

SQL Server 2022

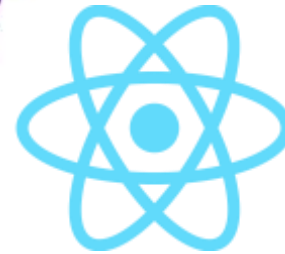
Neuerungen in der Übersicht



Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor



Azure Cosmos DB



Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
 - Problemanalyse und Lösungen
 - Technologieentscheidungen



A photograph of a rusted, abandoned car, possibly a pickup truck, in a field of tall green grass. In the background, there is a rocky cliff face and some bare trees. The scene is overgrown and appears to be in a rural or wilderness area.

SQL Server & Tooling

SQL Server

- RDBMS - Relational database management system
- SSAS – SQL Server Analysis Services
- SSIS – SQL Server Integration Services
- SSRS – SQL Server Reporting Services

SQL Server 2022

- Windows
- Red Hat Enterprise Server, SUSE Linux Enterprise Server & Ubuntu
- Docker

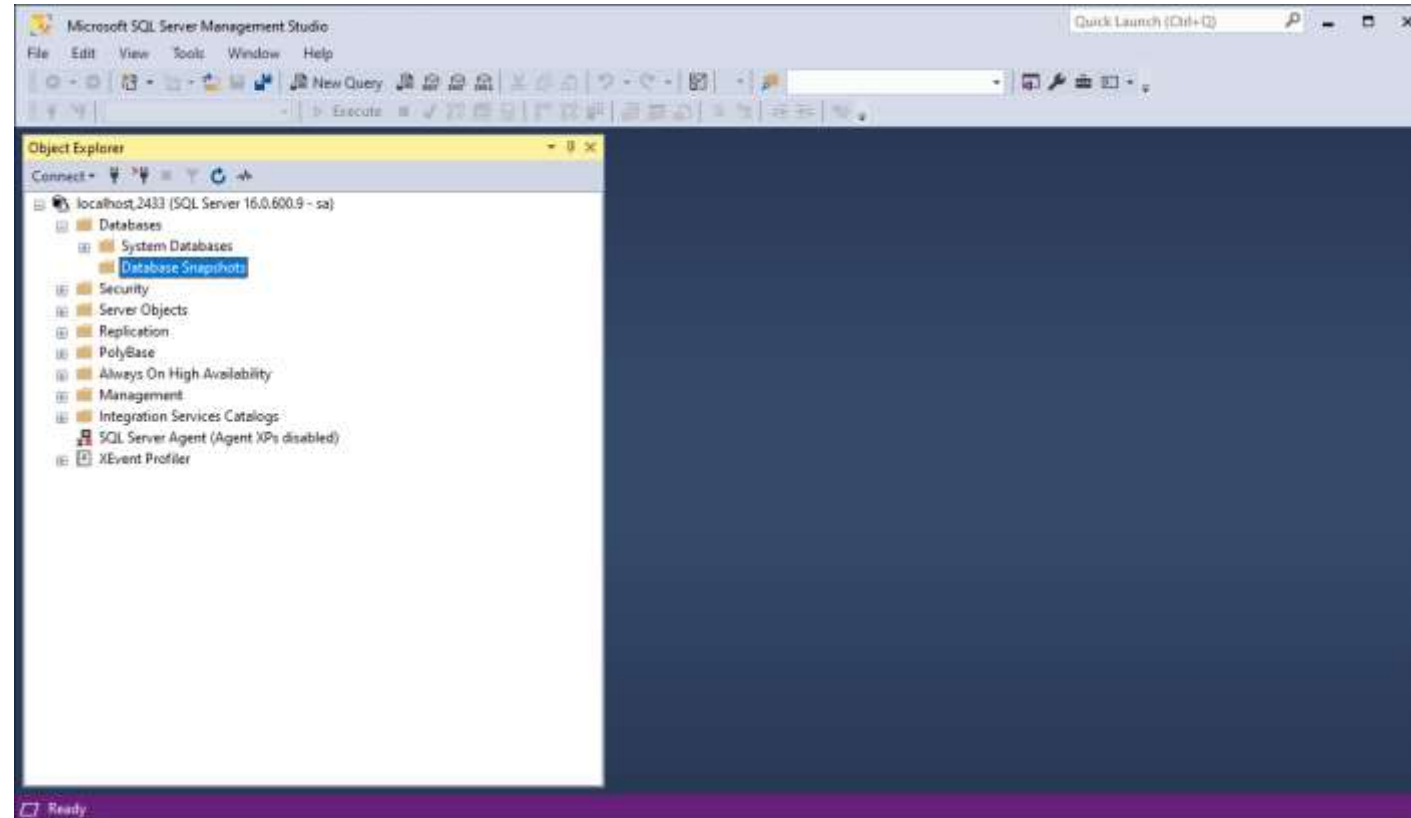
```
docker run -e ACCEPT_EULA=Y  
           -e SA_PASSWORD=P@ssw0rd99  
           -p 2433:1433  
           -d mcr.microsoft.com/mssql/server:2022-latest
```



https://hub.docker.com/_/microsoft-mssql-server

SQL Server Management Studio 19.0

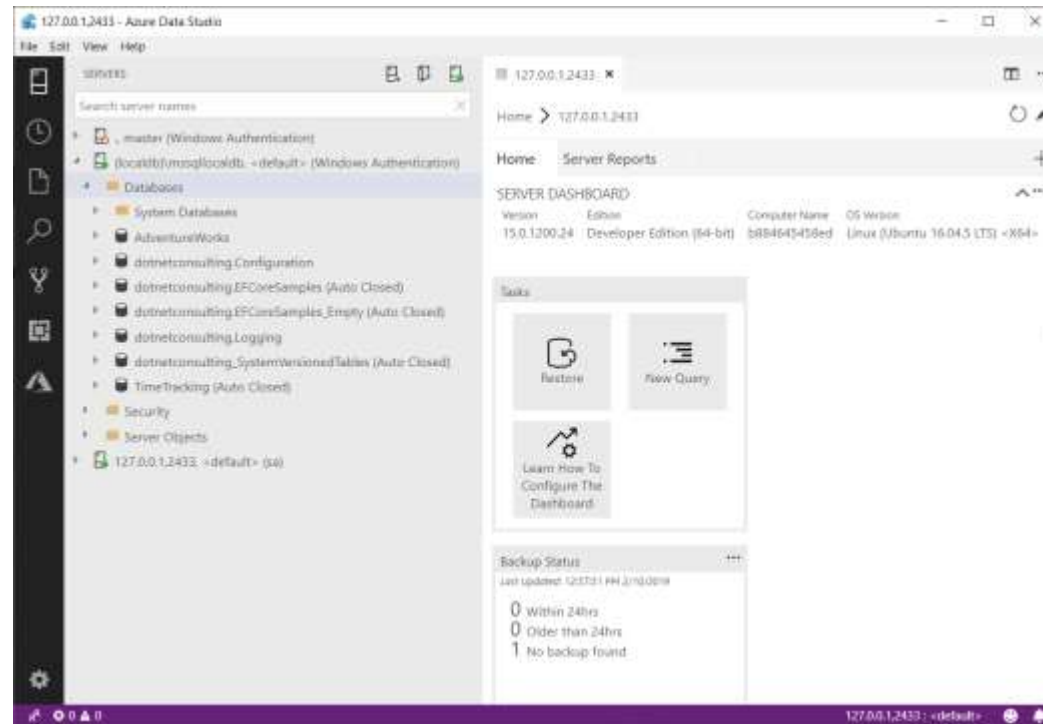
Windows only



<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms-19?view=sql-server-ver16>

Azure Data Studio

- Basiert auf Visual Studio Code
 - Windows
 - macOS
 - Linux



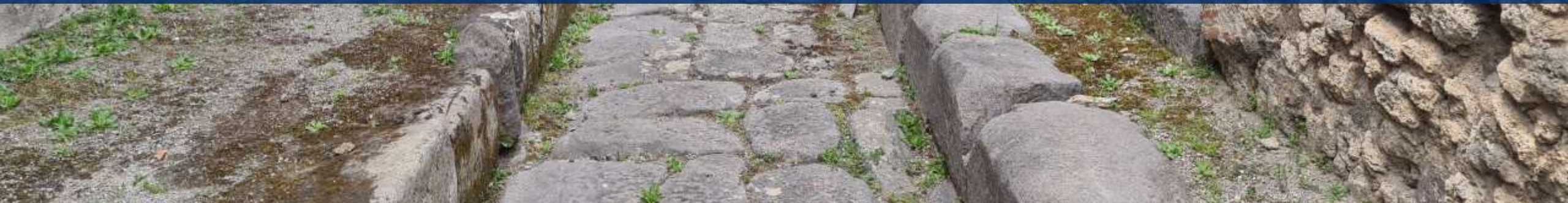
<https://docs.microsoft.com/en-us/sql/azure-data-studio>

Sonstiges

- VS Code
- SQLPackage.exe
- Sqlcmd.exe
- MSSQL-CLI

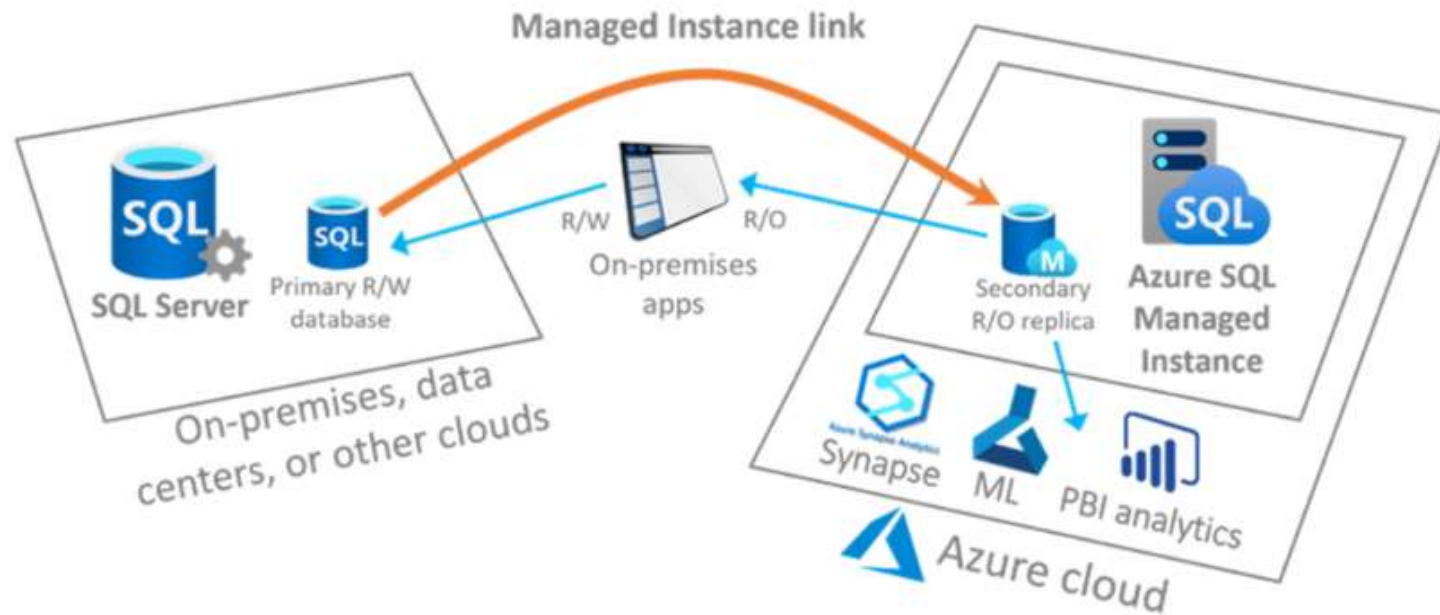


Availability



Link to Azure SQL Managed Instance

Replikation in die Azure Cloud



<https://cloudblogs.microsoft.com/sqlserver/2022/11/16/link-feature-for-azure-sql-managed-instance-connecting-sql-server-2022-to-the-cloud-reimagined/>



Security

Ledger Tables

Tabelle mit Nachweis von Manipulationen auf Dateiebene via Blockchain

Optionen

- Aktualisierbar
- Nur anfügen

 Demo 

Dynamic data masking

UNMASK-Recht auf Database-, Schema-, Table- oder Column-Ebene

```
-- UNMASK-Recht auf Spalte
GRANT UNMASK ON dbo.Mitarbeiter(Gehalt) TO UserMitUNMASK;
-- UNMASK-Recht auf Tabelle
GRANT UNMASK ON dbo.Mitarbeiter TO UserMitUNMASK;
-- UNMASK-Recht auf Schema
GRANT UNMASK ON SCHEMA::Data TO UserMitUNMASK;
-- UNMASK-Recht auf Datenbank
GRANT UNMASK TO UserMitUNMASK;
```

 Demo 



Performance

XML compression

XML ist umfangreich und kann nun komprimiert werden

- Out of row content
- Indizes

```
-- Compression von XML-Inhalten
```

```
ALTER INDEX PK_Table ON [dbo].[Table] REBUILD WITH (XML_COMPRESSION = ON);
```

A photograph of a bull's head with large, dark, curved horns mounted on a wooden wall. The background shows a thatched roof made of dried grass or straw. A semi-transparent blue horizontal band is overlaid across the middle of the image, containing the text 'Language (T-SQL)'.

Language (T-SQL)

Language

- Unterscheidbare Werte/ NULL Handling
- Zeichenketten trimmen
- Time series functions
- JSON-Funktionalitäten
- SELECT ... WINDOW clause
- FIRST_VALUE/ LAST_VALUE
- Aggregate functions
- Sonstige T-SQL Funktionen

Unterscheidbare Werte/ NULL Handling

“Gleich”/ “Ungleich” als “nicht unterscheidbar”/ “unterscheidbar”

```
SELECT * FROM sys.Databases WHERE 0 IS NOT DISTINCT FROM 0; -- =  
SELECT * FROM sys.Databases WHERE 0 IS NOT DISTINCT FROM 1; -- !=  
SELECT * FROM sys.Databases WHERE 0 IS NOT DISTINCT FROM NULL;  
SELECT * FROM sys.Databases WHERE NULL IS NOT DISTINCT FROM NULL;
```

A IS DISTINCT FROM B will decode to: $((A \neq B \text{ OR } A \text{ IS NULL OR } B \text{ IS NULL}) \text{ AND NOT } (A \text{ IS NULL AND } B \text{ IS NULL}))$

A IS NOT DISTINCT FROM B will decode to: $(\text{NOT } (A \neq B \text{ OR } A \text{ IS NULL OR } B \text{ IS NULL}) \text{ OR } (A \text{ IS NULL AND } B \text{ IS NULL}))$



IS NOT DISTINCT.sql

Zeichenketten trimmen

Unerwünschte Zeichen vom Anfang/ Ende einer Zeichenkette entfernen

<code>LTRIM()</code> / <code>RTRIM()</code>	Akzeptiert nun eine Auswahl an Zeichen, die entfernt werden
<code>TRIM()</code>	Akzeptiert nun eine Auswahl an Zeichen, die entfernt werden und die Angabe wo sie zu entfernen sind

LEADING	Entfernt vom Anfang (wie <code>LTRIM()</code>)
TRAILING	Entfernt vom Ende (wie <code>RTRIM()</code>)
BOTH	Kombination aus LEADING und TRAILING

LTRIM() / RTRIM() / TRIM()

Angaben, was wo entfernt werden soll

```
SELECT LTRIM(@caption, '-= ') AS [LTRIM],  
       RTRIM(@caption, '-= ') AS [RTRIM];  
  
SELECT TRIM(LEADING '-= ' FROM @caption) AS [LEADING],  
       TRIM(TRAILING '-= ' FROM @caption) AS [TRAILING],  
       TRIM(BOTH '-= ' FROM @caption) AS [BOTH];
```




TRIM.sql

Time series functions

Funktionen für Zeit- und Zahlenfolgen

DATE_BUCKET ()	Aufteilung von Zeiträumen in Intervalle seit Beginn
GENERATE_SERIES ()	Erzeugt eine Zahlenserie (Von..Bis, Schrittweite)

DATE_BUCKET ()

Aufteilung von Zeiträumen in Intervalle

```
DECLARE @date DATETIME2 = GETDATE();
DECLARE @origin DATETIME2 = '2022-01-01 00:00:00.000' -- null => 1900-01-01 00:00:00.000
SELECT 'Day', @date, DATE_BUCKET(DAY, 2, @date, @origin)
UNION ALL
SELECT 'Day', @date, DATE_BUCKET(DAY, 2, DATEADD(Day, 1, @date), @origin)
UNION ALL
SELECT 'Day', @date, DATE_BUCKET(DAY, 2, DATEADD(Day, 2, @date), @origin)
UNION ALL
SELECT 'Day', @date, DATE_BUCKET(DAY, 2, DATEADD(Day, 3, @date), @origin);
```




DATE_BUCKET.sql

DATETRUNC ()

Entfernen von Datumselementen

```
DECLARE @d DATETIME2 = GETDATE();  
SELECT 'Year', DATETRUNC(YEAR, @d); -- 2022-01-01 00:00:00.0000000  
SELECT 'Quarter', DATETRUNC(QUARTER, @d); --2022-10-01 00:00:00.0000000  
SELECT 'Month', DATETRUNC(MONTH, @d); -- 2022-12-01 00:00:00.0000000  
SELECT 'Week', DATETRUNC(WEEK, @d); -- 2022-12-11 00:00:00.0000000  
SELECT 'Iso_week', DATETRUNC(ISO_WEEK, @d); -- 2022-12-12 00:00:00.0000000  
SELECT 'DayOfYear', DATETRUNC(DAYOFYEAR, @d); -- 2022-12-17 00:00:00.0000000  
SELECT 'Day', DATETRUNC(DAY, @d); -- 2022-12-17 00:00:00.0000000  
SELECT 'Hour', DATETRUNC(HOUR, @d); -- 2022-12-17 19:00:00.0000000  
SELECT 'Minute', DATETRUNC(MINUTE, @d); -- 2022-12-17 19:45:00.0000000  
SELECT 'Second', DATETRUNC(SECOND, @d); -- 2022-12-17 19:45:17.0000000
```



DATETRUNC.sql

GENERATE_SERIES ()

Erzeugen einer numerischen Serie von Werten

```
-- Von 1 bis <=50 in 5er Schritten => 1..6..11..41..46
```

```
SELECT value FROM GENERATE_SERIES(/* Start */ 1, /*STOP*/ 50, /*STEP*/ 5);
```

```
-- Alle Wochen 2022
```

```
SELECT DATEADD(WEEK, value - 1, '2021-01-03'), value FROM GENERATE_SERIES(1, 52);
```



`GENERATE_SERIES.sql`

JSON-Funktionalitäten

Erweiterung der JSON-Funktionalitäten

<code>JSON_ARRAY ()</code>	Erzeugt ein JSON-Array (Neu)
<code>JSON_OBJECT ()</code>	Erzeugt ein einfaches JSON-Objekt (Neu)
<code>ISJSON ()</code>	Erweitert um die neue JSON Typen-Angabe
<code>JSON_PATH_EXISTS ()</code>	Prüft, ob ein JSON-Objekt den angegebenen Pfad besitzt (Neu)

JSON Type	
VALUE	Testet auf einen Wert (object, array, number, string, true, false, null)
ARRAY	Testet auf ein JSON-Array
OBJECT	Testet auf ein JSON-Objekt
SCALAR	Testet auf einen gültigen Skalarwert - Zahl, String, etc.



JSON.sql

SELECT ... WINDOW clause

Benanntes Abfragefenster für Vereinfachung von Abfragen

```
SELECT SalesOrderID, ProductID, OrderQty
    ,SUM(OrderQty) OVER win AS Total
    ,AVG(OrderQty) OVER win AS "Avg"
    ,COUNT(OrderQty) OVER win AS "Count"
    ,MIN(OrderQty) OVER win AS "Min"
    ,MAX(OrderQty) OVER win AS "Max"
FROM Sales.SalesOrderDetail
WHERE SalesOrderID IN(43659,43664)
WINDOW win AS (PARTITION BY SalesOrderID); ←
```




WINDOW Klausel.sql

FIRST_VALUE/ LAST_VALUE

Umgang mit NULL kontrollieren

```
SELECT Name, ListPrice,  
       FIRST_VALUE(Name) RESPECT NULLS OVER (ORDER BY ListPrice ASC)  
AS LeastExpensive  
FROM Production.Product  
WHERE ProductSubcategoryID = 37;
```



IGNORE NULLS	Ignoriert alle NULL Werte im Datensatz
RESPECT NULLS	Berücksichtigt alle NULL Werte im Datensatz



FIRST_VALUE LAST_VALUE.sql

Aggregate functions

Annäherungsfunktionen mit akzeptablen Fehlerbereich

<code>APPROX_PERCENTILE_DISC()</code>	Effiziente Alternative für größere Datenmengen für <code>PERCENTILE_DISC()</code>
<code>APPROX_PERCENTILE_CONT()</code>	Effiziente Alternative für größere Datenmengen für <code>PERCENTILE_CONT()</code>

APPROX_PERCENTILE_CONT | DISC ()

Percentile mit interpolierten/ diskreten Werten

-- Interpolierte Werte

```
SELECT DeptId,  
       APPROX_PERCENTILE_CONT(0.10) WITHIN GROUP(ORDER BY Salary) AS 'P10',  
       APPROX_PERCENTILE_CONT(0.90) WITHIN GROUP(ORDER BY Salary) AS 'P90'  
FROM #Employee  
GROUP BY DeptId;
```

-- Diskrete Werte

```
SELECT DeptId,  
       APPROX_PERCENTILE_DISC(0.10) WITHIN GROUP(ORDER BY Salary) AS 'P10',  
       APPROX_PERCENTILE_DISC(0.90) WITHIN GROUP(ORDER BY Salary) AS 'P90'  
FROM #Employee  
GROUP BY DeptId;
```




`APPROX_PERCENTILE.sql`

Sonstige T-SQLFunktionen

<code>GREATEST () / LEAST ()</code>	“Horizontales” Maxi-/ Minimum
<code>STRING_SPLIT ()</code>	Auf Wunsch mit laufender Nummer (Ordinal)
<code>ALTER TABLE..ADD CONSTRAINT</code>	Erstellung einer Einschränkung (Constraint) anhalten oder fortführen
<code>CREATE STATISTICS.. AUTO_DROP = ON</code>	Statistken, die bei einer Schemaänderung automatisch gelöscht werden (und nicht blockieren)

GREATEST () / LEAST ()

“Horizontales” Maxi-/ Minimum

```
-- Horizontales Maxi-/ Minimum  
SELECT GREATEST( '6.62', 3.1415, N'7' ) AS 'GREATEST',  
       LEAST( '6.62', 3.1415, N'7' ) AS 'LEAST';
```



GREATEST_LEAST.sql

STRING_SPLIT()

Auf Wunsch mit laufender Nummer (Ordinal)

```
-- SQL Server 2016+
```

```
SELECT * FROM STRING_SPLIT('Lorem ipsum dolor sit amet.', ' ');
```

```
-- SQL Server 2022+
```

```
SELECT * FROM STRING_SPLIT('Lorem ipsum dolor sit amet.', ' ', 1);
```




STRING_SPLIT.sql

Bit Operations

Bit Operationen als Funktionen

```
-- RIGHT_SHIFT()/ LEFT_SHIFT()
SELECT dbo.ConvertToBit(RIGHT_SHIFT(@number, 1)) '>> 1';
SELECT dbo.ConvertToBit(LEFT_SHIFT(@number, 1)) '<< 1';

-- BIT_COUNT()
SELECT dbo.ConvertToBit(@number), BIT_COUNT(@number) 'Number of 1s';

-- SET_BIT()/ GET_BIT()
SELECT GET_BIT(@number, 0) 'Pos 0', GET_BIT(@number, 1) 'Pos 1';
SELECT dbo.ConvertToBit(SET_BIT(@number, 0)) 'Set Pos 0';
```

Demo

Bit Operations.sql

Resumable add table constraints

Table Constraints können unterbrochen (und fortgeführt) werden

```
ALTER TABLE [dbo].[MyTable]
ADD CONSTRAINT PK_Constraint PRIMARY KEY CLUSTERED (...)
WITH (ONLINE = ON, MAXDOP = 2, RESUMABLE = ON, MAX_DURATION = 240);
```

```
ALTER INDEX ALL ON [dbo].[MyTable] PAUSE;
```

```
ALTER INDEX ALL ON [dbo].[MyTable] RESUME;
```

```
SELECT sql_text, state_desc, percent_complete FROM
sys.index_resumable_operations;
```

sql_text	state_desc	percent_complete
ALTER TABLE [dbo].[MyTable] (...)	RUNNING	43.552

 Demo 

CREATE STATISTICS WITH AUTO_DROP

Statistiken so erzeugen, das keine Schema-Änderung blockiert wird, sondern die Statistik automatisch löscht

```
UPDATE STATISTICS ... WITH AUTO_DROP = ON;
```

```
CREATE STATISTICS ... WITH AUTO_DROP = ON;
```

 Demo 

Fragen? Jetzt oder später!

Kontakt

 **E-Mail**
tkansy@dotnetconsulting.eu

 **Telefon**
[+49 \(0\) 6187 / 2009090](tel:+49(0)61872009090)

 **Microsoft Teams**
[Meet now](#)

 **LinkedIn**
[Link me](#)

 **XING**
[Xing me](#)

 **X (Twitter)**
[@tkansy](#)



www.dotnetconsulting.eu

SQL Server meets .NET (Core)- professionally!



Ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen mit Microsoft SQL Server- mit Allem, was dazu gehört- und was man vielleicht weglassen sollte.

