

Mustafa Girgin  
21290649

<https://github.com/antelcha/sqlprojeler>

# Veritabanı Versiyon Yönetimi ve Sürüm Kontrolü

## 1. İlk Kurulum ve Temel Yapılandırma

VersionControlDB veritabanı üzerinde temel versiyon yönetimi yapısını adım adım kurdum. Buradaki hedefim, veritabanı değişikliklerini kontrollü ve güvenli bir şekilde yönetmektir.

İlk olarak, `1\_initial\_setup.sql` dosyasında veritabanımızı ve temel tablolarımızı oluşturuyoruz. Bu aşamada:

- VersionControlDB adında yeni bir veritabanı oluşturuyoruz
- DatabaseVersion tablosunu oluşturuyoruz
- İlk versiyon kaydını (1.0.0) ekliyoruz
- Örnek bir Customers tablosu oluşturuyoruz

```

1  -- Veritabanı oluşturma
2  IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'VersionControlDB')
3  BEGIN
4      CREATE DATABASE VersionControlDB;
5  END
6  GO
7
8  USE VersionControlDB;
9  GO
10
11 -- Versiyon kontrol tablosu
12 BEGIN TRANSACTION;
13     IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'DatabaseVersion')
14     BEGIN
15         CREATE TABLE DatabaseVersion (
16             VersionID INT PRIMARY KEY IDENTITY(1,1),
17             VersionNumber VARCHAR(20) NOT NULL,
18             AppliedDate DATETIME DEFAULT GETDATE(),
19             Description NVARCHAR(500),
20             Status VARCHAR(50)
21         );
22
23         -- İlk versiyon kaydı
24         INSERT INTO DatabaseVersion (VersionNumber, Description, Status)
25         VALUES ('1.0.0', 'Initial database version', 'Completed');
26     END
27
28 -- Örnek tablo oluşturma
29 IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Customers')
30 BEGIN
31     CREATE TABLE Customers (
32         CustomerID INT PRIMARY KEY IDENTITY(1,1),
33         FirstName NVARCHAR(50),
34         LastName NVARCHAR(50),
35         Email NVARCHAR(100)
36     );
37 END
38 COMMIT TRANSACTION;

```

## 2. Şema Değişikliklerini İzleme Sistemi

`2\_schema\_tracking.sql` dosyasında, veritabanında yapılan tüm değişiklikleri otomatik olarak izleyen bir sistem kuruyoruz. Bu sistem:

- SchemaChanges tablosunu oluşturuyor
- DDL Trigger ekliyor
- Tüm şema değişikliklerini kaydediyor

```

USE VersionControlDB;
GO

-- Şema değişikliklerini izlemek için tablo oluşturma
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'SchemaChanges')
BEGIN
    CREATE TABLE SchemaChanges (
        ChangeID INT PRIMARY KEY IDENTITY(1,1),
        EventType NVARCHAR(100),
        ObjectName NVARCHAR(256),
        ObjectType NVARCHAR(100),
        SQLCommand NVARCHAR(MAX),
        LoginName NVARCHAR(256),
        ChangeDate DATETIME DEFAULT GETDATE()
    );
END
GO

-- DDL Trigger oluşturma
IF NOT EXISTS (SELECT * FROM sys.triggers WHERE name = 'TR_TrackSchemaChanges')
BEGIN
    EXEC('CREATE TRIGGER TR_TrackSchemaChanges
    ON DATABASE
    FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE,
    CREATE_PROCEDURE, ALTER_PROCEDURE, DROP_PROCEDURE,
    CREATE_FUNCTION, ALTER_FUNCTION, DROP_FUNCTION,
    CREATE_VIEW, ALTER_VIEW, DROP_VIEW
    AS
    BEGIN
        SET NOCOUNT ON;

        DECLARE @EventData XML = EVENTDATA();

        INSERT INTO SchemaChanges (
            EventType,
            ObjectName,
            ObjectType,
            SQLCommand,
            LoginName
        )
        VALUES (
            @EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(100)'),
            @EventData.value('(/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(256)'),
            @EventData.value('(/EVENT_INSTANCE/ObjectType)[1]', 'NVARCHAR(100)'),
            @EventData.value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]', 'NVARCHAR(MAX)'),
            @EventData.value('(/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(256)')
        );
    END;')
END
GO

```

### 3. Versiyon Yönetim Prosedürleri

`3\_version\_management.sql` dosyasında, versiyon yönetimi için gerekli stored procedure'leri oluşturuyoruz. Bu prosedürler:

- sp\_UpgradeDatabase: Yeni versiyona geçiş
- sp\_RollbackDatabase: Önceki versiyona dönüş
- sp\_GetVersionHistory: Versiyon geçmişi görüntüleme

```
USE VersionControlDB;
GO

-- Veritabanı versiyonunu yükseltme prosedürü
IF NOT EXISTS (SELECT * FROM sys.procedures WHERE name = 'sp_UpgradeDatabase')
BEGIN
    EXEC('CREATE PROCEDURE sp_UpgradeDatabase
        @TargetVersion VARCHAR(20),
        @Description NVARCHAR(500)
    AS
    BEGIN
        SET NOCOUNT ON;

        BEGIN TRY
            BEGIN TRANSACTION;

            -- Mevcut versiyonu kontrol et
            DECLARE @CurrentVersion VARCHAR(20);
            SELECT TOP 1 @CurrentVersion = VersionNumber
            FROM DatabaseVersion
            ORDER BY VersionID DESC;

            IF @CurrentVersion >= @TargetVersion
            BEGIN
                RAISERROR (N'Hedef versiyon mevcut versiyondan küçük veya eşit
olamaz.', 16, 1);
                RETURN;
            END

            -- Yeni versiyon kaydı ekle
            INSERT INTO DatabaseVersion (VersionNumber, Description, Status)
            VALUES (@TargetVersion, @Description, 'In Progress');
```

```

-- Burada yükseltme işlemleri gerçekleştirilir
-- (Örnek: ALTER TABLE, CREATE TABLE vb.)

-- Versiyon durumunu güncelle
UPDATE DatabaseVersion
SET Status = 'Completed'
WHERE VersionNumber = @TargetVersion;

COMMIT TRANSACTION;

END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    -- Hata durumunu kaydet
    UPDATE DatabaseVersion
    SET Status = 'Failed - ' + ERROR_MESSAGE()
    WHERE VersionNumber = @TargetVersion;

    RAISERROR (N'Versiyon yükseltme işlemi başarısız oldu.', 16, 1);
END CATCH
END')
END
GO

-- Veritabanı geri alma prosedürü
IF NOT EXISTS (SELECT * FROM sys.procedures WHERE name = 'sp_RollbackDatabase')
BEGIN
    EXEC('CREATE PROCEDURE sp_RollbackDatabase
        @TargetVersion VARCHAR(20)
    AS
    BEGIN
        SET NOCOUNT ON;

        BEGIN TRY
            BEGIN TRANSACTION;

            -- Mevcut versiyonu kontrol et
            DECLARE @CurrentVersion VARCHAR(20);
            SELECT TOP 1 @CurrentVersion = VersionNumber
            FROM DatabaseVersion

```

```

ORDER BY VersionID DESC;

IF @CurrentVersion <= @TargetVersion
BEGIN
    RAISERROR (N'Geri alma versiyonu mevcut versiyondan büyük veya eşit
olamaz.', 16, 1);
    RETURN;
END

-- Geri alma işlemi kaydı
INSERT INTO DatabaseVersion (VersionNumber, Description, Status)
VALUES (@TargetVersion, 'Rollback from ' + @CurrentVersion, 'In
Progress');

-- Burada geri alma işlemleri gerçekleştirilir
-- (Örnek: DROP TABLE, ALTER TABLE vb.)

-- Versiyon durumunu güncelle
UPDATE DatabaseVersion
SET Status = 'Completed'
WHERE VersionNumber = @TargetVersion;

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    -- Hata durumunu kaydet
    UPDATE DatabaseVersion
    SET Status = 'Rollback Failed: ' + ERROR_MESSAGE()
    WHERE VersionNumber = @TargetVersion;

    RAISERROR (N'Geri alma işlemi başarısız oldu.', 16, 1);
END CATCH
END')
END
GO

-- Versiyon geçmişini görüntüleme prosedürü
IF NOT EXISTS (SELECT * FROM sys.procedures WHERE name = 'sp_GetVersionHistory')
BEGIN

```

```

EXEC('CREATE PROCEDURE sp_GetVersionHistory
AS
BEGIN
    SELECT
        VersionNumber,
        AppliedDate,
        Description,
        Status
    FROM DatabaseVersion
    ORDER BY VersionID DESC;
END')
END
GO

```

#### 4. Versiyon Yükseltme Örneği

`4\_upgrade\_v2.sql` dosyasında 2.0.0 versiyonuna geçiş yapıyoruz. Bu aşamada:

- Customers tablosuna yeni kolonlar ekliyoruz
- Orders tablosunu oluşturuyoruz
- Versiyon bilgisini güncelliyoruz

```

USE VersionControlDB;
GO

-- Versiyon 2.0.0'a yükseltme örneği
BEGIN TRY
    BEGIN TRANSACTION;

    -- Yeni sürüm için değişiklikleri uygula
    -- 1. Customers tablosuna yeni kolonlar ekle
    IF NOT EXISTS (SELECT * FROM sys.columns WHERE object_id = OBJECT_ID('Customers')
AND name = 'PhoneNumber')
    BEGIN
        ALTER TABLE Customers
        ADD PhoneNumber NVARCHAR(20);
    END

    IF NOT EXISTS (SELECT * FROM sys.columns WHERE object_id = OBJECT_ID('Customers')
AND name = 'CreatedDate')
    BEGIN
        ALTER TABLE Customers
        ADD CreatedDate DATETIME DEFAULT GETDATE();
    END
END TRY

```

```

END

-- 2. Orders tablosunu oluşturun (eğer yoksa)
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Orders')
BEGIN
    CREATE TABLE Orders (
        OrderID INT PRIMARY KEY IDENTITY(1,1),
        CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
        OrderDate DATETIME DEFAULT GETDATE(),
        TotalAmount DECIMAL(18,2)
    );
END

-- Versiyon bilgisini güncelle
EXEC sp_UpgradeDatabase '2.0.0', 'Added phone number to Customers and created
Orders table';

COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;

    -- Hata bilgisini kaydet
    INSERT INTO DatabaseVersion (VersionNumber, Description, Status)
    VALUES ('2.0.0', 'Upgrade failed: ' + ERROR_MESSAGE(), 'Failed');

    RAISERROR (N'Versiyon yükseltme işlemi başarısız oldu.', 16, 1);
END CATCH;

```

## 5. Test ve Doğrulama

Son olarak `5\_test\_commands.sql` dosyası ile sistemin düzgün çalıştığını test ediyoruz. Bu testler:

- Örnek veri ekleme
- Versiyon geçmişi kontrolü
- Şema değişikliklerini görüntüleme
- Örnek raporlar oluşturma

```

USE VersionControlDB;
GO

```



```

-- 1. Versiyon geçmişini kontrol et
EXEC sp_GetVersionHistory;

-- 2. Şema değişikliklerini kontrol et
SELECT TOP 10 *
FROM SchemaChanges
ORDER BY ChangeDate DESC;

-- 3. Test verisi ekle
INSERT INTO Customers (FirstName, LastName, Email, PhoneNumber)
VALUES
    ('Ahmet', 'Yılmaz', 'ahmet@email.com', '5551234567'),
    ('Ayşe', 'Demir', 'ayse@email.com', '5557654321');

-- 4. Test siparişleri ekle
INSERT INTO Orders (CustomerID, TotalAmount)
VALUES
    (1, 150.50),
    (1, 75.25),
    (2, 225.00);

-- 5. Test sorguları
-- 5.1. Müşteri ve sipariş bilgilerini birleştirerek göster
SELECT
    c.CustomerID,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    c.Email,
    c.PhoneNumber,
    COUNT(o.OrderID) AS TotalOrders,
    SUM(o.TotalAmount) AS TotalSpent
FROM Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY
    c.CustomerID,
    c.FirstName,
    c.LastName,
    c.Email,
    c.PhoneNumber;

-- 5.2. Son 5 siparişi göster
SELECT TOP 5
    o.OrderID,
    c.FirstName + ' ' + c.LastName AS CustomerName,
    o.OrderDate,
    o.TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
ORDER BY o.OrderDate DESC;

```

```
-- 6. Geri alma testi (DİKKAT: Bu komutu sadece test ortamında çalıştırın!)
-- EXEC sp_RollbackDatabase '1.0.0';

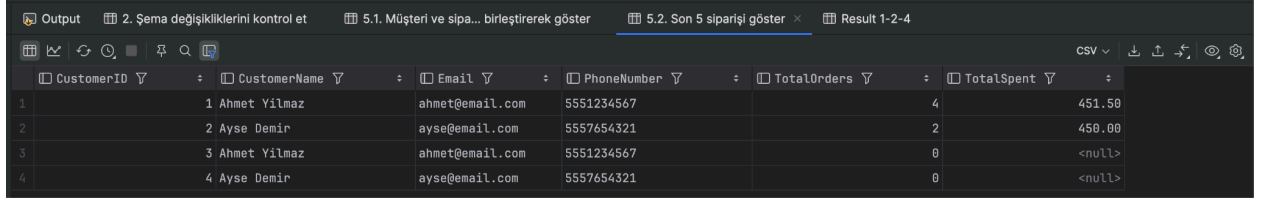
-- 7. Hata durumu testi
BEGIN TRY
    -- Kasıtlı hata oluşturun (duplicate primary key)
    INSERT INTO Customers (CustomerID, FirstName, LastName, Email)
    VALUES (1, 'Test', 'User', 'test@email.com');
END TRY
BEGIN CATCH
    PRINT 'Beklenen hata oluştu: ' + ERROR_MESSAGE();
END CATCH;

-- 8. Versiyon kontrolü
DECLARE @CurrentVersion VARCHAR(20);
SELECT TOP 1 @CurrentVersion = VersionNumber
FROM DatabaseVersion
ORDER BY VersionID DESC;

IF @CurrentVersion = '2.0.0'
    PRINT 'Versiyon yükseltme başarılı: ' + @CurrentVersion;
ELSE
    PRINT 'Versiyon yükseltme başarısız. Mevcut versiyon: ' + @CurrentVersion;
```

	VersionNumber	AppliedDate	Description	Status
1	4.0.0	2025-05-29 17:16:10.093	Added phone number to Customers and created Orders table	Completed
2	2.0.0	2025-05-29 17:15:53.533	Upgrade failed: String or binary data would be truncated in t...	Failed
3	2.0.0	2025-05-29 16:54:35.027	Upgrade failed: String or binary data would be truncated in t...	Failed
4	1.0.0	2025-05-29 16:38:16.317	Rollback from 2.0.0	Completed
5	2.0.0	2025-05-29 16:37:44.527	Added phone number to Customers and created Orders table	Completed
6	1.0.0	2025-05-29 16:36:20.610	Initial database version	Completed

	EventType	ObjectName	ObjectType	SQLCommand	LoginName	ChangeDate
1	5 CREATE_TABLE	Orders	TABLE	CREATE TABLE Orders (OrderID INT PRIMARY KEY IDENTIT...	sa	2025-05-29 16:37:44
2	4 ALTER_TABLE	Customers	TABLE	ALTER TABLE Customers ADD PhoneNumber NVARCHAR(20...	sa	2025-05-29 16:37:44
3	3 CREATE_PROCEDURE	sp_GetVersionHistory	PROCEDURE	-- Versiyon geçmişini görüntüleme prosedürü:CREATE PROCEDURE ...	sa	2025-05-29 16:37:37
4	2 CREATE_PROCEDURE	sp_RollbackDatabase	PROCEDURE	-- Veritabanı geri alma prosedürü:CREATE PROCEDURE sp_Rollbac...	sa	2025-05-29 16:37:37
5	1 CREATE_PROCEDURE	sp_UpgradeDatabase	PROCEDURE	-- Veritabanı versiyonunu yükseltme prosedürü:CREATE PROCEDUR...	sa	2025-05-29 16:37:37



The screenshot shows a database application window with multiple tabs. The active tab is '5.2. Son 5 siparişi göster'. Below the tabs, there is a table with the following columns: CustomerID, CustomerName, Email, PhoneNumber, TotalOrders, and TotalSpent. The table contains 4 rows of data.

	CustomerID	CustomerName	Email	PhoneNumber	TotalOrders	TotalSpent
1	1	Ahmet Yılmaz	ahmet@email.com	5551234567	4	451.50
2	2	Ayşe Demir	ayse@email.com	5557654321	2	450.00
3	3	Ahmet Yılmaz	ahmet@email.com	5551234567	0	<null>
4	4	Ayşe Demir	ayse@email.com	5557654321	0	<null>



The screenshot shows a database application window with multiple tabs. The active tab is 'Result 1-2-4'. Below the tabs, there is a table with the following columns: OrderID, CustomerName, OrderDate, and TotalAmount. The table contains 5 rows of data.

	OrderID	CustomerName	OrderDate	TotalAmount
1	4	Ahmet Yılmaz	2025-05-29 17:16:40.350	150.50
2	5	Ahmet Yılmaz	2025-05-29 17:16:40.350	75.25
3	6	Ayşe Demir	2025-05-29 17:16:40.350	225.00
4	1	Ahmet Yılmaz	2025-05-29 16:54:43.840	150.50
5	2	Ahmet Yılmaz	2025-05-29 16:54:43.840	75.25

Bu proje sayesinde:

- Veritabanı değişikliklerini kontrollü yapabiliyoruz
- Her değişikliği kayıt altına alıyoruz
- Hata durumunda geri dönebiliyoruz
- Kim, ne zaman, ne değiştirmiş, hepsini görebiliyoruz

Özellikle dikkat ettiğimiz noktalar:

1. Her değişikliği transaction içinde yapıyoruz
2. Hata kontrolü yapıyoruz
3. Geri alma planı hazırlıyoruz
4. Tüm değişiklikleri dokümente ediyoruz

[photo]

Son olarak, tüm versiyon geçmişini ve şema değişikliklerini gösteren bir özet ekran görüntüsü ekleyin. Bu görüntüde DatabaseVersion ve SchemaChanges tablolarının son durumları görünmeli.

Bu şekilde veritabanı versiyon yönetimini güvenli ve kontrollü bir şekilde yapabiliyoruz.