Chapter

2 Data Models

Bu bölümü tamamladıktan sonra şunları öğreneceksiniz:

2-1 Data Modellerinin neden önemli olduğu

2-2 Basic Data-Modeling Building Blocklarını Açıklayabilir hale gelme

2-3 İş kurallarının ne olduğunu ve veritabanı tasarımını nasıl etkilediğini tanımlayabilme

2-4 Başlıca veri modellerinin nasıl geliştiğini anlayabilme

2-5 Ortaya çıkan alternatif veri modellerini ve karşıladıkları ihtiyaçları bilme

2-6 Veri modellerinin soyutlama seviyelerine göre nasıl sınıflandırabileceğini açıklayabilme

Önizleme

Bu bölüm, veri modellemeyi inceler. Veri modelleme, veritabanı tasarımı sürecinin ilk adımıdır ve gerçek dünya nesneleri ile bilgisayar veritabanı arasında bir köprü görevi görür.

Veritabanı tasarımının en zorlu problemlerinden biri, tasarımcıların, programcıların ve son kullanıcıların veriyi farklı şekillerde görmesidir. Sonuç olarak, aynı veriye dair farklı bakış açıları, bir kuruluşun gerçek işleyişini yansıtmayan ve dolayısıyla son kullanıcı ihtiyaçlarını ve veri verimliliği gereksinimlerini karşılamayan veritabanı tasarımlarına yol açabilir. Bu tür hatalardan kaçınmak için veritabanı tasarımcıları, verinin doğasını ve kuruluş içindeki çeşitli kullanım alanlarını tam olarak tanımlamalıdır. Veritabanı tasarımcıları, programcılar ve son kullanıcılar arasındaki iletişim sık ve açık olmalıdır.

Veri modelleme, veritabanı tasarımındaki karmaşıklıkları azaltarak daha kolay anlaşılabilir soyutlamalar sunar ve varlıkları, ilişkileri ve veri dönüşümlerini tanımlayarak iletişimi netleştirir.

Öncelikle bazı temel veri modelleme kavramlarını ve mevcut veri modellerinin önceki modellerden nasıl geliştiğini öğreneceksiniz. Bu veritabanı modellerinin gelişimini takip etmek, kitabın geri kalanında ele alınacak veritabanı tasarım ve uygulama sorunlarını anlamanıza yardımcı olacaktır. Kronolojik sırayla, hiyerarşik ve ağ modelleri, ilişkisel model ve varlık-ilişki (ER) modeli ile tanışacaksınız.

Ayrıca, veri modelleme aracı olarak varlık-ilişki diyagramının (ERD) kullanımını ve ER diyagramlarında kullanılan farklı gösterimleri öğreneceksiniz. Daha sonra, nesne yönelimli (OO) model ve nesne/ilişkisel model ile tanışacaksınız. Ardından, günümüzde çok büyük sosyal medya veri kümelerini verimli ve etkili bir şekilde yönetme ihtiyacını karşılamak için kullanılan yeni NoSQL veri modelini öğreneceksiniz. Son olarak, farklı veri soyutlama seviyelerinin, aynı veriye dair farklı bakış açılarını nasıl uzlaştırdığını keşfedeceksiniz.

Veri Dosyaları ve Geçerli Formatlar

	MS Access	Oracle	MS SQL	MySQL
Ch02_InsureCo	Yes	Yes	Yes	Yes
Ch02_DealCo	Yes	Yes	Yes	Yes
Ch02_TinyCollege	Yes	Yes	Yes	Yes

Veri Dosyaları için detaylı bilgi: cengage.com

2-1 Veri modeli ve modelleme

data modeling (Veri modelleme)
Belirli bir problem alanı için özel bir veri modeli oluşturma süreci.

data model (Veri Modeli)
Genellikle grafiksel olarak gösterilen, karmaşık bir "gerçek dünya" veri yapısının temsili. Veri modelleri, Veritabanı Yaşam Döngüsü'nün veritabanı tasarım aşamasında kullanılır.

Veritabanı tasarımı, veritabanı yapısının son kullanıcı verilerini depolamak ve yönetmek için nasıl kullanılacağına odaklanır. Veritabanı tasarımının ilk adımı olan veri modelleme, belirli bir problem alanı için özel bir veri modeli oluşturma sürecini ifade eder. (Problem alanı, gerçek dünya ortamında açıkça tanımlanmış, belirli bir kapsamı ve sınırları olan, sistematik olarak ele alınacak bir alandır.)

Veri modeli, genellikle grafiksel olarak gösterilen, daha karmaşık gerçek dünya veri yapılarının nispeten basit bir temsilidir. Genel anlamda bir model, daha karmaşık bir gerçek dünya nesnesinin veya olayının soyutlamasıdır. Bir modelin temel işlevi, gerçek dünya ortamındaki karmaşıklıkları anlamanıza yardımcı olmaktır.

Veritabanı ortamında bir veri modeli, veri yapılarını ve bunların özelliklerini, ilişkilerini, kısıtlamalarını, dönüşümlerini ve diğer bileşenlerini belirli bir problem alanını desteklemek amacıyla temsil eder.

Not

Veri modeli ve veritabanı modeli terimleri sıklıkla birbirinin yerine kullanılır. Bu kitapta, veritabanı modeli terimi, bir veri modelinin belirli bir veritabanı sisteminde uygulanmasını ifade etmek için kullanılır.

Veri modelleme, yinelemeli ve ilerleyici bir süreçtir. Problem alanının basit bir anlayışıyla başlanır ve anlayış arttıkça, veri modelinin ayrıntı seviyesi de artar. Doğru bir şekilde yapıldığında, nihai veri modeli, tüm son kullanıcı gereksinimlerini karşılayacak bir veritabanı inşa etmek için gereken tüm talimatları içeren bir "plan" olur. Bu plan, hem metin açıklamaları içeren (açık ve kesin bir dille yazılmış) hem de ana veri öğelerini gösteren net ve faydalı diyagramlar içeren anlatımsal ve grafiksel bir doğaya sahiptir.

Not

Uygulama için hazır bir veri modeli en az aşağıdaki bileşenleri içermelidir:

- Son kullanıcı verilerini depolayacak veri yapısının bir açıklaması
- Verinin bütünlüğünü garanti altına almak için uygulanabilir kurallar seti
- Gerçek dünya veri dönüşümlerini destekleyecek bir veri manipülasyon metodolojisi

Geleneksel olarak, veritabanı tasarımcıları iyi bir veri modeli geliştirmelerine yardımcı olması için sağduyularına güvenirlerdi. Ne yazık ki, sağduyu genellikle bakış açısına bağlıdır ve çoğu zaman pek çok deneme-yanılma sonrasında gelişir. Örneğin, bu dersteki her öğrenci bir video akış sitesine yönelik bir veri modeli oluşturmak zorunda kalsa, her birinin farklı bir model ortaya koyması çok olasıdır. Hangisi doğru olur? Basit cevap şu olur: "Tüm son kullanıcı gereksinimlerini karşılayan model" ve birden fazla doğru çözüm olabilir! Neyse ki, veritabanı tasarımcıları, veritabanı modellemede hata yapma potansiyelini önemli ölçüde azaltan mevcut veri modelleme yapılarını ve güçlü veritabanı tasarım araçlarını kullanırlar. Aşağıdaki bölümlerde, mevcut veri modellerinin gerçek dünya verilerini nasıl temsil ettiğini ve farklı veri soyutlama seviyelerinin veri modellemeyi nasıl kolaylaştırdığını öğreneceksiniz.

2-2 Data Modellerinin Önemi

Veri modelleri, tasarımcı, uygulama programcısı ve son kullanıcı arasında etkileşimi kolaylaştırabilir. İyi geliştirilmiş bir veri modeli, veritabanı tasarımının geliştirildiği organizasyonun daha iyi anlaşılmasına bile katkı sağlayabilir. Kısacası, veri modelleri bir iletişim aracıdır. Veri modellemenin bu önemli yönü, bir müşterinin şu şekilde özetlediği gibi ifade edilmiştir: "Bu işi ben kurdum, yıllarca bu işle uğraştım ve şimdiye kadar tüm parçaların gerçekten nasıl bir araya geldiğini ilk defa gerçekten anladım."

Veri modellemenin önemi abartılamaz. Veri, bir sistem tarafından kullanılan en temel bilgiyi oluşturur. Uygulamalar, verileri yönetmek ve veriyi bilgiye dönüştürmeye yardımcı olmak için yaratılır, ancak veri farklı insanlar tarafından farklı şekillerde görülür. Örneğin, bir şirket yöneticisinin bakış açısını, bir şirket memurunun bakış açısıyla karşılaştırın. Her ikisi de aynı şirkette çalışıyor olsa da, yönetici, memura göre şirket verilerine daha geniş bir bakış açısına sahip olma eğilimindedir.

Farklı yöneticiler bile veriyi farklı şekillerde görür. Örneğin, bir şirket başkanı, şirketin bölümlerini ortak bir (veritabanı) vizyona bağlayabilmesi gerektiğinden, veriye daha evrensel bir bakış açısı benimseyecektir. Aynı şirketteki bir satın alma yöneticisi ise, veriye daha dar bir bakış açısına sahip olma eğilimindedir, tıpkı şirketin envanter yöneticisi gibi. Etkili bir şekilde, her departman yöneticisi, şirketin verilerinin bir alt kümesiyle çalışır. Envanter yöneticisi envanter seviyeleriyle, satın alma yöneticisi ise ürünlerin maliyeti ve bu ürünlerin tedarikçileriyle ilgilenir.

Uygulama programcıları, veriye farklı bir bakış açısına sahiptir ve daha çok veri konumu, formatı ve özel raporlama gereksinimleriyle ilgilenirler. Temelde, uygulama programcıları, şirket politikalarını ve prosedürlerini çeşitli kaynaklardan uygun arayüzlere, raporlara ve sorgu ekranlarına çevirirler.

Veri ve bilgi kullanıcıları ile üreticilerinin farklı bakış açıları genellikle kör insanlar ve fil hikayesini yansıtır: Filtin ağzını hisseden kör kişi, filin bacağını ya da kuyruğunu hisseden kişiden oldukça farklı bir bakış açısına sahip olacaktır. Tıpkı bunun gibi, tüm fili görmek gereklidir. Aynı şekilde, bir ev rastgele odalardan oluşan bir koleksiyon değildir; bir ev inşa etmek için, bir kişi önce planların sağladığı genel bakış açısına sahip olmalıdır. Benzer şekilde, sağlıklı bir veri ortamı, uygun bir veri modeline dayanan genel bir veritabanı planına ihtiyaç duyar.

İyi bir veritabanı planı mevcut olduğunda, uygulama programcısının veriye bakış açısının yöneticinin veya son kullanıcının bakış açısından farklı olması önemli değildir. Tersine, iyi bir veritabanı planı mevcut olmadığında, sorunlar ortaya çıkma olasılığı yüksektir. Örneğin, bir envanter yönetim programı ile bir sipariş giriş sistemi, çelişkili ürün numaralandırma şemaları kullanabilir ve bu da şirkete binlerce hatta milyonlarca dolara mal olabilir.
Unutmayın ki bir ev planı bir soyutlamadır; planın içinde yaşamazsınız. Benzer şekilde, veri modeli bir soyutlamadır; gerekli veriyi veri modelinden çıkaramazsınız. Tıpkı bir plan olmadan iyi bir ev inşa etmenin olasılığı düşük olduğu gibi, uygun bir veri modeli oluşturmadan iyi bir veritabanı oluşturmak da aynı şekilde olasılık dışıdır.

2-3 Veri Modelinin Temel Yapı Taşları (Data Model Basic Building Blocks)

varlık (entity)
Veri saklanabilecek bir kişi, yer, şey, kavram veya olaydır.
özellik (attribute)
Bir varlık veya nesnenin özelliğidir. Bir özelliğin adı ve veri tipi vardır.
ilişki (relationship)
Varlıklar arasındaki bir ilişkiyi tanımlar.
bir-e-bir (1:1 veya 1..1) ilişkisi
İki veya daha fazla varlık arasındaki ilişkilerde, bir varlık örneği yalnızca ilgili varlıkla bir örnekle ilişkilidir.
bir-e-çok (1:M veya 1..*) ilişkisi
İki veya daha fazla varlık arasındaki ilişkilerde, bir varlık örneği, ilgili varlıkların birçok örneğiyle ilişkilidir.
çok-e-çok (M:N veya ..) ilişkisi
İki veya daha fazla varlık arasındaki ilişkilerde, bir varlık örneği, ilgili varlıkların birçok örneğiyle ilişkilidir ve aynı şekilde, ilgili varlık örneği de bir varlık örneğiyle birçok kez ilişkilidir.

Tüm veri modellerinin temel yapı taşları varlıklar, özellikler, ilişkiler ve kısıtlamalardır. Bir varlık, veri toplanıp saklanacak bir kişi, yer, şey, kavram veya olaydır. Bir varlık, gerçek dünyada belirli bir nesne türünü temsil eder, yani bir varlık "ayırt edilebilir"dir—yani, her varlık örneği benzersiz ve farklıdır. Örneğin, bir MÜŞTERİ varlığı, John Smith, Pedro Dinamita ve Dana Strickland gibi birçok farklı müşteri örneğine sahip olabilir. Varlıklar, müşteriler veya ürünler gibi fiziksel nesneler olabileceği gibi, uçuş rotaları veya müzik konserleri gibi soyutlamalar da olabilir.
Bir özellik, bir varlığın karakteristiğidir. Örneğin, bir MÜŞTERİ varlığı, müşteri soyadı, müşteri adı, müşteri telefon numarası, müşteri adresi ve müşteri kredi limiti gibi özelliklerle tanımlanabilir. Özellikler, dosya sistemlerinde alanların karşılığıdır.
Bir ilişki, varlıklar arasındaki bir bağlantıyı tanımlar. Örneğin, müşteriler ve temsilciler arasında şöyle bir ilişki vardır: bir temsilci birçok müşteriye hizmet verebilir ve her müşteri yalnızca bir temsilci tarafından hizmet alabilir. Veri modelleri üç tür ilişki kullanır: bir-e-çok, çok-e-çok ve bir-e-bir. Veritabanı tasarımcıları genellikle sırasıyla 1:M veya 1..*, M:N veya .., ve 1:1 veya 1..1 kısaltmalarını kullanır. (M:N notasyonu, çok-e-çok ilişkisi için standart bir etiket olsa da, M:M etiketi de kullanılabilir.) Aşağıdaki örnekler, üç ilişki türü arasındaki farkları açıklamaktadır.

- **Bir-e-çok (1:M veya 1..*) ilişkisi.**

Bir ressam birçok farklı tablo yapar, ancak her biri yalnızca bir ressam tarafından yapılır. Bu nedenle, ressam ("bir") ile tablolar ("çok") arasında bir ilişki vardır. Bu ilişkiyi, "RESSAM tablo YAPAR" olarak tanımlayabiliriz ve veritabanı tasarımcıları bu ilişkiyi 1:M olarak etiketler. Benzer şekilde, bir müşteri ("bir") birçok fatura oluşturabilir, ancak her fatura ("çok") yalnızca bir müşteri tarafından oluşturulur. "MÜŞTERİ fatura OLUŞTURUR" ilişkisi de 1:M olarak etiketlenir.

- **Çok-e-çok (M:N veya ..) ilişkisi.**

Bir çalışan birçok iş becerisi öğrenebilir ve her iş becerisi birçok çalışan tarafından öğrenilebilir. Veritabanı tasarımcıları bu ilişkiyi "ÇALIŞAN beceri ÖĞRENİR" olarak etiketler ve M:N etiketini kullanır. Benzer şekilde, bir öğrenci birçok ders alabilir ve her ders birçok öğrenci tarafından alınabilir; dolayısıyla "ÖĞRENCİ ders ALIR" ilişkisi de M:N etiketini alır.

- **Bir-e-bir (1:1 veya 1..1) ilişkisi.**

Bir perakende şirketinin yönetim yapısı, her mağazanın yalnızca bir çalışan tarafından yönetilmesini gerektirebilir. Buna karşılık, her mağaza yöneticisi, bir çalışan olan mağaza yöneticisi, yalnızca bir mağazayı yönetir. Bu nedenle, "ÇALIŞAN mağaza YÖNETİR" ilişkisi 1:1 olarak etiketlenir.

Yukarıdaki tartışma, her ilişkiyi iki yönde de tanımlamıştır; yani ilişkiler bidireksiyondur:

- Bir MÜŞTERİ birçok FATURA oluşturabilir.
- Her bir FATURA yalnızca bir MÜŞTERİ tarafından oluşturulur.

Bir kısıtlama, veriye uygulanan bir sınırlamadır. Kısıtlamalar, veri bütünlüğünü sağlamaya yardımcı oldukları için önemlidir. Kısıtlamalar genellikle şu şekilde ifade edilen kurallar biçiminde ifade edilir:

- Bir çalışanın maaşı 6.000 ile 350.000 arasında olmalıdır.
- Bir öğrencinin GPA'sı 0.00 ile 4.00 arasında olmalıdır.
- Her dersin bir ve yalnız bir öğretmeni olmalıdır.

Varlıkları, özellikleri, ilişkileri ve kısıtlamaları doğru bir şekilde nasıl tanımlarsınız? İlk adım, modellediğiniz problem alanı için iş kurallarını açıkça tanımlamaktır. Bu kurallar, veri modelinizin temelini oluşturacak ve doğru varlıklar, özellikler ve ilişkiler belirlemenize yardımcı olacaktır.

Kısıtlama (Constraint)
Veri üzerinde konulan bir sınırlamadır ve genellikle kurallar şeklinde ifade edilir. Örneğin, "Bir öğrencinin GPA'sı 0.00 ile 4.00 arasında olmalıdır."

2-4 Business Rules (İş Kuralları)

Veritabanı tasarımcıları, veri modelini oluşturmak için kullanılacak varlıkları, özellikleri ve ilişkileri seçerken, genellikle bir organizasyondaki verilerin türlerini, verilerin nasıl kullanıldığını ve hangi zaman dilimlerinde kullanıldığını anlamak için derinlemesine bir anlayışa sahip olurlar. Ancak bu tür veriler ve bilgiler, yalnızca kendileri, toplam işin gereksinimlerini karşılayacak bir anlayış sağlamaz. Bir veritabanı bakış açısına göre, verilerin toplanması ancak doğru şekilde tanımlanmış iş kuralları yansıtıldığında anlam kazanır.

İş kuralı, belirli bir organizasyon içindeki bir politika, prosedür veya ilkenin kısa, net ve belirsiz olmayan bir açıklamasıdır. Aslında iş kuralları yanlış adlandırılmıştır: Herhangi bir organizasyona, büyük ya da küçük, bir iş yerine, bir devlet birimine, bir dini gruba veya bir araştırma laboratuvarına uygulandıkları için, veriyi depolayan ve bilgi üretmek için kullanan her tür organizasyona uygulanabilirler.

Bir organizasyonun operasyonlarının ayrıntılı bir tanımından türetilen iş kuralları, o organizasyonun ortamında eylemleri oluşturmak ve yürütmek için yardımcı olur. İş kuralları yazılı olarak ifade edilmeli ve organizasyonun operasyonel ortamındaki herhangi bir değişikliği yansıtacak şekilde güncellenmelidir.

Doğru yazılmış iş kuralları, varlıkları, özellikleri, ilişkileri ve kısıtlamaları tanımlamak için kullanılır. “Bir ajan birçok müşteriye hizmet verebilir, her müşteri yalnızca bir ajan tarafından hizmet verilebilir” gibi ilişki ifadelerini gördüğünüzde, iş kuralları devreye girmektedir. Bu kitapta, özellikle veri modelleme ve veritabanı tasarımına odaklanan bölümlerde iş kurallarının uygulamasını göreceksiniz.

İş kuralları etkili olabilmek için kolayca anlaşılabilir olmalı ve organizasyondaki herkesin kuralların ortak bir yorumunu paylaşmasını sağlamak için geniş bir şekilde yayımlanmalıdır. İş kuralları, şirketin gözünde verinin ana ve ayırt edici özelliklerini basit bir dille açıklar. İşte bazı iş kuralı örnekleri:

- Bir müşteri birçok fatura oluşturabilir.
- Bir fatura yalnızca bir müşteri tarafından oluşturulabilir.
- Bir eğitim oturumu, 10 kişiden az veya 30 kişiden fazla çalışan için planlanamaz.

Bu iş kuralları, varlıkları, ilişkileri ve kısıtlamaları tanımlar. Bazı iş kuralları ayrıca bir dizi veri işleme adımını (programlama koduna dönüşebilecek) belirleyebilir. Örneğin, ilk iki iş kuralı, iki varlık (MÜŞTERİ ve FATURA) ve bu iki varlık arasında bir 1:M ilişkisi oluşturur. Üçüncü iş kuralı, bir kısıtlama (en az 10 kişi ve en fazla 30 kişi) ve iki varlık (ÇALIŞAN ve EĞİTİM) tanımlar ve ayrıca ÇALIŞAN ile EĞİTİM arasındaki bir ilişkiyi ima eder.

İş Kuralı
Bir organizasyon içindeki bir politika, prosedür veya prensibin tanımını ifade eder. Örneğin, bir pilot 24 saatlik bir dönemde 10 saatten fazla görevde olamaz veya bir profesör bir dönemde en fazla dört ders verebilir.

2-4a İş Kurallarını Keşfetmek

İş kurallarının ana kaynakları, şirket yöneticileri, politika belirleyiciler, departman yöneticileri ve şirketin prosedürleri, standartları ve operasyonel kılavuzları gibi yazılı belgeler olabilir. İş kurallarını daha hızlı ve doğrudan keşfetmek için ise, son kullanıcılarla yapılan doğrudan görüşmeler önemli bir kaynak sağlar. Bu görüşmeler, iş kurallarının doğru ve etkili bir şekilde tanımlanmasına yardımcı olabilir.

Ne yazık ki, algılar farklılık gösterdiği için, son kullanıcılar bazen iş kurallarını belirtme konusunda daha az güvenilir bir kaynak olabilir. Örneğin, bir bakım departmanı teknisyeni, her teknisyenin bakım prosedürünü başlatabileceğini düşünebilir, ancak aslında yalnızca denetim yetkisi olan teknisyenler bu tür bir görevi yerine getirebilir. Bu tür bir ayrım önemsiz gibi görünebilir, ancak büyük yasal sonuçlar doğurabilir. Son kullanıcılar iş kurallarının geliştirilmesinde önemli katkılar sağlasa da, son kullanıcı algılarının doğrulanması önemlidir. Aynı işi yapan birkaç kişiyle yapılan görüşmeler çoğu zaman işin bileşenlerine dair çok farklı algılar ortaya koyar. Böyle bir keşif "yönetim sorunları"na işaret edebilir, ancak bu genel tespit veritabanı tasarımcısına yardımcı olmaz. Veritabanı tasarımcısının işi, bu farklılıkları uzlaştırmak ve iş kurallarının uygun ve doğru olduğundan emin olmak için uzlaşmayı doğrulamaktır.

İş kurallarını belirleme ve belgelemeye yönelik süreç, veritabanı tasarımı açısından birkaç nedenle çok önemlidir:

- Şirketin veri hakkındaki görüşünü standardize etmeye yardımcı olur.
- Kullanıcılar ve tasarımcılar arasında bir iletişim aracı olabilir.
- Tasarımcının verinin doğasını, rolünü ve kapsamını anlamasını sağlar.
- Tasarımcının iş süreçlerini anlamasını sağlar.
- Tasarımcının uygun ilişki katılım kurallarını ve kısıtlamaları geliştirmesine ve doğru bir veri modeli oluşturmasına olanak tanır.

Tabii ki, tüm iş kuralları modellenemez. Örneğin, "Hiçbir pilot 24 saatlik bir sürede 10 saatten fazla uçuş yapamaz" gibi bir iş kuralı doğrudan veritabanı modelinde modellenemez. Ancak, bu tür bir iş kuralı, uygulama yazılımı tarafından temsil edilebilir ve uygulanabilir.

2-4b İş Kurallarını Veri Modeli Bileşenlerine Çevirme

İş kuralları, varlıkların, özniteliklerin, ilişkilerin ve kısıtlamaların doğru bir şekilde tanımlanması için temel oluşturur. Gerçek dünyada, nesneleri tanımlamak için isimler kullanılır. İş ortamı, nesneleri takip etmek istiyorsa, bu nesneler için belirli iş kuralları olacaktır. Genel bir kural olarak, iş kuralındaki bir isim, modelde bir varlık olarak çevrilecektir ve bu isimleri birleştiren bir fiil (aktif veya pasif) bir ilişkiyi ifade eder. Örneğin, "bir müşteri birçok fatura oluşturabilir" iş kuralı iki isim içerir (müşteri ve fatura) ve bu isimleri ilişkilendiren bir fiil (oluşturmak). Bu iş kuralından şu çıkarımlar yapılabilir:

- Müşteri ve fatura, çevredeki ilgi çekici nesnelerdir ve kendi varlıklarıyla temsil edilmelidir.
- Müşteri ile fatura arasında bir "oluşturma" ilişkisi vardır.

İlişki tipini doğru bir şekilde belirlemek için, ilişkilerin çift yönlü olduğunu göz önünde bulundurmalısınız; yani her iki yönde de giderler. Örneğin, "bir müşteri birçok fatura oluşturabilir" iş kuralı, "bir fatura yalnızca bir müşteri tarafından oluşturulur" iş kuralıyla tamamlanır. Bu durumda, ilişki birden çoğa (1:M) olacaktır. Müşteri "1" tarafı, fatura ise "çok" tarafıdır.

İlişki türünü doğru bir şekilde belirlemek için genellikle iki soru sormalısınız:

1. Bir A örneğiyle kaç B örneği ilişkilidir?
2. Bir B örneğiyle kaç A örneği ilişkilidir?

Örneğin, öğrenci ile sınıf arasındaki ilişkiyi değerlendirirken şu iki soruyu sorabilirsiniz:

- Bir öğrenci kaç sınıfa kayıt olabilir? Cevap: Birçok sınıf.
- Bir sınıfa kaç öğrenci kaydolabilir? Cevap: Birçok öğrenci.

Bu nedenle, öğrenci ve sınıf arasındaki ilişki çoktan çoğa (M:N) bir ilişkidir. Kitap boyunca, varlıklar arasındaki ilişkileri belirlemek için birçok fırsatınız olacak ve yakında bu süreç ikinci doğa haline gelecektir.

2-4c Naming Conventions (İsimlendirme Kuralları)

İş kurallarının veri modeli bileşenlerine dönüştürülmesi sırasında, varlıklar, nitelikler, ilişkiler ve kısıtlamalar tanımlanır. Bu tanımlama süreci, nesneyi, problem alanındaki diğer nesnelerden benzersiz ve ayırt edilebilir bir şekilde adlandırmayı içerir. Bu nedenle, keşfettiğiniz nesneleri adlandırırken özel bir dikkat göstermek önemlidir.

Varlık adları, iş ortamındaki nesneleri tanımlayıcı olmalı ve kullanıcıların aşina olduğu terminoloji kullanılmalıdır. Bir nitelik adı da, o niteliğin temsil ettiği veriyi tanımlayıcı olmalıdır. Ayrıca, bir niteliğin adını, bulunduğu varlığın adı veya kısaltmasıyla öneklemek iyi bir uygulamadır. Örneğin, CUSTOMER varlığında, müşterinin kredi limiti CUS_CREDIT_LIMIT olarak adlandırılabilir. CUS, niteliğin CUSTOMER varlığına ait olduğunu gösterirken, CREDIT_LIMIT, nitelikte yer alacak veriyi kolayca tanımayı sağlar. Bu, ilerleyen bölümlerde, varlıklar arasındaki ilişkileri belirlemek için ortak nitelikler kullanma gerekliliğini öğrendiğinizde daha da önemli hale gelecektir. Doğru bir isimlendirme kuralı kullanmak, veri modelinin tasarımcı, uygulama programcısı ve son kullanıcı arasında iletişimi kolaylaştırma yeteneğini artıracaktır. Aslında, doğru bir isimlendirme kuralı, modelinizin kendiliğinden belge oluşturmasına önemli ölçüde yardımcı olabilir.

Çevrimiçi İçerik

Hiyerarşik ve ağ modelleri büyük ölçüde tarihsel ilgi taşır, ancak mevcut veritabanı profesyonellerinin ilgisini çeken bazı unsurlar ve özellikler içerir. Bu iki modelin teknik detayları sırasıyla Ek K ve L'de tartışılmaktadır, bunlar www.cengage.com adresinde mevcuttur. Ek G, nesne yönelimli (OO) modele ayrılmıştır. Ancak, ilişkisel modelin baskın piyasa varlığı göz önüne alındığında, kitabın çoğu, ilişkisel modele odaklanmaktadır.

Not

Modern veritabanı sistemleri, isimlendirme kurallarını kolaylaştıran nitelikler için tablo adı öneklerinin kullanılmasına izin verir. Örneğin, CUSTOMER.CREDIT_LIMIT, CUSTOMER tablosundaki CREDIT_LIMIT niteliğini belirtir.

2-5 Veri Modellerinin Evrimi

Daha iyi veri yönetimi arayışı, önceki modelin kritik eksikliklerini çözmeyi ve sürekli evrilen veri yönetimi ihtiyaçlarına çözüm sunmayı amaçlayan birkaç modele yol açmıştır. Bu modeller, bir veritabanının ne olduğunu, ne yapması gerektiğini, hangi yapıları kullanması gerektiğini ve bu yapıların uygulanmasında hangi teknolojilerin kullanılacağını belirten düşünce okullarını temsil eder. Belki kafa karıştırıcı bir şekilde, bu modeller, önceki bölümde tartışılan grafiksel veri modelleriyle aynı şekilde veri modelleri olarak adlandırılır. Bu bölüm, ana veri modellerinin kabaca kronolojik sırasıyla bir özetini sunar. "Yeni" veritabanı kavramlarının ve yapıların, "eski" veri modeli kavramlarına ve yapılarına şaşırtıcı derecede benzerlik gösterdiğini keşfedeceksiniz. Tablo 2.1, ana veri modellerinin evrimini izlemektedir.

hiyerarşik model

Veritabanı geliştirmesinin temelini oluşturan erken dönem bir veritabanı modelidir. Bu model, her kaydın bir segment olarak adlandırıldığı tersine çevrilmiş bir ağaç yapısına dayanır. En üstteki kayıt, kök segmenttir. Her segmentin, doğrudan altındaki segmentle 1:M ilişkisi vardır.

2-5a Hiyerarşik ve Ağ Modelleri

Hiyerarşik model, 1960'larda, Apollo roketinin 1969'da aya iniş yapması gibi karmaşık üretim projeleri için büyük miktarda veriyi yönetmek amacıyla geliştirilmiştir. Modelin temel mantıksal yapısı, ters bir ağaçla temsil edilir. Hiyerarşik yapı seviyeler veya segmentler içerir. Bir segment, dosya sisteminin kayıt türünün karşılığıdır. Hiyerarşide, daha yüksek bir katman, doğrudan altındaki segmentin ebeveyni olarak algılanır.

segment

Hiyerarşik veri modelinde, dosya sisteminin kayıt türünün karşılığıdır.

Table 2.1 Evolution of Major Data Models

Jenerasyon	Zaman	Veri Modeli	Örnekler	Yorumlar
İlk	1960–1970'ler	File System	VMS/VSAM	Başlangıçta IBM ana bilgisayar sistemlerinde kullanıldı, kayıtları yönetmeye odaklandı, ilişkileri değil.
İkinci	1970ler	Hierarchical and network	IMS, ADABAS, IDS-II	Erken dönem veritabanı sistemleri Navigasyonel erişim
Third	1970'lerin ortaları	Relational	DB2 Oracle MS SQL Server MySQL	Kavramsal basitlik Varlık-ilişki (ER) modellemesi ve ilişkisel veri modellemesi desteği
Dördüncü	1980'lerin ortaları	Objec t- oriented Objec t/relational (O/R)	Versant Objectivity/DB DB2 UDB Oracle	Nesne/ilişkisel, nesne veri türlerini destekler Yıldız Şeması, veri ambarı desteği sağlar Web veritabanları yaygınlaşır
Beşinci	1990'ların ortaları	XML Hybrid DBMS	dbXML Tamino DB2 UDB Oracle MS SQL Server PostgreSQL	Yapısız veri desteği O/R modeli XML belgelerini destekler Hibrit DBMS, ilişkisel veritabanlarına nesne ön yüzü ekler Büyük veritabanlarını (terabayt boyutunda) destekler
Gelişen Modeller: NoSQL	2000'lerin başları	Key-value store Column store	SimpleDB (Amazon) BigTable (Google) Cassandra (Apache) MongoDB Riak	Dağıtık, yüksek ölçeklenebilir Yüksek performans, hata toleransı Çok büyük depolama (petabaytlar) Seyrek veri için uygun Özel uygulama programlama arayüzü (API)

network model (ağ modeli)
Verileri 1:M ilişkilerinde kayıt türlerinin bir koleksiyonu olarak temsil eden erken dönem bir veri modeli.

schema (Şema)
Veritabanı nesnelerinin (tablolar, dizinler, görünümler ve sorgular gibi) birbirleriyle ilişkili mantıksal bir gruplandırılması.

subschema (Alt Şema)
Veritabanının, uygulama programlarıyla etkileşime giren kısmı.

data manipulation language (DML) (Veri Manipülasyon Dili)
Bir son kullanıcının veritabanındaki verileri manipüle etmesine olanak tanıyan komutlar kümesi; örneğin SELECT, INSERT, UPDATE, DELETE, COMMIT ve ROLLBACK.

data definition language (DDL) (Veri Tanımlama Dili)
Bir veritabanı yöneticisinin veritabanı yapısını, şemasını ve altşemasını tanımlamasına olanak tanıyan dil.

altında, buna çocuk denir. Hiyerarşik model, bir ebeveyn ve onun çocuk segmentleri arasında bir dizi birden fazla-tek (1:M) ilişkiyi tasvir eder. (Her ebeveynin birden fazla çocuğu olabilir, ancak her çocuğun yalnızca bir ebeveyni vardır.)

Ağ modeli, hiyerarşik modelden daha karmaşık veri ilişkilerini daha etkili bir şekilde temsil etmek, veritabanı performansını artırmak ve bir veritabanı standardı dayatmak amacıyla oluşturulmuştur. Ağ modelinde kullanıcı, ağ veritabanını 1:M ilişkilerindeki bir kayıt koleksiyonu olarak algılar. Ancak, hiyerarşik modelin aksine, ağ modeli bir kaydın birden fazla ebeveyni olmasına izin verir. Ağ veritabanı modeli bugün genellikle kullanılmasa da, ağ modelinde ortaya çıkan standart veritabanı kavramlarının tanımları modern veri modelleri tarafından hâlâ kullanılmaktadır:

Şema, veritabanı yöneticisi tarafından görülen tüm veritabanının kavramsal organizasyonudur.

Altşema, veritabanındaki verilerden istenen bilgiyi üreten uygulama programları tarafından "görülen" veritabanının bölümünü tanımlar.

Veri manipülasyon dili (DML), verilerin yönetilebileceği ortamı tanımlar ve veritabanındaki verilerle çalışmak için kullanılır.

Şema veri tanımlama dili (DDL), veritabanı yöneticisinin şema bileşenlerini tanımlamasına olanak tanır.

Bilgi ihtiyaçları arttıkça ve daha sofistike veritabanları ile uygulamalar gerektikçe, ağ modeli çok hantal hale gelmiştir. İstenmeyen sorgu yeteneğinin eksikliği, programcıları en basit raporları üretmek için bile gereken kodu oluşturmak zorunda bırakmıştır. Mevcut veritabanları sınırlı veri bağımsızlığı sağlasa da, veritabanındaki herhangi bir yapısal değişiklik hâlâ, veritabanından veri çeken tüm uygulama programlarında kaosa yol açabilirdi. Hiyerarşik ve ağ modellerinin dezavantajları nedeniyle, 1980'lerde bunlar büyük ölçüde ilişkisel veri modeliyle değiştirilmiştir.

2-5b The Relational Model (İlişkisel Model)

İlişkisel model, 1970 yılında IBM'den E. F. Codd tarafından, "A Relational Model of Data for Large Shared Databanks" (ACM İletişimleri, Haziran 1970, s. 377-387) başlıklı dönüm noktası niteliğindeki makalesinde tanıtılmıştır. İlişkisel model, hem kullanıcılar hem de tasarımcılar için önemli bir atılımı temsil ediyordu. Bir benzetme yapmak gerekirse, ilişkisel model, önceki "standart vites" veritabanlarının yerine "otomatik vites" bir veritabanı üretmiştir. Kavramsal basitliği, gerçek bir veritabanı devrimini başlatmak için zemin hazırlamıştır.

relational model
IBM'den E. F. Codd tarafından 1970'te geliştirilen ilişkisel model, matematiksel küme teorisine dayalıdır ve veriyi bağımsız ilişkiler olarak temsil eder. Her ilişki (tablo), kesişen satır ve sütunlardan oluşan iki boyutlu bir yapı olarak kavramsal olarak temsil edilir. İlişkiler, ortak varlık özelliklerinin (sütunlardaki değerlerin) paylaşılması yoluyla birbirine bağlanır.

Not

Bu bölümde sunulan ilişkisel veritabanı modeli, bir tanıtım ve genel bir bakıştır. Daha ayrıntılı bir tartışma, 3. Bölüm, İlişkisel Veritabanı Modeli'nde yer almaktadır. Aslında, ilişkisel model o kadar önemlidir ki, geri kalan bölümlerin çoğunda yapılacak tartışmaların temeli olarak hizmet edecektir.

İlişkisel modelin temeli, bir ilişki olarak bilinen matematiksel bir kavramdır. Soyut matematiksel teorinin karmaşıklığından kaçınmak için, bir ilişkiyi (bazen tablo olarak adlandırılır) kesişen satırlar ve sütunlardan oluşan iki boyutlu bir yapı olarak düşünebilirsiniz. İlişkideki her satır bir demet (tuple) olarak adlandırılır. Her sütun ise bir niteliği (attribute) temsil eder. İlişkisel model, aynı zamanda ileri düzey matematiksel kavramlara dayanan, veri manipülasyonu için kesin bir dizi yapı da tanımlar.

1970 yılında, Codd'un çalışması zeki ancak uygulanabilir olmayan bir fikir olarak kabul ediliyordu. İlişkisel modelin kavramsal sadeliği, bilgisayar yükü pahasına elde edilmişti; o dönemde bilgisayarlar, ilişkisel modeli uygulayacak güce sahip değildi. Neyse ki, bilgisayar gücü üssel bir şekilde arttı, işletim sistemi verimliliği de aynı şekilde gelişti. Daha da iyisi, bilgisayarların maliyeti hızla düştü, güçleri arttıkça. Bugün, ana bilgisayar atalarına kıyasla çok daha düşük bir maliyetle, kişisel bilgisayarlar bile Oracle, DB2, Microsoft SQL Server, MySQL gibi sofistike ilişkisel veritabanı yazılımlarını çalıştırabiliyor.

İlişkisel veri modeli, çok sofistike bir ilişkisel veritabanı yönetim sistemi (RDBMS) aracılığıyla uygulanır. RDBMS, hiyerarşik ve ağ DBMS sistemlerinin sağladığı aynı temel işlevleri, ilişkisel veri modelini daha kolay anlaşılır ve uygulanabilir kılan birçok diğer işlevle birlikte gerçekleştirir (1. Bölüm'de, DBMS Fonksiyonları bölümünde açıklandığı gibi).

table (relation)
İlişkisel modelde bir varlık kümesini temsil eden, kesişen satırlardan (varlıklardan) ve sütunlardan (özelliklerden) oluşan iki boyutlu bir yapı olarak algılanan mantıksal bir yapı.

tuple
İlişkisel modelde, bir tablo satırı.

relational database management system (RDBMS)
Bir ilişkisel veritabanını yöneten bir programlar topluluğudur. RDBMS yazılımı, bir kullanıcının mantıksal isteklerini (sorguları) fiziksel olarak konumlandırıp, istenen veriyi almak için komutlara dönüştürür.

Not

İşletme ortamlarında DBMS kurulumlarının ezici bir çoğunluğu RDBMS ürünleridir. Ancak, pratikte çoğu veri profesyoneli, daha spesifik olan "RDBMS" terimi yerine genel olarak "DBMS" terimini kullanır.

RDBMS'nin tartışmasız en önemli avantajı, kullanıcıdan ilişkisel modelin karmaşıklıklarını gizleme yeteneğidir. RDBMS, tüm fiziksel detayları yönetirken, kullanıcı, verilerin saklandığı bir dizi tablo olarak ilişkisel veritabanını görür. Kullanıcı, veriyi sezgisel ve mantıklı bir şekilde manipüle edebilir ve sorgulayabilir.

Tablolar, ortak bir öznitelik (bir sütundaki bir değer) aracılığıyla birbirine bağlanır. Örneğin, Şekil 2.1'deki CUSTOMER tablosu, aynı zamanda AGENT tablosunda bulunan bir satış temsilcisinin numarasını içerebilir.

Figure 2.1 Linking Relational Tables

Table name: AGENT (first six attributes) **Database name: Ch02_InsureCo**

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_INSURE_TYPE	CUS_INSURE_AMT	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	T1	100.00	05-Apr-2022	502
10011	Dunne	Leona	K	713	894-1238	T1	250.00	16-Jun-2022	501
10012	Smith	Kathy	W	615	894-2285	S2	150.00	29-Jan-2023	502
10013	Olowski	Paul	F	615	894-2180	S1	300.00	14-Oct-2022	502
10014	Orlando	Myron		615	222-1672	T1	100.00	28-Dec-2023	501
10015	O'Brian	Amy	B	713	442-3381	T2	850.00	22-Sep-2022	503
10016	Brown	James	G	615	297-1228	S1	120.00	25-Mar-2023	502
10017	Williams	George		615	290-2556	S1	250.00	17-Jul-2022	503
10018	Farriss	Anne	G	713	382-7185	T2	100.00	03-Dec-2022	501
10019	Smith	Olette	K	615	297-3809	S2	500.00	14-Mar-2023	503

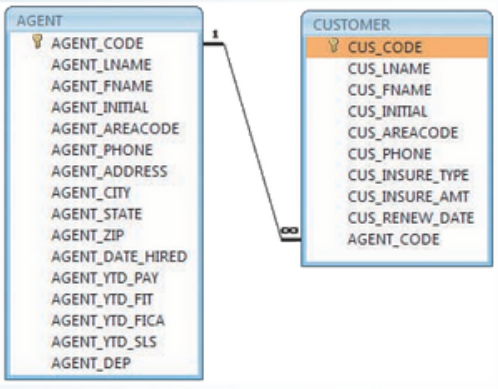
relational diagram
Bir ilişkisel veritabanının varlıklarını, bu varlıklar içindeki nitelikleri ve varlıklar arasındaki ilişkileri grafiksel olarak gösteren bir temsil.

Müşteri ve Satış Temsilcisi tabloları arasındaki ortak bağlantı, müşteri verilerinin bir tabloda ve satış temsilcisi verilerinin diğer tabloda saklanmasına rağmen bir müşteriyi satış temsilcisiyle eşleştirmenize olanak tanır. Örneğin, müşteri Dunne'ın temsilcisinin Alex Alby olduğunu kolayca belirleyebilirsiniz, çünkü müşteri Dunne için CUSTOMER tablosundaki AGENT_CODE 501'dir ve bu, Alex Alby'nin AGENT tablosundaki AGENT_CODE ile eşleşir. Tablolar birbirinden bağımsız olmasına rağmen, veriler arasında kolayca ilişki kurabilirsiniz. İlişkisel model, dosya sistemlerinde yaygın olarak bulunan çoğu fazlalığı ortadan kaldırmak için kontrol edilen minimum düzeyde bir fazlalık sağlar.

İlişki türü (1:1, 1:M veya M:N) genellikle bir ilişkisel şemada gösterilir, bunun bir örneği Şekil 2.2'de gösterilmiştir. İlişkisel diyagram, ilişkisel veritabanının varlıklarını, bu varlıklardaki nitelikleri ve bu varlıklar arasındaki ilişkileri temsil eder. Şekil 2.2'deki ilişkisel diyagram, bağlama alanlarını (bu örnekte AGENT_CODE) ve ilişki türünü (1:M) gösterir. Şekil 2.2'yi oluşturmak için kullanılan veritabanı yazılımı olan Microsoft Access, "çok" tarafı belirtmek için sonsuzluk simgesini (∞) kullanır. Bu örnekte, CUSTOMER "çok" tarafını temsil eder çünkü bir AGENT'in birden fazla CUSTOMER'ı olabilir. AGENT ise "1" tarafını temsil eder çünkü her CUSTOMER yalnızca bir AGENT'e sahiptir.

Online Content
Bu bölümdeki veritabanlarına www.cengage.com adresinden ulaşılabilir. Örneğin, Şekil 2.1'de gösterilen AGENT ve CUSTOMER tablolarının içerikleri, Ch02_InsureCo adlı veritabanında bulunmaktadır.

Figure 2.2 A Relational Diagram



Bir ilişkisel tablo, birbirine bağlı varlıkların bir koleksiyonunu saklar. Bu anlamda, ilişkisel veritabanı tablosu bir dosyaya benzer, ancak tablo ve dosya arasında önemli bir fark vardır: bir tablo, tamamen mantıksal bir yapı olduğu için tam veri ve yapı bağımsızlığı sağlar. Verilerin veritabanında nasıl fiziksel olarak saklandığı, kullanıcı veya tasarımcı için bir konu değildir; önemli olan algıdır. İlişkisel veritabanı modelinin bu özelliği, bir veri tabanı devrimini başlatan kaynağı oluşturmuştur ve bir sonraki bölümde bu konu derinlemesine ele alınacaktır.

İlişkisel veri modelinin hâkimiyetinin bir diğer nedeni ise güçlü ve esnek sorgu dilidir. Çoğu ilişkisel veritabanı yazılımı, kullanıcının ne yapılması gerektiğini belirttiği, ancak nasıl yapılması gerektiğini belirtmediği Structured Query Language (SQL) kullanır. RDBMS, SQL'i kullanarak kullanıcı sorgularını istenen veriyi almak için talimatlara dönüştürür. SQL, veriyi diğer veritabanı veya dosya ortamlarından çok daha az çaba ile çekmeyi mümkün kılar.

Bir son kullanıcı perspektifinden bakıldığında, SQL tabanlı bir ilişkisel veritabanı uygulaması üç bölümden oluşur: bir kullanıcı arayüzü, veritabanında saklanan bir dizi tablo ve SQL "motoru". Her biri şu şekilde açıklanabilir:

- Son kullanıcı arayüzü: Temelde, arayüz son kullanıcının verilerle etkileşime girmesine olanak tanır (otomatik olarak SQL kodu üreterek). Her arayüz, yazılım tedarikçisinin verilerle anlamlı bir etkileşim fikrinin ürünüdür. Ayrıca, veritabanı yazılımı alanında artık standart olan uygulama üreticileri yardımıyla kendi özelleştirilmiş arayüzünüzü de tasarlayabilirsiniz.
- Veritabanında saklanan tablo koleksiyonu: Bir ilişkisel veritabanında tüm verilerin tablolarda saklandığı kabul edilir. Tablolar, verileri son kullanıcıya anlaşılır bir şekilde "sunar". Her tablo bağımsızdır. Farklı tablolardaki satırlar, ortak niteliklerdeki ortak değerlerle ilişkilidir.
- SQL motoru: Çoğunlukla son kullanıcıdan gizli olan SQL motoru, tüm sorguları veya veri taleplerini çalıştırır. SQL motoru, DBMS yazılımının bir parçasıdır. Son kullanıcı, SQL kullanarak tablo yapılarını oluşturur ve veri erişimi ile tablo bakımını yapar. SQL motoru, tüm kullanıcı taleplerini genellikle perde arkasında ve son kullanıcının bilgisi olmadan işler. Bu nedenle, SQL'in ne yapılması gerektiğini söyleyen ancak nasıl yapılması gerektiğini belirtmeyen bir deklaratif dil olduğu söylenir. (SQL motoru hakkında daha fazla bilgiyi Bölüm 11, "Veritabanı Performansı Ayarlama ve Sorgu Optimizasyonu"nda öğreneceksiniz.)

RDBMS bazı görevleri perde arkasında gerçekleştirdiğinden, veritabanının fiziksel yönlerine odaklanmak gerekmez. Bunun yerine, sonraki bölümler ilişkisel veritabanının mantıksal kısmına ve tasarımına odaklanacaktır. Ayrıca, SQL detaylı bir şekilde Bölüm 7, "Structured Query Language (SQL)'e Giriş" ve Bölüm 8, "İleri Seviye SQL"de ele alınacaktır.

2-5c The Entity Relationship Model

İlişkisel veritabanı teknolojisinin kavramsal basitliği, RDBMS'lere olan talebi tetikledi. Buna karşılık, işlem ve bilgiye yönelik hızla artan gereksinimler, daha karmaşık veritabanı uygulama yapılarına olan ihtiyacı yarattı ve böylece daha etkili veritabanı tasarım araçlarına ihtiyaç duyuldu. (Örneğin, bir gökdelen inşa etmek, bir köpek kulübesi inşa etmekten daha detaylı tasarım faaliyetleri gerektirir.)

Karmaşık tasarım faaliyetleri, başarılı sonuçlar elde etmek için kavramsal basitlik gerektirir. İlişkisel model, hiyerarşik ve ağ modellerine göre büyük bir gelişme olmasına rağmen, yine de onu etkili bir veritabanı tasarım aracı yapacak özelliklerden yoksundu. Yapıları metinle tanımlamaktan ziyade grafiksel olarak incelemek daha kolay olduğundan, veritabanı tasarımcıları varlıkların ve ilişkilerinin resmedildiği bir grafik aracı kullanmayı tercih ederler. Böylece, varlık ilişki (ER) modeli (ERM), veri modelleme için yaygın olarak kabul görmüş bir standart haline gelmiştir.

Peter Chen, ER veri modelini ilk olarak 1976'da tanıttı; bir veritabanı yapısındaki varlıkların ve ilişkilerinin grafiksel gösterimi, ilişkisel veri modeli kavramlarını tamamladığı için hızla popüler hale geldi. İlişkisel veri modeli ve ERM, sıkı bir şekilde yapılandırılmış veritabanı tasarımının temelini oluşturmak için bir araya geldi.

entity relationship (ER) model (ERM) (Varlık-İlişki (ER) Modeli (ERM))
Varlık-ilişki modeli, varlıklar arasındaki ilişkileri (1:1, 1:M ve M:N) kavramsal düzeyde açıklayan bir veri modelidir. Bu model, ER diyagramları kullanarak varlıklar ve onların birbirleriyle olan ilişkilerini görsel olarak temsil eder.

entity relationship diagram (ERD)
Bir varlık ilişki modelinin varlıklarını, özniteliklerini ve ilişkilerini tasvir eden bir diyagram.

entity instance (entity occurrence)
İlişkisel bir tablodaki bir satır.

entity set
Benzer varlıkların bir koleksiyonu.

connectivity
Varlıklar arasındaki ilişki türü. Sınıflandırmalar arasında 1:1, 1:M ve M:N bulunur.

Chen notation
Bkz. varlık ilişki (ER) modeli.

Crow's Foot notation
A representation of the entity relationship diagram that uses a three-pronged symbol to represent the "many" sides of the relationship.

class diagram notation
Sınıf diyagramlarının oluşturulmasında kullanılan semboller kümesi.

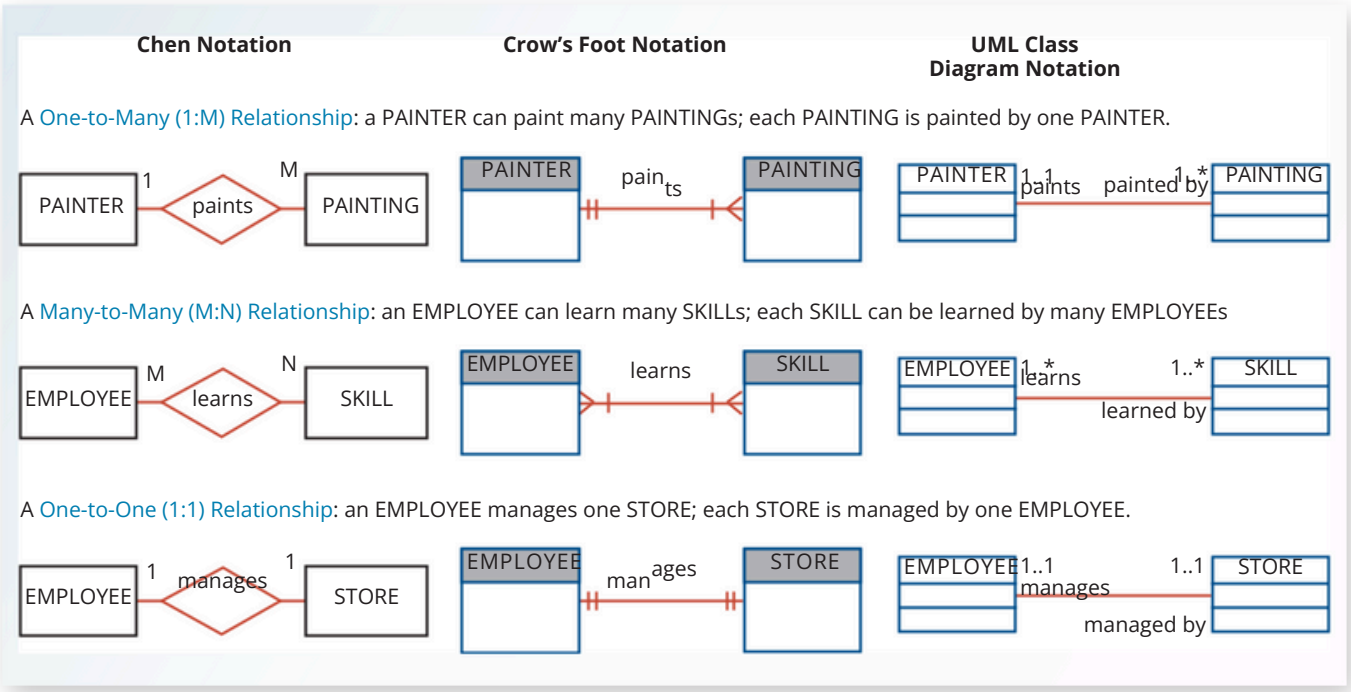
ER modelleri normalde bir varlık ilişki diyagramında (ERD) temsil edilir; bu diyagram, veritabanı bileşenlerini modellemek için grafiksel gösterimler kullanır. Bölüm 4, Varlık İlişki (ER) Modelleme bölümünde, veritabanlarını tasarlamak için ERD'leri nasıl kullanacağınızı öğreneceksiniz.

ER modeli aşağıdaki bileşenlere dayanmaktadır:
Varlık. Bu bölümün başlarında, bir varlık, hakkında veri toplanacak ve depolanacak herhangi bir şey olarak tanımlanmıştır. Bir varlık, ERD'de bir dikdörtgenle temsil edilir, buna varlık kutusu da denir. Varlığın adı, bir isim, dikdörtgenin ortasına yazılır. Varlık adı genellikle büyük harflerle ve tekil biçimde yazılır: PAINTERS yerine PAINTER ve EMPLOYEES yerine EMPLOYEE. Genellikle, ERD'yi ilişkisel modele uygularken, bir varlık bir ilişkisel tabloya eşlenir. İlişkisel tablodaki her satıra, ER modelinde varlık örneği veya varlık oluşumu denir. Benzer varlıkların bir koleksiyonuna varlık kümesi denir. Örneğin, Şekil 2.1'deki AGENT dosyasını AGENT varlık kümesindeki üç aracı (varlığı) içeren bir koleksiyon olarak düşünebilirsiniz. Teknik olarak konuşursak, ERD varlık kümelerini tasvir eder. Ne yazık ki, ERD tasarımcıları varlık kelimesini varlık kümesinin yerine kullanır ve bu kitap, herhangi bir ERD ve bileşenlerini tartışırken bu yerleşik uygulamaya uyacaktır.

Öznitelikler. Her varlık, varlığın belirli özelliklerini açıklayan bir öznitelikler kümesinden oluşur. Örneğin, EMPLOYEE varlığının Sosyal Güvenlik numarası, soyadı ve adı gibi öznitelikleri olacaktır. (Bölüm 4, özniteliklerin ERD'ye nasıl dahil edildiğini açıklar.)

İlişkiler. İlişkiler, veriler arasındaki ilişkileri açıklar. Çoğu ilişki, iki varlık arasındaki ilişkileri açıklar. Temel veri modeli bileşenleri tanıtıldığında, üç tür veri ilişkisi gösterilmiştir: bire çok (1:M), çoktan çoka (M:N) ve bire bir (1:1). ER modeli, ilişki türlerini etiketlemek için bağlantısallık terimini kullanır. İlişkinin adı genellikle aktif veya pasif bir fiildir. Örneğin, bir PAINTER birçok PAINTING boyar, bir EMPLOYEE birçok SKILL öğrenir ve bir EMPLOYEE bir STORE yönetir.
Şekil 2.3, üç ER notasyonunu kullanarak farklı ilişki türlerini göstermektedir: orijinal Chen notasyonu, Crow's Foot notasyonu ve Birleşik Modelleme Dili'nin (UML) bir parçası olan daha yeni sınıf diyagramı notasyonu.

Figure 2.3 The ER Model Notations



ER diyagramının sol tarafı, Peter Chen'in çığır açan makalesine dayanan Chen notasyonunu göstermektedir. Bu notasyonda, bağlantısallıklar her varlık kutusunun yanına yazılır. İlişkiler, bir ilişki çizgisi aracılığıyla ilgili varlıklara bağlı bir elmasla temsil edilir. İlişki adı elmasın içine yazılır.
Şekil 2.3'ün ortası, Crow's Foot notasyonunu göstermektedir. Crow's Foot adı, ilişkinin "çok" tarafını temsil etmek için kullanılan üç çatallı sembolden türetilmiştir. Şekil 2.3'teki temel Crow's Foot ERD'sini incelerken, bağlantısallıkların sembollerle temsil edildiğine dikkat edin. Örneğin, "1" kısa bir çizgi parçasıyla temsil edilir ve "M", üç çatallı "karga ayağı" ile temsil edilir. Bu örnekte, ilişki adı ilişki çizgisinin üzerine yazılır.
Şekil 2.3'ün sağ tarafı, UML notasyonunu (UML sınıf notasyonu olarak da bilinir) göstermektedir. Bağlantısallıkların semboller içeren çizgilerle (1..1, 1..*) temsil edildiğine dikkat edin. Ayrıca, UML notasyonu ilişkinin her iki tarafında da isimler kullanır. Örneğin, PAINTER ve PAINTING arasındaki ilişkiyi okumak için şunlara dikkat edin:
Bir PAINTER, 1..* sembolüyle belirtildiği gibi, birden çok PAINTING "boyar".
Bir PAINTING, 1..1 sembolüyle belirtildiği gibi, yalnızca bir PAINTER tarafından "boyanır".
Şekil 2.3'te varlıklar ve ilişkiler yatay bir biçimde gösterilir, ancak dikey olarak da yönlendirilebilirler. Varlığın konumu ve varlıkların sunulma sırası önemli değildir; sadece 1:M ilişkisini "1" tarafından "M" tarafına doğru okumayı unutmayın.
Bu kitapta Crow's Foot notasyonu tasarım standardı olarak kullanılmaktadır. Ancak, gerektiğinde bazı ER modelleme kavramlarını göstermek için Chen notasyonu kullanılmaktadır. Çoğu veri modelleme aracı, Crow's Foot veya UML sınıf diyagramı notasyonunu seçmenize olanak tanır. Sonraki bölümlerde göreceğiniz Crow's Foot tasarımlarını oluşturmak için Microsoft Visio Professional yazılımı kullanılmıştır.
ER modelinin olağanüstü görsel basitliği, onu baskın veritabanı modelleme ve tasarım aracı yapmaktadır. Bununla birlikte, veri ortamı gelişmeye devam ettikçe daha iyi veri modelleme araçları arayışı da devam etmektedir.

2-5d The Object-Oriented Model

Giderek karmaşıklaşan gerçek dünya sorunları, gerçek dünyayı daha yakından temsil eden bir veri modeline duyulan ihtiyacı göstermiştir. Nesne yönelimli veri modelinde (OODM), hem veriler hem de ilişkileri, nesne olarak bilinen tek bir yapıda bulunur. Buna karşılık, OODM, nesne yönelimli veritabanı yönetim sisteminin (OODBMS) temelini oluşturur.
Bir OODM, varlıkları tanımlamanın ve kullanmanın çok farklı bir yolunu yansıtır. İlişkisel modelin varlığı gibi, bir nesne de olgusal içeriğiyle tanımlanır. Ancak, bir varlıktan oldukça farklı olarak, bir nesne, nesne içindeki gerçekler arasındaki ilişkiler hakkında bilgi içerir ve ayrıca diğer nesnelerle olan ilişkileri hakkında da bilgi içerir. Bu nedenle, nesne içindeki gerçeklere daha fazla anlam verilir. OODM'nin semantik veri modeli olduğu söylenir, çünkü semantik anlamı ifade eder.
Sonraki OODM geliştirme, bir nesnenin ayrıca veri değerlerini değiştirme, belirli bir veri değerini bulma ve veri değerlerini yazdırma gibi üzerinde gerçekleştirilebilecek tüm işlemleri içermesine izin vermiştir. Nesneler veri, çeşitli ilişki türleri ve operasyonel prosedürler içerdiğinden, nesne kendi kendine yeterli hale gelir ve bu da onu -en azından potansiyel olarak- özerk yapılar için temel bir yapı taşı haline getirir.
OO veri modeli aşağıdaki bileşenlere dayanmaktadır:
Bir nesne, gerçek dünya varlığının bir soyutlamasıdır. Genel olarak, bir nesne, bir ER modelinin varlığına eşdeğer kabul edilebilir. Daha doğrusu, bir nesne bir varlığın yalnızca bir oluşumunu temsil eder. (Nesnenin semantik içeriği, bu listedeki birkaç madde aracılığıyla tanımlanır.)

Çevrimiçi İçerik
Bu bölüm yalnızca temel OO kavramlarını tanıtmaktadır. Nesne yönelimli kavramları ve prensipleri ayrıntılı olarak www.cengage.com adresindeki G Ekinden (Nesne Yönelimli Veritabanları) inceleyebilirsiniz.

object-oriented data model (OODM)
Temel modelleme yapısı bir nesne olan bir veri modeli.

objec t
Benzersiz bir kimliğe, gömülü özelliklere ve diğer nesnelerle ve kendiyle etkileşim kurma yeteneğine sahip gerçek dünya varlığının soyut bir temsili.

object-oriented database management system (OODBMS)
Verileri nesne yönelimli bir veritabanı modelinde yönetmek için kullanılan veri yönetimi yazılımı.

semantic data model
Verileri ve ilişkilerini nesne olarak bilinen tek bir yapıda modelleyen bir dizi veri modelinin ilki.

Öznitelikler bir nesnenin özelliklerini açıklar. Örneğin, bir KİŞİ nesnesi Ad, Sosyal Güvenlik Numarası ve Doğum Tarihi özniteliklerini içerir.

class
Paylaşılan yapıya (öznitelikler) ve davranışa (yöntemler) sahip benzer nesnelerin bir koleksiyonu. Bir sınıf, bir nesnenin veri temsilini ve bir yöntemin uygulamasını kapsar.

Benzer özelliklere sahip nesneler sınıflarda gruplandırılır. Sınıf, paylaşılan yapıya (öznitelikler) ve davranışa (yöntemler) sahip benzer nesnelerin bir koleksiyonudur. Genel anlamda, bir sınıf ER modelinin varlık kümesine benzer. Ancak, bir sınıf, yöntem olarak bilinen bir dizi yordam içermesi bakımından bir varlık kümesinden farklıdır. Bir sınıfın yöntemi, seçilen bir KİŞİ'nin adını bulma, bir KİŞİ'nin adını değiştirme veya bir KİŞİ'nin adresini yazdırma gibi gerçek dünya eylemini temsil eder. Başka bir deyişle, yöntemler geleneksel programlama dillerindeki yordamlara eşdeğerdir. OO terimleriyle, yöntemler bir nesnenin davranışını tanımlar.

method
Nesne yönelimli veri modelinde, bir eylemi gerçekleştirmek için adlandırılmış bir talimatlar kümesi. Yöntemler gerçek dünya eylemlerini temsil eder.

class hierarchy
Her üst sınıfın bir üst sınıf ve her alt sınıfın bir alt sınıf olduğu hiyerarşik bir ağaçtaki sınıfların organizasyonu. Ayrıca bkz. kalıtım.

Sınıflar bir sınıf hiyerarşisinde düzenlenir. Sınıf hiyerarşisi, her sınıfın yalnızca bir üst öğesi olduğu baş aşağı bir ağaca benzer. Örneğin, MÜŞTERİ sınıfı ve ÇALIŞAN sınıfı bir üst KİŞİ sınıfını paylaşır. (Bu açıdan hiyerarşik veri modeline olan benzerliğe dikkat edin.)

inheritance
Nesne yönelimli veri modelinde, bir nesnenin sınıf şemasındaki üzerindeki sınıfların veri yapısını ve yöntemlerini miras alma yeteneği

Kalıtım, sınıf hiyerarşisi içindeki bir nesnenin, üzerindeki sınıfların özniteliklerini ve yöntemlerini miras alma yeteneğidir. Örneğin, iki sınıf, MÜŞTERİ ve ÇALIŞAN, KİŞİ sınıfından alt sınıflar olarak oluşturulabilir. Bu durumda, MÜŞTERİ ve ÇALIŞAN, KİŞİ'den tüm öznitelikleri ve yöntemleri miras alacaktır.

Unified Modeling Language (UML)
Bir sistemi grafiksel olarak modellemek için şemalar ve semboller gibi araçlar sağlayan nesne yönelimli kavramlara dayalı bir dil.

Nesne yönelimli veri modelleri tipik olarak Birleşik Modelleme Dili (UML) sınıf diyagramları kullanılarak tasvir edilir. UML, bir sistemi grafiksel olarak modellemek için kullanabileceğiniz bir dizi diyagramı ve sembolü tanımlayan OO kavramlarına dayalı bir dildir. UML sınıf diyagramları, daha büyük UML nesne yönelimli sistem modelleme dilinde verileri ve ilişkilerini temsil etmek için kullanılır. UML'nin daha eksiksiz bir açıklaması için bkz. Ek H, Birleşik Modelleme Dili (UML).

class diagram
Verileri ve ilişkilerini UML nesne notasyonunda temsil etmek için kullanılan bir şema.

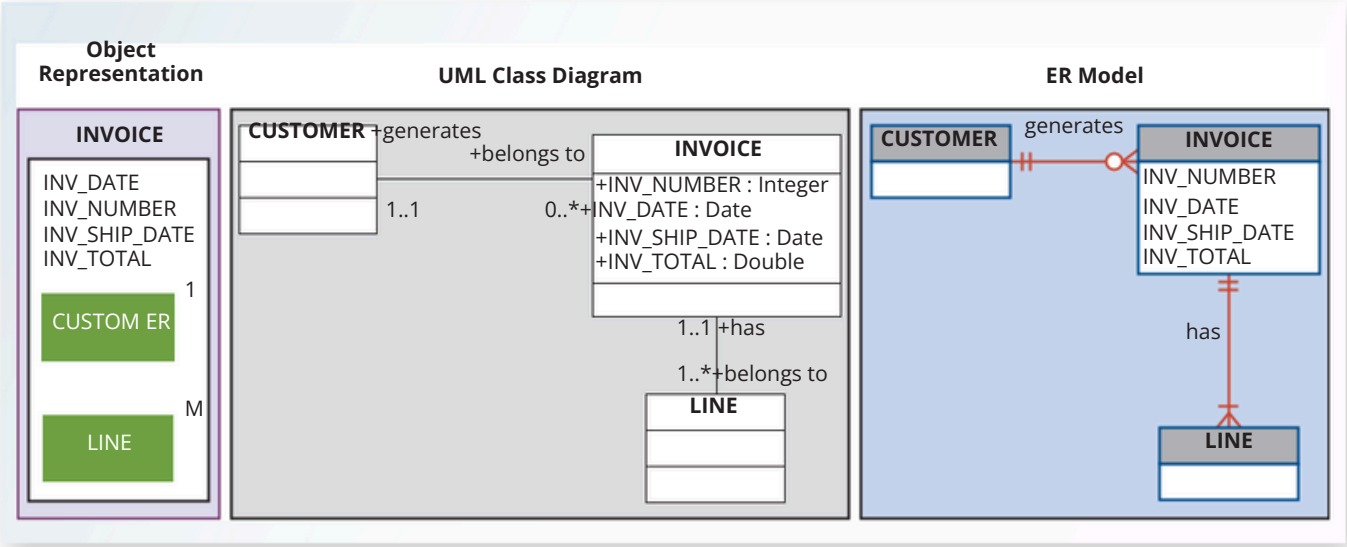
OODM'nin ana kavramlarını göstermek için basit bir faturalandırma problemini ele alalım. Bu durumda, faturalar müşteriler tarafından oluşturulur, her fatura bir veya daha fazla satıra atıfta bulunur ve her satır bir müşteri tarafından satın alınan bir öğeyi temsil eder. Şekil 2.4, bu basit faturalandırma problemi için nesne temsilini, eşdeğer UML sınıf diyagramını ve ER modelini göstermektedir. Nesne gösterimi, tek bir nesne oluşumunu görselleştirmenin basit bir yoludur. Şekil 2.4'ü incelerken, şunlara dikkat edin:

FATURA'nın nesne temsili, aynı nesne kutusu içindeki tüm ilgili nesneleri içerir. Bağlantısallıkların (1 ve M) ilgili nesnelerin FATURA ile ilişkisini gösterdiğine dikkat edin. Örneğin, MÜŞTERİ nesnesinin yanındaki "1", her FATURA'nın yalnızca bir MÜŞTERİ ile ilgili olduğunu gösterir. SATIR nesnesinin yanındaki "M", her FATURA'nın birçok SATIR içerdiğini gösterir.

UML sınıf diyagramı, bu basit faturalandırma problemini temsil etmek için üç ayrı nesne sınıfı (MÜŞTERİ, FATURA ve SATIR) ve iki ilişki kullanır. İlişki bağlantısallıklarının 1..1, 0..* ve 1..* sembolleriyle temsil edildiğine ve ilişkilerdeki nesnelerin oynadığı farklı "rolleri" temsil etmek için ilişkilerin her iki ucunda da adlandırıldığına dikkat edin.

ER modeli ayrıca bu basit fatura problemini temsil etmek için üç ayrı varlık ve iki ilişki kullanır. OODM ilerlemeleri, sistem modellemesinden programlamaya kadar birçok alanı etkiledi. (En çağdaş programlama dillerinin çoğu Java, Ruby, Perl, C# ve Python dahil olmak üzere OO kavramlarını benimsemiştir.) OODM'nin eklenen semantiği, karmaşık nesnelerin daha zengin bir şekilde temsil edilmesini sağladı. Bu da uygulamaların giderek daha karmaşık nesneleri yenilikçi yollarla desteklemesini sağladı. Bir sonraki bölümde göreceğiniz gibi, bu tür evrimsel ilerlemeler ilişkisel modeli de etkiledi.

Figure 2.4 A Comparison of the OO, UML, and ER Models



2-5e Object/Relational and XML

Daha karmaşık veri temsillerini destekleme talebiyle karşı karşıya kalan ilişkisel modelin ana satıcıları, modeli daha da geliştirdiler ve genişletilmiş ilişkisel veri modelini (ERDM) oluşturdular. ERDM, OO modelinin birçok özelliğini doğası gereği daha basit ilişkisel veritabanı yapısı içinde ekler. ERDM, nesneler (kapsüllenmiş veri ve yöntemler), sınıflara dayalı genişletilebilir veri türleri ve kalıtım gibi OO özelliklerini destekleyen yeni nesil ilişkisel veritabanlarına yol açtı. Bu nedenle, ERDM'ye dayalı bir VTYS genellikle nesne/ilişkisel veritabanı yönetim sistemi (O/R VTYS) olarak tanımlanır.

Bugün, çoğu ilişkisel veritabanı ürünü nesne/ilişkisel olarak sınıflandırılabilir ve OLTP ve OLAP veritabanı uygulamalarının baskın pazar payını temsil ederler. O/R VTYS'lerinin başarısı, modelin kavramsal basitliğine, veri bütünlüğüne, kullanımı kolay sorgu diline, yüksek işlem performansına, yüksek kullanılabilirliğe, güvenliğe, ölçeklenebilirliğe ve genişletilebilirliğe atfedilebilir. Buna karşılık, OO VTYS, bilgisayar destekli çizim/bilgisayar destekli üretim (CAD/CAM), coğrafi bilgi sistemleri (CBS), telekomünikasyon ve daha karmaşık nesneler için destek gerektiren multimedya gibi niş pazarlarda popülerdir.

Başlangıçtan itibaren, OO ve ilişkisel veri modelleri farklı sorunlara yanıt olarak geliştirildi. OO veri modeli, genel veri yönetimi görevlerinin geniş kapsamlı ihtiyaçları değil, çok özel mühendislik ihtiyaçlarını karşılamak için oluşturuldu. İlişkisel model, sağlam bir matematiksel temele dayalı olarak daha iyi veri yönetimine odaklanılarak oluşturuldu. Daha küçük bir problem alanına odaklanması göz önüne alındığında, OO pazarının ilişkisel veri modeli pazarı kadar hızlı büyümesi şaşırtıcı değildir.

Karmaşık nesnelerin kullanımı, İnternet devrimi ile bir destek aldı. Kuruluşlar iş modellerini İnternet ile entegre ettiklerinde, İnternetin kritik iş bilgilerine erişme, dağıtma ve paylaşma potansiyelini fark ettiler. Bu, İnternetin bir iş iletişim aracı olarak yaygın bir şekilde benimsenmesine neden oldu. Bu ortamda, Genişletilebilir İşaretleme Dili (XML), yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış verilerin verimli ve etkili bir şekilde paylaşılması için fiili standart olarak ortaya çıktı. XML verilerini kullanan kuruluşlar, kısa sürede kelime işlem belgeleri, web sayfaları, e-postalar ve diyagramlar gibi büyük miktarda yapılandırılmamış veriyi yönetmeleri gerektiğini fark ettiler. Bu ihtiyacı karşılamak için, yerel bir XML biçiminde yapılandırılmamış verileri yönetmek için XML veritabanları ortaya çıktı (XML hakkında daha fazla bilgi için bkz. Bölüm 15, Veritabanı Bağlantısı ve Web Teknolojileri). Aynı zamanda, O/R VTYS'leri, ilişkisel veri yapıları içinde XML tabanlı belgeler için destek ekledi. Geniş çapta uygulanabilir ilkelere sağlam bir şekilde dayanması nedeniyle, ilişkisel model, nesneler ve XML gibi yeni yetenek sınıflarını içerecek şekilde kolayca genişletilebilir.

extended relational data model (ERDM)
Nesne yönelimli modelin en iyi özelliklerini doğası gereği daha basit bir ilişkisel veritabanı yapısal ortamında içeren bir model. Bkz. genişletilmiş varlık ilişki modeli (EERM).

object/relational database management system (O/R DBMS)
Genişletilmiş ilişkisel modele (ERDM) dayalı bir VTYS. Birçok ilişkisel veritabanı araştırmacısı tarafından savunulan ERDM, ilişkisel modelin OODM'ye yanıtını oluşturur. Bu model, nesne yönelimli modelin birçok en iyi özelliğini doğası gereği daha basit bir ilişkisel veritabanı yapısı içinde içerir.

Extensible Markup Language (XML)
Veri öğelerini temsil etmek ve işlemek için kullanılan bir üst dil. Diğer işaretleme dillerinden farklı olarak, XML bir belgenin veri öğelerinin manipülasyonuna izin verir. XML, İnternet üzerinden siparişler ve faturalar gibi yapılandırılmış belgelerin değişimini kolaylaştırır.

İlişkisel ve nesne/ilişkisel veritabanları mevcut veri işleme ihtiyaçlarının çoğunu karşılasa da, bazı İnternet çağı kuruluşlarında bulunan bazı çok özel zorlukları ele almak için yeni nesil veritabanları ortaya çıktı.

2-5f Emerging Data Models: Big Data and NoSQL

Kuruluşların yıllar içinde biriktirdiği web verisi dağlarından kullanılabilir iş bilgileri elde etmek zorunlu bir ihtiyaç haline gelmiştir. Göz atma kalıpları, satın alma geçmişleri, müşteri tercihleri, davranış kalıpları ve Facebook, Twitter ve LinkedIn gibi kaynaklardan gelen sosyal medya verileri şeklindeki web verileri, kuruluşları yapılandırılmış ve yapılandırılmamış veri kombinasyonlarıyla boğmuştur. Ek olarak, akıllı telefonlar ve tabletler gibi mobil teknolojiler, ayrıca her türden sensör -GPS, RFID sistemleri, hava durumu sensörleri, biyomedikal cihazlar, uzay araştırmaları sondaları, araba ve havacılık kara kutuları- yanı sıra diğer İnternet ve hücresel bağlantılı cihazlar, çoklu biçimlerde (metin, resim, ses, video vb.) büyük miktarda veriyi otomatik olarak toplamanın yeni yollarını yaratmıştır. Veri alışverişi yapan ve toplayan İnternet bağlantılı cihazların bu ağı, Nesnelerin İnterneti (IoT) olarak bilinir. Toplanan veri miktarı her gün katlanarak artmaktadır. IoT, veri büyüme hızını hızlandırmıştır, öyle ki şu anda günlük olarak yaklaşık 2,5 kentilyon bayt veri oluşturulmaktadır. Veri büyümesinin hızlı temposu, sistem performansı ve ölçeklenebilirlik sonraki en büyük zorluklar olmak üzere, kuruluşlar için büyük bir zorluk olabilir. Günümüzün bilgi teknolojisi (BT) yöneticileri, bu hızla büyüyen verileri yönetme ihtiyacını daralan bütçelerle sürekli olarak dengelemektedir. Tüm bu birleşen eğilimleri (hızlı veri büyümesi, performans, ölçeklenebilirlik ve daha düşük maliyetler) yönetme ve bunlardan yararlanma ihtiyacı, "Büyük Veri" adı verilen bir olguyu tetiklemiştir. Büyük Veri, büyük miktarda web ve sensör tarafından oluşturulan veriyi yönetmenin ve ondan iş bilgisi elde etmenin yeni ve daha iyi yollarını bulma hareketini ifade ederken, aynı zamanda makul bir maliyetle yüksek performans ve ölçeklenebilirlik sağlar.

Internet of Things (IoT)
Sürekli olarak veri alışverişi yapan ve İnternet üzerinden veri toplayan, İnternet'e bağlı cihazlardan oluşan bir web. IoT cihazları uzaktan yönetilebilir ve veri toplamak ve İnternet'teki diğer cihazlarla etkileşim kurmak için yapılandırılabilir.

Big Data
Büyük miktarda web kaynaklı veriyi yönetmenin ve ondan iş bilgisi elde etmenin yeni ve daha iyi yollarını bulma hareketi, aynı zamanda makul bir maliyetle yüksek performans ve ölçeklenebilirlik sağlama.

3 Vs
Büyük Veri veritabanlarının üç temel özelliği: hacim, hız ve çeşitlilik.

Büyük Veri terimi, hukuktan istatistiğe, ekonomiye ve bilişime kadar birçok çerçevede kullanılmıştır. Terimin ilk olarak 1990'larda bir Silicon Graphics bilim adamı olan John Mashey tarafından bir bilişim çerçevesinde kullanıldığı görülmektedir.1 Ancak, Gartner Group'tan bir veri analisti olan Douglas Laney, Büyük Veri veritabanlarının temel özelliklerini ilk tanımlayan kişi gibi görünmektedir: hacim, hız ve çeşitlilik veya 3 V'ler.
Hacim, depolanan veri miktarlarını ifade eder. İnternet ve sosyal medyanın benimsenmesi ve büyümesiyle birlikte, şirketler müşterilere ulaşma yollarını çoğaltmıştır. Yıllar içinde ve teknolojik gelişmelerin faydasıyla, milyonlarca e-işlem için veriler günlük olarak şirket veritabanlarında depolanıyordu. Ayrıca, kuruluşlar son kullanıcılarla etkileşim kurmak için birden çok teknoloji kullanıyor ve bu teknolojiler dağlarca veri üretiyor. Sürekli büyüyen bu veri hacmi hızla petabayt boyutuna ulaştı ve hala büyüyor.

Hız, yalnızca verilerin büyüme hızı değil, aynı zamanda bilgi ve içgörü oluşturmak için bu verileri hızlı bir şekilde işleme ihtiyacını da ifade eder. İnternet ve sosyal medyanın ortaya çıkmasıyla birlikte, iş yanıt süreleri önemli ölçüde azalmıştır. Kuruluşların yalnızca hızla biriken büyük hacimli verileri depolaması değil, aynı zamanda bu tür verileri hızlı bir şekilde işlemesi de gerekir. Veri büyüme hızı, verilerin kuruluşa aktarıldığı farklı veri akışlarının sayısındaki artıştan da kaynaklanmaktadır (web, e-ticaret, tweetler, Facebook gönderileri, e-postalar, sensörler, GPS vb. yoluyla).

[1]Steve Lohr, "The origins of 'Big Data': An etymological detective story," *New York Times*, February 1, 2013.
[2]Douglas Laney, "3D data management controlling data volume, velocity and variety," META Group, February 6, 2011.

Çeşitlilik, toplanan verilerin birden çok veri biçiminde gelmesi gerçeğini ifade eder. Bu verilerin büyük bir kısmı, ilişkisel modele dayalı tipik operasyonel veritabanları tarafından işlenmeye uygun olmayan biçimlerde gelir.
3 V's çerçevesi, şirketlerin artık ne bildiğini, veritabanlarında toplanan veri miktarının boyut ve karmaşıklık açısından katlanarak arttığını göstermektedir. Geleneksel ilişkisel veritabanları, yapılandırılmış verileri yönetmede iyidir, ancak günümüzün iş ortamında toplanan veri miktarlarını ve türlerini yönetmek ve işlemek için uygun değildir.
Sorun şu ki, ilişkisel yaklaşım her zaman Büyük Veri zorlukları olan kuruluşların ihtiyaçlarıyla eşleşmiyor.
Yapılandırılmamış, sosyal medya ve sensör tarafından oluşturulan verileri satır ve sütunların geleneksel ilişkisel yapısına uydurmak her zaman mümkün değildir.
Milyonlarca satır çok biçimli (yapılandırılmış ve yapılandırılmamış) veriyi günlük olarak eklemek, kaçınılmaz olarak ilişkisel ortamda mevcut olmayabilecek daha fazla depolama, işlem gücü ve gelişmiş veri analiz araçlarına duyulmasına yol açacaktır. Büyük Veri sorunu için RDBMS ortamında gerekli olan yüksek hacimli uygulamalar türü, donanımı, depolamayı ve yazılım lisanslarını genişletmek için yüksek bir fiyat etiketiyle birlikte gelir.
OLAP araçlarına dayalı veri analizi, yüksek yapılandırılmış verilere sahip ilişkisel ortamlarda çok başarılı olduğunu kanıtlamıştır. Bununla birlikte, web kaynaklarından toplanan büyük miktarda yapılandırılmamış veride kullanılabilir veri için madencilik yapmak farklı bir yaklaşım gerektirir.
Veri yönetimi ihtiyaçlarına "tek beden herkese uyar" bir çözüm yoktur (ancak birçok köklü veritabanı satıcısı muhtemelen size bu fikri satmaya çalışacaktır). Bazı kuruluşlar için, Büyük Veri analizi için yüksek düzeyde ölçeklenebilir, hataya dayanıklı bir altyapı oluşturmak, işin hayatta kalması meselesi olabilir. İş dünyasında, teknoloji ile rekabet avantajı elde eden şirketlerin ve bunu kaçıranların birçok örneği vardır. Sadece kendinize şu soruları sorun:

- *Blackberry, gelişen Apple akıllı telefon teknolojisine hızla yanıt vermiş olsaydı, iş ortamı nasıl farklı olurdu?*
- *MySpace, Facebook'un meydan okumasına zamanında yanıt vermiş olsaydı.*
- *Blockbuster, Netflix iş modeline daha erken tepki göstermiş olsaydı.*
- *Barnes & Noble, Amazon'dan önce uygulanabilir bir İnternet stratejisi geliştirmiş olsaydı.*

Yayın televizyon ağları, Hulu, AppleTV ve Roku gibi akış hizmetlerine uyum sağlamada başarılı olacak mı? Sektör, değişen teknolojik olanaklara yanıt verirken ortaklıklar ve birleşmeler şüphesiz ev eğlencesi manzarasını değiştirecektir. Geleneksel haber kuruluşları, Y kuşağının değişen haber tüketim alışkanlıklarına uyum sağlayabilecek mi?
Büyük Veri analitiği, her tür şirket tarafından yeni tür hizmetler oluşturmak için kullanılıyor. Örneğin, Amazon başlangıçta düşük maliyetli bir sağlayıcı olarak "büyük kutu" mağazalarıyla rekabet ediyordu. Amazon sonunda depolama ve işleme teknolojilerinden yararlanarak film ve müzik hizmetini akışta sunma konusunda rekabet etmeye başladı ve daha yakın zamanda, tahmini nakliye gibi yenilikçi hizmetler oluşturmak için Büyük Veriden yararlandı. Tahmini nakliye, bir müşterinin bir ürünün ne zaman gerekli olacağını tahmin etmek için satın alma kalıplarını kullanır ve müşteri ihtiyaç duyduğunu fark etmeden önce müşteriye gönderir! Amazon ayrıca, doğal dil işlemeyi gerçekleştirmek için Alexa hizmetini kullanan Amazon Echo gibi ürünlerin satışında da başarılı olmuştur. Bu "sürekli dinleyen" cihazlar, Amazon'a mevcut hizmetleri iyileştirmek ve gelecekteki hizmetlerde yeniliği desteklemek için analiz edebileceği benzeri görülmemiş düzeylerde ve türlerde veri sağlayan dünyanın dört bir yanındaki evlere yerleştirilmiştir.
Şirketler, daha önce kullanılmayan Büyük Veri depolarından değer yaratmak için yeni Büyük Veri teknolojileri kullanıyor. Bu gelişmekte olan teknolojiler, kuruluşların işleme koymasına izin veriyor

büyük veri depolarını uygun maliyetli yollarla birden çok biçimde saklar. En sık kullanılan Büyük Veri teknolojilerinden bazıları Hadoop ve NoSQL veritabanlarıdır.

Hadoop, Java tabanlı, açık kaynaklı, yüksek hızlı, hataya dayanıklı dağıtılmış depolama ve hesaplama çerçevesidir. Hadoop, verileri depolamak ve işlemek için binlerce bilgisayar düğümünden oluşan kümeler oluşturmak için düşük maliyetli donanım kullanır. Hadoop, Google'ın dağıtılmış dosya sistemleri ve paralel işleme üzerine yaptığı çalışmalardan kaynaklanmıştır ve şu anda Apache Yazılım Vakfı tarafından desteklenmektedir.3 Hadoop'un çeşitli modülleri vardır, ancak iki ana bileşen Hadoop Dağıtılmış Dosya Sistemi (HDFS) ve MapReduce'dur.

Hadoop
Java tabanlı, açık kaynaklı, yüksek hızlı, hataya dayanıklı dağıtılmış depolama ve hesaplama çerçevesi. Hadoop, verileri depolamak ve işlemek için binlerce bilgisayar düğümünden oluşan kümeler oluşturmak için düşük maliyetli donanım kullanır

Hadoop Distributed File System (HDFS)
Yüksek hızlarda büyük miktarda veriyi yönetmek için tasarlanmış, yüksek düzeyde dağıtılmış, hataya dayanıklı bir dosya depolama sistemi.

Hadoop Dağıtılmış Dosya Sistemi (HDFS), büyük miktarda veriyi yüksek hızlarda yönetmek için tasarlanmış, yüksek düzeyde dağıtılmış, hataya dayanıklı bir dosya depolama sistemidir. Yüksek verim elde etmek için HDFS, bir kez yaz, çok oku modelini kullanır. Bu, veri yazıldıktan sonra değiştirilemeyeceği anlamına gelir. HDFS üç tür düğüm kullanır: dosya sistemi hakkında tüm meta verileri depolayan bir ad düğümü, sabit boyutlu veri bloklarını depolayan (diğer veri düğümlerine çoğaltılabilir) bir veri düğümü ve kullanıcı uygulaması ile HDFS arasında arayüz görevi gören bir istemci düğümü.

name node
Hadoop Dağıtılmış Dosya Sistemi'nde (HDFS) kullanılan üç düğüm türünden biri. Ad düğümü, dosya sistemiyle ilgili tüm meta verileri depolar. Ayrıca bkz. istemci düğümü ve veri düğümü.

MapReduce, hızlı veri analitiği hizmetleri sağlayan açık kaynaklı bir uygulama programlama arayüzüdür (API). MapReduce, verilerin işlenmesini paralel olarak binlerce düğüm arasında dağıtır. MapReduce, yapılandırılmış ve yapılandırılmamış verilerle çalışır. MapReduce çerçevesi iki ana işlev sağlar: Harita ve Azaltma. Genel olarak, Harita işlevi bir iş alır ve onu daha küçük iş birimlerine böler ve Azaltma işlevi düğümlerden oluşturulan tüm çıktı sonuçlarını toplar ve bunları tek bir sonuç kümesine entegre eder. MapReduce'un kendisi bugün oldukça sınırlı olarak görülse de, Büyük Verilerin nasıl işlendiğine dair paradigmayı tanımladı.

NoSQL, yapılandırılmış ve yapılandırılmamış verileri verimli bir şekilde depolayan büyük ölçekli dağıtılmış bir veritabanı sistemidir. NoSQL veritabanları, Bölüm 14, Büyük Veri ve NoSQL'de daha ayrıntılı olarak tartışılmaktadır.

data node
Hadoop Dağıtılmış Dosya Sistemi'nde (HDFS) kullanılan üç düğüm türünden biri. Veri düğümü, sabit boyutlu veri bloklarını (diğer veri düğümlerine çoğaltılabilir) depolar. Ayrıca bkz. istemci düğümü ve ad düğümü.

Hadoop teknolojileri, verilerin (yapılandırılmış veya yapılandırılmamış) dağıtıldığı, çoğaltıldığı ve düşük maliyetli ticari donanım ağı kullanılarak paralel olarak işlendiği Büyük Veri analitiği için bir çerçeve sağlar. Hadoop, verileri depolamanın ve yönetmenin yeni yollarını tanıttı. Karıştırılmayın: Hadoop ve NoSQL veritabanları genellikle birlikte tartışılır çünkü her ikisi de Büyük Veri sorunlarını ele almada bileşenlerdir. Ancak, Hadoop ne bir veritabanı ne de bir veri modelidir. Dağıtılmış bir dosya depolama ve işleme modelidir. Hadoop VTYS'si yoktur. NoSQL veritabanları veritabanlarıdır ve NoSQL modeli, verilerin ilişkisel olmayan bir şekilde depolanmasına ve işlenmesine farklı bir yaklaşımı temsil eder. NoSQL veritabanları, yapılandırılmamış verilerin işlenmesi için dağıtılmış, hataya dayanıklı veritabanları sağlar.

client node
Hadoop Dağıtılmış Dosya Sistemi'nde (HDFS) kullanılan üç düğüm türünden biri. İstemci düğümü, kullanıcı uygulaması ile HDFS arasında arayüz görevi görür. Ayrıca bkz. ad düğümü ve veri düğümü.

Büyük Veri analitiğinden elde edilen büyük kazanım potansiyeli ile, bazı kuruluşların web verisi dağlarında gizli olan bilgi zenginliğini madencilik yapmak ve rekabet avantajı elde etmek için NoSQL veritabanları gibi yeni ortaya çıkan Büyük Veri teknolojilerine yönelmesi şaşırtıcı değildir.

MapReduce
Hızlı veri analitiği hizmetleri sağlayan açık kaynaklı bir uygulama programlama arayüzü (API); kuruluşların büyük veri depolarını işlemesini sağlayan ana Büyük Veri teknolojilerinden biri.

Not

Bu, ilişkisel veritabanlarının Büyük Veri zorlukları olan kuruluşlarda bir yeri olmadığı anlamına mı geliyor? Hayır, ilişkisel veritabanları çoğu günlük işlem ve yapılandırılmış veri analitiği ihtiyaçlarını desteklemek için tercih edilen ve baskın veritabanları olmaya devam ediyor. Her VTYS teknolojisinin uygulama alanları vardır ve en iyi yaklaşım iş için en iyi aracı kullanmaktır. Perspektifte, nesne/ilişkisel veritabanları operasyonel pazar ihtiyaçlarının yüzde 98'ine hizmet ediyor. Büyük Veri ihtiyaçları için Hadoop ve NoSQL veritabanları seçenekler arasındadır. Bölüm 14, Büyük Veri ve NoSQL, bu seçenekleri daha ayrıntılı olarak tartışmaktadır.

NoSQL
Geleneksel ilişkisel veritabanı modeline dayanmayan yeni nesil bir veritabanı yönetim sistemi.

[3]Daha fazla bilgi: hadoop.apache.org.

NoSQL Databases

NoSQL Veritabanları

Amazon'da bir ürün aradığınızda, Facebook'ta arkadaşlarınıza mesaj gönderdiğinizde, YouTube'da bir video izlediğinizde veya Google Haritalar'da yol tarifi aradığınızda, bir NoSQL veritabanı kullanıyorsunuz. Herhangi bir yeni teknolojide olduğu gibi, NoSQL terimi birçok farklı teknoloji türüne gevşek bir şekilde uygulanabilir. Ancak, bu bölüm NoSQL'i Büyük Veri çağının özel zorluklarını ele alan ve aşağıdaki genel özelliklere sahip yeni nesil veritabanlarına atıfta bulunmak için kullanmaktadır:

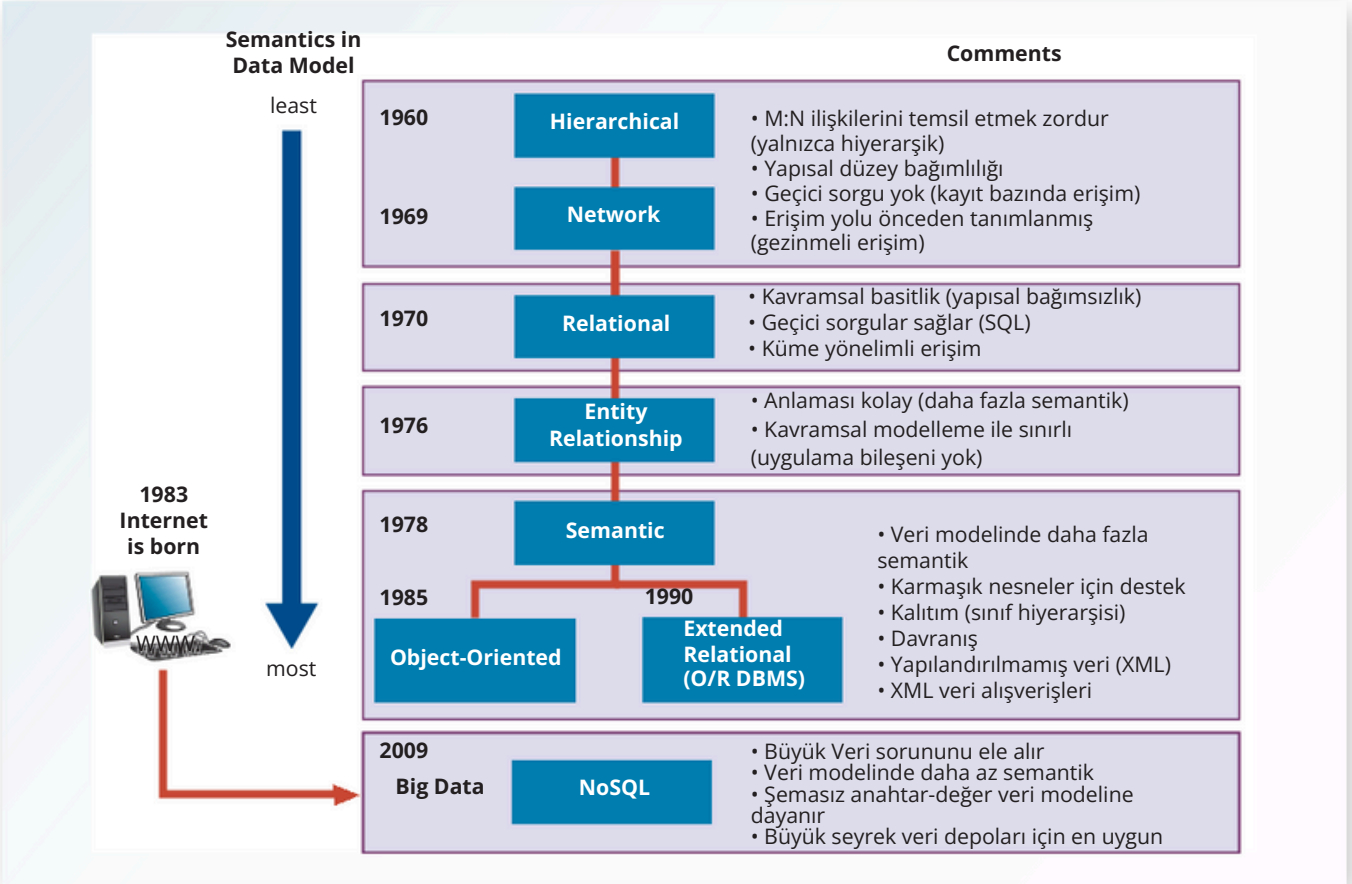
- İlişkisel modele ve SQL'e dayanmazlar; dolayısıyla NoSQL adı.
- Yüksek düzeyde dağıtılmış veritabanı mimarilerini desteklerler.
- Yüksek ölçeklenebilirlik, yüksek kullanılabilirlik ve hata toleransı sağlarlar.
- Çok büyük miktarda seyrek veriyi desteklerler (çok sayıda özniteliğe sahip veriler, ancak gerçek veri örneklerinin sayısı düşüktür).
- İşlem tutarlılığından ziyade performansa yöneliktirler.

Veri depolama ve manipülasyonuna çok kapsamlı ve uyumlu bir yaklaşım sağlayan ilişkisel modelin aksine, NoSQL modeli veri depolama ve manipülasyonuna yönelik çeşitli yaklaşımlar için geniş bir şemsiyedir. Bu yaklaşımların en yaygın olanları, Bölüm 14'te ayrıntılı olarak tartışıldığı gibi, anahtar-değer depoları, belge veritabanları, sütunlu veritabanları ve grafik veritabanlarıdır.

2-5g Data Models: Özet

DMBS'lerin evrimi her zaman giderek karmaşıklaşan gerçek dünya verilerini modellemenin ve yönetmenin yeni yollarını arayışıyla yönlendirilmiştir. En yaygın olarak tanınan veri modellerinin bir özeti Şekil 2.5'te gösterilmektedir.

Figure 2.5 The Evolution of Data Models



Bir veri modeli, veritabanının semantik bütünlüğünden ödün vermeden bir dereceye kadar kavramsal basitlik göstermelidir. Gerçek dünyadan daha zor kavramsallaştırılan bir veri modeline sahip olmak mantıklı değildir. Aynı zamanda, model açıklık ve alaka göstermelidir; yani, veri modeli belirsiz olmamalı ve problem alanına uygulanabilir olmalıdır. Bir veri modeli, gerçek dünyayı olabildiğince yakından temsil etmelidir. Bu hedef, modelin veri temsiline daha fazla semantik eklenerek daha kolay gerçekleştirilir. (Semantik, dinamik veri davranışıyla ilgilenirken, veri temsili gerçek dünya senaryosunun statik yönünü oluşturur.) Başka bir deyişle, model doğru ve eksiksiz olmalıdır—gerekli tüm veriler dahil edilmeli ve uygun şekilde açıklanmalıdır.

Gerçek dünya dönüşümlerinin (davranış) temsili, veri modelinin amaçlanan kullanımının gerektirdiği tutarlılık ve bütünlük özelliklerine uygun olmalıdır.

Her yeni veri modeli, önceki modellerin eksikliklerini giderir. Ağ modeli, hiyerarşik modelin yerini aldı çünkü ilki karmaşık (çoktan çoka) ilişkileri temsil etmeyi çok daha kolay hale getirdi. Buna karşılık, ilişkisel model, daha basit veri temsili, üstün veri bağımsızlığı ve kullanımı kolay sorgu dili aracılığıyla hiyerarşik ve ağ modellerine göre çeşitli avantajlar sunar; bu özellikler onu iş uygulamaları için tercih edilen veri modeli haline getirmiştir. OO veri modeli, zengin bir semantik çerçeve içinde karmaşık veri desteği getirmiştir. ERDM, ilişkisel modele birçok OO özelliği eklemiş ve iş ortamında güçlü pazar payını korumasına izin vermiştir. Son yıllarda, Büyük Veri olgusu, geleneksel veri yönetiminden bir kopuşu temsil eden verileri modellemenin, depolamanın ve yönetmenin alternatif yollarının geliştirilmesini teşvik etmiştir.

Tüm veri modellerinin eşit yaratılmadığına dikkat edin; bazı veri modelleri bazı görevler için diğerlerinden daha uygundur. Örneğin, kavramsal modeller üst düzey veri modelleme için daha uygunken, uygulama modelleri uygulama amaçları için depolanmış verileri yönetmek için daha uygundur. ER modeli kavramsal bir model örneğiyken, hiyerarşik ve ağ modelleri uygulama modelleri örnekleridir. Aynı zamanda, ilişkisel model ve OODM gibi bazı modeller hem kavramsal hem de uygulama modelleri olarak kullanılabilir. Tablo 2.2, çeşitli veritabanı modellerinin avantajlarını ve dezavantajlarını özetlemektedir.

Not

Tüm veritabanları, veritabanı içinde ortak bir veri havuzunun kullanılmasını varsayar. Bu nedenle, tüm veritabanı modelleri veri paylaşımını teşvik eder ve böylece bilgi adaları potansiyel sorununu azaltır.

Şimdiye kadar, daha belirgin veri modellerinin temel yapılarıyla tanıştınız. Her model, gerçek dünya veri ortamının anlamını yakalamak için bu tür yapıları kullanır. Tablo 2.3, çeşitli veri modelleri tarafından kullanılan temel terminolojiyi göstermektedir.