İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Ham Data Kullanılarak Azure Altyapısında Verinin İşlenmesi, Depolanması ve Veri Optimizasyon Yöntemleri

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Doğukan Ekinci

Proje Danışmanı: Dr. Öğr. Üyesi Mansur Alp TOÇOĞLU

Ocak 2024

Ham Data Kullanılarak Azure Altyapısında Verinin İşlenmesi, Depolanması ve Veri Optimizasyon Yöntemleri

Öz

Bu Projede, Ham Verinin İşlenmesi ve Depolanması için Veritabanı Kurulumunun Yapılması Üzerine Çalışma Yapılacaktır. Öncelikle Ham Verinin Azure Platformuna çekilmesi ve Synapse Altyapısına Olan Bağlantının Sağlanması için Azure Data Factory'deki Pipeline ve Aktivite Süreçlerinin Kurulumunun Yapılması, Ham Data Üzerinden ETL Süreçlerinin Oluşturulması Verinin Nasıl Optimize Edileceğinin Saptanması ve Senaryoların Oluşturulması Ardından Staging, ODS Ve Model Katmanlarının Modellenmesi Verinin Bu Katmanlar Arasında İşlenme Süreçlerinde Oluşturulacak View ve Stored Procedure Katmanlarının SQL Veri Dilinde Yapılması ve Karmaşık Veri İşleme Süreçlerinin Yönetilmesi ve Validasyon İşlemlerinin Yapılması.

Anahtar Sözcükler: Veri işleme, veritabanı, katman, sorgulama dili, ETL

Data Processing, Storage and Data Optimization Methods in Azure Infrastructure Using Raw Data

Abstract

In this project, work will be done on establishing a database for processing and storing raw data. First of all, setting up the Pipeline and Activity Processes in Azure Data Factory to pull the Raw Data to the Azure Platform and Ensuring the Connection to the Synapse Infrastructure, Creating ETL Processes on the Raw Data, Determining How to Optimize the Data and Creating Scenarios, Then Modeling Staging, ODS and Model Layers, Transferring the Data to These Layers Creating the View and Stored Procedure Layers to be Created in the Processing Processes in SQL Data Language, and Managing Complex Data Processing Processes and Performing Validation Procedures.

Keywords: Data processing, database, layer, query language, ETL

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

OLTP	Çevrimiçi Hareket İşleme
ETL	Extract, transform, load
DBMS	Database Management System

Bölüm 1

Giriş

Günümüzde kişiler, şirketler ve kurumlar ürettikleri veya üretilen bilgileri ihtiyaçları dahilinde saklama, yönetme, erişme ve analiz etme gibi ihtiyaçları mevcuttur. Kullanıcı herhangi bir bilgiye mesafe ve zaman ayrımı olmaksızın erişmek ister. Bu gibi ihtiyaçların akabinde çeşitli çözümler üretilmiştir. Tarihsel açıdan baktığımızda ilk çözümlerin dosya temelli sistemler olduğu görülmüştür. Dosya temelli sistemler, büyük ölçekli verilerin işlenmesinde dağıtılmasında ve depolanmasında zorluklar çıkarınca dosya temelli sistemden veritabanı denilen bir sistemin oluşturulmasıyla farklı isteklere de cevap verilebilmiştir. Farklı isteklere cevap verirken şirketler bu kuramdan hemen vazgeçmemiştir. Veritabanı sistemlerine entegre edilmiş ve bulut ortamında saklanması ve işlem görmesi devam etmektedir. Günümüz de şirketler bulut ortamında depoladığı ve verilerinin girdilerini oluşturduğu excel gibi dosya türleri mevcuttur. Verinin ilk girildiği excel dosyaları Azure Veritabanı gibi sistemlerde veri pipeline'ı oluşturularak önemli bir veri işleme sürecinin başlaması sağlanmış olmaktadır. Şirketler, kurumlar veya tüzel kişiler bu tarz veritabanları üzerine sistemlere çalışırken zamanla gelişen ihtiyaçlar ölçüsünde de bu sistemleri aynı zamanda geliştirmişlerdir. Veritabanı sistemleri günümüzde kullanan şirketler dahilinde özelleşmiş daha önce belirlenmiş isterler ölçütünde düşünsel olarak geliştirilmiş doğruluğu, çalışabilirliği şirketçe ihtiyaç haline getirilmiş katmanlar ve kurallar bütünü haline gelmiştir. Bu kurallar bütünü şirketlerin iş yapış biçimleri kıstasında gelişmiş ve kabul edilmiştir. Hangi katmanlarda verinin işlenmesi güvenlik katmanları view ve stored procedure oluşturulması gibi somutlaştırılmış verinin işlem basamaklarının oluşturulmasında önemli parametrelerdir. Bu önemli parametreler verinin işlenmesinde zaman içerisinde şirketlerin isterleri kapsamında gelişmiş ve

somut kurallar altında kullanılmaktadır. Bu çalışmada bu kurallar dahilinde işlemleştirilmiş insan kaynakları veri tabanı sisteminde de kullanılmaktadır.

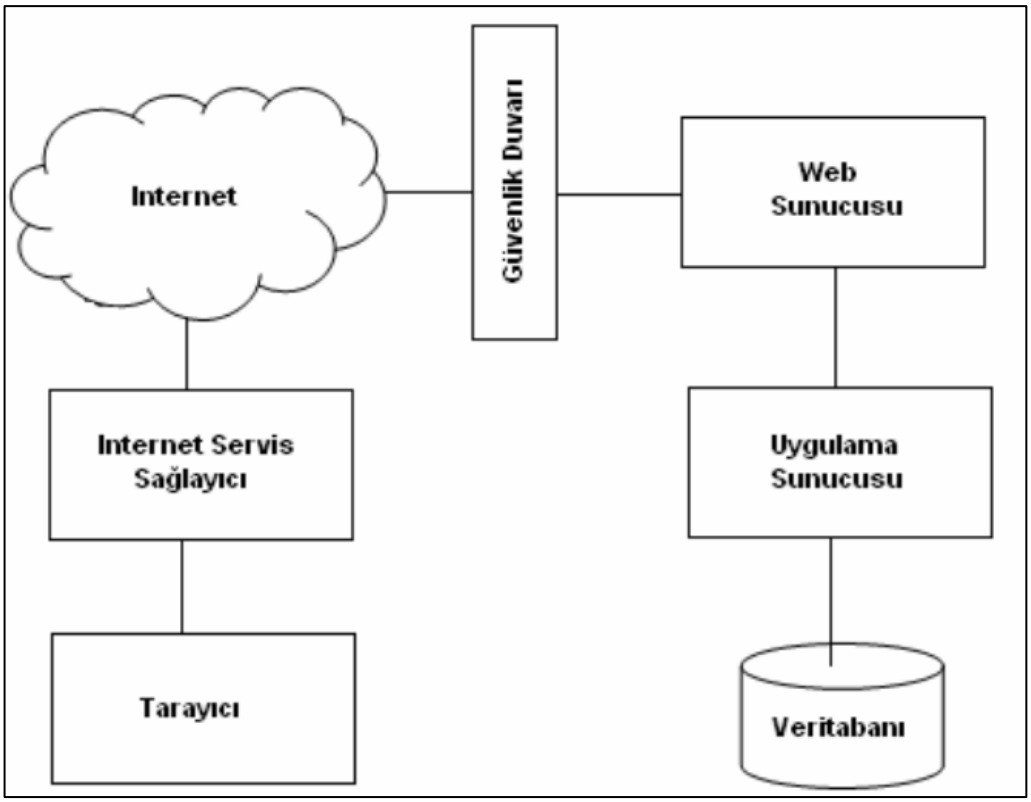
Bölüm 2

Genel Kavramlar

Bilginin işlenmesi saklanması ve gelecek nesillere iletilmesinde veritabanlarına önemli görevler düşmektedir. Şirketlerin, kurumların veya insanların iş yapışlarını ve hayatlarını kolaylaştıran veritabanları günümüzde yaygın şekilde kullanılmaktadır. Bu kısımda tanımları ve neden kullanılıyor gibi soruların cevabı aranacaktır.

2.1 Veritabanı Kavramı

Veritabanı verilerin bilgisayar sistemlerinde elektronik olarak depolandığı yerdir. Günümüzde bilgiler veriler sıklıkla dinamik bir şekilde çevrimiçi saklanır. Veritabanları saklanılan verinin bir kez tanımlanan ve ardından tanımlanan kullanıcılar tarafından erişilebildiği verilerin depo alanı olarak görülebilir. Veriler bu tip veritabanlarında saklanabilir, silinebilir, değiştirilebilir. Veritabanları yapıları itibariyle birkaç özelliği bünyesine barındırmalıdır. (Watt ve Eng, 2012) Öncelikle mantılıklı ve tutarlı olmalıdır. Her veri belli bir alanda depolanmalıdır. Her veritabanı belli bir amaç için oluşturulmalı ve gerçek dünya verilerini içermelidir. Veritabanı için Immon dört temel özellik tanımlamıştır. Bir veritabanı tasarlamak, veritabanında depolanacak veri tiplerini, yapılarını ve kısıtlamalarını belirlemeyi de içermektedir. (Prados, vd., 2005)



Şekil 2.1: Veritabanı ve Ağ bağlantısının bileşenleri (Önder, E., 2005).

2.1.1 SQL

Tarihsel gelişimi açısından bakıcak olursak Edgar F. Codd tarafından 1970 yılında ilişkisel veri modeli olarak önerilmiştir. Ardından 1970'lerin ortasında Don Chamberlin ve Raymond Boyce tarafından deflaratif sorgu dili geliştirilmiştir. (Florescu ve Fourny, 2013). SQL dilinin ilk adımları olan bu gelişmeler 1986'da American National Standards Institute tarafından ve 1987'de ISO tarafından da SQL'in resmi bir standart olarak kabul etmesiyle gelişimine devam etmiştir. SQL (Structured Query Language) ilişkisel veritabanı üzerindeki ihtiyaç duyulan işlemleri gerçekleştirmek için kullanılan yapısal bir sorgulama dilidir. Sorgulama dili ihtiyacı büyük miktarlarda veri hacimlerinin oluşmasıyla, verilere erişimde sorunlar ortaya çıkmasıyla beraber artmıştır sorunlardan biride yüzlerce tablonun bulunduğu veritabanına erişimdi. SQL bu ve bunun gibi sorunlara çözüm amacıyla oluşturulmuştur.

2.1.1.1 Basit SQL İfadeleri

Select: Veritabanındaki tabloalardan kayıtların çekilmesini sağlar.

Update: Bir tabloda ki kaydın güncellenmesini sağlar.

Delete: Kaydın silinmesini sağlar.

Insert: Yeni kaydın eklenmesini sağlar.

Create View: View görünümünün oluşturulmasını sağlar.

Create Proc: Saklı yordamın oluşturulmasını sağlar.

Truncate Sorgusu: Tabloyu silmeden içinde ki verinin silinmesini sağlar.

2.1.2 Sorgu Optimizasyonu

Veritabanlarını yönetmek amacıyla çeşitli sorgulamaların yapılması gerekmektedir. Veritabanı yönetim sistemi verilen sorguyu işlemek ve çıktı elde etmek için birden fazla erişim yolu vardır. Her erişim yolu çıktılar açısından eşdeğerdir ama maliyet ve sorgunun çalışması için gereken süre optimizasyon problemini de beraberinde getirmektedir.(Ioannıdıs, 1996). Problemin çeşitlerine göre birden fazla çözüm yolu oluşturulmaktadır. Bazı çözüm yöntemleri özellikle yüksek oranda verinin işlendiği veritabanların da iş yükünün azaltılmasında ve Azure ortamında işlenmesinde önemli ölçüde fayda/yarar sağlamaktadır.

2.1.3 Sorgu Optimizasyon Teknikleri

Örneğin Veritabanında tabloları boşaltmak için delete komutu yerine truncate komutu kullanımı daha iyi bir sonuç verecektir. Geçici tablolarda index oluşturulmalı veritabanı performansı açısından önemlidir. Alt sorgu kullanmak yerine aynı çıktıyı alınabilecek daha basit sorgular kullanılmalıdır. Bunlar gibi örneklerle sorgular optimize edebilebilir.

2.2 İlişkisel Veritabanı Nedir?

(Jatana vd., 2012) İlişkisel veritabanı E.F Codd tarafından 1970 yılında icat edildi. İlişkisel veri modelini destekleyen sistemlerin gelişimiyle elektronik ortamda paylaşılan verinin erişimine veritabanı sistemleriyle erişimin mümkün olması anlamına gelmektedir.(Haskin, 1982) Bir ilişkisel veritabanı, veri öğelerinin tablo halinde saklandığı, düzenlendiği ve veriye birçok farklı şekilde erişilebileceği veya yeniden bir araya getirilebileceği bir modeldir.

2.2.1 İlişkisel Veritabanının Temel Özellikleri

2.2.2 Normalizasyon

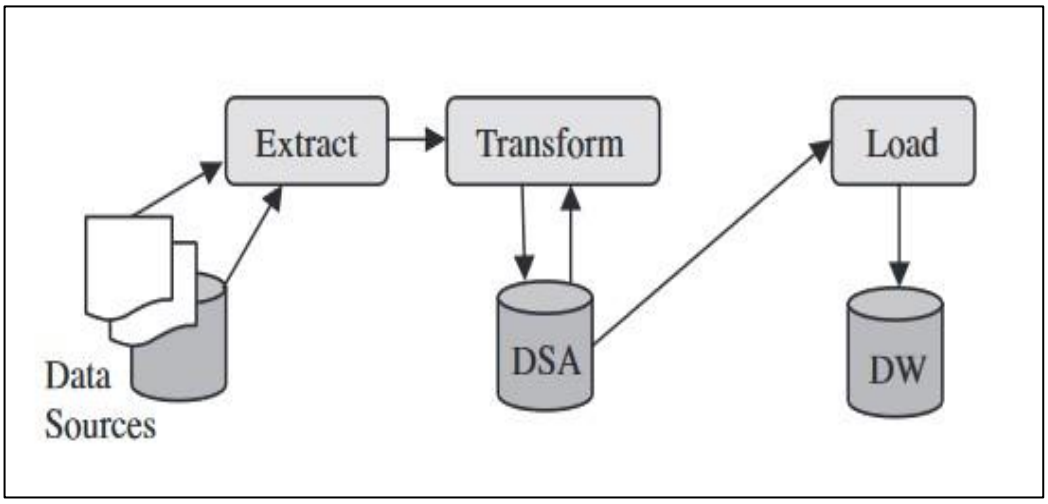
Normalizasyon, elimizde olan bir veri topluluğunu birbiriyle ilişkili veriler halinde daha küçük tablolara ayırma bölme işlemidir. Normalizasyon yapılmasında ki temel amaçlar veri tekrarının önlenmesi, veri tutarsızlığını minimilize etmek ve verideki sistemsel hataları gidermektir.(Uragun, 2011).

2.2.3 Denormalizasyon

Veritabanı tasarlarımı kavramsal, mantıksal ve fiziksel aşamalarını içerir. Aslında öncelik verinin tasarım aşamasındayken önce normalleştirilmesi ardından performans gibi sebeplerden dolayı verinin denormalize edilmesidir. Denormalizasyonda performans artışı ve veri üzerinde yapılacak bazı işlemleri basitleştirmek için kullanılan bir işlemdir. Yazılım sektöründe her tablo üzerinde ayrı ayrı işlem yapmak özellikle sıkışık zamanda performansı etkilediği gibi stresi de arttırmaktadır. Denormalize edilmiş tablolarda işlem yapmak zaman tasarrufu yapmaya olanak sağlamaktadır. Veri karmaşıklığı giderilmesi akabinde de zamansal ekonomiklikte sağlanmaktadır.

2.2.4 ETL

ETL, extract, transform ve load işlemlerinden oluşan bir veri entegrasyonu işlemidir. Bir veri tabanı projesinde kaynaktan gelen verinin çıkartılması, istenilen formata tabloların dönüştürülmesi, değiştirilmesi ve veri ambarına veya istenilen yere yükleme işlemlerinin kendisidir. (extraction, transformation, loading). ETL sürecine girecek veri yapılandırılmamış, yapılandırılmış yada yarı yapılandırılmış formatta olabilir. İşlemden geçmiş veri genellikle ilişkisel veya yarı yapılandırılmış bir formatta olabilir yada veri ambarları OLTP, istenilen herhangi bir formatta dosya, web sayfaları, çeşitli belge türleri veya hatta akış halinde gelen veriler olabilir. Veri işleme sürecinde kurum yada şirket veride istenilmeyen şeylerin işlenmesini istemez bu süreçte de kirli verinin temizlenmesi sağlanmaktadır. Temizleme ve dönüştürme işleminni akabinde kurulan bir veri ambarında veriler iş birimi tarafından istenilen aralıklarla ETL sürecinin yenilenmesi yapılmaktadır. Azure altyapısında triggerlanma işlemi genellikle gece saatlerinde yapılmaktadır. Gece saatlerinde yapılmasının temel amacına sisteme minimum yük getirilmesi amaçlanmış olur. Trigger süreciyle veri güncellenmesi süreci hem tekrarlanmış olmakta hemde yaşanılan hatalar düzeltilene kadar sadece bir günlük veri kaybı yaşanmış olmaktadır. (Mukherjee ve Kar, 2017)



Şekil 2.2: ETL süreçleri örnek gösterimi

2.2.4.1 Initial ETL

Genellikle veri ambarı hayata geçmeden önce sadece bir kere yapılan, iş zekası ve analizciler (Power Bi) tarafından kullanımına hazır hale gelmesi için yapılan ve verinin ilgili kısımlarının alındığı süreçtir. Nadir durumlarda yapılmaktadır.

2.2.4.2 Incremental ETL

Veri ambarını aşamalı olarak yenilemek ve güncel olarak tutmak için yapılır. İş birimi tarafından belirlenen saat ve günlerde incremental olarak data getirilir ve böylelikle çıktılar güncel olur. Getirilen datalar düzenlenmek için kullanılmaktadır. Yapılmasında ki başlıca amaçları sıralamak gerekirse öncelikle daha hızlı verinin işlenmesi amaçlanmaktadır. ETL süreci zaman ve iş yükü açısından maliyet bindirmektedir. Etkileşime daha az boyutta veri gireceğinden verinin işlenmesi süreci daha az vakit alır ve firmaların tasarruf etmesini sağlar. Diğer bir sebep ise risk yönetimidir. İşlem yapılacak veri miktarı azalacağından risk potansiyelide azalmaktadır. Bazen verinin yüklenmesi başarısız olabilmektedir. Verinin yüklenmesi başarısız oldupundan düzeltme işlemleri yapıldıktan sonra yüklenmesi daha az vakit almakta ve hatayı bulma süreci daha az vakit almaktadır. Gelen veriyi doğrulamakta böylelikle daha az zaman almaktadır. Diğer bir artısı da performans açısından daha tutarlı olmasıdır. Incremental ETL de sadece veri değişiklikleri durumunda veri transferi yaparak performansı maksimum seviyede tutması amaçlanmaktadır.

2.2.5 Dimension Tablo

Bir dimension tablosundaki nesnelerin her örneği hakkında açıklamaları ayrıntıları içerir. Özellik olarak incelendiğinde sayısal veriler içermemektedir. Bir işletme hakkında detaylı bilgi içerdiğinden tanımlayıcı öznitelikleri depolamak için dimension tablo kullanılmaktadır. Fact tablo ve Dimension Tablo birbiriyle ilişkilidir. Fact tablo olaylara karşılık gelirken dimension tablosu nesnelere, eşyalara, insanlara karşılık gelmektedir. veri ambarı modeli tasarlanırken temel amaç, basit ve mümkün olan en hızlı sorgu için yapmaktır. Bu yüzden dimension tablo tasarımında dimension tablolar elden geldikçe sade tutulmalıdır.

2.2.6 Fact Tablo

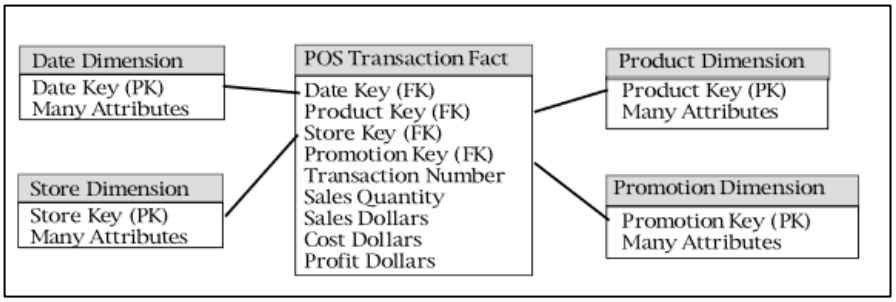
Fact tablo yapısı gereği belirli bir iş süreci hakkında niceliksel verileri içeren bir tablodur. Genellikle bir iş öğesini, bir işlemi veya işi analiz etmek için kullanılabilecek bir olayı temsil etmektedir. Fact tablo çok sayıda satır içerirken daha az sütun içermektedir. Fact tablo sayısal veriler içermekle beraber tarih bilgisi içeren tabloların da bir çoğu fact tablo olmaktadır. (Patel ve Patel, 2012)

2.2.7 Şema Nedir?

Veritabanı şeması verilerin nasıl depolandığını, birbirleriyle olan ilişkilerini ve kendi içerisinde barındırdığı kuralları gösteren bir varlık ilişki göstergesidir. DBMS sistemi içerisinde bulunur. Veritabanı güvenliği için farklı şemalar arasında farklı erişim izinleri ayarlanabilir böylece izinsiz girişler engellenmiş olmaktadır. Veritabanı içerisinde ki verileri farklı şemalar arasında aktarmamızı da sağlamış olur böylece daha kolay veri transferi ve düzenlenmesi sağlanmış olur.Ayrıca şema kullanımı veritabanı optimizasyonunda anahtar rol oynamaktadır. Örneğin staging, ODS ve Model katmanlarının kullanımı.

2.2.7.1 Kar Tanesi Şeması

Veriambarı mimarisi tasarımında kullanılmaktadır. Analitik sorgu yapmak için oluşturulmaktadır. Dimension ve Fact tablo olarak ayrılmaktadır. Genellikle merkezinde Fact tablo olmakta çevresinde dimension tablolar olmaktadır.(O'Neil vd., 2009)

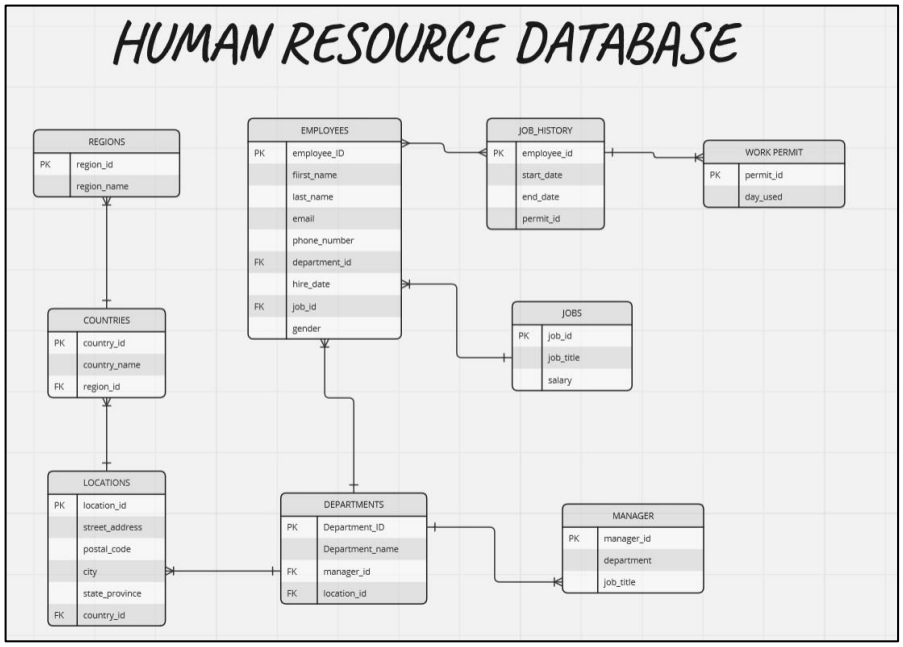


Şekil 2.3: Örnek kar tanesi şeması

Bölüm 3

Veri ve Yöntem

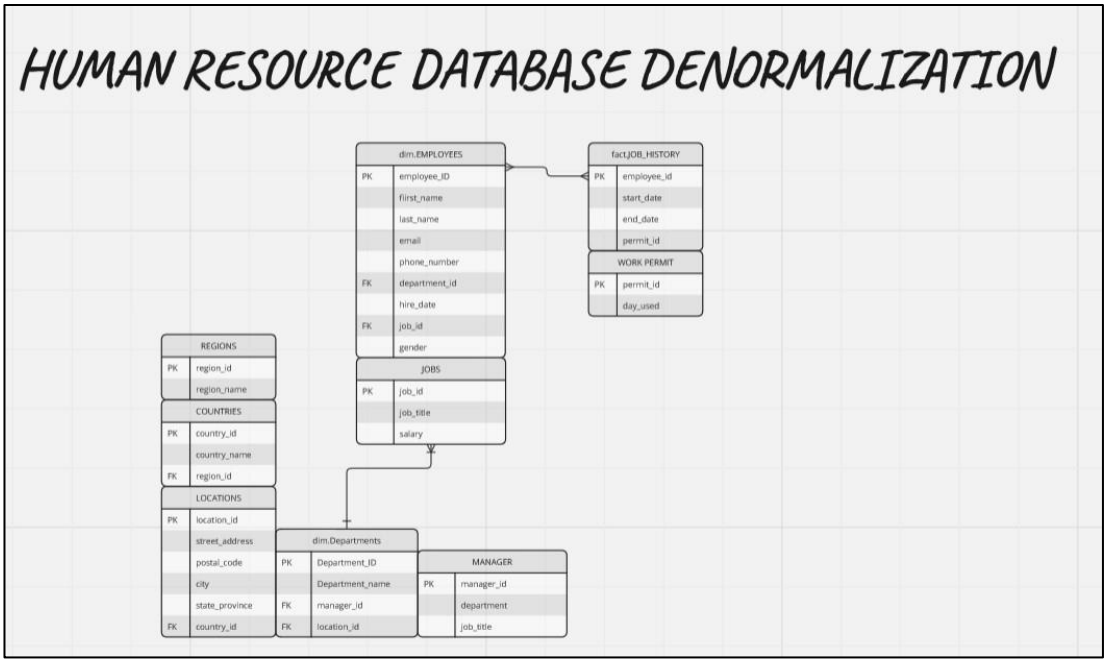
Bu projede, Microsoft'un Azure uygulaması kullanılmıştır. Bu çalışma için oluşturulmuş Human Resource verisi kullanılmıştır. Gizlilik ihlal olmaması açısından herhangi bir şirket verisi kullanılmamıştır. Veri Mockaroo web sitesinde rastgele olacak şekilde oluşturulmuştur. Verilerin oluşturulması sürecinde senaryoya en uygun şekilde hareket edilmiştir. Veri tipleri, veri miktarı senaryoda verinin SQL Server Management Studio Management Studio'ya aktarımı için sorun yaşanmaması maksadıyla aynı tipte olacak şekilde ayarlanmıştır. Gerçeğe en yakın olacak şekilde veri işleme planı oluşturulmuştur. Verilerin işlenmesi sürecinde farklı senaryolarla hangi yöntemlerinin izlenmesi gerektiği belirtilmiştir. Veri işleme sürecinde hatalarla karşılaşılmaması açısından veriler olabilecek en iyi şekilde hazırlanmıştır. Veri işleme sürecinde noktalama işaret hatası, Primery Key'in null ifade gelmesi veya tutarsız veri gelmesi gibi hatalarla karşılaşılmaması açısından veri oluşturma sürecinde göz önünde bulundurulmuştur.



Şekil 3.1 : İnsan Kaynakları Veritabanı

3.1 Denormalizasyon İşlemi

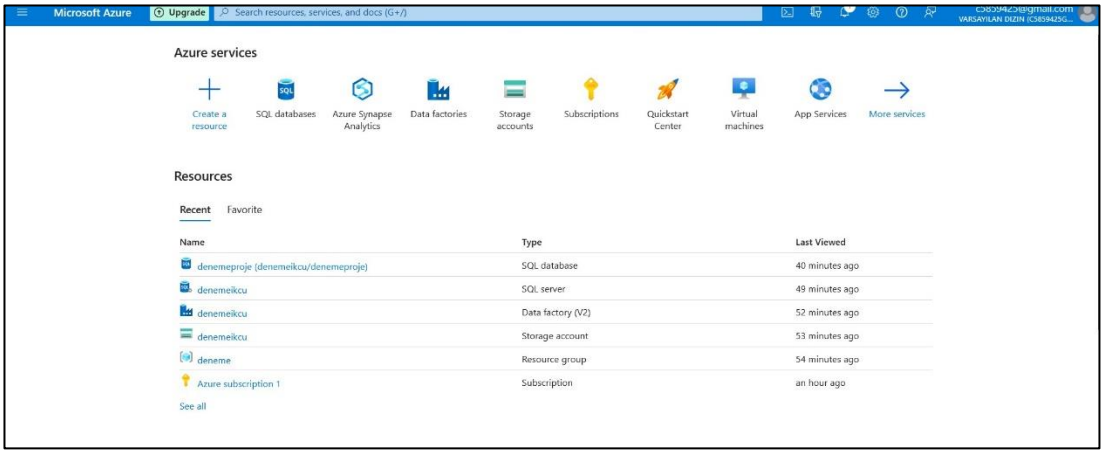
Human Resource Veritabanında tablolar halinde gelen veri denormalizasyon işlemine tabi tutuldu. Dokuz farklı başlıkta tablolar halinde gelen veri üç ana tablo halinde denormalizasyon işlemine tabi tutuldu. İşlemin yapılmasında temel amaç performans kaygılarının giderilmesi ve veri üzerinde yapılacak birleştirme işlemleriyle diğer katmanlarda yapılacak işlemleri kolaylaştıran basitleştimek amacıyla bir işlem yapılmıştır. Çalışmada ayrı ayrı her tablo üzerinde çalışma yapmak zaman maliyet açısından sorun çıkarması ve projenin daha anlazılır olmasını engellemektir. Denormalize edilen veriler üzerinde dokuz olan tablo sayısı üçe indirgenerek zamandan ve her tablonun işlem süresinden tasarruf edilerek maliyetler minimalize edilmiştir. Denormalizasyon sonucunda işlem görecek veri sayısı artacağından aynı zamanda sorgu performansı açısından da önemle dikkat edilmelidir. Alt sorgu kullanmaktan özellikle çekinilmiştir. Buda sorgu performanda açısından pozitif bir yaklaşımdır.



Şekil 3.2: İnsan Kaynakları Veritabanı Denormalizasyon İşlemi Sonucu

3.2 Veri İşleme Süreci

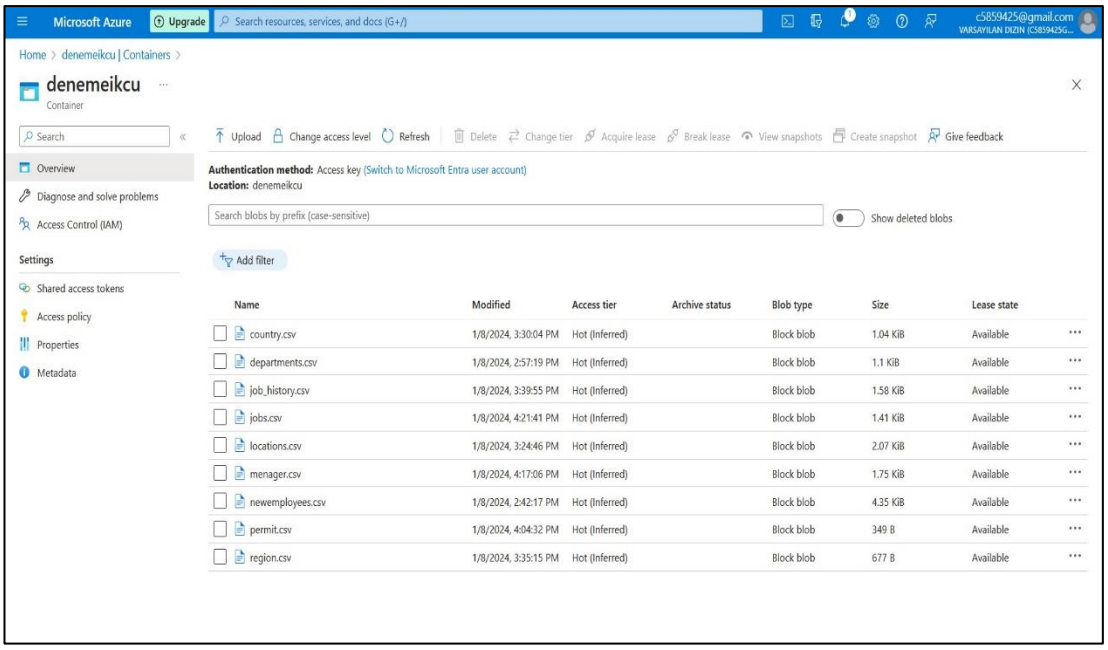
Verinin işleme sürecinde öncelikle verinin işlenme patikalarının altyapısı oluşturulmalıdır. Öncelikle SQL database kurulumu yapıldı.Kurulumdan sonra Datafactory'den Storage Account kurulumu Storage Account da ise csv dosyalarının aktarılacağı Container kısmı oluşturulmuştur. Bu arada firewall ayarları verinin güvenli bir şekilde güncellenip saklanması sağlanmıştır.



Şekil 3.3: Azure servisi oluşturulan kaynaklar

3.2.1 Azure Altyapısına Verinin Yüklenmesi

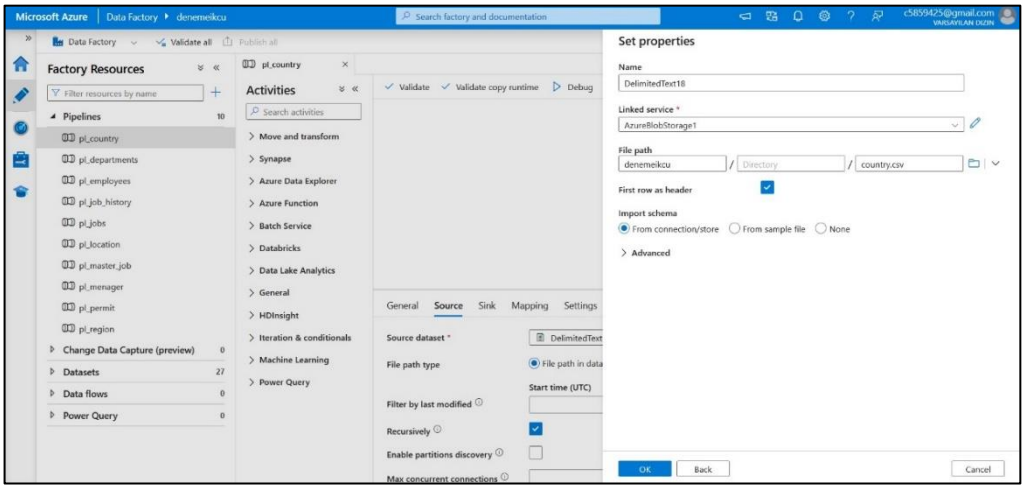
Azure Blob Storage veri depolama hizmeti kullanılarak container bölümünde bulunan upload kısmında verileri csv dosya türünde newemployes, permit, region, menager, locations, jobs, job_history, departments ve country isminde dokuz adet dosya yüklemesi yapılmıştır.



Şekil 3.4: Azure servisi container bölümü yüklenen csv dosyaları

3.2.2 Azure Altyapısından Verinin Çekilmesi

Veri Azure altyapısından blob storage container kısmına yüklenen verilerin linked services aracılığıyla blob Storageden azure data factorye çekilerek pipeline oluşturularak içerisinde datayı almak için oluşturulmuş Copy Data aktivitesinde bulunan source kısmından source dataset bölümünden new dedikten sonra blob storage seçiliyor ardından csv dosya türü seçilerek file path kısmından blob storage a daha önce yüklenen csv dosyası seçilerek source ayarı yapılır.



Şekil 3.5 : Pipeline oluşturma süreç gösterimi

Pipeline yapılanan verinin SQL Server Menagement Studio'da gösterilmesi oluşturulması için her tablo için dokuz farklı create table sql sorgusu yazılıp verinin SQL Server Menagement Studio'da gösterimi aktarımı sağlanmıştır.Aşağıda ki görselde staging şemasında Regions tablosunun oluşturulması SQL sorgusu vardır.

```
CREATE TABLE [staging].[Regions](
    [region_id] [nvarchar](max) NULL,
    [region_name] [nvarchar](max) NULL
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Şekil 3.6 : Tablo oluşturma örnek gösterimi

Regions tablosunu oluşturulduğunda verilerin geldiğini kontrol amaçlı Select top 1000 sorgusu yapılmaktadır.

```
1  SELECT TOP (1000) [region_id]
2        ,[region_name]
3    FROM [staging].[Regions]
4
```

Şekil 3.7 : Tablo oluşturma örnek gösterimi

Veriler aşağıda ki görselde ki gibi doğru gelmektedir. Verilerin doğruluğu kontrol edildikten sonra sonraki aşamalar için gerekli düzenlemeler ve sorgular yapılmıştır.

100 %

Results | Messages

	region_id	region_name
1	1	Indiana
2	2	California
3	3	North Carolina
4	4	Colorado
5	5	Florida
6	6	California
7	7	Wisconsin
8	8	New Mexico
9	9	Kansas
10	10	Colorado
11	11	Colorado
12	12	California
13	13	Illinois
14	14	Texas
15	15	New York
16	16	Florida

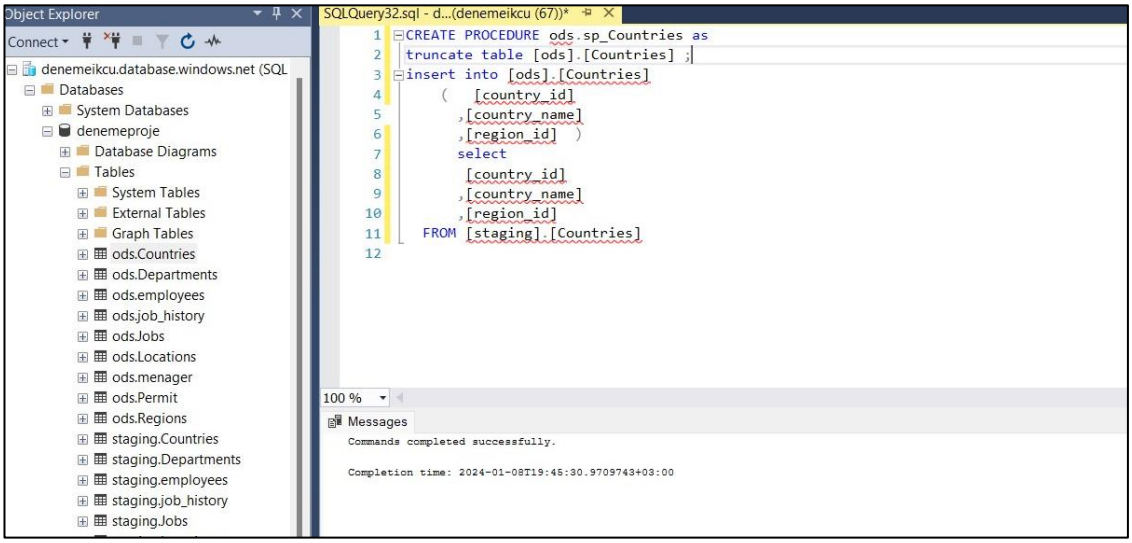
Şekil 3.8 : Sorgu sonucu

Sonraki aşamada ods şemasında dokuz adet tabloda da create table SQL sorgusu yapılmıştır.

```
CREATE TABLE [ods].[Regions](
    [region_id] [nvarchar](3) NOT NULL,
    [region_name] [nvarchar](25) NULL
) ON [PRIMARY]
```

Şekil 3.9 : Ods şemasında örnek regions tablosu oluşturma

Staging katmanından ODS katmanına verilerin aktarımı Stored Precedure SQL sorgusu ile mümkündür. Stored procesure sorgusu Create Proc sorgusuyla oluşturulmaktadır. Aşağıda çalışmada kullanılan örnek Procesure sorgusu mevcuttur. Sorguda verilerin aktarılacağı ods katmanının içi truncate sorgusuyla boşaltılarak veri çoklamasının önüne geçilmektedir. Ardından veriler "insert into" komutuyla staging katmanından ods katmanına aktarımı gerçekleştirilmiştir. Burada ki en önemli şey veri çoklamasının önüne geçilmesidir. Truncate table sorgusunun eklenmesiyle veri çoklanmasının önüne geçilmiştir.,



Şekil 3.10 : Örnek Procedure oluşturma

3.2.3 Veri İşleme Katmanları

3.2.3.1 Staging Katmanı

Staging Katmanı verinin ilk geldiği ham verinin bulunduğu katmandır. Bu katmanda ham veri işlenmek üzere bulunmaktadır. Kaynaktan çekilen verinin olduğu haliyle geldiği ve işlenmek üzere hazır bulunduğu kısımdır. Burada ki veriler esas çıktı olarak

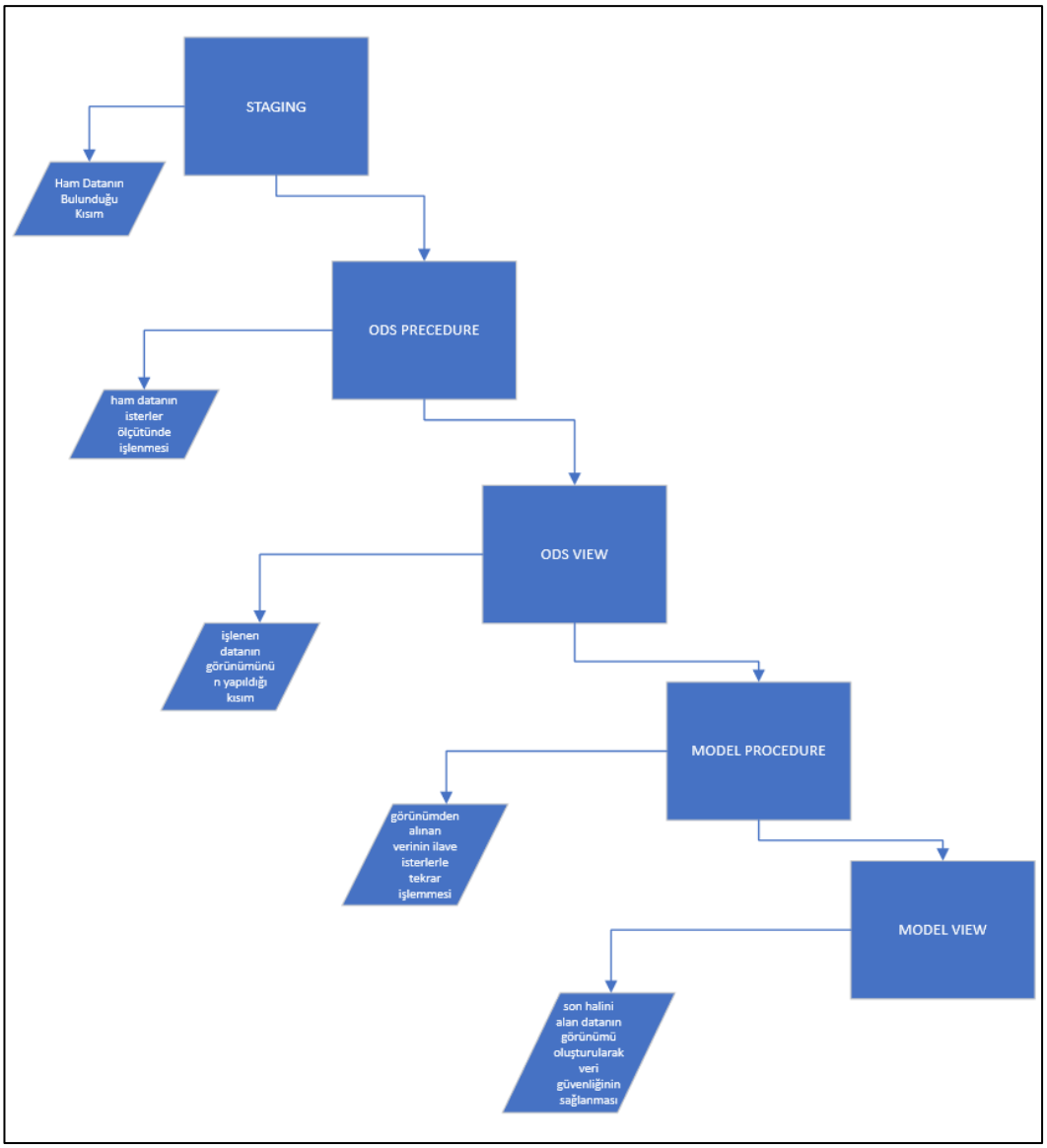
kullanılmazlar. İş biriminin isterleri ölçüsünde işlenmek için bu katmanda saklanırlar. Bu katmanda işletmelerin kurallarına bağlı olmak koşuluyla genellikle veriler her gün silinip tekrar yüklenmektedir.

3.2.3.2 ODS Katmanı

ODS katmanı staging katmanında gelen verinin stored procedure sorgusu ile staging katmanından ODS katmanına verinin aktarılmasıyla oluşmaktadır. Bu katmanda genelde staging katmanıyla aynı olmaktadır. Kolonlar aktarılır ve kolon çıkarma işlemi yapılmaz. Bu katmanda fazla değişikliğe uğramayan verilerin view görüntüsü oluşturulur. Sonra gelen Model katmanın oluşturulması amacıyla model katmanı için veriler oluşturulan ODS view kısmından çekilmektedir. Veri güvenliği ve performans gibi kriterler göz önüne alınırsa view oluşturulması da bir o kadar önemli olmaktadır.

3.2.3.3 Model Katmanı

Model katmanı ise ods katmanından oluşturulan denormalizasyon işlemi yapılan verilerin ods view en stored procedure ile çekilmesiyle yapılmıştır. Model katmanı veri güvenliği, performans gibi kriterler sebebiyle oluşturulmaktadır. Dikkatsiz bir çalışanın yanlışlıkla sildiği katmanlar göz önüne alındığında üç katmanlı çalışmak veri güvenliği açısından oldukça önemlidir. Rapora gelen son veriler bu katmandan gelmektedir. Aşağıda ki görselde verinin işlem basamakları gösterilmektedir. Hangi katmanda hangi şemalarda verinin aktarıldığı ve işlemlerin tek tek gösterilmektedir.



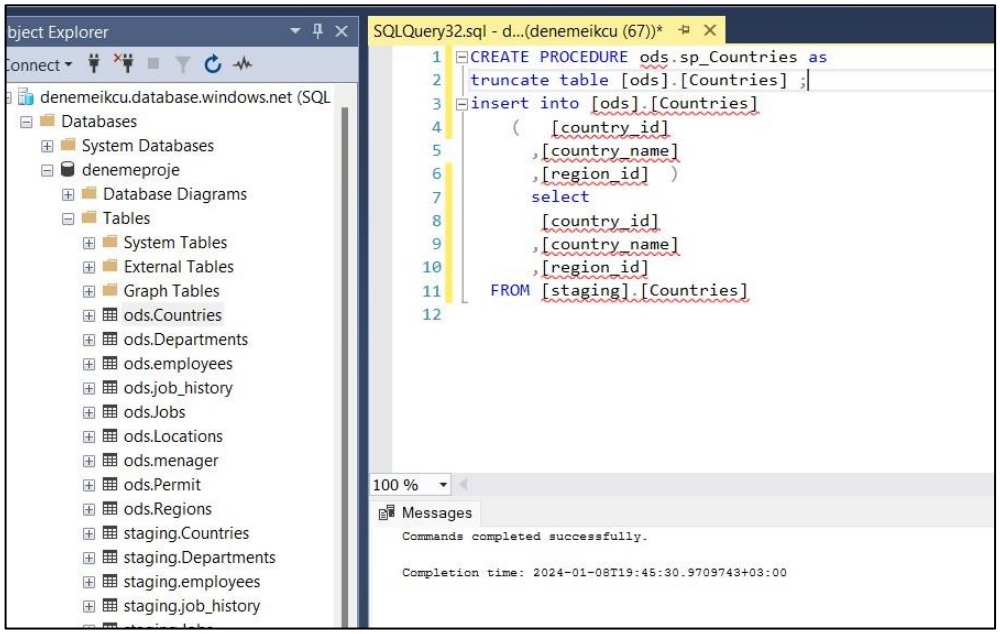
Şekil 3.11 : Veri İşlem Basamakları Gösterimi

3.2.4 View ve Stored Procedure

View kullanımı birden fazla tablonun join işlemiyle birleştirilip yeni bir tablo elde edilmesiyle oluşturulan sanal tablolardır. Fiziksel değildir. Burada ki amaç istenilen ölçüde kısıtlanmış veriye ulaşılarak basitleştirilmiş görünüm üzerinden daha hızlı çıktı almaktır. Yeni isimlendirme yapılarak amaca uygun daha az karmaşık bir isim yapısına da kabuşmuş olmaktadır yeni sanal tablomuz. Herhangi bir değişiklik talebinde mevcut tablolar üzerinden işkem yapmak yerine view üzerinden işlem yapılarak ana veri korunmuş olmaktadır böylelikle amaç dışı veri değişikliğinin de önüne geçilmiş olmaktadır. Proje çalışmasında join işlemi yapılarak [ods].[factJob_HistoryView],

[ods].[dimEmployeesView],[ods].[dimDepartmentView],[model].[factJob_HistoryView], [model].[dimEmployeesView], [model].[dimDepartmentView] görünümleri oluşturulmuştur.

Stored Procedure Türkçe ismiyle saklı yordam SQL sorgu dilinde oluşturulur. Oluşturulmasında ki amaç performans, güvenlik ve iş akışı gibi çekinceler göz önünde bulundurularak oluşturulmaktadır. Saklı yordam create procedure sorgusuyla oluşturulmaktadır. Bu projede iki farklı katmanda stored procedure oluşturulmuştur. Her katman önce ki katmanın stored procedure katmanından veriyi çekerek katmanlar arası bağlantıda oluşturulmuş olmaktadır. Çalışmada fark edildiyse verilerin güvenliği ve farklı katmanlar arası depo edilmesiyle yanlış veya hatalı müdahalelerde sistemin hızlı bir şekilde tekrardan işlemesine de önemli bir katkı sunmaktadır stored procedureler. Stored Procedure oluşturulurken veri çoklanmasının önlenebilmesi için truncate table sorgusu kullanılmaktadır.



Şekil 3.12 : Örnek Procedure oluşturma sorgusu

3.2.5 Veri Tipi ve Boyutu

Ods katmanı oluştururken veri türü ve boyutunda değişimler yaparak sistemde daha az bellek kullanımı, daha az depo alanı kullanılarak performansın artmasına ve

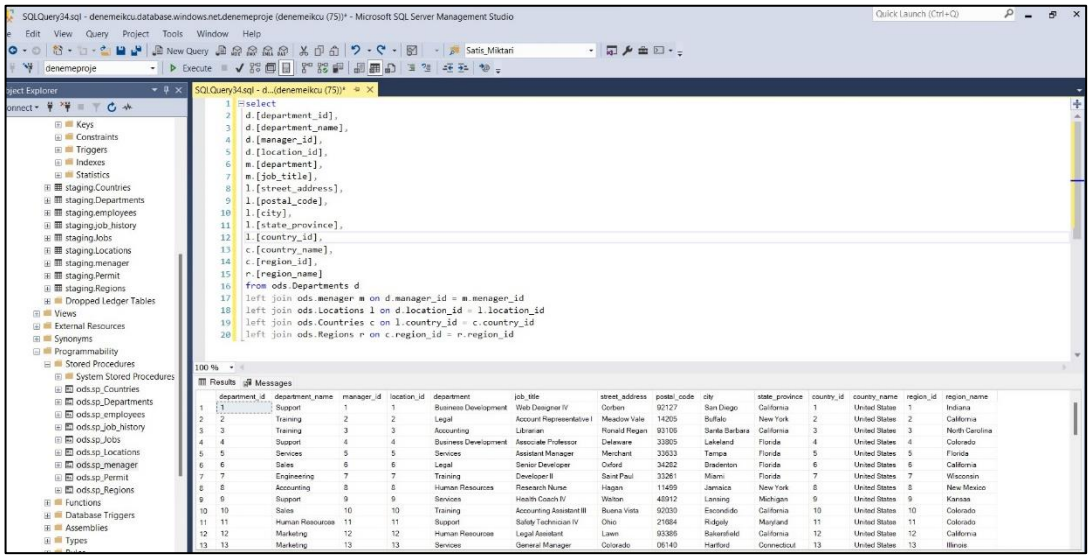
maliyetlerin azalması sağlanmaktadır. Örneğin bu çalışmada country_id sütununda ki maksimum karakter sayısını bulmak için select max len sorgusu kullanılarak maksimum olabilecek karakter sayısı bulunmuş olmaktadır. Bu çalışmada böyle bir sorgulama yaparak verinin maksimum karakter sayısı bulunmuş ve yeni katmanda karakter sayısı kadar yer alan tablo oluşturularak bellekten ve hacimden kazanç sağlanmış olmaktadır. Yapılan işlem verinin işlem basamaklarından bir tanesidir.

Şekil 3.13 : Örnek maksimum sayıda karakter sayısı bulma sorgusu

Şekil 3.14 : Ods şemasında Countries tablosunu oluşturma sorgusu

3.2.6 Denormalizasyon Sorgusu ve View Oluşturma

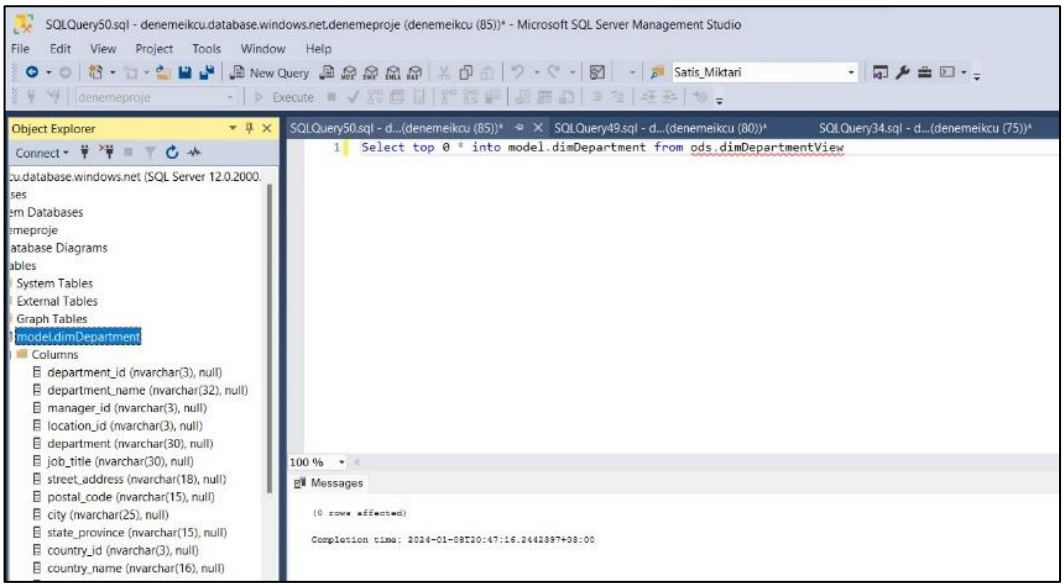
Tabloların denormalize edilme süreci sql sorgu dilinde left join işlemiyle gerçekleştirilmiştir. Left join kullanılmasının sebebi ise ana tabloda ki kayıtlarla eşleşen sağ tablo kayıtlarının getirilmesinin sağlanması ve ana tabloda ki tüm tabloları sağ tarafta kalan tabloda ise eşleşen tabloları alarak denormalize işleminin sorgu basamağı gerçekleştirilmiş olmaktadır. Aşağıdaki görselde ods view oluşturulmadan önce yapılan join sorgularının kontrolü sağlanması gösterilmektedir. Burada verilerin istenildiği gibi geldiği kontrol edildikten sonra create ods view yapılarak view katmanı oluşturulmuştur.



Şekil 3.15 : Örnek tabloları birleştirme sorgusu

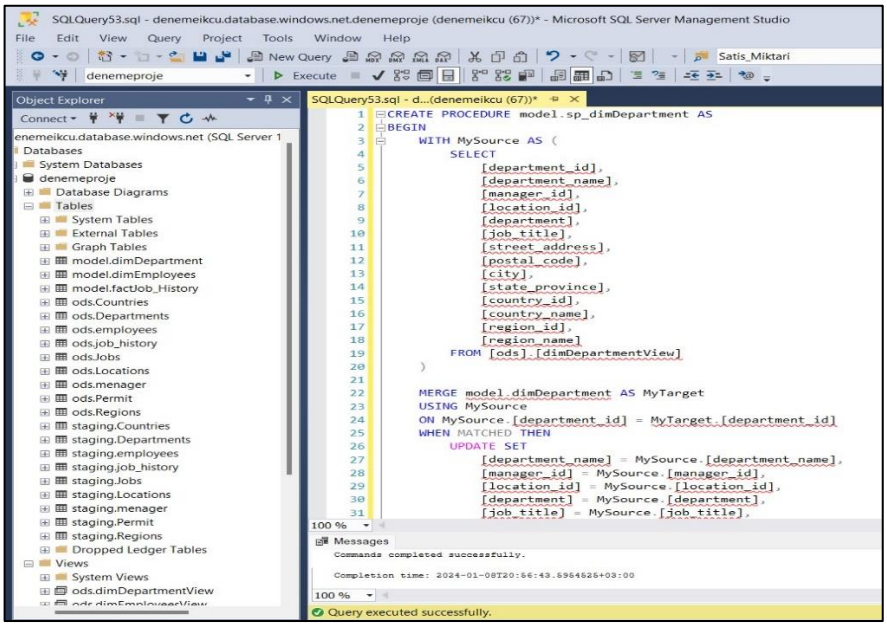
3.2.6.1 ODS View'den Model Katmanına Veri Aktarımı

Model şemasında tablo oluşturmak için, ODS şemasında hazırladığımız View' deki yapı kullanılır. Bu aktarım için "SELECT TOP 0 * INTO [model].[HEDEF_TABLO] FROM [ods].[KAYNAK_TABLO_View]" komutunu kullanılır. Bu komutu kullanılmasında ki başlıca sebep ODS şemasındaki View' de hazırlanan tablo yapısını herhangi bir veri aktarımı yapmadan Model şemasına aktarmaktır. Bu amaçla bu SQL sorgusu kullanılmıştır. Şekil 3.16'da gösterilmiştir.

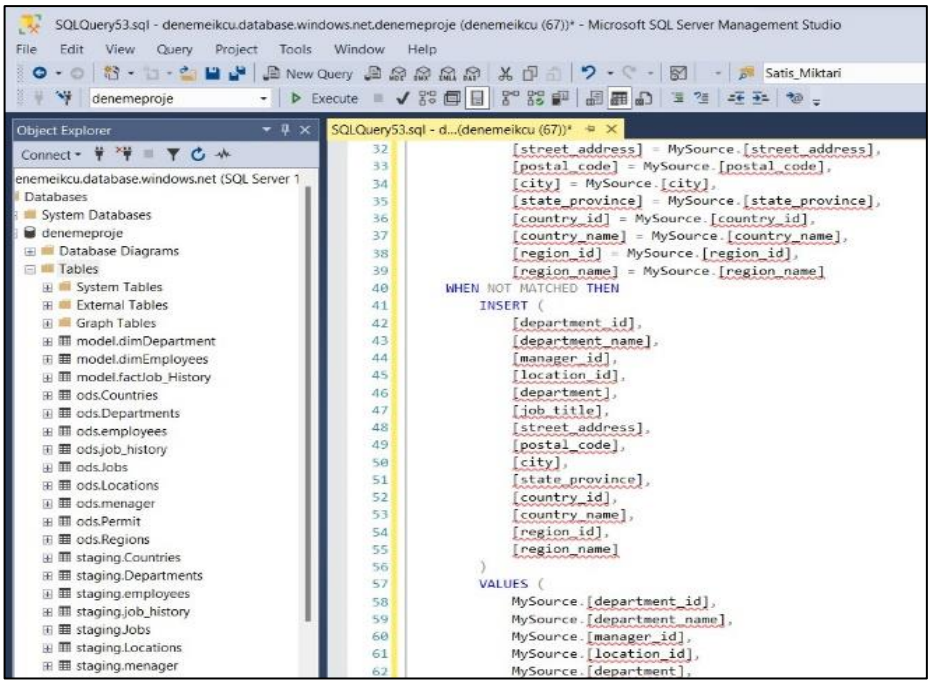


Şekil 3.16 : Örnek sorgu

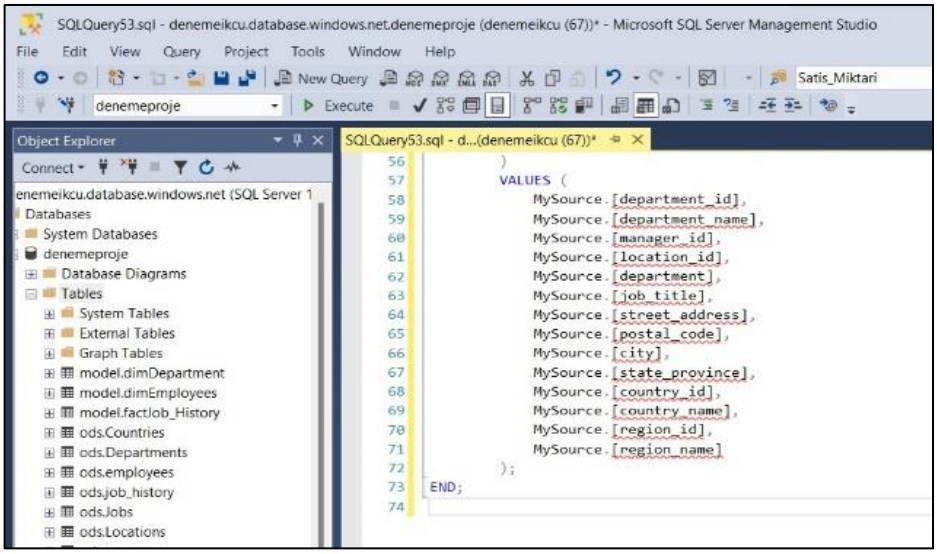
Merge sorgusu kullanılarak verinin alındığı kaynağın olduğu kolonları karşılaştırır. İçeride unique idlerle eşleşen veri varsa güncel olup olmadığını kontrol eder. Güncel değilse günceller, güncelse veri transferi gerçekleşmez. Eğer eşleşen veri yoksa veriyi tabloya ekler.



Şekil 3.17 : Örnek procedure oluşturma ve merge sorgusu

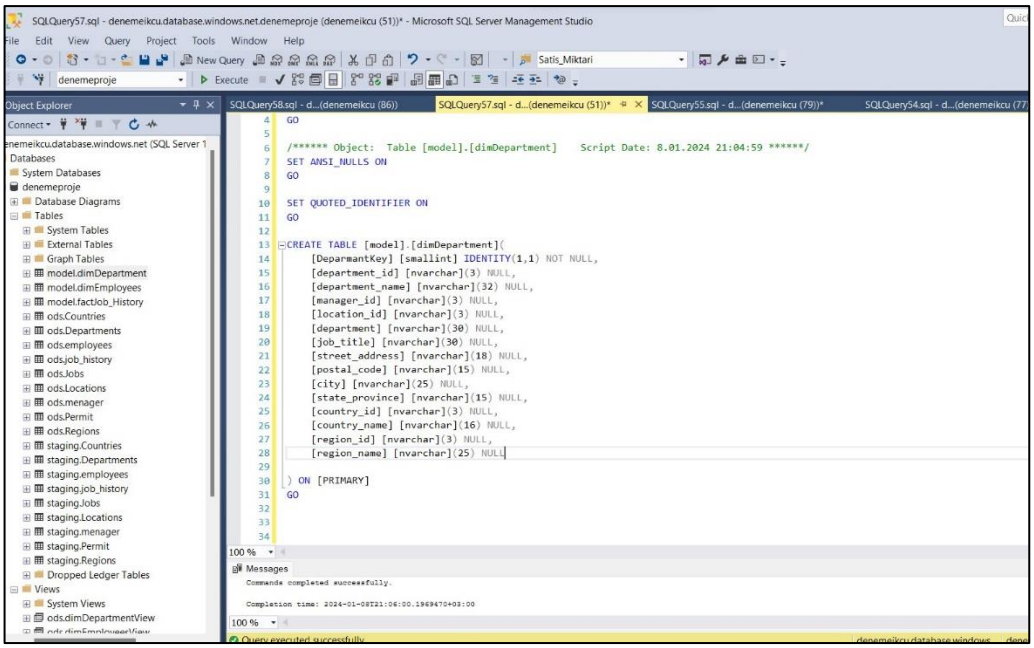


Şekil 3.18 : Örnek procedure oluşturma ve merge sorgusu devamı



Şekil 3.19 : Örnek procedure oluşturma ve merge sorgusu devamı

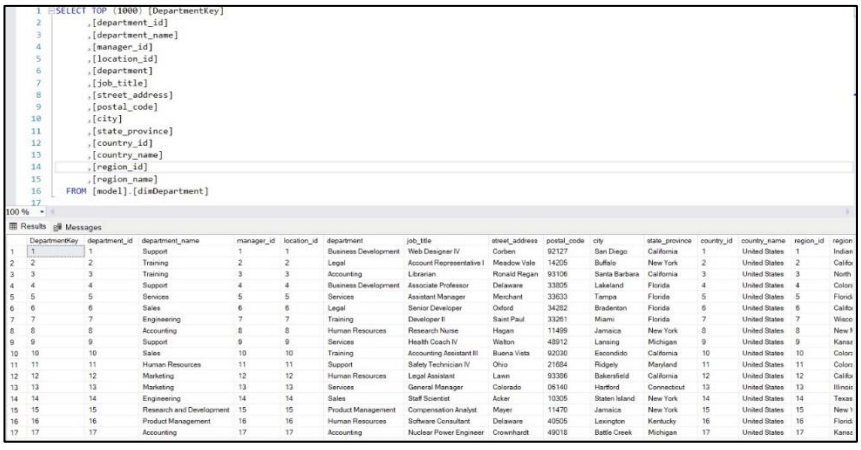
Dimension tablolar için benzersin key olması için otomatik key oluşturma ihtiyacı doğmuştur. Diğer tablolarla ilişki kurabilmek için unique keyden kurulması gerekmektedir. Bu yüzden identity(1,1) sorgusu kullanılmıştır. Oluşturulan bu primary keyler fact tablolarda kullanılır.



Şekil 3.20 : Örnek model şemasında dimDepartment tablosu oluşturma sorgusu

3.2.7 Elde Edilen Örnek Veri

Bu çalışmada en son nihai çıktı örneği aşağıda ki şekil 3.21, şekil 3.22 ve şekil 3.23'te verilmiştir. Tabloların birleştirilmiş ve işlemden geçirilmiş hali nihai çıktı olarak [model].[dimDepartment] , [model].[factJob_History] ve [model].[dimDepartment] modellerin çıktıları alınmıştır.



Şekil 3.22: [model].[dimDepartment] veri çıktısı

```
SELECT TOP (1000) [employee_id]
      ,[start_date]
      ,[end_date]
      ,[permit_id]
      ,[day_used]
  FROM [model].[factJob_History]
```

100 %

Results | Messages

	employee_id	start_date	end_date	permit_id	day_used
1	1	2018-05-07 17:44:22.000	NULL	1	19
2	2	2023-11-11 05:22:55.000	NULL	2	18
3	3	2022-07-14 21:43:10.000	NULL	3	18
4	4	2018-03-25 03:23:19.000	NULL	4	21
5	5	2022-07-17 11:00:36.000	NULL	5	8
6	6	2022-06-28 05:04:12.000	NULL	6	3
7	7	2022-09-20 06:00:43.000	NULL	7	7
8	8	2019-07-11 15:40:00.000	NULL	8	15
9	9	2023-12-19 06:15:23.000	NULL	9	24
10	10	2018-08-31 14:54:00.000	NULL	10	25
11	11	2023-03-30 20:53:48.000	NULL	11	15
12	12	2022-10-25 04:35:20.000	NULL	12	20
13	13	2022-03-23 01:07:22.000	NULL	13	25
14	14	2021-02-25 13:19:31.000	NULL	14	27
15	15	2020-04-18 01:40:37.000	NULL	15	6
16	16	2023-02-17 10:40:52.000	NULL	16	13
17	17	2022-07-10 18:50:25.000	NULL	17	1
18	18	2018-04-24 02:11:59.000	NULL	18	14
19	19	2021-02-27 20:26:23.000	NULL	19	23
20	20	2021-03-04 17:30:39.000	NULL	20	12
21	21	2023-12-17 18:32:31.000	NULL	21	11
22	22	2022-04-02 16:41:45.000	NULL	22	12
23	23	2020-10-04 08:43:50.000	NULL	23	10
24	24	2021-02-04 06:09:39.000	NULL	24	4
25	25	2022-11-12 13:48:20.000	NULL	25	28
26	26	2020-06-03 11:39:03.000	NULL	26	15
27	27	2022-01-22 05:18:51.000	NULL	27	28
28	28	2019-08-26 00:03:50.000	NULL	28	15
29	29	2020-09-03 15:03:25.000	NULL	29	4

Şekil 3.21: [model].[factJob_History] veri çıktısı

```
SELECT TOP (1000) [EmployeeKey]
      ,[employee_id]
      ,[first_name]
      ,[last_name]
      ,[email]
      ,[phone_number]
      ,[department_id]
      ,[hire_date]
      ,[job_id]
      ,[gender]
      ,[job_title]
      ,[salary]
  FROM [model].[dimEmployees]
```

100 %

Results | Messages

	EmployeeKey	employee_id	first_name	last_name	email	phone_number	department_id	hire_date	job_id	gender	job_title	salary
1	1	1	Marris	Phil	mphil0@timesonline.co.uk	480-714-8166	1	2019-08-10 04:19:29	1	Female	VP Quality Control	6493
2	2	2	Tonnie	Mouser	tmouser1@princeton.edu	658-453-2365	2	2019-11-02 17:09:50	2	Male	General Manager	24021
3	3	3	Hill	Bahike	hbahike2@shutterfly.com	708-804-8021	3	2017-11-08 03:31:45	3	Male	Physical Therapy Assistant	18390
4	4	4	Trish	Occleshaw	toccleshaw3@earthlink.net	298-189-3418	4	2021-11-03 03:17:49	4	Female	Recruiting Manager	23792
5	5	5	Ferdie	West-Frimley	fwestfrimley4@yale.edu	926-818-1120	5	2018-05-12 21:22:58	5	Male	Pharmacist	11023
6	6	6	Richardo	Jailler	rjailler5@loc.gov	780-750-6383	6	2023-09-11 00:30:03	6	Male	Analyst Programmer	15623
7	7	7	Che	Harsant	charsant6@dailymail.co.uk	558-313-5982	7	2018-07-29 17:02:56	7	Male	Design Engineer	3228
8	8	8	Vaughn	Buxey	vbuxey7@google.cn	121-413-4790	8	2023-11-02 21:54:37	8	Male	Programmer IV	19366
9	9	9	Amble	Littlewood	alittlewood8@infoseek.co.jp	773-629-5406	9	2018-07-14 23:50:25	9	Male	Assistant Media Planner	12305
10	10	10	Nelly	Stother	nstother9@instagram.com	709-939-4197	10	2016-11-03 23:58:46	10	Female	Nuclear Power Engineer	8077
11	11	11	Cirilo	Kirkebye	ckirkebyea@wunderground.com	167-638-4247	11	2022-12-28 11:05:55	11	Male	Graphic Designer	15264
12	12	12	Julietta	Finlow	jfinlowb@wufoo.com	255-119-8092	12	2022-11-24 06:13:18	12	Polygender	Teacher	10252
13	13	13	Nicola	Dyke	ndykec@upenn.edu	595-595-4352	13	2022-06-22 15:00:23	13	Male	Pharmacist	10339
14	14	14	Casie	Ching	cchingd@squidoo.com	277-754-2501	14	2021-02-18 16:12:13	14	Female	Sales Representative	7622
15	15	15	Nicolle	Heathfield	nheathfielde@uiuc.edu	805-731-9621	15	2022-07-17 10:37:11	15	Female	Quality Engineer	11077
16	16	16	Hercule	Mourgue	hmourguef@canalblog.com	605-402-2942	16	2022-05-20 03:38:24	16	Male	Account Representative IV	22411
17	17	17	Florentia	Franckton	ffranckton@dropbox.com	685-355-9232	17	2020-10-27 09:37:02	17	Female	Operator	17838
18	18	18	Cam	Iredell	ciredellh@bandcamp.com	714-416-5992	18	2022-11-28 17:01:09	18	Male	Senior Quality Engineer	13789
19	19	19	Olga	Keppie	okeppiei@lulu.com	199-562-4231	19	2020-07-14 20:34:03	19	Female	Electrical Engineer	21230
20	20	20	Wolf	Tenbrug	wtenbrugj@arizona.edu	413-331-0918	20	2023-01-06 13:45:56	20	Male	Paralegal	16783
21	21	21	Valentine	Van Geffen	vangeffenk@stumbleupon.com	932-646-5800	21	2021-12-09 05:57:19	21	Female	Geologist II	13282

Şekil 3.23: [model].[dimEmployees] veri çıktısı

Bölüm 4

Sonuçlar ve Öneriler

4.1 Sonuçlar

Projede örnek bir veritabanı oluşturuldu. Oluşturulan veriler adım adım işlem basamaklarında işleme tabi tutulmuştur. Aşağıda ki sonuç basamaklarında belirlenen kurallar çerçevesinde yürütülen veri işleme basamakları hangi basamakta denormalize işlemine tabi tutulacağından final basamağında ilave isterler doğrultusunda yapıalcak işlemlere kadar detaylı şekilde aktarılmıştır.

- Veri güvenliği açısından ham datalar Mockaroo internet sitesinde belli kriterlerde rastgele oluşturuldu.
- Datalar insan kaynakları veritabanı sistemine uygun olacak şekilde oluşturuldu.
- Üç farklı katmanda staging, ods ve model şemalarında tabloların oluşturulması planlandı.
- Oluşturulan tablolara verilerin aktarımı için gerekli altyapı işlemleri yapılması işlem patikaları planlandı.
- Staging şemasında SQL Server Management Studio'da tablo oluşturuldu.
- Microsoft Azure platformuna İnsan Kaynakları veritabanı oluşturulması amacıyla dokuz farklı csv dosya türünde kaynak dosyaları container kısmına yüklendi.
- Veriler Azure Blob Storage yüklenerek verilerin daha performanlı işlenmesi sağlanmış oldu.

- Container kısmından depolanan verinin işlenmesi için dokuz adet pipeline oluşturuldu.
- Dokuz adet csv dosyaları için oluşturulan pipelinelerın SQL Server Management Studio'ya aktarımı için sistemler birbirine bağlandı.
- Staging katmanına verilerin aktarımı Microsoft Azure bulut platformundan sağlandı.
- Dokuz farklı tablo için Staging katmanında SQL Server Management Studio'da SQL sorgusuyla create edildi.
- Staging katmanına veri aktarımı pipelinelar aracılığıyla sağlandı.
- ODS ve Model katmanları SQL sorgu dilinde create edildi.
- Veri aktarım sürecinde her katmanda ihtiyaç analizi sonucunda denormalize işlemi yapıldı.
- Denormalize işlemi sonucunda fact.JOB_HISTORY, dim.Employees ve dim.Departments tabloları oluşturuldu.
- fact.JOB_HISTORY tablosu job_history ve work permit tablolarının birleşimiyle, dim.Employees tablosu employees ve jobs tablolarının birleşimiyle, dim.Departments tablosu departments, manager, locations, countries ve regions tablolarının birleştirilmesiyle oluşturuldu.
- Staging katmanından ODS katmanına veri aktarımı için Stored Procedureler oluşturuldu.
- Veritabanında gereksiz veri tutulmasının önüne geçilmesi için ods katmanında düzenlemeler yapıldı.
- Normalizasyon işleminin ODS View'de yapıldı.
- ODS view katmanları denormalize işleminde oluşturulan tabloların SQL sorgu dilinde create edilmesiyle yapıldı.

- Model katmanı model.dimDepartment, model.dimEmployees ve model.factJob_History tablolarına Stored Procedure ile veri aktarımı ods viewden sağlandı.
- Model katmanına verinin aktarımı stored procedure ile ods view den sağlandı.
- Model katmanında tarih verisinin yıl-ay-gün olacak şekilde ayarlandı.
- Dimension tablolarda otomatik key atama işlemi yapıldı.
- İşlemler sonucunda veriler istenilen halde çıktı alındı.
- Hazırlanmış verilerin çıktısı model view den alınmaktadır.
- Raporlama için veriler model katmanının viewinden alınmaktadır.

4.2 Öneriler

- Oluşturulan pipelinelara Azure trigger aktivitesi eklenerek istenilen gün ve saatte otomatik olarak tetiklenmesini ve verinin kaynaktan gelmesi sağlanabilirdi.
- Farklı veri kaynakları kullanılabilirdi. Örneği Postgre Sql kaynağından veri çekilebilirdi.
- Model katmanında işlenen veri Power Bi'da gösterimi yapılabilirdi.
- Sorgulama performansını arttırmak için indexler kullanılabilirdi
- Sorgulama ve veri performansı için "with (clustered columnstore index, distribution = replicate)" sorgusu kullanılabilirdi. Bu sorgu verinin sıkıştırılmasında ve depolanmasında etkisi olabilirdi.
- Azure'da farklı aktiviteleri kullanarak ham verinin kısmen ön işlenmesi data factory'de sağlanabilirdi.

Kaynaklar

Florescu D., Fourny G., (2013). JSONiq: The History of a Query Language. *IEEE,* 86–90, doi: 0.1109/MIC.2013.97

Ioannıdıs E.Y., (1996). Query optimization. *ACM Journals,* 121-123, https://doi.org/10.1145/234313.234367

Jatana, N., Puri, S., Ahuja, M., Kathuria, I., Gosain, D. (2012). A Survey and Comparison of Relational and Non-Relational Databases. *International Journal of Engineering Research &Technology*, 1,1-5. https://doi.org/10.1142/9781848168701_0002

Patel A.R., Patel F.J.M., (2012). Data Modeling Techniques For Data Warehouse. *International Journal of Multidisciplinary Research,* ISSN Number: ISSN 2231 5780

Haskin L.R., (1982). On Extending the Functions of a Relational Database System. *ACM Digital Library,* Article ID CMO-89791-073-7/82/006/0207

Watt A., Eng N. (2012). *Database Design - 2nd Edition* . B.C. Open Textbook project.

Uragun B., Rajan R., (2011). Developing an appropriate data normalization method. *10th International Conference on Machine Learning and Applications, Honolulu, HI, USA.* Doi: 10.1109/ICMLA.2011.53

Mukherjee R., Kar P., (2017). A Comparative Review Of Data Warehousing ETL Tools With New Trends And Industry Insight. *IEEE 7th International Advance Computing Conference, Hyderabad, India.* doi: 10.1109/IACC.2017.0192

Prados, F., Boada, I., Soler J., Poch J. (2005, July 09-19). *An Automatic Correction Tool For Relational Database Schemas.* 2005 6th International Conference on Information Technology Based Higher Education and Training, Santo Domingo,DominicanRepublic.https://doi.org/10.1109/ITHET.2005.1560331

O'Neil P., O'Neil E., Chen X., Revilak S. (2009). *The Star Schema Benchmark and Augmented Fact Table Indexing*. First TPC Technology Conference, Lyon, France. Doi:10.1007/978-3-642-10424-4_17

Önder, E. (2005). Yönetim Bilişim Sistemleri KapsamındaWeb Tabanlı İlişkisel Veritabanı Yönetim Sistemleri ve Bir Uygulama [Yüksek lisans tezi, İstanbul Üniversitesi]. YÖK Ulusal Merkezi. https://tez.yok.gov.tr/UlusalTezMerkezi/