

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



VİZE PROJE RAPORU

(BLM4522) AĞ TABANLI PARALEL DAĞITIM SİSTEMLERİ

Perizat SAGYNBEKOVA

21290895

GitHub (<https://github.com/lmpasbaa/MSSQL.git>)

25.04.2025

PROJE 1. VERİTABANI PERFORMANS OPTİMİZASYONU VE İZLEME

➤ **PROJE AMACI:** Bu proje kapsamında SQL Server Management Studio (SSMS) üzerinde örnek bir büyük veritabanı (AdventureWorks) kullanılarak performans izleme, indeks yönetimi, disk alanı kullanımı, veri yoğunluğu yönetimi ve optimizasyon konularında işlemler gerçekleştirmektir. Sistemin daha verimli çalışmasını sağlamak ve ileri düzey veritabanı yönetimi becerilerini geliştirmektir. Ayrıca, sistemdeki performansı izlemek için SQL Server araçları kullanılır.

➤ **KULLANILAN VERİTABANI:** AdventureWorks2022

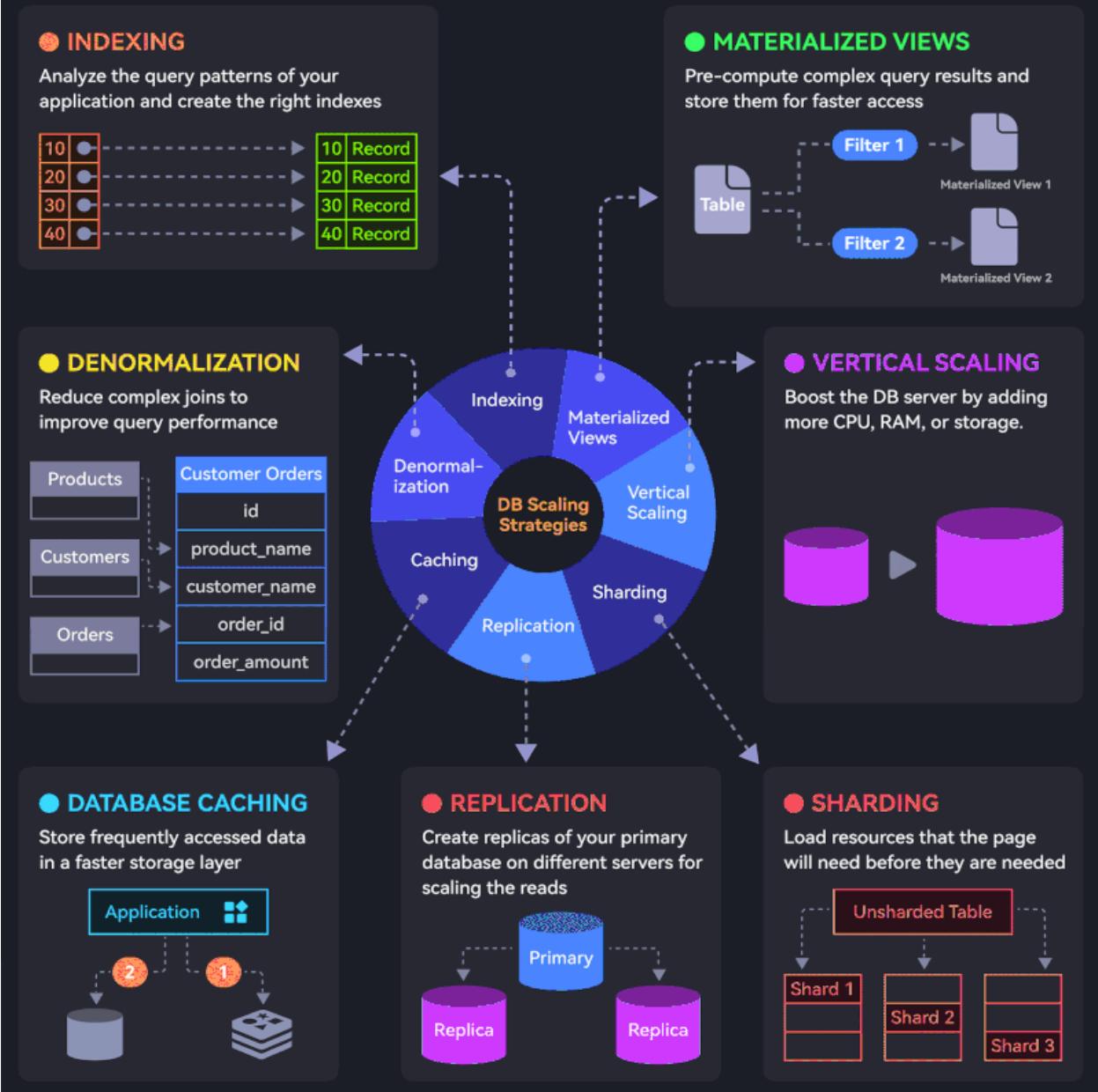
Yapılanlar

- AdventureWorks2022 dosyası indirildi.
- SSMS üzerinden Restore Database işlemiyle sistemde kuruldu.

➤ **VERİTABANI PERFORMANS OPTİMİZASYONU İÇİN ÖLÇEKLENDİRME NEDEN VE NE ZAMAN GEREKLİDİR?**

Database Scaling Cheatsheet

ByteByteGo



➤ PROJE ADIMLARI:

a) Performans İzleme

Araç: SQL Server Profiler

Amaç: Ağır veya yavaş sorguları tespit etmek

Adımlar:

- SQL Server Profiler'ı başlatmak

- Yeni bir "Trace" oluşturmak
- Şablonunu seçmek (TSQL_Duration)
- Hedef veritabanını seçmek (AdventureWorks)
- 10-15 dakika sistemdeki işlemleri izlemek
- En uzun süren sorguları tespit etmek ve kaydetmek

b) Dynamic Management Views (DMV) ile Performans Analizi

```

SQLQuery2.sql - C...\(COMP\FLEX V (55))*
SELECT TOP 10
    total_worker_time / execution_count AS Avg_CPU_Time,
    execution_count,
    total_worker_time,
    total_elapsed_time,
    query_hash,
    SUBSTRING(st.text,
        (qs(statement_start_offset/2)+1,
        ((CASE qs(statement_end_offset
            WHEN -1 THEN DATALENGTH(st.text)
            ELSE qs(statement_end_offset END - qs(statement_start_offset)/2) + 1
        ) AS query_text
    FROM sys.dm_exec_query_stats AS qs
    CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS st
    ORDER BY Avg_CPU_Time DESC;

```

Results

Avg_CPU_Time	execution_count	total_worker_time	total_elapsed_time	query_hash	query_text
589181	1	589181	589248	0x8FB5273EAD907CE	SELECT udf.name AS [Name], udf.object_id AS [ID], u...
2763	1	2763	2764	0xBDE9632F667F9792	SELECT clmns.column_id AS [ID], clmns.name AS [Na...
2397	1	2397	2398	0x9D2B733D94FCDB4F	SELECT clmns.name AS [Name], clmns.column_id AS ...
1078	1	1078	1078	0x5F7A717E52939E4F	SELECT clmns.column_id AS [ID], clmns.name AS [Na...
1067	7	7469	7473	0x2B02C67D1FC4AB4	SELECT clmns.name AS [Name], clmns.column_id AS ...
872	1	872	873	0xA7C5D937EEE6D57B	SELECT param.is_readonly AS [IsReadOnly], param.n...
853	1	853	853	0xB59B3DBBEF4650B0	SELECT NULL AS [Text], ISNULL(smudf.definition, ss...
646	1	646	646	0xE2F07EBE13A6079	SELECT param.parameter_id AS [ID], param.name AS...

Query executed successfully.

Açıklama: En çok CPU harcayan sorguları listeler. Sorgu metinleri ve çalışma istatistiklerini gösterir.

c) İndeks Analizi ve Yönetimi

Amaç: Gereksiz veya eksik indeksleri bulmak

Adımlar:

- İndeks kullanımını incelemek

Optimize Edilmemiş Sorgu:

SQLQuery2.sql - C...(COMP\FLEX V (55))*

```
SELECT *
FROM Sales.SalesOrderHeader
WHERE YEAR(OrderDate) = 2014
```

Results

	SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	Accou
1	63363	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63363	NULL	10-40:
2	63364	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63364	NULL	10-40:
3	63365	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63365	NULL	10-40:
4	63366	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63366	NULL	10-40:
5	63367	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63367	NULL	10-40:
6	63368	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63368	NULL	10-40:
7	63369	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63369	NULL	10-40:
8	63370	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63370	NULL	10-40:
9	63371	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63371	NULL	10-40:
10	63372	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63372	NULL	10-40:

Query executed successfully.

Açıklama: **YEAR()** fonksiyonu kullanmak, indekslerin kullanılmasını engeller. Bu sorgu optimize edilmelidir.

Optimize Edilmiş Sorgu:

SQLQuery2.sql - C...(COMP\FLEX V (55))*

```
SELECT *
FROM Sales.SalesOrderHeader
WHERE OrderDate >= '2014-01-01'
    AND OrderDate < '2015-01-01';
```

Results

	SalesOrderID	RevisionNumber	OrderDate	DueDate	ShipDate	Status	OnlineOrderFlag	SalesOrderNumber	PurchaseOrderNumber	Accou
1	63363	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63363	NULL	10-40:
2	63364	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63364	NULL	10-40:
3	63365	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63365	NULL	10-40:
4	63366	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63366	NULL	10-40:
5	63367	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63367	NULL	10-40:
6	63368	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63368	NULL	10-40:
7	63369	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63369	NULL	10-40:
8	63370	8	2014-01-01 00:00:00.000	2014-01-13 00:00:00.000	2014-01-08 00:00:00.000	5	1	SO63370	NULL	10-40:

Query executed successfully.

Açıklama: **YEAR()** fonksiyonu yerine tarih aralığı kullanmak, indeks kullanımını sağlar ve sorgu hızını artırır.

```

SQLQuery2.sql - C...\COMP\FLEX V (55)*  x
SELECT
    OBJECT_NAME(i.object_id) AS TableName,
    i.name AS IndexName,
    i.index_id,
    dm_ius.user_seeks,
    dm_ius.user_scans,
    dm_ius.user_lookups,
    dm_ius.user_updates
FROM sys.indexes AS i
INNER JOIN sys.dm_db_index_usage_stats AS dm_ius
    ON i.object_id = dm_ius.object_id
    AND i.index_id = dm_ius.index_id
WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1;

```

100 % 1 Results Messages

TableName	IndexName	index_id	user_seeks	user_scans	user_lookups	user_updates
SalesOrderHeader	PK_SalesOrderHeader_SalesOrderID	1	0	4	0	0

Query executed successfully.

Açıklama: İndekslerin ne kadar kullanıldığını gösterir. Kullanılmayan indeksler tespit edilebilir.

- Gereksiz veya hiç kullanılmayan indeksleri kaldırma

`DROP INDEX IndexName ON TableName;`

d) Soru Optimizasyonu

Amaç: Yavaş çalışan sorguları daha verimli hale getirmek.

Yapılacaklar:

- SQL Profiler veya DMV ile yavaş sorguları belirlemek
- EXPLAIN ya da Execution Plan'ı analiz etmek
- JOIN, WHERE, GROUP BY ifadelerini sadeleştirmek
- INDEX veya TEMP TABLE kullanmak

`CREATE INDEX IX_Customer_LastName
ON Customer (LastName);`

Açıklama: Sorgu performansını artırmak için yeni bir indeks oluşturur.

e) Disk Alanı ve Veri Yoğunluğu Yönetimi

- Kontrol:

```
SQLQuery2.sql - C...(COMP\FLEX V (55))*
```

```
EXEC sp_spaceused;
```

	database_name	database_size	unallocated space
1	AdventureWorks2022	272.00 MB	1.79 MB

	reserved	data	index_size	unused
1	202968 KB	97216 KB	85816 KB	19936 KB

100 %

Results Messages

Query executed successfully.

COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (55) | AdventureWorks2022 | 00:00:00 | 2 rows

Açıklama: Veritabanının toplam boyutunu ve boş alanını gösterir.

- Veri aralığını veya parçalanmış indeksleri yeniden düzenleme (fragmentation fix):

ALTER INDEX ALL ON TableName REBUILD;

Açıklama: Tüm indeksleri yeniden oluşturur. Diskteki veri düzenini iyileştirir, performansı artırır.

f) Erişim ve Rol Yönetimi

Amaç: Veritabanı yetkilisi rollerini ayırmak

CREATE ROLE read_only_role;

GRANT SELECT ON dbo.TableName TO read_only_role;

Açıklama: Okuma yetkisi olan yeni rol oluşturur.

The screenshot shows a SQL query window titled "SQLQuery2.sql - C...(COMP\FLEX V (55))". The query is:

```
CREATE USER readonlyuser FOR LOGIN readonlylogin;
EXEC sp_addrolemember 'read_only_role', 'readonlyuser';
```

The "Messages" pane below shows the results:

```
Commands completed successfully.  
Completion time: 2025-04-24T10:00:55.1907735+03:00
```

A status bar at the bottom indicates: COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (55) | AdventureWorks2022 | 00:00:00 | 0 rows.

Açıklama: Sadece okuma yetkisine sahip kullanıcı tanımlanır.

➤ KAYNAKLAR

1. https://www.youtube.com/watch?v=_1IKwnbscQU&list=TLPQMjQwNDIwMjUq_dK_15bbIg&index=8
2. Kullanılan Veritabanı: <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms>
3. Anlatım Videosu: <https://youtu.be/Iw1PrMjMHzE>
4. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery2.sql>

PROJE 2. VERİTABANI YÜKSELTME VE SÜRÜM YÖNETİMİ

- **PROJE AMACI:** Bir veritabanının eski sürümünden daha yeni bir sürüme yükseltilmesini gerçekleştirmek, sürüm kontrol mekanizmaları ile değişikliklerin izlenmesini sağlamak ve yükselme sürecine ait test ve geri dönüş planlarını tanımlamaktır.

➤ KULLANILAN VERİTABANI VE ORTAM

Veritabanı: AdventureWorks2022

Platform: Microsoft SQL Server 2022

Araç: SQL Server Management Studio (SSMS)

➤ VERİTABANI YÜKSELTME PLANI

The screenshot shows a SQL Server Management Studio window titled "SQLQuery4.sql - C...(COMP\FLEX V (51))". The query window contains the following SQL code:

```
SELECT name, compatibility_level
FROM sys.databases
WHERE name = 'AdventureWorks2022';
```

The results pane shows a single row of data:

name	compatibility_level
AdventureWorks2022	160

The status bar at the bottom indicates "Query executed successfully." and "1 rows".

Açıklama: Mevcut veritabanının incelenmesi.

The screenshot shows a SQL Server Management Studio window titled "SQLQuery4.sql - C...(COMP\FLEX V (51))". The query window contains the following SQL code:

```
BACKUP DATABASE AdventureWorks2022
TO DISK = 'C:\Belgeler\Yurt, Üniversite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak'
WITH FORMAT, MEDIANAME = 'AdventureWorksBackup', NAME = 'Full Backup';
```

The results pane shows the execution message and completion time:

```
Processed 25384 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
BACKUP DATABASE successfully processed 25386 pages in 0.325 seconds (610.228 MB/sec).

Completion time: 2025-04-24T23:03:23.5409641+03:00
```

The status bar at the bottom indicates "Query executed successfully." and "0 rows".

Açıklama: Yedek alma.

```
SQLQuery4.sql - C...(COMP\FLEX V (51))*
USE master;
GO
ALTER DATABASE AdventureWorks2022 SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
RESTORE DATABASE AdventureWorks2022
FROM DISK = 'C:\Belgeler(Yurt, Üniversite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak'
WITH
MOVE 'AdventureWorks2022' TO 'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\AdventureWorks2022.mdf',
MOVE 'AdventureWorks2022_log' TO 'C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\AdventureWorks2022_log.:1
REPLACE';
GO
ALTER DATABASE AdventureWorks2022 SET MULTI_USER;
GO

100 % < >
Messages
Commands completed successfully.

Completion time: 2025-04-24T23:48:00.6793905+03:00

100 % < >
Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (51) | master | 00:00:00 | 0 rows
```

Açıklama: Yeni sürümde restore etme.

```
SQLQuery4.sql - C...(COMP\FLEX V (51))*
ALTER DATABASE AdventureWorks2022
SET COMPATIBILITY_LEVEL = 160;

100 % < >
Messages
Commands completed successfully.

Completion time: 2025-04-24T23:51:05.8856312+03:00

100 % < >
Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (51) | AdventureWorks2022 | 00:00:00 | 0 rows
```

Açıklama: Uyumlu sürüm ayarlama.

➤ SÜRÜM YÖNETİMİ

```
SQLQuery4.sql - C...(COMP\FLEX V (51))* X
CREATE TABLE AuditSchemaChanges (
    EventTime DATETIME,
    LoginName NVARCHAR(100),
    EventType NVARCHAR(100),
    ObjectName NVARCHAR(100),
    CommandText NVARCHAR(MAX)
);
GO
CREATE TRIGGER trg_SchemaAudit
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
BEGIN
    INSERT INTO AuditSchemaChanges
    SELECT
        GETDATE(),
        SYSTEM_USER,
        EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]', 'NVARCHAR(MAX)');
END;

100 % < Messages
Commands completed successfully.

Completion time: 2025-04-24T23:55:12.1293424+03:00

100 % < Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (51) | AdventureWorks2022 | 00:00:00 | 0 rows
```

Açıklama: DDL Trigger ile şema değişikliği takibi.

```
SQLQuery4.sql - C...(COMP\FLEX V (51))* X
ALTER TABLE Person.ContactType ADD Description NVARCHAR(100);

100 % < Messages
(1 row affected)

Completion time: 2025-04-25T00:00:31.0638380+03:00

100 % < Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (51) | AdventureWorks2022 | 00:00:00 | 0 rows
```

Açıklama: Test amaçlı şema değişikliği.

➤ TEST VE GERİ DÖNÜŞ PLANI

- Geri dönüş için alınan tam yedek kullanılabilir
- Yükseltme sonrası test sorguları:

SQLQuery4.sql - C...(COMP\FLEX V (51))* X

```
SELECT TOP 10 * FROM Person.ContactType;
```

100 %

Results Messages

ContactTypeID	Name	ModifiedDate	Description
1	Accounting Manager	2008-04-30 00:00:00.000	NULL
2	Assistant Sales Agent	2008-04-30 00:00:00.000	NULL
3	Assistant Sales Representative	2008-04-30 00:00:00.000	NULL
4	Coordinator Foreign Markets	2008-04-30 00:00:00.000	NULL
5	Export Administrator	2008-04-30 00:00:00.000	NULL
6	International Marketing Manager	2008-04-30 00:00:00.000	NULL
7	Marketing Assistant	2008-04-30 00:00:00.000	NULL
8	Marketing Manager	2008-04-30 00:00:00.000	NULL
9	Marketing Representative	2008-04-30 00:00:00.000	NULL
10	Order Administrator	2008-04-30 00:00:00.000	NULL

Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) COMP\FLEX V (51) AdventureWorks2022 00:00:00 10 rows

- İşlevsel test, CRUD işlemlerinin denenmesi
- Sorun durumunda:

SQLQuery4.sql - C...(COMP\FLEX V (51))* X

```
RESTORE DATABASE AdventureWorks2022
FROM DISK = 'C:\Belgeler(Yurt, Üniversite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak' WITH REPLACE;
```

100 %

Messages

Processed 25384 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.
 Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
 RESTORE DATABASE successfully processed 25386 pages in 0.530 seconds (374.196 MB/sec).

Completion time: 2025-04-25T00:09:16.2456016+03:00

Query executed successfully. COMP\SQLEXPRESS (16.0 RTM) COMP\FLEX V (51) master 00:00:01 0 rows

➤ SONUÇ

Veritabanı başarıyla yeni sürümeye geçirildi ve şema değişiklikleri izlenebilir hale getirildi. Ayrıca da geri dönüş planı test edildi ve çalıştığı doğrulandı.

➤ KAYNAKLAR

1. Kullanılan Veritabanı: <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms>
2. Anlatım Videosu: <https://youtu.be/BBNB-ZdF2vU>
3. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery4.sql>

PROJE 3. VERİTABANI GÜVENLİĞİ VE ERIŞİM KONTROLÜ

- **PROJE AMACI:** Microsoft SQL Server kullanılarak veritabanı güvenliği sağlanması hedeflenmektedir. Kullanıcı erişim yönetimi, veri şifreleme, SQL Injection testleri ve audit log (kayıt) sistemlerinin uygulanması gerçekleştirilecektir.

➤ KULLANILAN VERİTABANI VE ORTAM

Veritabanı: AdventureWorks2022

Platform: Microsoft SQL Server 2022

Araç: SQL Server Management Studio (SSMS)

Windows Authentication / SQL Server Authentication

➤ PROJE ADIMLARI:

a) Erişim Yönetimi

Kullanıcı Oluşturma (SQL Authentication):

```
SQLQuery6.sql - C...(COMP\FLEX V (59))* ✎ X
CREATE LOGIN ogrenciLogin WITH PASSWORD = 'GucluParola123!';
CREATE USER ogrenciUser FOR LOGIN ogrenciLogin;
EXEC sp_addrolemember 'db_datareader', 'ogrenciUser';

100 % ◀
Messages
Commands completed successfully.

Completion time: 2025-04-25T12:01:54.6899991+03:00

100 % ◀
Query executed successfully. 🔒 COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (59) | AdventureWorks2022 | 00:00:00 | 0 rows
```

Yetkiyi Geri Alma:

```
SQLQuery6.sql - C...(COMP\FLEX V (59))* ✎ X
EXEC sp_droprolemember 'db_datareader', 'ogrenciUser';

100 % ◀
Messages
Commands completed successfully.

Completion time: 2025-04-25T12:05:08.7957955+03:00

100 % ◀
Query executed successfully. 🔒 COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (59) | AdventureWorks2022 | 00:00:00 | 0 rows
```

b) Veri Şifreleme (TDE - Transparent Data Encryption)

Ana Anahtar Oluşturma:

The screenshot shows a SQL query window titled "SQLQuery6.sql - C... (COMP\FLEX V (59))". The query is:

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MasterKeySifresi123!';
```

The results pane shows:

100 % ↶ Messages
(1 row affected)
Completion time: 2025-04-25T12:10:32.2923983+03:00

100 % ↶ 0 Query executed successfully.

COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (59) | master | 00:00:00 | 0 rows

Sertifika Oluşturma:

```
CREATE CERTIFICATE TDESertifika
WITH SUBJECT = 'Veritabanı Şifreleme Sertifikası';
```

Veritabanı Şifreleme Anahtarı (DEK) oluşturma:

```
USE AdventureWorks2022;
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TDESertifika;
```

Şifrelemeyi Etkinleştirme:

```
ALTER DATABASE AdventureWorks2022 SET ENCRYPTION ON;
```

c) SQL Injection Testi ve Önleme

Kötü Yazılmış Dinamik SQL (Savunmasız):

```
SQLQuery6.sql - C...(COMP\FLEX V (59))* X
DECLARE @sorgu NVARCHAR(MAX)
SET @sorgu = 'SELECT * FROM Users WHERE Username = ''' + @username + '''
EXEC sp_executesql @sorgu
```

Doğru Kullanım (Parametreli Sorgu ile):

```
SQLQuery6.sql - C...(COMP\FLEX V (59))* X
EXEC sp_executesql N'SELECT * FROM Users WHERE Username = @kullaniciAdi',
N'@kullaniciAdi NVARCHAR(50)',
@kullaniciAdi = 'admin';

100 % < >
Results Messages
UserID Username Password Role
1 admin admin123 Admin
```

Query executed successfully.

COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (59) | AdventureWorks2022 | 00:00:00 | 1 rows

d) Audit Logları (SQL Server Audit)

Server Audit Oluşturma:

```
USE master;
GO
CREATE SERVER AUDIT GUVENLIK_AUDIT
TO FILE (FILEPATH = 'C:\AuditLogs\' );
ALTER SERVER AUDIT GUVENLIK_AUDIT WITH (STATE = ON);
```

Database-Level Audit Oluşturma:

The screenshot shows the SSMS interface with a query window titled "SQLQuery6.sql - C... (COMP\FLEX V (59))". The query itself creates a database audit specification named "GUVENLIK_DB_AUDIT" for the "AdventureWorks2022" database, specifying audits for SELECT and INSERT operations on the "Users" table by the "ogrenciUser" login.

```
USE AdventureWorks2022;
GO
CREATE DATABASE AUDIT SPECIFICATION GUVENLIK_DB_AUDIT
FOR SERVER AUDIT GUVENLIK_AUDIT
ADD (SELECT ON OBJECT::[Users] BY ogrenciUser),
ADD (INSERT ON OBJECT::[Users] BY ogrenciUser)
WITH (STATE = ON);
```

Below the query window, the "Messages" pane displays the successful completion of the command:

Commands completed successfully.
Completion time: 2025-04-25T12:34:23.6435184+03:00

At the bottom of the interface, a status bar indicates the session details:

100 % COMP\SQLEXPRESS (16.0 RTM) | COMP\FLEX V (59) | AdventureWorks2022 | 00:00:00 | 0 rows

➤ SONUÇ

Bu projede SQL Server üzerinde temel güvenlik önlemleri başarıyla uygulanmıştır. Kullanıcı erişimi sınırlandırılmış ve hassas veriler şifrelenmiş, injection'a karşı önlem alınmış ve kullanıcı aktiviteleri kayıt altına alınmıştır.

➤ KAYNAKLAR

1. Kullanılan Veritabanı: <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms>
2. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery6.sql>

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



FİNAL PROJE RAPORU

(BLM4522) AĞ TABANLI PARALEL DAĞITIM SİSTEMLERİ

Perizat SAGYNBEKOVA

21290895

GitHub (<https://github.com/lmpasbaa/MSSQL.git>)

30.05.2025

PROJE 1. VERİTABANI YEDEKLEME VE FELAKETTEN KURTARMA PLANI

PROJE AMACI: Veritabanı yedekleme ve felaket kurtarma planı, veri kaybını önlemek ve sistem kesintilerinde hızlı kurtarma sağlamak için kritik öneme sahiptir. Bu rapor, bir veritabanı sisteminin güvenliğini sağlamak ve veri kaybını en aza indirmek amacıyla oluşturulan yedekleme ve felaketten kurtarma planlarını içermektedir. SQL Server kullanılarak tam, artık ve fark yedekleme stratejileri, otomatik yedekleme zamanlayıcıları, felaket senaryoları ve yedek test süreçleri detaylandırılmıştır.

HEDEF: Veri kaybını önlemek; Kesinti süresini en aza indirmek; Yedeklerin güvenilirliğini test etmek; Uyumluluk gereksinimlerini karşılamak.

- ✓ **Veritabanı Yedekleme**, dosyaların periyodik olarak anlık görüntülerini alarak bir kopyasını oluşturma işlemidir.
- ✓ **Felaketten Kurtarma Planı**, bir organizasyona kesintiden sonra (felaket sonrasında) tüm hayatı altyapıları, veritabanlarını, uygulamaları ve hizmetleri kurtarma yeteneği kazandıran süreçler, politikalar, prosedürler ve temel ölçümlerin birleşimidir.

1. GİRİŞ

1.1. YEDEKLEME STRATEJİLERİ

1.1.1. Tam Yedekleme (Full Backup)

Genellikle günlük veya haftalık periyotlarla büyük veri değişikliklerinden önce veya sonra tüm veritabanının bir kopyasını alır. En hızlı geri yükleme seçeneğidir ve tek başına kullanılabilir fakat fazla depolama alanı gerektirir.

1.1.2. Artık Yedekleme (Transaction Log Backup)

Tam yedekten sonra yapılan tüm işlemleri kaydeder. Felaket anında “point-in-time restore” ile belirli bir saniyeye kadar geri dönüş sağlanır ve sıkılıkla (örneğin her 15 dakikada bir) yapılması önerilir.

1.1.3. Fark Yedekleme (Differential Backup)

Tam yedekleme ile birlikte kullanılmakta ve son tam yedekten sonra değişen verileri yedekler. Avantajı ise disk alanı kullanımını daha azdır.

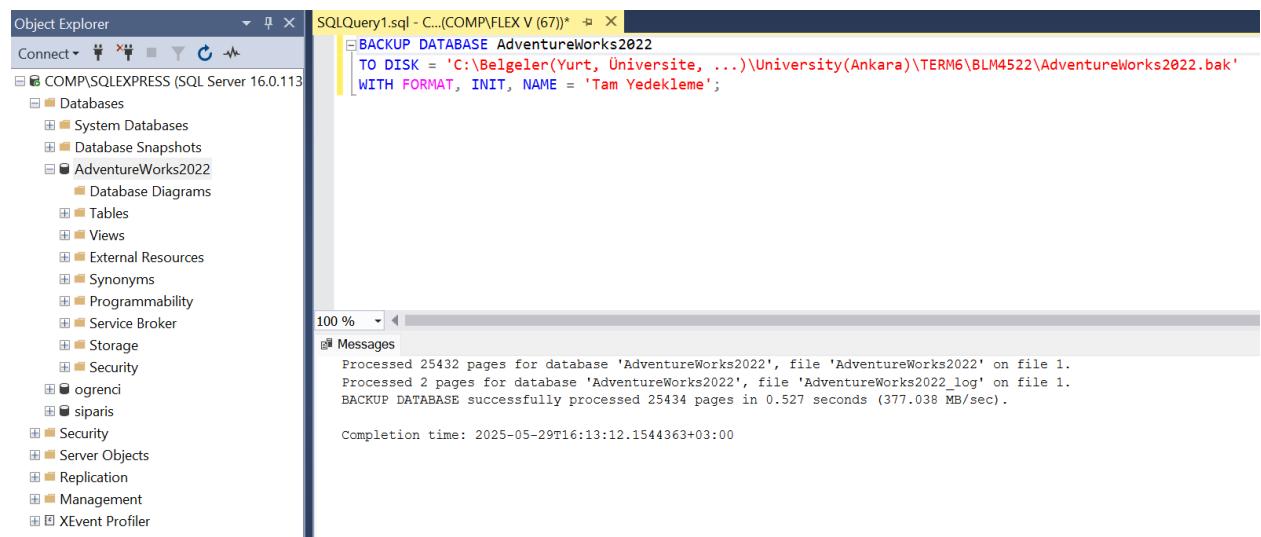
1.2. ZAMANLAYICILARLA OTOMATİK YEDEKLEME

1.2.1. SQL Server Agent Kullanımı

SQL Server Agent ile yedekleme işleri zamanlanabilir. Örnek:

- Tam yedekleme: Her gece saat 02:00
- Artık (ya da artırımlı) yedekleme: Her 15 dakikada bir
- Fark yedekleme: Günde 2 kez (12:00 ve 18:00)

1.2.2. Script Örneği (Tam Yedekleme)



The screenshot shows the SSMS interface with the Object Explorer on the left and a query window on the right. The query window contains the following T-SQL code:

```
BACKUP DATABASE AdventureWorks2022  
TO DISK = 'C:\Belgeler\Yurt, Universite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak'  
WITH FORMAT, INIT, NAME = 'Tam Yedekleme';
```

Below the code, the Messages pane displays the execution results:

```
Processed 25432 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.  
BACKUP DATABASE successfully processed 25434 pages in 0.527 seconds (377.038 MB/sec).  
Completion time: 2025-05-29T16:13:12.1544363+03:00
```

Adım adım otomatik tam yedekleme Job'u:

```

| USE msdb;
| GO
| EXEC sp_add_job
|   @job_name = N'OrnekDB_TamYedekJob';
|   GO
| EXEC sp_add_jobstep
|   @job_name = N'OrnekDB_TamYedekJob',
|   @step_name = N'Tam Yedek Al',
|   @subsystem = N'TSQL',
|   @command = N'
|     BACKUP DATABASE [OrnekDB]
|     TO DISK = ''D:\Yedekler\OrnekDB_TamYedek.bak''
|     WITH FORMAT, INIT, NAME = ''OrnekDB - Tam Yedek''
|
|   '
|   @retry_attempts = 5,
|   @retry_interval = 5;
| GO
| EXEC sp_add_schedule
|   @schedule_name = N'Her Gece 02:00',
|   @freq_type = 4, -- günlük
|   @freq_interval = 1,
|   @active_start_time = 020000;
| GO
| EXEC sp_attach_schedule
|   @job_name = N'OrnekDB_TamYedekJob',
|   @schedule_name = N'Her Gece 02:00';
| GO
| EXEC sp_add_jobserver
|   @job_name = N'OrnekDB_TamYedekJob';
| GO

```

1.3. FELAKETTEN KURTARMA SENARYOLARI

1.3.1. Kaza ile Silinen Veriler (Veritabanını Belirli Bir Zamana Geri Yükleme – Point-in-Time Restore)

Transaction log yedekleri kullanılarak belirli bir zamana geri dönüş yapılabilir.

Senaryo: Bir kullanıcı yanlışlıkla önemli bir tabloyu sildi.

Çözüm: En son tam yedekten geri yükle. İşlem günlüğünü (log backup) silme işleminden önceki bir zamana kadar uygula.

Point-in-time restore örneği:

```

RESTORE DATABASE AdventureWorks2022
FROM DISK = 'C:\Beigeler\Yurt, Üniversite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak'
WITH NORECOVERY;
RESTORE LOG AdventureWorks2022
FROM DISK = 'veritabani_log.trn'
WITH STOPAT = '2025-05-29 10:15:00', RECOVERY;

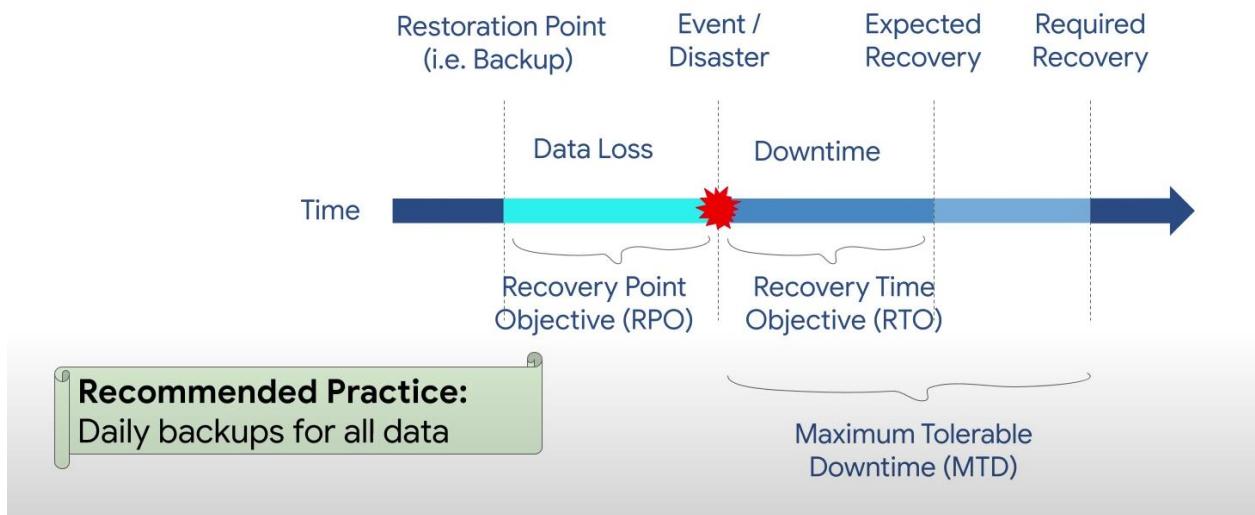
```

100 %

Messages

Processed 25432 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.
 Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.
 RESTORE DATABASE successfully processed 25434 pages in 0.326 seconds (609.506 MB/sec).

Completion time: 2025-05-29T17:28:20.8949295+03:00



1.3.2. Donanım Arızası

Database mirroring veya log shipping gibi çözümlerle yedek sunucuya geçiş sağlanabilir.

1.4. VERİTABANI YANSITMA (DATABASE MIRRORING)

Senaryo: Sunucu çöktü, hızlı bir şekilde yedek sunucuya geçiş yapılmalı.

Çözüm: Principal (Ana sunucu) ve Mirror (Yedek sunucu) arasında veritabanı aynalama kurulumu yapılır.

```
-- Principal sunucuda
ALTER DATABASE [AdventureWorks2022]
SET PARTNER = 'TCP://MirrorServer:5022';

-- Mirror sunucuda
ALTER DATABASE [AdventureWorks2022]
SET PARTNER = 'TCP://PrincipalServer:5022';
```

1.5. TEST YEDEKLEME SENARYOLARI

1.5.1. Yedeklerin Doğruluğunu Test Etme

Periyodik olarak yedekten geri yükleme testleri yapılmalıdır ve test sunucularında yedeklerin başarıyla yüklenip yüklenmediği kontrol edilmelidir.

Senaryo	Test Sıklığı	Yöntem
Yedekten Geri Yükleme	Aylık	<code>RESTORE DATABASE</code> komutu ile test ortamında geri yükle
Yedek Büyünlük Kontrolü	Haftalık	<code>RESTORE VERIFYONLY</code> komutu
Felaket Simülasyonu	Yıllık	Tam sunucu çökmesi senaryosu ile DR testi

1.5.2. Script Örneği (Test Yükleme)

Yedek bütünlük kontrolü:

```

Object Explorer
Connect ▾ X X
COMPRESSLEXPRESS (SQL Server 16.0.113)
Databases
System Databases
Database Snapshots
AdventureWorks2022
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
ogrenci
siparis
Security
Server Objects
Replication
Management
XEvent Profiler

SQLQuery1.sql - C...(COMPRESSLEXPRESS V (67))*
RESTORE VERIFYONLY
FROM DISK = 'C:\Belgeler\Yurt, Üniversite, ...)\University(Ankara)\TERM6\BLM4522\AdventureWorks2022.bak';

100 %
Messages
The backup set on file 1 is valid.
Completion time: 2025-05-29T18:31:37.3966557+03:00

```

SONUÇ ve ÖNERİLER

- ✓ Her zaman 3-2-1 kuralı uygulanmalı: 3 kopya (birincil + 2 yedek); 2 farklı ortam (örneğin, disk + bulut); 1 off-site yedek (fiziksel felaketlere karşı).
- ✓ Büyük veritabanlarında yedekleme çok zaman alıyorsa, **Compressed Backup** kullanılmalı: **WITH COMPRESSION**.
- ✓ Yedekler tek yerdeyse, farklı fiziksel disklere veya bulut ortamına da yedek alınabilir.
- ✓ Yetkisiz erişim riski varsa, yedek dosyalarını şifrelenmeli: **WITH ENCRYPTION** (SQL Server Enterprise)
- ✓ Otomatik bildirim için, SQL Agent Job tamamlandığında e-posta ile bilgilendirme ayarlanması.

Yedeklerin gerçekten çalıştığından emin olmak önemli ve bunun için de yedekleme testleri mutlaka periyodik olarak yapılmalıdır.

KAYNAKLAR

1. <https://www.youtube.com/watch?v=Z-6GkXCTPic&list=TLPOQMjkwNTIwMjWm44UozrR5Og&index=4>
2. Kullanılan Veritabanı: <https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms>
3. Anlatım Videosu: <https://youtu.be/YV0IEwAlN88?si=U87O8LFiE6ZBXOjd>
4. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery1.sql>

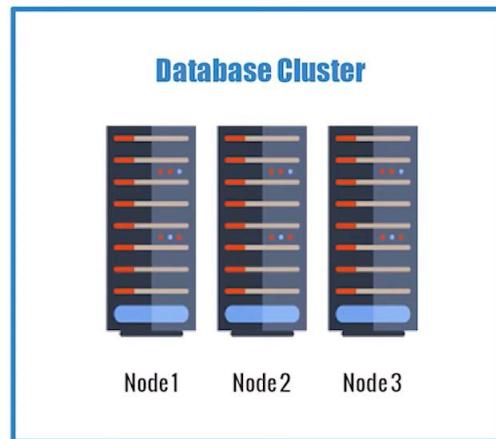
PROJE 2. VERİTABANI YÜK DENGELİME VE DAĞITIK VERİTABANI YAPILARI

PROJE AMACI: Bu projenin amacı, kurumsal düzeyde bir veritabanı sisteminin yüksek erişilebilirlik, yüksek performans ve kesintisiz hizmet sağlayacak şekilde nasıl yapılandırılabilceğini göstermektir. Bu kapsamında:

- Dağıtık veritabanı yapıları ile verilerin farklı sunucularda/coğrafi lokasyonlarda yönetilmesi,
- Yük dengeleme stratejileri sayesinde sistem kaynaklarının verimli kullanımı,
- Veritabanı replikasyonu ile veri tutarlığının sağlanması,
- Always On Availability Groups ve Database Mirroring gibi teknolojilerle kesintisiz erişim ve otomatik failover süreçlerinin uygulanması,
- Gerçek dünya senaryoları ile bu yapıların nasıl entegre edileceğinin anlaşılması amaçlanmaktadır.

- ✓ **Yük Dengeleme**, yüksek performans, ölçülebilirlik ve kesintisiz hizmet vermek amacıyla gelen veri/iş yükünün birden fazla veritabanı veya sunucuya dengeli şekilde dağıtılmıştır (bir e-ticaret sitesi örneği düşünelim: Sipariş alma işlemleri PRIMARY sunucudan yapılırken, ürün listeleme, yorum gösterme gibi okuma işlemleri SECONDARY sunucuya yönlendirilir. Bu yönlendirme, uygulama kodunda veya SQL Server'in "Read-Only Routing" özelliğiyle yapılır).
- ✓ **Dağıtık Veritabanında**, veriler veritabanını oluşturan birden fazla bilgisayara dağıtılr. Ve tüm veritabanı tek bir bilgisayarda çalışır (büyük firmalar (Amazon, Trendyol, Hepsiburada gibi), Türkiye ve Avrupa'da farklı veri merkezlerinde aynı veritabanının kopyalarını tutar. Kullanıcı, coğrafi olarak en yakın sunucuya bağlanır. Bu sayede sorgular daha hızlı döner (latency azalır)). Dağıtılmış veritabanları daha fazla veri depolamamıza; işi bilgisayarlar arasında bölerek büyük miktardaki veriler üzerinde çok daha hızlı sorgular gerçekleştirmemize; donanım veya ağ arızaları durumunda hata toleransını artırarak sistemlerimizi daha dayanıklı hale getirmemize olanak tanır.

Distributed Databases



Distributed database “clusters” are made up of individual “nodes”

1. GİRİŞ

1.1. BİRDEN ÇOK VERİTABANI YÖNETİMİ VE REPLİKASYON TEKNİKLERİ

- ✓ **Veritabanı Replikasyonu (SQL Server Replication)**, bir veritabanındaki verilerin birden fazla sunucuya (subscriber) otomatik olarak çoğaltıması ve senkronize edilmesidir.

Replikasyon türleri:

- Snapshot Replication (belirli aralıklarla verinin tam kopyasını alır)
- Transactional Replication (anlık veri değişimlerini hemen gönderir (gerçek zamanlıya yakın))
- Merge Replication (hem publisher hem subscriber veri değiştirirse birleştirme yapar yani çift yönlü veri senkronizasyonu sağlar)

1.1.1. Kullanım Senaryosu (Transactional Replication)

Örnek: İstanbul'daki ana sunucudaki verilerin anlık olarak Ankara'daki bir şubeye aktarılması.

```
SQLQuery2.sql - C...(COMP\FLEX V (62))*  X
-- Distributor
EXEC sp_adddistributor @distributor = N'MAIN-SERVER';

-- Publisher
EXEC sp_addpublication @publication = N'OrnekYayin', @status = N'active';

-- Subscriber
EXEC sp_addsubscription
@publication = N'OrnekYayin',
@subscriber = N'SUBE-SERVER',
@destination_db = N'OrnekDB_Kopya',
@subscription_type = N'Push';
```

1.2. YÜK DENGELEME YÖNTEMLERİ: ALWAYS ON AVAILABILITY GROUPS VE DATABASE MIRRORİNG

1.2.1. SQL Server Always On Availability Groups

Çoklu veritabanlarının yüksek erişilebilirlik ve yük dengeleme için yapılandırılmasıdır. Birincil veritabanından ikincil veritabanlarına veri çoğaltma yapar.

Senaryo: Ana sunucu (ServerA) ile yedek sunucu (ServerB) arasında veri senkronizasyonu ve okuma yük dengelemesi sağlanacak.

Adımlar:

- WSFC kümesi oluşturulur.
- SQL Server'da Always On özelliğini aktif edilir.
- Availability Group oluşturulur (SSMS veya T-SQL).
- Replika sunucular belirlenir (Primary & Secondary).
- Listener (bağlantı noktası) ile uygulamalar bağlanır.

Örnek kod (Availability Group oluşturma – avantajı, uygulamalar okuma işlemlerini otomatik olarak yedek sunuculardan yapabilir. Yük dengelenmiş olur):

```
CREATE AVAILABILITY GROUP [OrnekAG]
WITH (
    AUTOMATED_BACKUP_PREFERENCE = SECONDARY
)
FOR DATABASE [OrnekDB]
REPLICA ON
```

```

N'SERVERA' WITH (
    ENDPOINT_URL = 'TCP://SERVERA:5022',
    AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
    FAILOVER_MODE = AUTOMATIC,
    SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY)
),
N'SERVERB' WITH (
    ENDPOINT_URL = 'TCP://SERVERB:5022',
    AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
    FAILOVER_MODE = AUTOMATIC,
    SECONDARY_ROLE (ALLOW_CONNECTIONS = READ_ONLY)
);

```

1.2.2. Database Mirroring (daha eski bir teknoloji)

Bir veritabanının birincil ve ikincil sunucuya yansıtılması (1 veritabanı için geçerlidir).

```

CREATE ENDPOINT [Mirroring]
STATE = STARTED
AS TCP (LISTENER_PORT = 5022)
FOR DATABASE_MIRRORING (
    ROLE = PARTNER
);
-- Veritabanı restore edilmesi (NORECOVERY ile)
-- Mirroring
ALTER DATABASE Adventureworks2022
SET PARTNER = 'TCP://SERVERB:5022';

```

Modlar:

- High safety (senkron + otomatik geçiş)
- High performance (asenkron)
- High safety without automatic failover

Gerçek Hayat Senaryosu Örneği:

 **Firma:** TrendyX

 **Lokasyon:** İstanbul (Ana sunucu), Frankfurt (Yedek sunucu)

 **Uygulama:**

- İstanbul'daki PRIMARY sunucu siparişleri işler (INSERT, UPDATE).
- Almanya'daki SECONDARY sunucu sadece SELECT işlemleri için kullanılır.
- Availability Group ile bağlantılar trendyx-db-listener üzerinden yapılır.
- PRIMARY sunucu çökerse failover otomatik olarak Almanya sunucusuna gerçekleşir.
- Kullanıcılar bu geçiş fark etmez, web sitesi çalışmaya devam eder.

1.3. FAİLOVER SENARYOLARI

1.3.1. Always On Failover Stratejileri

- Otomatik Failover: Önceden tanımlanmış bir ikincil sunucuya otomatik geçiş sağlar. Genellikle 2-3 düğümlü senkron commit kullanır.

- Planlı Manuel Failover: Bakım veya yükseltme durumlarındadır. Veri kaybı olmaz.,
- Zorunlu Manuel Failover (Force Failover): Birincil sunucu çöktüğünde ve otomatik failover başarısız olduğunda kullanıma geçer. Fakat potansiyel veri kaybı riski vardır.

1.3.2. Performans Optimizasyonu

- Okuma İş Yükü Dengeleme: İkincil sunuculara okuma işlemlerini yönlendirme ve Connection string'de **ApplicationIntent=ReadOnly** kullanımı yapılabilir.

- Yönlendirme Stratejileri: Round-robin DNS; Donanım veya yazılım yük dengeleyicileri; Uygulama katmanında yönlendirme gibi işlemler gerçekleştirilebilir.

1.3.3. İzleme ve Bakım

- Sistem DMV'leri: **sys.dm_hadr_availability_group_states**; **sys.dm_hadr_availability_replica_states**; **sys.dm_hadr_database_replica_states**.

- Performans Metrikleri: Veri senkronizasyon gecikmesi; Log send queue size; Redo queue size

Bu teknikler, SQL Server ortamlarında yüksek erişilebilirlik, felaket kurtarma ve performans optimizasyonu sağlamak için kritik öneme sahiptir.

SONUÇ

Bu proje kapsamında, yüksek erişilebilirlik, veri tutarlılığı ve sistem performansını maksimize etmeye yönelik olarak **veritabanı yük dengeleme ve dağıtık veritabanı yapıları** incelenmiş ve uygulamaya yönelik senaryolarla desteklenmiştir.

Projede özellikle aşağıdaki başlıklar desteklenmiştir:

- ✓ **Always On Availability Groups** kullanılarak, birden fazla sunucu arasında hem veri senkronizasyonu hem de otomatik failover sağlanması.
- ✓ **Read-Only Routing** özelliği ile okuma işlemleri (ürün listeleme, yorum gösterme gibi) otomatik olarak **SECONDARY replica**'ya yönlendirilmiş, böylece PRIMARY sunucunun yükünün hafifletilmesi.
- ✓ **Replication teknikleri** ile veri çoğaltma ve coğrafi yedekliliğin sağlanması.
- ✓ **Failover senaryolarının** test edilmesi, sunucu kesintilerinde sistemin nasıl otomatik olarak yedek sunucuya geçtiğinin gözlemlenmesi.

KAYNAKLAR

1. <https://www.youtube.com/watch?v=J-sj3GUrq9k>
2. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery3.sql>

PROJE 3. VERİ TEMİZLEME VE ETL SÜREÇLERİ TASARIMI

PROJE AMACI: Bu proje, kurum içindeki dağınık, tutarsız ve kalitesiz verileri standartlaştırarak güvenilir, analize hazır ve iş kararlarını destekleyecek bir veri altyapısı oluşturmayı hedeflemektedir. Temel odak noktaları şunlardır:

- Veri Kalitesinin Arttırılması: Eksik, hatalı ve tekrarlanan verilerin tespiti ve düzeltilmesi; Farklı kaynaklardan gelen verilerin standart formata dönüştürülmesi
- ETL (Extract, Transform, Load) Süreçlerinin Otomasyonu: Veri entegrasyonunun manuel süreçlerden kurtarılarak zaman ve maliyet tasarrufu sağlanması; SQL tabanlı veri dönüşümleriyle tutarlı ve ölçülebilir bir veri akışı oluşturulması
- Raporlama ve Analiz için Optimize Edilmiş Veri Yapısı: İş zekası (BI) ve makine öğrenimi (ML) modellerine temiz ve yapılandırılmış veri sağlanması; Veri ambarlarında (data warehouse) tarihsel veri takibi ile trend analizlerinin kolaylaştırılması

- Uyumluluk ve Denetim İzlenebilirliği: KVKK, GDPR gibi düzenlemelere uygun veri doğrulama ve loglama mekanizmalarının kurulması: Veri kalite raporlarıyla şeffaf ve denetlenebilir bir süreç yönetimi
- ✓ **Veri Temizleme**, hatalı, eksik, yinelenen veya tutarsız verilerin tespit edilmesi ve temizlenmesidir.

1. GİRİŞ

1.1. Veri Temizleme

Çözmeye çalıştığımız sorunla ilgili eksik, yanlış veya alakasız yani kirli verilerin eksiksiz, doğru ve alaklı veriler haline getirmektir yani temizlemektir.

1.1.1. SQL ile Veri Temizleme Teknikleri

Eksik verilerin yönetimi:

```
-- NULL değerleri varsayılan değerlerle değiştirme
UPDATE Musteriler
SET Telefon = 'Belirtilmemiş'
WHERE Telefon IS NULL;
-- Eksik kayıtları silme
DELETE FROM Siparisler
WHERE MusteriID IS NULL OR UrunID IS NULL;
```

Tutarsız verilerin düzeltilmesi:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* X

-- Format düzeltme (telefon numaraları için)
UPDATE Musteriler
SET Telefon = REPLACE(REPLACE(Replace(Telefon, ' ', ''), '(', ''), ')', '')
WHERE Telefon LIKE '%(%' OR Telefon LIKE '%)%';
-- Yanlış veri tiplerini düzeltme
UPDATE Urunler
SET Fiyat = CAST(REPLACE(Fiyat, ',', '.') AS DECIMAL(10,2))
WHERE Fiyat LIKE '%,%';
```

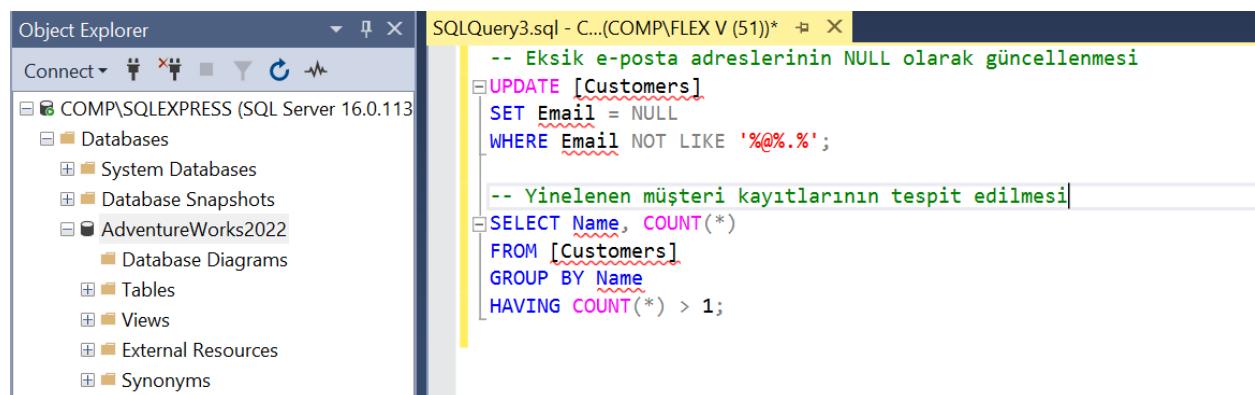
Tekilleştirme (Deduplication):

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* X

-- Yinelenen kayıtları bulma ve silme
WITH CTE AS (
    SELECT *,
        ROW_NUMBER() OVER (PARTITION BY TCKimlikNo ORDER BY KayitTarihi DESC) AS RN
    FROM Musteriler
)
DELETE FROM CTE WHERE RN > 1;
```

Örnek:

Eksik veya geçersiz e-posta adreslerinin temizlenmesi (SQL):



The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure for 'COMP\SQLEXPRESS (SQL Server 16.0.113)'. The 'Tables' node under 'AdventureWorks2022' is expanded, showing various table names like 'CustomerDemographic', 'Customer', etc. On the right, the main query editor window has a yellow title bar 'SQLQuery3.sql - C...(COMP\FLEX V (51))* X'. It contains two parts of T-SQL code. The first part, enclosed in a comment block, updates the 'Customers' table by setting 'Email' to NULL for rows where it does not contain a valid email address pattern ('%@%.%'). The second part, also in a comment block, selects the count of rows for each 'Name' value from the 'Customers' table, grouping by 'Name' and filtering for counts greater than 1.

```
-- Eksik e-posta adreslerinin NULL olarak güncellenmesi
UPDATE [Customers]
SET Email = NULL
WHERE Email NOT LIKE '%@%.%';

-- Yinelenen müşteri kayıtlarının tespit edilmesi
SELECT Name, COUNT(*)
FROM [Customers]
GROUP BY Name
HAVING COUNT(*) > 1;
```

1.2. ETL (Extract, Transform, Load) Süreçleri

ETL süreçleri, büyük veri kümelerinin işlenmesi, temizlenmesi ve hedef sistemlere yüklenmesi için kritik bir altyapı sağlar.

ETL Araçları:



1.2.1. Veri Dönüşürme

Farklı formatlardaki verileri analiz için tutarlı hale getirmek.

Örnek: Tarih formatlarını standardize etmek ve şehir adlarını büyük harfe çevirmek:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))*  ↗ X

-- Tarihi 'yyyy-MM-dd' biçimine dönüştürme
SELECT CONVERT(VARCHAR(10), OrderDate, 120) AS StandardDate FROM Orders;
-- Şehir isimlerini büyük harfe dönüştürme
UPDATE Customers
SET City = UPPER(City);
```

Standartlaştırma Teknikleri:

Veri formatlarını birleştirmeye:

SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X

```
-- Tarih formatlarını standart hale getirme
UPDATE Siparisler
SET SiparisTarihi = CASE
    WHEN SiparisTarihi LIKE '__._._.' THEN
        CONVERT(DATETIME, SiparisTarihi, 104) -- DD.MM.YYYY
    WHEN SiparisTarihi LIKE '_-__-' THEN
        CONVERT(DATETIME, SiparisTarihi, 110) -- MM-DD-YYYY
    ELSE TRY_CONVERT(DATETIME, SiparisTarihi)
END;
```

Farklı kaynakları entegre etme:

SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X

```
-- Farklı sistemlerden gelen müşteri verilerini birleştirme
INSERT INTO DimMusteri (MusteriKey, Ad, Soyad, Sehir, Ulke)
SELECT
    c.CustomerID,
    CASE
        WHEN c.FirstName IS NULL THEN p.Ad ELSE c.FirstName
    END,
    CASE
        WHEN c.LastName IS NULL THEN p.Soyad ELSE c.LastName
    END,
    COALESCE(c.City, p.Sehir, 'Bilinmiyor'),
    COALESCE(c.Country, p.Ulke, 'Bilinmiyor')
FROM CRM.Customers c
FULL OUTER JOIN POS.Musteriler p ON c.Email = p.Eposta;
```

Hiyerarşik verileri düzleştirme:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- JSON verisini ilişkisel modele dönüştürme
INSERT INTO UrunOzellikleri (UrunID, OzellikAdi, Deger)
SELECT
    UrunID,
    j.OzellikAdi,
    j.Deger
FROM Urunler
CROSS APPLY OPENJSON(UrunDetay, '$.ozellikler')
WITH (
    OzellikAdi NVARCHAR(100),
    Deger NVARCHAR(255)
) AS j;
```

1.2.2. Veri Yükleme

Temizlenmiş ve dönüştürülmüş verilerin hedef veritabanına (data warehouse gibi) aktarılması.

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- Temizlenmiş verileri DataWarehouse tablosuna yükleme
INSERT INTO DW_Customers (CustomerID, Name, Email, City)
SELECT CustomerID, Name, Email, City
FROM Staging_Customers
WHERE Email IS NOT NULL;
```

Yükleme Stratejileri:

Tam yükleme:

```
TRUNCATE TABLE HedefTablo;
INSERT INTO HedefTablo
SELECT * FROM KaynakTablo;
```

Artımlı yükleme:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- Değişen kayıtları bulma ve güncelleme
MERGE INTO HedefTabelo AS target
USING KaynakTabelo AS source
ON target.ID = source.ID
WHEN MATCHED AND (target.CheckSum <> CHECKSUM(source.*)) OR target.CheckSum IS NULL) THEN
    UPDATE SET target.* = source.* target.GuncellemeTarihi = GETDATE()
WHEN NOT MATCHED THEN
    INSERT (ID, ...) VALUES (source.ID, ...);
```

Partition Switching (büyük veri yüklemeleri için):

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- Geçici tablo oluşturma ve verileri yükleme
CREATE TABLE Temp_Tablo (...) WITH (PARTITION = ...);

-- Verileri işleme
INSERT INTO Temp_Tablo SELECT * FROM Kaynak WHERE ...;

-- Partition değiştirme
ALTER TABLE HedefTabelo SWITCH PARTITION X TO Temp_Tablo;
```

1.2.3. Veri Kalitesi Raporları

Temizlenen verilerin ne kadar düzeltildiği, hangi tür hataların giderildiği gibi analizler sunmak.

Veri Kalitesi Metrikleri ve Raporlanması

Temel veri kalitesi ölçümleri:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- Eksik değer analizi
SELECT
    COUNT(*) AS ToplamKayit,
    SUM(CASE WHEN Ad IS NULL THEN 1 ELSE 0 END) AS EksikAd,
    SUM(CASE WHEN Soyad IS NULL THEN 1 ELSE 0 END) AS EksikSoyad,
    SUM(CASE WHEN Telefon IS NULL THEN 1 ELSE 0 END) AS EksikTelefon,
    (SUM(CASE WHEN Ad IS NULL OR Soyad IS NULL OR Telefon IS NULL THEN 1 ELSE 0 END) * 100.0 / COUNT(*)) AS EksikYuzde
FROM Musteriler;
```

Veri tutarlılık raporları:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- Referans bütünlüğü kontrolü
SELECT
    'Siparisler' AS TabloAdi,
    COUNT(*) AS GecersizKayitSayisi
FROM Siparisler s
LEFT JOIN Musteriler m ON s.MusteriID = m.MusteriID
WHERE m.MusteriID IS NULL;
```

Veri temizleme özet raporu:

```
SQLQuery3.sql - C...(COMP\FLEX V (51))* ✎ X
-- ETL sürecinde yapılan değişikliklerin özeti
SELECT
    'Format Düzeltme' AS IslemTipi,
    COUNT(*) AS EtkilenenKayitSayisi
FROM Musteriler
WHERE Telefon LIKE '%(%' OR Telefon LIKE '%)%'
UNION ALL
SELECT
    'Eksik Değer Doldurma',
    COUNT(*)
FROM Musteriler
WHERE Telefon = 'Belirtilmemis';
```

SONUÇ

Bu proje kapsamında, büyük ve çeşitli veri kaynaklarının analiz edilebilir hale getirilmesi amacıyla ETL (Extract, Transform, Load) süreçleri detaylı olarak tasarlanmıştır. Özellikle verilerin kalitesini artırmak, anlamlı ve tutarlı hale getirmek için veri temizleme, dönüştürme ve yükleme işlemleri sistematik biçimde gerçekleştirilmiştir.

Projede yapılan temel çalışmalar şunlardır:

- ✓ **Veri Temizleme:** Hatalı formatta, eksik veya yinelenen verileri tespit etme; SQL sorguları kullanarak bu verileri standartlaştırılmış ve tutarsızlıklar giderilmiş hale dönüştürmek.
- ✓ **Veri Dönüştürme:** Farklı kaynaklardan gelen veriler tek bir yapıda ve analiz edilebilir formatta dönüştürüllererek işlenebilir hale getirilmiştir. Örneğin, tarih formatları ve adres bilgilerindeki farklar düzeltilmiştir.
- ✓ **Veri Yükleme:** Temizlenmiş verilerin, hedef sistemlere (örneğin veri ambarına) başarıyla yüklenmesi ve raporlama sistemlerinin kullanımına sunulması.
- ✓ **Veri Kalitesi Raporları:** Temizleme sürecinin başarısının ölçümü, eksik ya da hatalı veri oranlarının raporlanarak sürecin doğruluğu ve etkinliğinin analiz edilmesi.

Bu süreçler sayesinde, sistemdeki veri tutarlılığı ve güvenilirliği önemli ölçüde artırılır. Ayrıca, veri analizi ve raporlama süreçleri daha sağlıklı ve hızlı bir şekilde gerçekleştirilebilir.

Bu projenin çıktıları, özellikle büyük veri setleriyle çalışan kurumlar için karar destek sistemlerinin temelini oluşturacak niteliktedir. ETL süreçlerinin doğru planlanması ve uygulanması sayesinde, veriden maksimum fayda sağlanması mümkün olduğu söylenebilir.

KAYNAKLAR

1. <https://www.youtube.com/watch?v=rnmoT0P6Oyg>
2. Anlatım Videosu: <https://www.youtube.com/watch?v=7bq16kJ01FY>
3. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery5.sql>

PROJE 4. VERİTABANI YEDEKLEME VE OTOMASYON ÇALIŞMASI

PROJE AMACI: Bu projenin amacı, SQL Server ortamında veritabanı yedekleme işlemlerini otomatikleştirerek yönetim süreçlerini kolaylaştırmak, yedek bütünlüğünü düzenli olarak denetlemek ve olası yedekleme hatalarına karşı sistem yöneticilerini zamanında bilgilendirebilecek otomasyon sistemleri kurmaktır.

GEREKEN ARAÇLAR:

- SQL Server Management Studio (SSMS): Veritabanı yönetimi ve T-SQL scriptleri için.
- SQL Server Agent: Otomatikleştirme ve zamanlanmış yedekleme işleri için.
- PowerShell: Script bazlı otomasyon ve loglama işlemleri için.
- E-posta sunucusu (SMTP): Hatalı işlemlerde ya da Job başarısız olduğunda e-posta uyarıları göndermek için.

1. GİRİŞ

Veritabanı yedekleme, olası donanım arızaları, insan hataları veya siber saldırılar gibi durumlarda veri kaybını önlemek için kritik öneme sahiptir. Özellikle otomatikleştirilmiş yedekleme sistemleri, veritabanı yöneticilerinin manuel işlem yükünü azaltır ve hata olasılığını düşürür.

1.1. SQL Server Agent ile Otomatik Yedekleme

The screenshot shows a SQL query window titled "SQLQuery2.sql - C... (COMP\FLEX V (62))". The query is:

```
BACKUP DATABASE AdventureWorks2022  
TO DISK = 'C:\Backups\AdventureWorks2022_FULL.bak'  
WITH FORMAT, INIT, NAME = 'Full Backup';
```

Below the query window is a "Messages" pane. It displays the following output:

```
Processed 25432 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.  
BACKUP DATABASE successfully processed 25434 pages in 0.436 seconds (455.732 MB/sec).  
  
Completion time: 2025-06-08T23:30:01.7637006+03:00
```

Task Scheduler

File Action View Help

Task Scheduler Library

Name	Status	Triggers	Next Run Time	Last Run Time
Launch Adob...	Ready	At 19:25 every day	9.06.2025 19:25:00	7.06.2025 15:07:39
MicrosoftEd...	Ready	Multiple triggers defined	9.06.2025 22:19:30	7.06.2025 22:56:43
MicrosoftEd...	Ready	At 21:49 every day - After triggered, repeat every 1 hour for a duration of 1 day.	8.06.2025 23:49:30	8.06.2025 01:30:48
OneDrive Per...	Ready	At 22:00 on 1.05.1992 - After triggered, repeat every 1.000000 indefinitely.	9.06.2025 23:22:34	7.06.2025 15:07:39
OneDrive Re...	Ready	At 23:47 on 3.06.2025 - After triggered, repeat every 1.000000 indefinitely.	8.06.2025 23:47:41	7.06.2025 23:47:42
OneDrive Sta...	Ready	At log on of COMPFLEX V		3.06.2025 09:53:09
Weekly_Full_...	Ready	At 23:59 on 8.06.2025	8.06.2025 23:59:01	30.11.1999 00:00:00
ZoomUpdat...	Ready	At 00:44 every day - After triggered, repeat every 12:00:00 for a duration of 1 day.	9.06.2025 00:44:00	8.06.2025 17:49:26

General Triggers Actions Conditions Settings History (disabled)

Name: Weekly_Full_Backup_AdventureWorks

Location: \

Author: COMPFLEX V

Description: AdventureWorks2022 için haftalık tam yedekleme

Security options

When running the task, use the following user account:
FLEX V

Run only when user is logged on
 Run whether user is logged on or not
 Do not store password. The task will only have access to local resources

Actions

- Task Scheduler Library
 - Create Basic Task...
 - Create Task...
 - Import Task...
 - Display All Running Tasks
 - Enable All Tasks History
 - New Folder...
 - View
 - Refresh
 - Help
- Selected Item
 - Run
 - End
 - Disable
 - Export...
 - Properties
 - Delete
 - Help

Services

File Action View Help

Services (Local)

Name	Description	Status	Startup Type	Loc
Smart Card Removal Policy	Allows the s...	Running	Manual	Loc
SNMP Trap	Receives tra...	Running	Manual	Loc
Software Protection	Enables the ...	Running	Automatic (De...	Ne
Spatial Data Service	This service i...	Running	Manual	Loc
Spot Verifier	Verifies pote...	Running	Manual (Trigg...	Loc
SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic (De...	NT
SQL Server (MSSQLSERVER01)	Provides sto...	Running	Automatic (De...	NT
SQL Server (SQLEXPRESS)	Provides sto...	Running	Automatic (De...	NT
SQL Server Agent (MSSQLSE...	Provides sto...	Running	Automatic	NT
SQL Server Agent (MSSQLSE...	Provides sto...	Running	Manual	NT
SQL Server Agent (SQLEXPRE...	Provides sto...	Running	Disabled	Ne
SQL Server Browser	Provides sto...	Running	Disabled	Loc
SQL Server CEIP service (MSS...	Provides sto...	Running	Automatic (De...	NT
SQL Server CEIP service (MSS...	Provides sto...	Running	Automatic (De...	NT
SQL Server CEIP service (SQL...	Provides sto...	Running	Automatic (De...	NT
SQL Server VSS Writer	Provides sto...	Running	Automatic	Loc
SSDP Discovery	Provides sto...	Running	Manual	Loc
State Repository Service	Provides sto...	Running	Automatic	Loc
Steam Client Service	Provides sto...	Running	Manual	Loc
Still Image Acquisition Events	Provides sto...	Running	Manual	Loc
Storage Service	Provides sto...	Running	Automatic (De...	Loc

Start Stop Pause Resume Restart All Tasks Properties Help

Extended Standard

Start service SQL Server Agent (MSSQLSERVER) on Local Computer

```
[Administrator: C:\WINDOWS\SYSTEM32\cmd.exe] - X
Msg 911, Level 16, State 11, Server ComP, Line 1
Database 'AdventureWorks2022' does not exist. Make sure that the name is entered correctly.
Msg 3013, Level 16, State 1, Server ComP, Line 1
BACKUP DATABASE is terminating abnormally.
Msg 3634, Level 16, State 1, Server ComP, Line 13
The operating system returned the error '5(Access is denied.)' while attempting 'RestoreContainer::ValidateTargetForCreation' on 'C:\Backups\DBYedek.mdf'.
Msg 3156, Level 16, State 8, Server ComP, Line 13
File 'AdventureWorks2022' cannot be restored to 'C:\Backups\DBYedek.mdf'. Use WITH MOVE to identify a valid location for
the file.
Msg 3634, Level 16, State 1, Server ComP, Line 13
The operating system returned the error '5(Access is denied.)' while attempting 'RestoreContainer::ValidateTargetForCreation' on 'C:\Backups\DBYedek_log.ldf'.
Msg 3156, Level 16, State 8, Server ComP, Line 13
File 'AdventureWorks2022_log' cannot be restored to 'C:\Backups\DBYedek_log.ldf'. Use WITH MOVE to identify a valid loca
tion for the file.
Msg 3119, Level 16, State 1, Server ComP, Line 13
Problems were identified while planning for the RESTORE statement. Previous messages provide details.
Msg 3013, Level 16, State 1, Server ComP, Line 13
RESTORE DATABASE is terminating abnormally.
Press any key to continue . . .
```

1.1.1. Tam Yedek Zamanlanmış Görev ve Log Yedekleme – 15 Dakikada Bir

```

SQLQuery4.sql - C...(COMP\FLEX V (51))*  ↗ ×

USE msdb;
GO
IF NOT EXISTS (SELECT 1 FROM sys.dm_server_services WHERE servicename LIKE 'SQL Server Agent%' AND status_desc = 'Running')
BEGIN
    RAISERROR('SQL Server Agent servisi çalışmıyor. Lütfen başlatın ve tekrar deneyin.', 16, 1);
    RETURN;
END
-- Job oluşturma
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysjobs WHERE name = 'Weekly_Full_Backup_AdventureWorks')
BEGIN
    EXEC dbo.sp_add_job
        @job_name = N'Weekly_Full_Backup_AdventureWorks',
        @enabled = 1,
        @description = N'AdventureWorks2022 veritabanı için haftalık tam yedek';

    EXEC sp_add_jobstep
        @job_name = N'Weekly_Full_Backup_AdventureWorks',
        @step_name = N'Backup Database',
        @subsystem = N'TSQL',
        @command = N'
            BACKUP DATABASE [AdventureWorks2022]
            TO DISK = N'E:\Backups\Adventureworks_Full_$(ESCAPE_SQUOTE(DATE)).bak'
            WITH COMPRESSION, STATS = 10, CHECKSUM;
        ',
        @database_name = N'master';
    EXEC dbo.sp_add_schedule
        @schedule_name = N'Weekly_Saturday_2AM',
        @freq_type = 8, -- Weekly
        @freq_interval = 64, -- Saturday
        @active_start_time = 020000, -- 02:00:00
        @freq_recurrence_factor = 1; -- Haftalık tekrar
    EXEC sp_attach_schedule
        @job_name = N'Weekly_Full_Backup_AdventureWorks',
        @schedule_name = N'Weekly_Saturday_2AM';
    EXEC dbo.sp_add_jobserver
        @job_name = N'Weekly_Full_Backup_AdventureWorks',
        @server_name = N'(local)';
    PRINT 'Job başarıyla oluşturuldu: Weekly_Full_Backup_AdventureWorks';
END
ELSE
BEGIN
    PRINT 'Job zaten mevcut: Weekly_Full_Backup_AdventureWorks';
END
GO

USE msdb;
GO
-- Log yedekleme job'u
IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysjobs WHERE name = 'Log_Backup_AdventureWorks_15Min')
BEGIN
    EXEC msdb.dbo.sp_add_job
        @job_name = N'Log_Backup_AdventureWorks_15Min',
        @enabled = 1,
        @description = N'AdventureWorks2022 transaction log yedekleri (15 dakikada bir)';
    EXEC msdb.dbo.sp_add_jobstep
        @job_name = N'Log_Backup_AdventureWorks_15Min',
        @step_name = N'Backup Log',
        @command = N'
            BACKUP LOG [AdventureWorks2022]
            TO DISK = N'E:\Backups\AdventureWorks_Log_$(ESCAPE_SQUOTE(DATE)).trn'
            WITH COMPRESSION, STATS = 5;
        ',
        @database_name = N'master';
    EXEC msdb.dbo.sp_add_schedule
        @schedule_name = N'Every_15_Minutes',
        @freq_type = 4, -- Daily
        @freq_interval = 1,
        @freq_subday_type = 4, -- Minutes
        @freq_subday_interval = 15,
        @freq_recurrence_factor = 1;
    EXEC msdb.dbo.sp_attach_schedule
        @job_name = N'Log_Backup_AdventureWorks_15Min',
        @schedule_name = N'Every_15_Minutes';
    EXEC msdb.dbo.sp_add_jobserver
        @job_name = N'Log_Backup_AdventureWorks_15Min',
        @server_name = N'(local)';
    PRINT 'Job başarıyla oluşturuldu: Log_Backup_AdventureWorks_15Min';
END
ELSE
BEGIN
    PRINT 'Job zaten mevcut: Log_Backup_AdventureWorks_15Min';
END
GO

```

100 % < Messages
SQLServerAgent is not currently running so it cannot be notified of this action.
Job basariyla olusturuldu: Log_Backup_AdventureWorks_15Min
Completion time: 2025-06-02T01:01:37.0752702+03:00

1.2. PowerShell ile Yedekleme Otomasyonu

- ✓ **BackupRestoreDB.ps1 PowerShell script dosyası**, otomatik işlemleri (backup alma, dosya taşıma, servis kontrolü vs.) yapmak için komutları topluca çalıştırmanı sağlamaktadır. Bu projede .ps1 dosyası SQL Server'dan bir veritabanının (AdventureWorks2022) yedeklemesini almakta ve eğer eski yedek varsa onu silip yenisini yüklemektedir (restore). İşlem sonucunu da bir log dosyasına yazar.
Avantajları şu şekildedir:
 - Tek seferde yedekleme ve geri yükleme işlemini otomatik yapar.
 - El ile komut yazmamız gereklidir.
 - Hataları log dosyasına yazar, takip etmesi kolaydır.

```

# Set current date and time for logging
$Tarih = Get-Date -Format "yyyy-MM-dd HH:mm:ss"
$BackupDir = "C:\Backups"
$LogDosyasi = "$BackupDir\backup_log.txt"

# SQLCMD commands
$SQLKomut = @"
BACKUP DATABASE AdventureWorks2022
TO DISK = '$BackupDir\DBYedek.bak'
WITH FORMAT,
      MEDIANAME = 'DBYedekMedya',
      NAME = 'DB Tam Yedek';

IF EXISTS (SELECT name FROM sys.databases WHERE name = 'DBYedek')
BEGIN
    ALTER DATABASE DBYedek SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE DBYedek;
END;

RESTORE DATABASE DBYedek
FROM DISK = '$BackupDir\DBYedek.bak'
WITH
      MOVE 'AdventureWorks2022' TO '$BackupDir\DBYedek.mdf',
      MOVE 'AdventureWorks2022_log' TO '$BackupDir\DBYedek_log.ldf',
      REPLACE;
"@

# SQL Server instance name (adjust if needed)
$instance = "localhost"

try {
    # Run the SQLCMD using PowerShell
    sqlcmd -S $instance -Q $SQLKomut
    Add-Content -Path $LogDosyasi -Value "$Tarih - Backup and restore completed successfully."
}
catch {
    Add-Content -Path $LogDosyasi -Value "$Tarih - ERROR: $($_.Exception.Message)"
}

```

- ✓ **BackupRestoreDB.bat Batch dosyası** .ps1 gibi bir PowerShell dosyasını veya başka komutları çift tıklamayla çalıştırır. Bu projede .bat dosyası .ps1 dosyasını çalıştırır.

```
@echo off  
PowerShell.exe -ExecutionPolicy Bypass -File "C:\Backups\BackupRestoreDB.ps1"  
pause
```

SQLQuery7.sql - C...(COMP\FLEX V (55))*

```
-- BACKUP THE DATABASE  
BACKUP DATABASE AdventureWorks2022  
TO DISK = 'C:\Backups\DBYedek.bak'  
WITH FORMAT,  
      MEDIANAME = 'DBYedekMedya',  
      NAME = 'DB Tam Yedek';  
GO
```

100 %

Messages

```
Processed 25432 pages for database 'AdventureWorks2022', file 'AdventureWorks2022' on file 1.  
Processed 2 pages for database 'AdventureWorks2022', file 'AdventureWorks2022_log' on file 1.  
BACKUP DATABASE successfully processed 25434 pages in 0.287 seconds (692.331 MB/sec).
```

```
Completion time: 2025-06-07T18:10:20.1216117+03:00
```

```
-- VIEW HEADER INFO FROM THE BACKUP FILE
RESTORE HEADERONLY
FROM DISK = 'C:\Backups\DBYedek.bak';
GO
```

100 %

Results Messages

	BackupName	BackupDescription	BackupType	ExpirationDate	Compressed	Position	DeviceType	UserName	ServerName	DatabaseName	DatabaseVersion	FileId
1	DB Tam Yedek	NULL	1	NULL	0	1	2	COMP\FLEX V	Comp\SQLExpress	AdventureWorks2022	957	1

```
-- DROP DATABASE IF EXISTS
IF EXISTS (SELECT name FROM sys.databases WHERE name = 'DBYedek')
BEGIN
    ALTER DATABASE DBYedek SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
    DROP DATABASE DBYedek;
END
GO
```

100 %

Messages

Commands completed successfully.

Completion time: 2025-06-07T18:12:48.1154193+03:00

```
-- RESTORE THE DATABASE FROM BACKUP FILE
RESTORE DATABASE DBYedek
FROM DISK = 'C:\Backups\DBYedek.bak'
WITH
    MOVE 'AdventureWorks2022' TO 'C:\Backups\DBYedek.mdf',
    MOVE 'AdventureWorks2022_log' TO 'C:\Backups\DBYedek_log.ldf',
    REPLACE;
GO
```

100 % ▶

Messages

```
Processed 25432 pages for database 'DBYedek', file 'AdventureWorks2022' on file 1.
Processed 2 pages for database 'DBYedek', file 'AdventureWorks2022_log' on file 1.
RESTORE DATABASE successfully processed 25434 pages in 0.587 seconds (338.499 MB/sec).

Completion time: 2025-06-07T18:13:51.6117139+03:00
```

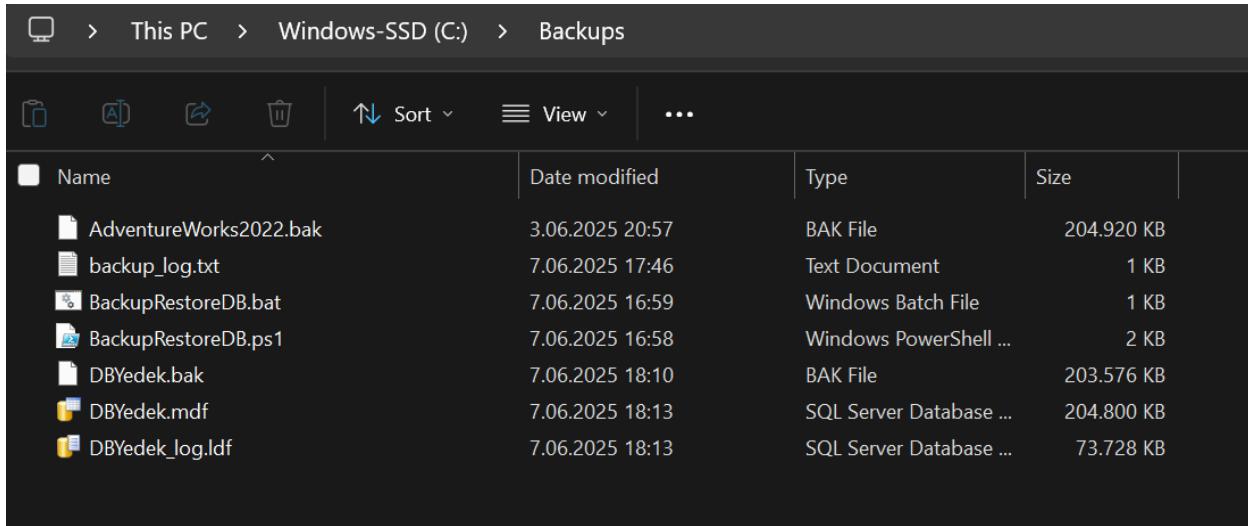
```
RESTORE FILELISTONLY
FROM DISK = 'C:\Backups\DBYedek.bak';
GO
```

```
RESTORE FILELISTONLY FROM DISK = 'C:\Backups\DBYedek.bak',
```

100 % ▶

Results Messages

LogicalName	PhysicalName	Type	FileGroupName	Size	MaxSize	FileId	CreateLSN	DropLSN	UniqueId
1 AdventureWorks2022	C:\Program Files\Microsoft SQL Server\MSSQL16.SQL...	D	PRIMARY	209715200	35184372080640	1	0	0	FBAD6D5F-27
2 AdventureWorks2022_log	C:\Program Files\Microsoft SQL Server\MSSQL16.SQL...	L	NULL	75497472	2199023255552	2	0	0	3DF51AFC-02f



1.3. Denetim ve Raporlama

Son yedekleri kontrol eden sorgu:

```

SELECT
    database_name AS [Database],
    CASE type
        WHEN 'D' THEN 'Full Backup'
        WHEN 'L' THEN 'Log Backup'
    END AS [Backup Type],
    backup_start_date AS [Start Time],
    backup_finish_date AS [Finish Time],
    DATEDIFF(MINUTE, backup_start_date, backup_finish_date) AS [Duration (Min)],
    CAST(backup_size/1048576 AS DECIMAL(10,2)) AS [Size (MB)]
FROM msdb.dbo.backupset
WHERE database_name = 'AdventureWorks2022'
ORDER BY backup_finish_date DESC;

```

Database	Backup Type	Start Time	Finish Time	Duration (Min)	Size (MB)
AdventureWorks2022	Full Backup	2025-05-29 16:13:11.000	2025-05-29 16:13:12.000	0	198.78
AdventureWorks2022	Full Backup	2025-04-25 02:37:54.000	2025-04-25 02:37:54.000	0	198.40
AdventureWorks2022	Full Backup	2025-04-25 02:32:40.000	2025-04-25 02:32:41.000	0	198.40
AdventureWorks2022	Full Backup	2025-04-24 23:39:39.000	2025-04-24 23:39:40.000	0	198.40
AdventureWorks2022	Full Backup	2025-04-24 23:03:23.000	2025-04-24 23:03:23.000	0	198.40
AdventureWorks2022	Full Backup	2023-05-23 11:56:14.000	2023-05-23 11:56:15.000	0	200.09

SONUÇ

Bu proje kapsamında, SQL Server ortamında yedekleme işlemleri otomatikleştirilmiş, hem tam hem log yedekleme stratejileri uygulanmış, ayrıca PowerShell scripti ile kolay otomasyon ve log takibi sağlanmıştır.

Projenin çıktıları:

- SQL Server Agent ile zamanlanmış tam ve log yedekleme işlerinin kurulumu.
- PowerShell scripti ile manuel komut gereksinimini ortadan kaldırın bir yedekleme otomasyonu.
- Backup başarılı/başarısızlık durumlarının loglanması.
- Olası felaket durumlarda geri yükleme (restore) işleminin hızlı yapılabilmesi.

Bu yapı, kurumsal veritabanı yönetimi için sürdürülebilir, güvenli ve kolay yönetilebilir bir yedekleme altyapısı sunmaktadır.

KAYNAKLAR

1. GitHub Linki: <https://github.com/Impasbaa/MSSQL/blob/main/SQLQuery7.sql>
2. Microsoft Docs: <https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/restore-a-database-backup-using-ssms?view=sql-server-ver17>