ANKA e-DERGİ
Journal of Phoenix
(Cilt 1/Sayı 1/Yıl 2016)

Hibernate Kalıcı Katmanın Gerçekleştirilmesi

Halit ÇETİNER[a*], İbrahim ÇETİNER[b]

[a] *Süleyman Demirel Üniversitesi, Keçiborlu M.Y.O., Bilgisayar Programcılığı Bölümü, Isparta, TÜRKİYE*
[b] *Süleyman Demirel Üniversitesi, Keçiborlu M.Y.O., Otomotiv Teknolojisi Bölümü, Isparta, TÜRKİYE*
***Sorumlu yazarın e-posta adresi:** halitcetiner @sdu.edu.tr*

Özet:
Geleneksel ilişkili veri tabanı üzerinde geleneksel yöntemler kullanılarak veri tabanı işlemleri gerçekleştirilebilmektedir. Klasik yöntemleri kullanmanın yerine nesneye yönelik bir programlama mantığını kullanan nesne ilişkili teknolojilerin geliştirilmesinde artış görülmektedir. Yapılan bu çalışmada bu teknolojilerden biri olan Hibernate teknolojisi incelenmiştir. Bu teknolojinin yapısal olarak mimarisi incelenmiştir. Aynı zamanda teknolojinin uygulanması sırasında meydana getirdiği değişimler incelenmiştir.

Anahtar Kelimeler: Java Swing, Kalıcı Nesneler, POJO, Hibernate 4.3, Sql Server 2012

Implementation of the Hibernate Persistence Layer

Abstract:
Database operations can be performed using conventional methods on conventional relational database. In the development of object related technologies using object-oriented programming instead of classical methods is seen to increase. In this study, Hibernate technology is one of these technologies, which has been examined. Structural architecture of this technology is examined. We also worked on the changes caused during implementation of the technology.

Key words: Java Swing, Persistence Object, POJO, Hibernate 4.3, Sql Server 2012

1. GİRİŞ

Bugün, birçok yazılım geliştirici kurumsal bilgi sistemleri (KBS) ile çalışma gerçekleştirmektedir. Bu uygulamaların çeşitleri arasında da çoklu fiziksel yerlerdeki birçok kullanıcı arasında bu bilgiyi paylaşan ve yapısal bilgiyi kayıt eden, oluşturan ve yönetebilen uygulamalardan oluşmaktadır. KBS bilgisinin depolanması SQL tabanlı veri tabanı yönetim sistemleri kullanımı ile gerçekleştirilmektedir. Her şirket SQL veri tabanına kariyeri boyunca en az bir kez karşılaşmıştır. Çoğunlukla şirketin çekirdeğindeki ilişkisel veri tabanı teknolojisi birimi tamamen bağımlıdır. Geçen yıllar içerisinde Java programlama dilinin geniş uyumluluğu yazılım geliştirebilmek için nesne tabanlı çekim örneklerinin üstünlük kazanmasını sağlamıştır. Geliştiriciler nesne tabanlı veri tabanı yönetim sistemlerinin faydaları noktasında odaklanmışlardır. Ancak, şirketlerin büyük çoğunluğu pahalı ilişkisel veri tabanı sistemlerinde uzun vadeli yatırımlara bağlı olarak hareket etmektedir. Amaç sadece eski verileri yeni nesne tabanlı farklı uygulama çeşitleri tarafından erişilmesi sağlamak değil aynı zamanda satıcıların ürünlerini garanti altına almaktır. Ancak, ilişkisel sistemde verinin tabular gösterimi nesne tabanlı Java uygulamalarında kullanılan nesnelerin ağlarından

temel olarak farklıdır. Bu farklılık nesne/ilişkisel paradigma uyumsuzluğuna yol açmaktadır. Geleneksel olarak uyumsuzluğun önemi ve maliyeti göz ardı edilmiştir ve uyumsuzluğu çözmek için geliştirilen araçlar yetersiz olduğu belirlenmiştir. Bununla birlikte, Java geliştiricileri uyumsuzluk için ilişkisel teknolojileri suçlarken, veri uzmanları nesne teknolojisini suçlamaktadır.

ORM uyumsuzluk problemlerini otomatik olarak çözmek için verilen bir isimdir. Aynı zamanda bilgisayar yazılımındaki ORM nesne tabanlı programlama dilleri ve ilişkisel veri tabanlarında uygun olmayan tip sistemleri arasında veri dönüştürebilmek için bir programlama tekniğidir. ORM teknolojileri nesne tabanlı mimari sistem ve ilişkisel çevre arasında aracılık etmektedir. Bu programlama dilleri içerisinde kullanılabilen sanal nesne veri tabanı etkisi yaratmaktadır. Nesne ilişkisel eşleşmeyi gerçekleştirebilmek için ücretsiz ve ticari paketler bulunmaktadır. Geliştiriciler için yorucu ve bıktırıcı veri erişim kodları ile uğraşmak yerine ORM teknolojisinin doğuşunu hızlandırmak daha fazla kabul görmüştür. ORM katman ile oluşturulan uygulamalarda SQL şemalarını anlamak veya iç nesnedeki değişiklikler ile daha fazla uğraşmak, daha az satıcıya özgü olmak, daha fazla ölçüle bilirlik ve daha ucuz olması beklenebilir.

Geleneksel ilişkisel veri tabanları, Hibernate sayesinde nesne/ilişkisel eşleştirme araçları ile dönüşüme uğramaktadır. Böylece Hibernate gelişmiş veriler için raporlar üretmek, araştırma veya sorgulama yapmak gibi hizmetleri sağlayan ve veri yapılarını anlayarak eşleşen kalıcı nesneler üzerinden veri tabanlarının işlenmesini sağlayan bir Java teknolojisidir. Gaving King Hibernate teknolojisini geliştirmeye 2001 yılının sonlarında başladı. Aynı zamanda karmaşık veri modelleri ile çözülmesi zor problemlerin ölçeklenmesini de sağlayacak şekilde bir mimari oluşturmaya çalıştı. Hibernate geliştiricileri istekleri karşılayabilmek için teknik liderlik, geliştirici üreticiliğine vurgu yaparak problemlerin çözümünde basit yaklaşımları ön plana çıkarmıştır. Hibernate binlerce uygulamada ve on binlerce kullanıcı tarafından kullanılmaktadır. Gaving King ve ekibi istekleri karşılayabilmek için tam zamanlı olarak tüm performanslarını projenin isteklerine harcamışlardır (Bauer and King, 2005). JBoss Inc. Şirketi tarafından Hibernate satın alınması ile ticari olmayan ve bağımsız bir platformda geliştirilen teknolojisi ticari olarak destek ve eğitim yöntemlerini kazanmıştır. ORM teknolojileri artan bir şekilde ana akım olmaya başladıkça, Hibernate sağlamış olduğu dinamiklik, esneklik daha da önem kazanmaya başlamıştır.

(Kasapbaşı, 2010) yaptıkları çalışmasında farklı ORM uygulamalarını değerlendirme gerçekleştirerek, karşılaştırmalar gerçekleştirmeye çalışmıştır.

(Xue and Zhu, 2009) yaptıkları çalışmada finansal yönetimde kullanılmak üzere veri üzerinde raporlama işlemlerini gerçekleştirmeye yarayacak Hibernate teknolojisini kullanan bir çalışma gerçekleştirmiştir.

(Cao et. al., 2009) yaptıkları çalışmada Hibernate teknolojisi kullanarak hastane tıp depolama sistemi gerçekleştirmiştir. Bu sistemde okunabilirliği, tekrar kullanılabilirliği artırmaya çalışmışlardır.

2. NESNE İLİŞKİSEL KALICILIK

Kalıcı veriyi yönetme yaklaşımı, her yazılım projesine çalışmaya başlandığında bir anahtar tasarım kararıdır. Kalıcı veri, Java uygulamaları için yeni veya alışılmamış gereksinim değildir. Bu noktada yazılımcı benzer ve iyi kurgulanmış kalıcı çözümler arasından basit bir

seçim yapmak istemektedir. Java toplulukları için kalıcılık sıcak bir konudur. Birçok geliştirici problemin kapsamına bile daha hakim değildir. Bu noktada farklı çözüm önerileri bulunmaktadır. SQL ve JDBC deki OOGS (oluşturmak, okumak, güncellemek, silmek) olarak ifade edebileceğimiz temel işlemler mi otomatikleştirilmelidir, yoksa SQL diyalektleri ile veri tabanı yönetim sistemi mi başarılmalıdır, yoksa SQL'i tamamen ortadan kaldıracak yeni bir teknoloji mi geliştirilmelidir şeklinde farklı sorular yönetilmiştir. Sonuç olarak ilerleyen zamanlarda çözüm önerisi olarak gün geçtikçe daha da artış gösteren ve sorunlara kalıcı çözüm sunan ORM teknolojisi kabul görmüştür. Bu noktada açık kaynak kodlu bir ORM teknolojisi olan Hibernate Java dili ile geliştirilmeye çalışılmıştır.

Hibernate, Java 'da kalıcı veriyi yönetme problemini tamamen çözmeyi amaçlayan bir projedir. Bu proje ilişkisel veri tabanı ile etkileşimli uygulamaların gerçekleştirilmesine aracılık etmektedir. Hibernate müdahaleci olmayan bir çözüm yöntemi sunmayı hedeflemektedir. Sunmak istediği proje geliştirme sırasında geliştirici iş mantığı ve kalıcı sınıflar yazarken, birçok Hibernate özgü kuralları ve tasarım örüntülerini takip etmek zorunda kalmamaktadır. Böylece Hibernate yeni ve var olan birçok uygulamaya kolay ve yumuşak bir şekilde bütünleşme imkânı sağlamaktadır. Bu aynı zamanda uygulamanın diğer katmanları üzerinde herhangi bir değişiklik gerçekleştirmeden uygulama geliştirilmesine imkân tanımaktadır.

Hemen hemen bütün uygulamalar kalıcı veri gerektirmektedir. Kalıcılık uygulama geliştirmede temel kavramlardan birisidir. Verinin kalıcılığını gerçekleştirebilmek için ilişkisel veri tabanları ile çalışmak gerekmektedir. İlişkisel veri tabanları son derece esnek ve veri yönetimi için dinamik bir yaklaşım sunmaktadır. İlişkisel veri tabanı yönetim sistemleri sadece Java'ya özgü değildir. Aynı zamanda ilişkisel veri tabanı belirli uygulamalara özgü de değildir. İlişkisel teknoloji farklı teknolojiler arasında veya farklı uygulamalar arasında veri paylaşımına olanak sağlamaktadır. İlişkisel teknoloji birçok farklı sistemlerin ve teknoloji platformlarının genel paydasıdır. Bu yüzden ilişkisel veri modeli iş varlıklarının genel anlamda kurumsal planda sunumudur. İlişkisel veri tabanı yönetim sistemleri SQL tabanlı uygulama programlama ara yüzlerine sahiptir.

Hibernate etkinliğini kullanabilmek için, ilişkisel modeli ve SQL'i sağlam bir şekilde anlamak ön koşul olarak görülmektedir. Hibernate uygulamalarında performansı ayarlamada SQL bilgisi önemli bir faktör haline gelmektedir. Hibernate uygulamalarında birçok kod görevlerinin otomatik olarak tekrar ettiğini düşünebiliriz, ancak bu durum kalıcı teknolojisi bilgisinin modern SQL veri tabanlarının tüm gücünü kullanabilmek için genişletilmesinden kaynaklanmaktadır. Bu durumda kalıcı verilerin yönetiminde anlaşılması gereken asıl hedef esnekliktir.

Java uygulamalarında SQL veri tabanları ile çalışıyorken, Java veri tabanı bağlantısını (JDBC) kütüphanesi aracılığıyla yapmaktadır. Bu SQL Java kodu içerisine gömülerek gerçekleştirilmektedir. Bu düşük seviye veri erişim görevleri olarak görülmektedir. Birçok uygulama geliştirici veri erişimi noktasında iş problemleri ile daha fazla ilgilenmektedir. Sıkıcı ve mekanik olan detaylar çalışmaları açık olmayan bir duruma getirmektedir.
Veri erişim noktasında var olan sıkıntılar veri tabanlar üzerindeki varlık birimlerini sınıf örnekleri ile ifade edilerek düzenlenmesine imkân sağlamıştır. Karmaşık nesneleri almak ve kaydedebilmek için nesne tabanlı olarak ifade edilen tablolar üzerinden kodlama gerçekleştirilmeye başlanmıştır. Modern ilişkisel veri tabanları verinin sıralama, araştırma ve toplanmasına imkân tanıyarak kalıcı verinin yapısal sunumunun oluşturulmasını sağlamaktadır. Veri tabanı yönetim sistemleri veri entegrasyonu ve eş zamanlılık

yönetiminden sorumludur. Aynı zamanda çoklu kullanıcı ve uygulama arasında veri paylaşımından sorumludur. Bir veri tabanı yönetim sistemi veri seviye güvenliği sağlamaktadır.

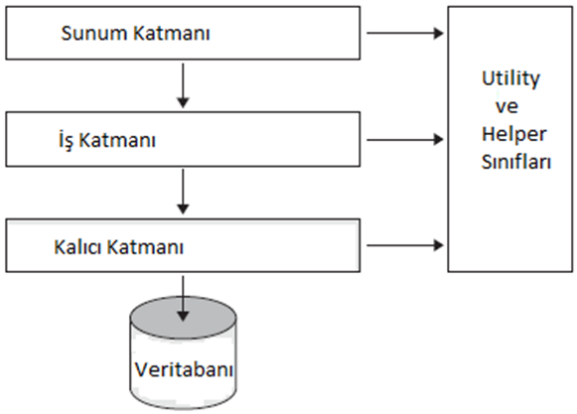
Genelde veri tabanlarında odaklanılan noktalar depolama, organizasyon ve yapısal verinin alımı, eş zamanlılık, veri bütünleşmesi ve veri paylaşımı olarak özetlenmektedir. Bu noktalar kapsamında var olan problemleri çözebilmek için modeller geliştirilmektedir. Hibernate teknolojisinin temelini oluşturan bu modeller veri tabanında tabloların, Java uygulamalarında sınıflarını otomatik olarak oluşturmaktadır.

2.1. Katmanlar

Orta veya büyük boyutlu uygulamalarda, katman ile sınıfları organize etmek için hassas davranmak gerekmektedir. Kalıcılık bir katmandır. Diğer katmanlar sunum, iş akış ve iş mantık katmanıdır. Bu katmanlar çapraz kesim (cross cutting) katmanlar olarak isimlendirilmektedir. Genel olarak çapraz kesim katmanları günlük, yetkilendirme ve işlem sınırlandırma gibi işlevleri içermektedir. Nesne tabanlı mimari sunum yapan katmanları da kapsamaktadır. Katmanlı sistem mimarisinde kalıcı katmanı ayırabilmek için kalıcılıktan sorumlu tüm sınıf ve bileşenleri gruplamak gerekmektedir.

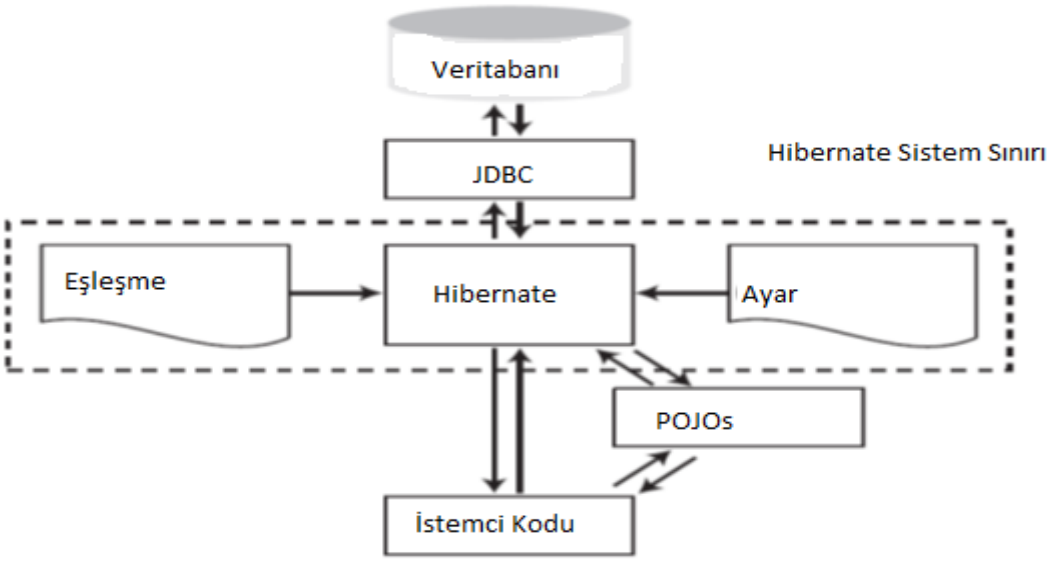
Katmanlı mimari değişik katmanları tanımlayabilmek için ara yüzler tanımlamaktadır. Katmanlar yukarıdan aşağıya iletişim kurmaktadır. Bir katman sadece aşağısındaki katmana doğrudan bağımlıdır. Her katman aşağısındaki katman hariç diğer katmanlardan habersizdir. Farklı uygulamalar katmanları farklı biçimde gruplamaktadır. Bu yüzden farklı katmanlar tanımlamaktadır. Yüksek seviye uygulama mimarisi 3 katman kullanmaktadır. Bunlardan birincisi sunum, ikincisi iş katmanı ve üçüncüsü de kalıcı katmandır.

Sunum katmanı, kullanıcı ara yüz mantığı en üstte bulunmaktadır. Sayfa kontrolü, ekran gezinme formları gibi kullanıcı ile doğrudan bağlantı halinde olan katmandır. İş katmanı herhangi bir iş kurallarını gerçekleştirmek veya problem alanındaki kullanıcılar tarafından anlaşılmayı gerektiren sistem gereksinimlerinden sorumludur. Son olarak da kalıcı katmanda tanımlı olan modelleri tekrar kullanabilmeyi sağlamaktadır. Bir ya da daha fazla veri stokları üzerinde veri kaydetmek, sorgulamak gibi CRUD işlemlerinden sorumlu sınıflar ve bileşenlerin gruplandığı katmandır. Veri tabanı Java uygulamaları haricinde de mevcuttur. İlişkisel şemaları ve kayıtlı prosedürler gibi verileri, fonksiyonları tutmaktadır. Helper/Utility sınıfları hata yakalama sınıfları gibi uygulamanın her katmanında kullanılabilen altyapısal bir kümedir. Katmanlar Şekil 1.'de gösterilmiştir.



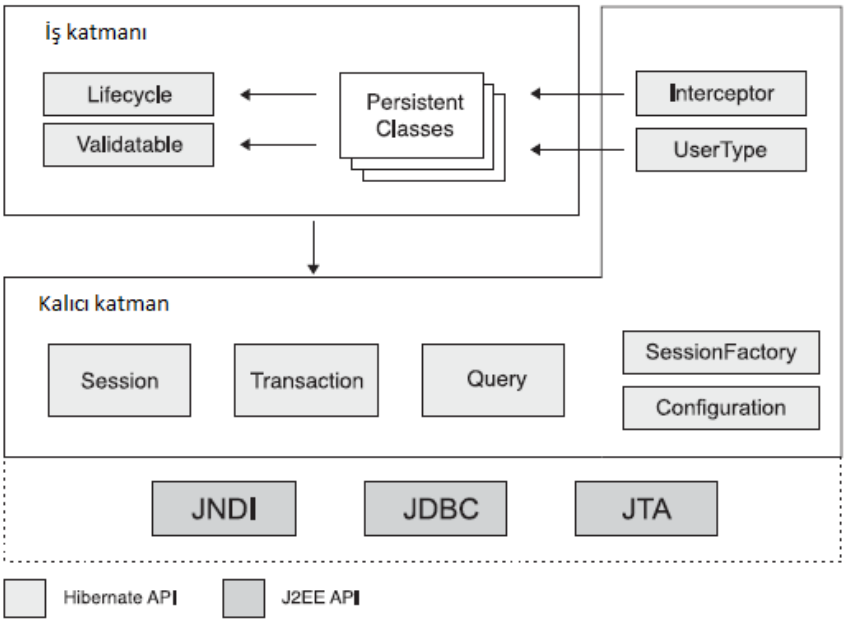
Şekil 1. *Kalıcı Katmanın Katmanlı Mimarideki Gösterimi*

Birçok önemli uygulama ilişkisel veri tabanı içermektedir. Birçok ticari uygulamaların dayanak noktası kataloglar, müşteri listeleri, bağlantı listeleri, yayınlanmış metin ve mimari tasarımlar gibi sıralı bilgilerin büyük ölçekli depolanmasından oluşmaktadır. Online olarak gerçekleştirilen uygulamaların büyük bir kısmı veri tabanlarını kullanmaktadır. Geleneksel JDBC kütüphaneleri yerine daha kapsamlı ilişkisel veri tabanı ile uygun bir ilişki kurulmasını sağlayacak Hibernate teknolojisi iyi bir tercih olarak düşünülmektedir. Hibernate düz eski Java nesneleri (POJO) yardımıyla veri tabanında oluşturulan kayıtları kaydetmektedir. Şekil 2. Hibernate teknolojisinin veri tabanı ile istemci kodu arasında uygulamaya nasıl uyum sağladığını göstermektedir. Geleneksel Java nesnelerinin doğrudan kalıcılığını sağlamak için nesne ilişkisel eşleşme terimi kullanılmaktadır. Yani Java'daki nesnelerin, veri tabanındaki ilişkisel varlıklar ile otomatik olarak eşleşmesini sağlamaktadır.



Şekil 2. *Java Uygulamalarında Hibernate Rolü (Minter and Linwood, 2006)*

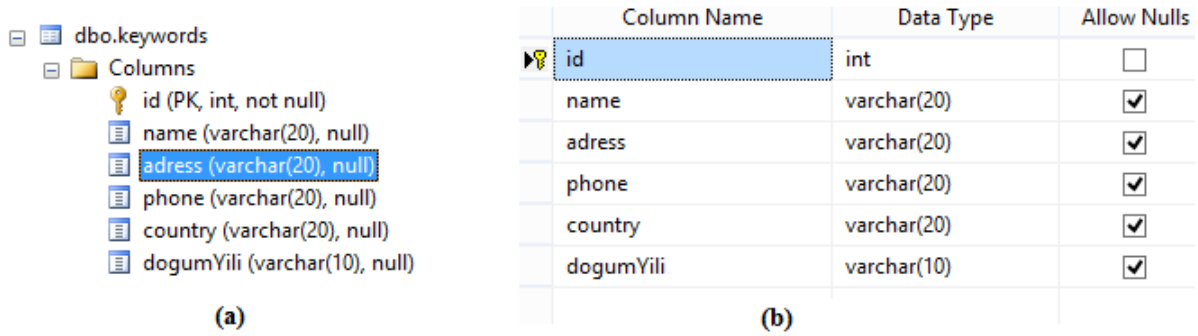
Geleneksel katmanlı uygulamalarda iş katmanı, kalıcı katmanın bir istemcisi olarak davranmaktadır. Bazı basit uygulamalarda kalıcı mantıktan iş mantığını kolayca ayırmak zordur. Uygulamalar tarafından çağrılan ara yüzler, temel CRUD ve sorgulama işlemlerini gerçekleştirmektedir. Bu ara yüzler Hibernate'taki iş/kontrol mantık uygulamalarının ana noktalarıdır. Bunlar Session, Transaction ve Query 'dir. Sırasıyla oturum HibernateUtil. java sınıfından veri tabanı ile eşleşmiş ayar dosyasını kullanarak veri tabanına bağlantı sağlamaktadır. Aynı zamanda Session veri tabanına kaydetme, silme işlemlerini POJO sınıfları aracılığıyla üretilen sınıfların yardımıyla gerçekleştirmektedir. Transaction kaydedilen verilerin onaylanması ya da geriye dönülmesi işlemlerini gerçekleştirmektedir. Query işlemleri Hibernate sorgularını gerçekleştirmemizi sağlamaktadır. Ara yüzler Configuration sınıflarını ayarlamak için altyapısal kodlamasını oluşturmaktadır. Callback arayüzleri ise Hibernate içerisinde oluşan Interceptor, Lifecycle, Validatable gibi olayların etkileşimine izin vermektedir (Bauer and King, 2005).



Şekil 3. *Katmanlı Mimaride Hibernate API'lerinin Yüksek Seviyeli Görünümü*

3. MATERYAL VE METOT

Geleneksel ilişkisel veri tabanları ile haberleşme teknolojilerinin yerine ORM tabanlı olarak çalışan Hibernate 4.3. teknolojisi kullanan bir çalışma gerçekleştirilmiştir. Gerçekleştirilen çalışmada Sql Server 2012'de oluşturulan bir tablo üzerinde temel CRUD işlemlerinin gerçekleştirilmesi sağlanmıştır.



Şekil 4. *(a) Kullanılan Veri Tabanı Tablosunun Veri Tipleri, (b) Nesne Penceresindeki Tablo Özellikleri*

Şekil 4.'te gösterilen şekilde keywords adlı bir tablo görmekteyiz. Oluşturulan tablo adına Hibernate tarafından sınıf oluşturulacaktır. Oluşturulan sınıflar POJO sınıfı olarak otomatik olarak oluşturulacaktır.

Şekil 2.'de gösterilen şekilde, geleneksel Java nesnelerinin doğrudan kalıcılığının sağlanmasına nesne ilişkisel eşleşme olduğu anlatılmaktadır. Yani, bu Java'daki nesnelerin veri tabanındaki ilişkisel varlıklar ile eşleşmesini göstermektedir. POJO'lar herhangi bir Java nesnesi olabilmektedir. Hibernate çok az kısıtla POJO'lara izin vermektedir. Hibernate burada verdiği tek kural özel varsayılan olarak yapıcı metodun tanımlanmak zorunda olmasıdır. Burada tablo üzerinde tüm CRUD işlemleri tablo adından türetilerek tanımlanan sınıf

aracılığıyla gerçekleştirilecektir. Bu sınıftan da 3 farklı yapıcı metodun tanımlanmış olması 3 farklı şekilde bu sınıftan nesne tanımlanabileceğini göstermektedir. Şekil 5.'de yapıcı metotların tanımlanmış halleri gösterilmektedir. 1. yapıcı metot ile herhangi bir parametreyi başlangıçta veremezken, 2.yapıcı metotta sadece id sütünü bilgisi, 3.yapıcı metotta ise tüm tablodaki sütunları parametre olarak verebilmekteyiz. özel erişim tipi ile tanımlanmış olan tüm değişkenler tablodaki sütunları temsil etmektedirler.

```
public class Keywords  implements java.io.Serializable {


    private int id;
    private String name;
    private String adress;
    private String phone;
    private String country;
    private String dogumYili;

    public Keywords() {
    }


    public Keywords(int id) {
        this.id = id;
    }
    public Keywords(int id, String name, String adress, String phone, String country, String dogumYili) {
       this.id = id;
       this.name = name;
       this.adress = adress;
       this.phone = phone;
       this.country = country;
       this.dogumYili = dogumYili;
    }
```

Şekil 5. *POJO Sınıflarındaki 3 Farklı Yapıcı Metodun Gösterimi*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-confi
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver</property>
    <property name="hibernate.connection.url">jdbc:sqlserver://localhost\HCETINER:1433;databaseName=kisi</property>
    <property name="hibernate.connection.username">hcetiner</property>
    <property name="hibernate.connection.password">ps090*21</property>
    <property name="hibernate.dialect">org.hibernate.dialect.SQLServerDialect</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.query.factory_class">org.hibernate.hql.internal.classic.ClassicQueryTranslatorFactory</property>
    <property name="connection.pool_size">1</property>
    <property name="current_session_context_class">thread</property>
    <mapping resource="kisi/entity/Keywords.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Şekil 6. *Kişi Adlı Veri Tabanı Bağlantısı ve Eşleşme Sayfasının Bilgilerini Tutan Ayar Dosyası*

Şekil 6.'da gösterilen ayar dosyasında veri tabanı bağlantı bilgileri (hibernate.connection.driver_class), veri tabanı için bağlantı yolu (hibernate.connection.url), veri tabanı kullanıcı adı (hibernate.connection.username), veri tabanı kullanıcı parolası (hibernate.connection.password), veri tabanında veri tabanına değişiklik gösteren diyalekt (hibernate.dialect), SQL kodunun gösterimi (hibernate.show_sql, veri tabanı sorgularının HQL adlı Hibernate sorgu diline dönüşüm sağlayacak sınıfı seçen özellik olan özelliği atamak (hibernate.query.factory_class), Hibernate veri tabanı bağlantı havuzunda bekleyen bağlantıların sayısının limitini göstermektedir (connection.pool_size), açılan mevcut oturumun bağlanılıp yönetileceğini göstermektedir (current_session_context_class) ve

<mapping resource="kisi/entity/Keywords.hbm.xml"/> ile de tablo ile eşleşmesi bulunan xml dosyasının yolunu göstermektedir.

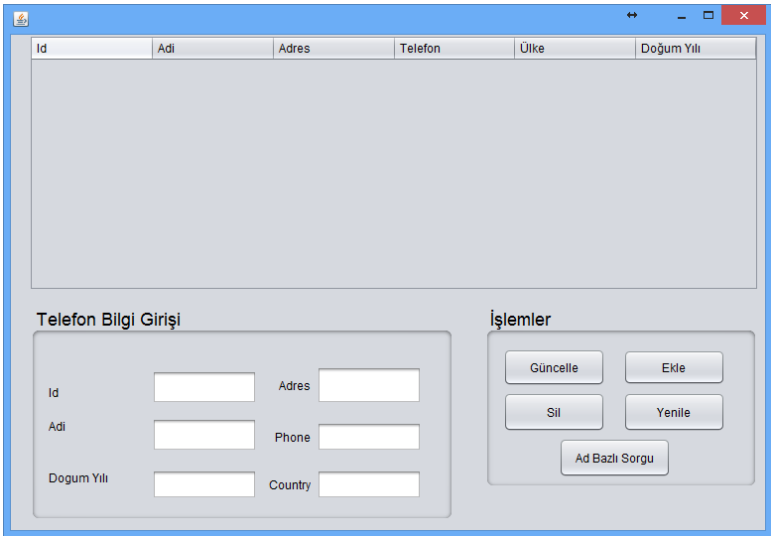
```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 11.Kas.2015 00:42:15 by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
    <class name="kisi.entity.Keywords" table="keywords" schema="dbo" catalog="kisi" optimistic-lock="version">
        <id name="id" type="int">
            <column name="id" />
            <generator class="assigned" />
            <!--<generator class="increment" />-->
        </id>
        <property name="name" type="string">
            <column name="name" length="20" />
        </property>
        <property name="adress" type="string">
            <column name="adress" length="20" />
        </property>
        <property name="phone" type="string">
            <column name="phone" length="20" />
        </property>
        <property name="country" type="string">
            <column name="country" length="20" />
        </property>
        <property name="dogumYili" type="string">
            <column name="dogumYili" length="10" />
        </property>
    </class>
</hibernate-mapping>
```

Şekil 1. *Kişi Adlı Veri Tabanı ile İlişkili POJO Sınıfı Bilgileri ve Tablo Bilgileri ile Sınıf Bilgilerinin Tutulduğu Eşleşme Dosyası*

Şekil 7.'de, Şekil 4.b'de gösterilen tablo bilgilerine göre üretilen Java POJO sınıfına uygun olarak otomatik üretilen hbm.xml uzantılı dosyanın içeriği gösterilmektedir. Bu eşleşme dosyasında tablodaki sütün bilgilerinin eşleştikleri açık bir şekilde görmekteyiz.

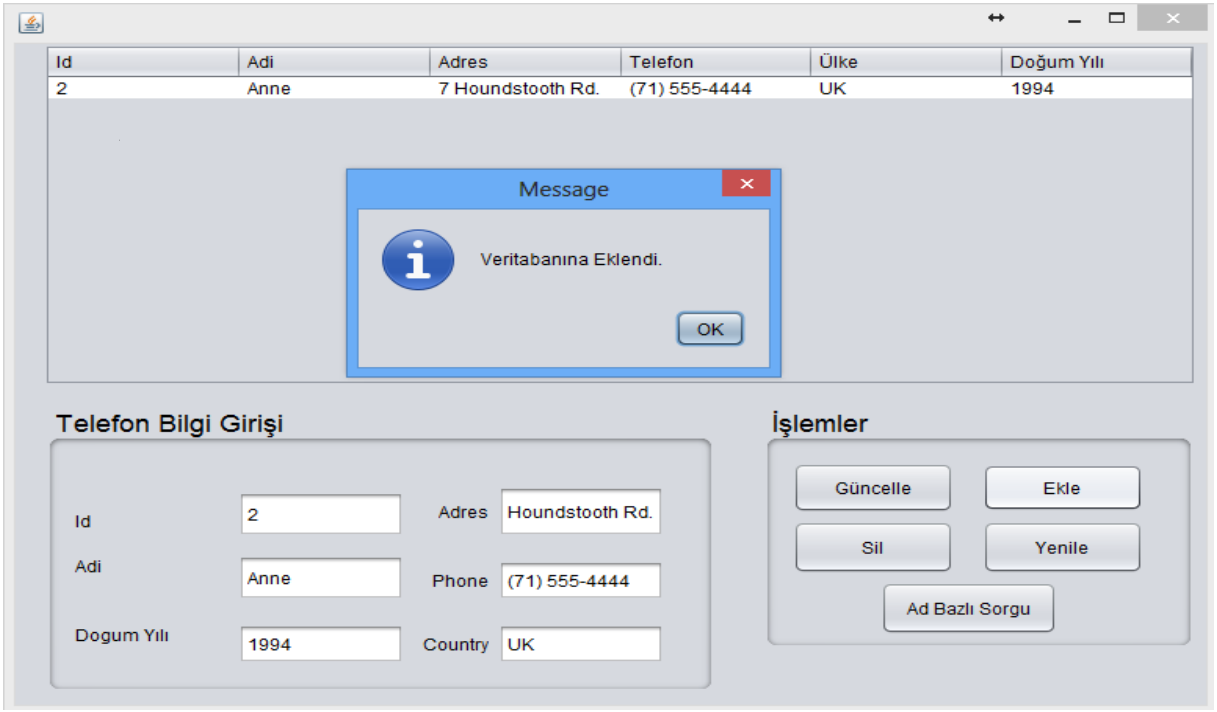
4. ARAŞTIRMA BULGULARI VE TARTIŞMA

Şekil 1., 2. ve 3. 'te Hibernate teknolojisinin yapısal alt yapısı gösterilmeye çalışıldı. Şekil 4., 5., 6. ve 7.'de ise bir gerçekleştirilen telefon uygulamasında kisi adlı bir Sql server veri tabanında tutulan verilerin ayar, eşleşme dosyaları gösterilmiştir. Şekil 8.'de geliştirilmiş olan basit bir Java Swing uygulaması üzerinden arka planda Hibernate teknolojisi kullanarak temel CRUD işlemlerinin gerçekleştirilmesi sağlanmıştır.



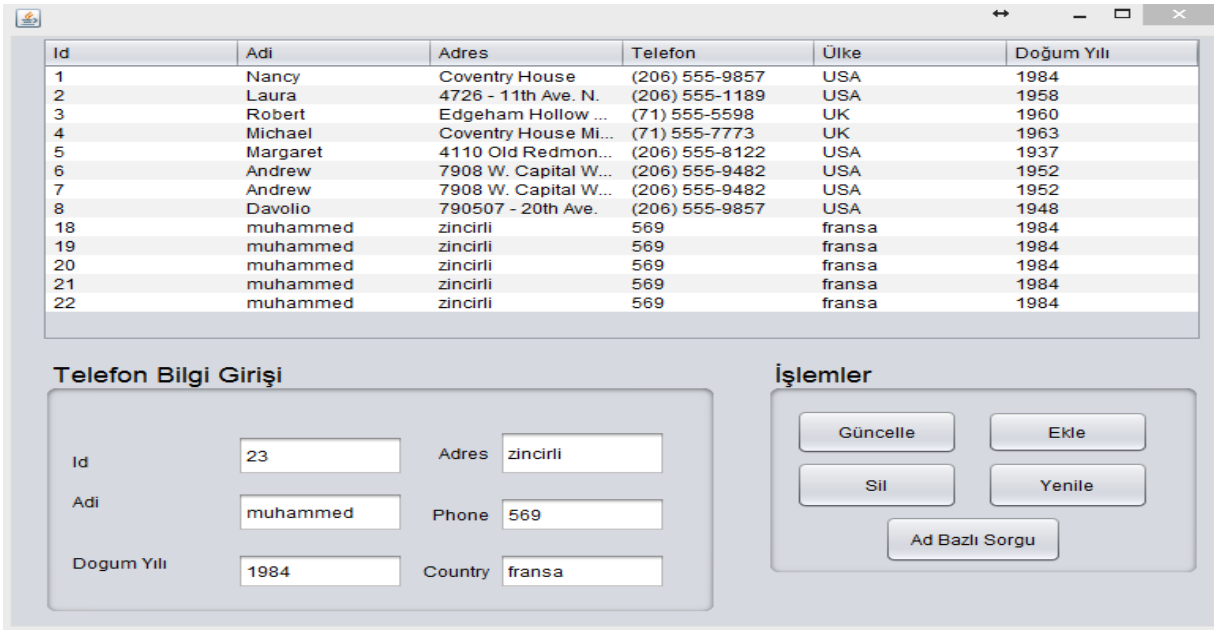
Şekil 8. *Java Swing ara yüzü*

Şekil 8.'de HQL komutları ile Sql server veri tabanı üzerinde ekleme, silme, güncelleştirme, sorgulama ve isim bazlı sorgulama gerçekleştirebilecek bir ara yüzü göstermektedir.



Şekil 2. *HQL komutları ile veri tabanına ekleme işlemi sonucu*

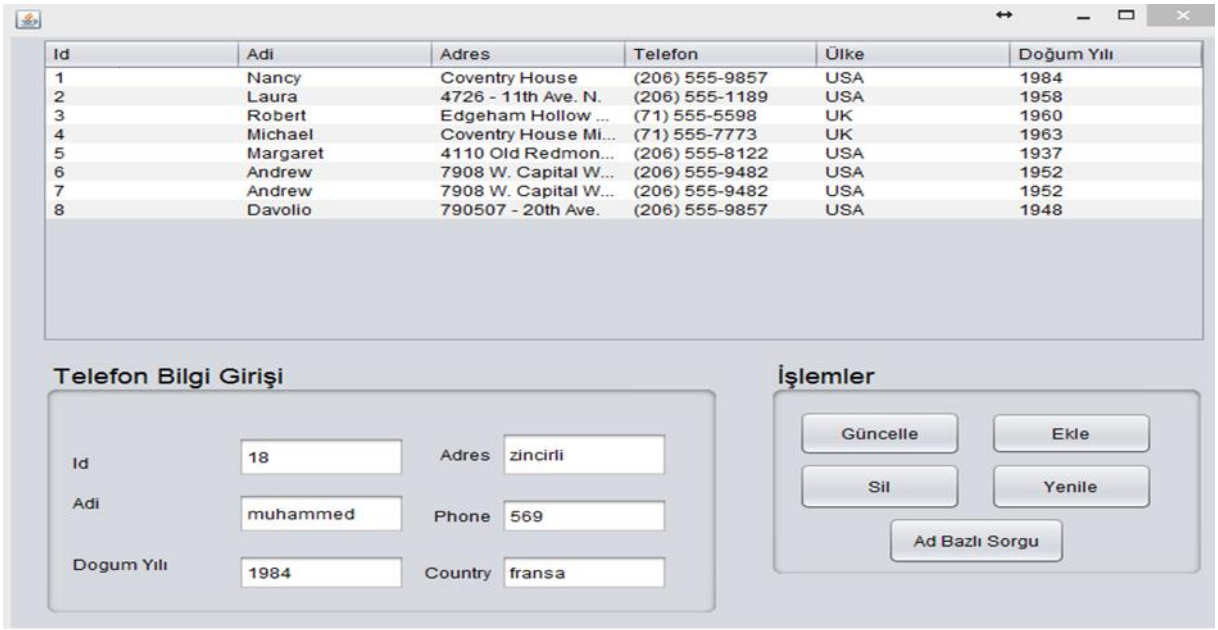
Otomatik olarak Hibernate teknolojisi tarafından üretilen sorgu Hibernate: insert into kisi.dbo.keywords (name, adress, phone, country, dogumYili, id) values (?, ?, ?, ?, ?, ?) şeklinde konsola yazdırılmıştır. Şekil 9.'da gösterilen şekilde ekleme işlemi için otomatik olarak oluşturulan sorgu sonucundaki ekleme işlemini göstermektedir.



Şekil 10. *HQL komutları ile veri tabanından tüm kayıtların getirilmesi*

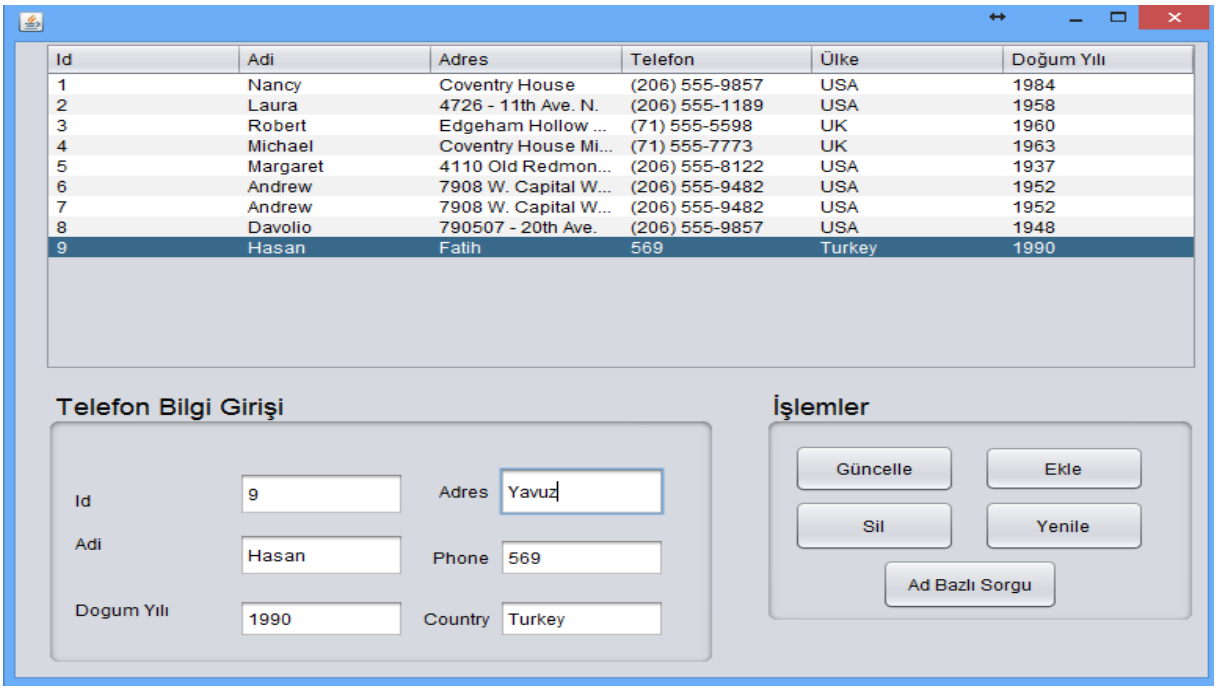
Şekil 10.'da gösterilen şekilde muhammed isimli girişler bulunmaktadır. Veri tabanında muhammed isimli kayıtlar gibi düzenli ve düzensiz bir şekilde giriş yapılan tüm kayıtlar

getirilmiştir. Şekil 11. bu girişleri silme işlemi için otomatik olarak oluşturulan sorgu sonucundaki silme işlemini göstermektedir. Otomatik olarak Hibernate teknolojisi tarafından üretilen sorgu Hibernate: delete from kisi.dbo.keywords where id=? şeklinde konsola yazdırılmıştır.



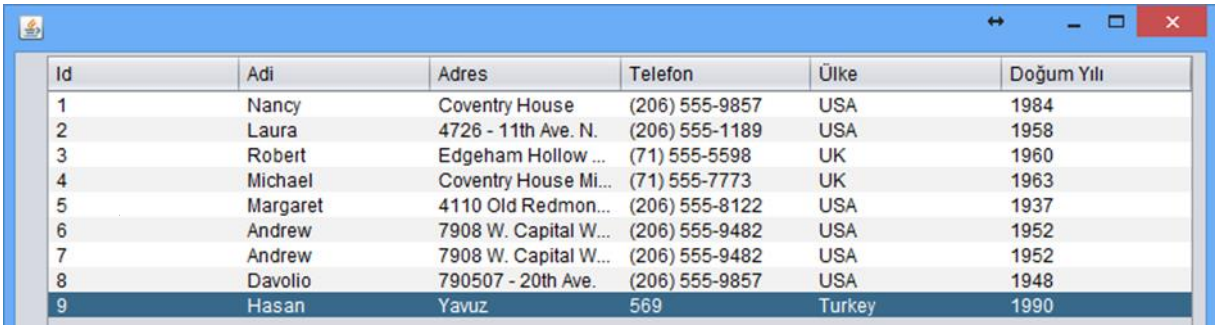
Şekil 11. *Veri tabanındaki bazı kayıtların silinmesinin gerçekleştirilmesi*

Son olarak veri tabanındaki name sütununa göre sorgulama ve güncelleme işlemleri bulunmaktadır.



Şekil 12. *Veri tabanındaki bir kaydın güncelleştirilmesi için veri girişi*

Şekil 12. güncelleştirme işlemi için veri girişini göstermektedir. Otomatik olarak Hibernate teknolojisi tarafından üretilen sorgu Hibernate: update kisi.dbo.keywords set name=?, adress=?, phone=?, country=?, dogumYili=? where id=? şeklinde konsola yazdırılmıştır.



Id	Adi	Adres	Telefon	Ülke	Doğum Yılı
1	Nancy	Coventry House	(206) 555-9857	USA	1984
2	Laura	4726 - 11th Ave. N.	(206) 555-1189	USA	1958
3	Robert	Edgeham Hollow ...	(71) 555-5598	UK	1960
4	Michael	Coventry House Mi...	(71) 555-7773	UK	1963
5	Margaret	4110 Old Redmon...	(206) 555-8122	USA	1937
6	Andrew	7908 W. Capital W...	(206) 555-9482	USA	1952
7	Andrew	7908 W. Capital W...	(206) 555-9482	USA	1952
8	Davolio	790507 - 20th Ave.	(206) 555-9857	USA	1948
9	Hasan	Yavuz	569	Turkey	1990

Şekil 13. *Veri tabanındaki bir kaydın güncelleştirilmesinin gerçekleştirilmesi*

Şekil 13.'ten güncelleme işlemi sonucunda kaydın hem JTable üzerindeki değerinin, Şekil 14.'ten de veri tabanındaki değerinin değiştiği görülmektedir.

Results Messages

	id	name	adress	phone	country	dogumYili
1	1	Nancy	Coventry House	(206) 555-9857	USA	1984
2	2	Laura	4726 - 11th Ave. N.	(206) 555-1189	USA	1958
3	3	Robert	Edgeham Hollow Winc	(71) 555-5598	UK	1960
4	4	Michael	Coventry House Mine	(71) 555-7773	UK	1963
5	5	Margaret	4110 Old Redmond Rd	(206) 555-8122	USA	1937
6	6	Andrew	7908 W. Capital Way	(206) 555-9482	USA	1952
7	7	Andrew	7908 W. Capital Way	(206) 555-9482	USA	1952
8	8	Davolio	790507 - 20th Ave.	(206) 555-9857	USA	1948
9	9	Hasan	Yavuz	569	Turkey	1990

Şekil 14. *Veri tabanı güncelleştirme işlemi sonucundaki durumunun gösterimi*

5. SONUÇ

Geleneksel ilişkili veri tabanı üzerinde geleneksel yöntemler kullanılarak CRUD işlemleri gerçekleştirilebilmektedir. Klasik yöntemleri kullanmanın yerine nesneye yönelik bir programlama mantığını kullanan bir teknoloji kullanılmıştır. Bu teknolojide tabloların sütunlarının dinamik olarak POJO sınıfları, ayar dosyaları, eşleşme dosyalarının oluşturulduğu görülmüştür. Aynı zamanda, SQL/JDBC bilmeden HQL kod yapısı ile çok az kod bilgisi ile tüm sorgulamalar gerçekleştirilebilmektedir. En önemli artısı bu işlemlerin tablo adı ile aynı isimde oluşan sınıflar aracılığıyla gerçekleştirilmesi en önemli noktalardan biridir. Hibernate teknolojisinin performansı incelenerek ORM araçlarından proje geliştirmede çok büyük kolaylıklar getirdiği belirlenmiştir.

KAYNAKLAR

Bauer, C., & King, G. (2005). Hibernate in action. Erişim Tarihi: 10.11.2015. https://www.cpe.ku.ac.th/~plw/oop/e_book/hibernate_in_action.pdf

Cao, Y., Yang, Y., Wang, H., & Zeng, P. (2009, June). Development of Hospital Medicine Storage Information System Based on B/S Architecture. In Web Mining and Web-based Application, 2009. WMWA'09. Second Pacific-Asia Conference on (pp. 271-274). IEEE.

Kasapbaşı, M.C. (2010). Nesneye Dayalı Programlarla Nesne İlişki Haritalanması. Akademik Bilişim'10 - XII. Akademik Bilişim Konferansı Bildiriler, 10-12 Şubat 2010.

Minter, D., & Linwood, J. (2006). Beginning Hibernate: From Novice to Professional. Dreamtech Press.

Xue, M., & Zhu, C. (2009, May). Design and implementation of the hibernate persistence layer data report system based on j2ee. In Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on (pp. 232-235). IEEE.

Cao, Y., Yang, Y., Wang, H., & Zeng, P. (2009, June). Development of Hospital Medicine Storage Information System Based on B/S Architecture. In Web Mining and Web-based Application, 2009. WMWA'09. Second Pacific-Asia Conference on (pp. 271-274). IEEE.

Kasapbaşı, M.C. (2010). Nesneye Dayalı Programlarla Nesne İlişki Haritalanması. Akademik Bilişim'10 - XII. Akademik Bilişim Konferansı Bildiriler, 10-12 Şubat 2010.

Minter, D., & Linwood, J. (2006). Beginning Hibernate: From Novice to Professional. Dreamtech Press.

Xue, M., & Zhu, C. (2009, May). Design and implementation of the hibernate persistence layer data report system based on j2ee. In Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on (pp. 232-235). IEEE.