

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ



BLM4522 - Ağ Tabanlı Paralel Dağıtım Sistemleri Proje Raporu

NURSENA TAŞKIRAN

21290665

<https://github.com/nursenataskiran>

Veri tabanı Güvenliği ve Erişim Kontrolü (3.proje)

1. Erişim Yönetimi

1.1. SQL Server Authentication kullanarak erişim yönetimi:

Öncelikle security->login klasörüne erişim sağlıyoruz. Daha sonra new login ekranına ulaşarak gerekli bilgileri giriyoruz.

Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script Help

Login name: new_user Search...

☐ Windows authentication

☐ Microsoft Entra ID authentication

☒ SQL Server authentication

Password: ****

Confirm password: ****

☐ Specify old password

Old password:

☒ Enforce password policy

☒ Enforce password expiration

☒ User must change password at next login

☐ Mapped to certificate

☐ Mapped to asymmetric key

☐ Map to Credential

Add

Mapped Credentials

Credential	Prov
------------	------

Remove

Default database: master

Default language: <default>

OK Cancel

Yukarıda bulunan ekranda gerekli bilgileri girdikten sonra user mapping kısmına veritabanımızı ekliyoruz.

Login - New

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script Help

Users mapped to this login:

Map	Database	User	Default Schema
<input checked="" type="checkbox"/>	AdventureWorks2019	new_user	AdventureWorks2019
<input type="checkbox"/>	blm4522		
<input type="checkbox"/>	BookRatingDB		
<input type="checkbox"/>	master		
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		

☐ Guest account enabled for: AdventureWorks2019

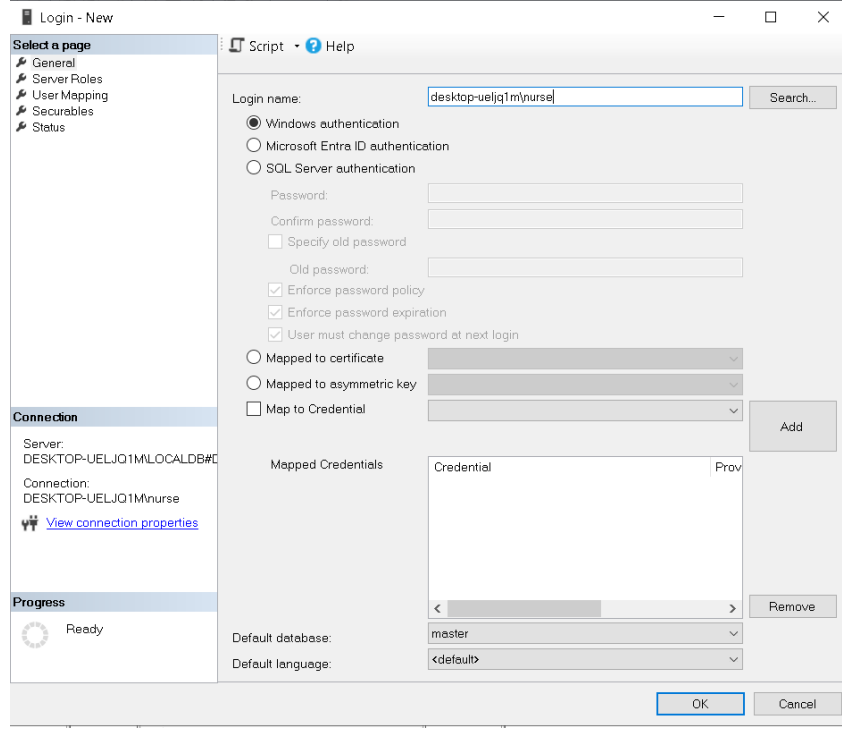
Database role membership for: AdventureWorks2019

- ☐ db_accessadmin
- ☐ db_backupoperator
- ☐ db_datareader
- ☐ db_datawriter
- ☐ db_dtladmin
- ☐ db_deniedreader
- ☐ db_deniedwriter
- ☐ db_owner
- ☐ db_securityadmin
- ☒ public
- ☒ RaporGorme

OK Cancel

1.2 Windows Authentication

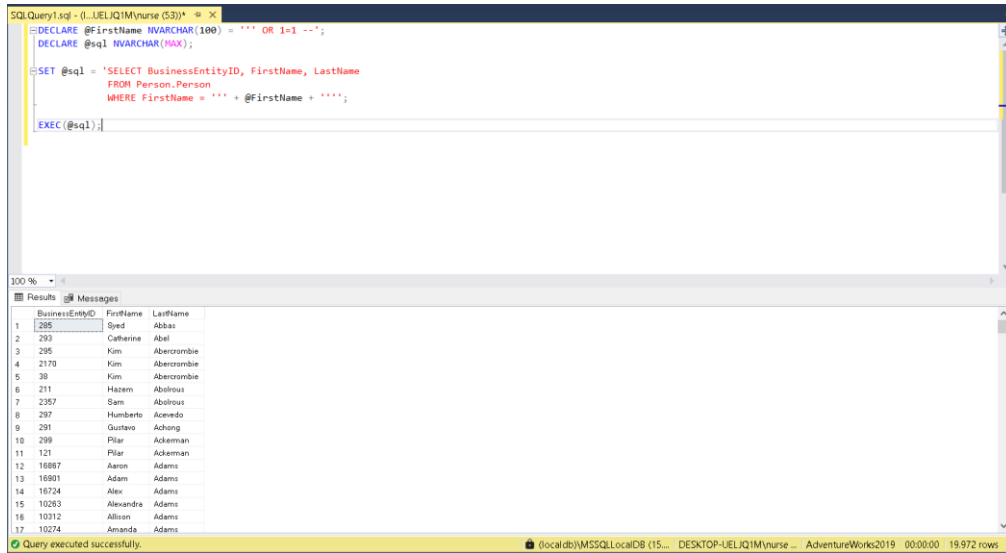
Aynı işlemleri tekrarlayacağız fakat bu sefer Windows authentication kısmını kullanacağız. Kullanıcı adımızı öğrenmek için öncelikle cmd'ye whoami komutunu giriyoruz. Daha sonra kullanıcı adımızla ekrandaki bilgileri doldurarak veritabanına erişim yetkisini tanımlıyoruz.



2. SQL Injection Testleri

Bu çalışmada, SQL Injection (veritabanına dışarıdan zararlı sorgu sokma) saldırısının nasıl yapılabildiğini ve buna karşı nasıl korunabileceğimizi göstereceğiz. Hem savunmasız bir örnek hem de güvenli bir örnek yapacağız.

İlk olarak, kullanıcının ismini kontrol eden bir sistem yazıyormuşuz gibi bir senaryo kuruyoruz. Ancak bu sorguyu güvensiz şekilde kurduk. Bu durumda sorgu tüm kayıtları getiriyor. Yani sistem **herkesi listelemiş oluyor**. Bu, klasik bir SQL injection örneği.



```
SQLQuery1.sql - (\\...\\UELQ1M\\nurse (53))*
--DECLARE @FirstName NVARCHAR(100) = 'OR 1=1 --';
--DECLARE @sql NVARCHAR(MAX);

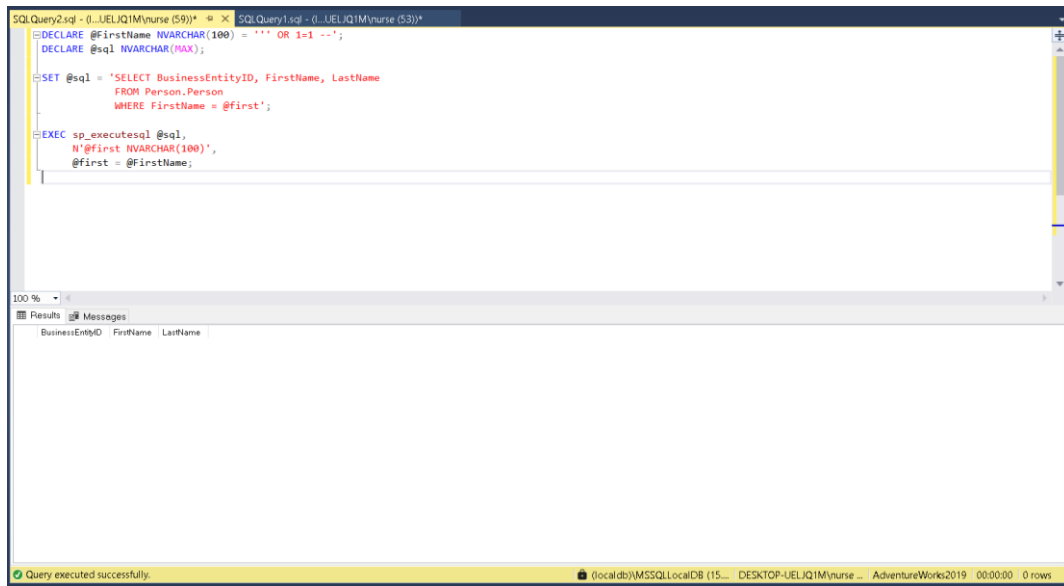
--SET @sql = 'SELECT BusinessEntityID, FirstName, LastName
--FROM Person.Person
--WHERE FirstName = ' + @FirstName + ''';

--EXEC (@sql);
```

BusinessEntityID	FirstName	LastName
295	Syed	Abbas
293	Catherine	Abel
295	Kim	Abercrombie
2170	Kim	Abercrombie
39	Kim	Abercrombie
211	Hazem	Abolrous
2357	Sam	Abolrous
297	Humberto	Acevedo
291	Gustavo	Achong
299	Pilar	Ademman
121	Pilar	Ademman
16887	Aaron	Adams
16901	Adam	Adams
16724	Alex	Adams
10293	Alexandre	Adams
10312	Allison	Adams
10274	Amanda	Adams

Query executed successfully.

Aynı işlemi bu sefer doğru şekilde, yani parametrelili sorgu kullanarak yapıyoruz. Burada ise zararlı giriş çalışmadı. Çünkü sorgunun içine doğrudan yazmak yerine, dışarıdan parametre olarak gönderdik. Böylece OR 1=1 gibi ifadeler anlamını yitiriyor.



```
SQLQuery2.sql - (\\...\\UELQ1M\\nurse (59))*
--DECLARE @FirstName NVARCHAR(100) = 'OR 1=1 --';
--DECLARE @sql NVARCHAR(MAX);

--SET @sql = 'SELECT BusinessEntityID, FirstName, LastName
--FROM Person.Person
--WHERE FirstName = @first';

--EXEC sp_executesql @sql,
--N'@first NVARCHAR(100)',
--@first = @FirstName;
```

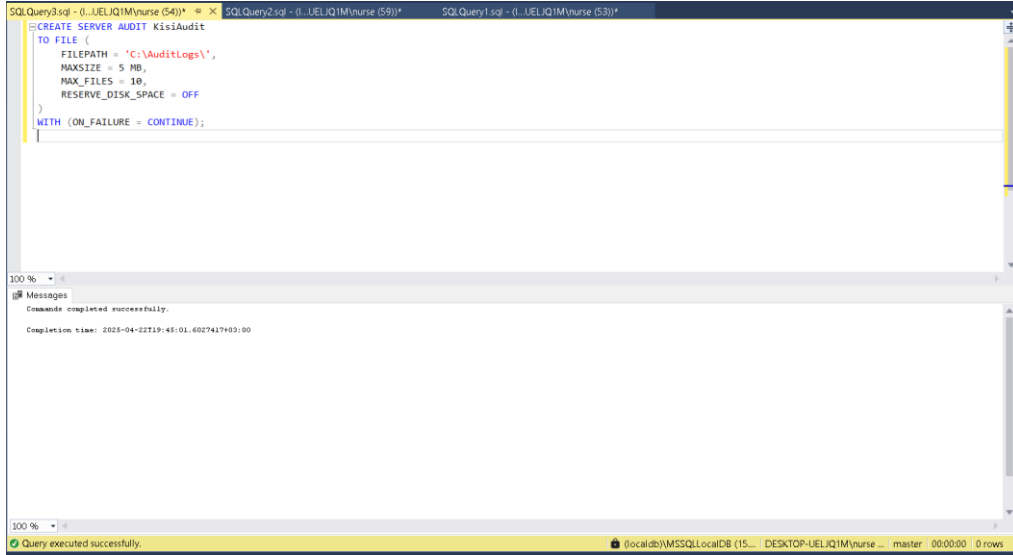
BusinessEntityID	FirstName	LastName
------------------	-----------	----------

Query executed successfully.

Bu testler doğrultusunda şu sonuca varabiliriz: Eğer SQL sorgularını doğrudan string birleştirerek yazarsak sistem dışarıdan gelen saldırılara karşı savunmasız kalabilir. Ama parametrelili sorgular kullanırsak bu saldırıları kolayca engelleyebiliriz.

3. Audit Logları

Bu çalışmada, SQL Server'da hangi kullanıcının ne zaman hangi işlemi yaptığını izleyebilmek için Audit (denetim kaydı) sistemi kuracağız. Böylece veri güvenliği açısından önemli olan erişimlerin ve işlemlerin loglanması sağlanacak.



```
SQLQuery3.sql - (\\...\\UELIQ1M\\nurse (54)) * SQLQuery2.sql - (\\...\\UELIQ1M\\nurse (59)) * SQLQuery1.sql - (\\...\\UELIQ1M\\nurse (53)) *
CREATE SERVER AUDIT KisiAudit
TO FILE (
FILEPATH = 'C:\\AuditLogs\\',
MAXSIZE = 5 MB,
MAX_FILES = 10,
RESERVE_DISK_SPACE = OFF
)
WITH (ON_FAILURE = CONTINUE);
```

100 %

Messages

Commands completed successfully.

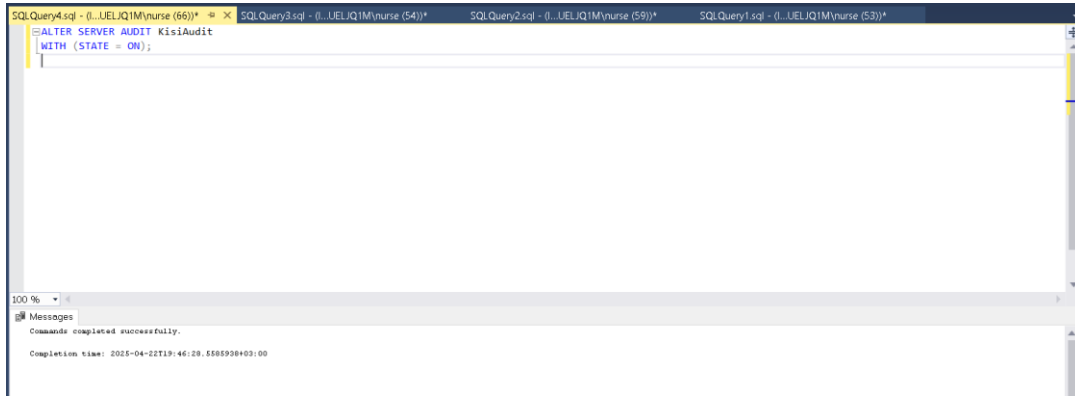
Completion time: 2025-04-22T19:45:01.6027417+03:00

100 %

Query executed successfully.

\\localdb\\MSSQLLocalDB (15... DESKTOP-UELIQ1M\\nurse ... master 00:00:00 0 rows

Audit'in çalışmaya başlaması için onu aktif hale getiriyoruz.



```
SQLQuery4.sql - (\\...\\UELIQ1M\\nurse (66)) * SQLQuery3.sql - (\\...\\UELIQ1M\\nurse (54)) * SQLQuery2.sql - (\\...\\UELIQ1M\\nurse (59)) * SQLQuery1.sql - (\\...\\UELIQ1M\\nurse (53)) *
ALTER SERVER AUDIT KisiAudit
WITH (STATE = ON);
```

100 %

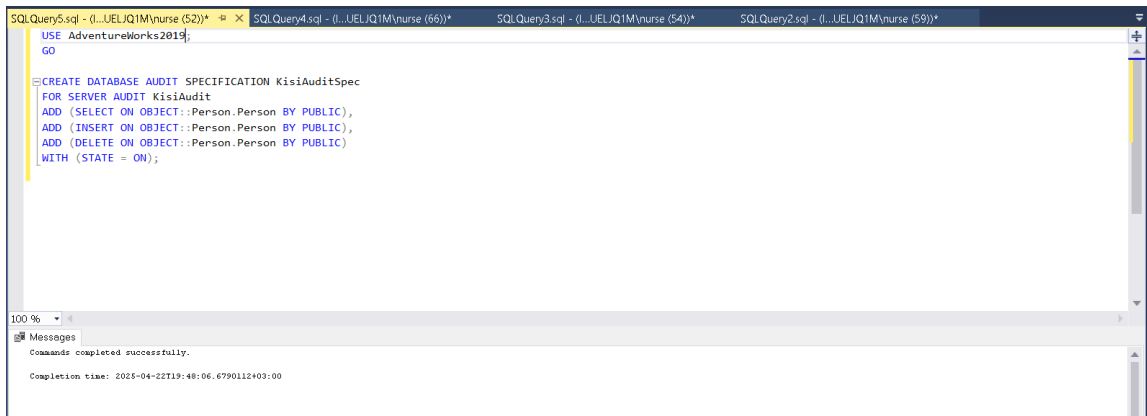
Messages

Commands completed successfully.

Completion time: 2025-04-22T19:46:28.5589389+03:00

-Person.Person tablosu için **SELECT**, **INSERT** ve **DELETE** işlemleri izleniyor.

-PUBLIC ile tüm kullanıcılar denetime dahil edildi



```
SQLQuery5.sql - (\\...\\UELIQ1M\\nurse (52)) * SQLQuery4.sql - (\\...\\UELIQ1M\\nurse (66)) * SQLQuery3.sql - (\\...\\UELIQ1M\\nurse (54)) * SQLQuery2.sql - (\\...\\UELIQ1M\\nurse (59)) *
USE AdventureWorks2019;
GO

CREATE DATABASE AUDIT SPECIFICATION KisiAuditSpec
FOR SERVER AUDIT KisiAudit
ADD (SELECT ON OBJECT::Person.Person BY PUBLIC),
ADD (INSERT ON OBJECT::Person.Person BY PUBLIC),
ADD (DELETE ON OBJECT::Person.Person BY PUBLIC)
WITH (STATE = ON);
```

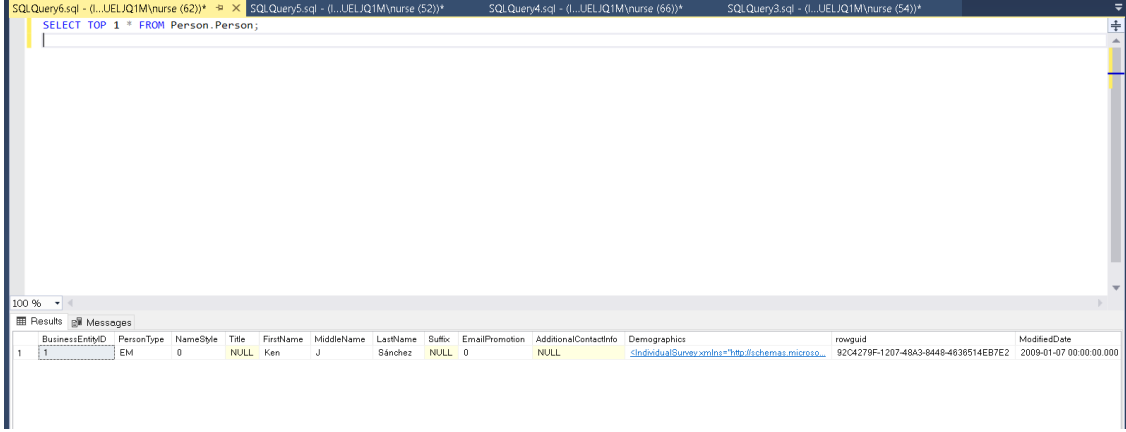
100 %

Messages

Commands completed successfully.

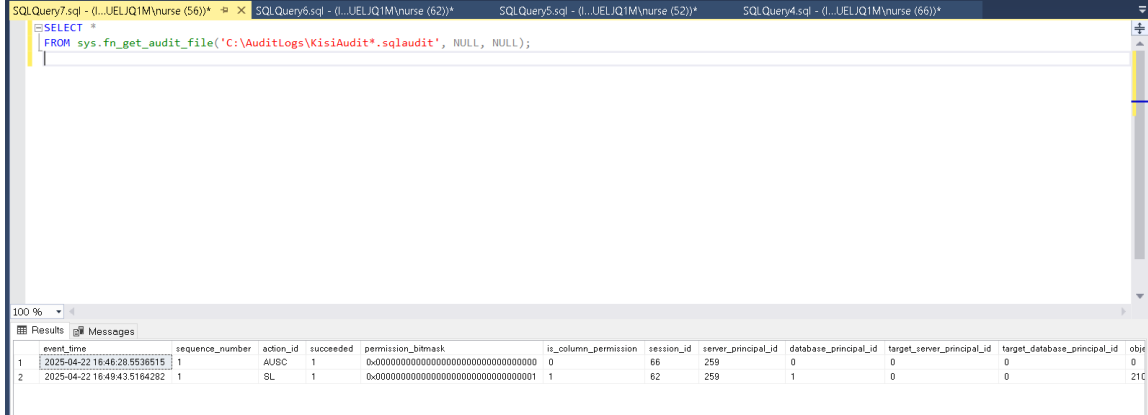
Completion time: 2025-04-22T19:48:06.6790112+03:00

Bu sorgunun çalıştırılması, audit sistemi tarafından kaydedilecektir. Aynı şekilde veri ekleme ve silme işlemleri de loglara yazılır.



The screenshot shows a SQL Server Enterprise Manager window with a query window titled 'SQLQuery6.sql - (\\...\\UELIQ1M\\nurse (62))*'. The query is 'SELECT TOP 1 * FROM Person.Person;'. The results pane shows a single row of data from the Person.Person table.

BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo	Demographics	rowguid	ModifiedDate
1	EM	0	NULL	Ken	J	Sánchez	NULL	0	NULL	IndividualSurvey.xmlns="http://schemas.microsoft.com/SQL/Server/SQLCollation/SQL_Latin1_General_CI_AS_KS_WS"	92C4279F-1207-48A3-B448-4638514EB7E2	2009-01-07 00:00:00.000



The screenshot shows a SQL Server Enterprise Manager window with a query window titled 'SQLQuery7.sql - (\\...\\UELIQ1M\\nurse (56))*'. The query is 'SELECT * FROM sys.fn_get_audit_file('C:\\AuditLogs\\KisiAudit*.sqlaudit', NULL, NULL);'. The results pane shows two rows of data from the sys.fn_get_audit_file function.

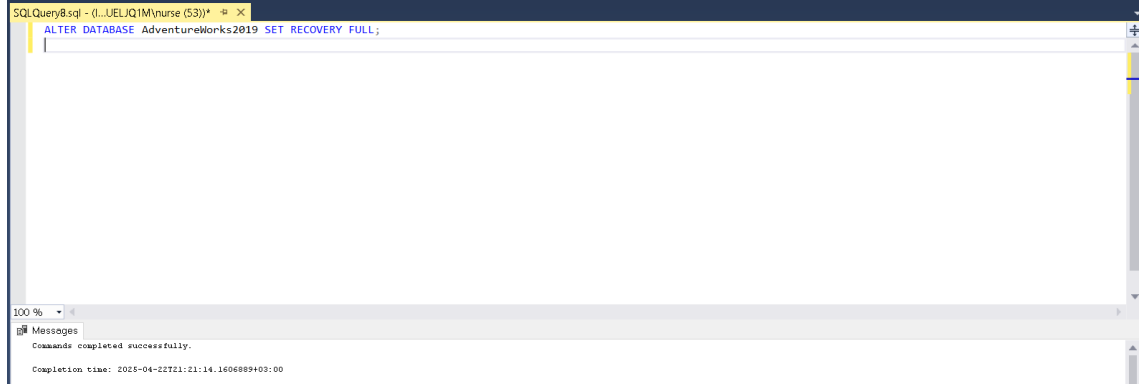
event_time	sequence_number	action_id	succeeded	permission_bitmask	is_column_permission	session_id	server_principal_id	database_principal_id	target_server_principal_id	target_database_principal_id	object_id
2025-04-22 18:46:28.9536515	1	AUDC	1	0x00000000000000000000000000000000	0	66	259	0	0	0	0
2025-04-22 18:49:43.5164282	1	SL	1	0x00000000000000000000000000000001	1	62	259	1	0	0	210

Bu işlem sayesinde SQL Server üzerinde yapılan önemli işlemleri denetleyip geçmişte neler yapıldığını kolayca görebiliyoruz. Özellikle kritik tablolar üzerinde yapılan işlemlerin takip edilmesi veri güvenliği açısından oldukça önemli.

Veritabanı Yedekleme ve Felaketten Kurtarma Planı (2. Proje)

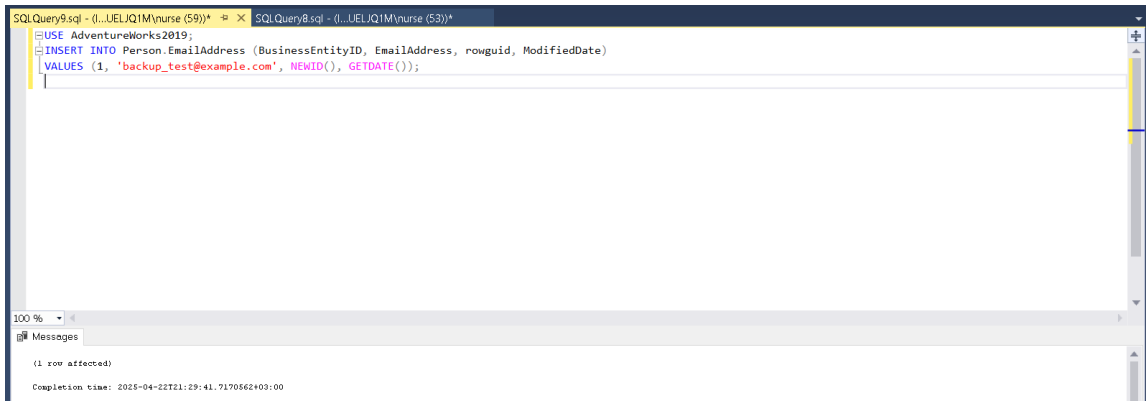
1. SQL Server Backup

Transaction Log yedeği alabilmek için recovery model ayarını yapıyoruz.



```
SQLQuery8.sql - (\\...\\UEL\\Q1M\\nurse (53)) *  
ALTER DATABASE AdventureWorks2019 SET RECOVERY FULL;  
  
Messages  
Commands completed successfully.  
Completion time: 2025-04-22T21:21:14.160688+03:00
```

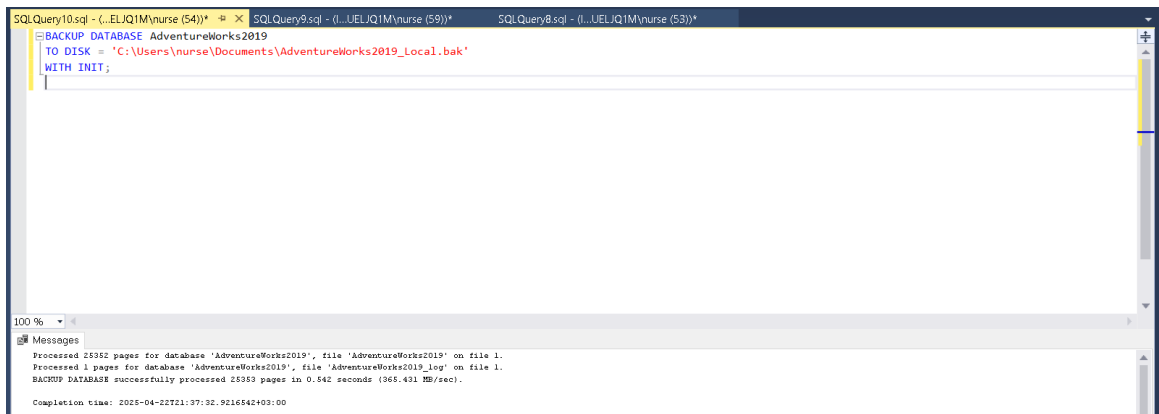
Yedekleme öncesi Person.EmailAddress tablosuna örnek veri eklendi:



```
SQLQuery9.sql - (\\...\\UEL\\Q1M\\nurse (59)) *  
USE AdventureWorks2019;  
INSERT INTO Person.EmailAddress (BusinessEntityID, EmailAddress, rowguid, ModifiedDate)  
VALUES (1, 'backup_test@example.com', NEWID(), GETDATE());  
  
Messages  
(1 row affected)  
Completion time: 2025-04-22T21:29:41.717056+03:00
```

SSMS üzerinden AdventureWorks2019 veritabanı için Full Backup alındı.

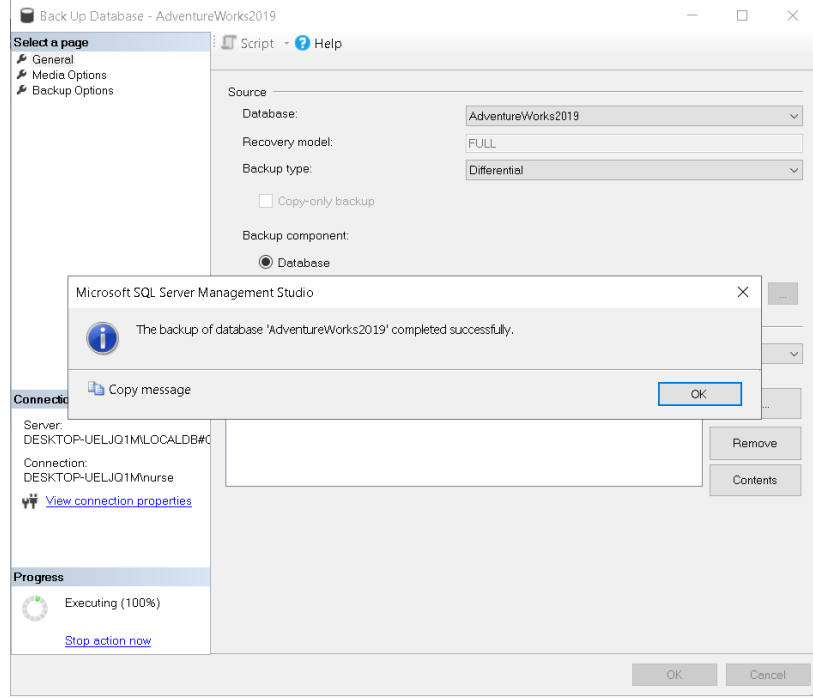
Backup Type: Full, dosya adı: AdventureWorks2019_Full.bak



```
SQLQuery10.sql - (\\...\\UEL\\Q1M\\nurse (54)) *  
BACKUP DATABASE AdventureWorks2019  
TO DISK = 'C:\\Users\\nurse\\Documents\\AdventureWorks2019_Local.bak'  
WITH INIT;  
  
Messages  
Processed 25352 pages for database 'AdventureWorks2019', file 'AdventureWorks2019' on file 1.  
Processed 1 pages for database 'AdventureWorks2019', file 'AdventureWorks2019_log' on file 1.  
BACKUP DATABASE successfully processed 25353 pages in 0.542 seconds (365.431 MB/sec).  
Completion time: 2025-04-22T21:37:32.921654+03:00
```

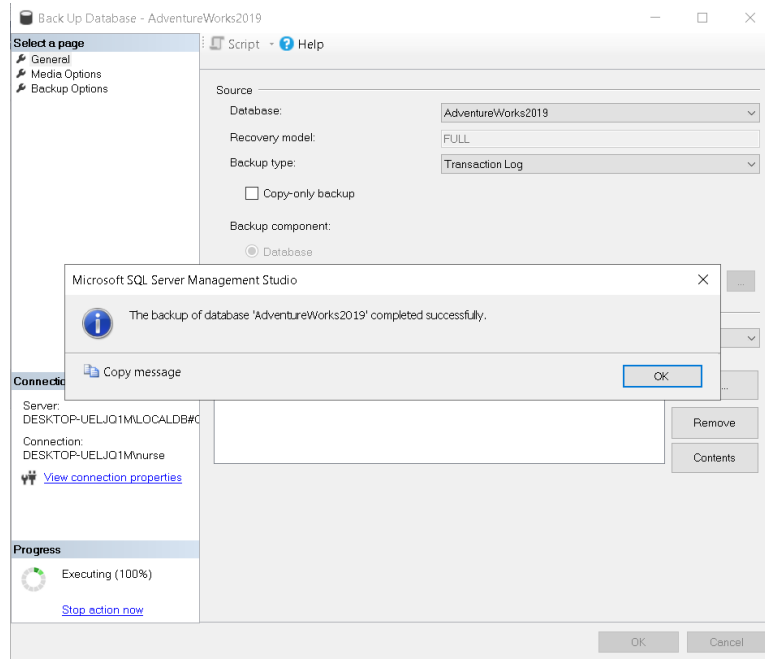
Yeni veri eklendikten sonra differential yedek alındı:

Backup Type: Differential, dosya adı: AdventureWorks2019_Diff.bak



Yeni veri eklendikten sonra transaction log yedeği alındı:

Backup Type: Transaction Log, dosya adı: AdventureWorks2019_Log.trn



2. Felaketten Kurtarma Senaryosu

Bu senaryoda, Foreign Key ilişkisi bulunmayan bir tabloyu (örneğin HumanResources.Department) yanırlıklıla silerek bir veri kaybı durumu simüle edilecek ve ardından önceden alınmış bir tam yedek dosyası (.bak) kullanılarak bu veri geri yüklenecektir.

İlk olarak, veritabanının mevcut durumunu korumak adına tam bir yedek alınır.

```
SQLQuery3.sql - (\\...\\UELQ1M\\nurse (52)) *  
USE master;  
GO  
BACKUP DATABASE AdventureWorks2019  
TO DISK = 'C:\\Backup\\AdventureWorks2019_FULL.bak'  
WITH FORMAT, INIT, NAME = 'Full Backup - AdventureWorks2019';  
  
Messages  
Processed 25302 pages for database 'AdventureWorks2019', file 'AdventureWorks2019' on file 1.  
Processed 1 pages for database 'AdventureWorks2019', file 'AdventureWorks2019_log' on file 1.  
BACKUP DATABASE successfully processed 25303 pages in 0.550 seconds (360.116 MB/sec).  
Completion time: 2025-04-25T00:10:33.7544208+03:00
```

```
SQLQuery7.sql - (\\...\\UELQ1M\\nurse (62)) * SQLQuery6.sql - (\\...\\UELQ1M\\nurse (54)) * SQLQuery3.sql - (\\...\\UELQ1M\\nurse (52)) *  
-- Tüm veritabanındaki FK'leri devre dışı bırak  
EXEC sp_msforeachtable "ALTER TABLE ? NOCHECK CONSTRAINT ALL";  
  
-- Silme işlemini yap  
DELETE FROM Person.Person;  
  
-- Tüm FK'leri tekrar aktif hale getir  
EXEC sp_msforeachtable "ALTER TABLE ? WITH CHECK CHECK CONSTRAINT ALL";  
  
Messages  
(0 rows affected)  
Completion time: 2025-04-25T00:31:08.5169506+03:00
```

Foreign Key kısıtlaması nedeniyle herhangi bir tabloyu direkt olarak silinemez. Bu yüzden test amaçlı olarak FOREIGN KEY'leri Geçici olarak devre dışı bırakarak Person.Person tablosu siliniyor.

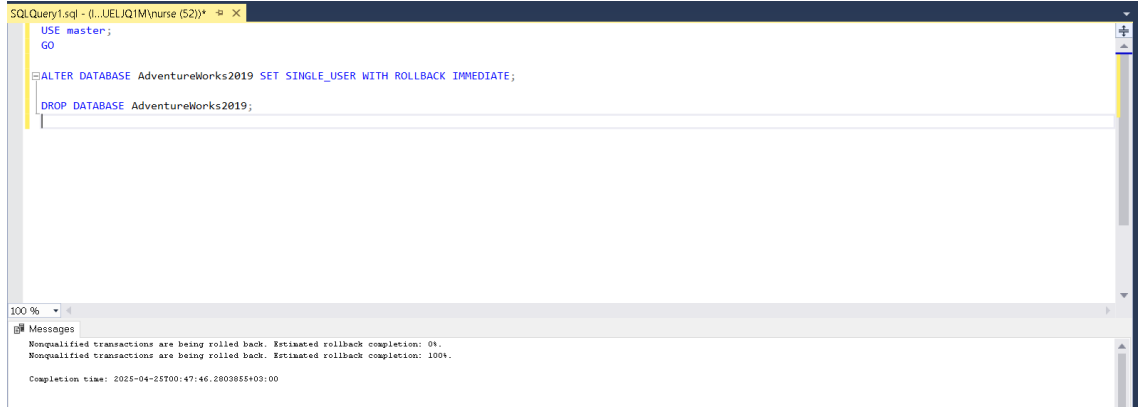
```
SQLQuery7.sql - (\\...\\UELQ1M\\nurse (62)) * SQLQuery6.sql - (\\...\\UELQ1M\\nurse (54)) * SQLQuery3.sql - (\\...\\UELQ1M\\nurse (52)) *  
USE AdventureWorks2019;  
GO  
SELECT COUNT(*) FROM Person.Person;  
  
Results Messages  
(No column name)  
1 0
```

Silme işlemi yapıldıktan sonra veritabanı çalışmaya devam ediyorsa, geri yükleme işleminden önce veritabanı **tek kullanıcı moduna** alınmalı ve aktif bağlantılar sonlandırılmalıdır. Ardından mevcut veritabanı silinmelidir.

Öncelikle veritabanını tek kullanıcı moduna alınır.

Daha sonra DROP DATABASE AdventureWorks2019; komutu ile veritabanını silinir.

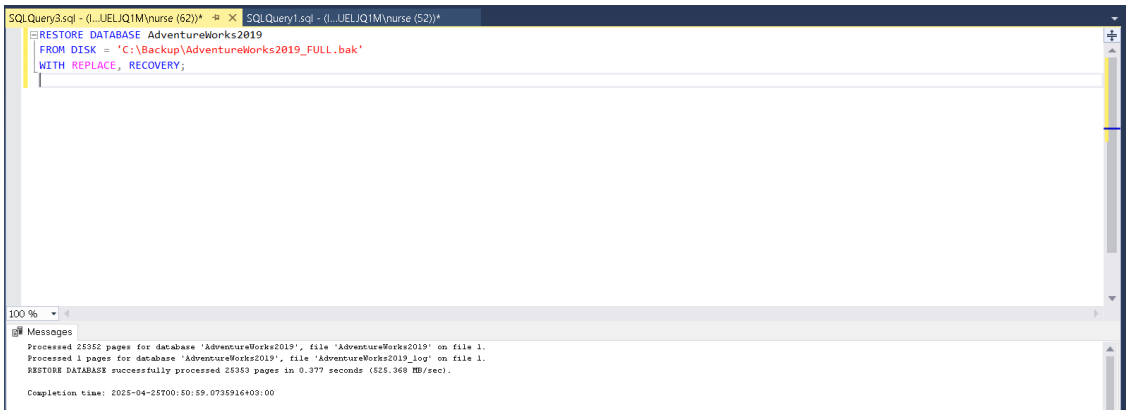
Daha önce alınan .bak dosyası (tam yedek) kullanılarak, veritabanı silinmeden önceki hâline geri getirilir.



```
SQLQuery1.sql - (\\...\\UELIQ1M\\nurse (52))*  
USE master;  
GO  
ALTER DATABASE AdventureWorks2019 SET SINGLE_USER WITH ROLLBACK IMMEDIATE;  
DROP DATABASE AdventureWorks2019;
```

100 %
Messages
Nonqualified transactions are being rolled back. Estimated rollback completion: 0%.
Nonqualified transactions are being rolled back. Estimated rollback completion: 100%.
Completion time: 2025-04-25T00:47:46.2803855403:00

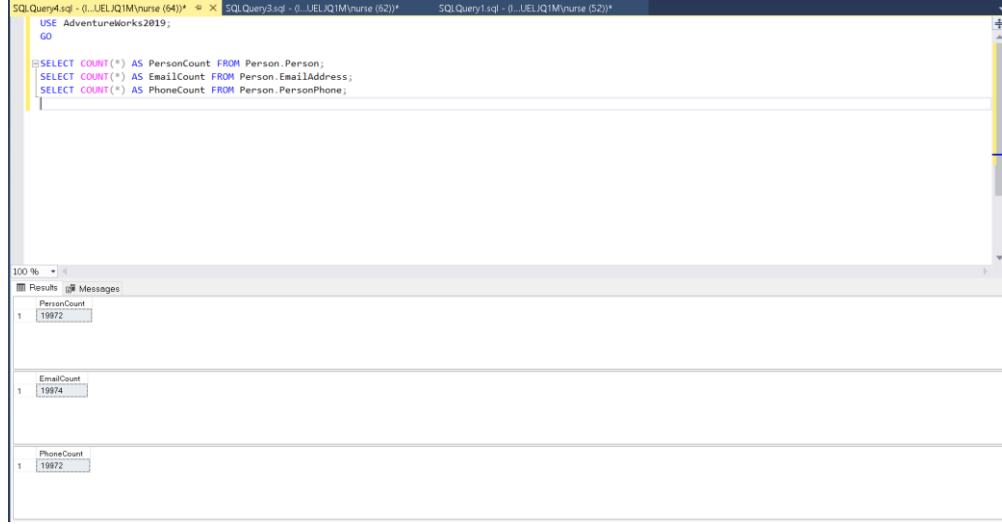
Veritabanının geri yüklenmesi için SSMS (SQL Server Management Studio) üzerinden master veritabanına bağlı bir sorgu penceresi açılarak aşağıdaki komut çalıştırılmıştır:



```
SQLQuery3.sql - (\\...\\UELIQ1M\\nurse (52))* SQLQuery1.sql - (\\...\\UELIQ1M\\nurse (52))*  
RESTORE DATABASE AdventureWorks2019  
FROM DISK = 'C:\\Backup\\AdventureWorks2019_FULL.bak'  
WITH REPLACE, RECOVERY;
```

100 %
Messages
Processed 25352 pages for database 'AdventureWorks2019', file 'AdventureWorks2019' on file 1.
Processed 1 pages for database 'AdventureWorks2019', file 'AdventureWorks2019_log' on file 1.
RESTORE DATABASE successfully processed 25353 pages in 0.377 seconds (525.369 MB/sec).
Completion time: 2025-04-25T00:50:59.0735916403:00

Veritabanı geri geldikten sonra, silinen tabloların içeriği aşağıdaki sorgular ile kontrol edilmiştir.



The screenshot shows three SQL queries in the SQL Server Enterprise Manager interface. The first query is a simple 'USE' statement. The second and third queries are 'SELECT COUNT(*)' statements for 'PersonCount', 'EmailCount', and 'PhoneCount' respectively. The results pane shows the counts for each table: PersonCount is 19972, EmailCount is 19974, and PhoneCount is 19972.

```
USE AdventureWorks2019;
GO

SELECT COUNT(*) AS PersonCount FROM Person.Person;
SELECT COUNT(*) AS EmailCount FROM Person.EmailAddress;
SELECT COUNT(*) AS PhoneCount FROM Person.PersonPhone;
```

PersonCount
19972

EmailCount
19974

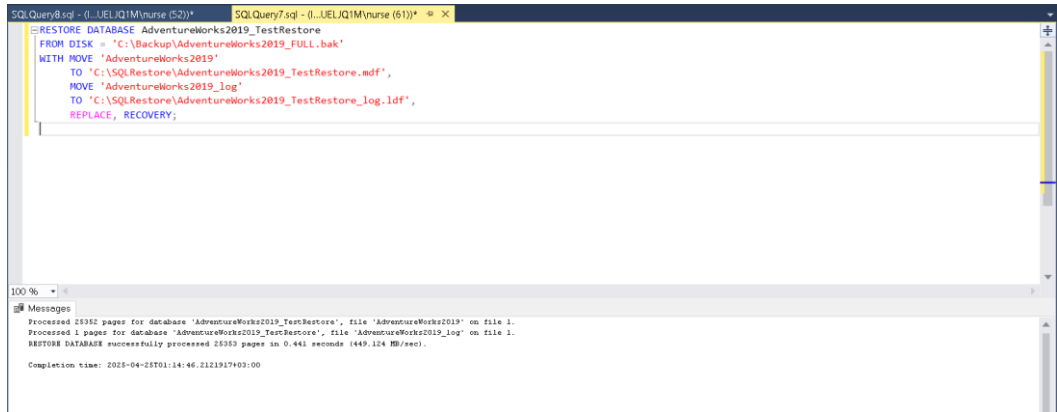
PhoneCount
19972

3. Test Yedekleme Senaryoları – Uygulama ve Doğrulama Aşamaları

Bu bölümde, AdventureWorks2019 veritabanı için önceden alınmış olan tam yedek dosyasının (.bak) sağlamlığını ve kullanılabilirliğini test etmek amacıyla geri yükleme işlemi gerçekleştirilmiştir. Geri yükleme farklı bir adla yapılmış ve ardından yedek dosyasının veri bütünlüğü test edilmiştir.

Alınan tam yedek dosyasının kullanılabilirliğini test etmek için .bak dosyası, AdventureWorks2019_TestRestore adında yeni bir veritabanı olarak geri yüklenmiştir. Geri yükleme işlemi sırasında veritabanı dosyalarının saklanacağı dizin olarak C:\SQLRestore\ klasörü seçilmiştir.

Bu komutun başarıyla tamamlanmasıyla birlikte sistem, AdventureWorks2019 veritabanının yedeğini farklı bir isimle ve farklı bir dizine geri yüklemiştir. Bu işlem yedeğin fiziksel olarak sağlam ve hatasız olduğunu doğrular.

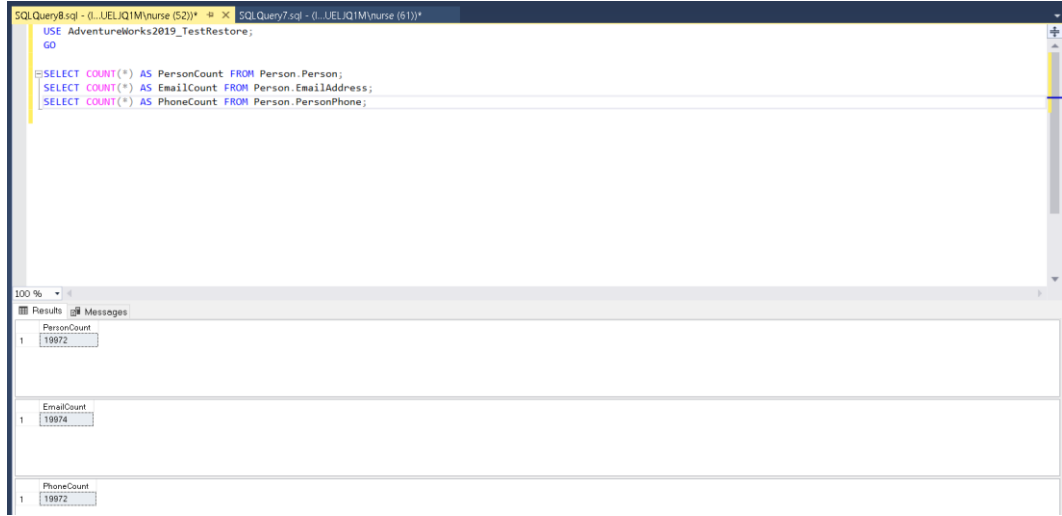


The screenshot shows a SQL query in the SQL Server Enterprise Manager interface. The query is a 'RESTORE DATABASE' command for 'AdventureWorks2019_TestRestore'. The command includes 'FROM DISK' pointing to a backup file, 'WITH MOVE' for the data and log files, and 'REPLACE, RECOVERY' options. The Messages pane shows the successful completion of the restore operation, including the number of pages processed and the completion time.

```
RESTORE DATABASE AdventureWorks2019_TestRestore
FROM DISK = 'C:\Backup\AdventureWorks2019_FULL.bak'
WITH MOVE 'AdventureWorks2019'
TO 'C:\SQLRestore\AdventureWorks2019_TestRestore.mdf',
MOVE 'AdventureWorks2019_log'
TO 'C:\SQLRestore\AdventureWorks2019_TestRestore_log.ldf',
REPLACE, RECOVERY;
```

Processed 25352 pages for database 'AdventureWorks2019_TestRestore', file 'AdventureWorks2019' on file 1.
Processed 1 pages for database 'AdventureWorks2019_TestRestore', file 'AdventureWorks2019_log' on file 1.
RESTORE DATABASE successfully processed 25353 pages in 0.441 seconds (469.124 MB/sec).
Completion time: 2025-04-25T01:14:46.2121917403:00

Yedek dosyasının eksiksiz ve kullanılabilir olduğunu göstermek için, geri yüklenen veritabanındaki bazı önemli tabloların veri sayıları sorgulanmıştır:



```
USE AdventureWorks2019_TestRestore;
GO

SELECT COUNT(*) AS PersonCount FROM Person.Person;
SELECT COUNT(*) AS EmailCount FROM Person.EmailAddress;
SELECT COUNT(*) AS PhoneCount FROM Person.PersonPhone;
```

PersonCount
19972

EmailCount
19974

PhoneCount
19972

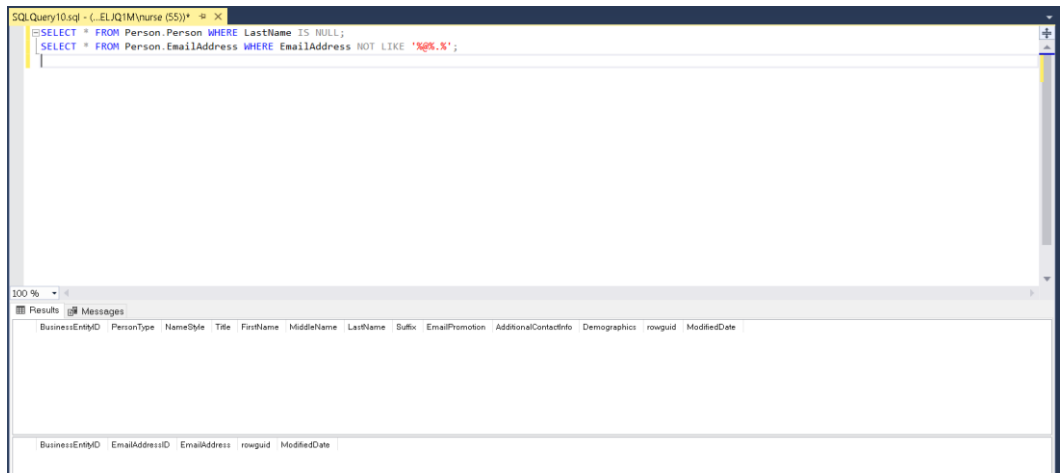
Veri Temizleme ve ETL Süreçleri Tasarımı (5.proje)

Bu projenin amacı, SQL Server üzerinde çalışan bir veritabanındaki hatalı, eksik ya da biçimsel olarak tutarsız verileri tespit edip temizlemek; ardından temiz verileri standart bir biçimde yeni bir tabloya aktarmak ve veri kalitesine dair temel raporlar oluşturmaktır. Bu süreç, ETL (Extract, Transform, Load) modelinin temel prensiplerini uygulamalı olarak yansıtır.

1.Veritabanı Temizleme Denetimi

İlk adımda, AdventureWorks2019 veritabanındaki Person.Person ve Person.EmailAddress gibi sık kullanılan tablolar taranarak NULL değerler, tutarsız veriler ve hatalı formatlar araştırılmıştır.

Bu sorgular sonucunda herhangi bir **eksik, hatalı veya uygunsuz veri bulunmamıştır**. AdventureWorks2019, Microsoft tarafından test ve öğretim amaçlı oluşturulmuş bir veritabanı olduğundan, veri kalitesi açısından oldukça yüksek standartlara sahiptir.



```
SELECT * FROM Person.Person WHERE LastName IS NULL;
SELECT * FROM Person.EmailAddress WHERE EmailAddress NOT LIKE '%@%.%';
```

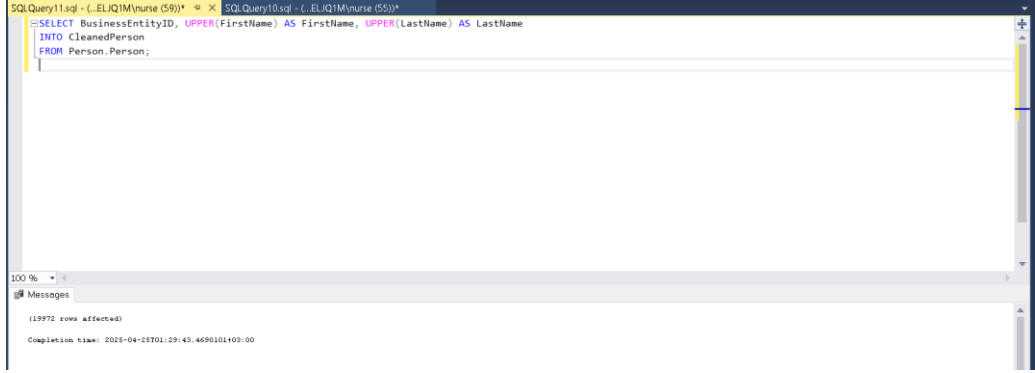
BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo	Demographics	rowguid	ModifiedDate
------------------	------------	-----------	-------	-----------	------------	----------	--------	----------------	-----------------------	--------------	---------	--------------

BusinessEntityID	EmailAddressID	EmailAddress	rowguid	ModifiedDate
------------------	----------------	--------------	---------	--------------

2. Dönüştürme İşlemleri (Transform)

Hatalı veri bulunmamasına rağmen, veri üzerinde örnek bir dönüşüm işlemi gerçekleştirilmiştir. Bu, ETL sürecinin "Transform" adımını simüle etmek amacıyla yapılmıştır.

Bu işlemle, Person.Person tablosundaki verilerin ad ve soyad sütunları büyük harfe çevrilmiş ve sonuçlar CleanedPerson adlı yeni bir tabloya aktarılmıştır.

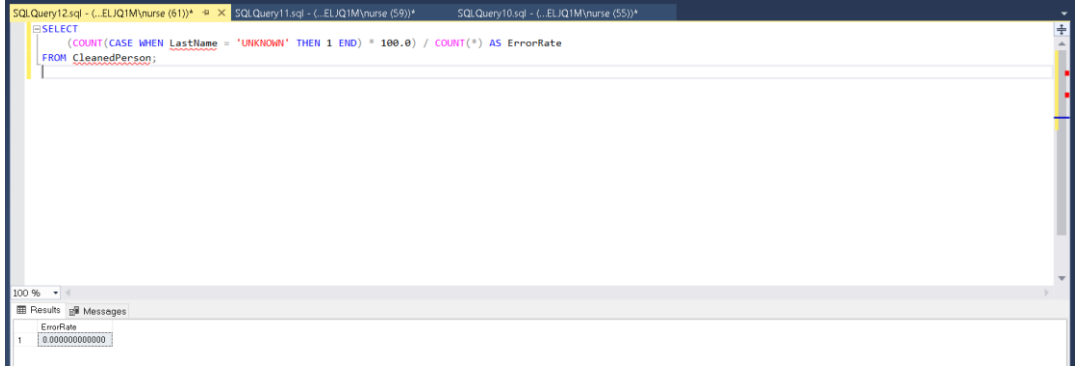


```
SQL Query11.sql - (E:\JQM\murse (59)) * SQL Query10.sql - (E:\JQM\murse (55)) *
SELECT BusinessEntityID, UPPER(FirstName) AS FirstName, UPPER(LastName) AS LastName
INTO CleanedPerson
FROM Person.Person;
```

100 %
Messages
(19972 rows affected)
Completion time: 2025-04-23T01:29:43.4690101+03:00

3. Veri Kalitesi Raporu

Hatalı veriler olmamasına rağmen, örnek bir veri kalitesi raporu senaryosu hazırlanmıştır. Aşağıdaki sorgu ile UNKNOWN gibi varsayılan atama yapılmış kayıtların oranı hesaplanmıştır.



```
SQL Query12.sql - (E:\JQM\murse (61)) * SQL Query11.sql - (E:\JQM\murse (59)) * SQL Query10.sql - (E:\JQM\murse (55)) *
SELECT
    (COUNT(CASE WHEN LastName = 'UNKNOWN' THEN 1 END) * 100.0) / COUNT(*) AS ErrorRate
FROM CleanedPerson;
```

100 %
Results Messages
ErrorRate
1 0.00000000000000

Bu sonuç, veri kalitesinin yüksek olduğunu ve temizleme gereksinimi bulunmadığını göstermektedir.

Veri tabanı Performans Optimizasyonu ve İzleme (1.proje)

Bu projede, veritabanı üzerindeki yavaş sorgular ve yüksek kaynak tüketimi oluşturan işlemler tespit edilip optimize edilecektir. Ayrıca indeks yönetimi ve veritabanı izleme teknikleri uygulanacaktır.

1. Yavaş Sorguların Tespiti

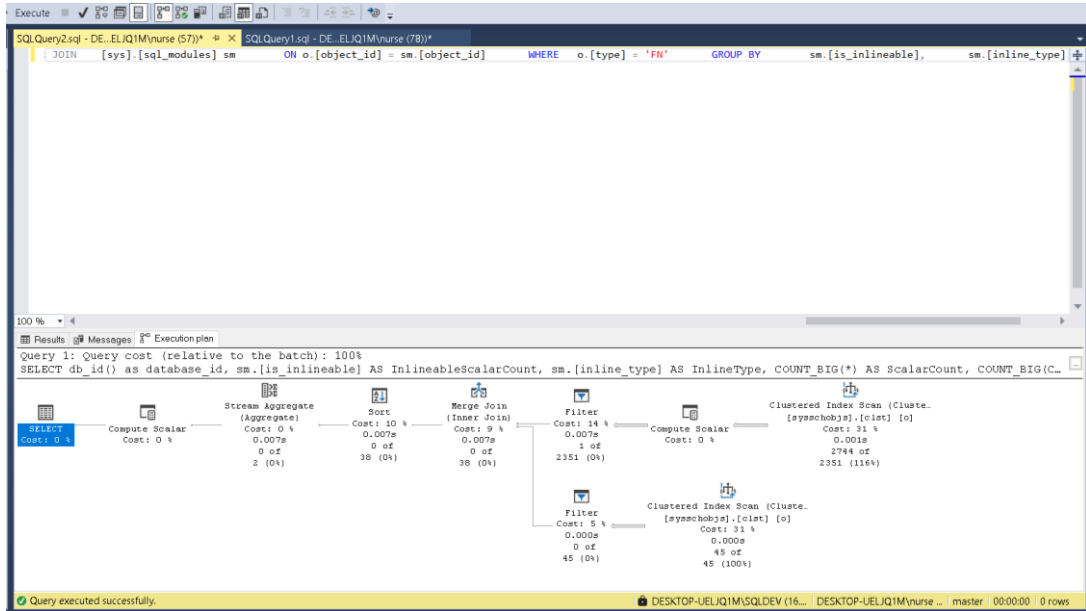
Veritabanı üzerinde çalışan sorguların performansını izlemek için sys.dm_exec_query_stats DMV'si kullanılmıştır. Bu sayede en yüksek ortalama çalışma süresine sahip ilk 10 sorgu listelenmiştir.



Rank	AvgElapsedTime	ExecutionCount	QueryText
1	267079	1	SELECT db_id() as database_id, sm.[is_inlinable] AS InlinableScalarCount, sm.[inline_type] AS Inlin...
2	148520	1	SELECT db_id() AS database_id, c.system_type_id, c.user_type_id, c.is_sparse, c.is_column_t...
3	107792	1	IF EXISTS (SELECT TOP 1 container_uri FROM managed_backup_hn_backup_db_config(NULL) WHERE contai...
4	91360	1	IF EXISTS (SELECT TOP 1 container_uri FROM managed_backup_hn_backup_instance_config() WHERE contai...
5	51578	1	IF EXISTS (SELECT TOP 1 credential_name FROM smart_admin_hn_backup_db_config(NULL) WHERE credent...
6	49752	1	SELECT db_id() AS database_id, o.type AS object_type, b.type AS index_type, p.data_compressio...
7	40958	1	IF EXISTS (SELECT TOP 1 credential_name FROM smart_admin_hn_backup_instance_config() WHERE credent...
8	33558	1	WITH TablesAndViews AS (SELECT object_id, 'table' AS object_type FROM sys.tables WITH(INCLUDE) UNION ...
9	33200	1	SELECT CASE WHEN name like '%msdtcp.dll%' THEN 'msdtcp' WHEN name like '%sqlqdbc.x...
10	31411	1	SELECT DB_ID() AS database_id, it.remote_data_archive_enabled, temporal_type, is_memory_opti...

2. Execution Plan Analizi

Yavaş sorgulardan biri Execution Plan sekmesinde analiz edilmiştir. Bu analiz sonucunda sorgunun tabloyu doğrudan taradığı ve Clustered Index Scan yaptığı görülmüştür. Bu da performans sorununa işaret etmektedir.



3. Eksik İndeks Önerisi ve Oluşturulması

sys.dm_db_missing_index_details DMV'si ile yapılan analizde, Sales.SalesOrderHeader tablosuna yönelik bir eksik indeks önerisi elde edilmiştir. Bu öneri doğrultusunda aşağıdaki indeks oluşturulmuştur.

```
SQLQuery3.sql - DE..ELJQ1M\nurse (61))* SQLQuery2.sql - DE..ELJQ1M\nurse (57))* SQLQuery1.sql - DE..ELJQ1M\nurse (78))*
SELECT
    migs.avg_total_user_cost * migs.avg_user_impact * (migs.user_seeks + migs.user_scans) AS ImprovementMeasure,
    mid.statement AS TableName,
    mid.equality_columns,
    mid.inequality_columns,
    mid.included_columns
FROM sys.dm_db_missing_index_groups mig
JOIN sys.dm_db_missing_index_group_stats migs ON migs.group_handle = mig.index_group_handle
JOIN sys.dm_db_missing_index_details mid ON mig.index_handle = mid.index_handle
ORDER BY ImprovementMeasure DESC;
```

ImprovementMeasure	TableName	equality_columns	inequality_columns	included_columns
158.02055394	[AdventureWorks2019].[Sales].[SalesOrderHeader]	NULL	[OrderDate] [SubTotal]	[TaxAmt] [Freight]

```
SQLQuery5.sql - DE..ELJQ1M\nurse (58))* SQLQuery3.sql - DE..ELJQ1M\nurse (61))* SQLQuery2.sql - DE..ELJQ1M\nurse (57))* SQLQuery1.sql - DE..ELJQ1M\nurse (78))*
CREATE NONCLUSTERED INDEX IX_SalesOrderHeader_OrderDate_SubTotal
ON Sales.SalesOrderHeader (OrderDate, SubTotal)
INCLUDE (TaxAmt, Freight);
```

Commands completed successfully.

Completion time: 2025-05-27T18:41:37.2578019+03:00

4. Performans İyileşmesinin Gözlemlenmesi

İndeks oluşturulduktan sonra aynı sorgu tekrar çalıştırılmış ve Execution Plan analizinde bu kez Index Seek işleminin kullanıldığı görülmüştür. Bu, sorgunun artık daha hızlı çalıştığını göstermektedir.

```
SQLQuery6.sql - DE..ELJQ1M\nurse (53))* SQLQuery5.sql - DE..ELJQ1M\nurse (58))* SQLQuery3.sql - DE..ELJQ1M\nurse (61))* SQLQuery2.sql - DE..ELJQ1M\nurse (57))*
SELECT SalesOrderID, OrderDate, SubTotal, TaxAmt, Freight
FROM Sales.SalesOrderHeader
WHERE OrderDate BETWEEN '2013-01-01' AND '2013-06-30'
AND SubTotal > 1000;
```

Query 1: Query cost (relative to the batch): 100%

SELECT [SalesOrderID], [OrderDate], [SubTotal], [TaxAmt], [Freight] FROM [Sales].[SalesOrderHeader] WHERE [OrderDate]>=@1 AND [OrderDate]<=@2 AND [SubTo...

Index Seek (NonClustered)
[SalesOrderHeader].[IX_Sales...
Cost: 100 %
0.000s
2107 of
1052 (113%)

5. Gereksiz İndeks Analizi

sys.dm_db_index_usage_stats DMV'si ile sistemde kullanılmayan ancak güncellenen indeksler sorgulanmıştır. Ancak AdventureWorks2019 veritabanı kısa süre önce yüklendiği için DMV henüz yeterli veri üretmemiştir ve sorgu sonucu boştur.

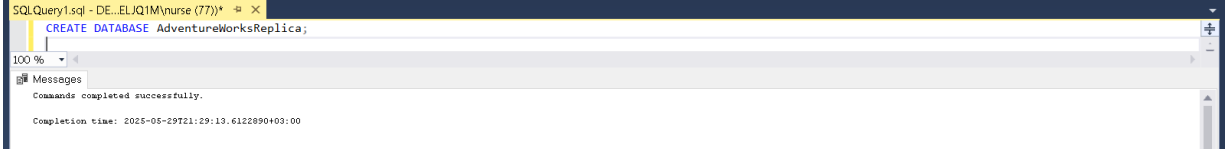
```
SQLQuery7.sql - DE..ELJQ1M\nurse (52))* SQLQuery6.sql - DE..ELJQ1M\nurse (53))* SQLQuery5.sql - DE..ELJQ1M\nurse (58))* SQLQuery3.sql - DE..ELJQ1M\nurse (61))*
-- Gereksiz indekslerin tespiti
SELECT
    OBJECT_NAME(i.object_id) AS TableName,
    i.name AS IndexName,
    i.index_id,
    user_seeks, user_scans, user_lookups, user_updates
FROM sys.dm_db_index_usage_stats us
JOIN sys.indexes i ON i.object_id = us.object_id AND i.index_id = us.index_id
WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1
AND user_seeks = 0
AND user_scans = 0
AND user_lookups = 0
ORDER BY user_updates DESC;
```

TableName	IndexName	index_id	user_seeks	user_scans	user_lookups	user_updates
-----------	-----------	----------	------------	------------	--------------	--------------

Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları (4.proje)

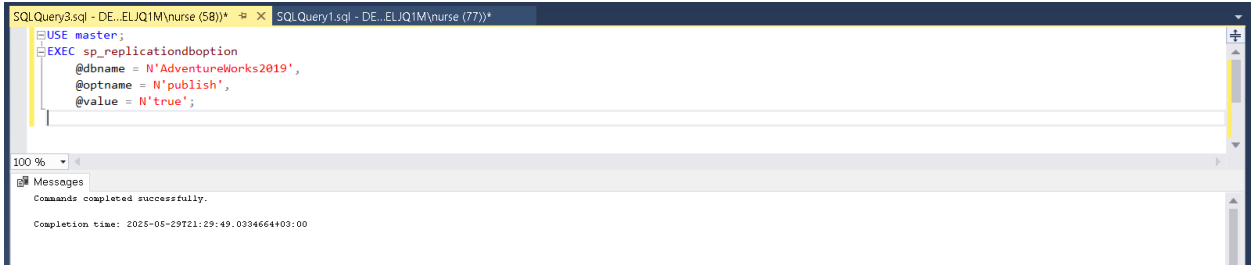
1. Subscriber Veritabanı Oluşturulması

Replikasyonun veri aktaracağı hedef veri tabanı olarak AdventureWorksReplica adlı yeni bir veri tabanı oluşturulmuştur.



2. Replikasyonun Etkinleştirilmesi

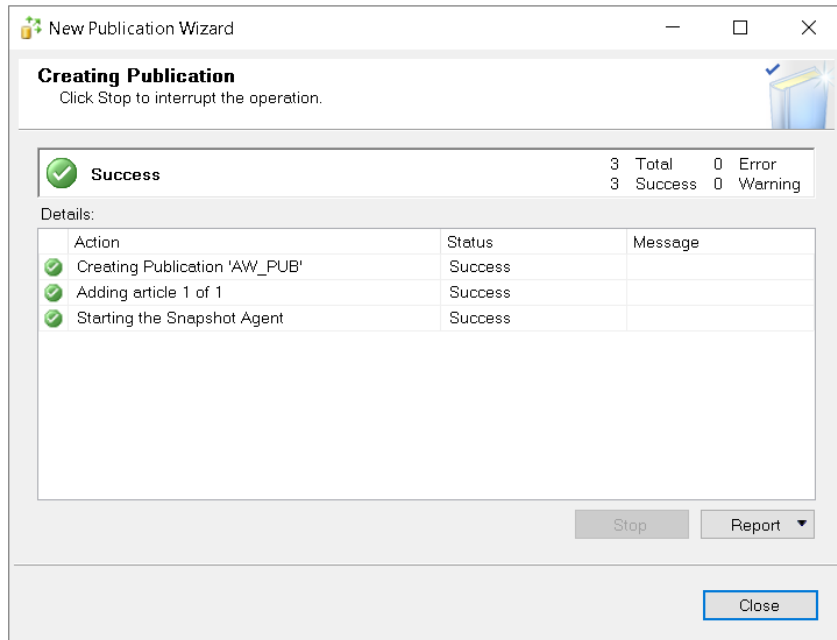
AdventureWorks2019 veritabanı için replikasyon özelliği etkinleştirilmiştir.



3. Publication Oluşturulması

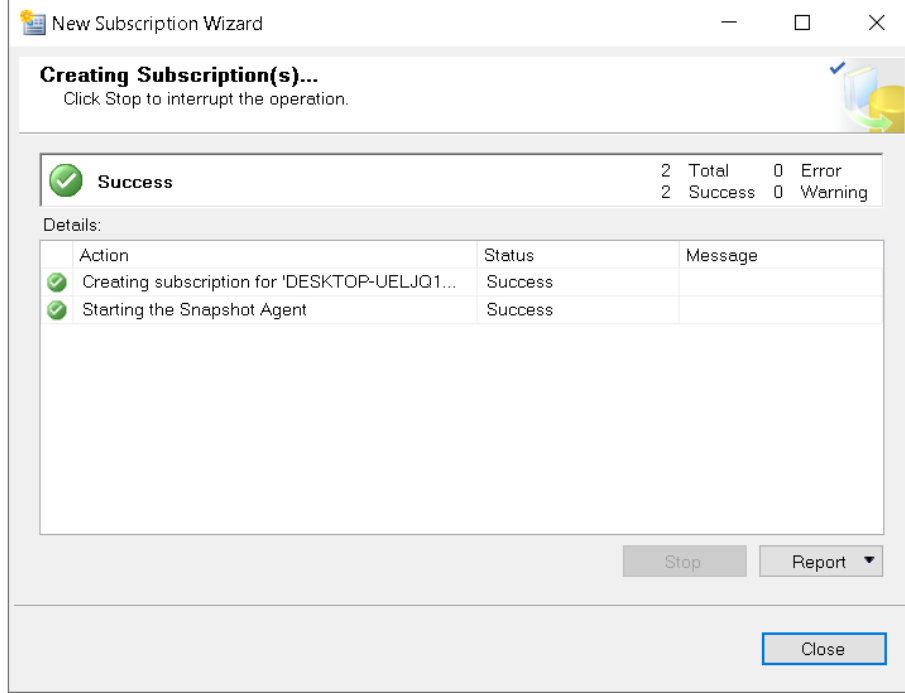
SQL Server Management Studio (SSMS) kullanılarak yeni bir publication oluşturulmuştur:

- Veritabanı olarak AdventureWorks2019 seçilmiştir.
- Replikasyon türü olarak Transactional Replication seçilmiştir.
- Replike edilecek tablo olarak Person.Person belirlenmiştir.
- Snapshot Agent otomatik olarak başlatılmış ve başarılı şekilde tamamlanmıştır.



4. Subscription Oluřturulması

Publisher'dan veri alacak abonelik AdventureWorksReplica veritabanı iin oluřturulmuřtur. Subscription Wizard bařarıyla tamamlanmıř ve Snapshot Agent tekrar alıřtırılmıřtır.



Veritabanı Yükseltme ve Sürüm Yönetimi (6.proje)

Bu projenin amacı, AdventureWorks2019 veritabanı üzerinde yapılan yapısal değişikliklerin otomatik olarak takip edilmesini sağlayarak bir sürüm kontrol sistemi oluşturmaktır. Bu kapsamda, DDL (Data Definition Language) trigger kullanılarak tablo oluşturma, değiştirme ve silme işlemleri otomatik olarak loglanmış; bu işlemler geri döndürülebilir hale getirilmiştir.

1. Sürüm Yükseltme Simülasyonu: Yeni Tablo Oluşturma

AdventureWorks2019 veritabanına “Customers_New” adında yeni bir tablo eklenmiştir. Bu işlem, veri yapısının yeni sürüme geçiş sürecini simüle etmektedir.

```
SQLQuery4.sql - DE...ELJQ1M\vnurse (66)*
USE AdventureWorks2019;
GO

CREATE TABLE Customers_New (
    CustomerID INT PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Email NVARCHAR(100)
);
```

100 %

Messages

Commands completed successfully.

Completion time: 2025-05-26T19:10:47.2564147+03:00

2. Şema Değişikliklerini Loglamak için Tablo

Veritabanı üzerinde yapılan CREATE, ALTER ve DROP işlemlerini izlemek amacıyla “SchemaChangeLog” adında özel bir log tablosu oluşturulmuştur.

```
SQLQuery5.sql - DE...ELJQ1M\vnurse (77)* SQLQuery4.sql - DE...ELJQ1M\vnurse (66)*
CREATE TABLE SchemaChangeLog (
    ChangeID INT IDENTITY(1,1) PRIMARY KEY,
    EventType NVARCHAR(100),
    ObjectName NVARCHAR(100),
    TSQLCommand NVARCHAR(MAX),
    ChangeDate DATETIME DEFAULT GETDATE()
);
```

100 %

Messages

Commands completed successfully.

Completion time: 2025-05-26T19:12:09.3722077+03:00

3. DDL Trigger ile Otomatik Takip

Veritabanı seviyesinde tanımlanan bir DDL Trigger sayesinde tablo oluşturma, değiştirme ve silme işlemleri otomatik olarak loglanmaktadır. Bu sayede manuel takip ihtiyacı ortadan kaldırılmıştır.

```
SQLQuery6.sql - DE...ELJQ1M\vnurse (68)* SQLQuery5.sql - DE...ELJQ1M\vnurse (77)* SQLQuery4.sql - DE...ELJQ1M\vnurse (66)*
CREATE TRIGGER trg_SchemaChangeLog
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
    INSERT INTO SchemaChangeLog (EventType, ObjectName, TSQLCommand)
    SELECT
        EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'NVARCHAR(MAX)')
END;
```

100 %

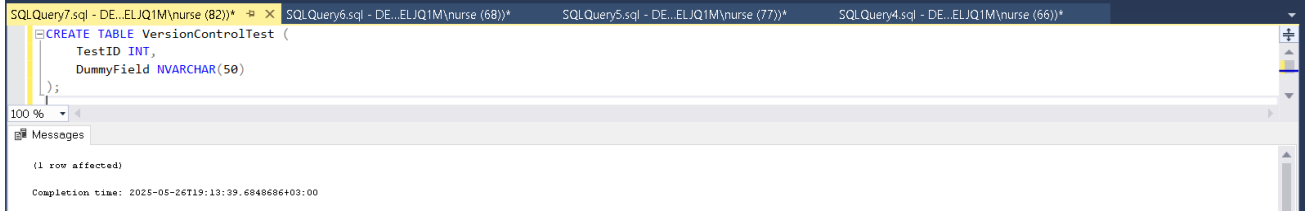
Messages

Commands completed successfully.

Completion time: 2025-05-26T19:12:40.1821478+03:00

4. Test Tablosu ile Trigger Kontrolü

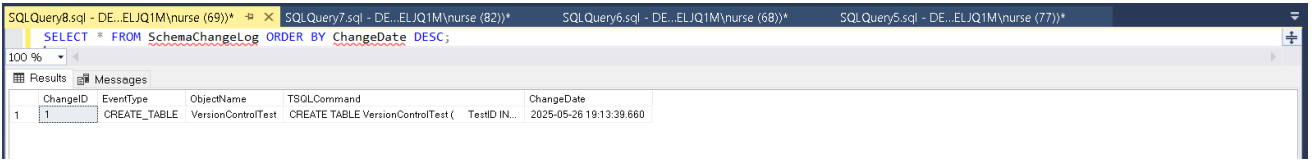
Test amacıyla “VersionControlTest” adında bir tablo oluşturulmuş, bu işlem trigger tarafından algılanarak SchemaChangeLog tablosuna kaydedilmiştir.



```
SQLQuery7.sql - DE...ELJQ1M\nurse (82))* SQLQuery6.sql - DE...ELJQ1M\nurse (68))* SQLQuery5.sql - DE...ELJQ1M\nurse (77))* SQLQuery4.sql - DE...ELJQ1M\nurse (66))*
CREATE TABLE VersionControlTest (
    TestID INT,
    DummyField NVARCHAR(50)
);
100 %
Messages
(1 row affected)
Completion time: 2025-05-26T19:13:39.684866+03:00
```

5. Log Kayıtlarının İncelenmesi

SchemaChangeLog tablosu üzerinden yapılan sorguda, CREATE_TABLE işleminin başarıyla yakalandığı ve detaylarının kaydedildiği görülmüştür.

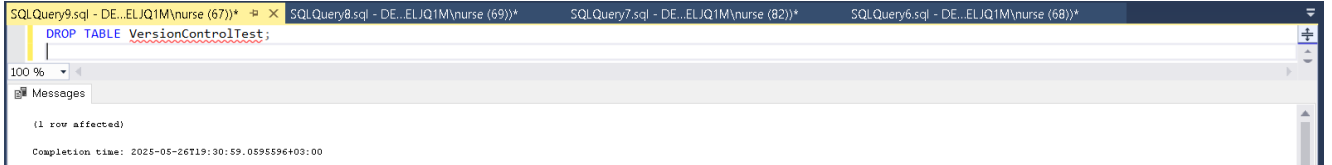


```
SQLQuery8.sql - DE...ELJQ1M\nurse (69))* SQLQuery7.sql - DE...ELJQ1M\nurse (82))* SQLQuery6.sql - DE...ELJQ1M\nurse (68))* SQLQuery5.sql - DE...ELJQ1M\nurse (77))*
SELECT * FROM SchemaChangeLog ORDER BY ChangeDate DESC;
100 %
Results Messages
ChangeID EventType ObjectName TSOLCommand ChangeDate
1 1 CREATE_TABLE VersionControlTest CREATE TABLE VersionControlTest ( TestID IN... 2025-05-26 19:13:39.660
```

6. Test ve Geri Dönüş Planı

Proje kapsamında oluşturulan DDL trigger, veritabanı şemasında yapılan tüm CREATE, ALTER ve DROP işlemlerini logladığı için herhangi bir yapısal değişikliğin geri alınabilirliğini garanti altına almaktadır.

Test tablosunun yanlışlıkla oluşturulduğunu varsayarsak, aşağıdaki komutla silinebilir



```
SQLQuery9.sql - DE...ELJQ1M\nurse (67))* SQLQuery8.sql - DE...ELJQ1M\nurse (69))* SQLQuery7.sql - DE...ELJQ1M\nurse (82))* SQLQuery6.sql - DE...ELJQ1M\nurse (68))*
DROP TABLE VersionControlTest;
100 %
Messages
(1 row affected)
Completion time: 2025-05-26T19:30:59.0595596+03:00
```

Bu silme işlemi de trigger tarafından loglanacaktır. Böylece loglara bakarak aynı tabloyu yeniden oluşturmak mümkün hale gelir.

Bu yapı sayesinde her tablo değişikliği izlenebilir hale getirilmiş ve sürüm takibi etkin bir şekilde sağlanmıştır. Sistem, gelecekteki sürüm geçişleri ya da beklenmeyen yapı değişikliklerine karşı koruma sağlamaktadır.

Veritabanı Yedekleme ve Otomasyon Çalışması (7.proje)

Bu projenin amacı AdventureWorks2019 veritabanının düzenli ve otomatik bir şekilde yedeklenmesini sağlamak, SQL Server Agent kullanarak günlük yedekleme job'ı oluşturmak ve bu yedekleme işlemlerine dair raporlama yapılmasını sağlamaktır.

1. SQL Server Agent Job Tanımı

İlk olarak "AW_Daily_Backup_Job" adında bir job oluşturulmuştur. Bu job, veritabanı yedekleme işlemini otomatikleştirmek amacıyla yapılandırılmıştır.

The 'New Job' dialog box is shown with the following details:

- Select a page:** General, Steps, Schedules, Alerts, Notifications, Targets.
- Name:** AW_Daily_Backup_Job
- Owner:** DESKTOP-UELJO1M\nurse
- Category:** [Uncategorized (Local)]
- Description:** (Empty text area)
- Connection:** Server: DESKTOP-UELJO1MSOLDEV, Connection: DESKTOP-UELJO1M\nurse, View connection properties
- Progress:** Ready
- Buttons:** OK, Cancel

2. Yedekleme Komutu – Job Step

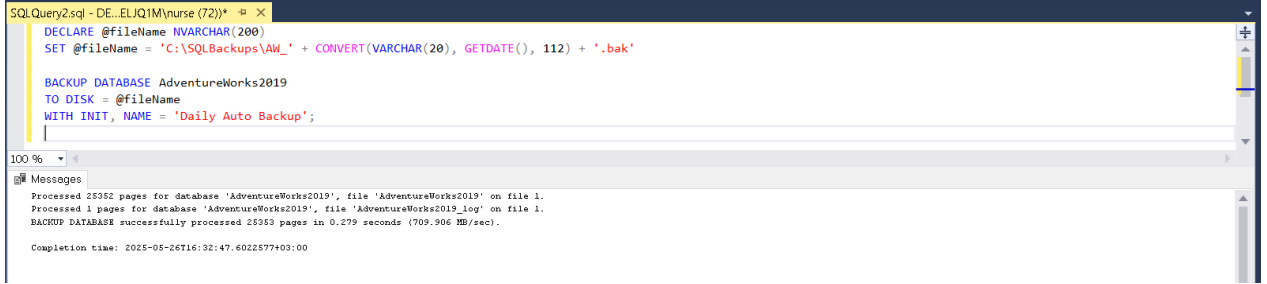
Job içinde "BackupStep" adlı bir adım tanımlanmış ve bu adımda T-SQL komutu ile günlük yedekleme işlemi gerçekleştirilmiştir. Dosya adları dinamik olarak tarih içerecek şekilde ayarlanmıştır.

The 'New Job Step' dialog box is shown with the following details:

- Select a page:** General, Advanced.
- Step name:** BackupStep
- Type:** Transact-SQL script (T-SQL)
- Run as:** (Empty dropdown)
- Database:** AdventureWorks2019
- Command:** (Empty text area)
- Buttons:** Open..., Select All, Copy, Paste, Parse
- Connection:** Server: DESKTOP-UELJO1MSOLDEV, Connection: DESKTOP-UELJO1M\nurse, View connection properties
- Progress:** Ready
- Buttons:** Previous, Next, OK, Cancel

3. Test Amaçlı Elle Backup Alma

Otomatik işlemten önce yukarıdaki komut manuel olarak çalıştırılarak test edilmiştir. Komut sorunsuz çalışmış ve yedekleme işlemi başarıyla tamamlanmıştır.



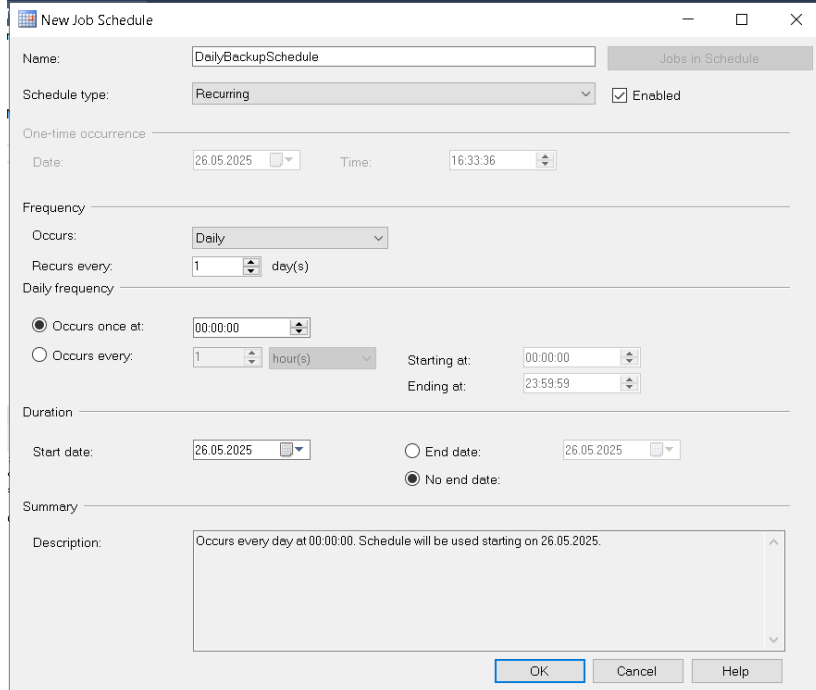
```
SQLQuery2.sql - DE...ELJQ1M\nurse (72)) *  X
DECLARE @fileName NVARCHAR(200)
SET @fileName = 'C:\SQLBackups\AW_' + CONVERT(VARCHAR(20), GETDATE(), 112) + '.bak'

BACKUP DATABASE AdventureWorks2019
TO DISK = @fileName
WITH INIT, NAME = 'Daily Auto Backup';

100 %
Messages
Processed 25352 pages for database 'AdventureWorks2019', file 'AdventureWorks2019' on file 1.
Processed 1 pages for database 'AdventureWorks2019', file 'AdventureWorks2019_log' on file 1.
BACKUP DATABASE successfully processed 25353 pages in 0.279 seconds (709.906 MB/sec).
Completion time: 2025-05-26T16:32:47.602577+03:00
```

4. Günlük Zamanlama Ayarları

Job, her gün saat 00:00:00'da çalışacak şekilde zamanlanmıştır. Schedule adı "DailyBackupSchedule" olarak tanımlanmıştır.



The 'New Job Schedule' dialog box is shown with the following settings:

- Name: DailyBackupSchedule
- Schedule type: Recurring
- Enabled: ☒
- One-time occurrence: Date: 26.05.2025, Time: 16:33:36
- Frequency: Occurs: Daily, Recurs every: 1 day(s)
- Daily frequency: ☒ Occurs once at: 00:00:00, ☐ Occurs every: 1 hour(s), Starting at: 00:00:00, Ending at: 23:59:59
- Duration: Start date: 26.05.2025, ☐ End date: 26.05.2025, ☒ No end date
- Summary: Description: Occurs every day at 00:00:00. Schedule will be used starting on 26.05.2025.

5. Yedekleme Raporlama

Aşağıdaki T-SQL sorgusu ile AdventureWorks2019 veritabanına ait geçmiş yedekleme işlemleri sorgulanmıştır. Her bir yedeğin başlangıç ve bitiş zamanı, süresi, boyutu ve dosya yolu listelenmiştir

SQLQuery3.sql - DE...ELQ1M\nurse (71))* SQLQuery2.sql - DE...ELQ1M\nurse (72))*

```
SELECT
    database_name,
    backup_start_date,
    backup_finish_date,
    DATEDIFF(SECOND, backup_start_date, backup_finish_date) AS DurationSeconds,
    backup_size / 1024 / 1024 AS Size_MB,
    physical_device_name
FROM msdb.dbo.backupset b
JOIN msdb.dbo.backupmediafamily m
    ON b.media_set_id = m.media_set_id
WHERE database_name = 'AdventureWorks2019'
ORDER BY backup_finish_date DESC;
```

100 %

Results Messages

	database_name	backup_start_date	backup_finish_date	DurationSeconds	Size_MB	physical_device_name
1	AdventureWorks2019	2025-05-26 16:32:47.000	2025-05-26 16:32:47.000	0	199.08203125000	C:\SQLBackups\Aw_20250526.bak
2	AdventureWorks2019	2025-05-26 16:28:03.000	2025-05-26 16:28:03.000	0	199.08203125000	C:\SQLBackups\AdventureWorks2019_FULL.bak
3	AdventureWorks2019	2023-05-08 12:08:45.000	2023-05-08 12:08:45.000	0	199.08593750000	C:\SQLBackups\AdventureWorks2019.bak