenekli bir dil olduğu ve SQL’in önemi anlaşılabilir. Bu nedenle V.Basic, Delphi, Asp, PHP gibi programlama dillerinde SQL desteğini kullanarak işlemlerinizi kolaylaştırınız.

12. SQL 'de YÖNETİMSEL FONKSİYONLAR

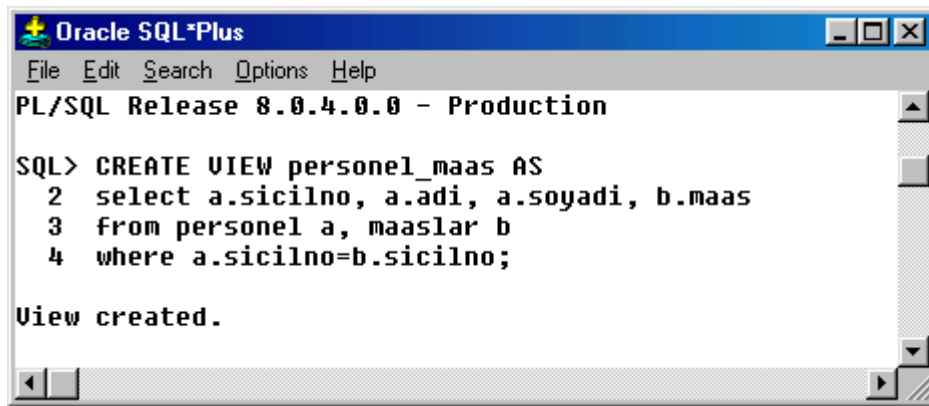
12.1. VIEWS (Tablo Görünümü)

Tablo görünümüleri veri tabanında tanımı olan tablolardan sorgulama sonucunda elde edilir. Tabloların tersine fiziksel bir yer tutmazlar. Uygulamalarda pek çok pratiklik sağlarlar.

Görüntü oluşturabilmek için Select cümlesi kullanılması gerekir. Bir görüntü bir yada daha fazla tablodan oluşturulabileceği gibi bir başka görüntüden (view) de oluşturulabilir.

Örnek-1) **SQL> CREATE VIEW personel_maas AS**
2 **select a.sicilno, a.adi, a.soyadi, b.maas**
3 **from personel a, maaslar b**
4 **where a.sicilno=b.sicilno;**

Personele ait sicilno, ad ve soyad bilgisi Personel tablosunda, maaş bilgisi ise maaslar 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.

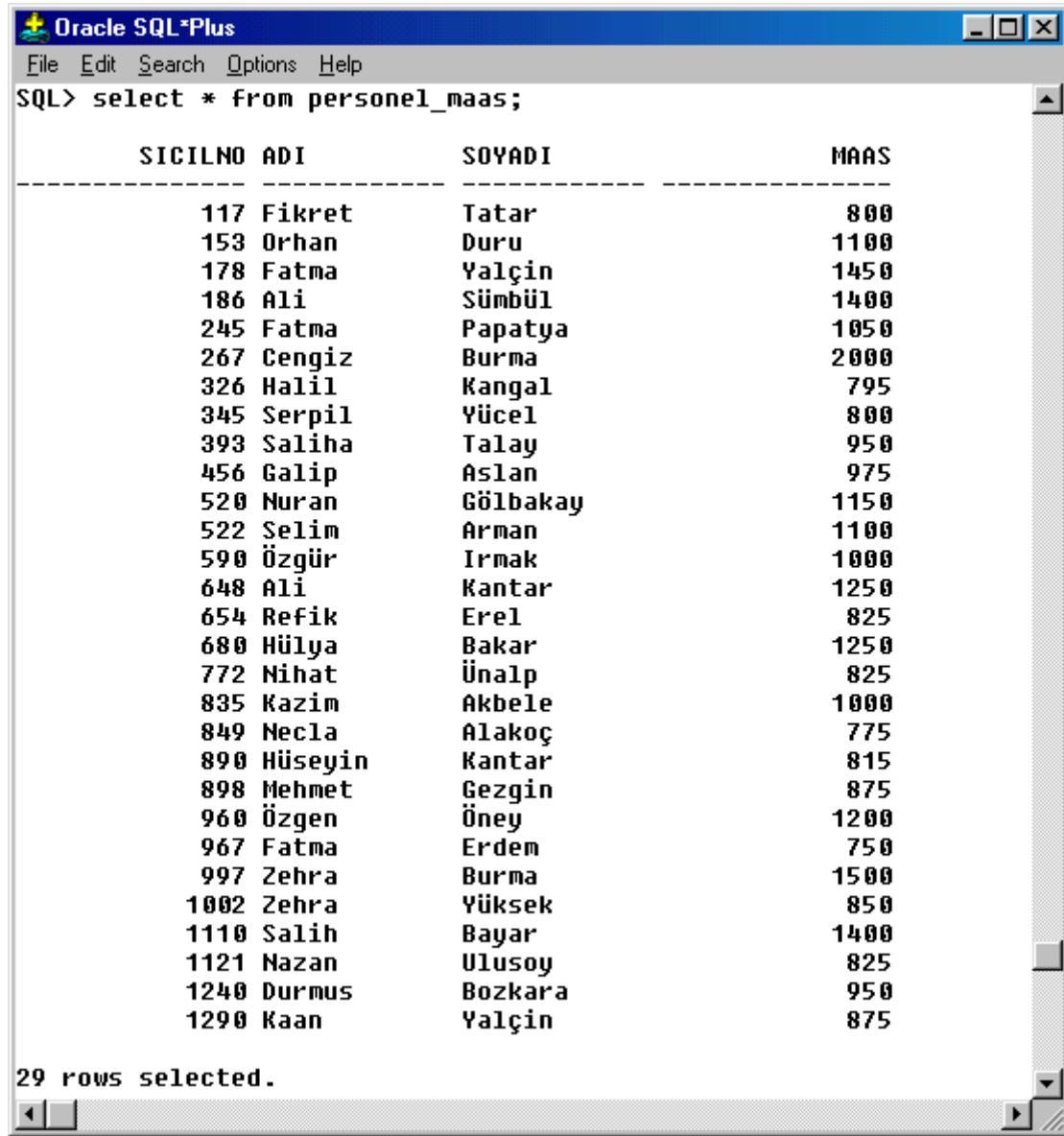


Şekil 12.1.1. View Ekranı-1

“**Personel_maas**” oluşturulan view ismidir. As deyiminden, sonra view alanlarının hangi bilgilerden oluşacağı yazılır. Bu ise select komutu ile öğrenilen klasik select sorgusudur. Tablo görünümüleri, tablolarda yapılan değişikliklerden etkilenmezler. Tabloya yeni bir kayıttın eklenmesi, kayıtların silinmesi veya değiştirilmesi, tabloya view'de olmayan yeni bir kolonun eklenmesi veya silinmesi gibi bir değişiklik view'lerin (görünümlerin) yapısını değiştirmez.

Örnek-2) **SQL> SELECT * FROM personel_maas;**

Personel ve personele ait maaş bilgisi personel_maas görünümünden listelenmiştir.



SICILNO	ADI	SOYADI	MAAS
117	Fikret	Tatar	800
153	Orhan	Duru	1100
178	Fatma	Yalçın	1450
186	Ali	Sümbül	1400
245	Fatma	Papatya	1050
267	Cengiz	Burma	2000
326	Halil	Kangal	795
345	Serpil	Yücel	800
393	Saliha	Talay	950
456	Galip	Aslan	975
520	Nuran	Gölbakay	1150
522	Selim	Arman	1100
590	Özgür	Irmak	1000
648	Ali	Kantar	1250
654	Refik	Erel	825
680	Hülya	Bakar	1250
772	Nihat	Ünalp	825
835	Kazim	Akbele	1000
849	Necla	Alakoç	775
890	Hüseyin	Kantar	815
898	Mehmet	Gezgin	875
960	Özgen	Öney	1200
967	Fatma	Erdem	750
997	Zehra	Burma	1500
1002	Zehra	Yüksek	850
1110	Salih	Bayar	1400
1121	Nazan	Ulusoy	825
1240	Durmus	Bozkara	950
1290	Kaan	Yalçın	875

Şekil 12.1.2. View Ekranı-2

View ler tıpkı tablolar gibi işlem görürler. Ama fiziksel olarak diskte yer kaplamazlar.

View'lerin faydaları aşağıda sıralanmıştır.

1. Tablolardan değişik kullanıcı gruplarına uygun alanların ayrılması sağlanır. Bu da güvenlik açısından önemlidir.
2. Birkaç tablodan elde edilen sürekli kullanılan karmaşık sorgulardan view oluşturularak sürekli hazır tutulur ve her seferinde bu karmaşık sorgu yazılmak yerine view den seçim yapılır.
3. Karmaşık sorgular için yada tabloda bulunan özel satırlar için tablo görünümüleri oluşturularak uygulamalarda pratiklik sağlanır.

12.2. CREATE TABLESPACE (Tablo Uzaı = Veri Alanı)

Kullanıcılara ait olan nesnelerin veritabanında mantıksal olarak tutulduğu yere tablo uzaı ismi verilir. Bir tablo uzaı oluşturulurken, bu tablo uzaının verilerinin hangi veri dosyasına konulacağı ve bu dosyanın dizini ile büyüklüğü bildirilmelidir.

```
CREATE TABLESPACE tbs_esef  
DATAFILE 'c:\orasql\tbs_esef.dat' SIZE 10M  
DEFAULT STORAGE (INITIAL 10K NEXT 50K  
MINEXTENTS 1 MAXEXTENTS 999)  
ONLINE;
```

Tbs_esef tablespace için verilen isimdir. C:\orasql veri alanının yaratılacağı dizin ve tbs_esef.dat ise diskte görüntülenecek olan veri alanının ismidir.

SIZE bilgisi veri dosyasının diskte fiziksel olarak kaplayacağı yeri belirler. Burada 10M, 5K gibi değerler girilebilir. INITIAL bildirisi tablo uzaı oluşturulduğunda, ilk alacağı genişleme'nin büyüklüğünü belirler. Next tablo uzaı oluşturulduktan sonra alacağı genişlemelerin büyüklüğünü belirler. MINEXTENTS tablo uzaı oluşturulduğunda ilk olarak alacağı minimum genişleme sayısının belirtildiği bölümdür. MAXEXTENTS bir tablo uzaının ilk olarak aldığı genişleme de dahil olmak üzere alabileceği maksimum genişleme sayısının belirtildiği bölümdür.

12.3. CREATE USER (Kullanıcı)

Kullanıcı veritabanı nesnelerinin sahibidir. Kullanıcılar, nesneleri oluşturur, kullanır ve silerler. Oracle veritabanı ilk kurulduğunda standart olarak üç kullanıcı tanımlanır. Bunlardan bir SYS kullanıcısıdır. SYS kullanıcısı veri sözlüğünün sahibi olan kullanıcıdır. Tüm nesneleri oluşturma hakkına sahiptir ve diğer bütün kullanıcıların nesnelerine erişebilir. SYS kullanıcısının ilk şifresi "change_on_install" olarak belirlenmiştir. İkinci kullanıcı SYSTEM kullanıcısıdır. SYSTEM kullanıcısı veri sözlüğünü kullanma hakkına sahiptir. Önemli nesneleri oluşturma hakkına da sahiptir. İlk şifresi "manager" olarak belirlenmiştir. Diğer kullanıcıların nesnelerine erişme hakkına da sahiptir. Üçüncü kullanıcı SCOTT kullanıcısıdır. SCOTT kullanıcısı veritabanına başlangıçta yüklenen demo tabloların sahibidir. Bu kullanıcının nesneleri kullanılarak SQL denemeleri yapılabilir.

"CREATE USER" komutunu SYS ve SYSTEM kullanıcıları standart olarak kullanabilir. Bu hak diğer kullanıcılara da verilebilir. Her kullanıcının nesnelerini tutmak için bir tablo uzaı oluşturmak sistemin performansı açısından gereklidir. Kullanıcı oluşturulurken bu tablo uzaı o kullanıcıya atanır.

```
CREATE USER ogrenci  
IDENTIFIED BY ogrenci1  
DEFAULT TABLESPACE tbs_esef  
QUOTA UNLIMITED ON tbs_esef
```

Yukarıdaki örnekte öğrenci adında bir kullanıcı oluşturulmuştur. Kullanıcının şifresi IDENTIFIED BY ile “ogrenci1” olarak belirtilmiştir. Kullanıcının kendi nesnelerini oluşturacağı tablo uzayı için ise “tbs_esef” tablo uzayı bildirilmiştir. Kullanıcının bu tablo uzayındaki tüm alanı kullanabileceği QUOTA UNLIMITED ile belirlenmiştir.

12.4. CREATE ROLE

Rol veritabanındaki hakların toplanmış haline denir. Veritabanı yöneticisi rolleri kullanarak sistemin güvenliğini daha kolay sağlayabilir. Roller Oracle tarafından önceden tanımlanmış roller ve kullanıcı tanımlı roller olarak iki şekilde düşünülebilir. Oracle tarafından önceden tanımlanan roller beş tanedir:

Rol	Atanmış Haklar
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	CREATE CLUSTER, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER
DBA	“WITH ADMIN OPTION” ile birlikte bütün sistem hakları
EXP_FULL_DATABASE	SELECT ANY TABLE, BACKUP ANY TABLE, SYS.INCVID, SYS.INCFIL ve SYS.INCEXP tablolarına INSERT, UPDATE ve DELETE hakkı
IMP_FULL_DATABASE	BECOME USER, WRITEDOWN

```
CREATE ROLE tablo_incele  
GRANT SELECT ON personel TO tablo_incele
```

Yukarıdaki örnekte tablo_incele isimli bir rol oluşturulmuştur. Daha sonra bu role personel tablosu üzerinde listeleme işlemi yapma hakkı verilmiştir. Böylece bu rolün atandığı kullanıcı personel tablosu üzerinde “SELECT” komutunu çalıştırabilir. Bunun dışındaki insert, update, delete vb. komutları çalıştıramaz.

12.5. CREATE INDEX

Daha öncede bahsedildiği gibi indeks tablodaki kayıtlara daha hızlı erişim için kullanılan nesnelerdir. Bir indeks oluşturabilmek için “CREATE ANY INDEX” sistem hakkına sahip olmak gerekir. İndeks bir tablonun bir alanı üzerinde tanımlanabileceği gibi birden fazla alan üzerinde de tanımlanabilir.

```
CREATE INDEX indeks1 ON personel(sicilno)
```

Personel tablosu üzerinde sicilno alanı için index1 isimli bir index yaratılmıştır.

12.6. CREATE SEQUENCE

Sıra, sıralı olarak artan alanlar için veritabanında tutulan nesnedir. Sıra tekrarlamayan sayısal bilgi elde etmek için kullanılır. Örneğin sicil numaraları, öğrenci numaraları, abone numaraları, müşteri numaraları gibi. Örneğin birden başlayan ve birer birer artan bir sıra yaratmak için:

```
CREATE SEQUENCE sicilno START WITH 1 INCREMENT BY 1
```

12.7. GRANT

Sistem ya da nesne haklarının kullanıcılara veya rollere atanması için kullanılan komuttur.

```
GRANT DELETE ON personel TO ogrenci
```

Yukarıdaki örnekte ogrenci kullanıcıasına personel tablosunda silme yapma yetkisi verilir.

12.8. REVOKE

Sistem ya da nesne haklarının kullanıcılardan veya rollerden geri alınması için kullanılan komuttur.

```
REVOKE DELETE ON personel FROM ogrenci
```

Yukarıdaki örnekte ogrenci kullanıcıısından personel tablosunda silme yapma yetkisi geri alınır.

KAYNAKLAR

- Yaşar Gözüdereli, Veritabanı Programlamaya Giriş, Byte Dergisi Eğitim Dizisi, Temmuz 2003
- Yaşar Gözüdereli, Veritabanı Programlama II, Byte Dergisi Eğitim Dizisi, Şubat 2004
- Osman Nihat Şen, Oracle, SQL, SQL*Plus, PL/SQL ve Veritabanı Yönetimi, İkinci Baskı, Beta Yayınları, İstanbul, 2000
- Aydın Şekihanov, Oracle8i A Practical Guide To SQL, PL/SQL, Developer 6, Atılım University Publications, 2001
- Oracle 8i Enterprise Edition Documentation
- Bilkent Üniversitesi Bilgisayar Merkezi, Veri Tabanı Yönetim Sistemine Giriş ve SQL, Temmuz 1996
- [http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri\(csutcu%20dr%20tezinin%20bir%20bolumu\).pdf](http://iletisim.marmara.edu.tr/bilisim/veri%20modelleri(csutcu%20dr%20tezinin%20bir%20bolumu).pdf)
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2010>
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2011>
- <http://www.yazilimgrubu.com/dokuman.php?sayfa=2&no=%2012>
- <http://www.farukcubukcu.com/downloads/sqlkomutlari.doc>
- <http://www.farukcubukcu.com/downloads/sqlkomutlari.doc>
- http://www.bto.yildiz.edu.tr/dosyalar/ebt2_konu5.doc
- http://www.bto.yildiz.edu.tr/dosyalar/ebt2_konu4.doc
- <http://www.verivizyon.com>
- <http://www.verivizyon.com/detail.asp?cid=110>
- <http://www.verivizyon.com/detail.asp?cid=96>
- <http://cisen.odtu.edu.tr/2002-7/veritabani.php>
- <http://serdar.ktg.com.tr/oracle.htm>
- http://www.geocities.com/cakmak_cengiz/
- http://www.olgun.com/main_sql_destek.asp
- www.oracle.com