İlişkisel Veritabanlarında Denormalizasyon Etkisi: Bir Anket Uygulaması

Denormalization Effects on Relational Databases: A Survey Application

Erdinç Uzun, H. Nusret Buluş, Cihat Erdoğan, Heysem Kaya
Bilgisayar Mühendisliği Bölümü, Namık Kemal Üniversitesi, Tekirdağ, Türkiye
erdincuzun@nku.edu.tr

***Özetçe*— Veritabanı tasarımında, gereksiz özellikler ekleyerek veya özellikleri bir araya getirerek veritabanından okuma işlemini hızlandırmak için yapılan işlemlere denormalizasyon denir. Bu çalışmada, fakülte için geliştirilen anket değerlendirme sisteminde yapılan denormalizasyon işlemleri ve getirileri anlatılmıştır. Denormalizasyon işlemi sayesinde sonuçları elde etmede yaklaşık 38 kat bir hız kazanımı sağlanırken yaklaşık 130 kat depolama maliyeti kazancı elde edilmiştir.**

Anahtar Kelimeler — denormalizasyon, veritabanı tasarımı, performans.

***Abstract*— In database design, processes that are aimed to speed up reading by adding redundant fields or agglomerating available fields are called denormalization. In this study, the denormalization processes applied to the questionnaire evaluation system, developed for Çorlu Faculty of Engineering, and their advantages are explained. The denormalization process provides a speed up gain of 38 times, along with a storage cost gain of 130 times.**

Keywords — denormalization, database design, performance.

I. GİRİŞ

Veritabanı tasarımı son derece önemli bir yazılım geliştirme aşamasıdır. Tasarım işleminde normalizasyon [1] ve E-R (Entity - Relationship) [2] kavramları karşımıza çıkar. 1970'li yıllarda ortaya atılan normalizasyon kuralları günümüzde halen uygulanmaktadır. Chen tarafından 1976'larda önerilen E-R modeli ise daha hızlı veritabanı tasarımı yapmaya olanak sağlar. E-R modeli de normalizasyon kurallarına uyan bir modeldir. Normalizasyon kurallarına uymak her ne kadar doğru veritabanı tasarımı açısından önemli olsa da kötü sistem performansı gibi bir kusuru olabilir [3]. Normalizasyon kurallarının dışına çıkılarak bu kusur giderebilir. Bu çalışmada, bir anket uygulamasında normalizasyon kusurları ve denormalizasyon işleminin performans olumlu şekilde nasıl etkileyeceği anlatılacaktır.

Normalizasyon, veri tekrarını azaltarak bir ilişkisel veritabanında özellikleri ve tabloları düzenleme işlemleridir. Normalizasyon kavramının ilk konusu olan birincil normal form (1NF) Edgar Codd tarafından 1970 yılında tanıtılmıştır. 1971 yılında ikinci ve üçüncü normal formlar (2NF ve 3NF) tanımlanmıştır [4]. 1974 yılında Edgar Codd ve Raymond F. Boyce tarafından Boyce-Codd Normal Form (BCNF) sunulmuştur. 3NF durumuna gelmiş bir veritabanı tasarımı normal duruma gelmiş olarak kabul edilir. Bunun yanı sıra, 4NF ve 5NF kuralları normalizasyon işleminde kullanılabilir olsa da uygulama tarafında fazla tercih edilmemektedir [5]. Bu çalışmada, öncelikle 3NF durumuna gelmiş bir tasarımın oluşturabileceği performans sorunlarına normalizasyon kurallarının dışına çıkılarak çözüm üretilebileceği ve bu çözümün performans getirisi incelenecektir.

Günümüzde internetteki verinin hızlı bir şekilde artması verinin sorgulanabilme sorununu ortaya çıkarmıştır. Çünkü bir veritabanındaki kayıt sayısı arttıkça o kayıtlar üzerinden bilgi çıkarımı konusu da zorlaşmaktadır. Bu noktada indeksleme [6] ve OLAP (OnLine Analytical Processing) [7] bir çözüm olarak düşünülebilir. İndeksleme her ne kadar hızlı sorgulamaya imkân tanısa da sorgulama zamanını optimize etmek için ek bir depolama alanına ihtiyaç duyar. Birçok popüler veritabanı indekslemeye destek vermektedir. OLAP ise verideki önemli özelliklerin belirlenip bu özelliklere ait hızlı sorgulama yapılabilecek bir yapı tasarlayan yazılımlardır. Popüler veritabanlarında OLAP desteği ek yazılımlar ile olmaktadır. Bu çalışmada, indekslemenin getirileri ve OLAP bir yapıya gerek kalmadan verinin çok hızlı bir şekilde sorgulanabilmesine olanak sağlayacak bir veritabanı tasarımı yapılacaktır. Bu tasarımda en temel normalizasyon kurallarının dışına çıkılacaktır.

Denormalizasyon, sorgulama sonuçlarını daha hızlı şekilde elde etmek için veritabanı tasarım tarafında değişiklikler yapmaktır. Bu değişiklikleri yaparken veritabanı tarafındaki normalizasyonda kesinlikle önerilmeyen ek özellikler eklenebilir veya mevcut

özellikler birleştirilebilir. Amaç, daha hızlı bir şekilde veriler üzerinde sorgulama veya okuma yapabilmektir.

Bu çalışmada, fakültemiz için geliştirilen anket yazılımının veritabanı tasarımındaki denormalizasyon işlemi anlatılacaktır. Anket uygulama sonuçlarının hızlı bir şekilde öğretim üyesine gösterilmesi için denormalizasyon işlemi uygulanmıştır. Denormalizasyon işleminin performans sonuçları standart normalizasyon işlemi sonucu elde edilen veritabanı ve bu veritabanı üzerinde yapılan indeksleme sonucu elde edilen sonuçlar ile karşılaştırılmıştır.

İkinci bölümde, denormalizasyon konusunda yapılan çalışmalar anlatılacaktır. Üçüncü bölüm anket problemi ve normalizasyon sonucu elde edilen veritabanı tasarımına ayrılmıştır. Dördüncü bölümde, denormalizasyon işlemi sonucu elde edilen veritabanı tasarımı ve yazılım tarafında yazılan kodların işlevi anlatılacaktır. Beşinci bölüm veritabanları üzerinde yapılan testleri ve karşılaştırma sonuçlarını içermektedir. En son bölümde güncel teknolojiler içinde denormalizasyon işleminin önemi tartışılıp, gelecek çalışmalar hakkında bilgi verilecektir.

II. DENORMALİZASYON

Normalizasyon her ne kadar veritabanı tasarımı konusunda temel kuralları içerse de bu kurallar belirgin verimsizlikler oluşturabilir. Diğer taraftan, denormalizasyon işlemi sayesinde sorgulama hızı arttırılabilir ama bu işlem yapılırken normalizasyon kurallarının dışına çıkılmış olur. İlk olarak, Schkolnick ve Sorenson denormalizasyon işleminin temellerini tanıtmışlardır [3]. Bu çalışmada, denormalizasyon işleminin sistemi iyi bilen ve veri modeli anlayabilecek bir kullanıcı ile veritabanı tasarımı yapıldığında veritabanı performansının arttırılabileceği belirtilmiştir. Cerpa ise denormalizasyon gibi bir işlemin uzman bir veritabanı tasarımcısı tarafından yapılması gerektiğini belirtmiştir [8]. Birçok araştırmacı farklı denormalizasyon tipleri geliştirirken bu işlemin dikkatli bir şekilde yapılması gerektiğinin altını çizmiştir [9, 10].

İyi bir denormalizasyon işlemi performans kazancı olarak dönerken, veritabanı esnekliği açısından bir kayıp söz konusudur. Denormalizasyon yapılırken istenen performansa ulaşmak için veri miktarı, hangi veritabanı yönetim sisteminin kullanıldığı, işletim sistemi ve hatta donanım dikkate alınmalıdır [11].

Bir veritabanı tasarımında tavsiye edilen temel kural "Önce normalize et, eğer performans problemi var ise denormalize işlemi uygula" dır. Denormalizasyon işleminin kesin kuralları yoktur ve veritabanı tasarımcısının sorguların yavaşlamasına karşın bulduğu çözümler olarak düşünülebilir [12]. Örneğin bu çözümler

- ✓ birden fazla tablodan JOIN yapan bir tasarımcının JOIN işleminin sorguları yavaşlattığını görünce ek tablodaki gerekli verileri ana tabloya taşıması,
- ✓ aynı tabloda ve/veya başka tablolarda bulunan verileri kullanarak SQL (Structured Query Language) fonksiyonları ile hesaplanıp bulunan bir değeri her gösterimde hesaplanmasın diye ayrı bir özellik olarak ekleme,
- ✓ birden fazla sütunu/satırı birleştirip tek sütunluk özelliğe çevirme,
- ✓ çok kullanılan verilerin ayrı tabloya taşınması

şeklinde gruplanabilir. Bu çözümler veritabanı tasarımcısının veya uygulama geliştiricisinin performans kriterini düşünüp bulduğu kişisel çözümlerdir. Probleme dayalı olarak çok farklı çözümlerde üretilebilir. Bu çalışmada anket uygulaması için önerilen çözüm ve bu çözümün performans getirileri gösterilecektir.

Veriyi daha hızlı bir şekilde sorgulamak için indeksleme ve OLAP gibi teknikler de kullanılır. İndeksleme işleminde tablolardaki özelliklerden hızlandırılmasını istenen özellik veya özellikler indekslenebilir. MySQL, PostgreSQL, Oracle, MS SQL Server gibi birçok popüler veritabanı yönetim sistemi farklı indeksleme türlerine destek vermektedir [5]. OLAP için ayarlanan veritabanında tablo veya tablolardan belirli özellikler belirlenir ve bu özelliklerden çok boyutlu bir veri modeli oluşturulur. Çok boyutlu veri modeli sayesinde karmaşık analitik sorgu cevapları çok hızlı bir şekilde elde edilebilir [6]. Hem indeksleme hem de OLAP ek alanlara ihtiyaç duyarlar. Bu ek alanlar bir taraftan dezavantaj oluştururken diğer taraftan performans kazandırırlar. İndeksleme işlemi basit sorguları hızlandırmak için kullanılırken OLAP daha karmaşık analitik sorgulara cevap verebilir. OLAP ayrı araçlar olarak sunulurken indeksleme ise birçok veritabanı yönetim sistemi için standart olmuştur. Bu çalışma da, indekslemenin sorgu performansına katkısı ayrıca incelenecektir.

Uzun yıllar fazla araştırmanın yapılmadığı denormalizasyon kavramı büyük veri ve büyük veriler üzerinde veri madenciliği gibi konular ile son yıllarda tekrar üzerinde durulur bir konu haline gelmiştir. Ordonez ve ark. gelişmiş E-R modellerinde veri madenciliği için veri seti yapmada denormalizasyon işleminin önemi üzerinde durmuşlardır [13]. Lee ve Zheng [14] ile Ho ve ark. [15] ilişkisel modelden NoSQL modele geçişte denormalizasyon işleminin nasıl yapılması gerektiği üzerinde durmuşlardır. Bu çalışmanın amacı bir uygulamada doğru yapılmış bir denormalizasyon işleminin getirilerini literatürle paylaşmaktır.

III. UYGULAMA VE NORMALİZASYON ÇÖZÜMÜ

Bu bölümde uygulama hakkında bilgi verildikten sonra üretilen normalizasyon çözümü açıklanacaktır. Bir veritabanı tasarımında uygulama tarafının çok iyi dinlenmesi ve bu uygulamaya uygun bir tasarım yapılması çok önemlidir. Fakültemiz öğrenci görüşlerini toplamak için bir anket düzenleme konusunda bölümümüzden destek istenmiştir. İlk görüşmede, ankette istenilen bilgileri

toplanarak normalizasyon kurallarına uygun bir tasarım yapılmıştır. Anketi kimin doldurduğu bilgisinin tutulmaması istenirken bir kez anketi dolduran öğrencinin bir daha anket doldurmaması istenmiştir. Normalizasyonu istenen veriler aşağıdaki gibidir.

Veri sözlüğü
Anket Soru, Cevap, Öğrenci, Ders, Dönem

A. Klasik Tasarım: Normalizasyon

Öncelikle bir anket dolduran öğrencinin bir daha anket doldurması engellemek için yapılmış normalizasyon işlemi ile başlayalım. Oluşturulan tablolar, özellikleri ve birincil anahtarlar aşağıda verilmiştir.

Normalize durum
Öğrenci(Öğrenci_No, Adı, Soyadı, …)
Bölüm(Bölüm_No, Bölüm Adı, …)
Ders(Ders_Kodu, Ders Adı, Bölüm_No, Öğretim_Üyesi_No …)
Öğretim_Üyesi(Öğretim_Üyesi_No, Adı, Soyadı, Bolum No, …)
Dönem(Dönem_No, Dönem Adı)
Anket_Kontrol(Öğrenci_No, Ders_Kodu, Dönem_No)

Öğrenci anket doldurmadan önce anket_kontrol tablosundan öğrencinin o ders ve belirli bir dönem için anket doldurduğunun bilgisi alınır. Eğer öğrenci bu tabloda var ise o ders için anket kullanılması engellenir. Bu bölüme denormalizasyon işlemi uygulanmayacaktır. Şimdi anketin sonuçlarının tutulacağı tabloya gelelim.

Normalize durum
Soru(Soru No, Soru, Bölüm_No)
Anket_Cevap (Anket_No, Ders_No, Donem_No, Soru_No, Cevap)

Öğrenci derse ait anketi doldurmadıysa anket doldurma işlemini başlattığında soru tablosundan seçilen dersin bölümüne göre sorular getirilir. Öğrenci tüm soruları cevapladıktan sonra Anket_Cevap tablosuna cevaplar eklenir. Bu tabloda öğrenciye ait bir bilgi tutulmamaktadır. Birincil anahtar olarak otomatik artan bir sayı "Anket No" özelliği tutulur. "Ders No", "Donem No", "Soru No" ilişki kurmak için kullanılmışken Cevap 1-5 arasında öğrencinin seçtiği değer bilgisi tutmaktadır. Kullanıcı tarafından veriler alındıktan ve ayrıştırıldıktan sonra ekleme SQL ifadesi çalıştırılmalıdır.

B. Ankete Veri Ekleme

Anket_Cevap tablosuna ekleme yaparken toplu INSERT SQL ifadesi kullanmaya dikkat edilmelidir. Örneğin 20 sorudan oluşan bir anketin ekleme işlemi için ayrı 20 INSERT yerine tek INSERT SQL ifadesi yazılmalıdır. Örneğin

```
INSERT INTO anket_cevap (Ders_No, Donem_No, Soru_No, Cevap) VALUES (2, 1, 1, 4), (2, 1, 2, 3), (2, 1, 3, 5), …
```

VALUES ifadesinden sonraki kısımda birden fazla verinin nasıl eklendiğinin örneği verilmiştir. Bu kısma uygulamayı geliştirirken dikkat edilmesi gerekmektedir.

Anket öğrenciler tarafından doldurulduktan sonra istatiksel sonuçların öğretim üyesine gösterilmesi gerekir. Bu noktada iyi bir şekilde tasarlanmış tablo veya tablolar üzerinden SQL ile kolayca raporlar hazırlanabilir.

C. Anket Sonuçların Elde Edilmesi

Öğretim üyesine rapor hazırlanırken SQL fonksiyonları kullanır. SQL fonksiyonları birden fazla satırdan bilgi edinmek için kullanılır. Aşağıdaki gibi SQL ile öğretim üyesi için sonuçlar kolayca hazırlanır.

```
SELECT s.Soru, AVG(Cevap), MIN(Cevap), MAX(Cevap) , STD(Cevap) FROM Anket_Cevap a
INNER JOIN Soru s ON a.Soru_No = s.Soru_No
WHERE a.Ders_No=X AND a.Donem_No=Y
GROUP BY a.Soru_No
```

Seçilen satırların ortalaması için AVG, minimumu bulmak için MIN, maksimumu bulmak için MAX ve standart sapmasını bulmak için STD fonksiyonları kullanılmıştır. INNER JOIN işlemi "Soru" tablosundan soruları çekmek işlemini gerçekleştirir. Ayrıca GROUP BY kullanılarak cevaplar Soru_No özelliğine göre gruplandırılmıştır. Ders_No ve Donem_No bilgileri öğretim üyesinin seçimine göre belirlenir. Öğretim üyesi X ve Y değerlerini farkında olmadan bir tıklama ile uygulama tarafına göndermektedir. Bu durum saldırısı kolay bir yapı gibi durmaktadır. Tabii ki diğer kullanıcıların bu SQL ifadesine ulaşamaması gerekir. Başka bir deyişle, güvenlik konusu da önemlidir. Ancak bu çalışma sadece veritabanı tarafı ile ilgilenmektedir.

D. İndeksleme

Bir önceki bölümdeki SQL ifadesinde INNER JOIN işlemi, WHERE sorgusundaki kısıtlar ve GROUP BY ifadesindeki özellik üzerine işlemler yapılmaktadır. Bu işlemler veri içinde arama ve gruplama içerir. Bu işlemleri hızlandırmanın en temel yöntemi indekslemedir. Çalışmamızda "Soru_No" ve "Ders_No - Donem_No" alanlarına indeks uygulanmıştır. Farklı indeksleme yöntemleri varken veritabanı yönetim sistemleri kendisine has indeksleme yöntemleri barındırmaktadır. Bu çalışmada MySQL'in kullandığı B+ Tree arama algoritması indeks yöntemi olarak kullanılmıştır. Test bölümde indeks içermeyen durum ve indeks içeren durum kıyaslanacaktır.

IV. DENORMALİZASYON ÇÖZÜMÜ

Denormalizasyon işlemi performans kazanımları düşülüp sadece gerekli tablolara uygulanması yeterli bir işlemdir. Anket uygulamasının en fazla verinin eklendiği tablo Anket_Cevap tablosudur. Bu tabloda her öğrencinin anketi için anketteki soru sayısı kadar satır açılmaktadır. Örneğin bir dönemdeki bir ders için 20 soruluk bir anketi 40 öğrenci doldurduğunda 800 satırlık bir veri girişi yapılmış olur. SQL fonksiyonları sayesinde bu veri içinden

analitik bilgilere ulaşabilir. Satır sayısı ders, dönem, soru ve öğrenci sayısına bağlı olarak çok fazla satırdan oluştukça analitik bilgileri elde etme etkinliğinin yavaşlaması kaçınılmazdır. Bu durumda, denormalizasyon işlemi bir çözüm olarak düşünülebilir. Yapacağımız denormalizasyon işleminde amacımız bir ders ve bir dönem için yüzlerce satırda tutulan veriyi tek satıra indirgemek olacaktır.

Anket_Cevap_Denor (Ders_No, Donem_No, Veri)

Anket_Cevap_Denor tablosuna "Veri" özelliğinde her bir öğrenciden gelen sonuçların tutulduğu bir özellik eklenmiştir. Denormalize tabloda: Anket_Cevap Anket_No, Soru_No ve Cevap bölümleri çıkartılmıştır. Öğrencin işaretlemelerinden gelen Soru_No ve Cevap verileri "Veri" özelliği içinde tutulacaktır. Başka bir deyişle birden fazla satırda tutulan bir veri tek sütunda tutulur hale çevrilecektir.

A. "Veri" Özelliğinin Yapısı

Derse ve döneme ait veriler tek satırda tutulacağı için "Veri" özelliği uygulamaya özgü şekilde tasarlanmalıdır. Geliştirdiğimiz anket uygulamasında JSON formatında öğrencilerden gelen sonuçlar saklanmıştır.

Soru_1=4-20,2-28,5-18,3-18,1-16&Soru_2=1-22,5-19,4-21,2-22,3-16&Soru_3=4-20,3-19,2-17,1-14,5-30&Soru_4=2-23,1-22,5-19,4-15,3-21&…

Buradaki formatta Soru_No=Cevap_No-Frekans şeklinde bir yapı tasarlanmıştır. Örneğin Soru 1'e 4 şıkkına 20, 2 şıkkına 20, 5 şıkkına 18, 3 şıkkına 18 ve 1 şıkkına 16 işaretleme gelmiştir. Bu bilgiler sayesinde Soru 1 için ortama, minimum, maksimum ve standart sapma kolaylıkla hesaplanabilir. "&" operatörü ile de diğer tüm sorular birbirine bağlanmıştır. Bu yapıda soru sayısı değişken olabilir. Aslında, normalizasyon işlemi sonucunda yüzlerce satırda tutulan bir veri denormalizasyon işlemi sayesinde tek bir özellik içinde tutulabilir hale gelmiştir.

B. Ekleme ve Güncelleme

Gelen bir anket verisi normalizasyon işlemini sonucu oluşturulan Anket_Cevap tablosuna INSERT kullanımı ile kolay bir şekilde eklenebilir. Ancak, denormalizasyon işlemi sonrası veri tek satırda tutulacağı için aşağıdaki Algoritma 1 gibi bir kaba koda ihtiyaç vardır.

Anketin doldurulduğu Ders No ve Donem No bilgileri X ve Y değişkenlerine atılmıştır. Ayrıca, kullanıcıdan gelen bir anket değerlendirmesi aşağıdaki gibidir.

Soru_1=3&Soru_2=4&Soru_3=2&Soru_4=3&…

Kullanıcıdan gelen değerlendirme Soru ve Cevap bilgisine ayrılıp bir Hashtable değişkenine atılır. Hashtable bir anahtar ve değer olmak üzere iki değişkenden oluşur. Anahtar üzerinden O(1) zaman karmaşıklığı ile başka bir deyişle çok hızlı arama yapmak mümkündür. Ayrıca anahtar üzerinden kolayca değere ulaşılabilir.

```
X = Ders No ve Y = Donem No
HashTable Soru_Cevap = Kullanıcıdan Gelen Soru_No ve Cevap bilgisi
al.
HashTable Mevcut_Veri = Soru_No = Cevap-Sayısı
Mevcut_Veri = SELECT Veri FROM Anket_Cevap_DeNor WHERE
Ders_No=X AND Donem_No=Y

IF Mevcut_Veri = = NULL THEN
BEGIN
 yeni_veri_sql = ""
 FOREACH Soru_Cevap BEGIN
  IF yeni_veri_sql = = "" THEN
   yeni_veri_sql = "Soru_" + Soru_No + "=" + Cevap + "-1"
  ELSE
   yeni_veri_sql = ",Soru_" + Soru_No + "=" + Cevap + "-1"
  END IF
 END FOREACH
    INSERT INTO anket_cevap_denor (Ders_No, Donem_No, Veri)
    VALUES (X, Y, yeni_veri_sql)
END IF
ELSE
 update_veri_sql = " "
 FOREACH Soru_Cevap BEGIN
  geçici = Mevcut_Veri[Soru_No]
  HashTable Cevap_Sayi = Cevap ve Sayı bilgilerini geçici değişkeni
  ayrıştır.
  update_temp = ""
  FOR EACH Cevap_Sayi BEGIN
   IF Soru_Cevap.Contains(Cevap_Sayi.Cevap) THEN
     Cevap_Sayi.Sayi++
   ELSE
     Cevap_Sayi.Sayi = 1
   END IF
   IF update_temp = = " " THEN
    update_temp = Cevap_Sayi.Cevap + "-" + Cevap_Sayi.Sayi
   ELSE
    update_temp += "," + Cevap_Sayi.Cevap + "-" + Cevap_Sayi.Sayi
   END IF
  END FOR EACH
  IF update_veri_sql = = "" THEN
   update_veri_sql = update_temp
  ELSE
   update_veri_sql = "&" + update_temp
  END IF
 END FOREACH
  UPDATE anket_cevap_denor SET Veri= update_veri_sql WHERE
  Ders_No=X AND Donem_No=Y
END ELSE
```

Algoritma 1. Denormalize edilmiş tabloya veri ekleme veya veriyi güncelleme kaba kodu

Öncelikle bu ders için bir veri girilip girilmediği bir SELECT sorgusu ile bulunur. Eğer Select sorgusundan dönen cevap boş ise bu ankete ilk girişin yapılacağı anlamına gelir. Bu durumda tüm cevaplar dolaşılır ve ilk giriş olduğu için Cevapların yanına 1 bilgisi eklenir.

Örneğin;

Soru_1=3-1&Soru_2=4-1&Soru_3=2-1&…

şeklinde bir dizgi hazırlanır. Bu dizgi INSERT sorgusu ile veritabanına kaydedilir. Bundan sonra gelen aynı derse ve döneme ait tüm anketler için aynı satır güncellenecektir. Başka bir deyişle bundan sonra SELECT sorgusundan dönen cevap boş olmayacaktır.

SELECT sorgusundan dönen boş olmadığında kullanıcıdan gelen her cevabın veri tabanından dönen mevcut veri içinde araması yapılır. Örneğin kullanıcıdan

Soru_1=1&Soru_2=4&Soru_3=5&…

gibi bir cevap geldiğinde tüm veriler teker teker değiştirilir. Örneğin Soru_1 için mevcut veri Soru_1=3-1 şeklindedir. Kullanıcıdan gelen değer Soru_1=1'dir. Bu durumda Algoritma 1'e göre update_temp değişkeni içinde aşağıdaki gibi bir değer döndürülür.

Soru_1=3-1, 1-1

Tüm sorular aynı şekilde düzenlenir ve aşağıdaki bir veri elde edilir.

Soru_1=3-1, 1-1&Soru_2=4-2&Soru_3=2-1,5-1&…

Bu veri her gelen anket değerlendirmesinden sonra güncellenir. Öğretim üyesi değerlendirme sonuçlarını görmek istediğinde bu veri üzerinden sonuçlar hazırlanacaktır.

C. Anket Sonuçlarının Elde Edilmesi

Denormalizasyon işleminden sonra veri tek satırda tutulduğu için sorgulanmasının daha hızlı olması beklenmektedir. Deney bölümünde performans karşılaştırması ve kazancı incelenecektir. Bu çalışmada bu veri üzerinden ortalama, minimum, maksimum ve standart sapma değerleri hesaplanacaktır. Algoritma 2'deki bir soru için bu hesaplamanın nasıl yapılacağı gösterilmiştir. Benzer şekilde tüm sorular için hesaplama yapılır. Bu sayede öğretim üyesi tüm sorular için yapılmış değerlendirmeyi görebilir.

Algoritma 2'de birinci döngü ile minimum, maksimum, cevap toplamı (t_sayi) ve cevap / cevap sayısı toplamı (toplam) hesaplanmıştır. Örneğin Soru_1=4-20,2-28,5-18,3-18,1-16 için minimum değer 1, maksimum değer 5, toplam 296 ve t_sayi 100'dir. Ortalama değer 2.96'dır.

```
HashTable Cevap_Sayi = Cevap ve Sayı bilgilerini geçici değişkeni
ayrıştır.
toplam = 0
min = 6
max = 0
t_sayi=0
FOREACH Cevap_Sayi BEGIN
  IF min > Cevap_Sayi.Cevap  THEN
    min = Cevap_Sayi.Cevap
  END IF
  IF max > Cevap_Sayi.Cevap THEN
    max = Cevap_Sayi.Cevap
  END IF
  toplam += Cevap_Sayi.Cevap * Cevap_Sayi.Sayi
  t_sayi += Cevap_Sayi.Sayi
END FOREACH
ortalama = toplam / t_sayi
std=0
FOREACH Cevap_Sayi BEGIN
  std += Cevap_Sayi.Sayi * (Cevap_Sayi.Cevap - ortalama)^2
END EACH
std = KAREKÖK( std / t_sayi )
```

Algoritma 2. Denormalize edilmiş veriden bilgi çıkarımı

Standart sapmanın hesaplanabilmesi için ortalamaya ihtiyaç vardır. Ortalama değer hesaplandıktan Algoritma 2'deki ikinci döngü sayesinde standart sapma hesaplanacaktır. Algoritma 2'e göre std yani standart sapma değeri 1.36'dır. Bu hesaplamalar tüm sorular için yapılmalıdır.

V. DENEYLER

Normalizasyon ve denormalizasyon işlemi sonucu elde edilen tasarımların karşılaştırması için otomatik bir veri seti hazırlanmıştır. Bu veri setinde 32 bölüm, her bölüm için 30 ders, bir dönemlik her ders için 100 öğrenci anketi ve her anket için 20 soru olduğu var sayılmıştır. Anket sorusuna cevap için 1-5 arası rastgele bir sayı seçimi yapılmıştır. Ayrıca, normalizasyon işleminde hem indeksleme yapılmamış tablo hem de indeksleme yapılmış tablo yaratılmıştır. 20 soru aynı anda normalizasyon, indeksli normalizasyon ve denormalize tabloları olmak üzere 3 farklı tabloya da kaydedilmiştir. Her bölümde 30 ders için anketler bitirildikten sonra o bölüm dersleri için sonuçların elde edilmesi üç farklı tablo için de test edilmiştir. Öncelikle üç farklı tablo için veri ekleme/güncelle süreleri kıyaslanacaktır.

Deneyler, 64 bit - 1.60 Ghz işlemcili 4 GB RAM'e ve SSD diske sahip bir bilgisayar kullanılmıştır. Deneyler üç defa yapılmış ve ortalaması alınmıştır. Veritabanı olarak seçilen MySQL'in 5.6.14 versiyonu kullanılmıştır. Yazılım test kodları C# dilinde .NET Framework 4 kütüphanesi fonksiyonları kullanılarak yazılmıştır. Zaman ölçüm işlemlerinde .NET son versiyonlarında karşımıza çıkan ve performans ölçümleri için önerilen Stopwatch kütüphanesi kullanılmıştır.

A. Tablolara Veri Ekleme veya Güncelleme

Anketi bir öğrenci doldurmuş gibi 20 sorunun cevabı rastgele hazırlanmış ve normalizasyon işlemi sonucu elde edilmiş normalize tablolara toplu ekleme işlemi yapılmıştır. Normalize tablolar indekslenmemiş ve indekslenmiş olarak iki farklı tabloya ayrılmıştır. Diğer taraftan denormalizasyon işleminden elde edilen denormalize tabloya ise ekleme/güncelleme işlemi yapılmıştır. Denormalize tabloda ekleme veya güncelleme işlemlerinden biri yapıldığı için ayrı ayrı incelenmiştir. Ayrıca her iki işlemin ortalama sonucu ayrıca hesaplanmıştır. Bu tablolara yapılan ekleme/güncelleme işlemlerin sonucu elde edilen sonuçlar Tablo 1'de verilmiştir.

İşlem Türü	ms	Std
Normalize tabloya ekleme	2.729	0.417
Normalize indekslenmiş tabloya ekleme	3.326	0.834
Denormalize tabloya veri ekleme	4.096	0.735
Denormalize tabloda veri güncelleme	3.565	0.636
Denormalize tablo (Ekleme/Güncelleme)	4.090	0.764

Tablo 1. Tablolara ekleme/güncelleme sürelerinin karşılaştırılması

Bu sonuçlara göre indeksleme yapılmamış bir normalize tabloya ekleme işlemi ortalama 2.729 ms'de (milisaniye) gerçekleşebilmektedir. Standart sapmanın (std) 0.417 ms olması ekleme işleminde veri sayısının artmasına rağmen büyük farklılıklar olmadığını göstermektedir. İndeksleme işlemi uygulanmış bir tabloda ekleme süresi 3.326 ms çıkmıştır.

İndeksleme yapılmış bir tabloda ek veriler eklenmesi bu performans düşüşüne neden olmuştur. Bu performans düşüşün anket sonuçlarına ulaşmadaki etkisi ayrıca incelenecektir. Denormalize edilmiş tabloda ders ve dönem bilgisine bağlı olarak tüm veriler aynı satırda tutulmaktadır. Denormalize edilmiş tabloda veri yoksa ekleme işlemi ve veri varsa güncelleme işlemi yapılmaktadır. Ortalama 4.090 ms'de verinin eklendiği/güncellendiği görülmektedir. Ekleme ve güncelleme süreleri arasında beklenildiği gibi ufak bir farklılık göze çarpmaktadır. Ancak indekslenmemiş ve indekslenmiş normalize edilmiş tablolara veri ekleme işlemin daha hızlı yapıldığı görülmektedir.

Denormalizasyon işleminin ekleme süresini yavaşlattığı görülmektedir. Bir sonraki bölümde bu yavaşlamanın anket sonuçlarını elde etmedeki getirisi incelenecektir.

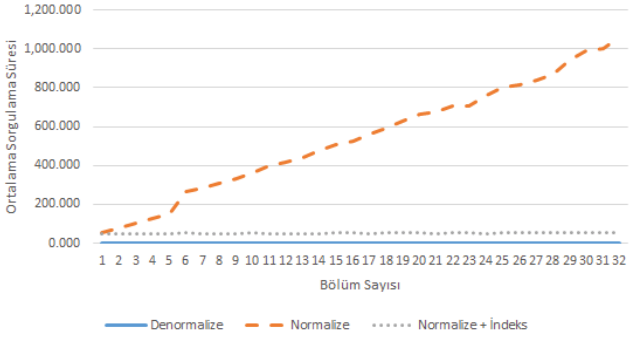
B. *Anket Sonuçlarına Ulaşma Süresi*

Bu bölümde, indekslenmemiş ve indekslenmiş normalize tablo üzerine üçüncü bölüm C başlığında anlatılan SQL ifadesinin zaman ölçümleri yapılacaktır. Ayrıca, denormalize edilmiş dördüncü bölüm C başlığındaki tek satırdan verinin ayrıştırılıp sonuçların elde edilmesinin getirisi incelenecektir. Ortalama sonuçlar Tablo 2'de verilmiştir.

Tablo 2'deki sonuçlara göre indeksleme işlemi beklenildiği gibi anket sonuçlarının yaklaşık 10.6 kat daha hızlı elde edilmesini sağlamaktadır. Diğer taraftan denormalizasyon işlemi ortalama 1.368 ms gibi çok hızlı bir geri dönüş değerine sahiptir. Görüldüğü gibi iyi düzenlenmiş bir denormalizasyon işlemi anket sonuçlarının hızlı bir şekilde elde edilmesini sağlamaktadır. Diğer bir deyişle normalize tabloya göre 399.5 ve indekslenmiş normalize tabloya göre ise 37.6 kat daha hızlı bir şekilde sonuçlar elde edilmiştir. Her ne kadar ekleme/güncelleme işleminde denormalize tablo 4.090 ms'lik bir değere sahip olsa da sorgulama performans getirisi açısından denormalizasyon işlemi sonuç elde etme konusunda göz önüne alınması gereken bir işlemdir.

Sorgulama Türü	ms	Std
Normalize tablo	546.735	294.270
Normalize indekslenmiş tablo	51.494	4.415
Denormalize tablo	1.368	0.485

Tablo 2. Anket sonuçlarının elde edilmesi



Şekil 1. Kayıt sayısının anket sorgulama süresine etkisi

Tablo Türü	Veri	İndeksleme
Normalize tablo	86.6 MB	-
Normalize indekslenmiş tablo	83.6 MB	110.8 MB
Denormalize tablo	1.5 MB	-

Tablo 3. Veritabanındaki tablo boyutları

C. *Kayıt Sayısının Anket Sonuçlarına Ulaşma Süresine Etkisi*

Kayıt sayısının artması anket sonuçlarının elde edilmesini yavaşlatabilecek bir işlemdir. Bu test için her bölüme ait bir dönemdeki 30 ders ve her bir ders için 100 anket sonucu (ankete katılan 100 öğrenci) olmak üzere her bir anket 20 sorudan oluştuğu var sayılmıştır. Her bir bölüme normalize edilmiş tablolarda 60,000 satır eklendi.

Denormalize tabloda ise 30 ders için 30 satır yeterli olmuştur. Her bölümden sonra anket sorgulama sonuçlara süresinin karşılaştırması Şekil 1'de verilmiştir. Normalize edilmemiş tablo 60,000 satırı 30 ders için ortalama 55 ms sorgulanırken 32 bölüm eklendikten sonra kayıt sayısı 1,920,000 olduğunda sorgulama süresi 1 sn'in üzerine çıkmıştır. Bu sonuçlara göre sadece normalizasyon işlemi uygulanmış bir tablo veri sayısı arttıkça sorgulama hızının azalması kaçınılmazdır. Diğer taraftan normalize edilmiş ve indekslenme uygulanmış tabloda 48 ms ile başlayan sorgulama süresi 32 bölüm eklendikten sonra 56 ms'e çıkmıştır. Bu durum indekslemenin önemini gösterir. Son olarak, denormalize tabloda sorgulama süresi 0.9 ms ile başlarken kayıt eklendikçe 1.9 ms civarına çıktığı görülmektedir.

D. *Tabloların Dosya Boyutu*

İndeksleme işlemi için ek bir depolama alanına ihtiyaç vardır. Denormalizasyon işleminde her ne kadar bilgiler tek satırda da toplansa daha fazla boyutta bir metin özelliğine ihtiyaç vardır. Tablo 3, veritabanında oluşturulan tablo boyutlarını vermektedir.

Tablo 3'teki tablolardın her birinde INNODB engine kullanılmıştır. İndeksleme işlemi için B+ Tree kullanılmıştır. Tablo 3'teki sonuçlara göre normalize edilmiş ve normalize edilmemiş veri miktarı arasında pek fark yoktur. Ancak, normalize edilip ayrıca indekslenmiş tabloda ayrıca 110.8 MB gibi büyük bir alan kullanılmıştır.

Bu alan anket sorgulama hızını arttırsa da ek depolamanın maliyeti olduğu göstermektedir. Denormalize sonrası veriler için VARCHAR veri türünden 1000 karakterden oluşan bir alan ayrılmıştır. Derse ait tüm bilgiler bu alanda tutulmaktadır. Denormalize tablo ise 1.5 MB bir alan kapladığı görülmektedir. Bu sonuca göre denormalizasyon işleminin doğru bir şekilde uygulandığında depolama maliyetinin de azaltıldığı görülmektedir.

VI. SONUÇ

Denormalizasyon, sorgulama performansını arttırmak için veritabanı tasarımında normalizasyon kurallarının dışına çıkılmasıdır. Bu kuralların dışına çıkan kişi veritabanı yönetim sistemleri ve uygulama geliştirme tarafında bilgisinin yanı sıra uygulama analizini çok iyi yapmalıdır. Bu çalışmada, anket uygulaması geliştirmede bu kuralların dışına çıkmanın getirileri incelenmiştir.

Çalışma sonucunda, denormalizasyon işlemi sayesinde normalize edilmiş ve indekslenmiş tabloya göre anket sonuçlarını elde etme hızı ortalama 51.5 ms'den 1.37 ms'e kadar düşürülmüştür. Başka bir deyişle anket sorgulama işleminde denormalizasyon sayesinde yaklaşık 38 katlık bir kazanım söz konusudur. Normalize edilmiş ve indekslenmiş bir tablonun depolama maliyeti 194.4 MB'ı bulmuştur. Denormalize tabloda ise bu maliyet 1.5 MB civarındadır. Depolama açısından 130 katlık bir iyileşme söz konudur.

Denormalizasyon işlemi özellikle NoSQL veritabanları [16] gibi güncel konuları anlatmadan önce temel konu niteliğindedir. Ancak yaptığımız incelemede normalizasyon işleminin çoğu üniversitede anlatılması rağmen denormalizasyon işleminin konu olarak anlatılmadığı görülmektedir. Bu çalışma diğer taraftan, doğru yapılmış bir denormalizasyon işleminin performans getirilerini göstermektedir. Özellikle iş hayatında yapılan iyi denormalizasyon işlemlerinin maliyetli OLAP yazılımlarına ihtiyacı azaltacağı kanaatindeyiz.

NoSQL veritabanları bazı alanlarda klasik veritabanı yönetim sistemlerinin yerini almaktadır. İleriki çalışmalarda veriyi NoSQL veritabanları ile klasik veritabanlarında tutmanın incelemesi yapılacak ve hangi durumda hangi tür veritabanında tutmanın yararlı olacağı araştırılacaktır. Özellikle NoSQL veritabanlarında kullanılan fonksiyonlar ile klasik veritabanlarındaki SQL fonksiyonlarının incelenmesi amaçlanmaktadır.

VII. BİLGİLENDİRME

Geliştirilen yapı Akıllı Ders Yönetim Sistemine (ADYS - adys.nku.edu.tr) entegre edilmiş ve Namık Kemal Üniversitesi Çorlu Mühendislik Fakültesi tarafından 2015-2016 öğretim yılı Bahar döneminde kullanılmıştır. ADYS, 18.12.2013 tarihinden bu yana NKUBAP.00.17.AR.13.15 protokol nolu 'Akıllı Ders Yönetim Sistemi ile Programlama Ödevleri için İntihal Tespiti Uygulaması' başlıklı projemiz kapsamında NKU-BAP tarafından desteklenmiştir. Tüm desteklerinden dolayı üniversitemize teşekkür ederiz.

Çalışmadaki yazılım kodlarına ve veritabanı tablo yaratma SQL'lerine aşağıdaki web adresinden ulaşılabilir: http://bilgmuh.nku.edu.tr/erdincuzun/post/denormalizasyon-bir-anket-uygulamasi

KAYNAKÇA

[1] Codd, E. F. "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM 13 (6): 377–387. doi:10.1145/362384.362685, 1970.

[2] Chen, P. "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems 1 (1): 9–36. doi:10.1145/320434.320440, 1976.

[3] Finkelstein, S., Schkolnick, M. ve Tiberio, P. "Physical Database Design For Relational Databases". ACM Transactions on Database Systems, vol. 13, no. 1, pp. 91-128, 1988.

[4] Codd, E.F. "Further Normalization of the Data Base Relational Model". IBM Research Report RJ909, 1971.

[5] Harrington, Jan L. "Relational Database Design and Implementation: Clearly Explained". Elsevier-Morgan Kaufmann, Chapter 6, pp. 105-126, 2009.

[6] Powell, G. "Chapter 8: Building Fast-Performing Database Models". Beginning Database Design ISBN 978-0-7645-7490-0, Wrox Publishing, 2006.

[7] Mailvaganam, H. "Introduction to OLAP - Slice, Dice and Drill!". Data Warehousing Review. 2007.

[8] Cerpa, N. "Pre-physical data base design heuristics". Information Management, vol. 28, no. 6, pp. 351-359, 1995.

[9] Hanus, M. "To normalize or denormalize, that is the question". Proceedings of 19th International Conference for the Management and Performance Evaluation of Enterprise Computing Systems, San Diego, CA, 1994, pp. 416– 423, 1994.

[10] Rodgers, U. "Denormalization: why, what, and how?". Database Programming & Design, (12) 46–53, 1989.

[11] Coleman, G. "Normalizing not only way". Computerworld,(12) 63– 64, 1989.

[12] Nizam, A. "Veritabanı Tasarımı İlişkisel Veri Modeli ve Uygulamaları". Bölüm 9: Denormalizasyon, sy. 159 – 178, Papatya Yayıncılık, 2011.

[13] Ordonez, C., Maabout, S., Matusevich, D. S. ve Cabrera, W. "Extending ER models to capture database transformations to build data sets for data mining". in Data and Knowledge Engineering, vol 89, syf. 38-54, DOI: 10.1016/j.datak.2013.11.002, 2014.

[14] Lee, C.-H. ve Zheng, Y.L. "SQL-to-NoSQL Schema Denormalization and Migration: A Study on Content Management Systems". IEEE International Conference on SMC, pp. 2022 – 2026, DOI: 10.1109/SMC.2015.353, 2015.

[15] Ho, L.-Y., Hsieh, M.-J., Wu, J.-J. ve Liu, P. "Data Partition Optimization for Column-Family NoSQL databases", IEEE International Conference on Smart City/SocialCom/SustainCom together with DataCom 2015, pp. 668-675, DOI: 10.1109/SmartCity.2015.146, 2015.

[16] "NoSQL Relational Database Management System: Home Page". Strozzi.it. Erişim tarihi: 22.07.2016.