

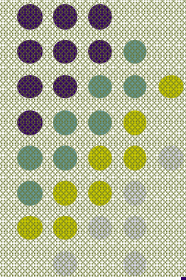
# VERİ TABANI DERS NOTLARI

## 1.DERS : GİRİŞ, TANIMLAR

### Kaynaklar:

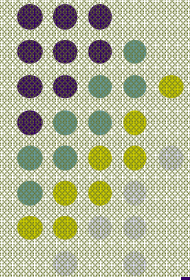
- *Prof. Dr. Ünal Yarımağan*, Veri Tabanı Sistemleri, Akademi Yayıncılık, 2010
- *Elmasri, Navathe*, Fundamentals of Database Systems, 6th Ed., Pearson, 2011

# AMAÇLAR



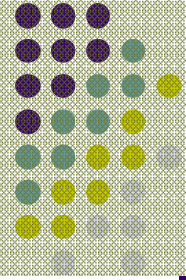
- Veritabanı yönetim sistemleri temel kavramlarını anlama
- Kavramsal veri modeli ile veritabanı tasarımı
  - Varlık-Bagıntı modeli (ER)
- İlişkisel Model
- İlişkisel sorgu dili SQL'i öğrenme ve uygulama
- Bir ilişkisel veritabanı yönetim sistemi kullanarak
  - Veritabanı tasarımı
  - Veritabanı gerçekleştirimi
  - Veritabanı üzerinde sorgulama yapma

# Veritabanı Türleri&Uygulamaları



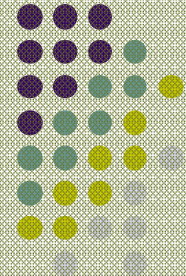
- Sayısal ve Metinsel Veritabanları
- Multimedya Veritabanları
- Coğrafik Bilgi Sistemleri
- Veri Ambarları
- Gerçek-zamanlı ve Aktif Veritabanları

# Konu Başlıkları



- VT nedir?
- VTYS nedir?
- Veri modeli nedir?
- Örnek bir veri tabanı: ÜNİVERSİTE
- VT kullanıcı Türleri
- VT dilleri
- VTYS olanakları ve dosya sisteminden farkları
- VTYS ana işlevsel modülleri
- VTYS 3-şema mimarisi ve veri bağımsızlığı
- VT kullanım/erişim mimarileri (2-katlı, 3 –katlı)
- VT türleri

# Veri Tabanı Nedir?

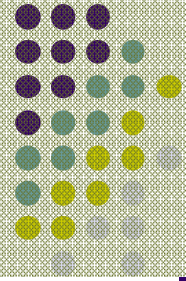


- Veri, bir anlamı olan ve kaydedilebilen gerçekler. (Bir kişinin ismi, adresi, telefon numarası vs.)
- Veri, olguların, kavramların veya talimatların, insan tarafından veya otomatik yolla, iletişim, yorumlama ve işleme amacına uygun bir biçimde ifadesidir(ANSI Tanımı)

Veritabanı ise

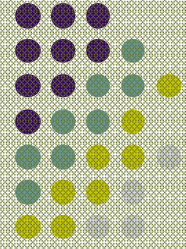
- Birden çok uygulama/kullanıcı tarafından erişilen
- Gereksiz yinelenmelerden arınmış
- Düzenli bir şekilde saklanan
- Birbiriyle ilişkili (*uyumlu olarak*)
- Sürekli, fakat statik olmayan (güncelleme yapılan)
- Belirli bir amaç için bir araya getirilmiş **veri topluluğudur.** <sup>5</sup>  
(*küçük bir dünya*). Şirket, bakanlık, üniversite, stok takip vs

# VT nedir?



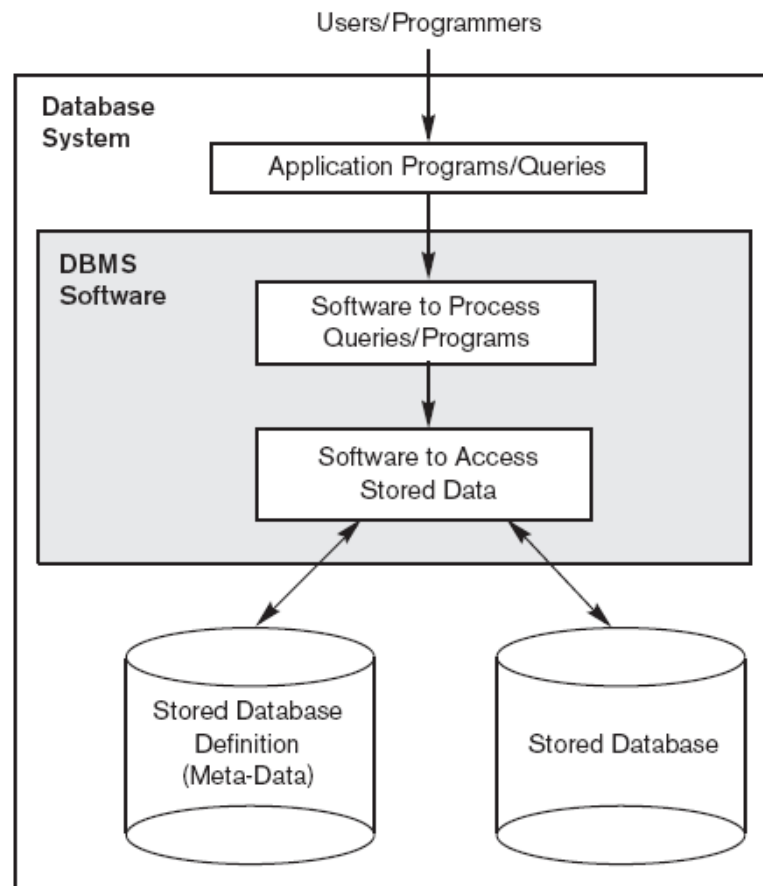
- VT büyüklüğü / karmaşıklığı için bir kısıtlama yok
- Modellediği minidünya gerçekçi ve güvenilir olması için, olası değişiklikler VT'ye olabildiğince hızlı yansıtılmalı.
- Örnek : Amazon.com (ticari bir veritabanı)
  - 2 teraB ( $10^{12}$  B)
  - 20 milyon kitap
  - 200 sunucu bilgisayar üzerinde kayıtlı
  - Günlük 15 milyon kullanıcı
  - Yaklaşık 100 admin (Alış/satışa rağmen güncel tutma)

# VTYS Nedir?



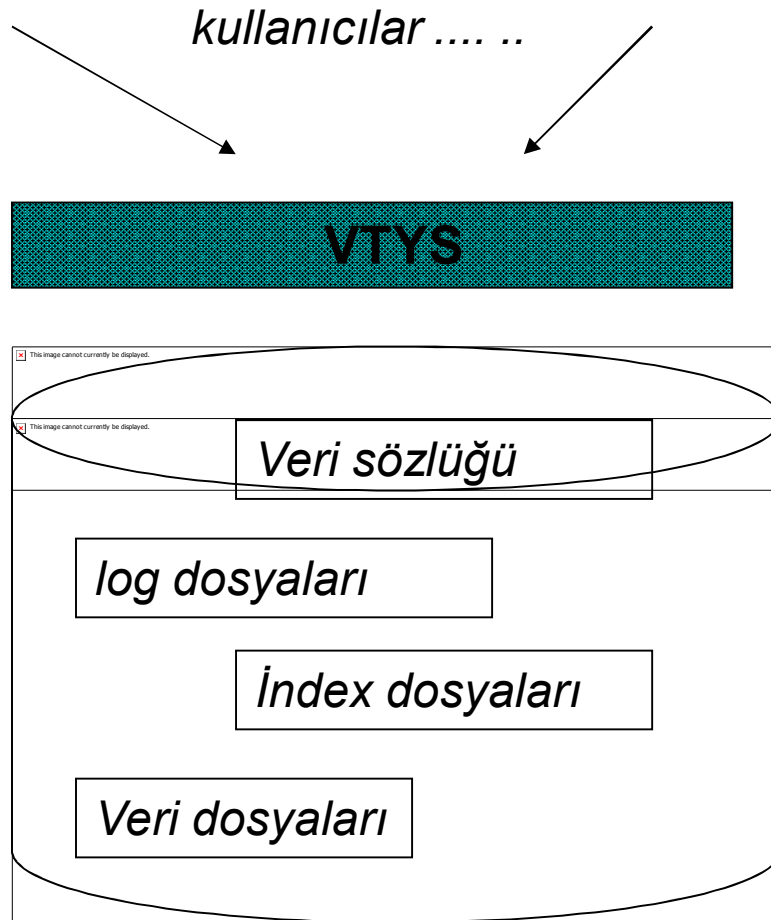
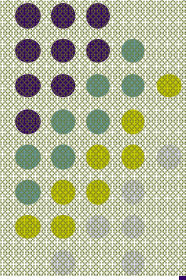
- Veri tabanı ile ilgili her türlü *işletimsel gereksinimleri* (oluşturma, bakım, güncelleme) karşılamak için kullanılan **karmaşık, merkezi** yazılım sistemine VTYS denir. VT oluşturma ve bakımı için **programlar topluluğu**. **VTYS genel olarak şu olanakları sağlar:**
  - VT tanımlanması(define), gerçekleşmesi(oluşturulması, construct), kullanımı (manipulate, update, query, report), paylaşımı (share)
  - Kontrollü veri tekrarı
  - Verimli erişim metodları ile sorgu işleme
  - Çok kullanıcıli hizmet,
  - Farklı kullanıcı arayüzleri,
  - Karmaşık iş kısıtlamalarının tanımlanması ve gerçekleşmesi,
  - Güvenlik tanımlamaları ve sağlanması (veri kurtarma/yedekleme),
- Veritabanına, **gerek işletim sistemi gerek diğer kullanıcılar (uygulama programları gibi) doğrudan erişemez**; ancak VTYS üzerinden erişim yapılır.

# DB Environment



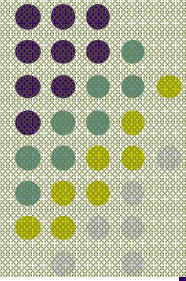
**Figure 1.1**  
A simplified database  
system environment.

# Genel VT / VTYS yapısı



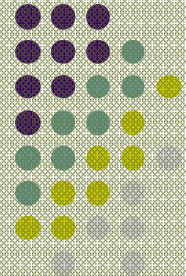
- **Veri sözlüğü**, veri tabanı tanımlarının (*metadata*) saklandığı dosyalardır. DB katalogu da denir.
- **İndex dosyaları**, fiziksel erişim dosyalarıdır.
- **Log dosyaları** güvenlik amaçlı (kurtarma için) dosyalardır.

# Terimler



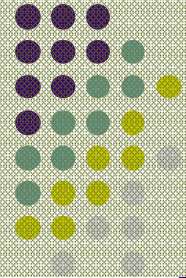
- **Uygulama programı:** veri için istek veya sorgu göndererek VT'ye erişim yapan program.
- **Sorgu:** VT'deki verinin alınmasına (retrieve) neden olan işlem.
- **Transaction:** Bazı verilerin VT'den okunmasına ve bazı verilerin VT'ye yazılmasına neden olan hareket, iş.

# Veri Modeli Nedir?



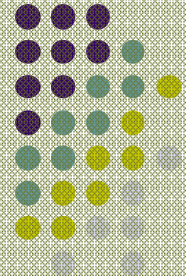
- Gerçek dünya verilerini kavramsal ve mantıksal seviyede düzenlemek için kullanılan *yapı ve kavramlar bütünü* olarak tanımlanır.
- **Veri Modeli** :Veritabanı yapısını, veriler arası ilişkileri ve veritabanının uyacağı kısıtlamaları tanımlamak için kullanılan kavramlar kümesi.
- Veri Modeli İşlemleri :
  - Veritabanından bilgi alma/güncellemeler ile veri modeli kavramlarına dayanarak belirlenen işlemler.
- Örnek; E-R (entity-relationship) modeli, ilişkisel model (RM)
- Genel VT tasarımı:
  - kavramsal tasarım
  - mantıksal düzenleme, varlık ve bağıntıların belirlenmesi
  - veri tipleri, değer aralığı, uzunluk belirlenmesi
  - veri bütünlüğü kısıtlamalarının belirlenmesi
  - fiziksel tasarım tercihleri
  - kullanıcıların belirlenmesi ve güvenlik ayarları

# Categories of Data Models



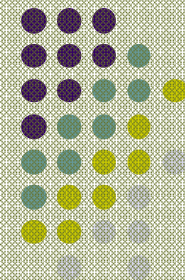
- **Conceptual (high-level, semantic) data models: (Kavramsal modelleme)**
  - Provide concepts that are close to the way many users perceive data. (Also called entity-based or *object-based* data models.)
- **Implementation (representational) data models: (Gerçekleştirimsel / Gösterimsel Modelleme)**
  - used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).
- **Physical (low-level, internal) data models: (Fiziksel Modelleme)**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals

# Schemas versus Instances



- Database Schema:
  - The ***description*** of a database.
  - Includes descriptions of the database structure, data types, and the constraints on the database.
- Schema Diagram:
  - An ***illustrative*** display of (most aspects of) a database schema.
- Schema Construct:
  - A ***component*** of the schema or an object within the schema, e.g., STUDENT, COURSE.
- Database State/instance:
  - The actual data stored in a database at a ***particular moment in time***. This includes the collection of all the data in the database.
  - Also called database instance (or occurrence or snapshot).
  - The term *instance* is also applied to individual database components, e.g. *record instance*, *table instance*, *entity instance*

# Schema vs. State (Üniversite Öğrenci-Ders Sistemi)



## STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

**Figure 2.1**

Schema diagram for the database in Figure 1.2.

## COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

## PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

## GRADE\_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

## COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

## GRADE\_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

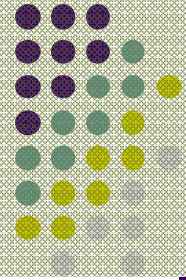
## PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

**Figure 1.2**

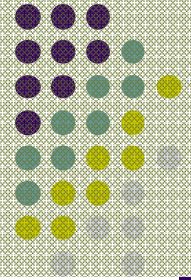
A database that stores student and course information.

# Örnek Bir Veritabanı (Kavramsal Veri Modeli)



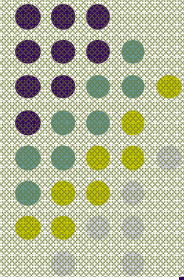
- Üniversite ortamının bir parçası (Mini-dünya)
- Bazı mini-dünya varlıkları (Varlık-bağıntı modeli ile)
  - ÖĞRENCİLER
  - DERSLER
  - BÖLÜMLER
  - HOCALAR
- Bazı mini-dünya ilişkileri (Varlık-bağıntı modeli ile)
  - ÖĞRENCİLER DERSLERİ *alır*.
  - Bazı DERSLERİN önkoşul DERSLERİ *vardır*.
  - HOCALAR DERSLERİ *verir*.
  - BÖLÜMLER DERSLERİ *açar*.
  - ÖĞRENCİLER BÖLÜMLERE *kayıtlıdır*.

# An Example (cont'd.)



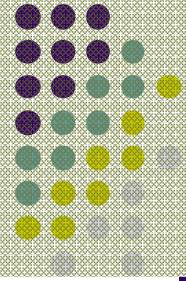
- Examples of queries:
  - Retrieve the transcript
  - List the names of students who took the section of the 'Database' course offered in fall 2008 and their grades in that section
  - List the prerequisites of the 'Database' course
- Examples of updates:
  - Change the class of 'Smith' to sophomore
  - Create a new section for the 'Database' course for this semester
  - Enter a grade of 'A' for 'Smith' in the 'Database' section of last semester

# History of Data Models



- Network Model
- Hierarchical Model
- Relational Model
  - **Collection of records**
  - Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82.
  - Now in several commercial products (e.g. DB2, ORACLE, MS SQL Server, SYBASE, INFORMIX).
  - Several free open source implementations, e.g. MySQL, PostgreSQL
  - SQL relational standards: SQL-89 (SQL1), SQL-92 (SQL2), SQL-99, SQL3, SQL2006, SQL2008
- Object-oriented Data Models / Object-Relational Models
  - **Collection of objects and object-specific methods**
  - One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
  - Additionally, systems like O2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
  - Object Database Standard: ODMG-93, ODMG-version 2.0, ODMG-version 3.0.
- Hierarchical models (*new generation*)
  - XML: collection of records having subrecords; like tree of values.

# VT kullanıcıları



- VT Sorumlusu (Admin, DBA)

- VT tasarımından işletimine kadar herşeyinden sorumlu kişi(ler)
- VT erişim yetkilerini tanımlama ve kullanımı kontrol
- Sistem için gerekli s/w,h/w desteğini belirler
- Güvenlik için yedekleme ve kurtarma işlemleri,
- VT kullanımını ve başarımını takip etmek, gerekli değişiklikleri yapmak

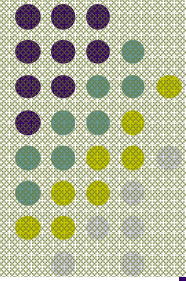
- Tasarımcı (Designer)

- Verinin her aşamada modellenmesi(yapısı, içeriği ve kısıtlamaları ile). Gerçekleme öncesi aşamalardan sorumludur. VT kullanıcıları ile haberleşir gereksinim analizi yapar. Veritabanı üzerinde fonksiyon ve işlemleri tanımlar. Genel olarak bütün kullanıcılar ile yakın temas.
- Hangi veri saklanacak, şema tanımları
- Veriler için uygun yapıları belirleme.

- Sistem Çözümleyici (System analysts)

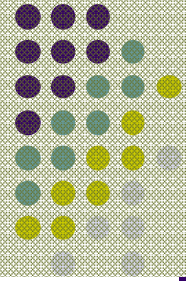
- Son kullanıcıların gereksinimlerini belirler

# VT kullanıcıları (Devam)



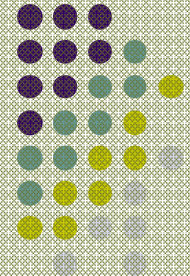
- VT uygulama yazılımcısı (**Application Programmer**)
  - VT erişimi yapacak uygulamaları tasarlama ve gerçekleştirme
- Son kullanıcılar (**End user**)
  - VT'deki bilgilere sorgular ve raporlar aracılığıyla erişirler veya veritabanı içeriğini güncellerler.
  - Uygulama programı kullanıcıları
  - SQL dili kullanıcıları
  - Rastgele kullanıcılar

# DBMS Languages



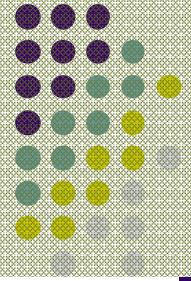
- Data Definition Language (DDL) : Veri Tanımlama Dili(VTD)
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
- Data Manipulation Language (DML) (Veri İşleme Dili)
  - High-Level or Non-procedural Languages: These include the relational language SQL
    - May be used in a standalone way or may be embedded in a programming language
  - Low Level or Procedural Languages:
    - These must be embedded in a programming language

# DBMS Programming Language Interfaces



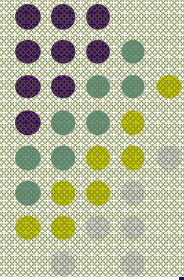
- VTYS komut konsolu
- **Programmer interfaces** for embedding DML in a programming languages:
  - **Embedded Approach:** e.g. embedded SQL (for C, C++, etc.), SQLJ (for Java)
  - **Procedure Call Approach:** e.g. JDBC for Java, ODBC for other programming languages
  - **Database Programming Language Approach:** e.g. ORACLE has PL/SQL, a programming language based on SQL; language incorporates SQL and its data types as integral components
- **User-friendly interfaces**
  - Menu-based, popular for browsing on the web, Forms-based, Graphics-based (*Point and Click, Drag and Drop, etc.*)
  - Natural language: requests in written English
- **Application Development Environments** and CASE (computer-aided software engineering) tools : *PowerBuilder* (Sybase), *JBuilder* (Borland)<sup>21</sup>, *JDeveloper 10G* (Oracle)

# Database System Utilities



- To perform certain functions such as:
  - Loading data stored in files into a database. Includes data conversion tools.
  - Backing up the database periodically on tape.
  - Reorganizing database file structures.
  - Report generation utilities.
  - Performance monitoring utilities.
  - Other functions, such as sorting, user monitoring, data compression, etc.

# Niye VTYS? *dosya-sistemi niye yetersiz?*



```
public static List<String> getStudents1997() {
    List<String> result = new ArrayList<String>();
    FileReader rdr = new FileReader("students.txt");
    BufferedReader br = new BufferedReader(rdr);
    String line = br.readLine();
    while (line != null) {
        String[] vals = line.split("\t");
        String gradyear = vals[2];
        if (gradyear.equals("1997"))
            result.add(vals[1]);
        line = br.readLine();
    }
    return result;
}
```

(a) Using a file system model

```
select SName from STUDENT where GradYear = 1997
```

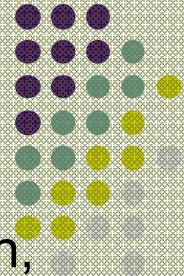
(b) Using the relational model

## **Figure 1-3**

Two ways to retrieve the name of students graduating in 1997

- Veri sorgulama kolaylığı

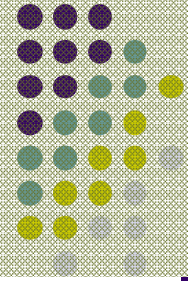
# Niye VTYS? *dosya-sistemi niye yetersiz?*



AMAÇ: yavaş saklama unitesinde saklanan, çok kişi tarafından erişilen, büyük veri yığınlarında, veri kurtarma desteği ile gerçek zamanda hizmet vermek için VTYS'na ihtiyacımız var. Bu amaca hizmet edecek aşağıdaki maddeler sıralanabilir:

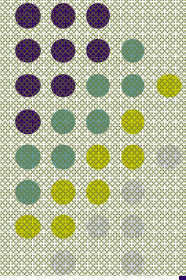
- Üstveri(metadata): Veri tabanında saklanan verinin tanımları (meta data) ve diğer kısıtlamalar da saklanıyor. Farklı küçük dünyalar aynı veri tabanında saklanabiliyor. (Buna *Program-veri bağımsızlığı* denir.)
  - *Dosya sistemi uygulaması sadece o uygulama için yazılmış.*
- Kayıt saklama ve erişimde
  - *“güçlü” veri yapıları ihtiyacı*
  - *Tampon kullanımı*
- program-operasyon bağımsızlığı ihtiyacı
- Verinin farklı görünümü (multiple views): Kullanıcılara veritabanının sadece kendilerini ilgilendiren belli bir görünümünün sunulması
- Paylaşım (sharing) ve çoklu hareket işleme (transaction processing) imkanı
  - *Eşzamanlılık*
  - *Veri kurtarma ve geri sarma*
  - *Güvenlik ve yetkilendirme*

# Avantajlar (Devam)

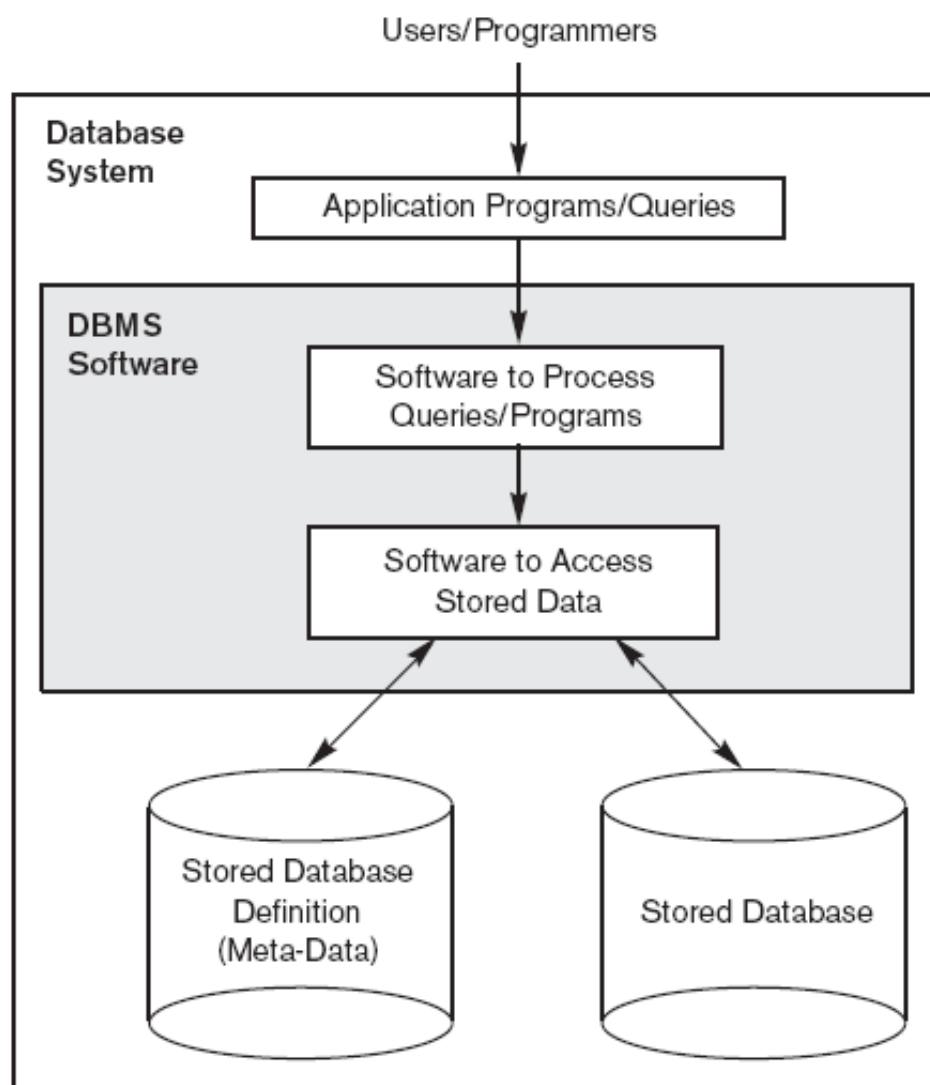


- Verilerin tekrarını önleme, yazılım geliştirme ve bakım işlemlerinde kolaylık
- Yedekleme ve kurtarma hizmetleri
- Değişik kullanıcı sınıfları için farklı arayüzler
- Veriler arasında karmaşık ilişkiler tanımlayabilme
- Veriler üzerinde bütünlük kısıtlamaları ile verilerin tutarlı olmasını sağlama
- Uygulama geliştirme zamanının kısalması
- Bilginin güncelliği: Havayolları, otel ve araba kiralama gibi çevrim-içi sistemlerinde çok önemlidir.
- Kullanımın yaygınlaştırılması.
- Veri güvenliği, gizliliğinin sağlanması, erişimin yetkilendirilmesi
- Standartların uygulanması

# VTYS Ne Zaman Kullanılmaz

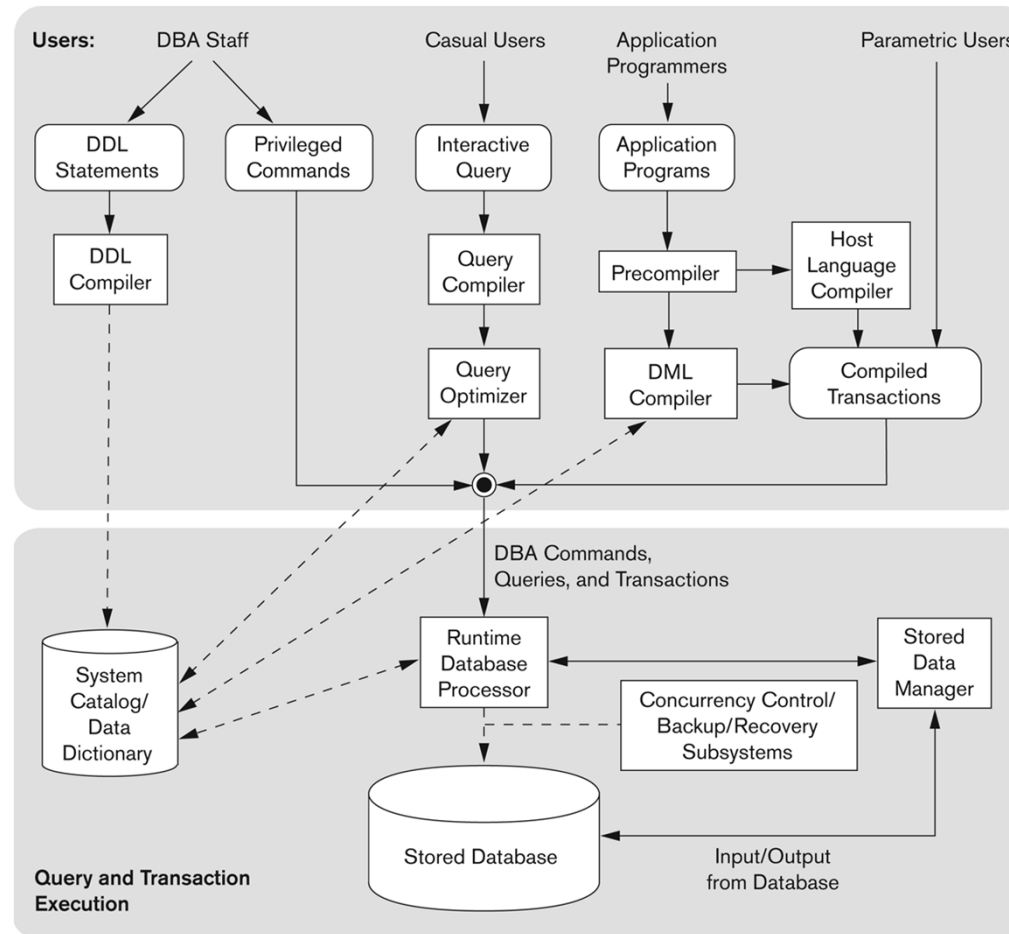


- VTYS kullanmanın maliyetleri
  - Yüksek yatırım maliyeti ve ek donanım gereksinimleri
  - Güvenlik, eşzamanlı erişim kontrolü, kurtarma ve bütünlük kısıtlamalarını sağlamanın getireceği yük
- VTYS'nin tamamen gereksiz olduğu durumlar
  - Veritabanı ve uygulamalarının basit, iyi tanımlanmış ve kolay kolay değişmez olduğu durumlar
  - VTYS'nin karşılayamayacağı gerçek zamanlı işlem gereksinimleri
  - Veriye birden fazla kullanıcının erişimesinin gerekmediği durumlar
- VTYS'nin yetersiz olduğu durumlar
  - Modelleme sınırlamalarından dolayı veritabanında karmaşık verilerin tanımlanamaması
  - VTYS'nin kullanıcıların özel işlem gereksinimlerini destekleyememesi



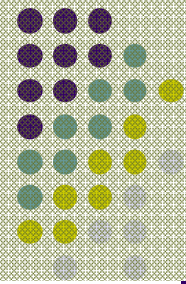
**Figure 1.1**  
A simplified database  
system environment.

# Typical DBMS Component Modules

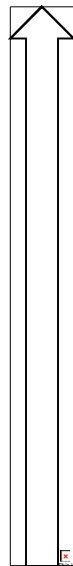
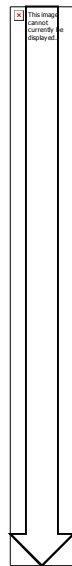


**Figure 2.3**  
Component modules of a DBMS and their interactions.

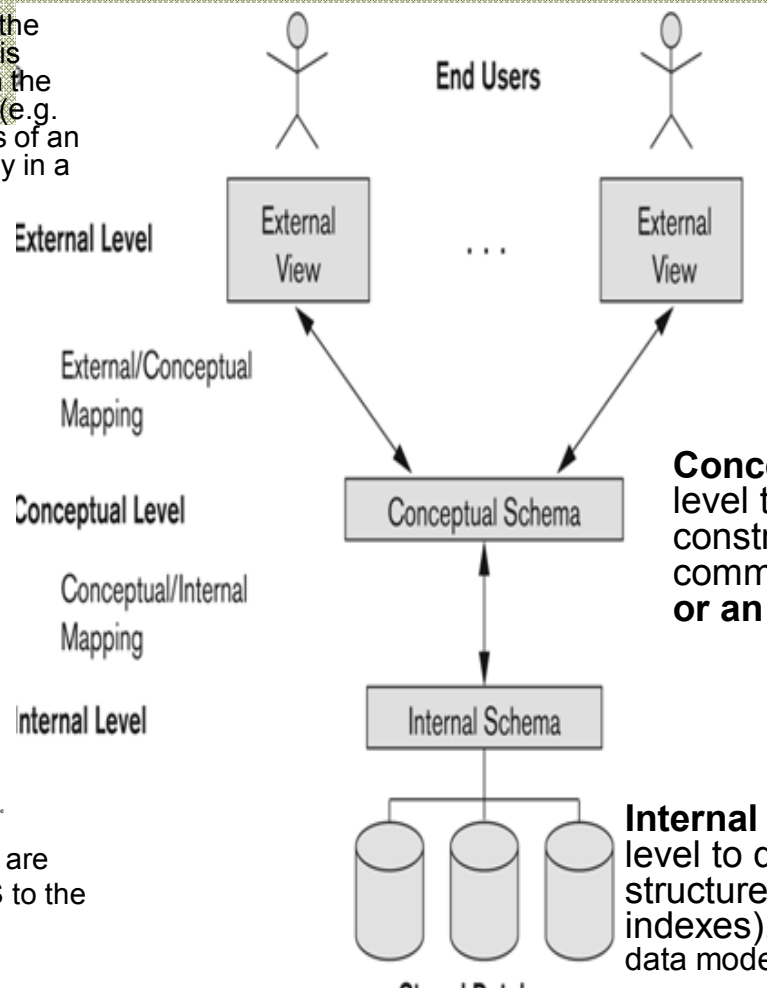
# The three-schema architecture



Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)



Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.



**External schemas** at the external level to describe the various user views. Usually uses the same data model as the conceptual schema.

**Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users. Uses a **conceptual** or an **implementation** data model.

**Internal schema** at the internal level to describe physical storage structures and access paths (e.g. indexes). Typically uses a **physical** data model.

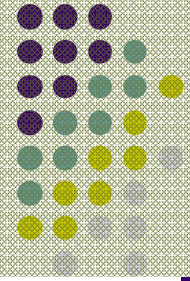
How data is used?

What data is inside?

How data is stored?

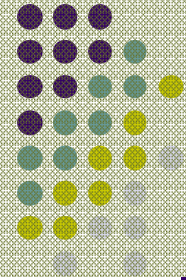
- Proposed to support DBMS characteristics of:
  - Program-data independence.**
  - Support of **multiple views** of the data.
- Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

# Data Independence

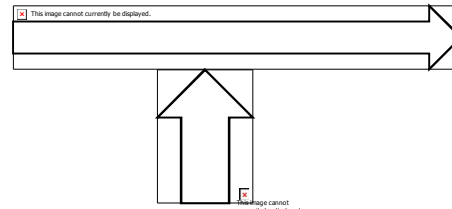


- **Physical Data Independence:**
  - The capacity to change the internal schema without having to change the conceptual schema. *(For example, the internal schema may be changed when certain file structures are reorganized or new indexes are created to improve database performance)*
- **Logical Data Independence:**
  - The capacity to change the conceptual schema without having to change the external schemas and their associated application programs.
- When a schema at a lower level is changed, only the **mappings** between this schema and higher-level schemas need to be changed in a DBMS that fully supports data independence.
- The higher-level schemas themselves are **unchanged**.
  - Hence, the application programs need not be changed since they refer to the external schemas.

# Physical Data Independence



Select Sname  
from STUDENT  
where GradYear=1997

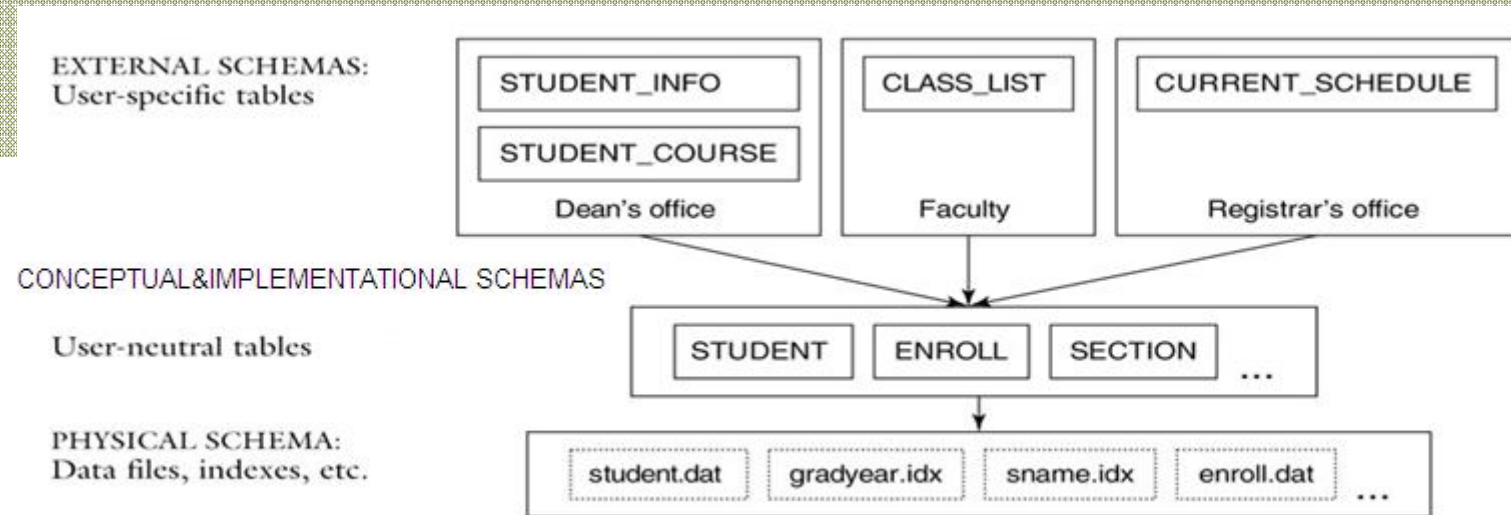


Database catalog  
(conceptual and  
physical schemas)

```
public static List<String> getStudents1997() {  
    List<String> result = new ArrayList<String>();  
    FileReader rdr = new FileReader("students.txt");  
    BufferedReader br = new BufferedReader(rdr);  
    String line = br.readLine();  
    while (line != null) {  
        String[] vals = line.split("\\t");  
        String gradyear = vals[2];  
        if (gradyear.equals("1997"))  
            result.add(vals[1]);  
        line = br.readLine();  
    }  
    return result;  
}
```

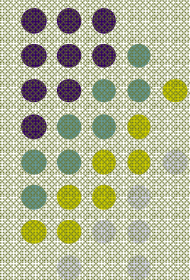
- Ease of use
- Query optimization
- Isolation from changes to physical schema

# Logical Data Independence

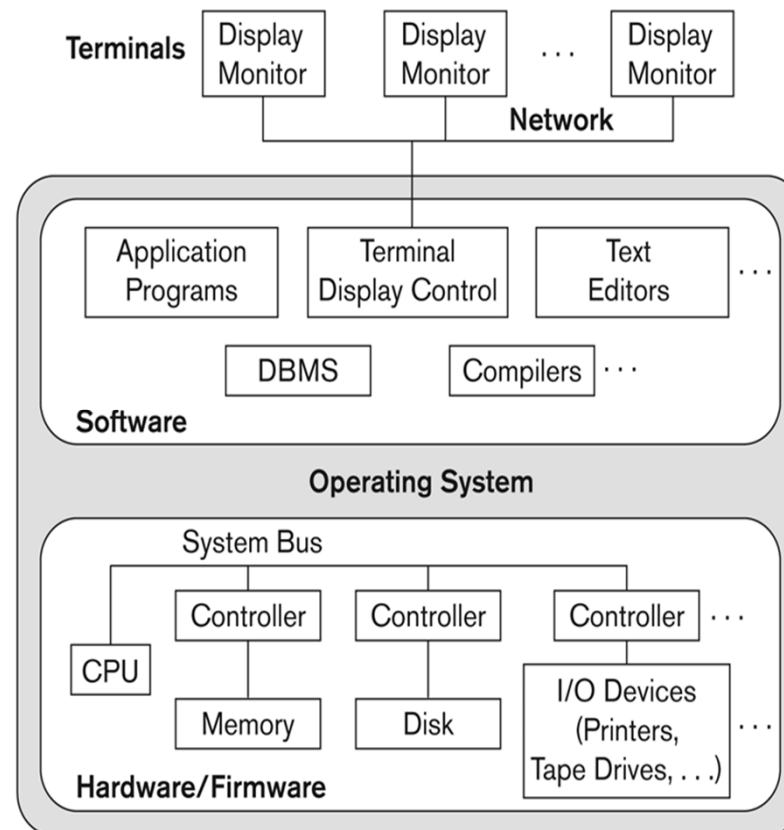


- In a database, if users have their own external schemas, then this db support logical data independence.
- Adv. of logical data dependence:
  - Customized external schema
  - Isolation from changes to conceptual schema
    - Suppose a change in conceptual schema: split STUDENT table into 2 tables: CURR\_STUDENT and ALUMNI. What happens to external schema?
  - Better security

# Centralized and Client-Server DBMS Architectures

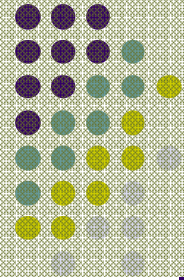


- Centralized DBMS:
  - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
  - User can still connect through a remote terminal – however, all processing is done at centralized site.



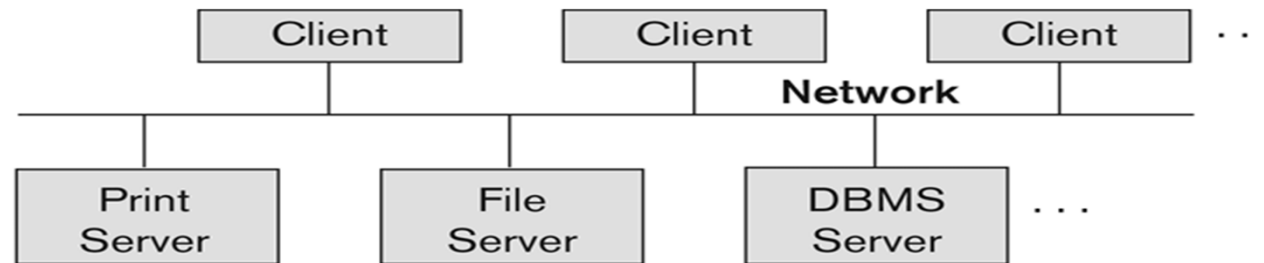
**Figure 2.4**  
A physical centralized architecture.

# Basic 2-tier Client-Server Architectures



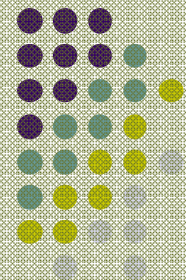
- Specialized Servers with Specialized functions: *Print server, File server, DBMS server, Web server, Email server...*
- Clients can access the specialized servers as needed

**Figure 2.5**  
Logical two-tier  
client/server  
architecture.



- Applications running on clients utilize an Application Program Interface (**API**) to access server databases via standard interface such as:
  - ODBC: Open Database Connectivity standard
  - JDBC: for Java programming access
- Client and server must install appropriate client module and server module software for ODBC or JDBC
- Variations of clients are possible (tightly vs loosely coupled): e.g., in some object DBMSs, more functionality is transferred to clients including data dictionary functions, optimization and recovery across multiple servers, etc.

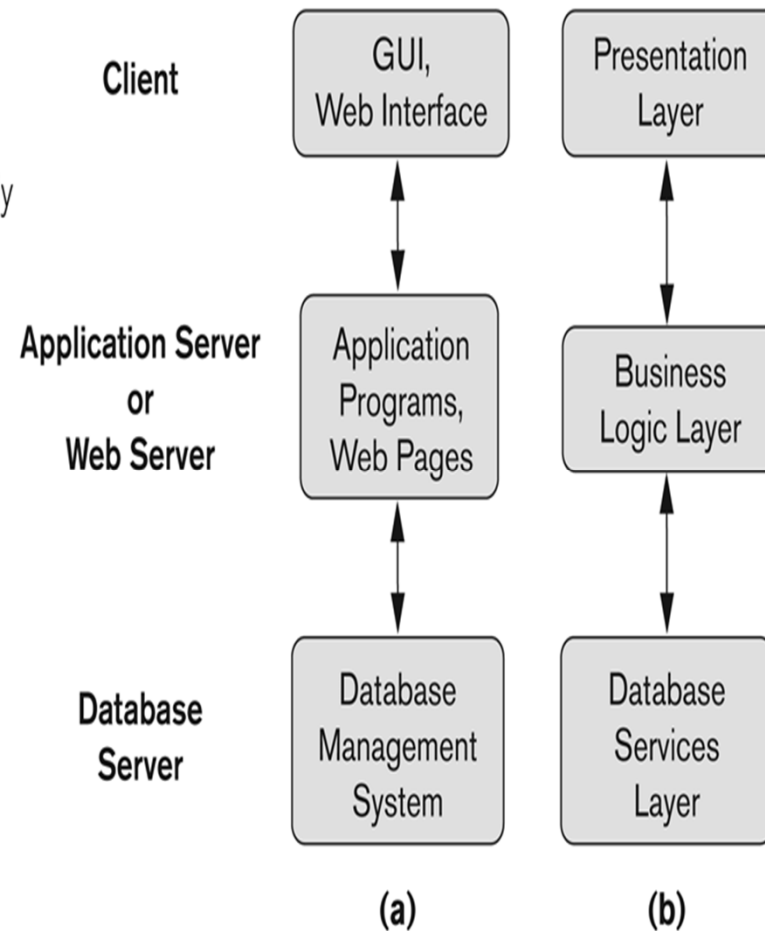
# Three Tier Client-Server Architecture



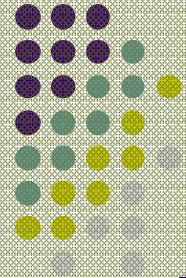
- Common for Web applications
- Three-tier Architecture Can Enhance Security:
  - Database server only accessible via middle tier (Application Server or Web Server)
  - Clients cannot directly access database server

**Figure 2.7**

Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.

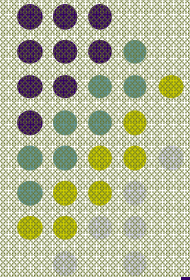


# Classification of DBMSs



- Based on the data model used
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented, Object-relational. XML
- Other classifications
  - Single-user (*typically used with personal computers or embedded dbms*)  
vs. multi-user (*most DBMSs*).
  - Centralized (*uses a single computer with one database*)  
vs. distributed (*uses multiple computers, multiple databases*)
- Homogeneous DDBMS
- Heterogeneous DDBMS
- Federated or Multidatabase Systems
- Distributed Database Systems have now come to be known as client-server based database systems because:
  - They do not support a totally distributed environment, but rather a set of database servers supporting a set of clients.

# Extension to DB capabilities, New Applications



- Database applications have evolved
  - Current trends: Information retrieval (IR), Web
- DB for Scientific applications
- Multimedia Databases: Image/videos
- Data mining
- Spatial/temporal DB
- IR: *deals with text in general, books, manuscripts, library-based articles*
- For more information about DBMS:  
[http://en.wikipedia.org/wiki/Database\\_management\\_system](http://en.wikipedia.org/wiki/Database_management_system)