Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 22, No 1, 163-174, 2007

J. Fac. Eng. Arch. Gazi Univ.
Vol 22, No 1, 163-174, 2007

VERİTABANI BAĞIMSIZ UYGULAMA YAZILIMI GELİŞTİRME YÖNTEMİ

Yaşar Güneri ŞAHİN ve Halil İbrahim BÜLBÜL*
Bilgisayar Mühendisliği Bölümü, Mühendislik Mimarlık Fakültesi, Yaşar Üniversitesi, İzmir
* Bilgisayar Eğitimi Bölümüi Endüstriyel Sanatlar Eğitim Fakültesi, Gazi Üniversitesi, Ankara
yasar.sahin@yasar.edu.tr , bhalil@gazi.edu.tr

(Geliş/Received: 23.02.2006 ; Kabul/Accepted: 09 .11.2006)

ÖZET

Uygulama yazılımlarında veritabanı bağımsızlığı; veritabanında sadece tablolar halinde verilerin ve bu tablolara ait indekslerin bulunması, diğer tüm ilişki ve işlemlerin uygulama katmanında bir veri modülü (data module) üzerinden yapılması olarak tanımlanmaktadır. Bu yöntemle geliştirilmiş yazılımlar da veritabanı bağımsız uygulama yazılımı olarak adlandırılmaktadır. Yöntemin uygulanması ile veritabanı seçiminin işletmelere bırakılması, istenildiği an küçük çaplı veritabanı yönetim sistemlerinin kullanılması, veri yedekleme işleminin kolaylaştırılması ve veritabanı yönetim sistemi için gerekli uzman personel istihdamının azaltılması sağlanacaktır. Böylece hem veritabanı başlangıç hem de veritabanı bakım-onarım zaman ve yönetim giderlerinde bir düşüş elde edilmiş olacak ve KOBİ'lerin bilişim alanındaki ekonomik yükü azaltılmış olacaktır. Aynı zamanda veritabanı bağımsız geliştirilecek bir uygulama yazılımıyla, veritabanı yapısına esneklik getirilecek ve uygulama katmanından yapılan işlemlerle yönetilmesi kolaylaşacaktır. Bu çalışmada, var olan veritabanı yönetim sistemleri kullanımı incelenmiş ve bu veritabanları üzerinde aşamalarıyla birlikte, uygulama geliştirme araçları bileşenlerinin özellikleri kullanılarak, veritabanı bağımsız bir uygulama yazılımı geliştirme yöntemi sunulmuştur. Ayrıca her iki yöntemle (bağımlı ve bağımsız) geliştirilebilecek uygulama yazılımlarının maliyet ve performans açısından karşılaştırılması yapılmış ve avantaj ve dezavantajları sunulmuştur.

Anahtar Kelimeler: Veritabanı, uygulama yazılımı, yazılım geliştirme, veritabanı bağımsızlığı, KOBİ.

A METHOD FOR DATABASE INDEPENDENT APPLICATION SOFTWARE DEVELOPMENT

ABSTRACT

Database independency can be defined as: a platform that a database includes only data in form of tables and indexes, and all other relations and processes about data are loaded in and managed by a data module located in application software. If a software is developed using this method, it can be called as database independent application software. An effective result of this method is that selection of databases can be handed over to the enterprises. Thus, the usage of small sized database management systems, comfortable backup environment could be provided and the number of the employed specialized personnel that are necessary for a database management system could be decreased. Therefore, a decrease in the KOBI's economic burden can be achieved through the decrease in both initial costs and database maintenance service expenses. At the same time, an easy management can be provided for the database management systems with their carrying out of the relations at the application level and flexibility can be brought to the database structure by this independent database management system. In this study, usages of the existing database management systems are investigated and through using the features of the application development components and tools along with the levels on these databases a method of developing independent application software has been presented. Moreover, both methods (database independent and dependent) have been compared with the aspect of economical burden and performance, next advantages and disadvantages of the method have been presented.

Keywords: Database, application software, software development, database independency, SME.

1. GİRİŞ (INTRODUCTION)

Sayıları gittikçe çoğalan veritabanı yönetim sistemlerinin, maliyeti ve kullanımı için gerekli uzmanlık düzeyi çok farklılık gösterdiği için, uygulama yazılımı ile birlikte hangi veritabanı yönetim sisteminin kullanılacağını belirlemek, uygulamanın yapılacağı ortamdan çok başlangıç ve bakım maliyeti üzerine yoğunlaşmayı zorunlu kılmıştır.

Günümüzde veritabanları ile ilgili yapılan araştırmaların bir çoğu veri tabanlarının hızı, güvenliği, erişilebilirliği ve yapılacak işlemlerin hatasız olabilmesi üzerine yoğunlaşmıştır. Ancak satış politikaları nedeniyle, başlangıç ve bakım maliyetleri hep ikinci plana itilmiştir. Bazı büyük veritabanı yazılım üretici firmaları tarafından yapılan araştırma geliştirme (ARGE) çalışmaları, çoğunlukla maliyeti ikinci planda bırakıp bütün yükü son kullanıcıya yükleyerek, uygulama yazılımlarının bu duruma çözüm getirmelerini beklemişlerdir.

Özellikle, KOBİ'lerde maliyet öneminin ön planda olduğu durumlarda, geliştirilecek uygulamaların bakım ve işletme maliyetlerinin düşük seviyede olmasını gerektirir. Ayrıca, her veritabanı platformunda sorunsuz çalışabilmesi, istenildiği anda verilerin transfer edilerek başka bir veritabanı yazılımının kullanılması, verilerin güvenliğinin bozulmaması ve işlerin aksamadan gerçekleşmesi için uygulama katmanı bağımlı ve veritabanı bağımsız olarak planlanması büyük önem arz etmektedir.

Veritabanı bağımsız yazılım geliştirme, veritabanında sadece kaydedilmiş verilerin (tablolar halinde) ve indekslerin bulunması ve bu verilere ait ilişkiler, yapılar, tetiklemeler, sınırlandırmalar ve kısıtlamalar gibi veriye ait özelliklerin uygulama yazılımı katmanında bulundurulması ve yönetilmesi temeline dayanır. Veritabanından bağımsız yapılacak bir uygulama yazılımı, veritabanı yönetim sistemi seçme serbestliği getireceği gibi istenildiği anda yedek ve paralel veri depolanması olanağını da sunmuş olacaktır. Ayrıca, veritabanlarında sadece kaydedilmiş veriler ve indeksler bulunduğu için, verilerin veritabanları arasında aktarım sorunu ortadan kalkmış olacaktır [1,2].

Bu çalışmada; veritabanı bağımsız bir uygulama geliştirme yöntemi sunulmuştur ve veritabanı katmanında yapılan ve uygulama katmanında yapılabilecek işlemlerin, veritabanı katmanından uygulama katmanına nasıl aktarılabileceği gösterilmiştir. Ayrıca bu yöntem kullanılarak veritabanı bağımsız bir uygulama yazılımı geliştirilmiştir. Geliştirilen yazılım, standart SQL yapısı dışında kalan ve değişik veritabanı yönetim sistemlerine göre özel olarak üretilen PL/SQL, 4GL, DSQL gibi özellikli yapılar kullanılmadan yapılmış ve ortak bir platform oluşturmak hedeflenmiştir.

1.1. İlgili Çalışmalar (Related Works)

Veritabanı yönetim sistemi kullanımının yaygınlaşmasından başlayan ve günümüze kadar süren zaman içinde veritabanı iyileştirme, hızlandırma, kontrol yöntemleri gibi bir çok konu ile ilgili bir çok araştırma ve çalışma yapılmıştır. Genel olarak değerlendirildiklerinde bu araştırmalar var olan bilgilerden yola çıkılarak veritabanı yönetimini optimal kullanımını amaçlamaktadırlar.

Keller ve Basu çalışmalarında istemci-sunucu mimarisinde çalışan ilişkisel veritabanlarında yeni bir önbellek yöntemi geliştirmişlerdir. Bu yöntemle sunucu tarafında çalıştırılan tüm sorgulama sonuçlarının her istemcide oluşturulan önbellekte [3]; Altınel ve arkadaşları tarafından yapılan diğer bir çalışmada ise, önbellek tablosu (Cache Table) ismiyle bir veritabanı nesnesi geliştirilerek, uzaktaki veritabanı tablosunun kısmi ya da tüm içeriğinin devamlı olarak bu tabloda bulundurulması amaçlanmıştır. Böylelikle yapılacak tüm sorgulama işlemlerinin dinamik olarak eş zamanlı gerçekleşmesiyle sorgulama sırasında harcanacak zamanın azaltılması hedeflenmiştir [4].

İnternet üzerinde elektronik-iş uygulamalarında kullanılan veritabanları için yapılan bir diğer çalışmada, Bornhövd ve arkadaşları çok katmanlı ortamlarda, ara katman üzerinde yapılacak bir önbellek uygulaması üzerinde durulmuştur. Bu yöntemle herhangi bir ara katman üzerinde uygulanacak önbellek işleminin, hem katmana bağlı işlemlerin hızlanacağı hem de gereksiz başvuruların (transactions) önüne geçileceği ileri sürülmüştür [5].

Post ve Kagan tarafından yapılan bir çalışmada, veritabanı yöneticileri, uygulama yazılımcıları ve orta alan uzmanlarının çok farklı istekleri olduğunu ortaya koymuştur. Veritabanı yöneticilerinin bile, veritabanı konusunda uzmanlıklarının sınırlı olduğu ve veritabanı ile ilgili tüm bilgilere sahip olamadıkları belirtilmiştir. Aynı zamanda kullanılabilecek veritabanı yazılımının çokluğu sonucunda, hangi veritabanın seçilmesi gerektiğinin de ayrı bir sorun olduğu ortaya çıkartılmıştır. Çizelge 1'de 104 veritabanı geliştirici ve yöneticisi ile yapılan anketten elde edilen veritabanı yazılımı tercihleri ortalamaları verilmiştir. Tüm bu sonuçlar aslında

Çizelge 1. Veritabanı geliştiricileri ve yöneticileri VTYS tercihleri

Veritabanı Sistemi	Sayı	Önemlilik Bölümü		Oylama Bölümü	
		Ortalama	Std.Sap.	Ortalama	Std.Sap.
Access	29	6.90	3.98	5.00	3.45
Oracle	24	7.54	3.19	6.25	3.17
MS SQL Server	19	7.37	3.48	5.68	3.16
Fox Pro	18	6.89	3.95	6.22	3.70
Ingres	13	6.69	4.05	7.31	2.46
Omni	10	6.90	4.25	6.40	4.48
DBase	9	6.56	4.22	7.44	3.61
Informix	9	7.56	3.24	6.78	2.91
IBM DB2	9	7.67	3.12	7.78	1.56
Paradox	9	8.44	0.73	6.56	2.55
SyBase	7	5.57	4.12	5.86	3.80
Progess	3	6.67	5.77	6.00	5.29
Other	26	6.88	3.39	6.08	2.83

veritabanı bağımsız uygulama geliştirmenin önemini ortaya koymaktadır [6]. Mah ve Chung çalışmalarında, birçok kendine has özelliği olan özerk veritabanını birbirleriyle bir bütünlük sağlayacak şekilde çalışabilmesi için bir yöntem sunmuşlardır [7].

Diğer yandan Grufman ve arkadaşları ise, dağıtık ve çoklu ortamlarda birbirinden çok farklı veritabanı sistemlerinin ilişkilendirilip birlikte çalışmasını ve bu ortamda kullanılan bütünlük sınırlandırmalarının yerel olarak yapılabileceği görüşüne dayanan bir çalışma yapmışlardır.

Bu çalışmayı Aberdeen and Linköping Üniversitesi ile ortak oluşturulan bir çalışma grubuyla bir proje haline dönüştürmüşlerdir [8]. Veritabanı iyileştirme üzerinde yapılan diğer bir çalışmada ise Mayol ve Teniente bütünlük sınırlandırmalarının kontrolü ve bakışların güncellenmesi üzerine bir yöntem geliştirmişler ve bu yöntemle birlikte bütünlük sınırlandırmaları kontrolünde ortaya çıkan reddedilme durumunda yapılacak fazladan güncelleme işlemlerinin önüne geçmeye çalışmışlardır. [9,10].

Schewe çalışmasında, tetiklemeler sonucu ortaya çıkan sınırlandırma bütünlüğü sorununa yeni bir yaklaşımla bir yol oluşturarak çözüm bulmaya çalışmıştır. Geri dönülemeyen işlemler üzerinde etkili olabilecek bir yöntem olduğu belirtilen çalışmada hiper-grafik yöntemiyle ilişkilendirilmiş kritik-yol tanımı araştırılarak ortaya konmuştur [11].

Bütün bu çalışmalar diğer iyileştirme çalışmaları gibi ilgilendikleri alanlara yeni bir çok yöntem kazandırmasına rağmen başlangıç maliyetin düşürülmesi ve veritabanı sistemi işletme giderlerinin azaltılması konusundaki araştırmalara ışık yeterince ışık tutamamaktadır.

Parker "C" İçin Veritabanı Bağımsız Soyutlama Katmanı (Database Independent Astraction Layer for C) çalışmasında veritabanı bağımsız ve uygulama yazılımlarında kullanılabilecek bir katman tasarlamıştır. Ancak bu çalışma şu an kullanılan veritabanı bağlantı araçları olan ODBC, ADO, Midas gibi çalışmaktadır. Ayrıca C dilinde kendine has bir dil geliştirmiş ve bir sistem olmaktan çok bir alt program gibi çalışmakta olduğu için bir çok uygulama yazılımcısı tarafından kullanılabilir bir halde değildir [12].

Bu çalışmada, yukarıda belirtilen veritabanı ile ilgili çalışmaların dışında veri tabanı bağımsızlığının elde edilmesi planlanmış ve yapılan uygulama ve testlerde bunun gerçekleştirilebileceği görülmüştür.

2. VERİTABANI BAĞIMSIZ UYGULAMA YAZILIMI GELİŞTİRME YÖNTEMİ (A METHOD FOR DATABASE INDEPENDENT APPLICATION SOFTWARE DEVELOPMENT)

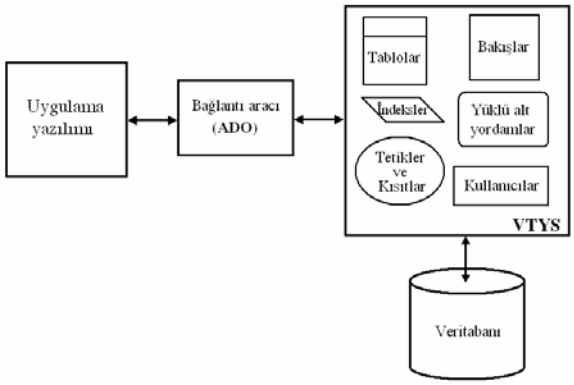
Veritabanı bağlantılı uygulama yazılımı geliştirme süreci, veritabanı yazılımı, yazılım geliştirme aracı ve veritabanına bağlantı nesnesinin seçimi ile başlayan ve uygun yazılım kodlarının üretilmesiyle devam eden bir süreçtir. Bu süreç içinde genelde iki genel aşama üzerinde işlemler yoğunlaşır. Bunlardan birincisi veritabanı üzerinde yapılacak işlemlerdir. Bu işlemler veritabanın yaratılması, tablo ve kullanıcıların tanımlanması, ön gerekliliklerin tanımlanması, bakışların ve tetiklerin oluşturulması gerekliyse saklı alt programların kodlanması gibi işlemleri içerir. İkinci aşamada ise, veritabanı yazılımında tanımlanmış bilgilerin uygulama katmanıyla birlikte kullanıcıya ulaştırılması ve kullanıcı tarafından girilen bilgi ve sorgulamaların veritabanına ulaştırılması için kullanılan kodlamaların uygulama katmanında oluşturulmasıdır. Bu aşamalar kullanılan platforma göre değişiklik gösterebilir. Örneğin internet üzerinde yapılan bir bağlantı varsa, bu iki ana katman arasına iletişim protokolleri gibi bazı diğer katmanlar eklenebilir. Ancak temel olarak ek katmanlar hazır nesnelerden oluştuğu için yapılacak olan kodlama işlemleri genelde iki ana katman olan veritabanı ve uygulama yazılımı katmanlarında yapılmaktadır.

2.1. Veritabanı Bağımsızlığı Nedir? (What Is Database Independency)

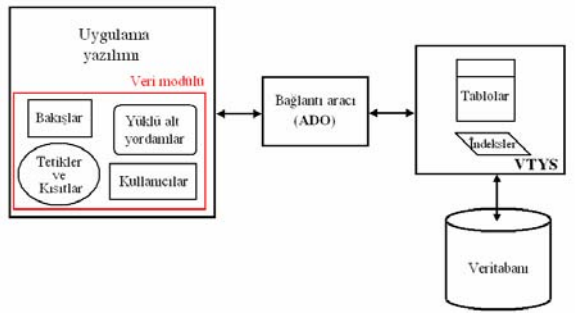
Veritabanı bağımsızlığı, bir uygulama yazılımının veritabanı yönetim sistemi gözetmeksizin üzerinde verilerin bulunduğu herhangi bir VTYS ile (tablolarda bulunan veri düzenin farklı olmaması koşuluyla) bir değişikliğe gerek duyulmadan birlikte çalışabilmesidir.

Veritabanı yönetim sistemleri, verilere ait ilişkilerin bulundurulduğu bir çok değişik yönteme sahiptir. Veritabanı bağımsız uygulama geliştirme yöntemi, VTYS'de bulunan ve uygulama katmanından da yapılabilecek bu yöntemlerin uygulama katmanından yapılması prensibine dayanmaktadır. Şekil 1'de veritabanı bağımlı bir uygulama yazılımı şeması görülmektedir.

Tablolar ve indeksler varitabanının bir parçasıdırlar. Bununla birlikte verilere ait diğer ilişkisel işlemler (Bakışlar, tetikler ve kısıtlar, yüklü alt yordamlar, kullanıcılar ve diğer bazı özellikler) VTYS tarafından yönetilebileceği gibi uygulama katmanında bulunan bir veri modülü (data module) tarafından da yönetilebilmektedirler.



Şekil 1. Veritabanı bağımlı bir uygulama yazılımı bloğu



Şekil 2. Veritabanı bağımsız bir uygulama yazılımı

Şekil 2'de veritabanı bağımsız bir uygulama yazılımı şeması görülmektedir. Burada, Şekil 1'de VTYS üzerinde bulunan bakışlar, yüklü alt yordamlar, kullanıcılar, tetikler ve kısıtlar bağımsız uygulamada uygulama katmanı içinde yer alan bir veri modülü içinde yer almaktadırlar. Veri modülü içinde yer alan bu ilişkisel özellikler uygulama katmanından da yönetilebileceği için VTYS üzerinde sadece tablolar ve indeksler vardır.

Veritabanı bağımsız uygulama yönteminde, tabloları ve indeksleri içeren herhangi bir ilişkisel veritabanı yönetim sistemi uygulama bloğunda kullanılabilecektir. Böylece hem veritabanı seçimi serbestliği hem de veritabanı yönetimi uzmanlığı gerektiren durum ortadan kaldırılmış olacaktır.

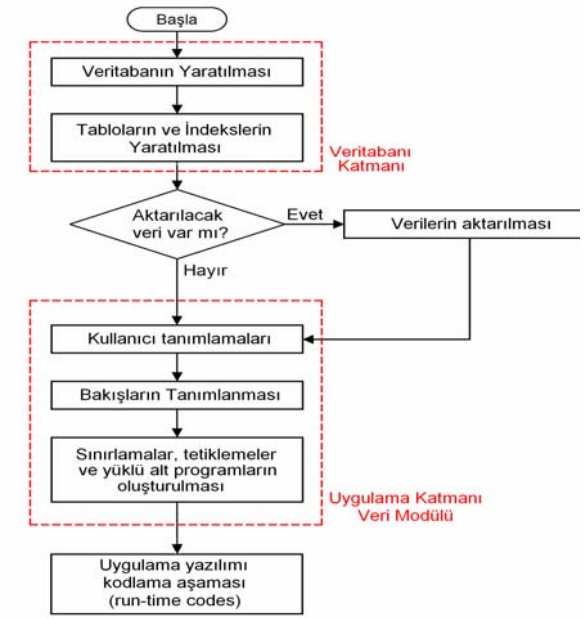
2.2. Veritabanı Bağımsız Uygulama Yazılımı Geliştirme Öncesi İşlemler (Required Processes Before Developing A Database Independent Application Software)

Bu bölümde dikkat edilmesi gereken, veritabanı bağımlı uygulamalarda veritabanı yaratılmasından-sınırlamalar, tetikler ve yüklü alt programlara kadar olan tüm adımlar veritabanı katmanı üzerinde yapılıyor olmasıdır. Tüm veritabanı katmanında yapılan işlemler, veritabanı bağımlı olmayı gerektirir ve böyle bir yapının kullanıldığı uygulama yazılımı projelerinde herhangi bir VTYS değişikliği (aynı VYTS kullanılması durumunda da) söz konusunda olduğunda, tüm adımların, yeni VYTS'ye göre düzenlenmesi gereklidir. Veritabanı bağımsız uygulama yöntemi bu zorunluluğu ortandan kaldırmaktadır. Şekil 3'te veritabanı bağımsız uygulama yöntemi ile üretilecek uygulama yazılımı kodlaması (run-time codes) öncesi veri ile ilgili işlem sırası verilmiştir.

Burada veritabanı yaratma, tablo ve indekslerin oluşturulması haricindeki tüm veri ile ilgili işlemlerin uygulama katmanında bulunan veri modülü içinde yapıldığı görülmektedir. Böylelikle herhangi bir VTYS değişimi söz konusu olduğunda, veritabanı katmanında yapılması gereken işlem çok azalmış olacak ve fazladan bir işlem yapma olasılığı ortadan kalkmış olacaktır. Bu özellikten dolayı uygulama yazılımı veritabanı bağımsız olarak nitelendirilmektedir.

2.3. Veri Modülü Yaratılması ve Veritabanı Bağlantısının Sağlanması (Creation of Data Module and Establishing the Database Connection)

Öncelikli olarak veri modülü, uygulama katmanı içinde bağımsız bir modül olarak tasarlanmalıdır. Buna ek



Şekil 3. Veritabanı bağımsız uygulama yöntemi ile üretilecek uygulama yazılımı kodlaması (run-time codes) öncesi veri ile ilgili işlem sırası

olarak içinde kullanılacak SQL, tablo ve bağlantı gibi bileşenlerin veritabanı bağımsızlığı sağlayabilmesi için özellikle parametrik tanımlamalar yapılmalıdır. Yapılacak ilk işlem uygulama katmanındaki veri modülünün ayrı bir modül olarak bir ünite halinde oluşturulmasıdır. Bu işlemden sonra veri modülünün içine veritabanı ile bağlantıyı sağlayacak olan veritabanı bileşenlerinin yerleştirilmesidir. Bu modülde kullanılabilecek bileşenler VT bağlantı bileşeni, genel sorgulamalar için kullanılacak SQL bileşenleri, tablolar üzerinde işlemlerinin (ekleme, silme, güncelleme vb) yapılabileceği tablo bileşenleri, saklı alt yordamların ve fonksiyonların yönetileceği SaklıAltYordam bileşeni, veri kümeleri üzerinde işlem yapmayı sağlayan VeriKümesi bileşeni ve verilerin taşınmasını sağlayacak VeriKaynağı bileşenleridir (bu bileşenler Borland Delphi ve C++ Builder uygulama geliştirme araçlarında görülebilir).

Kullanılacak bileşenlerin sayısı, veritabanı üzerinde bulunan tablo sayısı ve yapılması düşünülen işlemler miktarıyla doğru orantılıdır. Örnek olarak 10 adet tablo bulunan bir veritabanı için en az 10 adet SQL bileşeni ya da tablo bileşeni kullanılmalıdır. Tüm tablolar için tek bir SQL ya da tablo bileşenin kullanılması uygulama çalışma sırasında (run-time) bileşen bilgileri değişikliği gerektireceği için sistemin performans kaybını neden olacaktır, bu nedenle her tablo için ayrı bir SQL bileşeni kullanılmalıdır.

2.4. Veritabanı Kullanıcı, Rol Tanımları ve Bakışların Yönetimi (Managing Users, Roles and Views)

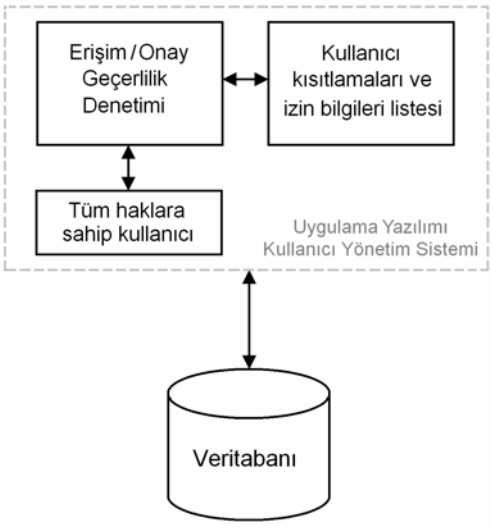
Veritabanı güvenliği ve kolay kullanımı açısından çok önemli iki unsur olan kullanıcı ve roller, genellikle veri tablolarının erişim ve veri tabanı üzerinde yapılacak diğer işlemlerin kişilere göre sınırlandırılması için kullanılırlar. Kullanıcıların ve rollerinin tanımlanması,

daha önce anlatılan veritabanı ve veri tablosu yaratılmasında kullanılan üç yöntemle de yapılabileceği gibi, uygulama katmanında da veritabanına ihtiyaç duymadan tanımlanması mümkündür.

Kullanıcı denetimlerinin uygulama katmanından yapılması: mevcut genel kullanıcı özelliğiyle birlikte diğer tüm özel kullanıcılara ait giriş/çıkış izinlerinin ve onay belgelerinin bir listede tutulması ve bu kullanıcıya ait bir istek sonucunda, isteğe bağlı izinin bu listeden kontrol edilerek verilmesi ya da reddedilmesi ile anlamına gelmektedir. Bu yöntemin kullanılmasıyla, gerekli izinlerin ya da onayların uygulama katmanından yapılması sonucunda veritabanına yapılan başvuru süresinde belirgin bir azalma söz konusu olacaktır. Böylelikle hem bağlantı başvuru trafiği azalmış hem de hız açısından bir fayda sağlanmış olacaktır. Bununla birlikte uygulama katmanından yönetilecek bir kullanıcı denetimi ile veritabanı yönetim sistemleri tarafından sağlanmayan bazı özellikler de (yazdırma/listeleme/ giriş-çıkış onayı vb.) denetlenebilecek ve çok daha geniş bir güvenlik ağı yerleştirme imkanı sağlanmış olacaktır.

Şekil 4'te uygulama katmanı tarafından yapılan kullanıcı yönetim sistemi şeması verilmiştir. Veritabanında bulunan veri tabloları üzerinde yapılabilecek işlemler için belirlenmiş kriterlerle (yazma/silme/değiştirme) birlikte, uygulama içinde kullanılan diğer bazı özelliklere ait kullanıcı istekleri, kullanıcının işlem başvurusu, daha önceden belirlenmiş kullanıcı kısıtlamaları ve izin bilgileri listesindeki bilgilerle karşılaştırılır.

Bu listede başvuruya ait izin bilgisi olumlu ise kullanıcının başvurusu veritabanına tüm haklara sahip bir kullanıcı başvurusu gibi gönderilir. Kullanıcı başvurusu liste tarafından onaylanmayan bir başvuru ise, denetim sonucunda veritabanına başvuru yapılmadan direkt olarak kullanıcıya isteğinin kısıtlamalar doğrultusunda reddedildiği mesajı gönderilir. Böylece, izin alınmayan bir başvuru için veritabanı hem meşgul edilmemiş hem de gerekli mesaj uygulama yazılımının kullanıcısına kendi dilinde bildirilmiş olacaktır.



Şekil 4. Uygulama katmanı tarafından yapılan kullanıcı yönetim sistemi

2.5. Tetiklerin Yaratılması ve Yönetilmesi (Managing and Creating Triggers)

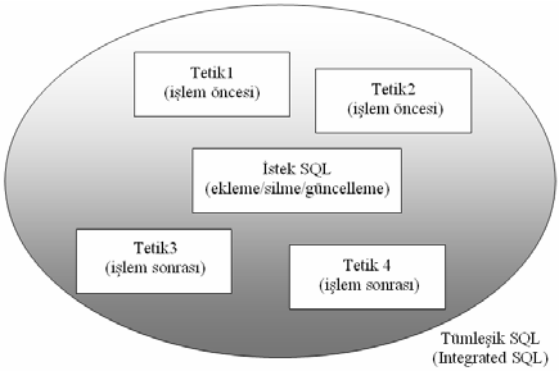
Bu yöntem kullanılarak yapılacak tetik yönetimi, tetiklerin veritabanı üzerinde tanımlanmadan tetik gerektiren sorgu ifadesi kullanımı öncesi ya da sonrası tetik sorgusunun çalıştırılması esasına dayanır [13,14]. Diğer bir deyişle, tablo üzerinde yapılacak bir değişiklik diğer bir tablo üzerinde değişiklik gerektiriyorsa, iki değişiklik için gerekli SQL cümlelerinin tanımlanmasının ardından öncelikli olarak çalıştırılacak cümlenin ardından diğer cümlenin çalıştırılmasıdır. Her iki cümlenin de çalıştırılması ile hem orijinal işlem hem de tetik işlemi gerçekleştirilmiş olacaktır.

Tetik işlemi ile orijinal işlem ayrı SQL cümlelerinde tanımlanmış olabileceği gibi SQL iyileştirmesi ile bir SQL cümlesi ile de gerçekleştirilebilmektedir. Böylece, bir başvuru süresince iki işlemin de halledilmesi mümkün olacaktır. Şekil 5'te tetik gerektiren bir SQL işlemi için (kayıt ekleme, kayıt silme ya da kayıt üzerinde değişiklik vb) uygulanabilecek tümleşik SQL yapısı görülmektedir. Burada görülebileceği gibi, başvuruda bulunulan bir işlemle birlikte uygulanabilecek tetiklerin bir arada çalıştırılması ya da başvuruda bulunulan işlem ve tetiklerin özelliğine belirli bir hiyerarşide SQL'lerin çalıştırılması mümkündür. Şekil 5'te görülen tetiklerden Tetik1 ve Tetik2 öncelikli (Before) tetik olduğu için başvuruda bulunulan SQL cümlesinden önce çalıştırılmalı ya da tümleşik SQL cümlesinde ilk sıraları almalıdır. Aynı şekilde Tetik3 ve Tetik4 sonda (After) uygulaması olduğu için başvuru SQL cümlesinden sonra yada tümleşik SQL cümlesinin sonunda yer almalıdır.

Bir diğer yol olarak kullanılabilecek tetik yönetimi ise, tetik işlemi gerektiren durumlarda SQL'leri birbiri ardına çağıracak bir fonksiyon tanımlamaktır. Bu yöntemde uygulama katmanında tetikler farklı birer fonksiyon olarak da tanımlanabilirler. Farklı tanımlanmış fonksiyonlar birbiri ardına sıra ile çağırılarak tetik işlemleri gerçekleştirilebilmektedir.

2.6. Sınırlamaların Yönetilmesi (Managing Constraints)

Sınırlamalar, çok çeşitli biçimlerde kullanılabileceği için veritabanı yönetim sistemi üzerinde ayrıca tanım-



Şekil 5. Tetiklerle birlikte çalıştırılan tümleşik SQL yapısı

lanabilen bir yere sahip değildirler. Bu nedenle veritabanı yönetim sistemleri, kullanılacak sınırlama tipine göre, SQL sorguları ya da ön gerekliliklerle bu işlemlerin yapılmasına olanak sağlamışlardır.

Sınırlamalar, içsel ve dışsal (internal and external) olmak üzere iki ana başlık altında toplanırlar [15,16]. Dışsal olarak nitelendirilen sınırlamalar uygulama yazılımı tarafından çok rahat bir şekilde kullanılan sınırlamalardır. Bunlara ek olarak içsel sınırlandırmalar olarak nitelendirilen, kalıtımsal (inherent), dolaylı (implicit) ve açık (explicit) sınırlandırmalar da veritabanı yönetim sistemlerinde kullanılmaktadırlar [15-17]. Bu tanımlara ek olarak sınırlandırmaları bağımsız (independent): veritabanı yönetim sistemi içinde herhangi bir gereksinime ihtiyaç duymayan ve bağımlı (dependent) olarak iki gruba ayırmak da mümkündür.

2.6.1. Bağımsız tip sınırlamaları yönetimi (Managing independent type constraints)

Bağımsız sınırlamalar, verinin giriş düzenine göre bütünlüğünün sağlanması için kullanılan ve diğer hiçbir tablo ya da veri işlemine bağımlı olmayan sınırlamalardır. Bunlara örnek olarak, sayılar için sadece rakamların girilmesi veya 1 haneli bir üye numarası için 10 karakterden az ya da fazla karakter girilmemesi zorunluluğu gösterilebilir.

Şekil 6'da bağımsız sınırlama tipinde olan giriş alan denetimi kısıtları için giriş kullanılabilecek bir maskesi şeması verilmiştir. Maske, giriş alanlarının karakteristik özelliklerini belirtir izin şemasıdır. Diğer bir deyişle, maske biriminin gösterdiği alana girilmesine izin verilecek karakterlerin özelliklerini belirtir. Şekil 6'daki şemada A ile gösterilen değerler alan içinde bulunan karakterleri, M ile gösterilen değerler ise alanın o karakterine ait özelliği belirten maske birimidir. Şekil 6'da gösterilen maskeye göre M ile gösterilen maske biriminde hangi tip karakterler belirtilmiş ise A ile gösterilen o alana sadece o karakterin girilmesine izin verilir.

Giriş alan sınırlamaları için kullanılabilecek bir diğer yöntem ise, giriş tuşları denetimidir. Alan üzerinde bulunurken sadece girilmesine izin verilen karakterlerin basılmasına izin vermekte ve diğer basılan tuşlar göz ardı edilmektedir. Böylece alana girilebilecek karakterler daha girilmeden kontrol edilmiş olacak ve hata oluşmadan engellenmesi sağlanacaktır.

2.6.2. Bağımlı dış sınırlama yönetimi (Managing dependent explicit constraints)

Bu tip sınırlamalar, tetiklemeler dahil tüm veri yapısını ilgilendiren sınırlamalardır. Salt veri bulunan bir

Şekil 6. Giriş alanı için maske kullanım şeması

veritabanı yapısıyla çalışan bir uygulama yazılımı ile sınırlama yönetimi, kullanılan veri işleme tekniklerine göre değişiklik gösterirler. Örneğin, bir veri tablosundan bulunan bir verinin değiştirilebilmesi ya da yeni bir verinin eklenebilmesi için başka tablolardaki yada aynı tablodaki diğer verilere ihtiyaç duyulması ve bu verilerin elverdiği ölçüde işlemlerin yapılabilmesi için birden fazla işlem yapılması gerekir. Bu gereklilik için kullanılabilecek değişik yöntemler, sistem performansını düşürebileceği gibi yükseltebilecek nitelikte de olabilirler.

Küçük ve orta ölçekli işletmelerde kullanılabilecek ve dağıtılmış olmayan veritabanı sistemleri için en uygun bağımlı sınırlama yönetimi, uygulanacak işlemle birlikte kontrollerin yapılması yöntemidir. Bu yöntem tümleşik SQL yapısıyla birlikte uygulanmasıyla, sistem performansında oluşabilecek hız düşüklüklerinin de bir ölçüde önüne geçilmiş olacaktır.

2.7. Saklı Alt Programların Yönetilmesi (Managing Stored Procedures)

Bu alt programların amacı veri işleme ve denetim mekanizmasına işlerlik ve hız kazandırmaktır. Bir çok veritabanı yönetim sistemi tarafından tanınmasına rağmen bazı veritabanı yönetim sistemleri alt programlardan yararlanmaz ve bu alt programların uygulama katmanından yapılması zorunluluğunu getirir.

Genel olarak veritabanı bütünlüğüne bakıldığında alt programlar, tetikler ve bütünlük sınırlamaları aynı amaca hizmet eden ve şekilsel farklılıklarla birbirlerinden ayrılan yöntemlerdir. Saklı alt programların uygulama katmanından da yönetimi mümkündür. Uygulama katmanından yapılacak alt program yönetimi, PL/SQL gibi özellikli dillerin kullanımı yerine, uygulama geliştirme aracının kendi diliyle yazılacağı için çok kolay ve daha verimli çalışmayı sağlayacaktır. Bu getirdiği kolaylığa rağmen, sistem performansında bir düşüşe neden olabilecektir. Ancak bu düşüş veri trafiğinin yoğun olmadığı küçük veri yapıları için önemli bir unsur olmayacaktır.

Uygulama katmanından yapılacak bir saklı alt program, geliştirme aracına ait alt program özelliklerini kullanarak, SQL cümleleri ya da SQL cümlesi yerine geçebilecek geliştirme aracına ait bileşenlerin kullanımıyla isteklere ait gerekli sonuçların elde edilmesini sağlamalıdır. Bu işlem, hem SQL cümle yapısının ve bileşene ait özelliklerin hem de rutin denetleme işlemlerinin tümleşik bir şekilde kullanılmasıyla mümkündür.

2.8. Uygulama Yazılımı Kodlama Yönetimi (Managing Run-Time Codes)

Veritabanı bağımsız uygulama yazılımı geliştirme sürecinde, veritabanı işlemlerinin bitmesiyle birlikte, uygulama yazılımı içinde kullanılacak ve veritabanı işlemlerini kolaylaştıracak kodlama işlemleri büyük önem taşımaktadır. Kodlama işlemi yapılırken kullanılacak alt programlardan-yazılımın diline, SQL cümlelerinden-

form yapısına kadar bir çok alanda dikkat edilmesi gereken kurallar olmalıdır. Bu kuralların bazıları standart yazılım geliştirme süreci kurallarını da içerir. Bu bölümde bu kurallar kısaca anlatılacaktır.

Bu kurallara ek olarak yazılım mühendisliği için geliştirmiş SPICE - ISO 15504 (Software Process Improvement and Capability Determination), IEEE 12207 ve yazılım süreç iyileştirme yöntemleri (benimseme süreci, yeterli ve nitelikli kaynak yönetimi) gibi bazı yazılım mühendisliği standartlarında uyulması da verimliliği artıracaktır [18,19].

2.9. Bileşen Kullanımı (Component Usage)

Nesneye yönelik programlama tekniği gelişimiyle, üretilen yazılım geliştirme araçlarının tümü, bileşen (component) tabanlı kullanımı sunmuştur. Bileşenler yazılım kodlama sürecini düşürdükleri gibi aynı zamanda da kolay kullanımlarıyla uygulama yazılımlarına işlevsellik, değişkenlik ve esneklik kazandırmışlardır. Özellikle, nesneye yönelik programlama ile oluşturulmuş bileşen sınıfları ile veritabanı bağlantısı ve üzerinde işlem yapma oldukça kolaylaşmıştır. SQL cümleleri yerine bağlantı nesnesiyle kullanılan bileşenlerin özellikleri kullanılması, bazı durumlarda hem sistem performansının artması hem de kullanım kolaylığı sağlamaktadır.

3. VERİTABANI BAĞIMSIZ UYGULAMA YAZILIMI MALİYET ANALİZİ VE PERFORMANS TESTİ (COST ANALYSIS ANF PERFORMANCE TEST FOR DATABASE INDEPENDENT APPLICATION SOFTWARE)

Veritabanı bağımsız geliştirilmiş bir uygulama yazılımı ile veritabanı bağımlı geliştirilmiş bir uygulama yazılımı arasında hem geliştirme hem de bakım-onarım süreci içinde maliyet, insan kaynakları ve süre açısından bir çok değişiklik vardır. Bu değişikliklerin avantaj ve dezavantajlarıyla incelenmesi hangi yöntemin hangi ortamda daha kullanışlı olduğunun anlaşılması bakımından önemlidir.

Yazılım geliştirme süreci, yazılımın kullanılması düşünülen sektör ve büyüklüğüne göre değişen ve planlama ile başlayarak ve bunu takip eden, analiz, geliştirme, uygulama ve bakım onarım aşamaları ile devam eden bir süreçtir. Bu süreç içinde kullanılacak insan gücü ve bunlar için gerekli süre ve maliyet, sektörün büyüklüğü ve yapılacak işlemlerin boyutuna göre belirli yöntemlerle hesaplanmalı ve uygulama süreci bu işlemlere göre takip edilecek yöntemi belirleyerek başlatılmalıdır.

3.1. Veritabanı Bağımsız Yazılım Maliyet Analizi (Cost Analysis of Database Independent Application)

Bu bölümde yazılım sektöründe çalışan firmalardan elde edilen gerçek verilerle yapılan maliyet analizleri ve oluşturulan tablo bilgileri doğrultusunda, KOBİ'ler için veritabanı bağlantılı uygulama yazılımı geliştirme süreçleri ve veritabanı yönetim sistemleri seçimi, başlangıç ve bakım onarım maliyetlerinin değişik yöntemler kullanarak nasıl azaltılabileceği sunulmuştur. Ayrıca veritabanı uzman personeli istihdamı yapılmadan geliştirilebilecek veritabanı bağımsız uygulama yazılımı yönteminin maliyet açısından getirileri verilmiştir. Bu bilgiler doğrultusunda veritabanı bağımsız uygulamaların KOBİ'lerde kullanımı ile tam zamanlı ve/veya yarı zamanlı uzman personel istihdamı gerekliliği ortadan kalkmasına bağlı maliyet düşüşü ve işletme yönetiminde basitleşmesi işaret edilmiştir.

3.2. Örnek Yazılım Üretim Aşamaları ve Fiyatlandırması (Sample Development Steps and Pricing)

Yazılım, kullanılacağı işletmenin gereksinimlerinden, test ve uygulama aşamasına kadar bir çok aşama sonrasında ortaya çıkartılmaktadır. Her aşama kendine has özelliğe ve fiyat politikasına sahiptir ve fiyatlandırma, sabit fiyat (fixed price) yöntemi, zaman-materyal (time-material) yöntemi ya da her ikisinin karışımı bir yöntem kullanılarak elde edilir [29].

Ticari uygulama yazılımları kapsam olarak çok farklılık göstereceği için, bu bölümde hesaplaması yapılan ticari yazılım paketi sadece stok takibi, cari takip (borç-alacak takibi) ve fatura modülü ile sınırlandırılmıştır. Ayrıca veritabanı fiyatları ve bakım onarım giderleri ayrı ayrı hesaplanmıştır. Çizelge 2'de piyasada bulunan bazı veritabanı yönetm sistemleri lisans ve bakım onarım giderleri verilmiştir.

Çizelge 3'te veritabanı bağımsız üretilecek ticari işletme yazılımı için gerekli işlemler ve bu işlemlere ait süre ve fiyat değerleri verilmiştir. Ortalama bir yazılımcının günlük yazılım geliştirme ücreti 100$ olarak belirlenmiştir. Bu değerler göz önüne alınarak, yazılım ilk değerlendirme aşamasından test aşamasına kadar geçen süre 60 iş günü ve 100$ adam/gün fiyat baz

Çizelge 2. Veritabanı sunucu ve istemci lisans ve bakım onarım ücretleri [20-28].

VTYS	Sürüm	VT Boyut	Sunucu+ 10 İstemci Lisans[1]	Yıllık bakım ücretleri [1]	Toplam Maliyet
DB2	OLAP S.	Büyük Ölçek	4 610$	1 000$	5 510$
Oracle	9i Std.Edition	Büyük Ölçek	3 000$	660$	3 660$
Microsoft SQL Serv.	2000	Büyük Ölçek	2 249$	1 000$	3 249$
Informix	IDS Ent.Edition	Büyük Ölçek	2 065$	300$	2 365$
SyBase	ASE v8.0	Büyük Ölçek	999$	1 000$	1 999$
MySQL	Pro.	Orta-Alt	495$	0$	495$
Microsoft Access	2003	Orta-Alt	247$	0$	247$
PostgreSQL	7.4.3	Orta-Alt	Ücretsiz	Ücretsiz	Ücretsiz
Paradox	7.0	Alt Ölçek	Ücretsiz	Ücretsiz	Ücretsiz
				Ortalama	1 947$

[1]: Fiyatlar Mart 2005 fiyatlarıdır.

Çizelge 3. Veritabanı bağımsız ticari işletme yazılımı için gerekli işlem, süre ve fiyat değerleri

Yapılan İşlem (Açıklama)	Süre (Gün)	Toplam	Ücret ($)	Toplam
İhtiyaçların belirlenmesi aşaması	7	7	700	700
Analiz aşaması	7	14	700	1 400
Planlama aşaması	7	21	700	2 100
Tasarım aşaması	((15))**	21	((1 500))**	2 100
Veri tabanı seçimi ve tasarımı	(7)*	21	(700)*	2 100
Veri tablolarının belirlenmesi	2	23	200	2 300
Sahaların ve kayıt desenlerinin belirlenmesi	1	24	100	2 400
İndeks ve anahtar alanların belirlenmesi	1	25	100	2 500
Tablolar arası ilişkilerin kurulması	1	26	100	2 600
Tetikleyici ve prosedür kodlarının tasarımı	1	27	100	2 700
Veri tabanı seçimi	1	28	100	2 800
Ara yüzlerin tasarımı ve geliştirme araçları	(8)*	28	(800)*	2 800
Yazılımın adı	-	28	-	2 800
Simge tasarımı	-	28	-	2 800
Kapak resmi	1	29	100	2 900
Kullanılacak derleyici ya da yorumlayıcı tespiti	-	29	-	2 900
Kodlama ve iş akış diyagramları hazırlığı	2	31	200	3 100
Veri giriş ekranlarının tasarlanması	2	33	200	3 300
Sorgulama ekranlarının tasarlanması	1	34	100	3 400
Yazıcı ve ekran raporlarının tasarlanması	2	36	200	3 600
Kodlama aşaması	14	50	1 400	5 000
Test aşaması	5	55	500	5 500
Kurulum ve uygulama başlangıç aşaması	5	60	500	6 000

alındığında toplam fiyat 6 000$ olacaktır. Çizelge 4'de ise veritabanı bağımlı üretilecek Ticari yazılım için gerekli ücret ve zaman bilgileri verilmiştir.

Bu çizelgede ise yine yazılım kodlama için 100$ adam/gün fiyat ve veritabanı için 150$ adam/gün baz alındığında toplam fiyat 6 350$ toplam süre ise 53 iş günü olacaktır.

Çizelge 3 ve Çizelge 4'te görüleceği gibi, veritabanı bağımlı üretilecek bir yazılım, veritabanı bağımsız üretilecek bir yazılıma göre paralel çalışma imkanından dolayı daha kısa sürede bitecektir. Ancak Çizelge 3 ve Çizelge 4'te gösterildiği gibi veritabanı bağımlı uygulama üretimi daha pahalı olacaktır. Bunlara ek olarak, üretim sırasında kullanılan personelin bakım ve onarım için istihdam edilmesi durumunda, veritabanı bağımsız uygulama için 1 diğeri için 2 personel bulundurma zorunluluğu vardır.

3.3. Örnek Yazılım Toplam Maliyeti (Total Cost of Sample Application)

Bu bölümde, önceki bölümde verilen değerler kullanılarak, veritabanı bağımlı ve bağımsız üretilecek iki adet yazılım için bir yıllık bakım onarım giderleri dahil gerekli personel ve parasal maliyetler verilmiştir.

Çizelge 5'de küçük ve orta büyüklükteki bir işletmede istihdam edilebilecek yazılım ve veritabanı uzmanı için gerekli yıllık personel maliyetleri, Çizelge 6'da ise bu

Çizelge 4. Veritabanı bağımlı ticari işletme yazılımı için gerekli işlem, süre ve fiyat değerleri

Yapılan İşlem (Açıklama)	Süre (Gün)	Toplam	Ücret (US$)	Toplam
İhtiyaçların belirlenmesi aşaması	7	7	700	700
Analiz aşaması	7	14	700	1 400
Planlama aşaması	7	21	700	2 100
Tasarım aşaması	((8))**	21	((800))**	2 100
Veri tabanı seçimi ve tasarımı	**(7)***	**21**	**(1 050)***	2 100
Veri tablolarının belirlenmesi	**2**	**21**	300	2 400
Sahaların ve kayıt desenlerinin belirlenmesi	**1**	**21**	150	2 550
İndeks ve anahtar alanların belirlenmesi	**1**	**21**	150	2 700
Tablolar arası ilişkilerin kurulması	**1**	**21**	150	2 850
Tetikleyici ve prosedür kodlarının tasarımı	**1**	**21**	150	3 000
Veri tabanı seçimi	**1**	**21**	150	3 150
Ara yüzlerin tasarımı ve geliştirme araçları	(8)*	21	(800)*	3 150
Yazılımın adı	-	21	-	3 150
Simge tasarımı	-	21	-	3 150
Kapak resmi	1	22	100	3 250
Kullanılacak derleyici ya da yorumlayıcı tespiti	-	22	-	3 250
Kodlama ve iş akış diyagramları hazırlığı	2	24	200	3 450
Veri giriş ekranlarının tasarlanması	2	26	200	3 650
Sorgulama ekranlarının tasarlanması	1	27	100	3 750
Yazıcı ve ekran raporlarının tasarlanması	2	29	200	3 950
Kodlama aşaması	14	43	1 400	5 350
Test aşaması	5	48	500	5 850
Kurulum ve uygulama başlangıç aşaması	5	53	500	6 350

* Bu miktarlar alt başlık altında yer alan süre ve ücretlerin toplamını göstermektedir.
** Bu miktarlar başlık altında yer alan alt başlıkların toplamını göstermektedir.

Çizelge 5. Tam zamanlı personel yıllık giderleri ($)[30]

Personel Niteliği	Yıllık Taban Ücret	Vergi	SSK Primleri	Yemek +Yol	Toplam Maliyet
Uygulama Uzmanı	12 400	2 400	2 965	2 480	20 245
Veritabanı Uzmanı	15 500	2 840	3 670	2 480	24 490

Çizelge 6. Bakım onarım için ayrılacak yarı zamanlı personel giderleri

Personel Niteliği	Yıllık Bakım-Onarım Gün Sayısı	Gün Ücreti ($)	Toplam Maliyet ($)
Uygulama Uzmanı	25	100	2 500
Veritabanı Uzmanı	20	150	3 000

işletmenin yazılımı bir firmaya yaptırması durumunda personel istihdamı yapmadan yıllık yazılım için gerekli bakım-onarım maliyetleri verilmiştir.

Tüm bu bilgiler ışığında, üretilebilecek ticari işletme yazılımı için gerekli maliyetler, üretim şekillerine ve veritabanı ortalama maliyetine göre Çizelge 7'de verilmiştir. Çizelge 7'de görülebileceği gibi, veritabanı bağımsız üretilecek bir yazılım hem personel istihdamı hem de bakım-onarım anlaşmalı yarı zamanlı çalışma ile üretim olsun her iki durumda da daha düşük maliyetli olacaktır. Aynı şekilde veritabanı bağımsız üretilecek bir yazılım, hem ücretli veritabanı kullanımı hem de ücretsiz veritabanı kullanımı söz konusu olduğunda daha düşük maliyetli (%30-%50 arası) olacaktır.

Ücretsiz ve düşük ölçekli veritabanlarının kullanımı, özellikle finansal bilgilerin kullanıldığı ve bilgilerin önemli olduğu durumlarda çok kullanışlı olmayacaktır. Özellikle istemci-sunucu mimarisi kullanılan ortamlarda veritabanlarının güvenliği söz konusu olduğunda, düşük ölçekli ve ücretsiz veritabanları kullanımı istemci tarafta bir sorgu ya da ekleme işlemi gerçekleştirilirken istemcinin işletim sisteminden kaynaklanan kilitlenme durumuna geçmesinde verilerin kaybolma riski söz konusu olacaktır. Bu bilgiler ışığında ve riskler düşünülerek kullanılacak veritabanlarının seçilmesi daha doğru olacaktır.

3.4. Veritabanı Bağımsız Yazılım Performans Testi
(Performance Test for Database Independent Application Software)

Verim ve gecikme performans ölçümlerinin önemli göstergelerindendir. Belirli bir miktarda verinin geri döndürüldüğü durumda, verim belirli bir zaman dilimi (genellikle bir saniye) içinde işlenen istemci isteği (transaction) sayısıdır. Gecikme ise istekle, isteğe karşılık gelen verinin geliş süresinde geçen zamanın göstergesidir [31]. Veritabanı performans testleri yapılırken önemli konulardan birisi birim maliyete düşen iş miktarıdır [32]. Bu bölümde veritabanı bağımsız geliştirilmiş bir uygulama yazılımı ile veritabanı bağımlı geliştirilmiş bir uygulama yazılımı arasında sistem performansını ölçmek için gerçekleştirilmiş bazı test sonuçları sunulmuştur. Bu test sonuçları ücretsiz elde edilebilecek (VTYS Lisansı kullanılmadan sadece VT kullanıldığı için) "Microsoft Access", "Oracle 8.1.7i Personal" ve "Microsoft SQL Server 2000 Personal" veritabanları üzerinde elde edilen test sonuçlarıdır. Test ortamında kullanılan donanım, yazılım ürünleri ve veriler Çizelge 8'deki gibi listelenmiştir:

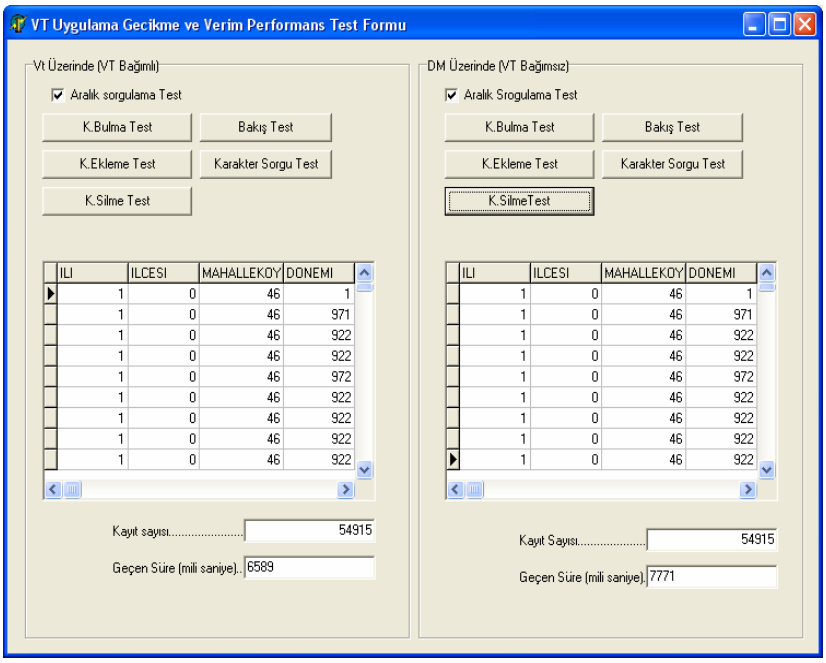
Çizelge 7. Ticari işletme yazılımı için üretim şekillerine ve veritabanı maliyetine göre fiyat listesi ($)

Yazılım Üretim, Bakım Onarım Cinsi	Veritabanı Ücreti	Yazılım Ücreti	Personel Giderleri	Bakım Onarım Giderleri	Toplam Maliyet
Yıllık personel istihdamı ile üretilecek veritabanı bağımlı yazılım	1 947[1]	0	44 735	0	46 682
Yıllık personel istihdamı ile üretilecek veritabanı bağımsız yazılım	1 947[1]	0	20 245	0	22 192
Yarı zamanlı çalışma ile üretilecek veritabanı bağımlı yazılım	1 947[1]	6 350	0	5 500	13 797
Yarı zamanlı çalışma ile üretilecek veritabanı bağımsız yazılım	1 947[1]	6 000	0	2 500	10 447

[1] Çizelge 2'de yer alan ortalama veritabanı maliyetidir.

Çizelge 8. Test ortamı değerleri

Test Ortamı		Açıklama
Donanım	Sunucu	Pentium IV 1.8Ghz (Single CPU)
	İstemciler	20 x Pentium IV 1.8Ghz (Single CPU)
Yazılım	İşletim sistemi	Windows XP Professional
	VTYS	• Microsoft Access • Oracle 8.1.7i • Microsoft SQL Server 2000
	Geliştirme aracı	Borland Delphi 5.0 Ent.
	Bağlantı aracı	ADO (2.8)
	Bench Mark yazılımı	Developed
Veri	Kayıt sayısı	60 000
	Tablo sayısı	6
	Sorgudaki en çok kayıt sayısı	45 000
	Test tipleri	• VT bağlantı (DB connectivity) test • Bakış (View) test (big and small size separately) • Kayıt bulma (Record fetch) test • Aralık sorgulama (Range query) test • Kayıt ekleme (Record insertion) test • Kayıt silme (Record deletion) test • Kayıt değiştirme (Record updating) test



Şekil 7. Uygulama performans testleri için üretilmiş test formu

DBMonster gibi bir çok firma tarafından veritabanı performans ve yük altında stres testi yapan yazılım geliştirilmiştir [33]. Ancak, bu testler veritabanı performansını ölçmektedir. Bu durumda üretilen test yazılımları, uygulama ve veritabanı arasındaki veri alışveriş hızı ile ilgili bir standart test yapamamaktadır.

Bu çalışmada yapılan uygulamalara yönelik veritabanı bağımlı ve bağımsız uygulama yazılımların performans testinin yapılabilmesi için ayrı bir yazılım geliştirilmiştir. Şekil 7'de geliştirilen performans test yazılımına ait form görülmektedir. Bu yazılımda bir veri modülü oluşturularak (VT bağımsız), VTYS'nin içinde (VT bağımlı) oluşturulmuş işlemlerle karşılaştırma yapılarak sonuçlar bir çizelgede sunulmuştur.

Çizelge 9'da yapılan testler sonucunda elde edilen sürelerin tümü yer almaktadır. Çizelgeden görülebileceği gibi, toplam gecikme sürelerinde bağımsız uygulama yönteminde bir performans düşüşü mevcuttur.

Karma bağımlı uygulama süresi : 22,56 saniye
Karma bağımsız uygulama süresi : 25,89 saniye
Toplam performans kaybı süresi : 3,33 saniye (25,89 - 22,56)
Toplam performans kaybı oranı : %12,93 (3,33 / 25,89 = 0,1286)

Test sonucunda işlem gören kayıt sayısı 93 668 adettir. Toplam karma reaksiyon zamanları bağımlı uygulama için 22,56 saniye, bağımsız uygulama için 25,89 saniyedir. Bu sonuçlara göre bağımlı uygulama yöntemi bağımsız uyulama yöntemine göre %12,86 daha hızlı olduğu görülmektedir.

Performans düşüklüğü oranları, özellikle testlerde uygulanan sorgulama ya da işlemin içerdiği veri sayısının büyük olduğu durumlarda artmaktadır. Tek kayıt ekleme silme ve güncelleme işlemlerinde performansta herhangi bir değişim söz konusu olmamakla birlikte çok küçük oranda (yaklaşık 2 ms) performansta bir artış gözlemlenmiştir.

Yöntemin önerildiği ve uygulandığı ortamlar KOBİ gibi düşük ölçekli veri ortamlarıdır. Kullanılan kayıt sayısının düşük olduğu bu tip ortamlarda ortaya çıkan

Çizelge 9. VTYS üzerinde ve uygulama katmanından yapılan seçme işlemler için (gecikme) performanslar

İşlem Tipi	Kayıt Sayısı	Ms. Access (saniye)		Ms. SQL Server 2000 (saniye)		Oracle 8.1.7.i (saniye)	
		Bağımsız	Bağımlı	Bağımsız	Bağımlı	Bağımsız	Bağımlı
VTYS Bağlantı Hızı	0	3,00	3,00	5,00	5,00	6,00	6,00
Bakış Testi (küçük ölçekli)	1 151	0,12	0,10	0,22	0,19	0,28	0,23
Bakış Testi (büçük ölçekli)	68 308	6,38	5,59	5,98	5,23	4,79	4,12
Kayıt Bulma	1	0,05	0,05	0,04	0,04	0,04	0,04
Aralık Sorgulama (küçük ölçekli)	1 617	0,18	0,14	0,19	0,16	0,21	0,17
Aralık Sorgulama (büçük ölçekli)	16 297	1,93	1,71	1,43	1,30	1,41	1,29
Kayıt Ekleme	1	0,04	0,04	0,05	0,05	0,05	0,05
Kayıt Silme	1	0,08	0,08	0,10	0,10	0,12	0,12
Kayıt Güncelleme	6 312	0,99	0,82	0,62	0,48	0,59	0,46
TOPLAM	**93 688**	**9,77**	**8,53**	**8,63**	**7,55**	**7,49**	**6,48**

yaklaşık %13 lük hız farkı oldukça düşük bir farktır ve fiyatın performansa önceliği düşünüldüğünde göz ardı edilebilecek bir orandır. KOBİ'lerde teknolojinin kullanımının yaygınlaştırılması için kullanılacak veritabanı uygulama yazılımlarının düşük maliyetli olması gerektiği düşünüldüğünde, ortaya çıkan sonuçlarda maliyette elde edilen düşüş performansta ortaya çıkan düşüşten daha önemlidir.

4. SONUÇ (CONCLUSION)

VTYS kullanımı sonucu, yüklü bir başlangıç (lisans) maliyeti, yüksek VTYS işletme ve bakım-onarım giderleri bunlara ek olarak bu teknolojiyi kullanacak fazladan uzman personel istihdamı gerekliliği ortaya çıkmıştır. VTYS sistemleri kullanımının daha da yaygınlaştırılması, tüm bilişim alanlarında kullanımının sağlanması ve yapılan işlemlerin hızlandırılması açısından bu maliyetlerin düşük seviyelere indirilmesi büyük önem arz etmektedir.

Yapılan maliyet ve performans analizleri sonucunda, veritabanı bağımsız geliştirilecek bir uygulama yazılımının, veritabanı bağımlı geliştirilecek bir uygulama yazılımına göre, maliyet açısından: istihdam edilen personel durumuna bağlı olarak %30 ile %50 arasında daha düşük maliyete sahip olduğu, performans açısından da yaklaşık %12,86 daha düşük performansa sahip olduğu gözlemlenmiştir. Bu sonuçlar değerlendirildiğinde, maliyetin performanstan öncelikli olduğu durumlarda yöntemin uygulanabilir olduğu ve küçük ve orta ölçekli işletmeler için veritabanı yönetim sistemiyle birlikte çalışan bağımsız bir uygulama yazılımı ile başlangıç maliyetlerinde ve işletme giderlerinde büyük bir düşüş sağladığı gözlemlenmiştir. Böylece, KOBİ'lerin düşük maliyetlerle bilgisayar teknolojisi kullanmasının kolaylaştırılması ve bu sayede hem güncel olarak teknolojiyi takip etmeleri hem de üretime daha fazla katkı yapabilmeleri sağlanabilecektir. Veritabanı bağımsız yapılacak bir uygulama yazılımı geliştirilen yöntemiyle elde edilebilecek olumlu sonuçlar aşağıdaki şekilde sıralanmıştır:

- Veritabanı seçimi kullanıcıya bırakıldığı ve herhangi bir veritabanının (açık kodlu ve ücretsiz veritabanları dahil) kullanılmasına olanak tanıdığı için, veritabanı yönetim sistemi başlangıç maliyetinin kullanıcı tarafından uygun koşullarda elde edilmesi.
- Veritabanı yönetim sistemi üzerinde karmaşık işlemlerin yapılması gerekemediği için, bu konuda çalışması gereken uzman personel istihdamının zorunluluğunun ortadan kalkması.
- Veritabanı yönetim sistemlerinde yapılan bakım, onarım gibi işlemler için fazladan işletme giderlerine ihtiyaç olmaması.
- Bu yöntem kullanılarak, veritabanı yönetim sistemleri hakkında çok fazla bilgiye sahip olmayan yazılım geliştiricilerin veritabanıyla yazılım üretebilme olanağının sağlanması.
- SQL kullanabilen her veritabanı üzerinde kolayca kullanılabilir olduğu için, istenildiği an veritabanı sisteminin değiştirilebilmesi ve yedek almada büyük kolaylık sağlanması.
- Bazı yöntemlerin kullanımıyla (Tümleşik SQL, Başvuru öncesi kontrol, Hazır veritabanı bileşenleri gibi) veritabanı başvuru trafiğinin azaltılması ve sistem performansının düşmesinin engellenmesi.
- Veritabanı üzerinde sadece verilerin bulunmasından dolayı, çok kolay anlaşılabilir yalın bir veri yapısına sahip olması.
- Tüm şifreleme ve yetkilendirme işlemlerinin uygulama tarafından yönetilmesi ve veritabanı erişimi ve veri iletişimi sağlanmak için tek ve güvenli bir şifre kullanılması sayesinde, veritabanı güvenliğinin en yüksek derecede uygulanabilmesi.

Ayrıca bu yöntemle geliştirilecek bir uygulama yazılımı için, bu yöntemi iyi analiz etmiş ve sistem performansını etkilemeyecek tarzda şematik yazılım üretebilen bir uygulama yazılımcısına ihtiyaç duyulacağı tespit edilmiştir.

KAYNAKLAR (REFERENCES)

1. Yu, E.K., "Database Independence! Myth or Reality?", **Senior Solutions Architect, Advanced Solutions Group**, University of South Carolina. www.asg.sc.edu/pdf/ DatabaseIndependence.pdf., 3-6, 2003.
2. Awerbuch B., Bar-Noy A., Information Technology Laboratory, "Database Independent Data Routing Scheme", **Technical Report No : 2000-003**, Medical University of South Carolina., 1-2, 2000.
3. Keller A.M., Basu J., "A Predicate-Based Caching Scheme for Client-Server Database Architectures", **The VLDB journal,** 5 : 35-47, 1996.
4. Altınel M., Bornhövd C., Krishnamurthy S., Mohan C., Pirahesh H., Reinwald B.,"Cache Tables : Paving the Way for Adaptive Database Cache", **Proceeding of the 29th VLDB Conference**, Berlin-Germany, 1-12, 2003.
5. Bornhövd C., Altınel M., Krishnamurthy S., Mohan C., Pirahesh H., Reinwald B.,"DBCache: Middle-tier Database Caching for Highly Scalable e-Business Architectures", **SIGMOD International Conference**, San Diego, CA., 1-2, 2003.
6. Post G., Kagan A., "Database Management Systems: Design Considirations and Attribute Facilities", **Elsevier-The Journal Systems and Software**, 56: 183-193, 2001.
7. Mah P.S., Chung S.M., "Schema Integration and Transaction Management for Multidatabases", **The Journal of Information Sciences**, 111: 153-188, 1998.
8. Grufman S., Samson F., Embury S.M., Gray Peter M.D., Risch T., "Distributing Semantic Constraints Between Heterogeneous Databases",

13th **International Conference on Data Engineering, ICDE'97**, Birmingham, England, 1-9, 1997.
9. Mayol E., Teniente E., "A Review of Integrity Constraint Maintenance and View Updating Techniques", **Unversitat Politecnica De Catalunya Research Report No: LSI-03-5-R**, Barcelona, Spain, 1-36, 2003.
10. Qian X., "Integrity Constraint Reformulation for Efficient Validation", **Proceedings of the 13th VLDB Conference**, Brighton, UK, 417-425, 1987.
11. Schewe K.D., "Consistency Enforcement in Entity-Relationship and Object-Oriented Models", **Data & Knowledge Engineering,** 28: 121-140, 1998.
12. Parker D.A., "Database Independent Abstraction Layer for C - libdbi Driver Author's Guide", **Neon Goat Productions**, 6-41, 2002.
13. Ceri S., Widom J., "Deriving Production Rules for Constraint Maintenance", **Proceedings of the 16th Conference on Very Large Databases**, Brisbane, Australia, 566-577, 1990.
14. Yoo S.B., Cha S.K., "Integrity Maintenance in A Heterogeneous Engineering Database Environment", **Data & Knowledge Engineering**, 21: 347-363, 1997.
15. Brodie M.L., "On the Development of Data Model, In M. L. Brodie, J. Mylopoulos, J. W. Schmidt ed., On Conceptual Modeling, Perspectives from Artificial Intelligence, Databases, and Programming Languages, **Springer-VerlagPub**., 87-114, 1984.
16. Geppert A., Dittrich K.R., "Specification and Implementation of Consistency Constraints, in Object-Oriented Database System", Applying Programming-by-Contract, G. Lausen (ed.): **Proceedings. GI Conference BTW**, Dresden, Germany, March., 5-10, 1995.
17. Dittrich K.R., "Object-Oriented Data Model Concepts. In A. Dogac, M. T. Özsu, A. Biliris, T. K. Sellis (ed.)", **Advances in Object-Oriented Database Systems, Computing and Systems Sciences**, 130, 1994.
18. Mathew P.K., SPICE is an Effective Model for Small Companies, http://www.ciol.com/content /news/interviews/101021901.asp, 2004.
19. Demirörs O., Yazılım Süreç Iyileştirme, http://www.bg.com. tr/yayinlar/ MAM-SPI.pdf, 2005.
20. Oracle for Small and Midsize Businesses Products, http://www.oracle.com/solutions/mid/index.html 2005.
21. IBM DB2 Personel Edition Prices, https://www-112.ibm.com/software/howtobuy/ buyingtools/paexpress/Express?P0=E1&part_number=D5B69LL&catalogLocale=en_US&Locale=en_US&country=USA, 2005.
22. IBM DB2 DB2 UDB Enterprise Server Edition Processor Prices, https://www-112.ibm.com/ software/howtobuy/buyingtools/paexpress/ Express?P0=E1&part_number=D518GLL,D518JLL,D53Y2LL&catalogLocale=en_US&Locale=en_US&country=USA (2005).
23. IBM Informix Dynamic Server Enterprice and WorkGroup Edition Prices https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1&part_number= D6DCRLL&catalogLocale=en_US&Locale=en_US&country=USA, 2005.
24. IBM Informix Extended Parallel Server Prices, https://www-112.ibm.com/software/howtobuy/ buyingtools/paexpress/Express?P0=E1&part_number=D6DAXLL,D5282LL,D6DB5LL,D6DBDLL,D53KKLL&catalogLocale=en_US&Locale=en_US&country=USA, 2005.
25. MYSQL Prices, http://www.globalink.com.sg/ mysql/prices/, 2005.
26. Sybase-Aberdeen, Mass-Deployment Database Embedded in Workgroup Environment Cost-Of-Ownership Study, www.sybase.com/content/1020326/Sybase2.pdf 2005.
27. Borland Porducts Shop, http://amos.shop.com/ cc.class/cc?sy=products&main=ccn_search&okp=1&search_form=&ccsyn=260&ost=Software%20Borland&st=Borland%20Software&ccsid=376132130-26429&ccn test=1, 2005.
28. PostgreSQL Support & Maintenance Service, http://osb.sra.co.jp/ PostgreSQL/Service/service-en.php, 2005.
29. Yazılım Geliştirme Aşamaları, http://yazilimproje.tripod. com /proje.htm, 2004.
30. Muhtasar Vergileri ve SSK Aylık Sigorta Primleri Listesi, http://www.calismahayati.net/ guncel.htm 2005.
31. Microsoft .Net, Performans Karşılaştırması: İşlem Denetimi, http://www.microsoft.com/ turkiye/net/yazilimgelistiriciler/performance.asp 2005.
32. MySQL Helps, Set New World Records for Speed & Price-Performance in Independent Benchmarks, http://www.mysql.com/news-and-events/press-release/release_2004_27.html, 2005.
33. DBMonster,http://dbmonster.kernelpanic.pl/ manual/ 2005.