

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



BLM 4522
Ağ Tabanlı Paralel Dağıtım Sistemleri

Şahin Berk Çelik
21290630
Github:berk-celik

Ağ Tabanlı Paralel Dağıtım Sistemleri dersinde kullanmak üzere 2024 yılı boyunca orta ölçekli bir organizasyon için tedarik işlemlerini simüle eden bir sentetik veri seti kullandım. Bu veri seti, çeşitli tedarikçilerden ve alıcılardan birden fazla kategorideki (elektronik, mobilya, kırtasiye vb.) satın alımları içerir. 9 sütun, 500 satırdan oluşmaktadır. Küçük bir bölümü aşağıdaki gibidir.

	TransactionID	ItemName	Category	Quantity	UnitPrice	TotalCost	PurchaseDate	Supplier	Buyer
1	TXN001	Desk Chair	Furniture	10	113.15	1131.5	2024-04-19 00:00:00.000	TechMart Inc.	Kelly Joseph
2	TXN002	Stapler	Office Supplies	16	12.62	201.92	2024-07-06 00:00:00.000	CloudSoft Corp.	Kelly Joseph
3	TXN003	Annual Software License	Software	1	5649.34	5649.34	2024-09-10 00:00:00.000	TechMart Inc.	Kelly Joseph
4	TXN004	Notepad	Stationery	13	2.92	37.96	2024-01-21 00:00:00.000	FumiWorks Ltd.	Luis Holland
5	TXN005	Notepad	Stationery	19	1.39	26.41	2024-02-03 00:00:00.000	TechMart Inc.	Cynthia Jenkins
6	TXN006	Printer	Electronics	19	150.94	2867.86	2024-11-28 00:00:00.000	FumiWorks Ltd.	Stephanie Bennett
7	TXN007	Notepad	Stationery	8	2.73	21.84	2024-02-19 00:00:00.000	OfficeSupplies Co.	Todd James
8	TXN008	Notepad	Stationery	4	2.42	9.68	2024-09-11 00:00:00.000	FumiWorks Ltd.	Aaron Hopkins
9	TXN009	Printer Ink	Stationery	13	11.89	154.57	2024-04-12 00:00:00.000	FumiWorks Ltd.	Kevin Adams
10	TXN010	Whiteboard	Furniture	19	100.82	1915.58	2024-12-14 00:00:00.000	FumiWorks Ltd.	Aaron Hopkins

Bu veri setini burada bulabilirsiniz:

<https://www.kaggle.com/datasets/shahriarkabir/company-purchasing-dataset>

1. Veritabanı Yedekleme ve Felaketten Kurtarma Planı

Bu bölümde bir veritabanının yedekleme ve felaketten kurtarma planlarının tasarlanması anlatılacaktır. Sırasıyla Tam, Artımlı, Fark yedeklemeleri, Zamanlayıcılarla yedekleme, Felaketten kurtarma senaryoları ve Test yedekleme senaryoları anlatılacaktır.

1.1. Tam Yedekleme

Burada veritabanının tamamı yedeklenir, komutu şu şekildedir.

```
BACKUP DATABASE BLM4522
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522.bak';
```

1.2. Fark Yedekleme

Son tam yedekten sonra deęiřen deęiřiklikler yedeklenir. komutu řu řekildedir.

```
BACKUP DATABASE BLM4522  
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_diff.bak' WITH DIFFERENTIAL;
```

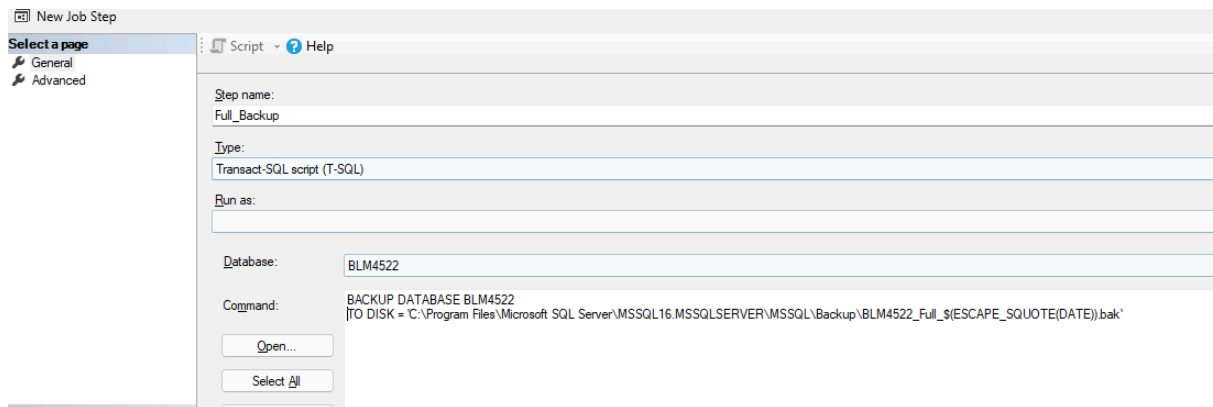
1.3. Artımlı Yedekleme

Son yedekten sonra deęiřen deęiřiklikler yedeklenir. Fark yedeklemeden farkı, fark yedekleme en son yedeklenen **tam** yedekten sonraki deęiřikleri yedekler, artımlı yedekleme ise son yedeklemeden sonraki deęiřiklikleri yedekler. Komutu řu řekildedir.

```
BACKUP LOG BLM4522  
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_log.trn'
```

1.4. Zamanlayıcılarla Yedekleme

Zamanlayıcılarla yedekleme yapabilmek için SQL Server Agent aracılığı ile Job'lar oluşturulur. Job bölümünün step kısmına komutumuz girilir.



New Job Step

Select a page

- General
- Advanced

Script Help

Step name: Full_Backup

Type: Transact-SQL script (T-SQL)

Run as:

Database: BLM4522

Command: BACKUP DATABASE BLM4522
TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_Full_\$(ESCAPE_SQUOTE(DATE)).bak'

Open...

Select All

Schedule kısmında hangi aralıklarla yedekleme yapılacağı belirlenir.

1.5. Felaketten Kurtarma Senaryoları

Bir veritabanındaki önemli bir tablo yanlışlıkla silindiğinde bu felaketi geri almak için tam yedekleme ve transaction log yedeği kullanılır.

a. Veritabanı Yedeğini Geri Yükleme (Full Backup)

Veritabanını tam yedeğinden geri yüklemek için aşağıdaki komut kullanılır.

```
RESTORE DATABASE BLM4522  
FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_full.bak'  
WITH REPLACE;
```

Eğer veritabanındaki bir veri silindiyseniz ve tam yedeği geri yüklediyseniz, ancak bu işlemin ardından yeni veriler de eklenmişse, Artımlı yedeği kullanılarak sadece silinen verileri geri getirebilirsiniz.

```
RESTORE LOG VeritabaniAdi  
FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_log.trn'  
WITH NORECOVERY;
```

1.6. Kaydedilen Yedeklerin Doğruluğunu Test Etme

Aşağıdaki komut sayesinde yedek dosyamızın bütünlüğünü kontrol edebiliriz.

```
RESTORE VERIFYONLY  
FROM DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\Backup\BLM4522_full.bak'
```

2. Veritabanı Yedekleme ve Otomasyon Çalışması

2.1. Veritabanı Yedekleme İşlemini Otomatikleştirmek

SQL Server Agent kullanarak yedekleme işlemini otomatikleştirme 1.4.'de detaylı bir şekilde anlatılmıştır.

2.2. T-SQL Kullanarak Yedekleme Raporları Oluşturma

Aşağıdaki sorgu sayesinde geçmiş yedekleme kayıtlarını listelenir. Bu sorgu ile hangi veritabanının, ne zaman yedeklendiği ve nereye kaydedildiği aşağıdaki şekildeki gibi detaylı olarak listelenir.

```

SELECT
    database_name,
    backup_start_date,
    backup_finish_date,
    physical_device_name,
    type AS backup_type
FROM msdb.dbo.backupset bs
JOIN msdb.dbo.backupmediafamily bmf
    ON bs.media_set_id = bmf.media_set_id
WHERE database_name = 'BLM4522'
ORDER BY backup_finish_date DESC;

```

100 %

Results

Messages

	database_name	backup_start_date	backup_finish_date	physical_device_name	backup_type
1	BLM4522	2025-04-25 00:36:36.000	2025-04-25 00:36:36.000	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	I
2	BLM4522	2025-04-25 00:33:27.000	2025-04-25 00:33:27.000	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	D
3	BLM4522	2025-04-25 00:33:08.000	2025-04-25 00:33:08.000	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	D
4	BLM4522	2025-04-25 00:28:58.000	2025-04-25 00:28:58.000	C:\Program Files\Microsoft SQL Server\MSSQL16.MSS...	D

2.3. Otomatik Yedekleme Uyarıları

Öncelikle Database Mail ayarlarımızı yaparız

Manage Existing Account

Choose the account to view, change, or delete.



Account name:

mail



Delete

Description:

Outgoing mail server (SMTP)

E-mail address:

skyout800@gmail.com

Display name:

berk

Reply e-mail:

skyout800@gmail.com

Server name:

smtp.gmail.com

Port number:

587

☒ This server requires a secure connection (SSL)

SMTP Authentication

☐ Windows Authentication using Database Engine service credentials

☒ Basic authentication

User name:

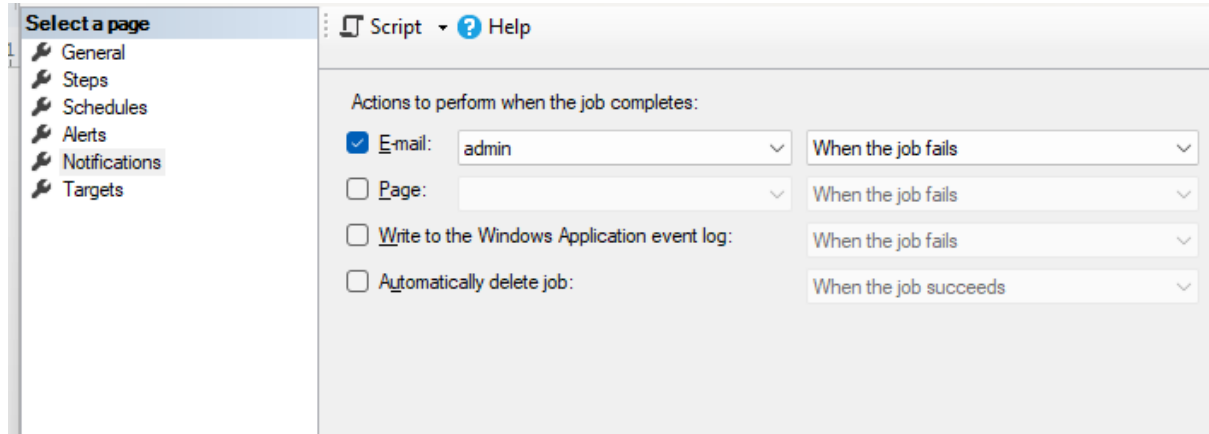
skyout800@gmail.com

Password:

Confirm password:

☐ Anonymous authentication

Ardından da Server Agent'a bunu tanıtmamız gerekir ki Mail sistemini kullanabilsin. Bunun için önce operator tanımlanır ve Job'umuzun properties kısmından "When the job fails" seçilir. Bu sayede işlemimizde hata oluşursa mail yoluyla bildirim gönderilir.



3. Veri tabanı Güvenliği ve Erişim Kontrolü

Bu bölüm veritabanı güvenliği ve erişim kontrolü konularını ele almaktadır. Genel olarak kullanıcı kimlik doğrulama yöntemleri, veri şifreleme, injection saldırılarına karşı koruma ve SQL Server Audit özelliklerinin kullanımından bahsedilecektir.

3.1. Erişim Kontrolü

Veritabanına erişimi SQL Server Authentication ve Windows Authentication ile sağlayabiliriz. SQL Server Authentication için öncelikle sunucuda oturum açabilecek kullanıcı oluşturulur, ardından da veritabanında bu kullanıcı oluşturulur.

```
CREATE LOGIN deneme_kullanici WITH PASSWORD = '1234';  
  
USE BLM4522;  
CREATE USER deneme_kullanici FOR LOGIN deneme_kullanici;
```

Bu kullanıcının erişebileceği tabloları belirleyebiliriz. Sadece istediğimiz tablolara erişim hakkı verip istediğimiz tablolara erişim hakkını yasaklayabiliriz.

```
GRANT SELECT on dbo.spend_analysis_dataset$ TO deneme_kullanici;  
DENY SELECT on dbo.spend_analysis_dataset$ TO deneme_kullanici;
```

GRANT SELECT, kullanıcıya o tabloya erişim hakkı tanır iken, DENY SELECT ise erişim hakkına izin vermez.

Windows Authentication kullanarak erişim sağlamak istiyor isek yukarı olduğu gibi bir kullanıcı ekleyip ardından oluşturmamız gerekmektedir.

```
CREATE LOGIN [berk\berk] FROM WINDOWS;  
USE BLM4522;  
CREATE USER [berk\berk] FOR LOGIN [berk\berk];
```

3.2. Veri Şifreleme

Veritabanının yetkisiz erişimlere karşı korunması amacıyla Transparent Data Encryption (TDE) yöntemi uygulanabilir. TDE, veritabanındaki tüm verilerin disk düzeyinde şifrlenmesini sağlar.

```
-- MASTER KEY oluşturulur (master veritabanında)  
USE master;  
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'SertifikaSifresi123!';  
  
-- Sertifika oluşturulur  
CREATE CERTIFICATE BLM4522_Cert  
WITH SUBJECT = 'BLM4522 Sertifikasi';  
  
-- Hedef veritabanına geçilir  
USE BLM4522;  
  
-- TDE için şifreleme anahtarı oluşturulur  
CREATE DATABASE ENCRYPTION KEY  
WITH ALGORITHM = AES_256  
ENCRYPTION BY SERVER CERTIFICATE BLM4522_Cert;  
  
-- Şifreleme etkinleştirilir  
ALTER DATABASE BLM4522 SET ENCRYPTION ON;
```


Öncelikle bir MASTER KEY oluşturulur, ardından sertifika oluşturulur. Son olarak da TDE için şifreleme anahtarı oluşturulur ve şifreleme etkinleştirilir.

3.3. SQL Injection Testleri

SQL Injection, kötü niyetli kullanıcıların web uygulamalarındaki veri giriş alanları aracılığıyla SQL sorgularına müdahale ederek veritabanına yetkisiz erişim sağlamaya çalıştığı bir saldırı türüdür.

```
SELECT * FROM dbo.user WHERE dbo.user = 'admin' AND sifre = '1234' OR 1=1;
```

Eğer bir kullanıcı yukarıdaki gibi bir sorgu girerse tüm user'lara erişebilir. Çünkü komuttaki OR 1=1 ifadesi her zaman doğru döndüreceğinden kimlik doğrulaması atlnaıp sistemdeki tüm kullanıcılar listelenebilir. SQL Injection'ı engellemenin en etkili yollarından biri parametrelili sorgular kullanmaktır. Bu yöntemle kullanıcıdan alınan veriler doğrudan sorguya gömülmez; veri ve sorgu mantıksal olarak ayrılır.

```
using (SqlCommand cmd = new SqlCommand("SELECT * FROM dbo.user WHERE username = @username AND password = @password", connection)){  
    cmd.Parameters.AddWithValue("@username", username);  
    cmd.Parameters.AddWithValue("@password", password);}
```

3.4. Audit Logları

Audit log, SQL Server'da yapılan işlemleri izleyip kayıt altına alan bir güvenlik mekanizmasıdır. Özellikle kim, ne zaman, hangi veritabanı nesnesi üzerinde hangi işlemi yaptı gibi bilgileri saklar. Bu sayede veritabanı yöneticisi, sistemde gerçekleşen olayları sonradan inceleyebilir. Örneğin bir tablo silindiyse, bunun hangi kullanıcı tarafından ve hangi tarihte yapıldığını öğrenmek mümkündür.

```
--Audit Özelliğini Açarız  
CREATE SERVER AUDIT BLM4522_Audit  
TO FILE (FILEPATH = 'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\AuditLogs\');
```

Yukarıda Audit loglarının yazılacağı dosya konumunu belirten bir audit nesnesi oluşturulmuştur.

Ardından Audit'i etkinleştiririz.

```
-- Audit'i etkinleştiririz  
ALTER SERVER AUDIT BLM4522_Audit WITH (STATE = ON);
```

Aşağıdaki ifade sayesinde de tablomuzda yapılan bütün SELECT ve INSERT işlemlerinin hareketlerini izleme özelliği tanımlarız.

```
-- Belirli hareketleri izlemek için AUDIT SPECIFICATION tanımlarız  
CREATE DATABASE AUDIT SPECIFICATION BLM4522_DBSpec  
FOR SERVER AUDIT BLM4522_Audit  
ADD (SELECT ON OBJECT::spend_analysis_dataset$ BY [public]),  
ADD (INSERT ON spend_analysis_dataset$ BY [public]);
```

Son olarak da Audit Spec'imizi aktif hale getiririz.

```
ALTER DATABASE AUDIT SPECIFICATION BLM4522_DBSpec WITH (STATE = ON);
```

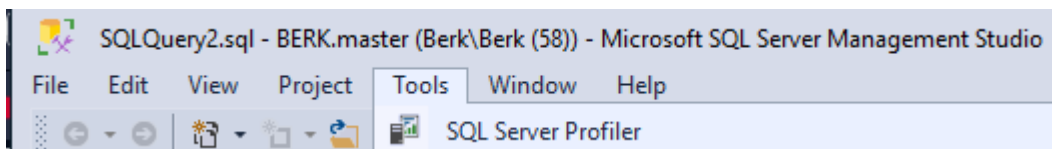
Altteki komut ile de Audit kayıtlarını okuyup inceleyebiliriz.

```
SELECT * FROM sys.fn_get_audit_file  
('C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\AuditLogs\*', NULL, NULL);
```

4. Veritabanı Performans Optimizasyonu ve İzleme

Bu bölümde veritabanımızın performans optimizasyonu ve izlenmesi bölümünde kullanılan yöntemler anlatılacaktır.

4.1. SQL Server Profiler Kullanma



SQL Server Profiler, SQL Server üzerindeki sorguların ve olayların gerçek zamanlı olarak izlenmesini sağlayan bir araçtır. Veritabanında hangi sorguların çalıştığını, ne kadar sürdüğünü, kim tarafından çalıştırıldığını ve sistem kaynaklarını

nasıl kullandığını takip ederek performans sorunlarının tespitine yardımcı olur. Ayrıca hataların izlenmesi, deadlock'ların analizi ve uygulama ile veritabanı arasındaki etkileşimlerin incelenmesi gibi durumlarda kullanılır. SSMS'imizin Tools bölümünün altından seçilir. Database'imizi seçtikten sonra aşağıdaki gibi bir tablo karşımıza çıkar. Buradan Template olarak TSQL_Duration seçilir ve çalıştırılır.

Trace Properties

General | Events Selection

Trace name:

Trace provider name:

Trace provider type: version:

Use the template:

☐ Save to file:
Set maximum file size (MB):
☒ Enable file rollover
☐ Server processes trace data

☐ Save to table:
☐ Set maximum rows (in thousands):

☐ Enable trace stop time:
☒ Set trace duration (in minutes):

Run Cancel Help

Ardından bize en maliyetli sorguları tablo halinde verir.

SQL Server Profiler

File Edit View Replay Tools Window Help

Untitled - 1 (BERK)

EventClass	Duration	TextData	SPID	BinaryData
SQL: BatchCompleted	29	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	31	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	32	if exists(select * from sys.server_e...	57	
SQL: BatchCompleted	36	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	37	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	43	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	43	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	48	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	48	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	50	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	50	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	51	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	54	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	54	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	57	SELECT target_data FROM sy...	57	
SQL: BatchCompleted	68	if exists(select * from sys.server_e...	57	
SQL: BatchCompleted	130	DECLARE @msticks...	57	
SQL: BatchCompleted	136	DECLARE @msticks...	57	

Trace Start

4.2 DMV'lerle Performans Analizi

DMV'ler, SQL Server'ın iç yapısına dair anlık bilgiler sunan özel sistem görünümleridir. Sunucunun çalışma durumu, sorgu performansı, bellek kullanımı, indeks durumu gibi pek çok konuda bilgi sağlayarak sistem yöneticilerine performans analizi ve sorun giderme konularında yardımcı olurlar.

```
SELECT
    migs.avg_total_user_cost * (migs.avg_user_impact / 100.0) * migs.user_seeks AS ImprovementMeasure,
    mid.statement AS TableName,
    mid.equality_columns,
    mid.inequality_columns,
    mid.included_columns
FROM sys.dm_db_missing_index_group_stats migs
JOIN sys.dm_db_missing_index_groups mig ON migs.group_handle = mig.index_group_handle
JOIN sys.dm_db_missing_index_details mid ON mig.index_handle = mid.index_handle
ORDER BY ImprovementMeasure DESC;
```

Örnek olarak yukarıdaki kod eksik indexleri bulmamızı sağlar.

```
SELECT
    OBJECT_NAME(i.object_id) AS TableName,
    i.name AS IndexName,
    i.index_id,
    s.user_seeks,
    s.user_scans,
    s.user_lookups,
    s.user_updates,
    (ISNULL(s.user_seeks, 0) + ISNULL(s.user_scans, 0) + ISNULL(s.user_lookups, 0)) AS TotalReads,
    s.user_updates AS TotalWrites
FROM sys.indexes AS i
LEFT JOIN sys.dm_db_index_usage_stats AS s
    ON i.object_id = s.object_id AND i.index_id = s.index_id
WHERE OBJECTPROPERTY(i.object_id, 'IsUserTable') = 1
    AND i.type_desc <> 'HEAP'
ORDER BY TotalReads ASC, TotalWrites DESC;
```

Bu kod ise gereksiz indexleri bulmamızı sağlar. Sorgu hızını arttırmak için bu gereksiz indexleri kaldırmayı düşünebiliriz.

4.3 Roller Ve Eriřim

```
CREATE ROLE VeriOkuyucu; -- örnek bir rol
GRANT INSERT, UPDATE, DELETE ON SCHEMA::dbo TO VeriOkuyucu;

EXEC sp_addrolemember 'VeriOkuyucu', 'deneme_kullanici';
```

Yukarıdaki gibi “CREATE ROLE” sorgusuyla yeni bir rol oluşturabiliriz.

Oluşturduğumuz yeni role “GRANT” aracılığı ile istediğimiz yetkileri verip, daha sonrasında bu rolü önceden oluşturmuş olduğumuz ‘deneme_kullanici’ kullanıcısına verebiliriz.

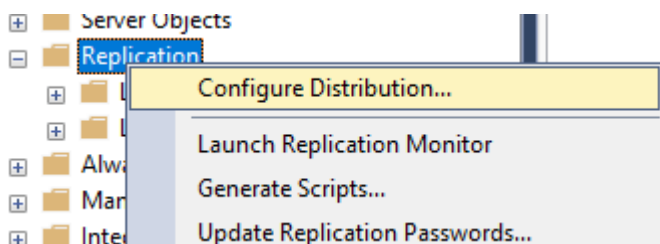
5. Veritabanı Yük Dengeleme ve Dağıtık Veritabanı Yapıları

Bu bölümde, birden fazla veritabanının senkronize çalışmasını sağlamak ve sistem yükünü dengelemek amacıyla SQL Server’da replikasyon, yük dengeleme ve failover (otomatik geçiş) stratejileri ele alınacaktır. Amaç; veri tutarlılığını koruyarak performansı artırmak, sistem kesintilerine karşı dayanıklılığı güçlendirmek ve kritik uygulamaların yüksek erişilebilirliğini garanti altına almaktır.

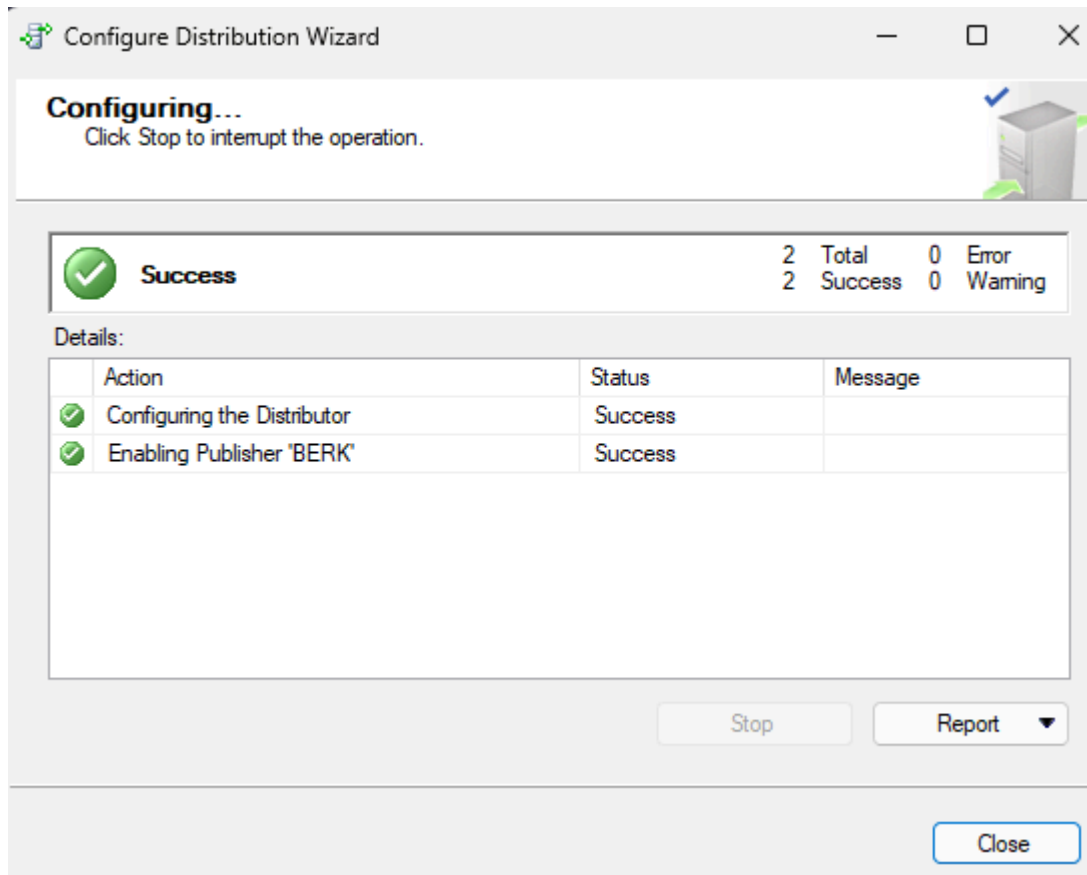
5.1. Veritabanı Replikasyonu

Veritabanı replikasyonu, SQL Server’da verilerin bir sunucudan başka bir sunucuya kopyalanarak çoğaltılmasını ve senkronize edilmesini sağlayan bir tekniktir. Bu sayede veriler, birden fazla noktada güncel olarak tutulabilir ve sistemler arası veri tutarlılığı sağlanır. Replikasyon, özellikle veri dağıtımı, raporlama sistemleri ve yedeklilik amaçları için kullanılır.

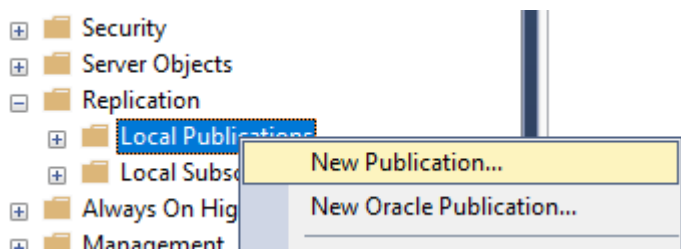
Sırası ile Distributor’ü Yapılandırma, Publication Oluşturma ve Subscription Oluşturma adımlarından oluşur. Distributor’ü yapılandırmak için öncelikle SSMS’imizin Replication kısmından Configure Distribution’a tıklanır.



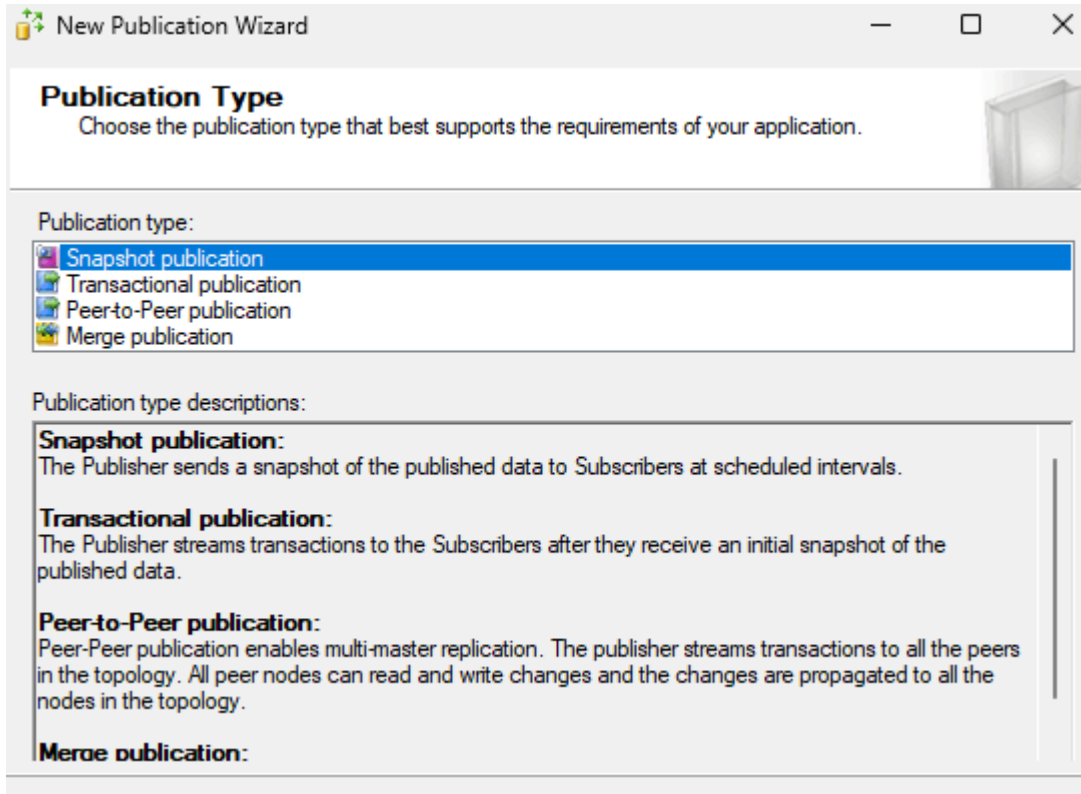
Ardından bize verilen komutlar takip edilerek kendi cihazımızı ya da başka bir cihazı distributor olarak ayarlarız.



Ardından publisher(yayıncı)'ı ayarlamamız gerekmektedir.

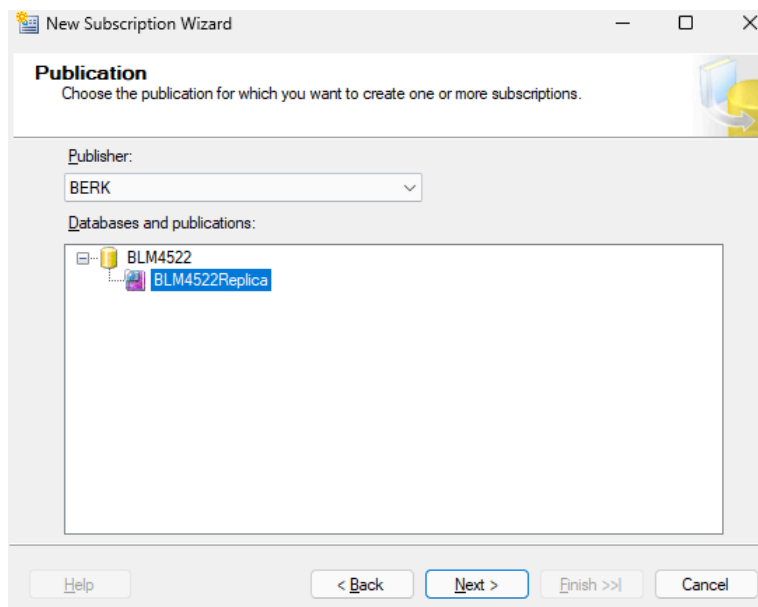


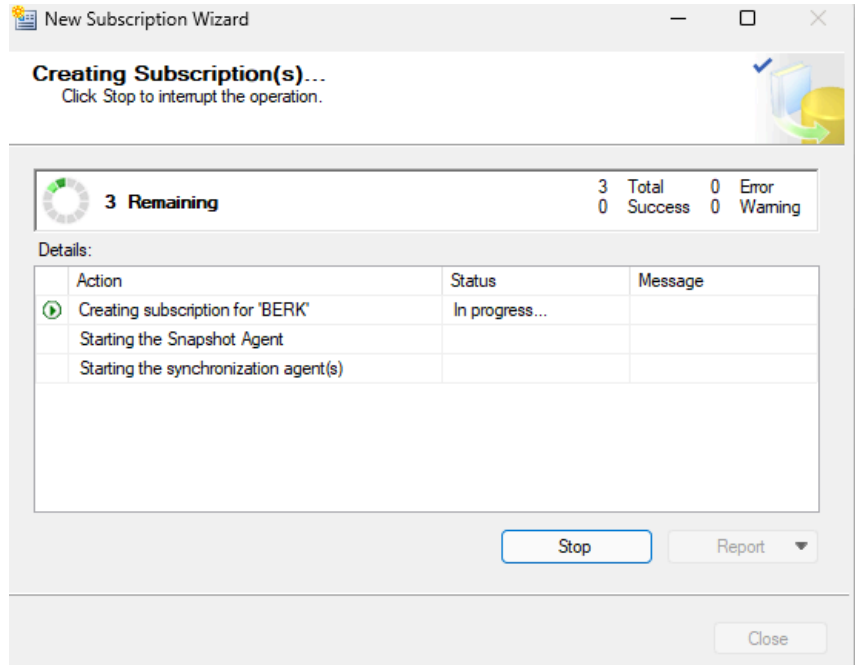
New Publication kısmına tıklanır ve ardından komutlar takip edilir. Ne türde bir yayın yapmak istediğimiz seçilir.



Snapshot publication, belli aralıklarla tablo verisini kopyalar ve gönderir.

Transactional publication ise tabloda bir değişim olduğunda anında kopyalama ve gönderme işlemi yapar. Ardından Local Subscriptions kısmından yayınlamış olduğumuz tabloya abone olunur.



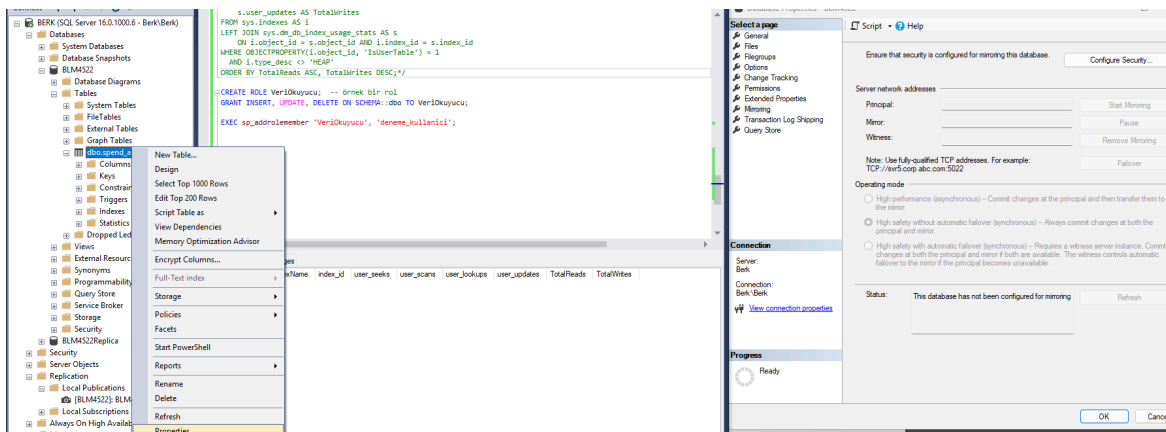


5.2. Yük Dengeleme

SQL Server'da yük dengeleme ve failover (otomatik geçiş) stratejileri, yüksek erişilebilirlik sağlamak ve hizmet kesintilerini en aza indirmek için kullanılır. Database Mirroring ve Always On Availability en yaygın kullanılan yöntemlerindendir.

5.2.1 Database Mirroring

Database Mirroring, SQL Server'da bir veritabanını yedek bir sunucuda birebir kopyalayarak yüksek erişilebilirlik sağlayan bir yapıdır. Ana sunucu çalışamaz hale geldiğinde, sistem manuel veya otomatik olarak yedek sunucuya geçiş yapabilir. Bu sayede veri güvenliği ve hizmet sürekliliği sağlanır.



Yukarıda görüldüğü üzere tablomuzun properties kısmına tıklarız ve sağdaki ekrandan mirroring'e girer ve konfigürasyonunu yaparız.

The screenshot shows the 'Configure Database Mirroring Security Wizard' window, specifically the 'Principal Server Instance' step. The window title is 'Configure Database Mirroring Security Wizard'. The main heading is 'Principal Server Instance' with a subtitle 'Specify information about the server instance where the database was originally located.' Below this, there is a dropdown menu for 'Principal server instance:' with 'BERK' selected. A checkbox labeled 'Encrypt data sent through this endpoint' is checked. Below the checkbox, there is a 'Listener port:' field with '5022' entered, and an 'Endpoint name:' field with 'Mirroring' entered. A note at the bottom states: 'NOTE: If the principal, mirror or witness are instances on the same server, their endpoints must use different ports.' At the bottom of the window, there are buttons for 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

Buradan hangi server'ımızın mirroring yapılacağını belirleriz. Listener portumuz 5022'dir. Ardından da hangi server'ımıza kopyalanacağını seçeriz. Daha sonrasında 2 serverimizi de birbiriyle eşleriz ve gerekli mirroring querylerini yazabiliriz.

6. VERİ TEMİZLEME VE ETL SÜREÇLERİ TASARIMI

Bu bölümde, büyük veri kümeleri üzerinde ETL (Extract, Transform, Load) süreçleri tasarlanarak veriler analiz için uygun hale getirilecektir. Süreç; verilerin farklı kaynaklardan alınması (extract), hatalı, eksik veya tutarsız verilerin temizlenip standart formata dönüştürülmesi (transform) ve doğru veritabanlarına yüklenmesi (load) adımlarını içerir. Ayrıca, veri kalitesini izlemek amacıyla hatalı kayıt sayıları, düzeltilen alanlar gibi bilgileri içeren raporlar hazırlanacaktır.

Visual Studio kullanarak SSIS aracılığı ile bir ETL paketi oluşturmak dağa sağlıklı olacaktır ancak ben burada direkt olarak SSMS üzerinden nasıl

yapılabileceğinden bahsedeceğim. Öncelikle veri kaybını önlemek için yeni bir tablo oluştururuz.

```
--Yanlışlıkla veri kaybını önlemek için geçici bir tablo oluştururuz.  
SELECT * INTO [BLM4522].[dbo].[STG_spend_analysis_dataset$] FROM [BLM4522].[dbo].[spend_analysis_dataset$];
```

Ardından hatalı verileri düzenler ve ayıklarız.

```
-- 1. Eksik veya NULL alanları düzelt (örnek: Buyer alanı)  
UPDATE [BLM4522].[dbo].[STG_spend_analysis_dataset$]  
SET Buyer = 'Bilinmiyor'  
WHERE Buyer IS NULL;  
  
-- 2. Hatalı Toplam Tutar (TotalCost) değerlerini düzelt  
-- TotalCost = Quantity * UnitPrice olmalı  
UPDATE [BLM4522].[dbo].[STG_spend_analysis_dataset$]  
SET TotalCost = Quantity * UnitPrice  
WHERE TotalCost <> Quantity * UnitPrice;  
  
-- 3. Tutarsız kategori adlarını standartlaştır (örnek: 'stationary' → 'Stationery')  
UPDATE [BLM4522].[dbo].[STG_spend_analysis_dataset$]  
SET Category = 'Stationery'  
WHERE Category IN ('stationary', 'Statnry');  
  
-- 4. Yinelenen kayıtları sil (aynı TransactionID ile birden fazla kayıt varsa sadece en yenisini tut)  
WITH Tekrarlar AS (  
    SELECT *, ROW_NUMBER() OVER (  
        PARTITION BY TransactionID  
        ORDER BY PurchaseDate DESC  
    ) AS rn  
    FROM [BLM4522].[dbo].[STG_spend_analysis_dataset$]  
)  
DELETE FROM Tekrarlar WHERE rn > 1;
```

Tüm bunlar yapıldıktan sonra da verimizi yeni, temiz bir tabloya yükleriz.

```
--STG'de değiştirilen verileri temize çekeriz.  
SELECT * INTO [BLM4522].[dbo].[Clean_spend_analysis_dataset$] FROM [BLM4522].[dbo].[STG_spend_analysis_dataset$];
```

7. Veritabanı Yükseltme ve Sürüm Yönetimi

Bu bölümde, mevcut veri tabanının daha yeni bir SQL Server sürümüne güvenli şekilde yükseltilmesi hedeflenmektedir. Süreç; yükseltme öncesi planlama, şema değişikliklerinin sürüm kontrolüyle izlenmesi ve yükseltme sonrası testlerin yapılması ve gerekirse eski duruma geri dönüş stratejisinin hazırlanmasını içermektedir. Amaç, veri kaybı ve kesinti riskini en aza indirerek sorunsuz bir geçiş sağlamaktır.

Şu anda kullanmakta olduğum Microsoft SQL sürümüm son sürüm olan Microsoft SQL Server 2022 olduğu için versiyon yükseltmeyi bir senaryo üzerinden anlatacağım.

Senaryomuz veritabanını eski bir SQL Server sürümünden (örneğin 2016) daha yeni bir sürüme (örneğin 2022) taşımak.

- SQL Server Data Migration Assistant (DMA) aracını kullanarak eski veritabanının yeni sürümle uyumluluğu kontrol edilir.
- Eski sürümde veritabanını yedeklenir.

```
BACKUP DATABASE [BLM4522] TO DISK = 'C:\Backup\BLM4522.bak';
```

- Yeni sunucuda geri yüklenir.
- Uyumluluk seviyesi güncellenir

```
ALTER DATABASE [BLM4522] SET COMPATIBILITY_LEVEL = 160; -- örnek: SQL Server 2022
```

DDL Triggers kullanarak sürüm değişikliklerini gözlemleyebiliriz. Bu bize veritabanı yapısında kim, ne zaman, ne değiştirdi, bunları gösterir.

```
CREATE TABLE SchemaChangesLog (  
    EventTime DATETIME,  
    EventType NVARCHAR(100),  
    ObjectName NVARCHAR(256),  
    ObjectType NVARCHAR(100),  
    UserName NVARCHAR(256)  
);  
  
CREATE TRIGGER DDL_SchemaChange  
ON DATABASE  
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE, CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE  
AS  
BEGIN  
    INSERT INTO SchemaChangesLog (EventTime, EventType, ObjectName, ObjectType, UserName)  
    SELECT  
        GETDATE(),  
        EVENTDATA().value('/EVENT_INSTANCE/EventType[1]', 'NVARCHAR(100)'),  
        EVENTDATA().value('/EVENT_INSTANCE/ObjectName[1]', 'NVARCHAR(256)'),  
        EVENTDATA().value('/EVENT_INSTANCE/ObjectType[1]', 'NVARCHAR(100)'),  
        SYSTEM_USER;  
END;
```

Bu tetikleyici, tablolar/prosedürler üzerinde yapılan değişiklikleri otomatik olarak kaydeder.