



Veri Tabanı Yönetim Sistemleri

Veri tabanlarına Giriş

Dr. Öğr. Üyesi Elvan DUMAN

Bölüm 1

Veri tabanlarına Giriş

Ders 1: Hedefler / Kazanımlar



- ▶ Veri tabanı yaklaşımının neden doğduğunu açıklayabilme
- ▶ Veri tabanı ve VTYS (DBMS) kavramlarını ayırt edebilme
- ▶ DDL ve DML'nin farkını örnekle açıklayabilme
- ▶ Kontrollü erişim başlıklarını (güvenlik, bütünlük, eşzamanlılık, kurtarma) sınıflandırabilme

Ders 1: Ana Kavram Haritası



- Dosya tabanlı sistem : problemler
- Veri tabanı yaklaşımı : çözüm fikri
- DBMS : bileşenler ve roller
- DDL/DML : şema/veri ayrımı

Veritabanı Uygulama Örnekleri



- ▶ Bankacılık ve Finans Sistemleri
- ▶ Hastane ve Sağlık Yönetim Sistemleri
- ▶ Üniversite Öğrenci Bilgi Sistemleri
- ▶ E-Ticaret ve Online Alışveriş Siteleri
- ▶ Otel ve Rezervasyon Yönetim Sistemleri
- ▶ Üretim ve Tedarik Zinciri Yönetimi
- ▶ Kütüphane Yönetim Sistemleri
- ▶ Sosyal Medya Platformları

Dosya Tabanlı Sistemlerin Kısıtları



- ▶ **Bağımsız Veri Yönetimi:** Her uygulama kendi veri dosyasını oluşturur ve yönetir. Veri paylaşımı ve bütünlüğü sağlamak zordur.
- ▶ **Veri Tekrarı ve Tutarsızlık:** Aynı veriler birden fazla dosyada saklanabilir, bu da tutarsızlıklara yol açabilir.
- ▶ **Esneklik Eksikliği:** Yeni bir uygulama eklemek veya var olanı güncellemek zaman alıcıdır, çünkü her program kendi veri yapısını kullanır.
- ▶ **Düşük Güvenlik ve Kontrol:** Kullanıcı erişim denetimi sınırlıdır; yetkisiz erişim veya veri kaybı riskleri yüksektir.
- ▶ **Örnekler:** Raporlama sistemleri, basit müşteri kayıt uygulamaları, eski muhasebe sistemleri.

Veri Tabanı Yaklaşımı



► Neden Ortaya Çıktı?

- Geleneksel sistemlerde veri, uygulama programlarına gömülü olarak saklanıyordu.
- Veri tanımlamaları ayrı ve bağımsız bir yapıda değildi.
- Uygulama programları dışında doğrudan veriye erişim veya düzenleme imkanı yoktu.

► Sonuç:

- Veri tabanı kavramı doğdu.
- Veri Tabanı Yönetim Sistemi (VTYS) (DBMS) geliştirildi.

k ilişili

- Bir organizasyonun ihtiyaç duyduğu bilgiyi erişilebilir şekilde düzenleyen, mantıksal olarak ilişkili verilerin paylaşımlı olarak toplandığı sistemdir.

- **Schema:** Verinin tasarımı (tablolar/ilişkiler/kısıtlar)
- **Instance:** Verinin o anki içeriği (kayıtlar): Mantıksal olarak ilişkili verilerden oluşur. Organizasyonun varlıklarını, özelliklerini ve ilişkilerini kapsar.
- **Metadata:** Şema tanımlarının tutulduğu katalog. Program-veri bağımsızlığını sağlamak için verinin tanımını içerir.

Veri Bağımsızlığı (Program–Veri Bağımsızlığı)

Amaç: Veride yapılan değişiklikler, uygulama kodunu mümkün olduğunca az etkilesin.

1. **Sorun (dosya tabanlı sistemde)** Uygulama programı, verinin **nasıl tutulduğunu** (dosya yapısı/alan sırası/format) bilir. Veri yapısı değişince **program da değişmek zorunda kalır**.
2. **Veri tabanı yaklaşımı ne sağlar?** Uygulama, veriye **DBMS üzerinden** erişir. Uygulama “veri dosyasını” değil, **şemayı ve sorguyu** kullanır.
3. **Veri bağımsızlığı ne demek?**
 - Depolama değişebilir (indeks eklendi, dosya düzeni değişti, tablo başka diske taşındı) ama uygulamanın kullandığı sorgular **değişmeden** çalışır.
 - Şemada sınırlı değişiklikler (ör. yeni sütun ekleme) uygulamayı minimum etkiler.
4. **“Katalog (Metadata) neden gerekli?”** DBMS, “verinin yapısını” **katalogda** tutar: Tabloların adları, sütunlar, veri tipleri, birincil anahtar, yabancı anahtar, kısıtlar, Yetkiler, indeksler (varsa)
Uygulama bir sorgu gönderdiğinde DBMS: Yapıyı **katalogdan** okur. Veriye nasıl erişeceğini kendisi planlar. Böylece **verinin saklanma detayı uygulamadan gizlenir**.

Örnek: Öğrenci Bilgi Sistemi (Schema–Instance–Metadata)

Amaç: Veride yapılan değişiklikler, uygulama kodunu mümkün olduğunca az etkilesin.



1. Schema (Tasarım / Yapı): Değişmesi zor, “plan” gibi düşünülebilir.

Öğrenci(OgrNo, Ad, Soyad, Bolum),

Ders(DersKodu, DersAdi, AKTS),

Kayıt(OgrNo, DersKodu, Donem, Not) Not: “Kayıt” tablosu, Öğrenci ile Ders arasındaki ilişkiyi tutar.

2. Instance (O anki veri / içerik) Her dönem değişen “gerçek kayıtlar”.

- 2026 Bahar dönemi için: (221045067, “VT101”, “2026-Bahar”, 85), (221045067, “PRG201”, “2026-Bahar”, 90)

3. Metadata (Katalogda tutulan bilgiler)

- Tablolar ve sütunlar (isim + veri tipi)
- Kısıtlar: **PK**: Öğrenci.OgrNo, Ders.DersKodu,
- Not alanı 0–100 aralığında olmalı gibi kurallar

Veri Tabanı Yönetim Sistemi (VTYS)



► Tanım

- Kullanıcılara veri tabanlarını tanımlama, oluşturma, yönetme ve erişim kontrolü sağlama imkanı sunan bir yazılım sistemidir.

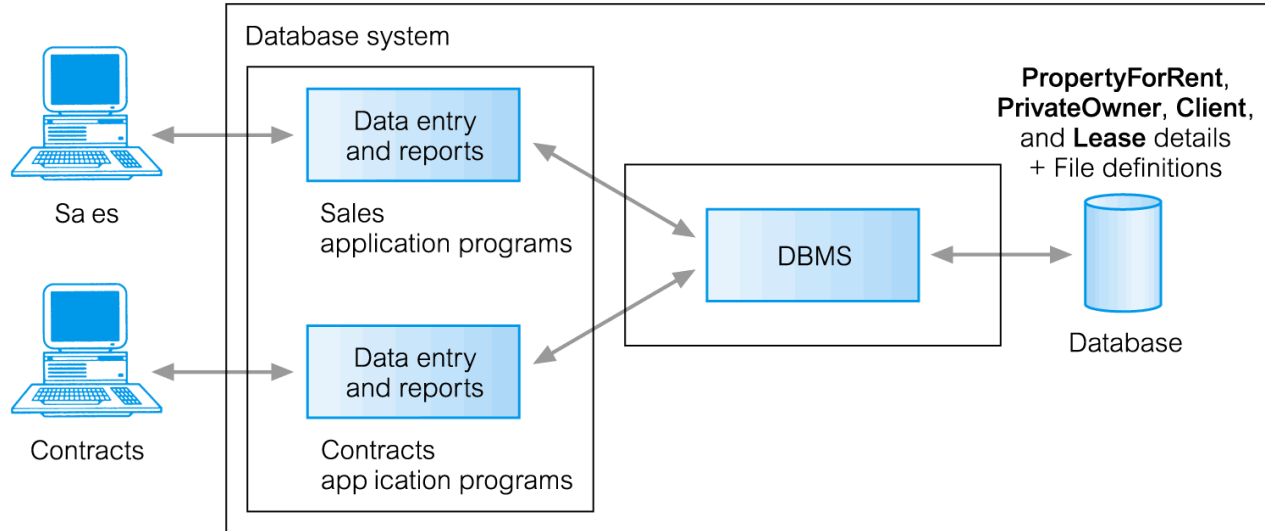
► Temel Fonksiyonları

- Veri saklama, güncelleme ve silme işlemlerini yönetmek.
- Kullanıcı yetkilendirmesi ile erişimi denetlemek
- Veri bütünlüğünü ve güvenliğini sağlamak.
- Büyük miktarda veriyi etkili bir şekilde yönetmek
- Aynı anda birden fazla kullanıcının veri üzerinde işlem yapmasını desteklemek.

► Veri Tabanı Uygulama Programı:

- Veritabanı ile etkileşimli çalışan ve uygun istekleri VTYS'ye ileten bilgisayar programıdır.
- Örneğin, müşteri bilgilerini sorgulayan bir e-ticaret uygulaması veya hastane yönetim sistemi.

Veri Tabanı Yönetim Sistemi (VTYS)



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)

Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)

Veri Tanımlama Dili (Data Definition Language - DDL)



► Amaç

- Veri tabanındaki veri yapılarının oluşturulmasını, değiştirilmesini ve silinmesini sağlar.

► Özellikler

- Veri tiplerini, tabloları ve ilişkileri tanımlar.
- Kısıtlamalar (constraints) ile veri bütünlüğünü korur.
- Tanımlamalar, sistem kataloğunda saklanır.

► Veri Tabanı Uygulama Programı:

- CREATE TABLE → Yeni bir tablo oluşturur.
- ALTER TABLE → Var olan bir tabloyu değiştirir.
- DROP TABLE → Bir tabloyu siler.

Veri İşleme Dili (Data Manipulation Language - DML)



► Amaç

- Veritabanındaki verileri eklemek, güncellemek, silmek ve sorgulamak için kullanılır.

► Özellikler

- Kullanıcıların veriye erişmesini ve işlem yapmasını sağlar.
- Verileri filtreleme ve analiz etme imkanı sunar.

► Veri Tabanı Uygulama Programı:

- SELECT : Verileri sorgulamak için kullanılır.
- INSERT : Yeni veri eklemek için kullanılır.
- UPDATE : Mevcut veriyi günceller.
- DELETE : Veri silmek için kullanılır.

Transaction (İşlem) Nedir?



- ▶ **Tanım:** Bir işi tamamlamak için yapılan **birden fazla veri tabanı adımının** tek bir bütün gibi çalıştırılmasıdır. Kullanıcı açısından “tek iş” görünür, ama içeride birden çok SQL çalışabilir.
- ▶ **Neden gerekli?** Çünkü veri tabanında işler genellikle **tek adımlı değildir**. Örnek: Para transferi, stok düşme + sipariş oluşturma, kayıt ekleme + ilişki ekleme.
- ▶ **Örnek senaryo:** Para transferi ($A \rightarrow B$ 100 TL)
 - A hesabından 100 düş
 - B hesabına 100 ekle
 - Bu iki adım **birlikte** başarılı olmalı.
 - Biri olursa diğeri olmazsa: veri hatalı olur (para kaybolur / iki kere yazılır).

Transaction (İşlem) Nedir?



- ▶ **Commit / Rollback**

- ▶ **Commit** : Transaction içindeki tüm adımlar başarılıysa, değişiklikleri **kalıcı hale getirir**.
- ▶ **Rollback** : Hata olursa, transaction içinde yapılan değişiklikleri **geri alır** (başlangıç durumuna döndürür).

- ▶ Sistem çökmesi / elektrik kesilmesi gibi durumlarda:
 - Commit edilmiş işlemler korunmalı
 - Yarım kalmış işlemler geri alınmalı

ACID

Transaction'ların güvenilir çalışması için 4 temel kural.



- 1. Atomicity (Bölünmezlik):** Transaction içindeki adımlar **tek paket** gibidir: Ya tamamen olur, ya hiç olmaz. Hata olursa: rollback ile eski hale dönülür.
- 2. Consistency (Tutarlılık):** Transaction bittiğinde veri tabanı **kurallara uygun** kalır. Örnek kurallar (kısıtlar):
 - Not 0–100 aralığında olmalı
 - Kayıt tablosundaki OgrNo mutlaka Öğrenci tablosunda olmalı
 - Bakiye negatif olamaz.
- 3. Isolation (Yalıtım)**
 - Aynı anda birden fazla kullanıcı işlem yaparken, işlemler birbirini **bozmamalı**.
 - Her transaction, sanki sistemde **tek başına çalışıyormuş** gibi doğru sonuç üretmeli.
 - **Örn:** Aynı stok ürününü iki kişinin aynı anda satın alması: stok “eksiye” düşmemeli.
- 4. Durability (Kalıcılık):** Commit edilen işlem, sistem çökse bile **kaybolmamalı**. Disk yazımı / log mekanizmaları ile garanti edilmeli.

Veri Tabanına Kontrollü Erişim



Bir veri tabanına erişim, çeşitli kontrol mekanizmaları ile güvence altına alınmalıdır. Bu sistemler şunları içerir.

- ▶ **Güvenlik Sistemi:** Yetkisiz erişimi engeller, Kullanıcı doğrulama (authentication) ve yetkilendirme (authorization) mekanizmalarını içerir.
- ▶ **Bütünlük Sistemi:** Veri bütünlüğünü korur ve hatalı verileri engeller. Kısıtlamalar (constraints) ile tutarsız veri girişini önler.
- ▶ **Eşzamanlılık Kontrol Sistemi:** Birden fazla kullanıcının aynı anda veri üzerinde işlem yapmasını düzenler. Çakışmaları ve veri kaybını önlemek için kilitleme (locking) mekanizmaları kullanır.
- ▶ **Kurtarma Kontrol Sistemi:** Sistem çökmesi veya hata durumlarında verilerin kurtarılmasını sağlar. Geri alma (rollback) ve tekrar işleme (redo) mekanizmalarını içerir.
- ▶ **Kullanıcı Erişim Kataloğu:** Kullanıcıların erişebileceği veri tabloları ve şemalar hakkında bilgi içerir. Yetkilere bağlı olarak hangi verilerin görüntülenebileceğini belirler.

Görünümler (Views)



► Tanım

- Her kullanıcıya, veritabanında kendisine özel bir veri görünümü sağlar.
- Fiziksel veri saklama yapısını değiştirmeden, kullanıcıya belirli veri alt kümelerini gösterir.

► Özellikler

- Bir görünüm, veritabanının belirli tablolarından seçilmiş bir alt kümedir.
- Kullanıcıların yalnızca yetkili oldukları verilere erişmesini sağlar.
- Hassas verileri gizleyerek güvenliği artırır.

► Avantajları

- **Güvenlik:** Kullanıcıya sadece gerekli bilgileri sunar, hassas verileri gizler.
- **Kolaylık:** Sık kullanılan sorgular için basitleştirilmiş bir yapı sunar.
- **Bağımsızlık:** Fiziksel veri değişikliklerinden kullanıcıları etkilenmeden korur.
- Detaylar ilerleyen derslerde (SQL – VIEW)

Veri Tabanı Yönetim Sistemi (VTYS) Bileşenleri



► Donanım

- VTYS'nin çalıştığı fiziksel altyapıdır. Bir kişisel PC'den büyük ölçekli bilgisayar ağlarına kadar değişebilir.
- Sunucular, depolama cihazları ve ağ bileşenleri içerebilir.

► Yazılım

- VTYS'nin çalışmasını sağlayan sistemlerdir. VTYS'nin çalıştığı işletim sistemleri, ağ yazılımları ve uygulama programları yazılım kategorisindedir.

► Veri

- Organizasyon tarafından kullanılan ve tanımlanan bilgidir.
- Verinin organizasyon içindeki yapısına şema (schema) denir.
- Şema, verinin nasıl düzenlendiğini, hangi ilişkilerde olduğunu ve nasıl erişileceğini belirler.

► İşlemler

- VTYS'nin verimli çalışması için uygulanan kurallar ve işlemlerdir.

► İnsanlar

- Veri tabanı sistemlerini yöneten, geliştiren ve kullanan kişilerden (rollerden) oluşur.

Veri Tabanındaki Roller



1. Veri Yöneticisi (Data Administrator - DA)

- Organizasyonun genel veri yönetim politikalarını belirler.
- Veri güvenliği, veri standartları ve veri paylaşımı gibi konulardan sorumludur.
- Verinin organizasyon içinde nasıl yönetileceğini belirler ve veri yönetimi stratejilerini oluşturur.

2. Veri Tabanı Yöneticisi (Database Administrator - DBA)

- Veri tabanının kurulumu, yönetimi ve bakımından sorumludur.
- Yedekleme, geri yükleme, güvenlik ve performans optimizasyonu işlemlerini yürütür.
- Kullanıcı yetkilendirme ve erişim kontrollerini sağlar.

3. Veri Tabanı Tasarımcıları (Database Designers)

- **Mantıksal Tasarımcılar:** Verinin organizasyonel ihtiyaçlara göre nasıl yapılandırılacağını belirler.
- **Fiziksel Tasarımcılar:** Verinin fiziksel olarak nasıl saklanacağını, indeksleme ve depolama stratejilerini oluşturur.

Veri Tabanındaki Roller



4. Uygulama Programcıları (Application Programmers)

- Veri tabanı ile çalışan yazılım ve uygulamaları geliştirir.
- VTYS'yi kullanarak kullanıcıların veri girişi yapmasını ve sorgulama işlemlerini gerçekleştirmesini sağlar.
- SQL sorguları, prosedürler ve entegrasyon işlemleri gibi konularda çalışır.

5. Son Kullanıcılar (End Users)

- VTYS'ye erişerek veri sorgulama, giriş yapma ve raporlama işlemlerini gerçekleştirir.
- İki tür son kullanıcı vardır:
 - **Güçlü Kullanıcılar (Power Users):** Gelişmiş sorgular ve analizler yapabilen deneyimli kullanıcılar.
 - **Normal Kullanıcılar (Casual Users):** Önceden tanımlanmış sorgular ve uygulamalar aracılığıyla veri tabanını kullanan kişiler.

Veri Tabanındaki Roller – Örnek Sorumluluklar



► Database Administrator (DBA)

- Kullanıcı/rol oluşturma, yetkilendirme (grant/revoke)
- Yedekleme planı, restore testi
- Çökme sonrası kurtarma prosedürü (recovery)
- Performans ayarı: indeksleme, bakım, izleme
- Güvenlik politikalarının uygulanması (erişim, audit)

► Data Administrator

- “Hangi veri toplanacak, kim hangi veriyi görebilir?” veri politikası
- Kurumsal veri standartları (isimlendirme, veri tanımı, veri sözlüğü)
- Veri kalitesi kuralları (zorunlu alanlar, doğrulama ilkeleri)
- Gizlilik/saklama süreleri gibi yönetsel kurallar
- Departmanlar arası veri tutarlılığı ve sahiplik (data ownership)

Veri Tabanındaki Roller – Örnek Sorumluluklar



► Database Designer

- ER diyagramı / kavramsal model tasarlama
- Tablolar-iliskiler-kardinalite belirleme
- PK/FK seçimi, normalizasyon kararları
- Kısıtların tasarımı (unique, check, referential integrity)
- View'lar için mantıksal tasarım (kim neyi görmeli)

► Uygulama Programcısı kartları

- CRUD ekranları ve iş kurallarını kodlama
- DB bağlantısı, sorgu yazma, hata yönetimi
- Transaction kullanımı: commit/rollback
- Input doğrulama (format, boş geçme, limit)
- Yetki kontrolünü uygulama katmanında doğru uygulama (role-based akış)

Veri Tabanındaki Roller – Örnek Sorumluluklar



► End User kartları (son kullanıcı)

- Sistemi kullanarak veri girme/güncelleme (hasta kaydı, öğrenci kaydı, sipariş)
- Rapor alma / sorgu ekranlarını kullanma (hazır raporlar)
- Hatalı veri bildirimi ve geri bildirim (kaliteye katkı)
- Süreç adımlarını takip etme (onay, talep, kayıt işlemi)
- “Kural dışı durum”ları işaretleme (ör. eksik bilgi, uyumsuzluk)

Veri Tabanı Sistemleri Tarihi



1. İlk Jenerasyon (1960'lar - 1970'ler)

- Hiyerarşik Veri Tabanı Modeli (Hierarchical Model)

- Veriler, ağaç yapısında (parent-child ilişkisiyle) düzenlenmiştir.
- Örnek: IBM'in IMS (Information Management System) veri tabanı.

- Ağ (Network) Veri Tabanı Modeli

- Veriler, graf yapısı (kavşak noktaları ve bağlantılar) ile saklanır.
- Daha esnek veri ilişkileri sağlar, ancak karmaşıklığı yönetmek zordur.
- Örnek: IDMS (Integrated Database Management System).

Veri Tabanı Sistemleri Tarihi



2. İkinci Jenerasyon (1970'ler - Günümüz)

- **İlişkisel Veri Tabanı Modeli (Relational Model)**
 - Edgar F. Codd tarafından önerilmiştir.
 - Veriler, tablolar (relations) içinde saklanır ve ilişkiler matematiksel kurallara dayanır.
 - **SQL (Structured Query Language) geliştirilerek standart hale getirilmiştir.**
 - **Örnek:** IBM DB2, Oracle, MySQL, PostgreSQL, Microsoft SQL Server.
 - Güçlü veri bütünlüğü, tutarlılık ve güvenlik sağlar.
 - Büyük ölçekli sistemlerde hala en güvenilir seçeneklerden biridir.
- İlişkisel Veri Tabanı Yönetim Sistemleri (RDBMS) hala en yaygın kullanılan veri tabanı modelidir.

Veri Tabanı Sistemleri Tarihi



3. Üçüncü Jenerasyon (1990'lar - Günümüz)

- Nesne-İlişkisel Veri Tabanı Modeli (Object-Relational Model)

- İlişkisel modelin yeteneklerini genişleterek nesne yönelimli programlamayı destekler.
- Karmaşık veri tipleri (JSON, XML, resimler, videolar) saklanabilir.
- **Örnek: PostgreSQL, Oracle Object-Relational DB.**
- **Örnek:** IBM DB2, Oracle, MySQL, PostgreSQL, Microsoft SQL Server.

- Nesne Tabanlı Veri Tabanı Modeli (Object-Oriented Model)

- Veriler tamamen nesne yönelimli paradigmalara göre saklanır ve işlenir.
- Java, C++ gibi programlama dilleriyle daha iyi entegre olur.
- **Örnek:** db4o, ObjectDB, Versant.

Veri Tabanı Sistemleri Tarihi



4. Dördüncü Jenerasyon (2000'ler - Günümüz)

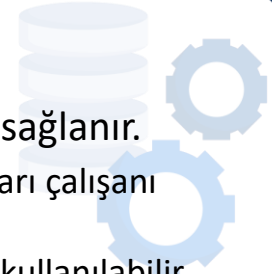
- **NoSQL Veri Tabanları (SQL dışındaki sistemler)**
- Örnekler: MongoDB, Cassandra, Redis, DynamoDB.
- **Özellikler**
 - Yapılandırılmamış veya yarı yapılandırılmış verileri saklar.
 - Graf tabanlı, belge tabanlı, anahtar-değer tabanlı gibi farklı türleri vardır.
 - Büyük ölçekli, hızlı veri işleme gerektiren sistemlerde kullanılır.
- **NewSQL Veri Tabanları (SQL + NoSQL özelliklerini birleştiren yeni nesil sistemler)**
 - Örnekler: Google Spanner, CockroachDB, MemSQL
- **Özellikler**
 - Geleneksel SQL yeteneklerini korurken yüksek ölçeklenebilirlik sunar.
 - Dağıtık veri yönetimi yapısı ile büyük veri işleme yeteneğine sahiptir. Örn. Büyük veri, bulut bilişim

Veri Tabanı Yönetim Sistemi (VTYS) Avantajları



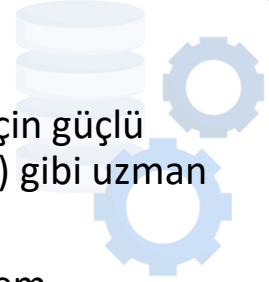
- ▶ **Veri Tekrarının Kontrolü:** Aynı verinin birden fazla yerde saklanmasını önler. Gereksiz veri kopyası azalır. Depolama maliyetini düşürür.
- ▶ **Veri Tekrarı ve Tutarsızlık:** Aynı veriler birden fazla dosyada saklanabilir, bu da tutarsızlıklara yol açabilir.
- ▶ **Verinin Paylaşımı:** Çok kullanıcıli sistemlerde, yetkilendirme dahilinde veriye ortak erişim sağlanır.
 - Ağ üzerinden dağıtık veri tabanları ile erişim sağlanabilir.
 - Farklı departmanlar veya şubeler aynı veriye güvenli bir şekilde ulaşabilir.
- ▶ **Veri Bütünlüğünün Geliştirilmesi:** VTYS, veriler üzerinde kurallar (constraints) tanımlayarak hatalı girişleri engeller.
 - Örneğin, bir müşteri kaydının telefon numarası boş bırakılamaz gibi kurallar koyulabilir.
 - Anahtar kısıtlamaları (Primary Key, Foreign Key) ile veri ilişkileri güvence altına alınır.

Veri Tabanı Yönetim Sistemi (VTYS) Avantajları



- ▶ **Güvenliğin Geliştirilmesi:** Kullanıcı yetkilendirme ve erişim kontrolleri ile veri güvenliği sağlanır.
 - Örneğin, bir muhasebe departmanı çalışanı sadece finansal verileri görebilirken, insan kaynakları çalışanı personel bilgilerine erişebilir.
 - Hassas veriler için şifreleme (encryption) ve kimlik doğrulama (authentication) mekanizmaları kullanılabilir.
- ▶ **Standartların Uygulanması:** SQL gibi standart diller sayesinde veri tabanı yönetimi evrensel hale gelir.
 - **Şirket içi veri yönetimi politikaları daha düzenli uygulanabilir.**
 - Organizasyon içinde veri paylaşımında tutarlılık sağlanır.

Veri Tabanı Yönetim Sistemi (VTYS) Dezavantajları



- ▶ **Yüksek Maliyet:** Kurulum, bakım ve lisans maliyetleri yüksektir. Büyük ölçekli VTYS'ler için güçlü donanım ve altyapı gerektirir. Özel geliştirme ve yönetim için veritabanı yöneticisi (DBA) gibi uzman personele ihtiyaç duyulur.
- ▶ **Karmaşıklık:** VTYS'ler, geleneksel dosya tabanlı sistemlere kıyasla daha karmaşıktır. Sistem yöneticileri ve geliştiriciler için öğrenme eğrisi yüksektir. Kurallar, yetkilendirme ve veri bütünlüğü işlemleri yönetim açısından ekstra bilgi gerektirir.
- ▶ **Bağımlılık (Bağılılık Sorunu):** VTYS'ye bağımlılık arttıkça, sistem arızaları veya erişim problemleri büyük kayıplara neden olabilir. Merkezi bir sistem olması nedeniyle, sunucu çökmesi durumunda tüm sistem etkilenebilir.
- ▶ **Performans Sorunları (Ölçeklenebilirlik):** Çok büyük veri kümeleri ve çoklu kullanıcı işlemleri sistemin performansını düşürebilir. Veritabanı iyi tasarlanmazsa, sorgular yavaş çalışabilir ve işlem süreleri uzayabilir.
- ▶ **Güvenlik Riskleri:** Siber saldırılar, yetkisiz erişimler ve veri hırsızlığı gibi riskler içerir.
- ▶ **Yedekleme ve Kurtarma Zorlukları:** Veri kaybı yaşandığında, sistemin geri yüklenmesi uzun sürebilir.

Bölüm Sonu - Teşekkürler

Dr. Öğr. Üyesi Elvan DUMAN – eduman@mehmetakif.edu.tr