**Veritabanı Yaklaşımı**

Dosya tabanlı sistemin kullanılmasından kaynaklanan zorluklar, veritabanı yaklaşımı olarak adlandırılan büyük miktarda kurumsal bilginin yönetiminde yeni bir yaklaşımın geliştirilmesine yol açmıştır.

Veritabanları ve veritabanı teknolojisi, iş, eğitim ve tıp dahil olmak üzere bilgisayarların kullanıldığı çoğu alanda önemli bir rol oynamaktadır. Veritabanı sistemlerinin temellerini anlamak için bu alandaki bazı temel kavramları tanıtarak başlayacağız.

**İş dünyasında veritabanlarının rolü**

Sadece arkadaşları ve aileleri hakkında bilgi depolamak için olsa bile herkes bir şekilde bir veritabanı kullanır. Bu veriler, bir kelime işlemci programı kullanılarak bir bilgisayarda yazılabilir veya saklanabilir veya bir elektronik tabloya kaydedilebilir. Ancak, verileri depolamanın en iyi yolu, veritabanı yönetim yazılımı kullanmaktır. Bu, verileri çeşitli şekillerde saklamanıza, işlemenize ve almanıza olanak tanıyan güçlü bir yazılım aracıdır.

**verinin anlamı**

Veriler, nesneler ve kavramlar hakkında ölçümler veya istatistikler gibi olgusal bilgilerdir. Verileri tartışmalar için veya bir hesaplamanın parçası olarak kullanırız. Veri bir kişi, bir yer, bir olay, bir eylem veya bir dizi şeyden herhangi biri olabilir. Tek bir olgu, bir veri öğesi veya bir veri öğesidir.

Veriler bilgiyse ve çalıştığımız iş bilgiyse, bunları nerede sakladığınızı görmeye başlayabilirsiniz. Veriler şurada saklanabilir:

Dosya dolapları

E-tablolar

klasörler

defterler

Listeler

Masanızda kağıt yığınları

Tüm bu öğeler bilgi depolar ve bir veritabanı da öyle. Veritabanlarının mekanik doğası gereği, sahip oldukları bilgileri yönetmek ve işlemek için müthiş bir güce sahiptirler. Bu, barındırdıkları bilgileri işiniz için çok daha yararlı hale getirebilir.

**Veritabanı Nedir?**

Veritabanı, belirli bir organizasyonun faaliyetlerini desteklemek için kullanılan ilgili verilerin ortak bir koleksiyonudur. Bir veritabanı, bir kez tanımlanan ve daha sonra erişilen bir veri deposu olarak görülebilir.

**veritabanı aşağıdaki özelliklere sahiptir:**

Gerçek dünyanın bazı yönlerinin bir temsili veya gerçek dünya bilgilerini temsil eden bir veri öğeleri (olgular) koleksiyonudur.

Bir veritabanı mantıklı, tutarlı ve kendi içinde tutarlıdır.

Bir veritabanı, belirli bir amaç için verilerle tasarlanır, oluşturulur ve doldurulur.

Her veri öğesi bir alanda veya bir sütunda depolanır.

Alanların birleşimi bir tablo oluşturur. Örneğin, bir çalışan tablosundaki her alan, tek bir çalışanla ilgili verileri içerir.

Bir veritabanı birçok tablo içerebilir. Örneğin, bir üyelik sistemi, bir adres tablosu ve bireysel üye tablosu içerebilir.

**Bir Veritabanının Özellikleri ve Faydaları**

Veritabanı yaklaşımını dosya tabanlı sistem veya yaklaşımdan ayıran bir takım özellikler vardır. Bu sayfa, veritabanı sisteminin yararlarını (ve özelliklerini) açıklar.

**Bir veritabanı sisteminin kendi kendini tanımlayan doğası**

Bir veritabanı sistemi, yalnızca veritabanının kendisini değil, aynı zamanda veritabanındaki verileri ve tablolar arasındaki ilişkileri tanımlayan ve tanımlayan meta verileri de içerdiğinden, kendi kendini tanımlayan olarak adlandırılır.

**Program ve veri arasındaki yalıtım**

Dosya tabanlı sistemde, veri dosyalarının yapısı uygulama programlarında tanımlanır, bu nedenle bir kullanıcı bir dosyanın yapısını değiştirmek isterse, o dosyaya erişen tüm programların da değiştirilmesi gerekebilir.

Öte yandan, veritabanı yaklaşımında, veri yapısı programlarda değil sistem kataloğunda saklanır.

**Birden çok veri görünümü desteği**

Bir veritabanı, birden çok veri görünümünü destekler. Görünüm, sistemin belirli kullanıcıları için tanımlanmış ve ayrılmış veritabanının bir alt kümesidir.

**Veri paylaşımı ve çok kullanıcılı sistem**

Mevcut veritabanı sistemleri birden fazla kullanıcı için tasarlanmıştır. Yani birçok kullanıcının aynı anda aynı veritabanına erişmesine izin verirler. Bu erişim, eşzamanlılık kontrol stratejileri adı verilen özellikler aracılığıyla sağlanır.

**Veri fazlalığının kontrolü**

Veritabanı yaklaşımında ideal olarak, her bir veri öğesi veritabanında yalnızca bir yerde saklanır. Bazı durumlarda, sistem performansını iyileştirmek için veri fazlalığı hala mevcuttur, ancak bu fazlalık uygulama programlaması tarafından kontrol edilir ve veritabanı tasarlanırken mümkün olduğunca az fazlalık getirilerek minimumda tutulur.

**Bilgi paylaşımı**

Bir kuruluş için tüm verilerin bir veritabanı sistemi içinde entegrasyonunun birçok avantajı vardır. İlk olarak, çalışanlar ve sisteme erişimi olan diğerleri arasında veri paylaşımına izin verir. İkincisi, kullanıcılara belirli bir miktarda veriden entegrasyon olmadan mümkün olandan daha fazla bilgi üretme yeteneği verir.

**Bütünlük kısıtlamalarının uygulanması**

Veritabanı yönetim sistemleri, kullanıcıların geçerli bilgiler girmesini ve veri bütünlüğünü korumasını sağlamak için belirli kısıtlamaları tanımlama ve uygulama yeteneği sağlamalıdır. Veritabanı kısıtlaması, belirli bir biçimi kullanan posta kodu veya Şehir alanına geçerli bir şehir ekleme gibi bir tabloya nelerin girilebileceğini veya düzenlenebileceğini belirten bir kısıtlama veya kuraldır.

**Yetkisiz erişimin kısıtlanması**

Bir veritabanı sisteminin tüm kullanıcıları aynı erişim ayrıcalıklarına sahip olmayacaktır. Örneğin, bir kullanıcı salt okunur erişime sahip olabilir (yani, bir dosyayı okuma ancak değişiklik yapmama yeteneği), diğer bir kullanıcı ise bir dosyayı hem okuma hem de değiştirme yeteneği olan okuma ve yazma ayrıcalıklarına sahip olabilir.

**Veri bağımsızlığı**

Bir veritabanı yönetim sisteminin bir diğer avantajı, veri bağımsızlığına nasıl izin verdiğidir. Başka bir deyişle, sistem veri açıklamaları veya verileri tanımlayan veriler (meta veriler) uygulama programlarından ayrılır.

**Hareket işleme**

Bir veritabanı yönetim sistemi, eşzamanlılık denetimi alt sistemlerini içermelidir. Bu özellik, birkaç kullanıcı aynı bilgiyi güncellese bile, işlem sırasında verilerin tutarlı ve geçerli kalmasını sağlar.

**Birden çok veri görünümü için provizyon**

Bir DBMS, doğası gereği, birçok kullanıcının veri tabanına tek tek veya aynı anda erişmesine izin verir. Kullanıcıların eriştikleri verilerin nasıl ve nerede saklandığının farkında olmaları önemli değildir.

**Yedekleme ve kurtarma tesisleri**

Yedekleme ve kurtarma, verilerinizi kaybolmaktan korumanızı sağlayan yöntemlerdir. Veritabanı sistemi, verileri yedeklemek ve kurtarmak için ağ yedeklemesinden ayrı bir işlem sağlar.

**Veritabanı Yönetim Sistemi Nedir?**

Bir veritabanı yönetim sistemi (DBMS), kullanıcıların veritabanları oluşturmasını ve bakımını yapmasını ve bunlara tüm erişimi kontrol etmesini sağlayan bir programlar topluluğudur. Bir VTYS'nin birincil amacı, kullanıcıların bilgileri alması ve saklaması için hem uygun hem de verimli bir ortam sağlamaktır.

Veritabanı yaklaşımı ile aşağıda gösterildiği gibi geleneksel bankacılık sistemine sahip olabiliriz. Bu banka örneğinde, paylaşılan kurumsal veritabanına erişmek için Personel Departmanı, Hesap Departmanı ve Kredi Departmanı tarafından bir VTYS kullanılmaktadır.

Bir DBMS kullanarak, topladığımız ve veritabanına eklediğimiz bilgiler artık kazara düzensizliğe tabi değildir. Daha erişilebilir ve çalışmalarımızın geri kalanıyla entegre hale geliyor. Bir veritabanı kullanarak bilgileri yönetmek, sahip olduğumuz verilerin stratejik kullanıcıları olmamızı sağlar.

Bir veritabanının işlem gücü, barındırdığı verileri manipüle etmesine olanak tanır, böylece şunları yapabilir:

Sort

Match

Link

Aggregate

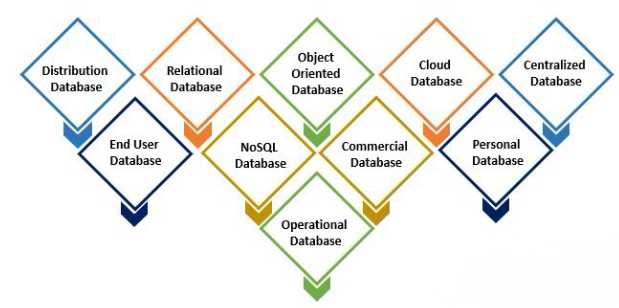
Skip fields

Calculate

Arrange

DBMS Sınıflandırması

Veritabanı yönetim sistemleri, tümü aşağıda açıklanan veri modeli, kullanıcı numaraları ve veritabanı dağılımı gibi çeşitli kriterlere göre sınıflandırılabilir.



Günümüzde kullanılan en popüler veri modeli ilişkisel veri modelidir. Oracle, MS SQL Server, SQLite ve MySQL gibi iyi bilinen İlişkisel Veritabanı Yönetim Sistemleri (RDBMS) bu modeli destekler.

Öte yandan NoSQL (Yalnızca SQL veya ilişkisel olmayan) veri modeli var. MongoDB gibi iyi bilinen NoSQL veritabanları,

Veritabanı sistemleri için iki ana dağıtım sistemi vardır ve bunlar sırayla DBMS'yi sınıflandırmak için kullanılabilir.

**Merkezi sistemler**

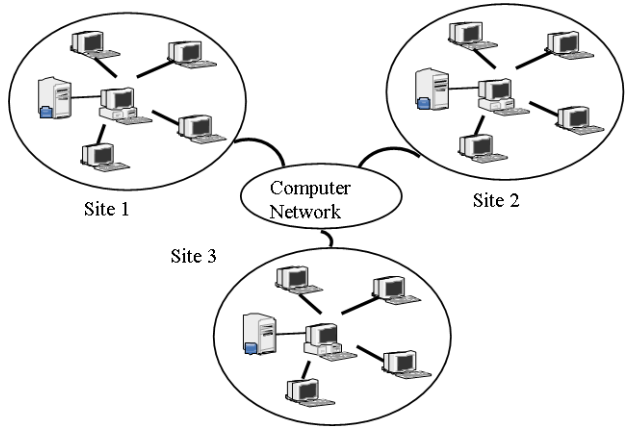
Merkezi bir veritabanı sistemi ile, DBMS ve veritabanı, diğer birçok sistem tarafından da kullanılan tek bir sitede depolanır. Bu, aşağıdaki şekil ile gösterilmiştir.

mikroskop içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Dağıtılmış veritabanı sistemi**

Dağıtılmış bir veritabanı sisteminde, gerçek veritabanı ve VTYS yazılımı, aşağıda gösterildiği gibi bir bilgisayar ağı ile bağlanan çeşitli sitelerden dağıtılır.



Başlangıçta SQL olmayan veya ilişkisel olmayan anlamına gelen bir NoSQL, verilerin depolanması ve alınması için bir mekanizma sağlayan bir veritabanıdır. Bu veriler, ilişkisel veritabanlarında kullanılan tablo ilişkileri dışındaki yollarla modellenir. NoSQL veritabanları gerçek zamanlı web uygulamalarında ve büyük veride kullanılmakta ve zaman içinde kullanımları artmaktadır. NoSQL sistemleri bazen SQL benzeri sorgu dillerini destekleyebileceklerini vurgulamak için Sadece SQL Değil olarak da adlandırılır.

Bir NoSQL veritabanı tasarım kolaylığı, makine kümelerine daha basit yatay ölçekleme ve kullanılabilirlik üzerinde daha hassas kontrol içerir. NoSQL veritabanları tarafından kullanılan veri yapıları, NoSQL'de bazı işlemleri daha hızlı hale getiren ilişkisel veritabanlarında varsayılan olarak kullanılanlardan farklıdır.

NoSQL Veritabanı ve İlişkisel Veritabanı:

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Veritabanı Kullanıcıları**

Son kullanıcılar, işleri sorgulamak, güncellemek ve rapor oluşturmak için bir veritabanına erişim gerektiren kişilerdir.

Uygulama kullanıcıları, günlük görevleri gerçekleştirmek için mevcut bir uygulama programına erişen kişilerdir.

Gelişmiş kullanıcılar, veritabanına erişmek için kendi yöntemlerine sahip olanlardır. Bu, sistemde sağlanan uygulama programını kullanmadıkları anlamına gelir. Bunun yerine, kendi uygulamalarını tanımlayabilir veya ihtiyaçlarını doğrudan sorgu dillerini kullanarak tanımlayabilirler.

Bu uzman kullanıcılar, MS Access gibi kullanımı kolay menü yönlendirmeli komutlar sağlayan hazır program paketleri kullanarak kişisel veritabanlarını korurlar.

Bu kullanıcılar, saklanan verilere erişmek için özel uygulama programları uygular. Görevlerini yerine getirmek için DBMS'lere aşina olmaları gerekir.

Bu, veritabanına erişim yetkisi vermekten, veritabanının kullanımını izlemekten ve tüm veritabanı sisteminin kullanımını desteklemek için tüm kaynakları yönetmekten sorumlu bir kuruluştaki bir kişi veya bir grup kişi olabilir.

**Anahtar terimler**

**concurrency control strategies**: eşzamanlılık kontrol stratejileri: birkaç kullanıcının aynı anda aynı veri öğesine erişmesine izin veren bir veritabanının özellikleri

**data elements** :veri öğeleri: gerçek dünya bilgilerini temsil eden gerçekler

**data type** :veri türü: bir alanda izin verilen veri türünü belirler, örneğin yalnızca sayılar

**data uniqueness**: veri benzersizliği: hiçbir kopya girilmemesini sağlar

**database**: veritabanı: belirli bir organizasyonun faaliyetlerini desteklemek için kullanılan ilgili verilerin ortak bir koleksiyonu.

**database constraint**: veritabanı kısıtlaması: bir tabloda neyin girilmesine veya neyin düzenlenmesine izin verildiğini belirleyen bir kısıtlama

**database management system (DBMS)**: veritabanı yönetim sistemi (DBMS): Kullanıcıların veritabanları oluşturmasını ve bakımını yapmasını ve bunlara tüm erişimi kontrol etmesini sağlayan bir programlar topluluğu

**metadata**: meta veriler: veritabanındaki tablolar arasındaki verileri ve ilişkileri tanımlar ve açıklar

**read and write privileges**: okuma ve yazma ayrıcalıkları: bir dosyayı hem okuma hem de değiştirme yeteneği

**read-only access**: salt okunur erişim: bir dosyayı okuma ancak değişiklik yapmama yeteneği

**self-describing**: Kendi kendini tanımlayan: Bir veritabanı sistemi, yalnızca veritabanının kendisini değil, aynı zamanda veritabanındaki verileri ve tablolar arasındaki ilişkileri tanımlayan ve tanımlayan meta verileri de içerdiği için kendi kendini tanımlayan olarak adlandırılır.

**table**: tablo: alanların bir kombinasyonu

**view**: görünüm: veritabanının bir alt kümesi

**concurrency**: eşzamanlılık: veritabanının, işlem işlemeyi olumsuz etkilemeden aynı kayda birden fazla kullanıcının erişimine izin verme yeteneği

**data element**: veri öğesi: tek bir gerçek veya bilgi parçası

**data inconsistency:** veri tutarsızlığı: aynı verinin çeşitli kopyalarının çakıştığı bir durum

**data isolation**: veri yalıtımı: bir işlem tarafından yapılan değişikliklerin diğer eşzamanlı kullanıcılar ve sistemler tarafından ne zaman ve nasıl görünür hale geleceğini belirleyen bir özellik

**data integrity**: veri bütünlüğü: bir veritabanındaki verilerin doğru ve tutarlı olduğunun bakım ve güvencesini ifade eder.

**data redundancy**: veri fazlalığı: bir alanın birden fazla tabloda güncellenmesi gerektiğinde bir veritabanında oluşan bir durum

**database approach**: veritabanı yaklaşımı: büyük miktarda kurumsal bilginin yönetimine izin verir

**database management software**: veritabanı yönetim yazılımı: verileri çeşitli şekillerde saklamanıza, işlemenize ve almanıza olanak tanıyan güçlü bir yazılım aracı

**file-based system**: dosya tabanlı sistem: veri dosyalarını işlemek için tasarlanmış bir uygulama programı

**İlişkisel Modelin Temel Kavramları**

Bu konumuzda, dersimizin odak noktası olan İlişkisel Veri Modelini tanıyacağız.

Bir RDBMS, sunduğu faydaları belirli koşullar altında sunar. İhtiyaçlarınızı iyi belirlemeli, verilerinizi iyi bilmeli ve verilerinizi ihtiyaçlarınıza göre modellemelisiniz.

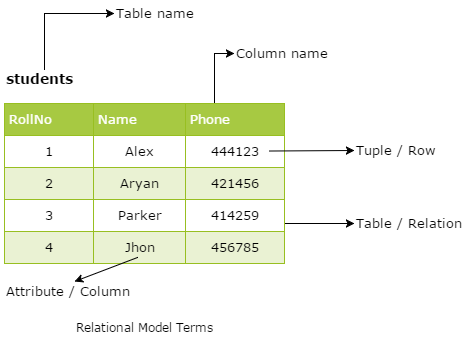
Verimli modelleme için, İlişkisel Veritabanı kavramlarına aşina olmalısınız. Anormalliklerden ve fazlalıklardan korunmak için gerekli argümanları bilmelisiniz. Tablolarınızı olması gerektiği gibi düzenlemeli ve ilişkileri doğru tanımlayıp oluşturmalısınız.

Tüm bunları başarmak için, verilerin depolandığı sütunlar veya tablolar üzerinde bazı kısıtlamalara ihtiyacınız var. Bu kısıtlamalar bazen doğal koşullardan kaynaklanabilir, bazen de iş, şirket, veri veya tercihlerinizden kaynaklanan kısıtlamalar olabilir.

Elbette, İlişkisel modelin kısıtlamalar, birincil anahtar ve yabancı anahtar gibi temel yapı taşlarını bilmek, özelliklerini bilmek ve bunları veri modelinize uygulamak, ilişkisel veri modelinin ön koşullarıdır.

**İlişkisel Veri Modeli Terimleri**

İlişkisel veri modeli 1970 yılında C. F. Codd tarafından bulunmuştur. Günümüzde en yaygın veri modellerinden biri olduğu söylenmektedir.



İlişkisel model aşağıdakilere yol açmıştır:

Veri teorisi/ilişki/kısıtlama

Veritabanı tasarım yaklaşımları

Yapılandırılmış sorgu dili (SQL) olarak adlandırılan standart veritabanı erişim dili

Modern veritabanı yönetim sistemleri

İlişkisel veri modeli, evreni “birbiriyle ilişkili ilişkilerin (veya tabloların) bir bileşimi” olarak tanımlar.

**1. İlişki**

Tablo veya dosya olarak da bilinen bir ilişki, bir adla temsil edilen bir etki alanları listesinin Kartezyen ürününün bir alt kümesidir. Ve bir tablo içinde, her satır bir grup ilgili veri değerini temsil eder. Bir satır veya kayıt, bir demet olarak da bilinir. Tablodaki sütun bir alandır ve öznitelik olarak da adlandırılır. Başka bir deyişle, öznitelikler kayıtları tanımlamak için kullanılır ve her kaydın bir dizi özniteliği vardır.

**2. Tablo**

Bir veritabanı birden çok tablo içerir ve her tablo verileri depolar.

**Tablo Özellikleri:**

Bir tablonun benzersiz bir adı vardır.

Yinelenen satır yok.

Hücreler ve nitelikler atomik olmalıdır.

Sütunlardaki girişler aynı etki alanındandır ve aşağıdaki gibi aynı veri türüne sahip olmalıdırlar:

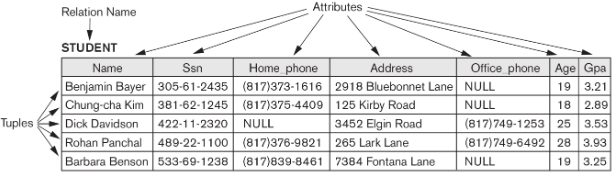
sayı (sayısal, tamsayı, float, smallint,…)

character string)

date

logical (doğru veya yanlış)

Her özelliğin ayrı bir adı vardır.



**3. Sütun**

İlişkisel bir veritabanı, bir etki alanına ait farklı bilgileri organize bir şekilde depolar. Veritabanını etkin bir şekilde kullanabilmek ve yönetebilmek, bu organizasyon yöntemini anlamayı gerektirir.

Temel depolama birimlerine sütunlar veya alanlar veya nitelikler denir. Bunlar, içeriğinizin bölünebileceği temel veri bileşenlerini barındırır. Hangi niteliklerin gerekli olduğuna karar verirken, verilerin sahip olduğu bilgileri düşünmeniz gerekir. Örneğin, veritabanında saklayacağınız bilgilerin ortak bileşenlerini çıkarmak ve bir öğeyi diğerinden ayıran özelliklerden kaçınmak.

**4. Etki alanı**

Etki alanı, bir sütunun sahip olabileceği nitelikleri taşıyan orijinal atomik değerleri ifade eder. Bir atomik değer artık bölünemez ve kendi başına bir anlamı olan bir yapıyı ifade eder. Örneğin:

Cinsiyet alanının bir dizi olasılığı vardır: Erkek, Kadın

Ayın etki alanı, olası tüm ayların kümesine sahiptir: {Jan, Feb, Mar…}.

Age alanı, 0'dan büyük ve 150'den küçük tüm kayan noktalı sayıların kümesidir.

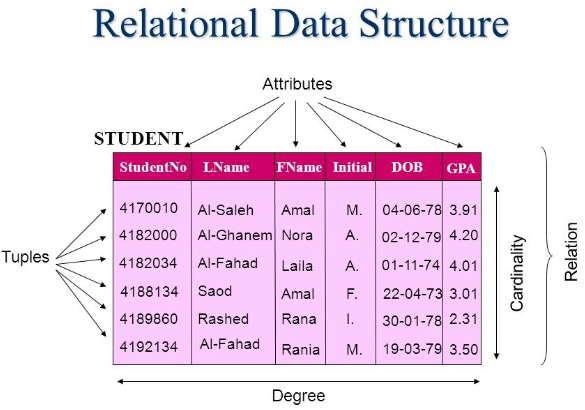
**5. İlişki**

İlişkiler, atomik yapıda bilgi taşıyan tabloların birbirine yapışmasını sağlayan köprülerdir.

**6. Kayıt**

Kayıtlar, müşteri veya ürün gibi ilgili alanları içerir. Daha önce belirtildiği gibi, bir demet, kayıt için kullanılan başka bir terimdir.

Kayıtlar ve alanlar, tüm veritabanlarının temelini oluşturur. Basit bir tablo, bir veritabanı depolama projesinde kayıtların ve alanların birlikte nasıl çalıştığının en net resmini verir.



tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

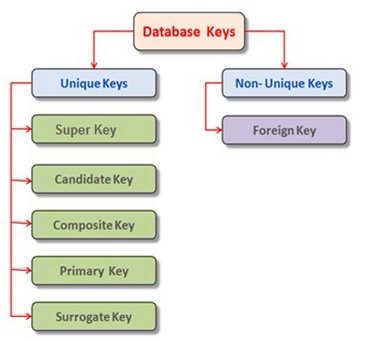
tablo içeren bir resim

Açıklama otomatik olarak oluşturuldutablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Anahtar Kısıtlamalar**

Aday anahtar, Bileşik anahtar, Birincil anahtar (PK), İkincil anahtar, Alternatif anahtar ve Yabancı anahtar (FK) gibi çeşitli anahtar türleri vardır.



**Candidate key**

Candidate key, benzersiz ve minimal olan basit veya bileşik bir anahtardır. Benzersizdir, çünkü bir tablodaki hiçbir iki satır aynı anda aynı değere sahip olamaz. Minimumdur, çünkü benzersizliği elde etmek için her sütun gereklidir.

**Composite key**

Composite key, iki veya daha fazla öznitelikten oluşur, ancak minimum düzeyde olmalıdır.

**Primary key**

Primary key (PK), tüm varlık kümesi için bir tanımlama mekanizması olarak kullanılmak üzere veritabanı tasarımcısı tarafından seçilen bir aday anahtardır. Bir tablodaki demetleri benzersiz bir şekilde tanımlamalı ve boş olmamalıdır. Primary key, özniteliğin altı çizilerek ER modelinde belirtilir.

Primary key, bir tablodaki her satırı benzersiz şekilde tanımlayan bir sütun veya sütun grubudur. PRIMARY KEY kısıtlamasını kullanarak bir tablo için birincil anahtar oluşturursunuz.

**💡İpuçları:**

Birincil anahtarın iki veya daha fazla sütunu olması durumunda, tablo kısıtlaması olarak PRIMARY KEY kısıtlamasını kullanmanız gerekir.

Her tablo yalnızca bir birincil anahtar içerebilir.

SQL Server, tüm birincil anahtar sütunları için NOT NULL kısıtlamasını otomatik olarak ayarlar.

SQL Server ayrıca bir birincil anahtar oluşturduğunuzda otomatik olarak benzersiz bir kümelenmiş dizin (veya bu şekilde belirtilmişse kümelenmemiş bir dizin) oluşturur.

**Secondary key**

Secondary key, kesinlikle alma amacıyla kullanılan bir niteliktir (bileşik olabilir).

**Alternate key**

Alternatif anahtarlar, birincil anahtar olarak seçilmeyen tüm aday anahtarlardır.

**Foreign key**

Foreign key (FK), başka bir tablodaki birincil anahtara başvuran bir tablodaki bir niteliktir. FK boş olabilir. Hem yabancı hem de birincil anahtarlar aynı veri türünde olmalıdır.

FOREIGN KEY (FK) kısıtlaması, değerleri başka bir tablonun PRIMARY KEY (PK) ile eşleşen bir sütunu veya sütunların birleşimini tanımlar.

Bir FK'deki değerler, ilişkili tablodaki PK değerleri güncellendiğinde/değiştirildiğinde otomatik olarak güncellenir.

FK kısıtlamaları, başka bir tablonun PK veya UNIQUE kısıtlamasına başvurmalıdır.

FK için sütun sayısı, PK veya UNIQUE kısıtlaması ile aynı olmalıdır.

WITH NOCHECK seçeneği kullanılırsa, FK kısıtlaması bir tablodaki mevcut verileri doğrulamaz.

Bir FK kısıtlamasına katılan sütunlarda dizin oluşturulmaz.

Bu örnekte, çalışan tablosundaki alan kimliği, departmanlar tablosundaki alan kimliğinin bir FK'sidir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**İlişkisel Bütünlük Kısıtlamaları**

Kısıtlamalar, ilişkisel bir modelde çok önemli bir özelliktir. Aslında, ilişkisel model, nitelikler veya tablolar üzerindeki iyi tanımlanmış kısıtlamalar teorisini destekler.

Kısıtlamalar, bir tasarımcının veritabanındaki verilerin anlamını belirlemesine izin verdiği için kullanışlıdır. Kısıtlamalar, DBMS'leri verilerin anlambilimi karşılayıp karşılamadığını kontrol etmeye zorlayan kurallardır.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Etki Alanı Bütünlüğü**

Etki alanı, ilişkideki niteliklerin değerlerini kısıtlar ve ilişkisel modelin bir kısıtlamasıdır. Ancak, yalnızca etki alanı kısıtlamalarıyla kullanıldığında belirlenemeyen veriler için gerçek dünya semantikleri vardır.

Hangi veri değerlerine izin verilip verilmediğini ve bir öznitelik için hangi biçimin uygun olduğunu belirtmek için daha spesifik yollara ihtiyacımız var. Örneğin, Çalışan Kimliği (EID) benzersiz olmalıdır veya çalışan Doğum Tarihi [1 Ocak 1950, 1 Ocak 2000] aralığındadır. Bu tür bilgiler, bütünlük kısıtlamaları adı verilen mantıksal ifadelerde sağlanır.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Bilgi tutarlılığı**

Bilgi bütünlüğü, bir yabancı anahtarın eşleşen bir birincil anahtara sahip olmasını veya boş olmasını gerektirir. Bu kısıtlama iki tablo (ebeveyn ve çocuk) arasında belirtilir; bu tablolardaki satırlar arasındaki yazışmaları korur. Bu, bir tablodaki bir satırdan başka bir tabloya yapılan referansın geçerli olması gerektiği anlamına gelir.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Varlık Bütünlüğü**

Varlık bütünlüğünü sağlamak için her tablonun bir birincil anahtarı olması gerekir. Ne PK ne de herhangi bir parçası boş değerler içeremez.

Bunun nedeni, birincil anahtar için boş değerlerin bazı satırları tanımlayamayacağımız anlamına gelmesidir. Örneğin, ÇALIŞAN tablosunda, bazı kişilerin telefonu olmayabilir, çünkü Telefon birincil anahtar olamaz.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Kurumsal Kısıtlamalar**

Kurumsal kısıtlamalar – bazen anlamsal kısıtlamalar olarak da adlandırılırlar – kullanıcılar veya veritabanı yöneticileri tarafından belirtilen ek kurallardır ve birden çok tabloya dayanabilir.

**İşte bazı örnekler.**

Bir sınıfta en fazla 30 öğrenci olabilir.

Bir öğretmen bir yarıyılda en fazla dört ders verebilir.

Bir çalışan beşten fazla projede yer alamaz.

Bir çalışanın maaşı, çalışanın yöneticisinin maaşını aşamaz.

**İş kuralları**

Gereksinimler toplanırken kullanıcılardan iş kuralları alınır. Gereksinim toplama süreci çok önemlidir ve sonuçları, veritabanı tasarımı oluşturulmadan önce kullanıcı tarafından doğrulanmalıdır. İş kuralları yanlışsa tasarım yanlış olur ve sonuçta oluşturulan uygulama kullanıcılar tarafından beklendiği gibi çalışmaz.

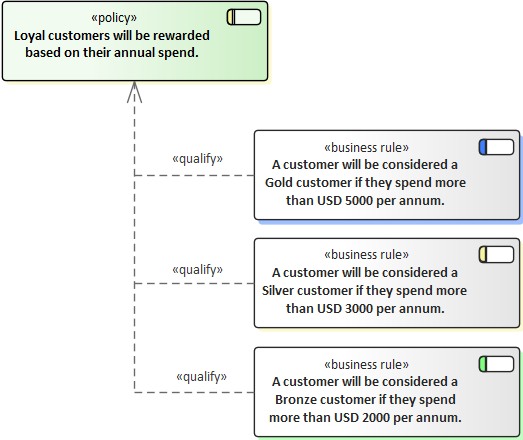
**İş kurallarına bazı örnekler:**

Bir öğretmen birçok öğrenciye ders verebilir.

Bir sınıfta en fazla 35 öğrenci olabilir.

Bir ders birçok kez verilebilir, ancak yalnızca bir eğitmen tarafından.

Her öğretmen ders vermiyor.



**Tablo veya Sütun Kısıtlamaları**

İsteğe Bağlı Sütun Kısıtlamaları, yeni bir kayıt için bir değer başlatmak için kullanılan NULL, NOT NULL, UNIQUE, PRIMARY KEY ve DEFAULT'tur. NULL sütun kısıtlaması, boş değerlere izin verildiğini belirtir; bu, bu sütun için bir değer olmadan bir satır oluşturulabileceği anlamına gelir. NOT NULL sütun kısıtlaması, yeni bir satır oluşturulduğunda bir değerin sağlanması gerektiğini belirtir.

Örneklemek için, 7 öznitelik veya alan içeren departmanlar tablosunu oluşturmak için CREATE TABLE Sections SQL deyimini kullanacağız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İlk alan, BIGINT alan tipine sahip id'dir. Kullanıcı bu alanı boş bırakamaz (NOT NULL).

Benzer şekilde, ikinci alan, 20 uzunluğunda bir VARCHAR alan tipine sahip addır.

Tüm tablo sütunları tanımlandıktan sonra, birincil anahtarı oluşturmak için CONSTRAINT kelimesiyle tanımlanan bir tablo kısıtlaması kullanılır:

CONSTRAINT pk\_1 PRIMARY KEY(id)

Bu sütun için benzersiz, artımlı bir değer sağlamak için isteğe bağlı IDENTITY sütun kısıtlamasını kullanabiliriz. Kimlik sütunları, tablo için benzersiz satır tanımlayıcısı olarak hizmet etmek üzere genellikle PRIMARY KEY kısıtlamalarıyla birlikte kullanılır. IDENTITY özelliği, tinyint, smallint, int, decimal veya sayısal veri tipine sahip bir sütuna atanabilir. Bu kısıtlama:

Sıralı sayılar üretir.

Varlık bütünlüğünü zorlamaz.

IDENTITY özelliğine yalnızca bir sütun sahip olabilir.

Tamsayı, sayısal veya ondalık veri türü olarak tanımlanmalıdır.

IDENTITY özelliğiyle bir sütun güncellenemez.

NULL değerler içeremez.

Varsayılanlar ve varsayılan kısıtlamalar sütuna bağlanamaz.

IDENTITY[(seed, increment)]

Seed – kimlik sütununun ilk değeri

Increment – son artış sütununa eklenecek değer

UNIQUE kısıtlaması, yinelenen değerlerin bir sütuna girilmesini önler.

Varlık bütünlüğünü zorlamak için hem PK hem de UNIQUE kısıtlamaları kullanılır.

Bir tablo için birden çok UNIQUE kısıtlaması tanımlanabilir.

Mevcut bir tabloya UNIQUE kısıtlaması eklendiğinde, mevcut veriler her zaman doğrulanır.

Boş değerleri kabul eden sütunlara UNIQUE bir kısıtlama yerleştirilebilir. Yalnızca bir satır NULL olabilir.

UNIQUE bir kısıtlama, seçilen sütunda otomatik olarak benzersiz bir dizin oluşturur.

Bu, UNIQUE kısıtlamasını kullanan bir örnektir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

CHECK kısıtlaması, bir tabloya girilebilecek değerleri kısıtlar.

WHERE yan tümcesine benzer arama koşulları içerebilir.

Aynı tablodaki sütunlara başvurabilir.

Bir CHECK kısıtlaması için veri doğrulama kuralı, bir boole ifadesi olarak değerlendirilmelidir.

Kendisine bağlı bir kuralı olan bir sütun için tanımlanabilir.

Bu, CHECK kısıtlamasının genel sözdizimidir:



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

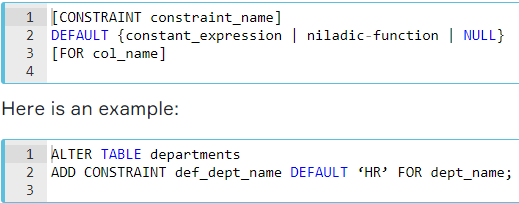
DEFAULT kısıtlaması, kullanıcının bir sütun sağlamaması durumunda bir sütun için otomatik olarak eklenen bir değer sağlamak için kullanılır.

Bir sütunda yalnızca bir DEFAULT olabilir.

DEFAULT kısıtlaması, zaman damgası veri türü veya kimlik özelliği olan sütunlarda kullanılamaz.

DEFAULT kısıtlamalar, oluşturulduklarında otomatik olarak bir sütuna bağlanır.

DEFAULT kısıtlamasının genel sözdizimi şöyledir:



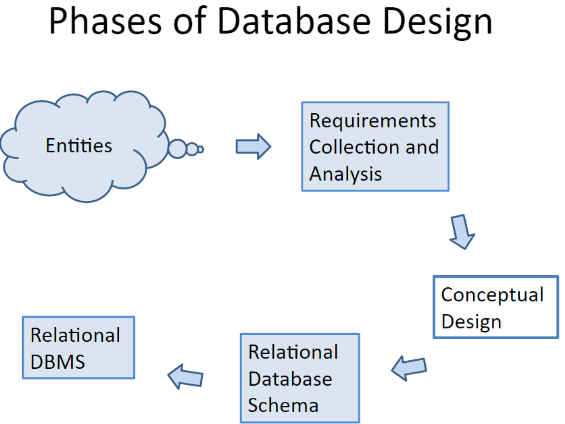
**Veritabanı Tasarımının Aşamaları**

Veri modelleme, veritabanı tasarımı sürecindeki ilk adımdır. Bu adım bazen kavramsal tasarım olarak da adlandırılan üst düzey ve soyut bir tasarım aşaması olarak kabul edilir. Bu aşamanın amacı aşağıdakileri tanımlamaktır:

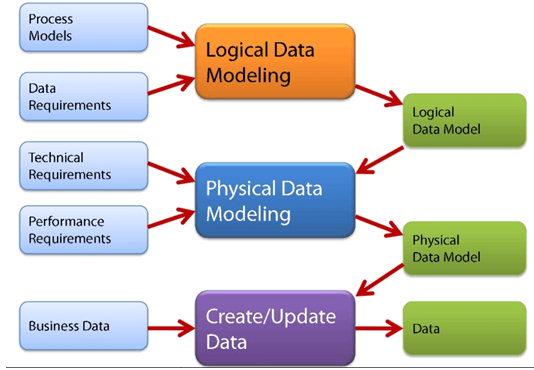
Veritabanında bulunan veriler (ör. varlıklar: öğrenciler, öğretim görevlileri, kurslar, konular)

Veri öğeleri arasındaki ilişkiler (örneğin, öğrenciler öğretim görevlileri tarafından denetlenir; öğretim görevlileri ders verir)

Veri kısıtlamaları (örneğin, öğrenci numarası tam olarak sekiz basamaklıdır; bir konunun yalnızca dört veya altı kredisi vardır)



İkinci adımda, veri öğeleri, ilişkiler ve kısıtlamaların tümü, üst düzey veri modeli tarafından sağlanan kavramlar kullanılarak ifade edilir. Bu kavramlar uygulama detaylarını içermediğinden, veri modelleme sürecinin sonucu, veritabanı yapısının (yarı) resmi bir temsilidir. Bu sonucun anlaşılması oldukça kolaydır, bu nedenle tüm kullanıcı gereksinimlerinin karşılandığından emin olmak için referans olarak kullanılır.



Üçüncü adım, veritabanı tasarımıdır. Bu adım sırasında, iki alt adımımız olabilir: biri, belirli bir VTYS'nin veri modelinde bir veritabanını tanımlayan veritabanı mantıksal tasarımı ve dahili veritabanı depolama yapısını, dosya organizasyonunu veya indekslemeyi tanımlayan veritabanı fiziksel tasarımı olarak adlandırılan bir adımdır. teknikler. Bu iki alt adım, veritabanı uygulaması ve işlemler/kullanıcı arayüzleri oluşturma adımlarıdır.

Veritabanı tasarım aşamalarında, veriler belirli bir veri modeli kullanılarak temsil edilir. Veri modeli, verileri, veri ilişkilerini, veri anlambilimini ve veri kısıtlamalarını açıklamak için bir kavramlar veya gösterimler topluluğudur. Çoğu veri modeli, veritabanındaki verileri işlemek için bir dizi temel işlemi de içerir.

**Veritabanı Geliştirme Yönergeleri**

Aşağıdakiler, gerçek veritabanı tasarımı (mantıksal model) için güçlü bir temel geliştirmeye yardımcı olacak genel yönergelerdir.

**💡İpuçları:**

Bilgi toplama aşamasında keşfedilen tüm varlıkları belgeleyin.

Her bir varlığa ait tüm nitelikleri belgeleyin. Aday ve birincil anahtarları seçin. Her varlık için anahtar olmayan tüm özniteliklerin tam işlevsel olarak birincil anahtara bağlı olduğundan emin olun.

Bir başlangıç ER diyagramı geliştirin ve uygun personelle gözden geçirin. (Bunun yinelemeli bir süreç olduğunu unutmayın.)

Çok değerli nitelikler ve yinelenen gruplar için yeni varlıklar (tablolar) oluşturun. Bu yeni varlıkları (tabloları) ER diyagramına dahil edin. Uygun personel ile gözden geçirin.

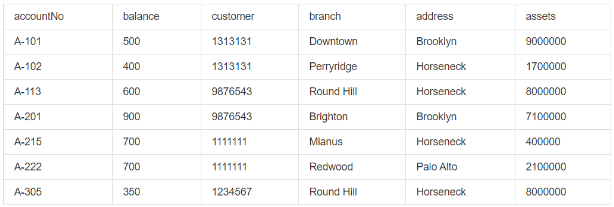
Tabloları normalleştirerek ER modellemesini doğrulayın.

**Neden Normalleşmeye ihtiyacımız var?**

Genel olarak, iyi bir ilişkisel veritabanı tasarımı, gerekli tüm öznitelikleri ve ilişkileri kapsamalıdır. Tasarım bunu minimum miktarda depolanmış bilgi ile ve fazla veri olmadan yapmalıdır.

Veritabanı tasarımında artıklık genellikle istenmeyen bir durumdur çünkü güncellemelerden sonra tutarlılığın korunmasında sorunlara neden olur. Ancak artıklık bazen performans iyileştirmelerine yol açabilir; örneğin, verileri bağlamak için bir birleştirme yerine artıklık kullanılabildiğinde. Birleştirme, ilgili iki tabloya dayalı bilgi almanız gerektiğinde kullanılır.

Aşağıdaki rakamı göz önünde bulundurun: müşteri 1313131, bir kez hesap no için olmak üzere iki kez görüntülenir. A-101 ve tekrar A-102 hesabı için. Bu durumda, tabloda silme anormallikleri olmasına rağmen müşteri numarası gereksiz değildir. Ayrı bir müşteri tablosuna sahip olmak bu sorunu çözecektir. Ancak, bir şube adresi değişecekse, birden çok yerde güncellenmesi gerekir. Müşteri numarası tabloda olduğu gibi bırakılsaydı, şube tablosuna ihtiyacınız olmayacaktı ve herhangi bir birleştirme gerekmeyecek ve performans artırıldı.



**Anomalies**

**1. Ekleme Anomalisi:**

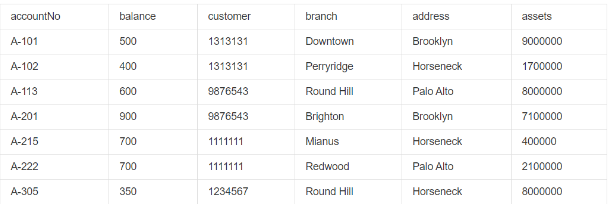
Bir tabloya tutarsız bilgiler eklerken bir ekleme anomalisi oluşur. Hesap no gibi yeni bir kayıt eklediğimizde. Aşağıdaki şekilde A-306, şube verilerinin mevcut satırlarla tutarlı olup olmadığını kontrol etmemiz gerekiyor.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

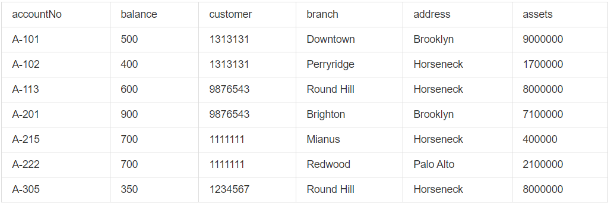
**2. Anomaliyi Güncelle:**

Aşağıdaki şekildeki Round Hill şubesi gibi bir şube adres değiştirirse, o şubeye atıfta bulunan tüm satırları güncellememiz gerekir. Mevcut bilgilerin yanlış değiştirilmesine güncelleme anomalisi denir.



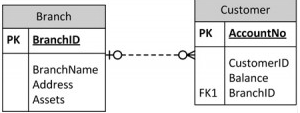
**3. Silme Anomalisi:**

Silinmemesi gereken öznitelikler içerebilecek bir kaydı sildiğinizde, bir silme anormalliği oluşur. Örneğin, aşağıdaki şekilde Downtown şubesindeki A-101 hesabı gibi bir şubedeki son hesapla ilgili bilgileri kaldırırsak, tüm şube bilgileri kaybolur.



A-101 satırını silmeyle ilgili sorun, Downtown şubesinin nerede olduğunu bilmememiz ve müşteri 1313131 ile ilgili tüm bilgileri kaybetmemizdir. Bu tür güncelleme veya silme sorunlarından kaçınmak için orijinal tabloyu birkaç küçük parçaya ayırmamız gerekir. her tablonun diğer tablolarla minimum düzeyde örtüştüğü tablolar.

Her banka hesabı tablosu, aşağıdaki şekilde gösterildiği gibi Şube veya Müşteri gibi yalnızca bir tüzel kişi hakkında bilgi içermelidir.



Bu uygulamaya uyulması, şube bilgileri eklendiğinde veya güncellendiğinde sadece bir kaydı etkilemesini sağlayacaktır. Böylece müşteri bilgileri eklendiğinde veya silindiğinde şube bilgileri yanlışlıkla değiştirilmeyecek veya hatalı kaydedilmeyecektir.

**Anomaliler nasıl önlenir?**

Anormallikler olmadan tablolar oluşturmaya yönelik en iyi yaklaşım, tabloların normalleştirilmesini sağlamaktır ve bu, işlevsel bağımlılıkları anlayarak gerçekleştirilir. FD, bir tablodaki tüm niteliklerin o tabloya ait olmasını sağlar. Başka bir deyişle, fazlalıkları ve anormallikleri ortadan kaldıracaktır.

**Örnek**: ayrı Proje ve Çalışan tabloları

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ayrı Proje ve Çalışan tablolarını kullanarak verileri ayrı tutarak:

Bir bütçe değiştirilirse hiçbir anormallik oluşturulmaz.

Çalışan atanmamış projeler için kukla değerlere gerek yoktur.

Bir çalışanın katkısı silinirse, hiçbir önemli veri kaybolmaz.

Bir çalışanın katkısı eklenirse hiçbir anormallik oluşturulmaz.

**Fonksiyonel Bağımlılık**

Varlık ilişkisel (ER) modeli için geliştirilen önemli bir teori, işlevsel bağımlılık (FD) kavramını içerir. Bunu çalışmanın amacı, veriler arasındaki ilişkileri anlamanızı geliştirmek ve pratik veritabanı tasarımına yardımcı olmak için yeterli formalizm kazanmaktır.

Kısıtlamalar gibi, FD'ler de uygulama alanının semantiğinden alınır. Esasen, işlevsel bağımlılıklar, bireysel özelliklerin nasıl ilişkili olduğunu tanımlar. FD'ler, bir ilişki içindeki nitelikler arasında bir tür kısıtlamadır ve iyi bir ilişkisel şema tasarımına katkıda bulunur. Bu bölümde şunlara bakacağız:

Fonksiyonel bağımlılığın temel teorisi ve tanımı

Normalleştirme olarak da adlandırılan şema tasarımlarını iyileştirme metodolojisi

İşlevsel bağımlılık (FD), iki öznitelik arasındaki, tipik olarak PK ile bir tablodaki diğer anahtar olmayan öznitelikler arasındaki bir ilişkidir. Herhangi bir R ilişkisi için, Y özniteliği, X özniteliğine (genellikle PK) işlevsel olarak bağlıdır, eğer X'in her geçerli örneği için, X'in bu değeri Y'nin değerini benzersiz bir şekilde belirlerse. Bu ilişki aşağıdaki temsil ile gösterilir:

**X ———–>Y**

Yukarıdaki FD diyagramının sol tarafına determinant, sağ tarafına ise bağımlı denir. İşte birkaç örnek.

Aşağıdaki ilk örnekte, SIN Adı, Adresi ve Doğum Tarihini belirler. SIN verildiğinde, tablodaki diğer özelliklerden herhangi birini belirleyebiliriz.

**SIN ———-> Name, Address, Birthdate**

İkinci örnek için, SIN ve Course, tamamlanan tarihi (DateCompleted) belirler. Bu aynı zamanda bileşik bir PK için de çalışmalıdır.

**SIN, Course ———> DateCompleted**

Üçüncü örnek, Başlığı ISBN'nin belirlediğini gösterir.

**ISBN ———–> Title**

**Normalizasyon Örneği**

Aşağıdaki şekil bir çalışan proje tablosu örneğini göstermektedir. Bu tablodan şunu varsayabiliriz:

EmpID ve ProjectID, bileşik bir PK'dir.

Proje Kimliği Bütçeyi belirler (yani, Proje P1'in bütçesi 32 saattir).

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi, aşağıdaki adımlarda bu tabloda oluşabilecek bazı olası anormalliklere bakalım.

Action: {S85, 35, P1, 9} satırını ekleyin

Problem: Çakışan bütçelere sahip iki grup var

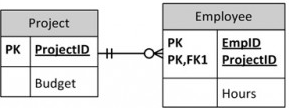
Action: {S79, 27, P3, 1} demetini sil

Problem: Adım #3, P3 projesinin bütçesini siler

Action: {S75, 32, P1, 7} tanımlama grubunu {S75, 35, P1, 7} olarak güncelleyin

Problem: Adım #5, proje P1'in bütçesi için farklı değerlere sahip iki demet oluşturur

Solution: Projeler ve Çalışanlar için aşağıda gösterildiği gibi ayrı bir tablo oluşturun.



**Normalization**

Normalleştirme, veritabanı tasarım sürecinin bir parçası olmalıdır. Ancak, normalizasyon sürecini ER modelleme sürecinden ayırmak zordur, bu nedenle iki teknik aynı anda kullanılmalıdır.

Bir kuruluşun veri gereksinimlerinin ve operasyonlarının büyük resmini veya makro görünümünü sağlamak için bir varlık ilişki diyagramı (ERD) kullanın. Bu, ilgili varlıkların, niteliklerinin ve ilişkilerinin tanımlanmasını içeren yinelemeli bir süreçle oluşturulur.

Normalleştirme prosedürü, belirli varlıkların özelliklerine odaklanır ve ERD içindeki varlıkların mikro görünümünü temsil eder.

**Normalleştirme Nedir?**

Normalleştirme, tasarım anlayışları sağlayan ilişkisel teorinin dalıdır. Bir tabloda ne kadar fazlalık olduğunu belirleme işlemidir. Normalleşmenin amaçları şunlardır:

İlişkisel bir şemada artıklık düzeyini karakterize edebilme

Fazlalığı ortadan kaldırmak için şemaları dönüştürmek için mekanizmalar sağlayın

Normalleştirme teorisi, ağırlıklı olarak fonksiyonel bağımlılıklar teorisine dayanır. Normalizasyon teorisi, altı normal form (NF) tanımlar. Her normal form, bir şemanın karşılaması gereken bir dizi bağımlılık özelliği içerir ve her normal form, güncelleme anormalliklerinin varlığı ve/veya yokluğu hakkında garanti verir. Bu, daha yüksek normal formların daha az fazlalığa sahip olduğu ve sonuç olarak daha az güncelleme sorunu olduğu anlamına gelir.

**Normal Formlar**

Herhangi bir veritabanındaki tüm tablolar, daha sonra tartışacağımız normal formlardan birinde olabilir. İdeal olarak, yalnızca PK'den FK'ye minimum fazlalık istiyoruz. Diğer her şey diğer tablolardan türetilmelidir. Altı normal form vardır, ancak biz sadece ilk dördüne bakacağız, bunlar:

İlk normal form (1NF)

İkinci normal form (2NF)

Üçüncü normal form (3NF)

Boyce-Codd normal formu (BCNF)

BCNF nadiren kullanılır.

**İlk Normal Form (1NF)**

İlk normal biçimde, her satır ve sütunun kesişiminde yalnızca tek değerlere izin verilir; bu nedenle, tekrar eden gruplar yoktur.

Yinelenen grup içeren bir ilişkiyi normalleştirmek için yinelenen grubu kaldırın ve iki yeni ilişki oluşturun.

Yeni ilişkinin PK'si, orijinal ilişkinin PK'si ile benzersiz tanımlama için yeni oluşturulan ilişkiden bir özniteliğin birleşimidir.

**1NF için süreç**

1NF sürecini açıklamak için örnek olarak bir Okul veritabanından aşağıdaki Student\_Grade\_Report tablosunu kullanacağız.

**Öğrenci\_Notu\_Raporu** (ÖğrenciNo, ÖğrenciAdı, Ana Dal, DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu, Not)

Öğrenci Not Raporu tablosunda tekrar eden grup ders bilgileridir. Bir öğrenci birçok ders alabilir.

Yinelenen grubu kaldırın. Bu durumda, her öğrenci için ders bilgisidir.

Yeni tablonuz için PK'yi tanımlayın.

PK, nitelik değerini (StudentNo ve CourseNo) benzersiz bir şekilde tanımlamalıdır.

Ders ve öğrenci ile ilgili tüm özellikleri kaldırdıktan sonra geriye öğrenci ders tablosu (StudentCourse) kalır.

Öğrenci tablosu (Öğrenci) artık tekrar eden grup kaldırılmış olarak ilk normal biçimindedir.

İki yeni tablo aşağıda gösterilmiştir.

**Öğrenci** (ÖğrenciNo, ÖğrenciAdı, Binbaşı)

**StudentDers**(ÖğrenciNo, DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmeninKonumu, Notu)

**1NF anomalileri nasıl güncellenir**

**ÖğrenciDers** (ÖğrenciNo, DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu, Not)

Yeni bir ders eklemek için bir öğrenciye ihtiyacımız var.

Kurs bilgilerinin güncellenmesi gerektiğinde tutarsızlıklar yaşayabiliriz.

Bir öğrenciyi silmek için bir kursla ilgili kritik bilgileri de silebiliriz.

**İkinci Normal Form (2NF)**

İkinci normal form için, ilişki ilk önce 1NF'de olmalıdır. İlişki, yalnızca ve yalnızca PK tek bir öznitelik içeriyorsa otomatik olarak 2NF'dedir.

İlişkinin bileşik bir PK'si varsa, o zaman anahtar olmayan her öznitelik, PK'nin bir alt kümesine değil, tüm PK'ye tamamen bağımlı olmalıdır (yani, kısmi bağımlılık veya büyütme olmamalıdır).

**2NF için süreç**

2NF'ye geçmek için bir tablonun önce 1NF'de olması gerekir.

Öğrenci tablosu, tek sütunlu bir PK'ye sahip olduğu için zaten 2NF'dedir.

Öğrenci Kursu tablosunu incelerken, tüm niteliklerin tamamen PK'ya bağlı olmadığını görüyoruz; özellikle, tüm kurs bilgileri. Tamamen bağımlı olan tek nitelik nottur.

Kurs bilgilerini içeren yeni tabloyu tanımlayın.

Yeni tablo için PK'yi tanımlayın.

Üç yeni tablo aşağıda gösterilmiştir.

**Öğrenci** (ÖğrenciNo, ÖğrenciAdı, Binbaşı)

**Ders Notu** (ÖğrenciNo, DersNo, Not)

**Ders Eğitmeni** (DersNo, DersAdı, EğitmenNo, EğitmenAdı, EğitmenKonumu)

**2NF anomalileri nasıl güncellenir**

Yeni bir eğitmen eklerken bir kursa ihtiyacımız var.

Kurs bilgilerinin güncellenmesi, eğitmen bilgileri için tutarsızlıklara yol açabilir.

Bir kursun silinmesi eğitmen bilgilerini de silebilir.

**Üçüncü Normal Form (3NF)**

Üçüncü normal formda olması için, ilişkinin ikinci normal formda olması gerekir. Ayrıca tüm geçişli bağımlılıklar kaldırılmalıdır; anahtar olmayan bir öznitelik, işlevsel olarak başka bir anahtar olmayan özniteliğe bağlı olmayabilir.

**3NF için süreç**

Geçişli bir ilişkiye sahip tabloların her birinden geçişli ilişkilerdeki tüm bağımlı öznitelikleri ortadan kaldırın.

Bağımlılığı kaldırılmış yeni tablo(lar) oluşturun.

Her tablonun bir belirleyicisi olduğundan ve hiçbir tablonun uygunsuz bağımlılıklar içermediğinden emin olmak için değiştirilen tabloların yanı sıra yeni tabloları kontrol edin.

Aşağıdaki dört yeni tabloya bakın.

**Öğrenci** (ÖğrenciNo, ÖğrenciAdı, Binbaşı)

**Ders Notu** (ÖğrenciNo, DersNo, Not)

**Ders** (DersNo, DersAdı, EğitmenNo)

**Eğitmen** (EğitmenNo, EğitmenAdı, EğitmenKonumu)

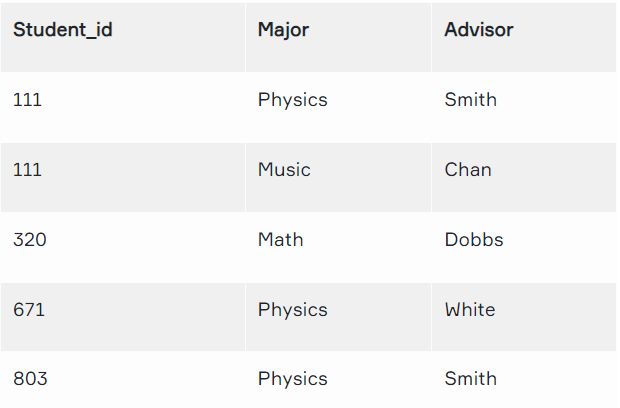
Bu aşamada üçüncü normal formda herhangi bir anormallik olmamalıdır.

**Boyce-Codd Normal Formu (BCNF)**

Bir tablonun birden fazla aday anahtarı olduğunda, ilişki 3NF'de olsa bile anormallikler ortaya çıkabilir. Boyce-Codd normal formu, 3NF'nin özel bir halidir. BCNF'de bir ilişki, ancak ve ancak her belirleyicinin bir aday anahtar olması durumundadır.

**BCNF Örnek 1**

Aşağıdaki tabloyu inceleyin (St\_Maj\_Adv).



Bu tablo için anlamsal kurallar (veritabanına uygulanan iş kuralları):

Her Öğrenci birkaç konuda uzmanlaşabilir.

Her Binbaşı için, belirli bir Öğrencinin yalnızca bir Danışmanı vardır.

Her Binbaşının birkaç Danışmanı vardır.

Her Danışman sadece bir Binbaşı tavsiye eder.

Her Danışman, bir Ana Dalda birkaç Öğrenciye tavsiyede bulunur.

Bu tablo için işlevsel bağımlılıklar aşağıda listelenmiştir. Birincisi bir aday anahtarıdır; ikincisi değil.

Student\_id, Binbaşı ——> Danışman

Danışman ——> Binbaşı

Bu tablo için anomaliler şunları içerir:

Sil – öğrenci danışman bilgilerini siler

Ekle - yeni bir danışmanın bir öğrenciye ihtiyacı var

Güncelleme – tutarsızlıklar

Not: Tek bir öznitelik aday anahtar değildir.

PK Student\_id, Major veya Student\_id, Advisor olabilir.

BCNF ile St\_Maj\_Adv ilişkisini azaltmak için iki yeni tablo oluşturursunuz:

St\_Adv (Student\_id, Danışman)

Adv\_Maj (Danışman, Binbaşı)

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

FD1 – MüşteriNo, MülakatTarihi –> MülakatZamanı, PersonelNo, RoomNo (PK)

FD2 – staffNo, mülakatTarihi, mülakatZamanı –> clientNO (aday anahtarı: CK)

FD3 – odaNo, görüşmeTarihi, görüşmeZamanı –> personelNo, clientNo (CK)

FD4 – personelNo, görüşmeTarihi –> odaNo

BCNF'de bir ilişki, ancak ve ancak her belirleyicinin bir aday anahtar olması durumundadır. Dördüncü FD için ilk üç FD'yi (Client\_Interview2 tablosu) ve başka bir tabloyu (Personel Odası tablosu) içeren bir tablo oluşturmamız gerekiyor.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

Normalizasyon ve Veritabanı Tasarımı

Veritabanı tasarımının normalleştirme işlemi sırasında, tablo yapıları oluşturulmadan önce önerilen varlıkların gerekli normal formu karşıladığından emin olun. Birçok gerçek dünya veri tabanı, uygun olmayan şekilde tasarlanmış veya zaman içinde uygun olmayan şekilde değiştirilirse anormalliklerle yüklenmiştir. Mevcut veritabanlarını yeniden tasarlamanız ve değiştirmeniz istenebilir. Tablolar uygun şekilde normalleştirilmemişse, bu büyük bir girişim olabilir.

**Anahtar Terimler ve Kısaltmalar**

**Boyce-Codd normal form (BCNF)**: Boyce-Codd normal formu (BCNF): 3. NF'nin özel bir durumu

**first normal form (1NF)**: ilk normal form (1NF): her satır ve sütunun kesişiminde yalnızca tek değerlere izin verilir, bu nedenle tekrar eden gruplar yoktur

**normalization**: normalleştirme: bir tabloda ne kadar fazlalık bulunduğunu belirleme süreci

**second normal form (2NF):** ikinci normal form (2NF): ilişki 1NF'de olmalıdır ve PK tek bir öznitelik içerir

**semantic rules: anlamsal kurallar:** veritabanına uygulanan iş kuralları

**third normal form (3NF):** üçüncü normal form (3NF): ilişki 2NF'de olmalı ve tüm geçişli bağımlılıklar kaldırılmalıdır; anahtar olmayan bir öznitelik, işlevsel olarak başka bir anahtar olmayan özniteliğe bağlı olmayabilir