

Veritabanlarına ve SQL'e Giriş

Devrim GÜNDÜZ

Teknoloji Destek Merkezi -- www.tdmsoft.com

devrim@gunduz.org

<http://seminer.linux.org.tr>

<http://www.gunduz.org>

Giriş

- Bu seminerde, aşağıdaki konular anlatılacaktır:
 - Veritabanı tanımı
 - Veritabanı türleri
 - Veritabanlarının kullanım alanları
 - İlişkisel (Relational) veritabanlarının açıklanması
 - SQL nedir?
 - SQL Örnekleri
 - E-posta listeleri

Veritabanı Nedir?

www.m-w.com :

“a usually large collection of data organized especially for rapid search and retrieval (as by a computer.”

Veritabanı Nedir?

- Belirli bir tarzda organize edilmiş bilgi “koleksiyon”udur.
- En az bir tablodan oluşmak zorundadır.
- Veritabanı programını oluşturan tablolar ise veri alanlarından oluşur (data field).

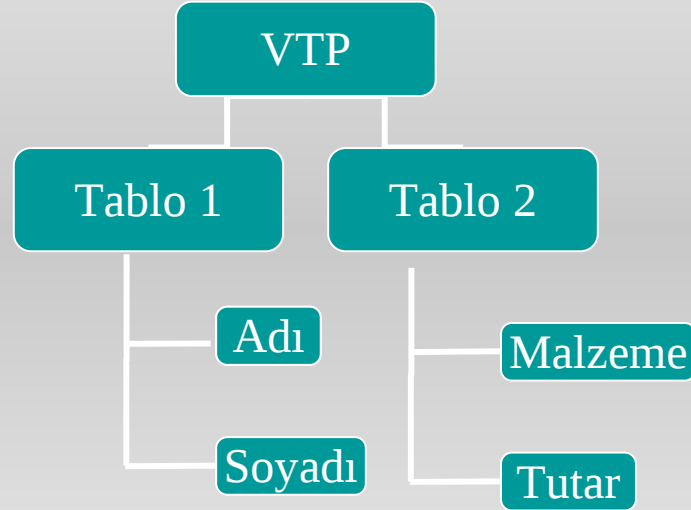
Veritabanı Nedir?

- Kitaplıklar, uygulamalar ve yardımcı programların birleşmesinden oluşur.
- Verilerin saklanması ve yönetilmesi ile ilgili konulardaki ayrıntılardan veritabanı yöneticilerini kurtarır.
- Kayıtların güncellenmesi ve kayıtlar üzerinde araştırma yapılması da mümkündür.

Veritabanı Tipleri

- Hiyerarşik Veritabanı
- İlişkisel Veritabanı (Relational Type)
- Nesnesel Veritabanı

Hiyerarşik Veritabanı



İlişkisel Veritabanı

- 1970: “A Relational Model of Data for Large Shared Data Banks”
- E. F. Codd
 - Relation mantığı
 - Tablolar -> Gerçek dünyadaki nesnelere

İlişkisel Veritabanı

- Birden fazla tablodan oluşabilir.
- Birbirlerinin yerine kullanılabilir.
- Tablolar satır ve sütundan oluşur.

İlişkisel Model

- - Bilgi Kuralı: bütün veriler tablolarda tutulmalı
- Erişim Garantisi: Tablolardaki her kayıda bir şekilde ulaşılabilmeli
- Sistematik boş (null) değer desteği: bütün değerlerden farklı bir boş değer işlenebilmeli
- Aktif ilişkisel katalog: Veritabanı tanımını veritabanı dili kullanılarak sorgulanabilmeli.
 - » Koray Toksöz'e teşekkürler.

İlişkisel Model

- Ayrıntılı veri dili: En az bir dil tanımlı olmalı
 - View güncelleme kuralı: Güncellenebilen tüm viewler sistem tarafından güncellenebilmeli
 - Küme düzeyi kayıt girme, silme ve güncelleme: Sadece seçmek yetmez
 - Fiziksel Veri Bağımsızlığı: Uygulama programları, fiziksel erişim değişikliklerinden etkilenmemeli.
 - » Koray Toksöz'e teşekkürler.

İlişkisel Model

- Mantıksal veri bağımsızlığı: Tablo yapılarındaki değişikliklerden uygulama programları etkilenmemeli.
 - Bütünlük Bağımsızlığı: Bütünlük kuralları tanımlanabilmeli ve atlatılamamalı
 - Dağıtım Bağımsızlığı: Uygulamalar verinin dağılımından etkilenmemeli
 - “Yıkılmama”: Bütünlük kuralları atlatılamamalı.
 - » Koray Toksöz'e teşekkürler.

İlişkisel Veritabanı

- Kullanıcının programı kullanırken ona **sık sık soracağı soruların** neler olacağı tespit edilir.
- “Gerçekleştirilecek olan veritabanı programından beklenen neler ve bu veritabanında hangi bilgilerin olması gerekli?” sorusunun yanıtı bulunur.

İlişkisel Veritabanı

- Tablolardaki kayıtlar matematiksel açıdan tuple olarak tanımlanırlar.
- Bir tuple, tanımlanmış bir veri tipi olan bileşenlerden oluşan sıralı grup olarak tanımlanır.
- Tüm tuplelar aynı sayıda ve tipte bileşenlerden oluşur.
 - {“10”, “Veritabanlarına Giriş”, “2002-06-12”}
 - {“11”, “PostgreSQL Veritabanı Sunucusu”, “2002-06-26”}

İlişkisel Veritabanı

- Örnekteki her bir tuple da 3 bileşen bulunmaktadır:
 - Ankara'daki 2002 yılındaki kaçınıcı seminer olduğu (integer)
 - Seminerin konusu (char)
 - Seminerin tarihi (timestamp)
- İlişkisel veritabanlarında bu “küme” ya da tabloya eklenen tüm kayıtlar aynı yapıda olmalıdırlar.

İlişkisel Veritabanı

- { “Veritabanlarına Giriş”, “2002-06-12”}
 - *eksik bileşen*
- {“10”, “Veritabanlarına Giriş”, “2002-06-12” , “Devrim GÜNDÜZ”}
 - *fazla bileşen*
- {“2002-06-12”, “Veritabanlarına Giriş”, “10”}
 - *yanlış bileşen tipleri (yanlış sırada)*

İlişkisel Veritabanı

- tuple lardan oluşan bir tabloda aynı veriler bulunmaz.(No duplicate record).
 - İlişkisel veritabanlarındaki herhangi bir tabloda birbiriyle tamamen aynı iki kayıt (row or record) bulunamaz.
 - Gereksiz sınırlama?
- Sorun : İki kez aynı ürünün siparişi
- Çözüm : Tabloya eklenecek bir fazla bileşen

İlişkisel Veritabanı

- Bir kayıttaki her bir bileşen “atomik”, yani bir veri olmalıdır
 - Başka bir kayıt ya da diğer bileşenlerin listesi olamaz.
- Tablodaki bileşenlerin veri tipleri de üsttekilerle ve dolayısıyla tablo tanımlarındakiyle aynı olmalıdır.
 - Veritabanı tarafından desteklenen veri tiplerinden biri olmalıdır.

İlişkisel Veritabanı - Anahtarlar

- **key:** Birbiriyle eş kayıtları ayırmak için kullanılan bileşenlerdir.
- **primary key :** Tablodaki bir kaydı diğer tüm kayıtlardan ayırmak için kullanılan bileşendir.
 - “unique” yapar.
 - Tüm ilişkisel veritabanlarında her bir tablo ya da relationda mutlaka primary key olmalıdır.

Tablolar

- Aynı konu ile ilgili olan bilgiler belirlenmelidir.
- Index olarak kullanılacak alanlar, zaman icinde deęiştirilebilecek şekilde belirlenir.
- Olası olan en yüksek seviyede yapısal bir şekilde tabloların oluşturulması sağlanmalıdır.

Veri alanları

- Aynı konu ile ilgili alanların kendi tablolarında toplanması sağlanmalı
- Veri tekrarı olmamalı
- Gereksiz alanlar kullanılmamalı
- Alanlar basite indirgenmelidir.

Veritabanından beklenenler

- Her düzeyde rapor alınabilme özelliđi
- Doğru ve hızlı sonuç verebilme özelliđi
- Sorulabilecek bütün sorulara yanıt verebilecek sorgulama dili
- Bilgilerin ve sonuçların tutarlılıđı

Neden Veritabanı?

- Gerçekten veritabanına gereksinmeniz var mı?
- Veritabanları, verilerin saklanması ve yönetilmesi için kullanılmalıdır.
- Küçük bilgiler için metin dosyaları yeterli olabilir.
- Amacınızın iyi belirlenmesi gerekir.

Neden Veritabanı?

- Veri sadece bir konuyu içeren bir listenin içinde mi?
- Sorun karmaşık mı?
- İstatiksel bir analiz mi yapmak istiyorsunuz?

Neden Veritabanı?

- Bir yönetim mi yapacaksınız?
- Metinsel veritabanları
- Kullanım kolaylığı

Neden Veritabanı?

- Bilimsel formüllere gereksinmeniz olacak mı?
- Veriyi paylaşma gereksinmeniz olacak mı?
- Veriyi webde sunacak mısınız?

Veritabanı Çeşitleri

- Öncelikle ne yapılacağına karar verilmelidir:
 - 1 Bu veritabanı ile neler yapacaksınız? Küçük bir şirket çalışanlarının özel bilgileri mi tutulacak, yoksa büyük bir şirketin binlerce müşterilerinin bilgileri mi?
 - 2 Sitenizi günde kaç kişi ziyaret edecek?

Veritabanı Çeşitleri

- 1 Aynı anda kaç işlem yapılacak?
1. Güvenlik ne ölçüde olacak?
2. Verilerinizin güvenliği ne ölçüde olacak?

SQL Nedir?

- SQL: Structured Query Language
- Veritabanı dilidir.
- Veri eklerken, silerken, güncellerken veya sorgularken kullanılır.
- ANSI ve ISO standardıdır.
- Select, Delete, Update, Insert

Veritabanı Yaratma

- `CREATE DATABASE veritabanı_adı;`
-
- `veritabanı_adı` adıyla boş bir veritabanı oluşturur.
-
- Örnek:
 - `Test1=# CREATE DATABASE lkd;`
 - `CREATE DATABASE`

Veritabanı kaldırma (silme)

- DROP DATABASE
 - test1=# DROP DATABASE lkd;
 - DROP DATABASE
-

Tablo yaratma

- Tablo yaratmak için, `CREATE TABLE` kullanılır.
- Tablo yaratırken, tablonun içindeki kolonlar da yaratılır. Yaratılan kolonların veri tipleri, veritabanı sunucusunda tanımlı veritipleri olmalıdır.
- Bu alanlar, daha sonra değiştirilebilir.

Tablo yaratma

- PostgreSQL:
CREATE TABLE bolumler
(
 sira_no SERIAL UNIQUE,
 bolum_no int2 PRIMARY KEY,
 bolum_adi varchar(30)
);

Tablo yaratma

- NOTICE: CREATE TABLE will create implicit sequence 'bolumler_sira_no_seq' for SERIAL column 'bolumler.sira_no'
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index 'bolumler_pkey' for table 'bolumler'
- NOTICE: CREATE TABLE / UNIQUE will create implicit index 'bolumler_sira_no_key' for table 'bolumler'
- CREATE TABLE

Tablo yaratma

- MySQL:
CREATE TABLE 'bolumler'
(
 sira_no auto_increment,
 bolum_no smallint PRIMARY KEY,
 bolum_adi varchar(30)
);

Tabloya veri girme

- `INSERT INTO <tablo> VALUES (<değerler>)`
<tablo> ile verilen tabloya <değerler> i girer.
- `INSERT INTO <tablo> <alanlar>VALUES (<değerler>)`
<tablo> ile verilen tablonun <alanlar> ile belirtilen alanlarına <değerler> i girer.

Tabloya veri girme

- INSERT INTO bolumler
VALUES (1, '12', 'Matematik');
- INSERT INTO bolumler (bolum_no,
bolum_adi)
VALUES ('13', 'Matematik');
- sıra_no alanına, daha önceden yaratılmış sequence
içindeki değer girilir, ardından değer 1 arttırılır.

Veri Sorgulama

- `SELECT <kolonlar> from <tablo>;`
- `<tablo>` adındaki tablodan belirtilen `<kolonlar>` ı seçer.
- Kolonlar yerine, `*` konursa, o tablodaki tüm alanlar seçilir.

Veri Sorgulama

- `SELECT * FROM bolumler;`
bolumler tablosundaki tüm kolonları seçer:
`test=# SELECT * FROM bolumler ;`
- `sira_no | bolum_no | bolum_adi`
- -----+-----+-----
- `1 | 12 | Matematik`
- `2 | 13 | Fizik Mühendisliği`
- `3 | 14 | Bilgisayar Programcılığı`
- (3 rows)

Veri Sorgulama

- test=# SELECT bolum_no, bolum_adi FROM bolumler ;
bolum_no | bolum_adi

```
-----+-----  
      12 | Matematik  
      13 | Fizik Mühendisliği  
      14 | Bilgisayar Programcılığı
```

(3 rows)

test=#

Tekrarsız veri sorgulama

- `SELECT DISTINCT <kolon> FROM <tablo>;`
<tablo> tablosundaki <kolon> kolonunun tekil değerlerini seçer.
- `test=# CREATE TABLE dersler2 (
test(# ders_no int2
test(# ders_adi varchar(40)
test(#);`
- `CREATE TABLE`

Tekrarsız veri sorgulama

- test=# SELECT * from dersler ;
ders_no | ders_adi

-----+-----

1 | Fizik

1 | Fizik

3 | Matematik

(3 rows)

Tekrarsız veri sorgulama

- test=# SELECT DISTINCT(ders_adi) FROM dersler;

ders_adi

Fizik

Matematik

(2 rows)

Tekrarsız veri sorgulama

- test=# SELECT DISTINCT(ders_adi),ders_no FROM dersler2;

ders_adi	ders_no
Fizik	1
Matematik	1

(2 rows)

Sıralamalı Sorgulama

- `SELECT <kolonlar> FROM <tablo>
ORDER BY <alan> [ASC|DESC]`
- `<alan>`'a göre artan/azalan sıralamada seçim yapar

Sıralamalı Sorgulama

- test=# SELECT * from bolumler ORDER BY bolum_no ASC;

sira_no	bolum_no	bolum_adi
1	12	Matematik
2	13	Fizik Mühendisliği
3	14	Bilgisayar Programcılığı

(3 rows)

Sıralamalı Sorgulama

- test=# SELECT * from bolumler ORDER BY bolum_no DESC;

sira_no	bolum_no	bolum_adi
3	14	Bilgisayar Programcılığı
2	13	Fizik Mühendisliği
1	12	Matematik

(3 rows)

Koşula Bağlı Sorgulama

- `SELECT <kolonlar> FROM <tablo>`
`WHERE <alan> <operatör> <değer>`
- `<tablo>` ile belirtilen tablodaki `<kolonlar>` ı, `<alan>` alanın `<operatör>` ve `<değer>` ile belirtilen koşullara uyan değer(ler)e göre seçer.
Eğer uyan bir kayıt yoksa, sonucu boş döndürür.

Koşula bağlı sorgulama

- test=# SELECT * FROM bolumler WHERE bolum_adi = 'Fizik Mühendisliği';
sira_no | bolum_no | bolum_adi
-----+-----+-----
2 | 13 | Fizik Mühendisliği
(1 row)
- test=# SELECT * FROM dersler WHERE ders_adi = 'Fizik';
ders_no | ders_adi
-----+-----
1 | Fizik
2 | Fizik
(2 rows)

Koşula bağlı sorgulama

- Operatörler

– <	Küçük
– >	Büyük
– =	Eşit
– <=	Küçük Eşit
– >=	Büyük Eşit
– <>	Eşit Değil

Aralık Sorgulaması

- `SELECT <kolonlar> FROM <tablo>WHERE <alan> BETWEEN <deger1> AND <deger2>;`

<tablo> tablosundaki kolonları, <alan> alanının <deger1> ve <deger2> arasındaki değerlerini seçer.

- Eğer uyan bir kayıt yoksa, sonucu boş döndürür.

Aralık Sorgulaması

- test=# SELECT * FROM dersler WHERE ders_no BETWEEN 2 AND 3;

```
ders_no | ders_adi
-----+-----
      2 | Fizik
      3 | Matematik
(2 rows)
```

Benzerli sorgulama – LIKE

- `SELECT <kolonlar> FROM <tablo> WHERE <kolon> LIKE '<deger>%';`
- `<tablo>` tablosundaki `<kolon>` kolonundan, değeri `<deger>` e benzeyen satırları seçer.
- Burada % işaretinin yeri önemlidir.
- `'<deger>%'` : `<deger>` ile başlayıp belirsiz devam eden
- `'%<deger>'` : sonu `<deger>` ile biten

Benzerli Sorgulama – LIKE

- test=# SELECT * FROM dersler
WHERE ders_adi LIKE 'F%';

ders_no	ders_adi
1	Fizik
2	Fizik
4	Fortran Programlama

(3 rows)

Benzerli Sorgulama – ILIKE

- `SELECT <kolonlar> FROM <tablo> WHERE <kolon> ILIKE '<deger>%';`
- `<tablo>` tablosundaki `<kolon>` kolonundan, değeri `<deger>` e benzeyen satırları seçer. `LIKE`' tan farkı, büyük-küçük harf ayrımı yapmamasıdır.
- Eğer uyan bir kayıt yoksa, sonucu boş döndürür.

Benzerli Sorgulama – ILIKE

- test=# SELECT * FROM dersler WHERE ders_adi ILIKE 'F%';
- ders_no | ders_adi
- -----+-----
- 1 | Fizik
- 2 | Fizik
- 4 | Fortran Programlama
- (3 rows)

Benzerli Sorgulama – ILIKE

- test=# SELECT * FROM dersler WHERE ders_adi ILIKE 'f%';
- ders_no | ders_adi
- -----+-----
- 1 | Fizik
- 2 | Fizik
- 4 | Fortran Programlama
- (3 rows)

Veri Güncelleme

- UPDATE <tablo> SET <kolon1>=<deger> ...;

<tablo> tablosundaki kolon1 kolonunun “TÜM” değerlerini <deger> yapar.

- UPDATE <tablo> SET <kolon1>=<deger> WHERE ...
- WHERE ile belirtilen kısma uyan alanlardaki <kolon1> kolonunun değer(ler)ini <deger> yapar.

Veri Güncelleme

- UPDATE dersler SET ders_adi='Fizik';
- UPDATE dersler SET ders_adi='Fizik 2'
WHERE ders_adi='Fizik';
- UPDATE dersler SET ders_adi='Fizik 2'
WHERE ders_adi LIKE 'Fi%';
- ...

Veri Silme

- DELETE FROM <tablo>;

<tablo> tablosundaki TÜM kayıtları siler.

DELETE FROM <tablo> WHERE <koşul>;

<tablo> tablosundan <koşul> koşuluna uyan kayıtları siler.

Veri Silme

- test=# DELETE FROM dersler;
- DELETE 4
- test=# SELECT * from dersler;
- ders_no | ders_adi
- -----+-----
- (0 rows)
-

Veri Silme

- test=# DELETE from dersler WHERE ders_adi LIKE 'F%';
- DELETE 3
- test=# SELECT * from dersler;
- ders_no | ders_adi
- -----+-----
- 3 | Matematik
- (1 row)

Aritmetiksel İfadeler

- SUM `select sum(brut) from personel`
- AVG `select avg(net) from personel`
- MAX `select max(brut) from personel`
- MIN `select min(brut) from personel`
- COUNT `select count(*) from personel`

Index Oluşturmak

- `CREATE INDEX <index adı> ON <tablo> (<kolon>,..)`

<tablo> tablosunda <kolon(lar)> ile belirtilen kolonlar üzerinde index oluşturur.

```
CREATE INDEX bolum_adi_idx ON
bolumler (bolum_no,bolum_adi);
```

Birden fazla tabloda sorgu

- `SELECT kolonlar> FROM <tablo1>
,<tablo2> where
<tablo1.alan1>=<tablo2.alan2>`

Tablo içinde veri harici deęişim

- ALTER TABLE <tablo> ...
- ALTER TABLE [ONLY] table [*]
- ADD [COLUMN] column type [column_constraint [...]]
- ALTER TABLE [ONLY] table [*]
- DROP [COLUMN] column [RESTRICT | CASCADE]
- ALTER TABLE [ONLY] table [*]
- ALTER [COLUMN] column { SET DEFAULT value | DROP DEFAULT }
- ALTER TABLE [ONLY] table [*]
- ALTER [COLUMN] column { SET | DROP } NOT NULL
-

Tablo içinde veri harici değişim

- ALTER TABLE [ONLY] table [*]
 RENAME [COLUMN] column TO new_column
- ALTER TABLE table
 RENAME TO new_table
- ALTER TABLE [ONLY] table [*]
 ADD table_constraint
- ALTER TABLE [ONLY] table [*]
 DROP CONSTRAINT constraint_name [RESTRICT |
 CASCADE]
- ALTER TABLE table
 OWNER TO new_own

Veritabanı Çeşitleri

- Yanlış bir kanı : “*Paralı ürünler iyidir, ücretsiz ürünler iyi değildir!*”
 - **Linux!**
- Bir veritabanınının ücretsiz olup olmamasından çok işinizi görüp görmeyeceği önemlidir.

Veritabanı Çeşitleri

- 1 MySQL
- 2 IBM DB2
- 3 Interbase
- 4 Informix
- 5 Progress
- 6 PostgreSQL
- 7 Oracle

E-posta listeleri

- linux-programlama@linux.org.tr Veritabanları için tartışma listesi (üye olmak için, <http://liste.linux.org.tr> web arayüzünü kullanabilirsiniz.)
- pgsql-tr-genel@postgresql.org (PostgreSQL Türkiye E-Posta Listesi . PostgreSQL üzerinde her türlü konu.)

Web sayfaları

- <http://foundries.sourceforge.net/databases>
- <http://www.PostgreSQL.org>
- <http://www.mysql.com>
- <http://www.oracle.com>
- <http://otn.oracle.com>

Belgenin güncel hali

- <http://www.gunduz.org>
- <http://seminer.linux.org.tr>
- <http://www.linux.org.tr/belgeler.php>

Veritabanlarına ve SQL'e Giriş

Devrim GÜNDÜZ

TDM Teknoloji Destek Merkezi -- www.tdmsoft.com

devrim@gunduz.org

<http://seminer.linux.org.tr>

<http://www.gunduz.org>

