Veritabanı Yönetim Sistemleri
Bölüm 1: Giriş

Database System Concepts, 7th Ed.

Bölüm İçeriği

- Veritabanı-Sistem Uygulamaları
- Veritabanı Sistemlerinin Amacı
- Verilerin Görüntülenmesi
- Veritabanı Dilleri
- Veri tabanı tasarımı
- Veritabanı Motoru
- Veritabanı Mimarisi
- Veritabanı Kullanıcıları ve Yöneticileri
- Veritabanı Sistemlerinin Tarihçesi

Veritabanı Sistemleri

- Bir **veri tabanı yönetim sistemi (DBMS)**, birbiriyle ilişkili verilerden ve bu verilere erişim sağlayan bir dizi programdan oluşan bir koleksiyondur.
- Genellikle **veri tabanı** olarak adlandırılan veri koleksiyonu, bir işletme/kurum ile ilgili bilgileri içerir.
- Bir DBMS'nin temel amacı, veritabanı bilgilerini hem **kullanışlı** hem de **verimli** bir şekilde depolamak ve erişmek için bir yol sağlamaktır.
- Veritabanı sistemleri aşağıdaki gibi veri koleksiyonlarını yönetmek için kullanılır:
  - Son derece değerli
  - Nispeten büyük
  - Genellikle aynı anda birden fazla kullanıcı ve uygulama tarafından erişilebilen.

Veritabanı Sistemleri

- İlk veritabanı sistemleri 1960'larda ticari verilerin bilgisayarla yönetilmesine yanıt olarak ortaya çıktı.
  - İlk veritabanı uygulamaları yalnızca basit, kesin olarak biçimlendirilmiş verileri içeriyordu. Örneğin öğrenci veri tabanı : Ogno, ad, soyad … gibi
- Modern bir veritabanı sistemi, büyük ve karmaşık bir veri koleksiyonunu yönetmek için kullanılan karmaşık bir yazılım sistemidir.
  - Günümüzde veritabanı uygulamaları karmaşık ilişkilere sahip verileri içerebilmektedir. Sosyal ağlara ait bir veritabanı düşünün metin, video, resim ve diğer kullanıcılarla olan bağlantılar gibi karmaşık yapılar içermektedir.

Veritabanı Uygulama Örnekleri

- **Kurumsal Bilgiler**
  - **Satışlar**: müşteriler, ürünler, satın almalar
  - **Muhasebe**: ödemeler, makbuzlar, varlıklar
  - **İnsan Kaynakları**: Çalışanlar, maaşlar, bordro vergileri hakkında bilgiler.
- **Üretim**: üretim, envanter, siparişler, tedarik zinciri yönetimi.
- **Bankacılık ve Finans**
  - **Bankacılık**: müşteri bilgileri, hesaplar, krediler ve bankacılık işlemleri.
  - **Kredi kartı işlemleri**: Kredi kartlarıyla yapılan alışverişler ve aylık ekstrelerin oluşturulması için.
  - **Finans**: Hisse senetleri ve tahviller gibi finansal araçların tutulması, satışı ve satın alınmasıyla ilgili bilgilerin depolanması; ayrıca gerçek zamanlı piyasa verilerinin depolanması için müşterilerin çevrimiçi ticaret yapmasını ve firmanın otomatik ticaret yapmasını sağlayacak veriler

Veritabanı Uygulama Örnekleri

- **Üniversiteler**: Öğrenci bilgileri, ders kayıtları ve notlar için (ek olarak insan kaynakları ve muhasebe gibi standart kurumsal bilgiler)
- **Havayolları**: Rezervasyon ve uçuş program bilgileri için. Havayolları, veri tabanlarını coğrafi olarak dağıtılmış bir şekilde ilk kullananlar arasındaydı.
- **Telekomünikasyon**: çağrıların, mesajların ve veri kullanımının kayıtları, aylık faturaların oluşturulması, arama kartlarındaki bakiyelerin takibi
- **Doküman veritabanları**: Yeni makalelerin, patentlerin, yayınlanmış araştırma makalelerinin vb. koleksiyonlarını tutmak için
- **Navigasyon sistemleri**: Yolların, tren sistemlerinin, otobüslerin vb. kesin rotaları ile birlikte çeşitli ilgi çekici yerlerin konumlarını tutmak için

Veritabanı Uygulama Örnekleri (Devam)

- **Web tabanlı servisler**
  - **Sosyal medya**: Kullanıcıların kayıtlarını tutmak için, kullanıcılar arasındaki bağlantılar (arkadaş / takip bilgileri gibi), kullanıcılar tarafından yapılan gönderiler, kullanıcılar hakkında beğenme bilgileri gönderiler, vs.
  - **Online perakendeciler**: Herhangi bir perakendecide olduğu gibi satış verilerinin ve siparişlerin kayıtlarını tutmak için, aynı zamanda bir kullanıcının ürün görüntülemelerini, arama terimlerini vb. izlemek için, o kullanıcıya önerilecek en iyi ürünleri belirlemek amacıyla.
  - **Online reklamlar**: Hedef kitle reklamları, ürün önerileri, haber makaleleri vb. sağlamak için tıklama geçmişinin kayıtlarını tutmak için. İnsanlar her web araması yaptıklarında, çevrimiçi bir satın alma işlemi gerçekleştirdiklerinde veya bir sosyal ağ sitesine eriştiklerinde bu tür veri tabanlarına erişirler.

Veritabanı Uygulamaları

Genel olarak konuşursak, veritabanlarının kullanıldığı **iki mod** vardır:

- **İlk mod**, çok sayıda kullanıcının veritabanını kullandığı, her kullanıcının nispeten küçük miktarlarda veri aldığı ve küçük güncellemeler yaptığı çevrimiçi işlem sürecini desteklemektir . Bu, veritabanı uygulamalarının kullanıcılarının büyük çoğunluğu için birincil kullanım modudur.

Veritabanı Uygulamaları

- **İkinci mod**, İkinci yöntem ise veri analitiğini desteklemek, yani sonuçlar çıkarmak, kuralları veya karar prosedürlerini çıkarmak ve daha sonra iş kararlarını yönlendirmek için kullanılan verileri işlemektir.
  - Örneğin, bankaların bir kredi başvurusunda bulunan kişiye kredi verip vermemeye karar vermesi gerekir, çevrimiçi reklam verenlerin belirli bir kullanıcıya hangi reklamı göstereceğine karar vermesi gerekir.
  - Bu görevler iki adımda ele alınır. İlk olarak, veri analizi teknikleri verilerden kuralları ve kalıpları otomatik olarak keşfetmeye ve tahmine dayalı modeller oluşturmaya çalışır.
  - Bu modeller, bireylerin girdi niteliklerini ("özellikler(features)") alır ve bir krediyi geri ödeme olasılığı veya bir reklama tıklama olasılığı gibi tahminler üretir ve bunlar daha sonra iş kararını vermek için kullanılır.

Veritabanı Sistemlerinin Amacı

Önceleri, veritabanı uygulamaları doğrudan **dosya sistemlerinin** üzerine kuruluydu ve bu da bazı deazavantajlara yol açtı:

- **Veri fazlalığı ve tutarsızlık:** Veriler birden fazla dosya formatında depolanır ve bu da bilgilerin farklı dosyalarda tekrarlanmasına neden olur.
- **Verilere erişimde zorluk**
  - Her yeni görevi gerçekleştirmek için yeni bir program yazmak gerekiyordu
- **Veri izolasyonu**
  - Veriler çeşitli dosyalara dağılmış olduğundan ve dosyalar farklı formatlarda olabileceğinden, uygun verileri geri getirecek yeni uygulama programları yazmak zordur
- **Bütünlük sorunları**
  - Bütünlük kısıtlamaları (örneğin, hesap bakiyesi > 0) açıkça belirtilmek yerine program kodunda " gömülü " haldeydi
  - Yeni kısıtlamalar eklemek veya mevcut kısıtlamaları değiştirmek zordu.

Veritabanı Sistemlerinin Amacı (Devam)

- **Güncellemelerin atomikliği**
  - Kısmi güncellemeler yapıldığında veritabanını tutarsız bir durumda bırakabilir
  - Örnek: Bir hesaptan diğerine para transferi ya tamamlanmalı ya da hiç gerçekleşmemelidir
- **Güvenlik sorunları**
  - Veritabanı sisteminin her kullanıcısı tüm verilere erişememelidir. Örneğin, bir üniversitede, bordro personelinin veritabanının yalnızca mali bilgileri içeren kısmını görmesi gerekir. Akademik kayıtlarla ilgili bilgilere erişmeleri gerekmez.
  - Ancak uygulama programları dosya işleme sistemine geçici bir şekilde eklendiğinden, bu tür güvenlik kısıtlamalarını uygulamak zordur.

Veritabanı Sistemlerinin Amacı (Devam)

- **Birden fazla kullanıcının eş zamanlı erişimi**
  - Sistemin genel performansı ve daha hızlı yanıt verilmesi adına birçok sistem, birden fazla kullanıcının verileri aynı anda güncellemesine izin verir.
    - Günümüzde büyük internet alışveriş siteleri, alışveriş yapanlar tarafından verilerine günde milyonlarca erişim sağlayabilmektedir.
  - Kontrolsüz eşzamanlı erişimler tutarsızlıklara yol açabilir.
    - Bu olasılığa karşı korunmak için sistem bir çeşit denetim sağlamalıdır. Ancak denetimin sağlanması zordur çünkü verilere daha önce koordine edilmemiş birçok farklı uygulama programı tarafından erişilebilir.

Bu zorluklar, 1960'lar ve 1970'lerde hem veritabanı sistemlerinin ilk gelişimini hem de dosya tabanlı uygulamaların veritabanı sistemlerine geçişini tetiklemiştir.

Veritabanı sistemleri yukarıdaki sorunların tümüne çözüm sunar.

Üniversite Veritabanı Örneği

- Kavramları açıklamak için örnek bir üniversite veri tabanı kullanacağız.
- Veriler aşağıdakilerle ilgili bilgilerden oluşur:
  - Öğrenciler
  - Eğitmenler
  - Sınıflar
- Uygulama programı örnekleri:
  - Yeni öğrenciler, eğitmenler ve dersler ekleme
  - Öğrencileri derslere kaydetme ve sınıf listeleri oluşturmak
  - Öğrencilere not vermek, not ortalamalarını hesaplamak ve transkriptleri oluşturmak

Verilerin Görüntülenmesi

- Bir **veritabanı sistemi**, birbiriyle ilişkili verilerden ve kullanıcıların bu verilere erişmesine ve bunları değiştirmesine olanak tanıyan bir dizi programdan oluşan bir koleksiyondur.
- Bir veritabanı sisteminin temel amacı, kullanıcılara verilerin soyut bir görünümünü sağlamaktır.
  - Yani sistem, verilerin nasıl depolandığı ve muhafaza edildiğine dair belirli ayrıntıları gizler.
  - Veri soyutlama: Veritabanındaki verileri temsil etmek için veri yapılarının karmaşıklığını, çeşitli veri soyutlama düzeyleri aracılığıyla kullanıcılardan gizlemektir.

Veri Modelleri

- Bir veritabanı yapısının temelinde **veri modeli** bulunur. **Veri modeli;** verileri, veri ilişkilerini, veri semantiğini ve tutarlılık kısıtlamalarını açıklamaya yönelik kavramsal araçların bir koleksiyonudur.
  - **Veri semantiği**, veritabanında depolanan verilerin anlamını, yorumunu ve bağlamını ifade eder. Verinin sadece biçimi veya yapısından ziyade, verinin arkasında yatan amaçlanan anlamı anlamaya odaklanır. Semantik (Anlambilim), verilerin yalnızca doğru değil, aynı zamanda amaçlanan uygulama veya iş bağlamı için anlamlı ve kullanışlı olmasını sağlamaya yardımcı olur.
  - Örneğin

Veri Semantiği

- Örneğin bir okul yönetim sistemi için aşağıdaki tabloları ele alalım.

1. **Tablo Tanımları:**

Tablo 1: Öğrenciler

ogrenci_id	isim	sinif
101	Meryem	3
102	Ali	5

Tablo 2: Dersler

ders_id	ders_adi
1	Matematik
2	Fen Bilgisi

Tablo 3: Notlar

ogrenci_id	ders_id	not
101	1	89
101	2	75
102	1	95

Veri Semantiği

- Örneğe yapısal düzeyde bakıldığında, tablolar düzenlenmiştir ve öğrenciler, konular ve notlar hakkında veriler depolanmıştır.
- Ancak, semantik olmadan ogrenci_id, ders_id, ve not sütunlarının gerçek dünyada neyi temsil ettiğini, bu tablolar arasındaki ilişkinin ne olduğunu bilemezsiniz.
- Veri semantiği, bu sütunların anlamını ve tabloların nasıl ilişkili olduğunu tanımlayarak devreye girer ve verileri anlamlı ve kullanışlı hale getirir.

Veri Semantiği

- Örnekteki tablolarla ilgili veri semantiği bize şunları açıklar:
  - **Birincil Anahtar (PK) ve Yabancı Anahtar (FK) İlişkileri**: ogrenci_id, Notlar tablosunda Öğrenciler tablosuna referans veren bir yabancı anahtardır. Bu semantik ilişki, notlar tablosundaki bir satırın belirli bir öğrenciyle ilişkili olduğunu belirtir.
    - ders_id, Notlar tablosunda Dersler tablosuna referans verir. Bu da notların hangi dersle ilgili olduğunu ifade eder.
    - Bu semantik ilişkiler olmadan, not sütunundaki verilerin hangi öğrenci ve dersle bağlantılı olduğu belirsiz olurdu.
  - **Sütunların semantik yorumu**:
    - ogrenci_id: Bir öğrenciye özgü olan kimlik numarasıdır.
    - ders_id: Bir derse özgü olan kimlik numarasıdır.
    - not: Bir öğrencinin belirli bir dersten aldığı notu temsil eder ve bu not, veritabanındaki bağlam içinde bir anlam taşır (örneğin, 100 üzerinden).

Veri Modelleri

Veri modelleri dört farklı kategoriye ayrılır:

- İlişkisel model
- Varlık-İlişki veri modeli (temel olarak veritabanı tasarımı için)
- Nesne tabanlı veri modelleri (Nesne yönelimli ve Nesne ilişkisel)
- Yarı yapılandırılmış veri modeli (JSON, XML)
- Diğer eski modeller:
  - Ağ modeli
  - Hiyerarşik model

İlişkisel Model

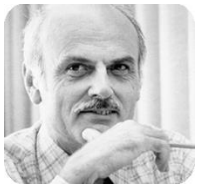
- İlişkisel model hem verileri hem de bu veriler arasındaki ilişkileri temsil etmek için bir tablo koleksiyonu kullanır. Her tablonun birden fazla sütunu vardır ve her sütunun benzersiz bir adı vardır. Tablolar aynı zamanda **ilişkiler** (**relation**) olarak da bilinir .
- İlişkisel modeldeki tablo verileri örneği

Sütunlar

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Satırlar

(a) The *instructor* table



Ted Codd
Turing Ödülü 1981

İlişkisel Model

- İlişkisel model, kayıt tabanlı (record-based) modelin bir örneğidir. Kayıt tabanlı modeller, veritabanının çeşitli türlerde sabit formatlı kayıtlar halinde yapılandırılmış olması nedeniyle bu şekilde adlandırılmıştır.
- Her tablo belirli bir türdeki kayıtları içerir. Her kayıt türü sabit sayıda alan (field) veya nitelik (attributes) tanımlar. Tablonun sütunları kayıt türünün niteliklerine karşılık gelir.
- İlişkisel veri modeli en yaygın kullanılan veri modelidir ve mevcut veritabanı sistemlerinin büyük çoğunluğu ilişkisel modele dayanmaktadır.

Örnek Bir İlişkisel Veritabanı

- Şekilde iki tablodan oluşan örnek bir ilişkisel veritabanı sunmaktadır: biri üniversitedeki öğretim elemanlarının ayrıntılarını, diğeri ise üniversitenin çeşitli bölümlerinin ayrıntılarını göstermektedir.

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Varlık-İlişki (Entity-Relationship) Modeli

- Varlık-ilişki (E-R) veri modeli, varlıklar olarak adlandırılan temel nesnelerden ve bu nesneler arasındaki ilişkilerden oluşan bir koleksiyon kullanır.
- Varlık, gerçek dünyada diğer nesnelerden ayırt edilebilen bir "şey" veya "nesne "dir.
- Varlık-ilişki modeli veritabanı tasarımında yaygın olarak kullanılmaktadır.

Yarı Yapılandırılmış (Semi-structured) Veri Modeli

- Yarı yapılandırılmış veri modeli, aynı tipteki veri öğelerinin farklı nitelik (attributes) kümelerine sahip olabileceği verilerin belirlenmesine izin verir.
- Bu, daha önce bahsedilen ve belirli bir türdeki her veri öğesinin aynı öznitelik kümesine sahip olması gereken veri modellerinin aksine bir durumdur.
- JSON ve XML (Extensible Markup Language) yaygın olarak kullanılan yarı yapılandırılmış veri temsilleridir.

Yarı Yapılandırılmış (Semi-structured) Veri Modeli

- Şekilde, aynı türdeki (örneğin "kişi") öğelerin farklı nitelik kümelerine sahip olabileceği yarı yapılandırılmış veri modelini gösteren bir JSON örneği yer almaktadır.
  - Bu örnekte:Alice, isim, yaş, e-posta ve telefon niteliklerine sahip.
  - Bob, isim, yaş ve adres niteliklerine sahip, ancak e-posta veya telefon bilgisi yok.
  - Charlie, isim, yaş, e-posta, iş_unvanı ve şirket niteliklerine sahip.
- Bu, yarı yapılandırılmış verinin esnekliğini gösterir; her bir öğe, aynı türde olmasına rağmen farklı niteliklere sahip olabilir.

```
[
  {
    "isim": "Alice",
    "yaş": 30,
    "e-posta": "alice@ornek.com",
    "telefon": "555-1234"
  },
  {
    "isim": "Bob",
    "yaş": 25,
    "adres": "Ana Cadde 123"
  },
  {
    "isim": "Charlie",
    "yaş": 35,
    "e-posta": "charlie@ornek.com",
    "iş_unvanı": "Mühendis",
    "şirket": "Teknoloji A.Ş."
  }
]
```

Nesne Tabanlı Veri Modeli

- Nesne yönelimli programlama (özellikle Java, C++ veya C#) yazılım geliştirme metodolojisinde baskın hale gelmiştir. Bu durum başlangıçta farklı bir nesne yönelimli veri modelinin geliştirilmesine yol açmıştır, ancak günümüzde nesne kavramı ilişkisel veritabanlarına iyi bir şekilde entegre edilmiştir.
- Nesneleri ilişkisel tablolarda saklamak için standartlar mevcuttur. Veritabanı sistemleri, prosedürlerin veritabanı sisteminde saklanmasına ve veritabanı sistemi tarafından yürütülmesine izin verir. Bu, ilişkisel modelin kapsülleme, metotlar ve nesne kimliği kavramlarıyla genişletilmesi olarak görülebilir.

Veri Soyutlama

- Sistemin kullanılabilir olması için verileri, verimli bir şekilde alması gerekir. Verimlilik ihtiyacı, veritabanı sistemi geliştiricilerini veritabanındaki verileri temsil etmek için karmaşık veri yapıları kullanmaya yöneltmiştir.
- Birçok veritabanı sistemi kullanıcısı bilgisayar eğitimi almadığından, geliştiriciler, kullanıcıların sistemle etkileşimlerini basitleştirmek için çeşitli **veri soyutlama seviyeleri** aracılığıyla karmaşıklığı kullanıcılardan gizler.

Veri Soyutlama Seviyeleri

- **Fiziksel seviye** : En düşük soyutlama düzeyi, verilerin gerçekte nasıl depolandığını açıklar. Fiziksel düzey, karmaşık düşük düzeyli veri yapılarını ayrıntılı olarak açıklar.
- **Mantıksal (Logical) seviye** : Bir sonraki daha yüksek soyutlama düzeyidir, veritabanında hangi verilerin depolandığını ve bu veriler arasında hangi ilişkilerin bulunduğunu açıklar. Dolayısıyla mantıksal düzey, az sayıdaki nispeten basit yapılar açısından tüm veri tabanını tanımlar.
  - Mantıksal düzeydeki basit yapıların gerçekleştirimi karmaşık fiziksel düzeydeki yapıları içerse de, mantıksal düzeydeki kullanıcının bu karmaşıklığın farkında olması gerekmez. Bu, **fiziksel veri bağımsızlığı** olarak adlandırılır.

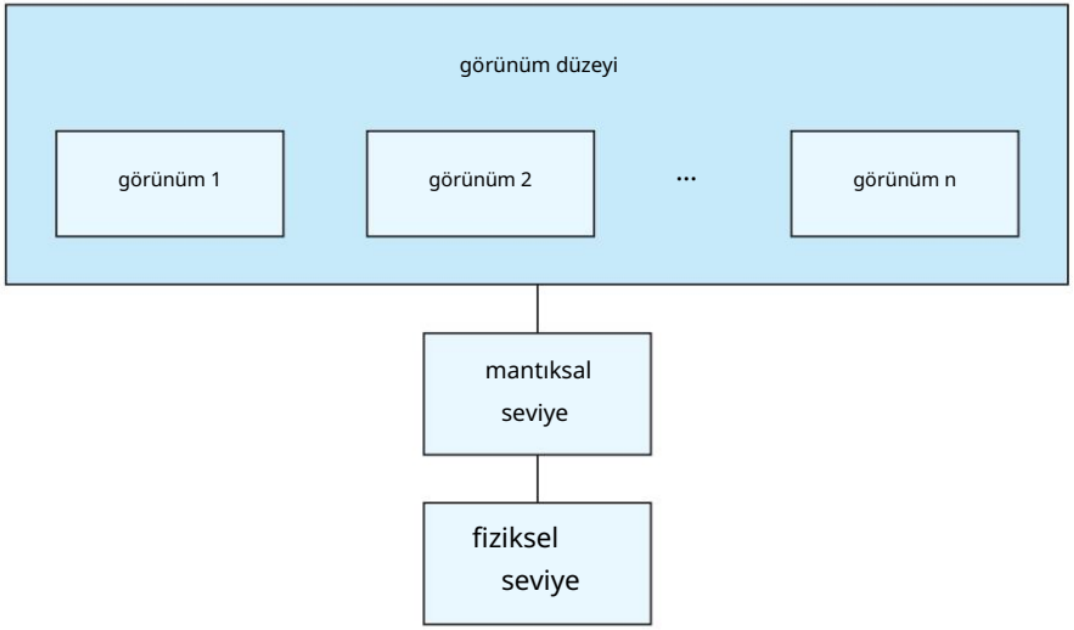
Veri Soyutlama Seviyeleri

(devamı)

- **Görünüm (view) düzeyi** :Mantıksal seviye daha basit yapılar kullansa da, büyük bir veritabanında saklanan bilgilerin çeşitliliği nedeniyle karmaşıklık devam eder. Görünüm düzeyi en yüksek soyutlama seviyesidir, belirli bir kullanıcının veya kullanıcı grubunun görebileceklerini tanımlar.
  - Veritabanı sisteminin birçok kullanıcısı veritabanının tüm bilgilerine ihtiyaç duymaz; bunun yerine, veritabanının yalnızca bir kısmına erişmeleri gerekir.
    - Görünüm seviyesi, veritabanının özelleştirilmiş bir perspektifini sunarak, veritabanının tamamı yerine yalnızca kullanıcıların ihtiyaç duydukları ilgili verileri görmelerini sağlar.
  - **Görünüm soyutlama düzeyi**, sistemle etkileşimlerini basitleştirmek için vardır. Sistem aynı veritabanı için birçok görünüm sağlayabilir.

Veri Soyutlama

- Şekilde üç soyutlama düzeyi arasındaki ilişki gösterilmektedir.
- İlişkisel model gibi veri modellerinin önemli bir özelliği, bu tür düşük seviyeli uygulama ayrıntılarını sadece veritabanı kullanıcılarından değil, veritabanı-uygulama geliştiricilerinden bile gizlemeleridir.
- Veritabanı sistemi, uygulama geliştiricilerin veri modelinin soyutlamalarını kullanarak verileri depolamasına ve almasına olanak tanır ve soyut işlemleri düşük seviyeli uygulamadaki işlemlere dönüştürür.



Veri Soyutlama

- Programlama dillerindeki veri türleri kavramına bir benzetme, soyutlama düzeyleri arasındaki ayrımı netleştirebilir. Birçok yüksek seviyeli programlama dili yapılandırılmış tip kavramını destekler. Bir kaydın türünü soyut olarak aşağıdaki gibi tanımlayabiliriz

```
type instructor = record
                    ID : char (5);
                    name : char (20);
                    dept_name : char (20);
                    salary : numeric (8,2);
                  end;
```

- Bu kod, dört alana sahip eğitmen (instructor) adında yeni bir kayıt türü tanımlar. Her alanın bir adı ve onunla ilişkili bir türü vardır. Örneğin, char(20) 20 karakterli bir string belirtir.

Veri Soyutlama

- **Fiziksel düzeyde**, bir eğitmen kaydı ardışık baytlardan oluşan bir blok olarak tanımlanabilir. Derleyici bu ayrıntı düzeyini programcılardan gizler. Benzer şekilde, veritabanı sistemi de en düşük seviyedeki depolama detaylarının çoğunu veritabanı programcılarından gizler.
- Öte yandan veritabanı yöneticileri, verilerin fiziksel organizasyonuna ilişkin belirli ayrıntıların farkında olabilirler.
  - Örneğin, tabloları dosyalarda saklamanın birçok olası yolu vardır. Bunun bir yolu, bir tabloyu bir dosyada bir dizi kayıt olarak saklamaktır; bir kaydın farklı niteliklerini ayırmak için özel bir karakter (virgül gibi) kullanılır ve kayıtları ayırmak için başka bir özel karakter (yeni satır karakteri gibi) kullanılabilir.

Veri Soyutlama

- Tüm nitelikler sabit uzunluktaysa, niteliklerin uzunlukları ayrı ayrı saklanabilir ve sınırlayıcılar dosyadan çıkarılabilir. Değişken uzunluktaki nitelikler, uzunluğun ve ardından verinin saklanmasıyla ele alınabilir.
- Veritabanları, kayıtların daha kolay alınmasını desteklemek için indeks adı verilen bir tür veri yapısı kullanır; bunlar da **fiziksel düzeyin** bir parçasını oluşturur.
- **Mantıksal düzeyde**, bu tür kayıtların her biri, önceki kod segmentinde olduğu gibi bir tip tanımıyla tanımlanır. Bu kayıt tiplerinin birbiriyle ilişkisi de mantıksal düzeyde tanımlanır; bir eğitmen kaydının dept_name değerinin department tablosunda görünmesi gerekliliği böyle bir ilişkiye örnektir.
  - Bir programlama dili kullanan programcılar bu soyutlama düzeyinde çalışırlar. Benzer şekilde, veritabanı yöneticileri de genellikle bu soyutlama düzeyinde çalışırlar.

Veri Soyutlama

- Son olarak, **görünüm (view) düzeyinde**, bilgisayar kullanıcıları veri türlerinin ayrıntılarını gizleyen bir dizi uygulama programı görürler.
- **Görünüm düzeyinde**, veritabanının çeşitli görünümleri tanımlanır ve bir veritabanı kullanıcısı bu görünümlerin bir kısmını veya tamamını görür.
- **Görünümler**, veritabanının mantıksal düzeyinin ayrıntılarını gizlemenin yanı sıra, kullanıcıların veritabanının belirli bölümlerine erişmesini önlemek için bir güvenlik mekanizması da sağlar.
  - Örneğin, üniversite kayıt bürosundaki memurlar, veritabanının yalnızca öğrenciler hakkında bilgi içeren bölümünü görebilir; eğitmenlerin maaşları hakkındaki bilgilere erişemezler.

Veri Tabanı Örnekleri (Instances) ve Şemaları (Schemas)

- Bilgi eklendikçe ve silindikçe veritabanları zamanla değişir. Belirli bir anda veritabanında saklanan bilgilerine veritabanının örneği **(database instance)** denir. Veritabanının genel tasarımına veritabanı şeması **(database schema)** denir.
- Bir veritabanı şeması, bir programdaki değişken bildirimlerine karşılık gelir. Her değişkenin belirli bir anda belirli bir değeri vardır. Bir programdaki değişkenlerin belirli bir andaki değerleri, bir veritabanı şeması örneğine **(instance)** karşılık gelir.
- Veritabanı sistemleri soyutlama düzeylerine göre bölümlere ayrılmış çeşitli şemalara sahiptir.
  - **Fiziksel şema**, veritabanı tasarımını fiziksel düzeyde tanımlarken, **mantıksal şema**, veritabanı tasarımını mantıksal düzeyde açıklar.
  - Bir veritabanı ayrıca , veritabanının farklı **görünümlerini (views)** tanımlayan, bazen alt şemalar (**subschemas**) olarak adlandırılan **görünüm düzeyinde** birkaç şemaya sahip olabilir .

Fiziksel Veri Bağımsızlığı

- Mantıksal şema, uygulama programları üzerindeki etkisi açısından açık ara en önemlisidir, çünkü programcılar uygulamaları mantıksal şemayı kullanarak oluştururlar.
- Fiziksel şema mantıksal şemanın altında gizlidir ve genellikle uygulama programlarını etkilemeden kolayca değiştirilebilir.
- Uygulama programları fiziksel şemaya bağlı değilse ve bu nedenle fiziksel şema değiştiğinde yeniden yazılmaları gerekmiyorsa **fiziksel veri bağımsızlığı** sergiledikleri söylenir**.**

Veritabanı Dilleri

- Bir veritabanı sistemi, veritabanı şemasını belirtmek için bir **veri tanımlama dili (data-definition language-DDL)** ve veritabanı sorgularını ve güncellemelerini ifade etmek için bir **veri manipülasyon dili (data-manipulation language- DML)** sağlar.
- Pratikte, veri tanımlama ve veri işleme dilleri iki ayrı dil değildir; bunun yerine SQL dili gibi tek bir veritabanı dilinin parçalarını oluştururlar. Neredeyse tüm ilişkisel veritabanı sistemleri SQL dilini kullanır.

Veri Tanımlama Dili (Data-Definition Language DDL)

- Bir veritabanı şemasını, **veri tanımlama dili (Data-Definition Language DDL)** adı verilen özel bir dille ifade edilen bir dizi tanımla belirtiriz. DDL ayrıca verilerin ek özelliklerini belirtmek için de kullanılır.
- Veri tabanı sistemi tarafından kullanılan depolama yapısını ve erişim yöntemlerini, **veri depolama ve tanımlama dili** olarak adlandırılan özel bir DDL türündeki bir dizi deyimle belirtiriz. Bu ifadeler, genellikle kullanıcılardan gizlenen veritabanı şemalarının uygulama ayrıntılarını tanımlar.
- Veritabanında depolanan veri değerleri belirli tutarlılık kısıtlamalarını karşılamalıdır. Örneğin, üniversitenin bir departmanın hesap bakiyesinin asla negatif olmamasını gerektirdiğini varsayalım.
- DDL, bu tür kısıtlamaları belirtmek için olanaklar sağlar. Veritabanı sistemi, veritabanı her güncellendiğinde bu kısıtlamaları kontrol eder. Genel olarak, bir kısıtlama veritabanı ile ilgili keyfi bir kısıtlama olabilir. Ancak, keyfi kısıtlamaların test edilmesi maliyetli olabilir. Bu nedenle, veritabanı sistemleri yalnızca minimum ek yük ile test edilebilen kısıtlamaları uygular.

Veri Tanımlama Dili (Data-Definition Language DDL)

Yaygın kullanılan bazı kısıtlamalar:

- **Etki Alanı (Domain) Kısıtlamaları**. Her nitelikle olası değerlerden oluşan bir etki alanı ilişkilendirilmelidir (örneğin, tamsayı türleri, karakter türleri, tarih/zaman türleri). Bir niteliğin belirli bir etki alanında olduğunu bildirmek, niteliğin alabileceği değerler üzerinde bir kısıtlama görevi görür. Etki alanı kısıtlamaları bütünlük kısıtlamalarının en temel şeklidir.
- **Referans Bütünlüğü:** Belirli bir nitelik kümesi için bir tabloda görünen bir değerin başka bir tablodaki belirli bir nitelik kümesinde de görünmesini sağlamak istediğimiz durumlar vardır (referans bütünlüğü).
  - Örneğin, her ders için listelenen bölüm, üniversitede gerçekten var olan bir bölüm olmalıdır. Daha açık bir ifadeyle, bir kurs kaydındaki departman adı değeri, departman tablosunun bazı kayıtlarında departman adı niteliğinde görünmelidir.

Veri Tanımlama Dili (Data-Definition Language DDL)

(devamı)

- **Yetkilendirme:** Veritabanındaki çeşitli veri değerleri üzerinde izin verilen erişim türüne göre kullanıcılar arasında farklılaştırma yapmak isteyebiliriz.
  - Bu farklılaştırmalar yetkilendirme terimleriyle ifade edilir, en yaygın olanları şunlardır:
    - verilerin okunmasına izin veren ancak değiştirilmesine izin vermeyen okuma yetkilendirmesi;
    - yeni verilerin eklenmesine izin veren ancak mevcut verilerin değiştirilmesine izin vermeyen ekleme yetkilendirmesi;
    - verilerin değiştirilmesine izin veren ancak silinmesine izin vermeyen güncelleme yetkilendirmesi;
    - ve verilerin silinmesine izin veren silme yetkilendirmesi.
  - Kullanıcıya bu yetki türlerinin hepsini, hiçbirini ya da bir kombinasyonunu atayabiliriz.

Veri Tanımlama Dili (Data-Definition Language DDL)

- DDL deyimlerinin işlenmesi, tıpkı diğer programlama dillerinde olduğu gibi, bazı çıktılar üretir. DDL'nin çıktısı olan **meta verileri**, **veri sözlüğüne (data dictionary)** yerleştirilir. Meta verileri **veriler hakkındaki verileri** (veritabanı şeması, bütünlük kısıtlamaları, yetkilendirmeler, vb.) içerir.
  - **Veri sözlüğü**, yalnızca veritabanı sisteminin kendisi (normal bir kullanıcı değil) tarafından erişilebilen ve güncellenebilen özel bir tablo türü olarak kabul edilir. Veritabanı sistemi, gerçek verileri okumadan veya değiştirmeden önce veri sözlüğüne danışır.
- SQL, tabloların veri türleri ve bütünlük kısıtlamaları ile tanımlanmasına olanak tanıyan zengin bir DDL sağlar. Örneğin, aşağıdaki SQL DDL deyimi *departman* tablosunu tanımlar:

```
create table department
    (dept_name      char (20),
     building       char (15),
     budget         numeric (12,2));
```

Veri İşleme Dili (Data-Manipulation language DML)

- Veri işleme dili (DML), kullanıcıların uygun veri modeli tarafından düzenlenen verilere erişmesine veya bunları değiştirmesine olanak tanıyan bir dildir. Erişim türleri şunlardır:
  - Veri tabanında depolanan bilgilerin geri alınması.
  - Veri tabanına yeni bilgilerin eklenmesi.
  - Veri tabanından bilgi silinmesi.
  - Veri tabanında saklanan bilgilerin değiştirilmesi.
- Temel olarak iki tür veri işleme dili vardır
  - **Prosedürel DML** - kullanıcının hangi verilere ihtiyaç duyulduğunu ve bu verilerin nasıl elde edileceğini belirtmesini gerektirir.
  - **Bildirime dayalı (Declarative) DML** - kullanıcının, bu verilerin nasıl alınacağını belirtmeden hangi verilere ihtiyaç duyulduğunu belirtmesini gerektirir.
  - Bildirime dayalı DML'lerin öğrenilmesi ve kullanılması genellikle prosedürel DML'lerden daha kolaydır.

Veri İşleme Dili (Data-Manipulation language DML)

- Sorgu (**Query**), bilginin alınmasını talep eden bir ifadedir. DML'nin bilgi almayı içeren kısmına **sorgulama dili (query language)** denir . Teknik olarak yanlış olmasına rağmen, **sorgulama dili** ve veri işleme dili terimlerinin eşanlamlı olarak kullanılması yaygın bir uygulamadır.
- Ticari veya deneysel olarak kullanımda olan çok sayıda veritabanı sorgulama dili vardır. En yaygın kullanılan **sorgulama dili, SQL'dir**.

SQL Sorgu Dili

- SQL sorgu dili prosedürel değildir. Bir sorgu girdi olarak bir veya birkaç tabloyu alır ve her zaman tek bir tablo döndürür.
- Comp.Sci bölümündeki tüm eğitmenleri bulma örneği

```
select name
        from instructor
        where dept_name = 'Comp. Sci.'
```

Uygulama Programından Veritabanı Erişimi

- Prosedürel olmayan sorgulama dilleri, genel amaçlı bir programlama dili kadar güçlü değildir; yani, genel amaçlı bir programlama dili kullanılarak mümkün olan ancak SQL kullanılarak mümkün olmayan bazı hesaplamalar vardır.
  - SQL ayrıca kullanıcılardan giriş, ekranlara çıkış veya ağ üzerinden iletişim gibi eylemleri de desteklemez.
  - Bu tür hesaplamalar ve eylemler, C/C++, Java veya Python gibi bir *ana* dilde, veritabanındaki verilere erişen gömülü SQL sorguları ile yazılmalıdır.
  - **Uygulama programları**, veritabanı ile bu şekilde etkileşim kurmak için kullanılan programlardır.

Uygulama Programından Veritabanı Erişimi

- Veritabanına erişmek için, DML ifadelerinin ana bilgisayardan yürütülecekleri veritabanına gönderilmesi gerekir. Bu en yaygın olarak DML ve DDL ifadelerini veritabanına göndermek ve sonuçları almak için kullanılabilen bir **uygulama-program arayüzü** (prosedürler dizisi) kullanılarak yapılır.
- Açık Veritabanı Bağlantısı (Open Database Connectivity ODBC) standardı, C ve diğer bazı dillerle kullanım için uygulama programı arayüzlerini tanımlar.
  - Java Veritabanı Bağlantısı (JDBC) standardı, Java diline karşılık gelen bir arayüz tanımlar.

Veri tabanı tasarımı

- Veritabanı tasarımı esas olarak veritabanı şemasının tasarımını içerir. Modellenen kuruluşun/işletmenin ihtiyaçlarını karşılayan eksiksiz bir veritabanı uygulama ortamının tasarımı, daha geniş bir dizi konuya dikkat edilmesini gerektirir
- Üst düzey bir veri modeli, veritabanı tasarımcısına, veritabanı kullanıcılarının veri gereksinimlerini ve veritabanının bu gereksinimleri karşılayacak şekilde nasıl yapılandırılacağını belirleyecek kavramsal bir çerçeve sağlar.

Veri tabanı tasarımı aşamaları

1. **Kullanıcı gereksinimlerinin orta çıkarılması ve spesifikasyonu:** Veritabanı tasarımının ilk aşaması, potansiyel veritabanı kullanıcılarının veri ihtiyaçlarını tam olarak karakterize etmektir. Veritabanı tasarımcısının bu görevi yerine getirmek için alan uzmanları ve kullanıcılarla kapsamlı bir şekilde etkileşime girmesi gerekir.
   - Bu aşamanın sonucu, kullanıcı gereksinimlerinin bir **spesifikasyonudur**.
2. **Kavramsal tasarım (conceptual-design) aşaması:** Daha sonra, tasarımcı bir veri modeli seçer ve seçilen veri modelinin kavramlarını uygulayarak bu gereksinimleri **veritabanının kavramsal şemasına** dönüştürür. Bu kavramsal tasarım aşamasında geliştirilen şema, işletmenin ayrıntılı bir genel görünümünü sağlar.
   - Tasarımcı, tüm veri gereksinimlerinin gerçekten karşılandığını ve birbiriyle çelişmediğini doğrulamak için şemayı gözden geçirir.

Veri tabanı tasarımı

- İlişkisel model açısından, kavramsal tasarım aşaması, veritabanında hangi nitelikleri yakalamak istediğimize ve çeşitli tabloları oluşturmak için bu nitelikleri nasıl gruplandıracağımıza ilişkin kararları içerir.
- "Ne" kısmı temelde bir iş kararıdır. Veritabanına hangi özellikleri kaydetmeliyiz? sorusu ile igilenir.
- "Nasıl" kısmı ise esas olarak bir bilgisayar bilimi problemidir. Hangi ilişki şemalarına sahip olmalıyız ve nitelikler çeşitli ilişki şemaları arasında nasıl dağıtılmalıdır? sorusu ile ilgilenir.
  - Bu problemin üstesinden gelmenin temelde iki yolu vardır.
    - Bunlardan ilki varlık-ilişki modelini kullanmak;
    - Diğeri ise girdi olarak tüm nitelikler kümesini alan ve bir tablolar kümesi oluşturan bir dizi algoritma (topluca normalleştirme olarak bilinir) kullanmaktır.

Veri tabanı tasarımı

3. **Tamamen geliştirilmiş bir kavramsal şema aşaması:** Tamamen geliştirilmiş bir kavramsal şema aşaması, kurumun/işletmenin fonksiyonel gereksinimlerini (**functional requirements**) gösterir.
   - Fonksiyonel gereksinimlerin spesifikasyonunda (**specification of functional requirements**), kullanıcılar veriler üzerinde gerçekleştirilecek işlem türlerini tanımlar.
   - Örnek işlemler arasında verilerin değiştirilmesi veya güncellenmesi, belirli verilerin aranması ve alınması ve verilerin silinmesi yer alır.
   - Kavramsal tasarımın bu aşamasında, tasarımcı **fonksiyonel gereksinimleri** karşıladığından emin olmak için şemayı gözden geçirebilir.

Veri tabanı tasarımı

Soyut bir veri modelinden veritabanının gerçekleştirimine geçme süreci iki tasarım aşamasında ilerler:

4. **Mantıksal tasarım aşaması (logical-design phase):** Mantıksal tasarım aşamasında tasarımcı üst düzey kavramsal şemayı, kullanılacak veritabanı sisteminin gerçekleştirim veri modeli ile eşleştirir.
5. **Fiziksel tasarım aşaması**: Tasarımcı, ortaya çıkan sisteme özgü veritabanı şemasını, veritabanının fiziksel özelliklerinin belirlendiği sonraki fiziksel tasarım aşamasında kullanır. Bu özellikler, dosya organizasyonu biçimini ve dahili depolama yapılarını içerir.

Veritabanı Motoru

- Veritabanı Motoru, verilerin bir veritabanında nasıl depolandığını, alındığını ve manipüle edildiğini yönetmekten sorumlu temel yazılım bileşenini ifade eder.
- Bir veritabanı sistemi, genel sistemin sorumluluklarının her biriyle ilgilenen modüllere bölünmüştür.
- Bir veritabanı sisteminin/motorunun fonksiyonel bileşenleri aşağıdakilere ayrılabilir:
  - Depolama yöneticisi (Storage Manager),
  - Sorgu işlemcisi (Query Processor),
  - Hareket yönetimi (Transaction Management)

Depolama Yöneticisi

- Depolama yöneticisi, veritabanında depolanan düşük seviyeli veriler ile sisteme gönderilen uygulama programları ve sorgular arasında arayüz sağlayan bir veritabanı sisteminin bileşenidir.
  - Depolama yöneticisi çeşitli DML ifadelerini düşük seviyeli dosya sistemi komutlarına dönüştürür.
- Depolama yöneticisi aşağıdaki görevlerden sorumludur:
  - İşletim sistemi dosya yöneticisiyle etkileşim
  - Verilerin verimli bir şekilde saklanması, alınması ve güncellenmesi

Depolama Yöneticisi

Depolama yöneticisi bileşenleri şunları içerir:

- **Yetkilendirme ve bütünlük yöneticisi**: Bütünlük kısıtlamalarının karşılanıp karşılanmadığını test eder ve kullanıcıların verilere erişim yetkisini kontrol eder
- **Hareket (Transaction) yöneticisi**: Sistem arızalarına rağmen veritabanının tutarlı (doğru) durumda kalmasını ve eşzamanlı hareket yürütmelerinin çakışma olmadan ilerlemesini sağlar.
- **Dosya yöneticisi**: Disk depolama alanında yer tahsisini ve verileri yönetir
- **Arabellek yöneticisi**: Verileri disk deposundan ana belleğe getirmekten ve ana bellekte hangi verilerin önbelleğe alınacağına karar vermekten sorumludur.
  - Arabellek yöneticisi, veritabanının ana belleğin boyutundan çok daha büyük veri boyutlarını işlemesini sağladığından veritabanı sisteminin kritik bir parçasıdır.

Depolama Yöneticisi (Devam)

- Depolama yöneticisi, fiziksel sistem gerçekleştiriminin bir parçası olarak çeşitli veri yapılarını gerçekleştirir:
  - **Veri dosyaları** - veritabanının kendisini saklar.
  - **Veri sözlüğü (data dictionary)** - veritabanının yapısı, özellikle veritabanı şeması hakkındaki meta verileri saklar.
  - **İndeksler** - veri öğelerine hızlı erişim sağlayabilir. Bir veritabanı indeksi, belirli bir değeri tutan veri öğelerine işaretçiler sağlar.
    - Bir kitaptaki indeks/içindekiler gibi, bir veritabanı indeksi de belirli bir değeri tutan veri öğelerine işaretçiler sağlar. Örneğin, belirli bir *ID'ye* sahip ***eğitmen*** kaydını ya da belirli bir *isme* sahip tüm ***eğitmen*** kayıtlarını bulmak için bir indeks kullanabiliriz.

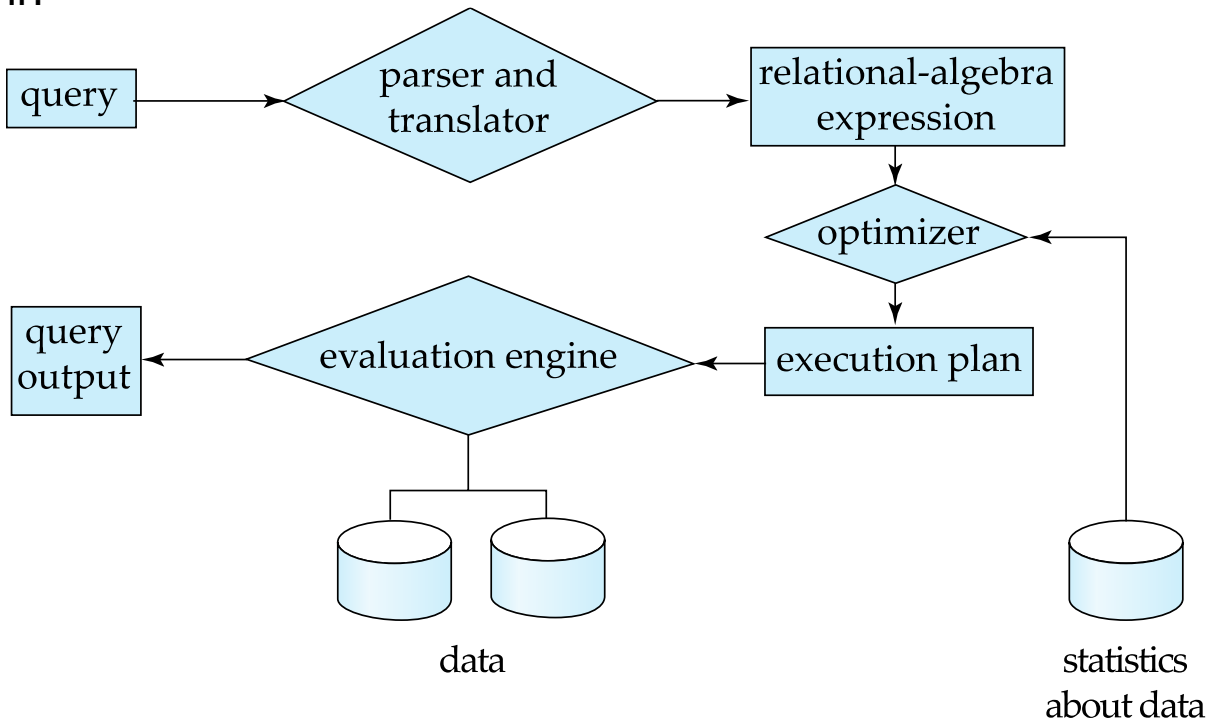
Sorgu İşlemcisi

- Sorgu işlemcisi bileşenleri şunları içerir:
  - **DDL yorumlayıcısı** - DDL ifadelerini yorumlar ve tanımları veri sözlüğüne kaydeder.
  - **DML derleyicisi** - bir sorgu dilindeki DML ifadelerini, sorgu değerlendirme motorunun anladığı alt düzey talimatlardan oluşan bir değerlendirme planına çevirir.
    - DML derleyicisi sorgu optimizasyonunu gerçekleştirir; yani çeşitli alternatifler arasından en düşük maliyetli değerlendirme planını seçer.
  - **Sorgu değerlendirme motoru (Query evaluation engine)** - DML derleyicisi tarafından oluşturulan düşük seviyeli talimatları yürütür.

Sorgu İşleme

Sorgu işleme süreci genel olarak 3 adımda gerçekleşir.

1. Adım gelen sorugunun ayrıştırılması ve ilişkisel cebir ifadelerine, daha alt düzey talimatlara dönüştürülmesidir.
2. Adımda çeşitli alternatifler arasından en düşük maliyetli değerlendirme planını seçme işlemi gerçekleştirilir.
3. Adımda ise değerlendirme motoru alt düzey talimatları yürüterek çıktı verir.



Hareket yönetimi

- **Hareket (transaction),** tek bir mantıksal iş birimi oluşturan işlemler (okuma, yazma, ekleme, güncelleme veya silme gibi) topluluğunu ifade eder.
  - Veritabanı hareketleri; bir ürün satın almak, bir kursa kaydolmak veya bir çek hesabına para yatırmak gibi olaylar tarafından tetiklenen gerçek dünya işlemlerini yansıtır.
  - Hareketlerin, bir müşterinin hesabının güncellenmesi, ürün envanterinin ayarlanması ve satıcının alacak hesaplarının güncellenmesi gibi birçok adım içermesi muhtemeldir.
  - Örneğin, bir hesaptan diğerine para transferi, her iki hesapta da birer kez gerçekleştirilecek update işleminden oluşan bir harekettir.
- **Hareket yönetimi bileşeni,** sistem arızalarına (örneğin, elektrik kesintileri ve işletim sistemi çökmeleri) ve işlem hatalarına rağmen veritabanının tutarlı (doğru) durumda kalmasını sağlar.

Hareket yönetimi

- Hareket yönetiminde işlemlerin ya hepsi gerçekleşmeli ya da hiç biri gerçekleşmemelidir. Bu ya hep ya hiç gereksinimine **atomiklik (atomicity)** denir.
  - Bunun bir örneği, bir bankacılık uygulamasında A hesabından B hesabına yapılan para transferidir. A hesabından B hesabına yapılan transfer işleminde A hesabından bakiye düşülmesi ve B hesabının bakiyesinin artırılması önemlidir.
- Ayrıca transfer işleminde veritabanının tutarlılığının korunması gerekir. Yani A ve B bakiyelerinin korunması gerekir. Bu gereksinime **tutarlılık (consistency)** denir.
- Son olarak, transferin başarılı bir şekilde yürütülmesinden sonra, A ve B hesaplarının bakiyelerinin yeni değerleri, sistem arızası olasılığına rağmen kalıcı olmalıdır. Bu kalıcılık gereksinimi **dayanıklık (durability)** olarak adlandırılır.

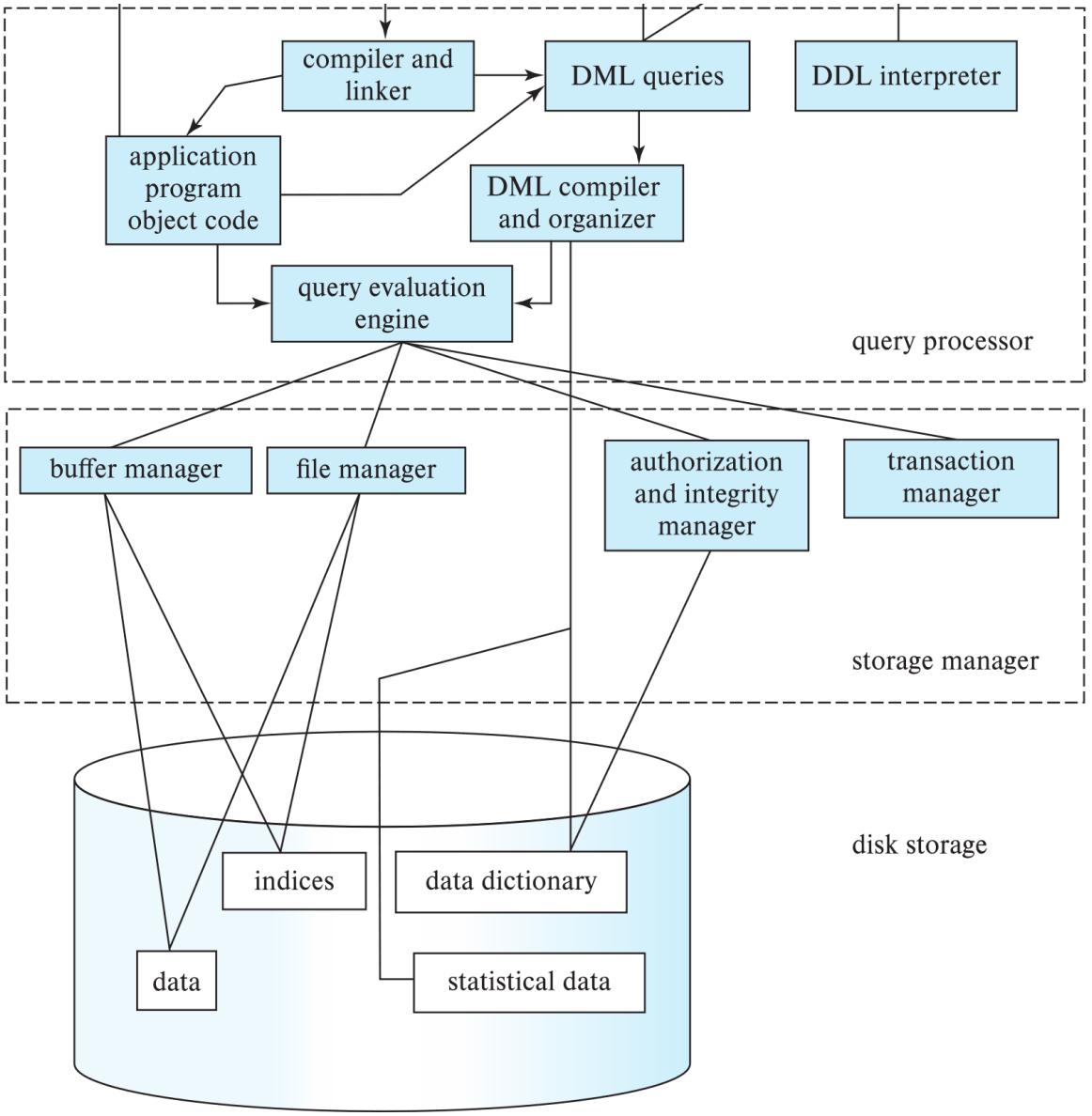
Hareket yönetimi

- **Hareket yöneticisi**,
  - **kurtarma yöneticisinden** ve
  - **eşzamanlılık kontrol yöneticisinden** oluşur.
- **Kurtarma yöneticisi: Atomiklik** ve **dayanıklılık** özelliklerinin sağlanması veritabanı sisteminin, özellikle de **kurtarma yöneticisinin** sorumluluğundadır.
  - Atomiklik özelliğini sağlamak istiyorsak, başarısız bir işlemin veritabanının durumu üzerinde hiçbir etkisi olmamalıdır. Bu nedenle, veritabanı, söz konusu işlem yürütülmeye başlamadan önceki durumuna geri getirilmelidir.
  - Bu nedenle veritabanı sistemi **hata kurtarma** işlemini gerçekleştirmeli, yani sistem arızalarını tespit etmeli ve veritabanını arızanın meydana gelmesinden önceki duruma geri getirmelidir.

Hareket yönetimi

- **Eşzamanlılık (concurrency) kontrol yöneticisi:** birden fazla hareket veritabanını eş zamanlı olarak güncellediğinde, her bir hareket doğru olsa bile verilerin **tutarlılığı** artık korunamayabilir.
  - Veritabanının **tutarlılığını** sağlamak için eşzamanlı hareketler arasındaki etkileşimi kontrol etmek, **eşzamanlılık kontrol yöneticisinin** sorumluluğundadır.

Veritabanı Mimarisi (Merkezi/Paylaşılan Bellek)



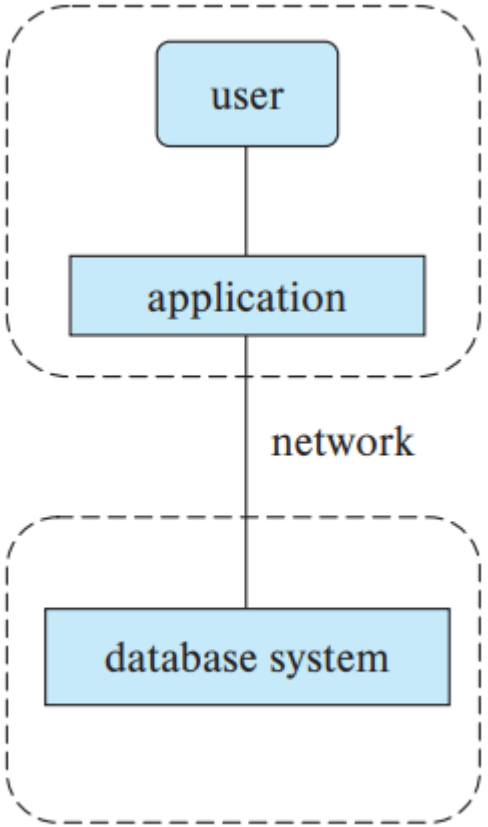
Veritabanı Mimarisi

- ***Merkezi veritabanları***
  - Önceki slaytta gösterilen merkezi mimari, birden fazla CPU'ya sahip olan ve paralel işlemden yararlanan, ancak tüm CPU'ların ortak bir paylaşılan belleğe eriştiği paylaşılan bellekli sunucu mimarilerine (shared-memory server architectures) uygulanabilir.
- Daha da büyük veri hacimlerine ve daha da yüksek işlem hızlarına ölçeklendirmek (scale up) için, ***paralel veritabanları (parallel databases)*** birden fazla makineden oluşan bir kümede (cluster) çalışacak şekilde tasarlanmıştır.
- Ayrıca, ***dağıtık veritabanları (distributed databases)***, coğrafi olarak ayrılmış birden fazla makinede veri depolama ve sorgu işlemeye izin verir.

Veritabanı Uygulamaları

Veritabanı uygulamaları genellikle iki veya üç bölüme bölünür:

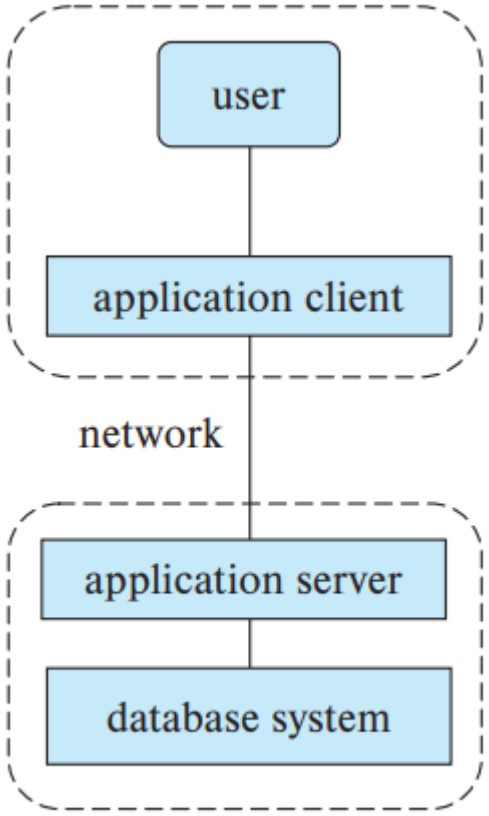
- **İki katmanlı mimari (three-tier architecture)** - Önceki nesil veritabanı uygulamaları, uygulamanın istemci makinede bulunduğu ve sorgu dili ifadeleri aracılığıyla sunucu makinedeki veritabanı sistemi işlevselliğini çağırdığı iki katmanlı bir mimari kullanıyordu.



(a) Two-tier architecture

Veritabanı Uygulamaları

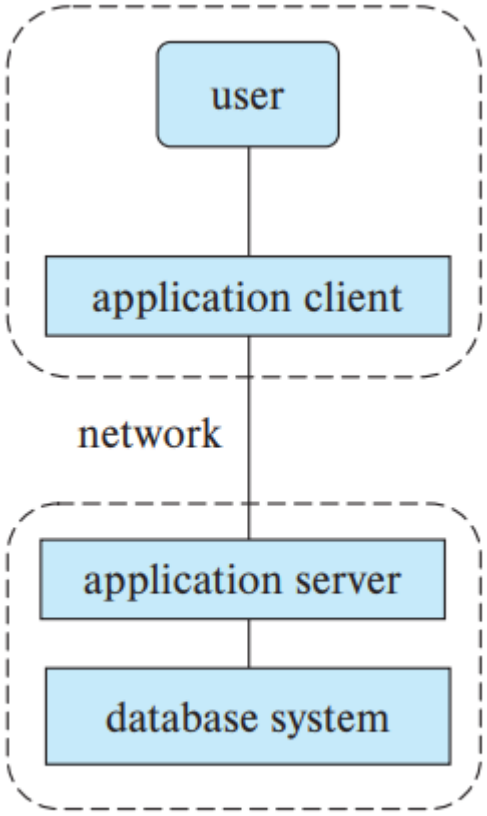
- **Üç katmanlı mimari (three-tier architecture)** – İki katmanlı mimarinin aksine, modern veritabanı uygulamaları, istemci makinenin yalnızca bir frontend (ön uç) görevi gördüğü ve herhangi bir doğrudan veritabanı çağrısı içermediği **üç katmanlı bir mimari** kullanır.
  - Web tarayıcıları ve mobil uygulamalar günümüzde en yaygın kullanılan uygulama istemcileridir. İstemci tarafı, genellikle bir form arayüzü aracılığıyla bir **uygulama sunucusuyla (application server)** iletişim kurar.
  - Üç katmanlı uygulamalar, iki katmanlı uygulamalara göre daha iyi güvenliğin yanı sıra daha iyi performans sağlar.



(b) Three-tier architecture

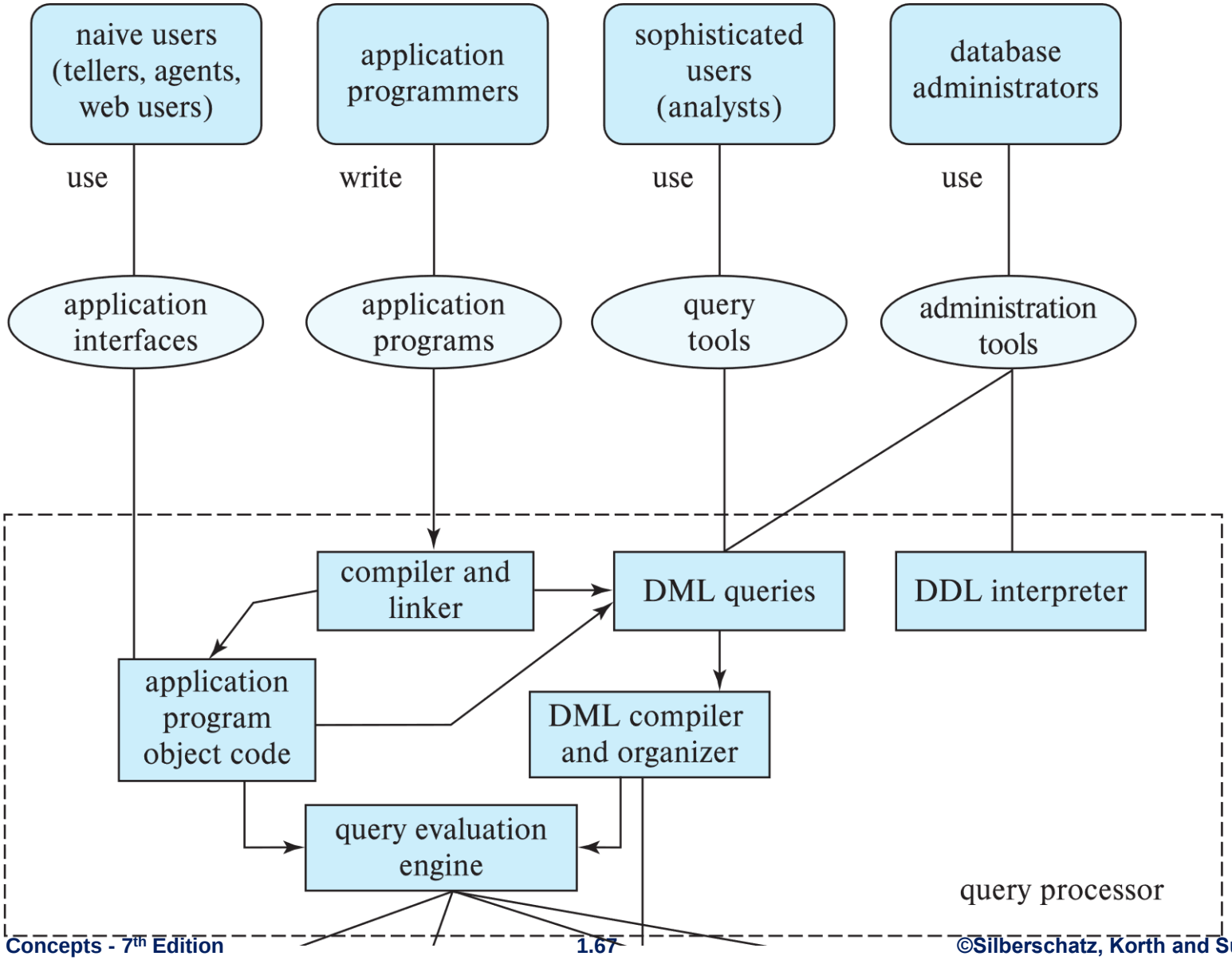
Veritabanı Uygulamaları

- **Üç katmanlı mimari (three-tier architecture)** - **Uygulama sunucusu** da verilere erişmek için bir veritabanı sistemiyle iletişim kurar.
  - **İş mantığı (business logic)**, verilerin uygulama içinde nasıl işlendiğini ve yönetildiğini tanımlayan kurallar ve prosedürler anlamına gelir, **uygulama sunucusunda** gerçekleştirilir ve frontend ile veritabanı (backend) arasında bir aracı görevi görür.
  - Bir e-ticaret uygulamasında bazı **iş mantığı** örnekleri:
    - Bir müşteri alışveriş sepetine bir ürün eklediğinde, uygulama, ürünün stokta olup olmadığını kontrol eder ve miktarı ayırır.
    - Ödeme sırasında uygulama, vergiler ve kargo dahil olmak üzere toplam maliyeti hesaplar ve geçerli indirimleri veya promosyon kodlarını uygular.
    - Sipariş onaylandıktan sonra, uygulama müşteriye bir bildirim e-postası gönderir ve veritabanındaki envanteri günceller.



(b) Three-tier architecture

Veritabanı Kullanıcıları



Veritabanı yöneticisi

Sistem üzerinde merkezi kontrole sahip olan kişiye **veritabanı yöneticisi (DBA) adı verilir.** Bir DBA'nın işlevleri şunları içerir:

- Şema tanımı: DBA, DDL'deki bir dizi veri tanımlama deyimini çalıştırarak orijinal veritabanı şemasını oluşturur
- Depolama yapısı ve erişim yöntemi tanımı: DBA, verilerin fiziksel organizasyonuna ilişkin ve oluşturulacak indekslere ilişkin bazı parametreler belirleyebilir.
- Şema ve fiziksel organizasyon değişikliği: DBA, kurumun değişen ihtiyaçlarını yansıtmak veya performansı artırmak için fiziksel organizasyonu değiştirmek amacıyla şema ve fiziksel organizasyonda değişiklikler yapar.
- Veri erişimi için yetki verilmesi

Veritabanı yöneticisi

(devamı)

- Rutin bakım
  - Veritabanının periyodik olarak yedeklenmesi
  - Normal işlemler için yeterli boş disk alanının bulunmasını sağlamak ve gerektiğinde disk alanını yükseltmek
  - Veritabanında çalışan işlerin izlenmesi ve bazı kullanıcılar tarafından gönderilen çok pahalı görevler nedeniyle performansın düşmemesinin sağlanması.

Veritabanı Sistemlerinin Tarihçesi

- 1950'ler ve 1960'ların başı:
  - Veri depolama için manyetik bantlar geliştirildi. Veri işleme, bir veya daha fazla banttan veri okuma ve yeni bir banda veri yazmadan oluşuyordu.
  - Veriler ayrıca delikli kart destelerinden girilebilir ve yazıcılardan çıktı olarak alınabilirdi.
  - Bantlar (ve kart desteleri) yalnızca sıralı olarak okunabiliyordu.

Veritabanı Sistemlerinin Tarihçesi

- 1960'ların sonu ve 1970'ler:
  - 1960'ların sonlarında sabit disklerin yaygın kullanımı veri işleme senaryosunu büyük ölçüde değiştirdi, çünkü sabit diskler verilere doğrudan erişime izin veriyordu.
  - Disk üzerindeki herhangi bir konuma sadece milisaniyeler içinde erişilebildiğinden, verilerin disk üzerindeki konumu önemsizdi.
    - Veriler böylece sıralılığın zorbalığından kurtuldu.
  - Disklerin ortaya çıkmasıyla birlikte, listeler ve ağaçlar gibi veri yapılarının disk üzerinde depolanmasına olanak tanıyan ağ ve hiyerarşik veri modelleri geliştirildi.
  - Edgar Codd'un 1970 yılında yayınladığı dönüm noktası niteliğindeki bir makale **ilişkisel modeli** ve **ilişkisel modelde veri sorgulamanın prosedürel olmayan yollarını tanımladı** ve ilişkisel veritabanları doğdu.
    - Bu çalışmayla ACM Turing Ödülünü kazandı

Veritabanı Sistemlerinin Tarihçesi

- 1970'lerin sonu ve 1980'ler:
  - İlişkisel model başlangıçta algılanan performans dezavantajları nedeniyle pratikte kullanılmadı; ilişkisel veritabanları mevcut ağ ve hiyerarşik veritabanlarının performansını karşılayamıyordu.
  - IBM Research't e çığır açan bir proje olan System R ile bu durum değişti ve daha verimli bir ilişkisel veritabanı sisteminin inşası için teknikler geliştirildi.
    - Tamamen işlevsel System R prototipi, IBM'in ilk ilişkisel veritabanı ürünü olan SQL/DS'e yol açtı.
  - Aynı dönemde Berkeley'deki Kaliforniya Üniversitesi'nde Ingres sistemi geliştiriliyordu.
    - Aynı adı taşıyan ticari bir ürüne yol açtı. Yine bu sıralarda Oracle'ın ilk sürümü piyasaya sürüldü.
  - IBM DB2, Oracle, Ingres ve DEC Rdb gibi ilk ticari ilişkisel veritabanı sistemleri, bildirimsel (declarative) sorguların daha verimli işlenmesine yönelik tekniklerin geliştirilmesinde önemli bir rol oynamıştır.

Veritabanı Sistemlerinin Tarihçesi

- 1970'lerin sonu ve 1980'ler:
  - 1980'lerin başında, ilişkisel veritabanları performans alanında bile ağ ve hiyerarşik veritabanı sistemleriyle rekabet edebilir hale gelmişti.
  - İlişkisel veritabanlarının kullanımı o kadar kolaydı ki sonunda ağ ve hiyerarşik veritabanlarının yerini aldılar.
    - Bu eski (ağ ve hiyerarşik) modelleri kullanan programcılar birçok düşük seviyeli uygulama detayı ile uğraşmak zorunda kalıyorlardı ve sorgularını prosedürel bir şekilde kodlamak zorundaydılar. En önemlisi, programlarını tasarlarken verimliliği göz önünde bulundurmaları gerekiyordu ki bu da çok fazla çaba gerektiriyordu.
    - Bunun aksine, ilişkisel bir veritabanında, neredeyse tüm bu alt düzey görevler veritabanı sistemi tarafından otomatik olarak gerçekleştirilir ve programcıyı mantıksal düzeyde çalışmakta serbest bırakır.
  - İlişkisel model, 1980'lerdeki hakimiyetinden bu yana veri modelleri arasında en üst sıralarda yer almaktadır.

Veritabanı Sistemlerinin Tarihçesi

- 1990'lar:
  - SQL dili öncelikle sorguların yoğun kullanıldığı karar destek uygulamaları için tasarlanmıştır, ancak 1980'lerde veritabanlarının temel dayanağı güncellemelerin yoğun kullanıldığı hareket işleme uygulamalarıydı.
  - 1990'ların başında karar destek ve sorgulama, veritabanları için önemli bir uygulama alanı olarak yeniden ortaya çıktı. Büyük miktarda veriyi analiz etmeye yönelik araçların kullanımında büyük bir artış görüldü.
    - Birçok veritabanı satıcısı bu dönemde paralel veritabanı ürünlerini tanıttı. Veritabanı satıcıları ayrıca veritabanlarına nesne-ilişkisel destek eklemeye başladılar.
  - 1990'ların en önemli olayı World Wide Web'in patlayıcı büyümesiydi. Veritabanları her zamankinden çok daha yaygın bir şekilde kullanılmaya başlandı.
    - Veritabanı sistemleri artık çok yüksek hareket işleme hızlarının yanı sıra çok yüksek güvenilirlik ve 7 x 24 kullanılabilirliği (haftanın 7 günü, günde 24 saat kullanılabilirlik, yani planlı bakım faaliyetleri için kesinti olmaması) desteklemek zorundaydı.

Veritabanı Sistemlerinin Tarihçesi

- 2000'ler
  - Veritabanı sistemlerinde depolanan **veri türleri** bu dönemde hızla gelişti. Yarı yapılandırılmış veriler giderek daha önemli hale geldi. **XML** bir veri değişim standardı olarak ortaya çıktı.
    - Daha sonra JavaScript veya diğer programlama dillerinden nesneleri depolamak için çok uygun olan daha kompakt bir veri değişim formatı olan **JSON**'un önemi giderek arttı.
    - Büyük ticari sistemlere XML ve JSON formatları için destek eklendikçe, bu tür veriler giderek artan bir şekilde ilişkisel veritabanı sistemlerinde depolandı.
    - Mekansal/Coğrafi veriler navigasyon sistemlerinde ve gelişmiş uygulamalarda yaygın olarak kullanılmaya başlandı.
    - Veritabanı sistemleri bu tür veriler için destek ekledi.

Veritabanı Sistemlerinin Tarihçesi

- 2000'ler
  - Sosyal ağ platformlarının hızlı bir şekilde büyümesi, insanlar arasındaki bağlantılar ve gönderdikleri veriler hakkında tablo şeklindeki satır ve sütun formatına uymayan verilerin yönetilmesi ihtiyacını doğurmuştur.
    - Bu da **grafik veritabanlarının** geliştirilmesine yol açtı.
  - On yılın ikinci yarısında, işletmelerde **veri analitiği** ve **veri madenciliği** kullanımı yaygınlaştı. Veritabanı sistemleri özellikle bu pazara hizmet etmek için geliştirildi.
    - Bu sistemler, büyük ticari veritabanı sistemlerinin geleneksel satır odaklı depolaması yerine tabloların sütunlara göre depolandığı "sütun depoları" gibi analitik işleme için uygun fiziksel veri organizasyonlarına sahipti.
  - Büyük veri hacimlerinin yanı sıra analitik için kullanılan verilerin çoğunun metinsel veya yarı yapılandırılmış olması, uygulama programcılarının verileri analiz ederken parallelizmi kullanmalarını kolaylaştırmak için ***map-reduce*** gibi programlama frameworklerinin geliştirilmesine yol açtı.
    - Zaman içinde bu özelliklere yönelik destek geleneksel veritabanı sistemlerine taşındı.

Veritabanı Sistemlerinin Tarihçesi

- 2000'ler
  - Yeni veri-yoğun uygulamaların çeşitliliği ve özellikle yeni kurulan firmaların hızlı gelişim ihtiyacı, hafif bir veri yönetimi biçimi sağlayan "**NoSQL (Not Only SQL)**" sistemlerine yol açmıştır.
    - Bu isim, bu sistemlerin her yerde bulunan veritabanı sorgu dili SQL'i desteklememesinden türetilmiştir, ancak bu isim artık genellikle "sadece SQL değil" anlamına gelmektedir.
    - **NoSQL** sistemlerde ilişkisel modele dayalı üst düzey bir sorgulama dilinin olmaması, programcılara yeni veri türleriyle çalışma konusunda daha fazla esneklik sağladı.
    - Bu sistemlerin geleneksel veritabanı sistemlerinin katı veri tutarlılığı desteğinin olmaması, bir uygulamanın dağıtık veri depolarını kullanmasında daha fazla esneklik sağladı.

Veritabanı Sistemlerinin Tarihçesi

- 2010'lar
  - NoSQL sistemlerinin tutarlılık desteğinin olmaması ve bildirimsel sorgulama desteğinin olmaması gibi sınırlamaları, ölçeklenebilirlik ve kullanılabilirlik gibi sağladıkları faydalar karşılığında birçok uygulama (örneğin sosyal ağlar) tarafından kabul edilebilir bulundu.
    - Ancak 2010'ların başlarına gelindiğinde, sınırlamaların programcılar ve veritabanı yöneticileri için hayatı önemli ölçüde daha karmaşık hale getirdiği açıktı.
    - Sonuç olarak bu sistemler, yüksek ölçeklenebilirlik ve kullanılabilirliği desteklemeye devam ederken daha katı tutarlılık kavramlarını destekleyecek özellikler sunacak şekilde evrim geçirdi.

Veritabanı Sistemlerinin Tarihçesi

- 2010'lar
  - İşletmeler, verilerinin depolanması ve yönetimi için giderek daha fazla dış kaynak kullanmaktadır.
  - Kurum içi sistemleri ve uzmanlığı sürdürmek yerine, işletmeler verilerini çeşitli müşteriler için dağıtık sistemlerin "**bulut**" hizmetlerinde depolayabilir.
    - Veriler kullanıcılara web tabanlı servisler aracılığıyla ulaştırılmaktadır.
    - Diğer kuruluşlar sadece verilerinin depolanmasını değil aynı zamanda tüm uygulamalarını da dışarıdan temin etmektedir.
      - **"Servis olarak yazılım (software as a service)"** olarak adlandırılan bu tür durumlarda, satıcı yalnızca bir kuruluş için verileri depolamakla kalmaz, aynı zamanda uygulama yazılımını da çalıştırır (ve bakımını yapar).

Bölüm Özeti

- Bir veritabanı yönetim sistemi (DBMS), birbiriyle ilişkili bir veri koleksiyonu ve bu verilere erişmek için bir program koleksiyonundan oluşur. Veriler belirli bir işletmeyi tanımlar.
- Bir VTYS'nin birincil amacı, insanların bilgi alma ve saklama işlemlerinde kullanmaları için hem uygun hem de verimli bir ortam sağlamaktır.
- Veritabanı sistemleri günümüzde her yerde bulunmaktadır ve çoğu insan her gün birçok kez veritabanlarıyla doğrudan ya da dolaylı olarak etkileşime girmektedir.
- Veritabanı sistemleri büyük verileri depolamak için tasarlanmıştır. Verilerin yönetimi, hem bilginin depolanması için yapıların tanımlanmasını hem de bilginin manipülasyonu için mekanizmaların sağlanmasını içerir.
  - Buna ek olarak, veritabanı sistemi, sistem çökmeleri veya yetkisiz erişim girişimleri karşısında depolanan bilgilerin güvenliğini sağlamalıdır. Veriler birden fazla kullanıcı arasında paylaşılacaksa, sistem olası anormal sonuçlardan kaçınmalıdır.

Bölüm Özeti

- Bir veritabanı sisteminin en önemli amaçlarından biri kullanıcılara verilerin soyut bir görünümünü sağlamaktır. Yani sistem, verilerin nasıl depolandığı ve muhafaza edildiğine dair belirli ayrıntıları gizler.
- Bir veritabanının yapısının altında yatan veri modelidir: verileri, veri ilişkilerini, veri semantiğini ve veri kısıtlamalarını tanımlamak için kullanılan kavramsal araçların bir koleksiyonu.
- İlişkisel veri modeli, veritabanlarında veri depolamak için en yaygın kullanılan modeldir. Diğer veri modelleri nesne yönelimli model, nesne-ilişkisel model ve yarı yapılandırılmış veri modelleridir.
- Veri manipülasyon dili (DML), kullanıcıların verilere erişmesini veya bunları manipüle etmesini sağlayan bir dildir.
  - Bir kullanıcının bu verileri tam olarak nasıl elde edeceğini belirtmeden yalnızca hangi verilere ihtiyaç duyulduğunu belirtmesini gerektiren prosedürel olmayan DML'ler günümüzde yaygın olarak kullanılmaktadır.

Bölüm Özeti

- Veri tanımlama dili (DDL), veritabanı şemasını ve verilerin diğer özelliklerini belirtmek için kullanılan bir dildir.
- Veritabanı tasarımı temel olarak veritabanı şemasının tasarımını içerir. Varlık-ilişki (E-R) veri modeli, veritabanı tasarımı için yaygın olarak kullanılan bir modeldir.
  - Verileri, ilişkileri ve kısıtlamaları görüntülemek için uygun bir grafiksel gösterim sağlar.
- Bir veritabanı sistemi çeşitli alt sistemlere sahiptir.
  - Depolama yöneticisi alt sistemi, veritabanında depolanan düşük seviyeli veriler ile uygulama programları ve sisteme gönderilen sorgular arasında arayüz sağlar.
  - Sorgu işlemcisi alt sistemi DDL ve DML ifadelerini derler ve yürütür.
  - Hareket yönetimi, sistem arızalarına rağmen veritabanının tutarlı (doğru) bir durumda kalmasını sağlar.
    - Hareket yöneticisi, eşzamanlı işlem yürütmelerinin çakışma olmadan devam etmesini sağlar.

Bölüm Özeti

- Bir veritabanı sisteminin mimarisi, veritabanı sisteminin üzerinde çalıştığı temel bilgisayar sisteminden büyük ölçüde etkilenir.
  - Veritabanı sistemleri merkezi veya birden fazla makineyi içeren paralel olabilir. Dağıtık veritabanları coğrafi olarak ayrılmış birden fazla makineye yayılır.
- Veritabanı uygulamaları tipik olarak istemci makinelerde çalışan bir frontend bölümüne ve backend'de çalışan bir bölüme ayrılır.
  - İki katmanlı mimarilerde, frontend doğrudan backend'de çalışan bir veritabanı ile iletişim kurar.
  - Üç katmanlı mimarilerde, backend kısmı kendi içinde bir uygulama sunucusu ve bir veritabanı sunucusuna ayrılır.
- Veri analizi teknikleri, verilerden kuralları ve örüntüleri otomatik olarak keşfetmeye çalışır.
  - Veri madenciliği alanı; yapay zeka araştırmacıları ve istatistik analistleri tarafından icat edilen bilgi keşif tekniklerini, son derece büyük veri tabanlarında kullanılmalarını sağlayan verimli uygulama teknikleriyle birleştirir.

Veritabanı Uzmanlığı Kariyerine Hazırlanmak

- Kullandığınız veritabanı türü (OLTP, OLAP veya NoSQL) veya ne tür veritabanı ortamında çalıştığınız (örneğin Oracle, Microsoft, IBM veya Hadoop) ne olursa olsun, bir veritabanı sisteminin başarısı büyük ölçüde veritabanı yapısının ne kadar iyi tasarlandığına bağlıdır.
- Bu ders boyunca, bir veritabanı uzmanı olarak kariyerinizin temelini oluşturan yapı taşlarını öğreneceksiniz.
- Bu yapı taşlarını anlamak ve bunları etkili bir şekilde kullanma becerilerini geliştirmek, sizi bir şirket içinde birçok farklı düzeyde veritabanlarıyla çalışmaya hazırlayacaktır.

Veritabanı Kariyer Fırsatlarının Bir Örneği

İş unvanı	Açıklama	Gerekli Örnek Beceriler
Veritabanı Geliştiricisi (Database Developer)	Veritabanı tabanlı uygulamalar oluşturma ve bakımını yapma	Programlama, veritabanı temelleri, SQL
Veritabanı Tasarımcısı (Database Designer)	Veritabanlarını tasarlama ve bakımını yapma	Sistem tasarımı, veritabanı tasarımı, SQL
Veritabanı yöneticisi (Database Administrator)	DBMS'ni ve veritabanlarını yönetme ve bakımını yapma	Veritabanı temelleri, SQL, şirket/kurum eğitimleri
Veritabanı Analisti (Database Analyst)	Karar destek raporlaması için veritabanları geliştirme	SQL, sorgu optimizasyonu, veri ambarları (data warehouses)
Veritabanı Mimarı (Database Architect)	Veritabanı ortamlarının tasarımı ve gerçekleştirimi (kavramsal, mantıksal ve fiziksel)	DBMS temelleri, veri modelleme, SQL, donanım bilgisi vb.

Veritabanı Kariyer Fırsatlarının Bir Örneği

İş unvanı	Açıklama	Gerekli Örnek Beceriler
Veritabanı Danışmanı (Database Consultant)	Şirketlerin iş süreçlerini iyileştirmek ve belirli hedeflere ulaşmak için veritabanı teknolojilerinden yararlanmasına yardımcı olma	Veritabanı temelleri, veri modelleme, veritabanı tasarımı, SQL, DBMS, donanım, satıcıya özel teknolojiler vb.
Veritabanı Güvenlik Sorumlusu (Database Security Officer)	Veri yönetimi için güvenlik politikalarını uygulama	DBMS temelleri, veritabanı yönetimi, SQL, veri güvenliği teknolojileri vb.
Cloud Computing Data Architect (Bulut Bilişim Veri Mimarı)	Yeni nesil bulut veritabanı sistemleri için altyapının tasarlanması ve uygulanması	İnternet teknolojileri, bulut depolama teknolojileri, veri güvenliği, performans ayarlama, büyük veritabanları vb.
Veri Bilimcisi (Data Scientist)	İçgörüler, ilişkiler ve öngörülebilir davranışlar oluşturmak için büyük miktardaki çeşitli verilerin analizi	Veri analizi, istatistik, ileri matematik, SQL, programlama, veri madenciliği, makine öğrenmesi, veri görselleştirme

Veritabanı Uzmanlığı Kariyerine Hazırlanmak

- Veritabanı teknolojileri,
  - artan miktarda veri (büyük veritabanları),
  - giderek çeşitlenen veri türleri (yarı yapılandırılmış ve yapılandırılmamış veriler) ve
  - artan işlem karmaşıklığı ve hızı

  gibi yeni zorluklara yanıt verecek şekilde sürekli olarak gelişmektedir.

Veritabanı Uzmanlığı Kariyerine Hazırlanmak

- Veritabanı teknolojileri hızla değişse de temel kavramlar ve beceriler değişmez. Bu dersteki veritabanı temelleri, geleneksel OLTP ve OLAP sistemlerinin yanı sıra aşağıdakiler gibi son teknoloji, karmaşık veritabanı teknolojileriyle çalışmak için de bir temel olacaktır:
  - **Çok büyük veritabanları (Very large databases -VLDBs).** Genellikle petabayt aralığındaki büyük miktarda veriyi destekleyen veritabanlarına olan ihtiyacı karşılarlar. (Bir petabayt 1.000 terabayttan fazladır.) VLDB satıcıları arasında Oracle Exadata, IBM Netezza, HP Vertica ve Teradata yer alır. Artık VLDB'lerin yerini Büyük Veri veritabanları (Big Data databases) alıyor.

Veritabanı Uzmanlığı Kariyerine Hazırlanmak

- (devamı) :
  - **Büyük Veri veritabanları (Big Data databases).** Cassandra (Facebook) ve BigTable (Google) gibi ürünler, büyük miktarlardaki "nontabular" verileri yöneten veritabanı uygulamalarının ihtiyaçlarını desteklemek için "columnar-database" teknolojilerini kullanıyor.
  - **Bellek içi veritabanları (In-memory databases).** Çoğu büyük veritabanı satıcısı, daha hızlı veritabanı işleme ihtiyacını karşılamak için bir tür bellek içi veritabanı desteği de sunar. Bellek içi veritabanları, verilerinin çoğunu daha yavaş olan ikincil depolama (sabit diskler) yerine birincil bellekte (RAM) depolar. Bellek içi veritabanları arasında IBM SolidDB ve Oracle TimesTen bulunur.
  - **Bulut veritabanları (Cloud databases).** Şirketler artık bulut veritabanı hizmetlerini ortamlarına hızlı bir şekilde veritabanı sistemleri eklemek ve aynı zamanda yeni bir DBMS'ine toplam sahip olma maliyetini düşürmek için kullanabilirler. Bulut veritabanı, yerel bir DBMS'nin tüm avantajlarını sunar, ancak şirketinizin ağ altyapısında yer almak yerine İnternette bulunur.

Kaynaklar

- Silberschatz, A., Korth, H. F., & Sudarshan, S. (7th edition). Database system concepts.
- Coronel, C., & Morris, S. (14th edition). Database systems: design, implementation and management. Cengage learning.

1.Bölümün Sonu