9-3 e Operasyon

Veritabanı değerlendirme ve test aşamasını geçtikten sonra operasyonel olarak kabul edilir. Bu noktada, veritabanı, yönetimi, kullanıcıları ve uygulama programları tam bir bilgi sistemi oluşturur.

Operasyonel aşamanın başlangıcı her zaman sistem evrimi sürecini başlatır. Hedeflenen tüm son kullanıcılar operasyon aşamasına girer girmez, test aşamasında öngörülemeyen sorunlar ortaya çıkmaya başlar. Sorunlardan bazıları acil "yama çalışması" gerektirecek kadar ciddiyken, diğerleri sadece küçük sıkıntılardır. Örneğin, veritabanı tasarımı web ile arayüz oluşturacak şekilde gerçekleştirilmişse, işlemlerin yoğunluğu iyi tasarlanmış bir sistemin bile tıkanmasına neden olabilir. Bu durumda, tasarımcılar darboğazın kaynağını tespit etmeli ve alternatif çözümler üretmelidir. Bu , işlemleri birden fazla bilgisayar arasında dağıtmak için yük dengeleme yazılımı kullanmayı, VTYS için mevcut önbelleği artırmayı . içerebilir. Operasyonel bir veritabanı sisteminin bir diğer kritik yönü de uyumluluktur. Veritabanı ve verileri, veri gizliliği, mahremiyet, fikri mülkiyet vb. ile ilgili sürekli değişen eyalet ve federal düzenlemelere uygun olmalıdır. Değişim talebi, tasarımcının sürekli endişesidir ve bu da 6. aşama olan bakım ve evrime yol açar.

9-3 f Bakım ve Evrim

Veritabanı yöneticisi, veritabanı içinde rutin bakım faaliyetleri gerçekleştirmeye hazır olmalıdır. Gerekli periyodik bakım faaliyetlerinden bazıları şunlardır:

- Önleyici bakım (yedekleme)
- Düzeltici bakım (kurtarma)
- Uyarlanabilir bakım (performansın artırılması, varlıkların ve niteliklerin eklenmesi vb.)
- Yeni ve eski kullanıcılar için erişim izinlerinin atanması ve sürdürülmesi
- Sistem denetimlerinin verimliliğini ve kullanışlılığını artırmak ve sistem performansını izlemek için veritabanı erişim istatistiklerinin oluşturulması
- Sistem tarafından oluşturulan istatistiklere dayalı periyodik güvenlik denetimleri
- Dahili faturalandırma veya bütçeleme amaçları için aylık, üç aylık veya yıllık sistem kullanım özetleri

Yeni bilgi gereksinimlerinin ortaya çıkma olasılığı ve ek raporlara ve yeni sorgu formatlarına olan talep, uygulama değişikliklerini ve veritabanı bileşenlerinde ve içeriklerinde olası küçük değişiklikleri gerektirir. Bu değişiklikler ancak veritabanı tasarımı esnek olduğunda ve tüm belgeler güncel ve çevrimiçi olduğunda kolayca uygulanabilir. Sonunda, en iyi tasarlanmış veritabanı ortamı bile artık bu tür evrimsel değişiklikleri içeremeyecek ve tüm DBLC süreci yeniden başlayacaktır.

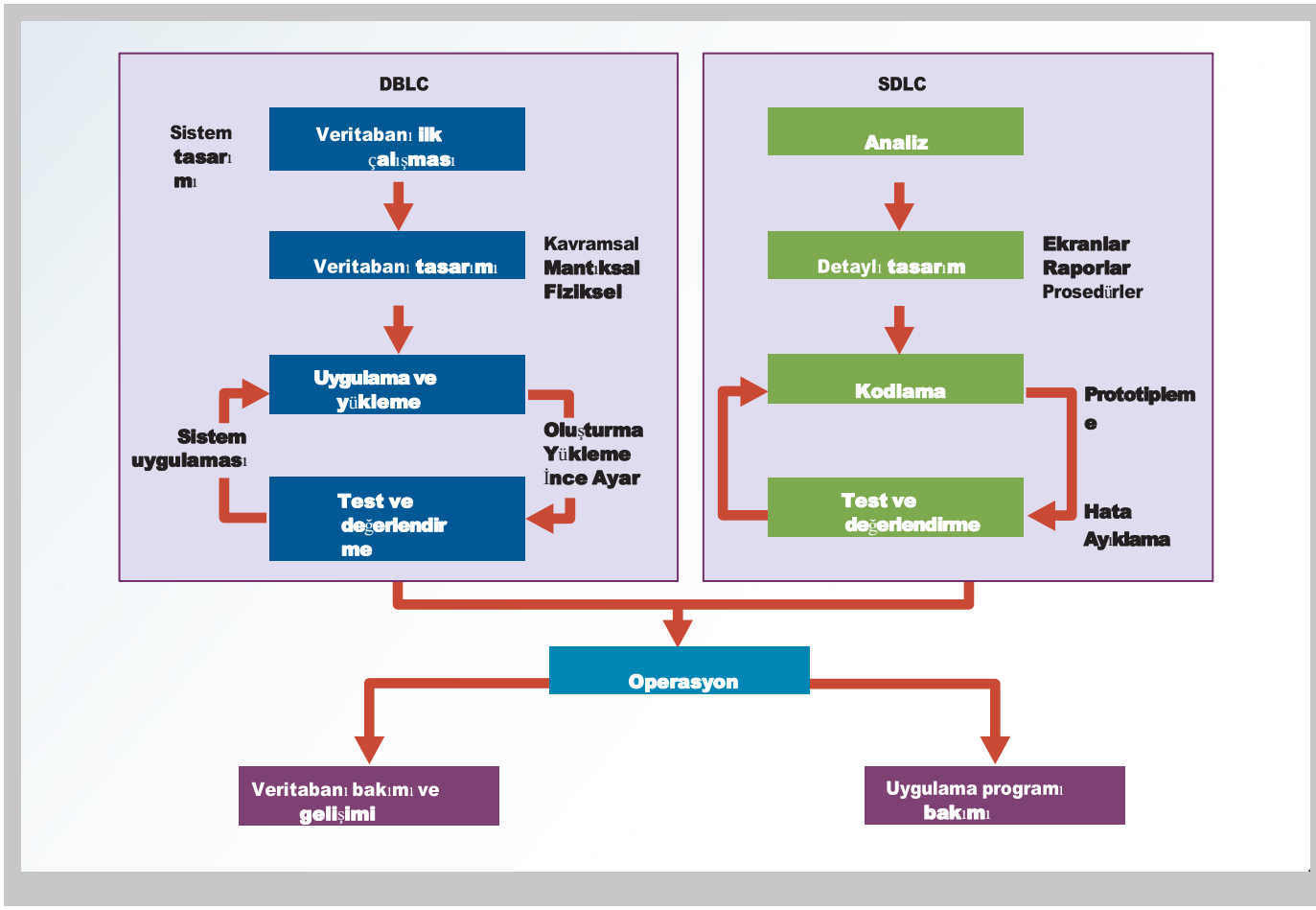
Gördüğünüz gibi, DBLC'de tanımlanan faaliyetlerin çoğu SDLC'dekilere benzemektedir. Bu şaşırtıcı olmamalıdır çünkü SDLC, DBLC faaliyetlerinin gerçekleştiği çerçevedir. SDLC ve DBLC içerisinde gerçekleşen paralel faaliyetlerin bir özeti Şekil 9.8'de gösterilmektedir.

9-4 Kavramsal Tasarım

DBLC'nin ikinci aşamasının, kavramsal tasarım, mantıksal tasarım ve fiziksel tasarım olmak üzere üç aşamadan ve ayrıca DBMS seçimi gibi kritik bir karardan oluşan veritabanı tasarımı olduğunu hatırlayın. **Kavramsal tasarım**, veritabanı tasarım sürecinin ilk aşamasıdır.

kavramsal tasarım
Gerçek dünya nesnelerini mümkün olduğunca gerçekçi bir şekilde temsil eden bir veritabanı yapısı modeli oluşturmak için veri modelleme tekniklerini kullanan bir süreç. Tasarım hem yazılımdan hem de donanımdan bağımsızdır.

Şekil 9.8 DBLC ve SDLC'deki Paralel Faaliyetler



Bu aşama, veritabanı yazılımından ve fiziksel detaylardan bağımsız bir veritabanı tasarlamaktır. Bu sürecin çıktısı, belirli bir sorun alanının ana veri varlıklarını, niteliklerini, ilişkilerini ve kısıtlamalarını tanımlayan kavramsal bir veri modelidir. Bu tasarım tanımlayıcı ve anlatı biçimindedir. Başka bir deyişle, genellikle grafiksel bir gösterimin yanı sıra ana veri unsurlarının, ilişkilerin ve kısıtlamaların metinsel açıklamalarından oluşur.

Bu aşamada veri modelleme, gerçek dünya nesnelerini mümkün olan en gerçekçi şekilde temsil eden soyut bir veritabanı yapısı oluşturmak için kullanılır. Kavramsal model, işin ve işlevsel alanlarının net bir şekilde anlaşılmasını sağlamalıdır. Bu soyutlama seviyesinde, kullanılacak donanım türü ve veritabanı modeli henüz belirlenmemiş olabilir. Bu nedenle, tasarım yazılım ve donanımdan bağımsız olmalıdır, böylece sistem daha sonra seçilen herhangi bir platformda kurulabilir.

Aşağıdaki **asgari veri kuralını** aklınızda bulundurun:

İhtiyaç duyulan her şey oradadır ve orada olan her şeye ihtiyaç vardır.

Başka bir deyişle, ihtiyaç duyulan tüm verilerin modelde olduğundan ve modeldeki tüm verilere ihtiyaç duyulduğundan emin olun. Veritabanı işlemlerinin gerektirdiği tüm veri elemanları modelde tanımlanmalı ve modelde tanımlanan tüm veri elemanları en az bir veritabanı işlemi tarafından kullanılmalıdır.

Ancak, minimum veri kuralını uygularken, aşırı kısa vadeli önyargılardan kaçının. Yalnızca işletmenin anlık veri ihtiyaçlarına değil, gelecekteki veri ihtiyaçlarına da odaklanın. Bu nedenle, veritabanı tasarımı gelecekteki değişiklikler ve eklemeler için yer bırakmalı ve işletmenin bilgi kaynaklarına yaptığı yatırımın kalıcı olmasını sağlamalıdır.

Kavramsal tasarımın Tablo 9.2'de listelenen dört adımı vardır.

Minimum veri kuralı

"İhtiyaç duyulan her şey oradadır ve orada olan her şeye ihtiyaç vardır" şeklinde tanımlanır. Başka bir deyişle, veritabanı işlemlerinin gerektirdiği tüm veri elemanları modelde tanımlanmalıdır ve tanımlanan tüm veri elemanları en az bir veritabanı işlemi tarafından kullanılmalıdır.

Tablo 9.2 Kavramsal Tasarım Adımları

ADIM	AKTİVİTE
1	Veri analizi ve gereksinimler
2	Varlık ilişkisi modelleme ve normalleştirme
3	Veri modeli doğrulaması
4	Dağıtık veritabanı tasarımı

Aşağıdaki bölümler bu adımları daha ayrıntılı olarak ele almaktadır.

9-4 a Veri Analizi ve Gereksinimler

Kavramsal tasarımın ilk adımı veri öğelerinin özelliklerini keşfetmektir. Etkili bir veri tabanı, başarılı karar verme süreçleri için temel bileşenleri üreten bir bilgi fabrikasıdır. Uygun veri elemanı özellikleri, uygun bilgiye dönüştürülebilen özelliklerdir. Bu nedenle, tasarımcının çabaları şunlara odaklanır:

- *Bilgi ihtiyaçları*. Ne tür bilgilere ihtiyaç vardır? , sistem tarafından hangi çıktıların (raporlar ve sorgular) üretilmesi gerekiyor, mevcut sistem hangi bilgileri üretiyor ve bu bilgiler ne ölçüde yeterli?
- *Bilgi kullanıcıları*. Bilgileri kim kullanacak? Bilgi nasıl kullanılacak? Çeşitli son kullanıcı veri görünümleri nelerdir?
- *Bilgi kaynakları*. Bilgi nerede bulunacak? Bilgi bulunduktan sonra nasıl çıkarılacak?
- *Bilgi anayasası*. Bilgiyi üretmek için hangi veri unsurlarına ihtiyaç vardır? Veri nitelikleri nelerdir? Verilerde hangi ilişkiler mevcuttur? Veri hacmi ne kadardır? Veriler ne sıklıkla kullanılıyor? Gerekli bilgileri üretmek için hangi veri kullanılacak?

Tasarımcı, gerekli bilgileri derlemek için bu soruların yanıtlarını çeşitli kaynaklardan elde eder:

- *Son kullanıcı veri görünümlerinin geliştirilmesi ve toplanması*. Veritabanı tasarımcısı ve son kullanıcı(lar) birlikte son kullanıcı veri görünümlerinin kesin bir tanımını geliştirir ve bunlar da veritabanının ana veri unsurlarının belirlenmesine yardımcı olmak için kullanılır.
- *Mevcut sistemin doğrudan gözlemlenmesi - mevcut ve istenen çıktı*. Son kullanıcı genellikle ister manuel ister bilgisayar tabanlı olsun mevcut bir sisteme sahiptir. Tasarımcı, verileri ve özelliklerini tanımlamak için mevcut sistemi gözden geçirir. Tasarımcı, veri türünü ve hacmini keşfetmek için girdi formlarını ve dosyaları (tabloları) inceler. Son kullanıcının halihazırda otomatik bir sistemi varsa, tasarımcı raporları desteklemek için gereken verileri tanımlamak üzere mevcut ve istenen raporları dikkatle inceler.
- *Sistem tasarım grubu ile arayüz oluşturma*. Bu bölümde daha önce de belirtildiği gibi, veritabanı tasarım süreci SDLC'nin bir parçasıdır. Bazı durumlarda, yeni sistemin tasarımından sorumlu sistem analisti aynı zamanda kavramsal veritabanı modelini de geliştirecektir. (Bu genellikle merkezi olmayan bir ortamda geçerlidir.) Diğer durumlarda, veritabanı tasarımı DBA'nın işinin bir parçası olarak kabul edilir. Bir DBA'nın varlığı genellikle resmi bir veri işleme departmanının varlığına işaret eder. DBA, veritabanını sistem analisti tarafından oluşturulan spesifikasyonlara göre tasarlar.

Doğru bir veri modeli geliştirmek için tasarımcının şirketin veri türleri, bunların kapsamı ve kullanımları hakkında kapsamlı bir anlayışa sahip olması gerekir. Ancak veri tek başına toplam iş hakkında gerekli anlayışı sağlamaz. Veritabanı açısından , verilerin ancak iş kuralları tanımlandığında anlamlı hale gelir. Bölüm 2'yi hatırlayın,

Veri Modelleri, bir *iş kuralının* belirli bir kuruluşun ortamındaki bir politika, prosedür veya ilkenin kısa ve kesin bir tanımı olduğunu belirtir. Bir kuruluşun faaliyetlerinin ayrıntılı bir tanımından türetilen iş kuralları, o kuruluşun ortamında eylemlerin oluşturulmasına ve uygulanmasına yardımcı olur. İş kuralları düzgün bir şekilde yazıldığında, varlıkları, atıfları, ilişkileri, bağlantıları, kardinaliteleri ve kısıtlamaları tanımlar.

Etkili olabilmek için iş kurallarının anlaşılması kolay olmalı ve kurumdaki herkesin kurallara ilişkin ortak bir yorumu paylaşmasını sağlamak için geniş çapta yaygınlaştırılmalıdır. Basit bir dil kullanan iş kuralları, *şirket tarafından görüldüğü* şekliyle verilerin temel ve ayırt edici özelliklerini tanımlar. İş kuralları örnekleri aşağıdaki gibidir:

- Bir müşteri bir hesap için çok sayıda ödeme yapabilir.
- Bir hesaptaki her ödeme yalnızca bir müşteriye yatırılır.
- Bir müşteri çok sayıda fatura oluşturabilir.
- Her fatura yalnızca bir müşteri tarafından oluşturulur.

Veritabanı tasarımındaki kritik rolleri göz önüne alındığında, iş kuralları gelişigüzel oluşturulmamalıdır. Kötü tanımlanmış veya yanlış iş kuralları, kuruluşun son kullanıcılarının ihtiyaçlarını karşılamayan veritabanı tasarımlarına ve uygulamalarına yol açar.

operasyonların tanımı
Bir kuruluşun faaliyet ortamını tanımlayan faaliyetlerin kesin, ayrıntılı, güncel ve kapsamlı bir şekilde gözden geçirilmiş bir tanımını sağlayan bir belge.

İdeal **olarak**, iş kuralları, bir kuruluşun çalışma ortamını tanımlayan faaliyetlerin kesin, güncel ve kapsamlı bir şekilde gözden geçirilmiş bir tanımını sağlayan bir belge olan **operasyonların** resmi bir **tanımından** türetilir. (Veritabanı tasarımcısı için çalışma ortamı hem veri kaynakları hem de veri kullanıcılarıdır). Doğal olarak, bir kuruluşun çalışma ortamı kuruluşun misyonuna bağlıdır. Örneğin, bir üniversitenin çalışma ortamı bir çelik üreticisinin, bir hava yolu şirketinin ya da bir huzurevinin çalışma ortamından oldukça farklı olacaktır. Yine de, kuruluşlar ne kadar farklı olsun, veri tabanı tasarımının veri *analizi* ve *gereksinimleri* bileşeni, veri ortamı ve veri kullanımı operasyonların bir tanımı içinde doğru ve kesin bir şekilde tanımlandığında geliştirilir.

Bir iş ortamında, operasyonların ve dolayısıyla iş kurallarının tanımlanması için ana bilgi kaynakları şirket yöneticileri, politika yapıcılar, departman yöneticileri ve şirket prosedürleri, standartlar ve operasyon kılavuzları gibi yazılı belgelerdir. İş kurallarına ilişkin daha hızlı ve doğrudan bir kaynak, son kullanıcılarla yapılan doğrudan görüşmelerdir. Ne yazık ki, algılar farklılık gösterdiğinden, iş kurallarının belirlenmesi söz konusu olduğunda son kullanıcı daha az güvenilir bir kaynak olabilir. Örneğin, bir bakım departmanı teknisyeni herhangi bir teknisyenin bir bakım prosedürünü başlatabileceğine inanabilir, oysa gerçekte yalnızca denetim yetkisine sahip teknisyenler böyle bir görevi yerine getirmelidir. Bu ayrım önemsiz görünebilir, ancak önemli yasal sonuçları vardır. Her ne kadar son kullanıcılar iş kurallarının geliştirilmesinde çok önemli katkılarda bulunsalar da, son kullanıcı algılarının doğrulanması önemlidir. Çoğu zaman, aynı işi yapan birkaç kişiyle yapılan görüşmeler, iş bileşenlerine ilişkin çok farklı algılar ortaya çıkarır. Böyle bir keşif "yönetim sorunlarına" işaret etse de, bu genel teşhis veritabanı tasarımcısına yardımcı olmaz. Bu tür sorunların keşfedilmesi durumunda, veritabanı tasarımcısının görevi farklılıkları uzlaştırmak ve iş kurallarının uygun ve doğru olduğundan emin olmak için uzlaştırma sonuçlarını doğrulamaktır.

İş kurallarının bilinmesi, tasarımcının işin nasıl yürüdüğünü ve verilerin şirket operasyonları içinde nasıl bir rol oynadığını tam olarak anlamasını sağlar. Sonuç olarak, tasarımcı şirketin iş kurallarını belirlemeli ve bunların verinin niteliği, rolü ve kapsamı üzerindeki etkisini analiz etmelidir.

İş kuralları, yeni sistemlerin tasarımında birkaç önemli fayda sağlar:

- Şirketin verilere bakışını standartlaştırmaya yardımcı olurlar.
- Kullanıcılar ve tasarımcılar arasında bir iletişim aracı oluştururlar.
- Tasarımcının verilerin doğasını, rolünü ve kapsamını anlamasını sağlarlar.
- Tasarımcının iş süreçlerini anlamasını sağlarlar.
- Tasarımcının uygun ilişki katılım kuralları ve yabancı anahtar kısıtlamaları geliştirmesini sağlarlar. Bölüm 4, Varlık İlişkisi (ER) Modellemesi.

Son nokta özellikle dikkat çekicidir; belirli bir ilişkinin zorunlu mu yoksa isteğe bağlı mı olduğu genellikle geçerli iş kuralının bir fonksiyonudur.

9-4 b Varlık İlişkisi Modelleme ve Normalleştirme

ER modelini oluşturmadan önce tasarımcı, tasarımın belgelendirilmesinde kullanılacak uygun standartları bildirmeli ve uygulamalıdır. Bu standartlar arasında diyagram ve sembollerin kullanımı, dokümantasyon yazım tarzı, düzen ve dokümantasyon sırasında uyulması gereken diğer kurallar yer alır. Tasarımcılar, özellikle de bir tasarım ekibinin üyesi olarak çalışıyorlarsa, bu çok önemli gerekliliği genellikle göz ardı ederler. Dokümantasyonun standartlaştırılmaması genellikle daha sonra iletişimde başarısızlık anlamına gelir ve iletişim başarısızlıkları genellikle kötü tasarım çalışmalarına yol açar. Buna karşılık, iyi tanımlanmış ve uygulanmış standartlar tasarım işini kolaylaştırır ve tüm sistem bileşenlerinin sorunsuz bir şekilde entegrasyonunu vaat eder (ancak garanti etmez).

İş kuralları genellikle varlıklar arasındaki ilişki(ler)in doğasını tanımladığından, tasarımcı bunları kavramsal modele dahil etmelidir. İş kurallarının tanımlanması ve ER diyagramları kullanılarak kavramsal modelin geliştirilmesi süreci Tablo 9.3'te gösterilen adımlar kullanılarak açıklanabilir.([8])

Tablo 9.3 ER Diyagramları Kullanarak Kavramsal Model Geliştirme

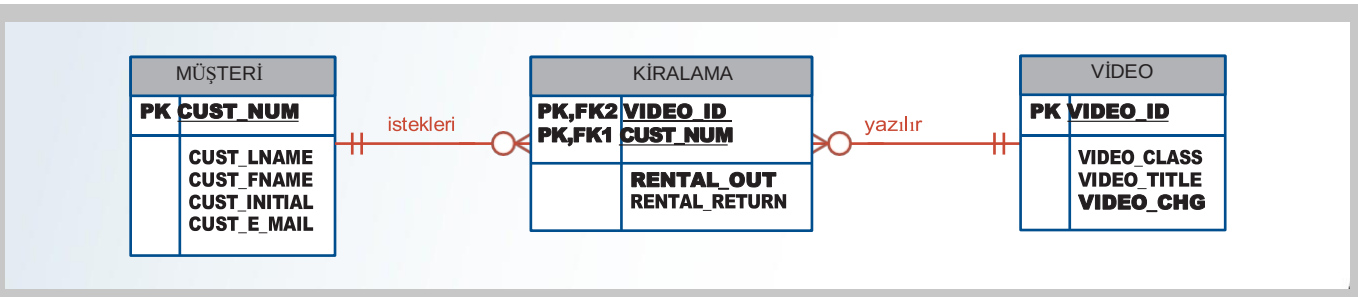
ADIM	AKTİVİTE
1	İş kurallarını belirleyin, analiz edin ve iyileştirin.
2	Adım 1'in sonuçlarını kullanarak ana varlıkları belirleyin.
3	Adım 1 ve 2'nin sonuçlarını kullanarak varlıklar arasındaki ilişkileri tanımlayın.
4	Her bir varlık için öznitelikleri, birincil anahtarları ve yabancı anahtarları tanımlayın.
5	Varlıkları normalleştirin. (Varlıkların bir RDBMS'de tablolar olarak uygulandığını unutmayın).
6	İlk ER diyagramını tamamlayın.
7	ER modelini son kullanıcıların bilgi ve işlem gereksinimlerine göre doğrulayın.
8	Adım 7'nin sonuçlarını kullanarak ER modelini değiştirin.

Tablo 9.3'te listelenen adımlardan bazıları eş zamanlı olarak gerçekleşir ve nor- malizasyon süreci gibi bazıları ek varlıklar ve/veya nitelikler için bir talep oluşturabilir ve böylece tasarımcının ER modelini revize etmesine neden olabilir. Örneğin, tasarımcı iki ana varlığı tanımlarken, iki ana varlık arasındaki çoktan çoğa ilişkiyi temsil eden bileşik köprü varlığını da tanımlayabilir.

Tekrar gözden geçirmek için, son kullanıcıları müşterilerin DVD film kiosk kiralamalarını takip etmek isteyen JollyGood Film Kiralama Şirketi için bir kavramsal model oluşturduğunuzu varsayalım. Şekil 9.9'da sunulan basit ER diyagramı, müşterilerin ve video kiralamalarının izlenmesine yardımcı olan bileşik bir varlığı göstermektedir. İş kuralları, VIDEO ve CUSTOMER varlıkları arasındaki ilişkilerin isteğe bağlı niteliğini tanımlar. Örneğin, müşterilerin bir videoyu almaları gerekmez. Bir videonun kioskta bulunması için teslim alınmış olması gerekmez. Bir müşteri birçok video kiralayabilir ve bir video birçok müşteri tarafından kiralanabilir. Özellikle, iki ana varlığı birbirine bağlayan bileşik RENTAL varlığına dikkat edin.

[8]Bakınız "Linking Rules to Models," Alice Sandifer ve Barbara von Halle, *Database Programming and Design*, 4(3), Mart 1991, s. 13-16. Kaynak eski gibi görünse de mevcut standart olmaya devam etmektedir. Teknoloji önemli ölçüde değişmiştir, ancak süreç değişmemiştir.

Şekil 9.9 Jollygood Film Kiralama ERD



Muhtemelen keşfedeceğiniz gibi, ilk ER modeli, sistemin gereksinimlerini karşılamadan önce birkaç revizyona tabi tutulabilir. Böyle bir revizyon süreci oldukça doğaldır. ER modelinin bir tasarım planı olduğu kadar bir iletişim aracı olduğunu da unutmayın. Bu nedenle, önerilen sistem kullanıcılarıyla bir araya geldiğinizde, ilk ER modeli "Gerçekten kastettiğiniz şey bu mu?" gibi sorulara yol açmalıdır. Örneğin, Şekil 9.9'da gösterilen ERD tamamlanmış olmaktan uzaktır. Açıkçası, tasarımın uygulanabilmesi için daha birçok niteliğin tanımlanması ve bağımlılıkların kontrol edilmesi gerekmektedir. Buna ek olaraktasarım henüz tipik video kiralama işlemlerini destekleyememektedir. Örneğin, her videonun kiralama amacıyla kullanılabilecek çok sayıda kopyası olması muhtemeldir. Ancak, Şekil 9.9'da gösterilen VIDEO varlığı kopyaların yanı sıra başlıkları da saklamak için kullanılırsa, tasarım Tablo 9.4'te gösterilen veri fazlalıklarını tetikler.

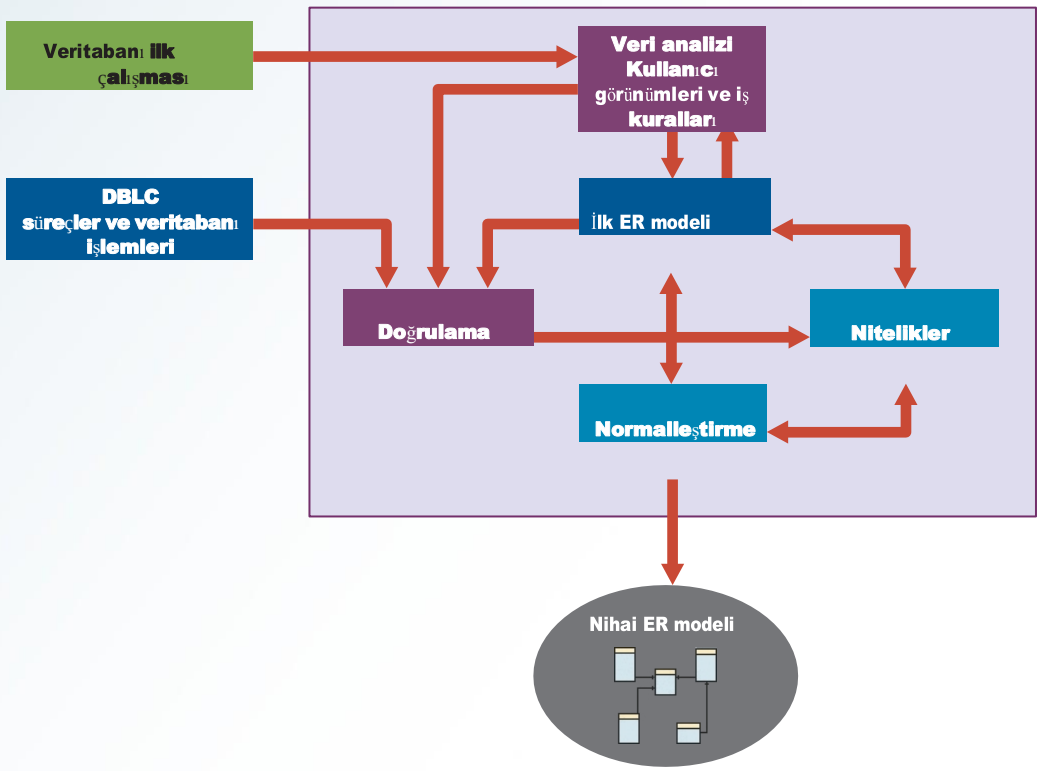
Tablo 9.4 Video Tablosundaki Veri Fazlalıkları

VIDEO_ID	VIDEO_TITLE	VİDEO_COPY	VIDEO_CHG	VIDEO_DAYS
SF-12345FT-1	Gezegen III'te Maceralar	1	$1.09	1
SF-12345FT-2	Gezegen III'te Maceralar	2	$1.09	1
SF-12345FT-3	Gezegen III'te Maceralar	3	$1.09	1
WE-5432GR-1	TipToe Kano ve Tyler 2: Bir Yolculuk	1	$1.09	2
WE-5432GR-2	TipToe Kano ve Tyler 2: Bir Yolculuk	2	$1.09	2

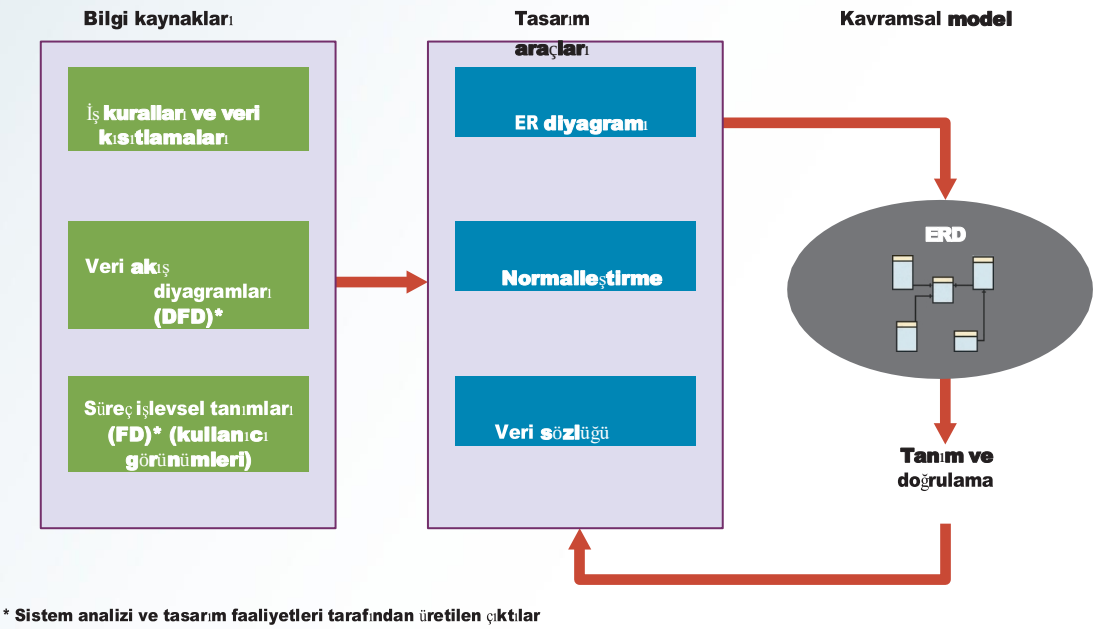
Şekil 9.9'da gösterilen ilk ERD, "Her başlık için birden fazla kopya mevcut mu?" sorusunun cevabını yansıtacak şekilde değiştirilmelidir. Ayrıca, ödeme işlemleri de desteklenmelidir. (Bölümün Problem 5'te bu ilk tasarımı değiştirme fırsatınız olacaktır).

Önceki tartışmadan, varlık ve öznitelik tanımı, normalleştirme ve doğrulama gibi ER modelleme faaliyetlerinin kesin bir sırayla gerçekleştiği izlenimini edinebilirsiniz. Aslında, ilk ER modelini tamamladıktan sonra, ER modelinin gerekli sistem taleplerini karşılayabilecek bir veritabanı tasarımını doğru bir şekilde temsil ettiğinden emin olana kadar faaliyetler arasında ileri geri hareket etme ihtimaliniz vardır. Faaliyetler genellikle paralel olarak gerçekleşir ve süreç yinelemelidir. Şekil 9.10 ER modelleme etkileşimlerini özetlemektedir. Şekil 9.11, tasarımcının kavramsal modeli üretmek için kullanabileceği tasarım araçları ve bilgi kaynakları dizisini özetlemektedir.

Şekil 9.10 ER Modelleme Birçok Faaliyete Dayalı Yinelemeli Bir Süreçtir



Şekil 9.11 Kavramsal Tasarım Araçları ve Bilgi Kaynakları



Tüm nesneler (varlıklar, nitelikler, ilişkiler, görünümler vb.), veri anormalliklerini ve fazlalık sorunlarını ortadan kaldırmaya yardımcı olmak için normalleştirme süreciyle birlikte kullanılan bir veri sözlüğünde tanımlanır. Bu ER modelleme süreci sırasında tasarımcı şunları yapmalıdır:

- Varlıkları, öznitelikleri, birincil anahtarları ve yabancı anahtarları tanımlayın. (Yabancı anahtarlar, varlıklar arasındaki ilişkilerin temelini oluşturur).
- Son kullanıcı ve işleme gereksinimlerini karşılamak için yeni birincil anahtar öznitelikleri ekleme konusunda kararlar alın.
- Bileşik ve çok değerli niteliklerin işlenmesine ilişkin kararlar alın.
- İşleme gereksinimlerini karşılamak için türetilmiş öznitelikler ekleme konusunda kararlar alın.
- Yabancı anahtarların 1:1 ilişkilerdeki yerleşimi hakkında kararlar verin.
- Gereksiz üçlü veya daha yüksek dereceli ilişkilerden kaçının.
- İlgili ER diyagramını çizin.
- Varlıkları normalleştirin.
- Tüm veri öğesi tanımlarını veri sözlüğüne dahil edin.
- Standart adlandırma kuralları hakkında kararlar verin.

Adlandırma kuralları gereksinimi önemlidir, ancak tasarımcının risk alması nedeniyle sıklıkla göz ardı edilir. Gerçek veritabanı tasarımı genellikle ekipler tarafından gerçekleştirilir. Bu nedenle, ekip üyelerinin adlandırma standartlarının tanımlandığı ve uygulandığı bir ortamda çalışmasını sağlamak önemlidir. Doğru dokümantasyon, tasarımın başarıyla tamamlanması için çok önemlidir ve adlandırma kurallarına bağlı kalmak veritabanı tasarımcılarına iyi hizmet eder. Aslında, kullanıcılardan gelen ortak bir nakarat şu şekilde görünmektedir: "Adlandırma kuralları konusunda neden bu kadar yaygara kopardığınızı bilmiyordum, ancak şimdi bu işi gerçekten yaptığım için, gerçek bir inanan haline geldim."

9-2 c Veri Modeli Doğrulaması

Veri modeli doğrulaması, kavramsal tasarım aşamasındaki son adımlardan biridir ve en kritik olanlardan biridir. Bu adımda ER modeli, veritabanı modeli tarafından desteklenebildiklerini doğrulamak için önerilen sistem süreçlerine karşı doğrulanmalıdır. Doğrulama, modelin bir dizi testten geçirilmesini gerektirir:

- Son kullanıcı veri görünümleri
- Gerekli tüm işlemler: SELECT, INSERT, UPDATE ve DELETE işlemleri
- Erişim hakları ve güvenlik
- İş dünyasının dayattığı veri gereksinimleri ve kısıtlamaları

Gerçek dünya veritabanı tasarımı genellikle ekipler tarafından yapıldığından, veritabanı tasarımı muhtemelen modüller olarak bilinen ana bileşenlere bölünmüştür. **Modül**, envanter, siparişler veya bordro gibi belirli bir iş fonksiyonunu ele alan bir bilgi sistemi bileşenidir. Bu koşullar altında her modül, kurumsal ER modelinin bir alt kümesi veya parçası olan bir ER segmenti tarafından desteklenir. Modüllerle çalışmak birkaç önemli amacı gerçekleştirir:

modül
(1) Özerk bir birim olarak uygulanabilen ve bazen bir sistem üretmek için birbirine bağlanan bir tasarım bölümü. (2) Envanter, siparişler veya bordro belirli bir işlevi yerine getiren bir bilgi sistemi bileşeni.

- Modüller (ve hatta içlerindeki segmentler) ekipler içindeki tasarım gruplarına devredilebilir ve bu da geliştirme çalışmalarını büyük ölçüde hızlandırır.
- Modüller tasarım işini basitleştirir. Karmaşık bir tasarımdaki çok sayıda varlık göz korkutucu olabilir. Her modül daha yönetilebilir sayıda varlık içerir.
- Modüller hızlı bir şekilde prototip haline getirilebilir. Uygulama ve uygulama programlamasındaki sorunlu noktalar daha kolay tespit edilebilir. Hızlı prototip oluşturma aynı zamanda büyük bir güven oluşturucudur.

- tamamı hızlı bir şekilde çevrimiçi hale getirilemese bile, bir veya daha fazla modülün uygulanması ilerleme kaydedildiğini ve sistemin en azından bir kısmının son kullanıcılara hizmet vermeye hazır olduğunu gösterecektir.

Modüller ne kadar kullanışlı olsalar da, kontrol edilmedikleri takdirde veritabanında hasara yol açabilecek ER model parçalarının gevşek bir koleksiyonunu temsil ederler. Örneğin, ER model parçaları:

- Aynı verinin örtüşen, yinelenen veya çelişen görünümlerini sunabilir
- Sistem modüllerindeki tüm süreçleri destekleyemeyebilir

Bu sorunlardan kaçınmak için, modüllerin ER parçalarının tek bir kurumsal ER modelinde birleştirilmesi daha iyidir. Bu süreç, merkezi bir ER modeli segmenti seçerek ve her seferinde bir ER modeli segmenti daha ekleyerek başlar. Her aşamada, modele eklenen her yeni varlık için, yeni varlığın kurumsal ER modelinde daha önce tanımlanmış bir varlıkla örtüşmediğini veya çakışmadığını doğrulamanız gerekir.

ER modeli segmentlerinin bir kurumsal ER modelinde birleştirilmesi, varlıkların dikkatli bir şekilde yeniden değerlendirilmesini ve ardından bu varlıkları tanımlayan niteliklerin ayrıntılı bir şekilde incelenmesini tetikler. Bu süreç birkaç önemli amaca hizmet eder:

- Öznitelik detaylarının ortaya çıkması, varlıkların kendilerinin de gözden geçirilmesine yol açabilir. Belki de ilk başta varlık olduğu düşünülen bazı bileşenlerin bunun yerine başka varlıkların içindeki nitelikler olduğu ortaya çıkacaktır. Ya da başlangıçta bir nitelik olarak düşünülen bir bileşenin, bir veya daha fazla yeni varlığın tanıtılmasını gerektirecek yeterli sayıda alt bileşen içerdiği ortaya çıkabilir.
- Öznitelik ayrıntılarına odaklanmak, birincil ve yabancı anahtarlar tarafından tanımlandıkları için ilişkilerin doğası hakkında ipuçları sağlayabilir. Yanlış tanımlanmış ilişkiler önce uygulama sorunlarına, daha sonra da uygulama geliştirme sorunlarına yol açar.
- İşleme ve son kullanıcı gereksinimlerini karşılamak için, mevcut bir birincil anahtarın yerine yeni bir birincil anahtar oluşturmak faydalı olabilir. Örneğin, Şekil 9.9'da gösterilen örnekte, VIDEO_ID ve CUST_NUM'dan oluşan orijinal birincil anahtarın yerine bir vekil birincil anahtar (RENTAL_ID) getirilebilir.
- Varlık detayları (nitelikler ve özellikleri) tam olarak tanımlanmadığı sürece, tasarımın normalleştirme kapsamını değerlendirmek zordur. Normalleştirme seviyelerinin bilinmesi, istenmeyen fazlalıklara karşı korunmaya yardımcı olur.
- Kaba veritabanı tasarım planının dikkatli bir şekilde gözden geçirilmesi muhtemelen revizyonlara yol açacaktır. Bu revizyonlar, tasarımın son kullanıcı gereksinimlerini karşılayabilmesini sağlamaya yardımcı olacaktır.

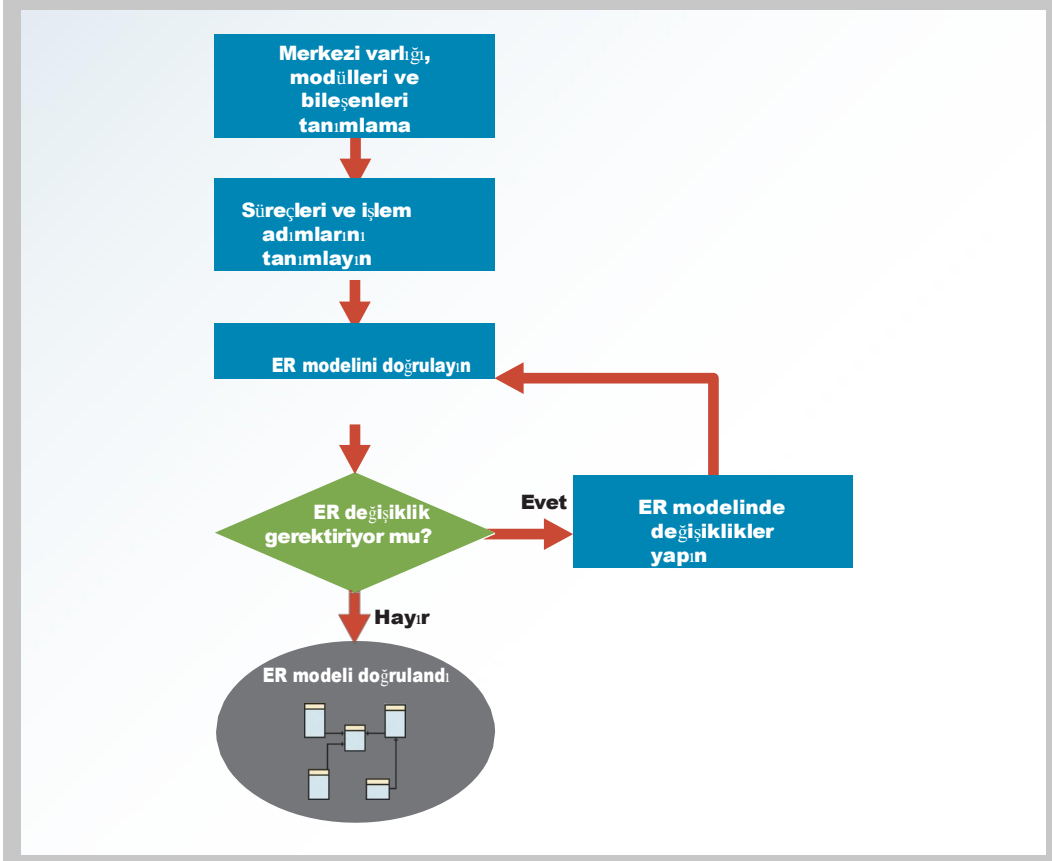
Birleştirme işlemi tamamlandıktan sonra, ortaya çıkan kurumsal ER modeli, modülün her bir karşı doğrulanır. ER modeli doğrulama süreci Tablo 9.5'te ayrıntılı olarak açıklanmıştır.

Tablo 9.5 ER Modeli Doğrulama Süreci

ADIM	AKTİVİTE
1	ER modelinin merkezi varlığını tanımlayın.
2	Her bir modülü ve bileşenlerini tanımlayın.
3	Her bir modülün işlem gereksinimlerini tanımlayın: Dahili: güncellemeler/eklemeler/silmeler/sorgular/raporlar Harici: modül arayüzleri
4	Tüm süreçleri modülün işleme ve raporlama gerekliliklerine göre doğrulayın.
5	Adım 4'te önerilen tüm gerekli değişiklikleri yapın.
6	Tüm modüller için Adım 2-5'i tekrarlayın.

Bu sürecin, ticari işlemlerin yanı sıra sistem ve kullanıcı gereksinimlerinin sürekli olarak doğrulanmasını gerektirdiğini unutmayın. Doğrulama sırası, sistemin her modülü için tekrarlanmalıdır. Şekil 9.12 sürecin yinelemeli yapısını göstermektedir.

Şekil 9.12 Yinelemeli ER Modeli Doğrulama Süreci



Doğrulama süreci, sistem operasyonlarının çoğunun odağı olan merkezi (en önemli) varlığın seçilmesiyle başlar.

Merkezi varlığı belirlemek için tasarımcı, modelin en fazla sayıda ilişkisine dahil olan varlığı seçer. ER diyagramında bu, kendisine diğerlerinden daha fazla çizgi bağlı varlıktır.

Bir sonraki adım, merkezi varlığın ait olduğu modülü veya alt sistemi belirlemek ve bu modülün sınırlarını ve kapsamını tanımlamaktır. Varlık, onu en sık kullanan modüle aittir. Her modül tanımlandıktan sonra, merkezi varlık modülün çerçevesi içine yerleştirilerek modülün ayrıntılarına odaklanmanız sağlanır.

Merkezi varlık/modül çerçevesi içinde şunları yapmalısınız

- *Modülün uyumluluğunu sağlayın*. **Uyumluluk** terimi, modülün varlıkları arasında bulunan ilişkilerin gücünü tanımlar. Bir modül *yüksek uyumluluk* göstermelidir; yani, varlıklar güçlü bir şekilde ilişkili olmalı ve modül eksiksiz ve kendi kendine yeterli olmalıdır.
- *Modül bağlantısını ele almak için her modülün diğer modüllerle olan ilişkilerini analiz edin*. **Modül bağlantısı**, modüllerin birbirinden ne ölçüde bağımsız olduğunu tanımlar. Modüller, diğer modüllerden bağımsız olduklarını gösteren *düşük bağlantı* göstermelidir. Düşük bağlantı, gereksiz modüller arası bağımlılıkları azaltarak gerçek anlamda modüler bir sistemin oluşturulmasını sağlar ve varlıklar arasındaki gereksiz ilişkileri ortadan kaldırır.

bütünlük
Aralarındaki ilişkilerin gücü bir modülün bileşenleri. Modüluyumluluğu yüksek olmalıdır.

modül bağlantısı
Modüllerin birbirinden ne ölçüde bağımsız olduğu.

Süreçler göre sınıflandırılabilir:

- Sıklık (günlük, haftalık, aylık, yıllık veya istisnalar)
- İşlem türü (INSERT veya ADD, UPDATE veya CHANGE, DELETE, sorgular ve raporlar, yığınlar, bakım ve yedeklemeler)

Tanımlanan tüm süreçler ER modeline göre doğrulanmalıdır. Gerekirse, uygun değişiklikler uygulanır. Süreç doğrulaması modelin tüm modülleri için tekrarlanır. Doğrulama sırasında kavramsal modele ilave varlıkların ve niteliklerin dahil edilmesini bekleyebilirsiniz.

Bu noktada, kavramsal bir model donanım ve yazılımdan bağımsız olarak tanımlanmıştır. Bu tür bir bağımsızlık, sistemin platformlar arasında taşınabilirliğini sağlar. Taşınabilirlik, başka bir VTYS'ye ve donanım platformuna geçişi mümkün kılarak veritabanının ömrünü uzatabilir.

9-4 d Dağıtık Veritabanı Tasarımı

Çoğu veritabanı için bir gereklilik olmasa da, bazılarının birden fazla coğrafi konuma dağıtılması gerekebilir. Veritabanına erişen süreçler de bir konumdan diğerine farklılık gösterebilir. Örneğin, bir perakende satış süreci ile bir depo depolama sürecinin farklı fiziksel konumlarda bulunması muhtemeldir. Veritabanı verileri ve süreçleri sistem genelinde dağıtılacaksa, bir veritabanının veritabanı parçaları olarak bilinen kısımları birkaç fiziksel konumda bulunabilir. Bir veritabanı **parçası**, belirli bir konumda depolanan bir veritabanının alt kümesidir. Veritabanı parçası, bir veya birden fazla tablodaki satırların veya sütunların bir alt kümesi olabilir.

Veritabanı parçası
Dağıtılmış bir veritabanının alt kümesi. Her ne kadar Parçalar, birtesisiçinde farklı yerlerde depolanabilir. bilgisayar ağında, tüm parçaların kümesi tek bir veritabanı olarak ele alınır.

Dağıtılmış veritabanı tasarımı, veritabanı bütünlüğünü, güvenliğini ve performansını sağlamak için veritabanı parçaları için optimum tahsis stratejisini tanımlar. Tahsis stratejisi, veritabanının nasıl bölümleneceğini ve her bir parçanın nerede saklanacağını belirler. Dağıtık süreçlerin getirdiği tasarım etkileri Bölüm 12, Dağıtık Veritabanı Yönetim Sistemleri'nde ayrıntılı olarak incelenmiştir.

9-5 DBMS Yazılım Seçimi

VTYS yazılımının seçimi, bilgi sisteminin sorunsuz çalışması için kritik öneme sahiptir. Sonuç olarak, önerilen VTYS yazılımının avantaj ve dezavantajları dikkatle incelenmelidir. Yanlış beklentilerden kaçınmak için, son kullanıcı hem VTYS'nin hem de veritabanının sınırlamalarından haberdar edilmelidir.

Satın alma kararını etkileyen faktörler şirketten şirkete değişmekle birlikte, en yaygın olanlardan bazıları şunlardır:

- *Maliyet*. Bu, orijinal satın alma fiyatının yanı sıra bakım, işletme, lisans, kurulum, eğitim ve dönüştürme maliyetlerini içerir.
- *DBMS özellikleri ve araçları*. Bazı veritabanı yazılımları, uygulama geliştirmeyi kolaylaştıran çeşitli araçlar içerir. Örneğin, örnek sorgu (QBE), ekran ressamları, rapor oluşturucular, uygulama oluşturucular ve veri sözlüklerinin kullanılabilirliği, hem son kullanıcı hem de uygulama programcısı için daha hoş bir çalışma ortamı yaratmaya yardımcı olur. Veritabanı yöneticisi olanakları, sorgu olanakları, kullanım kolaylığı, performans, güvenlik, para birimi kontrolü, işlem işleme ve üçüncü taraf desteği de VTYS yazılımı seçimini etkiler.
- *Temel model*. Veritabanı hiyerarşik, ağ, ilişkisel, nesne/ilişkisel, NoSQL, belge ve hatta bir grafik veri modeline dayalı olabilir. Bu bölümde ele alınan yöntemlerin çoğu ilişkisel veri modeline odaklansa da, temel tasarım süreçlerinin bazıları diğer veri modelleri için de geçerli olabilir.
- *Taşınabilirlik*. Bir VTYS platformlar, sistemler ve diller arasında taşınabilir olabilir.
- *DBMS donanım gereksinimleri*. Dikkate alınması gereken öğeler arasında işlemci(ler), RAM, disk alanı vb. yer alır.

9-6 Mantıksal Tasarım

Mantıksal tasarım, veritabanı tasarım sürecinin ikinci aşamasıdır. Mantıksal tasarımın amacı, belirli bir veri modeline dayanan ancak fiziksel düzeydeki ayrıntılardan bağımsız, kurum çapında bir veritabanı tasarlamaktır. Mantıksal tasarım, kavramsal modeldeki tüm nesnelerin seçilen veritabanı modeli tarafından kullanılan belirli yapılarla eşleştirilmesini gerektirir. Örneğin, ilişkisel bir VTYS için mantıksal tasarım, ilişkiler (tablolar), ilişkiler ve kısıtlamalar (başka bir deyişle, etki alanı tanımları, veri doğrulamaları ve güvenlik görünümleri) için spesifikasyonları içerir.
Mantıksal tasarım genellikle Tablo 9.6'da listelenen dört adımda gerçekleştirilir.

mantıksal tasarım
Tasarım aşamasında, kavramsal tasarımı seçilen VTYS'nin belirli yapılarıyla eşleştiren ve bu nedenle yazılıma bağlı olan bir aşama. Mantıksal tasarım, kavramsal tasarımı
Seçilen bir veritabanı yönetim sistemi için dahili modele.

Tablo 9.6 Mantıksal Tasarım Adımları

ADIM	AKTİVİTE
1	Kavramsal modeli mantıksal model bileşenleriyle eşleştirin.
2	Normalleştirme kullanarak mantıksal modeli doğrulayın.
3	Mantıksal model bütünlük kısıtlamalarını doğrulayın.
4	Mantıksal modeli kullanıcı gereksinimlerine göre doğrulayın.

Veri modelleme sürecinin çoğu gibi bu adımlar da sırayla değil, yinelemeli bir şekilde gerçekleştirilir. Aşağıdaki bölümler bu adımları daha ayrıntılı olarak ele almaktadır.

9-6a Kavramsal Modelin Mantıksal Model Bileşenleriyle Eşleştirilmesi

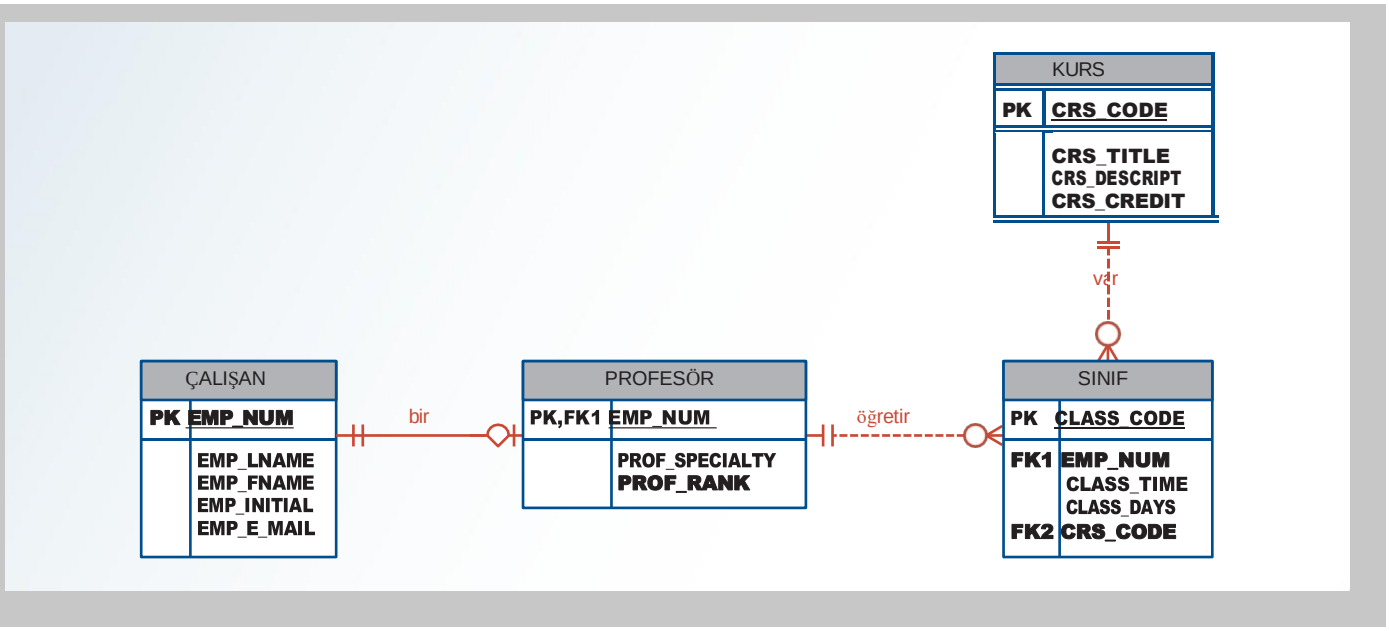
Mantıksal tasarımın oluşturulmasındaki ilk adım, kavramsal modeli seçilen veritabanı yapılarıyla eşleştirmektir. Bu kitap öncelikle ilişkisel veritabanlarıyla ilgilendiğinden ve mevcut veritabanı tasarım projelerinin çoğu ilişkisel veritabanı modeline dayandığından, bu bölüm ilişkisel yapıları kullanan mantıksal tasarıma odaklanmaktadır. Gerçek dünyada, mantıksal tasarım genel ER modelinin bir dizi ilişkiye (tablo), sütuna ve kısıt tanımına dönüştürülmesini içerir. Kavramsal modelin bir dizi ilişkiye dönüştürülmesi süreci Tablo 9.7'de gösterilmektedir.

Tablo 9.7 Kavramsal Modelin İlişkisel Modelle Eşleştirilmesi

ADIM	AKTİVİTE
1	Güçlü varlıkları eşleyin.
2	Süper tip/alt tip ilişkilerini eşleyin.
3	Zayıf varlıkları eşleyin.
4	İkili ilişkileri eşleyin.
5	Yüksek dereceli ilişkileri eşleyin.

Tablo 9.7'de belirtilen adımların sıralı değil, yinelemeli olduğunu unutmayın. Şekil 9.13'te gösterilen Basit Üniversite ER modeli örneği bu süreci göstermektedir.

Şekil 9.13 Basit Üniversite Kavramsal Modeli



Tablo 9.7'de belirtildiği gibi, mantıksal tasarım aşamasındaki ilk adım güçlü varlıkları tablolarla eşleştirmektir. Bölüm 4'ten, güçlü bir varlığın tüm ilişkilerinin "1" tarafında bulunan bir varlık olduğunu hatırlayın; yani, başka bir tablo için yabancı anahtar olan zorunlu bir niteliğe sahip olmayan bir varlık. Bu nedenle, tablolara çevrilecek ilk varlıklar EMPLOYEE ve COURSE varlıkları olacaktır. Bu durumda, tablo adını, sütunlarını ve özelliklerini tanımlarsınız. Örneğin, Simple College'ın güçlü varlıkları için ilişki tanımları şöyle olacaktır:

COURSE (CRS_CODE, CRS_TITLE, CRS_DESCRIPT, CRS_CREDIT)
BIRINCIL ANAHTAR: CRS_CODE

EMPLOYEE (EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_E_MAIL)
BIRINCIL ANAHTAR: EMP_NUM

Tüm güçlü varlıklar eşleştirildikten sonra, bir süper tip/alt tip ilişkisine dahil olan varlıkları veya zayıf varlıkları eşleştirmeye hazırsınız demektir. Simple College örneğinde, PROFESSOR varlığı EMPLOYEE varlığının bir alt türüdür. PROFESSOR aynı zamanda zayıf bir varlıktır çünkü birincil anahtarını EMPLOYEE'den alır ve varlığı EMPLOYEE'ye bağlıdır. Bu noktada, süper tip ve alt tip varlıklar arasındaki ilişkileri de tanımlamaya başlayabilirsiniz. Örneğin:

PROFESÖR (EMP_NUM, PROF_SPECIALTY, PROF_RANK)
BIRINCIL ANAHTAR: EMP_NUM
FOREIGN KEY: EMP_NUM REFERANSLARI ÇALIŞAN

Ardından, tüm ikili ilişkileri eşlemeye başlarsınız. Önceki örnekte, EMPLOYEE ile PROFESSOR arasındaki süper tip/alt tip ilişkisini tanımladınız. Bu, sürecin yinelemeli doğasını gösteren bir örnektir. Simple College ER modeli ile devam edersek, CLASS ilişkisini tanımlar ve PROFESSOR ve COURSE ile 1:M ilişkilerini tanımlarsınız:

CLASS (CLASS_CODE, *EMP_NUM*, CLASS_TIME, CLASS_DAYS, *CRS_CODE*)

BİRİNCİL ANAHTAR: SINIF_KODU
YABANCI ANAHTARLAR: EMP_NUM REFERANSLAR PROFESÖR
CRS_CODE REFERANSLAR KURS

Daha sonra, modeldeki tüm ilişkiler açıkça tanımlanana kadar üç veya daha fazla varlık arasındaki tüm ilişkilerle devam edeceksiniz. Mantıksal tasarımın tabloları Şekil 9.13'teki kavramsal tasarımda gösterilen varlıklara (ÇALIŞAN, PROFESÖR, DERS ve SINIF) karşılık gelmeli ve tablo sütunları da kavramsal belirtilen niteliklere karşılık gelmelidir. Bu sürecin nihai sonucu, bir sonraki adımın temelini oluşturacak olan ilişkiler, nitelikler ve ilişkilerin bir listesidir.

9-6b Normalleştirme Kullanarak Mantıksal Modeli Doğrulama

Mantıksal tasarım sadece uygun şekilde normalleştirilmiş tablolar içermelidir. Kavramsal modeli mantıksal modelle eşleme süreci bazı yeni öznitelikleri ortaya çıkarabilir veya yeni çok değerli veya bileşik öznitelikler keşfedilebilir. nedenle, tablolara yeni nitelikler eklenmesi veya mantıksal modele tamamen yeni tablolar eklenmesi çok muhtemeldir. Tanımlanan her tablo için (eski ve yeni), tüm özniteliklerin tanımlanan birincil anahtara tamamen bağımlı olduğundan ve tabloların en azından üçüncü normal formda (3NF) olduğundan emin olmalısınız.

Bu bölüm boyunca belirtildiği gibi, veritabanı tasarımı yinelemeli bir süreçtir. Normalleştirme gibi faaliyetler tasarım sürecinin farklı aşamalarında gerçekleşir. Bir adımı her tekrarladığınızda, model daha da rafine edilir ve daha iyi belgelenir. Yeni nitelikler oluşturulabilir ve uygun varlıklara atanabilir. Belirleyici ve bağımlı nitelikler arasındaki işlevsel bağımlılıklar değerlendirilir ve normalleştirme yoluyla veri anormallikleri önlenir.

9-6c Mantıksal Model Bütünlük Kısıtlamalarını Doğrulayın

Kavramsal modelin mantıksal bir modele çevrilmesi, öznitelik etki alanlarının ve uygun kısıtlamaların tanımlanmasını da gerektirir. Örneğin, Şekil 9.13'teki CLASS varlığında görüntülenen CLASS_CODE, CLASS_DAYS ve CLASS_TIME öznitelikleri için etki alanı tanımları bu şekilde yazılır:

CLASS_CODE	geçerli bir sınıf kodudur. Tip: sayısal Aralık: düşük değer51000 yüksek değer59999 Ekran formatı: 9999 Uzunluk: 4
SINIF_GÜNLERİ	geçerli bir gün kodudur. Tip: karakter Ekran formatı: XXX Geçerli girişler: MWF, TR, M, T, W, R, F, S Uzunluk: 3
CLASS_TIME	geçerli bir zamandır. Tür: karakter Ekran formatı: 99:99 (24 saatlik) Ekran aralığı: 06:00 - 22:00 Uzunluk: 5

Tanımlanan tüm bu kısıtlar mantıksal veri modeli tarafından desteklenmelidir. Bu aşamada, bu kısıtlamaları uygun ilişkisel model kısıtlamalarıyla eşleştirmeniz gerekir. Örneğin, CLASS_DAYS özniteliği, geçerli karakter kombinasyonlarının bir listesiyle sınırlandırılması gereken karakter verileridir. Burada, izin verilen tek değerlerin "MWF," "TR," "M, "T," "W," "R," "F," ve S" olmasını zorunlu kılmak için bu özniteliği bir CHECK IN kısıtlamasına sahip olacak şekilde tanımlarsınız. Bu adım sırasında, hangi niteliklerin zorunlu ve hangilerinin isteğe bağlı olduğunu da tanımlar ve tüm varlıkların varlık ve referans bütünlüğünü korumasını sağlarsınız.

Veritabanını kullanma hakkı da mantıksal tasarım aşamasında belirlenir. Tabloları kimlerin kullanmasına izin verilecek ve tabloların hangi bölümleri hangi kullanıcılar tarafından kullanılabilecek? İlişkisel bir çerçeve içinde, bu soruların yanıtları uygun görünümlerin tanımlanmasını gerektirir. Örneğin, belirli bir süreç, sınıf programları hakkında veri almak için aşağıdaki görünümün oluşturulmasını gerektirebilir:

```
CREATE VIEW vSCHEDULE AS
SEÇİNİZ     EMP_LNAME, EMP_FNAME, CLASS_CODE, CRS_TITLE,
            CLASS_TIME, CLASS_DAYS
FROM        PROFESÖR, SINIF, KURS
NEREDE      PROFESSOR.EMP_NUM 5 CLASS.EMP_NUM VE
            CLASS.CRS_CODE 5 COURSE.CRS_CODE
```

Bu aşamada tüm görünümlerin çözümlenebildiğinden ve verilerin gizliliğini sağlamak için güvenliğin uygulandığından emin olmak için özel dikkat gerekir. Ayrıca, dağıtılmış bir veritabanı tasarımıyla çalışıyorsanız, veriler birden fazla konumda depolanabilir ve her konum farklı güvenlik kısıtlamalarına sahip olabilir. Mantıksal model bütünlük kısıtlamalarını doğruladıktan sonra, modeli son kullanıcı gereksinimlerine göre doğrulamaya hazırsınız demektir.

9-6d Mantıksal Modeli Kullanıcı Gereksinimlerine Göre Doğrulayın

Mantıksal tasarım, yazılımdan bağımsız kavramsal modeli yazılıma bağımlı bir modele dönüştürür. Mantıksal tasarım sürecindeki son adım, tüm mantıksal model tanımlarının tüm son kullanıcı verileri, işlem ve güvenlik gereksinimlerine göre doğrulanmasıdır. Mantıksal modelin doğruluğunu sağlamak için Tablo 9.5'te gösterilene benzer bir süreç tekrar gerçekleştirilir. Artık sahne, sistemin seçilen DBMS/donanım ortamında çalışmasını sağlayacak fiziksel gereklilikleri tanımlamaya hazırdır.

fizîksel tasarim
Bir veritabanının veri depolama ve erişim özelliklerini eşleştiren veritabanı tasarımının bir aşaması. Çünkü bu özellikler Donanım tarafından desteklenen cihaz türlerinin bir fonksiyonu olarak, sistem fiziksel tasarımı tarafından desteklenen veri erişim yöntemleri hem donanıma hem de yazılıma bağlıdır. Ayrıca bkz. *fiziksel model*.

9-7 Fiziksel Tasarım

Fiziksel tasarım, veri tabanının bütünlüğünü, güvenliğini ve performansını sağlamak için veri depolama organizasyonunu ve veri erişim özelliklerini belirleme sürecidir. Bu, veritabanı tasarım sürecinin son aşamasıdır. Depolama özellikleri, donanım tarafından desteklenen cihaz türlerinin, sistem tarafından desteklenen veri erişim yöntemlerinin ve DBMS'nin bir fonksiyonudur. Fiziksel tasarım, yalnızca depolama aygıt(lar)verilerin erişilebilirliğini değil, sistemin performansını da etkileyen çok teknik bir iş haline gelebilir.

Fiziksel tasarım aşaması Tablo 9.8'deki adımlardan oluşur.

Çevrimiçi İçerik
Fiziksel tasarım özellikle önemlidir eski hiyerarşik sistemde ve Ek K ve L'de açıklanan ağ modelleri, Hiyerarşik Veritabanı Modeli ve Ağ Veritabanı Model, sırasıyla. Her iki eke de www.cengage.com .adresinden ulaşılabilir

Tablo 9.8 Fiziksel Tasarım Adımları

ADIM	AKTİVİTE
1	Veri depolama organizasyonunu tanımlayın.
2	Bütünlük ve güvenlik önlemlerini tanımlayın.
3	Performans ölçümlerini belirleyin.

Aşağıdaki bölümler bu adımları daha ayrıntılı olarak ele almaktadır.

9-7a Veri Depolama Organizasyonunu Tanımlama

Veri depolama organizasyonunu tanımlamadan önce, yönetilecek veri hacmini ve veri kullanım modellerini belirlemeniz gerekir.

- Veri hacmini bilmek, veritabanı için ne kadar depolama alanı ayıracağınızı belirlemenize yardımcı olacaktır. Bunu yapmak için tasarımcı, ER modeli doğrulaması sırasında kullanılana benzer bir süreç izler. Her tablo için olası tüm işlemleri, bunların sıklığını ve hacmini belirleyin. Her işlem için, veritabanına eklenecek veya veritabanından silinecek veri miktarını belirlersiniz. Bu bilgi, ilgili tabloda depolanacak veri miktarını belirlemenize yardımcı olacaktır.
- Tersine, yeni verilerin ne sıklıkla eklendiğini, güncellendiğini ve alındığını bilmek, tasarımcının veri kullanım modellerini belirlemesine yardımcı olacaktır. Kullanım şekilleri, özellikle dağıtık veritabanı tasarımında kritik öneme sahiptir. Örneğin, haftalık toplu yüklemeler veya aylık toplama raporları oluşturulacak mı? Sisteme ne sıklıkla yeni veri ekleniyor? Bu bilgiler, sistemin sorgu performans parametrelerini belirlemek için de faydalıdır.

Önceki iki bilgi parçasıyla donatılmış olan tasarımcı şunları yapmalıdır:

- *Her tablo için konumu ve fiziksel depolama organizasyonunu belirleyin*. Bölüm 9-3c'de gördüğünüz gibi, tablolar tablo alanlarında saklanır ve bir tablo alanı birden fazla tablodan veri tutabilir. Bu adımda, tasarımcı hangi tabloların hangi tablo alanlarını kullanacağını ve tablo alanlarının konumunu atar. Örneğin, çoğu ilişkisel veritabanında bulunan kullanışlı bir teknik, kümelenmiş tabloların kullanılmasıdır. **Kümelenmiş** tablo depolama tekniği, birbiriyle ilişkili iki tablodan ilgili satırları diskteki bitişik veri bloklarında depolar. Bu, verilerin sıralı olarak bitişik konumlarda depolanmasını sağlar, böylece veri erişim süresini azaltır ve sistem performansını artırır.

kümelenmiş tablo
Birbiriyle iki tablodan ilgili satırları diskteki bitişik veri bloklarında depolayan bir depolama tekniği.

- *Her tablo için kullanılacak dizinleri ve dizin türlerini belirleyin*. Önceki bölümlerde gördüğünüz gibi, indeksler bir sütundaki veri değerlerinin benzersizliğini sağlamak ve veri aramalarını kolaylaştırmak için kullanışlıdır. DBMS'nin her tablonun birincil anahtarı için otomatik olarak benzersiz bir dizin oluşturduğunu da biliyorsunuz. Bölüm 11'de çeşitli dizin organizasyonu türleri hakkında bilgi edineceksiniz. Bu adımda, gerekli tüm dizinleri tanımlar ve veri kullanım modellerine ve performans gereksinimlerine göre kullanılacak en iyi organizasyon türünü belirlersiniz.
- *Her tabloda kullanılacak görünümleri ve görünüm türlerini belirleyin*. Bölüm 8'de öğrendiğiniz gibi, görünümler kullanıcı veya işlem ihtiyaçlarına göre verilere erişimi sınırlamak için kullanışlıdır. Görünümler ayrıca işlemeyi ve son kullanıcı veri erişimini basitleştirmek için de kullanılabilir. Bu adımda tasarımcı, tüm görünümlerin uygulanabileceğinden ve yalnızca gerekli verileri sağladığından emin olmalıdır. Tasarımcı ayrıca VTYS tarafından desteklenen görünüm türlerini ve bunların sistem hedeflerinin karşılanmasına nasıl yardımcı olabileceğini de bilmelidir.

9-7b Bütünlük ve Güvenlik Önlemlerini Tanımlayın

Tabloların, dizinlerin ve görünümlerin fiziksel organizasyonu tanımlandıktan sonra, veritabanı son kullanıcılar için hazırdır. Ancak, kullanıcıların veritabanındaki verilere erişebilmeleri için önce kimliklerinin doğru bir şekilde doğrulanması gerekir. Fiziksel tasarımın bu adımında iki görev ele alınmalıdır:

- *Kullanıcı ve güvenlik gruplarını ve rollerini tanımlayın*. Kullanıcı yönetimi, veritabanı tasarımından çok veritabanı yönetiminin bir işlevidir. Ancak, bir tasarımcı olarak veritabanı güvenliğini düzgün bir şekilde uygulamak için farklı kullanıcı türlerini ve kullanıcı gruplarını bilmeniz gerekir. Çoğu DBMS uygulaması veritabanı rollerinin kullanımını destekler. **Veritabanı rolü**, bir kullanıcı ya da gruba birim olarak atanabilen bir dizi veritabanı ayrıcalığıdır. Örneğin, vSCHEDULE görünümüne Okuma erişimi olan bir Danışman rolü tanımlayabilirsiniz.

veritabanı rolü
Olarak atanabilecek dizi veritabanı ayrıcalığı bir kullanıcı veya gruba bir birim.

- *Güvenlik kontrolleri atayın*. DBMS, yöneticilerin bir kullanıcıya veya kullanıcı grubuna veritabanı nesneleri için belirli erişim hakları atamasına da olanak tanır. Örneğin, CLASS tablosunda mhendley kullanıcısına SELECT ve UPDATE erişim haklarını atayabilirsiniz. Bir erişim hakkı, belirli bir kullanıcıdan veya kullanıcı gruplarından da iptal edilebilir. Bu özellik, veritabanı yedeklemeleri, zamanlanmış bakım etkinlikleri ve hatta veri ihlali olayları kullanışlı olabilir.

9-7c Performans Ölçümlerinin Belirlenmesi

Veriler farklı konumlara dağıtıldığında fiziksel tasarım daha karmaşık hale gelir çünkü performans iletişim ortamının veriminden etkilenir. Bu tür karmaşıklıklar göz önüne alındığında, tasarımcıların fiziksel düzeydeki faaliyetlerin mümkün olduğunca çoğunu gizleyen veritabanı yazılımlarını tercih etmeleri şaşırtıcı değildir. İlişkisel modellerin bilgisayarın fiziksel özelliklerinin karmaşıklığını gizleme eğiliminde olmasına rağmen, ilişkisel veritabanlarının performansı fiziksel depolama özelliklerinden etkilenir. Örneğin performans; arama süresi, sektör ve blok (sayfa) boyutu, tampon havuzu boyutu ve disk plakaları ile okuma/yazma kafalarının sayısı gibi depolama ortamının özelliklerinden etkilenebilir. Buna ek olarak, bir dizinin oluşturulması gibi faktörler ilişkisel veritabanının performansı, yani veri erişim hızı ve verimliliği üzerinde önemli bir etkiye sahip olabilir.

Özetle, fiziksel tasarım performans ölçümü, önceki aşamalarda belirlenen son kullanıcı performans gereksinimlerini karşılamalarını sağlamak için VTYS ve ince ayar yapmakla ilgilenir.

Not

Kullanılabilecek veritabanı performansı ve sorgu optimizasyon teknikleri hakkında ayrıntılı bir tartışma için bkz. Bölüm 11, Veritabanı Performans Ayarlama ve Sorgu Optimizasyonu.

Önceki bölümlerde mantıksal ve fiziksel tasarım faaliyetlerine ilişkin tartışmalar birbirinden ayrılmıştır. Aslında, mantıksal ve fiziksel tasarım paralel olarak, tablo bazında gerçekleştirilebilir. Bu tür paralel faaliyetler, tasarımcının yazılım ve donanımın özelliklerinden tam olarak yararlanabilmesi için bunları tam olarak anlamasını gerektirir.

9-8 Veritabanı Tasarım Stratejileri

yukarıdan aşağıya tasarım
Bir sistemin ana yapılarını tanımlayarak başlayan ve daha sonra bu içindeki daha küçük birimleri tanımlamaya geçen bir tasarım felsefesi. Veritabanı tasarımında, bu süreç önce varlıkları tanımlar ve daha sonra varlıklar içindeki öznitelikleri tanımlar.

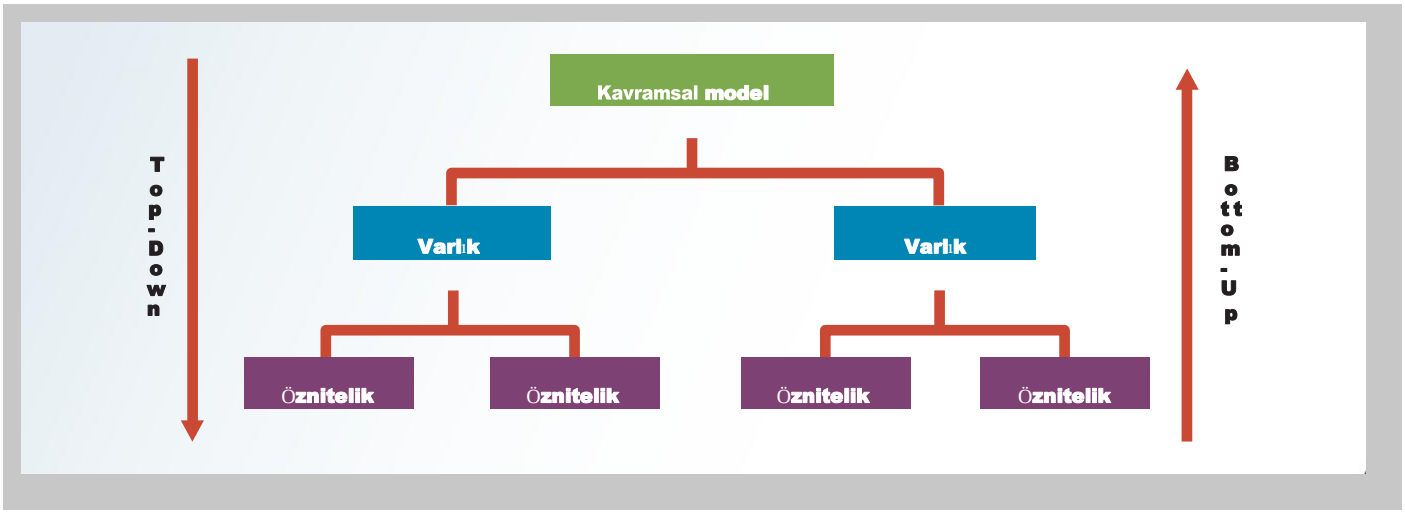
aşağıdan yukarıya tasarım
Bireysel tasarım bileşenlerini tanımlayarak başlayan ve daha sonra bunları daha büyük birimler halinde toplayan bir tasarım felsefesi. Veritabanı tasarımında süreç, niteliklerin tanımlanmasıyla başlar ve daha sonra bunları varlıklar halinde gruplandırır.

Veritabanı tasarımına yönelik iki klasik yaklaşım vardır:

- **Yukarıdan aşağıya tasarım**, veri setlerini tanımlayarak başlar ve ardından bu setlerin her biri için veri öğelerini tanımlar. Bu süreç, farklı varlık türlerinin tanımlanmasını ve her bir varlığın özniteliklerinin tanımlanmasını içerir.
- **Aşağıdan yukarıya tasarım** önce veri unsurlarını (öğeleri) tanımlar ve ardından bunları veri kümelerinde bir araya getirir. Başka bir deyişle, önce öznitelikleri tanımlar ve ardından bunları varlıklar oluşturacak şekilde gruplandırır.

Bu iki yaklaşım Şekil 9.14'te gösterilmektedir. Yukarıdan aşağıya veya aşağıdan yukarıya prosedürlere öncelikli vurgunun seçilmesi genellikle sorunun kapsamına veya kişisel tercihlere bağlıdır. Her ne kadar iki metodoloji birbirini dışlamaktan ziyade birbirini tamamlayıcı nitelikte olsa da, az sayıda varlık, nitelik, ilişki ve işlem içeren küçük veritabanları için aşağıdan yukarıya yaklaşıma öncelik vermek daha verimli olabilir. Varlıkların, ilişkilerin ve işlemlerin sayısının, çeşitliliğinin ve karmaşıklığının çok fazla olduğu durumlarda, öncelikle yukarıdan aşağıya bir yaklaşım daha kolay olabilir. Çoğu şirkette sistem geliştirme ve veritabanı tasarımı için standartlar zaten mevcuttur.

Şekil 9.14 Yukarıdan Aşağıya ve Aşağıdan Yukarıya Tasarım Sıralaması



Not

Öncelikle yukarıdan aşağıya bir yaklaşım seçildiğinde bile, mevcut tablo yapılarını revize eden normalleştirme süreci kaçınılmaz olarak aşağıdan yukarıya bir tekniktir. ER modelleri, niteliklerin ve varlıkların seçimi aşağıdan yukarıya olarak tanımlanabilse bile yukarıdan aşağıya bir süreç oluşturur. Hem ER modeli hem de normalleştirme teknikleri çoğu tasarımın temelini oluşturduğundan, yukarıdan aşağıya ve aşağıdan yukarıya tartışması gerçek bir farktan ziyade teorik bir ayrıma dayanıyor olabilir.

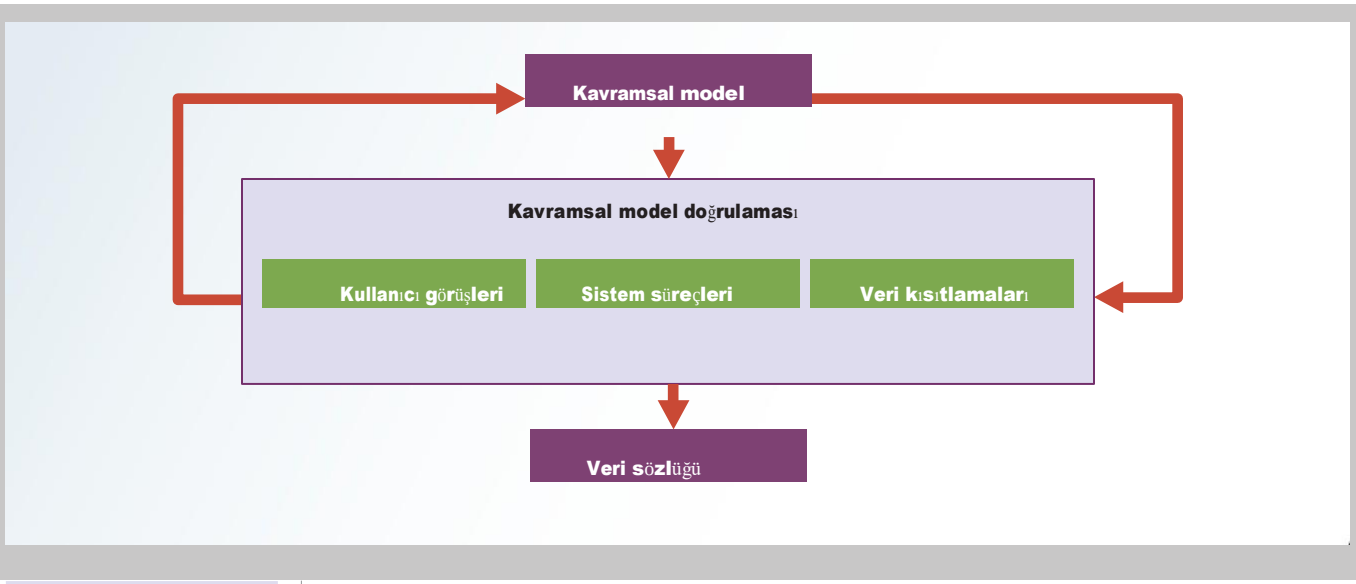
9-9 Merkezi ve Merkezi Olmayan Tasarım

Veritabanı tasarımına yönelik iki genel yaklaşım (aşağıdan yukarıya ve yukarıdan aşağıya), sistemin kapsamı ve büyüklüğü, şirketin yönetim tarzı ve şirketin yapısı (merkezi veya merkezi olmayan) gibi faktörlerden etkilenebilir. Bu faktörlere bağlı olarak, veritabanı tasarımı çok farklı iki tasarım felsefesine dayanabilir: merkezi ve merkezi olmayan.

Merkezi tasarım, veri bileşeninin nispeten az sayıda nesne ve yordama sahip olduğu durumlarda verimlidir. Tasarım oldukça basit bir veritabanında gerçekleştirilebilir ve temsil edilebilir. Merkezi tasarım nispeten basit, küçük veritabanları için tipiktir ve tek bir veritabanı yöneticisi veya küçük, gayri resmi bir tasarım ekibi tarafından başarılı bir şekilde yapılabilir. Şirket faaliyetleri ve sorunun kapsamı, tek bir tasarımcının bile sorun(lar)ı tanımlamasına, kavramsal tasarımı oluşturmasına, kavramsal tasarımı kullanıcı görüşleriyle doğrulamasına, tasarımın etkinliğini sağlamak için sistem süreçlerini ve veri kısıtlamalarını tanımlamasına ve tasarımın tüm gerekliliklere uygun olmasını sağlamasına izin verecek kadar sınırlıdır. (Merkezi tasarım küçük şirketler için tipik olsa da, bunun onlarla sınırlı olduğunu düşünme hatasına düşmeyin. Büyük şirketler bile nispeten basit bir veritabanı ortamında çalışabilir). Şekil 9.15 merkezi tasarım seçeneğini özetlemektedir. Merkezi tasarım yaklaşımında tek bir kavramsal tasarımın tamamlandığını ve ardından doğrulandığını unutmayın.

merkezî tasarım
Tüm veritabanı tasarım kararlarının küçük bir grup insan tarafından merkezi olarak yürütüldüğü bir süreç. aşağıya tasarım yaklaşımında problem alanı şu olduğunda uygundur Bir kuruluştaki tek bir birim veya departmanda olduğu gibi nispeten küçük.

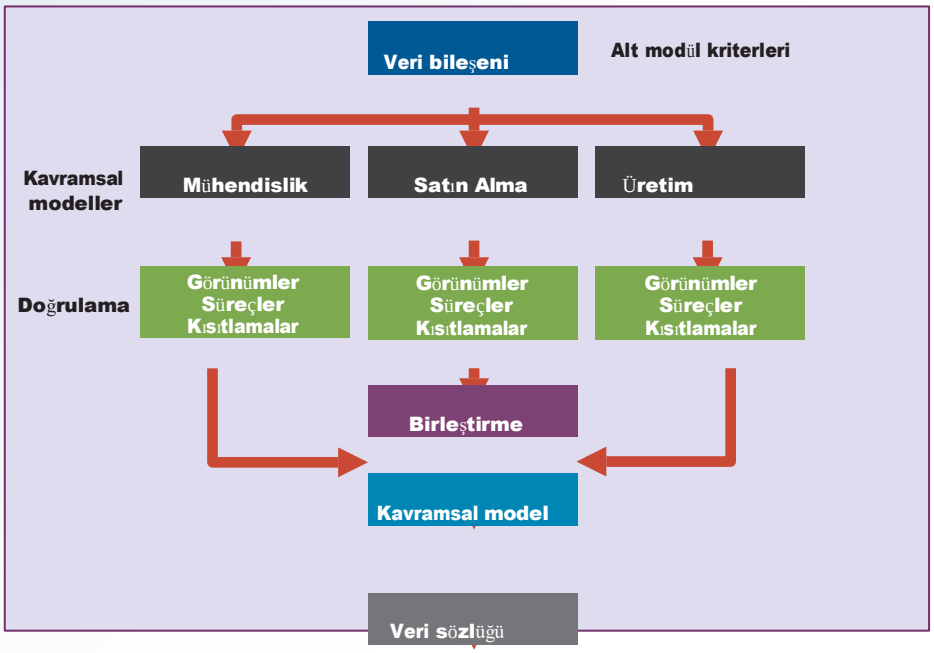
Şekil 9.15 Merkezi Tasarım



Merkezi olmayan tasarım
Kavramsal tasarım modellerinin kullanıldığı bir süreç Bir kuruluşun veritabanı gereksinimlerinin alt kümeleri, daha sonra eksiksiz bir tasarımda toplanır. Bu tür modüler tasarımlar, nispeten çok sayıda nesne ve prosedür içeren karmaşık sistemler tipiktir.

Merkezi olmayan tasarım, sistemin veri bileşeninin çok sayıda varlığa ve üzerinde çok karmaşık işlemlerin gerçekleştirildiği karmaşık ilişkilere sahip olduğu durumlarda . Merkezi olmayan tasarım, problemin kendisi birkaç operasyonel bölgeye yayıldığında ve her bir unsur tüm veri setinin bir alt kümesi olduğunda da sıklıkla kullanılır. (Bkz. Şekil 9.16.)

Şekil 9.16 Merkezi Olmayan Tasarım



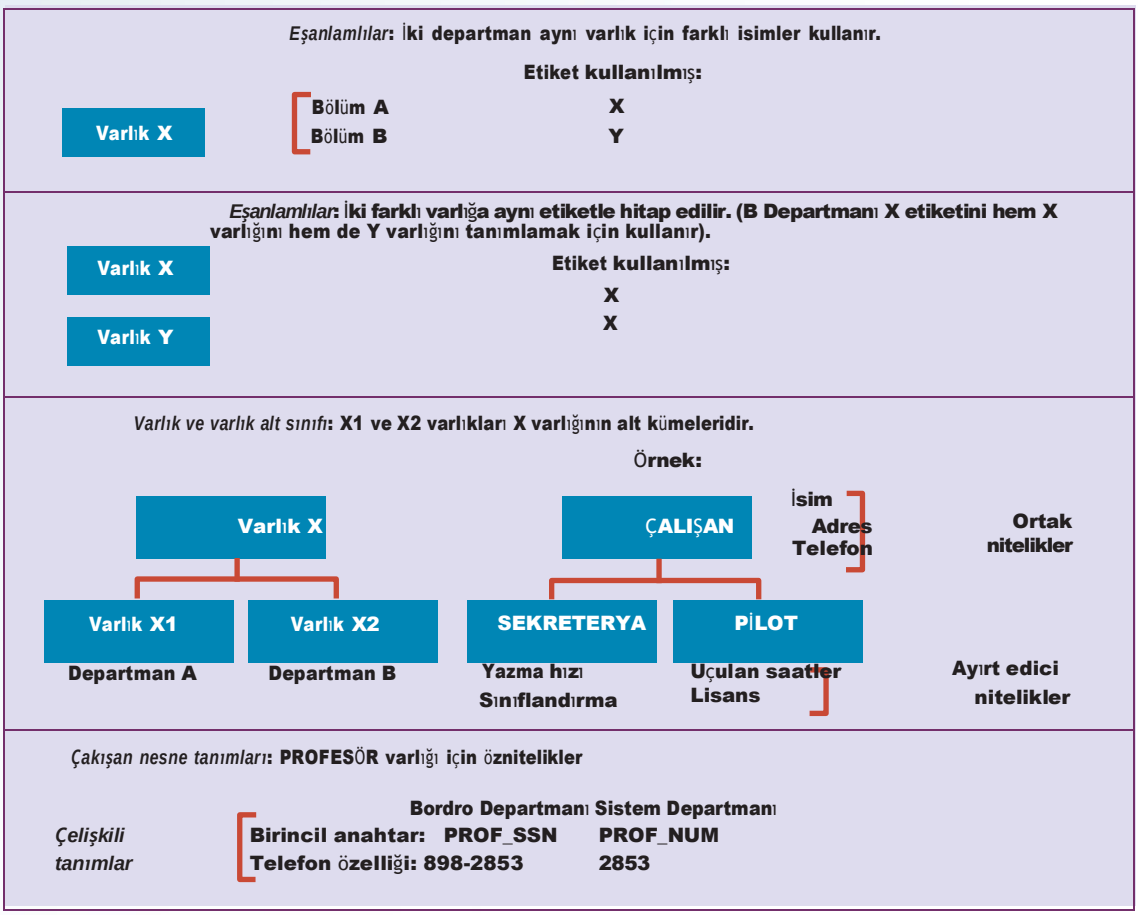
Büyük ve karmaşık projelerde, veritabanı genellikle tek bir kişi tarafından tasarlanamaz. Bunun yerine, özenle seçilmiş bir veritabanı tasarımcıları ekibi karmaşık bir veritabanı projesini ele alır. Merkezi olmayan tasarım çerçevesinde, veritabanı tasarım görevi birkaç modüle . Tasarım kriterleri belirlendikten sonra baş tasarımcı, tasarım alt kümelerini veya modüllerini ekip içindeki tasarım gruplarına atar.

Her tasarım grubu sistemin bir alt kümesini modellemeye odaklandığından, sınırların tanımı ve veri alt kümeleri arasındaki ilişkiler çok hassas olmalıdır. Her tasarım grubu, modellenen alt kümeye karşılık gelen bir kavramsal veri modeli oluşturur. Her bir kavramsal model daha sonra her modül için kullanıcı görüşleri, süreçler ve kısıtlamalarla ayrı ayrı doğrulanır. Doğrulama süreci tamamlandıktan sonra, tüm modüller tek bir kavramsal modele entegre edilir. Veri sözlüğü, kavramsal veri modelindeki tüm nesnelerin özelliklerini tanımladığından, entegrasyon sürecinde hayati bir rol oynar. Alt kümeler daha büyük bir kavramsal modelde toplandıktan sonra, baş tasarımcı bu modelin hala gerekli tüm işlemleri destekleyebildiğini doğrulamalıdır.

Toplama işleminin tasarımcının çeşitli toplama sorunlarının ele alınması gereken tek bir model oluşturmasını gerektirdiğini unutmayın. (Bkz. Şekil 9.17.)

- *Eşanlamlılar ve eşsesliler*. Çeşitli departmanlar aynı nesneyi farklı isimlerle tanıyabilir (eşanlamlılar) veya farklı nesnelere hitap etmek için aynı ismi kullanabilirler (eşsesliler). Nesne bir varlık, bir öznitelik veya bir ilişki olabilir.
- *Varlık ve varlık alt türleri*. Bir varlık alt türü, bir veya daha fazla departman tarafından ayrı bir varlık olarak görülebilir. Tasarımcı bu tür alt tipleri daha üst düzey bir varlığa entegre etmelidir.
- *Çelişen nesne tanımları*. Öznitelikler farklı türler (karakter, sayısal) olarak kaydedilebilir veya aynı öznitelik için farklı etki alanları tanımlanabilir. Kısıtlama tanımları da farklılık gösterebilir. Tasarımcı bu tür çakışmaları modelden kaldırmalıdır.

Şekil 9.17 Birleştirme Sorunlarının Özeti



Özet

- Bir bilgi sistemi, verilerin bilgiye dönüştürülmesine yardımcı olmak ve hem verileri hem de bilgileri yönetmek için tasarlanmıştır. Bu nedenle, veri tabanı bilgi sisteminin çok önemli bir parçasıdır. Sistem analizi, bir bilgi sistemi ihtiyacını ve kapsamını belirleyen süreçtir. Sistem geliştirme, bir bilgi sistemi oluşturma sürecidir.
- Sistem Geliştirme Yaşam Döngüsü (SDLC), bilgi sistemi içindeki bir uygulamanın geçmişini izler. SDLC beş aşamaya ayrılabilir: planlama, analiz, detaylı sistem tasarımı, uygulama ve bakım. SDLC, sıralı bir süreçten ziyade yinelemeli bir süreçtir.
- Veritabanı Yaşam Döngüsü (DBLC), bilgi sistemi içindeki veritabanının hikayesini tanımlar. DBLC altı aşamadan oluşur: veritabanı başlangıç çalışması, veritabanı tasarımı, uygulama ve yükleme, test ve değerlendirme, işletme ve bakım ve geliştirme. SDLC gibi DBLC de sıralı olmaktan ziyade yinelemelidir.
- Tasarımın kavramsal kısmı, iki temel tasarım felsefesine dayalı olarak çeşitli varyasyonlara tabi olabilir: aşağıdan yukarıya karşı yukarıdan aşağıya ve merkezileştirilmişe karşı merkezi olmayan.

Anahtar Terimler

aşağıdan yukarıya
tasarım sınırları
merkezi tasarım
kümelenmiş tablo
uyumluluğu
bilgisayar destekli yazılım mühendisliği (CASE)
kavramsal tasarım
veritabanı geliştirme
veritabanı parçası
Veritabanı Yaşam Döngüsü (DBLC) veritabanı rolü
merkezi olmayan tasarım
operasyonların tanımı
diferansiyel yedekleme
tam yedekleme
bilgi sistemi (IS)
mantıksal tasarımı
minimal veri kuralı
modülü
modül bağlanti
fiziksel tasarim
kapsami
sistem analizi sistem
geliştirme
Sistem Geliştirme Yaşam Döngüsü (SDLC)
yukarıdan aşağıya
tasarım işlem günlüğü
yedekleme sanallaştırma

İnceleme Soruları

1. Bilgi sistemi nedir? Amacı nedir?
2. Sistem analizi ve sistem geliştirme, bilgi sistemleri hakkındaki bir tartışmaya nasıl dahil edilir?
3. SDLC kısaltması ne anlama gelir ve bir SDLC neyi tasvir eder?
4. DBLC kısaltması ne anlama geliyor ve bir DBLC neyi tasvir ediyor?
5. Merkezi ve merkezi olmayan kavramsal veritabanı tasarımı arasındaki ayrımı tartışınız.

6. Kavramsal tasarımda minimal veri kuralı nedir? Neden önemlidir?
7. Veritabanı tasarımında yukarıdan aşağıya ve aşağıdan yukarıya yaklaşımlar arasındaki ayrımı tartışınız.
8. İş kuralları nedir? Bir veritabanı tasarımcısı için neden önemlidirler?
9. Veri tabanı tasarımında veri sözlüğünün işlevi nedir?
10. Bir ER diyagramının geliştirilmesinde hangi adımlar gereklidir? (*İpucu:* Bkz. Tablo 9.3.)
11. Bir ER modelinin doğrulanmasında yer alan faaliyetleri listeleyiniz ve kısaca açıklayınız.
12. Bir DBMS yazılımı seçiminde hangi faktörler önemlidir?
13. Mantıksal tasarım aşamasında gerçekleştirilen dört adımı listeleyiniz ve kısaca açıklayınız.
14. Fiziksel tasarım aşamasında gerçekleştirilen üç adımı listeleyiniz ve kısaca açıklayınız.
15. Veritabanı kurtarma yönetiminde hangi üç yedekleme seviyesi kullanılabilir? Her bir yedekleme seviyesinin ne yaptığını kısaca açıklayınız.

Problemler

1. ABC Araba Servis ve Onarım Merkezleri Sessiz Araba Bayiliğine aittir; ABC sadece sessiz arabalara servis ve onarım hizmeti vermektedir. Üç ABC merkezi tüm eyalete servis ve onarım hizmeti vermektedir.

   Üç merkezin her biri bir mağaza müdürü, bir resepsiyon görevlisi ve en az sekiz tamirci tarafından bağımsız olarak yönetilmekte ve işletilmektedir. Her merkezde tam stoklu bir parça envanteri bulunmaktadır.

   Her merkez ayrıca her aracın bakım geçmişinin tutulduğu manuel bir dosya sistemi tutmaktadır; yapılan onarımlar, kullanılan parçalar, maliyetler, servis tarihleri, sahibi vb. Envanter, satın alma, faturalama, çalışanların çalışma saatleri ve maaş bordrolarını takip etmek için de dosyalar tutulmaktadır.

   Bilgisayarlı bir veritabanı sistemi tasarlamak ve uygulamak üzere merkezin yöneticilerinden biri sizinle temasa geçti. Önceki bilgileri göz önünde bulundurarak aşağıdakileri yapın:

   a. Aşağıdaki adımların her birini doğru sırada etiketleyerek en uygun faaliyet sırasını belirtin. (örneğin, "Veritabanını yükle" adımının uygun ilk adım olduğunu düşünüyorsanız, bu adımı "1" olarak etiketleyin)

      _____Kavramsal modeli normalleştirin.

      _____Şirket faaliyetlerinin genel bir tanımını elde edin.

      _____Veritabanını yükleyin.

      _____Her bir sistem süreci için bir açıklama oluşturun.

      _____Sistemi test edin.

      _____Bir veri akış diyagramı ve sistem akış şemaları çizin.

      _____ER diyagramlarını kullanarak kavramsal bir model oluşturun.

      _____Uygulama programlarını oluşturun.

      _____Teknisyenlerle görüşün.

      _____Dosya (tablo) yapılarını oluşturun.

      _____Mağaza müdürüyle görüşün.

b. Sistemin içermesi gerektiğine inandığınız çeşitli modülleri açıklayın.

c. Bir veri sözlüğü sistemi geliştirmenize nasıl yardımcı olacak? Örnekler verin.

d. Mağaza müdürüne ne gibi genel (sistem) önerilerde bulunabilirsiniz? Örneğin, sistem entegre edilecekse hangi modüller entegre edilecek? Böyle bir entegre sistemden ne gibi faydalar elde edilebilir? Birkaç genel öneri ekleyin.

e. Kavramsal veritabanı tasarımı için en iyi yaklaşım nedir? Neden?

f. Sistemin sahip olması gereken en az dört raporu adlandırın ve açıklayın. Kullanımlarını açıklayın. Raporları kim kullanacak?

2. Sizden birçok şekil, boyut ve işlevde somun ve cıvata üreten bir üretim tesisi için bir bilgi sistemi oluşturmanız istendiğini varsayalım. Hangi soruları sorardınız ve cevaplar veri tabanı tasarımını nasıl etkilerdi?

   a. SDLC'nin nasıl olmasını öngörüyorsunuz?

   b. DBLC'nin ne olmasını öngörüyorsunuz?

3. Problem 2'de belirtilen işlevlerin aynısını daha büyük bir depolama operasyonu için gerçekleştirdiğinizi varsayalım. İki prosedür seti ne kadar benzerdir? Nasıl ve neden farklıdırlar?

4. Problem 1'de kullanılan aynı prosedür ve kavramları kullanarak, Bölüm 4'teki Tiny College örneği için nasıl bir bilgi sistemi oluştursunuz?

5. Bir video kiralama veritabanının tasarımı için uygun faaliyet sırasını yazın. (İlk ERD Şekil 9.9'da gösterilmiştir.) Tasarım, tüm kiralama faaliyetlerini, müşteri ödeme takibini ve çalışan çalışma programlarını desteklemeli ve hangi çalışanların videoları müşterilere teslim ettiğini takip etmelidir. Tasarım faaliyeti sırasını yazmayı bitirdikten sonra, veritabanı tasarımının başarıyla uygulanabileceğinden emin olmak için ERD'yi tamamlayın. (Tasarımın düzgün bir şekilde normalleştirildiğinden ve gerekli işlemleri destekleyebildiğinden emin olun).

6. Bir inşaat şirketinde, yeni bir sistem birkaç aydır yürürlüktedir ve şimdi yapılması gereken olası değişikliklerin/güncellemelerin bir listesi vardır. Her bir değişiklik/güncelleme için ne tür bir bakım yapılması gerektiğini belirtin: (a) düzeltici, (b) uyarlayıcı veya (c) mükemmelleştirici.

   a. Alanlardan birinin boyutunda bir hata tespit edildi ve güncellenmesi gerekiyor; durum alanının değiştirilmesi gerekiyor.

   b. Şirket, bu yeni hizmeti desteklemek ve mevcut verilerle entegre etmek için sistemi yeni bir tablo setiyle geliştirmeyi gerektirecek yeni bir hizmet türüne doğru genişliyor.

   c. Şirketin bazı devlet düzenlemelerine uyması gerekiyor. Bunu yapmak için mevcut sistem tablolarına birkaç alan eklenmesi gerekecektir.