

Ankara Üniversitesi

Bilgisayar Mühendisliği Bölümü

BLM4522 Ağ Tabanlı Paralel Dağıtım Sistemleri Dersi Raporu

Berkay Bozpınar – 21290178

Artun Cankar – 21290507

25.04.2025

Github:

<https://github.com/bbberkayyy/blm4522final>

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 Yüğü 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- >50 olanlar yüksek öncelikli 100)	
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

	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-8019-2C0C858619CA	2014-06-30 00:00:00.000
2	75123	121316	NULL	1	879	1	159.00	0.00	159.000000	75A89C6A-C60A-47EA-8A52-B52A9C435B64	2014-06-30 00:00:00.000
3	75123	121315	NULL	1	878	1	21.98	0.00	21.980000	C18B6476-429F-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-BE16-541C59B6E12C	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-AC43-451FE66D1C99	2014-06-30 00:00:00.000
8	75121	121310	NULL	1	921	1	4.99	0.00	4.990000	8B8ADC91-BF1A-4D8C-BFD0-95B587BC2567	2014-06-30 00:00:00.000

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate	MaritalStatus	Gender	HireDate	SalariedFlag	VacationHours	SickLeaveHours	CurrentFlag	rowguid
1	1	295847284	adventure-works/ken0	NULL	NULL	Chief Executive Officer	1969-01-29	S	M	2009-01-14	1	99	69	1	F01251E5-
2	3	509647174	adventure-works/roberto0	0x5AC0	2	Engineering Manager	1974-11-12	M	M	2007-11-11	1	2	21	1	9BBBF82C-
3	4	112457891	adventure-works/rob0	0x5AD6	3	Senior Tool Designer	1974-12-23	S	M	2007-12-05	0	48	80	1	59747955-4
4	6	998320692	adventure-works/jossee0	0x5ADE	3	Design Engineer	1959-03-11	M	M	2008-01-24	1	6	23	1	E39056F1-
5	7	134969118	adventure-works/dylan0	0x5AE1	3	Research and Devel...	1987-02-24	M	M	2009-02-08	1	61	50	1	4F46DECA-
6	10	879342154	adventure-works/micha...	0x5AE178	4	Research and Devel...	1984-11-30	M	M	2009-05-03	1	16	64	1	EAA43680-
7	11	974026903	adventure-works/lovidu0	0x5AE3	3	Senior Tool Designer	1978-01-17	S	M	2010-12-05	0	7	23	1	F68C7C19-
8	12	480168528	adventure-works/thierry0	0x5AE358	4	Tool Designer	1959-07-29	M	M	2007-12-11	0	9	24	1	1D955171-

	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.microso...	92C4279F-1207-48A3-8448-4636514EB7E2	2009-01-07 00:00:00.000
2	14	EM	0	NULL	Michael	I	Sullivan	NULL	2	NULL	IndividualSurvey.xmlns="http://schemas.microso...	9A7501DE-5CAF-4700-AB07-CC81102BB696	2010-12-23 00:00:00.000
3	15	EM	0	NULL	Sharon	B	Salavaria	NULL	2	NULL	IndividualSurvey.xmlns="http://schemas.microso...	BEB6A3CB-13F1-4B76-A3DE-FE9AC283A9...	2011-01-11 00:00:00.000
4	30	EM	0	NULL	Britta	L	Simon	NULL	0	NULL	IndividualSurvey.xmlns="http://schemas.microso...	5090C9A8-E39E-44D0-9133-14E7CF07998B	2009-01-22 00:00:00.000
5	31	EM	0	NULL	Margie	W	Shoop	NULL	2	NULL	IndividualSurvey.xmlns="http://schemas.microso...	5EAB3A3A-7BEA-4ADC-85F5-C934E2033825	2008-12-28 00:00:00.000
6	33	EM	0	NULL	Annik	O	Stahl	NULL	0	NULL	IndividualSurvey.xmlns="http://schemas.microso...	BFA2BDC8-208A-447B-84AE-72A215E9E134	2008-12-10 00:00:00.000
7	56	EM	0	NULL	Denise	H	Smith	NULL	0	NULL	IndividualSurvey.xmlns="http://schemas.microso...	77F4AC33-E165-49D7-BC6A-F3C2DC069C...	2009-01-29 00:00:00.000
8	72	EM	0	NULL	Steven	T	Seikoff	NULL	2	NULL	IndividualSurvey.xmlns="http://schemas.microso...	7455FEFF-9238-4C29-819F-A4DF6508C0F8	2008-11-24 00:00:00.000

	improvement_measure	create_index_statement	user_seeks	user_scans	schema_name	table_name
1	309.88	CREATE INDEX [IX_SalesOrderDetail_ProductID] ON ...	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 %

ResultsMessages

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-80SSB8BGIartun
65	65	2025-04-25 00:15:23.110	ALTER_TABLE	EmailAddress	ALTER TABLE [Person].[EmailAddress] NOCHECK CON...	LAPTOP-80SSB8BGIartun
66	66	2025-04-25 00:15:23.110	ALTER_TABLE	DatabaseVersions	ALTER TABLE [dbo].[DatabaseVersions] NOCHECK CO...	LAPTOP-80SSB8BGIartun
67	67	2025-04-25 00:15:23.117	ALTER_TABLE	Employee	ALTER TABLE [HumanResources].[Employee] NOCHECK...	LAPTOP-80SSB8BGIartun
68	68	2025-04-25 00:15:23.120	ALTER_TABLE	UpgradeTests	ALTER TABLE [dbo].[UpgradeTests] NOCHECK CONST...	LAPTOP-80SSB8BGIartun
69	69	2025-04-25 00:15:23.127	ALTER_TABLE	SalesOrderHead...	ALTER TABLE [Sales].[SalesOrderHeader] NOCHECK C...	LAPTOP-80SSB8BGIartun
70	70	2025-04-25 00:15:23.137	ALTER_TABLE	EmployeeDepart...	ALTER TABLE [HumanResources].[EmployeeDepartme...	LAPTOP-80SSB8BGIartun
71	71	2025-04-25 00:15:23.143	ALTER_TABLE	EmployeePayHi...	ALTER TABLE [HumanResources].[EmployeePayHistory...	LAPTOP-80SSB8BGIartun