ANKARA ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
BÖLÜMÜ

(BLM4522) Ağ Tabanlı Paralel Dağıtım Sistemleri

Mertcan Özdemir

Ömer Faruk Karagöz

21290194

21290585

Github: mertcan-ozdemir

Github repo: https://github.com/mertcan-ozdemir/BLM4522-21290194

veritabanı güvenliği ve erişimi
https://youtu.be/B4G72MCuPuM

veritabanı otomasyon ve bildirim
https://youtu.be/_M4R-XZu6eM

veritabanı yedekleme ve felaketten kurtarma senaryosu
https://youtu.be/xM3QI4Mqlm4

1. Veri tabanı Güvenliği ve Erişim Kontrolü

Bu bölümde MovieLensDB üzerinde gerçekleştirilen veritabanı güvenliği ve erişim kontrolü uygulamalarını kapsamaktadır. Temel olarak kullanıcı kimlik doğrulama yöntemleri, erişim izinlerinin yönetimi, SQL injection saldırılarına karşı alınan önlemler ve kullanıcı aktivitelerinin izlenmesi (audit loglama) konularına odaklanılmıştır.

1.1. Erişim Yönetimi

Veritabanına erişimi olan kullanıcıların kimlik doğrulama işlemleri SQL Server Authentication ve Windows Authentication ile sağlanabilir. Çalışmamızda SQL Authentication yöntemini tercih ettik.

```
-- SQL Auth ile yeni bir kullanıcı oluşturma
USE [master];
GO

CREATE LOGIN [mertcan_omer] WITH PASSWORD = 'MovieLens';
GO

USE MovieLensDB;
GO

CREATE USER [mertcan_omer] FOR LOGIN [mertcan_omer];
GO
```

İlk olarak master komutu ile öncelikle sistem veritabanına geçilmiştir. Çünkü SQL Server üzerinde yeni bir oturum tanımı yalnızca master veritabanı üzerinden yapılabilir.

Ardından CREATE LOGIN komutu ile SQL Server genelinde geçerli olacak yeni bir oturum tanımlanır. Ardından, kullanıcıya yetki verilmek istenen veritabanı olan MovieLensDB seçilmiştir.

Ve son olarak daha önce master veritabanında oluşturulan login'e karşılık gelen bir veritabanı kullanıcısı oluşturulur. Böylece mertcan_omer, MovieLensDB veritabanı içinde işlem yapabilir hale gelir. Bu adımları takip ederek SQL Server Authentication yöntemi ile yeni bir kullanıcı oluşturulmuştur.

```
-- Kullanıcıya sadece 'kullanici' tablosuna erişim izni ver
GRANT SELECT ON dbo.kullanici TO [mertcan_omer];
GO

-- Kullanıcıya 'rating' tablosuna erişim izni verilmesin
DENY SELECT ON dbo.rating TO [mertcan_omer];
GO
```

Oluşturduğumuz yeni kullanıcının erişebileceği ve erişemeyeceği tabloları burada GRANT (Erişim izni vermek için) ve DENY (Erişim izni reddetmek için) komutları ile belirtildi.

Yukarıda adımlarda SQL Server Authentication ile kullanıcı oluşturma ve o kullanıcıya erişim izinleri verme veya reddetme işlemlerinin nasıl yapıldığı gösterilmiştir.

1.2. Veri Şifreleme

Bu bölümde, MovieLensDB adlı veritabanının yetkisiz erişimlere karşı korunması amacıyla Transparent Data Encryption (TDE) yöntemi uygulanmıştır. TDE, veritabanındaki tüm verilerin disk düzeyinde şifrelenmesini sağlar.

1.2.1. Master Key Oluşturma

Master veritabanında şifreli işlemler için bir ana anahtar (master key) oluşturulmuştur

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'SifreliParola123!';
```

1.2.2. Sertifika Oluşturulması

Master key ile ilişkili bir sertifika oluşturulmuştur:

```
CREATE CERTIFICATE MovieLensCert
WITH SUBJECT = 'TDE Certificate';
```

1.2.3. Şifreleme Anahtarının Tanımlanması

Şifreleme işlemlerini gerçekleştirecek olan anahtar, hedef veritabanı olan MovieLensDB içinde tanımlanmıştır

```
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE MovieLensCert;
```

1.2.4. TDE'nin Aktif Edilmesi

Veritabanı şifrelemesi aşağıdaki komut ile aktif edilmiştir

```
ALTER DATABASE MovieLensDB
SET ENCRYPTION ON;
```

1.2.5. Kontrol Etme

Aşağıdaki komutla şifrelenip şifrelenmediği kontrol edilmiştir.

```sql
SELECT
    db.name,
    db.is_encrypted,
    dm.encryption_state,
    dm.percent_complete,
    dm.key_algorithm,
    dm.key_length
FROM sys.databases db
LEFT JOIN sys.dm_database_encryption_keys dm
ON db.database_id = dm.database_id;
```

100 %

Results | Messages

	name	is_encrypted	encryption_state	percent_complete	key_algorithm	key_length
1	master	0	NULL	NULL	NULL	NULL
2	tempdb	1	3	0	AES	256
3	model	0	NULL	NULL	NULL	NULL
4	msdb	0	NULL	NULL	NULL	NULL
5	proje	0	NULL	NULL	NULL	NULL
6	deneme	0	NULL	NULL	NULL	NULL
7	MovieLensDB	1	3	0	AES	256

1.3. 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.

```sql
-- Bu, sorguyu değiştirebilir ve tüm veritabanındaki verileri çekebilir.
SELECT * FROM dbo.user WHERE username = '' OR 1=1; --' AND password = 'password';
```

Yukarıdaki sorguda OR 1=1 ifadesi her zaman doğru döneceği için, kimlik doğrulaması atlanarak sistemdeki tüm kullanıcılar listelenebilir. SQL Injection'ı engellemenin en etkili yollarından biri parametreli sorgular kullanmaktır. Bu yöntemle kullanıcıdan alınan veriler doğrudan sorguya gömülmez; veri ve sorgu mantıksal olarak ayrılır. Örneğin, C# ile ADO.NET kullanarak parametreli bir sorgu örneği:

```
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);}
```

Yukarıdaki sorgu ile kullanıcının girdiği değerler @username ve @password parametrelerine atanır. Parametreler SQL sorgusundan **bağımsız** şekilde veritabanına iletilir. SQL komutuna dışardan müdahale edilemez hale gelir ve SQL Injection engellenmiş olur.

1.4. Audit Logları

SQL Server Audit özelliği, veri tabanı üzerindeki kullanıcı aktivitelerini detaylı şekilde izlemek ve kayıt altına almak için geliştirilmiştir. Burada, kullanıcı aktivitelerini izlemek için SQL Server Audit yapılandırmasının nasıl gerçekleştirileceğini adım adım açıklamaktadır. Örnek uygulama olarak MovieLens_Audit adlı bir denetim (audit) tanımlanmıştır.

```
-- Audit özelliğini açma
CREATE SERVER AUDIT MovieLens_Audit
TO FILE (FILEPATH = 'C:\AuditLogs\');
GO
```

Yukarıda Audit loglarının yazılacağı dosya konumunu belirten bir audit nesnesi oluşturulmuştur. MovieLens_Audit adlı bir denetim tanımlanmıştır. Log dosyaları C:\AuditLogs\ dizinine yazılacaktır.

```
-- Audit spesifik veritabanı işlemleri ekleme
CREATE SERVER AUDIT SPECIFICATION MovieLens_Audit_Specification
FOR SERVER AUDIT MovieLens_Audit
ADD (FAILED_LOGIN_GROUP),
ADD (DATABASE_OBJECT_PERMISSION_CHANGE_GROUP),
ADD (SCHEMA_OBJECT_ACCESS_GROUP)
WITH (STATE = ON);
GO
```

Yukarıda sunucu düzeyinde hangi aktivitelerin izleneceği belirlenmiştir.

FAILED_LOGIN_GROUP: Başarısız giriş denemelerini izlenmesini sağlar.

DATABASE_OBJECT_PERMISSION_CHANGE_GROUP: Veritabanı nesnelerinin izinlerinde yapılan değişikliklerin izlenmesini sağlar.

SCHEMA_OBJECT_ACCESS_GROUP: Şema nesnelerine yapılan erişim işlemlerin izlenmesini sağlar.

```
-- Audit özelliğini aktif hale getirme
ALTER SERVER AUDIT MovieLens_Audit
WITH (STATE = ON);
GO
```

Yukarıda Audit nesnesi çalışır hale getirilmiştir. Artık kullanıcı aktiviteleri izlenmeye başlanmıştır.

```
SELECT *
FROM fn_get_audit_file('C:\AuditLogs\*', NULL, NULL);
GO
```

Yukarıda Audit kayıtlarını incelemek için bir sorgu oluşturulmuştur. fn_get_audit_file fonksiyonu, belirttiğiniz dizindeki tüm audit loglarını okur ve görüntüler. Bu kayıtlar, kullanıcı hareketlerini detaylı bir şekilde sunar (tarih, kullanıcı adı, işlem türü vb.).

	event_time	sequence_number	action_id	succeeded	permission_bitmask	is_column_permission	session_id	server_principal_id	database_principal_id	target_server_principal_id	target_database_principa
1	2025-04-22 18:28:08.2966549	1	AUSC	1	0x00000000000000000000000000000000	0	66	259	0	0	0
2	2025-04-22 18:28:25.0764294	1	LGIF	0	0x00000000000000000000000000000000	0	0	0	0	0	0
3	2025-04-22 18:28:29.1885353	1	LGIF	0	0x00000000000000000000000000000000	0	0	0	0	0	0
4	2025-04-22 18:28:31.9007933	1	LGIF	0	0x00000000000000000000000000000000	0	0	0	0	0	0
5	2025-04-22 18:28:35.5810652	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
6	2025-04-22 18:28:35.5951253	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
7	2025-04-22 18:28:35.5951253	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
8	2025-04-22 18:28:35.6041257	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
9	2025-04-22 18:28:35.6061275	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
10	2025-04-22 18:28:35.6564253	1	SL	1	0x00000000000000000000000000000001	1	65	267	2	0	0
11	2025-04-22 18:28:35.6634149	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
12	2025-04-22 18:28:35.7724606	1	EX	1	0x00000000000000000000000000000020	0	59	267	2	0	0
13	2025-04-22 18:28:35.7823969	1	SL	1	0x00000000000000000000000000000001	1	59	267	2	0	0
14	2025-04-22 18:28:35.8434301	1	SL	1	0x00000000000000000000000000000001	1	70	267	2	0	0

Verilen sorgu sonucu yukarıdaki gibi bir tablo çıktısı verilir.

2. Veritabanı Yedekleme ve Felaketten Kurtarma Planı

Bu bölümde, SQL Server ortamında veritabanı yedekleme ve felaketten kurtarma planlarının nasıl oluşturulacağına bahsedilmektedir. Yedekleme stratejileri, otomatikleştirme, felaket senaryoları ve test süreçleri ele alınmıştır. Çalışmalarda MovieLens 100K veri seti kullanılmıştır.

2.1. Yedekleme Stratejileri

2.1.1. Tam Yedekleme

Veritabanının tamamı yedeklenir. Komutu şu şekildedir.

```
BACKUP DATABASE MovieLensDB
TO DISK = 'C:\Backups\movielens_full.bak';
```

2.1.2. Fark Yedekleme

Son tam yedekten sonra değişen veriler yedeklenir. Komutu şu şekildedir.

```
BACKUP DATABASE MovieLensDB
TO DISK = 'C:\Backups\movielens_diff.bak'
WITH DIFFERENTIAL;
```

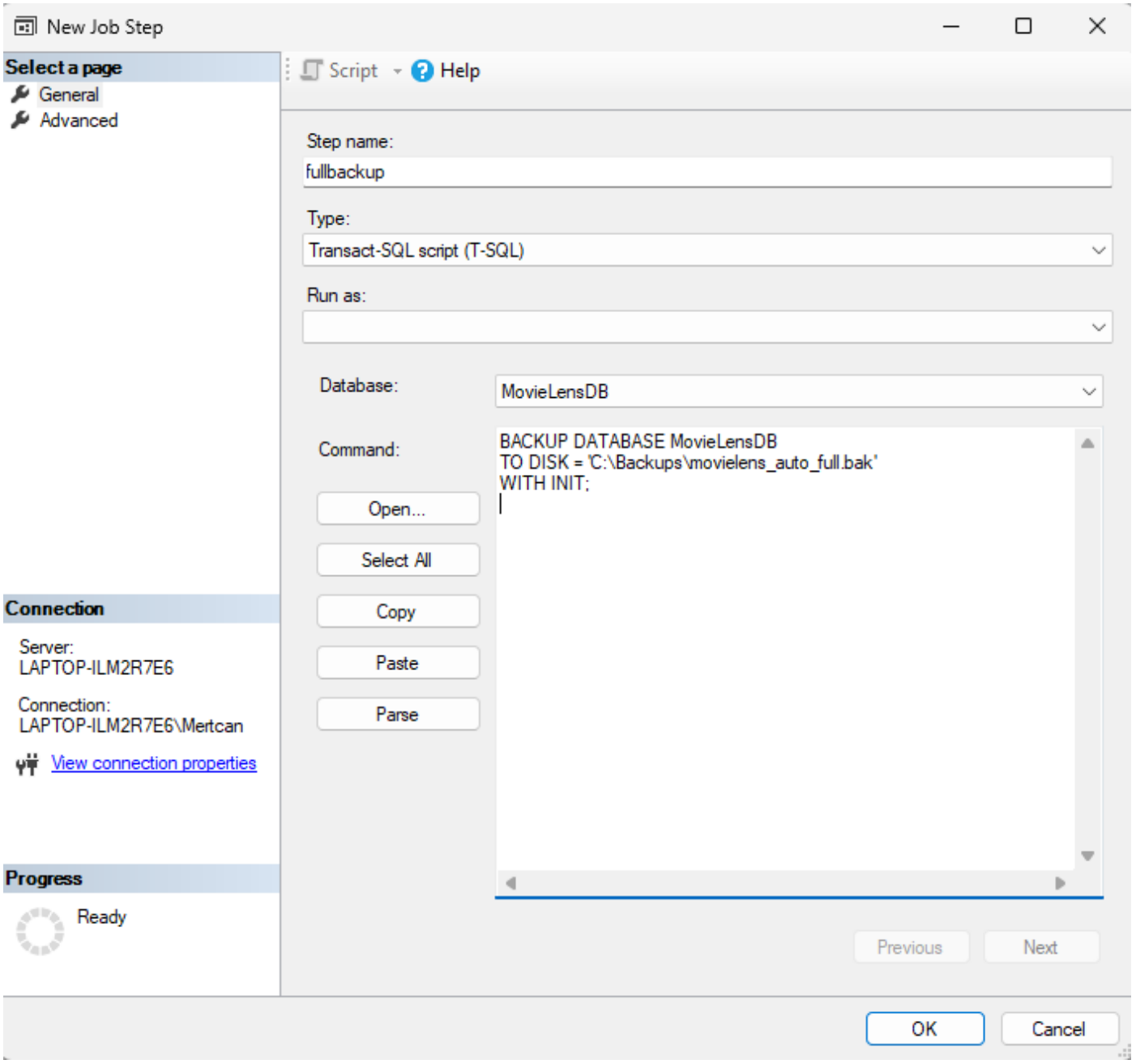
2.1.3. Artımlı Yedekleme

Veritabanında yapılan her değişikliği içeren log dosyaları yedeklenir.

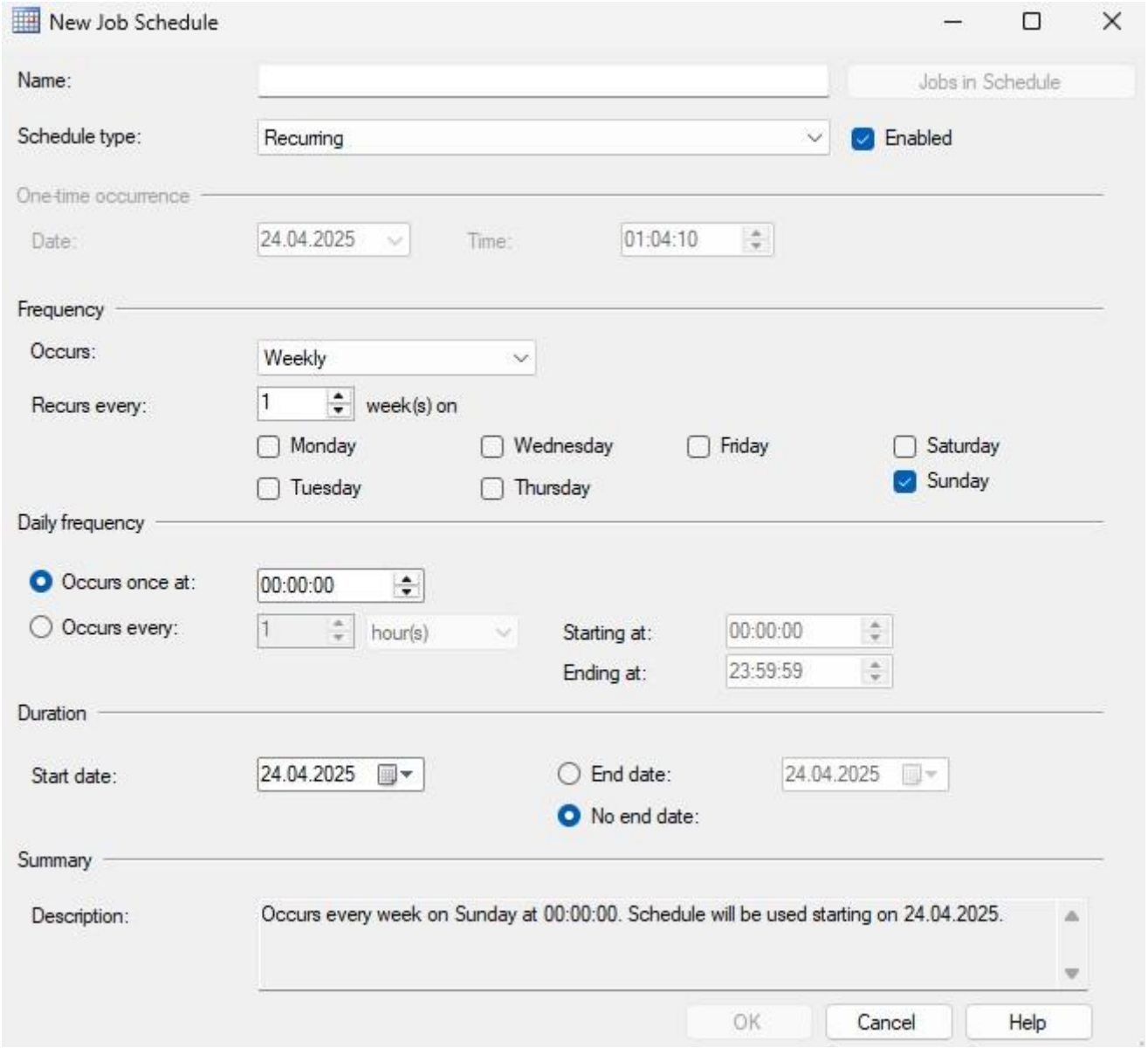
```
BACKUP LOG MovieLensDB
TO DISK = 'C:\Backups\movielens_log.trn';
```

2.2. Zamanlayıcılarla Yedekleme

Zamanlayıcılar ile yedekleme yapmak için SQL Server Agent üzerinden Job oluşturulabilir. Job oluşturmak için SQL Server Agent > New > Job seçilir. Açılan pencerenin sol üst kısmından steps seçilir. New butonuna basılıp tekrar açılan pencereden step için isim verilir, Database ve type olarak Transact-SQL script seçilir. Son olarak command girilir ve OK butonuna basılır.



Step tamamlandıktan sonra zamana bağlamak için schedule seçilir. Burada kaç günde veya hangi saat aralığında step'i çalıştıracağı seçilir.



2.3. Felaketten Kurtarma Senaryoları

Veritabanında bir tablo veya kendisi silinirse yedekler ile geri döndürülebilir.

```
USE master;
ALTER DATABASE MovieLensDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE MovieLensDB
FROM DISK = 'C:\Backups\movielens_full.bak'
WITH REPLACE;

ALTER DATABASE MovieLensDB SET MULTI_USER;
```

Veritabanı yedeğinin geri yüklenmesi sürecinde öncelikle sistem veritabanı olan master

veritabanına geçilmiştir, çünkü geri yükleme işlemleri sırasında hedef veritabanı kullanımda olamaz. Bu işlemden sonra, ALTER DATABASE MovieLensDB SET SINGLE_USER WITH ROLLBACK IMMEDIATE komutu ile MovieLensDB veritabanı tek kullanıcı moduna alınmış ve o an bağlı olan tüm kullanıcıların işlemleri iptal edilerek bağlantıları sonlandırılmıştır. Ardından, RESTORE DATABASE MovieLensDB FROM DISK = 'C:\Backups\movielens_full.bak' WITH REPLACE komutu ile belirtilen .bak dosyasından veritabanı geri yüklenmiştir. WITH REPLACE ifadesi, mevcut veri tabanının üzerine yazılmasına olanak tanımaktadır. Son olarak, ALTER DATABASE MovieLensDB SET MULTI_USER komutu ile veritabanı tekrar çok kullanıcı moduna geçirilmiş ve normal erişime açılmıştır. Bu işlemler sayesinde veri tabanı eski haline geri döndürülmüştür.

2.4. Test Yedekleme Senaryoları

Yedeklemelerin doğruluğunu kontrol etmek için sorfgu ile bir veri sildik ve yedekleri geri yükledikten sonra silinen veri geri geldi mi kontrol edildi.

Veri tabanımızda select * FROM kullanici WHERE user_id = 196 AND item_id = 242; sorgusunu çalıştırdığımızda user_id = 196 ve item_id = 242 denk olduğu bilgileri siler.

```
select * from dbo.kullanici WHERE user_id = 196 AND item_id = 242;
```

	user_id	item_id	rating	timestamp
1	196	242	3	881250949

Sorgusu bize böyle bir çıktı veriyor. DELETE FROM kullanici WHERE user_id = 196 AND item_id = 242; sorgusunu çalıştırırsak bunu silecektir. Bu satırı geri getirmek için yukarıda anlatıldığı gibi veri tabanı yedeği yüklenir ve aynı sorgu tekrar çalıştırıldığında user_id = 196 ve item_id = 242 çıktısının sonucu geri geldiği görülür.

3. Veritabanı Yedekleme ve Otomasyon Çalışması

3.1. SQL Server Agent ile yedekleme süreçlerini otomatikleştirme

SQL Server Agent kullanarak bir job ataması ile zamana dayalı otomatik yedekleme işlemi yapılmıştır. Bölüm 2.2.'de daha ayrıntılı olarak anlatılmıştır. Özet olarak SQL Server Agent'a yeni bir job atanır, çalışma sıklığı belirlenir (projemizde günlük olarak çalışacak şekilde atanmıştır) ve aşağıdaki şekildeki gibi yapacağı iş belirlenir (yedekleme).

```
BACKUP DATABASE MovieLensDB
TO DISK = 'C:\Backups\movielens_auto_full.bak'
WITH INIT;
```

3.2. T-SQL Scripting ile yedekleme raporları oluşturma

```
SELECT
    database_name,
    backup_start_date,
    backup_finish_date,
    type AS backup_type,
    physical_device_name
FROM msdb.dbo.backupset b
JOIN msdb.dbo.backupmediafamily m ON b.media_set_id = m.media_set_id
WHERE database_name = 'MovielensDB'
ORDER BY backup_finish_date DESC;
```

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

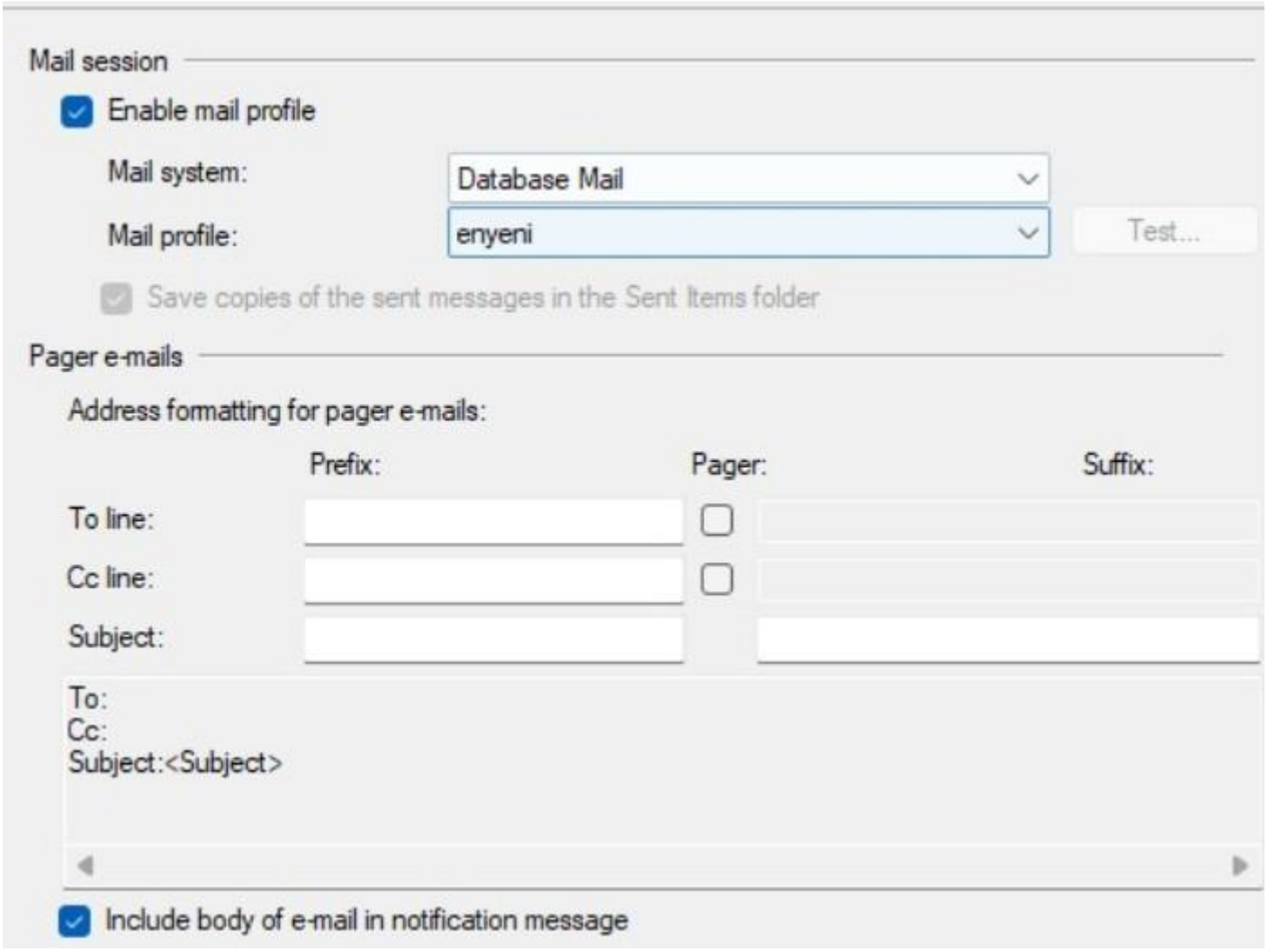
	database_name	backup_start_date	backup_finish_date	backup_type	physical_device_name
1	MovieLensDB	2025-04-23 17:26:12.000	2025-04-23 17:26:12.000	D	C:\Backups\movielens_auto_full.bak
2	MovieLensDB	2025-04-23 17:18:17.000	2025-04-23 17:18:17.000	L	C:\Backups\movielens_log.trn
3	MovieLensDB	2025-04-23 17:17:07.000	2025-04-23 17:17:07.000	I	C:\Backups\movielens_diff.bak
4	MovieLensDB	2025-04-23 17:16:17.000	2025-04-23 17:16:17.000	D	C:\Backups\movielens_full.bak

3.3. Otomatik Yedekleme Uyarıları

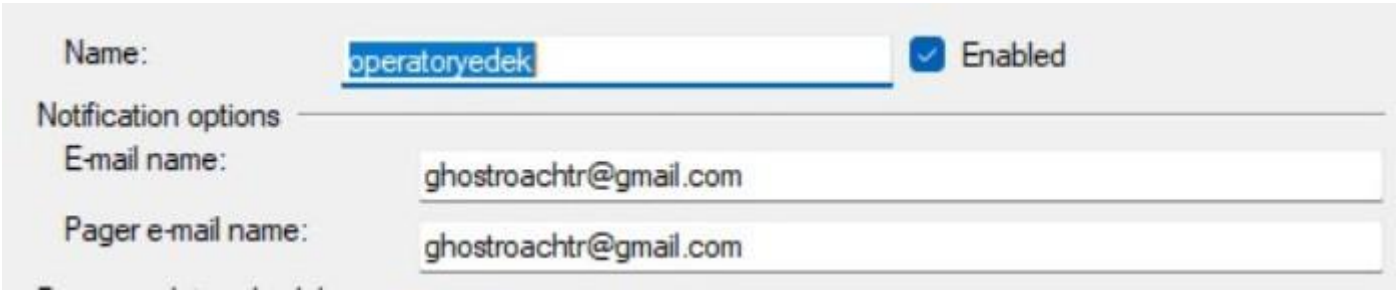
Yedekleme sırasında meydana gelen hataları kullanıcıya mail ile bildirmek otomasyon için çok önemlidir. Aşağıda bu işlemin nasıl yapıldığı açıklanmıştır.

İlk olarak bir mail profili oluşturulur ve SMTP sunucusu, port, kullanıcı adı/şifre gibi ayarları yapılır ve bir test maili göndererek doğruluğu kontrol edilir.

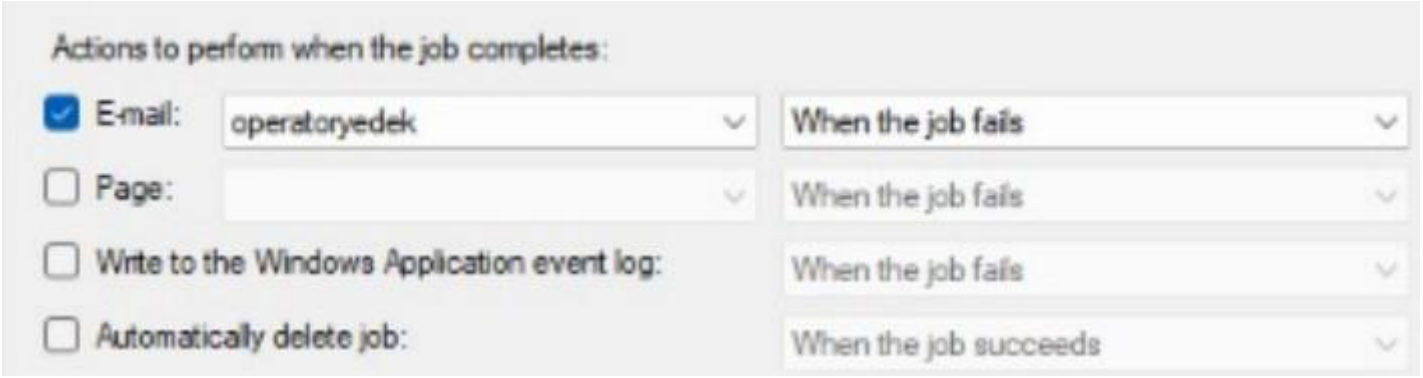
İlk aşamadan sonra SQL Server Agent'ın, Database Mail profilini kullanabilmesi için bunu tanıması gerekir. Tanımlama işlemini SQL Server Agent Alert System'dan aşağıdaki şekilde yapılır.



Ardından bildirim gönderilecek kişi ya da sistem (e-posta) adresi aşağıdaki gibi operatör olarak tanımlanır.



En son olarak yedekleme job'u başarısız olursa hangi operatöre, ne zaman bildirim gideceğini Job Notifications ile aşağıdaki gibi tanımlanır.



Yaptığımız işlemler sonucu job'un yedekleme hatası alması durumunda kullanıcıya mail gönderiyor mu test etmek için job'u kasıtlı olarak hata alacak şekilde çalıştırdık. Yaptıklarımız sonucunda mail adresimize Yedekleme başarısız maili aşağıdaki gibi gelmiş oldu.