Erste Anwendungen mit SSMS

Kathegorie
SQL Server

Denise
Robinson

Ich bin Denise und aktuell Studentin der Mathematik an der Uni Mainz.
Im Rahmen meines Praktikums bei der Mainzer Datenfabrik habe ich mich mit SQL-Datenbanken und SSMS beschäftigt und konnte so einen guten Einblick in den Umgang mit den Funktionen von SSMS bekommen und Eigenschaften von SQL-Datenbanken und Tabellen kennenlernen.
Am Ende meines Praktikums habe ich ein Skript erstellt, das nach dem Starten drei Tabellen erstellt und im Anschluss verschiedene Spalten der Tabellen ausgeben kann. Hierzu erkläre ich kurz, wie ich dabei vorgegangen bin.

Zunächst erstelle ich eine neue Datenbank (Test).

Mit **DROP TABLE IF EXISTS** lösche ich meine Tabellen, falls sie vorhanden sind, damit ich jedes Mal ohne Probleme mein Skript neu ausführen kann.

```
DROP TABLE IF exists dbo.KundInnen
DROP TABLE IF exists dbo.Artikel
DROP TABLE IF exists dbo.Umsaetze
GO
```

Mit **CREATE TABLE** definiere ich drei Tabellen (KundInnen, Artikel und Umsaetze). Die Spalten der KundInnen- sowie Artikel-Tabellen enthalten jeweils eine automatisch fortlaufende ID als Integer und mit **IDENTITY(1,1)** erstellt man fortlaufende Einträge ab 1, zu denen immer 1 hinzuaddiert wird. Bei der Tabelle Umsaetze werden zunächst nur die Spalten-Namen und Datentypen vorgegeben. Dieser Vorgang wird mit den folgenden Codes ausgeführt:

```
CREATE TABLE [dbo].[KundInnen](
     [KundInnenID] [int] IDENTITY(1,1) NOT NULL,
     [Name] [varchar](255) NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[Artikel](
     [ArtikelID] [int] IDENTITY(1,1) NOT NULL,
     [Name] [varchar](255) NOT NULL,
     [Preis] [int] NOT NULL
) ON [PRIMARY]
GO
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz

info@madafa.de
madafa.de

+49 6131 3331612

```
CREATE TABLE [dbo].[Umsaetze](
     [Bewegung] [bigint] NULL,
     [ArtikelID] INT NULL,
     [KundInnenID] INT NULL,
     [Stueckzahl] INT NULL
) ON [PRIMARY]
GO
```

Mit der Anweisung **INSERT INTO** füge ich der KundInnen- sowie der Artikel-Tabelle zufällige Namen in Form einer Zeichenfolge und in der Artikel-Tabelle eine zusätzliche Spalte mit zufälligen Preisangaben hinzu.
Da ich in der **CREATE TABLE** Anweisung beider Tabellen bereits die ID vorgegebe, muss ich diese Spalte hier nicht mehr berücksichtigen und gebe mit **GO 50** vor, dass 50 zufällige Einträge generiert werden sollen, da ich jeweils 50 Einträge in meinen Tabellen haben möchte.
Die folgenden Codes definieren die zufälligen Einträge:

```
INSERT INTO [dbo].[Artikel]
          ([Name],Preis)
SELECT CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97)+
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) [SomeVarchar],

 (ABS(CHECKSUM(NEWID())) % 100001) + ((ABS(CHECKSUM(NEWID())) %
100001) * 0.00001) [SomeNumeric]
 GO 50
INSERT INTO [dbo].[KundInnen] ([Name])
SELECT CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97)+
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) +
CHAR((ABS(CHECKSUM(NEWID())) % 26) + 97) [SomeVarchar]
GO 50
```

Zum Definieren der Umsatz-Tabelle benutze ich eine Schleife, da zwei meiner Spalten aus den Tabellen KundInnen und Artikel aufgerufen werden sollen.
Die Spalten sehen dann wie folgt aus:

1. Bewegung, mit zufälligem Datum als bigint
2. ArtikelID aus der Artikel-Tabelle
3. KundInnenID aus der KundInnen-Tabelle
4. Stueckzahl mit zufälligen ganzen Zahlen zwischen 0 und 100

Mit der **DECLARE**-Anweisung deklariere ich verschiedene Variablen , denen ich im Anschluss Werte zuweisen kann, die ich für meine **WHILE**-Schleife benötige:

```
DECLARE @DateStart    DATE = '2020-01-01'
     ,@DateEnd  DATE = '2020-09-01'
DECLARE @Tempvariable DATE
DECLARE @Bewegung BIGINT
DECLARE @ArtikelID INT
DECLARE @KundInnenID INT
DECLARE @Stueckzahl INT
```

Für die Schleife setze ich einen Counter, damit ich vorgeben kann, dass sie nach 50 Wiederholungen abbricht und starte die Schleife mit BEGIN:

```
DECLARE @Counter INT
SET @Counter=0
WHILE (@Counter <=50)
BEGIN
```


Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612

Mit der **INSERT INTO**-Anweisung gebe ich nun vor, welche Spalten gefüllt werden sollen und weise mit **SELECT** und **SET** den deklarierten Variablen verschieden Werte zu. Diese sieht wie folgt aus:

```
INSERT INTO [dbo].[Umsaetze]
           (Bewegung,ArtikelID,KundInnenID,Stueckzahl)
     SELECT   @Bewegung, @ArtikelID, @KundInnenID,@Stueckzahl
     SET @Tempvariable = (SELECT DATEADD(DAY, RAND() *
DATEDIFF(DAY, @DateStart, @DateEnd), @DateStart))
     SET @Bewegung = (SELECT FORMAT(@Tempvariable, 'yyyMMdd'))
     SET @ArtikelID = (SELECT TOP 1 ArtikelID FROM Artikel ORDER BY
NEWID())
     SET @KundInnenID = (SELECT TOP 1 KundInnenID FROM KundInnen
ORDER BY NEWID())
     SET @Stueckzahl = ((ABS(CHECKSUM(NEWID())) % 100) +
((ABS(CHECKSUM(NEWID())) % 100) * 0.00001) )
```

Zu dem Counter wird bis 50 immer 1 hinzu addiert und mit **END** nach 50 Wiederholungen abgebrochen.

```
SET @Counter  = (SELECT @Counter  + 1);
END;
```

Nachdem nun alle drei Tabellen definiert sind hat man die Möglichkeit verschiedene Werte der Tabellen ausgeben zu lassen.
Dieses Beispiel zeigt eine Tabelle, die in den Spalten das Buchungsdatum, den Namen des Artikels, den Einzelpreis, den Namen der Kund*innen sowie den jeweiligen Umsatz enthält und nach den Top 10-Einträgen der Umsatz-Spalte sortiert ist.

```
SELECT TOP 10 ums.Bewegung,art.[Name],art.Preis AS Einzelpreis,kd.[Name],
ums.Stueckzahl,ums.Stueckzahl * art.Preis AS Umsatz
FROM dbo.Umsaetze ums

INNER JOIN dbo.artikel art ON
ums.ArtikelID = art.ArtikelID
INNER JOIN dbo.KundInnen kd ON
ums.KundInnenID = kd.KundInnenID
ORDER BY Umsatz DESC
```

Diese Tabelle ist bei den beschriebenen Vorgängen entstanden:

Ergebnisse | Meldungen

	Bewegung	Name	Einzelpreis	Name	Stueckzahl	Umsatz
1	20200611	gecpmqghyg	96898	hjutcmtale	97	9399106
2	20200304	btjczsnxdt	94829	ztsmkhyigs	92	8724268
3	20200128	tyltejoszc	99716	jpbmnijezs	86	8575576
4	20200411	vtpjxyrfes	74418	oaznhqilwu	97	7218546
5	20200329	gecpmqghyg	96898	ixlrprhrdi	71	6879758
6	20200128	nxnhrxkdkh	85730	awsqdxksmt	60	5143800
7	20200120	vtpjxyrfes	74418	lenqhcvvyr	64	4762752
8	20200529	mbjnnuiiir	89410	efpcupnoqq	48	4291680
9	20200630	ankraevfji	48806	dcmjfbwfqs	72	3514032
10	20200510	hwkhxngnty	99727	dcmjfbwfqs	35	3490445

Mainzer Datenfabrik GmbH
Rotterdamer Platz 2, 55120 Mainz
info@madafa.de
madafa.de
+49 6131 3331612