



T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

VERİTABANI SİSTEMLERİNDE
GÜVENLİK ARTIRIMI – TADA SİSTEMİ

134361

Cihan KILIÇ
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI

Yrd.Doç.Dr. İbrahim SOĞUKPINAR

134361
T.C. YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

GEBZE

2003

Cihan KILIC'in tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..06.02.2003... tarih ve ..2003.1.05... sayılı kararıyla oluşturulan jüri tarafından Bilgisayar Müh. Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.



JÜRI

ÜYE

(Tez Danışmanı)

: Yrd. Doç. Dr. İbrahim SOBUKPINAR

ÜYE

: Yrd. Doç. Dr. Mehmet GÖKTÜRK

ÜYE

: Yrd. Doç. Dr. İlyas KANDEMİR

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..18.06.2003... tarih ve ..2003.1.18... sayılı kararı.



ÖZET



Veritabanları bilgisayarların kullanıldığı tüm alanlarda kritik rol oynamaktadır. Herhangi bir alandaki veritabanlarında verinin ve bilginin öneminden dolayı bu veritabanları güvenli ve korumalı olmalıdırlar.

Günümüz Veritabanı Yönetim Sistemlerinde (DBMS) veritabanı bağlantısı sağlayan bir çok uygulama programı mevcuttur. Veritabanı yöneticileri (DBA) güvenlik amacıyla son kullanıcıların bu tür programlarla veritabanı bağlantısı sağlamalarını istemezler. Çünkü erişim yetkileri düzgün yapılmamış olan veritabanına bu tür uygulama programları tarafından yapılan bir bağlantı sayesinde, kullanıcılar erişmemeleri gereken verilere erişim hakkına sahip olabilirler. Bunu sağlamak için kullanıcıların bu tür uygulama programları kullanarak veritabanı bağlantısı yapmasını engelleyip, veritabanı üzerindeki verilerin güvenliğinin sağlandığı uygulama programlarını kullandırmaya yöneltmek gerekir. Bu da uygulama programı içerisinde yapılacak olan kullanıcı erişim kontrolleri sayesinde gerçekleşir. Böylece çalışacak SQL ifadelerinin sınırlandırılması ya da kullanıcıların hangi verilere erişim sağlayabilmesi gerektiğinin kontrolü kolayca yapılabilir.

Bu çalışmada son kullanıcıların, sadece veritabanı güvenliğini tehlikeye sokmayan uygulama programlarıyla veritabanına bağlanmalarını sağlamak için bir yöntem geliştirilmiş ve bu sistem ile ilgili örnek uygulama ve kaynak kodları verilmiştir. Geliştirilen yöntem üçlü kimlik doğrulama (triple authentication) ve dinamik yetkilendirme (dynamic authorization) esaslı olarak çalışır. Bu sistemde veri güvenliğini sağlamak amacıyla iki farklı veritabanı kullanılmakta ve her kullanıcıya iki farklı kullanıcı kimliği tahsis edilmektedir. Kullanıcılar bu kullanıcı kimliklerinden bir tanesiyle muhatap olacaklar ve diğer kullanıcı kimliği ve şifre uygulama programı içerisinde tespit edilecektir. Ayrıca kullanıcılar veritabanı bağlantısını minimum yetkilerle gerçekleştirecek ve uygulama programı içerisinde kullanılan module göre gerekli yetkiler dinamik olarak verilecektir. TADA sistemi olarak adlandırılan bu yöntem, veritabanı bağlantısını daha güvenli kılmasına karşı zaman kaybı kabul edilebilir ölçüdedir.

SUMMARY

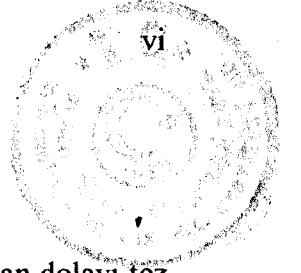


Databases play a critical role on all fields that computers being used. Databases should be secure and protected on any field because of the importance of data and information stored in it.

There are many methods that provide database connection on modern Database Management Systems (DBMS). But user authentication for DBMS is an important problem due to the password vulnerabilities. Database Administrators (DBA) generally do not want such applications to be used by the end users. Because there is always a risk of a user accession to information in database that the user having wrong access permission. However accessing to DBMS shall be allowed for using programs that provides database security. This is realized using access controls inside of application software. Using such restrictions, SQL queries could be controlled and users shall be restricted to access certain data.

A new method was proposed in this work to allow end users to connect DBMS only using programs that will not risk the security of the database and the sample source codes have been given related to such methods. The provided method works based on triple authentication and dynamic authorization. Two separate databases are used in this system to achieve data security and two sets of username and passwords are assigned to each user. The users will only be aware one of them, and other username and password will determined from within the application software. Furthermore the users will be connected to the database with minimum privileges and then privileges that are required according to the current module used by user within the application program will be given dynamically. Method proposed in this study named as TADA System provides more secure database connection however time consuming of method is acceptable.

TEŞEKKÜR

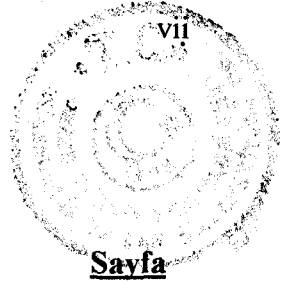


Yüksek Lisans Tezi çalışmalarım sırasında özverili yardımlarından dolayı tez danışmanım ve Gebze Yüksek Teknoloji Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı bölüm başkanı Yrd. Doç. Dr. İbrahim SOĞUKPUNAR başta olmak üzere, TÜBİTAK Ulusal Metroloji Enstitüsü Bilgi İşlem Merkezi sorumlusu Alper AKER' e ve çalışma arkadaşlarıma, hayatım boyunca maddi ve manevi desteklerini üzerimden eksiltmeyen anne ve babama, tez çalışmam sırasında bana büyük destek sağlayan eşim Hatice KILIÇ ile çalışmalarım esnasında en büyük moral kaynağım olan yaklaşık bir ay sonra dünyaya gözlerini açacak sevgili kızıma teşekkürlerimi bir borç bilirim.

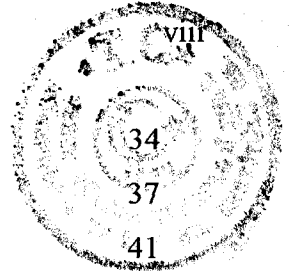
Cihan KILIÇ

28/04/2003, Kocaeli

İÇİNDEKİLER DİZİNİ



ÖZET	iv
SUMMARY	v
TEŞEKKÜR	vi
İÇİNDEKİLER DİZİNİ	vii
SİMGELER VE KISALTMALAR DİZİNİ	ix
ŞEKİLLER DİZİNİ	x
1. GİRİŞ	1
2. VERİTABANI GÜVENLİK KAVRAMI	4
2.1. Veritabanı Kavramı	4
2.2. Veritabanı Güvenliğine Niçin İhtiyaç Duyulur?	6
2.3. Güvenlik Gereksinimleri ve Sorunları	8
2.3.1. Birliktelik Kontrolleri	9
2.3.2. Veritabanı Bütünlüğü	10
2.3.3. Erişim Kontrolü	10
2.3.4. Kullanıcı Kimlik Doğrulama (Authentication)	13
2.3.5. Güvenlik Kısıtlamaları	13
2.4. Veritabanı Erişim Kontrol Teknikleri	16
2.4.1. İsteğe Bağlı Erişim Kontrolü	16
2.4.2. Zorunlu Erişim Kontrolü	18
2.4.2.1. Bütünlük Kilidi (Integrity Lock)	22
2.4.2.2. SeaView	23
2.4.2.3. SWORD	25
2.4.3. Rol Temelli Erişim Kontrolü	26
2.5. Dinamik Yetkilendirme	27
2.6. Veritabanı Güvenlik Problemleri	27
2.6.1. Inference Problemi	28
2.6.2. Polyinstantiation	29
2.6.2.1. Görünmeyen Polyinstantiation	30
2.6.2.2. Görünür polyinstantiation	32



2.6.2.3. Polyinstantiation Problemi İçin Çözümler	
3. TADA SİSTEMİ	
3.1. TADA Sisteminin İşleyişi	
4. GERÇEKLENEN UYGULAMA VE SONUÇLAR	45
4.1. TADA Sistemi Arayüzleri	45
4.2. Geliştirilen Foksiyonlar	49
4.2.1. Veri Şifreleme Fonksiyonu	49
4.2.2. Piyon Kullanıcıdan Asıl Kullanıcı Adını Elde Eden Fonksiyon	49
4.2.3. Asıl Kullanıcı Adına Ait Şifreyi Tespit Eden Fonksiyon	50
4.2.4. Şifre Korumalı Rol Oluşturma	52
4.2.5. Role Ait Şifreyi Tespit Eden Fonksiyon	52
4.2.6. Kullanıcı Yaratma Prosedürü	54
4.2.7. Kullanıcı Şifrelerini Değiştiren Veritabanı Job'u	55
4.2.8. Rol Şifrelerini Değiştiren Veritabanı Job'u	57
4.2.9. Son Kullanıcı Program Girişi	59
4.2.10. Modüle Göre İlgili Rollerini Kullanıcıya Atayan Prosedür	60
4.3. Performans ve Güvenlik Analizi	61
5. SONUÇ VE ÖNERİLER	63
KAYNAKLAR	66
ÖZGEÇMİŞ	68

SİMGELER VE KISALTMALAR DİZİNİ



DBA	Veritabanı Yöneticisi
DBMS	Veritabanı Yönetim Sistemi
I/O	Giriş / Çıkış
ISO / IEC	International Organization for Standardization / International Electrotechnical Commission
MLS / DBMS	Çok Düzeyli Veritabanı Yönetim Sistemi
RBAC	Rol Temelli Erişim Kontrolü
SQL	Yapısal Sorgulama Dili
TADA	Triple Authentication and Dynamic Authorization



ŞEKİLLER DİZİNİ



<u>Sekil</u>	<u>Sayfa</u>
2.1 Kısıtlama işlemi için tümleşik mimari	15
2.2. Ayrıcalıkların aktarımı ve geri alınması	17
2.3. Truva atı örneği	19
2.4. Bütünlük kilidi (Integrity Lock) mimarisi	22
2.5. Bütünlük kilidi veritabanı ve gerçekleştirilmesi	22
2.6. SeaView Mimarisi	23
2.7. SeaView tablosu örneği	24
2.8. SeaView örneği uygulaması	24
2.9. SWORD Veritabanı örneğinin iki görüşü	25
2.10. Çok düzeyli ilişki örneği	31
2.11. Polyinstantiated satır örneği	31
2.12. Polyinstantiated öge örneği	32
2.13. Çok düzeyli ilişki örneği	33
2.14. Polyinstantiated satır örneği	33
2.15. Belief modelinde çok düzeyli ilişki örneği	36
3.1. TADA sistemi iş akış diyagramı	37
3.2. Veritabanı ile uygulama programı arasındaki şifre eşlemesi örneği	39
3.3. Dinamik Yetkilendirme	41
3.4. TADA sisteminin uygulananışı	42
4.1. TADA Sistemi Giriş Arayüzü	45
4.2. Şifre Değiştirme Arayüzü	46
4.3. Logon Ekranı	46
4.4. Kullanıcı Bilgileri Arayüzü	47
4.5. Uygulama Programı Modülü	48
4.6. Yeni Kullanıcı Yaratma Arayüzü	48



1 GİRİŞ

Veritabanı ve veritabanı teknolojisi bilgisayar kullanımının gelişiminde büyük bir etkiye sahiptir. Veritabanları bilgisayarların kullanıldığı tüm alanlarda kritik rol oynamaktadır. Herhangi bir alandaki veritabanlarında verinin ve bilginin öneminden dolayı bu veritabanları güvenli ve korumalı olmalıdırlar. Veritabanları sadece veri depolamazlar, aynı zamanda kullanıcılara bilgiyi de sağlarlar. Veri herhangi bir bozulma veya yetkisiz erişimden korunmalı ve kullanıcılar eriştiği zaman bilgi kontrol edilmelidir. Veritabanı güvenliği, işletim sistemi veya uygulama programlarından ziyade veritabanı yönetim sisteminin sorumluluğunda olmalıdır.

Elektronik suçlar zamanla artmakta ve bu gerçek zamanlı sistemlerin teknolojik ihtiyaçlarını ihmal etmeden şirketlerin bilgi güvenliğine daha fazla önem vermesini sağlamaktadır. Bundan dolayı, bilgi sistemleri analizi ve tasarımı yapılırken güvenlik gereksinimleri hesaba alınır.

Günümüz veritabanı yönetim sistemleriyle (DBMS) birlikte gelen, veritabanı bağlantısı sağlayacak bir çok uygulama programı mevcuttur. (Oracle SQL Plus vb.) Kullanıcılar bu uygulama programları sayesinde veritabanına bağlandıkları zaman, kendilerine verilen yetkiler çerçevesinde yazdıkları yapısal sorgulama dili (SQL) ifadelerine göre veriler üzerinde işlem yapabilme yeteneğine sahip duruma gelmektedirler.

Veritabanı yöneticileri bu tür uygulamaların son kullanıcılar tarafından kullanılmasını istemezler. Çünkü erişim yetkileri düzgün yapılandırılmamış olan veritabanına bu tür uygulama programları tarafından bir bağlantı yapılması durumunda, kullanıcılar erişmemeleri gereken verilere erişim hakkına sahip olabilirler. Oysa kullanıcıların girecekleri SQL ifadeleri sınırlandırılmış olsa ya da sadece veritabanı yöneticisinin izin verdiği SQL ifadelerini kullanabilseler bu tür bir güvenlik tehdidi ortadan kaldırılmış olurdu.

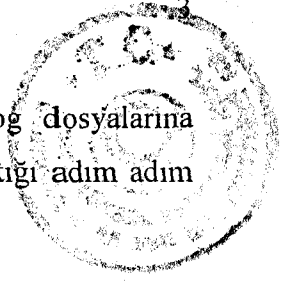
Bunu sağlamak için kullanıcıların bu tür uygulama programlarını kullanarak veritabanı bağlantısı yapmasını engelleyip, veritabanı üzerindeki verilerin

güvenliğinin sağlandığı uygulama programlarını kullandırmaya yöneltmek gerekir. Bu da uygulama programı içerisinde yapılacak olan kullanıcı erişim kontrolleri sayesinde gerçekleşir. Böylece çalışacak SQL ifadelerinin sınırlandırılması ya da kullanıcıların hangi verilere erişim sağlayabilmesi gerektiğinin kontrolü kolayca yapılabilir.

Son kullanıcıların veritabanı bağlantılarını, öngörülen güvenli bir uygulama programı ile sağlamanın getireceği avantajlar şu şekilde belirtilebilir :

- *İzinsiz veritabanı bağlantısının engellenmesi* : Yetkisiz kişilerin veritabanı bütünlüğünü bozacak illegal yazılımlar ile veritabanına bağlantıları engellenmiş olacaktır.
- *İzinsiz veri erişiminin engellenmesi* : Kullanıcılar kendi SQL cümleciklerini giremeyecekleri ve sadece uygulama programı tarafından izin verilen verileri görebileceği için izinsiz veri erişimi engellenmiş olur. Böylece veritabanı üzerindeki zorunlu erişim kontrolleri, uygulama programı bazında gerçekleştirilebilir.
- *Inference probleminden korunma* : Kullanıcı veriye direkt erişimi olmadan bilgiyi çıkarsamak için yasal veri ve sorguları kullandığı zaman inference gerçekleşir. Kullanıcılar kendi sorgularını oluşturamadıkları için, erişim yetkileri bulunmayan hassas verileri çıkarsamaları da mümkün olmayacaktır. Ancak uygulama programı içerisinde ulaşılacak verilere dikkat edilmelidir.
- *Polyinstantiation probleminden korunma* : Polyinstantiation problemi veri eklenildiği veya değiştirildiği zaman görülen bir problem olduğu için, uygulama yazılımı içerisinde veri ekleme ve değiştirme işleminin kontrollü ve düzgün bir şekilde yapılması polyinstantiation problemini çözecektir.
- *Kapsamlı denetleme sistemi* : Uygulama programı içerisinde, veritabanı

üzerinde yapılan her türlü işlem ayrıntılı bir şekilde log dosyalarına kaydedilerek hangi kullanıcının hangi veri üzerinde ne yaptığı adım adım takip edilebilir.



Veritabanı uygulama programı son kullanıcıların veritabanıyla alakalı isteklerini yerine getirecek şekilde oluşturulmalıdır. Aksi halde kullanıcı yetkili olduğu veriye dahi erişim sağlayamaz ise kullanıcıyı belirli programları kullandırmaya yöneltme düşüncesi tamamıyla dezavantajlı hale dönüşmüş olur.

Bu tez çalışmasında, veritabanı sistemlerinin güvenlik ihtiyacı göz önüne alınarak, güvenliğin artırılmasına yardımcı olacak yeni bir yöntem geliştirilmiştir. Tez çalışması dört (4) ana bölümden oluşmaktadır.

Birinci bölümde; tezde üzerinde çalışılan konuya giriş yapılmış ve veritabanına tek bir uygulama yazılımı ile bağlantı sağlamanın getireceği avantajlar anlatılmıştır.

İkinci bölümde; veritabanı güvenliği konusunda yapılan literatür araştırması sonucunda elde edilen bilgiler bulunmaktadır.

Üçüncü bölümde; veritabanı sistemlerinde güvenlik artırımı sağlamak amacıyla geliştirilmiş olan TADA sisteminin yapısı, işleyişi hakkında bilgi verilmiş ve TADA sisteminin performans ve güvenlik analizi yapılmıştır.

Dördüncü bölümde ise; bu tez çalışmasında yapılan teorik ve uygulama sonuçları tartışılmış ve bazı ileriye yönelik çalışmalar için önerilerde bulunulmuştur.

Ekler bölümünde ise TADA sistemine uygun dizayn edilmiş bir veritabanı uygulama yazılımının örnek kaynak kodları verilmiştir..

2 VERİTABANI GÜVENLİK KAVRAMI



2.1 Veritabanı Kavramı

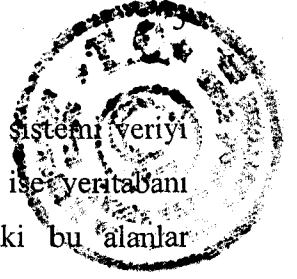
Veritabanı bazı anlamlı yollarla düzenlenmiş birbirleriyle ilişkili verilerin toplamıdır. Veriler arasındaki ilişkilerle bilgi nesneleri veya varlıklarından (entity) ibaretmiş gibi dikkate alınabilir. Veritabanındaki veriler birden çok uygulama tarafından paylaşılan ortak verilerdir. Veritabanı özel bir amaç için tasarlanır ve inşa edilir ve veritabanı yönetim sistemine bağlı olur. Veritabanı yönetim sistemi (DBMS), veritabanını yaratmak ve korumak için kullanıcıları etkinleştiren programların toplamıdır. Veritabanı veriyi düzenler ve günlük tutma, kurtarma ve yeniden organize olma gibi bazı servislere erişim sağlar.

Veritabanı kullanımının sağlamış olduğu faydalar şu şekilde özetlenebilir :

- Veri tekrarının azaltılması
- Veri bütünlüğünün sağlanması
- Veri paylaşımının sağlanması
- Veriler üzerindeki işlemlerin (sorgulama, güncelleme, raporlama vb.) çok rahat yapılabilmesi
- Verilerin gizliliğinin ve güvenliğinin sağlanması
- Veriler ve uygulamalarla ilgili her türlü standardın (verinin yapısı, gösterimi, adlandırılması, belgelenmesi gibi) uygulanabilmesi

Veritabanı sistemi, kullanıcıların erişimini kontrol eden ve veriyi düzenlemek için bazı kuralları tanımlayan veritabanı yöneticisine (DBA) ihtiyaç duyar. DBA veritabanına erişimi yetkilendirmekten ve kullanımı düzenleyip izlemekten sorumludur. Aynı zamanda, DBA güvenlik veya kötü sistem yanıt süresinin

bozulması gibi problemlere karşı da sorumludur. Veritabanı yönetim sistemi veriyi dosyalarda depolar; her bir dosya kayıtlardan oluşur. Her bir kayıt ise veritabanı öğelerini temsil eden alanlardan oluşur. Genellikle veritabanındaki bu alanlar arasında ilişkiler mevcuttur.



Veritabanı yöneticisinin görevlerini aşağıdaki gibi özetleyebiliriz :

- Veri ve veriler üzerinde yapılacak uygulama gereksinimlerini belirlemek, veritabanı içeriğini oluşturmak, veritabanı şemalarını tanımlamak.
- Bütünlük kısıtlamalarını belirleyip tanımlamak.
- Veritabanı kullanıcılarını ve her kullanıcının hangi veriler üzerinde hangi işlemleri yapmaya yetkili olduğunu belirlemek; kullanıcı ve kullanım yetkilerini tanımlamak.
- Veritabanı yönetim sisteminin sunduğu seçenekler çerçevesinde, veritabanının fiziksel yapısı ile ilgili parametreleri ve erişim yollarını belirlemek ve tanımlamak.
- Yedekleme, yeniden başlatma ve kurtarma stratejilerini belirlemek.
- Veritabanı sistemini sahiplenmek, işletimini izlemek, veritabanının sürekli olarak kullanıma açık olmasını sağlamak.
- Gereksinimlerdeki değişiklikleri izlemek ve bu değişikliklere paralel olarak veritabanı içeriği, şema tanımları, bütünlük kısıtlamaları, fiziksel yapı ile ilgili parametreler, erişim yolları, kullanıcılar ve kullanıcı yetkilerinde gerekli değişiklikleri oluşturmak ve tanımlamak.
- Veritabanı bütünlük kısıtlamalarının yeterliliğini izlemek; bütünlük kısıtlamaları ile ilgili gerekli değişiklikleri oluşturmak ve tanımlamak.

- Veritabanı kullanım istatistiklerini ve veritabanı başarımını izlemek; varsa sorunları ve yetersizlikleri belirlemek ve gerekli her türlü önlemi almak

Veritabanı ve veritabanı teknolojisi bilgisayar kullanımının gelişiminde büyük bir etkiye sahiptir. Veritabanları bilgisayarların kullanıldığı iş, mühendislik, tıbbi, hukuk, eğitim ve kütüphane bilimini kapsayan yerlerdeki tüm alanlarda kritik rol oynamaktadır. Herhangi bir alandaki veritabanlarında verinin ve bilginin öneminden dolayı bu veritabanları güvenli ve korumalı olmalıdırlar. Veritabanları sadece veri depolamazlar, aynı zamanda kullanıcılara bilgiyi de sağlarlar. Veri herhangi bir bozulma veya yetkisiz erişimden korunmalı ve kullanıcılar eriştiği zaman bilgi kontrol edilmelidir. Veritabanı güvenliği, işletim sistemi veya uygulama programlarından ziyade veritabanı yönetim sisteminin sorumluluğunda olmalıdır. Güvenlik, özellikle hassas bilgi kullanan herhangi bir veritabanı yönetim sisteminde ana konudur. Güvenlik yetkili kullanıcılara erişim hakkı vermekle başılır. Kullanıcılar genellikle veritabanı yöneticisi veya sistem yöneticisi tarafından ayarlanırlar.

2.2 Veritabanı Güvenliğine Niçin İhtiyaç Duyulur?

Bilindiği gibi, veritabanı bilgisi günümüzde kritik kaynaklardır. Endüstriyel, ticari, eğitim gibi herhangi bir alandaki kişiler bilgilere ve veritabanlarına bağlı olurlar. Veritabanı önemi çeşitli derecelerde olan veriyi içerir ve geniş çeşitlilikteki kullanıcılar arasında paylaşılır. Bundan dolayı korunmaya ve yönetilmeye ihtiyaç duyar, çünkü veritabanındaki herhangi bir değişiklik kendisini ve diğer ilgili veritabanlarını etkileyebilir. Günümüzde, çoğu veritabanı online'dır ve eş zamanlı olarak çoklu kullanıcılar tarafından erişilir. Bundan dolayı daha çok kontrol edilme ve veritabanının bütünlüğünü koruyabilme ihtiyacı duyarlar.

ISO/IEC (International Organization for Standardization / International Electrotechnical Commission)' te tanımlanıldığı gibi güvenlik, önemli ve kaliteli bir özelliktir. Bu standarda göre, iç ve dış kalitenin esas bileşenleri güvenilirlik, verim, kullanılabilirlik, işlevsellik, bakım yeteneği (korunabilirlik) ve taşınabilirliktir. Bu

standart, güvenliği aşağıdaki şekilde tanımlar:

“Güvenlik, yetkisiz kişiler veya sistemlerin okuyamaması - değiştirememesi ve yetkili kişilerin erişimine izin vermek için veri ve bilgiyi koruyan yazılım ürünlerinin kapasitesidir.”

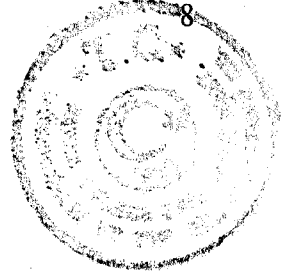
Genelde bilgi sistemleri hassas ve erişim kısıtlamasını dikkate alabilen bilgiyi kapsar. Sıklıkla sosyal güvenlik numaraları, adresler, telefon numaraları, dini, cinsi ve tıbbi veriler gibi özellikle korunması gereken, bireyler hakkındaki kişisel bilgileri içerirler. Eğer bu bilgi belirli güvenlik tedbirleri olmadan kullanılırsa kişisel haklar ihlal edilebilir.

Bilişim teknolojilerinin kullanımı, vatandaşın haklarını desteklediğinden emin olmak için hizmet eden anayasa maddesine göre düzenlenirler. Çoğu ülke -veri özellikle çok hassas kişisel bilgiler içeriyorsa- veriyi koruma kanunları koymuşlardır. Bilgi sistemleri veri güvenliğini garanti etmek için sadece para cezasına ve ihlal yasasına uygun cezaya çarptırılmaz, güvenliğin aşılmasından dolayı meydana gelebilecek olumsuz etkilere de maruz kalınır. (İşletmeye karşı güvensizlik duyulması, saygınlığın yitirilmesi vb.)

Bu faktörlerin tümünü içeren kanunlar koyan düzenleyici ve teknolojik etkenler, bilgi teknolojilerinde güvenliğin önemini doğrularlar. Ayrıca uzun vadede büyümeye ve bu teknolojik süreç içerisinde istikrara güvence vermenin ekonomik perspektifi vardır. Örneğin; Baskerville bunu üç yıllık zaman periyodu içerisinde bir veya daha fazla fiziksel ve mantıksal bilgi sistemi bozulmalarından sıkıntı çeken ABD şirketleri üzerinde yapmış olduğu araştırmayla doğrulamıştır. Bu çalışmaya göre, ABD şirketlerinin %12' sinde, ortalama 75.000\$' lık elektronik yolsuzluk olayı sonuçlanmaktadır. [1]

Veritabanı güvenliği esasen gizlik, bütünlük ve kullanılabilirlik ile alakalıdır. Bunlar sırasıyla, ihtiyaç duyulduğunda yetkisiz kişiler tarafından gizli bilginin bulunması, yetkisiz personel tarafından bilginin güncellenmesi ve bilginin kullanılabilirliğinin olmayışıyla ilgili problemlerle ilgilidir.

2.3 Güvenlik Gereksinimleri ve Sorunları



Güvenli bir veritabanı aşağıdaki koşulları sağlamalıdır.

1. Fiziksel bütünlüğe sahip olmalı. (Güç kesintileri veya doğal felaketlerden dolayı meydana gelebilecek veri kaybından koruma).
2. Mantıksal bütünlüğe sahip olmalı. (Veritabanının mantıksal yapısının korunması)
3. İhtiyaç olduğu zaman mevcut olmalı.
4. Sistem, denetleme sistemine sahip olmalı. (Veritabanı elemanlarına erişenleri veya değiştirenleri izlemek için)
5. Esas bütünlüğe (doğru veriye) sahip olmalı. (Her bir veritabanı ögesinin içerdiği verinin doğruluğu)
6. Erişim verinin hassasiyetine bağlı olarak bazı derecelerde kontrol edilmeli. (Böylece kullanıcıya sadece yetkili olduğu veriye erişimine izin verilecek ve farklı kullanıcılar kısıtlanmış olacaktır.)
7. Kullanıcı kimlik denetimi ile veritabanının yetkisiz kullanımı önlenmelidir.
8. Kullanılabilirlik, kullanıcılar yetkili oldukları genel ve tüm veriye veritabanında erişebilmelidir.
9. Hassas veri, inference (çıkarım) probleminden korunmalıdır.

Bu güvenlik alanları DBMS modelinin seçiminden doğrudan etkilenmektedir. Belirtilen bu gereksinimlerin anahtar hedefi, DBMS' i yetkisiz gözlem veya inference, yetkisiz modifikasyon ve hatalı güncellemelerden koruyarak depolanan

veriyi garanti altına almaktadır.



Erişim kısıtlamasının düzeyi, verinin hassasiyetine ve geliştiricinin en düşük ayrıcalığının derecesine bağlıdır. Tipik olarak DBMS' de korunur. Kullanıcı oturum açtığı zaman, kullanıcı için özel ayrıcalıkları elde eder.

Pfleeger' a göre erişim izni aşağıdaki kriterlerin bir ya da birkaçının yerine getirilmesine dayandırılabilir. [2]

1. Verinin mevcudiyeti : Verinin mevcut olmaması genellikle belirli veri öğelerinin başka bir özne (subject) tarafından kilitlenmesine sebep olur.
2. Verinin erişilebilirliği : Sadece yetkili kullanıcılar veriyi görür veya değiştirebilir. Tek düzeyli sistemde, bunu uygulamak nispeten daha kolaydır. Eğer kullanıcı yetkisiz ise, iletişim kontrolü çok daha zor yerine getirilir, çünkü DBMS kullanıcısının isteğe bağlı erişim ayrıcalıklarını mecbur etmelidir.
3. Doğruluğun sağlanması : Bu kayıtlı kullanıcının gerekliliğini kontrol etmek için normal çalışma saatlerinde erişimin kısıtlanmasını içerir.

2.3.1 Birliktelik Kontrolleri

Birliktelik kontrolleri verinin bütünlüğünü garanti etmek için yardım eder. Bu kontroller birden fazla kullanıcı aynı veri elemanını kullandığı zaman veri kullanma biçimini düzenlerler. Bunlar dağıtık sistemin etkin yönetiminde özellikle önemlidir. Çünkü çoğu durumda tek olmayan DBMS veri erişimini kontrol edemez. Eğer etkin birliktelik kontrolleri dağıtık sistem içerisine entegre edilemez ise çeşitli problemler çıkabilir. Bell ve Grisom, birliktelik problemlerinin üç olası kaynağını tanımlarlar.[2]

1. Kayıp Güncelleme : Başarılı bir güncelleme işlemi başka kullanıcı tarafından yanlışlıkla ya da kazara silinebilir.
2. Senkronize olmayan transactionlar bütünlüğü bozar.

3. Tekrarlanmayan okuma: Eriřilen veri hatalıdır, çünkü veri güncelleme esnasında elde edilmiştir.



Bu problemlerin her biri uygun kilitleme planı (sadece bir özne kilidin süresi için verilen varlığa erişir.) veya timestamp metodu (erken timestamp ile özne önceliği alır.) uygulayarak azaltılabilir veya ortadan kaldırılabilir.

2.3.2 Veritabanı Bütünlüğü

Veritabanı bütünlüğü, veritabanı güvenliğinin en önemli bileşenlerinden biridir. Veritabanı içeriğinin doğruluğu ile ilgilenir. Veritabanı kullanıldığı zaman, kullanıcılar veri değerlerinin doğruluğuna güvenmelidirler. Veri herhangi bir bozulmaya karşı korunmalıdır. Bozulma çeşitli yollarla olabilir : [3]

- *İllegal programlar çalıştırarak* : Yetkisiz kişi veritabanına eriştiği ve bazı verileri değiştirdiği veya veriyi karıştırmak yada değiştirmek için yetkisiz yolla herhangi bir programı çalıştırdığı zaman olur.
- *Disk bozulması* : Eğer kurtarmak için hiç yedek veya başka bir yol yoksa veri tamamen kaybolabilir.
- *Dış etkenler* : Yangın, deprem, güç kesilmeleri veya herhangi bir etken.

DBMS ve veritabanı yöneticisi, veritabanı bütünlüğünden sorumludur. DBMS sürekli olarak veritabanını ve transactionların günlüğünü korumalıdır. Çünkü her zaman sistem hatası olabilir. Eğer sistem hatası koruma olmadan olduysa, veritabanı bütünlüğü bundan etkilenir. Veritabanı yöneticisi de değişikliklerin sadece yetkili kişiler tarafından gerçekleştirilmesini sağlamalıdır.

2.3.3 Erişim Kontrolü

Erişim kontrolü; yöntem, mekanizma ve veritabanını yetkisiz kişilerin kullanımından veya erişiminden korumak için uygulanan prosedürlerin toplamıdır.

Çalıştırılan programlarla veriye erişim kontrol edilir ve yetkisiz erişime karşı paylaşılan veritabanındaki veri korunur. Erişim kontrolü veriyi hem görmeyi hem de değiştirmeyi içerir. Veri görünümünü kontrol ederek veri gizliliğini sağlamaya yardım eder ve veri değiştirmeyi kontrol ederek verinin doğruluğunu ve kalitesini geliştirmeye yardımcı olur. Erişim kontrolü veri kaynaklarının paylaştırıldığı ortamda daha çok önemlidir.

İlişkisel veritabanında erişim kontrolünün en yaygın biçimi view' dir. View mantıksal bir tablodur, SQL' de VIEW komutu ile oluşturulur. Bu tablo JOIN ve SELECT gibi ilave SQL komutları tarafından elde edilen veritabanındaki veriyi kapsar. Eğer veritabanı sınıflandırılmış ise, o zaman view için kaynak, kullanıcının sınıflandırma düzeyinde veya altında olan veritabanının altkümesidir. Kullanıcılar viewlarında veriyi okuyabilir veya değiştirebilir. Fakat view daha üst sınıflandırma düzeyindeki veri erişimine kullanıcılara olanak vermez. Gerçekten eğer view uygun bir biçimde tasarlanırsa yüksek sınıflandırma düzeyindeki verinin varlığından habersiz olacaktır.

Veri view kaynağında neyi içerebildiğini tanımlamak amacıyla veritabanındaki tüm veri erişim sınıflandırmasına maruz kalmalıdır. Dennig tarafından belirtilen çeşitli potansiyel erişim sınıflarını kullanmıştır. Bunlar şunları içerir:

1. *Tip bağımlılığı* : Sınıflandırma veri ile ilgili niteliklere dayanılarak belirlenir.
2. *Değer bağımlılığı* : Sınıflandırma veri değerine dayanılarak belirlenir.
3. *Kaynak düzeyi* : Yeni verinin sınıflandırılması veri kaynağının sınıflandırılmasına eşdeğer ayarlanır.
4. *Kaynak etiketi* : Veri, veriyi giren kullanıcı tarafından veya kaynak tarafından isteğe göre sınıflandırılır.

Güvenlik hedefi, inference saldırısının tehdidinin azaltılmasını içerdiği zaman, verinin sınıflandırılması ve gerekli viewların gelişimi çok daha karmaşık olur.

Inference tipik olarak yüksek düzeyli veriden elde edilen düşük sınıflandırma düzeyindeki veriden yapılır. (Inference hakkında ayrıntılı bilgi bölüm 2.6.1' de verilmiştir.)

Veritabanlarını yetkilendirebilmek için kullanılan yöntemlerin bir kısmı şöyledir :

- *Şifre* : Kullanıcı veritabanına erişebilmek için geçerli şifreyi sağlamalı.
- *Kriptografi (Şifreleme)* : Sadece yetkili kullanıcı belirli bir algoritma ve belirli bir anahtar kullanarak şifreli mesajı çözebilir.
- *Dilleri kısıtlayarak kullanıcılara kullandırmak* : Yetkisiz kullanıcıların kullanamaması için diller kısıtlanır.
- *Erişim kontrol listesi* : Kontrol listesi sadece yetkili kullanıcıları kapsar, bundan dolayı listede olmayan herhangi biri veriye erişemez.
- *Veri bağımlı erişim kontrolü* : Veriye erişimde yetki kararı erişilen verinin değerine dayanır. Bundan dolayı yetki kararı, veri değerine erişildikten ve incelendikten sonra yapılır. Program çevirisi esnasında, veriye erişmek isteyen kaynak kodu, önceden depoladıkları yetki kurallarına karşı kontrol edilir.
- *Giriş/Çıkış yordamlarının kullanımı* : Gerekli I/O işlemini güvenlik yordamlarını çağırarak yapmak için programlanabilmeyi gerçekleştiren sistem.
- *Otomatik sorgu değişikliği* : Kullanıcıların veritabanının yetkisiz kısımlarını görmelerini önlemek için ilgili DBMS' in gerçekleştirdiği otomatik sorgu değiştirme kontrolü vardır. Veritabanı ile etkileşimli kullanıcı erişim ihlali içermeyen diğer form içerisindeki erişim kontrol algoritmaları tarafından hemen değiştirilir.

2.3.4 Kullanıcı Kimlik Doğrulama (Authentication)

Kimlik doğrulama mekanizması her bir veritabanı için çok önemlidir. Bu mekanizma ile veritabanının yetkisiz kullanımını önlenir. Kimlik doğrulama işleminin farklı yöntemleri vardır.

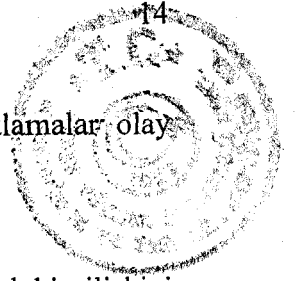
- *İşletim sistemi tarafından kimlik doğrulama* : Kullanıcılar, kullanıcı adı ve şifre belirtmeden işletim sistemi boyunca bazı veritabanlarına bağlanır. Bu durumda, kullanıcılar veritabanına eriştiği zaman, işletim sistemi, kullanıcıların kimlik denetimini yapar.
- *Network kimlik doğrulama servisi tarafından kimlik doğrulama* : Veritabanına bağlanılmak istenildiği zaman, kullanıcıları doğrulamak için kullanılan DCE, Kerberos ve SESAME gibi çeşitli network servisleri bulunmaktadır.
- *DBMS tarafından kimlik doğrulama* : İşletim sistemi veritabanı kullanıcı onayı için kullanılmadığında, veritabanının yetkisiz kullanımını önlemek için DBMS kullanıcıların kimlik denetimi yapar.

2.3.5 Güvenlik Kısıtlamaları

Güvenlik kısıtlamaları veriye güvenlik düzeylerini atayan kurallardır. Güvenlik kısıtlamalarının çeşitli tipleri şu şekilde tanımlanabilir : [4]

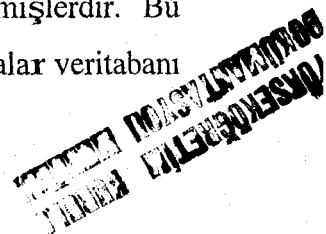
1. Veritabanını, ilişkiyi veya niteliği sınıflandıran kısıtlamalar. Bu kısıtlamalar basit kısıtlamalar olarak adlandırılır.
2. Birkaç verinin değerine bağlı olarak veritabanının herhangi bir parçasını sınıflandıran kısıtlamalar. Bu kısıtlamalar içerik esaslı (content-based) kısıtlamalar olarak adlandırılır.
3. Birkaç gerçek dünya olayının oluşmasına bağlı olarak veritabanının

herhangi bir parçasını sınıflandıran kısıtlamalar. Bu kısıtlamalar olay esaslı (event-based) kısıtlamalar olarak adlandırılır.

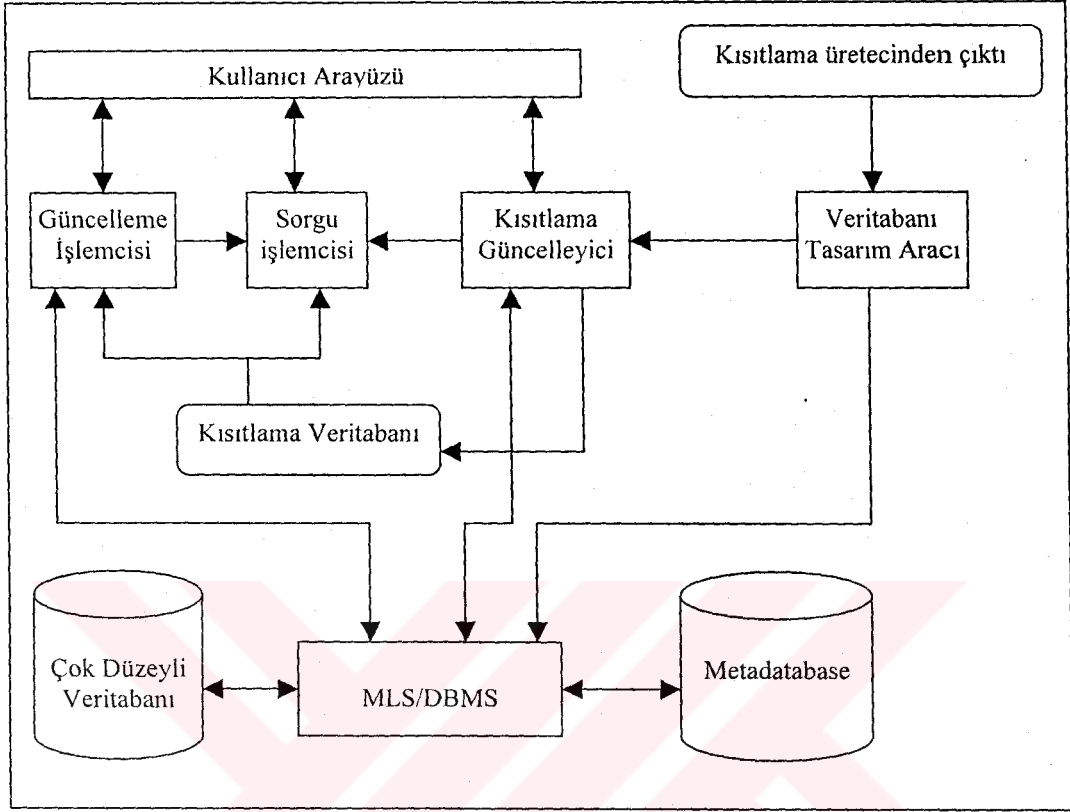


4. Veriler (satırlar, nitelikler, elemanlar vb. gibi) arasındaki ilişkiyi sınıflandıran kısıtlamalar. Bu kısıtlamalar ilişki esaslı (association-based) kısıtlamalar olarak adlandırılır.
5. Önceden yayımlanan bilgiye bağlı olarak veritabanının herhangi bir parçasını sınıflandıran kısıtlamalar. Bu kısıtlamalar yayım esaslı (release-based) kısıtlamalar olarak adlandırılır. Yayım esaslı kısıtlamalar iki tipte tanımlanmaktadır. Biri, diğer niteliğin herhangi bir değerinin yayımlanıp yayımlanmadığına bağlı olarak bütün niteliği sınıflandıran genel yayım kısıtlaması. Diğeri ise, diğer niteliğin değerinin yayımlanıp yayımlanmadığına bağlı olarak bütün niteliğin değerini sınıflandıran kişisel yayım kısıtlaması
6. Verinin toplamını sınıflandıran kısıtlamalar. Bu kısıtlamalar kümelenmiş (aggregate) kısıtlamalar olarak adlandırılır.
7. Birkaç verinin güvenlik düzeyine bağlı olarak veritabanının herhangi bir parçasını sınıflandıran kısıtlamalar. Bu kısıtlamalar düzey esaslı (level-based) kısıtlamalar olarak adlandırılır.
8. İçermeleri (uzatmaları) belirten kısıtlamalar. Bunlar mantıksal kısıtlamalar olarak adlandırılır.
9. Kısıtlamaları ve metadatayı sınıflandıran kısıtlamalar. Bunlar metadata kısıtlamaları olarak adlandırılır.

Güvenlik kısıtlamaları sınıflandırma politikasını uygular. Bu yüzden kısıtlamalar sadece yetkili kişiler tarafından işlenmeye zorunludur. Thursingham ve Ford kısıtlamaları işlemek için tümleşik bir yaklaşım dizayn etmişlerdir. Bu yaklaşıma göre, bazı kısıtlamalar sorgu işleme esnasında, bazı kısıtlamalar veritabanı



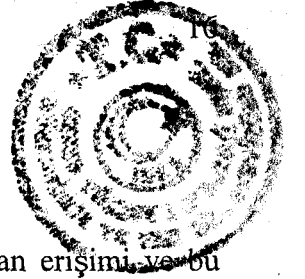
güncellemesi esnasında ve bazı kısıtlamalar da veritabanı dizaynı esnasında yürütülür. [4]



Şekil 2.1 Kısıtlama işlemi için tümleşik mimari

Her ne kadar sorgu kısıtlama işlemcisi, güncelleme kısıtlama işlemcisi ve veritabanı tasarım aracı ayrı modüller olsa da, tümü çok düzeyli dağıtık veritabanı sistemlerinde (MLS/DBMS) işletilen kısıtlamalara çözüm oluştururlar. Bu üç yaklaşım çok düzeyli (multilevel) ortamda işletilen güvenlik kısıtlamalarına tümleşik çözüm sağlar. Şekil 2.1' de Thurasingham ve Ford tarafından tasarlanan tümleşik mimari gösterilmektedir. [4] Bu mimaride, kısıtlama üretici tarafından üretilen kısıtlamalar ve şema, veritabanı tasarım aracı tarafından işletilirler. Değiştirilmiş kısıtlamalar, kısıtlama veritabanını güncellemek amacıyla kısıtlama güncelleyiciye verilir. Şema metadatabase'de depolanacak olan MLS/DBMS' e verilir. Kısıtlama veritabanındaki kısıtlamalar sorgu ve güncelleme kısıtlama işlemcisi tarafından kullanılırlar. Veri ve kısıtlamaların değiştiği dinamik ortamda, sorgu kısıtlama işlemcisi tüm ilgili kısıtlamaları inceleyerek kullanıcıların yetkisiz veriyi elde etmemelerini garanti etmelidir.

2.4 Veritabanı Eriřim Kontrol Teknikleri



Eriřim kontrolü, kaynaklara sadece yetkili personel tarafından erişimi ve bu personelin sadece yetkili faaliyetleri gerçekleřtirebilmesini garanti ettiğimiz veya garanti etmeyi denediğimiz mekanizmadır.

2.4.1 İsteęe Baęlı Eriřim Kontrolü

Çoęu DBMS' ler isteęe baęlı koruma politikasını uygulayarak erişim kontrolünü yürütür. İsteęe baęlı koruma politikaları, kullanıcı kimliğinin temelinde ve sistemdeki her kullanıcı ve nesne için belirtilen kuralları dikkate alarak kullanıcının bilgiye erişimini yönetir. Nesneye erişmek için kullanıcının talebi belirtilen yetkilere karşı kontrol edilir; eęer kullanıcı belirli bir durumda nesneye erişime yetkili ise, erişim kabul edilir; aksi halde reddedilir.

İsteęe baęlı korumayı uygulayan sistemlerde yetkileri yönetmek için uygulanabilen çok politika vardır. Bazı örnekler;

- *Merkezi yönetim* : Sadece bazı ayrıcalıklı kullanıcılar yetkileri verebilir ve geri alabilir.
- *Sahiplik temelli yönetim* : Nesnenin yaratıcısı nesneye erişimleri vermek ve geri almaya izin verir.
- *Merkezi olmayan yönetim* : Nesnenin sahibinin izniyle dięer kullanıcılara nesne yetkilendirmelerini vermek ve geri almak için izin verilir.

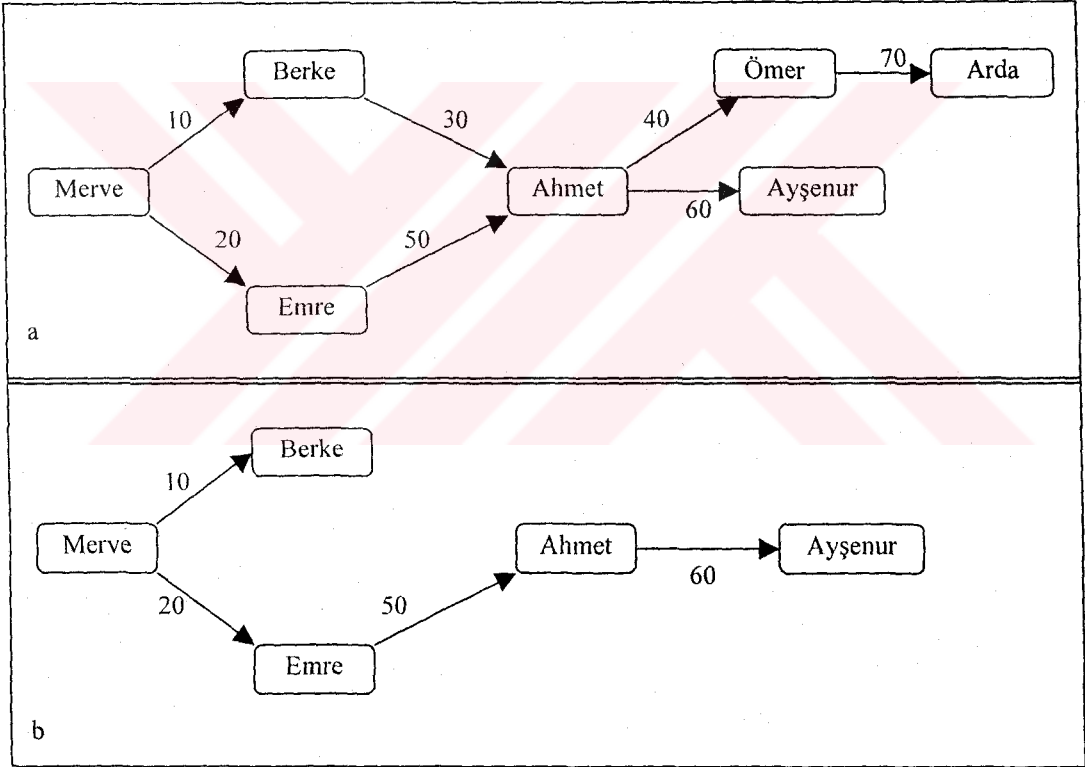
İsteęe baęlı politikalar, esnek olmalarından dolayı ticari sistemlerde kullanılır, farklı koruma gereksinimli ortamların çeřitlilięine elverişli olarak yapılır.

System R Yetkilendirme Modeli

Ticari DBMS' in parçası olarak gerçekleştirilen yetkilendirme modelidir. İlk

olarak Griffiths ve Wade tarafından geliştirilmiş ve daha sonra Fagin tarafından revize edilmiştir.[5] Bu model kullanıcılara verilen ayrıcalıkları dikkate alır. Tablolar üzerinde kullanılan olası ilişkisel veritabanı ayrıcalıkları *select*, *insert*, *delete*, *update* dir.

Kullanıcıya verilen yetkiler kullanıcının ayrıcalıklarını oluşturur. Kullanıcı sahip olduğu ayrıcalıkların bir kısmını başka kullanıcılara aktararak bu ayrıcalıkları paylaşmak isteyebilir. Aktarma yoluyla bir ayrıcalığa kavuşan her yeni kullanıcının bu ayrıcalığı başkalarına aktarma yetkisi vardır. Ayrıca her kullanıcı aktardığı ayrıcalıkları iptal etme yetkisine de sahiptir. Böylece veritabanı yöneticisi tarafından bir kullanıcıya verilen bir yetki, bu kullanıcı tarafından başkalarına aktarılabilir. Yetki aktarma işlemi zincirleme devam edebilir.



Şekil 2.2 Ayrıcalıkların aktarımı ve geri alınması

Kullanıcının başka bir kullanıcıya vermiş olduğu ayrıcalıkları geri alırken, geri alma işlemi tekrarlanan bir şekilde gerçekleşir. Geri alma işleminin nasıl gerçekleştiği Şekil 2.2' de örnekle gösterilmiştir. Şekil 2-2-a' da görüldüğü üzere "Ahmet", "Berke" den almış olduğu ayrıcalıkları "Ömer" e aktarmış ve de "Ömer" de "Ahmet" ten almış olduğu ayrıcalıkları "Arda" ya aktarmıştır. "Berke", "Ahmet"

e vermiş olduğu ayrıcalığı geri alması durumunda “Ahmet” in “Ömer” e vermiş olduğu ve de “Ömer” in “Arda” ya vermiş olduğu ayrıcalıklar da geri alınacaktır. (Şekil 2-2-b)

2.4.2 Zorunlu Erişim Kontrolü

Zorunlu erişim kontrolü, Bell ve La Padula tarafından işletim sistemleri için tasarlanan modele dayanır.[1] Bu kontrol politikası, güvenliğin farklı düzeylerinde özneler ve nesneleri sınıflandırır.

Genellikle, kaynaklara erişim kontrolü $\langle s, o, a \rangle$ anlamına gelen yetki kuralları kullanılarak yapılır. Özne s , erişilen nesne o , yapılan eylem a olarak tanımlanmıştır

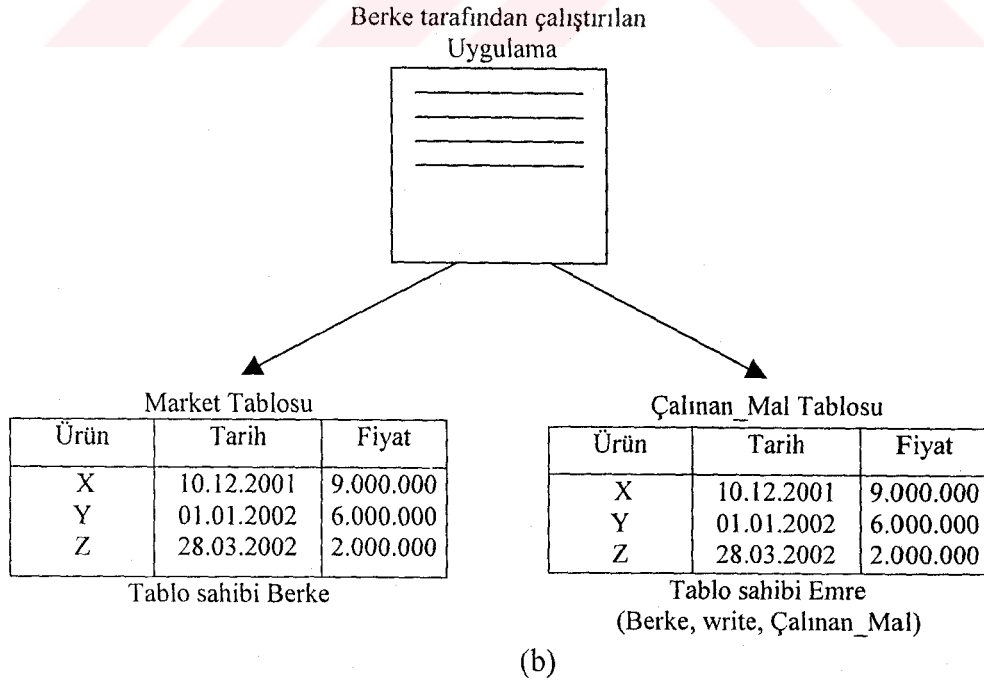
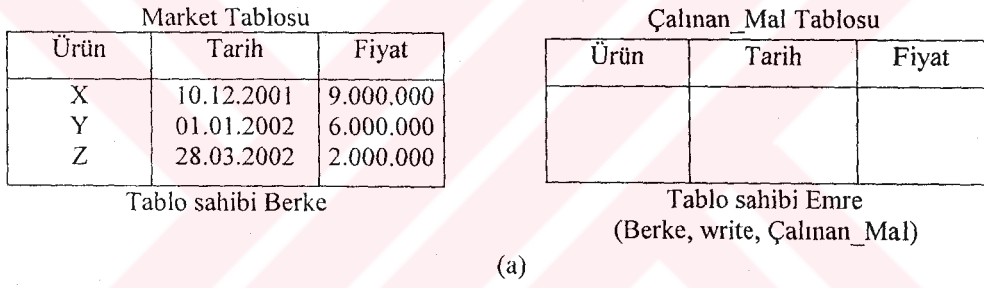
- Özneler nesnelere yetki verebilmek için eriştiği varlıklardır. Her ne kadar bunlar genelde kişisel kullanıcılar olsa da, aynı zamanda kullanıcı grupları, roller veya hatta kullanıcının adına gerçekleştirilen işlemler de olabilir.
- Nesneler kontrol etmek istediğimiz erişilen veritabanı elamanlarıdır.
- Eylemler, gerçekleştirilebilen olası işlemlerdir ve ilişkisel veritabanlarında bu işlemler genellikle *select*, *insert*, *delete* ve *update* şeklindedir.

Erişim sınıfı iki bileşen içerir : güvenlik düzeyi ve grup kümeleri. Güvenlik düzeyi, hiyerarşik olarak düzenlenen kümenin elemanıdır. Düzeyler Top Secret (TS), Secret (S), Confidential (C) ve Unclassified (U) olarak dikkate alınır ve aralarında $TS > S > C > U$ şeklinde bir ilişki vardır. Grup kümeleri ise düzensiz olarak kurulu (örneğin; NATO, Nükleer Enerji Santrali, Ordu gibi). Erişim sınıfları kısmen şu şekilde düzenlenir : Eğer c_1 'in güvenlik düzeyi c_2 'nin güvenlik düzeyine eşit veya daha büyük ve c_1 'in grupları c_2 'nin gruplarını içeriyor ise erişim sınıfı c_1 erişim sınıfı c_2 'yi kapsar. Ne $c_1 \geq c_2$ ne de $c_2 \geq c_1$ değil ise c_1 ve c_2 sınıfları *birbirleriyle kıyaslanamaz* denir.

Nesneyle ilgili erişim sınıfının güvenlik düzeyi nesneyi içeren bilginin hassasiyetini yansıtır. Kullanıcıyla ilgili erişim sınıfının güvenlik düzeyi ise, erişim yetkisi olmayan kullanıcılara hassas bilgiyi göstermeyen kullanıcı güvenilirliğini yansıtır.

Zorunlu koruma sistemlerindeki erişim kontrolü Bell ve Lapadula tarafından belirtilen aşağıdaki iki prensibi temel alır : [5]

- Eğer öznenin güvenlik düzeyi nesnenin güvenlik düzeyini kapsıyorsa, özne nesneyi okuyabilir.
- Eğer öznenin güvenlik düzeyi nesne düzeyinin aynısı ise özne nesneye erişip yazabilir. (Özne nesne düzeyini değiştirebilir)



Şekil 2.3 Truva atı örneği

Bu prensipler, düşük veya kıyaslanamaz düzeydeki nesneler içine direkt erişim için hassas nesnede bilgiyi engeller. Örneğin; şekil 2.3' de, eğer "Berke" "Market" tablosuna okuma erişimine izinli değilse, zorunlu erişim kontrolü altında "Market" tablosu, "Berke" için verilen erişim sınıfından daha yüksek veya kıyaslanamaz erişim sınıfına sahiptir. Fakat, o zaman "Market" tablosunu okuyabilen bir özne "Çalınan_Mal" tablosuna yazamaz ve truva atına bu yüzden işlevini tamamlamak için izin verilmez.

Erişim kontrol politikasını desteklemek amacıyla, düzeyler yoluyla veriyi sınıflandırmak için izin verilen veritabanı yönetim sistemini kullanmak gereklidir. Düzeyler yoluyla veriyi sınıflandırabilen bu veritabanları, "çok düzeyli (multilevel)" veritabanları olarak bilinir.

Çok düzeyli erişime sahip bir DBMS için özel problemler mevcuttur. Çok düzeyli erişim sisteminde, kullanıcılar tam veri erişiminden kısıtlanırlar. Belirli veri öğelerine kullanıcı erişimini kısıtlayan politikalar, gizlilik gereksinimlerinden meydana gelebilirler (Kullanıcı sadece konuyla ilgili bilgiye erişir.)

Veri çakışmalarından dolayı veri elemanlarının sınıflandırılması basit bir görev değildir. İlk hedef hassas olmayan tüm veriye erişimi veritabanı kullanıcılarına vermektir. İkinci hedef, yetkisiz gözlem veya inference' dan hassas veriyi korumaktır.

Diğer bir problem veri güvenlik sınıflandırılmasıdır. Millen ve Lunt' a göre bir veri elemanı sınıflandırılacağı zaman 3 boyutta sınıflandırılabilir. [2]

1. Veri sınıflandırılabilir.
2. Verinin varlığı sınıflandırılabilir.
3. Veriyi sınıflandırmak için gerekli sebep sınıflandırılabilir.

İlk boyutu değerlendirmek en kolay olandır. Sınıflandırılmış veriye erişim basitçe reddedilir. Diğer iki boyut daha çok düşünme ve daha çok yaratıcı stratejiler

gerektirir. Örneğin; yetkisiz bir kullanıcı, varlığı sınıflandırılan bir veriyi isterse, sistem bunu nasıl yanıtlar? Kötü bir şekilde planlanan yanıt, imkan dahilinde uzlaşılan veri hakkında inference yapması için kullanıcıya izin verir.

Halen daha, yüksek güvence ile zorunlu politikayı uygulayan işletim sistemleri vardır. Ancak, ilişkisel veritabanlarına uygulandığı zaman zorunlu politika çeşitli karmaşıklıklar ortaya çıkartır.

Zorunlu Etiketleme

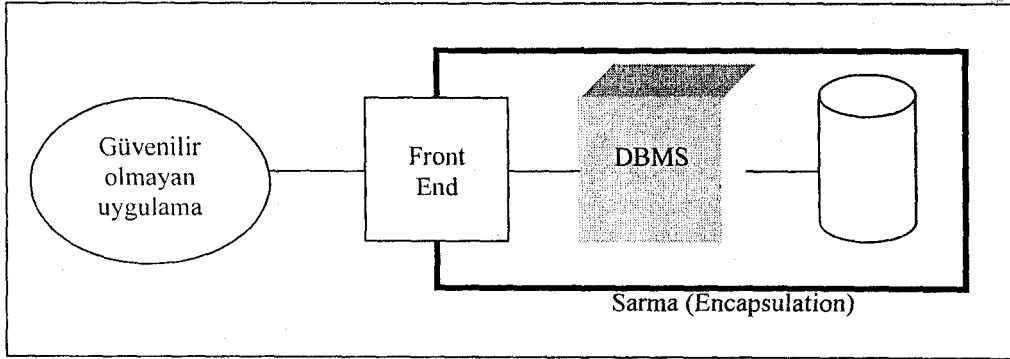
Güvenlik etiketini vermek için sistem içerisine veri giren kullanıcıya izin verilir ve bütün kullanıcılar için erişim yetkisi korunur. Eğer kullanıcının erişim yetkisi veri girildiği zaman en azından veriye verilen etiket kadar yüksek olursa, kullanıcı sadece bazı verileri öğrenebilir.

En ufak bir hata veya atlama zorunlu etiketleme sistemini başarısız yapabileceğinden dolayı, güvenliğin olması gibi çalışması için yeterli karmaşık yazılımı geliştirmek ve test etmek çok zordur. Bundan dolayı sistemin yazılımını göreceli olarak küçük güvenilir kısım ve büyük güvenilir olmayan kısma bölmek alışlagelmiştir. Öyle ki, kontrolün doğruluğu güvenilir olmayan yazılımın davranışına bakılmaksızın mutlaka garanti edilir. Küçük güvenilir kısım, güvenilir olmayan yazılım kontrolünü kabul ettirir. Tipik olarak güvenilir kısım işletim sistemi çekirdeğidir ve uygulamalar güvenilir olmayan kısım olur.

Zorunlu etiketleme yöntemlerini üç bölümde inceleyebiliriz :

1. Bütünlük Kilidi
2. SeaView
3. SWORD

2.4.2.1 Bütünlük Kilidi (Integrity Lock)



Şekil 2.4 Bütünlük kilidi (Integrity Lock) mimarisi

Veritabanlarında zorunlu etiketlemeyi sağlayan ilk girişimlerden biri bütünlük kilidi yaklaşımıdır. Buradaki fikir, sorgular ve sonuçlarını işleten güvenilir front-end ile bu veritabanında zorunlu etiketlemeyi kabul ettirmek için ve veriyi depolamak ve işletmek için güvenilir olmayan ilişkisel veritabanını kullanır. Veritabanı tüm erişimlerin front-end' inden geçerek ilerlediğinden emin olmak için bazı yolla sarılır.[6] Bu şekil 2.4' de gösterilmektedir.

	Normal Uygulama	Bütünlük Kilidi Uygulaması																				
Tablo	<table><tr><th colspan="2">Personel</th></tr><tr><th>ID</th><th>Ad</th></tr><tr><td>123</td><td>Ömer</td></tr><tr><td>456</td><td>Ayşenur</td></tr></table>	Personel		ID	Ad	123	Ömer	456	Ayşenur	<table><tr><th colspan="3">Personel</th></tr><tr><th>Label</th><th>ID</th><th>Ad</th></tr><tr><td>[L]</td><td>123</td><td>Ömer</td></tr><tr><td>[H]</td><td>456</td><td>Ayşenur</td></tr></table>	Personel			Label	ID	Ad	[L]	123	Ömer	[H]	456	Ayşenur
Personel																						
ID	Ad																					
123	Ömer																					
456	Ayşenur																					
Personel																						
Label	ID	Ad																				
[L]	123	Ömer																				
[H]	456	Ayşenur																				
Select	SELECT Ad FROM Personel WHERE ID=456	SELECT Ad FROM Personel WHERE Erişim Yetkisi>=Label AND ID=456																				
Update	UPDATE Personel SET Ad="Merve" WHERE ID=123	UPDATE Personel SET Ad="Merve" WHERE Erişim Yetkisi>=Label AND ID=123																				

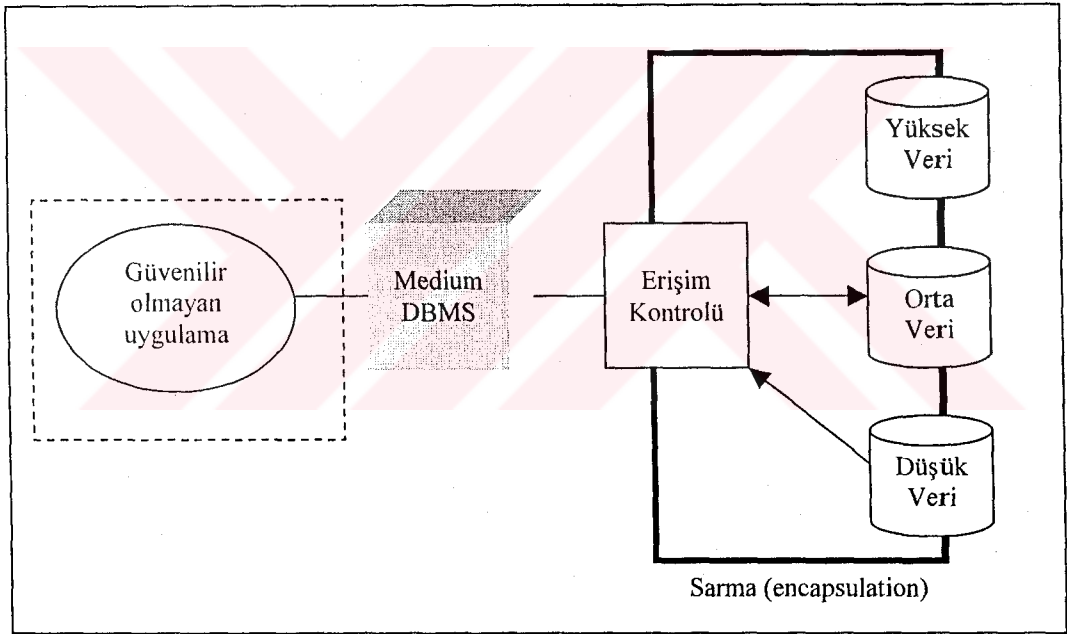
Şekil 2.5 Bütünlük kilidi veritabanı ve gerçekleştirilmesi

Etiketleme satır seviyesinde yapılır. Bu her bir tabloya ilave kolon ekleyerek uygulanır ve uygulamalar tarafından oluşturulan sorgular front-end' den geçerek ilerler gibi değiştirildiğinden dolayı kullanıcının erişim yetkisi olmadığı satırlar ihlal edilebilir. Şekil 2.5' de back-end veritabanında uygulanan ve uygulama tarafından görülen bir tablonun basit bir örneği verilmiştir. Burada ve diğer örneklerde low, medium ve high etiketleri, sırasıyla L, M ve H kısaltmalarıyla kullanılmıştır.

Bazı durumlarda satır etiketlerinin uygunsuz değiştirilebileceği varsayılmalıdır. Front-end bunu engelleyemez, fakat ikisini birden bağlamak için her bir satır ve etikete dijital imza uygulayabilir.

2.4.2.2 SeaView

İlişkisel veritabanlarına zorunlu etiketlemeyi eklemek için üzerinde çalışılan belki de en çok bilinen yöntem SeaView yöntemidir. [6] SeaView uygulaması çoklu güvenilir olmayan DBMS' leri kullanır. Her bir DBMS düşük etiketler ile veritabanından veriyi okuyabilir. Her bir DBMS gözleyebildiği tüm tablolar üzerinde özel güncellenebilir view ile uygulamalar sağlar. Bu view her bir alan üzerinde etiketler içerir.



Şekil 2.6 SeaView Mimarisi

DBMS' in kullanıcının erişim yetkisinden küçük etiketlerle veritabanını değiştiremediğinden ve kullanıcının erişim yetkisinden büyük etiketlerle veritabanını gözlemleyemediğinden emin olmak için sadece istenilen güvenilir yazılıma ihtiyaç vardır. SeaView çözüm yolunun mimarisi şekil 2.6' da gösterilmektedir.

SeaView veritabanında kabul edilen kısıtlamalar, güvenilir olmayan DBMS tarafından zorla uygulanırlar ve onların mecbur edilmeleri, temeli oluşturan veride

güvenilir erişim kontrol fonksiyonlarının kısıtlamalarına bağlı olur. Bu sebeple SeaView, herhangi bir inference probleminin rahatça zorunlu etiketleme sağlamasına açıktır.

SeaView tablosu, her bir SeaView veri kolonu için güvenilir olmayan veritabanlarının her birinde ayrı gerçekleştirme tablosu kullanarak uygulanır. Her bir gerçekleştirme tablosu birincil anahtar (primary key) için bir ve diğeri için bir olmak üzere iki kolona sahiptir.

PERSONEL		
Id	Ad	Meslek
123 [L]	Emre [L]	Araştırmacı [L]
456 [L]	Berke [L]	Doktor [L]
456 [L]	Berke [L]	Mühendis [H]
789 [L]	Ahmet [H]	NULL [H]

Şekil 2.7 SeaView tablosu örneği

SeaView tablosundaki satırlar kullanıcıya görünebilir olan gerçekleştirme tablolarında tüm satırlar birleştirilerek oluşturulur. Her bir satırdaki alanlar, değerini sağlayan tablonun etiketi tarafından etiketlenir. (Şekil 2.7 ve 2.8)

Ad [L]		Meslek [L]		Ad [H]		Meslek [H]	
Id	Ad	Id	Ad	Id	Ad	Id	Ad
123	Emre	123	Araştırmacı	789	Ahmet	456	Mühendis
456	Berke	456	Doktor				

Şekil 2.8 SeaView örneği uygulaması

SeaView veritabanı, normal veritabanı gereksinimlerine benzer olmayan karmaşık anlama sahiptir. Özellikle, SeaView birincil anahtarlarda eşsizlik kısıtlamalarını zorunlu kılmaz. SeaView tarafından desteklenen birincil anahtar kısıtlaması aynı birincil anahtar değeri için iki satıra izin verir. (yine de, onların alanlarının etiketleri farklı olması gerekir). Sonuçta SeaView' ın gerçek birincil anahtarı, birincil anahtar değerinden ve tüm alanların etiketlerinden oluşan bir satırdır. Bundan Polyinstantiation olarak söz edilir.

Polyinstantiation ile bütünlüğün genel kavramına sahip veritabanı uygulamalarını uygulamak zordur. Yukarıdaki örnekte veritabanı, personelin her bir

üyesinin bir ad ve bir mesleğe sahip olduğu temel bütünlük gereksinimini uygulaması olanaksızdır.

Eşsizlik kısıtlamasını uygulamak veritabanı için neden mümkün değildir? Çünkü, bütün anahtar değerleri, eşsiz olduğundan emin olmak amacıyla gözlenilmelidir ve bu, kullanıcıya bazı değerler hakkında bilgiyi ortaya çıkarır. Özellikle eğer çalışma anahtar değerini çoğaltmak için yapılırsa, isteğin reddedilmesi her zaman mevcut değeri açığa çıkartır. Eğer diğer kullanıcının yetkilendirilmesinden daha yüksek sınıflandırılırsa, kullanıcı bilmemesi gereken bir şeyi biliyor olur.

Bu nedenle, çoğu kez Polyinstantiation veritabanlarında zorunlu etiketlemenin kaçınılmaz sonucu olduğu tartışılır. Polyinstantiation, SeaView' da kaçınılmazdır. Çünkü alanın etiketi hem varlığı hem de değeri kapsar. Bu SWORD DBMS tarafından ispatlanmıştır.

2.4.2.3 SWORD

SeaView' a benzer şekilde SWORD DBMS, alan seviyesinde zorunlu etiketlemeyi destekler. Fakat SeaView' dan farklı olarak, klasik ilişkisel veritabanı bütünlük kısıtlamalarını koruyarak yapar. [6]

Düşük uygulama için görüş			Yüksek uygulama için görüş		
Id	Ad	Meslek	Id	Ad	Meslek
123 [L]	Emre [L]	Araştırmacı [L]	123 [L]	Emre [L]	Araştırmacı [L]
456 [L]	Berke [L]	Erişim yetkisi yok [H]	456 [L]	Berke [L]	Mühendis [H]

Şekil 2.9 SWORD Veritabanı örneğinin iki görüşü

SeaView ve SWORD arasındaki anahtar farklılık, SWORD etiketi sadece alanın değerini kapsayıp varlığını kapsamaz iken, SeaView' ın alan etiketleri alanın değerini ve varlığını yönetir. SWORD tablosunda satırın varlığı her zaman düşük ve kullanıcının yetkilendirilmesinden daha yüksek etiketlendiği alanda etiketlenir. Bu yetkisiz olarak adlandırılan özel değeri içerir. Bu şekil 2.9' da gösterilmiştir

26

Satırın varlığı düşük etiketlenildiği için, satırlar düşük yetkilendirme ile uygulamalar tarafından sadece yaratılabilir ve silinebilirler. Eğer yeni satır anahtar kolonda tüm alanlar düşük etiketlenildiği zaman eklenirse, yeni anahtar değerinin tek olmasını kanıtlamak için tüm anahtar değerlerini gözlemek mümkündür. Eğer herhangi bir satır kullanıcının yetkilendirilmesinden daha yüksek sınıflandırılmış anahtar değerine sahipse, eşsizlik için kontrol etmek mümkün değildir. Ancak kısıtlamanın kontrol edilemediğini ortaya çıkarmak mümkündür. Çünkü alanların etiketi kendilerini düşük etiketler. Böylece ekleme işlemi, herhangi bir hassas bilgiyi açığa çıkarttığından dolayı kabul edilmeyebilir.

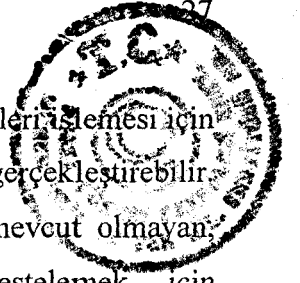
Anahtar alanına yüksek etiket verilmeli, fazla satır eklenmemelidir. Çoğu durumda bu denial-of-service (servislerin reddi) problemini oluşturur fakat anahtar kolona etiket kısıtlamasını uygulayarak kolayca önlenabilir

Her ne kadar SWORD polyinstantiation' dan kaçınsa da kullanımı yine de zordur. Çünkü çok düzeyli satırlar sadece iki aşamada yaratılabilir. Düşük yetkilendirme ile çalıştırılan ilk uygulama satır yaratmalı ve yüksek yetkilendirme ile çalıştırılan ikinci uygulama herhangi bir yüksek veriyle satırı güncellemelidir. Buda ortaya sıkıntılar çıkartır.

2.4.3 Rol Temelli Erişim Kontrolü

Rol temelli erişim kontrolü (RBAC), isteğe bağlı ve zorunlu erişim kontrol metotlarına alternatiftir. Rol temelli erişim kontrolünde izinler roller ile ilgilidir ve kullanıcı rollerin üyesi yapılır. Bu izin yönetimini çokça basitleştiren bir yoldur. Roller, organizasyondaki benzer fonksiyonlar ve sorumluluklara sahip kullanıcıların oluşturmuş olduğu fonksiyonel grupları temsil eder. [7]

Rol temelli erişim kontrol modellerinde erişim, sistemde kullanıcıların faaliyetlerine ilişkin kurulan işin temelinde düzenlenirler. Rol özel çalışma faaliyeti ile ilgili bir takım eylemler veya sorumluluklar olarak tanımlanır. Her bir kullanıcı için bütün erişimleri belirtmek yerine, nesneler üzerindeki yetkilendirmeler roller için belirtilebilir. Kullanıcılara roller atanarak yetkiler verilir. Roller, kullanıcıların



nesnelere erişimine arabuluculuk eder : her bir kullanıcıya belirli rolleri işlemesi için yetki verilir ve kullanıcı, rolün temelinde nesnelere erişimi gerçekleştirebilir. Özellikle, rol temelli politikalar güncel veritabanı sistemlerinde mevcut olmayan separation-of-duties (görevlerin ayrılması) gereksinimini desteklemek için kullanılabilirler. Separation-of-duties, kullanıcının kendi sistemini hatalı kullanmak için yeterli ayrıcalıkların kullanıcıya verilmemesi ilkesiyle ilgilidir. Örneğin; bir tabloya yetki veren kişi onu hazırlayan birinden farklı olmalıdır. Separation-of-duties, statik olarak çelişen rolleri (aynı kullanıcı tarafından işletilemeyen roller gibi) tanımlayarak veya dinamik olarak erişim süresince kontrolleri uygulayarak yürütülmelidir.

2.5 Dinamik Yetkilendirme

Yetkilendirme problemini çözmek için klasik yaklaşım RBAC modeline dayanır. Kullanıcı ayrıcalıkları roller tarafından statik olarak tanımlanır. Bu yaklaşım yüksek düzeyde sistem güvenliği sağlar.

Dinamik yetkilendirme, kullanıcılara ihtiyacı olduğunda gerekli yetkilerin verilmesi işlemidir. Bunu başarmak için RBAC yaklaşımı tek başına yeterli olmamaktadır. Bunun için Olivier et. al. tarafından önerilen dinamik erişim kontrolü yaklaşımı kullanılmaktadır.[8] Bu yaklaşım RBAC yaklaşımının güvenlik özelliklerinin ihlali olmadan kullanıcı ayrıcalıklarını işlemek için olanak sağlar. Bu tanıma göre ayrıcalıklar statik yetkilendirme işlemi esnasında otomatik olarak verilmez, fakat her bir işlem için ilave dinamik yetkilendirme gerçekleştirilir. Bir başka deyişle, klasik RBAC modeli statik potansiyel yetkilendirme sağlar iken, dinamik erişim kontrolü çalışma zamanında ilave kısıtlamalar sağlar. Dinamik yetkilendirme ile ilgili Gudes ve Tubman tarafından örnek bir çalışma yapılmıştır.[9]

2.6 Veritabanı Güvenlik Problemleri

Veritabanlarının gelişimi güvenlik problemlerini de beraberinde getirmiştir. Bu problemlerden en sık rastlanılan iki tanesi aşağıda açıklanmıştır.

2.6.1 Inference Problemi

Veritabanı güvenliğindeki tehlikelerden biri, yetkili kullanıcılar tarafından veritabanının hatalı kullanımıdır. Grubun dışındaki kimselere kişisel bilgilerin satılması örnek verilebilir. İstatistiksel veritabanı sistemlerindeki kişisel bilgilerin korunması için çeşitli erişim kontrol mekanizmaları önerilmektedir. Bu mekanizmalar istatistiksel veritabanları için belirli bir biçimde dizayn edilirler, genel amaçlı veritabanı sistemleri için uygulanamazlar. Çok düzeyli güvenli veritabanı sistemlerinde, inference olarak adlandırılan saldırıların tipi tanımlanır. Kullanıcı, veriye direkt erişim olmadan bilgiyi çıkarsamak (anlamak) için yasal veriyi kullandığı zaman inference gerçekleşir. Lunt, bazı inference problemlerinin veritabanı şemalarının yeniden dizayn edilerek ve özelliklerin tam anlamıyla sınıflandırılmasıyla önlenebileceğine işaret etmiştir.[10] Ancak, yeniden dizayn edilen veritabanı şeması güncelleme bozukluğu nedeniyle veri çoğalmasıyla sonuçlanır. Aynı zamanda, var olan uygulama programlarında da modifikasyonlar gerektirir.

Inference lar kullanıcılara veri erişimi için açıkça erişim hakkının verildiği yer olan isteğe bağlı erişim kontrollü sistemlerde gerçekleşir. Bazı durumlarda, kullanıcıların çalışmalarını engellemek için ihtiyaçlarından daha fazla erişim hakları verilir. Kullanıcıların veritabanını hatalı kullanmadıklarından emin olmak için, erişimlerini gözetleme ihtiyacı duyulur.

Kullanıcı erişimlerini takip etmek için basit bir yol, her bir kullanıcı sorgusunu incelemek ve hassas veriye erişen herhangi bir sorguyu reddetmektir. Ancak, kullanıcının veritabanından veriyi çıkarsamak (anlamak) için şüpheli olmayan sorgular dizisini kullanması mümkündür. Motro et al. benzer problemleri adreslemiştir, fakat çalışmaları inference saldırılarının yerine ortaya çıkan toplamaya odaklanır. İstatistiksel veritabanı güvenlik topluluğunda, kişisel kayıtları korumak için query-set-size kontrolü, hücre baskısı ve veri tedirginliği gibi teknikler önerilmiştir. Ancak bu teknikler genel amaçlı sorguları kullanan inference'ları algılamak için uygun değildir.



Her bir kullanıcı sorgusunun sonucu bir takım dönen satırlardır. Satırın birincil anahtarı dönmedikçe kullanıcı dönen her bir satırı tanımlayamaz. Ancak satırın nitelik değerlerinin belirli grubu, satırı eşsiz olarak tanımlayabilir. Eşsiz tanımlama kuralı bu durumu ele alır. Dönen satırı tanımlamak için başka bir yol, hali hazırda tanımlı olan dönen diğer satır ile karşılaştırmaktır.

Dönen satırların iki kümesi arasında olabilecek iki ilişki vardır. Bir olasılık sorguyla her bir dönen satır t_1 için, diğer sorguyla dönen satır t_2 bulunur, öyle ki t_1 ve t_2 veritabanında aynı satıra uyarlar. Bu durum inference kuralı kapsamında ele alınır. Diğer olasılık, sadece sorguyla dönen bazı satırların diğer sorgunun dönen bazı satırlarına uymasındır. Overlapping (örtüşme) inference kuralı, her iki sorguya mahsus ilgili dönen satırı tanımlar. Tamamlayıcı inference kuralı dönen satırın iki kümesi arasında farkı alarak ilgili dönen satırı tanımlar. İşlevsel bağımlılıklar inference kuralı, şema düzeyinde inference algılama şemasını simule etmek için tanıtılır.

İki sorgu arasında ilgili dönen satır bir kez tanımlanır, kullanıcı çıkarsılan (anlaşılan) sorgular üretir. Kullanıcı veritabanında doğrudan yayımlananın dışında anlaşılan sorguyla dönen satırı bilir. Örneğin; kullanıcı her iki sorguya mahsus dönen satır ile veya bir sorgudan dönen satır ile (diğer sorgudan değil) yeni sorguyu çıkarsayabilir (anlayabilir). Kullanıcı aynı zamanda tek sorgu içerisinde çeşitli sorguları birleştirebilir. [10]

2.6.2 Polyinstantiation

Polyinstantiation aynı isimde çoklu veri nesnelerinin aynı zamanda var olmasıyla ilgilidir. [11] Çoklu instantiation lar onların erişim sınıfları tarafından ayırt edilirler. Polyinstantiation aşağıda da gösterildiği gibi ilişkiler (veya veritabanının tamamı), satırlar ve veri öğelerini etkileyebilir. [5]

- Polyinstantiated ilişkiler aynı ilişki ismi tarafından tanımlanan ilişkililerdir fakat şemaları farklı erişim sınıflarına sahiptir.
- Polyinstantiated satırlar (varlık polyinstantiation olarak da adlandırılır)

aynı birincil anahtarlı satırlardır fakat farklı erişim sınıflarına karşı birincil anahtar ile ilgilidir.

- Polyinstantiated öğeler (nitelik polyinstantiation olarak da adlandırılır) farklı erişim sınıflarına sahip niteliğin değeridir fakat aynı birincil anahtar ve birincil anahtar nitelikleri için aynı erişim sınıflarıyla ilgilidir.

Farklı sınıflardaki özneler aynı ilişkilerde işletmek için izin verilmesi durumunda Polyinstantiation oluşur. Polyinstantiation aşağıdaki iki durumda ortaya çıkar :

1. Düşük kullanıcı, halihazırda daha yüksek veya kıyaslanamaz düzeydeki veriyi içeren alana veri eklediği zaman.
2. Yüksek kullanıcı, halihazırda düşük düzeydeki veriyi içeren alana veri eklediği zaman.

1 durumunda oluşan polyinstantiation, görünmeyen polyinstantiation olarak tanımlanır. Polyinstantiation' u oluşturan işlemi isteyen kullanıcı, polyinstantiation' un meydana geldiğinden haberdar olmadığı anlamına gelir. Buna karşın, 2 durumunda oluşan polyinstantiation, görünür polyinstantiation olarak tanımlanır.

2.6.2.1 Görünmeyen Polyinstantiation

Düşük kullanıcı, yüksek ve kıyaslanamaz düzeyde varolan satır ile aynı birincil anahtarlı satırı eklemek için sisteme sorar. DBMS aslında üç seçme hakkına sahiptir :

1. Halihazırda aynı birincil anahtar ile satır var olduğunu ve kayıt ekleme işlemini geri çevirdiğini kullanıcıya bildir.
2. Düşük düzeyde eklenen yeni satır ile daha yüksek veya kıyaslanamaz düzeydeki var olan satırı değiştir.
3. Daha yüksek veya kıyaslanamaz düzeyde var olan satırı değiştirmeksizin

düşük düzeydeki satırı ekle.

Seçenek 1 sakıncalıdır çünkü göremediği yüksek düzeyli bilginin kullanıcılarına bilgi vermeyi ifade eder, bu nedenle sinyal verme kanalı üretir.

Seçenek 2 de sakıncalıdır çünkü düşük kullanıcıya göremediği verinin üzerine yazmak için izin verir, böylece bütünlük tehlikeye sokulur.

Sadece seçenek 3 kabul edilebilirdir. Sonuç olarak, polyinstantiated varlık ortaya çıkar.

Halihazırda, kullanıcının değiştirmek istediğinden daha yüksek veya kıyaslanamaz düzeydeki bir değeri içeren alanı güncellemek için aynı adımlar uygulanır. Bunu görmek için, aşağıdaki örnekleri dikkate alalım :

PERSONEL						
Ad	C _{Ad}	Departman	C _{Departman}	Maaş	C _{Maaş}	TC
Merve	Low	Dep1	Low	600M	Low	Low
Omer	High	Dep2	High	750M	High	High
Aysenur	Low	Dep1	Low	700M	High	High

Şekil 2.10 Çok düzeyli ilişki örneği

Örnek 1 Şekil 2.10' da gösterilen çok düzeyli PERSONEL tablosunu dikkate alalım ve düşük kullanıcının aşağıdaki işlemi yaptığını varsayalım :

INSERT INTO Personel VALUES (Omer, Dep1, 600M)

Ad	C _{Ad}	Departman	C _{Departman}	Maaş	C _{Maaş}	TC
Merve	Low	Dep1	Low	600M	Low	Low
Omer	High	Dep2	High	750M	High	High
Aysenur	Low	Dep1	Low	700M	High	High
Omer	Low	Dep1	Low	600M	Low	Low

Şekil 2.11 Polyinstantiated satır örneği

İlişkide halihazırda "Omer" birincil anahtarlı bir satır varolduğu halde, bu satırın anahtarı düzey High' da sınıflandırılmıştır ve bundan dolayı kayıt eklemek

isteyen düşük kullanıcıya görünür değildir. Bundan dolayı, varolan birini değiştirmeksizin yeni satır tanıtılır. Sonuçta şekil 2.11' de oluşan tablo, "Omer" birincil anahtar ile satırlar polyinstantiated tır.

Şekil 2.10' daki PERSONEL tablosunu tekrar dikkate alalım ve düşük kullanıcının şimdi de aşağıdaki işlemi yaptığını varsayalım :

UPDATE Personel

SET Maas = '600M'

WHERE Ad = 'Aysenur'

Ad	C _{Ad}	Departman	C _{Departman}	Maaş	C _{Maaş}	TC
Merve	Low	Dep1	Low	600M	Low	Low
Omer	High	Dep2	High	750M	High	High
Aysenur	Low	Dep1	Low	700M	High	High
Aysenur	Low	Dep1	Low	600M	Low	Low

Şekil 2.12 Polyinstantiated öge örneği

İlişkide "Aysenur" için halihazırda "Maas" alanı için bir değer varolduğu halde, bu değer High sınıflandırılmıştır ve bundan dolayı ekleme yapmak isteyen düşük kullanıcıya görünür değildir. Bundan dolayı, varolan birini değiştirmeksizin yeni değer ilave edilir. Sonuçta şekil 2.12' de oluşan tablo, "Maas" niteliği "Aysenur" için polyinstantiated dır.

2.6.2.2 Görünür polyinstantiation

Halihazırda düşük erişim sınıfında varolan bir satır ile aynı birincil anahtara sahip satır eklemek için yüksek düzeyli kullanıcı ihtiyacı olduğu varsayılır. Burada da olası 3 seçenek vardır :

1. Halihazırda birincil anahtar ile satır var olduğunu ve eklemeyi geri çevirdiğini kullanıcıya bildir.
2. Yüksek düzeyde eklenilen yeni satır ile daha düşük düzeydeki var olan satırı değiştir.



3. Daha düşük düzeyde var olan satırı değiştirmeksizin yüksek düzeydeki satırı ekle.

Önceki bölümün aksine, seçim 1 herhangi bir kanal ortaya çıkarmaz. Düşük düzeyde varolan bir satırın yüksek kullanıcıya görünebilmesi için kullanıcıya çakışmanın olduğu bildirilir. Bununla beraber, yüksek kullanıcının kayıt ekleme işlemini geri çevirmek denial-of-service problemini belirtebilir. Bundan dolayı bu seçim tatmin edici değildir.

Seçenek 2 uygulanabilir değildir çünkü düşük düzeydeki bir satırı silmeyi belirtir. Bu satır düşük düzey kullanıcılar için görünmez olur, böylece sinyal verme kanalları ortaya çıkar.

Burada da sadece seçenek 3 kullanılabilir durumdadır.

Hali hazırda, yüksek kullanıcı düşük düzeydeki veriyi içeren alanı güncelleyeceği zaman da aynı tartışma uygulanır.

Ad	C _{Ad}	Departman	C _{Departman}	Maaş	C _{Maaş}	TC
Merve	Low	Dep1	Low	600M	Low	Low
Omer	Low	Dep2	Low	600M	Low	Low
Aysenur	Low	Dep1	Low	600M	Low	Low

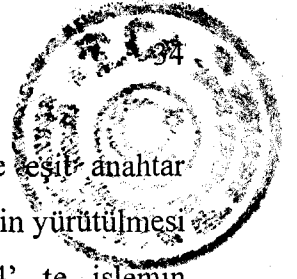
Şekil 2.13 Çok düzeyli ilişki örneği

Örnek 2 Şekil 2.13' te gösterilen çok düzeyli PERSONEL tablosunu dikkate alalım ve yüksek kullanıcının aşağıdaki işlemi gerçekleştirdiğini varsayalım :

```
INSERT INTO Personel
VALUES (Omer, Dep2, 900M)
```

Ad	C _{Ad}	Departman	C _{Departman}	Maaş	C _{Maaş}	TC
Merve	Low	Dep1	Low	600M	Low	Low
Omer	Low	Dep1	Low	600M	Low	Low
Aysenur	Low	Dep1	Low	600M	Low	Low
Omer	High	Dep2	High	900M	High	High

Şekil 2.14 Polyinstantiated satır örneği



Tabloda halihazırda düşük düzeyde sınıflandırılan “Omer” e eşit anahtar değerli bir satır bulunmaktadır. Böylece, fazladan kayıt ekleme işleminin yürütülmesi “Omer” için polyinstantiated satırlar ortaya çıkartır. Şekil 2.14’ te işlemin yürütülmesinden sonra ortaya çıkan ilişki gösterilmiştir.

2.6.2.3 Polyinstantiation Problemi İçin Çözümler

Polyinstantiation problemini adreslemek için çeşitli yaklaşımlar önerilmiştir. Her bir yaklaşım polyinstantiation’ ın anlamı ve kullanımı hakkında çeşitli bakış açıları yansıtır ve sonuç olarak avantaj ve dezavantajlara sahiptir. Bazı yaklaşımlar polyinstantiation’a izin vermeyi hedeflerken diğerleri önlemeyi hedefler.

Polyinstantiation’a izin veren yaklaşımlar, polyinstantiation’un çok düzeyli örneğinin kaçınılmaz bir parçası olduğu varsayımına dayanırlar.

Farklı güvenlik düzeylerindeki kullanıcılar, aynı gerçek dünya satırı için farklı nitelik değerleri görebilir (örneğin; aynı personel için farklı maaş değerleri) ve bu yüzden kullanıcılara bu değerleri başka şekilde güncellemek için izin verilmelidir. Bundan başka, polyinstantiation gizli hikayeler sağlamak için bazı durumlarda istenebilir.

Gerçek dünya varlıkları düşük düzey kullanıcılara görünür olduğu zaman gizli hikayeye ihtiyaç duyulur fakat o varlığın bazı nitelikleri varlığın kendisinden daha yüksek düzeyde sınıflandırılır. Gizli hikaye düşük düzeyli kullanıcıların yüksek düzeydeki değer hakkında bilgi edinmesini önlemek için düşük düzeydeki nitelik için makul bir değer sağlar. Örneğin; şekil 2.12’ deki çok düzeyli ilişkisine istinaden, “Aysenur” için düşük düzeyli maaş, düzey High’ da sınıflandırılan “Aysenur” in gerçek maaşını, düşük düzeyli kullanıcılardan gizlemek için gizli hikayeyi temsil edebilir.

Polyinstantiation’ u destekleyen modeller, polyinstantiated satırlar ve öğeler ile ilgili aşağıdaki anlamları dikkate alırlar :

- *Polyinstantiated satırlar gerçek dünya varlıklarına bakar.* Örneğin; şekil 2.14' teki çok düzeyli ilişkiye göre, adı "Omer" olan iki farklı personeli ifade eden iki satır vardır. Bu personellerden birisi "Dep1" departmanında çalışmakta ve "600M" maaş almakta ve diğeri "Dep2" departmanında çalışmakta ve "900M" maaş almaktadır.
- *Polyinstantiated öğeler aynı gerçek dünya varlıklarına bakar.* Örneğin; şekil 2.12' deki çok düzeyli ilişkiye göre, adı "Aysenur" olan iki satır "Dep1" departmanında çalışan aynı personeli ifade eder. Ancak, "Aysenur" un maaşı için iki farklı değer verilmiştir, bunlardan biri düşük düzeyde sınıflandırılan "600M" ve diğeri yüksek düzeyde sınıflandırılan "700M" dur.

Polyinstantiation' a izin veren modeller, Denning tarafından geliştirilen SeaView modelini ve Smith ve Winslett tarafından geliştirilen Belief-Based modelini içerir. SeaView modeli bölüm 2.4.2.2' de de belirtildiği gibi çok düzeyli ilişkilere dayanır ve satır ve nitelik polyinstantiation' unun her ikisini de destekler.

Belief-based modelinde, Smith ve Winslett kullanıcının gördüğünün ve inandığının arasındaki farkı ayırırlar. Özellikle, bir kullanıcı, kullanıcının erişim sınıfı tarafından kapsanan erişim sınıflarındaki tüm veriyi görebilir. Ancak, kullanıcı sadece kullanıcının erişim sınıfına eşit erişim sınıfına sahip veriye inanır.

Blief-based yaklaşımı aşağıdaki prensiplere dayanır :

- Anahtar nitelikler daima aynı tarzda sınıflandırılmalıdır.
- Anahtar olmayan öznitelikler daima aynı tarza sınıflandırılmalıdır, fakat muhtemelen anahtar nitelikler gibi aynı sınıflandırma düzeyinde olmazlar.
- Satırın anahtar olmayan nitelikler için tahsis ettiği sınıflandırma, satırın anahtarlı öznitelikleri için tahsis edilen sınıflandırmaya hakim olmalıdır.

Belief-based yaklaşımında, anahtar nitelikleri için aynı değerde ile fakat farklı sınıflandırma düzeylerindeki ile satırlar farklı gerçek dünya varlıklarını temsil eder. Anahtar nitelikler için, aynı değer ve erişim sınıfında fakat anahtar olmayan nitelikler için farklı erişim sınıfları ile satırlar aynı gerçek dünya varlığı hakkında farklı inançları (belief) temsil eder.

Ad	C _{Ad}	Departman	Maaş	TC
Berke	Low	Dep1	600M	Low
Berke	High	Dep2	800M	High
Ahmet	Low	Dep1	600M	Low
Ahmet	High	Dep2	800M	High

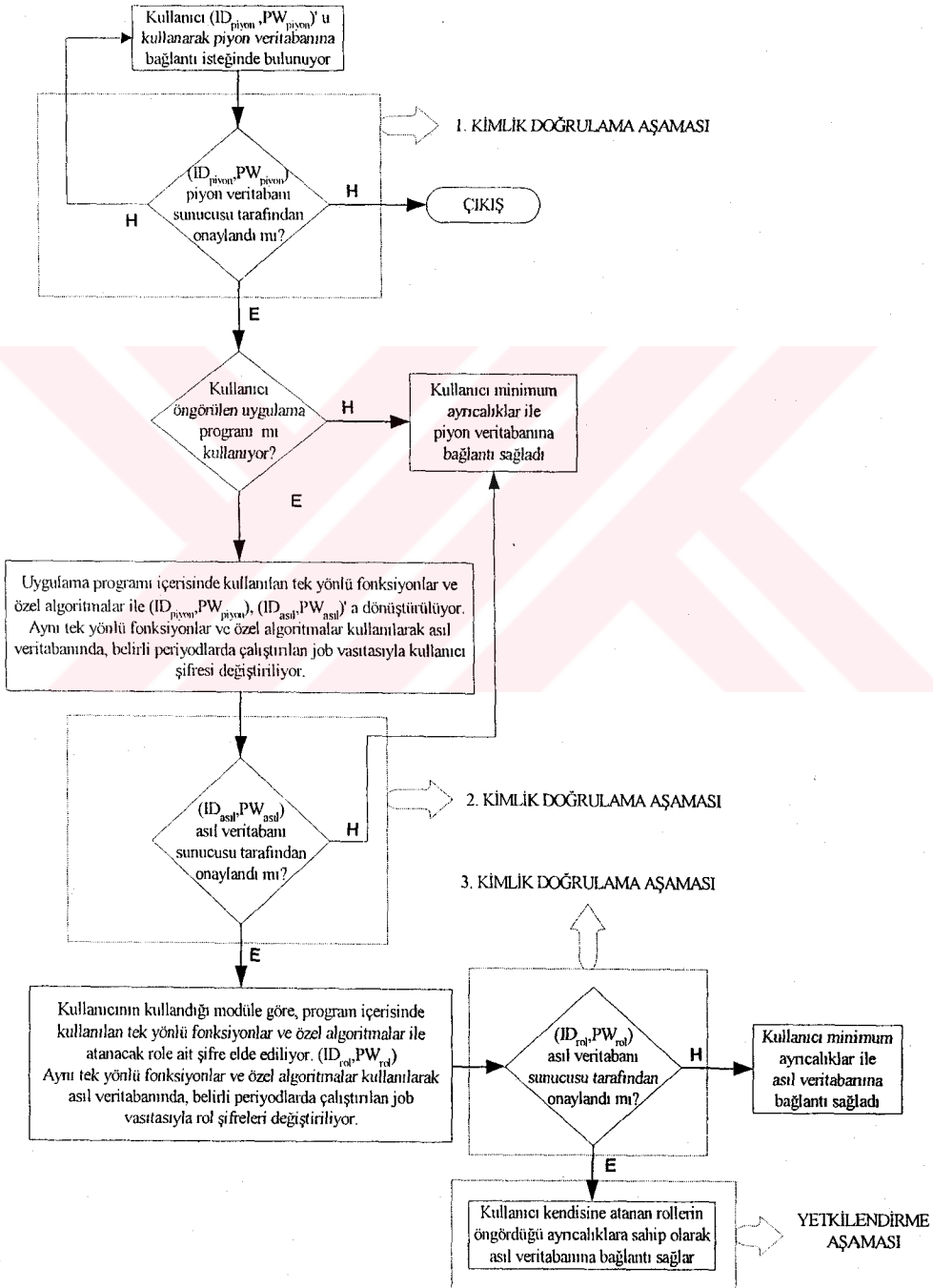
Şekil 2.15 Belief modelinde çok düzeyli ilişki örneği

Şekil 2.15’ te çok düzeyli ilişkide belief-based model örneği verilmektedir. “Ad” niteliğinde “Berke” olan iki satır, biri “Dep1” departmanında çalışan ve “600M” kazanan ve diğeri “Dep2” departmanında çalışan “800M” kazanan iki farklı personeli ima eder. “Ad” özniteliğinde “Ahmet” olan diğer iki satır da aynı personeli ima eder. Ancak, düşük kullanıcılar “Ahmet” in “Dep2” departmanında çalıştığını ve “600M” kazandığına inanırken yüksek kullanıcılar “Ahmet” in “Dep2” departmanında çalıştığını ve “800M” kazandığına inanacaktır.

Bazı yaklaşımlar polyinstantiationu önleme temelinde önerilmişlerdir. Özellikle varlık polyinstantiationu önlemek için yaklaşım, her kullanıcıya mümkün olan en düşük düzeyde sınıflandırarak görünür olması için bütün anahtar değerlerini ister. Varlık polyinstantiationu’ u önlemek için diğer olası yaklaşım, her bir birincil anahtar değerinin eşsiz sınıflandırmaya sahip olması için çeşitli düzeyler arasında birincil anahtarın etki alanını (domain) bölümler. Nitelik polyinstantiation’u özel “kısıtlanmış” değerler ortaya koyarak önlenebilir. Öge için değer yüksek düzeyde varolduğu zaman, “restricted (kısıtlanmış)” değer verilen düzeydeki eleman için kullanılır. Ancak bütün bu yaklaşımlar sistemin esnekliğini sınırlar.

3 TADA SİSTEMİ

TADA (Triple Authentication and Dynamic Authorization) sistemi şekil 3.1’de verilen iş akış diyagramında da görüldüğü üzere, güvenliği üç aşamalı kimlik doğrulama (authentication) ve tek aşamalı dinamik yetkilendirme (dynamic authorization) olmak üzere dört aşamada sağlamaktadır.



Şekil 3.1 TADA sistemi iş akış diyagramı

TADA sisteminde, veri güvenliğini sağlamak amacıyla iki ayrı veritabanı kullanılmaktadır. Veritabanlarından bir tanesi, kullanıcıların sadece ilk kimlik doğrulamayı sağlamak amacıyla herhangi bir veri içermeyecek piyon veritabanı (VT_{piyon}) olarak kullanılacaktır. Diğeri ise güvenliği sağlanacak olan verileri barındıran ve üzerindeki veriler üzerinde işlemler yapılacak olan asıl veritabanı ($VT_{asıl}$) olarak kullanılacaktır.

Her kullanıcıya iki tane kullanıcı adı tahsis edilmektedir. Bunlardan bir tanesi piyon kullanıcı (ID_{piyon}) olarak vazife yapacak, diğeri ise veritabanındaki veriler üzerinde işlemleri yapacak olan asıl kullanıcı ($ID_{asıl}$) olacaktır. Son kullanıcılar sadece piyon kullanıcı adı olarak nitelendirdiğimiz kullanıcı adı ve şifre (PW_{piyon}) ile muhatap olacaklar ve (ID_{piyon}, PW_{piyon}) ile VT_{piyon} veritabanına bağlanmaya çalışacaklardır.

1. *kimlik doğrulama aşamasında*, (ID_{piyon}, PW_{piyon}) için VT_{piyon} veritabanında kimlik doğrulama işlemi yapılmaktadır. Veritabanının kimlik doğrulama işlemi gerçekleştirmesi sonucunda ID_{piyon} kullanıcısı kendisine verilen minimum ayrıcalıklar ile VT_{piyon} veritabanına bağlantı sağlamaktadır.

Eğer ID_{piyon} kullanıcısı veritabanı yöneticisi tarafından öngörülen uygulama programını kullanıyor ise, uygulama programı içerisinde tek yönlü fonksiyonlar ve özel algoritmalar kullanılarak ($ID_{asıl}, PW_{asıl}$) elde edilmektedir.

$$ID_{asıl} = f(ID_{piyon} \oplus_1 f(ID_{piyon})) \quad (3.1)$$

Burada ID_{piyon} değerine tek yönlü f fonksiyonu uygulanmaktadır. Elde edilen değer ile ID_{piyon} değeri $\oplus_1$ özel işlemine tabi tutularak tek yönlü f fonksiyonu uygulanması sonucunda asıl kullanıcı adı ($ID_{asıl}$) elde edilir.

$$x_{pw} = f(T \oplus_2 k) \quad (3.2)$$

$$PW_{asıl} = f(ID_{asıl} \oplus_3 x_{pw}) \quad (3.3)$$

x_{pw} gizli anahtarı, güncel tarih ve saat değerini alan T timestamp ile kullanıcıya özel üretilen k anahtar değerinin $\oplus_2$ özel işlemine tabi tutulup tek yönlü f fonksiyonu uygulanması sonucunda elde edilen değeri göstermektedir.

PW_{asil} , ID_{asil} ve anahtar değeri (x_{pw}) arasında uygulanan özel $\oplus_3$ işleminin sonucunda uygulanan tek yönlü f fonksiyonu ile elde edilen değerdir.

Uygulama programından bağımsız olarak, VT_{asil} tarafında uygulama programı içerisinde kullanıcı şifresi tespiti için kullanılan tek yönlü fonksiyonlar ve özel algoritmaların aynıları kullanılarak belirli zaman periyodunda ID_{asil} ' a ait şifre otomatik olarak değiştirilmektedir. Böylece kimlik doğrulama aşamasının başarıyla geçilmesi için uygulama programı tarafından belirlenen kullanıcı şifresi ile VT_{asil} tarafında değiştirilen şifrenin aynı olması sağlanmaktadır.

VT_{asil} veritabanında 1' er dakikalık arayla çalışan algoritma sonucu değiştirilen ID_{asil} kullanıcı şifreleri

Çalışma Zamanı	Kullanıcı Şifresi
13:00:00	X
13:01:00	Y
13:02:00	Z
13:03:00	E
.	.
.	.

Uygulama programı kullanılarak VT_{asil} ' a bağlantı yapılacağı zaman, program içerisinde çalışan algoritma sonucunda ID_{asil} kullanıcısı için tespit edilen şifre

Bağlantı Zamanı	Tespit Edilen Şifre
13:00:30	X
13:01:00	Y
13:02:05	Z
13:02:50	Z
.	.
.	.

Şekil 3.2 Veritabanı ile uygulama programı arasındaki şifre eşleşmesi örneği

2. kimlik doğrulama aşamasında, uygulama programı içerisinde elde edilen (ID_{asil}, PW_{asil}) ile VT_{asil} veritabanına bağlantı sağlanması amaçlanmaktadır.

2. kimlik doğrulama aşaması da geçilip VT_{asil} veritabanıyla başarılı bir şekilde bağlantı kurulduktan sonra, kullanıcılara kullandıkları program içi modüle göre RBAC yardımıyla dinamik yetkilendirme yapılmaktadır. Bu işlem kullanıcılar ilk yaratılırken kendilerine pasif olarak atanan rollerden ilgili modülde kullanılacak olan rollerin aktif hale getirilmesi sonucunda gerçekleşmektedir. Rol atamalarının güvenli bir şekilde yapılmasını sağlamak için, rollerin şifre bağımlı olarak oluşturulması

gerekir. Yani herhangi bir kullanıcının bir rolü aktif hale getirebilmesi için rolün şifresine sahip olması gerekir. Ancak bu şifreyi girerek rolü üzerine alabilir ve gerekli erişim yetkilerine sahip hale gelir.

Burada dikkat edilmesi gereken konu rol şifresinin uygulama programı içerisinde sabit olarak tutulmaması gerektiğidir. Yani role ait şifre yardımıyla uygulama programı içerisinden rol ataması yapılabilir. Fakat role ait şifrenin değiştirilmesi durumunda, şifrenin uygulama programı içerisinden de değiştirilmesi gerekir ve bu da teknik açıdan kolay olmayabilir. Bunun için rol şifresinin de kullanıcı şifresi gibi dinamik olarak atanması gerekir. Bunun için yine uygulama programı içerisinde ve veritabanı tarafında tek yönlü fonksiyonlar ve özel işlemler kullanılmaktadır.

$$x_{rol} = f(T \oplus_4 k) \quad (3.4)$$

$$PW_{rol} = f_i(f(ID_{rol} \oplus_5 x_{rol})) \quad (3.5)$$

X_{rol} gizli anahtarı, güncel tarih ve saat değerini alan T timestamp ile role özel üretilen k anahtar değerinin $\oplus_4$ özel işlemine tabi tutulup tek yönlü f fonksiyonu uygulanması sonucunda elde edilen değeri göstermektedir.

PW_{rol} , ID_{rol} ve anahtar değeri (x_{rol}) arasında uygulanan özel $\oplus_5$ işleminin sonucunda uygulanan tek yönlü f fonksiyonu ile elde edilen değerdir.

3. *kimlik doğrulama aşamasında*, uygulama programı tarafından tespit edilen rol şifresi ile VT_{asil} da üretilen ilgili role ait şifrenin karşılaştırılması yapılmaktadır. Bu şifrelerin aynı olması program tarafından garanti edilmelidir. Aksi takdirde kullanıcıya gerekli erişim yetkileri verilemeyeceği için programın ilerleyişi de sona erecektir.

Dinamik yetkilendirme aşamasında ise, uygulama programı içerisinden atanacak rol (ID_{rol}) ve bu rol için elde edilen şifre (PW_{rol}) ile uygulama programı içerisinde kullanılan modüle göre kullanıcıya gerekli erişim yetkilerinin verilmesi

amaçlanmaktadır. Bu da uygulama programının istenilen bölümünde gerekli rollerin aktif hale dönüştürülmesiyle mümkündür. Bu aşama programın ilk çalıştırılması durumunda çalışan bir aşama değildir. Uygulama programı içerisinde çalıştırılan birbirinden farklı her modülde bu aşama ve 3. kimlik doğrulama aşaması uygun bir şekilde çalıştırılmaktadır.

Şekil 3.3' de verilen örneğe göre $ID_{asıl}$ kullanıcısı Modül₃' ü kullanmak istediği zaman ROL_2 ve ROL_3 rolleri şifre yardımıyla aktif hale getirilecektir. Modül₃' den çıkıp bu kez modül₁' e geçtiği zaman ise ROL_2 ve ROL_3 rolleri pasif duruma çevrilip ROL_1 aktif hale dönüştürülecektir.

$ID_{asıl}$ kullanıcısına aktarılan roller

Rol Adı	Durum
ROL_{min}	Aktif
ROL_1	Pasif
ROL_2	Pasif
ROL_3	Pasif

Uygulama programı içindeki modüller ve kullanıcıda bulunması gereken roller

Modül Adı	Gerekli Roller
Modül ₁	ROL_1
Modül ₂	ROL_2
Modül ₃	ROL_2, ROL_3
Modül ₄	ROL_1, ROL_3

Şekil 3.3 Dinamik Yetkilendirme

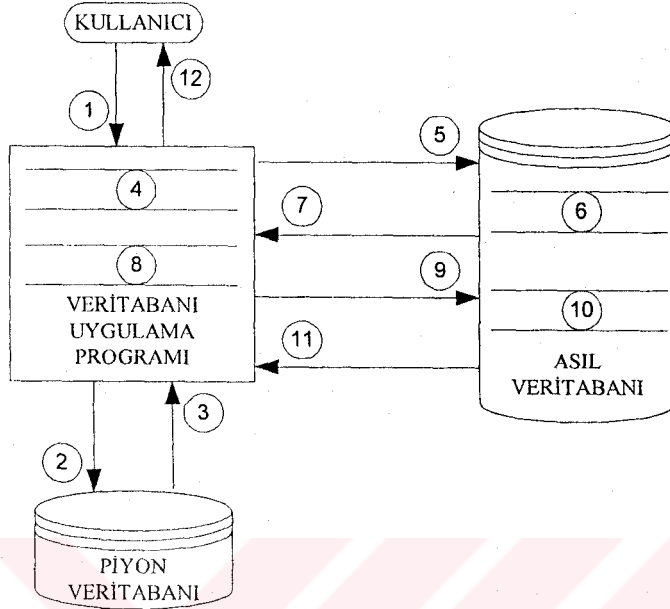
Yukarıda ifade edilen ve uygulanan tüm işlemlerin amacı son kullanıcıların veritabanına güvenilir uygulama programları aracılığıyla bağlanmalarını sağlamaktır. Eğer ki kullanıcı bu uygulama programını kullanmadığı takdirde VT_{piyon} veritabanına minimum yetkilerle bağlantı kurabilecek ve de bundan öteye gitmesi oldukça güçleşecektir.

3.1 TADA Sisteminin İşleyişi

Öncelikle veritabanı yöneticisi tarafından VT_{piyon} veritabanında ID_{piyon} adında kullanıcı hesabı açılır. (3.1) denklemi kullanılarak $VT_{asıl}$ veritabanında $ID_{asıl}$ kullanıcı hesabı açılır. Daha sonra (3.2) ve (3.3) denklemleri kullanılarak belirli periyotlarda çalışacak bir algoritma ile $ID_{asıl}$ kullanıcısının şifresinin değiştirilmesi sağlanır.

Veri erişimini kontrol etmek için gerekli yetkilendirmeleri içeren roller oluşturulur ve bu rollerden ilgili olanları $ID_{asıl}$ kullanıcısına pasif olarak atanır.

Yetkilendirme güvenliğini sağlamak amacıyla roller şifre kontrollü hale getirilir. Daha sonra (3.4) ve (3.5) denklemleri vasıtasıyla periyodik olarak çalışacak algoritma ile rollere ait şifrelerin değiştirilmesi sağlanır.



Şekil 3.4 TADA sisteminin uygulanışı

TADA sisteminin işleyişi ile ilgili şematik görünüm şekil 3.4' de verilmiştir. Aşağıda bu mekanizmanın adım adım işleyişi anlatılmaktadır.

1. Kullanıcı kendisine tahsis edilen kullanıcı adı ve şifresiyle (ID_{piyon}, PW_{piyon}), güvenli veritabanı uygulama programını kullanarak veritabanı bağlantısı kurmaya çalışıyor.
2. Kullanıcı kimlik doğrulama işlemini sağlamak amacıyla girilen (ID_{piyon}, PW_{piyon}), VT_{piyon} veritabanına gönderiliyor.
3. VT_{piyon} girilen kullanıcı adı ve şifrenin doğruluğunu onayladığını ya da onaylamadığını belirten mesajı uygulama programına geri döndürür. Eğer VT_{piyon} girilen (ID_{piyon}, PW_{piyon}) un yanlış olduğunu belirtirse, 12. adıma geçilerek kullanıcıya veritabanı bağlantısının yapılamadığını belirten mesaj gönderilir.

4. Eğer ki; VT_{piyon} (ID_{piyon}, PW_{piyon}) un doğruluğunu onaylaması durumunda, (3.1), (3.2) ve (3.3) denklemleri kullanılarak ($ID_{asıl}, PW_{asıl}$) elde ediliyor.
5. ($ID_{asıl}, PW_{asıl}$) kimlik doğrulama sağlamak amacıyla $VT_{asıl}$ veritabanına gönderiliyor.
6. $VT_{asıl}$ veritabanında periyodik olarak çalışan algoritma sonucu değiştirilen kullanıcı şifresi ile uygulama programının gönderdiği şifre ($PW_{asıl}$) karşılaştırılıyor.
7. $VT_{asıl}$ veritabanı, uygulama programı içerisinde elde edilen ($ID_{asıl}, PW_{asıl}$) in doğruluğunu onayladığını ya da onaylamadığını belirten mesajı uygulama programına geri döndürüyor. Burada ($ID_{asıl}, PW_{asıl}$) in $VT_{asıl}$ veritabanı tarafından onaylanmasının garanti edilmesi gerekir.

Bu adıma kadar kullanıcının veritabanları ile kimlik doğrulaması yapılmış fakat kullanıcı minimum yetkilerle veritabanına bağlanmıştır. Bundan sonraki adımlarda kullanıcının dinamik olarak yetkilendirilmesi sağlanmaktadır.

8. Uygulama programı içerisinde kullanılan modüle göre kullanıcıya aktarılacak olan rollere ait şifreler (3.4) ve (3.5) denklemlerine göre elde edilir.
9. Kullanıcıya atanacak olan rol adı (ID_{rol}) ve uygulama programı içerisinde elde edilen rol şifresi (PW_{rol}) doğruluğunu kontrol etmek amacıyla asıl veritabanına gönderiliyor.
10. $VT_{asıl}$ veritabanında periyodik olarak çalışan algoritma sonucu değiştirilen rol şifresi ile uygulama programının gönderdiği şifre (PW_{rol}) karşılaştırılıyor.
11. $VT_{asıl}$ veritabanı, aktif hale getirilecek rol için uygulama programı içerisinde elde edilen (ID_{rol}, PW_{rol}) ün doğruluğunu onayladığını ya da onaylamadığını belirten mesajı uygulama programına geri döndürür.

Burada (ID_{rol} , PW_{rol}) ün asıl veritabanı tarafından onaylanmasının garanti edilmesi gerekir.



12. Veritabanlarıyla kimlik doğrulama işlemleri başarılı bir şekilde gerçekleşirse, kullanıcıya gerekli yetkilere sahip olarak veritabanına bağlanıldığı mesajı iletilir.

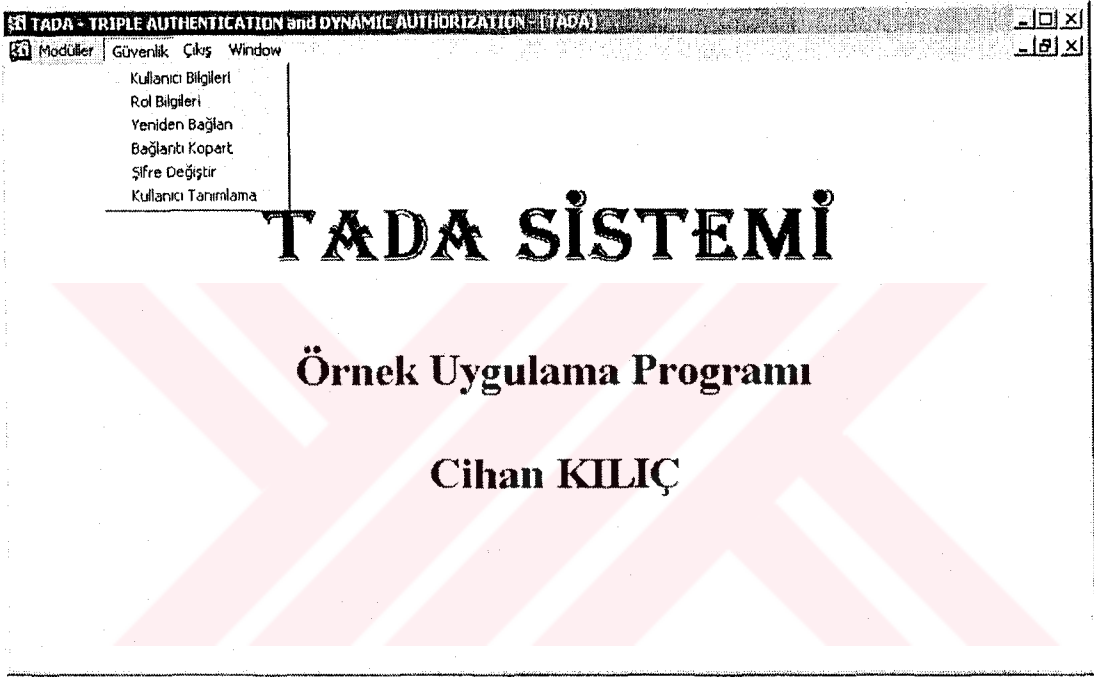
Böylece kullanıcı, uygulama programının izin verdiği ölçüde veritabanındaki veriler üzerinde işlem yapma hakkına sahip olur. Burada unutulmaması gereken en önemli husus, uygulama programının veritabanı güvenliğini tehlikeye atmayacak şekilde dizayn edilmesi gerektiği ve kullanıcılar için gereken tüm verilere güvenli bir şekilde erişimin sağlanması gerektiğidir.



4 GERÇEKLENEN UYGULAMA VE SONUÇLAR

Bu bölümde TADA sistemi kullanılarak yapılan örnek uygulama ekran görüntüleri ve kaynak kodları verilmiştir. DBMS olarak Oracle 9i ve uygulama yazılımı programlama dili olarak da Oracle Developer 6i kullanılmıştır.

4.1 TADA Sistemi Arayüzleri



Şekil 4.1 TADA Sistemi Giriş Arayüzü

TADA sistemi kullanılarak hazırlanan örnek uygulama programının giriş arayüzü şekil 4.1’ de verilmiştir. Veritabanı kullanıcısının programa giriş yapması durumunda, piyon kullanıcı adı asıl kullanıcı adına çevrilmektedir. Asıl kullanıcıya ait şifre tespiti yapıldıktan sonra kullanıcının asıl veritabanına bağlantısı sağlanır. Eğer asıl kullanıcı adının asıl veritabanında bulunmaması ya da asıl kullanıcı şifresinin herhangi bir sebepten dolayı yanlış tespit edilmesi durumunda kullanıcı piyon veritabanına bağlı durumda kalacak ve programın diğer aşamalarına geçişi mümkün olmayacaktır.

TADA - [Şifre Değiştir]

Window

Kullanıcı Adınız USER

Şimdiki Şifreniz *****

Yeni Şifreniz *****

Yeni Şifreniz (Tekrar) *****

ŞİFREYİ DEĞİŞTİR ÇIKIŞ

Record: 1/1 <DSC> <DBG>

Şekil 4.2 Şifre Değiştirme Arayüzü

Şekil 4.2’ de verilen şifre değiştirme arayüzünde, kullanıcılar kendilerine tahsis edilen şifreleri değiştirebilme hakkına sahip olmaktadır. Kullanıcı şifre değişikliği yapmak istediği zaman öncelikle eski şifresini ve sonra yeni şifresini girecek ve “Şifreyi Değiştir” düğmesine basması durumunda piyon veritabanında o kullanıcıya ait şifre değiştirilecektir.

Logon

Username: User

Password: ****

Database:

Connect Cancel

Şekil 4.3 Logon Ekranı

Kullanıcı programdan çıkmadan yeniden bağlantı kurmak istediği zaman karşısına şekil 4.3’ deki gibi logon ekranı gelecektir. Piyon kullanıcı adı ve şifresi girildikten sonra kullanıcı bilgileri piyon veritabanından kontrol edilecek, eğer

bilgiler doğruysa program içerisinde asıl kullanıcı adı ve şifresi tespit edilerek kullanıcının asıl veritabanına bağlantı kurması sağlanacaktır.

The screenshot shows a window titled "TADA - [Yeni Kullanıcı Tanımlama]". It has a menu bar with "İşlem", "Blok", "Düzenle", "Alan", "Kayıt", "Sorgu", "Yardım", "Program Hakkında", and "Window". Below the menu is a toolbar with various icons. The main area is divided into two sections: "Piyon Veritabanı Kullanıcı Bilgileri;" and "Asıl Veritabanı Kullanıcı Bilgileri;".

Piyon Veritabanı Kullanıcı Bilgileri;

Kullanıcı Adı	USER
Şifre	USER

Asıl Veritabanı Kullanıcı Bilgileri;

Kullanıcı Adı	88F866EE376BFA8120F802A94F893F
Şifre	66B17A8AC6B98E56494DD8C939CFE8

Below these sections are two boxes: "Aktif Roller" and "Pasif Roller".

Aktif Roller

ROL_MIN

Pasif Roller

ROL_1
ROL_2
ROL_3

Şekil 4.4 Kullanıcı Bilgileri Arayüzü

Şekil 4.4' de görüldüğü üzere kullanıcı bilgileri arayüzünde, kullanıcının piyon veritabanındaki kullanıcı adı ve şifresi ile asıl veritabanındaki kullanıcı adı ve şifresi gösterilmektedir. Bu arayüzde ayrıca kullanıcının asıl veritabanı üzerindeki sahip olduğu aktif ve pasif roller de görüntülenmektedir. Bu arayüz örnek uygulamaya yönelik sistemin çalışma prensibi konusunda bilgi vermek amacıyla hazırlanmıştır. Bu tür bir arayüzün gerçek bir uygulamada kullanılması çok sakıncalı bir durum ortaya koyacaktır.

Kullanıcı program içerisindeki bir modüle giriş yaptığı zaman, kullanıcı ilgili modülü kullanmaya yetkili ise o modül için gerekli roller set edilir. Kullanıcı ilgili modüle giriş yapmadan önce gerekli haklara sahip durumda değildir. Modülü çalıştırdığı zaman pasif olarak atanan rollerden gerekli olanları, program içerisinde ilgili rol için tespit edilen şifre yardımıyla kendisine atamaktadır.

YEREL İÇİŞİLER BAKANLIĞI
ULUSLARARASI İLİŞİLER GENEL MÜDÜRLÜĞÜ
T.C. İÇİŞİLER BAKANLIĞI

TADA - [MODUL 1]

İşlem Blok Düzenle Alan Kayıt Sorgu Yardım Program Hakkında Window

Set edilen roller

ROL 1
ROL 2

SELECT izni olan tablolar

ZPF_IS_ONEM_DERECE
ZPF_PLANLANAN_ISLER
ZPF_PLAN_DISI_ISLER
ZPF_SAPMA_NEDEN

INSERT izni olan tablolar

ACCESS\$
ZPF_PLANLANAN_ISLER
ZPF_PLAN_DISI_ISLER
ZPF_SAPMA_NEDEN
ZPF_IS_ONEM_DERECE

DELETE izni olan tablolar

ZPF_PLANLANAN_ISLER
ZPF_IS_ONEM_DERECE

UPDATE izni olan tablolar

ZPF_PLANLANAN_ISLER
ZPF_IS_ONEM_DERECE
ZPF_SAPMA_NEDEN

Şekil 4.5 Uygulama Programı Modülü

Kullanıcı uygulama modülüne giriş yaptığında set ettiği roller ve bu roller aracılığıyla yetki sahibi olduğu tablolar ve yetki tipleri şekil 4.5' de verilmiştir.

TADA - [Yeni Kullanıcı Tanımlama]

İşlem Blok Düzenle Alan Kayıt Sorgu Yardım Program Hakkında Window

Piyon Veritabanı Yöneticisi;

Kullanıcı Adı

Şifre

Asıl Veritabanı Yöneticisi;

Kullanıcı Adı

Şifre

Yeni Yaratılacak Kullanıcı Bilgileri;

Kullanıcı Adı

Şifre

Asıl Veritabanında Oluşan Kullanıcı Bilgileri;

Kullanıcı Adı

Şifre

KULLANICI YARAT

Şekil 4.6 Yeni Kullanıcı Yaratma Arayüzü

Şekil 4.6' da verilen kullanıcı yaratma arayüzünde görüldüğü gibi, her iki veritabanında da kullanıcı yaratılacağı için piyon ve asıl veritabanlarındaki DBA' in şifreleri girilmelidir. Kullanıcıya tahsis edilecek ve piyon veritabanında yaratılacak olan kullanıcı adı ve şifresi girildikten sonra, asıl veritabanında yaratılacak kullanıcı adı ve şifresi otomatik olarak program tarafından tespit edilmekte ve kullanıcı yaratılmaktadır. Kullanıcı yaratma arayüzü veritabanı yöneticileri tarafından kullanılabilir.

4.2 Geliştirilen Foksiyonlar

4.2.1 Veri Şifreleme Fonksiyonu

Veritabanı üzerinde DBA tarafından çalıştırılacak olan aşağıdaki fonksiyon yardımıyla veriler MD5 tek yönlü şifreleme algoritması kullanılarak şifreli hale dönüştürülür. Veri şifreleme işlemi oracle veritabanında kullanılan bir fonksiyon olan DBMS_OBFUSCATION_TOOLKIT.MD5 fonksiyonu kullanılarak yapılmaktadır.

Asıl veritabanında ve uygulama yazılımı içerisinde yapılacak olan şifreleme işlemlerinde de bu fonksiyon kullanılmaktadır.

```
CREATE OR REPLACE FUNCTION VERI_SIFRELE
(SIFRELENECEK_DEGER VARCHAR2) RETURN VARCHAR2 IS
raw_deger varchar2(100);
hex_deger varchar2(100);
BEGIN
RAW_DEGER := DBMS_OBFUSCATION_TOOLKIT.MD5 (INPUT_STRING =>
sifrelenecek_deger);
SELECT RAWTOHEX(raw_deger) INTO hex_deger FROM dual
RETURN hex_deger;
END;
```

4.2.2 Piyon Kullanıcıdan Asıl Kullanıcı Adını Elde Eden Fonksiyon

(3.1) denklemi dikkate alınarak piyon kullanıcı adından asıl kullanıcı adının

elde edildiği bu fonksiyon uygulama yazılımı tarafında çalışmaktadır. Yapılan işlemler ve üretilen kodlar veritabanı yöneticisi veya programcıya göre farklı şekillerde yapılabilir.

FUNCTION ASIL_KULLANICI_ID (PIYON_ID VARCHAR2) RETURN VARCHAR2 IS

f_piyon_id varchar2(32); /* (3.1) denklemindeki $f(ID_{piyon})$ değeri */
 ozel_islem_1 varchar2(1000); /* $\oplus_1$ özel işlemi sonucu elde edilen değer */
 asil_id varchar2(32); /* (3.1) denkleminde elde edilen ID_{asil} değeri */
 i integer;

BEGIN

/* ID_{piyon} değeri MD5 hash fonksiyonuna tabi tutuluyor, elde edilen değer
 f_piyon_id değişkenine aktarılıyor. */

f_piyon_id := DBA.Verı_Sıfrele (upper(piyon_id));

/* f_piyon_id ile piyon_id değerleri özel işleme tabi tutuluyor ve sonuçta elde edilen
 değer ozel_islem_1 değişkenine aktarılıyor. Bu işlemler daha karmaşık
 biçimlerde yapılabilir. */

ozel_islem_1:=null;

i:=1;

LOOP

EXIT WHEN i=Length(piyon_id);

ozel_islem_1:=ozel_islem_1 || f_piyon_id || upper(piyon_id);

i:=i+1;

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, veri_sıfrele fonksiyonu
 kullanılarak şifreleniyor ve yeni değer asil_id değişkenine aktarılıyor. */

asil_id := DBA.Verı_Sıfrele (ozel_islem_1);

RETURN substr(asil_id,1,30);

END;

4.2.3 Asıl Kullanıcı Adına Ait Şifreyi Tespit Eden Fonksiyon

(3.2) ve (3.3) denklemleri kullanılarak asıl kullanıcıya ait şifrenin tespit edildiği bu fonksiyon uygulama yazılımı tarafında çalışmaktadır. Yapılan işlemler ve

üretile kodlar veritabanı yöneticisi veya programcıya göre farklı şekillerde yapılabilir.



FUNCTION ASIL_KULLANICI_SIFRE (ASIL_ID VARCHAR2) RETURN VARCHAR2 IS

```
x_pw varchar2(32);          /* (3.2) denklemindeki xpw gizli anahtar değeri. */
t varchar2(20);             /* Timestamp değeri. */
k varchar2(1000);           /* Kullanıcıya özel üretilen anahtar değeri. */
ozel_islem_2 varchar2(10000); /*  $\oplus_2$  özel işlemi sonucu elde edilen değeri */
ozel_islem_3 varchar2(10000); /*  $\oplus_3$  özel işlemi sonucu elde edilen değeri */
asil_sifre varchar2(32);     /* (3.3) denklemin sonucu elde edilen PWasil değeri. */
i integer;
```

BEGIN

```
/* Güncel zaman değeri gun/ay/yıl saat:dakika formatında t değişkenine aktarılıyor. */
t := to_char(sysdate,'dd.mm.yyyy HH24:MI');
```

```
/* Kullanıcıya özel olarak üretilen kod k değişkenine aktarılıyor. Bu örnekte kod çok basit bir şekilde üretilmiştir. Bu kodlama karmaşık işlemler sonucunda bulunabilir. */
k := asil_id || 'e özel bir anahtar';
```

```
/* t ve k değerleri özel işleme tabi tutuluyor ve sonuçta elde edilen değeri
   ozel_islem_2 değişkenine aktarılıyor. Bu işlemler daha karmaşık biçimlerde yapılabilir. */
```

```
ozel_islem_2 := null;
```

```
i:=0;
```

LOOP

```
EXIT WHEN i=to_number(to_char(sysdate,'HH24'))+1;
```

```
ozel_islem_2 := ozel_islem_2 || t || k;
```

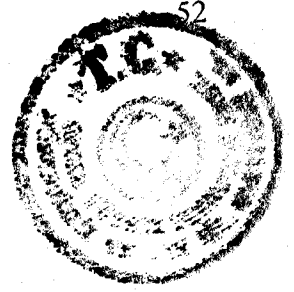
```
i:=i+1;
```

END LOOP;

```
/* Uygulanan özel işlem sonucu elde edilen değeri, veri_sifrele fonksiyonu kullanılarak şifreleniyor ve yeni değeri x_pw değişkenine aktarılıyor. */
```

```
x_pw := DBA.Veri_Sifrele (ozel_islem_2);
```

```
/* x_pw değeri ile asil_id değeri özel işleme tabi tutuluyor ve sonuçta elde edilen değeri
   ozel_islem_3 değişkenine aktarılıyor. Bu işlemler daha karmaşık biçimlerde yapılabilir. */
```



```
ozel_islem_3 := null;
i:=0;
LOOP
    EXIT WHEN i=to_number(to_char(sysdate,'MI'))+1;
    ozel_islem_3 := ozel_islem_3 || asil_id || x_pw;
    i:=i+1;
END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, veri_sifrele fonksiyonu
kullanılarak şifreleniyor ve yeni değer asil_sifre değişkenine aktarılıyor. */
asil_sifre := DBA.Veri_Sifrele (ozel_islem_3);
RETURN substr(asil_sifre,1,30);
END;
```

4.2.4 Şifre Korumalı Rol Oluşturma

Roller asıl veritabanı üzerinde oluşturulmalıdır. Şifre korumalı olarak rol oluşturmak için aşağıdaki SQL cümlecği kullanılır.

```
CREATE ROLE "ROL_1" IDENTIFIED BY "password"
```

4.2.5 Role Ait Şifreyi Tespit Eden Fonksiyon

Kullanıcıların rolleri aktif hale getirmeleri için rol şifrelerini bilmeleri gerekmektedir. (3.4) ve (3.5) denklemleri dikkate alınarak uygulama programı tarafında çalışan bu fonksiyon yardımıyla rollere ait şiflerin tespiti yapılmaktadır. Yapılan işlemler ve üretilen kodlar çok farklı şekillerde yapılabilir.

```
FUNCTION ROL_SIFRE(ROL_ID VARCHAR2) RETURN VARCHAR2 IS
```

```
x_rol varchar2(32);          /* (3.4) denklemindeki  $x_{rol}$  gizli anahtar değeri. */
t varchar2(20);              /* Timestamp değeri. */
k varchar2(1000);            /* Role özel üretilen anahtar değeri. */
ozel_islem_4 varchar2(10000); /*  $\oplus_4$  özel işlemi sonucu elde edilen değer */
ozel_islem_5 varchar2(10000); /*  $\oplus_5$  özel işlemi sonucu elde edilen değer */
rol_sifre varchar2(32);      /* (3.5) denklemin sonucu elde edilen  $PW_{rol}$  değeri. */
i integer;
```

BEGIN

/* Güncel zaman değeri gun/ay/yıl saat:dakika formatında t değişkenine aktarılıyor. */
t := to_char(sysdate,'dd.mm.yyyy HH24:MI');

/* Role özel olarak üretilen kod k değişkenine aktarılıyor. Bu örnekte kod çok basit
bir şekilde üretilmiştir. Bu kodlama karmaşık işlemler sonucunda bulunabilir. */
k := rol_id || 'e özel bir anahtar';

/* t ve k değerleri özel işleme tabi tutuluyor ve sonuçta elde edilen değer
ozel_islem_4 değişkenine aktarılıyor. Bu işlemler daha karmaşık biçimlerde
yapılabilir. */

ozel_islem_4 := null;

i:=0;

LOOP

exit when i=to_number(to_char(sysdate,'HH24'))+1;

ozel_islem_4 := ozel_islem_4 || t || k;

i:=i+1;

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, veri_sifrele fonksiyonu
kullanılarak şifreleniyor ve yeni değer x_rol değişkenine aktarılıyor. */

x_rol := DBA.Veri_Sifrele (ozel_islem_4);

/* x_rol değeri ile rol_id değeri özel işleme tabi tutuluyor ve sonuçta elde edilen
değer ozel_islem_5 değişkenine aktarılıyor. Bu işlemler daha karmaşık biçimlerde
yapılabilir. */

ozel_islem_5 := null;

i:=0;

LOOP

EXIT WHEN i=to_number(to_char(sysdate,'MI'))+1;

ozel_islem_5 := ozel_islem_5 || rol_id || x_rol;

i:=i+1;

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, veri_sifrele fonksiyonu
kullanılarak şifreleniyor ve yeni değer rol_sifre değişkenine aktarılıyor. */

rol_sifre := DBA.Veri_Sifrele (ozel_islem_5);

RETURN substr(rol_sifre,1,30);

END;

4.2.6 Kullanıcı Yaratma Prosedürü

Uygulama yazılımı tarafında DBA tarafından kullanılan bu prosedür iki aşamada çalışmaktadır. İlk olarak piyon veritabanında piyon kullanıcı yaratılmaktadır. İkinci olarak ise asıl veritabanında asıl kullanıcı adı ve şifresine göre kullanıcı yaratılmaktadır.

```

PROCEDURE KULLANICI_YARAT (PIYON_ID VARCHAR2, PIYON_SIFRE
    VARCHAR2) IS
    kayıt_sayisi Integer;

BEGIN
    /* Oluşturulmak istenen kullanıcı adının piyon veritabanında mevcut olup
        olunmadığının kontrolü yapılıyor. */
    SELECT count(*) INTO kayıt_sayisi FROM all_users WHERE upper(username) =
        Upper(piyon_id);
    IF kayıt_sayisi=0 THEN

        /* Eğer kullanıcı adı piyon veritabanında mevcut değilse kullanıcı yaratılıyor. */
        Forms_Ddl ('CREATE USER "'||upper(piyon_id)||'" IDENTIFIED BY
            "'||piyon_sifre||'" DEFAULT TABLESPACE "USERS"');

        /* Piyon kullanıcıya minimum ayrıcalık olarak CONNECT hakkı veriliyor. */
        Forms_Ddl ('GRANT "CONNECT" TO "' || upper(piyon_id)||'"');

        /* Piyon kullanıcının yaratılıp yaratılmadığının kontrolü yapılıyor. */
        SELECT count(*) INTO kayıt_sayisi FROM all_users WHERE
            upper(username) = upper(piyon_id);
        IF kayıt_sayisi=1 THEN
            Message('Kullanıcı yaratıldı...');

            /* Veritabanı yöneticisiyle asıl veritabanı bağlantısı kuruluyor. */
            Logout;
            Logon ('DBA','asdfqwe@asil_vt');

            /* Asıl kullanıcı adı ve şifresi kullanılarak kullanıcı yaratılıyor. */
            Forms_Ddl ('CREATE USER "'||asil_kullanici_id(piyon_id)||'" IDENTIFIED
                BY "'||asil_kullanici_sifre(asil_kullanici_id(piyon_id))||'"
                DEFAULT TABLESPACE "USERS"');
        
```

```

/* Kullanıcıya default roller aktif ve pasif şekilde aktarılıyor. */
Forms_Ddl ('GRANT "ROL_MIN" TO "'|| asil_kullanici_id(piyon_id) ||"'");
Forms_Ddl ('GRANT "ROL_1" TO "'|| asil_kullanici_id(piyon_id) ||"'");
Forms_Ddl ('GRANT "ROL_2" TO "'|| asil_kullanici_id(piyon_id) ||"'");
Forms_Ddl ('GRANT "ROL_3" TO "'|| asil_kullanici_id(piyon_id) ||"'");
Forms_Ddl ('ALTER USER "'|| asil_kullanici_id(piyon_id) ||"' DEFAULT
          ROLE "CONNECT");

ELSE
    Message('Kullanıcı yaratılamadı...!');
END IF;

ELSE
    Message('Böyle bir kullanıcı adı mevcut..!');
END IF;

END;

```

4.2.7 Kullanıcı Şifrelerini Değiştiren Veritabanı Job'u

Asıl kullanıcı için şifre tespitinde kullanılan fonksiyonun (ASIL_KULLANICI_SIFRE) aynıısı kullanılarak, kullanıcılara ait şifreler veritabanı tarafında 1 dakikalık periyotlar halinde ve dakika başlarında (örn. 10:01:00, 10:02:00 vb.) otomatik olarak değiştirilmektedir. Bu job DBA tarafından asıl veritabanı üzerinde çalıştırılmalıdır.

```
DECLARE jobno number;
```

```

BEGIN DBMS_JOB.SUBMIT(job => jobno,
    next_date => to_date('25.04.2003 00:00','dd.mm.yyyy HH:MI'),
    interval => '/*1:Mins*/ to_date(to_char(sysdate,"dd.mm.yyyy HH24:MI")||":00",
        "dd.mm.yyyy HH24:MI:SS")+1/(60*24)', what =>

```

```
DECLARE
```

```

    asil_id varchar2(30);          /* Asıl kullanıcı adı */
    x_pw varchar2(32);            /* (3.2) denklemindeki  $x_{pw}$  gizli anahtar değeri. */
    t varchar2(20);              /* Timestamp değeri. */
    k varchar2(1000);            /* Kullanıcıya özel üretilen anahtar değeri. */
    ozel_islem_2 varchar2(10000); /*  $\oplus_2$  özel işlemi sonucu elde edilen değer */
    ozel_islem_3 varchar2(10000); /*  $\oplus_3$  özel işlemi sonucu elde edilen değer */
    asil_sifre varchar2(32);      /* (3.3) denklemi sonucu elde edilen  $PW_{asil}$  değeri. */

```

i integer;

cursor cur1 is SELECT username FROM all_users WHERE length(username)=30;

BEGIN

/* Asıl veritabanı üzerindeki kullanıcı isimleri teker teker alınıyor. */

Open cur1;

LOOP

FETCH cur1 INTO asil_id;

EXIT WHEN cur1% notfound;

/* Güncel zaman değeri ASIL_KULLANICI_SIFRE fonksiyonundaki gibi
gun/ay/yıl saat:dakika formatında t değişkenine aktarılıyor. */

t := to_char(sysdate,'dd.mm.yyyy HH24:MI');

/* ASIL_KULLANICI_SIFRE fonksiyonundaki gibi kullanıcıya özel olarak üretilen
kod k değişkenine aktarılıyor. */

k := asil_id || "e özel bir anahtar";

/* t ve k değerleri ASIL_KULLANICI_SIFRE fonksiyonundaki gibi özel işleme tabi
tutuluyor ve sonuçta elde edilen değer özel_islem_2 değişkenine aktarılıyor. */

ozel_islem_2 := null;

i:=0;

LOOP

EXIT WHEN i=to_number(to_char(sysdate,'HH24'))+1;

ozel_islem_2 := ozel_islem_2 || t || k;

i:=i+1;

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, ASIL_KULLANICI_SIFRE
fonksiyonundaki gibi veri_sifreleme fonksiyonu kullanılarak şifreleniyor ve yeni
değer x_pw değişkenine aktarılıyor. */

x_pw := DBA.Veri_Sifrele (ozel_islem_2);

/* x_pw değeri ile asil_id değeri ASIL_KULLANICI_SIFRE fonksiyonundaki gibi
özel işleme tabi tutuluyor ve sonuçta elde edilen değer özel_islem_3
değişkenine aktarılıyor. */

ozel_islem_3 := null;

i:=0;

LOOP

EXIT WHEN i=to_number(to_char(sysdate,'MI'))+1;

ozel_islem_3 := ozel_islem_3 || asil_id || x_pw;

```

        i:=i+1;
    END LOOP;

    /* Uygulanan özel işlem sonucu elde edilen değer, ASIL_KULLANICI_SIFRE
       fonksiyonundaki gibi veri_sifrele fonksiyonu kullanılarak şifreleniyor ve yeni
       değer asil_sifre değişkenine aktarılıyor. */
    asil_sifre := DBA.Verı_Sifrele (ozel_islem_3);

    /* Ele alınan kullanıcının şifresi, tespit edilen şifreye göre değiştiriliyor. */
    EXECUTE IMMEDIATE "ALTER USER "||upper(asil_id)||" IDENTIFIED BY
        "||substr(asil_sifre,1,30)||""";

    END LOOP;
    Close cur1;
END;');
END;
/

```

4.2.8 Rol Şifrelerini Değiştiren Veritabanı Job'u

Asıl kullanıcı için şifre tespitinde kullanılan fonksiyonun (ROL_SIFRE) aynısı kullanılarak, kullanıcılara ait şifreler veritabanı tarafında 1 dakikalık periyotlar halinde ve dakika başlarında (örn. 10:01:00, 10:02:00 vb.) otomatik olarak değiştirilmektedir. Bu job DBA tarafından asıl veritabanı üzerinde çalıştırılmalıdır.

```

DECLARE jobno number;
BEGIN DBMS_JOB.SUBMIT(job => jobno,
    next_date => to_date('25.04.2003 00:00','dd.mm.yyyy HH:MI'),
    interval => '/*1:Mins*/ to_date(to_char(sysdate,"dd.mm.yyyy HH24:MI")||":00",
        "dd.mm.yyyy HH24:MI:SS")+1/(60*24)', what =>
    DECLARE
        rol_id varchar2(30);          /* Şifre korumalı rol adı */
        x_rol varchar2(32);          /* (3.4) denklemindeki xrol gizli anahtar değeri. */
        t varchar2(20);              /* Timestamp değeri. */
        k varchar2(1000);            /* Role özel üretilen anahtar değeri. */
        ozel_islem_4 varchar2(10000); /* ⊕4 özel işlemi sonucu elde edilen değer */
        ozel_islem_5 varchar2(10000); /* ⊕5 özel işlemi sonucu elde edilen değer */
        rol_sifre varchar2(32);      /* (3.5) denklemi sonucu elde edilen PWrol değeri. */
        i integer;

```

cursor cur1 is SELECT role FROM dba_roles WHERE password_required="YES";

BEGIN

/* Asıl veritabanı üzerindeki şifre korumalı rol isimleri teker teker alınıyor. */

Open cur1;

LOOP

FETCH cur1 INTO rol_id;

EXIT WHEN cur1% notfound;

/* Güncel zaman değeri ROL_SIFRE fonksiyonundaki gibi gun/ay/yıl saat:dakika formatında t değişkenine aktarılıyor. */

t := to_char(sysdate, 'dd.mm.yyyy HH24:MI');

/* ROL_SIFRE fonksiyonundaki gibi role özel olarak üretilen kod k değişkenine aktarılıyor. */

k := rol_id || "e özel bir anahtar";

/* t ve k değerleri ROL_SIFRE fonksiyonundaki gibi özel işleme tabi tutuluyor ve sonuçta elde edilen değer özel_islem_4 değişkenine aktarılıyor. */

ozel_islem_4 := null;

i:=0;

LOOP

EXIT WHEN i=to_number(to_char(sysdate, 'HH24'))+1;

ozel_islem_4 := ozel_islem_4 || t || k;

i:=i+1;

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, ROL_SIFRE fonksiyonundaki gibi veri_sifrele fonksiyonu kullanılarak şifreleniyor ve yeni değer x_rol değişkenine aktarılıyor. */

x_rol := DBA.Veri_Sifrele (ozel_islem_4);

/* x_rol değeri ile rol_id değeri ROL_SIFRE fonksiyonundaki gibi özel işleme tabi tutuluyor ve sonuçta elde edilen değer özel_islem_5 değişkenine aktarılıyor. */

ozel_islem_5 := null;

i:=0

LOOP

EXIT WHEN i=to_number(to_char(sysdate, 'MI'))+1

ozel_islem_5 := ozel_islem_5 || rol_id || x_rol;

i:=i+1;

```

END LOOP;

/* Uygulanan özel işlem sonucu elde edilen değer, ROL_SIFRE fonksiyonundaki
   gibi veri_sifrele fonksiyonu kullanılarak şifreleniyor ve yeni değer rol_sifre
   değişkenine aktarılıyor. */
rol_sifre := DBA.Veri_Sifrele (ozel_islem_5);

/* Ele alınan rolün şifresi, tespit edilen şifreye göre değiştiriliyor. */
EXECUTE IMMEDIATE "ALTER ROLE '||upper(rol_id)||' IDENTIFIED BY
                    '||substr(rol_sifre,1,30)||'";

END LOOP;
Close cur1;
END;);
END;
/

```

4.2.9 Son Kullanıcı Program Girişi

Uygulama tarafında çalışacak olan bu prosedür yardımıyla, son kullanıcılar piyon kullanıcı isimleri ve şifrelerini kullanarak piyon veritabanına bağlantı kurmaktadır. Daha sonra ASIL_KULLANICI_SIFRE fonksiyonu kullanarak piyon kullanıcı adından asıl kullanıcı adı ve şifresi elde edilmekte ve uygulama yazılımı içerisinde asıl veritabanına bağlantı kurulmaktadır.

```

PROCEDURE KULLANICI_BAGLANTI IS
    temp_kullanici_adi VARCHAR2(30);
    temp_sifre VARCHAR2(30);
    asil_id varchar2(30);
    asil_sifre varchar2(30);

BEGIN
    /* Kullanıcıya logon ekranı getiriliyor ve girdiği değerler değişkenlere aktarılıyor. */
    Logon_Screen;
    temp_kullanici_adi := Get_Application_Property(USERNAME);
    temp_sifre := Get_Application_Property(PASSWORD);
    IF temp_kullanici_adi is null or temp_sifre is null THEN
        Message('Kullanıcı Kodu ya da Şifre Hatalı ya da İşlemden Vazgeçilmiş.');
```

```

    ELSE

```

```

Logout;

/* Kullanıcı adı ve şifre doğru girilmiş ise piyon_vt ' ye bağlantı kuruluyor. */
Logon (temp_kullanici_adi,temp_sifre || '@piyon_vt');
END IF;

/* İlgili fonksiyonlar yardımıyla asıl kullanıcı adı ve şifresinin tespiti yapılıyor. */
asil_id:=asil_kullanici_id(user);
asil_sifre:=asil_kullanici_sifre(asil_kullanici_id(user));

/* Piyon veritabanı bağlantısı kesiliyor ve asıl veritabanı bağlantısı kuruluyor. */
Logout;
Logon(asil_id,asil_sifre|| '@asil_vt');
END;

```

4.2.10 Modüle Göre İlgili Rollerini Kullanıcıya Atayan Prosedür

Uygulama yazılımı tarafında çalışacak bu prosedür yardımıyla kullanıcı, kendisine ilk durumda pasif olarak atanan rollerden kullandığı modüle göre gerekli rolleri aktif hale geçirerek gerekli erişim yetkilerine sahip duruma gelecektir.

Modüllere göre atanacak olan roller, asıl veritabanında bir tabloda saklanarak dinamik olarak ya da program içerisinde statik olarak yapılandırılabilir. Aşağıda verilen örnekte modüle göre atanacak olan roller bir tabloda tutulmakta ve kullanılan modüle göre ilgili tablodan atanacak olan roller tespit edilmektedir.

```

PROCEDURE MODUL_ROL_ATA (P_MODUL_ADI VARCHAR2) IS
    atanacak_rol_adi varchar2(30);
    Cursor cur1 is SELECT rol_adi FROM DBA.MODUL_ATANACAK_ROLLER
                    WHERE modul_adi=p_modul_adi;

BEGIN
    /* Kullanılan modüle göre atanacak olan roller teker teker ele alınıyor. */
    Open cur1;
    LOOP
        EXIT WHEN cur1% notfound;
        FETCH cur1 INTO atanacak_rol_adi;
    
```

```

/* Ele alınan gerekli role ait şifre, daha önceden tanımlanmış olan fonksiyon
yardımıyla bulunarak aktif hale getiriliyor. */
Forms_Ddl ('SET ROLE "'||atanacak_rol_adi||'" IDENTIFIED BY
"'||Rol_Sifre(atanacak_rol_adi)||'"');
END LOOP;
Close cur1;
END;

```

Kullanılan modülden çıkıldığı zaman set edilen roller, aşağıdaki SQL ifadesi yardımıyla geriye alınır.

```
SET ROLE none
```

4.3 Performans ve Güvenlik Analizi

Son kullanıcılar illegal uygulama programları kullanarak piyon veritabanına bağlantı kurdukları zaman minimum erişim yetkilerine sahip olacaklar ve bu durum piyon veritabanında hassas veri bulunmadığı için hiçbir tehlike taşımayacaktır. Asıl veritabanındaki kullanıcı adı gizli olduğu ve son kullanıcı tarafından bilinmediği için asıl veritabanına bağlantı sağlanması zorlaştırılmıştır. Asıl kullanıcı adının tespit edilmesi durumunda dahi minimum erişim yetkileriyle bağlantı kurulduğu için tehlike boyutu çok düşük olacaktır.

Bu sistemde tehlike addedecek tek durum, oluşturulan algoritmaların kırılarak kullanıcı şifresi ve roller için üretilen şifrelerin bulunmasıdır. Eğer bir algoritmayı kırmak için;

- Gereken para, şifrelenmiş verinin değerinden daha fazlaysa,
- Gereken zaman, şifrelenmiş verinin saklı kalması için gereken zamandan daha uzunsa,
- Tek bir anahtar ile şifrelenmiş veri, algoritmayı kırmak için gerekenden daha az ise,

güvende olunduğu kabul edilebilir. [12]

Buna istinaden, kullanıcıya ve role ait şifrelerin birer dakikalık periyotlar halinde değiştirilmesi şifrelerin tespitini oldukça zorlaştıracaktır.

TADA sisteminde üç aşamalı kimlik doğrulama gerçekleştiği için her kimlik doğrulama işlemi esnasında bir zaman kaybı olmaktadır. Öyle ki;

- T_1 : 1. kimlik doğrulama işlemi için geçen süre
- T_{d123} : (1), (2) ve (3) denklemlerinin çalışma zamanı
- T_2 : 2. kimlik doğrulama işlemi için geçen süre
- T_{d45} : (4) ve (5) denklemlerinin çalışma zamanı
- T_3 : 3. kimlik doğrulama işlemi için geçen süre

olarak alırsak veritabanı bağlantısı sağlamak için geçen toplam zaman;

$$T_{\text{toplam_zaman}} = T_1 + T_{d123} + T_2 + T_{d45} + T_3 \quad (6)$$

olacaktır. Diğer veritabanı uygulama programları ise, kimlik doğrulama işlemini tek aşamada gerçekleştirdikleri için veritabanı bağlantısı için geçen toplam zaman T_1 olacaktır. Buna göre sistemin performansını etkileyen en büyük etken çalışan algoritmaların performansıdır. Bu algoritmaların düzgün dizayn edilmesi sistem performansını artıracaktır.

5 SONUÇ VE ÖNERİLER

Bu tez çalışmasında son kullanıcıların veritabanına, veritabanı yöneticileri tarafından onaylanan güvenli uygulama programları aracılığıyla bağlanmasını sağlamak amacıyla yeni bir sistem geliştirilmiş ve geliştirilen bu sistemle ilgili örnek uygulama kaynak kodları verilmiştir. Sistem yapısal olarak üç kimlik doğrulama (authentication) aşaması ve bir dinamik yetkilendirme (dynamic authorization) aşaması olmak dört aşamadan oluşmaktadır.

TADA sistemindeki her aşama birbiriyle ilişkili olarak veritabanı güvenliğini artırıcı mahiyette rol almaktadır. Geliştirilen bu sistem, son kullanıcıların veritabanı bağlantılarını sadece güvenilir uygulama programları aracılığıyla yapmaları sağlanmaktadır. Bunun neticesinde aşağıdaki sonuçlar elde edilir.

- Son kullanıcıların bütünlüğü bozacak illegal yazılımlar kullanarak veritabanı bağlantısı yapmaları engellenmiş olur.
- Kullanıcıların sadece uygulama programı tarafından izin verilen verilere ulaşma imkanları olduğu için izinsiz veri erişimleri engellenmiş olur.
- Kullanıcılar kendi sorgularını oluşturamadıkları için, erişim yetkileri olmayan hassas verileri de çıkarsamaları mümkün olmayacaktır.
- Kayıt ekleme ve değiştirme işlemleri, uygulama yazılımı içerisinde kontrollü bir şekilde yapılarak polyinstantiation problemi engellenmiş olur.
- Son kullanıcıların veritabanı üzerinde yapacakları bütün işlemler, sadece güvenilir uygulama programı aracılığıyla yapılacağı için yapılan işlemlerin kayıt edilmesi sonucunda kapsamlı bir denetleme sistemine sahip olunur.

Veritabanı uygulama programı, son kullanıcıların veritabanıyla ilgili gerekli tüm beklentilerini yerine getirecek şekilde oluşturulmalıdır. Bunun sağlanamaması durumunda kullanıcı yetkili olduğu verilere dahi erişim sağlayamayacağından dolayı

kullanıcıyı belirli programları kullandırmaya yöneltmek tamamıyla dezavantajlı hale dönüşmüş olur.

TADA sistemindeki kimlik doğrulama aşamalarında kullanılan algoritmalar veritabanı yöneticisi veya programcıya göre değişkenlik gösterebilir. Tek yönlü şifrelemede tercihe göre istenilen algoritmalar (MD4, MD5, RC5 vb.) kullanılabilir. Kullanılacak olan özel işlemler ve üretilen kodlar tamamen tercihe göre istenilen şekilde yapılabilir. Yani sistem yapısal olarak değiştirilmediği sürece istenilen işlem ve algoritmalar kullanılabilir.

Burada dikkat edilmesi gereken husus uygulama yazılımı ve veritabanı tarafında çalıştırılacak özel işlem ve algoritmaların sonucunun aynı olması gerekliliğidir. Aksi durumda şifre doğrulaması yapılamayacağı için sistemin işleyişi aksayacaktır.

Kullanıcılar sadece piyon kullanıcı olarak belirtilen kullanıcı adı ve şifresinden sorumlu olacaklardır. Kullanılacak özel işlem ve algoritmalar sonucunda piyon kullanıcı adından asıl kullanıcı adı ve şifresi tespit edilmektedir. Sonrasında asıl kullanıcı adı ve şifreyle veritabanı bağlantısı yapıp, kullanılan modüle göre dinamik yetkilendirmeler yapılmakta ve asıl kullanıcı adıyla veritabanı üzerinde işlem yapma hakkına sahip olunmaktadır. Asıl kullanıcı ve rol şifreleri birer dakikalık periyotlar içerisinde değiştirildiği için şifrelerin tespiti de pek fazla mümkün olmayacaktır.

Geliştirilen sistemde iki farklı veritabanı kullanılmaktadır. Bu veritabanları aynı bilgisayar üzerinde de yer alabilir, farklı bilgisayarlar üzerinde de yer alabilir. Farklı bilgisayarlar üzerinde yer alması durumunda bu bilgisayarlar arasındaki zaman senkronizasyonun sağlanması gerekir.

Bu çalışma, veritabanı sistemlerinde güvenlik artırımı sağlamak amacıyla yapılmış olan bir çalışmadır. TADA sistemi veritabanı güvenliği bütünüyle tek başına sağlayacak bir mekanizma değildir. DBMS özellikleriyle güvenliği sağlanmış olan bir veritabanına güvenli erişimi sağlamak amacıyla kullanılması durumunda güvenlik açısından iyi sonuçlar vermektedir.

Unutulmaması gereken bir şey var ki; hiçbir sistem tam anlamıyla güvenli değildir. Bu açıdan bakıldığında hedeflenen nokta, çıkartılan güvenlik politikalarıyla belirlenen stratejilere göre sağlanabilecek maksimum güvenliktir. Ayrıca gerekenden fazla güvenlik sağlamaya çalışmak ise maliyet artışı ve performans kayıplarını da beraberinde getirir.



KAYNAKLAR



- [1] Serrano M. A., Fernandez E., "Issues in Database and Data Warehouse Security", 5th World Multiconference on Systemics, Cybernetics and Informatics - The 7th International Conference on Information System Analysis and Synthesis, 2001
- [2] Coy S. P., "Security Implications of the Choice of Distributed Database Management System Model: Relational vs. Object-Oriented", 19th National Information Systems Security Conference, 1996
- [3] Al-Qahtani A., "Database Security", Computer Security Concepts Electronic Journal, Vol. 1, No 1, 2000
- [4] Thuraisingham B. And Ford W., "Security Constraint Processing in a Multilevel Secure Distributed Database Management System", IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No 2, 1995
- [5] Bertino E., Jajodia S. and Samarati P., "Database Security : Research and Practice" Information Systems, Vol.20, No.7, pp. 537-556, 1995
- [6] Wiseman S., "Database Security : Retrospective and Way Forward", Information Security Technical Report, Vol 6, No.2, pp. 30-43, 2001
- [7] Sandhu R.S., Coyne E.J., Feinstein H.L., Youman C.E., "Role-Based access control models", IEEE Computer 29 (2), pp. 38-47, 1996 .
- [8] Olivier M., Riet R, Gudes E., "Specifying Application-level Security in Workflow Systems", Proceedings of the Ninth International Workshop on Security of Data Intensive Applications (DEXA 98), pp 346-351, IEEE, 1998
- [9] Gudes E., Tubman A., "AutoWF-A secure Web workflow system using autonomous objects", Data&Knowledge Engineering, Vol.43, pp. 1-27, 2002

- 
- [10] Yip R. W. and Levitt K. N., "Data Level Inference Detection in Database Systems", IEEE Computer Security Foundations Workshop, Rockport, Massachusetts, June, 1998
- [11] Laferriere C., "Predicate Based Polyinstantiation in Multilevel Secure DBMS", Computers & Security, Vol.10, pp. 41-49, 1991
- [12] Schneier B., "Applied Cryptography", John Wiley&Sons, Inc. , Second Edition, 1996
- [13] Hwang M.S., Lee C.C. and Tang Y.L., "A Simple Remote User Authentication Scheme", Mathematical and Computer Modelling, Vol. 36, pp. 103-107, 2002
- [14] Lee C.C., Hwang M.S. and Yang W.P., "A Flexible Remote User Authentication Scheme Using Smart Cards", ACM Operating Systems Review, Vol. 36, No. 3, pp. 46-52, 2002
- [15] Harn L., Lin H.Y. and Yang S., "A Software Authentication System for Information Integrity", Computers & Security, Vol. 11, pp. 747-752, 1992
- [16] Jajodia S., "Database Security and Privacy", ACM Computing Surveys, Vol.28, No.1, 1996
- [17] Hardjono T., Zheng Y. and Seberry J., "Database Authentication Revisited", Computers & Security, Vol. 13, pp. 573-580, 1994
- [18] Qu T., "Database Security in Assets of Companies", Article in 7-110.501 Seminar on Network Security, 2001
- [19] Yarımağan Ü., "Veritabanı Sistemleri", Akademi Yayın Hizmetleri San. Tic. Ltd. Şti., 2000
- [20] Kevin L., "Oracle8 DBA Handbook", Oracle Press Editions, 1998

ÖZGEÇMİŞ



1979 yılında Kocaeli' de doğdu. 1995 yılında burslu olarak okuduğu Özel Alanya Lisesi' ndeki öğrenimini tamamladıktan sonra aynı sene içerisinde başladığı Kocaeli Üniversitesi Bilgisayar Bilimleri Mühendisliği Bölümü' nden 1999 yılında mezun oldu. 1999 Mart ayında TÜBİTAK Ulusal Metroloji Enstitüsü (UME) Bilgi İşlem Merkezi' nde iş hayatına atıldı ve halen aynı yerde çalışmaya devam etmektedir. Evli olan Cihan KILIÇ, veritabanı sistemleri yönetimi ve güvenliği, veritabanı programlama ve dinamik web programlama konularına ilgi duymaktadır.

