**Inline-Tabellenfunktion**

Gib mir alle Kunden aus der Kunde-Tabelle, deren Wohnort den Wert des übergebenen Parameters @Ort entspricht.

Benennung: KundenNachOrt(@Ort VARCHAR(20))

CREATE FUNCTION **KundenNachOrt(@Ort VARCHAR(20))**

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM Kunde WHERE Ort = @Ort

)

GO

**SELECT \* FROM KundenNachOrt('Aort')**

Hinweise:  
@Ort: Variable gleich, wie in der Quelltabelle

Wir haben bei Funktionen ähnliche STERN-Problematik, wie bei in der letzten Lektion behandelten VIEWS

--- > GLEICH erklären + Lösung, wie in Lektion 10

Natürlich können wir so eine Tabellenwert-Funktion, wie eine herkömmliche Tabelle zu anderen Datenmengen dazujoinen:

SELECT b.\*, kno.\* FROM KundenNachOrt('Musterstadt') kno

INNER JOIN Bestellung b ON kno.KundeId = b.KundeId

**Tabellenfunktionen mit mehreren Anweisungen**

CREATE FUNCTION AnzahlProdukteNachDatum(@Datum DATE)

RETURNS

@Ergebnis TABLE

(

Produkte INT, Datum Date

)

AS

BEGIN

INSERT @Ergebnis

SELECT

COUNT(bp.ProduktId),

@Datum

FROM [dbo].[BestellPosition] bp

INNER JOIN Bestellung b ON bp.BestellungId = b.BestellungId

INNER JOIN Produkt p ON bp.ProduktId = p.ProduktId

WHERE b.Zeitstempel = @Datum

RETURN

END

GO

In SQL kommen wir mit der letzteren – mit kopfgesteuerter Schleife gut aus. Kopfgesteuert bedeutet: Die Bedingung für die Ausführung der Schleife ist im oberen Bereich – „im Kopf“ definiert. Man bezeichnet sie auch „**WHILE-Schleife**“

**WHILE-SCHLEIFE**

DECLARE @Zaehler INT

SET @Zaehler=1

WHILE (@Zaehler <= 10)

BEGIN

SET @Zaehler = @Zaehler + 1

END

SELECT @Zaehler

**Mit eingeplantem Break**

DECLARE @Zaehler INT

SET @Zaehler=1

WHILE (@Zaehler <= 10)

BEGIN

SET @Zaehler = @Zaehler + 1

IF (@Zaehler = 3)

BREAK

END

SELECT @Zaehler

**Mit eingeplantem CONTNUE**

DECLARE @Zaehler INT

SET @Zaehler=1

WHILE (@Zaehler <= 10)

BEGIN

SET @Zaehler = @Zaehler + 1

IF (DATENAME(SECOND, GETDATE()) % 3 = 0)

BREAK

ELSE

CONTINUE

END

SELECT @Zaehler

**Cursor**

Ein Cursor ist ein **Datenbankobjekt**, das verwendet wird, um **zeilenweise über eine Ergebnismenge zu iterieren** (= schrittweise durchlaufen). **Normalerweise** werden SQL-Abfragen so geschrieben, dass sie alle betroffenen Zeilen **auf einmal zurückgeben** oder manipulieren. Ein Cursor hingegen erlaubt es, **jede Zeile einzeln** zu verarbeiten, was in bestimmten Situationen nützlich sein kann.

DECLARE @ProduktId INT

DECLARE @ProductName VARCHAR(50)

**DECLARE ProductCursor CURSOR FOR**

SELECT ProduktId, Name

FROM Produkt

OPEN ProductCursor

FETCH NEXT

FROM ProductCursor

INTO @ProduktId, @ProductName

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Product ID: ' + CAST(@ProduktId AS VARCHAR(10))

PRINT 'Product Name: ' + @ProductName

FETCH NEXT

FROM ProductCursor

INTO @ProduktId ,@ProductName

END

CLOSE ProductCursor

DEALLOCATE ProductCursor