Hochschule Karlsruhe – Data Engineering
WS 2021/2022
DSCB330 – Vorlesung 11 – Data Warehouse 2

Foto: Quirin Leppert

Themenübersicht

Data Integration

- Data Formats (csv, XML, json)
- Extract, Transform, Load
- Object Relation Mapper (ORM)
- Staging

Data Processing

- Relationale Datenbanken
- nicht-relationale Datenbanken
- Resource Description Framework (RDF)
- Ontologien
- **Data Warehouse**

Data Modelling

- Serialisierung
- OPC UA
- MQTT
- Pub/Sub
- Data pipelines
  - Apache Airflow
  - gRPC

Web-Service Architektur

- Front-End
- Backend for Frontend (BFF)
- Micro Services
- Docker Container

Security

- Security ist wichtig in allen Phasen der Softwareentwicklung und Datenbereitstellung.

Übung

- Erstellung eines Daten-Modells einer prozesstechnischen Anlage
- Statische Daten
- Dynamische Daten
- Auswertung der Daten

Data Warehouse
Slowly Changing Dimensions (3)

Behandlung von Slowly Changing Dimensions

+ Typ 1: Überschreibung des alten Attributwerts mit einem neuen Attributwert
- Nachteil: Verlust der Historie
- Anwendung, wenn die Änderung aufgrund eines fehlerhaften Eintrags in der Dimensions-Tabelle erfolgt.

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Slowly Changing Dimensions (4)

Behandlung von Slowly Changing Dimensions

+ Typ 2: Versionierung der Tupel in der Dimensions-Tabelle
- Nachteil: schlechtere Performance bei Abfragen (z.B. Verkaufszahlen des Produkts `prod1`)
- Im Beispiel wird in der Dimensions-Tabelle `Product` für das Produkt `prod1` eine neue Zeile mit der neuen Kategorie `cat2` ergänzt. Zusätzlich werden zwei Spalten für die Zeitspanne der Gültigkeit ergänzt.
- Mögliche Ergänzung um eine Spalte (Flag), die den Status der Zeile erläutert.

Product Key	Product Name	Discontinued	Category Name	Description	From	To	Row Status
p1	prod1	No	cat1	desc1	2010-01-01	2011-12-31	Expired
p11	prod1	No	**cat2**	desc2	2012-01-01	9999-12-31	Current
...	...	...	...	...	...	...	...

`Product` dimension table mit Status-Flag

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

Ursprünglicher `Product` dimension table (Star-Schema)

Product Key	Product Name	Discontinued	Category Name	Description	From	To
p1	prod1	No	cat1	desc1	2010-01-01	2011-12-31
p11	prod1	No	cat2	desc2	2012-01-01	9999-12-31
p2	prod2	No	cat1	desc1	2012-01-01	9999-12-31
p3	prod3	No	cat2	desc2	2012-01-01	9999-12-31
p4	prod4	No	cat2	desc2	2012-01-01	9999-12-31

angepasster `Product` dimension table

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Slowly Changing Dimensions (5)

Behandlung von Slowly Changing Dimensions

- \+ Typ 3: Einfügen einer Spalte für jede Änderung (attribute addition)
- \- Nachteil: Fehlende Historie (Gültigkeit)
- \- Im Beispiel wird bei einer Änderung der Kategorie eine neue Spalte für die Kategorie (`New Category`) und für die Beschreibung (`New Description`) eingefügt.

ProductKey	ProductName	Discontinued	CategoryName	Description
p1	prod1	No	cat1	desc1
p2	prod2	No	cat1	desc1
p3	prod3	No	cat2	desc2
p4	prod4	No	cat2	desc2

Ursprünglicher `Product` dimension table (Star-Schema)

Product Key	Product Name	Discontinued	Category Name	New Category	Description	New Description
p1	prod1	No	**cat1**	**cat2**	**desc1**	**desc2**
p2	prod2	No	cat1	Null	desc1	Null
p3	prod3	No	cat2	Null	desc2	Null
p4	prod4	No	cat2	Null	desc2	Null

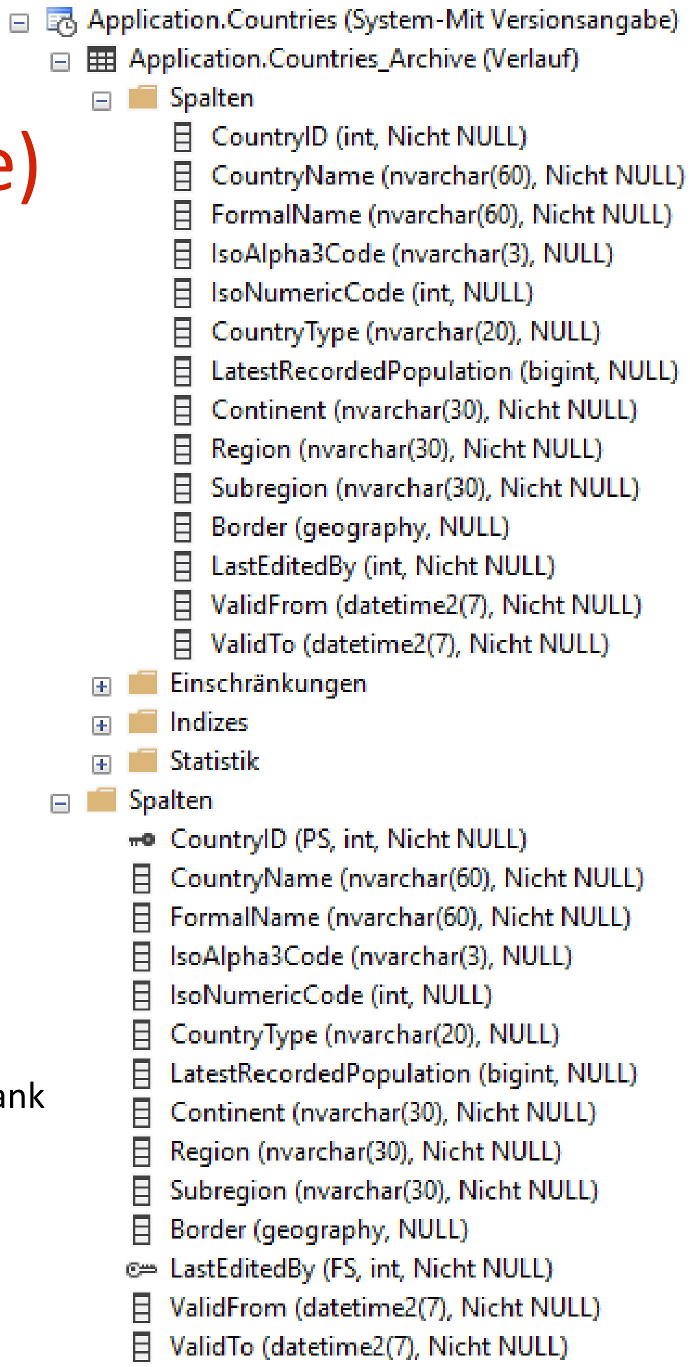
`Product` dimension table mit neuer Spalte bei Änderung der Kategorie

Quelle: A. Vaisman and E. Zimányi, Data Warehouse Systems, Data-Centric Systems and Applications, DOI 10.1007/978-3-642-54655-6, © Springer-Verlag Berlin Heidelberg 2014

Data Warehouse
Slowly Changing Dimensions (Temporal Table)

Nutzung von Temporal Tables mit MS SQL Server

+ T-SQL-Befehl `HISTORY_TABLE`
+ A system-versioned table exactly behaves like a dimension with type 2 changing behavior for all of its columns.



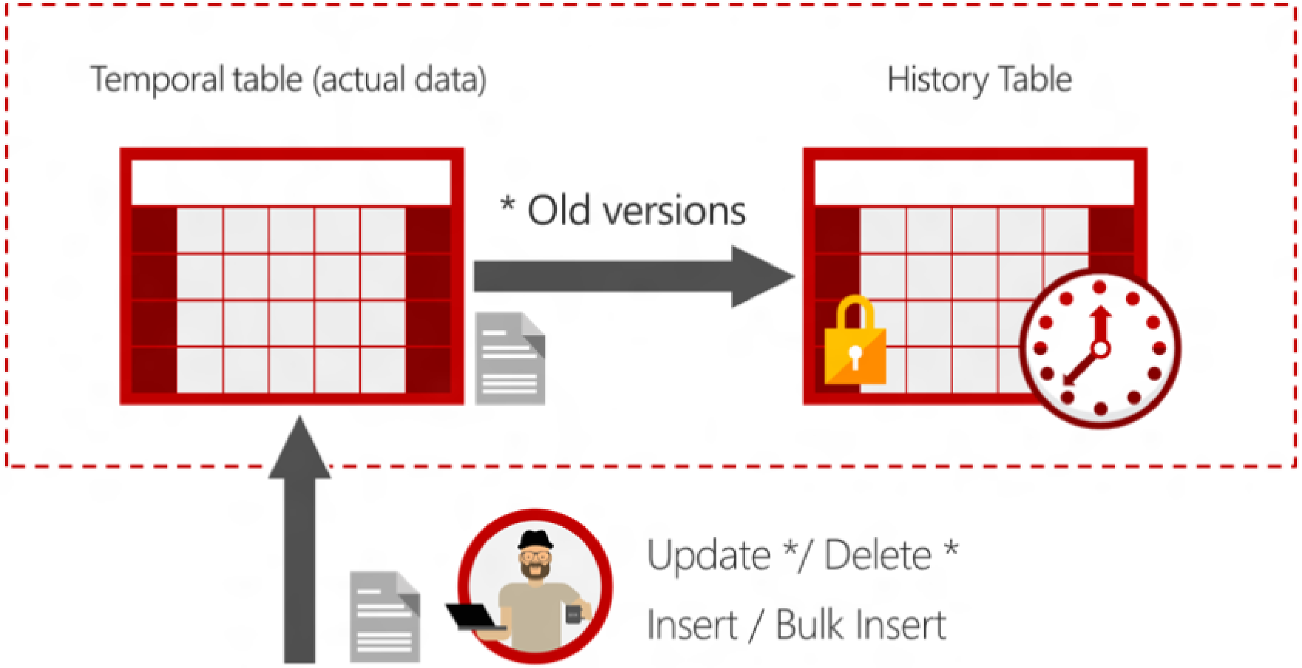
Beispiel aus WideWorldImporters-Datenbank

Quelle: https://www.mssqltips.com/sqlservertip/3680/introduction-to-sql-server-temporal-tables/

Data Warehouse
Temporal Table

Nutzung von Temporal Tables mit MS SQL Server

- Every temporal table has two explicitly defined columns, each with a datetime2 data type.
- System-versioning for a table is implemented as a pair of tables, a current table and a history table.
- Period start column: The system records the start time for the row in this column, typically denoted as the SysStartTime column.
- Period end column: The system records the end time for the row in this column, typically denoted as the SysEndTime column.



Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?redirectedfrom=MSDN&view=sql-server-ver15

Data Warehouse
Temporal Table

Beispiel

- Erzeugen einer Tabelle mit einem temporal table

```
CREATE TABLE dbo.Employee
(
  [EmployeeID] int NOT NULL PRIMARY KEY CLUSTERED
  , [Name] nvarchar(100) NOT NULL
  , [Position] varchar(100) NOT NULL
  , [Department] varchar(100) NOT NULL
  , [Address] nvarchar(1024) NOT NULL
  , [AnnualSalary] decimal (10,2) NOT NULL
  , [ValidFrom] datetime2 GENERATED ALWAYS AS ROW
START
  , [ValidTo] datetime2 GENERATED ALWAYS AS ROW END
  , PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
 )
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE =
dbo.EmployeeHistory));
```

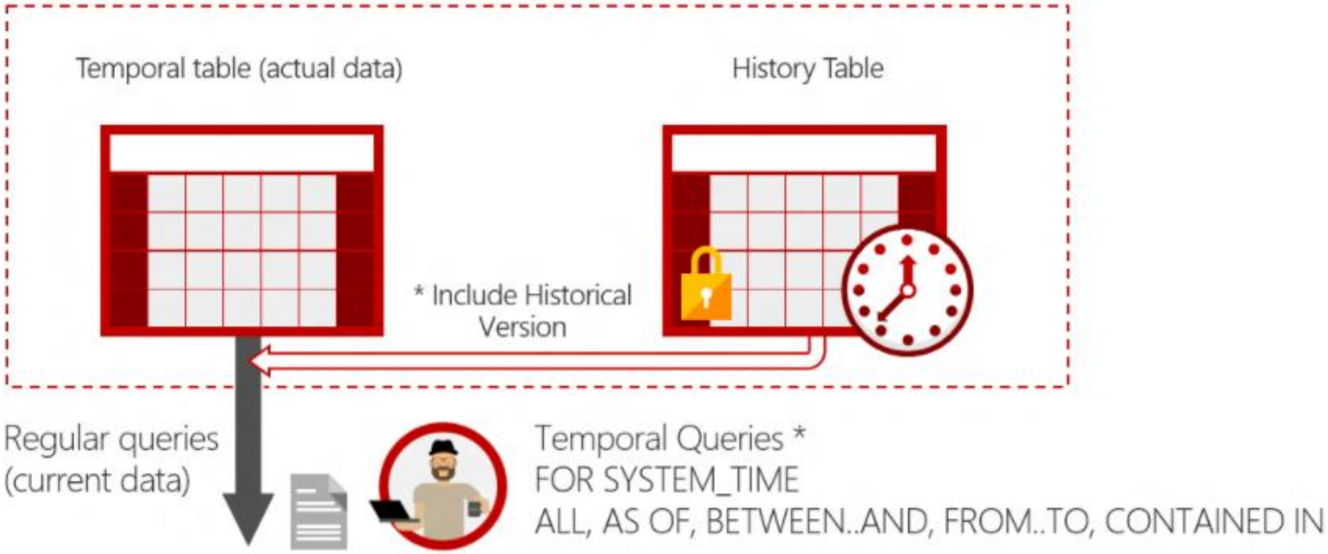
Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?redirectedfrom=MSDN&view=sql-server-ver15

Data Warehouse
Temporal Table

Beispiel

+ Abfragen einer Tabelle mit einem temporal table

```
SELECT * FROM Employee
  FOR SYSTEM_TIME
    BETWEEN '2014-01-01 00:00:00.0000000' AND
'2015-01-01 00:00:00.0000000’
      WHERE EmployeeID = 1000 ORDER BY ValidFrom;
```



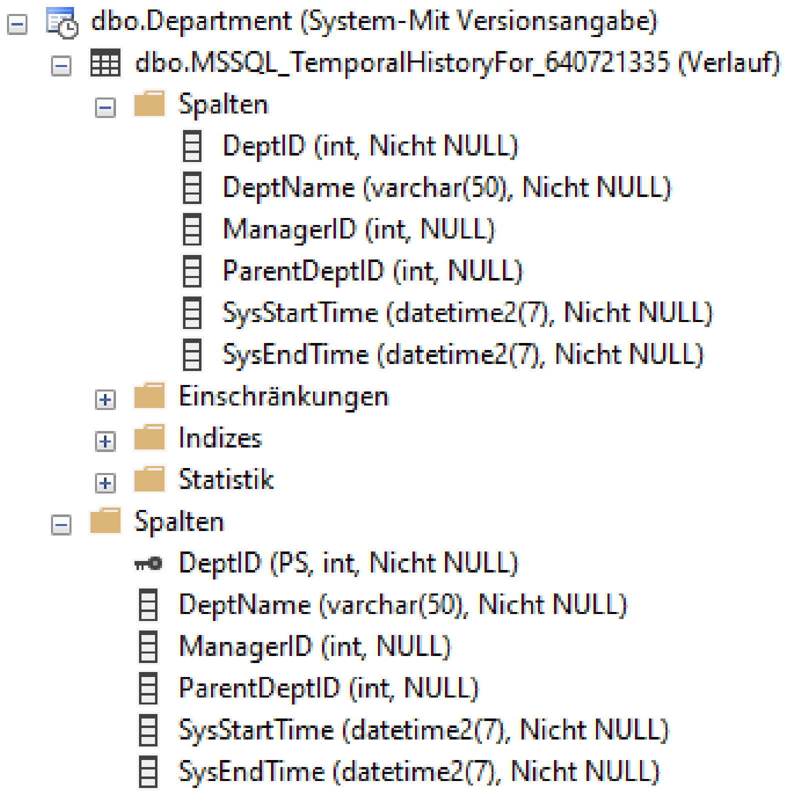
Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/temporal-tables?redirectedfrom=MSDN&view=sql-server-ver15

Data Warehouse
Temporal Table

Beispiel 2

+ Erzeugen einer Tabelle mit einem temporal table

```sql
CREATE TABLE Department
(    DeptID INT NOT NULL PRIMARY KEY CLUSTERED
   , DeptName VARCHAR(50) NOT NULL
   , ManagerID INT NULL
   , ParentDeptID INT NULL
   , SysStartTime DATETIME2 GENERATED ALWAYS AS ROW
START NOT NULL
   , SysEndTime DATETIME2 GENERATED ALWAYS AS ROW
END NOT NULL
   , PERIOD FOR SYSTEM_TIME
(SysStartTime,SysEndTime)
)
WITH (SYSTEM_VERSIONING = ON);
```



Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/creating-a-system-versioned-temporal-table?view=sql-server-ver15

Data Warehouse
Temporal Table

Beispiel 2

- Einfügen von Daten
- Der Zeitpunkt des Einfügens wird automatisch gespeichert

```sql
INSERT INTO [dbo].[Department]
  (  [DeptID]
        , [DeptName]
        , [ManagerID]
        ,[ParentDeptID]
  )
     VALUES
       (  10
       , 'Marketing'
       , 101
       , 1
       ) ;
```

- Abfrage des Eintrags mit Zeitpunkt des Einfügens

```sql
SELECT [DeptID], [DeptName],
[SysStartTime],[SysEndTime]
FROM [dbo].[Department]
--FOR SYSTEM_TIME AS OF '2021-12-
07T16:49:47.1219131Z' ;
```

Ergebnisse | Meldungen

	DeptID	DeptName	SysStartTime	SysEndTime
1	10	Marketing	2021-12-07 16:49:47.1219131	9999-12-31 23:59:59.9999999

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/modifying-data-in-a-system-versioned-temporal-table?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/relational-databases/tables/querying-data-in-a-system-versioned-temporal-table?view=sql-server-ver15

Data Warehouse
Temporal Table

Beispiel 2

- Änderung des Eintrags

```
UPDATE [dbo].[Department] SET [ManagerID] = 501
WHERE [DeptID] = 10
```

- Erneute Abfrage des Eintrags mit Zeitpunkt des Einfügens

```
SELECT [DeptID], [DeptName],
[SysStartTime],[SysEndTime]
FROM [dbo].[Department]
```

	DeptID	DeptName	SysStartTime	SysEndTime
1	10	Marketing	2021-12-08 08:16:11.9663472	9999-12-31 23:59:59.9999999

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/creating-a-system-versioned-temporal-table?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

+ Beispiel 1: Converting a Table to a Hierarchical Structure
+ Nutzung der AdventureWorks2019-Datenbank und des AdventureWorksDW2019-Data Warehouses. https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms
+ Schritt 1: Kopieren der Beispieltabelle HumanResources.Employee

```
USE AdventureWorks2019;
GO
  if OBJECT_ID('HumanResources.EmployeeDemo') is
not null
 drop table HumanResources.EmployeeDemo
 SELECT emp.BusinessEntityID AS EmployeeID,
emp.LoginID,
  (SELECT  man.BusinessEntityID FROM
HumanResources.Employee man
    WHERE
emp.OrganizationNode.GetAncestor(1)=man.Organizatio
nNode OR
    (emp.OrganizationNode.GetAncestor(1) = 0x AND
man.OrganizationNode IS NULL)) AS ManagerID,
       emp.JobTitle, emp.HireDate
INTO HumanResources.EmployeeDemo
FROM HumanResources.Employee emp ;
GO
```

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/tutorial-using-the-hierarchyid-data-type?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

+ Beispiel 1: Converting a Table to a Hierarchical Structure
+ Schritt 2: Struktur der Tabelle mit dem Manager als Hierarchie

```
SELECT
    Mgr.EmployeeID AS MgrID, Mgr.LoginID AS
Manager,
    Emp.EmployeeID AS E_ID, Emp.LoginID,
Emp.JobTitle
FROM HumanResources.EmployeeDemo AS Emp
LEFT JOIN HumanResources.EmployeeDemo AS Mgr
ON Emp.ManagerID = Mgr.EmployeeID
ORDER BY MgrID, E_ID
```

	MgrID	Manager	E_ID	LoginID	JobTitle
1	NULL	NULL	1	adventure-works\ken0	Chief Executive Officer
2	1	adventure-works\ken0	2	adventure-works\terri0	Vice President of Engineering
3	1	adventure-works\ken0	16	adventure-works\david0	Marketing Manager
4	1	adventure-works\ken0	25	adventure-works\james1	Vice President of Production
5	1	adventure-works\ken0	234	adventure-works\laura1	Chief Financial Officer
6	1	adventure-works\ken0	263	adventure-works\jean0	Information Services Manager
7	1	adventure-works\ken0	273	adventure-works\brian3	Vice President of Sales
8	2	adventure-works\terri0	3	adventure-works\roberto0	Engineering Manager
9	3	adventure-works\roberto0	4	adventure-works\rob0	Senior Tool Designer
10	3	adventure-works\roberto0	5	adventure-works\gail0	Design Engineer
11	3	adventure-works\roberto0	6	adventure-works\jossef0	Design Engineer
12	3	adventure-works\roberto0	7	adventure-works\dylan0	Research and Development Manager
13	3	adventure-works\roberto0	11	adventure-works\ovidiu0	Senior Tool Designer
14	3	adventure-works\roberto0	14	adventure-works\michael8	Senior Design Engineer
15	3	adventure-works\roberto0	15	adventure-works\sharon0	Design Engineer
16	7	adventure-works\dylan0	8	adventure-works\diane1	Research and Development Engineer
17	7	adventure-works\dylan0	9	adventure-works\gigi0	Research and Development Engineer
18	7	adventure-works\dylan0	10	adventure-works\michael6	Research and Development Manager
19	11	adventure-works\ovidiu0	12	adventure-works\thierry0	Tool Designer

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 3: Erstellen einer neuen Tabelle HumanResources.NewOrg unter Nutzung der **hierarchyid**-Spalte.

```
CREATE TABLE HumanResources.NewOrg
(
  OrgNode hierarchyid,
  EmployeeID int,
  LoginID nvarchar(50),
  ManagerID int
CONSTRAINT PK_NewOrg_OrgNode
  PRIMARY KEY CLUSTERED (OrgNode)
);
GO
```

- Erstellen einer temporären Tabelle, um die Anzahl der Unterknoten (Children) zu speichern.

```
CREATE TABLE #Children
   (
    EmployeeID int,
    ManagerID int,
    Num int
);
GO
```

- Erzeugen eines Index zur schnelleren Datenverarbeitung.

```
CREATE CLUSTERED INDEX tmpind ON
#Children(ManagerID, EmployeeID);
GO
```

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 4: Befüllen der temporären Tabelle

```
INSERT #Children (EmployeeID, ManagerID, Num)
SELECT EmployeeID, ManagerID,
  ROW_NUMBER() OVER (PARTITION BY ManagerID ORDER
BY ManagerID)
FROM HumanResources.EmployeeDemo
GO
```

- Inhalt der temporären Tabelle #Children

```
SELECT * FROM #Children ORDER BY ManagerID, Num
GO
```

Ergebnisse | Meldungen

	EmployeeID	ManagerID	Num
1	1	NULL	1
2	2	1	1
3	16	1	2
4	25	1	3
5	234	1	4
6	263	1	5
7	273	1	6
8	3	2	1
9	4	3	1
10	5	3	2
11	6	3	3
12	7	3	4
13	11	3	5
14	14	3	6
15	15	3	7

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 5: Befüllen der neuen Tabelle

```
WITH paths(path, EmployeeID)
AS (
-- This section provides the value for the root of
the hierarchy
SELECT hierarchyid::GetRoot() AS OrgNode,
EmployeeID
FROM #Children AS C
WHERE ManagerID IS NULL
UNION ALL
-- This section provides values for all nodes
except the root
SELECT
CAST(p.path.ToString() + CAST(C.Num AS varchar(30))
+ '/' AS hierarchyid),
C.EmployeeID
FROM #Children AS C
JOIN paths AS p
   ON C.ManagerID = P.EmployeeID
)
INSERT HumanResources.NewOrg (OrgNode,
O.EmployeeID, O.LoginID, O.ManagerID)
SELECT P.path, O.EmployeeID, O.LoginID, O.ManagerID
FROM HumanResources.EmployeeDemo AS O
JOIN Paths AS P
   ON O.EmployeeID = P.EmployeeID
GO
```

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 6: Abfrage der neuen Tabelle mit Konvertierung der **hierarchyid**-Spalte in ein Character-Format zur besseren Lesbarkeit.

```
SELECT OrgNode.ToString() AS LogicalNode, *
FROM HumanResources.NewOrg
ORDER BY LogicalNode;
GO
```

	LogicalNode	OrgNode	EmployeeID	LoginID	ManagerID
1	/	0x	1	adventure-works\ken0	NULL
2	/1/	0x58	2	adventure-works\terri0	1
3	/1/1/	0x5AC0	3	adventure-works\roberto0	2
4	/1/1/1/	0x5AD6	4	adventure-works\rob0	3
5	/1/1/2/	0x5ADA	5	adventure-works\gail0	3
6	/1/1/3/	0x5ADE	6	adventure-works\jossef0	3
7	/1/1/4/	0x5AE1	7	adventure-works\dylan0	3
8	/1/1/4/1/	0x5AE158	8	adventure-works\diane1	7
9	/1/1/4/2/	0x5AE168	9	adventure-works\gigi0	7
10	/1/1/4/3/	0x5AE178	10	adventure-works\michael6	7
11	/1/1/5/	0x5AE3	11	adventure-works\ovidiu0	3
12	/1/1/5/1/	0x5AE358	12	adventure-works\thierry0	11
13	/1/1/5/2/	0x5AE368	13	adventure-works\janice0	11
14	/1/1/6/	0x5AE5	14	adventure-works\michael8	3
15	/1/1/7/	0x5AE7	15	adventure-works\sharon0	3
16	/2/	0x68	16	adventure-works\david0	1
17	/2/1/	0x6AC0	17	adventure-works\kevin0	16
18	/2/2/	0x6B40	18	adventure-works\john5	16
19	/2/3/	0x6BC0	19	adventure-works\mary2	16

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 7: Optimierung der Tabelle durch Erzeugen mehrerer Indizes.
- Erzeugen eines Index für den Level in der Hierarchie

```
ALTER TABLE HumanResources.NewOrg
   ADD H_Level AS OrgNode.GetLevel() ;
CREATE UNIQUE INDEX EmpBFInd
   ON HumanResources.NewOrg(H_Level, OrgNode) ;
GO
```

- Erzeugen eines Index für den Arbeitnehmer (EmployeeID)

```
CREATE UNIQUE INDEX EmpIDs_unq ON
HumanResources.NewOrg(EmployeeID) ;
GO
```

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 1: Converting a Table to a Hierarchical Structure
- Schritt 8: Abfrage der Tabelle NewOrg

```
SELECT OrgNode.ToString() AS LogicalNode,
OrgNode, H_Level, EmployeeID, LoginID
FROM HumanResources.NewOrg
ORDER BY OrgNode;

SELECT OrgNode.ToString() AS LogicalNode,
OrgNode, H_Level, EmployeeID, LoginID
FROM HumanResources.NewOrg
ORDER BY H_Level, OrgNode;

SELECT OrgNode.ToString() AS LogicalNode,
OrgNode, H_Level, EmployeeID, LoginID
FROM HumanResources.NewOrg
ORDER BY EmployeeID;
GO
```

132 %

Ergebnisse | Meldungen

	LogicalNode	OrgNode	H_Level	EmployeeID	LoginID
1	/	0x	0	1	adventure-works\ken0
2	/1/	0x58	1	2	adventure-works\terri0
3	/1/1/	0x5AC0	2	3	adventure-works\roberto0
4	/1/1/1/	0x5AD6	3	4	adventure-works\rob0
5	/1/1/2/	0x5ADA	3	5	adventure-works\gail0
6	/1/1/3/	0x5ADE	3	6	adventure-works\jossef0
7	/1/1/4/	0x5AE1	3		
8	/1/1/4/1/	0x5AE158	4		

Sortiert nach OrgNode

	LogicalNode	OrgNode	H_Level	EmployeeID	LoginID
1	/	0x	0	1	adventure-works\ken0
2	/1/	0x58	1	2	adventure-works\terri0
3	/2/	0x68	1	16	adventure-works\david0
4	/3/	0x78	1	25	adventure-works\james1
5	/4/	0x84	1	234	adventure-works\laura1
6	/5/	0x8C	1	263	adventure-works\jean0
7	/6/				
8	/1/1/				

Sortiert nach Level und OrgNode

	LogicalNode	OrgNode	H_Level	EmployeeID	LoginID
1	/	0x	0	1	adventure-works\ken0
2	/1/	0x58	1	2	adventure-works\terri0
3	/1/1/	0x5AC0	2	3	adventure-works\roberto0
4	/1/1/1/	0x5AD6	3	4	adventure-works\rob0
5	/1/1/2/	0x5ADA	3	5	adventure-works\gail0
6	/1/1/3/	0x5ADE	3	6	adventure-works\jossef0
7	/1/1/4/	0x5AE1	3	7	adventure-works\dylan0
8	/1/1/4/1/	[illegible]	4	8	adventure-works\diane1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	...e-works\gigi0
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	...e-works\michael6

Sortiert nach EmployeeID

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-converting-a-table-to-a-hierarchical-structure?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 2: Erzeugen und Verwalten einer Tabelle mit dem **hierarchyid-Datentyp**
- Schritt 1: Erzeugen einer neuen Datenbank `HKA_HierarchyID`
- Schritt 2: Erzeugen einer Tabelle `EmployeeOrg` mit der Spalte `OrgNode` vom Typ `hierarchyid`

```
USE HKA_HierarchyID ;
GO
if OBJECT_ID('EmployeeOrg') is not null
 drop table EmployeeOrg
CREATE TABLE EmployeeOrg
(
   OrgNode hierarchyid PRIMARY KEY CLUSTERED,
   OrgLevel AS OrgNode.GetLevel(),
   EmployeeID int UNIQUE NOT NULL,
   EmpName varchar(20) NOT NULL,
   Title varchar(20) NULL  ) ;
GO
```

- Erzeugen eines Index für OrgLevel und OrgNode

```
CREATE UNIQUE INDEX EmployeeOrgNc1
ON EmployeeOrg(OrgLevel, OrgNode) ;
GO
```

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-2-creating-and-managing-data-in-a-hierarchical-table?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 2: Erzeugen und Verwalten einer Tabelle mit dem **hierarchyid-Datentyp**
- Schritt 2: Füllen der Tabelle `EmployeeOrg`
- Einfügen der obersten Hierarchie (root)

```
INSERT EmployeeOrg (OrgNode, EmployeeID, EmpName,
Title)
VALUES (hierarchyid::GetRoot(), 6, 'David',
'Marketing Manager') ;
GO
```

- Abfrage der Tabelle EmployeeOrg

```
SELECT OrgNode.ToString() AS Text_OrgNode,
OrgNode, OrgLevel, EmployeeID, EmpName, Title
FROM EmployeeOrg ;
```

Ergebnisse | Meldungen

	Text_OrgNode	OrgNode	OrgLevel	EmployeeID	EmpName	Title
1	/	0x	0	6	David	Marketing Manager

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-2-creating-and-managing-data-in-a-hierarchical-table?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 2: Erzeugen und Verwalten einer Tabelle mit dem **hierarchyid-Datentyp**
- Schritt 3: Einfügen eines Mitarbeiters
- Einfügen der obersten Hierarchie (root)

```
DECLARE @Manager hierarchyid
SELECT @Manager = hierarchyid::GetRoot()
FROM EmployeeOrg ;

INSERT EmployeeOrg (OrgNode, EmployeeID, EmpName,
Title)
VALUES
(@Manager.GetDescendant(NULL, NULL), 46, 'Sariya',
'Marketing Specialist') ;
```

- Abfrage der Tabelle EmployeeOrg

```
SELECT OrgNode.ToString() AS Text_OrgNode,
OrgNode, OrgLevel, EmployeeID, EmpName, Title
FROM EmployeeOrg ;
```

Ergebnisse | Meldungen

	Text_OrgNode	OrgNode	OrgLevel	EmployeeID	EmpName	Title
1	/	0x	0	6	David	Marketing Manager
2	/1/	0x58	1	46	Sariya	Marketing Specialist

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-2-creating-and-managing-data-in-a-hierarchical-table?view=sql-server-ver15

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 2: Erzeugen und Verwalten einer Tabelle mit dem **hierarchyid-Datentyp**
- Schritt 4: Erzeugen einer Stored Procedure (Funktion) zum Einfügen neuer Mitarbeiter

```
CREATE PROC AddEmp(@mgrid int, @empid int, @e_name
varchar(20), @title varchar(20))
AS
BEGIN
   DECLARE @mOrgNode hierarchyid, @lc hierarchyid
   SELECT @mOrgNode = OrgNode
   FROM EmployeeOrg
   WHERE EmployeeID = @mgrid
   SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
   BEGIN TRANSACTION
      SELECT @lc = max(OrgNode)
      FROM EmployeeOrg
      WHERE OrgNode.GetAncestor(1) =@mOrgNode ;

      INSERT EmployeeOrg (OrgNode, EmployeeID,
EmpName, Title)
      VALUES(@mOrgNode.GetDescendant(@lc, NULL),
@empid, @e_name, @title)
   COMMIT
END ;
GO
```

- Einfügen neuer Mitarbeiter

```
EXEC AddEmp 6, 271, 'John', 'Marketing Specialist';
EXEC AddEmp 6, 119, 'Jill', 'Marketing Specialist';
EXEC AddEmp 46, 269, 'Wanida', 'Marketing Assistant';
EXEC AddEmp 271, 272, 'Mary', 'Marketing Assistant';
```

- Abfrage der Tabelle EmployeeOrg

```
SELECT OrgNode.ToString() AS Text_OrgNode,
OrgNode, OrgLevel, EmployeeID, EmpName, Title
FROM EmployeeOrg ;
```

	Text_OrgNode	OrgNode	OrgLevel	EmployeeID	EmpName	Title
1	/	0x	0	6	David	Marketing Manager
2	/1/	0x58	1	46	Sariya	Marketing Specialist
3	/1/1/	0x5AC0	2	269	Wanida	Marketing Assistant
4	/2/	0x68	1	271	John	Marketing Specialist
5	/2/1/	0x6AC0	2	272	Mary	Marketing Assistant
6	/3/	0x78	1	119	Jill	Marketing Specialist

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-2-creating-and-managing-da

Data Warehouse
Hierarchie

Nutzung des hierarchyid-Datentyps

- Beispiel 2: Erzeugen und Verwalten einer Tabelle mit dem **hierarchyid-Datentyp**
- Schritt 5: Abfragen der Tabelle
- Mitarbeiter von Sariya (inklusive Sariya)

```sql
DECLARE @CurrentEmployee hierarchyid
SELECT @CurrentEmployee = OrgNode
FROM EmployeeOrg
WHERE EmployeeID = 46 ;

SELECT *
FROM EmployeeOrg
WHERE OrgNode.IsDescendantOf(@CurrentEmployee )=1;
```

	OrgNode	OrgLevel	EmployeeID	EmpName	Title
1	0x58	1	46	Sariya	Marketing Specialist
2	0x5AC0	2	269	Wanida	Marketing Assistant

- Mitarbeiter von Sariya (exklusive Sariya)

```sql
DECLARE @CurrentEmployee hierarchyid
SELECT @CurrentEmployee = OrgNode
FROM EmployeeOrg
WHERE EmployeeID = 46 ;

SELECT OrgNode.ToString() AS Text_OrgNode, *
FROM EmployeeOrg
WHERE OrgNode.GetAncestor(1) = @CurrentEmployee
```

	Text_OrgNode	OrgNode	OrgLevel	EmployeeID	EmpName	Title
1	/1/1/	0x5AC0	2	269	Wanida	Marketing Assistant

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-2-creating-and-managing-data-in-a-hierarchical-table?view=sql-server-ver15

Data Warehouse Queries

Beispiele für Abfragen Data Warehouse WorldWideImporters

- Gesamter Gewinn in Millionen USD

```
SELECT SUM(Profit)/ 1000000 as [TotalProfit Mio. USD]
FROM [WideWorldImportersDW].[Fact].[Sale]
```

- Gesamter Gewinn pro Stadt geordnet nach dem Gewinn

```
SELECT C.City, SUM(Profit) as TotalProfitPerCity
FROM [WideWorldImportersDW].[Fact].[Sale] S,
Dimension.City C
WHERE C.[City Key] = S.[City Key]
GROUP BY C.City
ORDER BY TotalProfitPerCity DESC
```

- Minimale und maximale Steuer (in %) nach Stadt

```
SELECT C.City, MIN([Tax Rate]) as [Min Tax Rate],
MAX([Tax Rate]) as [Max Tax Rate]
FROM [WideWorldImportersDW].[Fact].[Sale] S,
Dimension.City C
WHERE C.[City Key] = S.[City Key]
GROUP BY C.City
ORDER BY [Min Tax Rate], [Max Tax Rate]
```

- Durchschnittlicher Gewinn pro Rechnungsdatum

```
SELECT DISTINCT S.[Invoice Date Key], AVG(Profit)
as AvgProfit
FROM [WideWorldImportersDW].[Fact].[Sale] S
GROUP BY S.[Invoice Date Key]
ORDER BY S.[Invoice Date Key]
```

- alternativ

```
SELECT DISTINCT S.[Invoice Date Key], AVG(Profit)
OVER (PARTITION BY S.[Invoice Date Key]) as
AvgProfit
FROM [WideWorldImportersDW].[Fact].[Sale] S
ORDER BY S.[Invoice Date Key]
```

Data Warehouse Queries

Beispiele für Abfragen Data Warehouse WorldWideImporters

+ Erzeugen des Jahres / Monats aus einem Datum on-the-fly unter Nutzung einer common table expression und anschließender Nutzung für PARTITION mit year-to-date-Berechnung

```
USE [WideWorldImportersDW]
GO
WITH Profit_CTE ([Invoice Date Key],[YEAR], [MONTH], Profit, [City
Key]) AS
(SELECT [Invoice Date Key], YEAR([Invoice Date Key]) AS [YEAR],
MONTH([Invoice Date Key]) AS [MONTH]
,Profit,[City Key]
FROM [Fact].[Sale])
SELECT [Invoice Date Key],[City Key],[YEAR], [MONTH],
Profit,SUM(Profit)
OVER (PARTITION BY [City Key],[YEAR] ORDER BY [Invoice Date
Key],[MONTH], Profit ROWS UNBOUNDED PRECEDING) AS [SumProfit YTD]
FROM Profit_CTE
ORDER BY [City Key],[Invoice Date Key]
```

	Invoice Date Key	City Key	YEAR	MONTH	Profit	SumProfit YTD
1	2013-01-01	37955	2013	1	20.00	20.00
2	2013-01-02	37955	2013	1	90.00	110.00
3	2013-01-18	37955	2013	1	8.50	118.50
4	2013-01-18	37955	2013	1	132.00	250.50
5	2013-01-18	37955	2013	1	150.00	400.50
6	2013-01-18	37955	2013	1	360.00	760.50
7	2013-01-18	37955	2013	1	1080.00	1840.50
8	2013-01-31	37955	2013	1	17.00	1857.50
9	2013-01-31	37955	2013	1	684.00	2541.50
10	2013-02-18	37955	2013	2	68.00	2609.50
11	2013-02-18	37955	2013	2	68.00	2677.50
12	2013-02-18	37955	2013	2	200.00	2877.50
13	2013-02-18	37955	2013	2	218.40	3095.90
14	2013-02-18	37955	2013	2	1320.00	4415.90
15	2013-02-21	37955	2013	2	87.50	4503.40
16	2013-02-21	37955	2013	2	96.00	4599.40
118	2013-12-09	37955	2013	12	211.50	41844.15
119	2013-12-09	37955	2013	12	240.00	42084.15
120	2013-12-09	37955	2013	12	240.00	42324.15
121	2013-12-27	37955	2013	12	52.50	42376.65
122	2013-12-27	37955	2013	12	84.00	42460.65
123	2013-12-27	37955	2013	12	216.00	42676.65
124	2014-01-10	37955	2014	1	9.90	9.90
125	2014-01-10	37955	2014	1	37.50	47.40
126	2014-01-10	37955	2014	1	76.50	123.90
127	2014-01-10	37955	2014	1	2350.00	2473.90
128	2014-01-14	37955	2014	1	29.50	2503.40

Data Warehouse Queries

Beispiele für Abfragen Data Warehouse WorldWideImporters

- Erzeugen des Jahres / Monats aus einem Datum on-the-fly unter Nutzung einer common table expression und anschließender Nutzung für PARTITION mit year-to-date-Berechnung
- Ausgabe reduziert auf die year-to-date-Summe für den jeweiligen Monat

```
USE [WideWorldImportersDW]

GO

WITH Profit_CTE ([Invoice Date Key],[YEAR], [MONTH],
Profit, [City Key]) AS
(SELECT [Invoice Date Key], YEAR([Invoice Date Key]) AS
[YEAR],
MONTH([Invoice Date Key]) AS [MONTH]
,Profit,[City Key]
FROM [Fact].[Sale])

SELECT [Invoice Date Key],[City Key],[YEAR], [MONTH],
Profit,SUM(Profit)
OVER (PARTITION BY [City Key],[YEAR] ORDER BY [Invoice
Date Key],[MONTH], Profit ROWS UNBOUNDED PRECEDING) AS
[SumProfit YTD]
INTO #SumProfitYTD
FROM Profit_CTE
ORDER BY [City Key],[Invoice Date Key]

SELECT *, RANK() OVER (PARTITION BY [City Key]
,[YEAR],[MONTH] ORDER BY [SumProfit YTD] DESC) AS [Rank]
INTO #SumProfitYTDWithRank
FROM #SumProfitYTD
ORDER BY [City Key],[Invoice Date Key],[SumProfit YTD]

SELECT C.City, [YEAR], [MONTH], [SumProfit YTD]
FROM #SumProfitYTDWithRank YTD, Dimension.City C
WHERE [Rank]=1 AND YTD.[City Key] = C.[City Key]
ORDER BY YTD.[City Key],[Invoice Date Key],[SumProfit YTD]

DROP TABLE #SumProfitYTD

DROP TABLE #SumProfitYTDWithRank
```

Data Warehouse Queries

Beispiele für Abfragen Data Warehouse WorldWideImporters

- Erzeugen des Jahres / Monats aus einem Datum on-the-fly unter Nutzung einer common table expression und anschließender Nutzung für PARTITION mit year-to-date-Berechnung
- Ausgabe reduziert auf die year-to-date-Summe für den jeweiligen Monat

	City	YEAR	MONTH	SumProfit YTD
1	Rose Tree	2013	1	2541.50
2	Rose Tree	2013	2	7609.60
3	Rose Tree	2013	3	10698.90
4	Rose Tree	2013	4	14351.40
5	Rose Tree	2013	5	16505.90
6	Rose Tree	2013	6	19597.50
7	Rose Tree	2013	7	21897.50
8	Rose Tree	2013	8	26609.75
9	Rose Tree	2013	9	33670.15
10	Rose Tree	2013	10	37573.15
11	Rose Tree	2013	11	40508.65
12	Rose Tree	2013	12	42676.65
13	Rose Tree	2014	1	4929.40
14	Rose Tree	2014	2	5745.40
15	Rose Tree	2014	3	7905.20
16	Rose Tree	2014	5	10524.20
17	Rose Tree	2014	6	17808.45
18	Rothsville	2013	1	631.10
19	Rothsville	2013	2	5968.10
20	Rothsville	2013	3	6553.70
21	Rothsville	2013	4	9493.70
22	Rothsville	2013	5	15861.70

ETL
PolyBase

Extract, Transform & Load mit PolyBase

- PolyBase enables your SQL Server instance to query data with T-SQL directly from SQL Server, Oracle, Teradata, MongoDB, Hadoop clusters, Cosmos DB without separately installing client connection software. You can also use the generic ODBC connector to connect to additional providers using third-party ODBC drivers. PolyBase allows T-SQL queries to join the data from external sources to relational tables in an instance of SQL Server.
- A key use case for data virtualization with the PolyBase feature is to allow the data to stay in its original location and format. You can virtualize the external data through the SQL Server instance, so that it can be queried in place like any other table in SQL Server. This process minimizes the need for ETL processes for data movement. This data virtualization scenario is possible with the use of PolyBase connectors.

Quelle: https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-guide?view=sql-server-ver15

Hochschule Karlsruhe
University of
Applied Sciences

www.h-ka.de