

Ankara Üniversitesi

Bilgisayar Mühendisliği Bölümü

BLM4522 Ağ Tabanlı Paralel Dağıtım Sistemleri Dersi Raporu

Artun Cankar – 21290507

Berkay Bozpınar – 21290178

25.04.2025

Github:

https://github.com/DarkSiderVV/BLM4522_ProjeVideolar-

1. Veritabanı Performans Optimizasyonu ve İzleme

Bu rapor, AdventureWorks veritabanı üzerinde gerçekleştirilen performans analizi ve optimizasyon çalışmalarını özetlemektedir. SQL Server'ın sunduğu dinamik yönetim görünümleri (DMV'ler) kullanılarak veritabanı performansı izlenmiş ve iyileştirme fırsatları belirlenmiştir. Kod temel olarak 3 ana kısımdan oluşuyor:

1. Test İş Yükü Oluşturma

```
SELECT TOP 500 * FROM Sales.SalesOrderDetail ORDER BY SalesOrderID DESC;
SELECT * FROM HumanResources.Employee WHERE Gender = 'M';
SELECT * FROM Person.Person WHERE LastName LIKE 'S%';
```

Buradaki amaç, sorgu trafiği oluşturarak DMV'lerin doğru veri toplamasını sağlamak. Farklı sorgu desenleri (filtreleme, sıralama, LIKE kullanımı) ile performans sorunlarını tespit etmek.

Analiz Edilen Sorgular:

Sorgu	Açıklama	Performans Etkisi
Sales.SalesOrderDetail	Büyük bir tablodan 500 satır çekme	Disk I/O ve bellek kullanımı
HumanResources.Employee	Cinsiyet filtresi ile tarama	İndeks kullanım verimliliği
Person.Person	LIKE 'S%' ile metin arama	Full-table scan riski

2. Eksik İndeks Önerileri Analizi

```
SELECT TOP 20
    CONVERT(DECIMAL(10,2),
        migs.avg_total_user_cost * migs.avg_user_impact * (migs.user_seeks +
        migs.user_scans) / 100
    ) AS improvement_measure,

    'CREATE INDEX [IX_' + OBJECT_NAME(mid.object_id) + '_' +
    AS create_index_statement,

    migs.user_seeks,
    migs.user_scans,
    SCHEMA_NAME(o.schema_id) AS schema_name,
    OBJECT_NAME(mid.object_id) AS table_name
FROM sys.dm_db_missing_index_groups mig
WHERE mid.database_id = DB_ID('AdventureWorks')
ORDER BY improvement_measure DESC;
```

Sonuçlar ve Öneriler:

Ölçüm	Açıklama	Optimizasyon Önerisi
improvement_measure	İndeksin performansa etkisi (0-100)	>50 olanlar yüksek öncelikli
user_seeks	İndeksin kaç kez arandığı	Yüksekse, indeks kritik
create_index_statement	Otomatik üretilen SQL komutu	Uygulanarak sorgu hızı 2-10x artabilir

3. Tablo Boyutu ve İndeks Analizi

SELECT

```
SCHEMA_NAME(o.schema_id) AS schema_name,  
o.name AS table_name,  
SUM(p.rows) AS row_count,  
CAST(SUM(a.total_pages) * 8.0 / 1024 AS DECIMAL(10,2)) AS size_mb,  
COUNT(DISTINCT i.index_id) AS index_count,  
SUM(CASE WHEN i.type = 1 THEN 1 ELSE 0 END) AS clustered_indexes,  
SUM(CASE WHEN i.type = 2 THEN 1 ELSE 0 END) AS nonclustered_indexes
```

FROM sys.objects o

WHERE o.type = 'U'

AND o.is_ms_shipped = 0

AND o.name NOT LIKE 'sys%'

GROUP BY SCHEMA_NAME(o.schema_id), o.name

ORDER BY size_mb DESC;

Bu bölüm veritabanındaki tabloların boyutunu ve indeks yapılarını analiz etmektedir. Her kullanıcı tablosunun satır sayısını ve depolama boyutunu (MB) hesaplar, her tablo için toplam indeks sayısını gösterir, kümelenmiş ve kümelenmemiş indeksleri ayrı ayrı sayar, tablolar boyut sırasına göre listelenir (en büyük tablolar en üstte)

4. Programın Çıktısı

	schema_name	table_name	row_count	size_mb	index_count	clustered_indexes	nonclustered_indexes
1	Person	Person	99860	32.07	3	3	2
2	Sales	SalesOrderDetail	363951	30.77	3	1	2
3	Production	TransactionHist..	340329	9.96	3	1	2
4	Production	TransactionHist..	267759	7.90	3	1	2
5	Production	WorkOrderRout..	134262	6.64	2	1	1
6	dbo	DatabaseLog	6384	6.46	2	0	1
7	Production	WorkOrder	217773	6.21	3	1	2
8	Person	Address	117684	5.70	4	3	3

Results Messages															
SalesOrderDetail															
SalesOrderID	SalesOrderDetailID	CarrierTrackingNumber	OrderQty	ProductID	SpecialOfferID	UnitPrice	UnitPriceDiscount	LineTotal	rowguid	ModifiedDate					
1	75123	121317	NULL	1	712	1	8.99	0.00	8.990000	73646D26-0461-450D-8019-2C6C858619CA	2014-06-30 00:00:00.000				
2	75123	121316	NULL	1	879	1	159.00	0.00	159.000000	75A89C6A-C60A-47EA-8A52-B52A9C435864	2014-06-30 00:00:00.000				
3	75123	121315	NULL	1	878	1	21.98	0.00	21.980000	C18B8476-429F-4BB1-828E-2BE5F82A0A51	2014-06-30 00:00:00.000				
4	75122	121314	NULL	1	712	1	8.99	0.00	8.990000	84F1C363-1C50-4442-BE16-541C5986E12C	2014-06-30 00:00:00.000				
5	75122	121313	NULL	1	878	1	21.98	0.00	21.980000	8CAD6675-18CC-4F47-8287-97B41A8EE47D	2014-06-30 00:00:00.000				
6	75121	121312	NULL	1	707	1	34.99	0.00	34.990000	AF25E491-73B2-4EB8-B2FE-9A193C31A83F	2014-06-30 00:00:00.000				
7	75121	121311	NULL	1	930	1	35.00	0.00	35.000000	86638D4E-63C5-4014-AC43-451FE68D1C99	2014-06-30 00:00:00.000				
8	75121	121310	NULL	1	921	1	4.99	0.00	4.990000	8B8A0C91-BF1A-4D8C-BFD0-95B5878C2567	2014-06-30 00:00:00.000				
BusinessEntity															
BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate	MaritalStatus	Gender	HireDate	SalariedFlag	VacationHours	SickLeaveHours	CurrentFlag	rowguid	
1	1	295847284	adventure-works\ken0	NULL	NULL	1969-01-29	S	M	2009-01-14	1	99	69	1	F01251E5-98B8FB2C	
2	3	509647174	adventure-works\roberto0	0x5AC0	2	Engineering Manager	1974-11-12	M	2007-11-11	1	2	21	1	59747955-E39056F1-4F46DECA	
3	4	112457891	adventure-works\rob0	0x5AD6	3	Senior Tool Designer	1974-12-23	S	M	2007-12-05	0	48	80	1	EAA43680-59747955-E39056F1-4F46DECA
4	6	998320692	adventure-works\josset0	0x5ADE	3	Design Engineer	1959-03-11	M	2008-01-24	1	6	23	1	1D955171-4F46DECA	
5	7	134969118	adventure-works\idylan0	0x5AE1	3	Research and Development	1987-02-24	M	2009-02-08	1	61	50	1	4F46DECA	
6	10	879342154	adventure-works\michael0	0x5AE178	4	Research and Development	1984-11-30	M	2009-05-03	1	16	64	1	EAA43680-59747955-E39056F1-4F46DECA	
7	11	974026903	adventure-works\oviduo0	0x5AE3	3	Senior Tool Designer	1978-01-17	S	M	2010-12-05	0	7	23	1	F68C7C19-1D955171-4F46DECA
8	12	480168528	adventure-works\thierry0	0x5AE358	4	Tool Designer	1959-07-29	M	M	2007-12-11	0	9	24	1	1D955171-4F46DECA
Person															
BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo	Demographics	rowguid	ModifiedDate			
1	1	EM	0	NULL	Ken	J	Sánchez	NULL	0	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	92C4279F-1207-48A3-8448-4636514EB7E2	2009-01-07 00:00		
2	14	EM	0	NULL	Michael	I	Sullivan	NULL	2	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	9A7501DE-5CAF-4700-AB07-CC81102BB696	2010-12-23 00:00		
3	15	EM	0	NULL	Sharon	B	Salavaria	NULL	2	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	BEB463CB-13F1-4B76-A3DE-FE9AC283A9...	2011-01-11 00:00		
4	30	EM	0	NULL	Britta	L	Simon	NULL	0	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	5090C9A8-E39E-4A4D-9133-14E7CF079988	2009-01-22 00:00		
5	31	EM	0	NULL	Margie	W	Shoop	NULL	2	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	5EA8A3DA-7BEA-4ADC-85F5-C934E2033825	2008-12-28 00:00		
6	33	EM	0	NULL	Annik	O	Stahl	NULL	0	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	BFA2BDC8-208A-447B-844E-72A215E9E134	2008-12-10 00:00		
7	56	EM	0	NULL	Denise	H	Smith	NULL	0	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	77F4AC33-E165-49D7-BC6A-F3C2DC069C...	2009-01-29 00:00		
8	72	EM	0	NULL	Steven	T	Selkoff	NULL	2	NULL	<IndividualSurvey.xmlns="http://schemas.microsoft.com/SQLServer/ReportModel/ReportModel" />	7455FEFF-9238-4C29-819F-A4DF69D8C0F8	2008-11-24 00:00		
Index															
improvement_measure	create_index_statement	user_seeks	user_scans	schema_name	table_name										
1	309.88 CREATE INDEX [IX_SalesOrderDetail_ProductID] ON [SalesOrderDetail]	276	0	Sales	SalesOrderDetail										

5. Veri Temizleme ve ETL Süreçleri Tasarımı

SQL Server ortamında kapsamlı bir ETL (Extract, Transform, Load) süreci oluşturmaktadır. AdventureWorks2 veritabanından veri çekip temizleyerek yeni bir AdventureWorksCleaned veritabanına aktarmaktadır.

1. Hazırlık İşlemleri

```
CREATE DATABASE AdventureWorksCleaned;
GO
USE AdventureWorksCleaned;
GO
-- Create schema for our cleaned data
CREATE SCHEMA cleaned;
GO
```

Bu bölüm, temizlenmiş veriler için yeni bir veritabanı ve şema oluşturur.

2. Veri Çıkarma ve Temizleme (Extract & Transform)

2.1. Müşteri Verilerinin Temizlenmesi

```
SELECT
    c.CustomerID,
    ISNULL(p.Title, 'N/A') AS Title,
    ISNULL(p.FirstName, 'Unknown') AS FirstName,
    ISNULL(p.MiddleName, '') AS MiddleName,
    ISNULL(p.LastName, 'Unknown') AS LastName,
    -- Standardize phone numbers
    CASE
        WHEN pp.PhoneNumber LIKE '+%' THEN pp.PhoneNumber
        WHEN pp.PhoneNumber IS NULL THEN 'N/A'
        ELSE '+1' + REPLACE(REPLACE(REPLACE(REPLACE(pp.PhoneNumber, ' ', ''), '- ', ''), '(', ''), ')', ')')
    END AS PhoneNumber,
    -- [diğer alanlar ve dönüşümler]
    INTO cleaned.Customers
FROM AdventureWorks2019.Sales.Customer c
JOIN AdventureWorks2019.Person.Person p ON c.PersonID = p.BusinessEntityID
-- [diğer tablo birleştirmeleri]
```

Bu bölüm müşteri verilerini temizler: ISNULL fonksiyonuyla eksik değerler için varsayılan değerler atanır (ISNULL fonksiyonu). Telefon numaraları, e-posta adresleri, adres bilgileri standartlaştırılır.

2.2. Ürün Verilerinin Temizlenmesi

SELECT

```
p.ProductID,  
p.Name AS ProductName,  
ISNULL(p.ProductNumber, 'N/A') AS ProductNumber,
```

CASE

```
    WHEN p.MakeFlag = 1 THEN 'Manufactured'  
    WHEN p.MakeFlag = 0 THEN 'Purchased'  
    ELSE 'Unknown'
```

END AS ProductionType,

-- [diğer alanlar ve dönüşümler]

INTO cleaned.Products

FROM AdventureWorks2019.Production.Product p

LEFT JOIN AdventureWorks2019.Production.ProductSubcategory ssc ON p.ProductSubcategoryID =
ssc.ProductSubcategoryID

LEFT JOIN AdventureWorks2019.Production.ProductCategory sc ON ssc.ProductCategoryID =
sc.ProductCategoryID;

Bu bölüm ürün verilerini temizler: Eksik değerler için 'N/A' gibi varsayılan değerler atanır, boolean değerler Manufactured/Purchased olarak atanır,

2.3. Satış Verilerinin Temizlenmesi

SELECT

```
soh.SalesOrderID,  
soh.SalesOrderNumber,  
CONVERT(date, soh.OrderDate) AS OrderDate,  
-- [diğer alanlar]
```

CASE soh.Status

```
    WHEN 1 THEN 'In process'  
    WHEN 2 THEN 'Approved'
```

-- [diğer durum değerleri]

END AS StatusDescription,

-- [diğer alanlar]

INTO cleaned.SalesOrderHeaders

FROM AdventureWorks2019.Sales.SalesOrderHeader soh

```
LEFT JOIN AdventureWorks2019.Sales.CreditCard cc ON soh.CreditCardID = cc.CreditCardID;
```

Bu bölüm satış verilerini temizler: Tarih değerleri belirli formata dönüştürülür, durum kodları (Status) açıklayıcı metinlere dönüştürülür ve eksik değerler için varsayılanlar atanır

```
-- Sales order details
```

```
SELECT
```

```
    sod.SalesOrderID,  
    sod.SalesOrderDetailID,  
    -- [diğer alanlar]
```

```
INTO cleaned.SalesOrderDetails
```

```
FROM AdventureWorks2019.Sales.SalesOrderDetail sod;
```

3. Veri Kalitesi Raporları

3.1. Eksik Veri Raporu

```
CREATE VIEW cleaned.MissingDataReport AS
```

```
SELECT
```

```
    'Customers' AS TableName,  
    COUNT(CASE WHEN FirstName = 'Unknown' THEN 1 END) AS MissingFirstName,  
    COUNT(CASE WHEN LastName = 'Unknown' THEN 1 END) AS MissingLastName,  
    -- [diğer eksik veri metrikleri]
```

```
FROM cleaned.Customers
```

```
UNION ALL
```

```
SELECT
```

```
    'Products' AS TableName,  
    -- [ürünlerle ilgili eksik veri metrikleri]
```

```
FROM cleaned.Products;
```

Bu görünüm:

Eksik müşteri bilgilerini (isim, soyisim, telefon, vb.) raporlar ve eksik ürün bilgilerini (ürün numarası, renk, boyut, kategori) raporlar

3.2. Veri Standardizasyon Raporu

```

CREATE VIEW cleaned.StandardizationReport AS
SELECT
    'Phone Numbers' AS StandardizationArea,
    COUNT(*) AS TotalRecords,
    SUM(CASE WHEN PhoneNumber LIKE '+1%' THEN 1 ELSE 0 END) AS StandardizedRecords,
    CAST(SUM(CASE WHEN PhoneNumber LIKE '+1%' THEN 1 ELSE 0 END) AS FLOAT) / COUNT(*) *
100 AS PercentStandardized
FROM cleaned.Customers
UNION ALL
SELECT
    'Email Addresses' AS StandardizationArea,
    -- [e-posta standardizasyon metrikleri]
FROM cleaned.Customers;

```

Telefon numaralarının, e-posta adreslerinin ve veri dönüşümünün başarısını yüzde olarak rapor eder ve ölçer.

3.3. Veri Tutarlılık Raporu

```

CREATE VIEW cleaned.PriceConsistencyReport AS
SELECT
    p.ProductID,
    p.ProductName,
    p.ListPrice,
    AVG(sod.UnitPrice) AS AvgSellingPrice,
    CASE
        WHEN AVG(sod.UnitPrice) < 0.9 * p.ListPrice THEN 'Sold below 90% of list'
        WHEN AVG(sod.UnitPrice) > 1.1 * p.ListPrice THEN 'Sold above 110% of list'
        ELSE 'Within normal range'
    END AS PriceConsistencyStatus
FROM cleaned.Products p
JOIN cleaned.SalesOrderDetails sod ON p.ProductID = sod.ProductID
GROUP BY p.ProductID, p.ProductName, p.ListPrice;

```

Bu görünüm: Liste fiyatları ile gerçek satış fiyatları arasındaki tutarlılığı analiz eder, Normalden daha düşük veya yüksek fiyatla satılan ürünleri belirler.

4. Loglama ve İzleme

```

CREATE TABLE cleaned.ETLLog (
    LogID INT IDENTITY(1,1) PRIMARY KEY,

```



```
ProcessName VARCHAR(100),  
StartTime DATETIME,  
EndTime DATETIME,  
Status VARCHAR(20),  
RowsProcessed INT,  
ErrorMessage VARCHAR(MAX)
```

Bu tablo:

- 1) Her ETL işleminin başlangıç ve bitiş zamanını kaydeder
- 2) İşlem durumunu (Running, Completed, Failed) izler
- 3) İşlenen satır sayısını kaydeder
- 4) Hata mesajlarını saklar

5. ETL Sürecinin Çalıştırılması

```
EXEC cleaned.RunETLProcess;
```

```
SELECT * FROM cleaned.MissingDataReport;
```

```
SELECT * FROM cleaned.StandardizationReport;
```

```
SELECT * FROM cleaned.PriceConsistencyReport;
```

```
SELECT * FROM cleaned.ETLLog ORDER BY LogID DESC;
```

ETL sürecini başlatır ve veri kalitesi raporlarını görüntüler.

6. Programın Çıktısı

100 %

Messages

Result	
1	ETL Process Completed Successfully

	TableName	MissingFirstName	MissingLastName	MissingPhone	GeneratedEmail	MissingAddress
1	Customers	0	0	0	0	635
2	Products	0	248	293	209	0

	StandardizationArea	TotalRecords	StandardizedRecords	PercentStandardized
1	Phone Numbers	19143	19143	100
2	Email Addresses	19143	19143	100

	ProductID	ProductName	ListPrice	AvgSellingPrice	PriceConsistencyStatus
67	783	Mountain-200 Black, 42	2294,99	1776,3198	Sold below 90% of list
68	784	Mountain-200 Black, 46	2294,99	1836,2179	Sold below 90% of list
69	785	Mountain-300 Black, 38	1079,99	647,7976	Sold below 90% of list
70	786	Mountain-300 Black, 40	1079,99	647,7034	Sold below 90% of list
71	787	Mountain-300 Black, 44	1079,99	647,994	Sold below 90% of list
72	788	Mountain-300 Black, 48	1079,99	647,8971	Sold below 90% of list
73	789	Road-250 Red, 44	2443,35	1852,0458	Sold below 90% of list
74	790	Road-250 Red, 48	2443,35	1886,6291	Sold below 90% of list

	LogID	ProcessName	StartTime	EndTime	Status	RowsProcessed	ErrorMessage
1	1	Full ETL Process	2025-04-24 23:54:48.810	2025-04-24 23:54:51.367	Completed	172429	NULL

6. Veritabanı Yükseltme ve Sürüm Yönetimi

AdventureWorks veritabanının yükseltilmesi ve sürüm yönetimi için geliştirilen SQL kodunu analiz eder. Kod, aşağıdaki temel işlevleri yerine getirir. Kod dört ana bölümden oluşmaktadır:

1. Temizleme ve Hazırlık İşlemleri

```
IF EXISTS (SELECT * FROM sys.triggers WHERE name = 'tr_TrackSchemaChanges' AND
parent_class = 0)
BEGIN
    DROP TRIGGER tr_TrackSchemaChanges ON DATABASE;
END
GO

IF OBJECT_ID('dbo.UpgradeSteps', 'U') IS NOT NULL DROP TABLE dbo.UpgradeSteps;
IF OBJECT_ID('dbo.SchemaChanges', 'U') IS NOT NULL DROP TABLE dbo.SchemaChanges;
IF OBJECT_ID('dbo.DatabaseVersions', 'U') IS NOT NULL DROP TABLE dbo.DatabaseVersions;
IF OBJECT_ID('dbo.UpgradeTests', 'U') IS NOT NULL DROP TABLE dbo.UpgradeTests;
IF OBJECT_ID('dbo.RollbackUpgrade', 'P') IS NOT NULL DROP PROCEDURE dbo.RollbackUpgrade;
IF OBJECT_ID('dbo.ExecuteUpgrade', 'P') IS NOT NULL DROP PROCEDURE dbo.ExecuteUpgrade;
```

Yükseltme öncesi temiz bir başlangıç sağlar. Eski trigger, tablo ve prosedürleri kaldırarak çakışmaları önler. Bu temiz başlangıç yaklaşımı, yükseltme sürecinde hataları önler

2. Veritabanı Yükseltme Planı

```
IF OBJECT_ID('dbo.UpgradeSteps', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.UpgradeSteps (
        StepID INT PRIMARY KEY,
        StepName VARCHAR(100) NOT NULL,
        Description VARCHAR(500),
        ExecutionOrder INT NOT NULL,
        SQLScript NVARCHAR(MAX) NOT NULL,
        RollbackScript NVARCHAR(MAX),
        Status VARCHAR(20) DEFAULT 'Pending'
    );
END

DELETE FROM dbo.UpgradeSteps;
```

```

INSERT INTO dbo.UpgradeSteps VALUES
(1, 'Müşteri Veri Taşıma', 'SalesLT.Customer -> Sales.Customer', 1,
' INSERT INTO AdventureWorks.Sales.Customer...',
' DELETE FROM AdventureWorks.Sales.Customer', 'Pending'),
-- [diğer yükseltme adımları]

```

Bu bölüm, Yükseltme adımlarını planlı ve izlenebilir hale getirir.Örnekte AdventureWorksLT veritabanından AdventureWorks veritabanına veri taşıma adımları tanımlanmıştır. Yükseltme SQL'i (örneğin, veri taşıma), Rollback SQL'i (hata durumunda geri almak için) ve Durum takibi (Pending, Completed, Failed) bulunmaktadır.

3. Sürüm Yönetimi

```

IF OBJECT_ID('dbo.SchemaChanges', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.SchemaChanges (
        ChangeID INT IDENTITY(1,1) PRIMARY KEY,
        ChangeDate DATETIME DEFAULT GETDATE(),
        ChangeType VARCHAR(50),
        ObjectName VARCHAR(100),
        SQLScript NVARCHAR(MAX),
        ExecutedBy VARCHAR(100) DEFAULT SUSER_SNAME()
    );
END

-- DDL Trigger ile değişiklikleri kaydetme
CREATE TRIGGER tr_TrackSchemaChanges
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE
AS
BEGIN
    DECLARE @EventData XML = EVENTDATA();

    INSERT INTO dbo.SchemaChanges (ChangeType, ObjectName, SQLScript)
    VALUES (
        @EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'VARCHAR(50)'),
        @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]', 'VARCHAR(100)'),
        @EventData.value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]', 'NVARCHAR(MAX)')
    )

```

```

);
END;

-- Sürüm bilgisi tablosu
IF OBJECT_ID('dbo.DatabaseVersions', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.DatabaseVersions (
        VersionID INT PRIMARY KEY,
        VersionName VARCHAR(50) NOT NULL,
        ReleaseDate DATE NOT NULL,
        AppliedDate DATETIME,
        Notes VARCHAR(500)
    );
END

```

Bu bölüm, sürüm yönetimi mekanizmalarını kurar:

- 1) SchemaChanges tablosu şema değişikliklerini izlemek için kullanılır
- 2) DDL Trigger (tr_TrackSchemaChanges) oluşturulur:
 - a. CREATE, ALTER ve DROP işlemlerini dinler
 - b. Tablo ve prosedür değişikliklerini yakalar
 - c. Değişiklikleri SchemaChanges tablosuna kaydeder
 - d. EVENTDATA() fonksiyonu ile değişiklik detaylarını XML formatında alır
- 3) DatabaseVersions tablosu veritabanı sürümlerini izler

4. Test ve Geri Dönüş Planı

```

IF OBJECT_ID('dbo.UpgradeTests', 'U') IS NULL
BEGIN
    CREATE TABLE dbo.UpgradeTests (

```

```

        TestID INT PRIMARY KEY,
        TestName VARCHAR(100) NOT NULL,
        TestQuery NVARCHAR(MAX) NOT NULL,
        ExpectedResult VARCHAR(500) NOT NULL,
        ActualResult VARCHAR(500),
        TestStatus VARCHAR(20) DEFAULT 'NotRun'
    );
END

-- Temel test senaryoları (önce temizle)
DELETE FROM dbo.UpgradeTests;
INSERT INTO dbo.UpgradeTests VALUES
(1, 'Müşteri Sayısı Kontrolü',
 'SELECT COUNT(*) FROM AdventureWorksLT.SalesLT.Customer',
 'SELECT COUNT(*) FROM AdventureWorks.Sales.Customer',
 NULL, 'NotRun'),
-- [diğer test senaryoları]

-- Geri dönüş prosedürü
CREATE PROCEDURE dbo.RollBackUpgrade
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Yükseltme adımlarının ters sırada geri alınması
        DECLARE @RollbackSQL NVARCHAR(MAX) = '';

        SELECT @RollbackSQL = @RollbackSQL + RollbackScript + '; '
        FROM dbo.UpgradeSteps
        WHERE Status = 'Completed'
        ORDER BY ExecutionOrder DESC;

        -- [geri dönüş işlemleri]
    END TRY
    BEGIN CATCH
        -- [hata yönetimi]
    END CATCH;

```

END;

Bu bölüm, yükseltme sonrası test ve geri dönüş mekanizmalarını kurar: UpgradeTests tablosu test senaryolarını tanımlar. Her test için:

- 1) Test sorgusu
- 2) Beklenen sonuç
- 3) Gerçek sonuç (yürütme sonrası doldurulacak)
- 4) Test durumu

RollbackUpgrade saklı prosedürü geri dönüş işlemleri için:

- 1) İşlem (transaction) yönetimi içerir
- 2) Tamamlanan adımları ters sırada geri alır
- 3) Hata yönetimi mekanizması içerir

5. Yükseltmeyi Çalıştıran Ana Prosedür

```
CREATE PROCEDURE dbo.ExecuteUpgrade
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Yükseltme adımlarını çalıştır
        DECLARE @StepID INT, @SQL NVARCHAR(MAX);
        DECLARE @StepCursorStatus INT;

        DECLARE step_cursor CURSOR LOCAL FOR
        SELECT StepID, SQLScript
        FROM dbo.UpgradeSteps
        WHERE Status = 'Pending'
        ORDER BY ExecutionOrder;

        -- [yükseltme adımlarını çalıştırma kodu]
        -- Testleri çalıştır
        -- [test çalıştırma kodu]
        -- Sürüm bilgisini güncelle
        -- [sürüm güncelleme kodu]

        COMMIT TRANSACTION;

        SELECT 'Yükseltme başarıyla tamamlandı' AS Result;
    END TRY
```

```

BEGIN CATCH
    -- [hata yönetimi]

END CATCH;

END;

```

Bu bölüm, yükseltme süreci için ana prosedürü oluşturur: Adımları sırayla çalıştırır (Cursor kullanarak), testleri otomatik çalıştırır, hata durumunda rollback yapar ve sürüm geçmişini günceller.

6. Programın Çıktısı

100 %						
Results Messages						
1 Yükseltme başarıyla tamamlandı						
	TestID	TestName	TestQuery	ExpectedResult	ActualResult	TestStatus
1	1	Müşteri Sayısı Kontrolü	SELECT COUNT(*) FROM AdventureWorksLT.SalesLT.Cu...	SELECT COUNT(*) FROM AdventureWorks.Sales.Customer	847	Failed
2	2	Ürün Sayısı Kontrolü	SELECT COUNT(*) FROM AdventureWorksLT.SalesLT.Pr...	SELECT COUNT(*) FROM AdventureWorks.Production.Pr...	295	Failed
Result						
1 Geri dönüş başarıyla tamamlandı						
	ChangeID	ChangeDate	ChangeType	ObjectName	SQLScript	ExecutedBy
64	64	2025-04-25 00:15:23.103	ALTER_TABLE	SalesOrderDetail	ALTER TABLE [Sales].[SalesOrderDetail] NOCHECK CO...	LAPTOP-80SSB8B8Giartun
65	65	2025-04-25 00:15:23.110	ALTER_TABLE	EmailAddress	ALTER TABLE [Person].[EmailAddress] NOCHECK CON...	LAPTOP-80SSB8B8Giartun
66	66	2025-04-25 00:15:23.110	ALTER_TABLE	DatabaseVersions	ALTER TABLE [dbo].[DatabaseVersions] NOCHECK CO...	LAPTOP-80SSB8B8Giartun
67	67	2025-04-25 00:15:23.117	ALTER_TABLE	Employee	ALTER TABLE [HumanResources].[Employee] NOCHECK...	LAPTOP-80SSB8B8Giartun
68	68	2025-04-25 00:15:23.120	ALTER_TABLE	UpgradeTests	ALTER TABLE [dbo].[UpgradeTests] NOCHECK CONST...	LAPTOP-80SSB8B8Giartun
69	69	2025-04-25 00:15:23.127	ALTER_TABLE	SalesOrderHead...	ALTER TABLE [Sales].[SalesOrderHeader] NOCHECK C...	LAPTOP-80SSB8B8Giartun
70	70	2025-04-25 00:15:23.137	ALTER_TABLE	EmployeeDepart...	ALTER TABLE [HumanResources].[EmployeeDepartme...	LAPTOP-80SSB8B8Giartun
71	71	2025-04-25 00:15:23.143	ALTER_TABLE	EmployeePayHi...	ALTER TABLE [HumanResources].[EmployeePayHistory...	LAPTOP-80SSB8B8Giartun

2. Veritabanı Yedekleme ve Felaketten Kurtarma Planı

AdventureWorks veritabanı üzerinde gerçekleştirilen kapsamlı yedekleme ve felaketten kurtarma stratejilerini detaylandırmaktadır. SQL Server'ın yedekleme mekanizmaları kullanılarak farklı yedekleme türleri, otomatik zamanlama sistemleri ve çeşitli kurtarma senaryoları test edilmiştir.

1. Yedekleme Stratejileri ve Türleri

1.1. Tam Yedekleme (Full Backup)

```
BACKUP DATABASE AdventureWorks  
  
TO DISK = 'C:\Backups\AdventureWorks_Full.bak'  
  
WITH INIT, CHECKSUM;  
  
RESTORE VERIFYONLY  
  
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak'  
  
WITH CHECKSUM;
```

Bu bölümde tam yedekleme stratejisi uygulanmaktadır. Veritabanının tüm verilerini ve yapısını yedekler, INIT parametresi ile mevcut yedekleme dosyasını üzerine yazar ve CHECKSUM ile veri bütünlüğünü doğrular. RESTORE VERIFYONLY komutu yedekleme dosyasının bozulmadığını kontrol eder.

1.2. Diferansiyel Yedekleme (Differential Backup)

```
BACKUP DATABASE AdventureWorks  
  
TO DISK = 'C:\Backups\AdventureWorks_Diff.bak'  
  
WITH DIFFERENTIAL, CHECKSUM;  
  
RESTORE VERIFYONLY  
  
FROM DISK = 'C:\Backups\AdventureWorks_Diff.bak'  
  
WITH CHECKSUM;
```

Diferansiyel yedekleme son tam yedeklemeden sonra değişen verileri yedekler.

1.3. Transaction Log Yedekleme

```
ALTER DATABASE AdventureWorks SET RECOVERY FULL;  
  
GO
```

```
BACKUP LOG AdventureWorks

TO DISK = 'C:\Backups\AdventureWorks_Log.trn'

WITH CHECKSUM;

RESTORE VERIFYONLY

FROM DISK = 'C:\Backups\AdventureWorks_Log.trn'

WITH CHECKSUM;
```

Transaction log yedekleme için veritabanı FULL kurtarma moduna alınır. Log yedeklemeleri point-in-time kurtarma imkanı sağlar ve veri kaybını minimize eder.

2. Otomatik Yedekleme Zamanlayıcısı

2.1. SQL Server Agent Job Oluşturma

```
USE msdb;

EXEC sp_add_job

    @job_name = N' AdventureWorks_FullBackupJob' ;
```

SQL Server Agent kullanılarak otomatik yedekleme işi oluşturulur. msdb sistem veritabanında job bilgileri saklanır.

2.2. Job Adımı Tanımlama

```
EXEC sp_add_jobstep

    @job_name = N' AdventureWorks_FullBackupJob' ,

    @step_name = N' FullBackupStep' ,

    @subsystem = N' TSQL' ,

    @command = N' BACKUP DATABASE AdventureWorks TO DISK =
    '' C:\Backups\AdventureWorks_Full_Scheduled.bak'' WITH INIT;' ,

    @retry_attempts = 1,

    @retry_interval = 5;
```

Job adımı TSQL subsystem ile tanımlanır. Hata durumunda 1 kez tekrar deneme ve 5 dakika bekleme süresi ayarlanır.

2.3. Zamanlama Konfigürasyonu

```
EXEC sp_add_schedule
```

```
    @schedule_name = N'Daily2AM',
```

```
    @freq_type = 4, -- daily
```

```
    @freq_interval = 1,
```

```
    @active_start_time = 020000;
```

```
EXEC sp_attach_schedule
```

```
    @job_name = N'AdventureWorks_FullBackupJob',
```

```
    @schedule_id = 1;
```

```
EXEC sp_add_jobserver
```

```
    @job_name = N'AdventureWorks_FullBackupJob' ;
```

Bu bölüm otomatik zamanlama sistemini kurar. Job, her gün saat 02:00'da çalışacak şekilde ayarlanmıştır.

3. Felaketten Kurtarma Senaryoları

3.1. Tam Kurtarma İşlemi

```
USE master;
```

```
ALTER DATABASE AdventureWorks SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```

```
DROP DATABASE AdventureWorks;
```

```
RESTORE DATABASE AdventureWorks
```

```
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak'
```

```
WITH NORECOVERY, REPLACE, CHECKSUM;
```

```
RESTORE DATABASE AdventureWorks
```

```
FROM DISK = 'C:\Backups\AdventureWorks_Diff.bak'
```

```
WITH RECOVERY, CHECKSUM;
```

Bu bölüm felaketten kurtarma sürecini simüle eder: Mevcut veritabanı single user moduna alınır ve silinir, tam yedeklemeden NORECOVERY ile geri yüklenir, diferansiyel yedekleme RECOVERY ile tamamlanır ve CHECKSUM ile veri bütünlüğü doğrulanır.

3.2. Veri Kaybı Simülasyonu ve Test

```
USE AdventureWorks;
```

```
SELECT TOP 50 * FROM HumanResources.Department;
```

```
DELETE FROM HumanResources.Department WHERE GroupName = 'Sales and Marketing';
```

```
SELECT * FROM HumanResources.Department WHERE GroupName = 'Sales and Marketing';
```

Kasıtlı veri silme işlemi gerçekleştirilir: Silmeden önce veriler görüntülenir, kritik veriler HumanResources.Department tablosundan silinir ve silme işleminin başarısı doğrulanır.

3.4. Point-in-Time Kurtarma

```
USE master;
```

```
ALTER DATABASE AdventureWorks SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
```

```
RESTORE DATABASE AdventureWorks
```

```
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak'
```

```
WITH REPLACE, CHECKSUM;
```

```
USE AdventureWorks;
```

```
SELECT * FROM HumanResources.Department WHERE Name = 'Sales and Marketing';
```

Silinen veriler tam yedeklemeden geri yüklenir ve kurtarma işleminin başarısı kontrol edilir.

4. Test Yedekleme Senaryoları

4.1. Yedekleme Doğrulama Testleri

```
RESTORE VERIFYONLY
```

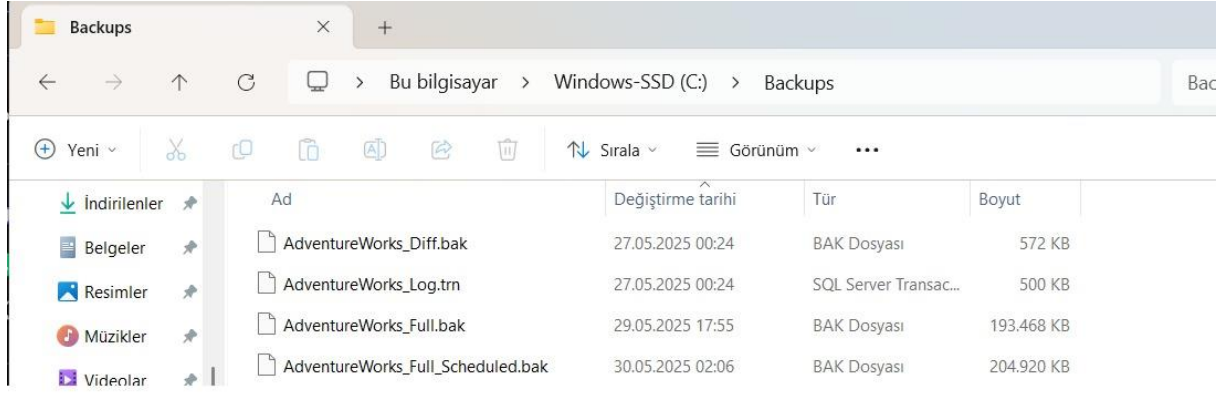
```
FROM DISK = 'C:\Backups\AdventureWorks_Full.bak';
```

```
RESTORE VERIFYONLY
```

```
FROM DISK = 'C:\Backups\AdventureWorks_Log.trn';
```

Bu bölüm yedekleme dosyalarının bütünlüğünü test eder: Tam yedekleme dosyasının korusiyonu kontrol edilir, transaction log yedekleme dosyasının geçerliliği doğrulanır ve RESTORE VERIFYONLY gerçek geri yükleme yapmadan dosyayı test eder.

5. Program Sonucu Oluşan Backup Databasele



3. Veritabanı Güvenliği ve Erişim Kontrolü

AdventureWorks veritabanı üzerinde gerçekleştirilen kapsamlı güvenlik uygulamalarını ve erişim kontrolü mekanizmalarını detaylandırmaktadır. SQL Server'ın güvenlik özellikleri kullanılarak kullanıcı yetkilendirme, veri şifreleme, SQL injection koruması ve audit sistemleri test edilmiştir.

1. Erişim Yönetimi ve Kullanıcı Yetkilendirme

1.1. SQL Server Authentication ile Login Oluşturma

```
CREATE LOGIN AdventureUser WITH PASSWORD = 'StrongP@ssword123!';
```

```
USE AdventureWorks;
```

```
CREATE USER AdventureUser FOR LOGIN AdventureUser;
```

```
GRANT SELECT ON SCHEMA::Sales TO AdventureUser;
```

Bu bölümde SQL Server Authentication kullanılarak güvenli login oluşturulur.

1.2. Windows Authentication ile Kullanıcı Yönetimi

```
USE AdventureWorks;
```

```
CREATE USER [laptop-80ssb8bg¥artun] FOR LOGIN [laptop-80ssb8bg¥artun];
```

```
GRANT SELECT, INSERT, UPDATE ON SCHEMA::HumanResources TO [laptop-80ssb8bg¥artun];
```

Windows Authentication ile domain/local kullanıcı entegrasyonu sağlanır. HumanResources şemasında SELECT, INSERT ve UPDATE yetkileri verilir, ancak DELETE yetkisi kısıtlanır.

1.3. Kullanıcı İmpersonation ve Yetki Testleri

AdventureUser Yetki Testi:

```
EXECUTE AS USER = 'AdventureUser';
```

```
SELECT TOP 5 * FROM Sales.Currency;
```

```
BEGIN TRY
```

```
    INSERT INTO Sales.Currency(CurrencyCode, [Name], ModifiedDate)
```

```
    VALUES ('AGF', 'Artun', 2010-04-30);
```

```
END TRY
```

```
BEGIN CATCH
```

```
    PRINT 'INSERT işlemi başarısız. Yetkiniz yok.';
```

```
END CATCH;
```

EXECUTE AS ile kullanıcı impersonation gerçekleştirilir. SELECT işlemi başarılı (yetki var), INSERT işlemi başarısız (yetki yok) ve yetki kontrolü TRY-CATCH ile test edilir.

Windows User Yetki Testi:

```
EXECUTE AS USER = 'laptop-80ssb8bg¥artun';
```

```
BEGIN TRY
```

```
    INSERT INTO HumanResources.Department ([Name], GroupName, ModifiedDate)
```

```
    VALUES ('TestDept', 'TestGroup', GETDATE());
```

```
    PRINT 'INSERT işlemi başarılı.';
```

```
END TRY
```

```
BEGIN CATCH
```

```
    PRINT 'INSERT işlemi başarısız.';
```

```
END CATCH;
```

```
BEGIN TRY
```

```
    DELETE FROM HumanResources.Department
```

```
    WHERE [Name] = 'UpdatedDept';
```

```
    PRINT 'DELETE işlemi başarılı.';
```

```
END TRY
```

```
BEGIN CATCH
```

```
    PRINT 'DELETE işlemi başarısız. Yetkiniz yok.' ;
```

```
END CATCH;
```

Bu test senaryosunda Windows kullanıcısının yetkileri doğrulanır. INSERT ve UPDATE işlemleri başarılı (yetki var), DELETE işlemi başarısız (yetki kısıtlı) ve REVERT ile orijinal kontekste dönüş sağlanır.

2. Veri Şifreleme (Data Encryption)

2.1. Master Key ve Certificate Oluşturma

```
USE AdventureWorks;
```

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'StrongMasterKeyP@ssword!';
```

```
CREATE CERTIFICATE AdventureWorksCert
```

```
WITH SUBJECT = 'Certificate for Column Encryption';
```

Veritabanı seviyesinde Master Key oluşturulur ve şifreleme için sertifika tanımlanır. Master Key tüm şifreleme hiyerarşisinin temelini oluşturur.

2.2. Symmetric Key ile Sütun Şifreleme

```
CREATE SYMMETRIC KEY AdventureSymKey
```

```
WITH ALGORITHM = AES_256
```

```
ENCRYPTION BY CERTIFICATE AdventureWorksCert;
```

```
CREATE TABLE dbo.EncryptedTest (
```

```
    Id INT IDENTITY PRIMARY KEY,
```

```
    Name NVARCHAR(100),
```

```
    EncryptedData VARBINARY(MAX)
```

```
);
```

AES_256 algoritması ile güçlü symmetric key oluşturulur. Şifrelenmiş verileri saklayacak test tablosu hazırlanır ve EncryptedData sütunu VARBINARY(MAX) tipinde tanımlanır.

2.3. Veri Şifreleme ve Deşifreleme İşlemleri

```
OPEN SYMMETRIC KEY AdventureSymKey
```

```

DECRYPTION BY CERTIFICATE AdventureWorksCert;

INSERT INTO dbo.EncryptedTest (Name, EncryptedData)

VALUES (

    'Müşteri A',

    ENCRYPTBYKEY(KEY_GUID('AdventureSymKey'), 'Gizli Bilgi 123')

);

CLOSE SYMMETRIC KEY AdventureSymKey;

```

Symmetric key açılır ve ENCRYPTBYKEY fonksiyonu ile veri şifrelenir. İşlem sonrası güvenlik için key kapatılır.

```

OPEN SYMMETRIC KEY AdventureSymKey

DECRYPTION BY CERTIFICATE AdventureWorksCert;

SELECT

    Id,

    Name,

    CONVERT(NVARCHAR, DECRYPTBYKEY(EncryptedData)) AS DecryptedData

FROM dbo.EncryptedTest;

CLOSE SYMMETRIC KEY AdventureSymKey;

```

Deşifreleme işleminde key tekrar açılır, DECRYPTBYKEY fonksiyonu ile veri çözülür ve CONVERT ile okunabilir formata dönüştürülür.

3. SQL Injection Koruması

3.1. Parametrelili Stored Procedure

```

USE AdventureWorks;

GO

CREATE PROCEDURE usp_GetEmployeeByName

    @LastName NVARCHAR(50)

AS

BEGIN

```



```
SET NOCOUNT ON;

SELECT BusinessEntityID, FirstName, LastName

FROM Person.Person

WHERE LastName = @LastName;

END;
```

SQL Injection saldırılarına karşı parametrelili stored procedure kullanılır. @LastName parametresi input validation sağlar, SET NOCOUNT ON performans optimizasyonu yapar ve parameterized query SQL injection'u önler.

3.2. SQL Injection Test Senaryoları

Güvensiz Sorgu (Örnek):

-- RISKLI: SQL Injection'a açık

```
"SELECT * FROM Users WHERE Username = '" + userInput + "'" "
```

Güvenli Sorgu (Uygulanan):

-- GÜVENLİ: Parametrelili approach

```
WHERE LastName = @LastName
```

Bu yaklaşımda malicious input otomatik olarak escape edilir ve SQL komutları çalıştırılmaz.

4. Audit Logları ve İzleme Sistemleri

4.1. Server Audit Oluşturma

```
USE master;

CREATE SERVER AUDIT Audit_AdventureWorks

TO FILE ( FILEPATH = 'C:¥AuditLogs¥' );

ALTER SERVER AUDIT Audit_AdventureWorks

WITH (STATE = ON);
```

Server seviyesinde audit nesnesi oluşturulur. Audit logları dosya sistemine kaydedilir ve C:\AuditLogs\ dizininde saklanır. Audit STATE = ON ile etkinleştirilir.

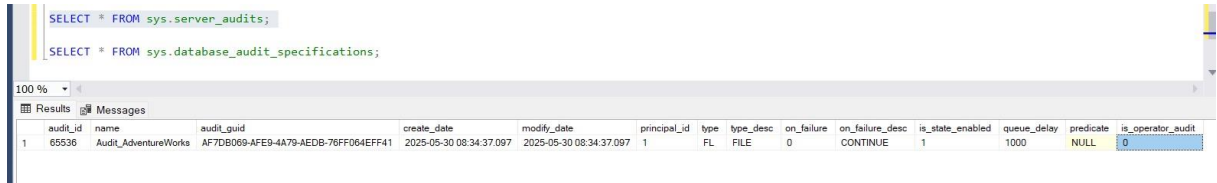
4.2. Database Audit Specification

```
USE AdventureWorks;
```

```
CREATE DATABASE AUDIT SPECIFICATION Audit_ReadWriteAccess
FOR SERVER AUDIT Audit_AdventureWorks
ADD (SELECT ON SCHEMA::Sales BY AdventureUser),
ADD (INSERT ON SCHEMA::Sales BY AdventureUser),
ADD (UPDATE ON SCHEMA::Sales BY AdventureUser)
WITH (STATE = ON);
```

Database seviyesinde audit specification tanımlanır. AdventureUser'ın Sales şemasındaki aktiviteleri izlenir: SELECT, INSERT ve UPDATE işlemleri loglanır ve WITH (STATE = ON) ile aktif hale getirilir.

5. Programın Çıktısı



audit_id	name	audit_guid	create_date	modify_date	principal_id	type	type_desc	on_failure	on_failure_desc	is_state_enabled	queue_delay	predicate	is_operator_audit
1	Audit_AdventureWorks	AF7DB069-AFE9-4A79-AEDB-76FF064EFF41	2025-05-30 08:34:37.097	2025-05-30 08:34:37.097	1	FL	FILE	0	CONTINUE	1	1000	NULL	0

4. Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları

Bu çalışma, iki farklı SQL Server instance'ı üzerinde AdventureWorks2 ve AdventureWorks3 veritabanlarını kullanarak dağıtık yapı ve veri çoğaltmayı kapsamaktadır. SQL Server Replication özelliği kullanılarak verilerin çoğaltılması, yük dengeleme ve failover senaryoları yapılandırılmıştır.

1. Hazırlık İşlemleri

Bu aşamada, veri çoğaltılması için bir SQL Server instance'ı Publisher (yayımcı), diğeri ise Subscriber (abonelik alan) olarak yapılandırılmıştır. Ayrıca dağıtıcı (Distributor) sunucusu tanımlanmıştır. Instance'lar aynı fiziksel makinada farklı olarak yapılandırılmıştır:

- Publisher veritabanı: AdventureWorks2 (LAPTOP-80SSB8BG)
- Subscriber veritabanı: AdventureWorks3 (LAPTOP-80SSB8BG\MSSQLSERVER01)

Dağıtım (Distribution) işlemi GUI üzerinden “New Publication Wizard” ve “New Subscription Wizard” sihirbazları ile gerçekleştirilmiştir.

2. Replikasyonun Yapılandırılması (Publisher, Distributor, Subscriber)

2.1 Dağıtıcı Tanımı

- “Configure Distribution” sihirbazı üzerinden LAPTOP-80SSB8BG sunucusu dağıtıcı olarak tanımlanmıştır.

- SQL Server Agent servisi başlatılarak dağıtım veritabanı (distribution) oluşturulmuştur.

2.2 Yayımcı (Publisher) Yapılandırması

- Veritabanı: AdventureWorks2 (Orjinal AdventureWorks2019.bak)
- Yayın türü: Transactional Publication
- Yayınlanan tablo(lar): Person.Person, Sales.Customer, Production.Product gibi sık kullanılan veri tabloları seçilmiştir.
- Güvenlik ayarlarında Snapshot Agent için Windows kullanıcısı (LAPTOP-80SSB8BG\artun) kullanılmıştır.

2.3 Abonelik (Subscription) Yapılandırması

- Abone sunucu: LAPTOP-80SSB8BG\MSSQLSERVER01
- Hedef veritabanı: AdventureWorks3 (Orjinal AdventureWorksLT2019.bak)
- Abonelik türü: Pull Subscription
- Verilerin Transactional Replication ile anlık olarak eşitlenmesi hedeflenmektedir.

3. İzleme ve Doğrulama

3.1 Replication Monitor Kullanımı

- Replication Monitor ekranı üzerinden yayımcı ve abone sunucular izlenmiştir.
- Veri çoğaltma işlemlerinin zamanlaması, hataları ve bekleme süreleri gözlemlenmiştir.

7. Veritabanı Yedekleme ve Otomasyon Çalışması

SQL Server Agent kullanılarak AdventureWorks veritabanı için tam yedekleme işlemleri otomatikleştirilmiş ve günlük olarak zamanlanmıştır. Yedekleme işlemleri başarıyla tamamlandığında veya başarısız olduğunda, ilgili bildirimler e-posta ile yöneticilere gönderilmektedir. Bu süreç aşağıdaki şekilde yapılandırılmıştır:

1. SQL Server Agent ile Zamanlanmış Yedekleme İşlemi

- Job Adı: FullBackupJob
- Adım Sayısı: 1 (Tam Yedekleme)
- Zamanlama: Her dakikada bir

Jobın çalıştırdığı yedekleme komutu şu şekildedir:

```
BACKUP DATABASE AdventureWorks3
```

```
TO DISK = 'C:\Backups\AdventureWorks3_Full_ Scheduled.bak'
```

```
WITH INIT;
```

Bu komut, belirtilen klasöre .bak uzantılı bir tam yedek dosyası oluşturur.

2. Database Mail Konfigürasyonu

- SMTP Sunucusu: smtp.gmail.com
- Port: 587
- SSL: Aktif
- Kimlik Doğrulama: Gmail uygulama şifresi kullanılarak yapılmıştır.

3. Otomatik Uyarı ve Bildirim

- Uyarı Tipi: Mail bildirimi
- Kapsam:
 - Yedekleme işlemi başarılı olursa → “Başarıyla tamamlandı” mesajı gönderilir.
 - Yedekleme işlemi başarısız olursa → Hata detayı ile birlikte uyarı e-postası gönderilir.
- Alıcı: kişisel Gmail adresi

4. Programın Çıktısı

[The job succeeded.] SQL Server Job System: 'AdventureWorks_FullBackupJob' completed on \\LAPTOP-80SSB8BG\MSSQLSERVER01. Gelen Kutusu x



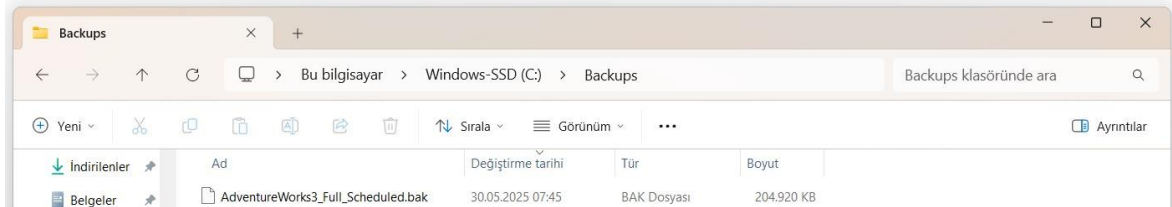
SQL ALERT <artun.9915@gmail.com>

Alıcı: ben

07:45 (0 dakika önce)



JOB RUN: 'AdventureWorks_FullBackupJob' was run on 30.05.2025 at 07:45:00
DURATION: 0 hours, 0 minutes, 4 seconds
STATUS: Succeeded
MESSAGES: The job succeeded. The Job was invoked by Schedule 9 (RunAsSQLAgentServiceStartSchedule). The last step to run was step 1 (FullBackupStep).



```

SELECT
    database_name,
    backup_start_date,
    backup_finish_date,
    backup_size / 1024 / 1024 AS BackupSize_MB,
    physical_device_name AS BackupFile,
    CASE type
        WHEN 'D' THEN 'Full'
        WHEN 'I' THEN 'Differential'
        WHEN 'L' THEN 'Log'
        ELSE type
    END AS BackupType
FROM msdb.dbo.backupset b
JOIN msdb.dbo.backupmediafamily m ON b.media_set_id = m.media_set_id
WHERE backup_start_date >= DATEADD(DAY, -7, GETDATE())
ORDER BY backup_start_date DESC;

```

100 %

Results Messages

	database_name	backup_start_date	backup_finish_date	BackupSize_MB	BackupFile	BackupType
1	AdventureWorks3	2025-05-30 07:51:00.000	2025-05-30 07:51:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
2	AdventureWorks3	2025-05-30 07:50:01.000	2025-05-30 07:50:02.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
3	AdventureWorks3	2025-05-30 07:49:02.000	2025-05-30 07:49:02.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
4	AdventureWorks3	2025-05-30 07:48:00.000	2025-05-30 07:48:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
5	AdventureWorks3	2025-05-30 07:47:01.000	2025-05-30 07:47:02.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
6	AdventureWorks3	2025-05-30 07:46:01.000	2025-05-30 07:46:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
7	AdventureWorks3	2025-05-30 07:45:02.000	2025-05-30 07:45:03.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
8	AdventureWorks3	2025-05-30 07:44:01.000	2025-05-30 07:44:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
9	AdventureWorks3	2025-05-30 07:43:00.000	2025-05-30 07:43:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
10	AdventureWorks3	2025-05-30 07:42:01.000	2025-05-30 07:42:02.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
11	AdventureWorks3	2025-05-30 07:41:01.000	2025-05-30 07:41:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
12	AdventureWorks3	2025-05-30 07:40:11.000	2025-05-30 07:40:12.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
13	AdventureWorks3	2025-05-30 07:40:00.000	2025-05-30 07:40:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full
14	AdventureWorks3	2025-05-30 07:39:01.000	2025-05-30 07:39:01.000	200.08593750000	C:\Backups\AdventureWorks3_Full_Scheduled.bak	Full