



- Welche Straßen gehen durch den Microsoft-Campus?
- Welche Gesamtfläche haben alle Parks, die maximal 1 km vom Microsoft-Campus entfernt sind?

SQL Server 2008 führt für die Speicherung solcher Daten zwei neue Datentypen ein:

- Der Typ **“geography”** speichert Punkte, Linien, Polygone und Kombinationen aus diesen Objekten. Dabei wird ein Koordinatensystem mit Längen- und Breitengraden verwendet, das eine kugelförmiges Erdmodell voraussetzt. Es wird erwartet, dass dieser Datentyp für geographische Daten am häufigsten verwendet wird. Außerdem können auch korrekte Ergebnisse bei einem Ellipsoid-Erdmodell errechnet werden. Damit können Fragen beantwortet werden wie: Wie groß ist die Fläche von Indonesien? Werde ich bei meinem Flug von Seattle nach Beijing über Nordkorea fliegen? Wo kann ich meine GPS-Daten speichern?
- Der Typ **“geometry”** entspricht den OGC-Vorgaben und setzt ein "flaches" Erdmodell voraus. Er wird verwendet für Kartenmaterial, das auf ebene Projektionen aufsetzt.

In der Grafik oben könnten wir die Straßen in einer Tabelle Roads speichern:

```
Roads(name varchar(30), location geography)
```

Das bedeutet, geography ist ein Datentyp wie jeder andere.

Der Datentyp geography besitzt eine Reihe von Methoden, mit denen spezielle Informationen berechnet werden können (Achtung: die Methoden sind case-sensitiv!):

STArea	Berechnet die Fläche eines Polygons
STLength	Länge
STIntersects	Überprüft, ob sich zwei Elemente schneiden
STIntersection	Berechnet den/die Schnittpunkte zweier Elemente
STUnion	Vereinigungsmenge zweier Flächen

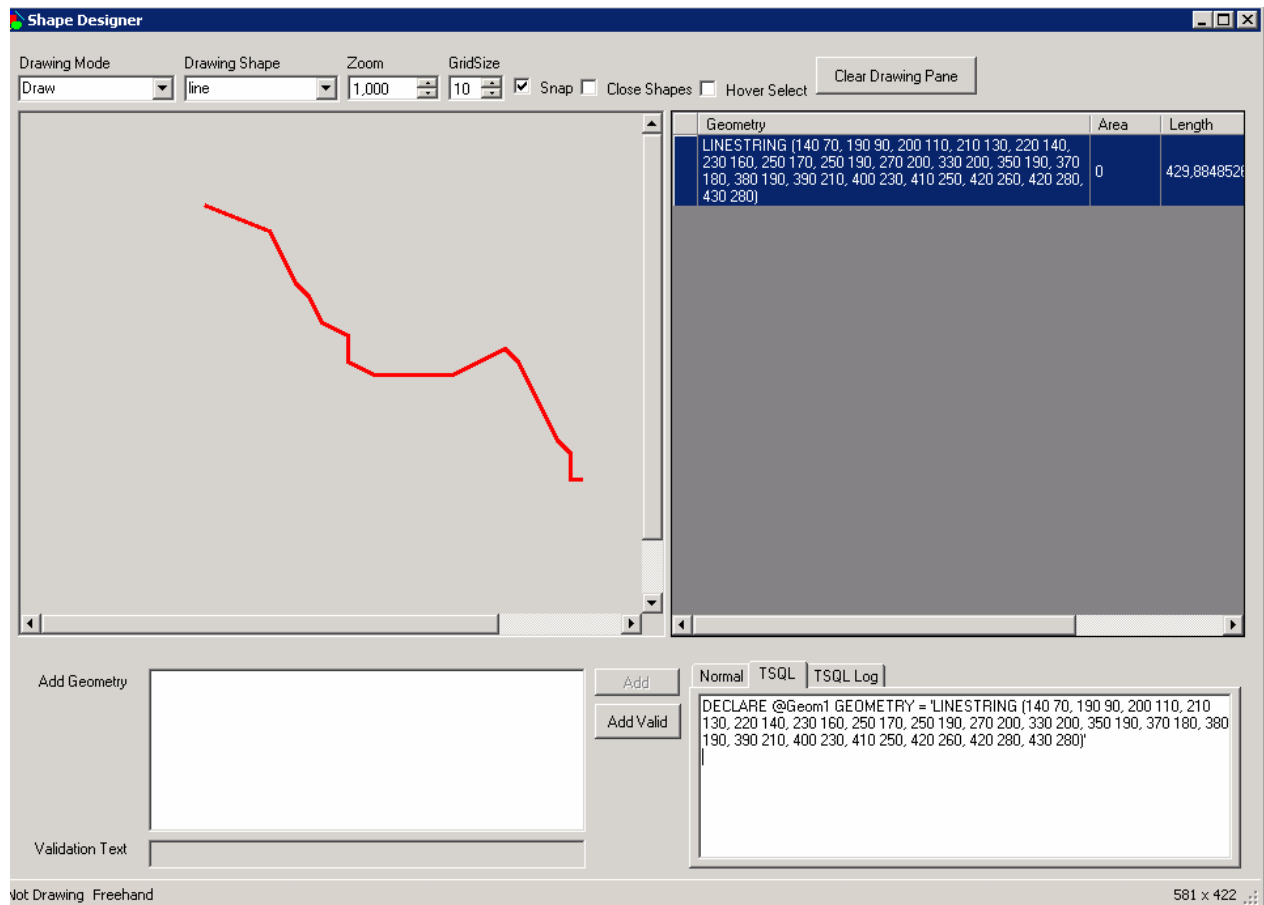
Beispiel: Wenn wir eine Variable @microsoft vom Typ geometry haben, die den Polygonzug des Microsoft-Campus enthält, können wir durch folgende Abfrage herausfinden, welche Straßen ihn schneiden:

```
SELECT name
FROM Roads
WHERE location.STIntersects(@microsoft) = 1
```

Hätten wir eine analoge Tabelle Parks, die alle Parks der USA enthält, so können wir die Frage nach der Fläche aller Parks wie folgt formulieren:

```
SELECT SUM(location.STArea())
FROM Parks
WHERE location.STDistance(@microsoft) < 1.0
```

Unterstützungstools: Spatial Designer (CodePlex-Homepage)



### Beispiel 1: Länge einer Linie

```
DECLARE @Geom1 GEOMETRY = 'LINESTRING (140 70, 190 90, 200 110, 210 130, 220 140, 230 160, 250 170, 250 190, 270 200, 330 200, 350 190, 370 180, 380 190, 390 210, 400 230, 410 250, 420 260, 420 280, 430 280)'
```

```
SELECT @Geom1.STLength()
```

Ergebnis:

429,884852692517

### Beispiel 2: Schnittpunkte zweier Linien

```
DECLARE @Geom1 GEOMETRY = 'LINESTRING (140 70, 190 90, 200 110, 210 130, 220 140, 230 160, 250 170, 250 190, 270 200, 330 200, 350 190, 370 180, 380 190, 390 210, 400 230, 410 250, 420 260, 420 280, 430 280)'
```

```
DECLARE @Geom2 GEOMETRY = 'LINESTRING (380 60, 360 60, 330 80, 320 90, 310 110, 300 120, 290 150, 280 170, 270 190, 260 210, 250 220, 220 240, 210 250, 190 260, 180 260)'
```

```
SELECT @Geom1.STIntersection(@Geom2)
```

Ergebnis:

0x00000000010C000000000A070400000000000C06840

Mit diesem Ergebnis fangen wir nichts an. Da es sich aber um SQLCLR-Datentypen handelt, können wir die Lesbarkeit des Ergebnisses wie folgt verbessern:

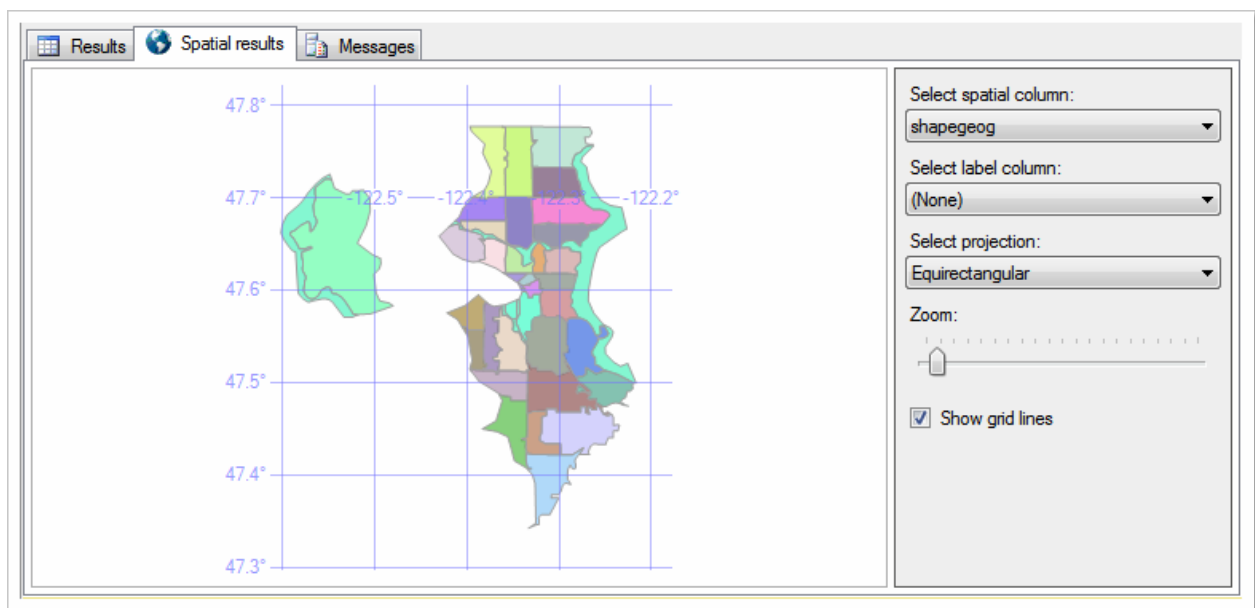
```
SELECT @Geom1.STIntersection(@Geom2).ToString()
```

Nun wird das Ergebnis mit x/y-Koordinaten ausgegeben:

POINT (266 198)

Um Spatial Data zu verstehen, muss eine Visualisierung durchgeführt werden:

	id	shapegeog
1	98101	0xE6100000010417000000C9B08A3732CF4740F03504C76595...
2	98102	0xE610000001041A000000C45A7C0A80D14740C58CF0F6209...
3	98103	0xE610000001041A000000B6BE4868CBD94740EF8E8CD5E6...
4	98104	0xE6100000010418000000400276BD4CB4740FDFFF2F519...
5	98105	0xE6100000010427000000124E0B5EF4D54740CAA48636009...
6	98106	0xE61000000104380000007C478D0931C54740312592E86595...
7	98107	0xE610000001041300000091B8C7D287D647401348895DDB9...
8	98108	0xE610000001042E000000B79C4B7155C947402DEDD45C6E...
9	98109	0xF610000001041F000000A5F78DAF3DD347404F4354F1CF9



Die Karten sind interaktiv: Sie können zoomen, Sie können die Resultate auch beschriften.

## 4 Einschränkungen (Constraints)

### 4.1 Spalten- und Tabelleneinschränkungen

Einschränkungen können Spalten- oder Tabelleneinschränkungen sein:

- Eine Spalteneinschränkung wird als Teil einer Spaltendefinition angegeben und gilt nur für diese Spalte.
- Eine Tabelleneinschränkung wird unabhängig von einer Spaltendefinition deklariert und kann für mehr als eine Spalte in einer Tabelle gelten. Tabelleneinschränkungen müssen verwendet werden, wenn mehr als eine Spalte in eine Einschränkung eingeschlossen werden muss.

### 4.2 Primary Key-Einschränkungen, Default-Einschränkungen

```
use Auftrag;
create table dbo.tArtikel
( ArtNr int identity(1,1),
  ArtBez nvarchar(50) NOT NULL,
  Einzelpreis money NOT NULL constraint DF_tArtikel_Einzelpreis default (0.0),
  constraint PK_tArtikel_ArtNr primary key nonclustered
  (ArtNr ASC) with (ignore_dup_key = off)
);
```

Wenn in einer Tabelle z. B. zwei oder mehr Spalten für den Primärschlüssel verwendet werden, müssen Sie eine Tabelleneinschränkung verwenden, um beide Spalten in den Primärschlüssel einzuschließen. Stellen Sie sich eine Tabelle vor, die Ereignisse aufzeichnet, die für einen Computer in einer Fabrik eintreten. Nehmen Sie weiterhin an, dass unterschiedliche Ereignistypen gleichzeitig eintreten können, dass jedoch nie zwei Ereignisse desselben Typs gleichzeitig eintreten. Dieser Sachverhalt kann in der Tabelle erzwungen werden, indem Sie die **type**- und die **time**-Spalte in einen Primärschlüssel einschließen, der zwei Spalten umfasst.

```
CREATE TABLE factory_process
( event_type int,
  event_time datetime,
  event_site char(50),
  event_desc char(1024),
  CONSTRAINT event_key PRIMARY KEY (event_type, event_time) )

CREATE TABLE tPLZ
(
  PLZ char(5) not NULL,
  Ort varchar(50) not NULL
  CONSTRAINT PK_tPLZ PRIMARY KEY (PLZ, Ort)
);
```

### 4.3 Foreign Key-Constraints

```
alter table dbo.tAuftrag
  add MitarbeiterNr int null;

alter table dbo.tAuftrag
  with check -- vorhandene Datensätze werden überprüft
```

```
add constraint FK_MitarbeiterNr_tMitarbeiter
foreign key (MitarbeiterNr)
references dbo.tMitarbeiter (MitarbeiterNr);

alter table dbo.tAuftrag -- beginnen Sie immer mit der Detailtabelle,
                        -- das ist die Tabelle, die den Fremdschlüssel enthält
with check              -- vorhandene Datensätze werden überprüft
add constraint FK_KdNr_tKunden
foreign key (KdNr)      -- Fremdschlüssel
references dbo.tKunden (KdNr) -- Bezug auf Primärschlüssel der anderen Tabelle
on update cascade;    -- Kaskadierungsoptionen
```

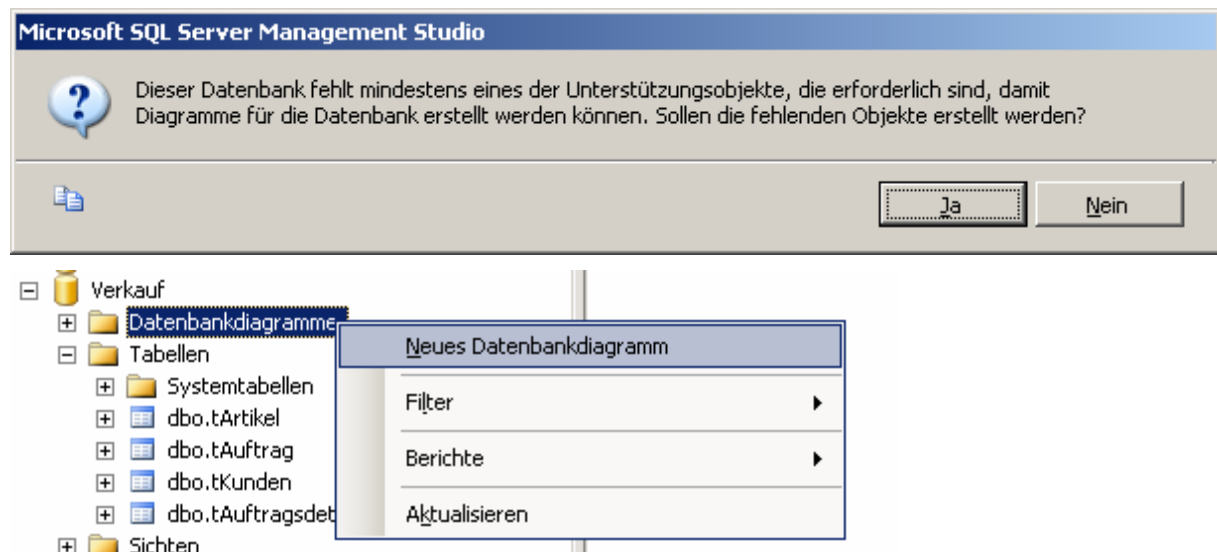
## 4.4 Fremdschlüsseleinschränkungen in Diagrammen erstellen

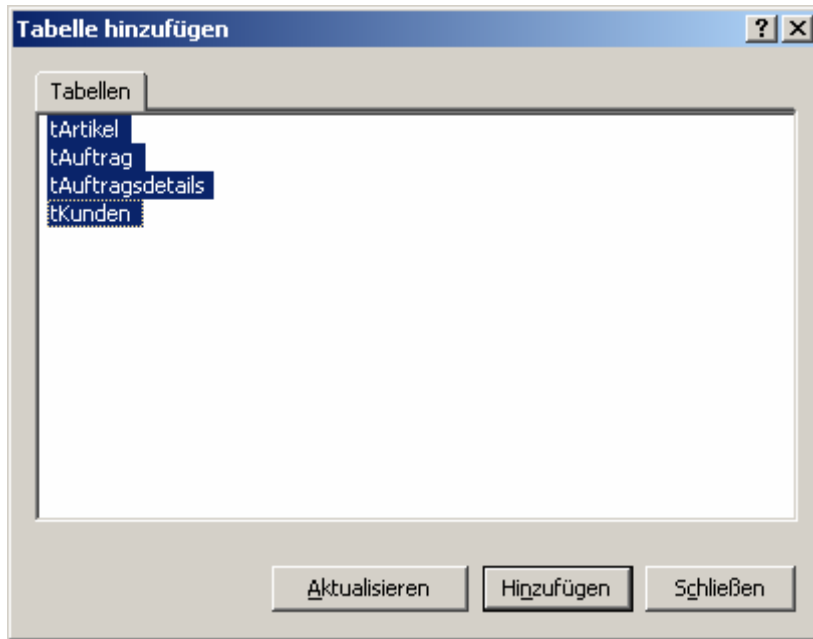
Die Erstellung von Fremdschlüsseleinschränkungen (Beziehungen) aktiviert einen Mechanismus im SQL Server, der die Integrität der eingegebenen Daten prüft (referentielle Integrität).


**Referenzielle Integrität:** Beispielsweise dürfen in einer Verkaufstabelle nur Kunden enthalten sind, die auch in der Kundentabelle angelegt sind



- Prüfung, ob Fremdschlüsselfelder vorhandenen Primärschlüsselfeldern entsprechen
- aus Mastertabelle können Datensätze (Tupel) erst dann gelöscht werden, wenn die verknüpften Datensätze in der Detailtabelle gelöscht werden


Im SQL Server 2008 Management Studio müssen dafür Datenbankdiagramme erstellt werden.






tKunden	
	KdNr
	Vorname
	Nachname

tAuftragsdetails	
	AuftrNr
	ArtNr
	Anzahl

tAuftrag	
	AuftrNr
	KdNr
	Datum

tArtikel	
	ArtNr
	ArtBez
	Einzelpreis

Die Beziehung wird erstellt, in dem die verknüpften Felder mit Drag and Drop verbunden werden.

**Tabellen und Spalten** [?] [X]

Beziehungsname:

Primärschlüsseltabelle:  Fremdschlüsseltabelle:

KdNr	KdNr

**Fremdschlüsselbeziehung** [?] [X]

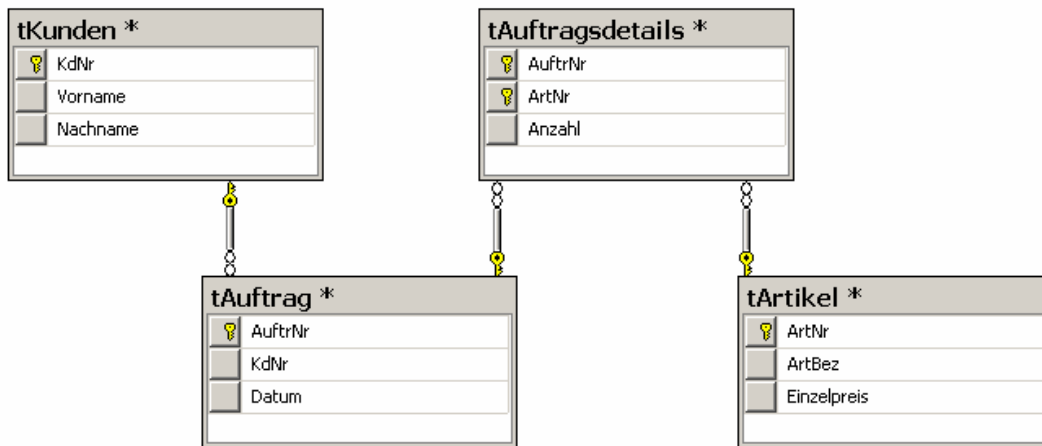
Beziehung (ausgewählt):

Eigenschaften für Beziehung (neu) werden bearbeitet. Die Tabellen- und Spaltenspezifikation-Eigenschaft muss ausgefüllt werden, bevor Beziehung (neu) akzeptiert wird.

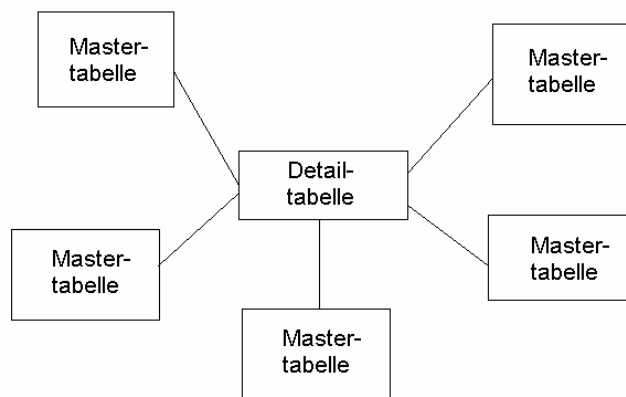
- (Allgemein)**
  - Tabellen- und Spaltenspezifik
 

Vorhandene Daten bei Erstell	Ja
------------------------------	----
- Datenbank-Designer**
  - Fremdschlüsseleinschränkung
  - Für Replikation erzwingen
- INSERT- und UPDATE-Spezif**
- Identität (ID)**

(Name)	FK_tAuftrag_tKunden
Beschreibung	



In einem Diagramm ist es meist günstig, wenn man die Mastertabellen (Stammdatentabellen mit Primärschlüsseln) sternförmig um die Detailtabellen (Fremdschlüsseltabellen) anordnet:



## 5 Sichten (Views)

Man sollte nie direkt mit den Tabellen, sondern immer mit Abfragen arbeiten.

```
use Auftrag
go

create view dbo.Auftragssicht
as
select
    tAuftragsdetails.AuftrNr,
    tAuftrag.Datum,
    tAuftrag.KdNr,
    tKunden.Vorname,
    tKunden.Nachname,
    tAuftragsdetails.ArtNr,
    tArtikel.ArtBez,
    tAuftragsdetails.Anzahl,
    tArtikel.Einzelpreis,
    Anzahl * Einzelpreis AS Zeilenpreis
from
    tKunden as k inner join tAuftrag as a
        on k.KdNr = a.KdNr
    inner join tAuftragsdetails as d
        on a.AuftrNr = d.AuftragsNr
    inner join tArtikel as art
        on d.ArtNr = art.ArtNr
where
    d.AuftrNr = 1;
```

### WICHTIG!

**Niemals verknüpfte Primärschlüsselfelder in der Abfrage verwenden!**

**Verknüpfte Felder in der Detailtabelle MÜSSEN in der Abfrage enthalten sein!**