T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

JAVA PROGRAMLAMA İLE JDBC, HİBERNATE, NOSQL 'İ KULLANARAK FARKLILIKLARIN ORTAYA ÇIKARILMASI VE UYGULAMALAR İLE GÖSTERİLMESİ

Yüksek Lisans Tezi

Tezi Hazırlayan: **Ferhat Cem CİHAN**

İSTANBUL, 2016

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

JAVA PROGRAMLAMA İLE JDBC, HİBERNATE, NOSQL 'İ KULLANARAK FARKLILIKLARIN ORTAYA ÇIKARILMASI VE UYGULAMALAR İLE GÖSTERİLMESİ

Yüksek Lisans Tezi

Tezi Hazırlayan:

Ferhat Cem CİHAN

Öğrenci No:

140820001

Danışman:

Yrd. Doç. Dr. Ediz ŞAYKOL

İSTANBUL, 2016

YEMİN METNİ

Yüksek lisans tezi olarak sunduğum "**Java Programlama ile Jdbc, Hibernate, NoSql'i kullanarak farklılıkların ortaya çıkarılması ve uygulamalar ile gösterilmesi**" başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiği ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım. 10.04.2016

Ferhat Cem Cihan

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ SAVUNMA SINAVI SONUÇ TUTANAĞI

Beykent Üniversitesi
Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Aşağıda tez adı belirtilen yüksek lisans öğrencisi.140820001.....no'lu FERHAT CEM CİHAN'ın .../.../... tarihinde yapılan tez savunma sınavı[1] sonucunda 45 dakika süreyle sunduğu ve savunduğu tezi hakkında[2] oybirliğiyle, ...kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Programı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Başlığı[3] : Java Programlama İle JDBC, HIBERNATE, NOSQL'i Kullanarak Farklılıkların Ortaya Çıkarılması Ve Uygulamalar İle Gösterilmesi

---

Tez Sınav Jürisi	**Öğretim Üyesi**	**İmza**
Danışman	**: YRD.DOÇ.DR. EDİZ ŞAYKOL**	
Üye	**: YRD.DOÇ.DR. EGE KİPMAN**	
Üye	**: DOÇ.DR. GÖKHAN SİLAHTAROĞLU**	

---

[1] Jüri üyeleri söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez savunma sınavına alır. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda jüri en geç onbeş gün içinde toplanarak adayı tez savunma sınavına alır. Tez savunma sınav süresi en az 45 dakikadır. Yüksek lisans tez savunma sınavı, tez çalışmasının sunulması ve bunu izleyen soru-yanıt bölümlerinden oluşur ve dinleyiciye açıktır. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-3)

[2]Tez sınavının tamamlanmasından sonra jüri, tez hakkında "kabul", "düzeltme" veya "red" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış sınav tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi başarısız bulunan öğrencinin Enstitü ile ilişiği kesilir. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve yönetmelikte belirtilen usullere uygun olarak tezini aynı jüri önünde yeniden savunur. Bu savunma sınavında da tezi kabul edilmeyen öğrencinin enstitü ile ilişiği kesilir.(Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-4)

[3]İleridedoğabilecekaksaklıklarınengellenmesiiçintezinbaşlığınınyazılmasıgerekmektedir.

TEŞEKKÜR

Tez çalışmamda bana danışmanlık yapan ve yönlendirmelerde bulunan **Yrd. Doç. Dr. Ediz ŞAYKOL** hocama ve tez çalışmalarım süresince bana sabırla destek olan aileme çok teşekkür ederim.

JAVA PROGRAMLAMA İLE JDBC, HİBERNATE, NOSQL 'İ KULLANARAK FARKLILIKLARIN ORTAYA ÇIKARILMASI VE UYGULAMALAR İLE GÖSTERİLMESİ

Tezi Hazırlayan: **Ferhat Cem Cihan**

ÖZET

Günümüzde nesne yönelimli programlama dillerinden biri olan Java her geçen gün çeşitli araçlar ile kullanılmaya başlanmıştır. Veri tabanı işlemleri için Java programlama dili ile beraber kullanılan birçok araç mevcuttur. Bu araçlar, veri tabanının hali hazırdaki program ile haberleşmesi, veri tabanı tablolarının çeşitli işlemler sonucunda programa modellenebilmesi, veri tabanı tablolarına verilerin kayıt edilebilmesi, veri silebilmesi, veri güncellemesi, veri bulunabilmesi, verileri ilişkili bir hale getirebilmesi işlemlerini ve var olan veri tabanı tablolarında değişiklik yapma bilme gibi imkanları bize sunar.

Birçok araç bunu yapabilirken kullanılan programlama dili ve seçilen araç önem taşımaktadır. Yanlış bir seçim sonucu var olan işler beklenenden daha uzun sürebilmekte olduğu için zaman ve maliyet kaybına yol açmaktadır. Seçimin çeşitli durumlarda yazılım geliştirici açışından avantaj ve dezavantajları ortaya çıkar.

Bu tez Java programlama dili ile Jdbc, Hibernate, NoSql geliştirmeleri yapılırken karşılaşılan farklılıkları ve benzerlikleri uygulamalar ile inceler ve bunların yazılım geliştiriciler açısından avantaj ve dezavantajlarını ortaya çıkarır.

Anahtar Kelimeler: Java, Jdbc, Hibernate, NoSql Farklılıkları, Uygulamaları, Yazılım Geliştiriciler Açısından Avantaj ve Dezavantajları.

REVEALING DIFFERENCES USING JDBC, HIBERNATE, NOSQL THROUGH JAVA PROGRAMMING AND INVESTIGATING THOSE BY MEANS OF PRACTICES

Presented By: **Ferhat Cem Cihan**

ABSTRACT

Java which is one of the object-oriented programming language at the present time, has begun to be used with various tools day by day. Many tools are available, being used together with Java programming language for data base operations. These tools offer us operations such as transmission of data base with available program, adapting of data base tables to program as a result of various operations, recording of data to data base tables, data delete, data update, data availability, making data associated to each other, and opportunities such as making changes in existing database tables.

As well as many tools have ability to accomplish forementioned issues, programming language used and tool selected are important. Since the works, yielded through materials selected wrongly, could last longer than expected, they lead to a loss of time and cost as well. The advantages and disadvantages of selection with regard to software developers will arise with various ocassions.

This thesis investigates the differences and similarities arose while developing of Jdbc, Hibernate, NoSql with Java programming language and reveals advantages and disadvantages of them in terms of software developers.

Key Words: Java, Jdbc, Hibernate, NoSql Differences, Practices, Their Advantages and Disadvantages in terms of Software Developers.

İÇİNDEKİLER

KISALTMALAR

IDE: Integrated Development Environment

JAR: Java Archive

JDBC: Java Database Connectivity

JVM: Java Virtual Machine

HQL: Hibernate Query Language

ORM: Object Relational Mapping

JSF: Java Server Faces

JSP: Java Server Pages

XML: Extensible Markup Language

JSON: JavaScript Object Notation

BSON: Binary Structured Object Notation

JDK: Java Development Kit

SDK: Software Development Kit

SQL: Structured Query Language

OOP: Object Oriented Programming

CMD: Command Prompt

JRE: Java Runtime Environment

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1 GİRİŞ

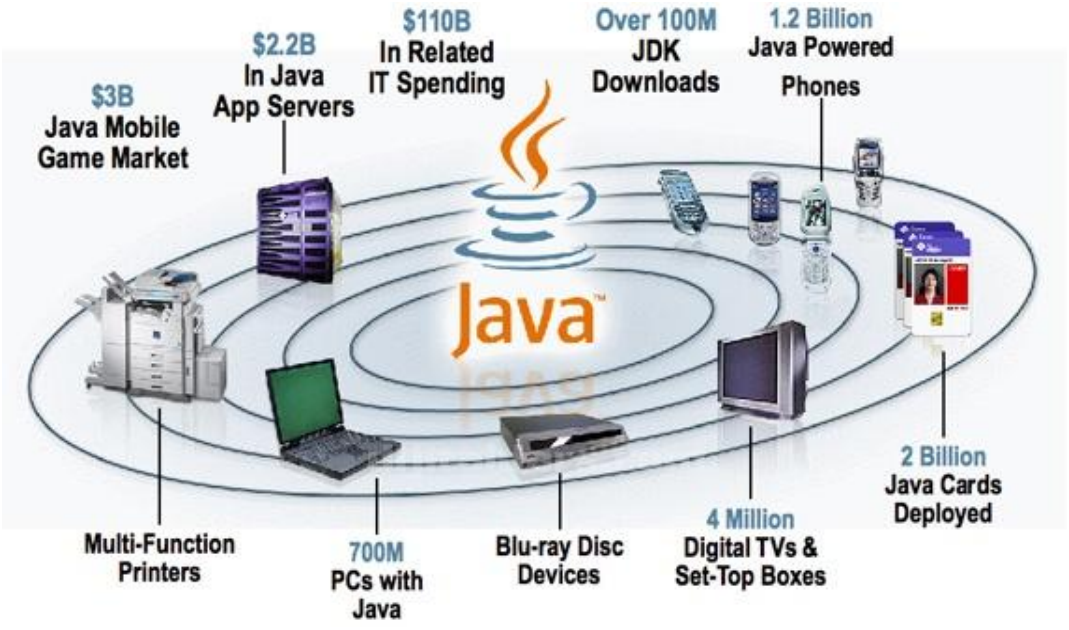
Java programlama dili günümüzün en yaygın ve esnek programlama dillerinden biridir. Java, Sun Microsysterms tarafından geliştirilmeye başlanmış olup, yüksek performanslı, çok işlevli, yüksek seviye adım adım işletilen bir dildir.[1] Javanın ilk sürümü 1995'te çıkmıştır.

Platform bağımsız çalışması, açık kaynak kodlu olması ve nesne yönelimli olması sonucunda kullanım alanları oldukça fazladır. [2]

Kişisel bilgisayarlar insanların yaşamında derin bir etki bırakmıştır ve işlerini yapmada kolaylık sağlamıştır.[3] Fakat sadece kişisel bilgisayarlar değil bazı elektronik cihazlarda insanların yaşamını kolaylaştırmaktadır.

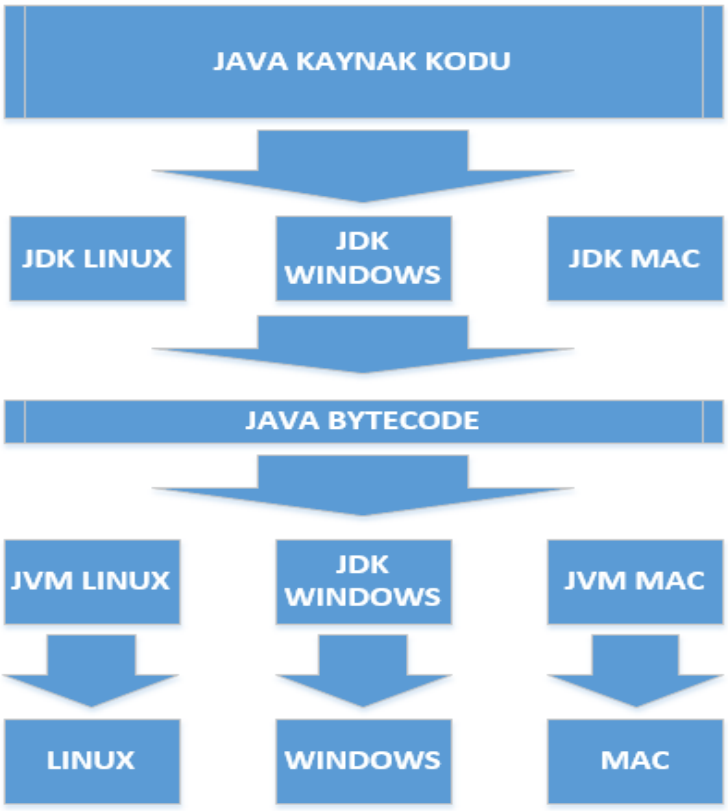
Java dili bilgisayar ağları üzerindeki farklı bilgisayar platformlarında kullanılacak yazılımları geliştirmek amacıyla da geliştirilmiştir. [1]

Cep telefonlarındaki uygulamalarda, cep telefonlarında, bilgisayarlarda, yazıcılarda, dijital televizyonlarda kullanılmaktadır. Ayrıca yeni üretilen programlanabilir birçok cihazda yazılım dili olarak Java programlama dilinin tercih edildiğini aşağıdaki Şekil 1.'de görmekteyiz.[5]



Şekil 1. Java Programlama Dilinin Kullanıldığı Cihazlar

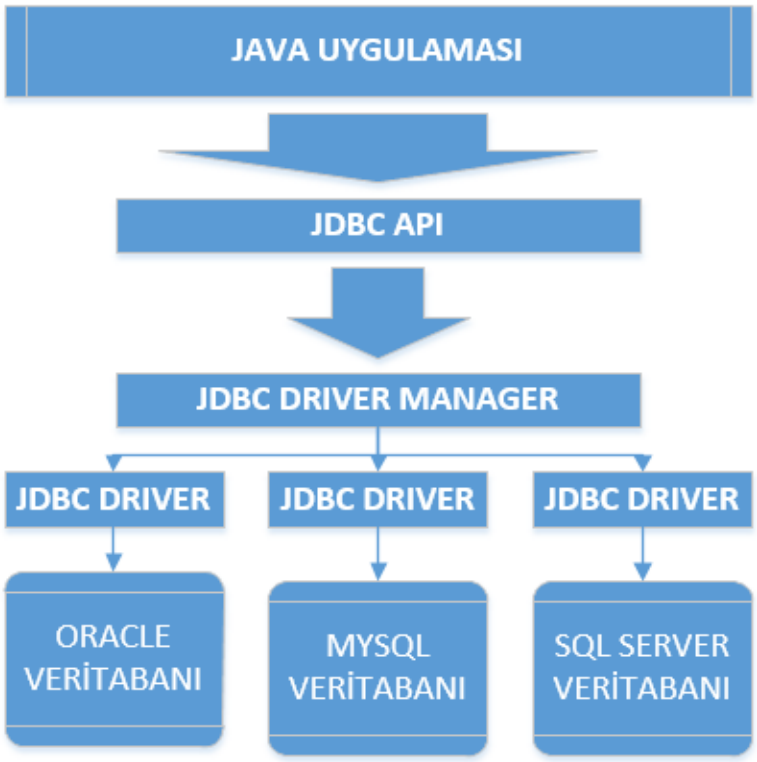
Java programlama dilinin en temel özelliklerinden biri de "Bir kere Java kodu yaz her yerde kullan" felsefesidir. İşletim sistemi göz etmeksizin herhangi bir sistem (framework) kullanmadan not defteri (notepad) üzerine yazılan Java kodu derlenerek de Şekil 2'de gösterildiği gibi çalışabilmektedir.



Şekil 2. Java Programlama Dili Çalışma Mantığı

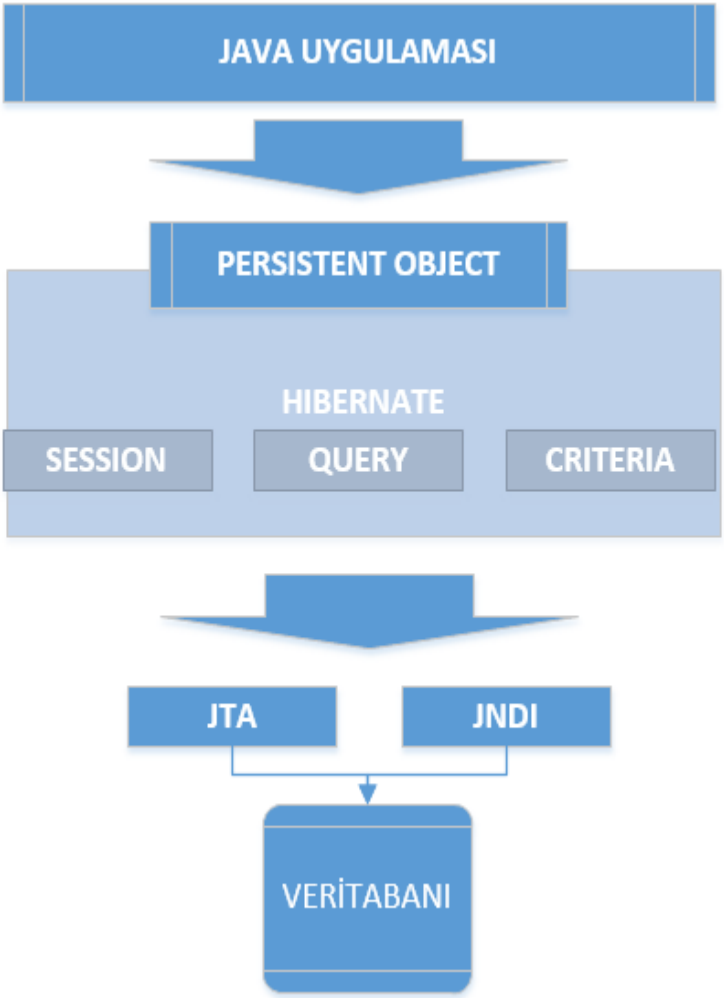
Java dilinin birçok teknoloji ile uyum içinde çalıştığını söyleyebiliriz. Ayrıca yazılım geliştiriciler açışından oluşturulan yazılım projelerinin esnek ve dinamik bir yapıda olması geliştiricilerin işini kolaylaştıran Java ile birlikte kullanılan araçlar vardır. En önemlilerine bakacak olursak, yazılan kod kalitesini analiz eden araçlar (SonarQube, Fisheye), Java sanal makinesi ayarlama (Eclipse Memory Analizer, JProfiler, VisualVM) ve izleme, performans (Tuning & Monitoring) araçları, test araçları (Selenium, JUnit), Java derleme (Builds, Maven,Ant) araçları, Java IDE'leri (Eclipse, NetBeans, IntelliJ), Java web programlama araç ve dilleri (JSP, JSF) ve Java da veri tabanı işlemleri yapmak için kullanılan çeşitli araç ve diller (Jdbc, Hibernate, NoSql) mevcuttur.

Veri tabanı işlemleri için Jdbc kullanılmak istenildiğinde Java ile veri tabanı arasında bir katman oluşturmak gerekmektedir. Bu katman yardımıyla veri tabanına bağlanılarak Java kodu üzerinde yazılan Sql Scriptleri(herhangi bir programlama dilinde yazılmış kod parçası) çalıştırılarak işlemler yapılabilir.[6] Bu katmanı oluşturmak ve kullanmak için her veri tabanına özgü olan Jar kütüphaneleri sağlanarak projelere eklemek yeterli olacaktır. Jdbc genel yapısı Şekil 3.'de gösterilmiştir.



Şekil 3. Jdbc ile Java Uygulaması Ve Veri Tabanı Bağlantısı Katmanları Genel Yapısı

Hibernate ise bir ORM aracı olmasının yanında yine veri tabanı işlemlerimde de kullanılır. Veri sorgulama, veri çekme işlemlerini sağladığı gibi Java kodu üzerinden veri tabanına tablolar oluşturma, tablolar arasında ilişkiler oluşturma işlemlerini de yapabilir. Ayrıca Hibernate' in bir sorgu dili olan HQL ile birçok işlem imkanını yazılım geliştiricilere sunmaktadır. Hibernate genel yapısı Şekil 4.'de gösterilmiştir.



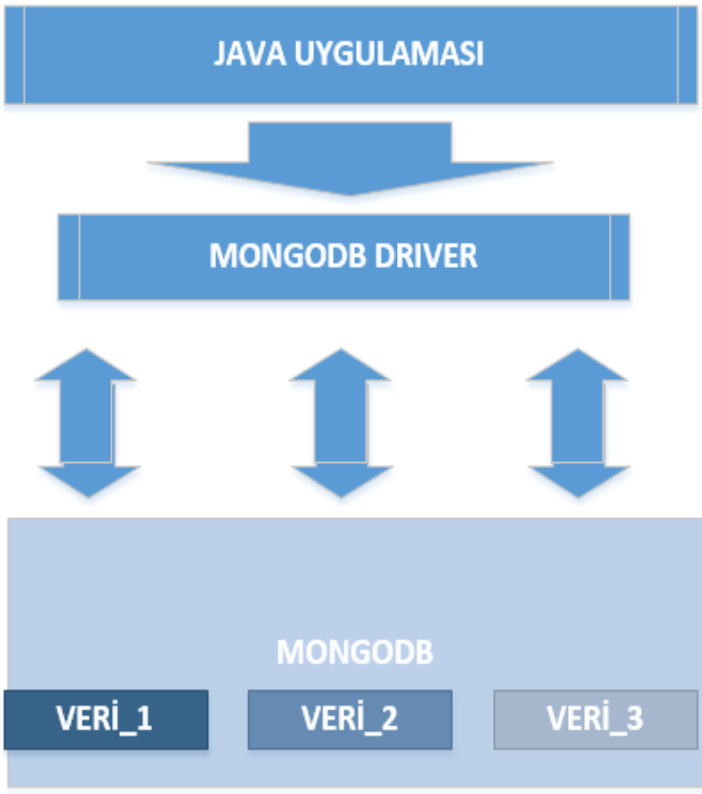
Şekil 4. Hibernate İle Java Uygulaması Ve Veri Tabanı Bağlantısı Katmanları Genel Yapısı

NoSql ise, ilişkili olmayan (No Relation) veya yalnızca Sql olmayan(Not Only Sql) anlamlarından yola çıkarak ortaya çıkarılan, yazılım geliştiricilerin Sql veya sorgu (query) ve ilişkisel veri tabanı mantıklarını geri plana koymaları sağlamaya çalışan bir çözüm olarak ortaya çıkarılmıştır. Dikey büyümeden ziyade yatay büyümeyi benimsemiştir. Birçok büyük internet şirketi (Google, Amazon, Facebook) verilerini klasik veri tabanlarında tutmak yerine NoSql sistemlere sahip olan veri tabanı yapısına taşımaya başlamışlardır.[7]

Bu tezde NoSql işlemleri belge yönelimli veri tabanı olan MongoDb kullanılmıştır. MongoDb verileri satır satır saklayıp daha sonrasında ilişkili olduğu

tablolara saklamak yerine belge (document) şeklinde XML, JSON, BSON formatlarında saklanırlar. [7]

Java uygulaması ile MongoDb arasındaki katman kullanılmış olan MongoDb veri tabanına özgü olan Jar ile sağlanmaktadır. Aradaki katman oluşturulduktan sonra Java uygulama kodları yardımıyla ilgili veri tabanında veri tabanı işlemleri Şekil 5.'de gösterildiği gibi yapılabilir bir hale getirilebilir.



Şekil 5. MongoDb İle Java Uygulaması Ve Veri Tabanı Bağlantısı Katmanları Genel Yapısı

Üç farklı kullanımın sahip olduğu avantajlar ve dezavantajlar vardır. Farklı durumlar için her biri farklı davranış sergileyebilir. Kullanılacak araç ve dil bu yüzden önem arz etmektedir. Oluşturulacak uygulamanın performans kriteri ve bu uygulamayı geliştirecek yazılım geliştiricilere sunmuş olduğu avantaj ve dezavantajlar göz ardı edilmemelidir. Daha önce yapılan çalışmalar değerlendirildiğinde detaylı olarak bu farktan bahsedilmediği görülmüştür. Ayrıca yazılım geliştiriciler açışından avantaj ve dezavantajları ele alınmamıştır.

Tez kapsamında bu araçların nasıl kullanıldığı, ihtiyaç duyulan Jar kütüphaneleri Java projelerine eklenerek geliştirmeler yapılmıştır. Oluşturulan veri tabanı ile bağlantı kurularak performans çalışmaları karşılaştırılmıştır. İhtiyaç duyulan ilişkisel bağlantıların nasıl oluşturulduğu veya veri tabanından bağımsız olarak nasıl oluşturulabileceği, birbirleri aralarındaki farklar kıyaslanarak uygulamalar ile tespit edilerek gösterilmiştir.

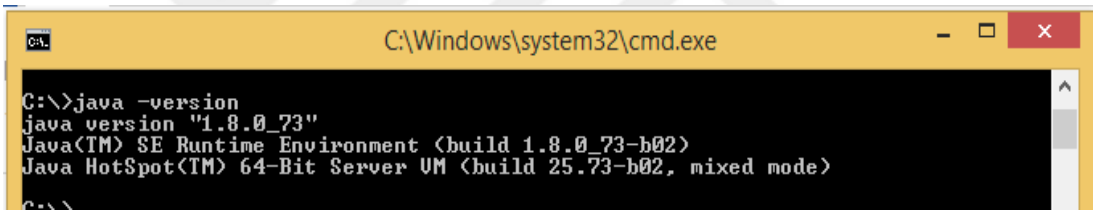
Ayrıca yazılım geliştiriciler açışından kullanılan Jdbc, Hibernate ve NoSql çeşitli kriterler açısından avantaj ve dezavantajları değerlendirilerek hangi durumda tercihin ne şekilde yapılması gerektiği daha uygun olabilir şeklindeki sorulara cevap bulunmuştur.

2 GELİŞTİRİLEN ÇALIŞMALAR

2.1 Jdbc, Hibernate Ve NoSql Uygulamaları İçin Ön Hazırlık

Java programlama dili kullanılarak oluşturulan Jdbc çalışmaları yapılabilmesi için veri tabanına bağlı bir sistem ve önceden oluşturulmuş bir veri tabanı olması gerekmektedir. Bir alt bölümde anlatılan ve tasarlanan veri tabanı şeması ve tabloları, kodu çalıştıracağınız bilgisayarda Java SDK nin yüklü olması, Java kodlarımızı tasarlayıp, geliştirebileceğiz ve derleyebileceğimiz bir IDE ve MySql Connector Java, MongoDb Connector Java ve Hibernate Jar kütüphanelerini sağladıktan sonra uygulamalar geliştirilebilir.

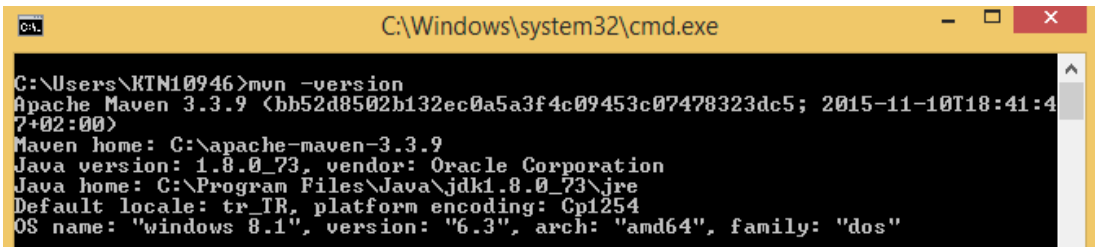
Kodu çalıştıracağınız bilgisayarda Javanın yüklü, çalışabilir ve hangi versiyonunun kullanıldığı öğrenmek için CMD üzerinden "java –version" komutunu yazarak Şekil 6.'da gösterildiği gibi öncelikle kontrol etmek gerekmektedir.



Şekil 6. Java Kurulum ve Versiyon Bilgisi Gösterimi

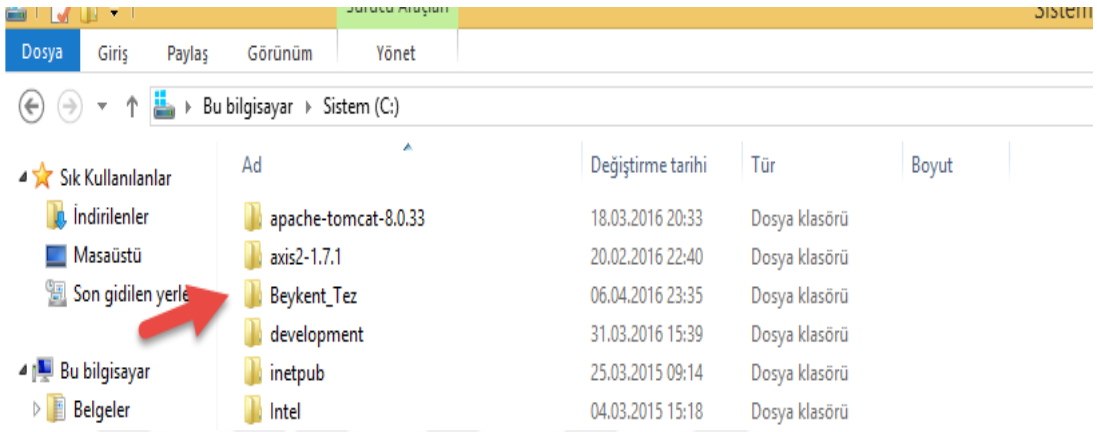
NoSql ve Hibernate uygulama çalışmalarında Java projelerinde karmaşıklığı azaltmak ve belirli bir standarda çekebilmek, kütüphane bağımlılıklarını ve kontrolünü kullanıcıdan kurtarmak için Maven (Java Deployment Tool) aracı kullanılmıştır.[8]

Bu aracı ücretsiz olarak kişisel bilgisayarınızda indirdiğinizde ve yüklediğinizde Şekil 7.'de gösterildiği üzere CMD üzerinden "mvn -version" komutunu yazarak kullanılabilirliğini test edebilirsiniz.

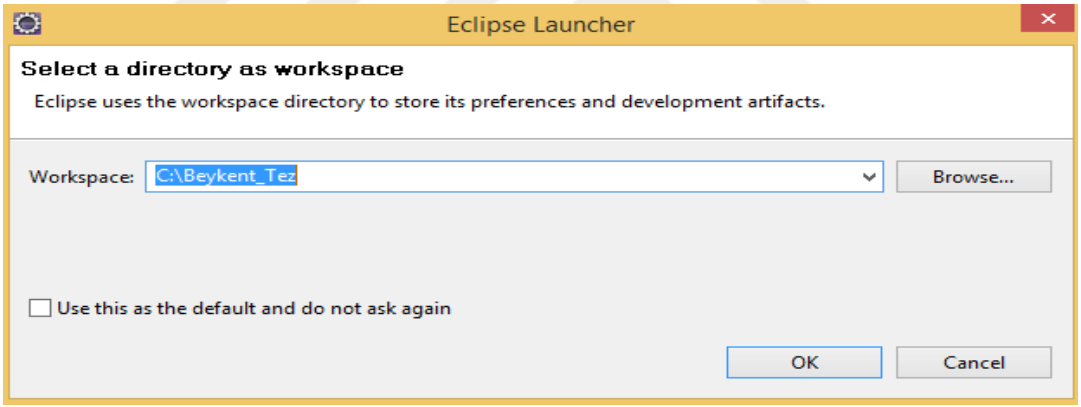


Şekil 7. Maven Kurulum ve Versiyon Bilgisi Gösterimi

Java çalışma ortamı olarak Eclipse IDE kullanılmıştır. Kişisel bilgisayarımda Şekil 8.'de gösterildiği gibi C:\ dizinini altına "Beykent_Tez" isimli bir klasör oluşturup, Şekil 9.'da ise Eclipse IDE de çalışma ortamı (workspace) olarak seçilmesi gösterilmiştir.



Şekil 8. Kişisel Bilgisayarda Çalışma Ortamı Oluşturulması



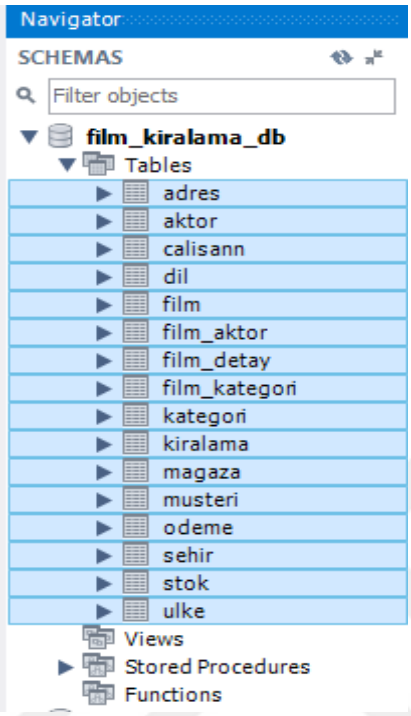
Şekil 9. Eclipse IDE Çalışma Ortamı Seçilmesi

2.2 Kullanılacak Veri Tabanı Tasarımı Çalışması

Jdbc uygulaması için gerekli olan veri tabanı tasarımı MySql veri tabanı yönetim sistemi kullanılarak tasarlanmıştır. Veri tabanı şemasının ismi "film_kiralama_db" olarak oluşturulmuştur. Oluşturulan şemanın altında 16 tablo mevcuttur. Bunlar; Aktör Tablosu, Adres Tablosu, Kategori Tablosu, Şehir Tablosu, Ülke Tablosu, Müşteri Tablosu, Film Tablosu, Film Aktör Tablosu, Film Kategori

Tablosu, Film Detay Tablosu, Stok Tablosu, Dil Tablosu, Ödeme Tablosu, Kiralama Tablosu, Çalışan Tablosu, Mağaza Tablosu.

Oluşturulan veri tabanı şemasının genel görüntüsü Şekil 10.'daki gibidir.



Şekil 10. Veri Tabanı Şemasının Genel Görüntüsü

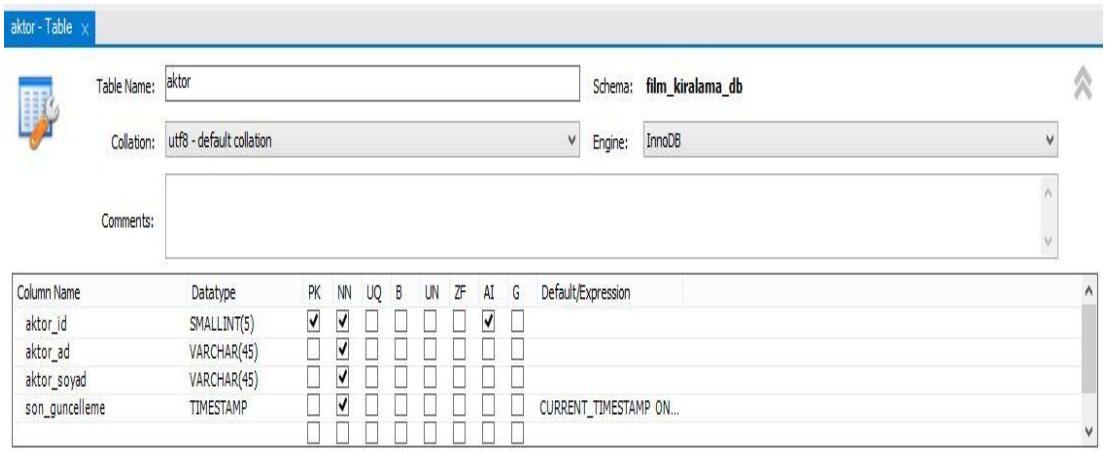
Uygulamaların çalışabilmesi için aşağıdaki veri tabanı modeli ve ilişkileri oluşturulmuştur. Oluşturulan veri tabanı aşağıdaki tabloları içermektedir.

Aktör Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
AKTOR_ID	SmallInt(5)	PrimaryKey, Auto Increment	Hayır
AD	Varchar(45)		Hayır
SOYAD	Varchar(45)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 1. Aktör Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 11.'de gösterilmiştir.



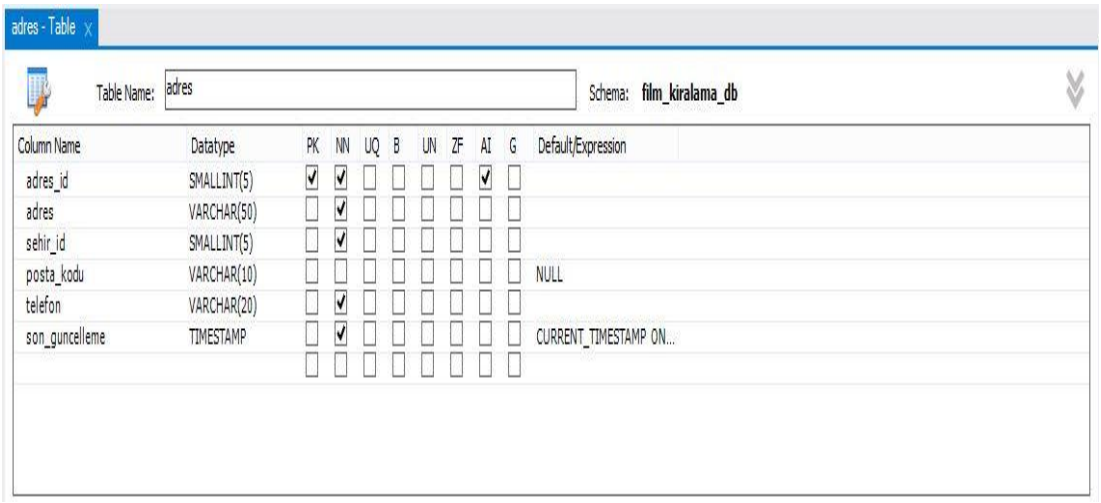
Şekil 11. Aktör Veri Tabanı Tablosu Görüntüsü

Adres Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
ADRES_ID	SmallInt(5)	Primary Key, Auto Increment	Hayır
ADRES	Varchar(50)		Hayır
SEHIR_ID	SmallInt(5)		Hayır
POSTA_KODU	Varchar(10)		Evet
TELEFON	Varchar(20)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 2. Adres Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 12.'de gösterilmiştir.



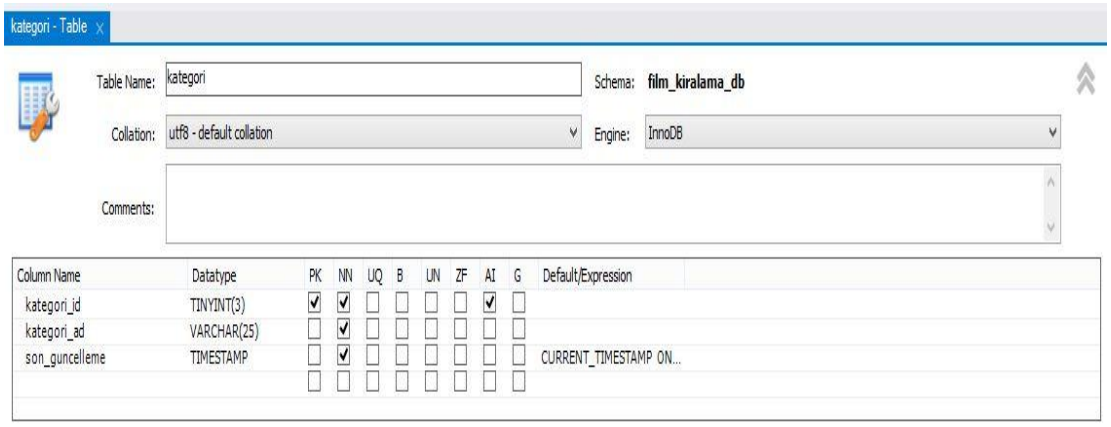
Şekil 12. Adres Veri Tabanı Tablosu Görüntüsü

Kategori Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
KATEGORI_ID	TinyInt(3)	Primary Key, Auto Increment	Hayır
KATEGORI_AD	Varchar(20)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 3. Kategori Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 13.'de gösterilmiştir.



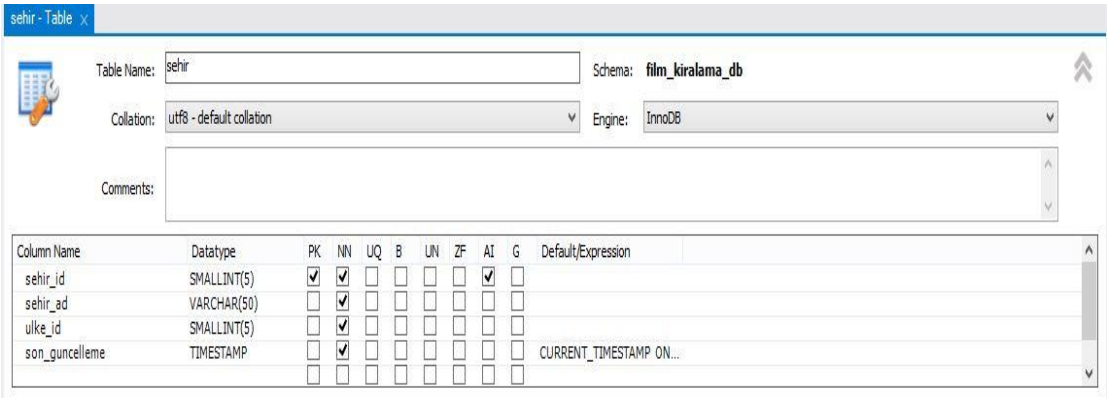
Şekil 13. Kategori Veri Tabanı Tablosu Görüntüsü

Şehir Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
SEHIR_ID	SmallInt(5)	Primary Key, Auto Increment	Hayır
SEHIR_AD	Varchar(50)		Hayır
ULKE_ID	SmallInt(5)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 4. Şehir Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 14.'de gösterilmiştir.



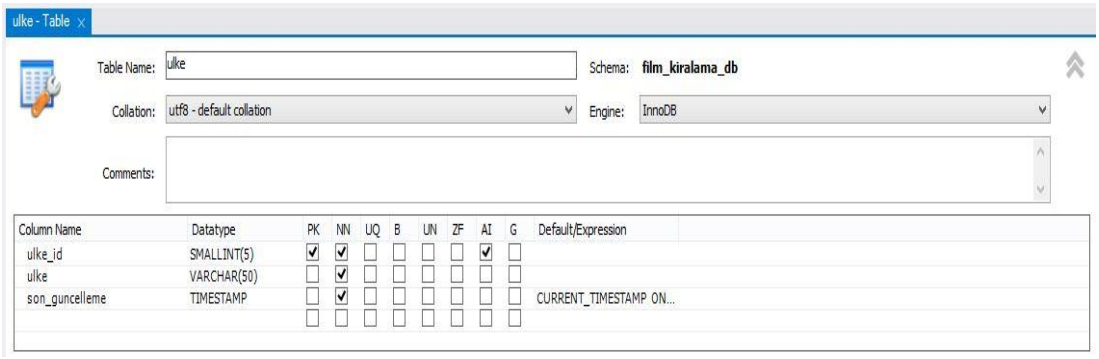
Şekil 14. Şehir Veri Tabanı Tablosu Görüntüsü

Ülke Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
ULKE_ID	SmallInt(5)	Primary Key, Auto Increment	Hayır
ULKE_AD	Varchar(50)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 5. Ülke Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 15.'de gösterilmiştir.



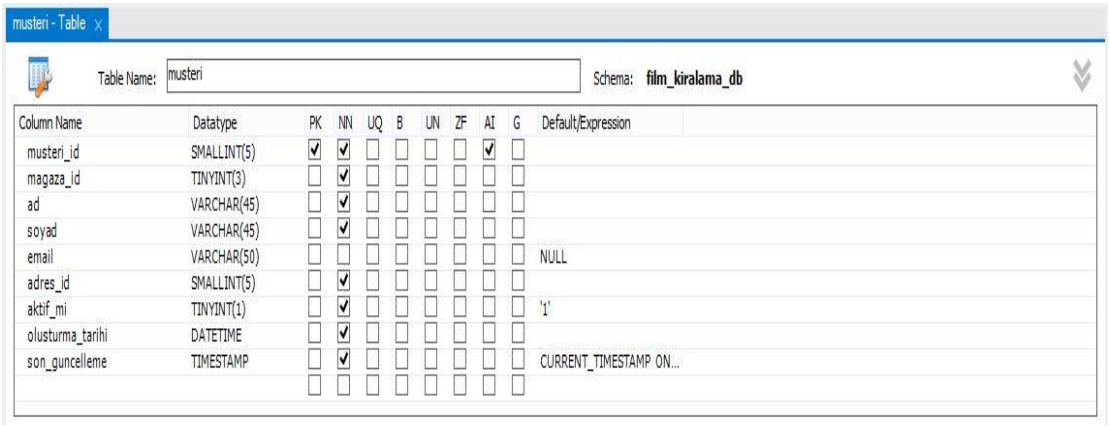
Şekil 15. Ülke Veri Tabanı Tablosu Görüntüsü

Müşteri Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
MUSTERI_ID	SmallInt(5)	PrimaryKey, Auto Increment	Hayır
MAGAZA_ID	TinyInt(3)		Hayır
AD	Varchar(45)		Hayır
SOYAD	Varchar(45)		Hayır
EMAIL	Varchar(50)		Evet
ADRES_ID	SmallInt(5)		Hayır
AKTIF_MI	TinyInt(1)		Hayır
OLUSTURMA_TARIHI	DateTime		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 6. Müşteri Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 16.'de gösterilmiştir.



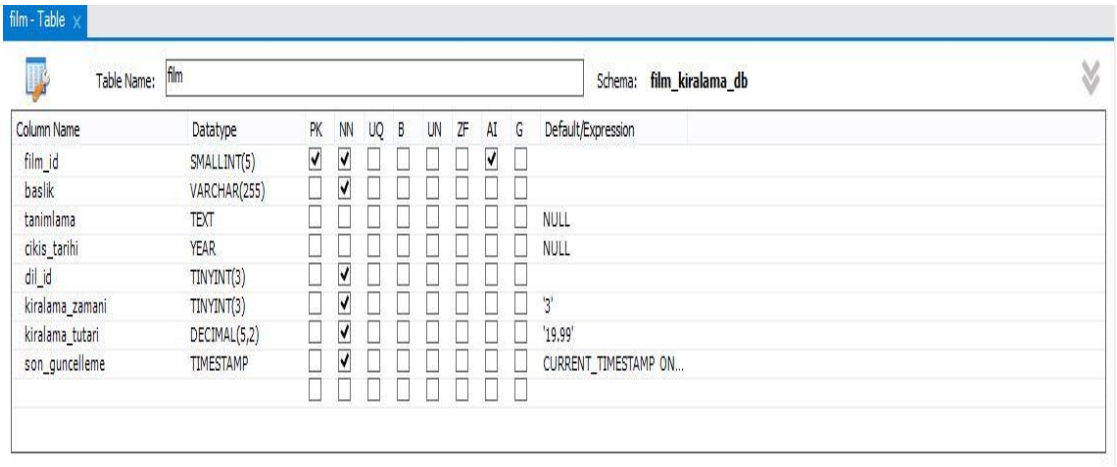
Şekil 16. Müşteri Veri Tabanı Tablosu Görüntüsü

Film Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
FILM_ID	SmallInt(5)	PrimaryKey, Auto Increment	Hayır
BASLIK	Varchar(255)		Hayır
TANIMLAMA	Text		Evet
CIKIS_TARIHI	Year		Evet
DIL_ID	TinyInt(3)		Hayır
KIRALAMA_ZAMANI	TinyInt(3)		Hayır
KIRALAMA_BEDELI	Decimal(5,2)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 7. Film Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 17.'de gösterilmiştir.



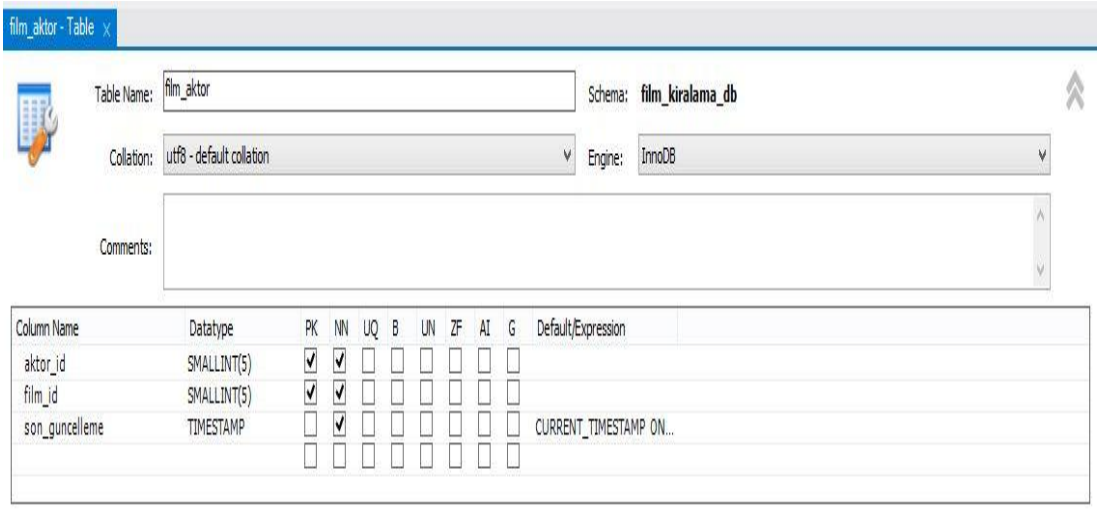
Şekil 17. Film Veri Tabanı Tablosu Görüntüsü

Film Aktör Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
ACTOR_ID	SmallInt(5)	Primary Key	Hayır
FILM_ID	SmallInt(5)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 8. Film Aktör Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 18.'de gösterilmiştir.



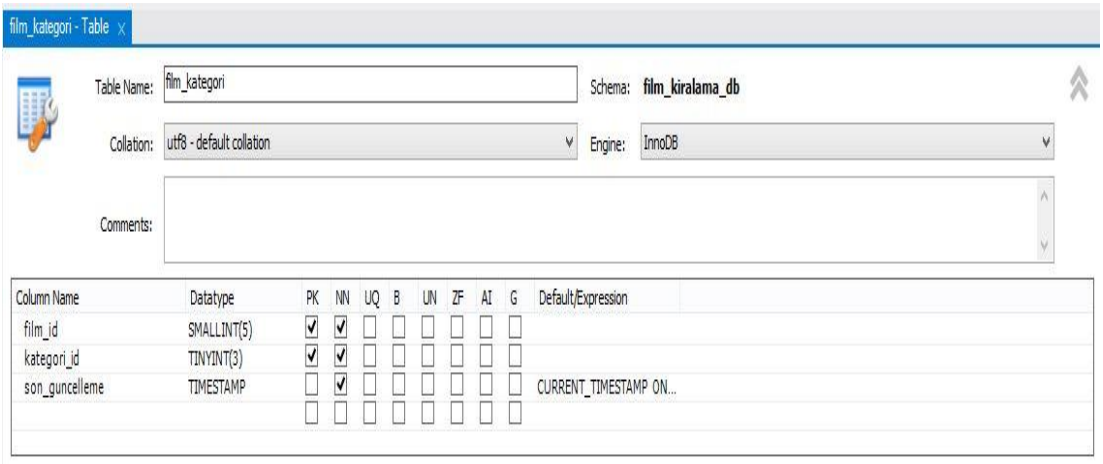
Şekil 18. Film Aktör Veri Tabanı Tablosu Görüntüsü

Film Kategori Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
FILM_ID	SmallInt(5)	Primary Key	Hayır
KATEGORI_ID	TinyInt(3)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 9. Film Kategori Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 19.'de gösterilmiştir.



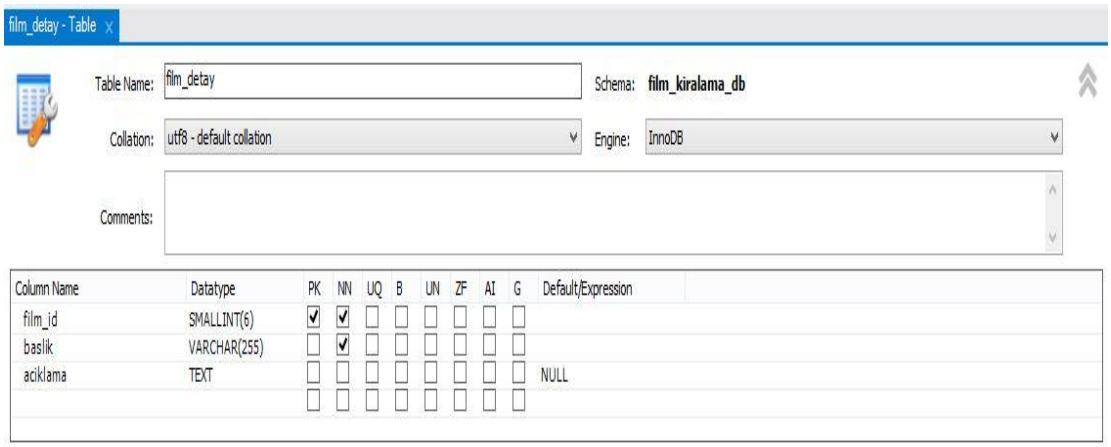
Şekil 19. Film Kategori Veri Tabanı Tablosu Görüntüsü

Film Detay Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
FILM_ID	SmallInt(5)	Primary Key, Auto Increment	Hayır
BASLIK	Varchar(255)		Hayır
ACIKLAMA	Text		Hayır

Tablo 10. Film Detay Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 20.'de gösterilmiştir.



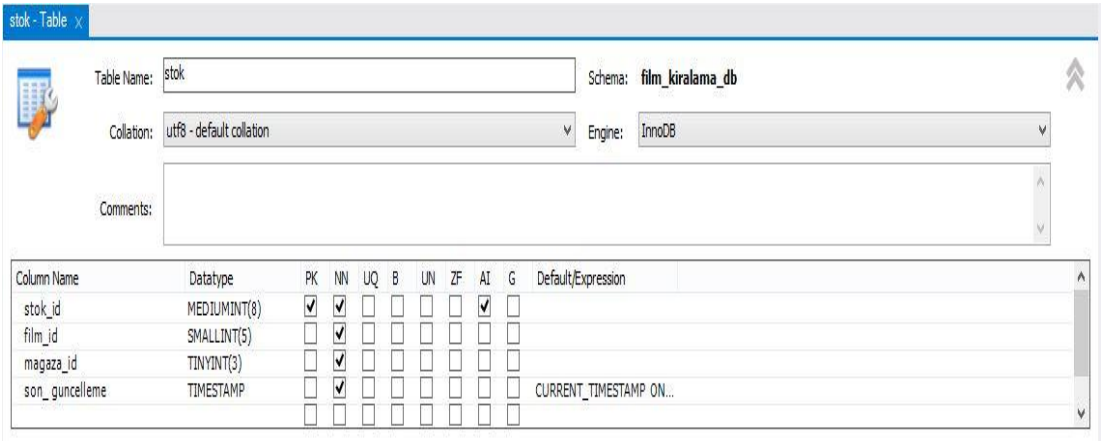
Şekil 20. Film Detay Veri Tabanı Tablosu Görüntüsü

Stok Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
STOK_ID	MediumInt(8)	PrimaryKey, Auto Increment	Hayır
FILM_ID	SmallInt(5)		Hayır
MAGAZA_ID	TinyInt(3)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 11. Stok Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 21.'de gösterilmiştir.



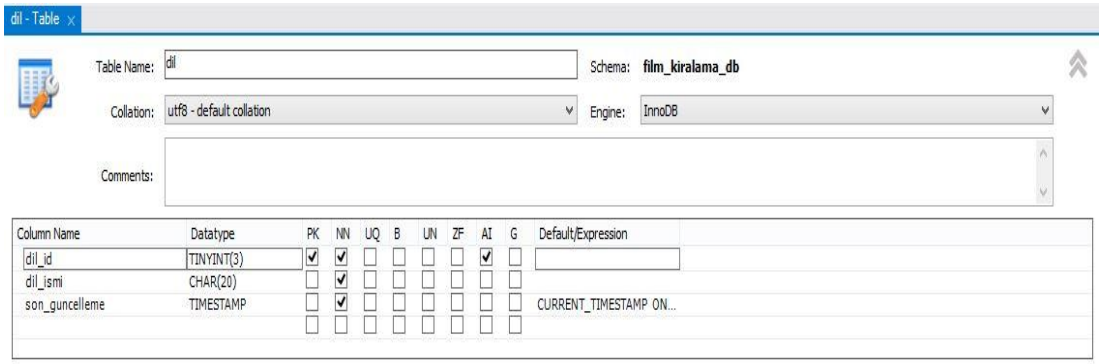
Şekil 21. Stok Veri Tabanı Tablosu Görüntüsü

Dil Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
DIL_ID	TinyInt(3)	Primary Key, Auto Increment	Hayır
DIL_ISMI	Char(20)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

Tablo 12. Dil Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 22.'de gösterilmiştir.



Şekil 22. Dil Veri Tabanı Tablosu Görüntüsü

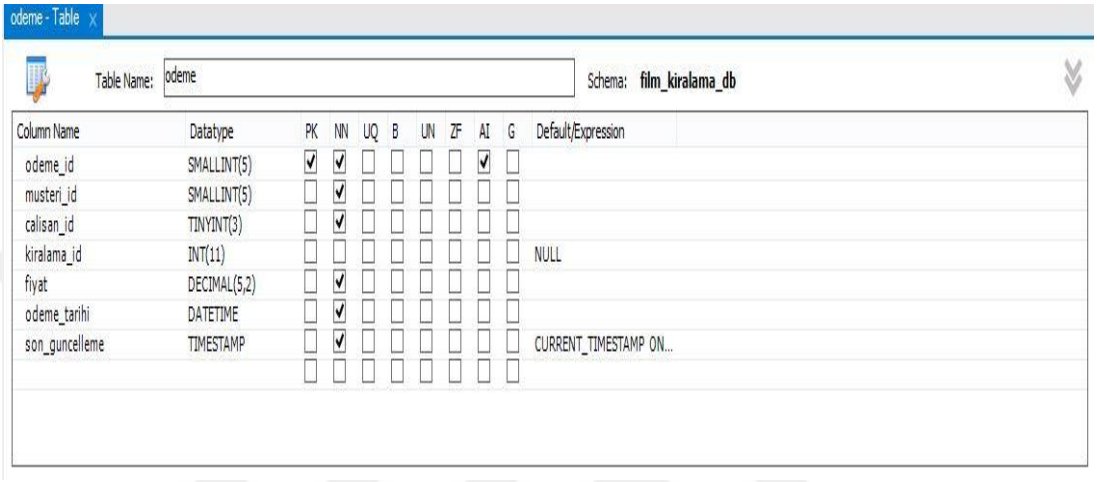
Ödeme Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
ODEME_ID	SmallInt(5)	Primary Key, Auto Increment	Hayır
MUSTERI_ID	SmallInt(5)		Hayır
CALISAN_ID	TinyInt(3)		Hayır
KIRALAMA_ID	Int(11)		Evet
FIYAT	Decimal(5,2)		Hayır
ODEME_TARIHI	DateTime		Hayır

SON_GUNCELLEME	TimeStamp		Hayır

Tablo 13. Ödeme Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 23.'de gösterilmiştir.



Şekil 23. Ödeme Veri Tabanı Tablosu Görüntüsü

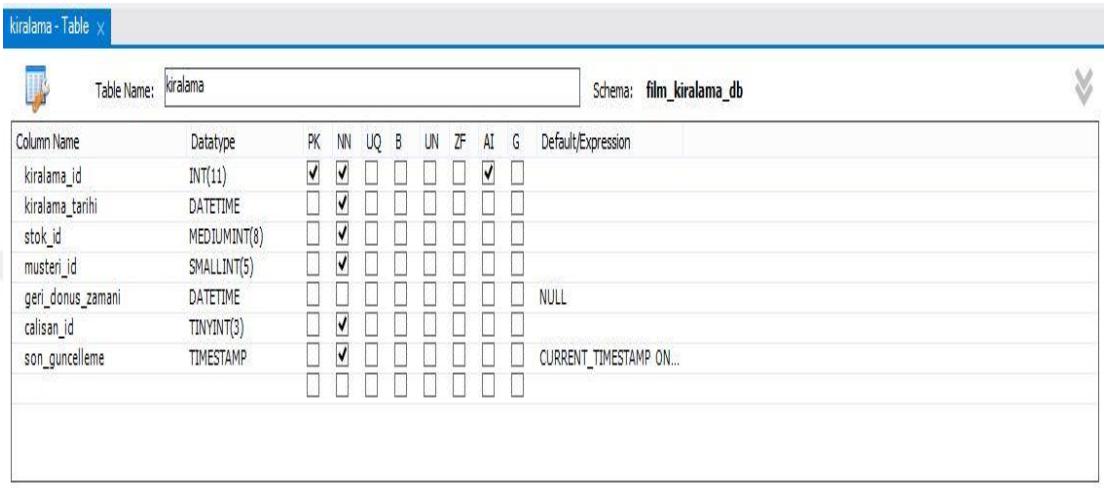
Kiralama Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
KIRALAMA_ID	Int(11)	PrimaryKey, Auto Increment	Hayır
KIRALAMA_ZAMANI	DateTime		Hayır
STOK_ID	MediumInt(8)		Hayır
MUSTERI_ID	SmallInt(5)		Hayır
GERI_DONUS_ZAMANI	DateTime		Evet
CALISAN_ID	TinyInt(3)		Hayır

SON_GUNCELLEME	TimeStamp		Hayır

Tablo 14. Kiralama Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 24.'de gösterilmiştir.



Şekil 24. Kiralama Veri Tabanı Tablosu Görüntüsü

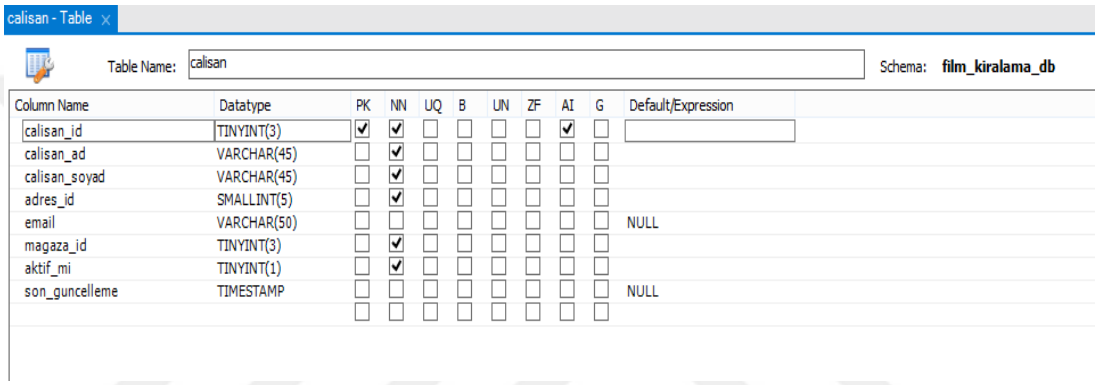
Çalışan Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
CALISAN_ID	TinyInt(3)	Primary Key, Auto Increment	Hayır
CALISAN_AD	Varchar(45)		Hayır
CALISAN_SOYAD	Varchar(45)		Hayır
ADRES_ID	SmallInt(5)		Hayır
EMAIL	Varchar(50)		Evet

MAGAZA_ID	TinyInt(3)		Hayır
AKTIF_MI	TinyInt(1)		Hayır
SON_GUNCELLEME	TimeStamp		Evet

Tablo 15. Çalışan Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 25.'de gösterilmiştir.



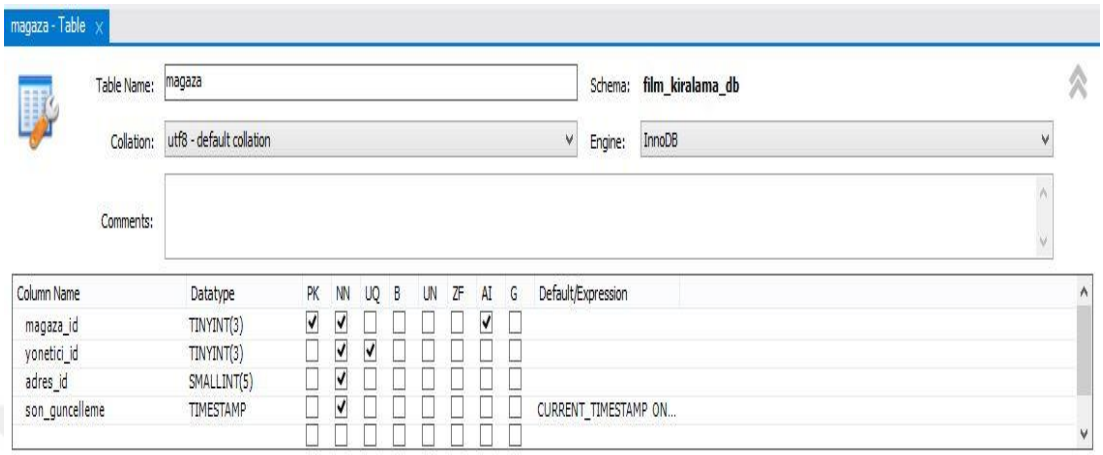
Şekil 25. Çalışan Veri Tabanı Tablosu Görüntüsü

Mağaza Tablosu:

Alan Adı:	**Veri Tipi:**	**Veri Özelliği:**	**Boş(Null) Olabilir mi?**
MAGAZA_ID	TinyInt(3)	Primary Key, Auto Increment	Hayır
YONETICI_ID	TinyInt(3)	Unique Constrant	Hayır
ADRES_ID	SmallInt(5)		Hayır
SON_GUNCELLEME	TimeStamp		Hayır

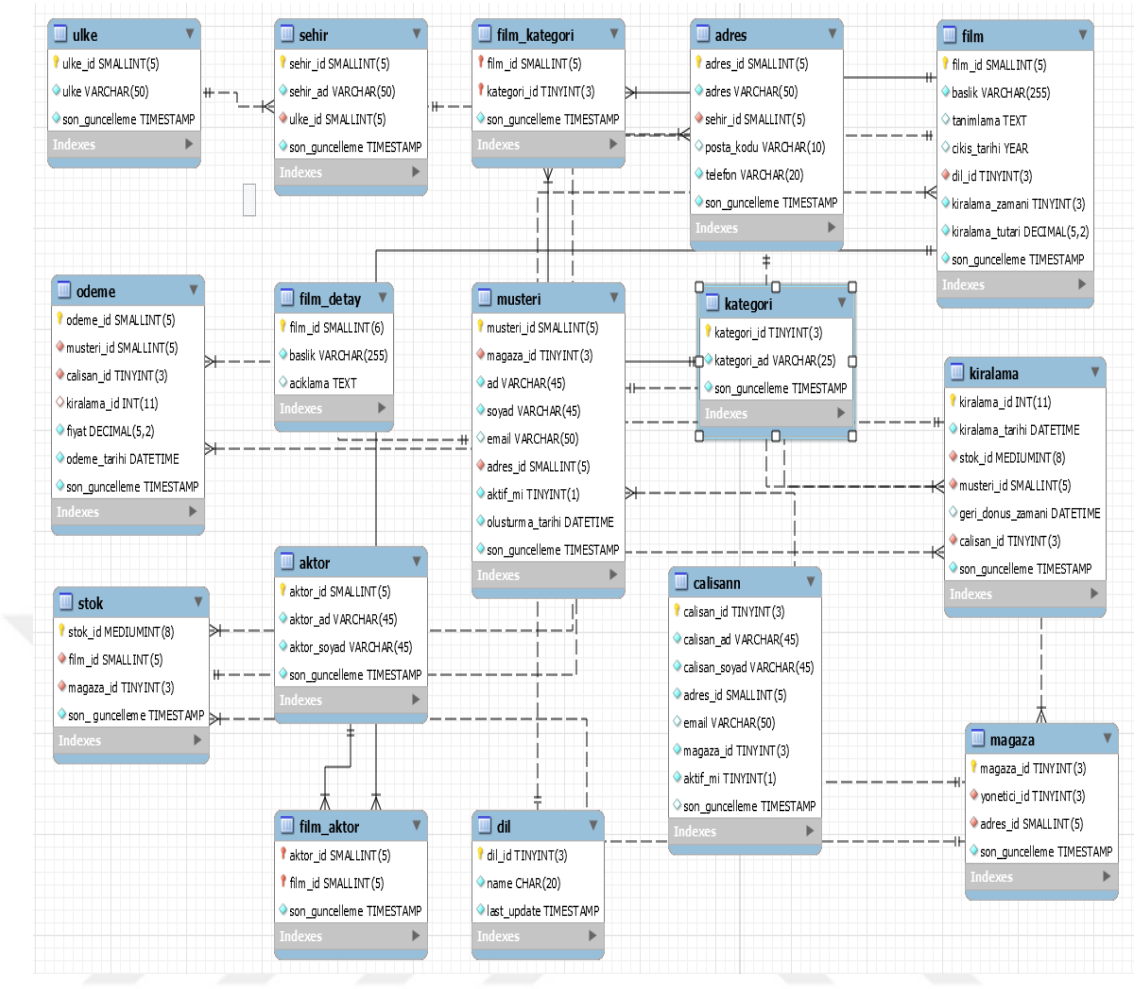
Tablo 16. Mağaza Veri Tabanı Tablosu

Veri tabanındaki görünüşü Şekil 26.'de gösterilmiştir.



Şekil 26. Mağaza Veri Tabanı Tablosu Görüntüsü

Oluşturulan veri tabanındaki şemada yer alan tabloları içeren diyagram Şekil 27.'de gösterilmiştir.

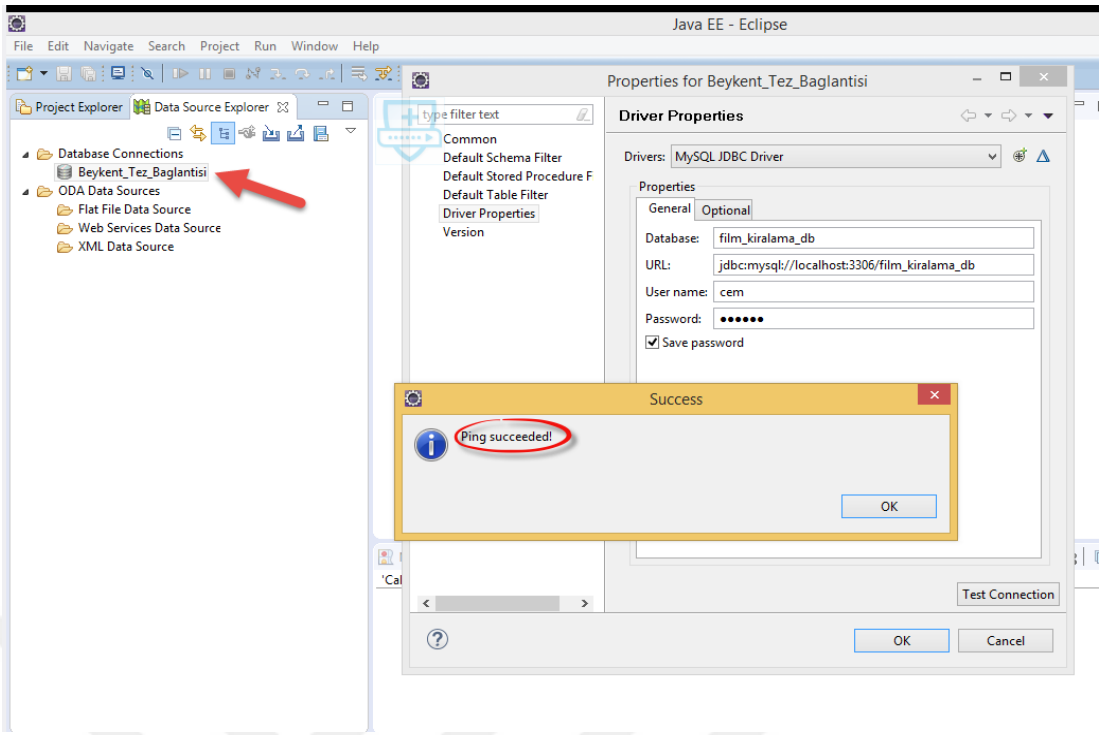


Şekil 27. Veri Tabanı İlişki Diyagramı

2.3 Jdbc Çalışmaları

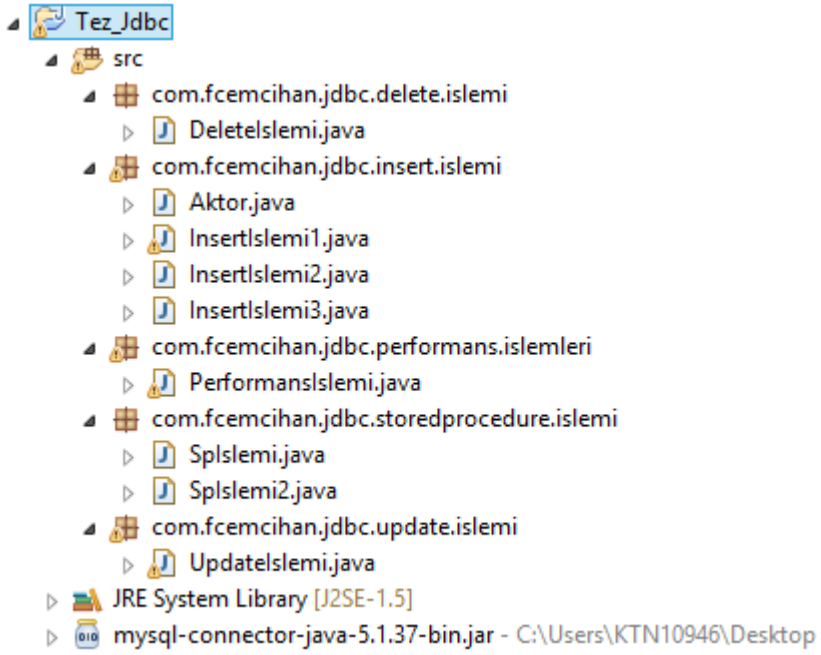
Kullanılmış olan veri tabanı tipine bağlı olarak Java programlama dili ve veri tabanı arasında katman oluşturabilmek adına Java MySql Connectar Jar kütüphanesi oluşturulan Java projesine eklenmiştir. [17]

Jar kütüphanesi eklendikten sonra MySql veri tabanı erişim bilgileri girilerek Eclipse IDE veri tabanı kaynak araştırma (database source explorer) sekmesinden veri tabanı ile Java projesinin haberleştiğini Şekil 28.'de görebiliyoruz.



Şekil 28. Eclipse IDE ve MySql Veri Tabanı Arasında Bağlantı Kurulması

Jdbc uygulamaları için oluşturulmuş olan Java projelerinin Eclipse IDE üzerindeki genel yapısı Şekil 30.'da gösterilmiştir.



Şekil 29. Jdbc Kullanılarak Oluşturulan Java Projeleri Genel Yapısı

Jdbc kullanarak çeşitli çalışmalar yapılmıştır. Üç farklı şekilde veri kayıt etme işlemi, veri kayıt silme, veri kayıt güncelleme ve performans işlemleri yapılmıştır.

2.3.1 Jdbc Kullanarak MySql üzerinde Var Olan Tabloya Veri Kayıt Etme Ve Eklenen Verileri Gösterme İşlemi Çalışması

Jdbc kullanarak önceden oluşturulmuş olan veri tabanı tablolarına kayıt işlemini yapan kod parçası aşağıdadır. İlgili sınıfta (class) sadece kayıt işlemini yapan metodun kodu eklenmiştir.

```
public static void main(String[] args) throws SQLException {
        Connection baglanti = null;
        Statement durum = null;
        ResultSet sonuc = null;
        String url = "jdbc:mysql://localhost:3306/film_kiralama_db";
        String kullanici = "cem";
        String sifre = "123456";
        try {
            baglanti = DriverManager.getConnection(url, kullanici,
sifre);
            durum = baglanti.createStatement();
            System.out.println("Aktor Tablosuna birden fazla "
                + "kayit ekleyecegim. Insert islemi yapıyorum...");
            Aktor aktorJohnny = new Aktor(500, "Johnny", "Depp",
                    Calendar.getInstance());
            Aktor aktorJim = new Aktor(501, "Jim", "Carrey",
                    Calendar.getInstance());
            List<Aktor> aktorList = new ArrayList<Aktor>();
            aktorList.add(aktorJohnny);
            aktorList.add(aktorJim);

            for (int i = 0; i < aktorList.size(); i++) {

                StringBuilder sql = new StringBuilder();
                sql.append("INSERT INTO AKTOR(aktor_id,
```

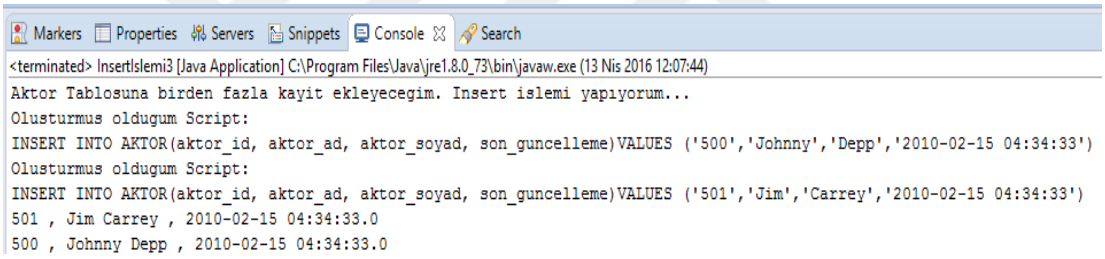
```
aktor_ad, aktor_soyad, son_guncelleme)VALUES ('');
                    sql.append(aktorList.get(i).getAktor_id());
                    sql.append("','");
                    sql.append(aktorList.get(i).getAktor_ad());
                    sql.append("','");
                    sql.append(aktorList.get(i).getAktor_soyad());
                    sql.append("','");
                    sql.append("2010-02-15 04:34:33");
                    sql.append("')");
                    System.out.println("Olusturmus oldugum Script:");
                    String sqlbirlestirmeSonuc = sql.toString();
                    System.out.println(sqlbirlestirmeSonuc);
                    durum.executeUpdate(sql.toString());
                }
                sonuc = durum.executeQuery("SELECT * FROM AKTOR
ORDER BY aktor_id DESC");
                while (sonuc.next()) {
                    System.out.println(sonuc.getString("aktor_id") + " , "
                            + sonuc.getString("aktor_ad") + " "
                            + sonuc.getString("aktor_soyad") + " ,
"+ sonuc.getString("son_guncelleme"));
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                if (sonuc != null) {
                sonuc.close();
                }
                if (durum != null) {
                durum.close();
                }
                if (baglanti != null) {
                baglanti.close();
```

```
                    }
                }
            }
```

Tablo 17. Jdbc İle Veri Ekleme Ve Veri Gösterme Java Kaynak Kodu

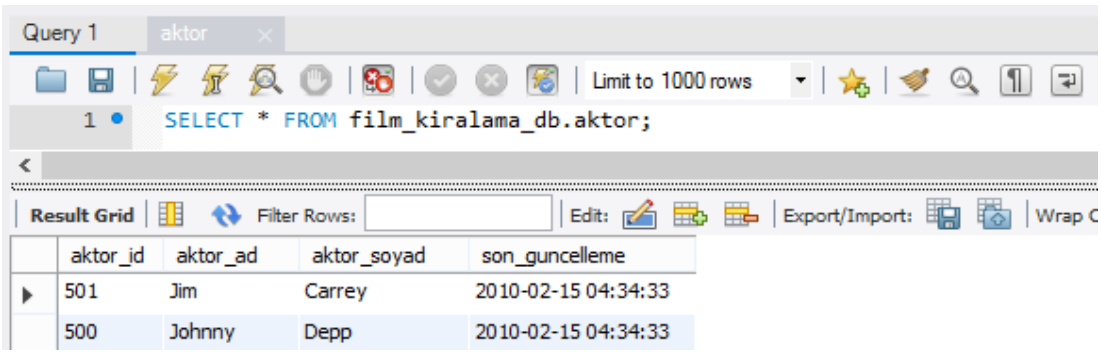
Jdbc kullanarak öncelikle MySql veri tabanına bağlanabilmek adına erişim bilgilerini bir değişkene atılıyor. Java'nın sağlamış olduğu sınıflar ve arayüzler (interface) yardımıyla oluşan bağlantı üzerinden Aktor sınıfından örnek almalar (instance) yapılarak yeni nesne tanımlanıyor. Yaratılan instancelar bir metod yardımıyla Sql Script yazılarak veri tabanındaki Aktor tablosuna kayıt ediliyor.

Kodun Eclipse IDE tarafından derlendikten sonra konsolundaki sonuç Şekil 11.'de gösterilmiştir.



Şekil 30. Jdbc İle Veri Ekleme Ve Veri Gösterme Eclipse IDE Çıktısı

MySql veri tabanındaki Aktor Tablosuna oluşturulan kayıtlar aşağıda görüldüğü gibi eklenmiştir.



Şekil 31. Jdbc İle Veri Ekleme Ve Veri Gösterme MySql Aktor Tablosu Görünümü

Görülmüş olduğu gibi Jdbc'yi Java da kullanabilmek için önceden oluşturulmuş olan bir veri tabanına ihtiyaç vardır. Var olan şema ve tablolarda Jdbc yardımıyla Sql Scriptler yazarak işlemleri yapabilmek mümkündür. Bunu yapabilmek için ise veri tabanı programlama bilgisine ihtiyaç duymaktadır. Sadece Jdbc ile Java kodu yazmak yeterli olmayacaktır.

2.3.2 Jdbc Kullanarak MySql Üzerinde Var Olan Verileri Güncelleme Ve Güncellenen Verileri Gösterme İşlemi Çalışması

Oluşturulan veri güncellenmek istendiğinde ise aşağıda metodun kod parçası kullanılmıştır.

```
public static void main(String[] args) throws SQLException {
		Connection baglanti = null;
		Statement durum = null;
		ResultSet sonuc = null;
		String url = "jdbc:mysql://localhost:3306/film_kiralama_db";
		String kullanici = "cem";
		String sifre = "123456";
		try {
			baglanti = DriverManager.getConnection(url, kullanici, sifre);
			durum = baglanti.createStatement();
			System.out.println("Update edilecek veriyi gösterelim...");
			calisanGoster(baglanti, "Jim", "Carrey");
			System.out.println("Jim Carrey icin guncelleme islemi basliyor...");
			String guncelDurum = "Carrey";
			int etkilenenSatirSayisi = durum.executeUpdate("UPDATE AKTOR SET aktor_soyad"
					+ "= 'CARREY' WHERE aktor_soyad =" + " '"+ guncelDurum + "' AND aktor_ad = 'Jim'");
			System.out.println("Jim Carrey icin guncelleme islemi bitti ve sonuc: ");
			calisanGoster(baglanti, "JIM", guncelDurum);
		} catch (Exception e) {
```

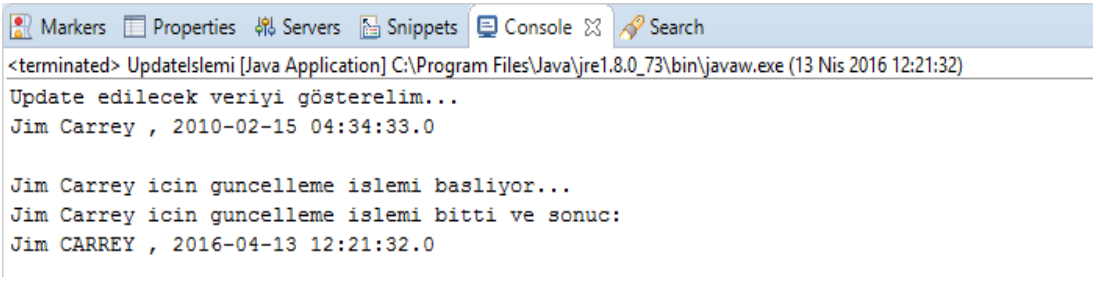
```
                e.printStackTrace();
            } finally {
                baglantiKapat(baglanti, durum, sonuc);
            }
        }
        private static void calisanGoster(Connection baglanti, String first_name,
String last_name) throws SQLException {
            PreparedStatement preSta = null;
            ResultSet rSet = null;
            try {preSta = baglanti.prepareStatement("SELECT aktor_ad,
aktor_soyad,  son_guncelleme FROM AKTOR WHERE aktor_soyad = ? "
                                + "AND aktor_ad = ?");
                preSta.setString(1, last_name);
                preSta.setString(2, first_name);
                rSet = preSta.executeQuery();
                while (rSet.next()) {
                    String aktor_soyad = rSet.getString("aktor_soyad");
                    String aktor_ad = rSet.getString("aktor_ad");
                    String                              son_guncelleme=
rSet.getString("son_guncelleme");
                    System.out.println(aktor_ad + " " + aktor_soyad + " ,
"+ son_guncelleme + "\n");
                }
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                durumKapat(preSta, rSet);
            }
        }
```

Tablo 18. Jdbc İle Veri Güncelleme Ve Güncellenen Veri Gösterme Java Kaynak Kodu

Veri tabanında var olan bir kayıt güncellemek istendiğinde yine Java kodları yardımıyla gerçekleştirilebilir. Var olan kayıt Sql Script'i yazarak güncellenmiştir ve

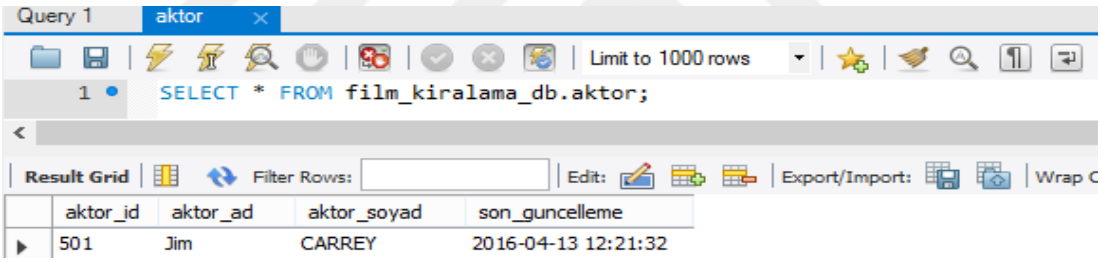
yine Script ile sorgu çalıştırarak güncellemenin gerçekleşip gerçekleşmediğini ekranda gösterilmiştir.

Çalıştırılan kod sonucunda Eclipse IDE konsolunun görüntüsü Şekil 32.'deki gibidir.



Şekil 32. Jdbc İle Veri Güncelleme Ve Güncellenen Veri Gösterme Eclipse IDE Çıktısı

Veri tabanında güncellenen kayıt ise Şekil 33.'de gösterilmiştir.



Şekil 33. Jdbc İle Veri Güncelleme Ve Güncellenen Veri Gösterme MySql Aktor Tablosu Görünümü

Gerçekleştirilen uygulamalarda çok fazla veri tabanı programlama bilgisine ihtiyaç duyulmasa da kod üzerinde büyük ve karmaşık sorgularda veri tabanı işlemleri yapabilmek adına bunu bilmek şarttır. Bu bilgiye sahip bir yazılım geliştirici kolaylıkla Jdbc ile Java kodları yardımıyla veri tabanı üzerinde işlemler yapabilir. Ayrıca yine tablolar arasındaki bağlantıları oluşturabilmek, bu bağlantılar ile sorgular çekebilmek çok basit bir hale gelecektir.

2.3.3 Jdbc Kullanarak Performans İşlemi Çalışması

Jdbc kullanarak oluşturulan veri tabanındaki Adres tablosuna 50000 tane kayıt ekleme işlemi yapan metodun kod parçası aşağıdadır.

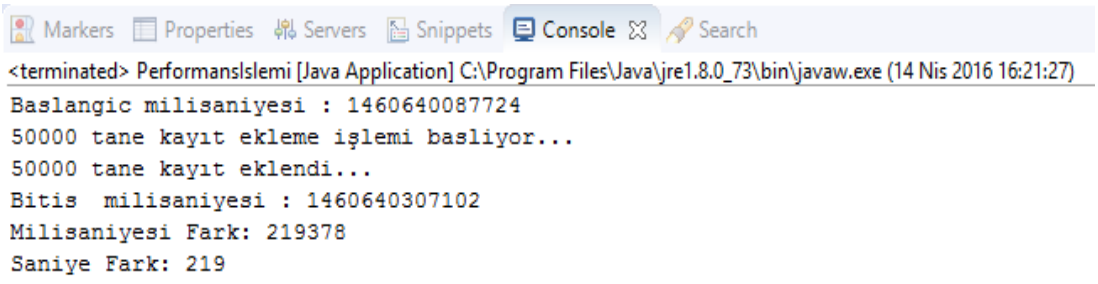
```
public static void main(String[] args) throws SQLException {
		Connection baglanti = null;
		Statement durum = null;
		ResultSet sonuc = null;
		String url = "jdbc:mysql://localhost:3306/film_kiralama_db";
		String kullanici = "cem";
		String sifre = "123456";
		try {
			baglanti = DriverManager.getConnection(url, kullanici, sifre);
			durum = baglanti.createStatement();
			long baslangic = System.currentTimeMillis();
			System.out.println("Baslangic milisaniyesi : " +
System.currentTimeMillis());
			// 50000 tane kayit ekleniyor.
			System.out.println("50000 tane kayıt ekleme işlemi
basliyor...");
			for (int i = 100; i < 50100; i++) {
				boolean etkilenenSatirSayisi = durum.execute(
"INSERT INTO ADRES(ADRES_ID, ADRES, SEHIR_ID, POSTA_KODU,
TELEFON, SON_GUNCELLEME)VALUES"+ " ('" + i + "','West California', '" +
38+ "', 'WES1500', '+45414565657', '2016-02-15 04:34:33')");
			}
			System.out.println("50000 tane kayıt eklendi...");
			long bitis = System.currentTimeMillis();
			System.out.println("Bitis milisaniyesi : " +
System.currentTimeMillis());
			long fark = bitis - baslangic;
			System.out.println("Milisaniyesi Fark: " + fark);
			long saniyeFark = fark / 1000;
			System.out.println("Saniye Fark: " + saniyeFark);
		} catch (Exception e) {
```

```
                e.printStackTrace();
            } finally {
                if (sonuc != null) {
                    sonuc.close();
                }
                if (durum != null) {
                    durum.close();
                }
                if (baglanti != null) {
                    baglanti.close();
                }
            }
        }
```

Tablo 19. Jdbc Performans Çalışması Java Kaynak Kodu

Adres isimli bir Java sınıfı oluşturarak bu sınıfa bağlı 50000 tane nesne oluşturulup, Jdbc yardımı ile veri tabanını kayıt işlemi gerçekleştirilirken başlangıç ve bitiş süreleri farkları ortaya çıkarılmıştır.

Bu kodun derlenmesi sonucu çıkan sonucun Eclipse IDE konsolundaki çıktısı Şekil 34.'de gösterilmiştir.



Şekil 34. Jdbc Performans Çalışması Eclipse IDE Çıktısı

Görüldüğü üzere 50000 kayıt, 219 saniye de veri tabanına kayıt edilmiştir. Yaklaşık olarak 3,65 dakikaya karşılık gelmektedir.

Yapılan Çalışma	Çalışma Süresi Saniye	Çalışma Süresi Dakika
Jdbc	219	3,65

Tablo 20. Jdbc Performans Çalışması Sonuçları

2.4 Hibernate Çalışmaları

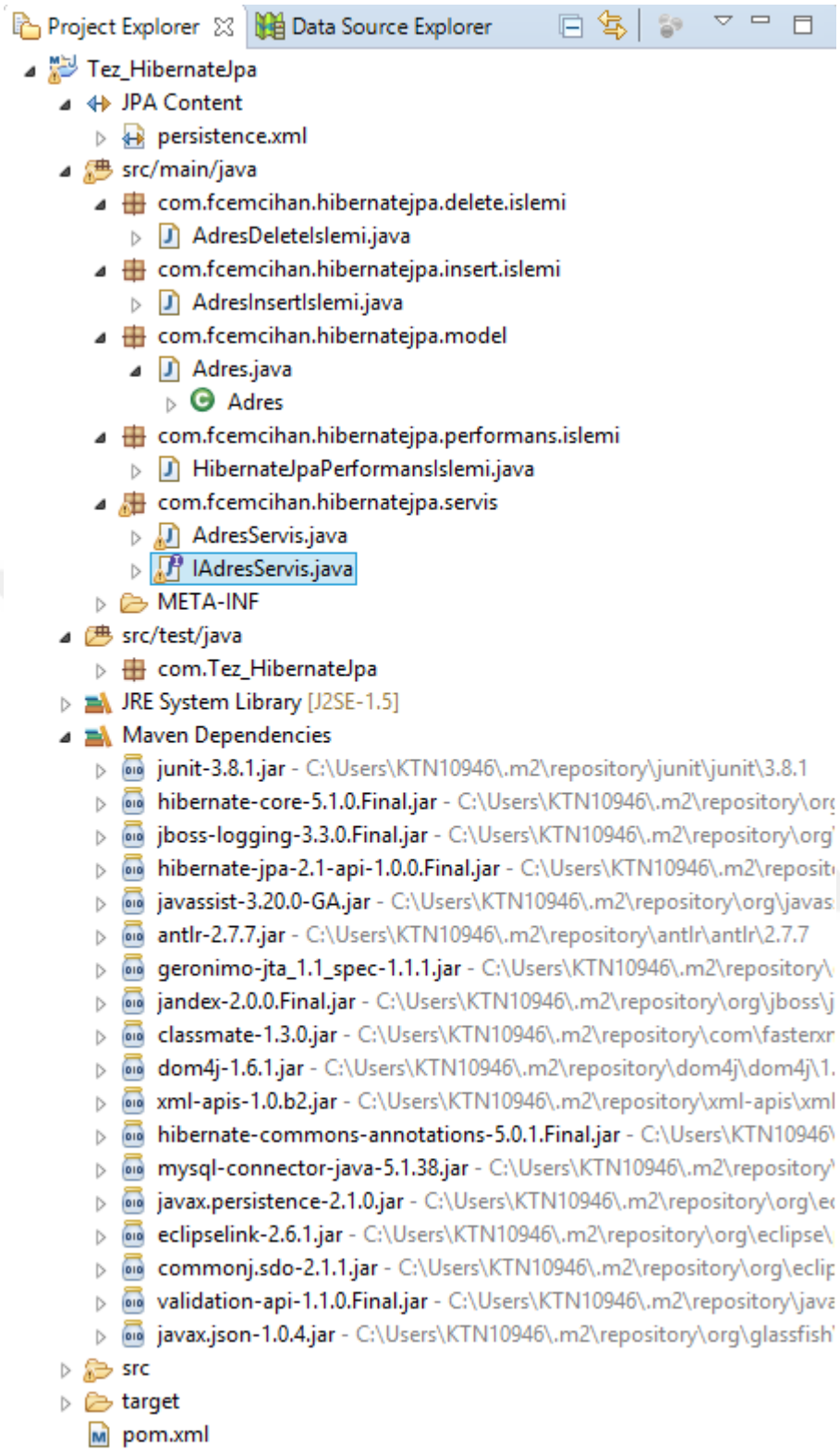
Hibernate çalışmasını yapabilmek adına Hibernate'in kütüphaneleri birden fazla olduğu için Maven proje yapısı oluşturulmuştur. Bunun için Eclipse IDE üzerinde Maven projesi seçilerek yaratılmıştır. İlgili Jarları ise Maven'ın sunmuş olduğu pom.xml üzerinden bağımlılıklar (java dependencies) sağlayarak internet üzerinden kütüphaneleri indirmesini sağlanmıştır. Bunun için pom.xml aşağıdaki şekilde düzenlenmiştir.

```
<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>5.1.0.Final</version>
</dependency>
<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>5.1.38</version>
</dependency>
```

Tablo 21. Hibernate İhtiyaç Duyulan Jar Kütüphanesi Ekleme

Bunun sonucunda Maven yardımıyla oluşturulan projeye Hibernate Jar kütüphaneleri eklenmiştir.

Eklenmiş olan bağımlılık kütüphaneleri ve bu tez kapsamındaki Hibernate çalışmaları için oluşturulan yapı Şekil 35.'de aşağıdaki şekilde yansımıştır.



Şekil 35. Hibernate Kullanılarak Oluşturulan Java Projeleri Genel Yapısı ve Bağımlılıkları

Hibernate kullanarak yukarıdaki çalışmalar gerçekleştirilmiştir. MySql veri tabanına veri kaydetme, veri güncelleme, veri silme ve performans işlemleri çalışmalarıdır. Bu çalışmaları yaparken aynı anda hem veri kayıt etme hem de veri

silme işlemi veya veriyi kayıt ederken aynı anda da güncelleme gibi çalışmalar gerçekleştirilmiştir. Oluşturulan veri tabanına verileri modellerken bu çalışmada ihtiyaç duyulmuştur.

Hibernate veri tabanında yeni tablo oluşturma ve bu tabloya veri ekleme işlemlerini gerçekleştirebilir. Kullanılmak istenen veri tabanına bağlantı persistence.xml yardımıyla sağlanır. Bağlanılmak istenen veri tabanı şeması, kullanıcı adı ve şifresi girilerek bağlantı sağlanabilir. Oluşturulan veri tabanı şemasına Hibernate'in bağlanabilmesi için düzenlenen persistence.xml kodları aşağıdadır.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"version="2.1">
<persistence-unit name="AdresUnit" transaction-type="RESOURCE_LOCAL">
                <class>com.fcemcihan.hibernatejpa.model.Adres</class>
                <properties>
<propertyname="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
                <property name="javax.persistence.jdbc.url"
                        value="jdbc:mysql://localhost:3306/film_kiralama_db" />
                <property name="javax.persistence.jdbc.user" value="cem" />
              <property name="javax.persistence.jdbc.password" value="123456" />
                <property name="eclipselink.ddl-generation" value="drop-and-create-
tables" />
                <property name="eclipselink.logging.level" value="OFF" />
                </properties>
        </persistence-unit>
</persistence>
```

Tablo 22. Hibernate Persistence.Xml Kaynak Kodları

Burada görüldüğü gibi Hibernate kullanabilmek için veri tabanı programlama bilgisine sahip olmak gerekmemektedir. Ayrıca yazılım geliştiricilerin yazılan Java

kodunun içinde herhangi bir Sql Script yazılmamaktadır. Bunları çeşitli Java sınıfları ve arayüzleri yardımıyla yapmak mümkündür. Fakat bazı sorgularda Hibernate'e has olan sorgu dili olan HQL kullanarak işlemler yapılabilmektedir.

Veri tabanında oluşturmak istediğiniz tablodaki alanları ve isimlerini Hibernate in bize sunmuş olduğu javax.persistence.entity kütüphanesini kullanarak oluşturulmuştur. Java kodu ile yazılmış olan sınıfı veri tabanındaki aynı değerler (mapping işlemi) ile modellenmiştir.

Adres isimli Java sınıfındaki alanlar aşağıdadır.

```
@Id
private int adres_id;
private String adres;
private int sehir_id;
private String posta_kodu;
private String telefon;
private String son_guncelleme;
```

Tablo 23. Adres Sınıfı Modelleme Java Kaynak Kodu

2.4.1 Hibernate Kullanarak MySql Üzerinde Var Olan Tabloya Veri Ekleme İşlemi Çalışması

Hibernate kullanarak veri tabanında var olan tabloya veri ekleme işlemi yapan metodunun kod parçası aşağıdadır.

```
public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory = Persistence
                .createEntityManagerFactory("AdresUnit");
        EntityManager entityManager = entityManagerFactory
                .createEntityManager();
        EntityTransaction entityTransaction = entityManager.getTransaction();
        IAdresServis adresServis = new AdresServis(entityManager);
```

```
                entityTransaction.begin();
                Adres adresJude = adresServis.adresOlustur(12, "West California",
38,"WES1500", "+1456565677", "2014-09-25 22:32:18");
                Adres adresJohnny = adresServis.adresOlustur(13, "Los Angeles",
463,"LOS2540", "+1456565677", "2011-09-25 22:32:18");
                Adres adresJim = adresServis.adresOlustur(14, "Newyork", 463,
                                "NEW1500", "+1456565677", "2016-09-25 22:32:18");

                entityTransaction.commit();
                System.out.println("Adres1 Eklendi: " + adresJude);
                System.out.println("Adres2 Eklendi: " + adresJohnny);
                System.out.println("Adres3 Eklendi: " + adresJim);
                @SuppressWarnings("unchecked")
                List<Adres> adresler = adresServis.tumAdresleriGoster();
                for (Adres adr : adresler) {
                        System.out.println("Adresler:" + adr);
                }
                entityManager.clear();
                entityManagerFactory.close();
        }
```

Tablo 24. Hibernate İle Var Olan Tabloya Veri Ekleme Java Kaynak Kodu

Bu kodun çalışması için gerekli AdresServis isimli Java sınıfındaki metodlar aşağıdadır.

```
public Adres adresOlustur(int adres_id, String adres, int sehir_id,
                        String posta_kodu, String telefon, String son_guncelleme) {

                Adres adreS = new Adres(adres_id, adres, sehir_id, posta_kodu,
telefon,son_guncelleme);
                entityManager.persist(adreS);
                return adreS;
```

```
    }
    public Adres adresGoster(int adres_id) {
        Adres adres = entityManager.find(Adres.class, adres_id);
        return adres;
    }
```

Tablo 25. AdresServis Java Kaynak Kodu

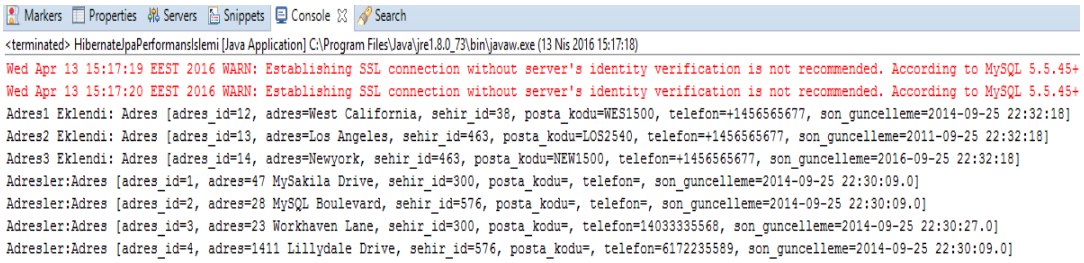
Veri ekleme işleminin çalışması için gerekli olan IAdresServis Java arayüzünün kodları aşağıdadır.

```
public interface IAdresServis {
    public Adres adresOlustur(int adres_id, String adres, int sehir_id,
                    String posta_kodu, String telefon, String son_guncelleme);

    public Adres adresGoster(int adres_id);
    public void adresSil(int adres_id);
    public List tumAdresleriGoster();
}
```

Tablo 26. IAdresServis Arayüzü Java Kaynak Kodu

Adres isimli Java sınıfı veri tabanındaki Adres tablosu ile aynı alanları içermektedir. Oluşturulan nesne tabanlı yapı ile Main metodu içindeki IAdresServis arayüzündeki metodların gövdeleri ile AdresServis sınıfının içinde kodlanmış Main metodunda direk olarak instance alınarak kullanılmıştır.

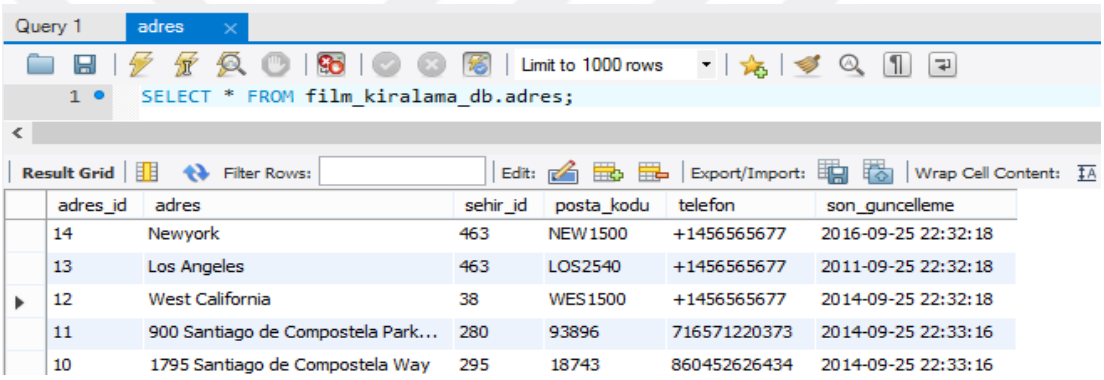
Eclipse IDE de kodun derlenmesi sonucunda konsol ekranının çıktısı Şekil 36.'da gösterilmiştir.



Şekil 36. Hibernate İle Var Olan Tabloya Veri Ekleme Eclipse IDE Çıktısı

Kod tarafından oluşturan veriler ve daha önceden tabloda yer alan verileri ekranda görüntülenmektedir.

Şekil 37.'de veri tabanındaki görüntüsü yer almaktadır.

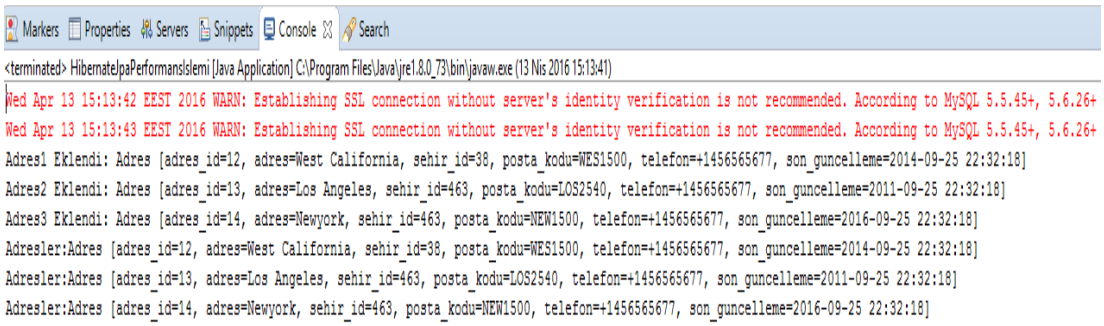


adres_id	adres	sehir_id	posta_kodu	telefon	son_guncelleme
14	Newyork	463	NEW1500	+1456565677	2016-09-25 22:32:18
13	Los Angeles	463	LOS2540	+1456565677	2011-09-25 22:32:18
12	West California	38	WES1500	+1456565677	2014-09-25 22:32:18
11	900 Santiago de Compostela Park...	280	93896	716571220373	2014-09-25 22:33:16
10	1795 Santiago de Compostela Way	295	18743	860452626434	2014-09-25 22:33:16

Şekil 37. Hibernate İle Var Olan Tabloya Veri Ekleme MySql Adres Tablosu Görünümü

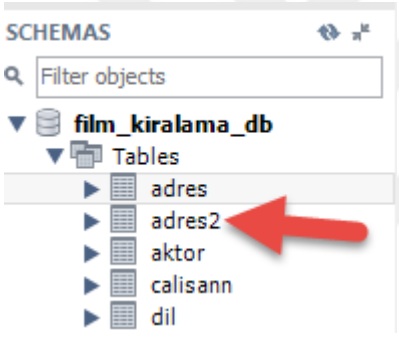
Eğer aynı isim ile bağlantı kurulan veri tabanında tablo var ise sadece veri eklemekle yetinecektir. Fakat aynı isimde bir tablo yok ise tabloyu oluşturup veri ekleme işini yapacaktır.

Kodumdaki Adres isimli sınıfımı Adres2 olarak değiştirip aynı kodu tekrar çalıştırdığımda ise konsol ekranımım son hali Şekil 38.'deki gibi olacaktır.

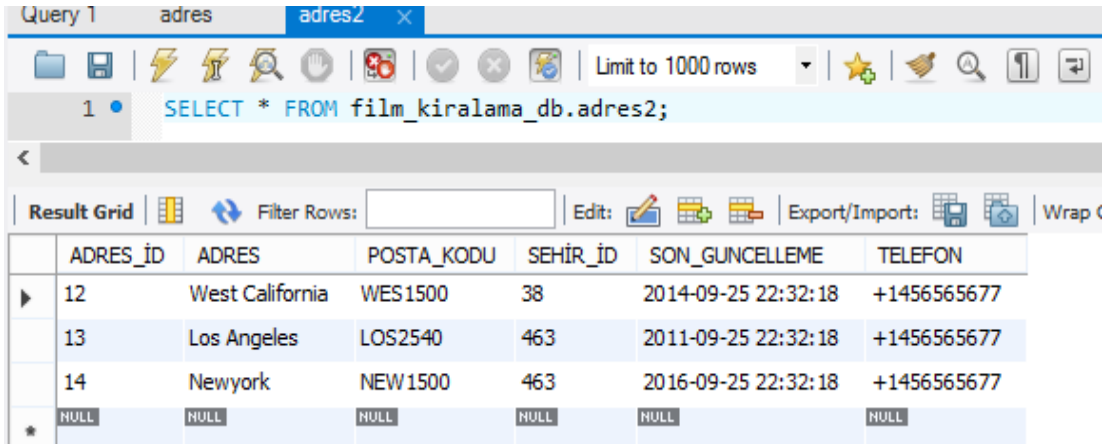


Şekil 38. Hibernate İle Tablo Oluşturma Ve Tabloya Veri Ekleme Eclipse IDE Çıktısı

Yeni bir tablo oluşturulduğu için sadece şu anda oluşturmuş olan veriler vardır. Veri tabanı tablolarına şu şekilde eklenmiş olup tablonun görünümü Şekil 39.'da gösterilmiştir.



Şekil 39. Hibernate İle Yeni Tablo Oluşturma MySql Adres2 Tablosu Görünümü



Şekil 40. Hibernate İle Tablo Oluşturma Ve Tabloya Veri Ekleme MySql Adres2 Tablosu Görünümü

2.4.2 Hibernate Kullanarak MySql Üzerinde Veri Kayıt Etme Ve Veri Silme İşlemi Çalışması

Aynı yapıyı kullanarak aynı anda hem veri ekleme hemen ardından da veri silme işlemi yapan metodun kod parçası aşağıdadır.

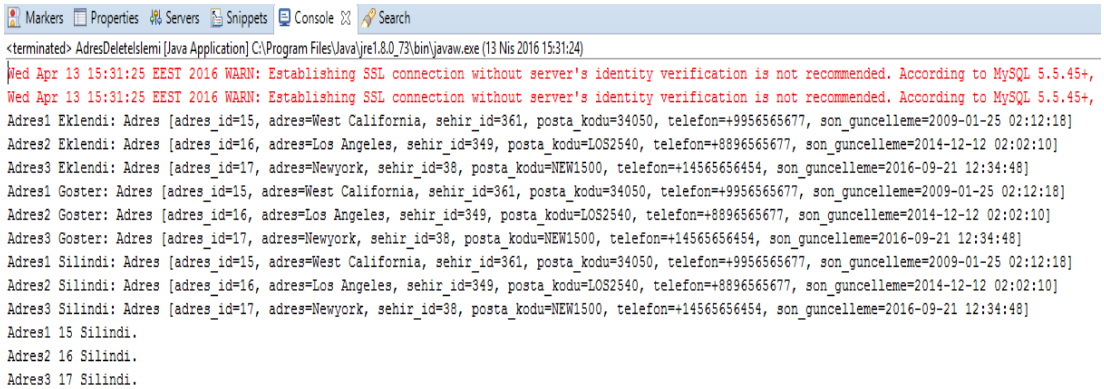
```
public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory = Persistence
                .createEntityManagerFactory("AdresUnit");
        EntityManager entityManager = entityManagerFactory
                .createEntityManager();
        EntityTransaction entityTransaction = entityManager.getTransaction();
        IAdresServis adresServis = new AdresServis(entityManager);
        entityTransaction.begin();
        Adres adresJennifer = adresServis.adresOlustur(15, "West California",361, "34050", "+9956565677", "2009-01-25 02:12:18");
        Adres adresMary = adresServis.adresOlustur(16, "Los Angeles", 349,"LOS2540", "+8896565677", "2014-12-12 02:02:10");
        Adres adresGeorge = adresServis.adresOlustur(17, "Newyork", 38, "NEW1500", "+14565656454", "2016-09-21 12:34:48");

        entityTransaction.commit();
        System.out.println("Adres1 Eklendi: " + adresJennifer);
        System.out.println("Adres2 Eklendi: " + adresMary);
        System.out.println("Adres3 Eklendi: " + adresGeorge);
        Adres adres1 = adresServis.adresGoster(15);
        System.out.println("Adres1 Goster: " + adres1);
        Adres adres2 = adresServis.adresGoster(16);
        System.out.println("Adres2 Goster: " + adres2);
        Adres adres3 = adresServis.adresGoster(17);
        System.out.println("Adres3 Goster: " + adres3);

        entityTransaction.begin();
```

```
            adresServis.adresSil(15);
            System.out.println("Adres1 Silindi: " + adres1);
            adresServis.adresSil(16);
            System.out.println("Adres2 Silindi: " + adres2);
            adresServis.adresSil(17);
            System.out.println("Adres3 Silindi: " + adres3);
            entityTransaction.commit();
            System.out.println("Adres1 " + 15 + " Silindi.");
            System.out.println("Adres2 " + 16 + " Silindi.");
            System.out.println("Adres3 " + 17 + " Silindi.");
            entityManager.clear();
            entityManagerFactory.close();
    }
```

Tablo 27. Hibernate İle Veri Ekleme Ve Veri Silme Java Kaynak Kodu

Adres sınıfının yardımıyla birden fazla Adres örnek alma işlemi gerçekleştirilmiştir. Ardından oluşturmuş olan bu yeni nesneleri tekrar silinmiştir.

Şekil 41.'de konsol çıktısı gösterilmektedir.



Şekil 41. Hibernate İle Veri Ekleme Ve Veri Silme Eclipse IDE Çıktısı

2.4.3 Hibernate Kullanarak Performans İşlemi Çalışması

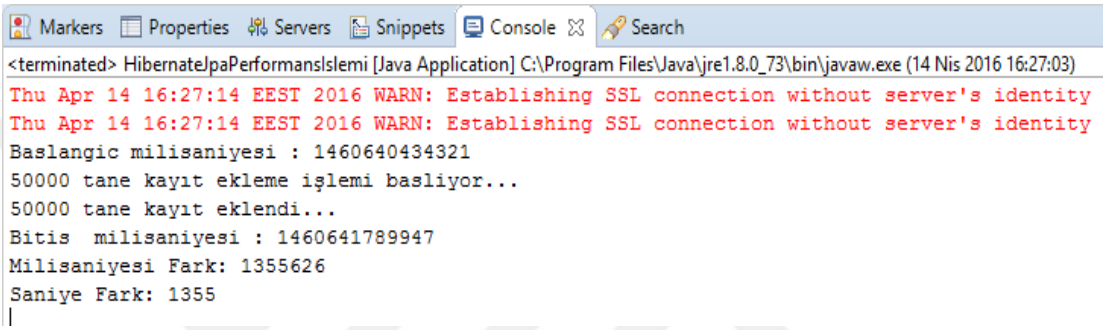
Hibernate kullanarak Jdbc çalışmasında olduğu gibi 50000 veriyi veri tabanına ekleme işlemini yapan metodun kod parçası aşağıdadır.

```
public static void main(String[] args) {
        EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("AdresUnit");
        EntityManager entityManager = entityManagerFactory.createEntityManager();
        EntityTransaction entityTransaction = entityManager.getTransaction();
        IAdresServis adresServis = new AdresServis(entityManager);
        List<Adres> adresler = new ArrayList<Adres>();
        long baslangic = System.currentTimeMillis();
        System.out.println("Baslangic milisaniyesi : " + System.currentTimeMillis());
        // 50000 tane kayit ekleniyor.
        System.out.println("50000 tane kayıt ekleme işlemi basliyor...");
        for (int i = 100; i < 51000; i++) {
            entityTransaction.begin();
            Adres adres = adresServis.adresOlustur(i, "West California", 38, "WES1500", "+45414565657",
                    "2016-04-01 08:32:18");
            entityTransaction.commit();
        }
        entityManager.clear();
        entityManagerFactory.close();
        System.out.println("50000 tane kayıt eklendi...");
        long bitis = System.currentTimeMillis();
        System.out.println("Bitis milisaniyesi : " + System.currentTimeMillis());
        long fark = bitis - baslangic;
        System.out.println("Milisaniyesi Fark: " + fark);
```

```
        long saniyeFark = fark / 1000;
        System.out.println("Saniye Fark: " + saniyeFark);
    }
```

Tablo 28. Hibernate Performans Çalışması Java Kaynak Kodu

Kodun çıktısının Eclipse IDE deki son hali aşağıdadır.



Şekil 42. Hibernate Performans Çalışması Eclipse IDE Çıktısı

50000 adet kayıt Hibernate kullanarak 1355 saniye yani yaklaşık olarak 22,5 dakikada veri tabanına kayıt edilmiştir.

Jdbc kullanılarak yapılan çalışmaya göre kıyaslandığında çok daha yavaş olduğu görülmektedir.

Yapılan Çalışma	**Çalışma Süresi Saniye**	**Çalışma Süresi Dakika**
Jdbc	219	3,65
Hibernate	1355	22,5

Tablo 29. Hibernate ve Jdbc Performans Çalışması Sonuçları

Hibernate, Jdbc ye göre bu çalışmada yaklaşık olarak 6 kat daha yavaş olduğu görülmektedir.

2.5 NoSql Çalışmaları

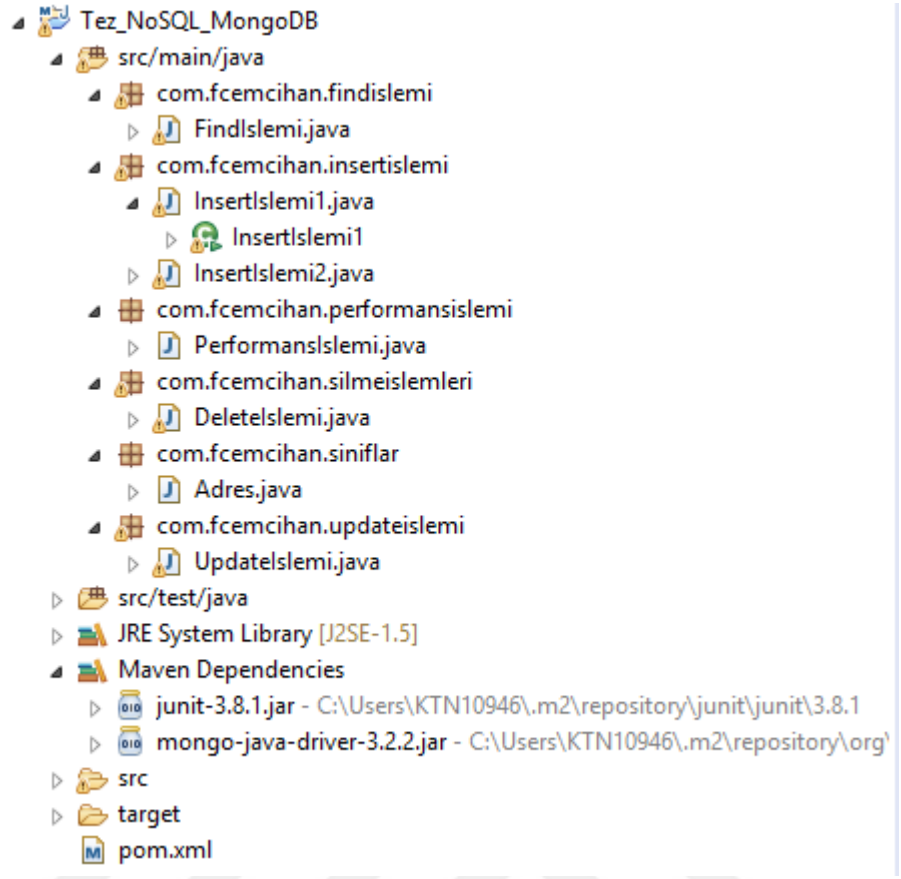
Bu tezde, NoSql çalışmaları için MongoDb veri tabanı kullanılmıştır. MongoDb ücretsiz olarak temin edilip kişisel bilgisayara yüklendikten sonra kurulumu yapılabilir. MongoDb veri tabanı üzerinde yapılan işlemlerin izlenmesi ve çeşitli veri tabanı işlemlerin yapılması mümkündür.[13]

Hibernate çalışmasında olduğu gibi Eclipse IDE üzerinde Maven aracı kullanılarak gerekli olan MongoDb Jar kütüphanesi pom.xml bağımlılıkları ile düzenlenerek projeye eklenmiştir. Eklenen Jar kütüphanesi mongo-java-driver-3.2.2.jar isimli Jar'dır ve pom.xml eklenen bağımlılık kodu ise aşağıdaki şekildedir.

```
<dependency>
		<groupId>org.mongodb</groupId>
		<artifactId>mongo-java-driver</artifactId>
		<version>3.2.2</version>
</dependency>
```

Tablo 30. NoSql İhtiyaç Duyulan Jar Kütüphanesi Ekleme

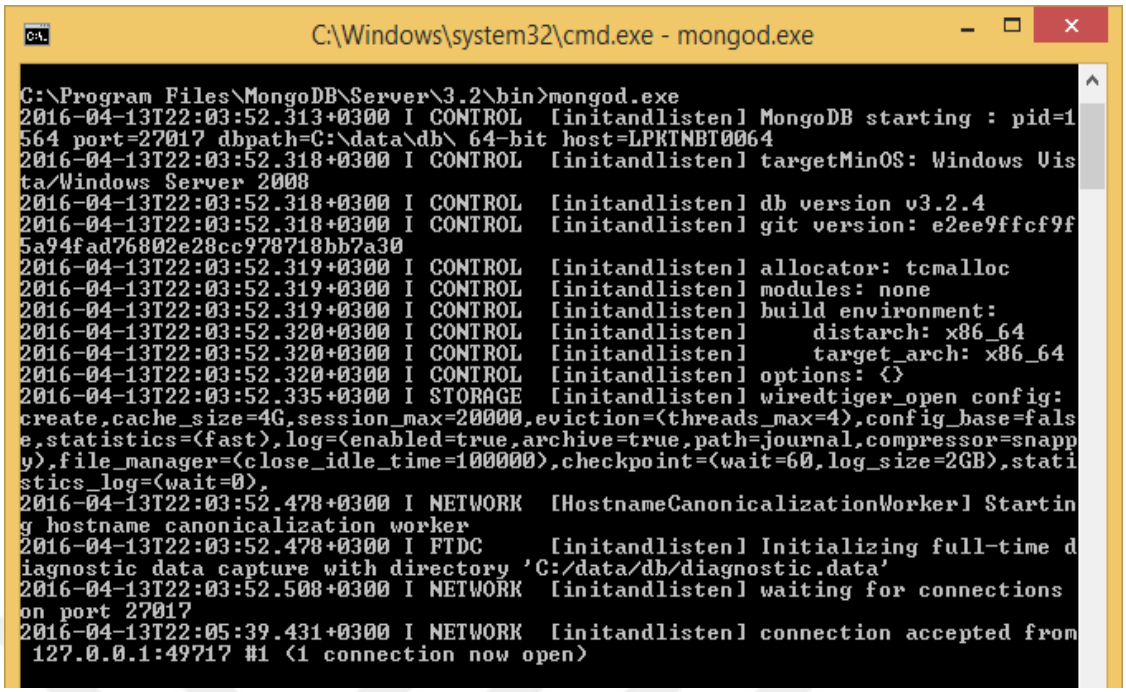
Eclipse IDE de NoSql çalışmaları için yaratılan Maven projesi Şekil 43.'deki yapıda oluşturulmuştur.



Şekil 43. NoSql Kullanılarak Oluşturulan Java Projeleri Genel Yapısı

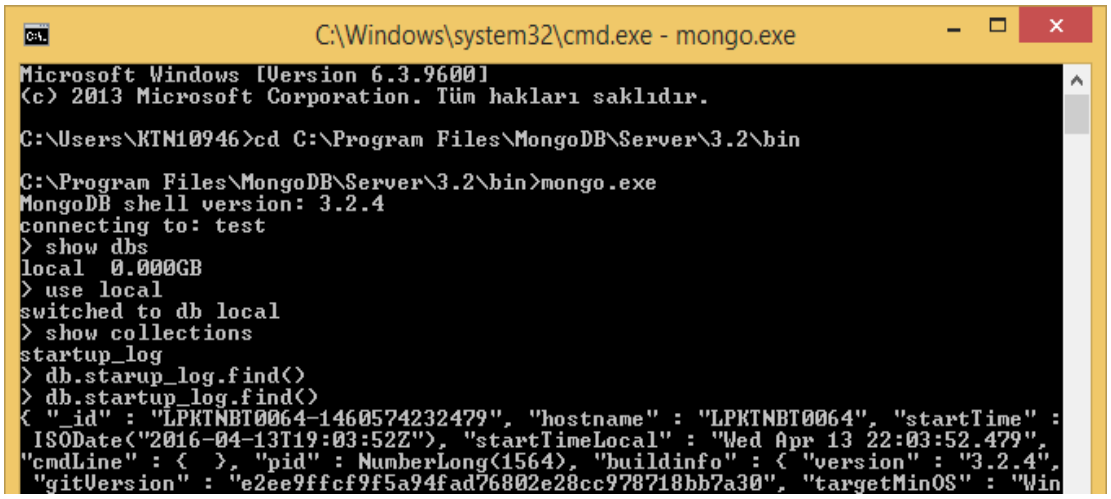
NoSql kullanarak MongoDb veri tabanı üzerinde çeşitli uygulamalar yazılarak Java kodları geliştirilmiştir. MongoDb veri tabanına veri kayıt etme, var olan verileri güncelleme, verileri silme ve performans işlemi çalışmaları yapılmıştır.

MongoDb veri tabanını kişisel bilgisayarınıza yüklendikten sonra veri tabanına bağlanmak için komut istemi(CMD) öncelikle MongoDb sunucusunu çalıştırıyoruz. Şekil 44.'de görüldüğü gibi sunucuyu çalıştırmak için komut istemi üzerinde MongoDb nin yüklü olduğu yolu seçerek "mongod.exe" komutunu yazarak çalıştırdığımızda bize varsayılan olarak portu 27017 şeklinde sunacaktır. [16]



Şekil 44. MongoDb Server'ını Çalıştırma

Şekil 45.'de gösterildiği gibi, MongoDb veri tabanındaki şemaları görmek için kullanıcı tarafında "mongo.exe" komutu yardımıyla veri tabanına bağlanıyoruz. Bağlantı gerçekleştikten sonra "show dbs" komutunu kullanıyoruz. Bu komut veri tabanında oluşmuş olan şemaları bize göstermeye yaramaktadır. Oluşturulmuş olan şemalar seçmek istendiğinde "use" + "şema ismi" şeklinde kodluyoruz. Sonrasında tabloları görmek için "show collections" komutu yazarak istedilen tabloya ulaşabiliyoruz. Tablodaki verileri görmek için ise "db."+ tablo ismi + ".find()" komutunu yazarak tablo içindeki verileri görebiliyoruz. Yine "db." + tablo ismi + count() diyerek tablodaki kayıtlı veri sayısını öğrenebilmekteyiz.[16]



Şekil 45. MongoDb Veri Tabanına Bağlanma

2.5.1 NoSql Kullanarak MongoDb Üzerinde Yeni Şema Oluşturma, Yeni Tablo Oluşturma Ve Veri Kayıt Etme İşlemi Çalışması

NoSql kullanarak veri tabanında yer almayan tabloyu oluşturma ve veri tabanına kayıt ekleme işi yapan metodun kod parçası aşağıdadır.

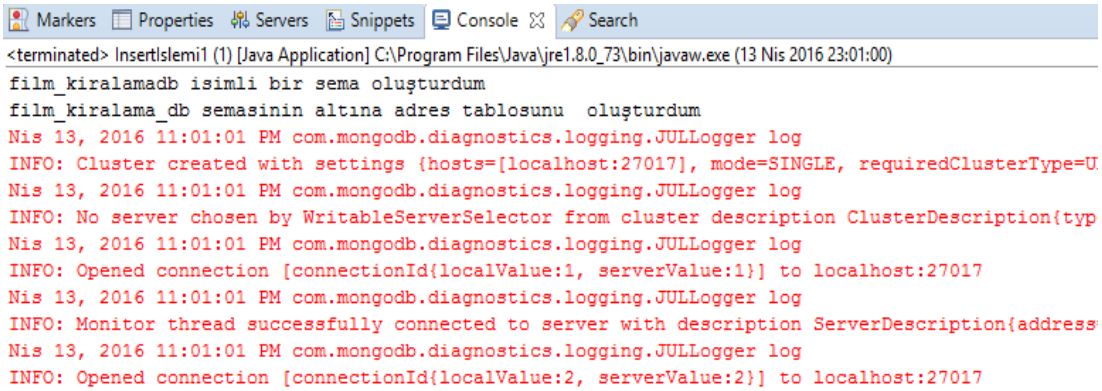
```
public static void main(String[] args) {
        System.out.println("film_kiralamadb isimli bir sema oluşturdum");
        System.out.println("film_kiralama_db semasinin altına adres tablosunu oluşturdum");
        Date bugun = new Date();
        MongoClient mongo = new MongoClient("localhost");
        DB db = mongo.getDB("film_kiralama_db");
        DBCollection dbcol = db.getCollection("adres");
        BasicDBObject adres = new BasicDBObject().append("id", 1)
                .append("adres", "beylikdüzü no:10")
                .append("sehir", "istanbul").append("posta_kodu", 34550)
                .append("telefon", 5370267).append("son_guncelleme", bugun);
        BasicDBObject adres2 = new BasicDBObject().append("id", 2)
                .append("adres", "eryaman blok:A-03  No:10")
```

```
                .append("sehir", "ankara").append("posta_kodu", 06550).append("telefon", 5320267).append("son_guncelleme", bugun);

                dbcol.insert(adres);
                dbcol.insert(adres2);
        }
```

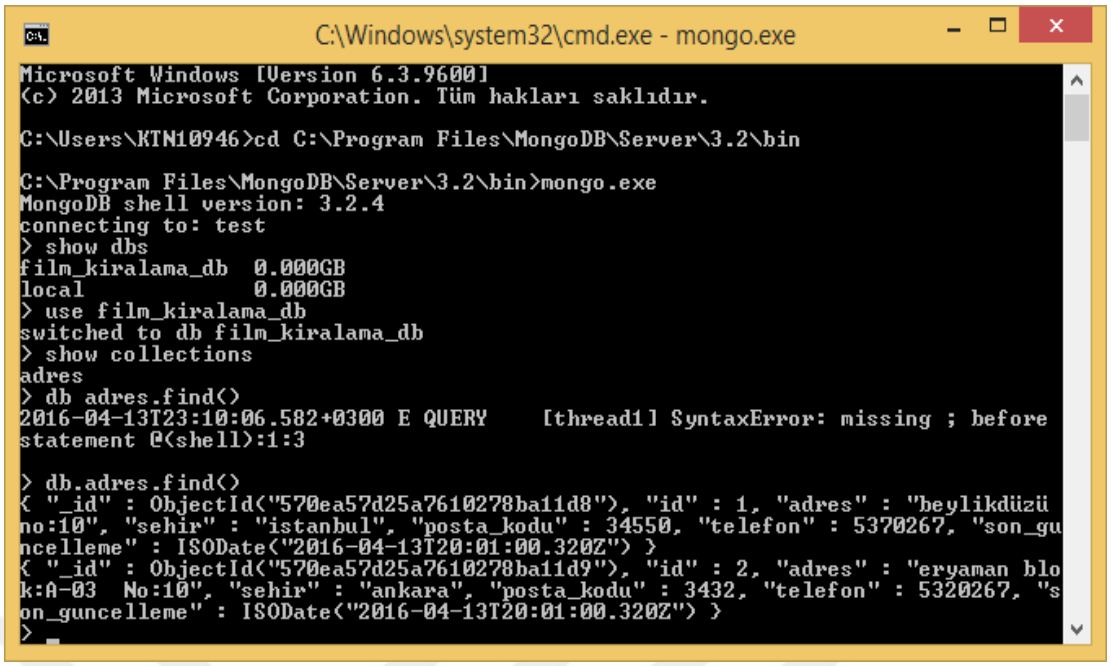
Tablo 31. NoSql İle MongoDb' de Şema Oluşturma, Tablo Oluşturma Ve Tabloya Veri Ekleme Java Kaynak Kodu

MongoDb veri tabanına "film_kiralama_db" isimli bir şema oluşturulmuştur ve bu şemanın içine Adres isimli bir tablo yaratılmıştır. Adres isimli tabloya ise Java kodu yardımıyla Adres nesneleri oluşturarak kayıt etme işlemi gerçekleştirilmiştir. Görüldüğü üzere NoSql Adres isimli bir Java kodu oluşturmaya gerek kalmadan bunu yapabilme gücünü kazandırmıştır. Java kodu üzerinde Mongo Jar kütüphanesinde yer alan BasicDBObject sınıfı yardımıyla istemiş olduğum nesne modellemesini gerçekleştirilmiştir.

Java kodumuzu derledikten sonra Şekil 46.'da Eclipse IDE konsolunun ve Şekil 47.'de MongoDb veri tabanının son halleri gösterilmiştir.



Şekil 46. NoSql İle MongoDb' de Şema Oluşturma, Tablo Oluşturma Ve Tabloya Veri Ekleme Eclipse IDE Çıktısı



Şekil 47. NoSql İle MongoDb' de Şema Oluşturma, Tablo Oluşturma Ve Tabloya Veri Ekleme MongoDb Görünümü

Yazılım geliştiriciler açışından oldukça kolay ve kullanılabilir bir belge yönelimli veri tabanıdır. Veri tabanı programlama diline ihtiyaç duyulmadan gerçekleştirilebilir. Yine bir Java sınıf modellemesine ihtiyaç duymamaktadır. Jar kütüphanelerinin içindeki sınıflar ve arayüzler sayesinde Java koduna yazılan kodlar ile veri tabanındaki şemayı ve tabloları oluşturmaya, tabloların içindeki veri işlemleri yapabilmek mümkündür.

2.5.2 NoSql Kullanarak MongoDb Üzerindeki Var Olan Veriyi Güncelleme İşlemi Çalışması

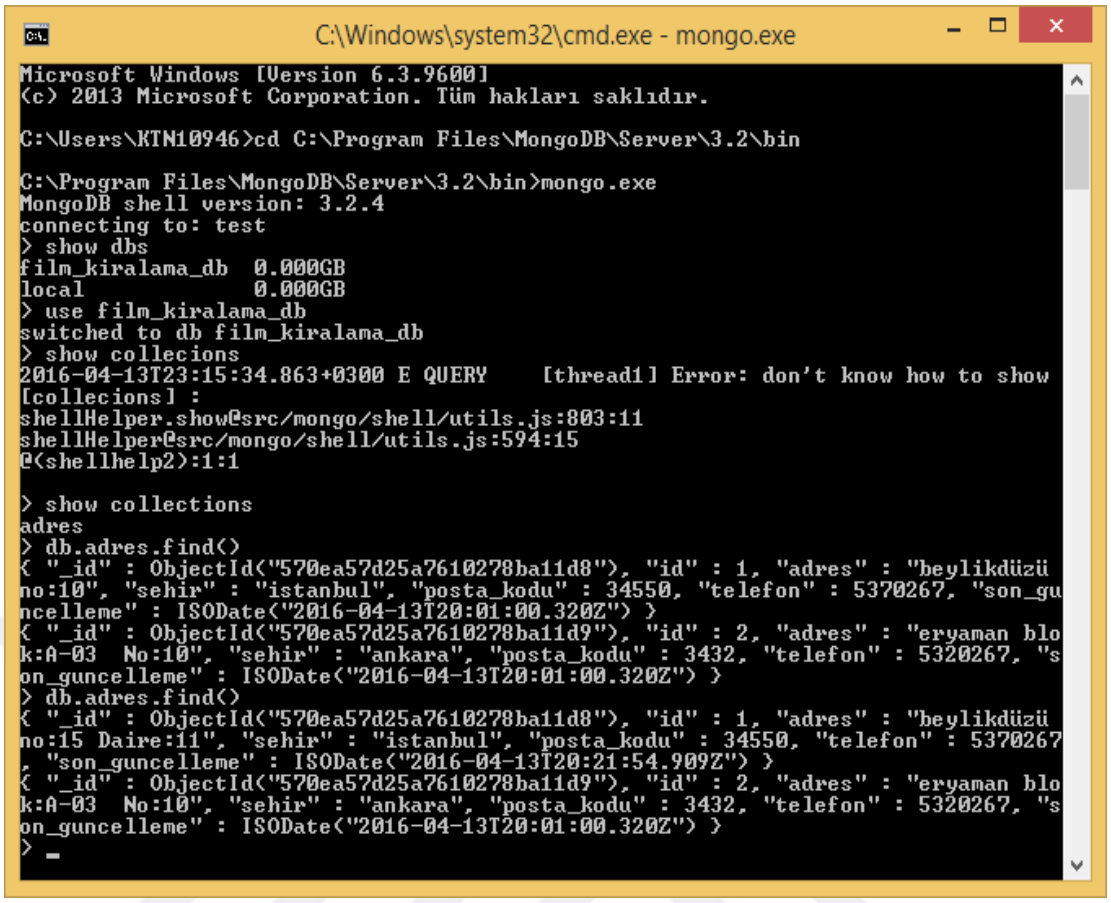
MongoDb üzerinde Java Maven projesi yardımı ile var olan tablodaki bir kaydı güncelleme işlemini yapan metodun kod parçası aşağıdadır.

```
public static void main(String[] args) {
        Date bugun = new Date();
        MongoClient mongo = new MongoClient("localhost");
        DB db = mongo.getDB("film_kiralama_db");
        DBCollection dbcol = db.getCollection("adres");
```

```
                BasicDBObject adres = new BasicDBObject().append("id", 1)
                                .append("adres", "beylikdüzü no:15 Daire:11")
                                .append("sehir",        "istanbul").append("posta_kodu",
34550).append("telefon",5370267).append("son_guncelleme", bugun);

                WriteResult sonuc = dbcol.update(new BasicDBObject("adres",
"beylikdüzü no:10"), new BasicDBObject("$set", adres));

                System.out.println(sonuc);
        }
```

Tablo 32. NoSql İle Veri Güncelleme Ve Güncellenen Veri Gösterme Java Kaynak Kodu

Var olan kaydı bulması için herhangi bir alanı yazmak yeterli olabilmektedir. Kodda Adres tablosundaki verilerden hangisinde güncelleme yapabilmesi gerektiğini bulabilmektedir. Yazılım geliştiricinin güncellenecek alan için sadece var olan bilgiyi girmesi ve güncellenecek veriyi girmesi yeterli olacaktır. Java kodunda herhangi bir Sql Script yazılmasına gerek yoktur. MongoDb çıktısını Şekil 48.'de gösterilmiştir.



Şekil 48. NoSql İle Veri Güncelleme Ve Güncellenen Veri Gösterme MongoDb Adres Tablosu Görünümü

2.5.3 NoSql Kullanarak Performans İşlemi Çalışması

NoSql çalışması için kullanılan MongoDb veri tabanına diğer 2 çalışmada olduğu gibi aynı sınıfı kullanarak ve aynı alanları kapsayan 50000 veri ekleme işini yapan metodun kod parçası aşağıdadır.

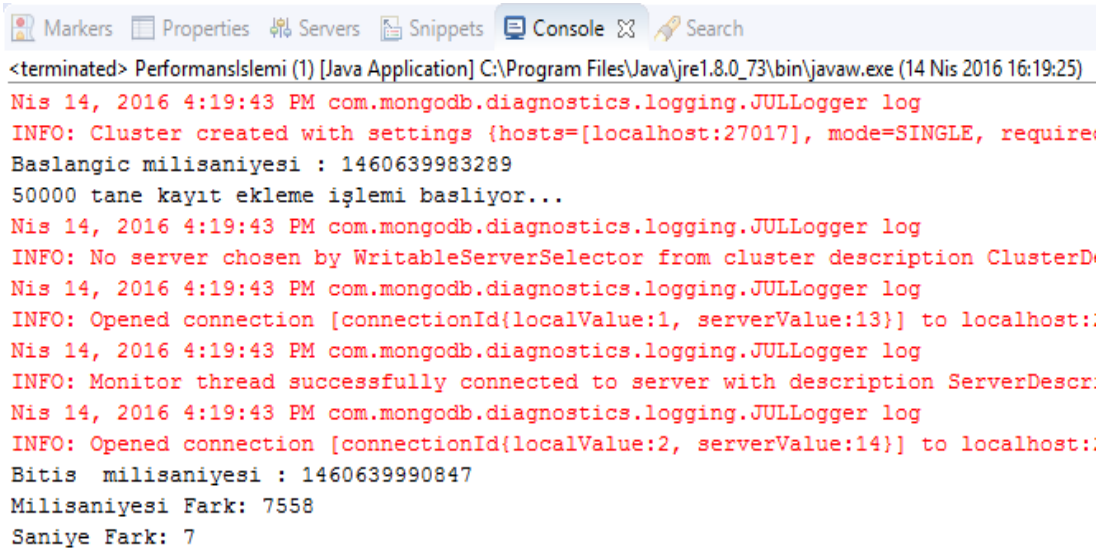
```
public static void main(String[] args) {
        Date bugun = new Date();
        MongoClient mongo = new MongoClient("localhost");
        DB db = mongo.getDB("film_kiralama_db");
        DBCollection dbcol = db.getCollection("adres");
        long baslangic = System.currentTimeMillis();
        System.out.println("Baslangic milisaniyesi : " +
System.currentTimeMillis());
```

```
                //50000 tane kayit ekleniyor.
                System.out.println("50000 tane kayıt ekleme işlemi basliyor...");

                for (int i = 100; i < 50100; i++) {
                        BasicDBObject adres = new BasicDBObject().append("id",
i).append("adres", "beylikdüzü no:10")
                                        .append("sehir",
"istanbul").append("posta_kodu", 34550).append("telefon", 5370267)
                                        .append("son_guncelleme", bugun);
                        dbcol.insert(adres);
                }

                long bitis = System.currentTimeMillis();
                System.out.println("Bitis milisaniyesi : " +
System.currentTimeMillis());

                long fark = bitis-baslangic;
                System.out.println("Milisaniyesi Fark: " + fark);

                long saniyeFark = fark /1000;
                System.out.println("Saniye Fark: " + saniyeFark);
        }
```

Tablo 33. NoSql Performans Çalışması Java Kaynak Kodu

50000 adet Adres nesnesi oluşturulmuştur.

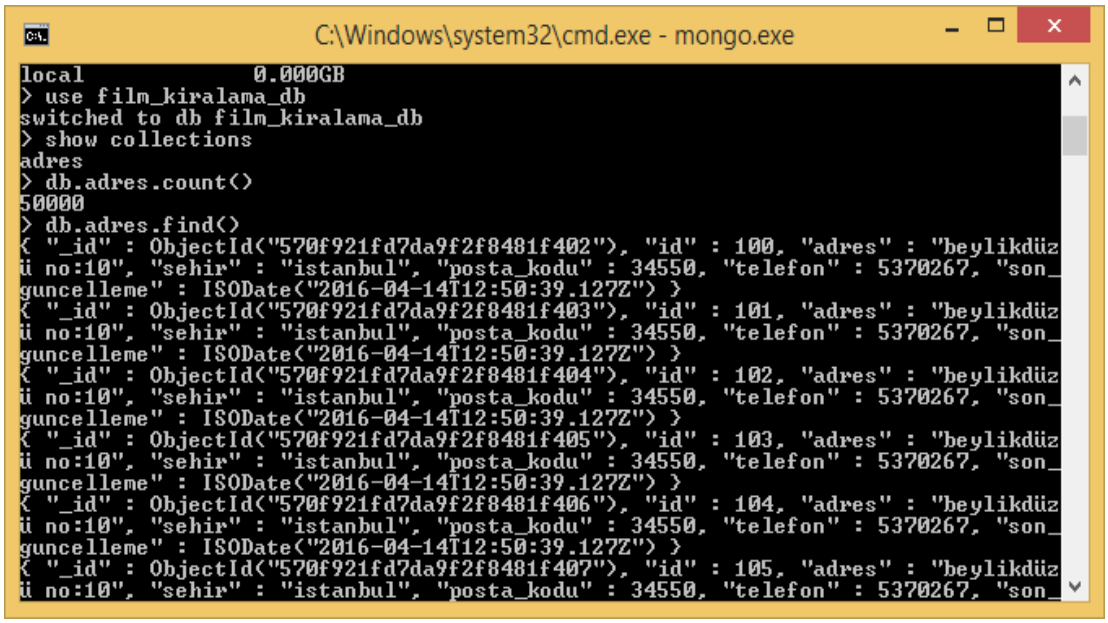
Kod Şekil 49.'da gösterilmiştir.



Şekil 49. NoSql Performans Çalışması Eclipse IDE Çıktısı

NoSql çalışmasında MongoDb ye 50000 kayıt 7 saniyede yaklaşık olarak 0,11 dakikada eklenmiştir.

MongoDb veri tabanında var olan tabloyu ve kaç kayıt olduğunu görmek için db.adres.count() komutunu yazmak yeterli olacaktır. Java kodu yardımıyla eklemiş olduğumuz verileri görmek istediğimizde ise db.adres.find() komutu kullanılmalıdır. Aşağıdaki şekilde bu 2 kod kullanılarak veri tabanına eklenen veri sayısı ve kayıtlar görülmektedir.[14]



Şekil 50. NoSql Performans Çalışması MongoDb Adres Tablosu Görünümü

NoSql çalışmasının diğer 2 çalışmaya göre oldukça hızlı olduğu görülmektedir. Diğer çalışmalar ile kıyaslandığında sonuçlar şu şekildedir.

Yapılan Çalışma	**Çalışma Süresi Saniye**	**Çalışma Süresi Dakika**
Jdbc	219	3,65
Hibernate	1355	22,5
NoSql	7	0,11

Tablo 34. NoSql, Hibernate, Jdbc Performans Çalışması Sonuçları

Diğer iki çalışma ile kıyaslandığında Hibernate'den yaklaşık olarak 193 kat, Jdbc'den ise yaklaşık olarak 31 kat daha hızlı olduğu görülmüştür.

3 KULLANILAN TEKNOLOJİLER

3.1 MySql Giriş Sürümü

MySql bir ilişkisel veri tabanı yönetim sistemidir. Veri tabanı yönetim sistemi "Veri tabanlarını tanımlamak, yaratmak, kullanmak, değiştirmek ve veri tabanı sistemleri ile ilgili her türlü işletimsel gereksinimleri karşılamak için tasarlanmış sistem ve yazılımdır.[9]

İlişkisel veri tabanı ise; çeşitli tabloların arasında ortak verilerden oluşturularak tasarlanmış veri tabanları olarak ifade edilebilir. Tablolar arasındaki ilişkiler çeşitli anahtarlar aracılığıyla gerçekleşebilir. Birincil Anahtar (Primary Key), Yabancı Anahtar (Foreign Key), Tekil Anahtar (Unique Key) kullanılarak ilişkiler oluşturulabilir.

MySql işlemlerini Sql adı verilen, yapılandırılmış sorgu dili denilen veri tabanından bilgi çekmeye sorgulamaya yarayan bir veri tabanı programlama dili ile gerçekleştirilmektedir.

Veri tabanı çalışmalarında kullandığım araç MySql Giriş Sürümü (Enterprise Edition) dür. Bu aracı ücretsiz olarak kullanıcı hesabı edinilerek indirilebilir.[10]

Kurulum nasıl yapılacağına ait kendi içindeki doküman okunarak sırasıyla adımları gerçekleştirerek kurulumun yapılması mümkündür. [11]

3.2 MongoDb 3.2 Versiyonu

MongoDb C++ dili ile yazılmış, ölçeklenebilir, doküman tabanlı, açık kaynak bir NoSql veri tabanı uygulamasıdır.[12]

Diğer eski yöntemlerden olan ilişkisel veri tabanlarına göre özellikle hız gerektiren işlemlerde seçilmektedir.

Bu tez çalışmasında NoSql işlemleri için MongoDb veri tabanı kullanılmıştır. Bu veri tabanını ücretsiz olarak internetten edinilebilir.[13] Ayrıca kurulum bilgilerini ve çalışma mantığını üretici firmanın sağlamış olduğu dokümanlardan bulmak mümkündür.[14]

3.3 Eclipse IDE Neon Versiyonu

Eclipse, açık kaynak kodlu bir geliştirme ortamıdır. Odak noktası Java ve Java teknolojileri ile çalışan araçlar olsa da bununla yetinmeyip C, C++, Java Script gibi programlama dilleri içinde uygun bir çalışma ortamına sahiptir.

Bu IDE' yi ücretsiz olarak kişisel bilgisayarınızın işletim sistemine uygun olarak internet üzerinden ücretsiz olarak indirebilirsiniz.[15] Bu tez çalışmasında kullanmış olan Eclipse Neon versiyonudur.

3.4 Kullanılan Java Kütüphaneleri ve Yönetim Araçları

3.4.1 Java Kütüphaneleri

Java programlama dili açık kaynak kodlu bir dil olduğumdan dolayı kütüphaneler bakımından oldukça zengindir. Kütüphane ise birçok sınıfı ve metodu bir arada bulunduran ve bir amaca hizmet etmek için programcılar tarafından geliştirilen kodlardır. Var olan daha önceden yazılmış hazır kütüphaneler yardımıyla yazılım geliştiriciler yapmak istedikleri işleri sıfırdan yapmak yerine daha hızlı ve verimli bir şekilde bu kütüphaneleri kullanabilirler.

Bu tezde kullanılmış olan kütüphaneler aşağıda belirtilmiştir.

JRE Sistem Kütüphaneleri, tüm Java uygulamalarını çalıştırmak için gerekli olan en az seviyedeki Java sistem kütüphanelerini içeren yapıdır. Kişisel bilgisayarınızda Java uygulaması çalıştırabilmek için JRE'nin bilgisayarınızda yüklü olması gerekmektedir.

Jdbc çalışması için oluşturulan Java projesi ile veri tabanı arasındaki bağlantı yönetmeye yarayan Jar kütüphanesi olarak "mysql-connector-java-3.0.17-ga-bin" isimli Jar kullanılmıştır.

Hibernate çalışması için Maven aracı üzerinde pom.xml koduna "mysql-connector-java 5.1.38" bağımlılık olarak verilerek "mysql-connector-java-5.1.38-mysql-connector-java-5.1.38.jar" isimli jar kullanılmıştır.

NoSql kullanılarak MongoDb veri tabanı üzerinde işlemler yapabilmek için oluşturulan katmanda Maven aracı üzerinden pom.xml koduna "mongo-java-driver 3.2.2" bağımlılığı verilerek "mongo-java-driver 3.2.2.jar" isimli Jar kütüphanesi kullanılmıştır.

3.4.2 Yönetim Araçları

Java proje geliştirirken açık kaynak kodlu olmasından dolayı birçok Jar kütüphanesi ve bağımlılık kullanılmaktadır. Bunları sağlarken bazen sorunlar ile karşılaşılmaktadır. Bunun için Java projesi geliştirme süreçlerini daha basit hale getirmek, bağımlılıkları kontrol altına almak ve belirli bir standart oluşturmak için Maven yönetim aracı kullanılmaktadır.[8]

Bu tezde Hibernate ve NoSql çalışmalarında Maven aracı kullanılmış olup, Maven projesi oluşturulmuştur.

4 DEĞERLENDİRMELER

4.1 Jdbc Değerlendirme Sonuçları

Bu tez de ortaya çıkarılan Jdbc çalışmasında yazılım geliştiricilerin veri tabanı programlama bilgisine sahip olması gerektiği açıkça görülmektedir. Bu bilgiye sahip olmadan Jdbc kullanarak Java kodları yazmak yeterli olmayacaktır.

Bu bilgiye sahip bir yazılım geliştirici ise rahatlıkta veri tabanından veri sorgulama, verileri kayıt etme, güncelleme, verileri silme gibi işlemleri yapabilir. Ayrıca ilişkisel olan tablolarda işlemler yaparken yeniden bir veri modellemesine ihtiyaç duyulmaksızın Java kodu üzerinde Sql Scriptleri yazarak hızlıca istenilen sonuçları elde etmek mümkündür.

Jdbc kullanılması gerektiğinde ortada bir veri tabanı olması gereklidir. Var olan şemalar ve tablolar üzerinden birçok işlemler yapabilmektedir. Bunu da Java Connector Jar kütüphaneleri ile sağlamak mümkündür.

Yapılan performans çalışmasında ise Hibernate'e göre 6 kat daha hızlı olduğu belirlenmiştir. NoSql den ise 193 kat daha yavaş bir performans göstermiştir. Yazılım geliştiriciler için veri tabanı programlama bilgisine sahip olanlar kullanım kolaylığı, karmaşık veri tabanlarını hükmetmek ve veri tabanı işlemlerinde hızlı olduğu Jdbc'yi tercih edebilirler.

4.2 Hibernate Değerlendirme Sonuçları

Hibernate çalışmalarında yazılım geliştiricilerin veri tabanı programlama bilgisine ihtiyaç duyulmamaktadır. Bu bilgiye sahip olmadan da veri tabanı işlemleri yapmak mümkündür. Ayrıca yine Hibernate in sorgulama dili olan HQL bilmek şart değildir. Bunları Java kodları yardımıyla yapmak mümkündür fakat geliştiriciler bu dili kullanmayı bilirse bazı noktalarda daha verimli çalışabilirler.

Herhangi bir veri tabanının olmasına gerek duymadan Hibertane Java kodu üzerinde oluşturacağı tabloya ait sınıfı modelleyerek, veri tabanı şeması ve tablosu oluşturabilmektedir. Bu kullanıcıya büyük kolaylık sağlamaktadır. Yine var olan bağlantı üzerinden tablo işlemleri yapabilmektedir. İlişkisel tablolarda da yine kod

üzerinden bu bağlantıları sağlayarak oluşturmak mümkündür. Veri tabanına bağlantı yine kullanılan veri tabanı tipine bağlı olarak Java Connector Jar kütüphaneleri ile sağlanabilir.

Performans çalışmasında ise Hibernate'in Jdbc den daha yavaş veri tabanı işlemleri yaptığı görülmüştür. Yaklaşık olarak 6 kat daha yavaş çalışmıştır. Veri tabanı programlama bilgisine sahip olmayan geliştirici, modelleme özelliğinden yararlanarak veri tabanı işlemleri yavaşta olsa Hibernate'i tercih ederek kullanabilir.

4.3 NoSql Değerlendirme Sonuçları

Yapılan NoSql çalışmasında yazılım geliştiricilerin herhangi bir programlama diline ihtiyaçları yoktur. NoSql'in kendine has olan bir sorgulama dili mevcut değildir. Sadece veri tabanında veri saklama şekli diğerlerinden farklıdır.

NoSql, Hibernate çalışmasında olduğu gibi veri tabanında şema ve tablo oluşturabilme özelliklerine sahiptir. Eğer bir veri tabanı var ise de bunun üzerinde veri tabanı işlemlerini yazılım geliştirici rahatlıkla yapabilir. Ayrıca yazılan Java kodları ile tablo alanları oluşturmak ve veri tiplerini belirlemek diğer 2 çalışmaya göre daha basit ve yalın biçimde yapılabilmektedir. Bağlantı diğer 2 çalışmada oluşturulan yapı ile benzer Java Connector Jar kütüphaneleri kullanılarak yapılmıştır.

Veri tabanı tablolarını modellemeye de ihtiyaç duymamaktadır. Oluşturulan nesneler yardımıyla belirlenen modelin nasıl olması gerektiğini anlayarak, kodlanan biçime göre veri tabanında tabloyu oluşturabilmektedir.

Performans çalışmasında ise diğer 2 uygulamaya göre oldukça hızlı veri tabanı işlemleri yaptığı görülmektedir. Yapılan aynı işlemler neticesinde Jdbc'den 31 kat, Hibernate'den ise 193 kat daha hızlı veri tabanı işlemleri yaptığı görülmüştür. Özellikle çok büyük verilerin bulunduğu veya çok büyük veriler ile işlem yapılırken, yazılım geliştiricilerin veri tabanı programlama bilgisine ihtiyaç duyulmadan ve basit kullanım kolaylığından dolayı NoSql tercih edilebilir.

Tez kapsamında yapılan çalışmalar neticesinde yazılım geliştiriciler açısından değerlendirme sonuçları aşağıda ifade edilmiştir.

	JDBC Uygulamaları	**Hibernate Uygulamaları**	**NoSql Uygulamaları**
Varolan Veri Tabanına İhtiyaç	Var	Yok	Yok
Veri Tabanı Kodlama Bilgisi İhtiyacı	Var	Yok	Yok
Kendine Has Veri Tabanı Dili	Yok(Sql Scriptler yazılabilir)	HQL	Yok
Veri Tabanın da Veri Saklama Şekli	Satır(row) bazlı	Satır(row) bazlı	Doküman(document) ve BSON bazlı
Veri Tabanı Tablosu Modelleme	Yok	Var	Yok
Java Sınıfı(Class) Modelleme	Yok	Var	Yok
Performans	31* (X)	(X)	193 * (X)
Hangi Durumda Tercih Edilmeli	*Karmaşık veri tabanı yapısına sahip uygulamalarda.	*Performansın gözardı edildiği uygulamalarda.	*Çok büyük veriler ile işlemler yapıldığında. *Performansın önemli olduğu çalışmalarda.
En Büyük Avantajı	*Karmaşıklığı kolayca yönetebilir. *İstenilen sonuçlar hızlıca elde edilebilir.	*Veri modelleme yeteneği. *Veri tabanı programlama bilgisine ihtiyaç olmaması.	*Çok performanslı işlemler yapması. *Dikey büyüme yerine yatay büyümeyi benimsemesi. *Kullanım kolaylığı.

Tablo 35. Yazılım Geliştiriciler Açısından Değerlendirme Sonuçları

5 SONUÇ

Tez çalışmasında Jdbc, Hibernate ve NoSql kullanarak birbirleri arasındaki farklar ve benzerlikler uygulamalar ile ele alınmış olup, yazılım geliştiriciler açışından avantaj ve dezavantajlarının neler olduğu anlatılmıştır.

Yazılım geliştiricilerin sahip olduğu teknik yeterliliğe göre hangi durumda hangi seçimin yapılması gerektiği ortaya çıkarılmıştır. Üç çalışmanın birbirilerine göre avantajları ve dezavantajlar olduğu görülmektedir. Geliştiricilerinin bu kıstasları göze alarak yazılım geliştirme süreçlerinde tercih etmesi gerekmektedir. Aksi takdirde olumsuz sonuçlar doğrulabilmekte zaman ve maliyet kaybı yaşanmasına yol açmaktadır.

Tez kapsamında ele alınan performans çalışmaları bizlere NoSql'in performans açışından diğer kullanımlara göre çok daha hızlı olduğunu göstermiştir. Aradaki farklar ve benzerlikler çok daha detaylı olarak ele alınarak geniş çaplı araştırmalar yapılabilir.

Yapılan çalışma ve uygulamalar ile yazılım geliştiricilerin geliştirecekleri uygulamalarda seçim yaparken dikkat edilmesi ve göz önünde bulundurulması gereken kriterlere dikkat çekerek başarıya ulaşmalarının yolunu göstermiştir.

KAYNAKLAR

[1] Wikipedia, (Mart 2016) http://tr.wikipedia.org/wiki/Java_(programlama_dili)

[2] M. Özdemir, Java Tabanlı Bir Elektronik Seçim Modeli Uygulaması, Yüksek lisans Tezi, Beykent Üniversitesi, İstanbul 2008

[3] A.Kiraz, Java Programlama Dilinin Web Üzerinden Sunumu, Yüksek Lisans Tezi, Sakarya Üniversitesi, Eylül 2006

[4] SUN, (Ocak 2016) http://www.sun.com/java/

[5] Java History, http://javahistory.blogspot.com.tr/

[6]JavaSETechnologies-Database(Mart2016) http://www.oracle.com/technetwork/java/javase/jdbc/index.html

[7] A. Tallal Aladily, The performance-Wise Comparison of The Most Widely Used NoSql Databases, Msc Thesis ,Kadir Has Univesity, June 2015

[8] Apache Maven Project, (Ekim 2015) https://maven.apache.org/

[9] MySQL, (Mart 2016) https://tr.wikipedia.org/wiki/MySQL

[10] MySQL Downloads, (Aralık 2015) https://www.mysql.com/downloads/

[11] MySQL Documentation, (Aralık 2015) http://dev.mysql.com/doc/

[12] Node JS Türkiye, (Mart 2016) http://www.nodejstr.com/]

[13]Run and Manage MongoDB with CloudManager,(Nisan 2016) https://www.mongodb.org/downloads#production

[14]The MongoDB 3.2 Manual, (Nisan 2016) https://docs.mongodb.org

[15] Eclipse, (Eylül 2015) https://eclipse.org/downloads/

[16]UseDatabaseCommands,(Nisan2016) https://docs.mongodb.org/manual/tutorial/use-database-commands/

[17] Mvn Repository, (Şubat 2016) http://mvnrepository.com/search?q=jdbc

ÖZGEÇMİŞ

26 Ocak 1984 tarihinde Kastamonu İli Taşköprü İlçesinde doğdum. İlköğretim eğitimimi Atatürk İlk Öğretim Okulunda aldım. Daha sonrasından Taşköprü Süper Lisesini okuduktan sonra Beykent Üniversitesi Bilgisayar Mühendisliği bölümünde üniversite eğitimime başladım ve 2010 yılında mezun oldum. 2014 yılında Beykent Üniversitesi Bilgisayar Mühendisliği Tezli Yüksek Lisans bölümünde yüksek lisans eğitimime başladım ve şu anda devam etmekteyim.

Ferhat Cem Cihan