**Stored Procedure ve Trigger neden kullanılır?**

Stored procedure ve trigger, veritabanı yönetim sistemlerinde kullanılan önemli araçlardır. İşlevsel olarak farklı amaçlara hizmet ederler.

Stored procedure (saklanmış işlem), bir veritabanı yönetim sistemindeki önceden tanımlanmış bir dizi SQL ifadesidir. Genellikle bir isimle tanımlanır ve bu isimle çağrılabilir. Stored procedure'lar, veritabanında tekrar tekrar kullanılabilen ve performans avantajı sağlayabilen işlemleri gerçekleştirmek için kullanılır. Aşağıda stored procedure'ların bazı kullanım alanları verilmiştir:

Veri manipülasyonu: Veritabanındaki verileri ekleme, güncelleme, silme veya sorgulama gibi işlemleri gerçekleştirmek için kullanılabilirler. Özellikle karmaşık ve tekrar eden işlemleri basitleştirebilir ve tekrar kullanılabilir hale getirebilirler.

Güvenlik: Stored procedure'lar, veritabanı üzerindeki yetkilendirme ve erişim kontrollerini uygulamak için kullanılabilir. Sadece belirli kullanıcılara veya rollerin yetkisine sahip olan kullanıcıların stored procedure'ları çalıştırmasına izin verilebilir.

Performans iyileştirmesi: Stored procedure'lar, veritabanı sorgularının önceden derlenmesini ve yürütme planlarının önceden oluşturulmasını sağlayarak performansı artırabilir. Aynı sorgunun birden fazla kez çalıştırılması gerektiğinde sorgu maliyetini azaltabilir.

Trigger (tetikleyici), veritabanında belirli bir olayın gerçekleşmesi durumunda otomatik olarak çalıştırılan bir veritabanı nesnesidir. Trigger'lar, bir tabloda yapılan bir eylem (örneğin, veri ekleme, güncelleme veya silme) gerçekleştiğinde belirli bir tepkiyi tetiklemek için kullanılır. Aşağıda trigger'ların bazı kullanım alanları verilmiştir:

Veri bütünlüğü: Trigger'lar, veri bütünlüğünü korumak için kullanılabilir. Örneğin, bir tabloya yeni bir kayıt eklendiğinde veya bir kayıt güncellendiğinde, ilgili diğer tablolardaki verileri otomatik olarak güncelleyebilir.

Veri denetimi: Trigger'lar, veritabanında belirli kuralları uygulamak için kullanılabilir. Örneğin, bir tabloya yapılan bir ekleme veya güncelleme işlemi öncesi belirli koşulları kontrol etmek ve işlemi engellemek veya değiştirmek için kullanılabilir.

Audit izleme: Trigger'lar, veritabanında yapılan değişiklikleri izlemek ve loglamak için kullanılabilir. Örneğin, bir tabloya yeni bir kayıt eklendiğinde veya bir kayıt güncellendiğinde, bu değişiklikleri bir audit tablosunda kaydedebilir.

Stored procedure'lar ve trigger'lar, veritabanı yönetim sistemi tarafından sağlanan işlevsellikleri kullanarak veritabanı tasarımı, yönetimi ve güvenliği açısından önemli bir rol oynarlar. Bu araçları kullanmak, veritabanı işlemlerini daha etkili ve güvenli hale getirebilir.

**Olumlu etkiler:**

Performans iyileştirmesi: Stored procedure'lar, veritabanı sorgularının önceden derlenmesini ve yürütme planlarının önceden oluşturulmasını sağlayarak performansı artırabilir. Bu, tekrar tekrar çalıştırılan sorguların daha hızlı yürütülmesini sağlayabilir.

Tekrar kullanılabilirlik: Stored procedure'lar, tekrar eden işlemleri tek bir yerde tanımlayarak tekrar kullanılabilirlik sağlar. Bu, kodun tekrar yazılmasını önler ve geliştirme sürecini hızlandırır.

Güvenlik: Stored procedure'lar, veritabanı üzerinde yetkilendirme ve erişim kontrollerini uygulamak için kullanılabilir. Yalnızca belirli kullanıcılar veya rollerin yetkisine sahip olan kullanıcıların stored procedure'ları çalıştırmasına izin verilebilir.

Veri bütünlüğü ve denetimi: Trigger'lar, veri bütünlüğünü korumak ve veritabanında belirli kuralları uygulamak için kullanılabilir. Bu, yanlış veya tutarsız veri girişlerini önleyebilir ve veri bütünlüğünü sağlayabilir.

Audit izleme: Trigger'lar, veritabanında yapılan değişiklikleri izlemek ve loglamak için kullanılabilir. Bu, izleme ve denetim amaçlarıyla veri değişikliklerini kaydetmeyi sağlar.

**Olumsuz etkiler:**

Karmaşıklık: Stored procedure ve trigger'ların karmaşık yapısı, geliştirme ve bakım süreçlerini karmaşıklaştırabilir. Uygun şekilde tasarlanmadığında, kodun anlaşılması ve hataların ayıklanması zorlaşabilir.

İşlem kontrolü: Stored procedure'lar ve trigger'lar, veritabanı yönetimi için daha fazla işlem kontrolü sağlar. Bununla birlikte, gereksiz veya yanlış kullanıldığında, işlem kontrolü karmaşıklaşabilir ve gereksiz kısıtlamalara veya yanlış sonuçlara yol açabilir.

Bakım zorlukları: Stored procedure'lar ve trigger'lar, güncelleme veya değişiklikler gerektiğinde bakım süreçlerini karmaşıklaştırabilir. Birden fazla stored procedure veya trigger arasındaki etkileşimlerin yönetimi zaman alabilir ve hatalara yol açabilir.

Yetersiz performans optimizasyonu: Yanlış kullanıldığında, stored procedure'lar ve trigger'lar performans sorunlarına yol açabilir. Örneğin, gereksiz sorguların veya işlemlerin tetiklenmesi performansı olumsuz etkileyebilir.

**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.

**Tabloya yeni bir müşteri eklemek için bir Stored Procedure örneği :**

-- Stored procedure oluşturma

CREATE PROCEDURE AddCustomer

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@Email VARCHAR(100)

AS

BEGIN

INSERT INTO Customers (FirstName, LastName, Email)

VALUES (@FirstName, @LastName, @Email)

END

**Trigger örneği:**

Aşağıda, bir trigger örneği verilmiştir. Bu örnekte, Orders tablosuna yeni bir sipariş eklendiğinde, ilgili Customer tablosunda müşterinin toplam sipariş sayısı güncellenmektedir.

-- Trigger oluşturma

CREATE TRIGGER UpdateOrderCount

ON Orders

AFTER INSERT

AS

BEGIN

-- Yeni eklenen siparişin müşteri kimliğini alın

DECLARE @CustomerId INT

SELECT @CustomerId = CustomerId FROM inserted

-- Müşterinin toplam sipariş sayısını güncelle

UPDATE Customers

SET OrderCount = OrderCount + 1

WHERE CustomerId = @CustomerId

END

**ÖRNEK:**

import psycopg2

# Veri tabanı bağlantısı oluşturma

conn = psycopg2.connect(database="your\_database", user="your\_user", password="your\_password", host="your\_host", port="your\_port")

# Bağlantı üzerinde bir imleç oluşturma

cursor = conn.cursor()

# Stored procedure veya trigger'ı çağırma

cursor.callproc('your\_stored\_procedure', [param1, param2, ...])

# Sonuçları alma

results = cursor.fetchall()

# Bağlantıyı kapatma

cursor.close()

conn.close()

Yukarıdaki örnekte PostgreSQL veri tabanı için psycopg2 kütüphanesi kullanılmıştır. Veri tabanı sürücüsü veya ORM kütüphanesi, kullandığınız veri tabanı sistemine bağlı olarak değişebilir. Ayrıca, kullanılan veri tabanı sistemine özgü belirli belgeleri ve örnekleri incelemek de faydalı olabilir.