CSD NOTLARI – TEMEL VERİTABANI

Veritabanı Nedir?

Bilgilerin saklanması ve geri alınması için organize edilmiş dosyalara veritabanı denilmektedir. Veritabanları tek bir dosya olarak organize edilebildiği gibi bir grup dosya biçiminde de organize edilebilir. Genellikle bu organizasyon istenen bilgilerin hızlı bir biçimde elde edilmesi amacıyla gerçekleştirilmektedir. Günümüzde uygulamaların pek çoğu küçük ya da büyük birtakım veritabanlarını kullanmaktadır.

Veritabanlarının organizasyonu için birtakım modeller (paradigmalar) kullanılmaktadır. Günümüzde bunlardan en çok tercih edileni "ilişkisel (relational)" veritabanı modelidir. Ancak farklı uygulamalarda farklı modellerin çeşitli avantajları vardır.

Veritabanı Yönetim Sistemleri (Database Management Systems - DBMS)

Veritabanı işlemleri ticari uygulamalarda uygulamanın performansı üzerinde en etkili olan öğelerdendir. Bu nedenle geliştiriciler veritabanı işlemlerini mümkün olduğunca hızlı yapan araçlar kullanmak isterler. Eskiden veritabanı işlemleri kütüphaneler ile yapılıyordu. Yani bu konuda uzmanlaşmış kişilerin ya da şirketlerin yazmış olduğu kütüphanelerdeki fonksiyonlarla veritabanlarına kayıt eklenip, sorgulamalar yapılıyordu. Ancak bu kütüphanelerin oldukça aşağı seviyeli bir yapısı vardı. Bunlarla çalışma genel olarak zordu. İşte ilk kez 70'li yılların sonlarına doğru "Veritabanı Yönetim Sistemi (VTYS)" ismi altında veritabanı işlemlerini yapan özel uygulamalar geliştirildi. Bu yazılımlar veritabanı işlemlerinden sorumlu oldular.

Bir yazılıma VTYS denebilmesi için onun bazı özelliklere sahip olması gerekmektedir. Bunlardan bazıları şunlardır:

1) Aşağı Seviyeli Dosya Formatlarıyla Kullanıcının İlişkisinin Kesilmiş Olması: VTYS'lerde kullanıcıların bilgilerin hangi dosyalarda ve nasıl organize edildiğini bilmelerine gerek kalmamaktadır. Yani adeta veritabanı kullanıcıya bir kara kutu biçiminde gösterilmektedir. Kullanıcı yalnızca ne yapacağını VTYS'ye iletir. İşlemleri VTYS yapar.

2) VTYS'ler yüksek seviyeli dekleratif dillerle kullanıcı isteklerini yerine getirmektedir. Bu dillerden en yaygın olanı "SQL (Structured Query Language)"dir. SQL asıl sorgulama işlemlerini yapan programların dili değildir. SQL kullanıcının VTYS'ye isteğini anlatmak için kullanılan bir dildir. VTYS bu isteği alır, motor kısmındaki C/C++ ile yazılmış kodlar yoluyla sonuçları elde eder ve kullanıcıya verir.

3) VTYS'ler client-server çalışma modeline sahiptir. Yani birden fazla kullanıcı VTYS'ye istekte bulunabilir. VTYS bu istekleri karşılar. Yani biz bir VTYS'yi bilgisayarımıza kurduğumuzda aynı zamanda bir server da kurmuş oluruz.

4) VTYS'lerde belli düzeylerde güvenlik ve güvenilirlik mekanizması (security and safety) oluşturulmuştur. Yani bilgiler bu sistemlerde kolayca bozulmazlar ve çalınmazlar.

5) VTYS'lerin çoğu yardımcı birtakım araçlar içermektedir. Örneğin backup-restore programları, yönetici programlar, kütüphaneler vs.

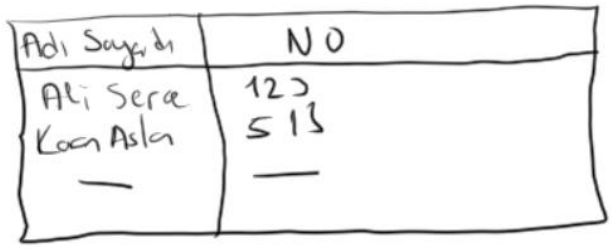
Pekiyi günümüzde en çok tercih edilen VTYS'ler nelerdir? Oracle firmasının Oracle isimli ürünü büyük veritabanları için kurumların en çok tercih ettiği VTYS'lerden biridir. Microsoft'un SQLServer isimli ürünü doğrudan Oracle ile rakip durumdadır. Pek çok kurum Sql Server'ı tercih etmektedir. Bunun dışında ücretli başka VTYS'ler de vardır. Ancak ücretsiz ve açık kaynak kodlu

da pek çok VTYS geliştirilmiştir. MySql açık kaynak kodlu bir projedir. Ancak bazı haklarını daha sonra Oracle satın almıştır. Açık kaynak kodlu olarak devam etmektedir. PostGreSQL diğer bir bedava ve açık kaynak kodlu VTYS'dir. Son dönemlerde gittikçe popülaritesi artmaktadır.

Bir grup VTYS aslında VTYS'lerin pek çok özelliğini barındırmasa da SQL kullanımına izin vermektedir. Bunların kurulum sorunları yoktur. Bunlar adeta bir veritabanı kütüphanesi gibi tek bir kütüphane dosyasından (örneğin DLL'den) oluşmuşlardır. Özellikle gömülü sistemlerde tercih edilmelerinden dolayı bunlara "Gömülü VTYS (Embedded DBMS)" de denilmektedir. Bunların en yaygın olanı şu günlerde SqLite'tır. SqLite hem Windows, hem Linux hem MAC OS X hem de mobil işletim sistemlerinde aynı biçimde kullanılmaktadır.

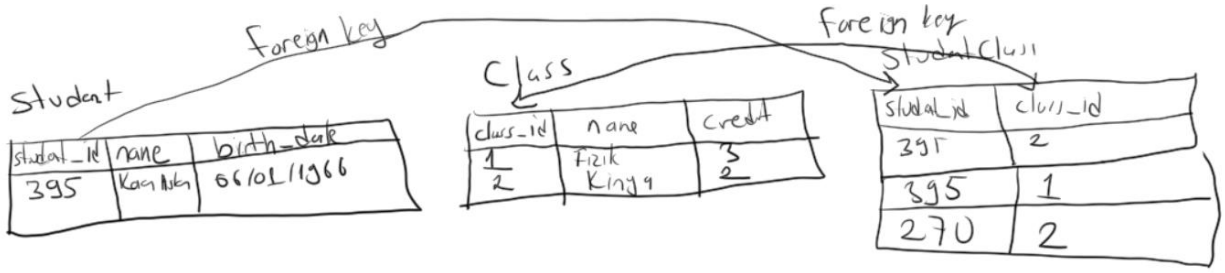
İlişkisel Veritabanları (Relational Databases)

İlişkisel veritabanları kabaca tablolardan (tables), tablolar da sütunlardan (fields) ve satırlardan (records) oluşmaktadır.



Tablolarda yinelenmeme garantisi verilen sütunlara "Birincil Anahtar (Primary Key)" denilmektedir. Genellikle her tablosunun bir tane birincil anahtara sahip olması tavsiye edilir.

İlişkisel veritabanlarında bilgiler birden fazla tabloda bulunuyor olabilir. Böylece bilgi elde edilirken birden fazla tablodan alınabilir. Örneğin:



İdeal olarak veritabanı tabloları tekrar bilgisi içermemelidir. Örneğin Bir öğrencinin adı ve soyadı birden fazla tabloda gereksiz bir biçimde bulundurulmamalıdır. Tabii tablolar arasında geçiş yapmak için ortak bir anahtara gereksinim duyulur. Bunun için oluşturulan sütunlara (alanlara) "yabancı anahtar (foreign key)" denilmektedir. Yukarıdaki veritabanında öğrencinin numarası onun hangi dersleri aldığı bilgisini elde etmek için kullanılmaktadır. Benzer biçimde öğrencinin derslerin id numaraları da o derslerin diğer bilgilerine erişmekte kullanılmaktadır. İşte büyük veritabanlarında böyle binlerce tablo bulunabilmektedir. Uygun bilginin elde edilmesi bir sürü tablo gezinerek yapılabilmektedir. Tabloların tekrarı engelleyecek biçimde düzenlenmesine veritabanı terminolojisinde "normalizasyon" denilmektedir. Veritabanı tablolarını tasarlamakta kullanılan yazılımsal araçlar da vardır. Fakat kursumuzda bunlar ele alınmayacaktır.

Temel SQL Bilgisi

CSD NOTLARI – TEMEL VERİTABANI

SQL büyük harf küçük harf duyarlılığı olan bir dil değildir. Dolayısıyla SQL komutlarının büyük harf ya da küçük harfli yazılmaları sorun oluşturmaz. Pek çok programcı bir yazım stili olarak SQL komutlarını büyük harflerle belirtmektedir. Ayrıca SQL komutları ‘;’ ile sonlandırılmaktadır. Ancak pek çok VTYS komutlar ‘;’ ile sonlandırılmasa bile bir sorun oluşturmamaktadır.

Veritabanın Yaratılması

Şüphesiz ilk iş bir veritabanının sıfırdan yaratılmasıdır. Bu işleme bir kez gereksinim duyulacağı için VTYS’lerinin yönetim ekranından yapılabilir. SQL’de veritabanı yaratmak için CREATE DATABASE komutu kullanılmaktadır. Komutun genel biçimi şöyledir:

```
CREATE DATABASE <isim>;
```

Veritabanı Tablolarının Yaratılması

Veritabanı yaratıldıktan sonra sıra tabloların yaratılmasına gelir. Yine tabloları biz VTYS’lerin sunduğu GUI araçlarıyla ya da doğrudan SQL cümleleriyle yaratabiliriz. Tabloların yaratımı sırasında tabloların isimleri, onların sütunlarının isimleri ve türleri tek tek belirtilir. Sütun türleri SQL standartlarında belirtilmiştir. Ancak farklı VTYS’ler kendi özgü türler de kullanmaktadır. Tipik sütun türleri (alanlara ilişkin türler) şunlardır:

CHARACTER(n)	En fazla n karakterlik yazıların tutulacağı alan için kullanılır. Genellikle VTYS yazı n karakterden küçük olsa bile n karakterlik yeri ayırmaktadır.
VARCHAR(n)	En fazla n karakterli yazıların tutulabileceği alan için kullanılır Ancak n’den az karakterli yazılar için n karakterlik yer ayrılmaz.
SMALLINT	Bu tür genellikle 2 byte uzunluğunda olan tamsayıları belirtmektedir. Ancak VTYS’den VTYS’ye değiebilir.
INTEGER	Bu tür genellikle 4 byte uzunluğunda olan tamsayıları belirtmektedir. Ancak VTYS’den VTYS’ye değiebilir.
BIGINT	Bu tür genellikle 8 byte uzunluğunda olan tamsayıları belirtmektedir. Ancak VTYS’den VTYS’ye değiebilir.
FLOAT	Genellikle 8 byte’lık gerçek sayı türünü turmak için kullanılmaktadır.
REAL	Genellikle 4 byte’lık gerçek sayı türünü turmak için kullanılmaktadır.
DOUBLE	Genellikle 8 byte’lık gerçek sayı türünü turmak için kullanılmaktadır.
DATE	Tarih bilgisi tutmak için kullanılır
TIME	Zaman bilgisi tutmak için kullanılır
BLOB	En fazla 2 byte uzunluğunda (65526) binary alan.
TINYBLOB	En fazla 1 byte uzunluğunda (256) binary alan
MEDIUMBLOB	En fazla 3 byte uzunluğunda (16777216) binary alan.
LONGBLOB	En fazla 4 byte uzunluğunda (4294967296) binary alan.
TINYTEXT	En fazla 1 karakter uzunluğunda (256) text alan
MEDIUMTEXT	En fazla 3 byte uzunluğunda (16777216) text alan.
TEXT	En fazla 2 byte uzunluğunda (65526) text alan.

Pek çok VTYS yukarıdaki türlerden başka pek çok tür de bulundurmaktadır. Örneğin MySQL’de işaretsiz tamsayı türleri de vardır.

Tablo yaratmak için CREATE TABLE Sql komutu kullanılır. Komutun yalın genel biçimi şöyledir:

```
CREATE TABLE <table_name>
(
    <column_name1> <data_type>(size),
```

```
    <column_name2> <data_type>(size),
    <column_name3> <data_type>(size),
    ....
);
```

Komutun ayrıntıları için Sql kaynaklarına başvurulabilir. Örneğin:

```
CREATE TABLE student_info(student_id INTEGER PRIMARY KEY AUTO, student_name
VARCHAR(45), student_bdate DATE);
```

Veritabanı üzerinde temel işlemler

Konu ne olursa olsun veriler üzerinde şu işlemler yapılabilir: (CRUD)

1. Create: Kayıt ekleme
2. Read: Sorgulama
3. Update: Güncelleme
4. Delete: Silme

Tabloya Kayıt Ekleme İşlemi

Tabloya kayıt eklemek için INSERT INTO Sql komutu kullanılır. Komutun genel biçimi şöyledir:

```
INSERT INTO <table_name> (column1,column2,column3,...)VALUES
(value1,value2,value3,...);
```

Bir çok VTYS' de kayıt eklerken yazısal sütunlar ile tarih ve zaman sütunları tek tırnak içerisine alınmalıdır. Bunun dışında sayısal sütun bilgileri doğrudan yazılır. Örneğin:

```
INSERT INTO student_info(student_name, student_bdate) VALUES ('John Lennon',
'1940/10/9');
```

Insert işlemi sırasında biz bazı sütunları belirtmeyebiliriz. Ancak belirtmediğimiz sütunlar için default değer tanımlamasının yapılmış olması gerekir. Bazı sütunlar "Auto Increment" olabilmektedir. Bu durumda VTYS kayıt ekleme sırasında önceki değerin bir fazlasını bu sütuna değer olarak verir.

WHERE Cümleciği

WHERE cümleceği Sql'de bir komut değildir. Bazı komutların içerisinde kullanılan bir kalıptır. Örneğin DELETE FROM komutunun, SELECT komutunun WHERE cümleciği kısımları vardır. WHERE cümleciği koşul belirtmektedir. Koşul belirtilirken sütun isimleri ve temel karşılaştırma operatörleri kullanılabilir. Örneğin:

```
WHERE student_id > 1250
```

Koşullarda mantıksal AND, OR ve NOT operatörleri kullanılabilir. Örneğin:

```
WHERE student_name = 'Kaan Aslan' AND student_id > 2450
```

LIKE opereratörü yazısal bir sütunun belli bir kalıba uygunluk koşulu için kullanılır. % joker karakteri "bundan sonra herhangi karakterler gelebilir" anlamındadır.

Örneğin:

```
WHERE student_name LIKE  'S%'
```

Burada ismi s ile başlayan öğrenciler için koşul verilmiştir.

Kayıt Silme İşlemi

Belli koşulları sağlayan kayıtların silinmesi DELETE FROM komutuyla yapılmaktadır. Komutun genel biçimi şöyledir:

```
DELETE FROM <tablo ismi> [WHERE cümleciği]
```

Örneğin:

```
DELETE FROM student_info WHERE student_name = ‘Ali Serçe’
```

Bu komutla ismi Ali Serçe olan tüm kayıtlar silinmektedir. Örneğin:

```
DELETE FROM student_info WHERE student_id > 100
```

Burada id’si 100’den büyük tüm kayıtlar silinmektedir.

Koşulu Sağlayan Kayıtların Elde Edilmesi

Koşulu sağlayan kayıtların elde edilmesi için SELECT komutu kullanılmaktadır. SELECT komutunun genel biçimi oldukça ayrıntılıdır. Pek çok cümlecik (örneğin WHERE cümleciği) komut içerisinde bulundurulabilmektedir.

SELECT komutun tipik kullanımı şöyledir:

```
SELECT <sütun listesi> FROM <tablo ismi> [WHERE cümleciği]
```

Sütun listesi yerine ‘*’ karakteri getirilirse tüm sütunlar anlaşılır. WHERE cümleciği kullanılmazsa tüm kayıtlar anlaşılır. Örneğin MySQL’in örnek “World” veritabanı için şöyle bir SELECT komutu yazmış olalım:

```
SELECT Code FROM country WHERE Name LIKE 'T%'
```

Bu komutla ilk harfi T ile başlayan tüm ülkelerin ülke kodları elde edilecektir. Örneğin:

SELECT komutunda VTYS’nin hazır bazı fonksiyonları kullanılabilmektedir. Her VTYS’nin birtakım hazır fonksiyonları vardır. Ancak bu konuda bir standart bulunmamaktadır. Örneğin MySql’de DAYOFMONTH isimli fonksiyon bir tarihin gün değerini verir. Biz de bu sayede aşağıdaki gibi bir SELECT komutu oluşturabiliriz:

```
SELECT * FROM student_info WHERE DAYOFMONTH(student_bdate) < 10
```

Örneğin:

```
SELECT * FROM student_info WHERE MOD(age, 10) = 0
```

Burada yaşı 10'un katlarında olan öğrencilerin bilgileri listelenmek istenmiştir. Örneğin:

VTYS'lerin fonksiyon listelerine onların dokümanlarından erişilebilir. Ancak bu fonksiyonların standart olmadığını yani her VTYS fonksiyonlarının birbirlerinden farklılık gösterebildiğini anımsatalım.

SELECT ile birden fazla tablodan bilgi alınabilir. Bu işleme genel olarak "join (birleştirme)" işlemi denilmektedir. Join işleminin INNER, LEFT, RIGHT ve FULL biçiminde türevleri vardır. Ancak bu join türevleri tüm VTYS'ler tarafından tam olarak desteklenmeyebilmektedir. Join işlemi denildiğinde default olarak INNER JOIN anlaşılmaktadır.

Join işlemi kartezyen çarpım işlemi biçiminde ele alınarak açıklanabilir. Bilindiği gibi iki kümenin kartezyen çarpımı sıralı ikililerden oluşmaktadır. Bu sıralı ikililerin ilk terimleri soldaki kümeden, ikinci terimleri sağdaki kümeden oluşturulur:

```
A X B = { (a, b) | a € A ve b € B}
```

İşte biz iki tabloyu bu biçimde kartezyen çarpım işlemine sokarsak iki tablonun eleman sayılarının çarpımı kadar kayıt elde etmiş oluruz. Sonra bu kayıtlardan WHERE cümlesi ile belirtilen koşulu sağlayanlar seçilirse bu işleme INNER JOIN denilmektedir. INNER JOIN sentaksı şöyledir:

```
SELECT <sütun listesi> FROM table1 INNER JOIN table2 ON <koşul>
```

Sütun ve koşul kısımlarında her iki tablonun sütunları bulundurulabileceğinden dolayı bir çakışma söz konusu olabilir. Çatışma durumunda sütun isimleri tablo isimleriyle araya '.' karakteri konularak niteliklendirilebilir. Aslında SQL kullanıcıları çakışma olmasa da sütunları hep tablo isimleriyle niteliklendirmektedir. Örneğin MySQL'in örnek "world" veritabanı için aşağıdaki sorgulamayı yapıyor olalım:

```
SELECT city.Name, country.Name FROM city INNER JOIN country ON city.CountryCode = country.Code
```

Burada biz sonuç olarak city tablosundaki isimleri ile country tablosoundaki isimleri beraber görüntülemek istemekteyiz. Ancak bu iki tablonun kartezyen çarpımındaki tüm satırlar için bu işlemler yapılmayacak. Yalnızca ON kısmında belirtilen koşulların sağlandığı satırlar elde edilecek. Bu işlemin sonucunda da biz tüm şehirlerin hangi ülkeye ilişkin olduğuna ilişkin bir liste elde ederiz.

Name	Name
Oranjestad	Aruba
Kabul	Afghanistan
Qandahar	Afghanistan
Herat	Afghanistan
Mazar-e-Sharif	Afghanistan
Luanda	Angola
Huambo	Angola
Lobito	Angola
Benguela	Angola

Örneğin:

```
SELECT country.Name, countrylanguage.Language, countrylanguage.Percentage FROM country INNER JOIN
countrylanguage ON country.Code = countrylanguage.CountryCode
```

Name	Language	Percentage
Tonga	English	0.0
Tonga	Tongan	98.3
Trinidad and Tobago	Creole English	2.9
Trinidad and Tobago	English	93.5
Trinidad and Tobago	Hindi	3.4
Tunisia	Arabic	69.9
Tunisia	Arabic-French	26.3
Tunisia	Arabic-French-English	3.2
Turkey	Arabic	1.4
Turkey	Kurdish	10.6
Turkey	Turkish	87.6

INNER JOIN işlemi için alternatif bir sentaks daha vardır. Bu sentaks doğrudan birden fazla tablonun isminin geçtiği SELECT cümlesi sentaksıdır. Örneğin:

```
SELECT city.Name, country.Name FROM city INNER JOIN country ON city.CountryCode = country.Code
```

INER JOIN işleminin eşdeğeri şöyle de yazılabilir:

```
SELECT city.Name, country.Name FROM city, country WHERE city.CountryCode = country.Code
```

Örneğin:

```
SELECT country.Name, countrylanguage.Language, countrylanguage.Percentage FROM country INNER JOIN countrylanguage ON country.Code = countrylanguage.CountryCode
```

INNER JOIN işleminin de eşdeğeri şöyle yazılabilir:

```
SELECT country.Name, countrylanguage.Language, countrylanguage.Percentage FROM country, countrylanguage WHERE country.Code = countrylanguage.CountryCode
```

LEFT JOIN işleminde sol taraftaki tablonun tüm satırları ve ON koşulunu sağlayan satırlar alınır. Sol taraftaki tablonun ON koşulunu sağlamayan satırlarının sağ taraf sütunları boş (NULL) biçimdedir. Örneğin:

```
SELECT city.Name, country.Name FROM city LEFT JOIN country ON city.CountryCode = country.Code AND country.Population > 50000000
```

Name	Name
Naogaon	Bangladesh
Sirajganj	Bangladesh
Narsinghdi	Bangladesh
Saidpur	Bangladesh
Gazipur	Bangladesh
Bridgetown	NULL
Antwerpen	NULL
Gent	NULL
Charleroi	NULL

RIGHT JOIN ise LEFT JOIN işleminin tersidir. Yani sağ taraftaki tablonun tüm satırları ve ON koşulunu sağlayan satırlar alınır. Sağ taraftaki tablonun ON koşulunu sağlamayan satırlarının sol taraf sütunları boş (NULL) biçimdedir. Örneğin:

```
SELECT city.Name, country.Name FROM city RIGHT JOIN country ON city.CountryCode = country.Code AND country.Population > 50000000
```

Name	Name
NULL	Belgium
NULL	Benin
NULL	Burkina Faso
Dhaka	Bangladesh
Chittagong	Bangladesh
Khulna	Bangladesh
Rajshahi	Bangladesh

FULL JOIN pek çok VTYS tarafından desteklenmemektedir. Bu işlemde sol taraftaki ve sağ taraftaki tabloların bütün satırları ayrıca bir de koşulu sağlayan satırlar elde edilir. Ancak koşulu sağlamayan satırların diğer tablo karşılıkları boş (NULL) olarak elde edilir.

Aslında SQL burada anlatılanlardan daha ayrıntılı bir dildir. Ancak kursumuzda bu kadar bilgi yeterli görülmüştür. Fakat ne olursa olsun ne kadar çok SQL bilinirse o kadar etkin işlemler yapılabilmektedir.