Stored Procedure Neden Kullanılır?

* Veritabanı İş Mantığını Merkezileştirme: Stored procedure'lar, veritabanı iş mantığını birleştirerek ve merkezi olarak saklayarak veritabanı uygulamalarını düzenler. Bu, aynı iş mantığını gerektiren farklı uygulama bileşenlerinin ayrı ayrı kodlanmasını önler ve tekrar kullanılabilirliği artırır.
* Performans ve Optimizasyon: Stored procedure'lar, veritabanı işlemlerini önceden derleyip saklar ve tekrar kullanımı kolaylaştırır. Bu, veritabanı sorgularının yeniden derlenmesi gerekliliğini ortadan kaldırır ve sorgu performansını artırabilir. Stored procedure'lar, karmaşık sorguları optimize etmek ve veritabanı kaynaklarını daha verimli kullanmak için de kullanılabilir.
* Güvenlik ve Erişim Kontrolü: Stored procedure'lar, veritabanı erişimini kontrol etmek ve güvenlik politikalarını uygulamak için kullanılabilir. Stored procedure'lar, sorguları parametrelerle çalıştırarak güvenlik açıklarını azaltabilir ve veritabanının hassas verilere erişimini kontrol altında tutabilir.
* Karmaşık İşlemleri Yönetme: Stored procedure'lar, veritabanında karmaşık işlemleri gerçekleştirmek için kullanılabilir. Birçok sorguyu ve veri manipülasyonunu içeren işlemleri tek bir işlem olarak gruplayarak, veritabanında veri bütünlüğünü sağlamak ve tutarlılığı korumak için kullanılabilir.
* Uygulama Katmanından Bağımsızlık: Stored procedure'lar, veritabanında işlemlerin tanımlanmasını sağlar ve böylece uygulama katmanının doğrudan SQL sorgularını içermesini önler. Bu, veritabanı şemasının değiştirilmesi veya optimize edilmesi gerektiğinde uygulama katmanını etkilemeden değişiklik yapmayı kolaylaştırır.

Stored Procedure Olumlu Yönleri

* Performans ve Veritabanı Optimizasyonu: Stored procedure'lar, veritabanı işlemlerini önceden derleyip saklayarak performansı artırır. Stored procedure'lar, veritabanı sunucusunda önbelleğe alınır ve tekrar kullanıma hazır hale getirilir, böylece sorgu yürütme süresi azalır. Ayrıca, stored procedure'lar, karmaşık işlemleri optimize etmek ve veritabanı kaynaklarını daha verimli kullanmak için indeksler ve diğer optimizasyon tekniklerini kullanabilir.
* Güvenlik ve Erişim Kontrolü: Stored procedure'lar, veritabanı erişimini kontrol etmek ve güvenlik politikalarını uygulamak için kullanılabilir. Stored procedure'lar, parametrelerle çalıştırılan sorgular aracılığıyla güvenlik açıklarını azaltır ve yetkilendirme kontrollerini yapar. Kullanıcılar, doğrudan tablolara veya sorgulara erişim yerine, sadece stored procedure'ları kullanarak veritabanına erişir ve işlem yapar.
* Tekrar Kullanılabilirlik: Stored procedure'lar, aynı iş mantığının tekrar tekrar kodlanmasını önleyerek ve merkezi olarak saklanarak tekrar kullanılabilirliği artırır. Stored procedure'lar, bir kez yazıldıktan sonra farklı uygulama bileşenlerinde veya farklı sorgularda kullanılabilir. Bu, kod tekrarını azaltır, geliştirme sürecini hızlandırır ve bakım maliyetlerini düşürür.
* Veri Bütünlüğü ve Tutarlılık: Stored procedure'lar, veritabanında karmaşık işlemleri gerçekleştirmek için kullanılırken veri bütünlüğünü sağlamak ve tutarlılığı korumak için kullanılabilir. Birden fazla sorgu ve veri manipülasyonunu içeren işlemler tek bir işlem olarak gruplandırılır ve bu işlemler atomik bir şekilde çalıştırılır. Bu, veritabanında veri bütünlüğünü korumaya yardımcı olur.
* Uygulama Katmanından Bağımsızlık: Stored procedure'lar, veritabanında işlemlerin tanımlanmasını sağlayarak uygulama katmanını SQL sorgularından bağımsız hale getirir. Bu, veritabanı şemasının değiştirilmesi veya optimize edilmesi gerektiğinde uygulama katmanını etkilemeden değişiklik yapmayı kolaylaştırır. Stored procedure'lar, veritabanı yönetimi ve iş mantığı ile ilgili değişiklikleri sadece stored procedure kodlarında yaparak uygulamaları güncellemeyi kolaylaştırır.

Stored Procedure Olumsuz Yönleri

* Karmaşıklık ve Bakım Zorlukları: Stored procedure'lar, karmaşık bir mantık içerebilir ve birden çok sorgu ve işlemi bir araya getirebilir. Bu durum, stored procedure'ların anlaşılmasını ve sürdürülmesini zorlaştırabilir. Bakım ve güncelleme süreçleri karmaşık olabilir ve doğru şekilde yönetilmezse hatalara yol açabilir.
* Platform Bağımlılığı: Stored procedure'lar, veritabanı yönetim sistemine özgü olabilir. Bir stored procedure, bir veritabanı yönetim sisteminde çalışırken başka bir sisteme taşındığında uyumluluk sorunları ortaya çıkabilir. Bu durum, veritabanı taşınabilirliğini ve sistem bağımsızlığını olumsuz etkileyebilir.
* Teknoloji Bağımlılığı: Stored procedure'lar genellikle SQL tabanlıdır ve veritabanı yönetim sistemi tarafından desteklenen programlama dilini kullanır. Bu durum, geliştiricilerin belirli bir teknolojiye bağlı kalmasına ve farklı diller veya araçlar kullanma esnekliğini kısıtlayabilir.
* Güncelleme Sorunları: Stored procedure'lar, her kullanıcı tarafından doğrudan değiştirilemez ve güncellenemez. Bu durum, stored procedure'larda yapılan hataları düzeltmeyi veya performans iyileştirmeleri yapmayı zorlaştırabilir. Stored procedure'ların güncellenmesi ve sürüm kontrolü, dikkatli bir şekilde yönetilmelidir.
* Test Edilebilirlik: Stored procedure'lar, genellikle birim testlerin veya otomatik test süreçlerinin zor olduğu bir ortamda çalışır. Stored procedure'lar doğrudan veritabanında çalıştığından, veritabanı durumunu ve test verilerini yönetmek karmaşık olabilir ve test senaryolarının oluşturulması zorlaşabilir.
* Güvenlik Riskleri: Stored procedure'lar, kullanıcıların veritabanına doğrudan erişimini sağlar. Yanlış veya kötü amaçlı stored procedure'lar, veritabanı güvenliğini tehlikeye atabilir veya kötü niyetli faaliyetlere yol açabilir. Bu nedenle, stored procedure'ların güvenlik açıklarına karşı dikkatlice tasarlanması ve denetlenmesi önemlidir.

Stored Procedure Örneği

Bir "customers" tablosunda yeni bir müşteri eklediğimizde, müşterinin toplam harcamasını otomatik olarak hesaplayan bir stored procedure oluşturacağız.

* Stored Procedure Oluşturma:

CREATE PROCEDURE CalculateTotalSpent  
 @customer\_id INT,  
 @order\_amount DECIMAL(10, 2)  
AS  
BEGIN  
 UPDATE customers  
 SET total\_spent = total\_spent + @order\_amount  
 WHERE customer\_id = @customer\_id;

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

Python'da veritabanı bağlantısı yapıldıktan sonra stored procedure'ları çağırmak için veritabanı bağlantısı üzerinden bir cursor oluşturarak SQL sorgularını çalıştırmanız gerekmektedir.

import pyodbc

# Veritabanı bağlantısı kur

conn = pyodbc.connect('Driver={SQL Server};Server=my\_server;Database=my\_database;Trusted\_Connection=yes;')

cursor = conn.cursor()

# Stored procedure'u çağırma

customer\_id = 1

order\_amount = 100.00

# Parametreli sorgu oluşturma

query = f"EXEC CalculateTotalSpent @customer\_id = ?, @order\_amount = ?"

# Parametre değerlerini göndererek stored procedure'u çağırma

cursor.execute(query, customer\_id, order\_amount)

# Veritabanı değişikliklerini kaydet

conn.commit()

# Veritabanı bağlantısını kapat

conn.close()

Yukarıdaki örnekte, **pyodbc** kütüphanesini kullanarak SQL Server veritabanına bağlanıyoruz. **cursor** nesnesi üzerinden SQL sorgularını çalıştırabiliyoruz. Stored procedure'u çağırmak için bir parametreli sorgu oluşturuyoruz ve **cursor.execute()** yöntemini kullanarak sorguyu çalıştırıyoruz. Parametre değerlerini **cursor.execute()** yöntemine göndererek stored procedure'u çağırıyoruz.

Daha önce örnekte olduğu gibi, **conn.commit()** ile veritabanı değişikliklerini kaydedebilir ve **conn.close()** ile veritabanı bağlantısını kapatabilirsiniz.

Bu şekilde Python ile veritabanı bağlantısı yapıldıktan sonra stored procedure'ları çağırabilir ve ilgili işlemleri gerçekleştirebilirsiniz. Stored procedure'un geri dönüş değeri varsa, **cursor.execute()** sonucunu alarak ilgili değeri elde edebilirsiniz.