SQL Server Data Types

Thomas Reinwart

Einleitung

Fast jede neue SQL Server Version hat in ihrer Feature Liste auch eine Erweiterung der SQL Datentypen dabei. Die SQL Datentypen bietet grundsätzlich die Möglichkeit die Daten so abzulegen, um sie in späterer Folge für den SQL Server Best möglich zu erreichen – also aus Performance und Memory Optimierung. Aber auch für den *Business Case* – der zukünftige Wertebereich sowie Format der Daten – ist hier bei der Wahl ausschlaggebend. Doch wie sieht es in der Realität aus, wer optimiert denn die Datentypen bei einem Upgrade der Datenbank Version tatsächlich und prüft die Auswirkungen auf die Schnittstelle und Anwendungen? Welche Fehlerquellen hier bei Datenbankabfragen lauern können, sehen wir uns hier an.

Mit SQL Server 2008 hinzugekommen

Das Speichern von Datum und Zeit Werte war zuvor immer lästig, entweder war der Minimum Wert der Jahre nicht mit 1 beginnend, oder man hatte nur Zeiten und musste aber zusätzlich ein Datum speichern.

Mit SQL Server 2008 kamen dann neue Datentypen hinzu:

datetime2: Hier beginnt das Datum mit 0001 und nicht mehr mit 1753 wie beim Datentyp datetime datetimeoffset: Hier ist eine Zeitzone inkludiert.

date: Damit lassen sich nur Tage abbilden. Time: Damit lassen sich nur Zeiten abbilden.

geography und **geometry**: Abbildung von Räumliche Daten.

Die Datentypen entsprechen der Open Geospatial Consortium (OGC) *Simple Features for SQL Specification Version 1.1.* [3]

Geography bietet die Unterstützung von planaren bzw. euklidischen Daten. Hier fallen etwa die GPS Position rein.

Geometry stellt einen räumlichen Datentyp dar.

Mit **Hierarchyid** kann eine hierarchische Verbindung dargestellt werde. Einsetzen kann man dies zur Abbildung einer organisatorischen Struktur, eines File-systems oder abhängigen Tasks in einem Projekt .

SQL Server 2012 Änderungen

Hier sind keine neuen Datentypen hinzugekommen, dafür sind einige als *deprecated* deklariert worden.

text: kann durch **varchar(max)** ersetzt werden. **ntext:** kann durch **nvarchar(max)** ersetzt werden. **image**: kann durch **varbinary(max)** ersetzt werden. Mehr Details dazu siehe auch online [1].

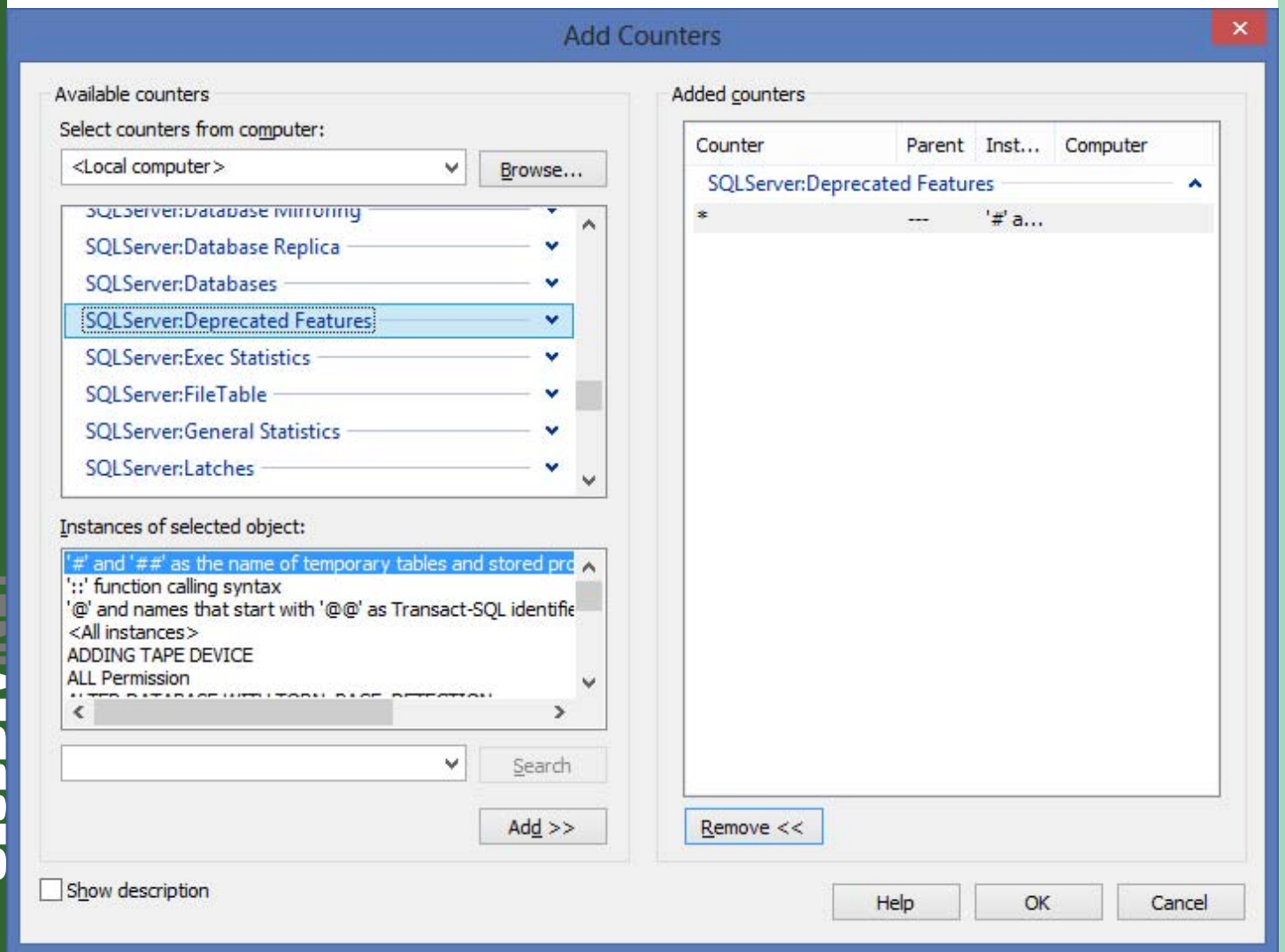
SQL Server 2014 Kompatibilität zur vorherigen Versionen

Hier wurden die gleichen Datentypen wie beim SQL Server 2012 als depricated deklariert. Neue Datentypen sind nicht hinzugekommen.

Über die *Breaking Changes* des SQL Server 2014 ist derzeit in der MSDN noch nichts Offizielles zu finden.

Columnstore Index SQL Server 2012 und 2014

Der Sql Server 2012 bot die Option an, einen *columnstore index* zu verwenden. Mittels dieses Index konnte man bis zu einer 10x höheren Performance eine 7x höhere Compression zu herkömmlichen



Tabellen erreichen. Aber leider mit dem Tribut, auf den darunter liegende Table nur im *readonly* modus zugreifen zu können. Im SQL Server 2014 wurde diese Limitierung aufgehoben. Mit dem *clustered columsstore index* kann diese Performance gehalten werden, der darunter verwendete Table kann mit allen DML Operationen *(insert, update, delete)* betrieben werden.

Systemmonitor nutzen um deprecated features am SQL Server anzuzeigen

Mit dem Start von *perfmon.exe* – dem Windows Systemmonitor, werden eine Vielzahl von Überwachungsmöglichkeiten des Systems angeboten. Neben Messungen des Betriebssystems wird hier auch eine Menge für den SQL-Server angeboten, beginnend von *Alerts, Jobs, Statistics, über Broker, Locks und Memory, Transaction* lässt sich hiermit auch die Verwendung von *Deprecated Features* aufspüren.

Datentyp	Wertebereich von	bis	Info
date	0001-01-01	9999-12-31	ab 2008 Auf 1 Tag genau
time	00:00:00.0000000	23:59:59.9999999	ab 2008 auf 100 ns genau
datetime	1753-01-01 00:00:00	9999-12-31 23:59:59:997	Auf ~3⅓ Millisekunden genau (.000, .003, .007, .010) Obsolete ab 2012
datetime2	0001-01-01	9999-12-31	auf 100ns genau ab 2008
datetimeoffset	0001-01-01 00:00:00	9999-12-31 23:59:9999999	auf 100ns genau ab 2008
smalldatetime	1900-01-01 00:00	2079-06-06 23:59	auf 1 Minute genau Obsolete ab 2012

Festlegen der Sprache am SQL Server

Die Liste der möglichen Sprachen erhält man mittels

```
Select * from sys.syslanguages
```

Will man die Deutsche Auslegung des Datums, setzt man

```
SET LANGUAGE German;
```

Überblick über die Kategorien von Datentypen am SQL Server

SQL Server Datum und Zeit [2]

Warum wurde gerade 1753-01-01 als Beginn Wert des Datentyps `datetime` gewählt?

Das hat historische Gründe, es gibt den Julianischen und den Gregorianischen Kalender. Die beiden Kalender differieren 10 bis 13 Tage, abhängig vom Jahrhundert. Die Kulturen wechselten den Kalender zu unterschiedlichen Zeitpunkten. So wechselte Großbritannien am 02.09.1752, durch die Differenz der Tage folgte der nächste Tag der 14.09.1752. Sybase hatte damals beschlossen, 1753 als frühestens Datum zu verwenden, um diese unterschiedliche Berechnungslogik früherer Datumswerte nicht durchführen zu müssen. SQL Server tat es ebenso. Andere Länder haben diesen Kalenderwechsel erst später vollzogen, die Türkei etwa erst 1927.

Beim Verwenden der Daten ist man oft mit Konvertierungen konfrontiert, eine weitere Fehlerquelle, die nur bei bestimmen Tagen des Jahres auftritt, wenn datetime als Cast Typ verwendet wird.

```
SET LANGUAGE British;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 107) AS British;
SET LANGUAGE US_English;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 107) AS US;
SET LANGUAGE German
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 104) AS German;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 102) AS ANSI;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 102) AS ISO;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 126) AS ISO8601;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 127) AS ISO8601WithTimezone;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 113) AS EuropeDefault;
```

British	US	German	ANSI	ISO	ISO8601	ISO8601 WithTimezone	Europe Default-Date
Dec 09, 2013	Sep 12, 2013	09.12.2013	2013.12.09	2013.12.09	2013-12-09T00:00:00	2013-12-09T00:00:00	09 Dez 2013 00:00:00

```
SET LANGUAGE German;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime), 107) AS DateOnlyDate;
SELECT CONVERT(varchar(20), CAST('2013-09-12' AS datetime2), 107) AS DateOnlyDate2;
SELECT CONVERT(varchar(20), CAST('2013-09-12T00:00:00' AS datetime), 107) AS ISODateTimeDate;
SELECT CONVERT(varchar(20), CAST('2013-09-12T00:00:00' AS datetime), 104) AS GermanDateTimeDate;
```

Wird `datetime2` beim Cast bei einem Wert verwendet, bei dem die Zeitangabe ohne `T` formatiert ist, ist das Resultat korrekt:

DateOnlyDate	DateOnlyDate2	ISODateTimeDate	GermanDateTimeDate
Dec 09, 2013	Sep 12, 2013	Sep 12, 2013	12.09.2013

```
SET LANGUAGE British;
SELECT CONVERT(varchar(20), CAST('20130912' AS datetime), 107) AS BritishDate;
SET LANGUAGE US_English;
SELECT CONVERT(varchar(20), CAST('20130912' AS datetime), 107) AS USDate;
SET LANGUAGE German;
SELECT CONVERT(varchar(20), CAST('20130912' AS datetime), 104) AS GermanDate;
```

Bei der Formatierung des Datumwertes ohne Bindestrich liefert allerdings auch das alte datetime die richtige Übersetzung:

BritishDate	USDate	GermanDate
Sep 12, 2013	Sep 12, 2013	12.09.2013

Die Datentypen date und time wurden mit dem SQL Server 2008 eingeführt, um nur Datum oder nur eine Zeit Informationen zu speichern. In den vorigen SQL Server Versionen war das Speichern einer solchen Information nur über den Datentype datetime möglich, was im Datenbestand zu unnötigen Daten führte. Im Zuge eine Migration auf eine aktuelle SQL Server Version sollte auch hier eine Entscheidung fallen, wie mit der Datenqualität in Zukunft umgegangen wird.

```
SET LANGUAGE German
SELECT CAST('20130912' AS datetime) as OnlyDateNoTime
SELECT CAST('20130912' AS datetime2) as OnlyDateNoTime2
SELECT CAST('14:23:58' AS datetime) as OnlyTimeNoDate
SELECT CAST('14:23:58' AS datetime2) as OnlyTimeNoDate2
SELECT CAST('14:23:58' AS time) as OnlyTime
SELECT CAST('20130912' AS date) as OnlyDate
```

OnlyDate NoTime	OnlyTime NoDate2	OnlyTime NoDate	OnlyTime NoDate2	Only Time	Only Date
2013-09-12	2013-09-12	1900-01-01	1900-01-01	14:23:58.0000000	2013-09-12

Wie man am Ergebnis sieht, speichert der SQL Server beim Datentype datetime und datetime2, wenn nur die Zeit angegeben wird, mit dem Begin Datum 1900-01-01. Und im anderen Fall, bei dem nur ein Datum verwendet wird, ist die Zeitangabe immer 00:00:00.000.

Das Ermitteln der aktuellen Zeit am Server, etwa für einen automatischen Timestamp, sind mehrere Varianten möglich, mit unterschiedlichem Ergebnis und Genauigkeit.

```
SELECT 'SYSDATETIME' AS FunctionName, SYSDATETIME();
SELECT 'SYSUTCDATETIME' AS FunctionName, SYSUTCDATETIME();
SELECT 'CURRENT_TIMESTAMP' AS FunctionName, CURRENT_TIMESTAMP;
SELECT 'GETDATE' AS FunctionName, GETDATE();
SELECT 'GETUTCDATE' AS FunctionName, GETUTCDATE();
```

SYSDATETIME	SYSUTCDATETIME	CURRENT_TIMESTAMP	GETDATE	GETUTCDATE
2013-08-06 13:13:09.1301427	2013-08-06 11:13:09.1301427	2013-08-06 13:13:09.130	2013-08-06 13:13:09.130	2013-08-06 11:13:09.130

GMT - GreenwichMean Time

War von 1884 bis 1928 die Weltzeit. Ist durch UTC abgelöst.

French TUC - temps universel coordonné

UTC - *Coordinated Universal Time – koordinierte Weltzeit*

Die heute gültige Weltzeit.

Die Anfangsbuchstaben für *Coordinated Universal Time* sollten eigentlich CUT sein, doch Aufgrund einer Kontroverse mit der TUC hat man sich auf UTC geeignet.

- Grundformat: `19850412T232050`
- Erweitertes Format: `1985-04-12T23:20:50`

Angabe mit Zeitzone:

- Grundformat: `19850412T232050+0100`

- Erweitertes Format: `1985-04-12T23:20:50+01:00`

Am SQL Server ist hier das 'T' in der Formatierung anzugeben.

Aber auch die Datentypen der Datenbankhersteller können unterschiedliche Wertebereiche aufweisen, die SQL Dialekte können also vom Standard abweichen. Hier lauert bereits die erste Fehlerquelle, wenn Daten etwa über Schnittstellen ausgetauscht werden und am jeweiligen System andere Datenbanken im Einsatz sind.

Hier etwa die Unterschiede von Datum und Zeit von MySQL 5.x

Datentyp	Wertebereich	Genauigkeit
datetime	01.01.1000 00:00:00 bis 31.12.9999 23:59:59	1 Sekunde
date	01.01.1000 bis 31.12.9999	1 Tag
time	–838:59:59 bis 838:59:59	1 Sekunde
year	1901 bis 2055	1 Jahr

SQL Server Numerische Datentypen

Datentyp	Wertebereich	Wertebereich
bit	0	1
tinyint	0	255
smallint	-32.768	32.767
int	-2.147.483.648	2.147.483.647
bigint	-9.223.372.036.854.775.808	9.223.372.036.854.775.808
decimal, numeric	$-10^{\pm 38}+1$	$10^{\pm 38}-1$
smallmoney	-214.748,3648	214.748,3647
money	-922.337.203.685.477,5808	922.337.203.685.477,580
float	$-3.40 * 10^{\pm 308}$	$3.40 * 10^{\pm 308}$
real	$-1.79 * 10^{\pm 38}$	$1.79 * 10^{\pm 38}$

Bei decimal und money gibt es unterschiedliche Ergebnisse durch Runden. Die Entscheidung des passenden Datentypes ist also bei einer Finanzanwendung eine Entscheidende.

```
DECLARE @dOne decimal(19,4) = 1,
@dThree decimal(19,4) = 3,
@mOne money = 1,
@mThree money = 3;

SELECT @dOne / @dThree * @dThree AS DecimalResult,
@dOne * @dThree / @dThree AS ReorderedDecimalResult,
@mOne / @mThree * @mThree AS MoneyResult,
@mOne * @mThree / @mThree AS ReorderedMoneyResults;
GO
```

DecimalResult	ReorderedDecimalResult	MoneyResult	ReorderedMoneyResults
1.000000	1.000000	0,9999	1,00

SQL Server Zeichenfolgen

Datentyp	Anzahl der möglichen Zeichen	Info
char(n)	8000	Non Unicode
nchar(n)	4000	UTF16
varchar(n)	8000	
varchar(max)	~ 2.000.000.000	
nvarchar(n)	4000	
nvarchar(max)	~ 1.000.000.000	
text	~ 2.000.000.000	Obsolete ab 2012
ntext	~ 1.000.000.000	Obsolete ab 2012

Unterschied zwischen Datentyp und Datentyp(n) : Unicode Zeichen

Unterschiede von Zeichenfolgen bei Oracle 11g

Datentyp	Länge	Parameter Länge	Ablage der Daten	Platz	Ablage
char	Fix	s für die Länge des Strings	Wenn der String kleiner ist als die angegebene Länge, wird mir Leerzeichen aufgefüllt. Maximal 2000 Bytes	Verbraucht viel Platz	Größere Daten werden abgeschnitten
varchar	variable	Maximale Länge des Strings	Kann maximal 4000 Bytes speichern	Belegt Leerzeichen als	Datengröße zu Beginn festgelegt
varchar(2)	variable	Maximale Länge des Strings	Kann maximal 4000 Bytes speichern	Belegt Leerzeichen als Null Werte	Datengröße zu Beginn festgelegt

Die Unterschiede von varchar und varchar(2) sind historisch bedingt, varchar hatte in älteren Oracle Version nur 2000 Bytes Platz zur Verfügung.

SQL Server Binary

Datentyp	Anzahl der möglichen Bytes	Info
binary(n)	8000	
varbinary(n)	8000	
varbinary(max)	~2.000.000.000	
image	~2.000.000.000	Obsolete ab 2012

SQL Server weitere Datentypen

Datentyp		Info
cursur		
timestamp/rowversion	Wird als Erkennung einer Datensatzänderung verwendet.	Timestamp ist durch rowversion zu ersetzen
hierarchyid		
uniqueidentifier		
sql_variant		
xml		
table		

Fazit

Einige Datentypen von Datenbank sind teilweise noch aus der frühen Zeit der EDV, als mögliche Datengrenzen noch in weiter Ferne lagen und die Schonung der Ressourcen noch im Vordergrund stand. Aber auch zwischen den Datenbanksystemen besitzen gleichnamigen Datentypen unter Umständen andere Wertebereiche, hier kann man sich nicht blind auf den Namen des Datentyps verlassen. Die Wahl eines passenden SQL Datentyps ist das A und O, die bei der erstmaligen Datenbankdefinition beginnt und bei Upgrade auf eine neue Datenbankversion nicht aufhört.

Links & Quellen

Deprecated Database Engine Features in SQL Server

[1] http://msdn.microsoft.com/en-us/library/ms143729.aspx

ISO Time Format

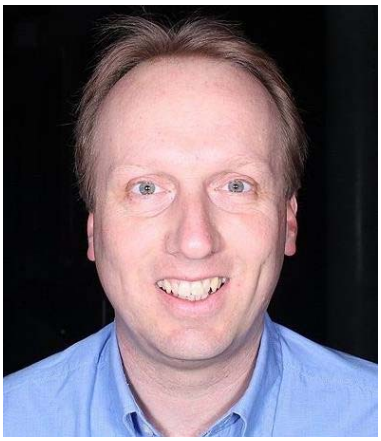
[2] http://wwp.greenwichmeantime.com/info/iso.htm

Opengeospatial Organisation

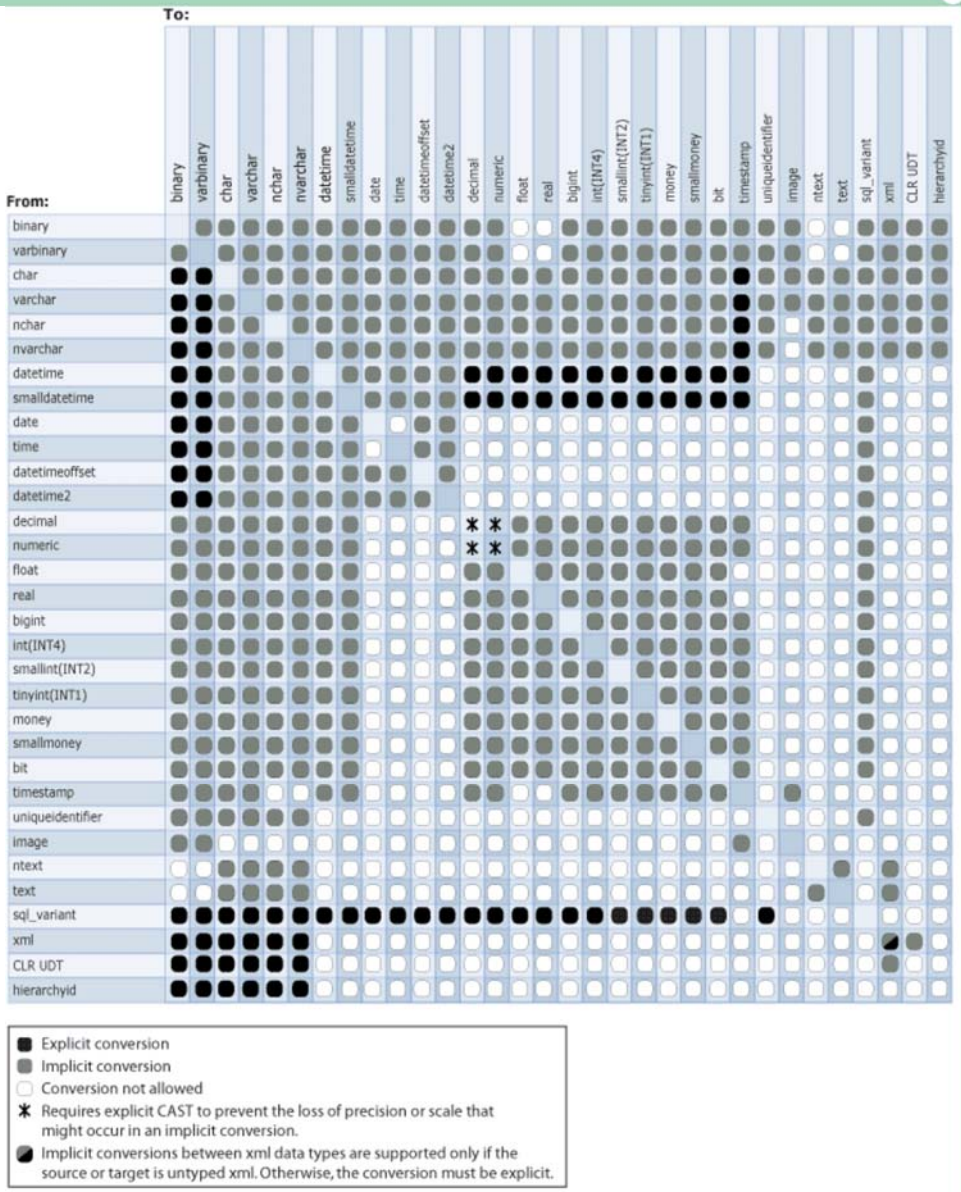
[3] http://www.opengeospatial.org/

Autorenbox

Thomas Reinwart verfügt über umfangreiche Berufserfahrung auf dem IT Sektor. In den letzten 20 Jahren war er in den Bereichen Softwareentwicklung, Softwaredesign, Architekt und als Consultant tätig. Fokus ist derzeit Microsoft .net und SQL Server, wo er alle aktuellen Microsoft-Zertifizierungen hat.



Email: office@reinwart.com



Inhaltsverzeichnis Kurs „Netzwerktechnik“ von Christian Zahler