

# SQL Server 2008 - Überblick und „Beyond Relational“

**Constantin Klein**

Software Architect

Freudenberg Forschungsdienste KG

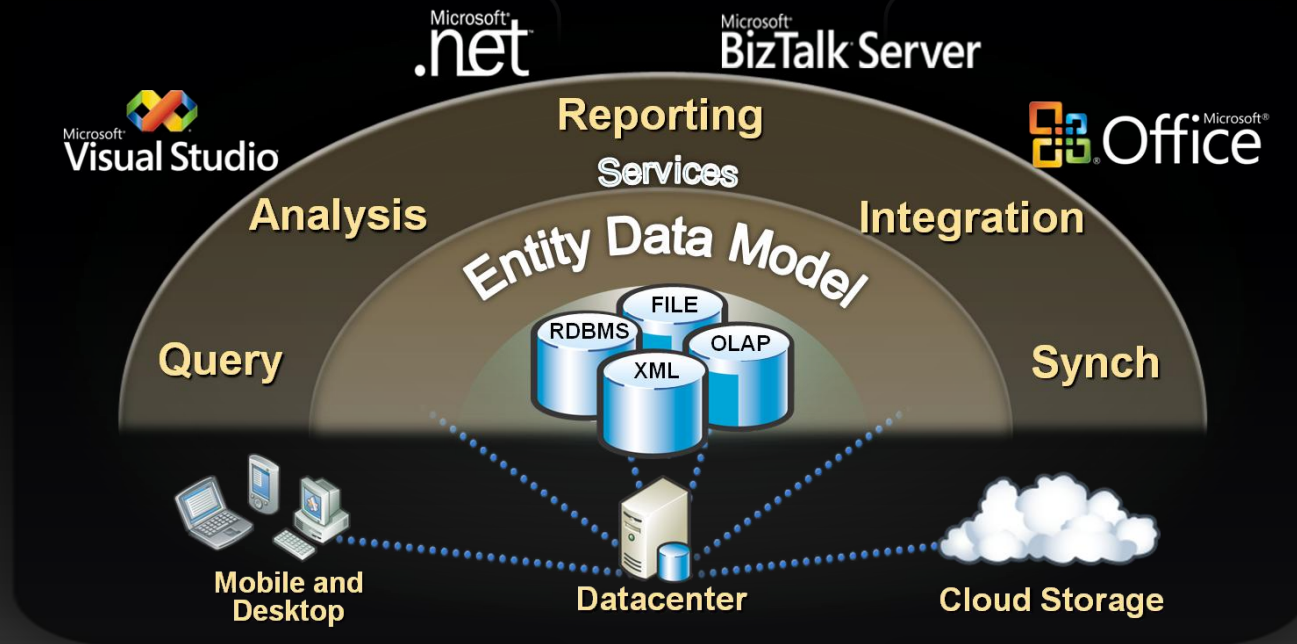
<http://kostjaklein.wordpress.com>

# Datenplattform-Vision von Microsoft

*Ihre Daten jederzeit und überall*

**Kontinuierliche Verfügbarkeit  
& Automatisierung**

**Mehr als relationale  
Speicherung**



**Produktive Entwicklung**

**Business Intelligence**

# Was ist neu?

## Microsoft® SQL Server™ 2008

- Transparent Data Encryption
- External Key Management
- Data Auditing
- Pluggable CPU
- Transparent Failover for Database Mirroring
- Declarative Management Framework
- Server Group Management
- Streamlined Installation
- Enterprise System Management
- Performance Data Collection
- System Analysis
- Data Compression
- Query Optimization Modes
- Resource Governor
- Entity Data Model
- LINQ
- Visual Entity Designer
- Entity Aware Adapters
- SQL Server Change Tracking
- Synchronized Programming Model
- Visual Studio Support
- SQL Server Conflict Detection
- FILESTREAM data type
- Integrated Full Text Search
- Sparse Columns
- Large User Defined Types
- Date/Time Data Type
- LOCATION data type
- SPATIAL data type
- Virtual Earth Integration
- Partitioned Table Parallelism
- Query Optimizations
- Persistent Lookups
- Change Data Capture
- Backup Compression
- MERGE SQL Statement
- Data Profiling
- Star Join
- Enterprise Reporting Engine
- Internet Report Deployment
- Block Computations
- Scale out Analysis
- BI Platform Management
- Export to Word and Excel
- Author reports in Word and Excel
- Report Builder Enhancements
- TABLIX
- Rich Formatted Data
- Personalized Perspectives
- ... and many more

## Enterprise-Datenplattform

- Produktive, richtlinienbasierte Verwaltung der Infrastruktur
- Sichere, zuverlässige Plattform für Ihre Daten
- Optimierte und vorhersehbare System-Performance



## Dynamische Entwicklung

- Beschleunigen Sie die Entwicklung mit Enterprise Developer
- Daten von überall aus synchronisieren



## Über relational hinaus

- Speicherung und Verwendung aller Arten von Dokumenten
- Bereitstellung von Standort-Intelligenz innerhalb Ihrer Anwendungen



## Business Intelligence

- Alle Ihre Daten in das Enterprise-Data-Warehouse integrieren
- Erreichen Sie alle Ihre Anwender mit einer zentralen BI-Plattform
- Unterstützen Sie jeden Anwender mit detaillierten Einblicken



# Agenda

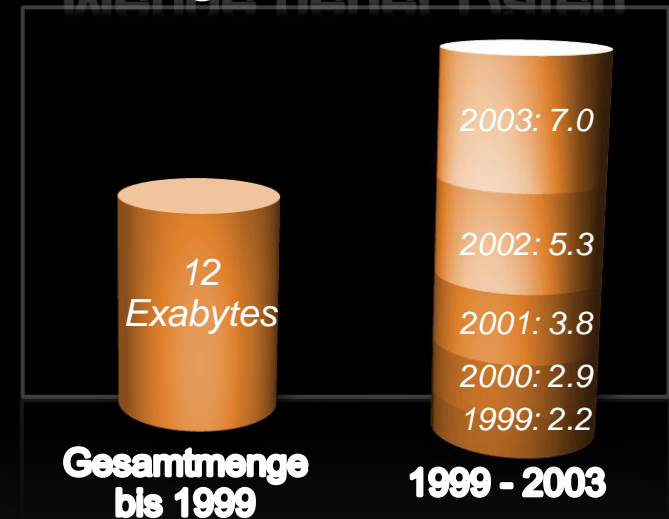
- Warum „Beyond Relational“?
- Filestream
- Geodaten
- Was gibt's sonst noch für Entwickler?

# Warum „Beyond Relational“

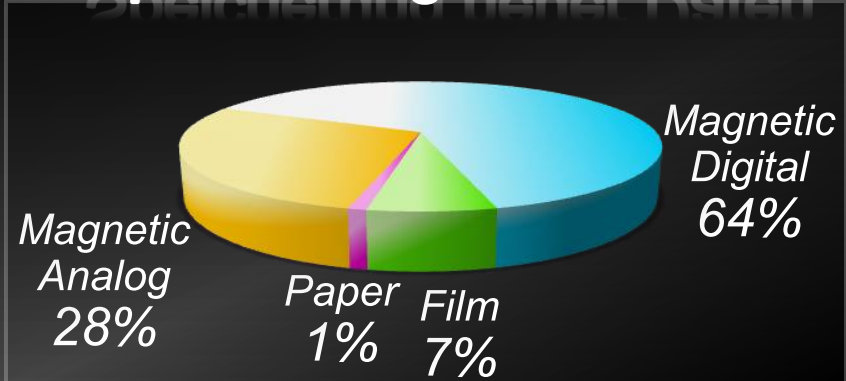
## Trends

- Menge neuer Daten steigt ständig an
- Der Großteil dieser Informationen liegt zwar digital, aber in Form von unstrukturierten Daten vor

### Menge neuer Daten



### Speicherung neuer Daten



# Heutige Herausforderungen

- Moderne Anwendungen müssen zunehmend auch diese Daten speichern und verwalten:
  - Bilder
  - Videos
  - Dokumente
  - ...
- Wie können diese Daten gespeichert werden?

# Heutige Herausforderungen



## Schmerz

- Unterschiede im Management von relationalen und nicht relationalen Daten
- Steigende Komplexität von Anwendungen und Duplizieren von Funktionalität
- Kompensierung nicht verfügbarer Dienste



## Ziel

- Reduzierung von Kosten für das Management unterschiedlicher Datentypen
- Vereinfachung der Anwendungsentwicklung für das Handling von relationalen und nicht relationalen Daten
- Erweiterung der für relationale Daten zur Verfügung stehenden Dienste auch für nicht relationale Daten

# Funktionen Übersicht

SQL Server 2005

SQL Server 2008



XML

- XML Data Type and Functions

- XML Upgrades



Documents &  
Multimedia

- Full Text Indexing

- Remote BLOB Store API
- FILESTREAM
- Integrated FTS



Spatial

- Fully supported Geometry and Geography data types and Functions

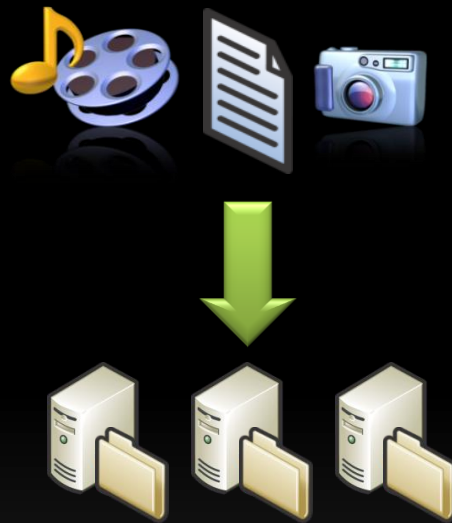


Relational  
Data

- User Defined Types

- Large UDTs
- Flexible Columns
- Wide Tables
- Filtered Indices
- HierarchyID

# Bisherige Lösungen für „BR“ Daten



Lösung

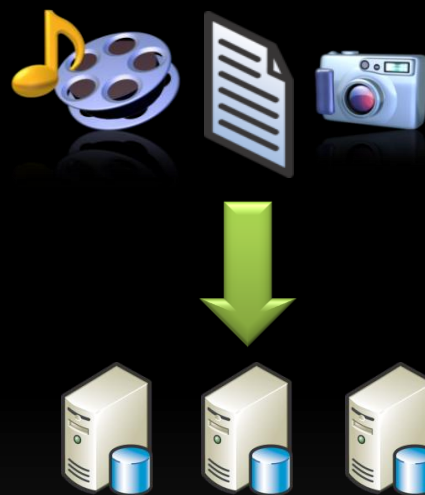
File System auf File Servern

Vorteil

- Kostengünstig
- Streaming Performance

Nachteil

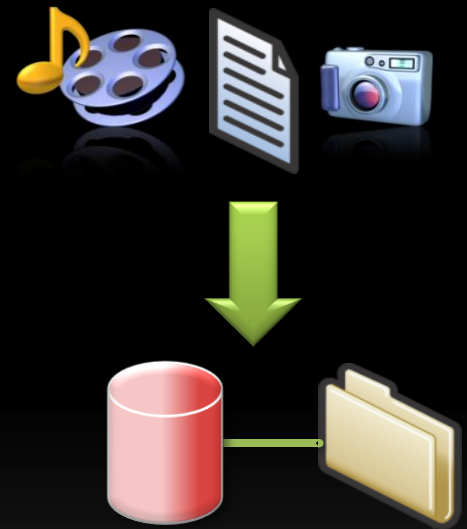
- Gemeinsame Entwicklung mit relationalen Daten
- Konsistenz



Speichern von BLOBs in Datenbanken

- Integriertes Management
- Transaktionale Konsistenz

- Schlechte Unterstützung für Datenstreaming
- Datei Größen Beschränkung (2GB)



DB + File System Integration

- Integriertes Management
- Garantierte Konsistenz
- Streaming Performance
- Einige V1 Einschränkungen

# Filestream – die Lösung

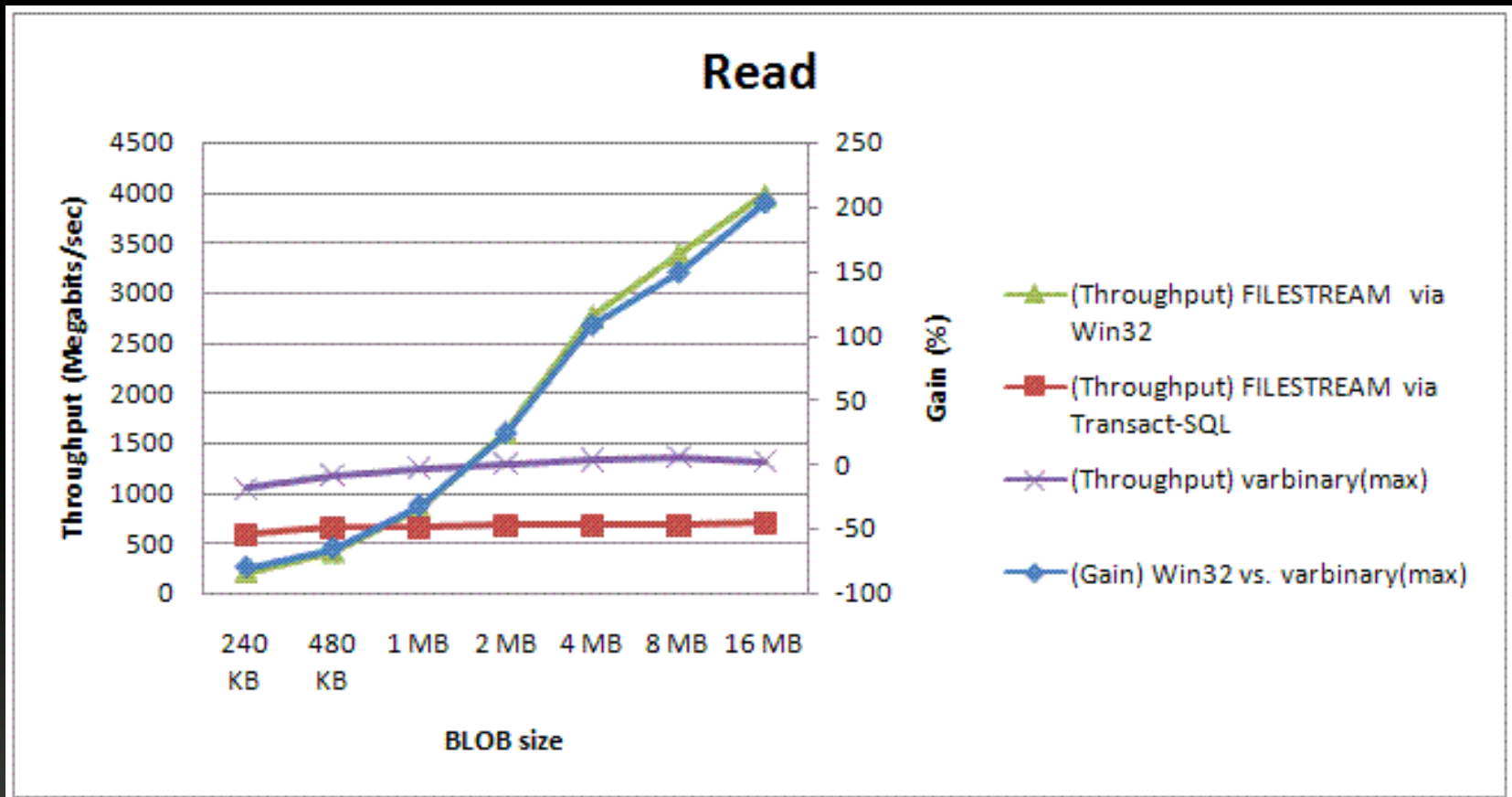
- Speicherattribut von varbinary(max)
- Unstrukturierte Daten werden direkt im File-System gespeichert (erfordert NTFS)
- Duales Programmiermodell
  - T-SQL (identisch zu SQL BLOB)
  - Win32 Streaming API mit T-SQL Transaktions Semantik
- SQL Server Sicherheitsmodell (identisch zu SQL BLOB)

**FILESTREAM**

# Wann Filestream?

- Die zu speichernden Daten sind im Schnitt größer als 1 MB
- Ein schneller lesender Zugriff ist wichtig
- Für kleinere Datengrößen bietet `varbinary(max)` in der Datenbank häufig eine bessere Performance und
- `varbinary(max)`, `varchar(max)` und XML Datentypen werden weiter voll unterstützt

# Read Performance



Quelle: FILESTREAM Whitepaper

# V1 Limitierungen

- Remote storage von FILESTREAM Daten wird (noch) nicht unterstützt
- DB Snapshot und DB Mirroring werden nicht supported
- Einige andere 2008er Features werden nicht unterstützt (z.B. SQL Encryption und TVPs)

# Vergleich der Speicherlösungen

Vergleichsmerkmal	File Server / File System	SQL Server (varbinary (max))	FILESTREAM
Maximale BLOB Größe	NTFS Laufwerksgröße	2 GB	NTFS Laufwerksgröße
Streaming Leistung	Exzellent	Schlecht	Exzellent
Sicherheit	Manuelle ACLs	Integriert	Integriert + automatische ACLs
Kosten pro GB	Niedrig	Hoch	Niedrig
Administrierbarkeit	Schwierig	Integriert	Integriert
Integration mit strukturierten Daten	Schwierig	Konsistenz auf Datensatzebene	Konsistenz auf Datensatzebene
Anwendungsentwicklung	Komplexer	Einfacher	Einfacher
Möglichkeiten zur Behebung von Datenfragmentierung	Exzellent	Schlecht	Exzellent
Performance von häufigen kleinen Updates	Exzellent	Mittelmäßig	Schlecht

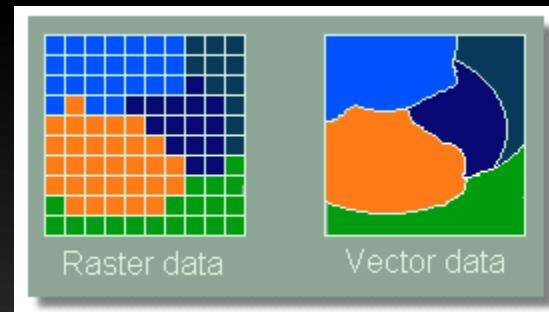
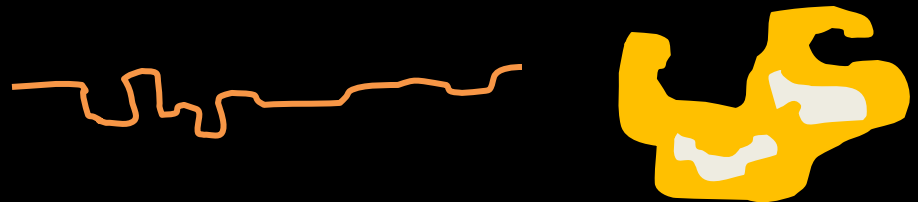
# Geodaten

# Geodaten

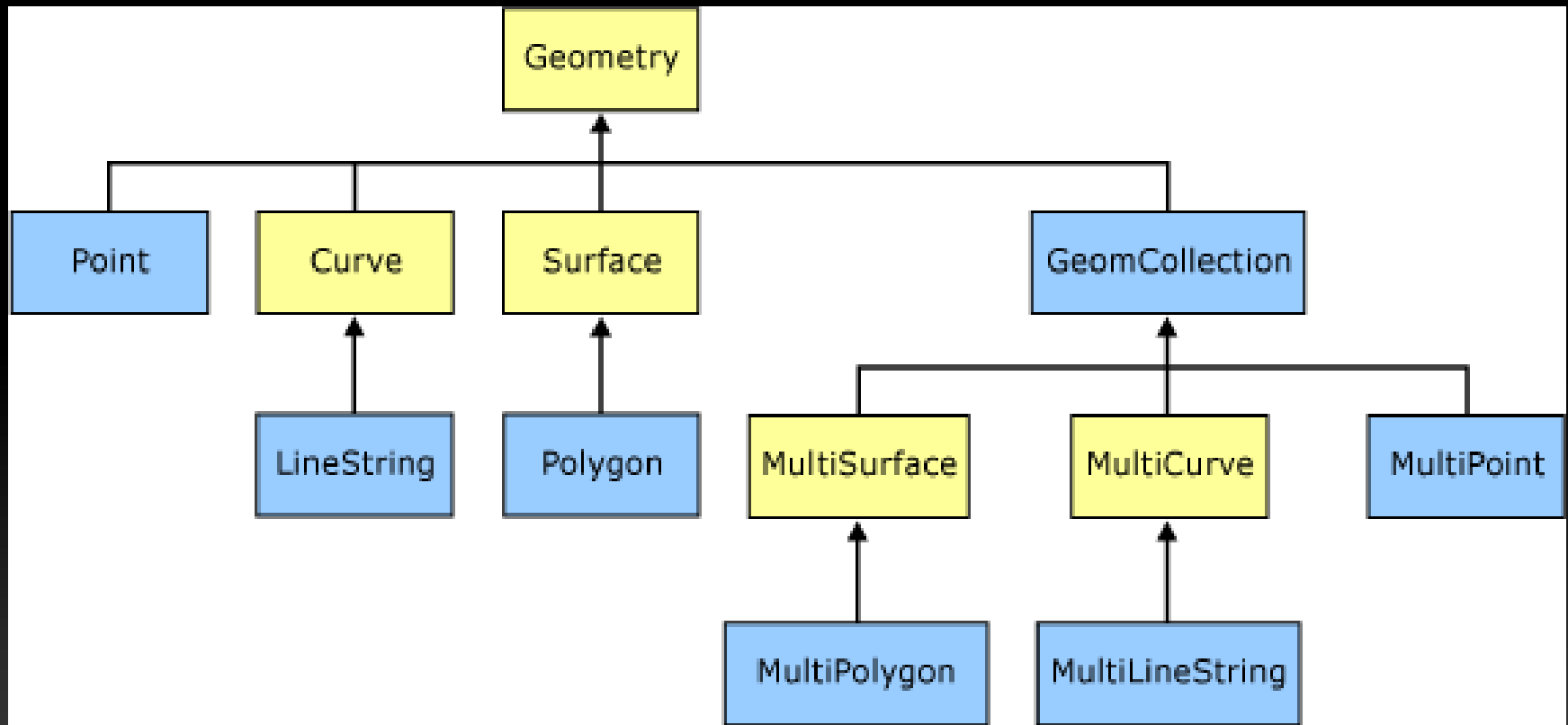
- Geodaten geben Antworten auf Fragen wie ...
  - Welche Straßen kreuzen ein bestimmtes Gelände?
  - Überlappen sich bestimmte Gebiete?
  - Welche italienischen Restaurants gibt es im Umkreis von 5 km?
- Geodaten sind schon jetzt Bestandteil vieler Datenbanken
  - Kundenadressen

# Was sind Spatial Daten?

- Das Open Geospatial Consortium (OGC) definiert eine Hierarchie von Datentypen
  - Point
  - Linestring
  - Polygon
  - MultiPoint
  - MultiLinestring
  - MultiPolygon
  - GeomCollection und
  - Nicht instanziiierbare Klassen auf denen diese Typen aufbauen



# OGC Hierarchie von Geodaten



# Der SQL Server 2008 und Geodaten

- Der SQL Server 2008 unterstützt zwei Typen von Geodaten
  - Geometry – flaches Erdmodell – planar
  - Geography – rundes Erdmodell – geodätisch
- Beide Arten unterstützen alle OGC Typen
  - InstanceOf kann die Typen unterscheiden
- Zunächst „nur“ Unterstützung von 2D Daten

# Planar und Geodätisch im Vergleich

- Planar
  - Unterstützung für Legacy Systeme
  - Geeignet für Innenräume (Architektur, ...)
  - Einfacher in der Berechnung
  - Konzept schwieriger für *geospatial*
- Geodätisch
  - Unterstützt existierende mapping Anforderungen für weite Entfernungen (Schiffe, Militär, ...)
  - Unterstützt neuere Geo-Anwendungen
  - Komplexer in der Berechnung
  - Einfacheres Konzept für *geospatial*

# Properties und Methods

- Geodaten Typen sind SQLCLR UDTs
  - '.' Syntax für Properties
  - '.' Syntax für Instanzmethoden
  - '::' Syntax für statische Methoden
  - Methoden und Properties sind case-sensitive

# Properties und Methods

- Jeder Typ unterstützt eine Anzahl von Methoden und Properties, die mit den Vorgaben des OGC übereinstimmt
  - für Geometry sind dies alle
  - für Geography die meisten
- Es gibt aber auch zusätzliche Erweiterungen
  - MakeValid
  - Reduce
  - ...

# Eingabe von Geodaten

- Geodaten werden binär gespeichert
- Instanzen des Typs können NULL sein
- Eingabe als
  - Well known Binary – ST[Type] FromWKB
  - Well known Text – ST[Type] FromText
  - Geography Markup Language (GML) –  
GeomFromGml
- Außerdem SQLCLR Funktionen
  - Parse
  - Point

# Geography

- Die meisten Verfügbaren Anwender Geodaten sind derzeit geodätisch
  - Alles was als Längengrad/Breitengrad ausgedrückt werden kann
  - Daher wird Geography der wahrscheinlich für die meisten Anwender interessantere Datentyp sein

# Some Code

- Neue Instanz

```
declare @g geography  
set @g = geography::Parse('POINT(47.6456 -122.12551)')
```

- Neue Tabelle

```
create table T(id int, region geography)
```

- Selektieren von einigen Daten

```
select * from T where region.STIntersects(@g) = 1
```

**SPATIAL**

# Date und Time

# Neue Zeiten sind angebrochen

Date

Time(n)

DateTimeOffset(n)

DateTime2(n)

# Date und Time

- Date
  - Enthält nur den Datumswert
  - Feste Größe von 3 byte
  - Spanne: 01.01.0001 – 31.12.9999
  - Bsp: 13-11-2008 oder 2008-11-13

# Date und Time

- Time(n)
  - Enthält nur den Zeitwert
  - Variable Größe von 3-5 byte
  - Genauigkeit bis zu 100 Nanosekunden möglich
  - Bsp: 08:15:30.11223

# Date und Time

- DateTimeOffset(n)
  - Enthält Datum und Uhrzeit inklusive der Zeitonenverschiebung
  - Variable Größe von 8-10 byte
  - Genauigkeit bis zu 100 Nanosekunden möglich
  - Bsp: 2008-11-13 08:15:30.11223 01:00

# Date und Time

- DateTime2(n)
  - Kombination aus Date und Time Datentyp
  - Variable Größe von 6-8 byte
  - Genauigkeit bis zu 100 Nanosekunden möglich
  - Spanne: 01.01.0001 – 31.12.9999
  - Bsp: 13-11-2008 08:15:30.11223

# Date und Time

- Erweiterung des SqlDbType Enum und der Clientbibliotheken

SQL Server Type	.NET Type
date	System.DateTime
time	System.TimeSpan
datetimeoffset	System.DateTimeOffset
datetime2	System.DateTime

**HierarchyID**

# HierarchyID Datentyp

- Perfekter Einsatz für
  - Unternehmensorganisation, Stücklisten, etc...
- Repräsentiert eine Position in einer Hierarchie
- Kompakte/effiziente Speicherung
- Beliebige Datensätze speichern/löschen
- Leistungsfähige Abfragen
  - GetRoot, GetLevel, IsDescendant, ReParent, etc.

# Hierarchische Daten in SQL 2005

- Parent ID <- ChildID
- Einfache Dateneingabe aber...
- Aufwändig in der Abfrage
  - Rekursion

# Hierarchische Daten in SQL 2008

- Neuer „Built-In“ Datentyp – HierarchyID
- SQLCLR basierte System UDT
  - Kann auch in .NET (VB.NET, C#) verwendet werden (SqlHierarchyId)
- Als path enumeration model implementiert (ORDPATH)
- HierarchyID (SQLCLR Typ) kann erweitert werden
  - Von dem Typ ableiten
  - Um Methoden und Properties erweitern

**Merge**

# Merge statement - Ausgangslage

- Für einen Abgleich müssen drei separate statements erzeugt werden (INSERT/UPDATE/DELETE) und nacheinander ausgeführt werden
- Eine explizite Transaktion muss über alle DML Operationen separat gepflegt werden um bei Fehlern das Rollback zu ermöglichen

# Merge statement - Anwendungsgebiete

- Bietet die Möglichkeit für die gleichzeitige Verwendung von DML Operationen (INSERT / UPDATE / DELETE) in einer Anweisung
- Bietet die Möglichkeit Änderungen in einem Batch durch den Client übertragen zu lassen und die Logik serverseitig auszuführen
- Befüllungen eines DataWareHouse
- BatchUpdate von Clients
- Synchronisieren von Datenquellen

# Merge statement - Syntax

```
[ WITH <common_table_expression> [,...n] ]  
MERGE  
[ TOP ( expression ) [ PERCENT ] ]  
[ INTO ] target_table [ [ AS ] table_alias ]  
    [ WITH ( <merge_hint> ) ]  
USING <table_source>  
(... 58 lines of syntax following...)  
;
```

# Merge statement - Aktionen

- **WHEN MATCHED**

- Ermöglicht Operationen wenn Datensätze sowohl in der Quelle als auch im Ziel vorhanden sind

- **WHEN NOT MATCHED [BY TARGET]**

- Ermöglicht Operationen wenn Datensätze in der Quelle aber nicht im Ziel existieren

# Merge statement - Aktionen

- **WHEN NOT MATCHED BY SOURCE**
  - Ermöglicht Operationen wenn Datensätze im Ziel aber nicht in der Quelle existieren

# Merge statement - Vorteile

- MERGE ist transaktional und benötigt keine explizite Transaktion für die beinhalteten DML Operationen
- MERGE löst Trigger nur einmal pro Aktion aus (BTW: Trigger werden pro Statement ausgelöst und NICHT pro Zeile! 😊)
- \$action ermöglicht es die ausgeführte Aktion (INSERT / UPDATE / DELETE) des Datensatzes zu erkennen

# Merge statement - Vorteile

- UPDATE ist nicht deterministisch: Möglichkeit von mehreren Datensätzen in der Quelltable (gefunden durch ON clause)
- MERGE ist deterministisch: Join (ON clause) erwartet einen eindeutigen Datensatz in der Quelltable

# Table Valued Parameter

# Table-valued parameter - Ausgangslage

- Batch Updates benötigten bisher mehrere Roundtrips zum Server um einen Abgleich zu machen
- Senden von strukturierten Daten war bisher nicht möglich, außer man konnte/wollte diese auf dem Server wieder zerlegen (TextArrays, XML, Binärdaten)
- Untypisierte Daten führten oftmals zu serverseitigen Fehlern

# Table-valued parameter - Ausgangslage

- Grundlage sind benutzerdefinierte Tabellentypen
  - Erweiterung der benutzerdefinierten Typen, der Typ wird in sys.table\_types persistiert
  - Erweiterung der bestehenden “Create Type” Syntax

```
CREATE TYPE myTableType AS  
TABLE (id INT, name  
NVARCHAR(100), qty INT);
```

```
DECLARE @MyVar myTableType
```

# Table-valued parameter - Anwendung

- Table-valued parameters
  - Verwendung von benutzerdefinierte Tabellentypen für die temporäre Speicherung von und den lesenden Zugriff auf die Daten (READONLY)
  - Vereint Vorteile von Tabellenvariablen und temporären Tabellen

# Table-valued parameter - Anwendung

- Ermöglicht die Übergabe als SqlParameter von Instanzen der Typen
  - DataTable
  - DbDataReader
  - System.Collections.Generic.IList<SqlDataRecord>
- Unterstützung der client stacks  
OLEDB/ODBC/ADO.NET
- Erweiterung der .NET (3.x) Clientbibliotheken durch den neuen SqlDbType „Structured“

# Table-valued parameter - Vorteile

- Ermöglicht große Mengen an Daten vom Client effizient (zur weiteren Verarbeitung) an den Server zu übertragen
- Bietet die Möglichkeit Parameter/Datensätze an Routinen in einer Variable zu übergeben (vs. XML, BLOB, CSV und andere Möglichkeiten die zusätzliche Verarbeitungen benötigen)
- Verringert bei großen Datenmengen die Anzahl der erzeugten Roundtrips erheblich

# Die kleinen Dinge

# Die kleinen Dinge im Leben...

- Row constructors

```
SELECT *  
FROM (VALUES  
      (10002,1,12,1,12),  
      (10002,1,12,1,12),  
      (10003,1,12,1,12)  
    )  
AS SomeOrders(OrderId, ProductId,  
UnitPrice, Quantity, Discount)
```

# Die kleinen Dinge im Leben...

- Row constructors

```
Insert Into [Order Details]  
VALUES  
    (10001,1,12,1,12)  
    , (10002,1,12,1,12)  
    , (10003,1,12,1,12)
```

# Die kleinen Dinge im Leben...

- Variableninitialisierungen

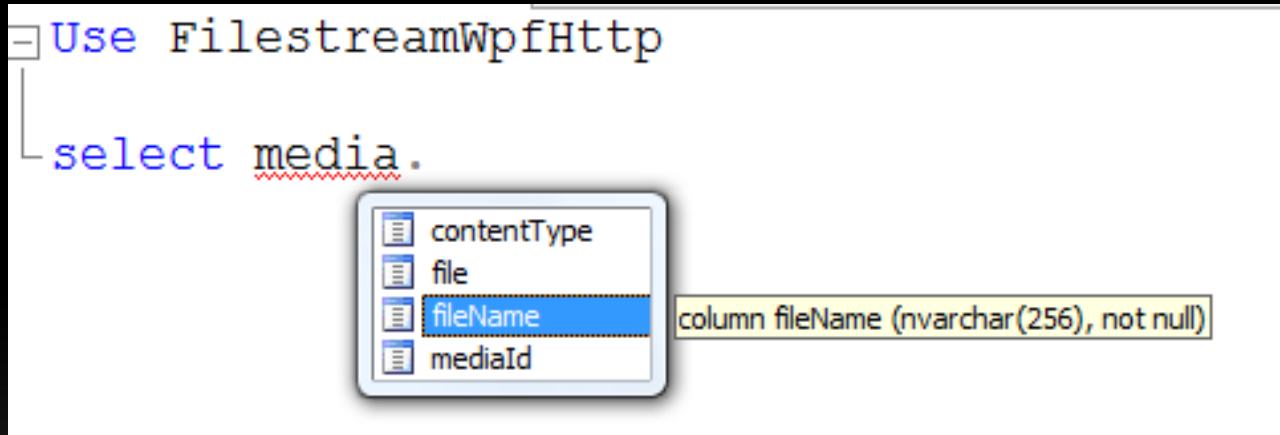
```
DECLARE @MyInt INT = 0
```

- Zuweisungsoperatoren
  - +=, -=, \*=, /=, &=, |=, %=, ^=

```
UPDATE [Order Details]  
SET Discount *= 1.1  
FROM [Order Details] OD  
INNER JOIN Orders OD  
ON O.OrderId = OD.OrderId  
WHERE OrderId = 10001
```

# Die kleinen Dinge im Leben...

- Intellisense :-)



# Was sollten Sie heute mitnehmen

- Es gibt wieder viel Neues im SQL Server 2008
- Für zukunftsfähige Anwendungen ist das Handling von nicht relationalen Daten ein MUSS
- Es gibt insbesondere in diesem Bereich viel Neues im SQL Server 2008
- ... und es lohnt sich, sich damit frühzeitig auseinanderzusetzen.

**Fragen ???**

**Feedback !!**

# Speakerinfo: Constantin Klein



- Constantin arbeitet als Anwendungsarchitekt und Entwickler bei der Freudenberg Forschungsdienste KG. Dort beschäftigt er sich hauptsächlich mit dem Design und der Entwicklung von Web-Informationssystemen und Datenbanken. Seit seinem Studium der Wirtschaftsinformatik gilt sein besonderes Interesse darüber hinaus allen aktuellen Themen im Microsoft .NET Umfeld, insbesondere aber dem Thema Softwarearchitektur. Er ist MCSD, MCITP Database Developer und MCPD Web + Enterprise Application Developer.
- Sein Blog finden Sie unter <http://kostjaklein.wordpress.com>

# Ressourcen

- Blog von Roger Doherty  
<http://blogs.msdn.com/rdoherty>
- Blog von Isaac K. Kunen  
<http://blogs.msdn.com/isaac>

## Tools für Spatial Data

<http://blogs.msdn.com/isaac/archive/2008/01/18/some-cool-tools-for-sql-server-spatial.aspx>

- Microsoft SQL Server Community Samples: End to End  
<http://codeplex.com/SQLSrvE2E>

# Ressourcen (2)

- Website des Open Geospatial Consortium (OGC)  
<http://www.opengeospatial.org>
- SpatialViewer  
<http://codeplex.com/SpatialViewer>
- Geoquery 2008  
<http://www.conceptdevelopment.net/Database/Geoquery>
- FILESTREAM Storage Whitepaper  
<http://msdn.microsoft.com/en-us/library/cc949109.aspx>

# Ressourcen (3)

- Microsoft SQL Server 2008 Feature Pack, April 2009  
<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=b33d2c78-1059-4ce2-b80d-2343c099bcb4>