Gespeicherte Prozeduren

Was ist das?

- Benannte Sammlung von T-SQL-Anweisungen oder .NET-Code
  - Im Grunde Programm, welches auf dem Server gespeichert wird
- Akzeptiert Eingabeparameter, gibt Ausgabeparameter zurück
- Gibt einen Statuswert zurück, welcher (Miß-)Erfolg anzeigt
- Dient u.A. zur Vereinfachung von Routine-Aufgaben oder der Automatisierung
- Mit Prozeduren können Anwendungen von Änderungen an der Datenbankstruktur einer Datenbank entkoppelt werden

Entwickler

- Im Zusammenhang mit anderen Objekten stellen Prozeduren eine vertrauenswürdige Schnittstelle zur Anwendungsprogrammierung dar
- Entwickler sollten:
  - Zur Rückgabe von Daten Ansichten, Funktionen und Prozeduren verwenden
  - Mithilfe von Prozeduren Daten ändern und neue Daten hinzufügen
  - Prozedurdefinition zentral an einer Stelle ändern, anstatt den Anwendungscode zu aktualisieren

Schnittstelle nutzen – C#-Beispiel

```
SqlCommand cmd = new SqlCommand();
SqlParameter par1 = new SqlParameter("@baumid", System.Data.SqlDbType.Int);

par1.Value = fb.Baum_ID;

cmd.Parameters.Add(par1);



cmd.CommandType = System.Data.CommandType.StoredProcedure;
cmd.CommandText = "usp_LoescheFrage";
cmd.Connection = sqlconn;
cmd.Connection.Open();
cmd.ExecuteNonQuery();
cmd.Connection.Close()
```

Erstellen einer gespeicherten Prozedur

```
CREATE PROCEDURE usp_ProjekteAnzeigen(@ang int)
AS
	SELECT ang.name, ang.beruf, pro.p_name, pro.p_beschr
	FROM ang INNER JOIN pro_ang ON ang.a_nr = pro_ang.a_nr
	INNER JOIN pro on pro_ang.p_nr = pro.p_nr
	WHERE ang.a_nr = @ang;
```

- Aufruf mit:

```
EXECUTE usp_ProjekteAnzeigen @ang = 198;
```

- Alternativ EXEC statt EXECUTE verwendbar, PROC statt PROCEDURE verwendbar
- Ändern einer Prozedur mit ALTER, löschen einer Prozedur mit DROP

Rückgabe

- Prozeduren geben immer einen Statuswert zurück
  - Statuswerte können explizit zugewiesen werden
  - Standardmäßig Rückgabewert 0 bei Erfolg, != 0 bei Mißerfolg

```
ALTER PROCEDURE usp_Teilen(@zaehler int, @nenner int)
AS
BEGIN
	IF @nenner = 0
		BEGIN
		SELECT NULL AS Ergebnis
		RETURN -1
		END
	ELSE
		SELECT @zaehler / @nenner AS Ergebnis
END
```

```
DECLARE @ergebnis int;
EXEC @ergebnis = usp_Teilen @zaehler = 42, @nenner = 7;
SELECT @ergebnis AS Rückgabewert;
```

	Ergebnis
1	6

	Rückgabewert
1	0

Hinweise zur Erstellung von Prozeduren

- Qualifizieren der Objektnamen in der Prozedur
- Erstellen jeweils einer Prozedur für eine Aufgabe
- Erstellen, Testen, Fehlerbehebung
- Vermeiden des Präfixes sp_ in dem Namen von Prozeduren
- Auf konsistente Verbindungseigenschaften achten
- Minimieren der Verwendung temporärer gespeicherten Prozeduren

Weiteres zu Prozeduren

- Prozeduren mit oder ohne Übergabeparameter
  - Hohe Flexibilität
  - Aufruf mit Parametern in korrekter Reihenfolge oder als benannte Parameter

```
EXECUTE usp_Teilen 75, 25;
EXEC usp_Teilen @nenner = 25, @zaehler = 75;
```

- Mit INSERT INTO <Tabelle> EXEC usp_ProcName können Ergebnisse von Prozeduren in Tabellen gespeichert werden

```
INSERT INTO Sales.MyOrders (orderid, custid, empid, orderdate, shipcountry, freight)
EXECUTE Sales.OrdersForCountry
```

Prozeduren mit OUTPUT-Parametern

- Mit Ausgabeparametern lassen sich Werte aus einer gespeicherten Prozedur zurückgeben
  - Vergleichbar mit der Rückgabe eines Resultsets
  - Parameter werden in Prozedur und in Abfrage gekennzeichnet

```
CREATE PROC usp_Division(@zähler float,  @nenner float, @ergebnis float OUTPUT)
AS BEGIN
        SET @ergebnis = @zähler / @nenner
        RETURN @ergebnis;
END

DECLARE @lösung float;
EXEC usp_Division 10,2, @lösung OUTPUT;
SELECT @lösung;
```

Dynamisches SQL

- T-SQL-Code, der in eine Zeichenfolge assembliert, als Befehl interpretiert und ausgeführt wird
- Verwaltungs- und Programmieraufgaben lassen sich so flexibel gestalten
- Zwei Möglichkeiten für die Ausführung von dynamischem SQL-Code
  - ➢ Der EXEC-Befehl akzeptiert Zeichenfolge aber keine Übergabe von Parametern
  - ➢ Systemprozedur sp_executesql (bevorzugt) unterstützt Parameter

```
EXEC ('SELECT GETDATE();');
```

```
DECLARE @befehl2 nvarchar(max) = 'SELECT * FROM ang WHERE ang.a_nr = @par1;';
DECLARE @def nvarchar(50) = N'@par1 int';
EXEC sp_executesql @befehl2, @def,  @par1 = 205;
```