6. NORMALİZASYON

Birlikte Düşünelim

İlişkisel veri tabanı, yalnızca veriyi tablolara bölmek midir?

Tablolara bölünmüş veri bütünleri, nasıl hala aynı bilgiyi taşımaya devam eder?

Bir nitelik, yanlış bir veri tabanı tablosu içerisine yerleştirilirse ne gibi risklerle karşı karşıya kalınır?

Kaçıncı normal form etkin bir veri tabanı için yeterlidir?

Tüm normal formlar uygulanmış bir veri tabanı tasarımı, mükemmel bir veri tabanı tasarımı mıdır?

Normal formları izleyen veri tabanı tasarımcıları her zaman aynı sonuca mı ulaşırlar?

Başlamadan Önce

Veri tabanı tasarımcılığı uzmanlık ve deneyimle doğrudan ilişkilidir. Ancak elbette ki bu uzmanlığı ve deneyimi kazanmak için izlenmesi gereken bir yol vardır. Bunlardan birincisi kuralları öğrenmek, ikincisi ile hatalarla karşılaşarak iyileştirmeler yapmaktır.

Veri tabanı tasarımı hazırlarken, bir ver kümesinin uygun bir veri tabanı tasarımına dönüştürülmesi sürecine normalizasyon adı verilir. Bu süreç aynı zamanda normal form adı verilen kurallar bütününün uygulanmasını da içermektedir. Bir veri tabanı tasarımcısı, tasarım gerçekleştirirken bu kuralları göz önünde bulundurmak zorundadır. Gerçekleştirilen bir veri tabanının tasarımı tüm normal formlara uygun olduğunda etkin bir çözüm olacağı kabul edilmektedir. Ancak elbette kullanım alanına göre ek ihtiyaçlar da ortaya çıkabilir.

Bir veri tabanı tasarımı kullanıma alındığında daha önce düşünülmemiş bazı kullanım şekilleri sebebiyle çeşitli problemler oluşmasına sebep olabilir. Bu da tasarımcılığın deneyimle güçlenmiş tarafının önemini göstermektedir. Zaman içerisinde farklı kullanım şekilleri ve hatalarla karşılaşan tasarımcılar; veri tabanı tasarımcılığı sürecinde normal formları uygulamanın yanı sıra karşılaşılabilecek hataları tahmin eder ve buna göre iyileştirmeler yaparlar. Bazı hatalar henüz meydana gelmeden tahmin edilir ve düzeltilirse “daha iyi” bir veri tabanı tasarımı elde edilmiş olacaktır.

Bu bölümde normalizasyon kavramı ve normal formları inceleyeceğiz.

6.1. Normalizasyon Süreci

Veri tabanı tasarlama süreci kolayca formüle edilebilen, herkes tarafından adım adım ve nesnel olarak gerçekleştirilebilecek bir süreç değildir. Her ne kadar önerildiğinden beri geçen süre içerisinde iyi bir veri tabanı tasarımını tarif eden, tasarım sürecinde dikkat edilmesi gereken adımları sunan ve veri tabanı tasarlamayı öğretmeyi hedefleyen çeşitli çalışma ve yayınlar gerçekleştirilmiş olsa da mükemmel bir sonuç elde edilebilmiş değildir. Konunun uzmanları dahi aynı probleme yaklaşırken farklı çözümler önerebilirler ve işin ilginci bu çözümlerden birini en iyi olarak seçmek mümkün olmayabilir. Bu sebeple veri tabanı tasarlama sürecinde amaç en etkin tasarımı üretmekten öte, hatasız bir tasarım üretmek üzerine kuruludur. Bir veri tabanının hatasız olarak kurgulanmış olması gerçek bir projede kullanılabilmesi için genellikle yeterlidir. Bu durumda veri tabanının kullanımında gelecekte oluşması muhtemel problemlerin öngörülerek bunlar için önlem alınması iyi bir veri tabanı tasarımcısı olmanın gerekliliklerinden biridir.

Dersin bu aşamasına kadar veri tabanı tasarlamanın kesin yargılardan oluşmadığını; deneyim ve uzmanlıklara oldukça bağlı olduğunu vurguladık. Bununla birlikte uzman olsalar dahi farklı tasarımcıların farklı veri tabanı tasarımları sunabileceklerini de söyledik. Tüm bunlar beraberinde şu soruyu getirmekte: İyi

bir veri tabanı tasarımcısı nasıl olunur? Deneyime bağlı bir süreçte “daha iyi” olmak ya da “yeterince iyi” olmak nasıl sağlanabilir? Bunun yanıtı; elde edilen tasarımın uygulama altında ne tür problemlerle karşılaştığı ve düzeltmek için hangi adımların atılması gerektiğini deneyimlemektir. Kağıt ortamında tasarlanan bir veri tabanının, kullanıma alınan bir sistem içerisinde çalıştırılması her zaman beklenildiği gibi olmayacaktır. Hangi noktalarda problem yaşanıldığına dair edinilen deneyim, gelecek veri tabanı tasarımlarında karşılaşılabilecek riskler listesine yeni bir madde eklenmesini ve bunu da gözeterek yeni veri tabanı tasarımının yapılmasını sağlayacaktır. Peki veri tabanı tasarımcılığını yeni öğrenirken ne yapacağız, hangi durumların riskli olduğunu, en azından temel risk ve hataları nasıl öğreneceğiz. İşte bunun yanıtı normalizasyon kavramının içerisindedir.

Veri tabanı tasarlama süreci aynı zamanda normalizasyon olarak da karşımıza çıkar. Normalizasyon, eldeki tüm niteliklerin birbirleriyle ilişkili ve veri tabanı ilkelerine aykırılık oluşturmayacak şekilde varlıkların içerisine yerleştirilme sürecidir (Hoffer, J.A., Ramesh, V & Heikki, T., 2016; Lee, H. 1995). Normalizasyon, iyi bir veri tabanı tasarımı elde edebilmek için uyulması gereken kurallar bütünü olarak da görülebilir. Bu kuralların her birine normal form adı verilmektedir. Literatürde 7 normal form ile karşılaşılsa da günlük hayatta genellikle ilk dördü dikkate alınmakta; ilk dört normal forma uyan tasarımlar normalize edilmiş kabul edilmektedir (Fagin, R., 1981; Hoffer, J.A., Ramesh, V & Heikki, T., 2016).

Normal formlar, veri tabanı tasarımı ve normalizasyonu sürecinde bize hazır bir reçete sunmazlar. Bunun yerine her biri veri tabanı tasarımına aykırılık teşkil eden durumları tanımlar ve ele aldığımız veri tabanı tasarımında bu aykırılığın yer almaması gerektiğini belirtirler. Veri tabanı tasarımını inceleyerek belirtilen riski taşıyıp taşımadığını analiz etmek ve tasarımı bu hatadan kurtarmak yine tasarımcının görevidir. Dolayısıyla kesin bir süreç olmamakla birlikte henüz daha performanslı bir yol da bulunamamıştır. Bu sebeple veri tabanı tasarımcılığı uzmanlık ve deneyim gerektiren bir süreç olarak görülmektedir.

Normal formlar, veri tabanının nasıl olması ve olmaması gerektiği konusunda çeşitli yargılar içeren kurallar bütünüdür. Örneğin birinci normal form, hücrelerin atomik yani tek bir değer almasıyla ilgilidir. Ancak bu normal formun uygulandığının bir testi mümkün değildir. Her bir hücre her seferinde gerçekten atomik kayıt mı alır? Bazı örneklerde veri tabanında adSoyad niteliği tanımlanabilir; ad ve soyad bilgisi bu alana kaydedilebilir. Ancak bu durumda, yalnızca ad niteliğine ihtiyaç duyulursa bu bilgi elde edilemeyecektir. Bu da aslında birinci normal formun uygulanmadığı anlamına gelebilir. Ya da bir telefon numarası kaydedilirken numara ve alan kodu birlikte kayıt altına alınmış olabilir ancak sadece alan koduna ihtiyaç duyulursa bu bilgi elde edilemeyebilir. Bu da birinci normal forma aykırılık içerir. Burada vurgulanan nokta, normal formların yargılarıyla birlikte ilgili veri tabanının kullanım amacının da birlikte değerlendirilmesi ve hatta gelecekte ihtiyaç duyulacak raporlama çeşitlerinin de tasarlama sürecinde ön görülmesi gerekmektedir. Tasarlanan bir veri tabanı tüm normal formlara uygun gözükürken ihtiyaç duyulan yeni bir rapor çeşidi sebebiyle bir anda kötü tasarlanmış bir veri tabanına dönüşebilir.

Bu başlık altında normalizasyon sürecinin bileşenleri olan normal formları inceleyeceğiz. Bu kısım dahilinde her bir normal form örneklerle açıklanacaktır.

6.2. Birinci Normal Form

Birinci normal form, veri yapısı içerisinde her bir içeriğin tek bir (atomik) değer almasını gerektirmektedir (Teorey, T. J., etc., 2011). Tablo ile 1NF’ye uymayan bir örnek sunulmuştur.

ad	soyad	iletisim
Lorem	Ipsum	05991234567
Dolor	Sit	05989876543,dolorsit@mail.com
Amet	Consectetur	consetetur@company.com
Adipiscing	Elit	05973456789

Kişilere ait ad, soyad ve iletisim bilgilerini içeren bir örnek görüyoruz. Burada iletisim niteliği hem telefon numarası hem de e-posta adresi kaydı için kullanılmış. Bu sebeple iki iletişim bilgisi birden bulunan kullanıcılar için hücre içerisinde çift kayıt sunulmuştur. Bu durum 1NF’ye aykırıdır. Çözüm için e-posta ve

telefon numarası için ayrı niteliklerin tanımlanması ve iletişim yerine bu niteliğin kullanılması gerekmektedir.

Kisi: ad, soyad, ePosta, telefon

Bu sayede tasarım, 1NF'ye uygun hale getirilmiştir.

Bazı durumlar için 1NF hala sağlanmamış olabilir mi? Lütfen inceleyiniz.

6.3. İkinci Normal Form

2NF'ye uygunluk için öncelikle 1NF'ye uygunluk gerekli ve ön şarttır. Sonraki normal formlarda da aynı kural geçerlidir. Yani her bir normal form, bir önceki normal forma uyum sağlanması gerektiği kuralıyla başlar.

Tanım gereği 2NF, bir tabloda anahtar olmayan bir niteliğin birincil anahtarın bir alt kümesi olması durumunda ortaya çıkmaktadır (Kent, W., 1983). Bu durumu birden fazla şekilde açıklayabiliriz. Bunlardan birincisi, bir tablo içerisinde birden fazla tabloya ait niteliklerin toplanması 2NF'ye aykırıdır. Bir diğeri, ele alınan tablo içerisinde iki farklı fonksiyonel bağımlılık denklemi kurulabiliyor olmasıdır. Bu durum da yeni bir tablo oluşturulması gerektiğine işarettir. Üçüncü ve son olarak, ele alınan tabloda iki farklı olguyu işaret eden iki farklı birincil anahtar olması da 2NF'ye aykırılık göstergesidir. Anlaşılacağı üzere bu problemin çözümü varlığı gerekli sayıda yeni tabloya bölmektir. Böylece elde edilecek yeni tablolar 2NF'ye uygun olacaktır. Şimdi bu normal forma aykırılığı ve çözümü bir örnek ile ele alalım. Aşağıdaki tablo 2NF'ye uygun olmayan bir örnek içermektedir.

ad	soyad	not	ders	bolum	programTuru
Lorem	Ipsum	85	Veritabanı Tasarımı	YBS	lisans
Dolor	Sit	76	Programlamaya Giriş	YBS	lisans
Amet	Consecte	93	Veritabanı Tasarımı	YBS	lisans

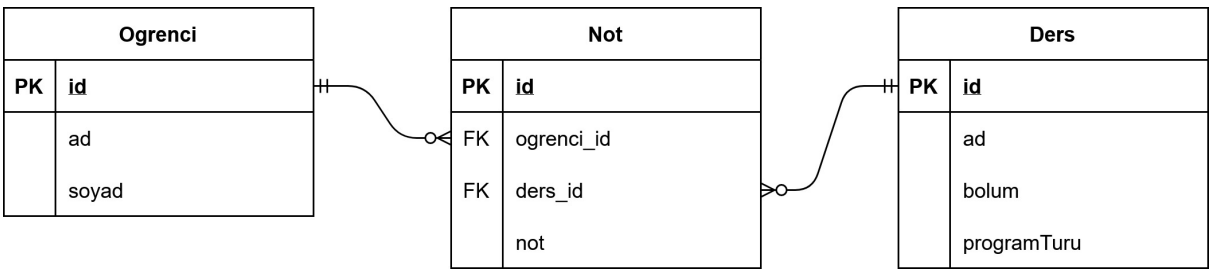
Örnekte öğrencilerin derslerden aldığı notlar yer alıyor. Ayrıca notların hangi derse ait olduğu ve bu dersin bulunduğu bölüm ile bölümün eğitim türü de belirtiliyor. Temelde iki grup bilgi sunduğundan bahsedilebilir. Birincisi öğrenci, ikincisi ise notun alındığı ders. Buradaki not öğrenci ile ilgili, bolum ve programTuru ise ders ile ilgilidir. Dolayısıyla iki bilgi grubundan bahsetmek mümkündür. Çözümlemek için Ogrenci ve Ders adlı iki varlık oluşturmak ve nitelikleri bu gruplara ayırmak gereklidir.

Burada bir alıştırma yapma fırsatı var, kaçırmayalım. Tablo ile verilen örneği bir kez de aşağıda görüyorsunuz.

SinavNotlari: ad, soyad, not, ders, bolum, programTuru

Bu örnek için iki varlık oluşturup nitelikleri bu varlıklara ekledikten sonra varlıkların arasındaki ilişki türünü belirleyiniz.

Bir dersin birçok öğrencisi olması muhtemel olduğu gibi, bir öğrencinin birçok derse ait sınava girmesi de muhtemeldir. Ancak bir öğrenci henüz hiç not almamış olabilir. Benzer şekilde bir dersin de henüz hiç sınavı yapılmamış olabilir. Bu açıklamalar doğrultusunda çözüm Şekil ile verilen gibidir.



Çözüm sayesinde Tablo ile verilen örneğin NF2'ye aykırılığı varlığın iki varlığa bölünmesi sayesinde giderilmiştir. Çözüm önerisindeki üçüncü (Not) varlık, Ogrenci ve Ders varlıklarının çoğa çok bağlantılı olması sebebiyle eklenmiştir. Temelde çözümün iki varlığa bölme işlemiyle elde edildiğini savunabiliriz.

Not: Dikkatinizi çekmiş olabileceği üzere; burada bolum ve programTuru ifadeleri de birer varlık olmayı hak edebilecek içeriğe sahip niteliklerdir. Ancak burada iki varlığa bölünme konu alınmıştır. Eğer örnek için çözümü irdelerken bu konuda takıldıysanız doğru yolda hızla ilerlediğinizi düşünebilirsiniz.

6.4. Üçüncü Normal Form

Bu normal form, bir varlık içerisindeki niteliklerin anahtar özelliğine sahip olanlar dışında hiçbir varlığa fonksiyonel bağımlı olmamasını gerektirmektedir (Bernstein, P. A., 1976). Bir diğer deyişle bir varlıkta anahtar özelliği başka bir nitelik tarafından taklit edilmemelidir. Tablo, 3NF'ye uyumlu olmayan bir örnek sunmaktadır.

id	no	ad	soyad	...
1	325896	Lorem	Ipsum	...
2	514896	Dolor	Sit	...
3	125870	Amet	Consecte	...

Tablo ile verilen örnek öğrenci kayıtlarını içeren bir varlığı temsil etmektedir. Bu varlıkta öğrenci bilgileri id niteliğine fonksiyonel bağımlı olmakla birlikte aynı zamanda no niteliğine de bağımlılık gösterdiği görülmektedir. Bu durum 3NF'ye aykırıdır. Çözümü için bu niteliklerden biri kaldırılabilir. Aşağıda bir çözüm önerisi sunulmaktadır.

Ogrenci: no, ad, soyad, ...

Böylece fonksiyonel bağımlılığı sağlayan tek bir nitelik kalmıştır. Bu haliyle sunulan varlık 3NF'ye uygundur.

Her ne kadar teoride 3NF uyulması gereken zorunlu kurallar arasında sayılsa da pratikte, projelerde bu kurala uyulmadığı ve bu konuda tasarımcının haklı çıktığı birçok uygulama ile karşılaşılabilmektedir. Bu sebeple 3NF'ye uyulmasının avantaj ve dezavantajlarını kısaca ele alalım.

3NF'ye uyulmaması durumu, aynı görevin birden fazla nitelik tarafından gerçekleştirilmesine sebep olabilir. Örnek üzerinden gidersek öğrenci işaret edici için sistemin bir kısmında Ogrenci.no bir kısmında ise Ogrenci.id kullanılırsa, geliştirme ve yönetimde, sonu hatalı veri bütünlüğüne gidebilecek birçok karmaşıklığa sebep olabilir. Bunun yanında ikinci bir nitelik veri hacminde de bir artışa sebep olacaktır.

Bunların yanı sıra; eğer geliştirme esnasında kurallar net olarak ortaya konulursa ve iyi bir dokümantasyon yapılırsa bu karmaşıklıklar ortadan kalkabilir. Bununla birlikte birincil anahtarın her zaman sistem tarafından atanan ve hiçbir şekilde sistem yöneticileri tarafından müdahale edilemeyen bir yapıda olması veri bütünlüğünü garantiye alabilir. 3NF'ye uymamak uğruna yapılacak bu hareket ile bazı dezavantajlar göze alınacak olsa bile veri bütünlüğünü her daim sürdürmek daha öncelikli bir hedef olmalıdır.

6.5. Boyce-Codd Normal Form

Fonksiyonel bağımlılığı tanımlarken bir nitelik kümesinin başka bir nitelik kümesi tarafından temsil etmesinden bahsetmiştik. Fonksiyonel bağımlılığın tanımına ve başlarda yaptığımız atölyelere göre temsil eden tarafta niteliklerden oluşan bir kümeden bahsetmek oldukça normaldi. Ancak bu normal form ile bu konuda bir kısıtlamaya gidiyoruz. BCNF, varlık içerisinde tek bir birincil anahtar olmasını ve nitelikler arasında bir kısmi fonksiyonel bağımlılık olmamasını gerektirmektedir (Teorey, T. J., etc., 2011). Yani fonksiyonel bağımlılıkların tarifini, BCNF'ye uyan tasarımlar için bir nitelik kümesinin tek bir nitelik tarafından temsil edilebilmesi şeklinde güncellememiz gerekmektedir. Fonksiyonel bağımlılık başlığı altında sunduğumuz çözümlerde sol tarafta sıklıkla bir nitelik kümesi kullanmıştık. Ama aynı zamanda bunun en iyi çözümünün bir id kullanmak olduğundan da bahsetmiştik. Bir varlık içerisinde birçok niteliği kullanarak

anahtar oluşturmak yerine id gibi bir nitelik ile bunu sağlamak neredeyse her zaman yeterli seviyede performans sağlayacaktır.

Bir varlık içerisinde, bir varlık kümesine fonksiyonel bağımlılık gerçekleşmesi durumunda bu çözüm id veya benzeri bir niteliğin eklenmesiyle kolayca çözülebiliyor ancak eğer veri kümesindeki mevcut fonksiyonel bağımlılık zaten tek bir niteliğe ise bu niteliği anahtar olarak kullanmak BCNF'ye uyum için yeterli oluyor. Teoride bu çözüm yeterli gözükse de pratikte veri kümesinde bulunan bir niteliğin anahtar olarak tanımlanması pek de tercih edilen bir yöntem değildir. Seçilecek, veri kümesinde var olan niteliğin anahtar olarak tercih edilmesi, gelecekte bu anahtarın içeriği, rolü ya da kullanım şeklinde bir güncelleme yapılması riski sebebiyle her zaman bir riski beraberinde getirmektedir. Öğrencilere ait verileri içeren bir veri kümesindeki öğrenci numarası niteliğini örnek alalım. Öğrenci bilgilerinin öğrenci numarasına fonksiyonel bağımlı olduğu söylenebilir. Böylece bir öğrenci varlığı için öğrenci numarası birincil anahtar olarak tercih edilebilecektir. Ancak zaman içerisinde kayıt olmuş tüm öğrencilerin sayısı çok artacağı için öğrenci numarasının sıfırlanması söz konusu olabilir. Sık karşılaşılan bir şey olup olmadığını bilmemekle birlikte bu örnek olayı yaşamış biri olarak söyleyebilirim ki bu durumda bir süre sonra aynı öğrenci numarasına sahip birden fazla öğrenci veri tabanında yer alabilir. Bu durum gerçekleştiği anda veri tabanı, veri bütünlüğünü sağlamak adına aynı numaralı yeni bir öğrenci kaydına izin vermez. Bu da sistemin bir arızası olarak karşılanır.

Tüm bu sebeplerden dolayı varlıklar içerisinde dışardan etki edilmeyecek, çalışma mekaniği değiştirilmeyecek ve birincil anahtar özelliklerini asla kaybetmeyecek niteliklerin seçilmesi oldukça önemlidir. Bunun da en kolay yolu herhangi bir sorgulama yapmadan her varlığa id niteliği eklemektir.

6.6. Dördüncü Normal Form

Verinin kendini tekrar etmesi veri tabanı tasarımları için önemli bir problemdir. Bu problem çok nadir koşullar dışında hoş görülmez ve giderilmesi gereklidir. 4NF, bir veri kümesinde bazı nitelikler farklı değerler alıyorken bir niteliğin aynı değeri tekrar alması durumunda ortaya çıkmaktadır (Fagin, R., 1977). Bir niteliğin aldığı değerler kendini tekrar ediyorsa bu niteliğin bulunduğu varlığın değiştirilmesi ya da yeni bir varlık tanımlanması gerekiyor olabilir. Bir önceki tablodan uyarlanan aşağıdaki Tablo, 4NF'ye uymayan bir örnek içermektedir.

ders	bolum	programTuru
Veri Tabanı Tasarımı	YBS	lisans
Programlamaya Giriş	YBS	lisans
Sistem Analizi ve Tasarımı	YBS	lisans

Tablo sırasıyla dersleri ve bu derslerin yer aldığı bölümler ile diploma programı türlerini sunmaktadır. Her bir satır, yeni bir ders için kayıt içerirken aynı bölümde ve aynı diploma programı seviyesinde birden fazla ders olması sebebiyle bu değerlerin tekrar ettiğini görmekteyiz. Bunun çözümü bölümler için yeni bir varlık oluşturmak ve bire çok bağlantı sebebiyle derslerin olduğu varlığa ikincil anahtar olarak bolum_id eklemektir.

Fark ettiyseniz bu problem genellikle kategorik değer alan niteliklerle karşımıza çıkmaktadır. Eğer bir nitelik kategorik değer alacaksa henüz herhangi bir kayıt girmeden 4NF'yi düşünebilir ve gerekiyorsa önlem alabilirsiniz.

6.7. Beşinci Normal Form

Literatürde PJNF (Projection-Join Normal Form) olarak da anılan 5NF, eldeki çözümün daha küçük varlıklara bölünüp bölünemeyeceğini konu alır (Fagin, R., 1979). Bu normal forma göre; bir veri tabanı tasarımındaki tüm varlıklar, veri kaybı yaşanmadığı sürece bölünebilecekleri en küçük varlıklara bölünmelidir. Eğer bir varlık iki varlığa bölünebiliyorsa 5NF'ye uyumsuzluktan bahsedilebilir.

6.8. Alan/Anahtar Normal Form

Alan/Anahtar Normal Form (Domain/Key Normal Form - DKNF) olarak da adlandırılan 6NF, veri tekrarının bir başka formunun veri tabanı tasarımında bulunmasını konu almaktadır. Bu normal forma göre bir nitelik, başka bir niteliğin bir fonksiyonu ise; diğer bir deyişle bir nitelik başka bir niteliğe bağlı olarak hesaplanabiliyorsa bu bir nevi veri tekrarıdır (Fagin, R., 1981). Bu tür bir veri tekrarı 6NF'ye uyumsuzluk anlamına gelmektedir.

Bir varlık içerisinde hem doğum tarihi hem de yaş niteliklerinin bulunması 6NF'ye uyumsuzluğa güzel bir örnek teşkil eder. Yaş, doğum tarihine bağlı kolaylıkla ve nesnel olarak hesaplanabilen bir niteliktir. Bu sebeple bu iki niteliği taşıyan bir varlık 6NF'ye aykırıdır.

Bölüm Özeti

Normalizasyon süreci; bir veri tabanının, ilişkisel veri tabanı ilkelerine uygun, veri bütünlüğünü sağlamış ve yüksek performansla çalışan bir veri yapısı elde etme sürecidir. Dersin önceki bölümlerinde iyi bir veri tabanı tasarımı yapmanın deneyimle mümkün olduğunu vurgulamıştık. Ancak elbette öğrenme sürecindeki tasarımcıların da veri tabanı tasarımı gerçekleştirebilmeleri mümkün olmalıdır. Veri tabanı tasarımı sürecinin formüle edilmesini sağlayan çeşitli kurallar bulunmaktadır. Bu kurallar normal formlar olarak anılırlar. Normal formlara uygun hale getirilen veri yapılarının normalize edildiği yani iyi bir tasarıma sahip olduğunu savunulmaktadır.

Normal formlar; birinci, ikinci, üçüncü, Boyce-Codd, dördüncü, beşinci ve alan/anahtar normal form olarak literatürde yer almaktadır. Her bir normal form ilk öncül olarak veri yapısının bir önceki normal formda olması gerektiği kuralını içerir. Bununla birlikte her normal form, veri yapısı için uyulması gereken yapıyı tarif eder ve kurallaştırır.

Birinci normal form verinin atomik yapıda olması gerektiğini içerir. İkinci normal form, aynı tablo içerisinde alt fonksiyonel bağımlılıklar içeren nitelik kümeleri bulunmaması gerektiğini, yani yeni tablolar oluşturulması gereken durumları açıklar. Üçüncü normal formda birden fazla birincil anahtar olmaması gerektiği vurgulanır. Nitelikler, birden fazla niteliğe fonksiyonel bağımlı olmamalıdır. Boyce-Codd da yine tek bir birincil anahtar olması ve ona bağımlılıkla ilgili kuralları belirtmektedir. Dördüncü normal form veri tekrarının olmaması gerektiği bilgisini içerir. Veri tekrarı veri tabanı yapıları için ciddi bir problemdir. Beşinci normal form, bir tablo daha fazla tabloya bölünebiliyorsa bölünmelidir kuralını içerir. Veri yapısı, mümkün olan en küçük nitelik gruplarına bölünmeli ve veri bütünlüğü korunmalıdır. Alan/anahtar normal form, bir niteliğin başka bir niteliği kullanarak türetilmemesi gerektiği bilgisini içerir. Örneğin kişilerle ilgili bir tabloda hem doğum tarihi hem de yaş bilgisi saklanmamalıdır.

Kaynakça

Akadal, E. 2020. Veritabanı Tasarlama Atölyesi. Türkmen Kitabevi, İstanbul.

Bernstein, P. A., 1976. Synthesizing third normal form relations from functional dependencies. ACM Transactions on Database Systems (TODS).

Fagin R. 1977. Multivalued dependencies and a new normal form for relational databases. ACM Transactions on Database Systems (TODS), 2(3):262–278.

Fagin, R. 1979. Normal forms and relational database operators. In Proceedings of the 1979 ACM SIGMOD international conference on Management of data, 153–160.

Fagin, R. 1981. A normal form for relational databases that is based on domains and keys. ACM Transactions on Database Systems (TODS), 6(3):387–415.

Hoffer, J. A., Ramesh, V. & Topi, H. 2016. Modern database management. Pearson.

Kent, W. 1983. A simple guide to five normal forms in relational data- base theory. Communications of the ACM, 26(2):120–125.

Lee, H. 1995. Justifying database normalization: a cost/benefit model. Information processing & management, 31(1):59–67.

Teorey, T. J., Lightstone, S. S., Nadeau, T & Jagadish, H. V. 2011. Database modeling and design: logical design. Elsevier.