Veri Tabanı Yönetimi ve SQL

Bu bölümde öğrenilecek kavramlar.

- **Veritabanı (VT)**
- **İlişkisel model**
- **Yabancı anahtar**
- **Veritabanı yöneticisi**
- **Veritabanı yaratmak**
- **Veritabanı Yönetim**
- **Sistemleri Mantıksal tasarım**
- **Anahtar öğe**
- **SQL komutları**
- **Veritabanı nesneleri**

Veri Tabanı

Büyük boyutlu veriler için iki temel problem

1. Nerede saklanacak

   Bir dosyaya yazdırmak çözüm olur mu ????

```
#include <stdio.h>
int main(){
    FILE *dosya = fopen("dosya.txt", "w");
    fprintf(dosya, "Merhaba 100 petabyte veri yazabilirmiyim amca");
    fclose(dosya);
    return 0;
}
```

2. Nasıl veri çekilecek

   Dosyadan okumakmı!

```
#include <stdio.h>
 int main(){
   FILE *dosya = fopen("dosya.txt", "r");
   while(! feof(dosya) ){
       putchar(fgetc(dosya));
   }
   fclose(dosya);
   return 0;
}
```

Veri Tabanı

- 1970'li yıllara kadar veri tabanı (VT) uygulamaları, dosya işleme uygulamaları biçiminde yapılmıştır.
- Verinin organize olarak tutulması ve üzerinde işlemler yapılması önemli ancak zor bir programlama faaliyeti olarak süregelmiştir.
- Verilerin tutulduğu tabloların matematiksel altyapısı ile ilgili olarak *T. E.* ***Codd*** tarafından yapılan çalışma (Codd-1970) ile birlikte bugün ilişkisel VT diye bilinen model ortaya atılmış ve günümüzün VT sistemlerinin temel altyapısını oluşturulmuştur.
- Bu modele göre her tablo matematiksel olarak varlıklardan oluşan bir kümedir ve kümeler için tanımlı işlemler tablolar üzerinde de geçerlidir.
- Veri tabanlarında birden çok tablo olabileceğinden, bunlar arasındaki ilişki de bir küme işlemi olarak tanımlanabilmektedir

Neden Veri Tabanı ?

- Sürekli güncellenmesi gereken değişken veriler
  - Bir işletmeye ait veriler
  - Bir kuruma ait veriler
  - Bir sisteme ait veriler …..
- Kayıpsız depolanması gereken veriler
- Birbiri ile ilişkili olarak kaydedilmesi gereken veriler
- Adeti ve boyutu çok olan veriler

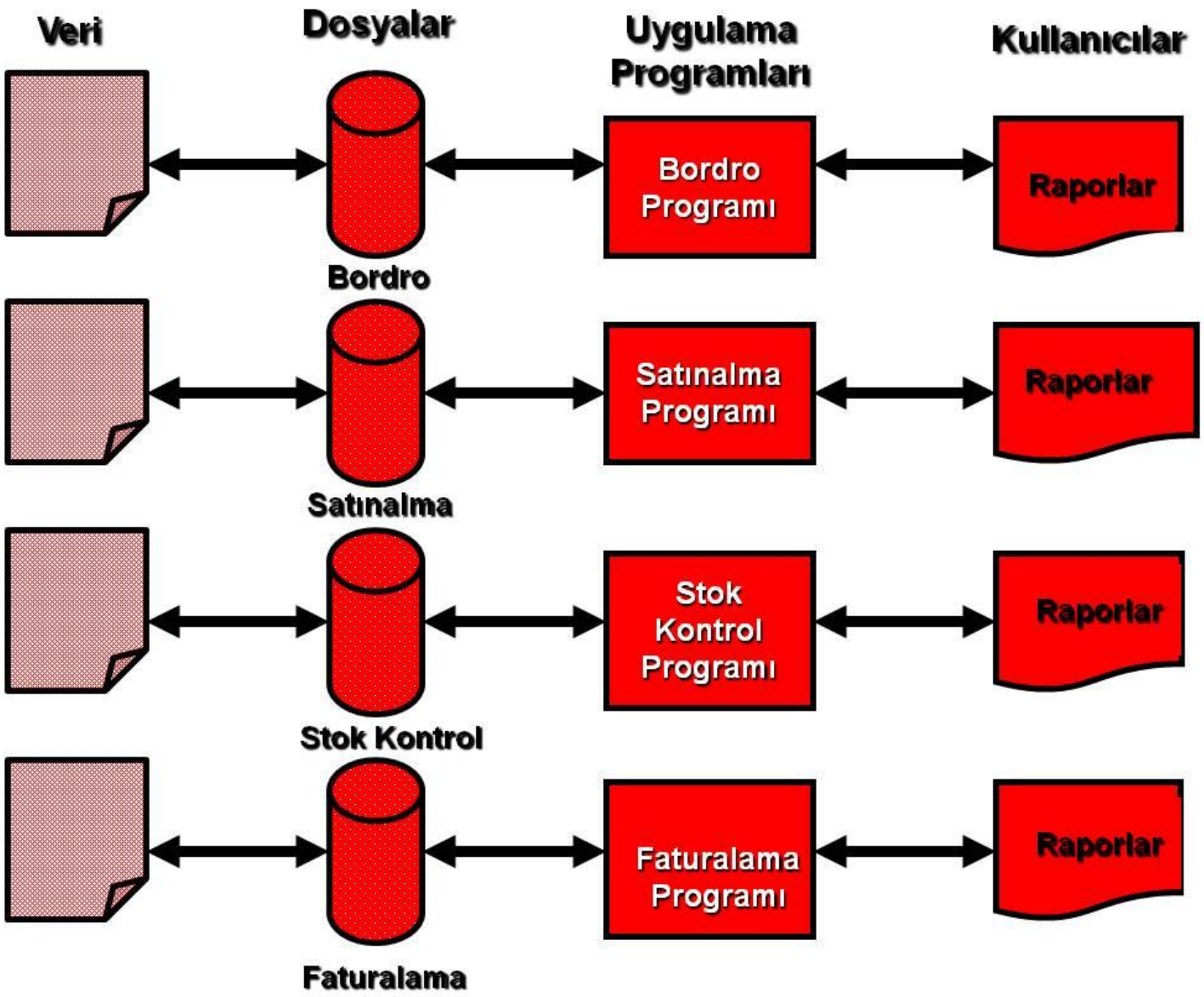
Veri Tabanı Yönetimi

- Veri tabanı (VT) birbirleri ile ilişkili verilerin bütünleşik olarak tutulduğu bir alandır.
- Veriler Tablolarda tutulur.
  - Alanlar
  - Kayıtlar

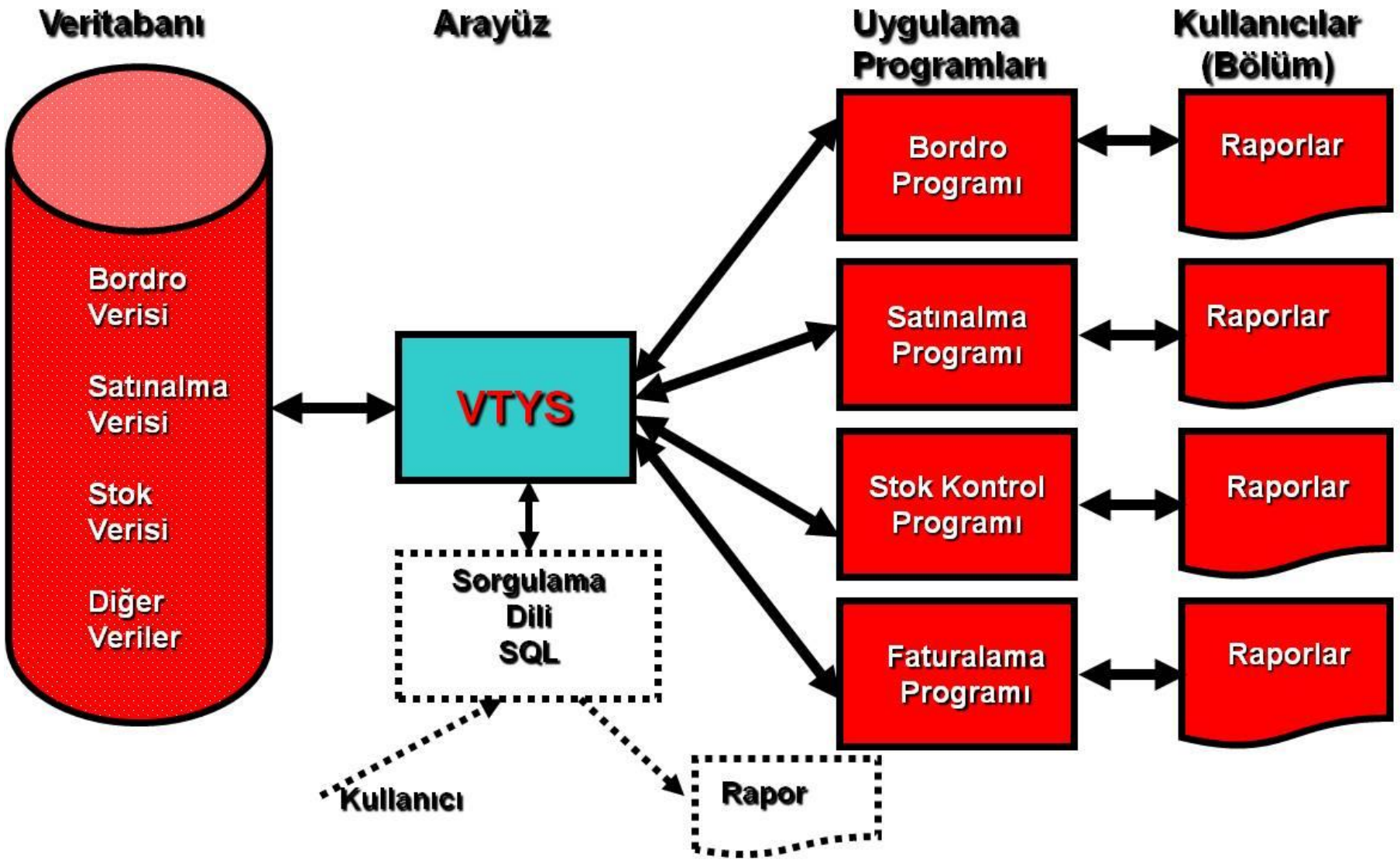
Employee	LastName	FirstName	Title	TitleOfCou	BirthDate	HireDate	Address
1	Davolio	Nancy	Sales Rep	Ms.	12/8/48	5/1/92	507 - 20th A
2	Fuller	Andrew	Vice Presi	Dr.	2/19/52	8/14/92	908 W. Ca
3	Leverling	Janet	Sales Rep	Ms.	8/30/63	4/1/92	722 Moss
4	Peacock	Margaret	Sales Rep	Mrs.	9/19/37	5/3/93	4110 Old F
5	Buchanan	Steven	Sales Mar	Mr.	3/4/55	10/17/93	14 Garrett
6	Suyama	Michael	Sales Rep	Mr.	7/2/63	10/17/93	Coventry H
7	King	Robert	Sales Rep	Mr.	5/29/60	1/2/94	Edgeham
8	Callahan	Laura	Inside Sal	Ms.	1/9/58	3/5/94	4726 - 11th
9	Dodsworth	Anne	Sales Rep	Ms.	1/27/66	11/15/94	7 Houndst

Veri Tabanı Yönetimi

- VT'yi basit olarak birden fazla tablodan (ilişkiden) oluşan bir sistem olarak tanımlamak da mümkündür.
- VT uygulamasına örnek olarak:
  - Türkiye Cumhuriyeti vatandaşlarının bilgilerinin tutulduğu veri tabanı,
  - Bankalarda mevduat hesapları ile ilgili verilerin tutulduğu mevduat veri tabanı,
  - Üniversitelerde öğrenciler, dersler, öğretim elemanları ve dersliklerle ilgili verilerin tutulduğu veri tabanı
- Verinin organize olarak tutulmasını, güncellemesini ve saklanmasını gerektiren her uygulama VT yaratmak zorundadır.


Veri
Dosyalar
Uygulama Programları
Kullanıcılar
Bordro Programı
Raporlar
Bordro
Satınalma Programı
Raporlar
Satınalma
Stok Kontrol Programı
Raporlar
Stok Kontrol
Faturalama Programı
Raporlar
Faturalama

- Veri tekrarlılığı
- Dosyalar arası kopukluk
- Program-veri bağımlılığı


Veritabanı
Arayüz
Uygulama Programları
Kullanıcılar (Bölüm)
Bordro Verisi
Satınalma Verisi
Stok Verisi
Diğer Veriler
VTYS
Sorgulama Dili SQL
Kullanıcı
Rapor
Bordro Programı
Satınalma Programı
Stok Kontrol Programı
Faturalama Programı
Raporlar
Raporlar
Raporlar
Raporlar

Veri Tabanı Yönetim Sistemleri

- VT üzerinde yapılabilecekler:
  - tablolarını yaratmak,
  - bu tablolara veri girişi yapmak,
  - var olan veride değişiklik yapmak
  - geçersiz verileri silmek gibi işlemlerin yapılması
- Veri tabanının yönetimini sağlayan yazılım sistemleri Veritabanı Yönetim Sistemleri (VTYS) olarak adlandırılır.
  - *Oracle,*
  - *MySQL,*
  - *DB2,*
  - *PostgreSQL,*
  - *Microsoft SQL Server*
- VTYS de veriler Structured Query Language (SQL) ile yönetilir.

İlişkisel Veri Tabanı ve İlişkisel Model

- İlişkisel veri tabanındaki temel yapı iki boyutlu bir tablodur.
- Tablonun sıralarında bir varlıkla ilgili kayıtlar tutulur.
- Örneğin, bir üniversitedeki öğrencilerle ilgili kayıtlar **ÖĞRENCİ** isimli ve sıra ve sütunlardan oluşan bir tabloda tutulabilir.
- **ÖĞRENCİ** tablosunun her sırasında bir öğrenci ile ilgili kayıt, sütunlarında ise kayıt içerisindeki verileri tutan veri öğeleri (alanları) bulunur.
- Öğrencinin kimlik numarası, ismi, genel not ortalaması, bölümü, bölüme giriş yılı, e-posta adresi ve telefon numarası gibi veriler sütunlarda tutulur.

KimlikNo	İsim	Bölüm	GNO	e-posta
04111067	Tolga Can	Bilgisayar	2,47	tolgacanfa calgarv.edu.ca
				toigacan(2>,hotmail .com
07233450	Serap Taner	İşletme		seraptanaer(2).calgarv.edu.ca
07230600	Mutlu Atanur	Bilgisayar		mutluatanur@calgarv.edu.ca
				mutlu@,gmail.com

İlişkisel Veri Tabanı ve İlişkisel Model

KimlikNo	İsim	Bölüm	GNO	e-posta
04111067	Tolga Can	Bilgisayar	2,47	tolgacanfa calgarv.edu.ca
				toigacan(2>,hotmail .com
07233450	Serap Taner	İşletme		seraptanaer(2).calgarv.edu.ca
07230600	Mutlu Atanur	Bilgisayar		mutluatanur@calgarv.edu.ca
				mutlu@,gmail.com

- **ÖĞRENCİ** tablosundaki *KimlikNo* öğesi belirleyici bir öğe olup **birincil anahtar (primary key)** olarak adlandırılır.
- VT'de bir varlık türü kümesi öğelerden oluşur ve aşağıdaki biçimde gösterilir:

  ÖĞRENCİ (KimlikNo. İsim, Bölüm, GNO, (E-posta))
- Bu örnekte birincil anahtar tek bir öğeden oluşmuştur. Ancak, bazı durumlarda birden fazla öğeden oluşan birincil anahtar kullanılabilir.
- Benzer biçimde, bir eğitim kuruluşunda görevli öğretim elemanları verileri **ÖĞRETİM_ELEMANI,** dersliklerle ilgili veriler **DERSLİK** ve bir dönemde açılan dersler **DERS** tablolarında tutulabilir.
- Bu tablodan bütünleşik olarak tutan bellek yapısı ise ***KURUM*** veritabanını oluşturacaktır. Bu yapı VT şeması olarak adlandırılır ve bu örnek için aşağıdaki biçimde gösterilebilir:

  KURUM {ÖĞRENCİ, ÖĞRETİMELEMANI, DERSLİK, DERS}

VT Tasarım Adımları

- Veritabanı tasarımı yazılım geliştirmede olduğu gibi bir sistem geliştirme hayat döngüsü süreci ile gerçekleştirilir. Bu süreç aşağıdaki adımlardan oluşur:
  - **Gereksinim analizi** - Problemin tanımı ve kullanıcı gereksiniminin belirlenmesi.
  - **Kavramsal tasarım** - VT'yi oluşturacak varlık türlerinin ve öğelerin belirlenmesi; tablolar arasındaki ilişkilerin belirlenmesi.
  - **Mantıksal tasarım** - Kavramsal tasarımdan ilişkisel modele geçiş.
  - **Sistemin uyarlanması** - Tabloların yaratılması, veri girişi ve uygulama programlarının ve ara yüzlerin yazılması.
  - **Sistemin test edilmesi ve bakımı.**

Gereksinim Analizi

- **Örnek problemin tanımı:** TBMM yeni dönemde seçilecek 550 milletvekili ve milletvekillerinin meclis içerisindeki faaliyetlerini saklamak üzere bir VT hazırlayacaktır.
  - Bu VT'de milletvekilleri ve bakmakla yükümlü oldukları aile bireyleri ile ilgili veriler ile milletvekillerinin yapacakları kanun teklifleri ve gensoru önergeleri hakkında veriler tutulacaktır.
  - Meclis Başkanlığı belli zaman aralıklarında, milletvekillerinin devamsızlıklarını ve faaliyetlerini raporlayıp kamuoyuna açıklayacaktır.
  - Bu problemin çözümü için, meclis faaliyetleri ve milletvekili verileri hakkında bir ön çalışma yapılması gerekmektedir.
- **Gereksinim analizi:**
  - Milletvekili sayısı 550 olarak verilmektedir.
  - Her milletvekilinin ortalama 4 kişinin bakımı ile yükümlü olduğu bilgisi bir önceki meclis istatistiklerinden edinilmiştir.
  - Yine bir önceki dönemde yapılan kanun teklifleri ve gensoru önerileri sayıları meclis başkanlığından alınmıştır.
  - Bu bilgiler VT büyüklüğü ve ek donanım yatırımları için gerekli olacaktır.
  - Eski dönem milletvekilleri ile görüşmeler yapılmış ve VT'de tutulması gereken veriler belirlenmiştir.
  - VT kullanıcıları milletvekilleri, cumhurbaşkanı, meclis başkanlığı ve meclis idari personelidir.
  - Milletvekilleri ile ilgili bazı verilerin önemli olduğu ve belirlenecek yetki seviyelerine göre kullanıcılara açılması gerektiği anlaşılmıştır.
  - Diğer bazı bulgular aşağıda sıralanmıştır:
    - Bir milletvekili ya bir parti üyesi ya da bağımsızdır.
    - Bir milletvekili görev süresince istediği kadar gensoru önergesi ve kanun teklifi verebilir.
    - Milletvekilinin seçim bölgesi bilgisi önemlidir.
    - Milletvekilleri komisyonlara üye olabilirler.
    - Milletvekillerinin bakmakla yükümlü oldukları kişilerin bilgileri tutulmalıdır.
- VT tasarımı bir yazılım şirketince düzenlenecek bir şartnameye göre ihale usulü ile yaptırılacağından, gereksinim analizinin bir bölümünde VT maliyeti de göz önüne alınmalı, donanım/yazılım, tasarım ve kodlama ve kullanıcı eğitimi gibi maliyetlerle ilgili bir ön çalışma yapılmalıdır.

Kavramsal Tasarım

- Gereksinim analizine göre VT'na konu olacak ve istenilen verileri saklayacak bir kavramsal model geliştirilmesi gereklidir.
- Bu modelde varlık türleri, barındıracağı varlıkların öğeleri, birincil anahtar ve varlıklar arasındaki ilişkiler belirlenir.
- Örnek problem için
  - ***MECLİS*** (VEKİL, YAKINLAR, TEKLİF, KOMİSYON) VT Şeması varlık türleri ve ayrıntılarıyla aşağıdaki şekilde belirlenmiştir:

    VEKİL (TCKNJsim, (TelefonNo), Bölge, DoğumTarihi, GörevBaşlangıç, GörevBitiş)

    YAKINLAR (İsim , YakınlıkDerecesi, DoğumTarihi)

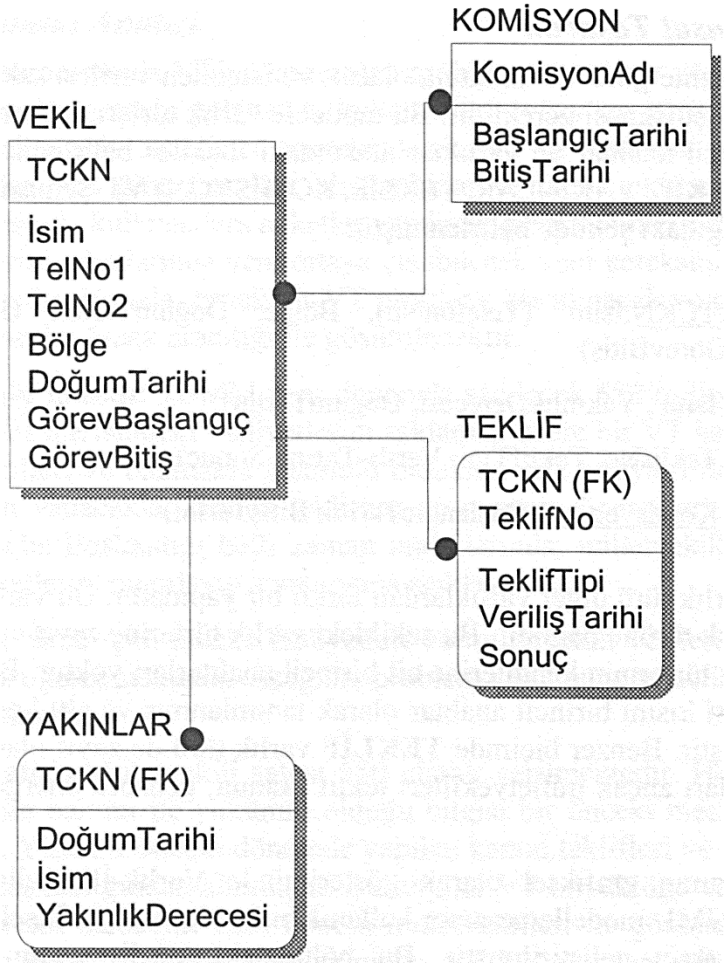
    TEKLİF (TekliiNo, TeklifTipi, VerilişTarihi, Sonuç)

    KOMİSYON (KomisyonAdı, BaşlangıçTarihi, BitişTarihi)

- YAKINLAR varlık türü diğer varlıklardan farklı bir yapıdadır. Bu varlık türünün varlığı VEKİL varlık türüne bağlıdır. Bu şekildeki varlık türlerine zayıf varlık türü adı verilir. Zayıf varlık türlerinin kendilerine ait birincil anahtarları yoktur. Bu yüzden geçici olarak, İsim öğesi kısmi birincil anahtar olarak tanımlanmış ve altı kesikli çizgi ile çizilerek belirtilmiştir. Benzer biçimde TEKLİF varlık türü de zayıf olarak tanımlanabilir. Teklif varlıkları ancak milletvekilleri teklif (kanun, gensoru önergesi) verildiğinde oluşacaktır.
- Kavramsal tasarımın grafiksel olarak gösteriminde Varlık-İlişki-diyagramları (ERD-Entity Relationship Diagram) ya da UML modelleme aracı kullanılabilir. Her iki grafiksel yöntem için de birçok yazılım aracı geliştirilmiştir.

Kavramsal Tasarım

- Yanda kavramsal tasarımın diyagramı verilmiştir.
- Bu diyagram üzerinde, tablolar arasındaki ilişkiler gereksinim analizine göre gösterilmiştir.
  - Örneğin bir milletvekilinin 0 ya da daha fazla teklif yapacağı gerçeği ***1***-*M* (bire-çok) ilişkisi ile belirtilmiş ve çok kısmı diyagramda içi dolu bir daire ile gösterilmiştir.
  - Diğer ilişki seçenekleri ise *M-M* (çoka-çok) ve ***1-1*** (bire-bir)'dir. M-M ilişkisi her iki ucu dolu daire ile gösterilmiştir. Buna göre bir vekilin birden çok komisyonda görev alabileceği ve aynı şekilde bir komisyonda birden çok vekilin üye olabileceği gerçeği yansıtılmaktadır.
  - Varlık türleri dikdörtgen, zayıf varlık türleri ise köşeleri yuvarlak dikdörtgen olarak gösterilmektedir. Zayıf varlık türlerinin bağlı olduğu varlık türleri belirleyen varlık türü olarak adlandırılmaktadır. Bu örnekte **VEKİL** belirleyen varlık türüdür
- TEKLİF ve YAKINLAR varlık türlerinde *TCKN* öğesi VEKİL varlık türündeki birincil anahtar olduğundan bu varlık türlerine yabancı anahtar (Foreign Key = FK) olarak tanımlanmaktadır.
  - Bunun nedeni TEKLİF ve YAKINLAR varlık türlerinin zayıf olmalarıdır. Sonuçta, YAKINLAR varlık türünün birincil anahtarı *{TCKN, İsim}* biçiminde bileşik öğe olarak ve TEKLİF varlık türünün birincil anahtarı *{TCKN, TeklifNo}* yine bileşik öğe olarak oluşmuştur



Mantıksal Tasarım

- Mantıksal tasarım aşamasında, bir önceki modeldeki kavramsal yapıdan ve ERD diyagramından yararlanılarak, ilişkisel modele geçiş işlemi gerçekleştirilir.
- Amaç kavramsal düzeyde açıklanan varlık türlerinden tabloların (ilişkilerin) elde edilmesidir. Bu geçiş oldukça standart bir yöntemle sağlanır.
- Yapılması gereken adımlara bakalım

Mantıksal Tasarım

1. İlk olarak yineleyen öğeler ele alınır.
   - Örnekte olduğu gibi bir milletvekilinin birden fazla telefon numarasının tutulması gerekmektedir.
   - Ancak, telefon numarası sayısı vekilden vekile değişmekte olduğundan tablodaki sütun sayısını önceden kestirmek mümkün değildir.
   - Çözüm **olarak, telefon numaralarının ayrı tutulduğu yeni bir tablo** yaratılır ve bu tabloya **VEKİL** varlık türünün birincil anahtarı taşınır.
   - Yeni tablonun birincil anahtarı *{TCKN, TelNo}* olarak seçilir. Bu adım sonucunda **VEKİL** varlık tipinden **VEKİLİ** ve **TELEFON** gibi iki tablo aşağıdaki gibi elde edilir:

     VEKİL1(TCKN, İsim, Bölge, DoğumTarihi, GörevBaşlangıç, GörevBitiş)

     TELNO (TCKN. TelNo)

Buna göre ilk tasarımda bir milletvekilinin en fazla iki telefonu saklanabilirken, bu yaklaşımla, sayı kısıtı ortadan kaldırılmıştır.

Mantıksal Tasarım

2. Her varlık için bir tablo yaratılmalı ve ERD diyagramındaki öğeler tablo öğeleri olarak seçilmelidir.

Birinci adımda **VEKİL** varlık tipi için **VEKİLİ** ve **TELEFON** tabloları yaratılmış durumdadır. Benzer biçimde KOMİSYON varlık tipi için yaratılan tablo şöyledir:

KOMİSYON (KomisyonAdı. BaşlangıçTarihi, BitişTarihi)

Mantıksal Tasarım

3. Zayıf varlıkların her biri için bir tablo yaratılmalı ve belirleyici varlık türlerinin birincil anahtarı bu tablolara eklenmelidir.

   Yeni tabloların birincil anahtarları, belirleyici varlık türlerinin birincil anahtarı ile zayıf varlık türlerinin kısmi anahtarının bileşimi olarak seçilmelidir. Buna göre yaratılan tablolar aşağıda verilmiştir:

   YAKINLAR (TCKN. İsim. YakınlıkDerecesi, DoğumTarihi)

   TEKLİF (TCKN. TeklifNo. TeklifTipi, VerilişTarihi, Sonuç)

Mantıksal Tasarım

4. tablolar arasındaki ilişkiler ele alınır.

- 1 -M ilişkilerinde, 1 tarafındaki birincil anahtar, yabancı anahtar olarak M tarafına götürülür.
  - Bu işlem zayıf varlık türleri olan **TEKLİF** ve **YAKINLAR** için doğrudan yerine getirilmiştir.
- M-M ilişki türünün VT'lerde fiziksel olarak gösterimi uygun değildir. Bunun yerine, M-M ilişki iki tane 1-M ilişki haline getirilir.
  - Bunu sağlamak için M-M ilişkiye giren iki varlık tipi arasında yeni bir tablo yaratılır ve eski varlık türlerinden bu yeni tabloya 1-M ilişki sağlanır.
  - **VEKİLİ** ile **KOMİSYON** varlık türleri arasında M-M ilişki mevcuttur. Buna göre, yeni tablolar ve bağlantılar aşağıdaki gibi oluşur:

VEKİLI_KOMİSYON (TCKN. KomisvonAdı)

Mantıksal Tasarım

5. Elde edilen son VT şeması aşağıdaki gibidir:

   VEKİLK<u>TCKN</u>. İsim, Bölge, DoğumTarihi, GörevBaşlangıç, GörevBitiş)

   TELEFON (<u>TCKN. TelNo</u>)

   KOMİSYON (<u>KomisvonAdı</u>. BaşlangıçTarihi, BitişTarihi)

   VEKİLI_KOMİSYON (<u>TCKN. KomisvonAdı</u>)

   YAKINLAR (<u>TCKN, İsim</u>, YakınlıkDerecesi, DoğumTarihi)

   TEKLİF (<u>TCKN. TeklifNo</u>. TeklifTipi, VerilişTarihi, Sonuç)

Mantıksal Tasarım

6. Son adım normalizasyon adımıdır. Normalizasyon VT tablolarının sistematik olarak ve bir amaca yönelik olarak alt tablolarla bölünme işlemidir.

Normalizasyon Örnek

Tablonun İlk Hali

Adı Soyadı	tel	ek bilgi	ders1	eğitmen1	ders2	eğitmen2	ders3	eğitmen3
Ali A	123	ceza almış	BBG	a	NTP	b	PRG1	c
Veli B	321		BBG	a	PRG1	c		
Ayşe C	213		VT	c				

Normalizasyon Örnek

- 1. Normal Form
  - Bir kayıtla ilgili tekrar eden sütunları yok et

ogrenci					
adı	soyadı	tel	ek bilgi	ders	eğitmen
ali	a	123	ceza almış	BBG	a
ali	a	123	ceza almış	NTP	b
ali	a	123	ceza almış	PRG1	c
veli	b	321		BBG	a
veli	b	321		PRG1	c
ayşe	c	2133		VT	d

Normalizasyon Örnek

- 2. Normal Form
  - Tekrar eden satırları yok edecek tablo bölmesi yap

ogrenci			
OGR_ID	adı	souadı	tel
1	ali	a	123
2	veli	b	321
3	ayşe	c	213

ogrenci_dersleri		
OGR_ID	ders	eğitmen
1	BBG	a
1	NTP	b
1	PRG1	c
2	BBG	a
2	PRG1	c
3	VT	d

Normalizasyon Örnek

- 3. Normal Form
  - Veri tekrarını azaltmak amacı ile tanım tabloları oluştur

ogrenci				
OGR_ID	adı	soyadı	ek bilgi	tel
1	ali	a	ceza almış	123
2	veli	b		321
3	ayşe	c		213

ogrenci_dersleri	
OGR_ID	DERS_ID
1	1
1	2
1	3
2	1
2	2
3	4

dersler		
DERS_ID	ders	eğitmen
1	BBG	a
1	NTP	b
1	PRG1	c
1	VT	d

Normalizasyon Örnek

- 4. Normal Form
  - Boş alanları yok edecek zayıf bağlantılı tablolar oluştur

ogrenci			
OGR_ID	adı	soyadı	tel
1	ali	a	123
2	veli	b	321
3	ayşe	c	213

ek_bilgi	
OGR_ID	bilgi
1	ceza almış

ogrenci_dersleri	
OGR_ID	DERS_ID
1	1
1	2
1	3
2	1
2	2
3	4

dersler		
DERS_ID	ders	eğitmen
1	BBG	a
1	NTP	b
1	PRG1	c
1	VT	d

Sistemin Uyarlanması

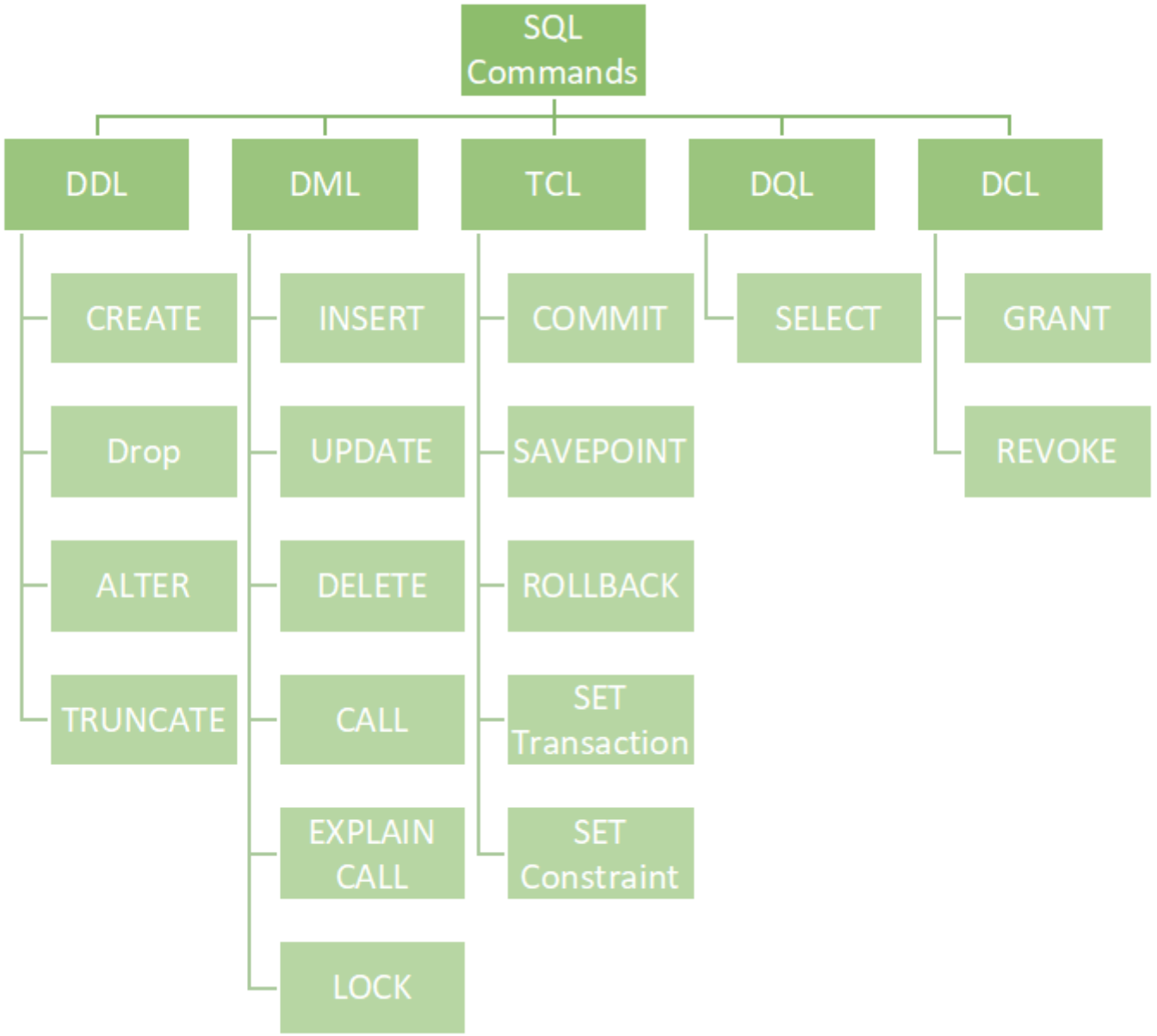
- Bu adımda mantıksal tasarım sonucunda elde edilen tablolar kullanılan VTYS'de fiziksel olarak yaratılır ve tablolara kayıt girişi yapılır. Yaratılan VT'nın en etkin biçimde kullanımı için bazı ayarlamalar yine bu adımda gerçekleştirilir. Fonksiyonel bazı işlemler ve hareketler için gerekli programlar ve kullanıcı arayüzleri yine bu adımda hazırlanır.

Sistemin Test Edilmesi ve Bakımı

- VT kullanıma açılmadan önce, testlerden geçirilerek doğruluğu ve geçerliliği sağlanır. Gerekli kullanıcı dokümanları bu aşamada sonlandırılır ve VT sistemi tamamlanmış olur

SQL

- **Bir** VT**'nin yaratılması,** yeni kayıt ekleme, kaydın saklanması ve değiştirilmesi, tabloya kısıtlar eklenmesi, var olan kısıtların değiştirilmesi gibi işlemler "Yapısal Sorgulama Dili" olarak bilinen SQL (Structured Query Language) ile gerçekleştirilir. SQL komutları VTYS'ne bağlı olarak ya bir SQL editörü aracılığı ile ya da bir başka programlama dili içerisinden gömülü olarak çalıştırılır. SQL komutları işlevlerine göre gruplara ayrılırlar:
  - **Veri Tanım Dili (DDL=Data Definition Language)** komutları VT tablolarının ve diğer nesnelerin fiziksel yapısını yaratmak ya da değiştirmek için kullanılan komutlardır.
  - **Veri Manipülasyon Dili (DML=Data Manipulation Language)** komutları kayıtların tabloya eklenmesi, tablodan çıkartılması ya da var olan kayıtlar üzerinde değişiklik yapmaya yarayan komutlardır.
  - **Veri Denetim Dili (DCL=Data Control Language)** komutları ise VT üzerindeki yetkilendirmeleri denetleyen komutlardır.
  - **Hareket Denetim Dili (TCL=Transaction Control Language)** komutları tablolar üzerindeki değişikliklerin ve hareketlerin VT'de kesin olarak saklanmasında ya da yapılan işlemden vazgeçildiğinde VT'yi bir önceki durumuna getirmede kullanılır.
  - **Veri Sorgulama Dili (DQL=Data Query Language)** komutu veri sorgulamada kullanılır


SQL Commands
DDL
DML
TCL
DQL
DCL
CREATE
Drop
ALTER
TRUNCATE
INSERT
UPDATE
DELETE
CALL
EXPLAIN CALL
LOCK
COMMIT
SAVEPOINT
ROLLBACK
SET Transaction
SET Constraint
SELECT
GRANT
REVOKE

Bazı SQL Komutları

Veri Tanım Dili (Data Definition Language) Komuttan	
Tablo Yapısını Yaratmak	
Komut	**Tanım**
CREATE TABLE	VT içerisinde yeni bir tablo yaratır.
CREATE TABLE ... AS	Var olan bir tablonun fiziksel yapısını ve kayıtlarının kullanarak yeni bir tablo yaratır.
Tablonun Yapısını Değiştirmek	
ALTER TABLE ... ADD	Tabloya yeni bir sütun ekler.
ALTER TABLE ... MODIFY	Var olan sütunun özelliklerini (boyut, veri tipi veya kabul edilmiş değer) değiştirir.
ALTER TABLE ... DROP COLUMN	Tablodan bir sütunu siler.
TRUNCATE TABLE ...	Tablonun yalnızca kayıtlarını siler.
DROP TABLE ...	Tabloyu VT'den kaldırır.
Veri Manipülasyon Dili (Data Manipulation Language) Komutları	
INSERT INTO...VALUES 0	Tabloya bir kayıt ekler.
DELETE FROM ...	Tablodaki kayıtlan siler.
UPDATE ... SET ...	Tablodaki hücre değerlerini değiştirir.
Hareket Denetim Dili (Transaction Control Language) Komutları	
COMMIT	Değişiklikleri tabloda kesin olarak saklar.
ROLLBACK	Değişiklikleri geri alır.
Veri Denetim Dili (Data Control Language) Komutları	
GRANT ... ON ... TO ...	Kullanıcılara tablo ve diğer nesneler üzerinde yetki verir.
REVOKE ... ON ... FROM ...	Verilen yetkileri geri alır
j **Veri Sorgulama Dili (Data Query Language) Komutları**	
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ... ORDER BY ...	Temel sorgulama konutudur. İstenilen kısıtlara uygun kayıtlan seçer ve gösterir.

Tablo Yaratmak

- Tablo yaratmak için kullanılan komut **CREATE TABLE**'dır. Komutun genel yapısı aşağıda verilmiştir. Köşeli parantez içindekiler zorunlu olmayan bölümlerdir

```
CREATE TABLE <tablo adı>

   ( sütun adı veri tipi [DEFAULT Kabul edilen değer],
     sütun adı veri tipi [DEFAULT kabul edilen değer],
     …/
     CONSTRAINT kısıt (sütun),
     CONSTRAINT kısıt   (sütun),
   ) ;
```

- Buradaki **DEFAULT** sözcüğü ile sütunlara ilk değer ataması yapılabilir. Veri türü öğenin alacağı değerler kümesine göre *VARCHAR2(n)* (En çok *n* karakterli dinamik dizgi), *DATE* (tarih değeri), *CHAR() (n* karakterli statik dizgi) ya da *NUMBER(m,n) {m* uzunlukta ve *n* ondalık basamaklı sayı) olabilir. Kısıt ise aşağıdakilerden birisidir:
  - **PRIMARY KEY** (Birincil anahtarı tanımlar.)
  - **FOREIGN KEY** (Yabancı anahtarı tanımlar.)
  - **UNIQUE** (İkincil anahtar için kullanılır.)
  - **CHECK** (Sütun değerinin denetimi için kullanılır.)
  - **NOT NULL** (Sütun değerinin boş olamayacağını gösterir.)
- Yukarıdaki **CREATE TABLE** komutunda kısıtlar tablo kısıtı olarak gösterilmiştir. Komutun bir başka kullanım şeklinde bazı kısıtlar sütun kısıtı olarak sütun tanımlanırken de verilebilir. Örnek tablolar yaratılırken her iki kullanımdan da yararlanılacaktır

Tablo Yaratmak

- Aşağıda örnek tabloları yaratmak üzere **CREATE TABLE** komutları verilmiştir

```
CREATE TABLE      VEKİLİ
  ( TCKN          VARCHAR2(11)  PRIMARY KEY,
    isim          VARCHAR2(20)  NOT NULL,
    Bölge         VARCHAR2(10)
                  ,
    DoğumTarihi   DATE,
    GorevBaslangi DATE DEFAULT  '28-TEM-2007',
    c
    GörevBitiş    DATE);
```

```
CREATE TABLE      TELEFON
  ( TCKN          VARCHAR2(11),
    TelNo         CHAR(12),
    CONSTRAINT    telefon_pk PRIMARY KEY(TCKN, TelNo),
    CONSTRAINT    tckn fkl FOREIGN KEY(TCKN) REFERENCES VE-
                  KİLİ (TCKN) ON DELETE CASCADE);
```

```
CREATE TABLE      KOMİSYON
( KomisyonAdı     VARCHAR2(11) PRIMARY KEY,
  BaşlangıçTari   DATE,
  hi
  BitişTarihi     DATE);
```

```
CREATE TABLE      VEKİLİ KOMİSYON
  ( TCKN          VARCHAR2(11),
    KomisyonAdı   VARCHAR2(20),
    CONSTRAINT    vekill_komisyon_pk      PRIMARY KEY(TCKN,
                  KomisyonAdı),
    CONSTRAINT    tckn fk2 FOREIGN KEY(TCKN) REFERENCES VEKİLİ
                  (TCKN) ON DELETE CASCADE),
    CONSTRAINT    komisyon adi fk FOREIGN KEY(KomisyonAdi)
                  REFERENCES   KOMISYON(Komisyonadi)   ON DELETE
                  CASCADE);
```

```
CREATE TABLE      YAKINLAR
   ( TCKN         VARCHAR2(11) REFERENCES VEKİLİ (TCKN),
     isim         VARCHAR2(20),
    YakınlıkDerece VARCHAR2(10),
    si
     DoğumTarihi  DATE,
     CONSTRAINT   yakinlar_pk PRIMARY KEY(TCKN, isim));
```

```
CREATE TABLE      TEKLİF
  ( TCKN          VARCHAR2(11) REFERENCES VEKİLİ (TCKN),
    TeklifNo      NUMBER(4),
    TeklifTipi    VARCHAR2(4),
    VerilisTarihi DATE DEFAULT sysdate,
    Sonuç         VARCHAR2(20),
    CONSTRAINT    teklif_pk PRIMARY KEY (TCKN, TeklifNo),
    CONSTRAINT    tekliftipi ck CHECK (TeklifTipi
                  IN('KANUN','GENSORU')));
```

Tablolara Veri Girişi

- VT'nin fiziksel yapısının yaratılmasından sonraki işlem tablolara veri girişidir. Tabloya veri girişi **İNSERT** komutu ile gerçekleştirilir, **İNSERT** komutunun biçimi aşağıdaki gibidir:

  İNSERT İNTO tabloadı (sütunadıl, sütunadı2, **...) VALUES** (öğe 1 değeri, öğe2değeri ,...,öğeKdeğeri);

örnek:

İNSERT İNTO VEKİLİ **VALUES** (11012345670, Ahmet Akan', 'ISTANBUL2', '10-OCA- 1970', NULL, DEFAULT);

Tablolara Veri Girişi

TELEFON

TCKN	TELNO
10544765309	536-2899191
10544765309	903124412525
10544765309	903126504242
11012345670	532-5101010
11012345670	903122405050
20238090001	903122871111
58628012100	903125405672
71034556000	542-7608686

- Bu tabloda birincil anahtar iki öğeden oluşmakta ve bu sayede bir milletvekiline ait birden fazla telefon numarasının kaydı mümkün olabilmektedir.

VEKIL1_KOMISYON

TCKN	KOMİSYONADI
10544765309	Af
11012345670	Af
11012345670	Bilişim
20238090001	Adalet
20238090001	Anayasa
71034556000	Anayasa

- Benzer biçimde, aşağıdaki tabloda, milletvekilleri ve görev aldıkları komisyonlar gösterilmektedir

KOMİSYON

KOMİSYONADI	BAŞLANGIÇT	BITIŞTARIH
Anayasa	01/08/2007	
Adalet	01/08/2007	
Bilişim	11/09/2007	
Af	11/09/2007	

YAKINLAR

TCKN	ISIM	YAKINLIKDE	DOĞUMTARIH
11012345670	Aysel Akan	Kızı	01/09/2000
11012345670	Murat Akan	Oğlu	23/04/2002
11012345670	Serap Akan	Eşi	14/07/1973
10544765309	Murat Durak	Eşi	29/06/1970
71034556000	Serap Tolun	Eşi	21/10/1958

- YAKINLAR tablosunda, milletvekillerin bakmakla yükümlü olduğu yakınlarının bilgileri tutulmaktadır

TEKLİF

TCKN	TEKLİFNO	VERIUSTAR	SONUÇ	TEKLIFT
20238090001	1001	01/09/2007	RED	GENSORU
20238090001	1002	21/04/2008	KABUL	KANUN
58628012100	2007	21/04/2008	KABUL	GENSORU
71034556000	2103	18/03/2008	KABUL	KANUN

- Aşağıdaki tabloda ise, milletvekillerinin dönem içerisinde meclise sundukları kanun teklifli ve gensoru önergeleri bilgileri tutulmaktadır

Sorgulama

- Veritabanlarındaki verilerden istatistikler çıkartılması, raporlar elde edilmesi, bazı kıstaslara uyan kayıtların elde edilmesi ve kayıtlan sıralama gibi amaçlarla, VT sorgulaması yapılır.
- Örneğin, bir kütüphanenin veritabanından, belli bir yazara ait kitapların bilgilerine erişmek, nüfus veritabanından bir vatandaşın bilgilerine erişmek, bir kişiye ait banka hesabındaki bakiyeyi öğrenmek gibi işlemler sorgulama komutu ile yerine getirilir.
- Çok sayıda kayıt arasından aranılan kıstaslara uyan kayıtlara hemen erişmek için kullanılan sorgulama komutu **SELECT**'tir.
- Bu komut ile bir ya da birden fazla tablodan sorgulama yapılabilir. Komutun genel yapısı aşağıdaki gibidir:

SELECT öğe 1, öğe2,... **FROM** tablol,tablo2,... **WHERE** kısıt
GROUP BY öğel, öğe2,... **HAVING** grupkısıtı **ORDER BY** öğel,öğe2,...;

- **SELECT** sözcüğünün yanına hakkında bilgi istenilen öğeler veya hesaplanacak terimler yazılır. Eğer tüm öğelerin değerleri isteniyorsa öğe isimlerini tek tek yazmak yerine yıldız (*) simgesi kullanılabilir.
- **FROM** sözcüğünün yanına sorgulamada kullanılacak tabloların isimleri yerleştirilir.
- Kayıtların tamamı seçilecekse **WHERE** kısıtı kullanılmaz. Ancak, çoğu kez bazı kıstaslara uyan kayıtların seçilmesi gerekeceğinden bu bölüme sonucu doğru ya da yanlış olan bir mantıksal ifade konulur.
- Mantıksal ifade **AND, OR** ve **NOT** mantıksal işleçleri ile birleşik ifade biçiminde de kullanılabilir. Bu mantıksal ifadenin doğru sonuç verdiği kayıtlar sonuç tablosu olarak gösterilir.
- **ORDER BY** ile sonuçların sıralı olarak gösterimi sağlanır. Kabul edilmiş sıra küçükten büyüğe doğrudur. Tersi için öğenin yanına **DESC** sözcüğünün konulması gerekir. Artan sıra için istenirse öğenin yanında **ASC** sözcüğü kullanılabilir

Sorgulama

- **Sorgu 1.** 1970'den önce doğmuş milletvekillerinin isimlerini ve partilerini elde ediniz. **SELECT** komutu ve VT'den alınan sonuç aşağıdaki gibidir

```
SELECT ISIM, PARTIADI, DOĞUMTARIHI
FROM VEKIL1
WHERE  DOĞUMTARIHI<'1-OCA-1970';
```

ISIM	PARTIADI	DOĞUMTARIH
Tahsin Mert	EP	18/03/1958
Orhan Tolun	YP	23/08/1955

- **Sorgu 2.** Yukarıdaki sorguyu değiştirerek, elde edilen tablonun doğum tarihlerine göre küçükten büyüğe doğru sıralı olmasını sağlayınız. Bunun için komutun sonuna **ORDER BY** sözcüğünü eklemek gerekmektedir

```
SELECT ISIM, PARTIADI, DOĞUMTARIHI
FROM VEKIL1
WHERE  DOĞUMTARIHI < '1-OCA-1970'
ORDER BY DOĞUMTARIHI;
```

ISIM	PARTIADI	DOĞUMTARIH
Orhan Tolun	YP	23/08/1955
Tahsin Mert	EP	18/03/1958

- **Sorgu 3.** ADANA ve ARTVİN milletvekillerinin isimlerini ve bölgelerini gösteriniz

```
SELECT ISIM, BOLGE
FROM VEKIL1
WHERE BOLGE='ADANA' OR BOLGE='ARTVIN';
```

ISIM	BOLGE
Tahsin Mert	ADANA
Orhan Tolun	ARTVIN

Sorgulama

- **Sorgu 4.** GENSORU önergesi veren milletvekillerinin isimlerini ve verdikleri önergelerin sonucunu gösteriniz. Bu sorgulama hem milletvekili bilgilerini hem de teklif bilgilerini istemektedir. Yani hem *VEKİLİ* hem de *TEKLİF* tablosu birlikte kullanılmalıdır. Bu işlem birleştirme (join) işlemi olarak bilinir. İki tablonun birleştirilmesinde, iki tabloya ortak olan öğe kullanılır. Bu öğelerden birisi genelde birleştirilen tabloların birinde yabancı anahtar diğerinde ise birincil anahtardır. Ortak olan öğelerin isimlerinin aynı olması gerekmez.. Bu kısıta birleştirme kısıtı adı verilmektedir. Aşağıda bu işi yapmak üzere iki birleştirme komutu verilmiştir. Sonuçta, gensoru önergesi veren milletvekillerinin isimleri listelenmiştir

```
SELECT ISIM, SONUC
FROM VEKIL1, TEKLIF
WHERE  VEKIL1.TCKN = TEKLIF.TCKN
AND TEKLIFTIPI='GENSORU';
```

veya

```
SELECT ISIM, SONUC
FROM VEKIL1 JOIN TEKLIF
USING (TCKN)
WHERE TEKLIFTIPI='GENSORU';
```

ISIM	SONUC
Tahsin Mert	RED
Ahmet Nurgül	KABUL
Orhan Tolun	KABUL

- **Sorgu** 5. Komisyonlarda çalışan milletvekili sayılarını veriniz. Bu sayılan elde etmek için *VEKİLİ KOMİSYON* tablosunun kullanılması gerekmektedir. SQL dilinde, matematiksel, metinsel ve tarih işlemlerini destekleyen birçok fonksiyon bulunmaktadır. **COUNT** bu fonksiyonlardan birisi olup, belirlenen kıstası sağlayan kayıtlan saymada kullanılmıştır. **GROUP BY** sözcüğü tabloyu komisyon adlarına göre alt kümelere ayırmakta ve her alt küme için sayma işlemi gerçekleştirilmektedir

```
SELECT KOMISYONADI, COUNT(*) MSAYISI
FROM VEKIL1_KOMISYON
GROUP BY KOMISYONADI;
```

KOMISYONADI	MSAYISI
Adalet	1
Af	2
Anayasa	2
Bilişim	1

Sorgulama

- **Sorgu 6.** Milletvekillerinin ve varsa eşlerinin isimlerini yazdırınız

```
SELECT VEKIL1.ISIM VEKIL, YAKINLAR.ISIM ESI
FROM VEKIL1, YAKINLAR
WHERE VEKIL1.TCKN=YAKINLAR.TCKN
AND YAKINLAR.YAKINLIKDERECESI='Eşi';
```

VEKIL	ESI
Ahmet Akan	Serap Akan
Alev Durak	Murat Durak
Orhan Tolun	Serap Tolun

- **Sorgu 7.** Milletvekillerinin telefon sayılarını TCKN ile birlikte veriniz

```
select tckn, count(tckn)
from telefon
group by tckn;
```

TCKN	COUNT(TCKN)
10544765309	3
11012345670	2
20238090001	1
58628012100	1
71034556000	1