Datentypen und Funktionen

- SQL Server ordnet Spalten, Ausdrücke, Variablen und Parametern Datentypen zu
- Datentypen bestimmen, welche Daten gespeichert werden können (int, Zeichen, Datum, Money, binäre Daten etc.)
- SQL Server verfügt über integrierte Datentypen
- Entwickler können auch benutzerdefinierte Typen erstellen/festlegen (Aliase in T-SQL)

Datentypen I

Datentyp	Bereich	Speicher
tinyint	0 bis 255	1
smallint	-32.768 bis 32.767	2
int	-2.147.483.648 bis 2.147.483.647	4
bigint	-2^63 bis 2^63-1 (ca. +/- 9 Quintillion)	8
bit	1, 0 oder NULL	1
decimal/numeric	-10^38+1 bis 10^38-1 bei max. Genauigkeit	5 bis 17
money	-922.337203.685.477,5808 bis 922.337203.685.477,5807	8
smallmoney	-214.748,3648 bis 214.748,3647	4
float(n)	-1,79E+308 bis -2,23E-308, 0 und 2,23E-308 bis 1,79E+308	Je nach n, 4 oder 8
real	-3,40E+38 bis -1,18E-38, 0 und 1,18E-38 bis 3,40E+38	4

Bei float(n) ist ‘n‘ die Anzahl der Bits, mit denen die Mantisse der Gleitkommazahl in wissenschaftlicher Schreibweise gespeichert wird.

Die Nachkommastellen von Gleitkommazahlen werden bei der Umwandlung in ganze Zahlen abgeschnitten

Datentypen II – Binäre Datentypen

Datentyp	Bereich	Speicher
binary(n)	1 bis 8000 Bytes	n Bytes
varbinary(n)	1 bis 8000 Bytes	n Bytes + 2
varbinary(max)	1 bis ca. 2,1 Milliarden Bytes	Länge + 2

- Gut geeignet für „unstrukturierte“ Daten wie Bilder o.ä.
- Sollte eventuell als Filestream abgelegt werden

Datentypenrangfolge

- Die Priorität bestimmt die Auswahl
- Eine niedrige Priorität wird bei Kombination in den Datentyp mit der höheren Priorität umgewandelt
- Konvertierung in einen Datentyp niedrigerer Priorität muss explizit (mit CAST-Funktion) erfolgen
- (niedrig aufwärts) char -> varchar -> nvarchar -> tinyint -> int -> decimal -> time -> date -> datetime -> xml

Wann Konvertierung?

- Verschieben von Daten, kopieren in andere Tabelle
- Implizit beim Vergleichen (WHERE-Klausel)
- Explizit beim Verwenden der CAST- oder CONVERT-Funktion
- Nicht alle denkbaren Konvertierungen sind zulässig!

  ➢ z.B. timestamp in nvarchar kann nur Chuck Norris!

CAST

- Konvertiert einen Wert mit einem Datentyp in einen Anderen
  - ➢ In SELECT- und WHERE-Klauseln anwendbar

- Syntax und Beispiel:

```
CAST(<Datenwert> AS <Datentyp>)

SELECT CAST(SYSDATETIME() AS date)
```

- Nichtkompatible Datentypen lösen Fehlermeldungen aus

CONVERT

- Konvertiert einen Wert mit einem Datentyp in einen Anderen
  - ➢ In SELECT- und WHERE-Klauseln anwendbar
- CONVERT ist SQL Server-eigene Funktion, kein Standard
- Format gibt Konvertierungsart an, Datum, Uhrzeit, num. Werte, XML, etc.
- Syntax und Beispiel:

```
CONVERT(<Datentyp>, <Wert>, <optionale Formatnummer>)

CONVERT(CHAR(8), CURRENT_TIMESTAMP, 112) AS ISO_FORMAT;
```

PARSE

- Konvertiert Zeichenfolgen in Datums- Uhrzeit- und Zahlentypen

PARSE-Element	Kommentar
Zeichenkette	Formatierte Eingabe varchar(4000)
Datentyp	Angeforderte Datentypausgabe
Kultur	Optionale Zeichenfolge im .NET-Kulturformat: de-DE, es-ES, ar-SA usw.

- Beispiel:

```
SELECT PARSE (’02/12/2016‘ AS datetime2 USING ‘en-US‘)
AS Parse_Ergebnis;
```

TRY_PARSE und TRY_CONVERT

- Ergebnis der Typkonvertierung als Rückgabewert
- Bei Fehlschlag NULL als Rückgabewert
- Beispiel:

```
SELECT TRY_PARSE (’SQL-Server‘ AS datetime2 USING ‘en-US‘)
AS ‘try_parse_Result‘;
```

	try_parse_Result
1	NULL

Textdatentypen

Datentyp	Bereich	Speicher
char(n) nchar(n)	1-8000 Zeichen	n Bytes, aufgefüllt 2*n Bytes, aufgefüllt
varchar(n) nvarchar(n)	1-8000 Zeichen	n+2 Bytes (2*n)+2 Bytes
varchar(max) nvarchar(max)	1-2^31-1 Zeichen	Tatsächliche Länge +2

- char, nchar sind in der Länge nicht veränderbar
- varchar, nvarchar haben eine variable Länge
- Zeichendaten durch einfache Anführungszeichen voneinander trennen

Texte verketten

- ‘+‘ verkettet Texte
- Verkettung mit NULL ergibt NULL als Rückgabewert

```
SELECT EmpID, LastName, FirstName,
           Firstname + N‘ ‘ + LastName AS FullName
FROM dbo.Employees;
```

- CONCAT() - Funktion, um NULL-Ausdrücke in leere Zeichenkette umzuwandeln

```
SELECT CustomerID, City, Region, Country,
          CONCAT(City, ‘, ‘ + Region, ‘, ‘ + Country) AS Location
FROM dbo.Customers;
```

Wichtige Funktionen

Funktion	Syntax	Anmerkungen
SUBSTRING()	SUBSTRING(expression, start, length)	Gibt einen Teil des Ausdrucks zurück
LEFT(), RIGHT()	LEFT(expression, integer_value) RIGHT(expression, integer_value)	LEFT() gibt den liken Teil der Zeichenfolge bis zu integer_value zurück, RIGHT() entsprechend
LEN() DATALENGTH()	LEN(expression) DATALENGTH(expression)	LEN() gibt die Anzahl der Zeichen des angegebenen Zeichenfolgeausdrucks zurück, ausschließlich nachfolgender Leerzeichen. DATALENGTH gibt die Anzahl der verwendeten Bytes zurück
CHARINDEX()	CHARINDEX(expressionToFind, expressionToSearch)	Sucht nach einem Ausdruck für einen anderen Ausdruck und gibt im Falle eines matches dessen Startposition zurück. Die Startposition ist optional
REPLACE()	REPLACE(expression, pattern, replacement)	Ersetzt alle Vorkommen eines gegebenen Zeichenfolgewerts durch einen anderen Wert
UPPER() LOWER()	UPPER(expression) LOWER(expression)	Wandelt Kleinbuchstaben in Großbuchstaben um und umgekehrt

Datum und Zeit I

Datentyp	Speicher(Byte)	Datumsbereich	Genauigkeit	Eingabeformat (empfohlen)
DATETIME	8	1.1.1753 – 31.12.9999	3 – 1/3 Millisekunden	‚YYMMDD hh:mm:ss:nnn‘
SMALLDATETIME	4	1.1.1900 – 6.6.2079	1 Minute	‚YYMMDD hh:mm:ss:nnn‘
DATETIME2	6 bis 8	1.1.0001 – 31.12.9999	100 Nanosekunden	‚YYMMDD hh:mm:ss:nnnnnn‘
DATE	3	1.1.0001 – 31.12.9999	1 Tag	‚YYYY-MM-DD‘
TIME	3 bis 5		100 Nanosekunden	‚hh:mm:ss:nnnnnn‘
DATETIMEOFFSET	8 bis 10	1.1.0001 – 31.12.9999	100 Nanosekunden	‚YY-MM-DD hh:mm:ss:nnnnnn [+/-]hh:mm‘

Datum und Zeit II

Datentyp	Sprachunabhängige Formate	Beispiele
DATETIME	'YYYYMMDD hh:mm:ss.nnn' 'YYYY-MM-DDThh:mm:ss.nnn' 'YYYYMMDD'	'20120212 12:30:15.123' '2012-02-12T12:30:15.123' '20120212'
SMALLDATETIME	'YYYYMMDD hh:mm' 'YYYY-MM-DDThh:mm' 'YYYYMMDD'	'20120212 12:30' '2012-02-12T12:30' '20120212'
DATETIME2	'YYYY-MM-DD' 'YYYYMMDD hh:mm:ss.nnnnnnn' 'YYYY-MM-DD hh:mm:ss.nnnnnnn' 'YYYY-MM-DDThh:mm:ss.nnnnnnn' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567' '2012-02-12 12:30:15.1234567' '2012-02-12T12:30:15.1234567' '20120212' '2012-02-12'
DATE	'YYYYMMDD' 'YYYY-MM-DD'	'20120212' '2012-02-12'
TIME	'hh:mm:ss.nnnnnnn'	'12:30:15.1234567'
DATETIMEOFFSET	'YYYYMMDD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYY-MM-DD hh:mm:ss.nnnnnnn [+|-]hh:mm' 'YYYYMMDD' 'YYYY-MM-DD'	'20120212 12:30:15.1234567 +02:00' '2012-02-12 12:30:15.1234567 +02:00' '20120212' '2012-02-12'

Datum und Zeit III

- In SQL Server ist es nicht möglich, Datums- oder Zeitwerte explizit einzugeben
- Eingabe als Zeichenliterale und Konvertierung (expl. / impl.)
- Formate sind sprachabhängig und könnten zu Problemen führen
- Best Practices: sprachunabhängige Formate / Zeichenfolgen verwenden

```
SELECT OrderID, CustomerID, OrderDate
FROM dbo.Orders
WHERE OrderDate = ‘20070825‘;
```

Datum und Zeit IV

- Datumswerte, die aus Zeichenliteralen konvertiert wurden, lassen häufig die Uhrzeitangaben aus
  - ➢ Abfragen mit Gleichheitsoperator entsprechen Mitternacht

```
SELECT OrderID, CustomerID, OrderDate
FROM dbo.Orders
WHERE OrderDate = ‘20070825‘;
```

  - ➢ Bei Abfragen Zeitwerte beachten! Bereichsfilter verwenden

```
SELECT OrderID, CustomerID, OrderDate
FROM dbo.Orders
WHERE OrderDate >= ‘20070825‘
AND OrderDate < ‘20070826‘;
```

Aktuelle Zeit

- Funktionen, welche das aktuelle Datum und Uhrzeit zurückgeben

Funktion	Rückgabetyp	Anmerkungen
GETDATE()	datetime	Das aktuelle Datum und die aktuelle Uhrzeit. Kein Zeitzonenoffset
GETUTCDATE()	datetime	Das aktuelle Datum und die aktuelle Uhrzeit in UTC
CURRENT_TIMESTAMP	datetime	Das aktuelle Datum und die aktuelle Uhrzeit. Kein Zeitzonenoffset. ANSI-Standard
SYSDATETIME()	datetime2	Das aktuelle Datum und die aktuelle Uhrzeit. Kein Zeitzonenoffset
SYSUTCDATETIME()	datetime2	Das aktuelle Datum und die aktuelle Uhrzeit in UTC
SYSDATETIMEOFFSET()	datetimeoffset	Das aktuelle Datum und die aktuelle Uhrzeit. Beinhaltet Zeitzonenoffset

Arbeiten mit Datum und Zeit

Funktion	Syntax	Anmerkungen
DATEADD()	DATEADD(datepart, interval, date)	Ergänzt das Datum mit einem Intervall und gibt den Datentyp des Datums zurück
EOMONTH()	EOMONTH(start_date, interval)	Gibt als Startdatum den letzten Tag des Monats mit optionalem Offset zurück
SWITCHOFFSET()	SWITCHOFFSET(datetimeoffset, time_zone)	Ändert den Zeitzonenoffset
TODATETIMEOFFSET()	TODATETIMEOFFSET(expression, time_zone)	Konvertiert datetime2 in datetimeoffset
DATEDIFF()	DATEDIFF(datepart, start_date, end_date)	Gibt die Differenz zwischen zwei Datumsangaben in dateparts zurück
ISDATE()	ISDATE(expression)	Bestimmt, ob eine datetime- oder smalldate-Zeitangabe ein gültiger Wert ist
DATENAME()	DATENAME(interval, date)	Gibt dem Datumsanteil seinen Namen (Wochentag, Monat)
DATEPART()	DATEPART(interval, date)	Spaltet von dem Datum das gewünschte Intervall ab

ISNUMERIC

- Prüft, ob ein Eingabeausdruck ein gültiger numerischer Datentyp ist (einschließlich float und money)
- Gibt 1 zurück, wenn numerisch; 0, wenn nicht

```
SELECT ISNUMERIC('SQL') AS Result;
```

	Result
1	0

```
SELECT ISNUMERIC('22.934') AS Result;
```

	Result
1	1

IIF

- Gibt einen von zwei Werten zurück, nachdem ein logischer Test durchgeführt wurde
- Syntax und Beispiel:

```
IIF(<Ausdruck>, <True-Wert>, <False-Wert>)
```

```
SELECT ProductID, ProductName, UnitPrice,
      IIF(UnitPrice > 40, ‚teuer‘, ‚billig‘) AS Preiskategorie
FROM dbo.Products;
```

CHOOSE

- Gibt ein Element einer Liste an einem Indexwert zurück

```
SELECT CHOOSE(2, ‘Geflügel‘, ‘Fisch‘, ‘Paarhufer‘) AS Lecker;
```

	Lecker
1	Fisch

ISNULL

- Ersetzt NULL durch einen angegebenen Wert

```
SELECT CustomerID, City, ISNULL(Region, ‘unbekannt‘) AS
            Region, Country
FROM dbo.Customers;
```

	CustomerID	City	Region	Country
1	COMMI	Sao Paulo	SP	Brazil
2	CONSH	London	unbekannt	UK
3	DRACD	Aachen	unbekannt	Germany
4	DUMON	Nantes	unbekannt	France
5	EASTC	London	unbekannt	UK
6	ERNSH	Graz	unbekannt	Austria
7	FAMIA	Sao Paulo	SP	Brazil

COALESCE

- Gibt den ersten Wert aus einer Liste zurück, der nicht NULL ist

```
SELECT CustomerID, City, Country,
      country + ‘, ‘ + COALESCE(Region, ‘, ‘) + City AS Ort
FROM dbo.Customers;
```

NULLIF

- Vergleicht zwei Ausdrücke, gibt NULL bei Gleichheit zurück,
- Den ersten Wert bei Ungleichheit

```
SELECT NULLIF(’Hallo’,’Hallo’);   → Ergebnis NULL
```

```
SELECT NULLIF(’Hallo’,’Holla’);   → Ergebnis Hallo
```