T.C.
MİLLÎ EĞİTİM BAKANLIĞI

ENDÜSTRİYEL OTOMASYON TEKNOLOJİLERİ

VERİ TABANI
481BB0085

Ankara, 2011

- Bu modül, mesleki ve teknik eğitim okul/kurumlarında uygulanan Çerçeve Öğretim Programlarında yer alan yeterlikleri kazandırmaya yönelik olarak öğrencilere rehberlik etmek amacıyla hazırlanmış bireysel öğrenme materyalidir.
- Millî Eğitim Bakanlığınca ücretsiz olarak verilmiştir.
- PARA İLE SATILMAZ.

İÇİNDEKİLER

AÇIKLAMALAR iii
GİRİŞ 1
ÖĞRENME FAALİYETİ–1 3
1. VERİ TABANININ TANIMI 3
1.1. Veri Tabanı 3
1.2. Veri Tabanı Çeşitleri 3
1.2.1. İlişkisel Veri Tabanı 3
1.3. PostgreSQL Veri Tabanı 4
1.3.1. PostgreSQL'in Kurulumu 5
1.3.2. PostgreSQL Sunucunun Çalışma Sistemi 6
1.3.3. PostgreSQL' in Başlatılması 7
1.3.4. Veri Tabanı Dosyası Oluşturma 7
1.3.5. Veri Tabanı Dosyası Silme 9
1.3.6. PostgreSQL Kullanıcısı Oluşturma 9
1.3.7. PostgreSQL Kullanıcısı Silme 10
1.3.8. Terminal Tabanlı PostgreSQL Sorgu Arayüzü 10
1.3.9. PostgreSQL Veri Tabanının Yedeklenmesi 15
1.3.10. PostgreSQL Tüm Veri Tabanlarının Yedeklenmesi 18
1.3.11. PostgreSQL Veri Tabanının Geri Yüklenmesi 18
UYGULAMA FAALİYETİ 19
ÖLÇME VE DEĞERLENDİRME 21
ÖĞRENME FAALİYETİ–2 23
2. YAPISAL SORGULAMA DİLİ 23
2.1. Yapısal Sorgulama Dili SQL'un Tanımı 23
2.1.1. DDL Veri Tanımlama Komutları 24
2.1.2. DML Veri İşleme Komutları 31
UYGULAMA FAALİYETİ 37
ÖLÇME VE DEĞERLENDİRME 40
ÖĞRENME FAALİYETİ–3 42
3. VERİ TABANI SORGULAMA 42
3.1. SELECT Komutuyla Tek Tablo Üzerinde Sorgulama Yapma 42
3.1.1. Koşulsuz Sorgulama 42
3.1.2. Koşullu Sorgulama 46
3.1.3. SELECT Komutunda Kullanılabilecek Aritmetiksel İfadeler 50
3.1.4. Gruplandırma İşlemi (GROUP BY) 53
3.2. SELECT Komutu ile Birden Fazla Tabloyu Sorgulama 54
3.2.1. UNION (Birleşim) İfadesi 57
3.2.2. EXCEPT (Fark) İfadesi 59
3.2.3. INTERSECT (Kesişim) İfadesi 60
3.2.4. İç İçe (Nested) SELECT İfadesi 60
3.2.5. ANY İfadesi 62
3.2.6. ALL İfadesi 64
3.2.7. EXISTS İfadesi 64
UYGULAMA FAALİYETİ 66
ÖLÇME VE DEĞERLENDİRME 68
MODÜL DEĞERLENDİRME 70

CEVAP ANAHTARLARI 71
KAYNAKÇA 72

AÇIKLAMALAR

KOD	**481BB0085**
ALAN	**Endüstriyel Otomasyon Teknolojileri**
DAL/MESLEK	**Alan Ortak**
MODÜLÜN ADI	**Veri Tabanı**
MODÜLÜN TANIMI	Veri tabanın temellerini anlatan öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	
YETERLİK	Veri tabanı sunucusunu hatasız bir şekilde kurmak
MODÜLÜN AMACI	**Genel Amaç** Bu modül ile gerekli ortam sağlandığında veri tabanı uygulamalarını hatasız yapabileceksiniz. **Amaçlar** **1.** Veri tabanı yönetim sisteminin kurulumunu ve kullanımını öğreneceksiniz. **2.** Yapısal sorgulama dili komutlarını kullanabileceksiniz. **3.** Veri tabanındaki tabloları sorgulayabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam:** Bilgisayar laboratuvarı **Donanım:** Bilgisayar, işletim sistemi
ÖLÇME VE DEĞERLENDİRME	Modül içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi değerlendireceksiniz. Öğretmen modül sonunda ölçme aracı (çoktan seçmeli test, doğru-yanlış vb.) kullanarak modül uygulamaları ile kazandığınız bilgi ve becerileri ölçerek sizi değerlendirecektir.

GİRİŞ

Sevgili Öğrenci,

İnsanoğlu, verileri yıllardır belli algoritmalar altında saklamaktadır. Devasa boyutlarda verilerin bir arada tutulması, istendiğinde bu verilere ulaşılması sorunları gün geçtikçe daha ciddi çözümlerin getirilmesi gerekliliğini ortaya koymuştur. Bu amaçla çeşitli veri tabanı yönetim sistemi türleri ortaya çıkmış ve bu tip programlar geliştirilmiş ve hatta geliştirilmeye devam edilmektedir.

Biz bu modülde hızla gelişmekte olan, büyük çapta veri depolama ve sorunsuzca sorgulama yapabileceğimiz veri tabanı yönetim sistemi olan PostgreSQL veri tabanını, yönetim sisteminin kurulumunu, veri tabanı oluşturulması ve bu veri tabanı içinde sorgulamalar yapmayı anlatmaya çalışacağız.

Modülde siz öğrencilere bu konularda çok da detaylı olmasa da bilgiler verilecektir. Modül bittiğinde bir veri tabanı yönetim sistemi olan PostgreSQL programını kullanmayı öğreneceksiniz. Ayrıca ortak bir platform olan veri tabanı sorgulama dili olan SQL ile veri tabanı içindeki tabloları sorgulayabileceksiniz.

ÖĞRENME FAALİYETİ–1

AMAÇ

Modül bittiğinde veri tabanı yönetim sisteminin kurulumunu ve kullanımını öğreneceksiniz.

ARAŞTIRMA

- Veri tabanı çeşitleri ve yapıları hakkında internetten doküman bulunuz.

1. VERİ TABANININ TANIMI

1.1. Veri Tabanı

Veri tabanı genel anlamıyla verilerin toplandığı ve bu verilerin çeşitli amaçlar için kullanıldığı yapılardır. Veri tabanı içinde veriler tekrarlanmaz. Günümüzde çeşitli alanda veri tabanları kullanılır. Örneğin bankalarda, büyük alışveriş mağazalarında, kütüphanelerde daha burada sayamadığımız birçok alanda veri tabanı kullanılmaktadır.

Veri tabanlarına ihtiyaca göre veri eklenebilir veya silinebilir. Gerektiğinde veri tabanındaki bilgiler sorgulanabilir ve kullanıcıya aktarılabilir.

1.2. Veri Tabanı Çeşitleri

Yapılarına göre üç çeşit veri tabanı bulunmaktadır.

- Hiyerarşik veri tabanı
- İlişkisel veri tabanı
- Nesnel veri tabanı

Modülde kullanılan veri tabanı ilişkisel veri tabanı olduğu için sadece ilişkisel veri tabanı anlatılacaktır.

1.2.1. İlişkisel Veri Tabanı

İlişkisel tip veri tabanı tablolardan oluşmaktadır. Tablolar veri tabanı içinde ortak özelliklere sahip olan verilerin depolandığı yapılardır. Tabloların sayısı bir veya birden fazla olabilir. Tablolar satır ve sütunlardan oluşur. Aynı tablo içinde veri tekrarı olmaz. Başka bir deyişle veri tekrarı yoktur.

Aşağıda örnek bir tablo verilmiştir. Bu tabloda okulda bulunan öğrencilerin numarasını, adı ve soyadlarını içersin.

1. Tablo : ogrenci_bilgi

	Sütun 1	Sütun 2	Sütun 3
	Numara	**Ad**	**Soyad**
Satır 1	11	Ümit	Cihan
Satır 2	12	Ahmet	Zapir
Satır 3	13	Okan	Su
Satır 4	14	Gökhan	Demir

Yukarıdaki tablo incelendiğinde sütunlar alanları, satırlar ise verileri ifade eder. Tabloda toplam 3 alan ve 4 verimiz bulunmaktadır. Ancak bu şekilde kullanıldığında bir ilişkisellikten bahsedilemez. Bir başka tablo oluşturarak bu iki tablonun nasıl ilişkilendirildiği takip edilsin.

2. Tablo : ogrenci_not

	Sütun 1	Sütun 2	Sütun 3	Sütun 4
	Numara	**Not1**	**Not2**	**Not3**
Satır 1	11	80	90	100
Satır 2	12	90	90	80
Satır 3	13	70	70	50
Satır 4	14	60	60	90

Görüldüğü üzere ikinci tabloda yine aynı öğrencilerin (bunları numaralarından anlaşılabilir.) bu defa notları tutulur.

Yukarıdaki örnekte olduğu gibi belli alanlar ile birbiri ile ilişkilendirilebilen (örneğimizde ilişki kurulan alan numara alanıdır) tablolardan oluşan veri tabanı çeşidine ilişkisel tip (Relational Type) veri tabanı denir.

Bu modülde ilişkisel tip veri tabanı olan PostgreSQL veri tabanı kullanılacaktır.

1.3. PostgreSQL Veri Tabanı

PostgreSQL ilişkisel modeli kullanan ve SQL standart sorgulama dilini kullanan bir VTYS (veri tabanı yönetim sitemi) dir.

VTYS : *Veri tabanına birden fazla kullanıcın ulaşmasını ve bu kullanıcıların isteklerine cevap verilmesini sağlayan programdır.*

PostgreSQL California Berkley Üniversitesi'nde yapılan çalışmalar sonrasında geliştirildi. İlk olarak 1977-1985 arasında "Ingres" adında ilişkisel veri tabanı olarak geliştirildi. Daha sonra Ingres'in kodu Relational Tech/ Ingres firması tarafından satın alındı.

Berkley'de bu veri tabanı üzerinde çalışmalar devam etti ve bu veri tabanı postgres adını aldı. Bu kod da başka bir firma tarafından satın alınarak "informix" adını aldı.

1994' te postgres veri tabanına SQL standardı eklendi. Bu veri tabanına da Postgres95 adı verildi. 1995 yılından itibaren "Postgres" tanınmaya ve kodları yayılmaya başladı. Gönüllü birçok programcı tarafından geliştirilmeye devam etti. Bu aşamadan sonra artık PostgreSQL olarak adlandırıldı ve günümüze kadar bu isimle geldi.

PostgreSQL'in birçok üstünlükleri ve avantajları vardır.

PostgreSQL aynı zamanda iyi performans veren, güvenli ve geniş özellikleri olan, kararlı bir veri tabanı yönetim sistemidir. Hemen hemen tüm Unıx ya da Unix türevi (Linux, FreeBSD gibi) işletim sistemlerinde çalışır. Ayrıca NT çekirdekli tüm Windows sistemlerde de çalıştırılabilir. PostgreSQL ücretsiz ve açık kodludur.

PostgreSQL birçok programlama dili tarafından desteklenmektedir. Standart olarak destekleyen diller şunlardır: C, C++, Java, Perl, Tcl/Tk, Python, PHP, ve Ruby, vs.

PostgreSQL SQL işlevlerine sahip bir veri tabanı programıdır. SQL çok karmaşık sorgulamaların bile yapılabileceği bir dildir. Böylelikle veri tabanı üzerinde veri değişiklikleri yapılabilir. Başka bir değişle bütün bu işlemler bir ekranda toplanabilir.

1.3.1. PostgreSQL'in Kurulumu

Bir postgreSQL veri tabanı iki farklı metotla kurulabilir. Bunlardan birincisi kaynak koddan derleyerek kurma, ikincisi ise RPM kurulum paketi ile kurmadır. Bu modüldeki uygulamada RPM kurulum paketinden kurulacaktır. Kurulum öncesi aşağıda adı geçen dosyaların tedarik edilmesi gerekir. Bu RPM dosyaları internetten temin edilebileceği gibi uygulamalarda kullanılan Fedora Core CD' si içindeki "/Fedora/RPMS" klasörü içinde de bulunabilir.

Kurulumu yapılabilen paketler ve ne işe yaradıkları aşağıdaki tabloda verilmiştir.

Paket ismi	Açıklaması
postgresql	Kurulum yapılacak asıl ana paket
postgresql-libs	PostgreSQL kütüphane dosyaları
postgresql-devel	PostgreSQL geliştirme aracı
Postgresql-odbc	PostgreSQL için ODBC (Windows istemciden bağlantı için)
Postgresql-jdbc	postgreSQL için java bağlantısı
Postgresql-perl	Perl için PostgreSQL
Postgresql-phyton	Phyton için PostgreSQL
Postgresql-server	PostgreSQL sunucu oluşturmak için gerekli programlar
Postgresql-tcl	Tcl için postgreSQL
Postgresql-test	PostgreSQL test birimi
Postgresql-tk	postgreSQL için tk kabuğu ve grafik ara yüzü
Postgresql-contrib	postgrSQL ile birlikte gelen kaynak.

Tablo 1.1. PostgreSQL kurulum RPM paketleri

Bu kurulumlardan yalnızca üç tanesini yapmak yeterli olacaktır.

"postgresql", "postgresql-libs" ve "postgresql-server" Rpm komutu ile dosyaları kurulacaktır.

İlk olarak "postgresq-libs" dosyasını kurmak için aşağıdaki komutu veriniz.

Not: *Bu komutları kurulum dosyalarının bulunduğu klasör içindeyken yaptığımız varsayılmıştır. Ayrıca rpm dosyalarının bir kısmını oluşturan sürüm bilgileri kurulumunuzda kullanacağınız rpm sürümüne göre mutlaka farklı olacaktır. Bu kurulumda postgreSQL in 8.1.4 versiyonu kullanılmıştır.*

1. Adım:

#rpm -ihv postgresql-libs-8.1.4-1.i386.rpm

2. Adım:

 # rpm -ihv postgresql-8.1.4-1.i386.rpm

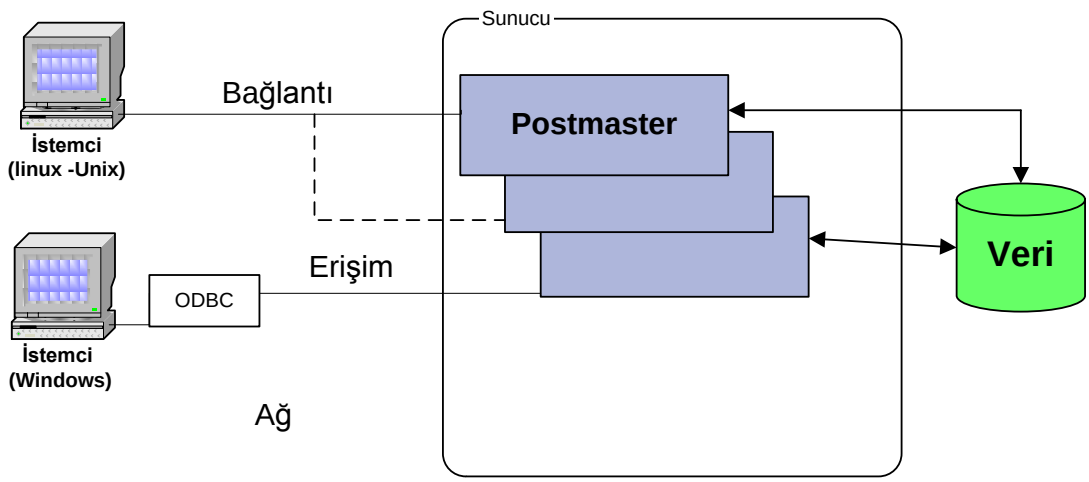
3. Adım:

#rpm -ihv postgresql-server-8.1.4-1.i386.rpm

1.3.2. PostgreSQL Sunucunun Çalışma Sistemi

Postmaster postgreSQL veri tabanının sunucusudur. Postmaster tek bir sunucu üzerinde çalışabilir. Veri tabanına erişebilecek programlar mutlaka sunucu üzerinde çalışır. Sunucu içinden bir program dahi veri tabanına direkt olarak ulaşamaz. Bu yapı istemci ile sunucun farklı bilgisayarlarda olmasına imkân tanır.

Yukarıda anlatılan yapının şeması şekildeki gibidir:



Şekil 1.1. PostgreSQL sunucusunun çalışması

1.3.3. PostgreSQL' in Başlatılması

Kurulum esnasında herhangi bir hata yapılmamış ise bilgisayarda "postgres" adında bir kullanıcı oluşturulacaktır. İstenirse bu kullanıcıya aşağıdaki komut ile bir şifre verilebilir. Bu güvenlik için gerekli bir basamaktır.

```
#passwd postgres
```

PostreSQL veri tabanı bu kullanıcı ile yönetilecektir bu nedenle "root" kullanıcısından "postgres" kullanıcısına aşağıdaki komut ile geçiş yapılacaktır.

```
#su  postgres (ENTER)
bash-3.1$
```

Şimdi de postgreSQL sunucusu başlatılsın. Bu da istenirse grafik ara yüzü kullanılarak istenirse de aşağıdaki komutu uygulanarak yapılabilir.

```
#service postgresql start
```

Sunucu başlatılan anda "initialize" işlemi otomatik olarak yapılacaktır. Bunun anlamı sunucu otomatik olarak veri tabanının oluşturulacağı klasörü hazırlayacaktır.

Bu klasör "/var/lib/psql" klasörüdür. "initialize" işleminde bu klasör oluşturulacak ve içine bazı dosyalar eklenecektir.

1.3.4. Veri Tabanı Dosyası Oluşturma

Postgresql'de veri tabanı oluşturma komutu "createdb" komutudur. "createdb" komutunun kullanımı aşağıdaki gibidir.

```
createdb [parametre] [veri tabanı adı]
```

Parametre kısmında kullanılabilecek parametreler ve açıklamaları aşağıdaki gibidir.

Parametre	Açıklaması
-E *encoding* veya **--encoding *encoding***	➢ Veri tabanının oluşturulacağı karakter seti (örneğin Türkçe için LATIN5)
-O sahip veya **--owner sahip**	➢ Veri tabanın kime ait olacağı belirtilir.
-h host veya **--host host**	➢ Yeni oluşturulacak veri tabanın bulunacağı bilgisayarın "hostname" i yazılır.
-P port veya **-port port**	➢ Veri tabanı sunucusunun port numarası belirtilir.

Tablo 1.2: Createdb komutunun parametreleri

Örnek olarak hiçbir parametre vermeden bir veri tabanı oluşturulabilir.

```
bash-3.1$createdb okul
CREATE DATABASE
```

Veri tabanı oluşturulmuş ise komutun hemen altında "CREATE DATABASE" yanıt görülür.

Bir sonraki örnekte ise "Türkçe karakter seti" kullanarak bir veri tabanı oluşturulabilir.

```
bash-3.1$createdb –E LATIN5  okul
CREATE DATABASE
```

1.3.5. Veri Tabanı Dosyası Silme

Veri tabanı sunucusundaki bir veri tabanını silmek için "dropdb" komutu kullanılabilir. Kullanımı aşağıdaki gibidir.

```
dropdb [parametre] [veri tabanı adı]
```

Parametre olarak "createdb" komutundaki parametrelerden "–H" ve "-P" parametrelerini kullanılabilir. Aşağıda örnek olarak "createdb" ile bir önceki konuda oluşturulan veri tabanını silelim.

```
bash-3.1$dropdb okul
DROP  DATABASE
```

Komut doğru olarak çalıştırıldığında ekrana "DROP DATABASE" mesajı gelecektir.

1.3.6. PostgreSQL Kullanıcısı Oluşturma

PostgreSQL kullanıcısı oluşturmak için aşağıdaki komut kullanılabilir.

```
createuser [parametre] [kullanıcı adı]
```

Kullanabilecek parametreler ve açıklamaları tablodaki gibidir.

Parametre	Açıklaması
-a	Oluşturulan kullanıcı başka kullanıcılar oluşturabilir.
-A	Yeni kullanıcı oluşturamaz.
-d	Kullanıcı veri tabanı oluşturabilir.
-D	Kullanıcı veri tabanı oluşturamaz.

Tablo 1.3: Createuser komutunun parametreleri

Aşağıdaki komutla bir kullanıcı oluşturalım.

```
bash-3.1$createuser bulent
Shall the role be a superuser? (y/n) n
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) n
CREATE ROLE
```

Bu şekilde bir komut ile kullanıcı oluşturulduğunda bize sırasıyla "Shall the role be a superuser?" "yeni kullanıcı superuser mi" sorusunu "n" ile cevap verirsek ardından "Shall the new user be allowed to create databases?" "yeni kullanıcı veri tabanı oluşturabilecek mi? ve "Shall the new user be allowed to create more new users?" "bu kullanıcı yeni bir kullanıcı oluşturabilecek mi?" sorularını soracaktır. Yukarıdaki sorulara verdiğimiz cevaplara göre yeni postgreSQL kullanıcısına sadece yeni veri tabanı oluşturma yetkisini verdik. Parametre kullanarak bu komutu aşağıdaki gibi yazabilirsiniz.

```
bash-3.1$createuser bulent –A –d
CREATE ROLE
```

1.3.7. PostgreSQL Kullanıcısı Silme

Bir PostgreSQL sunucusundaki kullanıcıları silmek için aşağıdaki komutu kullanabilirsiniz.

```
dropuser   [parametre] [kullanıcı adı]
```

Parametre olarak "–i " ifadesi kullanılır. Kullanıcı adı bölümüne ise var olan ve silmek istediğiniz kullanıcının adı girilir. "-i" parametresi kullanıldığında silme işlemi için onay sordurulur. Örnekte bu tip bir silme yapılacaktır.

```
bash-3.1$dropuser –i bulent
Role "bulent" will be permanently removed.
Are you sure ?(y/n) y
DROP ROLE
```

1.3.8. Terminal Tabanlı PostgreSQL Sorgu Arayüzü

PosgreSQL de terminal tabanlı olarak sorgulama arayüzünü kullanmak için "psql" komutu kullanılır. Komutun kullanımı aşağıdaki gibidir.

```
psql [parametre] [veri tabanı dosyası]
```

Psql komutu ile birlikte kullanılabilen parametreler ve anlamları aşağıdaki tablodaki gibidir.

Parametre	Açıklaması
-d veri tabanı	Terminal işlemi yapılacak veri tabanını seçmek için kullanılır.
-f dosya adı	Dosya içindeki SQL sorgulamalarının çağırılmasını sağlar.
-o dosya adı	Sorgu sonuçlarının yazılacağı dosya belirtilir.
-h host adı	PostgreSQL sunucusunun bulunduğu makine adı belirtilir.
-p port	PostgreSQL sunucu portu belirtilir.
-U kullanıcı adı	Veri tabanına bağlanacak kullanıcı ismi belirtilir
-V	Psql sürümünü göstermek için kullanılır.
-l	Var olan veritabanları listelenir.

Tablo1.4: Psql komutu parametreleri

Örnek 1:

Not: Uygulamalarımızı postgres kullanıcısında iken çalıştıracağız.

Geçiş için "#su postgres" komutunu kullanabilirsiniz. Ayrıca çalışmalarımızı /var/lib/pgsql/ klasörü altında yapmamızda fayda var.

```
bash-3.1$ psql -l
        List of databases
   Name    | Owner   | Encoding
  ---------+----------+----------
 denem1    | postgres | LATIN5
 denem2    | postgres | LATIN5
 latin5    | postgres | UTF8
 okul      | postgres | UTF8
 postgres  | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(7 rows)

bash-3.1$
```

Psql komutunun –l parametresi ile sunucumuzda var olan veri tabanlarını listelettik.

Örnek 2:

Bu örneğimizde öncelikle SQL komutlarının var olduğu bir dosya oluşturacağız.

Dosya adı : /var/lib/pgsql/olustur.sql

Aşağıdaki SQL komutlarını bu dosya içine yazınız.

```
drop table ogrenci;     // var ise ogrenci tablosunu silelim
create table ogrenci(ad varchar(20),soyad varchar(20));
insert into ogrenci values('mustafa','gunes');
select * from ogrenci;
```

Psql komutu ile bu dosya içindeki komutları okul adındaki veri tabanına bağlanarak çalıştıralım.

```
bash-3.1$ psql -f olustur.sql -d okul
DROP TABLE
CREATE TABLE
INSERT 0 1
   ad    | soyad
---------+-------
 mustafa | gunes
(1 row)

bash-3.1$
```

Örnek 3:

Daha önceden createdb komutu ile oluşturduğumuz "okul" adlı veri tabanına "psql" komutu ile bağlanmak;

```
bash-3.1$ psql -d okul
Welcome to psql 8.1.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

okul=#
```

Örnekte görüldüğü üzere –d parametresinin yanında bağlantı yapacağımız veri tabanın adını girdik. Böylelikle postgreSQL'in interaktif terminaline giriş yapmış olduk.

1.3.8.1. Terminal Tabanlı İnteraktif Arayüzün Kullanımı

Bir önceki konumuzda postgreSQL'in terminal tabanlı interaktif arayüzüne bağlanmıştık. İnteraktif terminal içinde kullanılabilecek bazı kısa yollar ve işlevleri aşağıdaki tabloda listelenmiştir.

Kısayol	İşlevi
\?	Terminal içinde kullanılabilecek tüm kısa yolları listeletir.
\q	Terminalden çıkış için kullanılır.
\c VERİ TABANI adı	Belirtilen veri tabanına bağlanılır.
\l	Veri tabanlarını listeler.
\dt	Bağlı bulunulan veri tabanı içindeki tabloları listeletir.
\d tabloadı	Tablo ile ilgili bilgiler görüntülenir (alanlar, tipleri vs.)

Tablo 1.5: Terminal ortamı kısa yollar

Şimdi bu kısa yollardan bazılarını kullanalım.

Örnek 1:

```
bash-3.1$ psql okul          (okul veri tabanına terminalden bağlantı)
Welcome to psql 8.1.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

okul=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 denem1    | postgres | LATIN5
 denem2    | postgres | LATIN5
 latin5    | postgres | UTF8
 okul      | postgres | UTF8
 postgres  | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(7 rows)
```

Örnekten de anlaşılabileceği gibi "psql okul" komutunu kullanarak okul veri tabanı bağlantılı terminale girdik. Terminal içinde iken "\l" komutunu kullanarak sunucumuzdaki tüm veritabanlarını listelettik.

Örnek 2:

```
okul=# \d ogrenci
          Table "public.ogrenci"
 Column |         Type                | Modifiers
 -------+-----------------------------+-----------
 ad         | character varying(20) |
 soyad    | character varying(20)  |

okul=#
```

Bu örneğimizde "\d ogrenci" komutu ile veri tabanımızda var olan "ogrenci" adlı tablomuzun özelliklerini gördük.

Örnek 3:

```
okul=# \q
bash-3.1$
```

"\q" kısa yoluyla da terminalden çıkış yaptık.

Terminal içinde birçok kısa yol bulunmaktadır. Kısa yollar ve ne işe yaradıklarını öğrenmek için terminalde "\?" kısa yolunu kullanabilirsiniz.

İnteraktif terminal içinde iken ayrıca direkt olarak SQL komutları da kullanabilirsiniz. Örneğin bir tablo oluşturabilir, silebilir veya içine kayıt ekleyebilirsiniz. Terminalde SQL komutları ";" olan kısma kadar icra edilir.

Örnek 4:

```
okul=# INSERT INTO ogrenci VALUES('gurcan','bildir'); enter
INSERT 0 1
okul=#
```

Yukarıdaki komut ile "ogrenci" tablosuna kayıt eklemiş olduk.

Bir diğer kullanım şekli aşağıdaki gibidir.

```
okul=# INSERT INTO  ogrenci VALUES (      enter
okul(# 'neslihan','bildir');        enter
INSERT 0 1
okul=#
```

Örnekte de görüldüğü üzere ilk satırın sonunda ";" işareti yok. İkinci satırın sonunda ise ";" var. Buradan icra edilen komutların sonunda mutlaka ";" olması gerektiği anlaşılıyor.

1.3.9. PostgreSQL Veri Tabanının Yedeklenmesi

PostgreSQL de oluşturduğumuz ve içine tablolar eklediğimiz, çeşitli ayarlamalar yaptığımız veya içine veri girdiğimiz veri tabanını bazen güvenlik amaçlı ya da başka bir sunucuda çalışmak amacıyla yedekleme ihtiyacı duyabiliriz. Yedekleme yapmak için de "**pg_dump"** komutunu kullanırız. Pg_dump komutunun kullanımı aşağıda verilmektedir.

```
Pg_dump  [veri tabanı]  [parametre]
```

"Pg_dump" komutu ile birlikte veri tabanının adı verilir ayrıca ihtiyaca binaen parametreler kullanılır. Kullanabilecek bazı parametreler ve anlamları tabloda verilmiştir.

Parametre	İşlevi
-f dosya adı	Yedeklemenin yapılacağı oluşturulacak dosya adı girilir.
-F format	Çıktı formatı belirlenir.
-t tablo	Sadece belirtilen tabloyu yedekler.
-a	Sadece verileri yedekler.

Tablo 1.6: pg_dump komutu parametreleri

Bundan sonraki adımda ise "pg_dump" komutunu kullanarak bazı örnekler yapalım.

Örnek:

```
bash-3.1$ pg_dump okul -f deneme1.sql
bash-3.1$
```

veya;

```
bash-3.1$ pg_dump okul  >deneme1.sql
bash-3.1$
```

Yukarıdaki örnekte okul veri tabanı dosyasının denem1.sql adında bir yedeği oluşturuldu. Şimdi de yedek dosyasın içeriğini görelim. "vi" editörü ile bakacağız. "vi" editörü neredeyse tüm Linux dağıtımları ile birlikte gelen bir editördür. Kullanımı ile ilgili dokümanları internetten rahatça bulabilirsiniz.

```
bash-3.1$ vi deneme1.sql--
-- PostgreSQL database dump
--
SET client_encoding = 'UTF8';
SET check_function_bodies = false;
SET client_min_messages = warning;
--
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres
--
COMMENT ON SCHEMA public IS 'Standard public schema';
SET search_path = public, pg_catalog;
SET default_tablespace = '';
SET default_with_oids = false;
--
-- Name: deneme; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--
CREATE TABLE deneme (
    "no" integer,
    adres character varying(20)
);
ALTER TABLE public.deneme OWNER TO postgres;
--
-- Name: ogrenci; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--
CREATE TABLE ogrenci (
    ad character varying(20),
    soyad character varying(20)
);
ALTER TABLE public.ogrenci OWNER TO postgres;
--
-- Data for Name: deneme; Type: TABLE DATA; Schema: public; Owner: postgres
--
COPY deneme ("no", adres) FROM stdin;
1       aydin
\.
--
-- Data for Name: ogrenci; Type: TABLE DATA; Schema: public; Owner: postgres
--
COPY ogrenci (ad, soyad) FROM stdin;
mustafa gunes
gurcan  bildir
gurcan  bildir
```

```
neslihan	bildir
neslihan	bildir
\.
--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--
REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;
--
-- PostgreSQL database dump complete
--
```

Yedek içinde görüldüğü üzere tüm "okul" veri tabanı içindeki tüm yapılan işlemlerin birer kaydı alındı. Bu yedek ile istenilen veri tabanı içine "okul" veri tabanı içinde yapılan işlemler dâhil edilebilir.

Şimdi de oluşan yedeği kullanarak şimdi oluşturacağımız "deneme4" adlı veri tabanına "okul" veri tabanındaki tüm tabloları ve diğer işlemleri yükleyelim.

```
bash-3.1$ createdb deneme4    // deneme4 adında veri tabanı oluştur.
CREATE DATABASE
```

"deneme4" veri tabanını oluşturduk. Şimdi de yedekten geri yükleme yapalım.

```
bash-3.1$ psql deneme4 -f deneme1.sql  //deneme4 veri tabanı içerisine
yedekleri yükle
SET
SET
SET
COMMENT
SET
SET
SET
CREATE TABLE
ALTER TABLE
CREATE TABLE
ALTER TABLE
REVOKE
REVOKE
GRANT
GRANT
```

Örnekten de anlaşıldığı gibi "okul" veri tabanından alınan yedekler "deneme4" adındaki veri tabanı içine aktarıldı. Aktarım işlemini **"psql deneme4 -f deneme1.sql**" komutu ile yaptık.

1.3.10. PostgreSQL Tüm Veri Tabanlarının Yedeklenmesi

PostgreSQL'de kayıtlı olan tüm veri tabanlarının yedeklerinin alınması için aşağıdaki komutu kullanabilirsiniz.

```
Pg_dumpall  [veri tabanı]  [parametre]
```

Örnek:

Örneğimizde postgreSQL de bulunan tüm veri tabanlarının yedeğini alıyoruz.

```
bash-3.1$ pg_dumpall  > hepsi.sql
bash-3.1$
```

Tüm veri tabanların yedeği metin(text) olarak hepsi.sql dosyasının içine yazılmış oldu.

1.3.11. PostgreSQL Veri Tabanının Geri Yüklenmesi

PostgreSQL de daha önceden oluşturduğumuz yedekleri geri yüklerken aşağıdaki komutu kullanmalıyız.

```
Pg_restore [veri tabanı]  [parametre]
```

Örnek:

Öncelikle "okul" veri tabanımızın tar uzantılı sıkıştırılmış bir yedeğini alalım.

```
bash-3.1$ pg_dump –Ft –b okul > okulyedek.tar
bash-3.1$
```

Bulunduğumuz klasörde "okulyedek.tar" adında bir dosya oluştu. Şimdiki basamakta ise bu oluşturulan yedek dosyasından "yeniokul" adlı veri tabanımıza geri yükleme yapalıdır.

```
bash-3.1$ pg_restore –d  yeniokul  okulyedek.tar
bash-3.1$
```

UYGULAMA FAALİYETİ

PostgreSQL veri tabanı sunucusunu bilgisayarınıza kurunuz.

İşlem Basamakları	Öneriler
➢ Fedora Core Linux işletim sistemi kurulu bir bilgisayarda PostgreSQL için gerekli paketleri internettten ya da işletim sistemi kurulum CD' sinden temin ediniz.	PostgreSQL sunucu kurulum paketleri şunlardır: ➢ postgresql-libs-8.1.4-1.i386.rpm ➢ postgresql-8.1.4-1.i386.rpm ➢ postgresql-server-8.1.4-1.i386.rpm
➢ Öğrenim faaliyeti-1' de anlatılan kurulum paketlerini bilgisayarınıza kurunuz.	➢ Postgres kullanıcısını oluşup oluşmadığını kontrol ediniz.
➢ Kurulum işlemi sonrasında PostgreSQL sunucunu başlatınız.	➢ Başlatma işleminin başarılı olup olmadığını kontrol ediniz.
➢ Veri tabanı initialize işlemini yapınız.	➢ Veri tabanı initialize işleminin sonucunda "/var/lib/psql" klasörü oluşturulup oluşturulmadığını kontrol ediniz.

UYGULAMA FAALİYETİ

PostgreSQL sunucusu kurulu olan bilgisayarınızda Türkçe karakterleri destekleyen bir "alistirma" adında veri tabanı dosyası oluşturunuz.

İşlem Basamakları	Öneriler
➢ Postgres kullanıcısına geçiniz.	➢ Geçiş yaparken "su postgres" komutunu kullanınız.
➢ Createdb komutunu kullanarak "alistirma" adında Türkçe karakter düzenini destekleyen veri tabanı oluşturunuz.	➢ Createdb komutunu kullanırken "–e" parametresini kullanınız.
➢ Veri tabanının oluşturup oluşturulmadığını kontrol ediniz.	➢ Psql –l komutu ile konrol işlemini yapabilirsiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi bir veri tabanı türü değildir?

A) Hiyerarşik veri tabanı
B) İlişkisel veri tabanı
C) Nesnel veri tabanı
D) Öznel veri tabanı

2. PostgreSQL veri tabanı yönetim sistemi ne tür bir veri tabanı sistemidir?

A) İlişkisel
B) Nesnel
C) Öznel
D) Hiyerarşik

3. Aşağıdaki komutlardan hangisi PostgreSQL'de veri tabanı oluşturmak için kullanılır?

A) Create database
B) Createvt
C) Makedb
D) Createdb

4. PostgreSQL!de veri tabanı sorgu arayüzüne geçiş yapmak için hangi komut kullanılır?

A) Postgres
B) Createdb
C) Psql
D) Pgsql

5. PostgreSQL sorgu arayüzünden çıkmak için aşağıdaki hangi komut kullanılmalıdır?

A) \q
B) Quit
C) Exit
D) End

6. "pg_dumpall > yedek.sql" komutu aşağıdaki işlemlerden hangisini yapar?

A) Yedek adlı veri tabanı dosyasının yedeğini alır.
B) PostgreSQL sunucusunda oluşturulmuş tüm veri tabanlarının yedeğini alarak "yedek.sql" adlı dosyaya aktarır.
C) "yedek.sql" adlı veri tabanı dosyasının yedeğini yine "yedek.sql" içine aktarır.
D) pg_dumpall ile yedek.sql dosyalarından hangisinin daha büyük olduğunu buldurur.

7. "pg_restore" komutunun görevi aşağıdakilerden hangisinde açıklanmıştır?

A) Veri tabanını yedekler.
B) Yedeklenmiş veri tabanını geri yükler.
C) Veri tabanını siler.
D) Veri tabanında yeni bir tablo oluşturur.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Yapısal sorgulama dili komutlarını kullanabileceksiniz.

ARAŞTIRMA

- SQL dili hakkında araştırma yapınız.
- SQL ile ilgili kitaplardan veya internetten yararlanabilirsiniz.

2. YAPISAL SORGULAMA DİLİ

2.1. Yapısal Sorgulama Dili SQL'un Tanımı

İlişkisel veri tabanı modeli ilk olarak Codd tarafından 1970 yılında oluşturuldu. İlişkisel veri tabanına ulaşmak ve içinde işlemler yapmak amacıyla da SQL "Structured Query Language" ortaya çıkmıştır. SQL kısaca veri tabanından veri almak veya veri tabanına veri eklemek için kullanılan bir dildir.

SQL standart bir veri tabanı sorgu dilidir. Birçok gelişmiş veri tabanı uygulamalarında kullanılabilir. Örneğin Acces, MySQL, PostgreSQL, SQL sunucu, Oracle SQL komutları iki başlık altında gruplandırılabilir:

- **DDL (Data Definition Language) : Veri tanımlama dili komutları**

Bazı komutları:

- **Create Table Komutu:** Tablo Oluşturma Komutu
- **Alter Table Komutu:** Tablo Değiştirme Komutu
- **Drop Table Komutu**: Tablo Silme Komutu
- **Create Index Komutu:** İndex Oluşturma Komutu
- **Drop Index Komutu:** İndex silme komutu

- **DML (Data Manipulation Language) : Veri işleme dili komutları**

Bazı komutları:

- INSERT INTO Komutu: Veri ekleme komutu
- UPDATE Komutu: Veri güncelleştirme (Değiştirme) komutu
- **DELETE Komutu**: Veri silme komutu
- **SELECT Komutu:** Veri tabanından veri seçme komutu

2.1.1. DDL Veri Tanımlama Komutları

Bu komutlar vasıtasıyla veri tabanında tablo oluşturulabilir, düzenlenebilir, tablolar silinebilir, tablolara yeni sütunlar eklenebilir veya silinebilir.

Bu bölümde komutlardan birkaçının nasıl kullanılacağını öğreneceğiz.

2.1.1.1. CREATE TABLE Komutu

Bu komut ile veri tabanı içinde bir tablo oluşturulabilir. Komutun kullanım düzeni aşağıdaki gibidir.

```
CREATE TABLE Tablo_Adı(alan_adı1 tip1,alan adı2 tip2 ,……..)
```

Not : SQL Komutlarını PostgreSQL veri tabanı sistemi içinde kullanacağız.Bu nedenle örneğimizi PostgreSQL oluşturacağımız "alistirma" veri tabanı içinde yapacağız. Veri tabanı oluşturma ve SQL komutlarını kullanacağımız psql terminaline girişi daha önceki öğrenme faaliyetinde görmüştünüz.

CREATE TABLE komutu içinde oluşturulabilecek tipler aşağıdaki tabloda verilmiştir.

Veri Tipi	Açıklama
CHAR	Karakter
CHAR(n)	Karakter katarı(örneğin n=6 karakter katarı=5 boşluk=0)
VARCHAR(n)	Karakter katarı (örneğin n=6 karakter katarı=5 boşluk=1)
TEXT	Karakter katarı (Bu değişkende herhangi bir karakter sayısı limiti yoktur.)
INT	Integer (4 byte)
SMALLINT	Integer (2 byte)
BIGINT	Integer (8 byte)
FLOAT	Kayan noktalı tek boyutlu
DOUBLE PRECISION	Kayan noktalı çift boyutlu
NUMERIC	Integer (Sınırlama: 1000 haneye kadar) ve Desimal
DECIMAL	Integer (Sınırlama:1000 haneye kadar digits) ve Desimal
DATE	Tarih
TIME(WITH TIMEZONE)	Saat, dakika, saniye
TIMESTAMP(WITH TIMEZONE)	Yıl, ay, gün, saat, dakika, saniye
INTERVAL	Aralık
BOOL	Mantıksal değer
SERIAL	Otomatik sayı vermek için kullanılır.

Tablo 2.1: Veri tipleri

Create Table komutu ile bir örnek yapalım.

Örnek 1:

Birinci örneğimizde "alistirma" adında bir veri tabanı oluşturulacak. Ardından bu veri tabanına terminalden giriş yapılacak.

```
[root@bulent ~]# cd /var/lib/pgsql/     "veri tabanı klasörüne gidiliyor"
[root@bulent pgsql]# su postgres        " postgres Kullanıcısına bağlanma"
bash-3.1$ createdb alistirma          "alistirma veri tabanı oluşturuluyor."
CREATE DATABASE
bash-3.1$ psql alistirma       "terminal ile alistirma veri tabanına
giriliyor."
Welcome to psql 8.1.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

alistirma=#
```

Bir sonraki adımda "alistirma" veri tabanı içinde "ogrenciler" adında bir tablo oluşturacağız.

```
alistirma=# CREATE TABLE ogrenciler(num int,ad varchar(20),soyad
varchar(20),
alistirma(# dogum_tarih date);
 CREATE TABLE
```

Komut sonrasında CREATE TABLE mesajı görülmüş ise tablo oluşturma işlemi başarıya ulaşmış demektir. Görüldüğü üzere "alistirma" veri tabanı içinde "ogrenciler" adında bir tablo oluşturuldu. Tablo içinde int tipinde "num" alanı 20 karakter uzunluklarında "ad" ve "soyad" alanları tarih "date" tipinde de "dogum_tarihi" alanı oluşturuldu.

CREATE TABLE içinde istendiğinde otomatik numara veren alanlarda oluşturabiliriz. Bunun için "serial" tipi kullanılır.

Örnek 2:

```
alistirma=# CREATE TABLE dersler(dersno serial,ders_adi
varchar(25));
NOTICE:  CREATE TABLE will create implicit sequence
"dersler_dersno_seq" for serial column "dersler.dersno"
CREATE TABLE
```

Görüldüğü üzere "dersno" alanının tipi "serial" olarak verildi. Tip "serial" verildiğinde "tabloadi_alan_seq" adında bir tablo daha oluşturulmuş oldu. Bu tablodan faydalanarak her yeni girilen kaydın numarasını otomatik verdirebiliriz. Aşağıdaki uygulamada bu alana veri girilmek gerektiğinde nasıl bir yol izleneceği anlatılmıştır.

```
alistirma=# insert into dersler
values(nextval('dersler_dersno_seq'),'matematik');
INSERT 0 1        "Tabloya veri girişi yaptık."
alistirma=# select * from dersler;      "Tablo içindeki tüm verileri görelim."
 dersno | ders_adi
---------+-----------
      1     | matematik
(1 row)

alistirma=# insert into dersler
values(nextval('dersler_dersno_seq'),'kimya');
INSERT 0 1  "Tabloya yeni bir veri girdik."
alistirma=# select * from dersler;   "Şimdi yine tablodaki tüm kayıtları
görelim."
 dersno | ders_adi
---------+-----------
      1     | matematik
      2     | kimya
(2 rows)

alistirma=#
```

Görüldüğü üzere "Create Table" komutu ile otomatik numara olan bir "dersno" alanı oluşturmuştuk ve yukarıdaki kodlarda ise oluşturulan bu tablo içerisine "nextval" fonksiyonunu kullanarak bir sonraki otomatik sayıyı tablomuza ekledik.

Not: ***"Insert Into"*** *komutu tabloya veri girmek için kullanılır. Bir sonraki konuda ayrıntılı olarak aktarılacaktır.* ***"Select"*** *komutu ise tablo içindekileri listelemek için kullanılmaktadır. Diğer öğrenme faaliyetinde ayrıntılı olarak açıklanacaktır.*

Kayıt ekleme sırasında "nextval" fonksiyonu yerine "currval" fonksiyonu kullanıldığında bir önceki otomatik numara verilecektir.

2.1.1.2. ALTER TABLE Komutu

Bir tablo oluşturulduktan sonra tabloya yeni alanlar eklemek veya tablodan alan silmek amacıyla "Alter Table" komutu kullanılır. "Alter Table" komutunun kullanımı aşağıdaki gibidir.

```
ALTER TABLE Tablo_Adı ADD/DROP COLUMN alan_adı;
```

Tabloya alan eklemek için "Add Column" tablodan alan silmek için ise "Drop Column" ifadesi kullanılır.

Yapılacak örnekte daha önceden oluşturduğunuz "ogrenciler" tablosuna yeni bir alan eklenecek.

Örnek 1:

```
alistirma=# select * from ogrenciler ;   "Tablo içindeki verileri listeleyelim."
 num |   ad       | soyad        | dogum_tarih
-----+------------+--------------+-------------
  10 | Mustafa    | NAZMAN       | 1978-10-26
  11 | Bulent     | VARDAL       | 1977-01-05
(2 rows)

alistirma=# ALTER TABLE ogrenciler ADD COLUMN adres
varchar(60);
ALTER TABLE   "Tabloya adres alanunu ekledik."
alistirma=# select * from ogrenciler ;   "Tablo içerisini tekrar listeleyelim."
 num |   ad       | soyad        | dogum_tarih | adres
-----+------------+--------------+-------------+--------------
  10 | Mustafa    | NAZMAN       | 1978-10-26  |
  11 | Bulent     | VARDAL       | 1977-01-05  |
(2 rows)
```

Örnekte "ogrenciler" tablosuna "Alter Table" komutu ile yeni bir "adres" adında bir alan eklenmiş oldu. Ekleme sırasında mutlaka eklenecek alanın tipi yazılmalıdır. Örnekte adres alanının tipi "varchar(60)" olarak verildi.

İkinci örnekte ise tablodan bir alan sileceğiz.

Örnek 2:

```
alistirma=# select * from ogrenciler;       "Tabloyu listeleyelim."
 num |   ad      | soyad         | dogum_tarih | adres
-----+----------+---------------+---------------+-------
  10 | Mustafa | NAZMAN   | 1978-10-26  |
  11 | Bulent    | VARDAL   | 1977-01-05  |
(2 rows)

alistirma=# ALTER TABLE ogrenciler DROP COLUMN adres ;
ALTER TABLE              "tablodan adres alanunu sildik."
alistirma=# select * from ogrenciler;   "tabloyu tekrar listeleyelim"
 num |   ad      | soyad         | dogum_tarih
-----+----------+--------------+-------------
  10 | Mustafa | NAZMAN | 1978-10-26
  11 | Bulent    | VARDAL | 1977-01-05
(2 rows)
```

Bu örneğimizde "Alter Table" komutu içinde "Drop Column" ifadesini kullanarak adres alanını sildik. Görüldüğü gibi adres alanının tipini yazmamıza gerek yok.

2.1.1.3. DROP TABLE Komutu

Var olan bir tabloyu içindeki veriler ile birlikte silmeye yarayan komuttur. Kullanımı aşağıda verilmiştir.

```
DROP TABLE Tablo_Adı ;
```

"Drop Table" ilgili bir örnek yapacağız

Örnek:

```
alistirma=# select * from ogrenciler;       "Tabloyu listeleyelim."
 num |   ad      | soyad         | dogum_tarih | adres
-----+----------+---------------+---------------+-------
  10 | Mustafa | NAZMAN   | 1978-10-26  |
  11 | Bulent    | VARDAL   | 1977-01-05  |
(2 rows)

alistirma=# DROP  TABLE ogrenciler;
DROP TABLE   "Ogrenciler tablosunu sildik."
alistirma=# select * from ogrenciler;   "Ogrenciler tablosunu listeleyelim."
ERROR:  relation "ogrenciler" does not exist   "ogrenciler tablosu yok
mesajı"
alistirma=#
alistirma=# \d     "Veri tabanı içerisndeki tabloları listeleyelim."
               List of relations
 Schema |       Name           |   Type         | Owner
-----------+-----------------------+------------+----------
 Public    | dersler                 | table         | postgres
 Public    | dersler_dersno_seq | sequence | postgres
(2 rows)     "listede ogrenciler tablosu yok."

alistirma=#
```

Örnekte daha önceden oluşturduğumuz "ogrenciler" adlı tabloyu siliyoruz. Silmek için "Drop Table ogrenciler;".

2.1.1.4. CREATE INDEX Komutu

Bir tablo içindeki verileri sırlamak için kullanılır. İndekslemedeki amaç veri tabanı içindeki verilere sıralı olması sayesinde daha hızlı bir şekilde ulaşmaktır. "Create Index" komutunun kullanımı aşağıdaki gibidir.

```
CREATE INDEX indeks_adı ON tablo_adı ( alan_adı1, alan_adı2,…..) ;
```

Şimdide "alistirma veri tabanının içinde öğrenciler tablosuna bağlı numaraları sıralatan bir indeks oluşturma çalışması yapacağız.

Örnek:

```
alistirma=# SELECT * FROM ogrenciler ;
 num  |ad          |soyad           | dogum_tarih
------+-----------+----------------+---------------------
  11  | Bulent     | VARDAL         | 1977-01-05
  10  | Mustafa    |  NAZMAN        | 1978-10-26
   8  | Ahmet      | OZKAN          | 1975-07-22
  15  | Telat      | GULER          | 1977-04-13
  13  | Murat      | OZDEVECI       | 1970-05-01

(5 rows)

alistirma=# CREATE INDEX ogr_num_ndx ON ogrenciler(num);
CREATE INDEX

alistirma=# \d ogrenciler
          Table "public.ogrenciler"
   Column    |         Type          | Modifiers
-------------+-----------------------+-----------
 num         | integer               |
 ad          | character varying(20) |
 soyad       | character varying(20) |
 dogum_tarih | date                  |
Indexes:
    "ogr_num_ndx" btree (num)
```

Tabloya bağlı olan indeks

İndeks oluşturma işlemi yapıldıktan sonra "\d tablo_adı" komutu yazıldığında tablonun tüm özellikleri ile birlikte tabloya bağlı olan "indeks" imi de görünmektedir. Bu uygulamada "ogrenciler" tablosunu "num" alanına göre indekslemiş olduk. Buradaki "btree" indeksleme metodunu ifade etmektedir.

2.1.1.5. DROP INDEX Komutu

"Drop Indeks" komutu var olan bir indeksi silmeye yarar. Kullanımı aşağıdaki gibidir.

```
DROP INDEX indeks_adı ;
```

Örnek:

```
alistirma=# DROP INDEX ogr_num_ndx;
DROP INDEX
alistirma=# \d ogrenciler
          Table "public.ogrenciler"
   Column     |         Type          | Modifiers
--------------+-----------------------+-----------
 num          | integer               |
 ad           | character varying(20) |
 soyad        | character varying(20) |
 dogum_tarih  | date                  |

alistirma=#
```

Örnekten de anlaşıldığı üzere indeks bileşeni silindi.

2.1.2. DML Veri İşleme Komutları

Tablodaki alanlar üzerinde işlem yaptıran komutlardır. Bu komutlarla tablo içindeki verilere ekleme, tablodaki verilerde düzeltme, veri silme gibi işlemler yapılabilir.

2.1.2.1. INSERT INTO Komutu

Tablo içine veri eklenmek istendiğinde bu komut kullanılır. Komutun kullanım şekli aşağıda verilmiştir.

```
INSERT INTO Tablo_Adı (alan1, alan2, alan3) VALUES (deger1,deger2,…….);
```

Örnek:

```
alistirma=# INSERT INTO ogrenciler VALUES(19,'Ahmet','ZAPIR','11-
27-1977');
INSERT 0 1
alistirma=# SELECT * FROM ogrenciler;
 num  |   ad    |  soyad     | dogum_tarih
------+---------+------------+-------------
  11  | Bulent  | VARDAL     | 1977-01-05
  10  | Mustafa | NAZMAN     | 1978-10-26
   8  | Ahmet   | OZKAN      | 1975-07-22
  15  | Telat   | GULER      | 1977-04-13
  13  | Murat   | OZDEVECI   | 1970-05-01
  19  | Ahmet   | ZAPIR      | 1977-11-27
(6 rows)
```

2.1.2.2. UPDATE Komutu

Tablo içindeki istenilen veri içinde değişiklik yapılma istendiğinde bu komut kullanılır. "Update" komutunun kullanım şekli aşağıdaki gibidir.

```
UPDATE Tablo_adı SET alan_adı=değer  WHERE koşul;
```

"Update" komutu ile ilgili örnek aşağıdadır.

Örnek:

```
alistirma=# SELECT * FROM ogrenciler ;
 num |   ad         |  soyad         | dogum_tarih
-------+-------------+---------------+-------------
  11   | Bulent       | VARDAL     | 1977-01-05
  10   | Mustafa      | NAZMAN     | 1978-10-26
   8   | Ahmet        | OZKAN      | 1975-07-22
  15   | Telat        | GULER      | 1975-04-13
  13   | Murat        | OZDEVECI   | 1970-05-01
  19   | Ahmet        | ZAPIR      | 1977-11-27
(6 rows)

alistirma=# UPDATE ogrenciler SET ad='Filiz' WHERE ad='Bulent';
UPDATE 1
alistirma=# SELECT * FROM ogrenciler ;
 num |   ad         |  soyad         | dogum_tarih
-------+-------------+---------------+-------------
  10   | Mustafa      | NAZMAN     | 1978-10-26
   8   | Ahmet        | OZKAN      | 1975-07-22
  15   | Telat        | GULER      | 1975-04-13
  13   | Murat        | OZDEVECI   | 1970-05-01
  19   | Ahmet        | ZAPIR      | 1977-11-27
  11   | Filiz        | VARDAL     | 1977-01-05
(6 rows)

alistirma=#
```

Tabloyu listeliyoruz.

Değişiklik sonrası tabloyu tekrar listeletiyoruz.

Örnekte "ad" alanı "Bulent" olan kaydın "ad" alanını "Filiz" olarak değiştirdik.

2.1.2.3. DELETE Komutu

Tablodan veri silmek istendiğinde bu komutu kullanabiliriz. "Delete" komutunun koşullu ve koşulsuz olmak üzere iki kullanım şekli vardır.

- **Koşulsuz kullanım**

```
DELETE FROM tablo_adı;
```

Bu kullanım metodunda "tablo_adı" ile belirtilen tablodaki tüm kayıtlar silinecektir. Bir örnek ile daha iyi anlayalım.

Örnek:

```
alistirma=# SELECT * FROM dersler;  "dersler tablosundaki verileri listeleme"
 dersno | ders_adi
--------+-----------
      1 | matematik
      3 | Kimya
      4 | Biyoloji
      5 | Fizik
      6 | Edebiyat
(5 rows)

alistirma=# DELETE FROM dersler;  "Dersler tablosundaki tüm verileri siliyoruz."
DELETE 5
alistirma=# SELECT * FROM dersler;
 dersno | ders_adi
--------+----------
(0 rows)
```

Dersler tablosunda 5 adet veri var.

Dersler tablosunda veri yok.

- **Koşullu kullanım**

```
DELETE FROM tablo_adı WHERE  koşul;
```

İstenildiği zaman "Delete" komutu ile birlikte "Where" sözcüğü de kullanılarak silme işlemi bir koşula bağlanabilir. Koşul işlemi içinde şu ifadeler kullanılabilir:

Karşılaştırma Operatörleri	Anlamı
=	Eşitlik
>	Büyük olma durumu
<	Küçük olma durumu
<>	Farklı olma durumu
<=	Küçük ya da eşit olma durumu
>=	Büyük ya da eşit olma durumu
!=	Farklı olma durumu

Tablo 2.2: Karşılaştırma operatörleri ve anlamları

Örnek 1:

Örnekte Adı "Filiz" olan kaydı silelim.

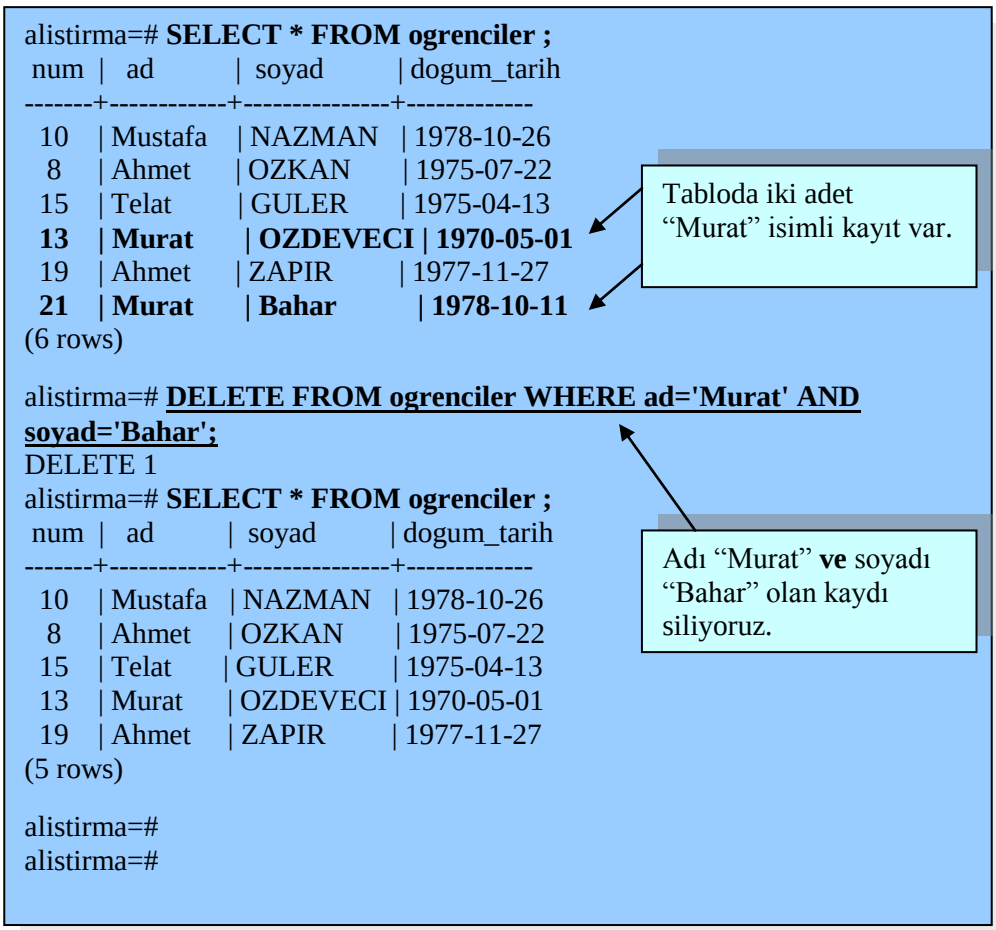
```
alistirma=# SELECT * FROM ogrenciler ;
 num |   ad       |  soyad         | dogum_tarih
------+------------+----------------+-------------
  10 | Mustafa    | NAZMAN    | 1978-10-26
   8 | Ahmet      | OZKAN     | 1975-07-22
  15 | Telat      | GULER     | 1975-04-13
  13 | Murat      | OZDEVECI  | 1970-05-01
  19 | Ahmet      | ZAPIR     | 1977-11-27
  11 | Filiz      | VARDAL    |1977-01-05
(6 rows)

alistirma=#
alistirma=# DELETE FROM ogrenciler WHERE ad='Filiz';
DELETE 1
alistirma=# SELECT * FROM ogrenciler ;
 num   |   ad       |  soyad         | dogum_tarih
--------+--------- +---------------+-------------
  10    | Mustafa | NAZMAN   | 1978-10-26
   8    | Ahmet   | OZKAN    | 1975-07-22
  15    | Telat   | GULER    | 1975-04-13
  13    | Murat   | OZDEVECI | 1970-05-01
  19    | Ahmet   | ZAPIR    | 1977-11-27
(5 rows)

alistirma=#
```

Silme işleminde birden fazla koşulda verilebilir. Koşullar birbirleri ile mantıksal olarak bağlanabilir. Kullanılan mantıksal deyimler ve anlamları tabloda verilmiştir.

Mantıksal İşlem	Anlamı
AND	Ve
OR	Veya
NOT	Değil

Tablo 2.3: Mantıksal değimler ve anlamları

Örnek 2:



Örnekten de anlaşılacağı üzere tablomuzda "Murat" adında iki kaydımız mevcuttur. Ancak bu kayıtların "soyad" alanları farklıdır. Sadece "ad" alanı "Murat" ve "soyad" alanı da "Bahar" olanı silmek istediğimizde bu iki koşulu birbirine "And" yani "ve" bağlacı ile bağlayabiliriz.

"Delete" ile ilgili örnekler artırılabilir. Örneğin doğum tarihi "1975" yılından önce olanları veya "1975" ten sonra olanları silmek isteyebiliriz. Bu duurmda "OR" bağlacını kullanabiliriz. Bu durumda komutumuz "DELETE FROM ogrenciler WHERE

dogum_tarih>1975 OR dogum_tarih<1975" olacaktır. Bu durumda tüm kayıtlar silinecektir. Nedeni de 1975 yılında doğanların birinci ayın birinde doğmamış olmalarıdır.

"Delete" komutu ile koşullandırma yaparken "Between", "Like", "is" gibi sözcüklerde kullanılabilir. Bu sözcükler listeleme komutu olan "Select" komutunda sırası gelince anlatılacaktır. Kullanım kuralları "Select" komutu ile aynıdır.

UYGULAMA FAALİYETİ

Aşağıdaki tabloda gösterilen alanlara sahip tablo oluşturunuz.

numara	üye adı	üye soyadı	üye telefon	üye adresi

İşlem Basamakları	Öneriler
➢ PostgreSQL de bir varitabanı oluşturun.	➢ Createdb komutunun kullanım şeklini inceleyiniz.
➢ Psql terminalini kullanınız.	➢ Terminale giriş komutunu kullanınız.
➢ Oluşturacağınız tablodaki alanların tiplerine karar veriniz.	➢ Tablodaki verilerin tiplerine karar verirken sayısal ya da karakterimi olacağına dikkat ediniz.
➢ "Create Table" komutu ile tablo oluşturma komutunu yazınız.	➢ "Create Table" komutunun kullanım kurallarına dikkat ediniz.
➢ Tablonun var olup olmadığını kontrol ediniz.	➢ Terminal komutlarından tabloları listeleyen komutu bulunuz.

UYGULAMA FAALİYETİ

Uygulama faaliyeti 1’ de oluşturduğunuz tabloya doğum tarihi alanını ekleyiniz.

İşlem Basamakları	Öneriler
➢ PostgreSQL terminal ortamına giriniz.	➢ Terminal kısayol komutlarını öğreniniz.
➢ Ekleyeceğiniz alanın tipine karar veriniz.	➢ Tarih tipinde veri ise “Date”tipini kullanabilirsiniz.
➢ “Alter Table” komutu ile alan ekleme işlemini yapınız.	➢ “Alter Table” komutunun kullanım şeklini öğreniniz.
➢ Terminal komutlarını kullanarak değişiklikleri test ediniz.	➢ Kısa yol tuşlarından \d komutunu kullanabilirsiniz.

UYGULAMA FAALİYETİ

Uygulama faaliyeti -1' de oluşturduğunuz ve uygulama faaliyeti -2' de yeni alan eklediğiniz tabloya aşağıdaki verileri ekleyerek numara alanına göre indeksleyiniz.

numara	üye adı	üye soyadı	üye telefon	üye adresi	Dogum tarihi
10	Mustafa	NAZMAN	123456	Bornova	10-10-1977
12	Ahmet	OZKAN	654321	Camdibi	11-11-1975

İşlem Basamakları	Öneriler
➢ PostgreSQL terminaline giriniz.	➢ Terminal kısa yol komutlarını öğreniniz.
➢ "Insert Into" komutu ile verileri giriniz.	➢ Komut içindeki verilerin dizilişine dikkat ediniz.
➢ Girilen verileri "Select" komutu ile kontrol ediniz.	➢ "Select" ile verilerin tümünü listeleyerek eklemenin başarılı olup olmadığını kontrol edebilirsiniz.

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdakilerden hangisi veri tanımlama komutlarından değildir?

A) Create Table
B) Drop Table
C) Delete
D) Alter Table

2. Aşağıdakilerden hangi komut ile yeni bir tablo oluşturulabilir?

A) Alter Table
B) Create Indeks
C) Createdb
D) Create Table

3. Aşağıdakilerden hangisi "Create Table" komutu ile oluşturulan tablo içindeki alan tiplerinden biri olamaz?

A) Table
B) Date
C) Varchar
D) İnteger

4. Aşağıdakilerden hangisi "Alter Table" komutunu tam olarak açıklar?

A) Yeni bir tablo oluşturmak için kullanılır.
B) Tablodaki alanların tipini değiştirmek için kullanılır.
C) Tabloya yeni bir alan eklemek ya da silmek için kullanılır.
D) Indeks oluşturmak için kullanılır.

5. "Alter Table" komutu ile hangisi kullanıldığında yeni bir alan ekleme işlemi yapılır?

A) Drop Column
B) Add Column
C) Sub Column
D) Create Column

6. Aşağıdaki hangi komut belirtilen tabloyu tamamen siler?

A) Delete
B) Remove
C) Drop Table
D) Remove Table

7. Tablo içine veri eklemek için aşağıdaki komutlardan hangisi kullanılır?

A) Add Table
B) Insert Into
C) Update
D) Select

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise bir sonraki öğrenme faaliyetine geçiniz.

ÖĞRENME FAALİYETİ–3

AMAÇ

Veri tabanındaki tabloları sorgulayabileceksiniz.

ARAŞTIRMA

Sevgili öğrenci, bu öğrenme faaliyetinden önce aşağıdaki hazırlıkları yapmalısınız.

- SQL'deki "Select" komutu ile sorgulama hakkında araştırma yapınız.

3. VERİ TABANI SORGULAMA

3.1. SELECT Komutuyla Tek Tablo Üzerinde Sorgulama Yapma

Tablodan ya da tablolardan veri seçmeye yarayan komuttur. "Select" Komutu SQL dilinin en önemli komutu olarak söylenebilir. Sorgular "Select" komutu ile yapılır. "Select" komutunun da "Delete" komutu gibi koşullu ve koşulsuz kullanım biçimleri vardır. Bunlara ilaveten "Select" komutuna özgü ifadeler de mevcuttur.

3.1.1. Koşulsuz Sorgulama

Select komutunun en basit kullanımı aşağıdaki gibidir.

```
SELECT Alan_Adı FROM Tablo_Adı ;
```

"Select" komutu içinde istenirse sadece bir alan ya da birden fazla alan listelenebilir. Tüm alanların listelenmesi istenirse alan_adı bölümüne "*" yazılmalıdır. Ayrıca istendiğinde birden fazla tabloda bir listede çıkarılabilir. Şimdi kullanımı ile ilgili örnekler yapalım.

Örnek:

Tablo Adı: ogrenciler

num	ad	soyad	dogum_tarih
10	Bulent	Vardal	01-05-1977
11	Deniz	Vardal	09-08-1998
12	Ceyda	Unal	04-28-2006

```
alistirma=# SELECT ad FROM ogrenciler;
  ad
--------
 Bulent
 Deniz
 Ceyda
(3 rows)
alistirma=# SELECT ad,soyad FROM ogrenciler;
  ad       | soyad
-----------+--------
 Bulent   | Vardal
 Deniz    | Vardal
 Ceyda    | Unal
(3 rows)
alistirma=# SELECT * FROM ogrenciler;
 num |   ad       | soyad       | dogum_tarih
-------+-----------+-------------+-------------
  10   | Bulent    | Vardal      | 1977-01-05
  11   | Deniz     | Vardal      | 1998-09-08
  12   | Ceyda     | Unal        | 2006-04-28
(3 rows)
```

"ogrenciler" tablosundan sadece "ad" alanı listelendi.

"ogrenciler" tablosundan "ad" ve "soyad" alanları listelendi.

"ogrenciler" tablosundaki tüm alanlar listelendi.

3.1.1.1. Tekrar Edilen Verilerin Yok Edilmesi (DISTINCT ifadesi)

Tablolarda aynı alanda birbirinin aynısı veriler olabilir. Biz bu verilerin listelenirken tekrarlanmamasını istiyorsak "Distinct" ifadesini kullanabiliriz.

Distinct ifadesine sahip bir Select komutunun kullanımı aşağıda verilmiştir.

```
SELECT DISTINCT Alan_Adı FROM Tablo_Adı ;
```

Distinct ifadesi ile ilgili bir örnek yapacağız.

Örnek:

Örneğimizi gerçekleştirmek için veri tabanı içinde aşağıdaki alanlara sahip bir "malzeme" adında tablo oluşturalım.

Tablo Adı: malzeme

mal_no (integer)	mal_ad (varchar(20))	mal_fiyat (integer)
1201	CDROM	40
1202	DVDROM	55
1200	CDROM	45
1204	DVDROM	60

Bu tablodaki malzeme çeşitlerini göstermek istediğimizde aşağıdaki komutu uygulamamamız gerekir.

```
alistirma=# SELECT DISTINCT mal_ad FROM malzeme;
 mal_ad
--------------
 CDROM
 DVDROM
(2 rows)
```

Anlaşılacağı üzere malzeme tablosundaki "mal_ad" alanında tekrar olmasını engelledik. Böylelikle malzeme tablosu içinde birbirinden farklı "mal_ad" alanını listeletmiş olduk.

3.1.1.2. Listelemenin Sıralı Olması (ORDER BY ifadesi)

"Select" komutunda sorgu sonucunun bir ya da daha fazla alana göre artan veya azalan sıralı bir şekilde gelmesini isteyebilir.

Bu durumda "Select" komutu içinde "Order By" ifadesini kullanılmalıdır. "Order By" ifadesinin kullanım şekli aşağıdaki gibidir:

```
SELECT  Alan_Adı FROM Tablo_Adı ORDER BY Alan_adı DESC/ASC ;
```

"Select" içinde kullanılan "Order By" ifadesi ile birlikte ASC ya da DESC ifadesi kullanarak sıralama metodunu belirleyebilirsiniz.

ASC 'nin anlamı sıralamanın seçilen alana göre artan, DESC ise sıralamanın azalan olmasını sağlar.

Örneklerle "Order By" ifadesini açıklayacağız.

Örnek:

İlk adımda "ogrenciler" tablosunda var olan verileri listeleteceğiz.

```
alistirma=# SELECT * FROM ogrenciler ;
 num |   ad    | soyad     | dogum_tarih
-----+---------+-----------+-------------
  10 | Bulent  | Vardal    | 1977-01-05
  11 | Deniz   | Vardal    | 1998-09-08
  12 | Ceyda   | Unal      | 2006-04-28
   5 | Gulen   | Vardal    | 1973-04-14
   9 | Filiz   | Unal      | 1978-04-06
(5 rows)
```

İkinci adımda tabloyu "num" alanına göre artan şekilde sıralayalım:

```
alistirma=# SELECT * FROM ogrenciler ORDER BY num ASC;
 num |    ad         | soyad      | dogum_tarih
-----+---------------+------------+-------------
   5 | Gulen         | Vardal     | 1973-04-14
   9 | Filiz         | Unal       | 1978-04-06
  10 | Bulent        | Vardal     | 1977-01-05
  11 | Deniz         | Vardal     | 1998-09-08
  12 | Ceyda         | Unal       | 2006-04-28
(5 rows)
```

Şimdiki uygulamada "num" alanına göre azalan şekilde sırlayalım:

```
alistirma=# SELECT * FROM ogrenciler ORDER BY num DESC;
 num |   ad    | soyad     | dogum_tarih
-----+---------+-----------+-------------
  12 | Ceyda   | Unal      | 2006-04-28
  11 | Deniz   | Vardal    | 1998-09-08
  10 | Bulent  | Vardal    | 1977-01-05
   9 | Filiz   | Unal      | 1978-04-06
   5 | Gulen   | Vardal    | 1973-04-14
(5 rows)
```

Şimdi de “soyad” a göre artan, “ad” a göre azalan sırlama yapalım:

```
alistirma=# SELECT * FROM ogrenciler ORDER BY soyad
ASC,ad DESC;
 num |   ad      | soyad      | dogum_tarih
-------+----------+------------+-------------
   9   | Filiz     | Unal       | 1978-04-06
  12   | Ceyda     | Unal       | 2006-04-28
   5   | Gulen     | Vardal     | 1973-04-14
  11   | Deniz     | Vardal     | 1998-09-08
  10   | Bulent    | Vardal     | 1977-01-05
(5 rows)
```

Görüldüğü üzere “soyad” ları artan sıralarken “soyad” ları aynı olanların “ad”larını azalan sıralattık. İstenirse ikisi de artan ya da azalan şekilde sıralanabilir.

3.1.2. Koşullu Sorgulama

“Select” komutu ile koşullu bir sorgulama yapılmak istendiğine komuta ek olarak “Where” ifadesi eklenmelidir. “Where” ifadesinin ardından koşul ya da koşullar dizisi eklenebilir.

Kullanım şekli aşağıda verilmiştir:

```
SELECT Alan_Adı FROM Tablo_Adı WHERE Koşul ya da koşullar ;
```

Sınamalarda kullanılabilecek karşılaştırma operatörleri daha önceki “Delete” komutunda tablo ile listelenmiştir.

Bu kullanım şeklini örneklerle daha iyi anlamaya çalışalım:

Örnek 1:

Bu örneğimizde daha önceden oluşturduğumuz, “ogrenciler” tablosunu kullanacağız. Öncelikle tabloda kayıtlı olan tüm verileri görelim.

```
alistirma=# SELECT  * FROM ogrenciler ;
 num |   ad      | soyad     | dogum_tarih
-------+----------+-----------+----------------
  10   | Bulent    | Vardal    | 1977-01-05
  11   | Deniz     | Vardal    | 1998-09-08
  12   | Ceyda     | Unal      | 2006-04-28
(3 rows)
```

Şimdiki durumda ise doğum tarihi 1980’den büyük olanları görelim:

```
alistirma=# SELECT * FROM ogrenciler WHERE Dogum_tarih>1980;
 num  | ad      | soyad     | dogum_tarih
--------+---------+-----------+-------------
  11    | Deniz   | Vardal    | 1998-09-08
  12    | Ceyda   | Unal      | 2006-04-28
(2 rows)
```

"Select" ifadesinde birden fazla koşul yerine getirilmek istendiğinde bu koşulları birbirleriyle gerekli mantıksal bağlantı kurarak yazmak gerekir. Kullanılabilecek bu mantıksal bağlantılar ve anlamları tablodaki gibidir:

Mantıksal Bağlaç	Anlamı
AND	Ve
OR	Veya
NOT	Değil

Bu tür komut kullanımı aşağıdaki gibi olacaktır:

SELECT Alan_Adı FROM Tablo_Adı WHERE Koşul1 Bağlaç **Koşul2** bağlaç **….. ;**

Bununla ilgili örnek yapalım.

Örnek 2:

Bu örnekte adı "Bulent" veya "Deniz" olanları listeletelim.

```
alistirma=# SELECT * FROM ogrenciler WHERE ad='Bulent' OR
ad='Deniz';
 num |  ad     | soyad    | dogum_tarih
------+--------+----------+-------------
  10 | Bulent  | Vardal   | 1977-01-05
  11 | Deniz   | Vardal   | 1998-09-08
(2 rows)
```

Örnek 3:

Şimdide "Not" ifadesi kullanarak "soyad" alanı "Vardal" olmayanları listeletelim.

```
alistirma=# SELECT * FROM ogrenciler WHERE NOT
Soyad='Vardal';
 num |  ad        | soyad   | dogum_tarih
-------+-----------+---------+-------------
  12   | Ceyda    | Unal     | 2006-04-28
   9   | Filiz      | Unal     | 1978-04-06
(2 rows)
```

Örneklerin sayısı daha da artırılabilir.

3.1.2.1. Koşullu Listelemede "IN" İfadesi

Birden fazla koşulu bağlaçsız bir şekilde ifade etmek gerektiğinde "In" ifadesi kullanılabilir. Kullanım şekli aşağıdaki gibidir.

```
SELECT Alan_Adı FROM Tablo_Adı WHERE alan_adi IN
(değer1,değer2….değerN);
```

Örnekle konuyu pekiştirelim.

Örnek:

"num" alanı 10 ya da 11 olanları listeletelim.

```
alistirma=# SELECT * FROM ogrenciler WHERE num IN (10,11);
 num |   ad   | soyad  | dogum_tarih
-----+--------+--------+-------------
  10 | Bulent | Vardal | 1977-01-05
  11 | Deniz  | Vardal | 1998-09-08
(2 rows)
```

3.1.2.2. Koşullu Listelemede Aralık Belirtme "Between" İfadesi

Koşullu listeleme yaparken aralı belirleme de kullanılacak yöntemlerden bir tanesi de "Between" ifadesi kullanmaktır. "Select" komutu içinde "Between" kullanımı aşağıdaki gibidir.

```
SELECT Alan_Adı FROM Tablo_Adı WHERE alan_adi BETWEEN
değer1 AND değer2
```

Between ifadesini bir örnekle anlamaya çalışalım.

Örnek:

Örnekte "num" alanındaki verilerin 1 ile 5 arasında olanlarını listeletelim.

```
alistirma=# SELECT * FROM ogrenciler WHERE num BETWEEN 1
AND 5;
 num  |  ad      | soyad  | dogum_tarih
------+--------+---------+-------------
   5     | Gulen | Vardal  | 1973-04-14
(1 row)
```

3.1.2.3. Karakter Katarı İçinde Olup Olmadığını Sorgulama "LIKE" İfadesi

"Select" komutu ile herhangi bir karakter katarı alanı içinde bir sözcük parçasının olup olmadığını ya da herhangi bir harfle başlayan veya biten kelimeleri listelemek istediğimizde "Like" ifadesini kullanabiliriz.

"Like" ifadesinin kullanım şekli aşağıdaki gibidir:

```
SELECT Alan_Adı FROM Tablo_Adı WHERE alan_adi LIKE
"%değer%";
```

"%" işareti koyulduğu tarafın ne olursa olsun anlamına geldiğini gösterir. Örneğin "al%" yazıldığında başlangıcı "al" olan tüm verileri ifade eder. Eğer her iki tarafa da "%" işareti eklenirse "%en%" içinde "en" olan tüm verileri ifade eder.

Bununa ilgili bir örnek yapalım;

Örnek:

```
alistirma=# SELECT * FROM ogrenciler WHERE ad LIKE '%en%';
 num |   ad   | soyad  | dogum_tarih
-----+--------+--------+-------------
  10 | Bulent | Vardal | 1977-01-05
  11 | Deniz  | Vardal | 1998-09-08
   5 | Gulen  | Vardal | 1973-04-14
(3 rows)
```

3.1.3. SELECT Komutunda Kullanılabilecek Aritmetiksel İfadeler

"Select" komutu içinde istendiğinde verileri toplatabilir, çarpabilir ya da bölüm ve benzeri işlem yaptırabiliriz. Yapabileceğimiz işlemlerin simgesi ve anlamı tablodaki gibidir.

Operatör	Görev
+	Toplama
-	Çıkarma
/	Bölme
*	Çarpama
^	Üs alma

Tablo 3.1: Operatörler ve anlamları

Şimdi yapacağımız birkaç örnekle aritmetiksel işlemleri kullanmayı öğrenelim. Yapacağımız örnek için aşağıdaki tabloyu oluşturalım.

Tablo Adı : Ürünler

mal_no	mal_ad	mal_fiyat	mal_adet	mal_KDV
10	Buzdolabı	700	5	18
11	Fırın	350	8	18
12	Televizyon	425	10	18
9	Fırın	250	21	18
15	Bulaşık makinesi	490	10	25

Örnekte malzemelerin ayrı ayrı KDV dâhil fiyatlarını gösterelim, KDV'li fiyat (fiyat*(kdv+100))/100 olarak bulunur.

Örnek:

```
alistirma=# SELECT mal_no,mal_ad,mal_fiyat*(mal_KDV+100)/100 AS
KDVdâhil, mal_KDV FROM ürünler ;
 mal_nu. |      mal_ad          | KDVdâhil   | mal_KDV
---------+----------------------+------------+---------
    10    | Buzdolabi            | 826        |    18
    11    | Firin                | 413        |    18
    12    | Televizyon           | 501        |    18
     9    | Firin                | 295        |    18
    15    | Bulasik Makinesi     | 612        |    25
(5 rows)
```

Yukarıdaki örnekte "AS" ifadesi ile sonucun hangi alanda çıkacağı gösterilir.

Select komutu ile kullanılan bazı fonksiyonlar ile veri sayısı buldurulabilir, veriler içindeki herhangi bir alanda kayıtlı en küçük ve en büyük değer buldurulabilir. SELECT içine but tip fonksiyonlar barındırır. Bu fonksiyonları sırayla öğrenelim.

3.1.3.1. COUNT Fonksiyonu

COUNT fonksiyonu sorgulama sonucunda ortaya çıkan verileri sayar. Kullanım şekli aşağıdaki gibidir.

```
SELECT COUNT(Alan_Adı) FROM Tablo_Adı WHERE koşul;
```

Count ifadesi ile birlikte koşul kullanarak sorgulama sonucunda kaç veri bulunduğunu saydırabiliriz.

Count fonksiyonu ile ilgili örneği inceleyelim.

Örnek:

```
alistirma=# SELECT COUNT(*) FROM urunler WHERE mal_fiyat>450;
 count
-------
     2
(1 row)
```

Bu sorguda fiyatı 500YTL' nin üzerinde kaç adet ürün olduğunu sorguladık. İstenirse sonuçta görülen count ifadesi yerine kendi yazdığımız bir ifade yazdırabiliriz.

```
alistirma=# SELECT COUNT(*) AS fiyati_450_den_fazla FROM urunler WHERE
mal_fiyat>450;
 fiyati_450_den_fazla
----------------------
                    2
(1 row)
```

AS ifadesi ile sonucumuza bir isim verdik.

3.1.3.2. Minimum ve Maksimum Fonksiyonu

Select komutu içinde belirtilen alandaki en küçük değeri bulmak için "MIN" en büyük değeri bulmak için "MAKS." fonksiyonu kullanılır.

Örnek:

```
alistirma=# SELECT MIN(mal_fiyat)AS en_ucuz_urun FROM URUNLER;
 en_ucuz_urun
--------------
          250
(1 row)
alistirma=#
alistirma=# SELECT MAX(mal_fiyat)AS en_pahali_urun FROM URUNLER;
 en_pahali_urun
----------------
           700
(1 row)
```

3.1.3.3. SUM Fonksiyonu

"SUM" fonksiyonu ile "Select" içinde seçilen herhangi bir alanın toplamını buldurabiliriz.

Örnek:

```
alistirma=# SELECT SUM(mal_fiyat)AS fiyatlar_toplam FROM urunler WHERE
mal_kdv=18;
 fiyatlar_toplam
-----------------
            1725
(1 row)
```

Bu örneğimizde KDV oranı %18 olan ürünlerin toplam fiyatını buldurmuş olduk.

3.1.3.4. AVG Fonksiyonu

"SUM" fonksiyonu ile "Select" içinde seçilen herhangi bir alanın toplamını buldurabiliriz.

Örnek:

```
alistirma=# SELECT AVG(mal_kdv)AS kdv_ortalamasi FROM urunler;
kdv_ortalamasi
--------------------
 19.4000000000000000
(1 row)
```

Üstteki örnekte KDV oranların ortalamasını buldurmuş olduk.

3.1.4. Gruplandırma İşlemi (GROUP BY)

Sorgulama sırasında bazen verileri belli alanlar göre gruplandırmak gerekebilir. Örnek verecek olursak "urunler" tablosunda KDV oranları %18 olanlar ile %25 olanların sayısını tek bir komutta görmek isteyebiliriz. Bu durumda KDV oranlarını gruplamak gerekir. Select komutunda gruplandırma işlemi yapmak istendiğinde "Group By" ifadesi kullanılır.

Şimdi az önce verdiğimiz örneği uygulayalım.

Örnek:

KDV'ye göre gruplayarak saydıralım.

```
alistirma=# SELECT mal_kdv,count(*) FROM urunler GROUP BY mal_kdv;
 mal_kdv | count
-----------+-------
      25     |     1
      18     |     4
(2 rows)
```

Görüldüğü üzere birbirinden farklı iki "KDV" oranına sahip ürünlerin sayılarını listelettik.

- **HAVING İfadesi**

"Having" ifadesi "Group By" ifadesi ile gruplanmış veriler üzerinde işlem yapmak amacıyla kullanılır. "Having" ifadesinde sonra mutlaka "SUM, AVG, MAKS., MIN" gibi aritmetiksel işlemler kullanılmalıdır. Örneğin, bir şirkette çalışanların ayrı ayrı bölümleri olduğunu düşünelim, bu bölümlerden maaş ortalamaları 1000 TL üzerindeki bölümlerin ve ortalama maaşlarının listelenmesini istediğimizde "Having" ifadesini kullanmalıyız.

Örnek:

Şimdi "Having" ifadesi ile bir sorgulama yapalım. Sorgulamamızda "matematik_notlari" tablosu olsun, sınıflardan not ortalamaları 50'nin üzerinde olanları sınıfları listeletelim. Öncelikle aşağıdaki tabloyu oluşturalım.

Tablo Adi : matematik_notlari

Num	Adi	Soyadi	Sinif	Notu
37	Ahmet	Zapir	10b	70
29	Umit	Cihan	10b	72
41	Cemil	Uysal	10a	75
38	Yusuf	Yildiz	10a	100
32	Bulent	Vardal	10c	35
35	Okan	Su	10c	50
34	Aslı	Topal	10b	100
42	Gokhan	Demir	10b	75

Şimdi üst kısımda sorulduğu gibi tablodan notlarının ortalamaları 50'nin üzerindeki sınıfları listeletelim.

```
alistirma=# SELECT sinif,avg(notu) AS ortalama  FROM matematik_notlari
GROUP BY sinif  HAVING avg(notu)>50;

sinif      | Ortalama
------------------------
10a        | 87.5000
10b        | 79.2500
(2 rows)
```

Görüldüğü üzere sonuç olarak 10C sınıfı listelenmedi çünkü 10C sınıfındaki matematik notlarının ortalaması 50' den büyük değildir.

3.2. SELECT Komutu ile Birden Fazla Tabloyu Sorgulama

Şu ana kadar "Select" komutu ile sadece tekbir tablo üzerinde sorgulama yaptık. Ancak SQL' in asıl kullanım amaçlarından ve en güçlü olduğu konulardan birisi de birden fazla tabloyu ilişkilendirerek sorgulamaktır.

Örnek 1:

"ogrenciler" tablosuna ek olarak "il" adında bir tablo oluşturalım. İçerisine üç adet kayıt giriyoruz.

Tablo adı: il

Num integer	İladi varchar(20)
10	Aydın
11	İzmir
12	Antalya

"ogenciler" tablosunda da üç adet veri var.

```
alistirma=# SELECT * FROM ogrenciler;
 num    |   ad       | soyad       | dogum_tarih
--------+----------+-------------+-------------
  10    | Bulent     | Vardal      | 1977-01-05
  11    | Deniz      | Vardal      | 1998-09-08
  12    | Filiz      | Unal        | 1978-03-14
(3 rows)
```

“il” tablosunda da üç adet veri var.

```
alistirma=# SELECT * FROM il;
 num  |  il adı
-------+---------
  10   | Aydın
  11   | İzmir
  12   | Antalya
(3 rows)
```

Şimdi de tabloların ikisini birden listeletelim:

```
alistirma=# SELECT * FROM ogrenciler,il;
 num  |   ad       | soyad      | dogum_tarih | num |  iladi
------+------------+------------+-------------+-----+---------
  10  | Bulent     | Vardal     | 1977-01-05  |  10 | Aydin
  11  | Deniz      | Vardal     | 1998-09-08  |  10 | Aydin
  12  | Filiz      | Unal       | 1978-03-14  |  10 | Aydin
  10  | Bulent     | Vardal     | 1977-01-05  |  11 | izmir
  11  | Deniz      | Vardal     | 1998-09-08  |  11 | izmir
  12  | Filiz      | Unal       | 1978-03-14  |  11 | izmir
  10  | Bulent     | Vardal     | 1977-01-05  |  12 | Antalya
  11  | Deniz      | Vardal     | 1998-09-08  |  12 | Antalya
  12  | Filiz      | Unal       | 1978-03-14  |  12 | Antalya
(9 rows)
```

Toplam 9 adet veri

Görüldüğü gibi iki tablo birden listelenmek istendiğinde 3 “ogrenciler” tablosundan 3 de “il” tablosundan veri tüm kombinasyonları olacak şekilde listelendi. Sonuç olarak 3x3=9 adet veri listelenmiş oldu. Diğer bir anlamda bu iki tablodaki veriler kartezyen çarpımı şeklinde listelendi.

İki tablodan sadece belirli alan adları listelenmek istendiğinde ise “Select” komutundaki alan_adı bölümüne “tablo_adi.alan_adi” ifadesi yazılmalıdır.

```
alistirma=# SELECT ogrenciler.ad,il.iladi FROM ogrenciler,il;
   ad      | iladi
-----------+---------
 Bulent    | Aydin
 Deniz     | Aydin
 Filiz     | Aydin
 Bulent    | izmir
 Deniz     | izmir
 Filiz     | izmir
 Bulent    | Antalya
 Deniz     | Antalya
 Filiz     | Antalya
(9 rows)
```

Yukarıdaki örnekten anlaşılacağı üzere "tablo_adi.alan_adi" şeklinde yazıldığında istenilen tablodan istenilen alanı çağırmış olduk. Ancak henüz istenilen sonuca ulaşamadık.

Şimdi her iki tablodan da istenilen kayda ait alanlar ile birleştirme işlemi yapalım. Görüldüğü üzere her iki tabloda "num" diye adlandırılan alanlar ortak. Bu alanların ortaklığı sayesinde bu her iki tabloyu birleştireceğiz. Burada önemli olan nokta her bir tablodaki alanların ayrı ayrı eşleştirilmesidir. Ogrenciler tablosundaki "num" alanını ifade ederken "ogrenciler.num" ifadesini kullanacağız. "il" tablosundaki "num" alanını ifade ederken de "il.num" ifadesini kullanacağız.

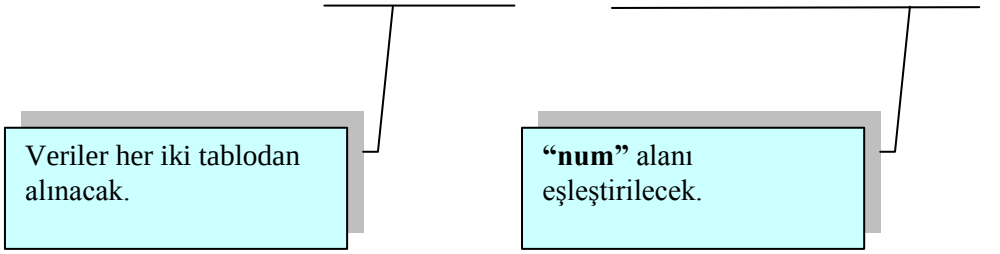
Şimdi bu her iki tablodaki kayıtları "num" alanları üzerinde birebir eşleştirelim yani "JOIN" işlemi yapalım. Bunun için aşağıdaki komutu kullanabiliriz.

Örnek:

```
alistirma=# SELECT *  FROM ogrenciler,il WHERE  ogrenciler.num=il.num;

 num     |   ad      | soyad     | dogum_tarih  | num   |  iladi
---------+-----------+-----------+--------------+-------+---------
  10     | Bulent    | Vardal    | 1977-01-05   |  10   | Aydin
  11     | Deniz     | Vardal    | 1998-09-08   |  11   | izmir
  12     | Filiz     | Unal      | 1978-04-06   |  12   | Antalya
(3 rows)
```

"Select" ifadesini inceleyelim.

SELECT * FROM ogrenciler,il WHERE ogrenciler.num=il.num;



Yukarıdaki örneğimizde tablodaki tüm alanları listeletmiş olduk. Ancak bazı durumlarda bu alanlardan sadece bir kısmını listeletmek isteyebiliriz. Aşağıdaki örnekteki gibi.

```
alistirma=# SELECT ogrenciler.num,ogrenciler.ad,il.iladi FROM ogrenciler,il
WHERE ogrenciler.num=il.num;
num |   ad   |  iladi
-----+--------+---------
  10 | Bulent | Aydin
  11 | Deniz  | izmir
  12 | Filiz  | Antalya
(3 rows)
```

SELECT ogrenciler.num,ogrenciler.ad,il.iladi FROM ogrenciler,il

WHERE ogrenciler.num=il.num;

Bu kısımda "**ogrenciler**" tablosundan "**num**" ve "**ad**" alanını aldık, "**il**" tablosundan da "**iladi**" alanını aldık

3.2.1. UNION (Birleşim) İfadesi

Select komutu içinde "UNION" ifadesi kullanılarak iki ayrı tabloyu birleştirebiliriz. Bu birleştirme sonucunda aynı kayda sahip veriler bir defa listelenecektir. Bir sonraki örnekte "UNION" ifadesi kullanacağız.

Bu amaçla aşağıdaki iki tabloyu oluşturmalıyız.

Tablo adı :malzeme1

malz_num integer	malz_ad varchar(20)	malz_fiyat integer
123	anfi cihazi	80
213	speaker	25
396	CD calar	95
443	speaker kablosu	10
445	tv	210

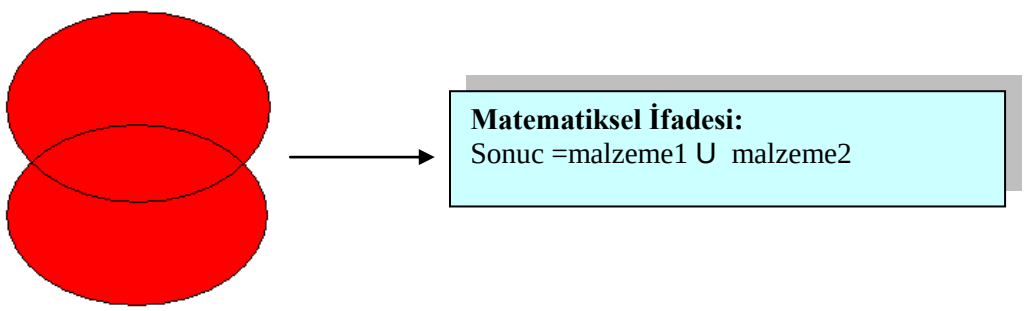
Tablo adı :malzeme2

malz_num integer	malz_ad varchar(20)	malz_fiyat integer
445	tv	210
213	speaker	25
444	buzdolabi	650
448	yazici	250

İki ya da daha fazla sorgu kümesinin birleşimi için şu koşullar gereklidir:

- Tablolardaki kolon sayıları eşit olmalıdır.
- Her kolon aynı tipte ve genişlikte veri alanı içermelidir.

Her iki tabloda da mevcut olan verilerin "malz_num" ları (213,445) olanlardır. Bu durumda bu malzeme numaralarına sahip veriler sadece bir kez listelenecektir.



Şekil 3.1: JOIN işleminin kümeler ile ifade edilmesi

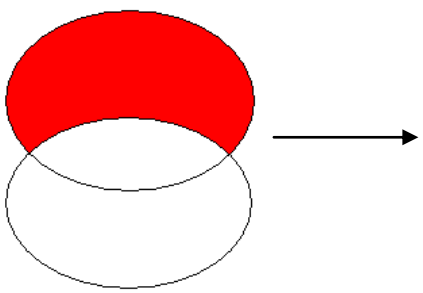
Şimdi "UNION" ifadesi ile sorgulayalım.

Örnek:

```
alistirma=#SELECT * FROM malzeme1 UNION SELECT * FORM malzeme2;
 malz_num |     malz_ad          | malz_fiyat
----------+----------------------+------------
      123 | anfi cihazi          |     80
      213 | speaker              |     25
      213 | speaker              |    150
      396 | CD calar             |     95
      443 | speaker kablosu      |     10
      444 | buzdolabi            |    650
      445 | tv                   |    210
      448 | yazici               |    250
(8 rows)
```

3.2.2. EXCEPT (Fark) İfadesi

İki tablo arasındaki farklı elemanları listeletmek istediğimizde "Ekscept" ifadesinden faydalanabiliriz. Aşağıdaki örnekte bu iki tablo arasındaki farklı olan elemanları listeletelim.



Matematiksel İfadesi:
Sonuc =malzeme1 - malzeme2

Şekil 3.2. Fark (EXCEPT) ifadesinin kümeler ile gösterimi

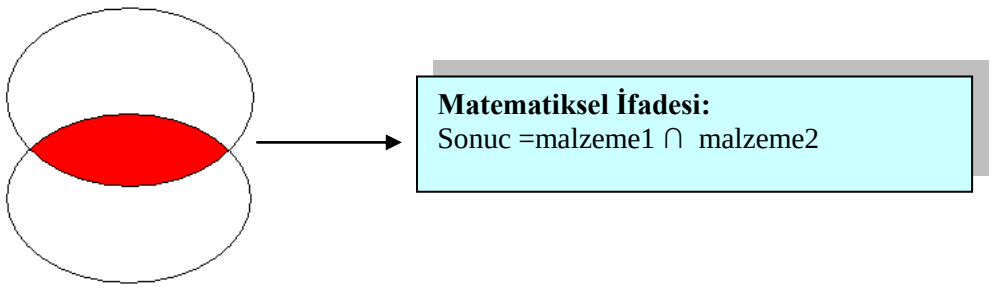
Örnek:

```
alistirma=# SELECT * FROM malzeme1 EXCEPT SELECT * FROM
malzeme2;
 malz_num |     malz_ad       | malz_fiyat
----------+-------------------+------------
      123 | anfi cihazi       |     80
      396 | CD calar          |     95
      443 | speaker kablosu   |     10
(3 rows)
```

Görüldüğü gibi mazleme1 tablosunda olup da malzeme2 tablosunda olmayan veriler listelendi.

3.2.3. INTERSECT (Kesişim) İfadesi

Kesişim ifadesi olan kesişim (INTERSECT) ile sadece iki tabloda da olan veriler listelenecektir.



Şekil 3.3. INTERSECT (Kesişim) ifadesinin kümeler ile gösterimi

Örnekle anlamaya çalışalım.

Örnek:

```
alistirma=# SELECT * FROM malzeme1 INTERSECT SELECT * FROM
malzeme2;
 malz_num | malz_ad | malz_fiyat
----------+---------+------------
      213 | speaker |     25
      445 | tv      |    210
(2 rows)
```

3.2.4. İç İçe (Nested) SELECT İfadesi

İç içe "Select" ifadesi sadece birden fazla tablonun olduğu durumlarda kullanılmaz. Sadece bir tabloda da iç içe "Select" ifadesi kullanılabilir. Bu durumları örneklerle açıklayalım.

Sadece bir tablo içinde iç içe "Select" komutunu kullanalım.

Örnek 1:

Bu örneğimizde malzeme1 tablosunda malzeme fiyatı, ortalama malzeme fiyatının üstünde olanları listeletmek istediğimizde de yazmamız gereken komutu yazacağız.

```
alistirma=# SELECT * FROM malzeme1 WHERE malz_fiyat>(SELECT
AVG(malz_fiyat) FROM malzeme1);
 malz_num | malz_ad       | malz_fiyat
----------+---------------+------------
      396 | CD calar      |         95
      445 | tv            |        210
(2 rows)
```

Örnekten de anlaşılacağı üzere ikinci "Select" ifadesi parantez içinde yazıldı.

Örnek 2:

Bu örneğimizde de en pahalı malzemeyi görelim.

```
alistirma=# SELECT * FROM malzeme1 WHERE malz_fiyat=(SELECT
MAX(malz_fiyat) FROM malzeme1);
 malz_num | malz_ad     | malz_fiyat
----------+-------------+------------
      445 | tv          |        210
(1 row)
```

Şimdide birden fazla tablo için iç içe "Select" kullanımı ile ilgili örnekler yapalım.

Örnek 3:

Bu örneğimizde malzeme1 tablosundan malzeme2 tablosundaki malzeme fiyatlarının içinde en az olandan daha ucuz olanları listeletelim.

```
alistirma=# SELECT *  FROM malzeme1 WHERE malz_fiyat<(SELECT
MIN(malz_fiyat) FROM malzeme2);
 malz_num |     malz_ad          | malz_fiyat
----------+----------------------+------------
      443 | speaker kablosu |         10
(1 row)
```

Şimdi de iç içe "Select" ifadesini birden fazla tabloda uygulayalım.

Birinci örneğimizde malzeme1 tablosunda malzeme2 tablosunun ortalama fiyatının altındaki fiyata sahip verileri listeletelim.

Örnek 4:

Örnekten de anlaşılacağı üzere ikinci "Select" ifadesi yani parantez içinde yazılmış olan ile "mazleme2" tablosundan ortalama malzeme fiyatını bulduarak birinci "Select" ifadesi içinde yazılan "Where" ifadesi ile malzeme1 tablosundaki fiyat ile karşılaştırdık.

3.2.5. ANY İfadesi

"Any" ifadesi sorgulamalarda "herhangi biri" anlamında kullanılır. Örneğin sınıftaki erkek öğrencilerin herhangi birinden daha yüksek not almış kız öğrencileri bulmak istediğimizde "any" ifadesini kullanabiliriz.

Şimdi yapacağımız bir örnekle "any" ifadesini anlayalım.

Örnek:

Bu sorguda şöyle bir ifade kullanalım: "malzeme1" tablosu içinde "malzeme2" tablosu içinde var olan ürünlerin herhangi birinin fiyatından daha yüksek fiyata sahip ürünleri listeletelim.

"ANY" ifadesini daha iyi anlayabilmek için öncelikle "malzeme1" tablosunda bulunan tüm kayıtları listeleyelim.

```
alistirma=# SELECT * FROM malzeme1;
 malz_num |     malz_ad        | malz_fiyat
----------+--------------------+------------
      123 | anfi cihazi        |         80
      213 | speaker            |         25
      396 | CD calar           |         95
      445 | tv                 |        210
      443 | speaker kablosu    |         10
(5 rows)
```

Şimdi de "malzeme2" tablosunu listeletelim.

```
alistirma=# SELECT * FROM malzeme2;
 malz_num |  malz_ad  | malz_fiyat
----------+-----------+------------
      445 | tv        |        210
      213 | speaker   |         25
      444 | buzdolabi |        650
      448 | yazici    |        250
      213 | speaker   |        150
(5 rows)
```

Şimdi de “any” ifadesi kullanarak örneğin başında ifade ettiğimiz sorgu sonucuna ulaşalım.

```
alistirma=# SELECT * FROM malzeme1 where malz_fiyat>ANY (SELECT
malz_fiyat FROM malzeme2);
 malz_num |   malz_ad   | malz_fiyat
----------+-------------+------------
      123 | anfi cihazi |         80
      396 | CD calar    |         95
      445 | tv          |        210
(3 rows)
```

Şimdi sonuçları teker teker değerlendirerek ”any” ifadesini anlamaya çalışalım.

1. veri malz_num=123, malz_ad= anfi cihazı , malz_fiyat=80

Görüldüğü gibi “malz_fiyat” verisi “malzeme2” tablosundaki en küçük olan “25” değerinden daha büyük, o nedenle sorgu sonucunda var.

2. veri malz_num=396, malz_ad= CD calar , malz_fiyat=95

Görüldüğü gibi “malz_fiyat” verisi “malzeme2” tablosundaki en küçük olan “25” değerinden daha büyük, o nedenle sorgu sonucunda var.

3. veri malz_num=445, malz_ad= tv , malz_fiyat=210

Görüldüğü gibi “malz_fiyat” verisi malzeme2 tablosundaki fiyatı “25” olan ve fiyatı “150” olan malzemelerin fiyatlarından yani herhangi birinden daha büyüktür. Sonuç olarak bu veri de sorguyu geçiyor.

Anlaşılacağı ürere “any” ifadesi veriler ancak tek tek incelendiğinden aydınlanıyor.

Bu ifadenin bir eşleniğini de aşağıdaki şekilde kullanabilirsiniz. İki sorgu sonucunda da aynı veriler listelenecektir.

```
alistirma=# SELECT * FROM malzeme1 WHERE malz_fiyat>(SELECT
MIN(malz_fiyat) FROM malzeme2);
 malz_num |   malz_ad   | malz_fiyat
----------+-------------+------------
      123 | anfi cihazi |         80
      396 | CD calar    |         95
      445 | tv          |        210
(3 rows)
```

3.2.6. ALL İfadesi

"ALL" ifadesi sorgularda "tümü, her biri " anlamında kullanılır. Örneğin bir sınıftaki bayan öğrencilerin tümünün notlarından daha yüksek nota sahip öğrenciyi bulmak istediğimizde bu ifadeyi kullanabiliriz. Şimdi bir örnek ile "all" ifadesini anlamaya çalışalım.

Şimdi tablolarımızla ilgili bir örnek yapalım.

Örnek:

Örneğimizde "malzeme2" tablosundan "malzeme1" tablosu içindeki fiyatların hepsinden daha büyük olanları malzemeleri seçtirelim.

```
alistirma=# SELECT * FROM malzeme2 WHERE malz_fiyat>ALL (SELECT
malz_fiyat FROM malzeme1);
 malz_num |  malz_ad     | malz_fiyat
----------+--------------+------------
      444       | buzdolabi  |       650
      448       | yazici       |       250
(2 rows)
```

3.2.7. EXISTS İfadesi

"EXISTS" ifade "var olan" anlamına gelir. "Exısts" ifadesinden sonra yazılan "Select" sorgu cümlesi ile yapılan sorgu sonucunda bir kayıt dahi var ise ilk "Select" ifadesi uygulanır. Eğer ikinci "Select" ifadesi ile yapılan sorgu sonucunda en az bir değer dönmüyor ise birinci "Select" komutu uygulanmaz.

Şimdi yapacağımız örnekler ile "Exısts" ifadesini anlamaya çalışalım.

Örnek 1:

Birinci durumda eğer mazlaeme2 tablosunda malz_adı alanında kayıtlı "tv" verisi bulunuyor ise ilk "Select" komutunun icra edilip edilmeyeceğine bakalım.

```
alistirma=# SELECT * FROM malzeme1 WHERE EXISTS (SELECT * FROM
malzeme2 WHERE malz_ad='tv');
 malz_num |     malz_ad        | malz_fiyat
----------+--------------------+------------
      123      | anfi cihazi          |        80
      213      | speaker              |        25
      396      | CD calar             |        95
      445      | tv                   |       210
      443      | speaker kablosu |        10
(5 rows)
```

Anlaşılacağı üzere “malzeme2” tablosunda “malz_ad” alanında kayılı “tv” verisi mevcut. Bu durumda “mazleme1” e ait olan “Select * From malzeme1” sorgusu çalıştırılacaktır.

Bir sonraki örneğimizde “malzeme2” tablosunda var olmayan bir bir “malz_ad” verisini sorgulayalım.

Örnek 2:

```
alistirma=# SELECT * FROM malzeme1 WHERE EXISTS (SELECT * FROM
malzeme2 WHERE malz_ad='matkap');
 malz_num | malz_ad | malz_fiyat
----------+---------+-----------
(0 rows)
```

Anlaşılacağı üzere “mazleme2” tablosu içinde “malz_ad” alanında “matkap” verisi olmadığı için malzeme1 sorgusu sonucunda hiçbir veri listelenemedi.

Bunun tam tersi işlemi içinde “Not Exists” ifadesi kullanabiliriz. “matkap” verisi var olmadığında listeleme yapabilmesi koşuluna ait komutu yazalım ve sonuçları yorumlayalım.

Örnek 3:

```
alistirma=# SELECT * FROM malzeme1 WHERE NOT EXISTS (SELECT *
FROM malzeme2 WHERE malz_ad='matkap');
 malz_num |     malz_ad        | malz_fiyat
----------+--------------------+-----------
      123 | anfi cihazi        |         80
      213 | speaker            |         25
      396 | CD calar           |         95
      445 | tv                 |        210
      443 | speaker kablosu    |         10
(5 rows)
```

Sonuçtanda anlaşılacağı üzere “malzeme2” tablosunda “matkap” verisi olmadığı için birinci “Select” sorgulaması sonucunda tablodaki veriler listelendi.

UYGULAMA FAALİYETİ

Aşağıda gösterilen tabloyu oluşturarak verileri giriniz. Giriş işleminden sonra 10A ve 10B sınıfındaki öğrencilerin not ortalamalarını ayrı ayrı gösterten sorguyu yapınız.

Numara	Ad	Soyad	Sınıf	Notu
21	Ahmet	Yaman	10b	68
22	Mehmet	Hanay	10b	72
23	Aslı	Topal	10a	88
28	Yusuf	Yıldız	10a	90
41	Cemil	Uysal	10b	75

İşlem Basamakları	Öneriler
➢ PostgreSQL terminaline giriniz.	➢ Terminal komutlarına dikkat ediniz.
➢ “Create Table” komutu kullanarak uygun alan tiplerine göre tabloyu oluşturunuz.	➢ Tablo oluştururken en uygun veri tiplerini kullanınız.
➢ “Select” komutunu kullanarak yukarıda istenen sorguyu gerçekleştiriniz.	➢ Group By ifadesini kullanınız.

UYGULAMA FAALİYETİ

Aşağıda gösterilen tabloyu oluşturarak verileri giriniz. Veri giriş işleminden sonra bu iki tabloyu bire bir eşleştirerek listeletiniz.

Numara	Adres	Telefon
21	Aydın	1234567
22	Ankara	6543216
23	Aydın	9871235
28	Denizli	1237894
41	İncirlova	1112223

İşlem Basamakları	Öneriler
➢ PostgreSQL terminaline giriniz.	➢ Terminal komutlarına dikkat ediniz.
➢ "Create Table" komutu kullanarak uygun alan tiplerine göre tabloyu oluşturunuz.	➢ Numara alanının tipinin diğer tablodaki numara ile aynı olmasına dikkat ediniz.
➢ "Select" komutunu kullanarak yukarıda istenen sorguyu gerçekleştiriniz.	➢ İki tabloyu da "Join" işlemi ile birleştiriniz. "Where" ifadesi kullanınız.

ÖLÇME ve DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyunuz ve doğru seçeneği işaretleyiniz.

1. Aşağıdaki komutlardan hangisi ile tablolar sorgulanabilir?

A) SELECT
B) INSERT INTO
C) UPDATE
D) QUERY

2. Sorgulama içinde koşul vermek için aşağıdaki ifadelerden hangisi kullanılabilir?

A) WHICH
B) WHERE
C) WHO
D) WHOM

3. "Select" ile sorgulama yaparken sorgulama ifadesinin artan bir sıralamada yapılması için aşağıdaki ifadelerden hangisi kullanılır?

A) ORDER BY alan DESC
B) ORDER BY alan ASC
C) GROUP BY alan ASC
D) LIST BY alan DESC

4. Select komutu içinde kullanılan "Where" ifadesi ile "Lıke" sözcüğü ile birlikte "ah%" ile sorgulama yapıldığında hangi veriler seçilir?

A) Verilen alan içindeki verilerden sonu "ah" olanlar
B) Verilen alan içindeki verilerden içinde "ah" olanlar
C) Verilen alan içindeki verilerden başlangıcı "ah" olanlar
D) Verilen alan içindeki verilerden başlangıcı "ah" olmayanlar

5. "SELECT * FROM ogrenciler WHERE num BETWEEN 21 AND 29" ifadesini açıklaması aşağıdakilerden hangisinde doğru olarak verilmiştir?

A) "ogenciler" tablosundaki num alanı 29 dan küçük olanlar
B) "ogrenciler" tablosundaki num alanı 21 den büyük olanlar
C) "ogrencier" tablosundaki num alanı 21 ile 29 arasında olanlar
D) "ogrenciler tablosundaki num ad alanı "a" ile başlayanlar

6. "SELECT ad FROM ogrenciler WHERE notu=(SELECT maks.(notu) FROM ogrenciler)" sorgusunun anlam aşağıdakilerden hangisinde tam doğru olarak verilmiştir?

A) "ogrenciler" tablosundaki en yüksek notlu öğrencinin adını gösterir.
B) "ogrenciler" tablosundaki en düşük notlu öğrencinin adını gösterir.
C) "ogrenciler" tablosundaki en yüksek notlu öğrencinin tüm bilgileri
D) "ogrenciler" tablosundaki notu olmayan öğrencileri gösterir.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarıyla karşılaştırınız. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt ettiğiniz sorularla ilgili konuları faaliyete geri dönerek tekrarlayınız. Cevaplarınızın tümü doğru ise "Modül Değerlendirme" ye geçiniz.

MODÜL DEĞERLENDİRME

Modülde yaptığınız uygulamaları tekrar yapınız. Yaptığınız bu uygulamaları aşağıdaki tabloya göre değerlendiriniz.

Bu modül kapsamında aşağıda listelenen davranışlardan kazandığınız becerileri **Evet** ve **Hayır** kutucuklarına (X) işareti koyarak kontrol ediniz.

Değerlendirme Ölçütleri	Evet	Hayır
1. İlişkisel veri tabanı oluşturabiliyor musunuz?		
2. Veri tabanı içinde tablolar oluşturabiliyor musunuz?		
3. Tablolara veri ekleyebiliyor musunuz?		
4. Tablodan veri silebiliyor musunuz?		
5. Veri güncellemesi yapabiliyor musunuz?		
6. Verileri sorgulayabiliyor musunuz?		

DEĞERLENDİRME

Değerlendirme sonunda "Hayır" şeklindeki cevaplarınızı bir daha gözden geçiriniz. Kendinizi yeterli görmüyorsanız öğrenme faaliyetlerini tekrar ediniz. Bütün cevaplarınız "Evet" ise bir sonraki modüle geçmek için öğretmeninize başvurunuz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1.	**D**
2.	**A**
3.	**D**
4.	**C**
5.	**A**
6.	**B**
7.	**B**

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1.	**C**
2.	**D**
3.	**A**
4.	**C**
5.	**B**
6.	**C**
7.	**B**

ÖĞRENME FAALİYETİ-3'ÜN CEVAP ANAHTARI

1.	**A**
2.	**B**
3.	**B**
4.	**C**
5.	**C**
6.	**A**

KAYNAKÇA

- MASUDA Yoichi, Bülent VARDAL, **Web Sistem Uygulamaları**, Jıca–Meb Endüstriyel Otomasyon Teknolojileri Kurulum Projesi, Eylül, 2004.

- UYSAL Mithat, **SQL Veri Tabanı Sorgulama Dil**i, Beta Yayınları, Mart 1999, İSTANBUL.

- GESCHWINDE Ewald,S, Hans-Jürgen CHÖNIG, **PostgreSQL Developer's Handbook**, Sams Publishing, Dec, 2001.

- http://web.sakarya.edu.tr/~erdal/SQL%20DERS%20NOTU%202005-2006.doc