2-5b The Relational Model (İlişkisel Model)

İlişkisel model, 1970 yılında IBM'den E. F. Codd tarafından, dönüm noktası niteliğindeki "Büyük Paylaşılan Veri Bankaları için İlişkisel Veri Modeli" (Communications of the ACM, Haziran 1970, s. 377–387) adlı makalesinde tanıtıldı. İlişkisel model, hem kullanıcılar hem de tasarımcılar için büyük bir atılımı temsil ediyordu. Bir benzetme yapmak gerekirse, ilişkisel model, kendisinden önce gelen "manuel vites" veritabanlarının yerine "otomatik vites" bir veritabanı sundu. Kavramsal sadeliği, gerçek bir veritabanı devriminin zeminini hazırladı.

ilişkisel model
IBM'den E. F. Codd tarafından 1970 yılında geliştirilen ilişkisel model, matematiksel kümeler teorisine dayanır ve veriyi bağımsız ilişkiler (tablolar) olarak temsil eder. Her ilişki (tablo), kavramsal olarak kesişen satırlar ve sütunlardan oluşan iki boyutlu bir yapı olarak temsil edilir. İlişkiler, ortak varlık özelliklerinin (sütunlardaki değerler) paylaşılması yoluyla birbirine bağlanır.

Not

Bu bölümde sunulan ilişkisel veritabanı modeli, bir giriş ve genel bakıştır. Daha ayrıntılı tartışma, 3. Bölüm: İlişkisel Veritabanı Modeli'nde yer almaktadır. Aslında, ilişkisel model o kadar önemlidir ki, kalan bölümlerin çoğundaki tartışmaların temelini oluşturacaktır.

İlişkisel modelin temeli, ilişki olarak bilinen matematiksel bir kavrama dayanır. Soyut matematiksel teorinin karmaşıklığından kaçınmak için, bir ilişkiyi (bazen tablo olarak adlandırılır) kesişen satırlar ve sütunlardan oluşan iki boyutlu bir yapı olarak düşünebilirsiniz. Bir ilişkideki her satıra kayıt (tuple) denir. Her sütun bir özniteliği temsil eder. İlişkisel model ayrıca, ileri düzey matematiksel kavramlara dayanan veri manipülasyonu yapıları için belirli bir seti tanımlar.

tablo (ilişki)
İlişkisel modelde bir varlık kümesini temsil eden, kesişen satırlardan (varlıklar) ve sütunlardan (öznitelikler) oluşan iki boyutlu bir yapı olarak algılanan mantıksal bir yapı.

Kayıt (Tuple) İlişkisel modelde, bir tablo satırı.

1970 yılında, Codd'un çalışması dahice olarak kabul edilse de pratik olmayan bir yaklaşım olarak görülüyordu. İlişkisel modelin kavramsal sadeliği, bilgisayar yükü pahasına elde edilmişti; o dönemdeki bilgisayarlar, ilişkisel modeli uygulama gücünden yoksundu. Neyse ki, bilgisayar gücü ve işletim sistemi verimliliği üssel olarak arttı. Dahası, bilgisayarların gücü arttıkça maliyetleri hızla azaldı. Bugün, ana bilgisayar atalarına kıyasla çok daha düşük maliyetli olan PC'ler, Oracle, DB2, Microsoft SQL Server, MySQL ve diğer ana bilgisayar ilişkisel yazılımlarını çalıştırabiliyor.

İlişkisel veri modeli, çok sofistike bir ilişkisel veritabanı yönetim sistemi (RDBMS) aracılığıyla uygulanır. RDBMS, hiyerarşik ve ağ tabanlı DBMS sistemlerinin sağladığı temel işlevlerin yanı sıra, ilişkisel veri modelini anlamayı ve uygulamayı kolaylaştıran birçok başka işlevi de yerine getirir (Bölüm 1'de, DBMS İşlevleri kısmında belirtildiği gibi).

İlişkisel Veritabanı Yönetim Sistemi (RDBMS)
İlişkisel bir veritabanını yöneten bir programlar kümesi. RDBMS yazılımı, bir kullanıcının mantıksal isteklerini (sorguları) fiziksel olarak verileri bulup almak için gereken komutlara dönüştürür.

Not

İşletme ortamlarında yapılan DBMS kurulumlarının büyük çoğunluğu RDBMS ürünleridir. Ancak, pratikte çoğu veri profesyoneli, daha spesifik olan "RDBMS" terimi yerine genel "DBMS" terimini kullanmaktadır. Bu, her seferinde köpeğimin bir pudel olduğunu belirtmeden "köpeğim" demeye benzer.

RDBMS'nin belki de en önemli avantajı, ilişkisel modelin karmaşıklıklarını kullanıcılardan gizleyebilmesidir. RDBMS, tüm fiziksel ayrıntıları yönetirken, kullanıcı veritabanını verilerin depolandığı bir tablo koleksiyonu olarak görür. Kullanıcı, veriyi sezgisel ve mantıklı bir şekilde manipüle edebilir ve sorgulayabilir.

Tablolar, ortak bir özniteliğin (bir sütundaki değer) paylaşılması yoluyla birbirine bağlanır. Örneğin, Şekil 2.1'deki CUSTOMER (MÜŞTERİ) tablosu, AGENT tablosunda da bulunan bir satış temsilcisi numarasını içerebilir.

Şekil 2.1 İlişkisel Tabloları Bağlama

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

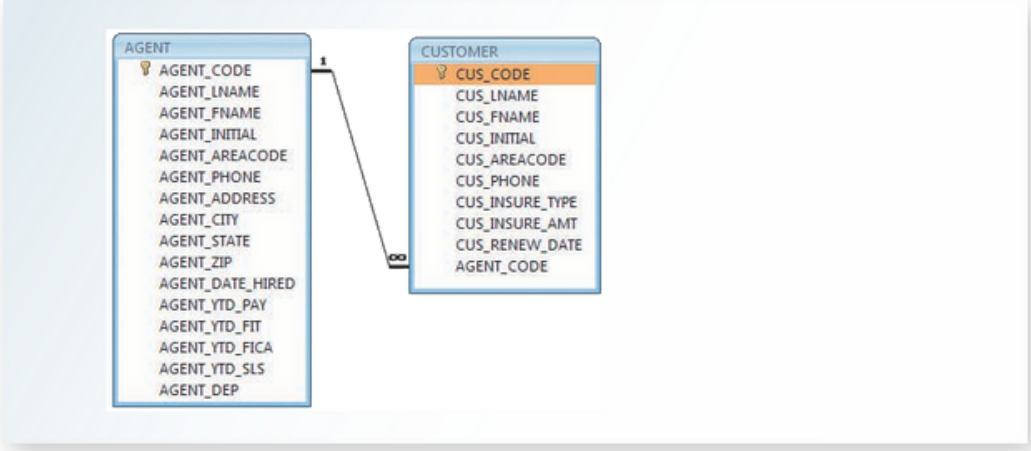
İlişkisel Diyagram
Bir ilişkisel veritabanının varlıklarını, bu varlıklardaki öznitelikleri ve varlıklar arasındaki ilişkileri grafiksel olarak temsil eden bir diyagram.

Çevrimiçi İçerik Bu bölümdeki veritabanlarına www.cengage.com adresinden erişilebilir. Örneğin, Şekil 2.1'de gösterilen AGENT ve CUSTOMER tablolarının içeriği, Ch02_InsureCo adlı veritabanında bulunmaktadır.

Müşteri (CUSTOMER) ve Ajan (AGENT) tabloları arasındaki ortak bağlantı, müşteri verilerinin bir tabloda ve satış temsilcisi verilerinin başka bir tabloda saklanmasına rağmen, bir müşteriyi satış temsilcisiyle eşleştirmenizi sağlar. Örneğin, müşteri Dunne'in temsilcisinin Alex Alby olduğunu kolayca belirleyebilirsiniz, çünkü müşteri Dunne için MÜŞTERİ tablosunun AGENT_CODE 501'dir ve bu, Alex Alby'nin AGENT tablosundaki AGENT_CODE ile eşleşir. Tablolar birbirinden bağımsız olsa da, tablolardaki verileri kolayca ilişkilendirebilirsiniz. İlişkisel model, dosya sistemlerinde yaygın olarak bulunan çoğu redundansı ortadan kaldırmak için minimum düzeyde kontrol edilen bir fazla veri (redundancy) sağlar.

İlişki türü (1:1, 1:M veya M:N), genellikle bir ilişkisel şemada gösterilir; bunun bir örneği Şekil 2.2'de gösterilmektedir. İlişkisel diyagram, ilişkisel veritabanının varlıklarını, bu varlıklardaki öznitelikleri ve varlıklar arasındaki ilişkileri temsil eder. Şekil 2.2'de, ilişkisel diyagram, bağlantı alanlarını (bu durumda AGENT KODU) ve ilişki türünü (1:M) göstermektedir. Şekil 2.2'yi oluşturmak için kullanılan veritabanı yazılımı Microsoft Access, "çok" tarafını belirtmek için sonsuzlu_k simgesini (∞) kullanır. Bu örnekte, CUSTOMER, bir AGENT ın birçok MÜŞTERİSİ' si olabileceği için "çok" tarafını temsil eder. AGENT ise her MÜŞTERİ' nin yalnızca bir AGENT' i olduğu için "1" tarafını temsil eder.

Şekil 2.2 İlişkisel Diyagram



Bir ilişkisel tablo, ilişkili varlıkların bir koleksiyonunu depolar. Bu açıdan, ilişkisel veritabanı tablosu bir dosyaya benzer, ancak tablo ile 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 depolandığı, kullanıcı ya da tasarımcı için önemli değildir; algı önemli olan şeydir. İlişkisel veri modelinin bu özelliği, bir sonraki bölümde derinlemesine incelenecek olup, gerçek bir veritabanı devrimine kaynaklık etmiştir. İlişkisel veri modelinin egemenliğe yükselmesinin bir diğer nedeni ise güçlü ve esnek sorgulama dilidir.

Çoğu ilişkisel veritabanı yazılımı, kullanıcının ne yapılması gerektiğini belirtirken nasıl yapılacağını belirtmesine gerek kalmadan belirlemesini sağlayan Yapılandırılmış Sorgulama Dili (SQL) kullanır. RDBMS, kullanıcı sorgularını, istenilen veriyi alacak şekilde talimatlara dönüştürmek için SQL kullanır.

SQL, veriyi, diğer veritabanı ya da dosya ortamlarından çok daha az çaba ile almayı mümkün kılar. Bir son kullanıcı perspektifinden bakıldığında, herhangi bir SQL tabanlı ilişkisel veritabanı uygulaması üç bileşenden oluşur: bir kullanıcı arayüzü, veritabanında saklanan bir dizi tablo ve SQL "motoru". Bu bileşenlerin her biri şu şekilde açıklanır:

- Son kullanıcı arayüzü temelde arayüz, son kullanıcının verilerle etkileşim kurmasını sağlar (SQL kodunu otomatik olarak oluşturarak). Her arayüz, yazılım satıcısının verilerle anlamlı bir etkileşim konusundaki vizyonunun bir ürünüdür. Ayrıca, günümüzde veritabanı yazılımlarında standart hale gelen uygulama üreticileri sayesinde kendi özelleştirilmiş arayüzünüzü tasarlayabilirsiniz.
- Veritabanında Saklanan Tabloların Koleksiyonu

  Bir ilişkisel veritabanın da, tüm veriler tablolar içinde saklanıyor olarak algılanır. Tablolar, verileri son kullanıcıya anlaşılması kolay bir şekilde “sunmak” için kullanılır. Her tablo bağımsızdır. Farklı tablolardaki satırlar, ortak özniteliklerde bulunan ortak değerler aracılığıyla birbiriyle ilişkilidir.

- SQL Motoru

  Büyük ölçüde son kullanıcıdan gizli olan SQL motoru, tüm sorguları veya veri taleplerini yürütür. SQL motorunun, DBMS yazılımının bir parçası olduğunu unutmayın. Son kullanıcı, tablo yapıları oluşturmak, veri erişimi sağlamak ve tablo bakımı yapmak için SQL kullanır. SQL motoru, kullanıcı isteklerini perde arkasında işler ve çoğu zaman son kullanıcı bunun farkında bile olmaz. Bu nedenle SQL, neyin yapılması gerektiğini belirten ancak nasıl yapılacağını belirtmeyen bildirimsel (deklaratif) bir dil olarak kabul edilir. (SQL motoru hakkında daha fazla bilgiyi Bölüm 11: Veritabanı Performans Ayarlama ve Sorgu Optimizasyonu başlığında öğreneceksiniz.)

RDBMS bazı görevleri perde arkasında gerçekleştirdiği için, veritabanının fiziksel yönlerine odaklanmak gerekli değildir. Bunun yerine, sonraki bölümler ilişkisel veritabanının mantıksal kısmına ve tasarımına odaklanmaktadır. Ayrıca, SQL konusu ayrıntılı olarak 7. Bölüm, "Yapılandırılmış Sorgu Diline (SQL) Giriş", ve 8. Bölüm, "İleri Düzey SQL" içinde ele alınmaktadır.

2-5c Varlık-İlişki Modeli (Entity Relationship Model)

İlişkisel veritabanı teknolojisinin kavramsal sadeliği, RDBMS'lere olan talebi tetikledi. Buna karşılık, hızla artan işlem ve bilgi gereksinimleri, daha karmaşık veritabanı uygulama yapılarının gerekliliğini ortaya çıkardı ve bu da daha etkili veritabanı tasarım araçlarına olan ihtiyacı doğurdu. (Örneğin, bir gökdelen inşa etmek, bir köpek kulübesi yapmaktan çok daha ayrıntılı tasarım faaliyetleri gerektirir.)

Karmaşık tasarım faaliyetlerinin başarılı sonuçlar verebilmesi için kavramsal sadelik gereklidir. İlişkisel model, hiyerarşik ve ağ modellerine kıyasla büyük bir gelişme olsa da, etkili bir veritabanı tasarım aracı olmasını sağlayacak bazı özelliklerden yoksundu. Yapıları metinle açıklamaktansa görsel olarak incelemek daha kolay olduğundan, veritabanı tasarımcıları varlıkları ve ilişkilerini görselleştiren bir grafik aracı kullanmayı tercih eder. Bu nedenle, Varlık-İlişki (ER) modeli (ERM), veri modelleme için yaygın olarak kabul gören bir standart haline gelmiştir.

Varlık-İlişki (ER) Modeli (ERM)
Varlıklar arasındaki ilişkileri (1:1, 1:M ve M:N) kavramsal düzeyde ER diyagramlarının yardımıyla tanımlayan bir veri modeli.

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

Varlık-İlişki Diyagramı (ERD)
Bir varlık-ilişki modelinin varlıklarını, özniteliklerini ve ilişkilerini gösteren bir diyagramdır.

Varlık Örneği (Varlık Görünümü)
İlişkisel bir tablodaki bir satır.

Varlık Kümesi
Benzer varlıkların bir koleksiyonu

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

Chen Gösterimi
Varlık-İlişki (ER) modeline bakınız.

Crow's Foot Gösterimi
ilişkisel varlık diyagramını (ERD) temsil eden ve ilişkinin "çok" (many) tarafını göstermek için üç uçlu bir sembol kullanan bir gösterim türüdür

Sınıf Diyagramı Gösterimi
Sınıf diyagramlarının oluşturulmasında kullanılan semboller kümesi.

ER modelleri genellikle, veritabanı bileşenlerini modellemek için grafiksel gösterimler kullanan varlık-ilişki diyagramı (ERD) ile temsil edilir. Veritabanlarını tasarlamak için ERD'leri nasıl kullanacağınızı 4. Bölüm, Varlık-İlişki (ER) Modelleme bölümünde öğreneceksiniz.

ER modelinin temel bileşenleri şunlardır:

• Varlık (Entity)

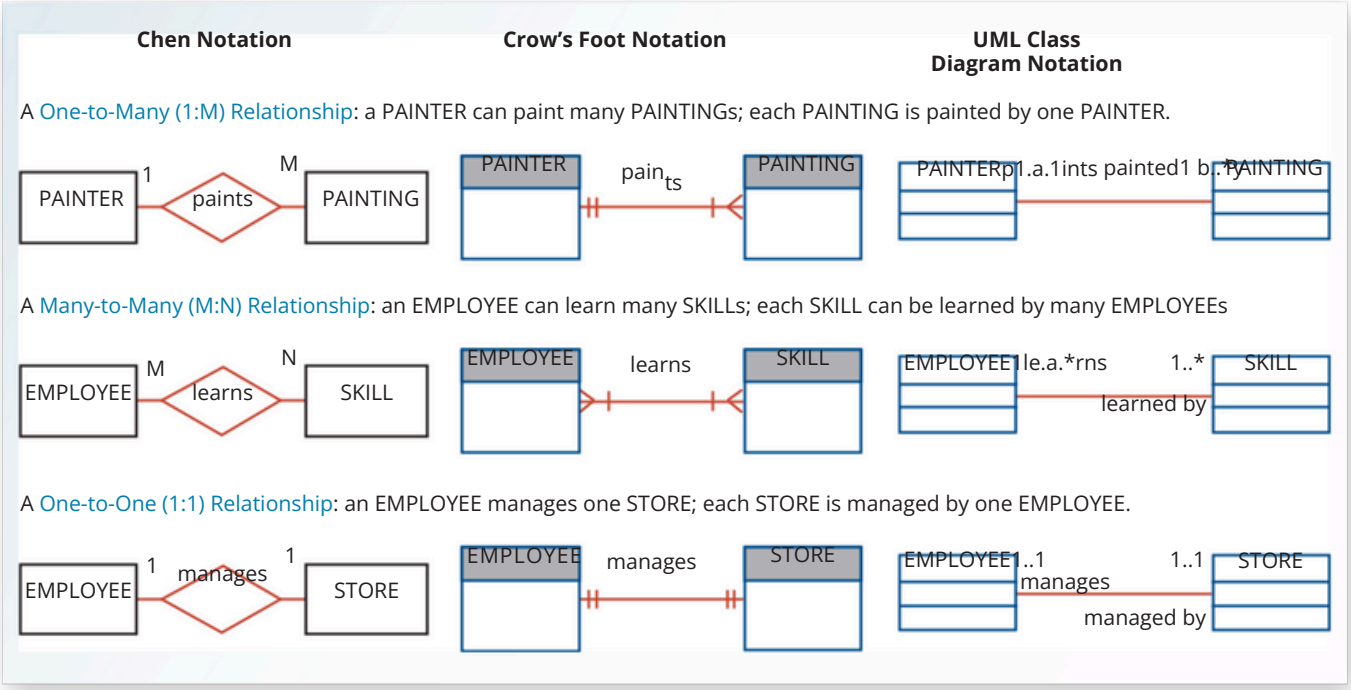
Bu bölümün başlarında, bir varlık, veri toplanıp saklanacak herhangi bir şey olarak tanımlanmıştı. Bir varlık, ERD'de bir dikdörtgen ile temsil edilir ve buna varlık kutusu da denir. Varlığın adı, dikdörtgenin ortasına yazılır. Varlık adı genellikle büyük harflerle ve tekil formda yazılır: PAINTER yerine PAINTERS, EMPLOYEE yerine EMPLOYEES. Genellikle, ERD'yi ilişkisel modele uygularken bir varlık, bir ilişkisel tabloya eşlenir. İlişkisel tablodaki her satır, ER modelinde bir varlık örneği veya varlık görünümü olarak bilinir. Benzer varlıklardan oluşan bir koleksiyon, bir varlık kümesi olarak adlandırılır. Örneğin, Şekil 2.1'deki AGENT dosyasını, AGENT varlık kümesindeki üç ajanı (varlıkları) içeren bir koleksiyon olarak düşünebilirsiniz. Teknik olarak, ERD varlık kümelerini gösterir. Ancak, ERD tasarımcıları, varlık kelimesini varlık kümesi yerine kullanılır ve bu kitap, herhangi bir ERD ve bileşenleri tartışırken bu yerleşik uygulamaya uyacaktır.

•Öznitelikler (Attributes). Her varlık, varlığın belirli özelliklerini tanımlayan bir öznitelikten oluşur Örneğin,
EMPLOYEE varlığı, bir Sosyal Güvenlik numarası, soyadı ve adı gibi özniteliklere sahip olacaktır. (Bölüm 4, özniteliklerin ERD'ye nasıl dahil edileceğini açıklar.)

•İlişkiler, veriler arasındaki bağlantıları tanımlar. Çoğu ilişki, iki varlık arasındaki bağlantıları açıklar. Temel veri model bileşenleri tanıtıldığında, üç tür veri ilişkisi gösterilmiştir: birden çoğa (1:M), çoktan çoğa (M:N) ve birden bireye (1:1). ER modeli, ilişki türlerini etiketlemek için bağlantısallık terimini kullanır. ilişkinin adı genellikle bir etkin veya pasif fiilidir. Örneğin, bir PAINTER birden fazla PAINTING yapar, bir EMPLOYEE birçok SKILL öğrenir ve bir EMPLOYEE bir mağazayı yönetir.

Şekil 2.3, üç ER notasyonu kullanarak farklı ilişki türlerini gösterir: orijinal Chen notasyonu, Crow's Foot notasyonu ve daha yeni sınıf diyagramı notasyonu, ki bu da Unified Modeling Language (UML)'nin bir parçasıdır.

Şekil 2.3 ER Model Notasyonları



ER diyagramının sol tarafı, Peter Chen'in önemli makalesine dayanan Chen notasyonunu gösterir. Bu notasyonda, bağlantılılıklar her varlık kutusunun yanına yazılır. İlişkiler, ilgili varlıklara ilişki çizgileriyle bağlanan bir elmas şekliyle temsil edilir. İlişkinin adı, elmasın içine yazılır.

Şekil 2.3'ün ortasında Crow's Foot notasyonu gösterilmektedir. "Crow's Foot" ismi, ilişkinin "çok" tarafını temsil etmek için kullanılan üç uçlu sembolden türetilmiştir. Şekil 2.3'teki temel Crow's Foot ERD'sini incelediğinizde, bağlantılılıkların sembollerle temsil edildiğini göreceksiniz. Örneğin, "1" kısa bir çizgi parçası ile, "M" ise üç uçlu "crow's foot" sembolü ile temsil edilir. Bu örnekte, ilişki adı ilişki çizgisinin üst kısmına yazılır.

Şekil 2.3'ün sağ tarafında UML notasyonu (aynı zamanda UML sınıf notasyonu olarak da bilinir) gösterilmektedir. Bu notasyonda, bağlantılılıklar sembollerle (1..1, 1..*) temsil edilen çizgilerle gösterilir. Ayrıca, UML notasyonu ilişkilerin her iki tarafında da isimler kullanır. Örneğin, PAINTER ve PAINTING arasındaki ilişkiyi okumak için şu şekilde değerlendirilir:

- Bir PAINTER "boyar" birden fazla PAINTINGi, bu ilişkiyi gösteren 1..* sembolüyle belirtildiği gibi
- Bir PAINTING "tarafından boyanmıştır" yalnızca bir PAINTER tarafından, bu ilişkiyi gösteren 1..1 sembolüyle belirtilmiştir.

Şekil 2.3'te varlıklar ve ilişkiler yatay bir formatta gösterilmiştir, ancak bunlar dikey olarak da düzenlenebilir. Varlıkların konumu ve sırasının önemi yoktur; yalnızca 1:M ilişkisinin "1" tarafından "M" tarafına doğru okunması gerektiğini unutmayın.

Bu kitapta Crow's Foot notasyonu, tasarım standardı olarak kullanılmaktadır. Ancak, gerekli olduğunda bazı ER modelleme kavramlarını açıklamak için Chen notasyonu da kullanılmaktadır. Çoğu veri modelleme aracı, Crow's Foot veya UML sınıf diyagramı notasyonlarından birini seçmenize olanak tanır. Microsoft Visio Professional yazılımı, sonraki bölümlerde göreceğiniz Crow's Foot tasarımlarını oluşturmak için kullanılmıştır.

ER modelinin olağanüstü görsel basitliği, onu baskın bir veri tabanı modelleme ve tasarım aracı yapmaktadır. Bununla birlikte, veri ortamı gelişmeye devam ettikçe daha iyi veri modelleme araçları arayışı da devam etmektedir.

Online concept
Bu bölüm yalnızca temel OOP (nesne yönelimli programlama) kavramlarını tanıtmaktadır. nesne yönelimli veritabanları ve ilkeleri hakkında ayrıntılı bilgiye ek G, Nesne Yönelimli Veritabanları bölümünden, www.cengage.com adresinden ulaşabilirsiniz.

2-5d Nesne Yönelimli Model

Giderek daha karmaşık hale gelen gerçek dünya problemleri, verilerin gerçek dünyayı daha yakın bir şekilde temsil edebileceği bir veri modeline olan ihtiyacı gösterdi. Nesne Yönelimli Veri Modeli (OODM) içinde hem veri hem de ilişkileri, bir nesne olarak bilinen tek bir yapıda yer alır. Bu arada, OODM, Nesne Yönelimli Veri Tabanı Yönetim Sistemi (OODBMS) için temel teşkil eder.

Nesne Yönelimli Veri Modeli (OODM)
Temel modelleme yapısı bir nesne olan bir veri modelidir.

Bir OODM, varlıkları tanımlama ve kullanma konusunda oldukça farklı bir yaklaşımı yansıtır. İlişkisel modelin varlığı gibi, bir nesne de gerçek içerikleriyle tanımlanır. Ancak, bir varlıktan çok farklı olarak, bir nesne, nesne içindeki gerçekler arasındaki ilişkilerle ilgili bilgilerin yanı sıra diğer nesnelerle olan ilişkileriyle ilgili bilgileri de içerir. Bu nedenle, nesne içindeki gerçekler daha büyük bir anlam taşır. OODM, anlamı ifade eden "semantik" terimi nedeniyle, semantik veri modeli olarak adlandırılır.

Nesne
Kendine özgü bir kimliği, gömülü özellikleri ve diğer nesnelerle ve kendisiyle etkileşimde bulunabilme yeteneği olan, gerçek dünya varlıklarının soyut bir temsilidir.

Sonraki OODM geliştirmeleri, bir nesnenin aynı zamanda içindeki verilerle ilgili tüm işlemleri de içermesini sağlamıştır. Bir nesnenin üzerinde gerçekleştirilebilecek işlemler, veri değerlerini değiştirme, belirli bir veri değerini bulma ve veri değerlerini yazdırma gibi işlemleri içerebilir. Nesneler, verileri, çeşitli ilişki türlerini ve operasyonel prosedürleri içerdiğinden, nesne kendi başına bir yapı haline gelir ve bu da onu—en azından potansiyel olarak—bağımsız yapılar için temel bir yapı taşı yapar.

Nesne Yönelimli Veri Tabanı Yönetim Sistem (OODBMS)
Nesne yönelimli bir veri tabanı modelinde verileri yönetmek için kullanılan veri yönetim yazılımıdır.

OO veri modeli aşağıdaki bileşenlere dayanır:

* Bir nesne, gerçek dünya varlıklarının bir soyutlamasıdır. Genel terimlerle, bir nesne, bir ER modelinin varlığına denk sayılabilir. Daha hassas bir şekilde, bir nesne yalnızca bir varlık örneğini temsil eder. (Nesnenin anlamlı içeriği, bu listedeki birkaç öğe aracılığıyla tanımlanır.)

Semantik Veri Modeli
Veriyi ve veriler arasındaki ilişkileri, bir nesne olarak bilinen tek bir yapıda modelleyen bir dizi veri modelinin ilki.

sınıf
Paylaşılan yapı (özellikler) ve davranış (metodlar) ile benzer nesnelerin bir koleksiyonu. Bir sınıf, bir nesnenin veri temsilini ve bir metodun uygulamasını kapsüller.

metod
Nesne yönelimli veri modelinde, bir eylemi gerçekleştirmek için adlandırılmış bir talimat seti. Metodlar, gerçek dünyadaki eylemleri temsil eder.

sınıf hiyerarşisi
Sınıfların, her ana sınıfın bir üst sınıf ve her çocuk sınıfın bir alt sınıf olduğu hiyerarşik bir ağaç şeklinde düzenlenmesi. Ayrıca, miras (inheritance) ile de bağlantılıdır.

miras
Nesne yönelimli veri modelinde, bir nesnenin, üstündeki sınıfların veri yapısını ve metodlarını miras alabilme yeteneği.

Bir sistemin grafiksel olarak modellenmesi için diyagramlar ve semboller gibi araçlar sağlayan, nesne yönelimli kavramlara dayalı bir dil.

Sınıf Diyagramı
Veri ve bunların ilişkilerini UML nesne notasyonunda temsil etmek için kullanılan bir diyagram.

• Öznitelikler, bir nesnenin özelliklerini tanımlar. Örneğin, bir PERSON nesnesi, Ad, Sosyal Güvenlik Numarası ve Doğum Tarihi gibi özniteliklere sahip olabilir.

•Benzer özelliklere sahip nesneler, sınıflarda gruplanır. Bir sınıf, benzer nesnelerin paylaşılan yapısı (öznitelikler) ve davranışları (metodlar) ile bir koleksiyonudur. Genel anlamda, bir sınıf, ER modelindeki varlık kümesine benzer. Ancak bir sınıf, bir varlık kümesinden farklıdır çünkü bir dizi işlem (metod) içerir. Bir sınıfın metodu, seçilen bir PERSON'un adını bulmak, bir PERSON'un adını değiştirmek ya da bir PERSON'un adresini yazdırmak gibi gerçek dünya eylemlerini temsil eder. Diğer bir deyişle, metodlar geleneksel programlama dillerindeki prosedürlerin karşılığıdır. OO (Nesne Yönelimli) terimleriyle, metodlar bir nesnenin davranışını tanımlar.

•Sınıflar, bir sınıf hiyerarşisinde düzenlenir. Sınıf hiyerarşisi, her sınıfın yalnızca bir üst sınıfı (ebeveyni) olduğu ters bir ağaç şeklinde benzer. Örneğin, CUSTOMER sınıfı ve EMPLOYEE sınıfı, bir ebeveyn PERSON sınıfını paylaşır. (Bu, hiyerarşik veri modeline benzerliğine dikkat edin.)

•Miras alma, sınıf hiyerarşisindeki bir nesnenin, üstündeki sınıflardan öznitelikleri ve metodları miras alma yeteneğidir. Örneğin, CUSTOMER ve EMPLOYEE adlı iki sınıf, PERSON sınıfından alt sınıflar olarak oluşturulabilir. Bu durumda, CUSTOMER ve EMPLOYEE, PERSON sınıfından tüm öznitelikleri ve metodları miras alacaktır.

•Nesne yönelimli veri modelleri, genellikle Birleşik Modelleme Dili (UML) sınıf diyagramları kullanılarak gösterilir. UML, nesne yönelimli (OO) kavramlar üzerine kurulmuş bir dil olup, bir sistemi grafiksel olarak modellemek için kullanabileceğiniz bir dizi diyagram ve sembol tanımlar. UML sınıf diyagramları, veriyi ve verinin ilişkilerini daha büyük UML nesne yönelimli sistem modelleme dilinde temsil etmek için kullanılır. UML hakkında daha ayrıntılı bir açıklama için, Ek H, Birleşik Modelleme Dili (UML)'ye bakın.

OODM'nin ana kavramlarını göstermek için, basit bir faturalama problemine bakalım. Bu durumda, faturalar müşteriler tarafından oluşturulur, her fatura bir veya daha fazla satırı referans alır ve her satır bir müşterinin satın aldığı bir öğeyi temsil eder. Şekil 2.4, bu basit faturalama problemi için nesne temsilini, karşılık gelen UML sınıf diyagramını ve ER modelini gösterir. Nesne temsili, tek bir nesne örneğini görselleştirmenin basit bir yoludur

Şekil 2.4'ü incelerken, aşağıdakilere dikkat edin:

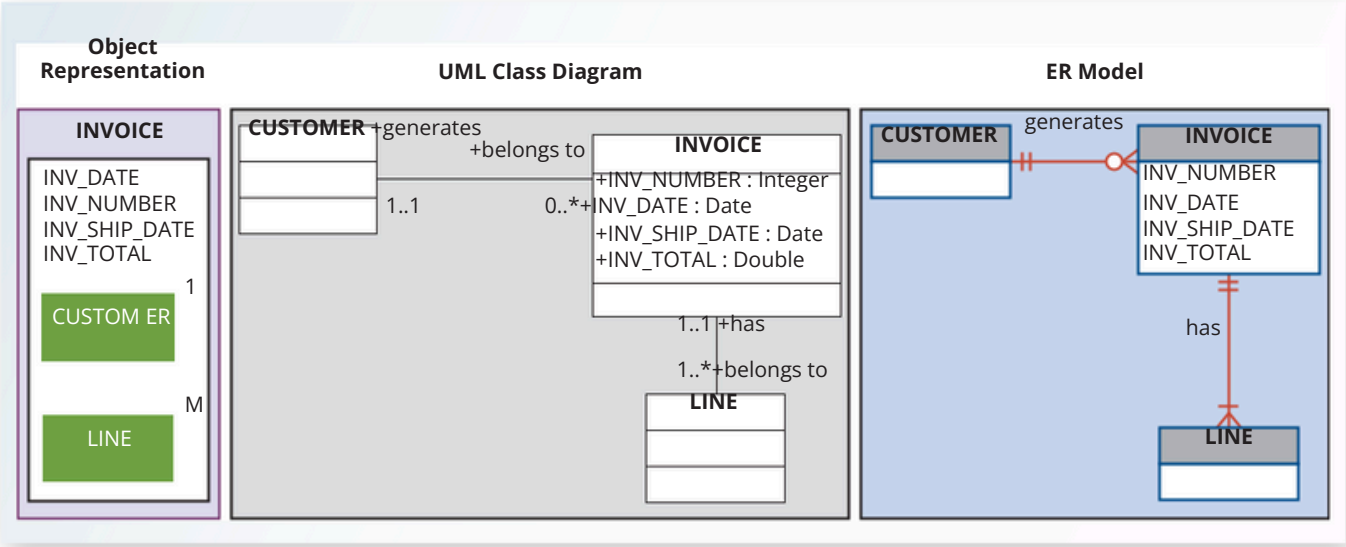
• INVOICE'in nesne temsili, aynı nesne kutusunda tüm ilgili nesneleri içerir. Bağlantıların (1 ve M) INVOICE ile ilgili nesnelerin ilişkisini gösterdiğine dikkat edin. Örneğin, CUSTOMER nesnesinin yanındaki "1", her INVOICE'un yalnızca bir CUSTOMER ile ilişkili olduğunu gösterir. LINE nesnesinin yanındaki "M", her INVOICE'un birden çok LINE içerdiğini belirtir. UML sınıf diyagramı üç ayrı kullanır.

•UML sınıf diyagramı, Bu basit faturalama problemini temsil etmek için üç nesne sınıfı (CUSTOMER, INVOICE ve LINE) ve iki ilişki kullanılır. İlişki bağlantı tiplerinin 1..1, 0.., ve 1.. sembolleriyle temsil edildiğine ve ilişkilerin her iki ucunda nesnelerin ilişkilerdeki farklı "rollerini" temsil etmek için adlandırıldığına dikkat edin.

•ER modeli de bu basit fatura sorununu temsil etmek için üç ayrı varlık ve iki ilişki kullanır.

OODM'nin ilerlemeleri, sistem modellemeden programlamaya kadar birçok alanı etkiledi. (Çoğu çağdaş programlama dili, Java, Ruby, Perl, C# ve Python dahil olmak üzere, OO kavramlarını benimsemiştir.) OODM'nin eklenen anlam katmanları, karmaşık nesnelerin daha zengin bir şekilde temsil edilmesini sağladı. Bu da uygulamaların, giderek daha karmaşık nesneleri yenilikçi şekillerde desteklemesine olanak tanıdı. Bir sonraki bölümde göreceğiniz gibi, bu evrimsel ilerlemeler, ilişkisel model üzerinde de etkili olmuştur.

Figure 2.4 OO, UML ve ER Modellerinin Karşılaştırması



2-5e Nesne/İlişkisel ve XML Daha karmaşık veri temsillerini destekleme talebiyle karşı karşıya kalan ilişkisel modelin ana satıcıları, modeli daha da geliştirdi ve genişletilmiş ilişkisel veri modelini (ERDM) yarattılar. ERDM, doğal olarak daha basit olan ilişkisel veritabanı yapısına birçok OO modelinin özelliğini ekler. ERDM, nesneler (veri ve yöntemlerin kapsüllenmesi), sınıflara dayalı genişletilebilir veri türleri ve kalıtım gibi OO özelliklerini destekleyen yeni bir nesil ilişkisel veritabanlarının doğmasına yol açtı. Bu nedenle, ERDM tabanlı bir DBMS genellikle nesne/ilişkisel veritabanı yönetim sistemi (O/R DBMS) olarak tanımlanır.

Genişletilmiş İlişkisel Veri Modeli (ERDM)
Nesne yönelimli modelin en iyi özelliklerini, doğası gereği daha basit bir ilişkisel veritabanı yapısal ortamında içeren bir modelidir. Genişletilmiş Varlık-İlişki Moıdeli (EERM)'ye bakınız.

Nesne/İlişkisel Veri Tabanı Yönetim Sistemi (O/R DBMS)
Genişletilmiş İlişkisel Veri Modeline (ERDM) dayanan bir DBMS'dir. ERDM, birçok ilişkisel veritabanı araştırmacısı tarafından savunulmuş olup, ilişkisel modelin OODM'ye (Nesne Yönelimli Veri Modeli) verdiği yanıttır. Bu model, nesne yönelimli modelin en iyi özelliklerinin çoğunu, doğası gereği daha basit bir ilişkisel veritabanı yapısında içerir.

Genişletilebilir İşaretleme Dili (XML)
Veri öğelerini temsil etmek ve işlemek için kullanılan bir metalangıdır. Diğer işaretleme dillerinin aksine, XML bir belgenin veri öğelerinin işlenmesine izin verir. XML, siparişler ve faturalar gibi yapılandırılmış belgelerin İnternet üzerinden değiştirilmesini kolaylaştırır.

Bugün, çoğu ilişkisel veritabanı ürünü nesne/ilişkisel olarak sınıflandırılabilir ve bunlar, OLTP ve OLAP veritabanı uygulamalarının dominant pazar payını temsil etmektedir. O/R DBMS'lerinin başarısı, modelin kavramsal basitliğine, veri bütünlüğüne, kullanımı kolay sorgulama diline, yüksek işlem performansına, yüksek kullanılabilirliğe, güvenliğe, ölçeklenebilirliğe ve genişletilebilirliğe bağlanabilir. Buna karşılık, OO DBMS, daha karmaşık nesneleri desteklemesi gereken bilgisayar destekli çizim/bilgisayar destekli üretim (CAD/CAM), coğrafi bilgi sistemleri (GIS), telekomünikasyon ve 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 yerine, çok belirli mühendislik ihtiyaçlarını ele almak için yaratıldı. İlişkisel model ise sağlam bir matematiksel temele dayalı olarak daha iyi veri yönetimine odaklanarak yaratıldı. Daha küçük bir sorun alanı setine odaklanması göz önüne alındığında, OO pazarının ilişkisel veri modeli pazarına göre bu kadar hızlı büyümemesi şaşırtıcı değildir.

Karmaşık nesnelerin kullanımı, İnternet devrimiyle birlikte hız kazandı. Organizasyonlar iş modellerini İnternet ile entegre ettiklerinde, bunun kritik iş bilgilerini erişme, dağıtma ve değiştirme potansiyelini fark ettiler. Bu, İnternet'in bir iş iletişim aracı olarak yaygın şekilde benimsenmesine yol açtı. Bu ortamda, Genişletilebilir İşaretleme Dili (XML), yapılandırılmış, yarı yapılandırılmış ve yapılandırılmamış verilerin verimli ve etkili bir şekilde değişimi için fiili standart olarak ortaya çıktı. XML verisi kullanan organizasyonlar, kısa sürede kelime işleme belgeleri, web sayfaları, e-postalar ve diyagramlar gibi büyük miktarda yapılandırılmamış veriyi yönetmeleri gerektiğini fark ettiler. Bu ihtiyacı karşılamak için XML veritabanları, yapılandırılmamış veriyi yerel XML formatında yönetmek için ortaya çıktı (daha fazla bilgi için Bölüm 15'e, Veritabanı Bağlantısı ve Web Teknolojileri'ne bakınız).