Stored Procedure ve Trigger neden kullanılır?

Stored Procedure, belirli bir işlevi yerine getirmek için önceden tanımlanmış bir veya daha fazla SQL komutu içeren bir kod parçacığıdır. Stored Procedure'lar aşağıdaki avantajlara sahiptir[1]:

- Hız: Stored Procedure'lar ilk çalıştıklarında derlenir ve sonraki çalışmalarda yeniden derlenmez. Bu da performansı artırır.

- Parametre Kullanımı: Stored Procedure'lar içine parametre alarak farklı değerlerle çalışabilir. Bu da esneklik sağlar.

- Esneklik: Stored Procedure'lar içinde başka Stored Procedure'lar çağırabilir, güncelleme ve ekleme işlemleri yapabilir, programlama dillerindeki gibi komutlar kullanabilir.

- Güvenlik: Stored Procedure'lar SQL injection saldırılarına karşı koruma sağlar. Ayrıca kritik raporlar için yetki verilebilir.

Trigger ise, bir tablo üzerinde belirli bir işlem (ekleme, güncelleme, silme) gerçekleştiğinde otomatik olarak tetiklenen ve başka bir işlem yapılmasını sağlayan bir kod parçacığıdır. Trigger'ların kullanım amacı aşağıdaki gibidir[2][3][4]:

- Veri Bütünlüğü: Trigger'lar sayesinde tablolardaki verilerin tutarlılığı sağlanabilir. Örneğin, bir tablodan veri silindiğinde başka bir tablodan da ilgili veri silinebilir.

- Veri Geçmişi: Trigger'lar sayesinde tablolardaki verilerin değişim geçmişi kaydedilebilir. Örneğin, bir tablodaki veri güncellendiğinde başka bir tabloya eski ve yeni değerleri yazılabilir.

- Veri Denetimi: Trigger'lar sayesinde tablolardaki verilere erişim kısıtlanabilir veya denetlenebilir. Örneğin, bir tabloya ekleme yapılması için belirli bir koşulun sağlanması gerektiği kontrol edilebilir.

Olumlu ve Olumsuz etkileri:

Stored Procedure'ın olumlu etkileri şunlardır:

- Hız: Stored Procedure'lar ilk çalıştıklarında derlenir ve sonraki çalışmalarda yeniden derlenmez. Bu da performansı artırır. Örneğin, bir tablodan veri çekmek için her seferinde aynı sorguyu çalıştırmak yerine, bu sorguyu Stored Procedure olarak tanımlayıp çağırmak daha hızlı olacaktır.

- Parametre Kullanımı: Stored Procedure'lar içine parametre alarak farklı değerlerle çalışabilir. Bu da esneklik sağlar. Örneğin, bir tablodan belirli bir kriteri sağlayan verileri çekmek için her seferinde

farklı sorgular yazmak yerine, bu kriteri parametre olarak alan bir Stored Procedure yazmak daha kolay olacaktır.

- Esneklik: Stored Procedure'lar içinde başka Stored Procedure'lar çağırabilir, güncelleme ve ekleme işlemleri yapabilir, programlama dillerindeki gibi komutlar kullanabilir. Bu da karmaşık işlemleri daha kolay yapmaya olanak sağlar. Örneğin, bir tabloya veri eklemek için hem insert hem de update komutlarını içeren bir Stored Procedure yazmak mümkündür.

- Güvenlik: Stored Procedure'lar SQL injection saldırılarına karşı koruma sağlar. Ayrıca kritik raporlar için yetki verilebilir. Bu da veri güvenliğini artırır. Örneğin, bir tabloya veri eklemek için kullanıcıdan gelen değerleri doğrudan sorguya dahil etmek yerine, bu değerleri parametre olarak alan bir Stored Procedure kullanmak daha güvenli olacaktır.

Stored Procedure'ın olumsuz etkileri şunlardır:

- Bakım Zorluğu: Stored Procedure'lar çok sayıda ve karmaşık olduğunda bakım yapmak zorlaşabilir. Örneğin, bir Stored Procedure'da yapılan değişikliğin başka Stored Procedure'lara etkisi olup olmadığını takip etmek zor olabilir.

- Taşınabilirlik Sorunu: Stored Procedure'lar veritabanına bağımlı olduğundan farklı veritabanları arasında taşınması zor olabilir. Örneğin, SQL Server'da yazılan bir Stored Procedure'ı Oracle'a taşımak için uyumlu hale getirmek gerekebilir.

Trigger'ın olumlu etkileri şunlardır:

- Veri Bütünlüğü: Trigger'lar sayesinde tablolardaki verilerin tutarlılığı sağlanabilir. Örneğin, bir tablodan veri silindiğinde başka bir tablodan da ilgili veri silinebilir.

- Veri Geçmişi: Trigger'lar sayesinde tablolardaki verilerin değişim geçmişi kaydedilebilir. Örneğin, bir tablodaki veri güncellendiğinde başka bir tabloya eski ve yeni değerleri yazılabilir.

- Veri Denetimi: Trigger'lar sayesinde tablolardaki verilere erişim kısıtlanabilir veya denetlenebilir. Örneğin, bir tabloya ekleme yapılması için belirli bir koşulun sağlanması gerektiği kontrol edilebilir.

Trigger'ın olumsuz etkileri şunlardır:

- Performans Düşüklüğü: Trigger'lar her işlemde tetiklendiği için performansı düşürebilir. Örneğin, bir tabloya çok sayıda veri eklendiğinde her veri için tetiklenen bir Trigger olması işlem süresini uzatabilir.

- Bakım Zorluğu: Trigger'lar çok sayıda ve karmaşık olduğunda bakım yapmak zorlaşabilir. Örneğin, bir Trigger'da yapılan değişikliğin başka Trigger'lara etkisi olup olmadığını takip etmek zor olabilir.

- Taşınabilirlik Sorunu: Trigger'lar veritabanına bağımlı olduğundan farklı veritabanları arasında taşınması zor olabilir. Örneğin, SQL Server'da yazılan bir Trigger'ı Oracle'a taşımak için uyumlu hale getirmek gerekebilir.

Python ile veri tabanı bağlantısı yapıldıktan sonra stored procedure ya da trigger kullanıldığı durumda python içerisinde bunu nasıl çağırabiliriz/nasıl bir mantık kurulabilir?

Veri tabanı bağlantısını oluşturun: Python'da veri tabanı bağlantısı oluşturmak için ilgili veri tabanı sürücüsünü kullanmanız gerekmektedir. Bu, genellikle veri tabanı bağlantı dizesi, kullanıcı adı ve şifre gibi bilgileri içerir.

Bağlantı üzerinde bir imleç oluşturun: Bağlantı üzerinde SQL ifadelerini yürütmek için bir imleç (cursor) oluşturmanız gerekebilir. Bu imleç, veri tabanına sorgular göndermek ve sonuçları almak için kullanılacaktır.

Stored procedure veya trigger'ı çağırın: İşlem yapmak istediğiniz stored procedure veya trigger'ın adını belirleyin ve imleç üzerinden bu adı kullanarak çağırın. Örneğin, PostgreSQL'de bir stored procedure çağırmak için "CALL" veya "EXECUTE" ifadesini kullanabilirsiniz.

Gerekli parametreleri ve değerleri belirtin: Eğer stored procedure veya trigger parametre alıyorsa, bu parametreleri belirtmeniz gerekmektedir. Parametreler, SQL ifadesinde yer tutucular veya isimlendirilmiş parametreler olarak kullanılabilir ve bu parametrelere değerler atanmalıdır.

Sonuçları işleyin: Stored procedure veya trigger'ın geri dönüş değerleri veya sonuçları varsa, bu sonuçları imleç üzerinden alabilir ve Python içinde işleyebilirsiniz.

Örnek:

```
CREATE PROCEDURE USER_LOGIN_SP
(
 @Username VARCHAR(50),
 @Pass VARCHAR(50)
) AS
 BEGIN
Select * from ApplicationUsers
WHERE Username = @Username AND Password = @Pass
END
 EXEC USER_LOGIN_SP 'Begüm erva ' '8520'
```