Matija Lah
Mi Lambda, Matija Lah, s.p.

Implementing Supertypes and Subtypes in SQL Server

Speaker Introduction

Matija Lah

- Lawyer
- SolidQ CEE, Mentor
- Microsoft MVP (Data Platform), 2007 – 2017/2018
- 18+ years of SQL Server experience
  - Specializing in Legal Information Management and Natural Language Processing
- Current focus areas
  - Database Engine
  - Integration Services
  - Analysis Services (incl. Data Mining)
- Based in Ljubljana, Slovenia
- mlah@solidq.com, @MatijaLah

#ntk19

Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

Business Case

Different entities

Some *common* attributes, some *specific* attributes

Same relationships

E.g. in *sales*, a *customer* is a *person* or a *company*

Type

Person: *natural* person, or *legal* person

Supertype

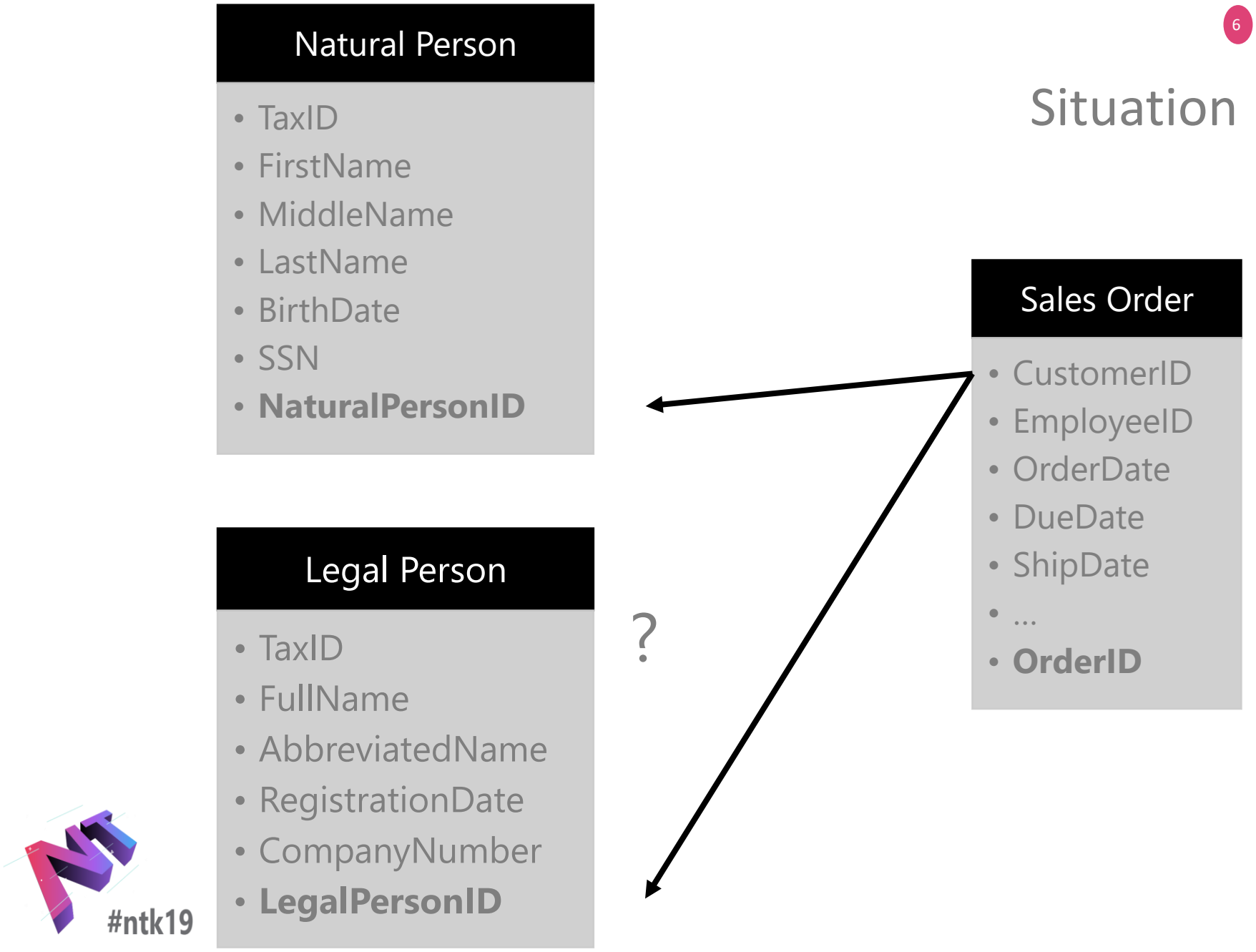
Common person attributes
(Tax ID, Friendly name, etc.)

Subtype

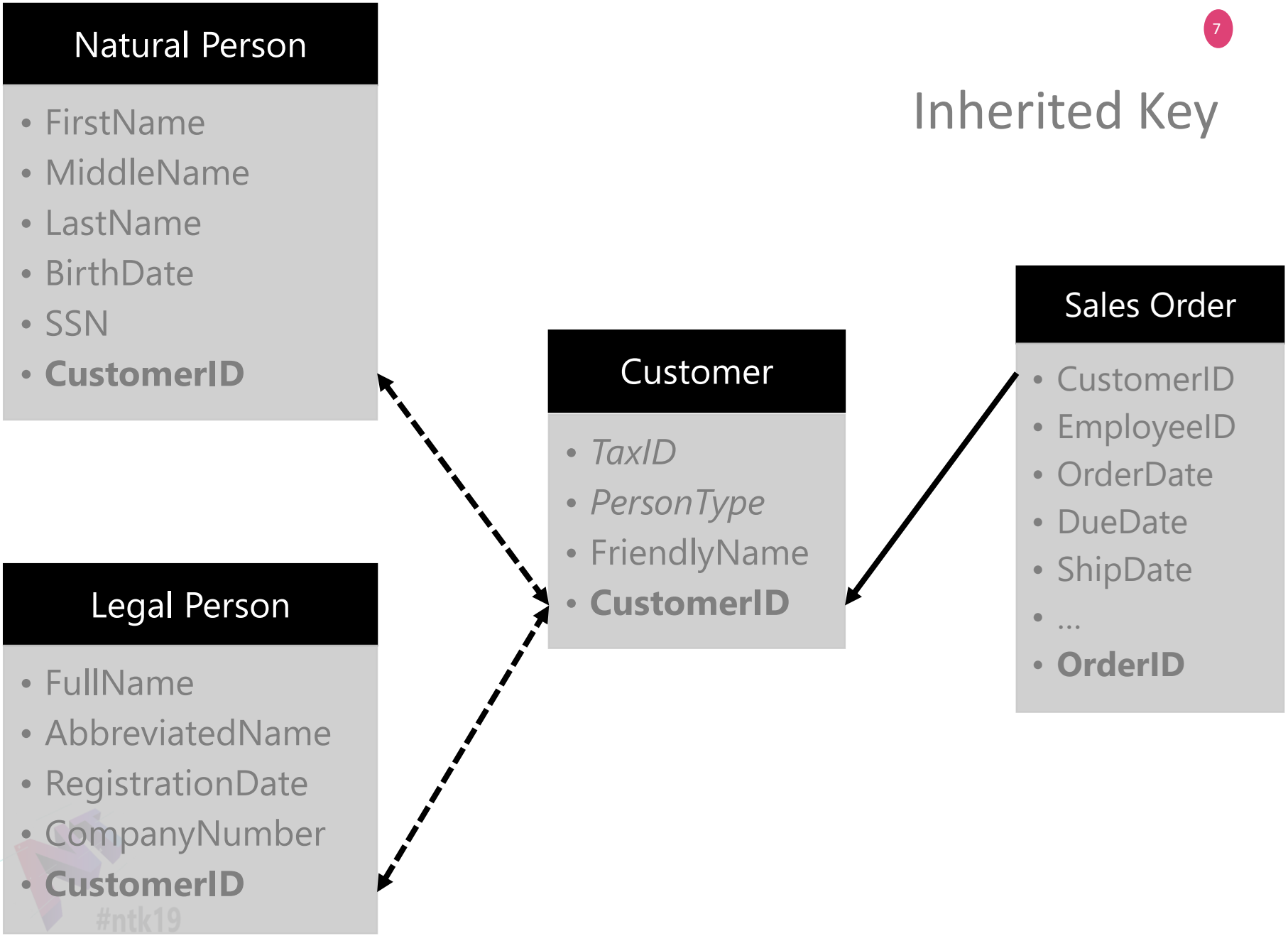
Specific natural person attributes
(First name, Last name, Middle name, Date of birth, Social security number, etc.)

Specific legal person attributes
(Full name, Abbreviated name, Registry number, Date of registration, etc.)

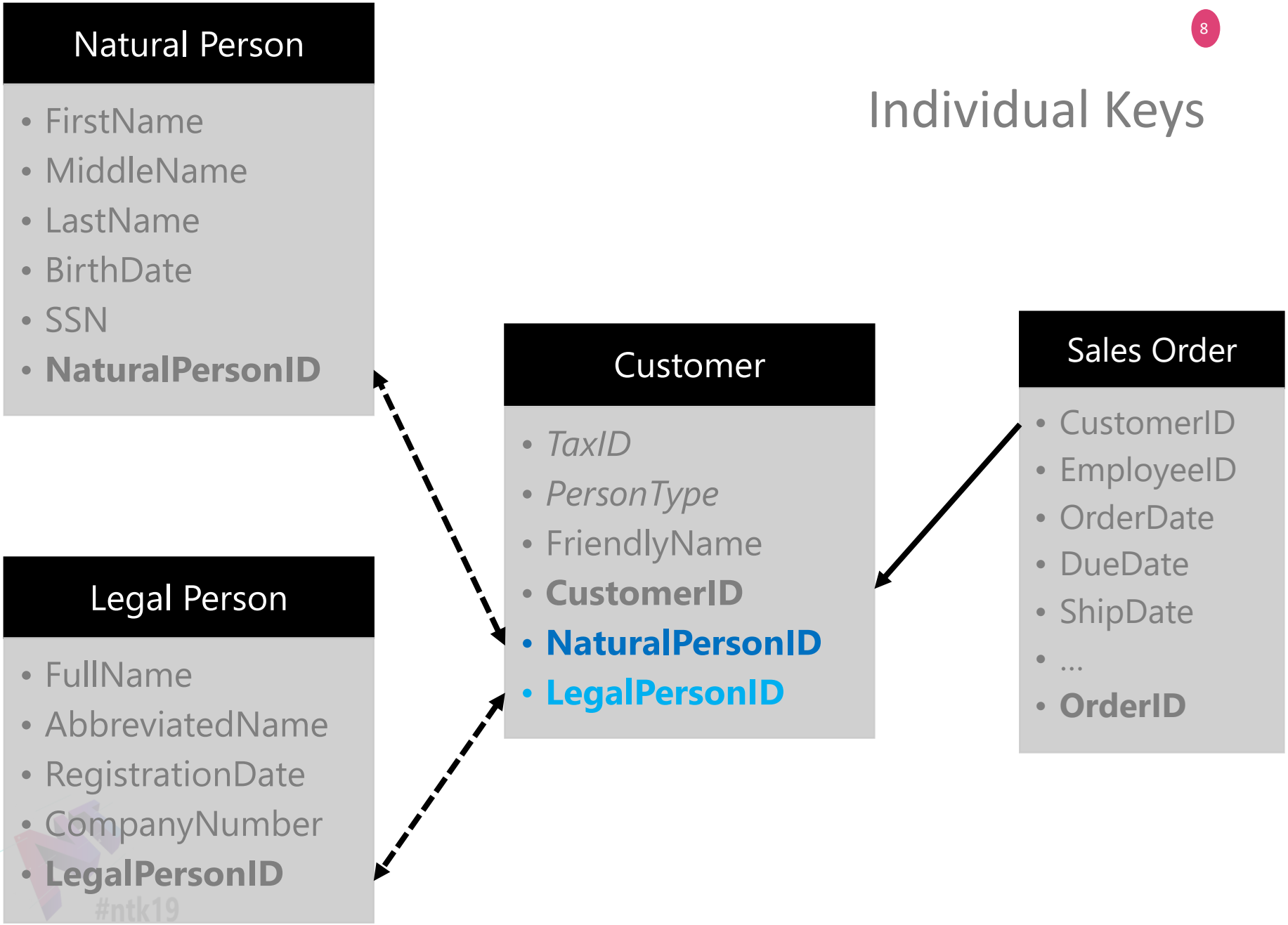
Situation


Natural Person
• TaxID
• FirstName
• MiddleName
• LastName
• BirthDate
• SSN
• NaturalPersonID
Legal Person
• TaxID
• FullName
• AbbreviatedName
• RegistrationDate
• CompanyNumber
• LegalPersonID
?
Sales Order
• CustomerID
• EmployeeID
• OrderDate
• DueDate
• ShipDate
• …
• OrderID

Inherited Key



Individual Keys



Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

Sparse Columns

- Both types in one table
  - Supertype attributes in „regular“ columns
  - Subtype attributes in *sparse* columns
- Differentiation with column and table constraints

Demo 1

Sparse Columns

Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

XML Document

- Both types in one table
  - Supertype attributes in columns
  - Subtype attributes as XML documents in an XML column
- XML
  - Complex SQL Server native data type
    (XML Schema, XPath, XQuery)
  - Built-in retrieval/manipulation methods
    (`.nodes()`, `.exist()`, `.query()`, `.value()`, `.modify()`)
  - T-SQL binding functions
    (`sql:column()`, `sql:variable()`)
  - T-SQL constructs to generate XML data
    (`FOR XML`)
- Differentiation with separate XML namespaces
  (and XML Schemas, of course)

Demo 2

XML Documents

Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

JSON

- Both types in one table
  - Supertype attributes in columns
  - Subtype attributes in JSON – in a NVARCHAR/VARCHAR column
- JSON
  - Not a SQL Server native data type
  - Built-in functionalities to access JSON entities in textual data (JSON Path)
  - Built-in retrieval/manipulation functions (`ISJSON()`, `OPENJSON()`, `JSON_QUERY()`, `JSON_VALUE()`, `JSON_MODIFY()`)
  - T-SQL constructs to generate JSON data (`FOR JSON`)
- Differentiation depends on the specific implementation

Demo 3

JSON

Agenda

Types, supertypes, and subtypes

Sparse Columns

XML Documents

JSON

CLR User-defined Types

CLR User-defined Types (UDT)

- Both types in one table
  - Supertype attributes in columns
  - Subtype attributes in a CLR UDT column
- CLR UDT
  - Complex SQL Server data type
    (implemented as a DOT.NET CLR assembly)
  - Not native, but as close as
- Differentiation within CLR UDT
  - All dynamic properties outside Database Engine, but accessible in T-SQL
    (data access fully integrated)

Demo 4

CLR User-defined Type

Comparison

	Multiple Tables	Sparse Columns	XML	JSON	CLR UDT
Completeness	*	Constraints	XML Schema	✗	CLR
Differentiation	Separate Tables	Constraints	XML Namespaces	**	CLR
Manipulation	✓	✓	XML DML	JSON DML	CLR
Retrieval	✓	✓	XPath/XQuery	JSON Path	CLR
Indexing	✓	✓	✓	Computed columns	Computed columns
Full-text Search	✓	✓	✓	✓***	✗

* „Tricky“
** Depends on JSON structure
*** No native JSON IFilter

Recommendations

- OLTP
  - Any of the methods, but maintain completeness!
- DW
  - Sparse columns
    - Client applications (e.g. OLAP, Excel, Power*) expect columns
  - Data marts
    - Type-independent
    - Person-centric
    - Company-centric
  - Assumes completeness is maintained in OLTP!

Further Reading

Use Sparse Columns

(https://docs.microsoft.com/en-us/sql/relational-databases/tables/use-sparse-columns)

XML Data (SQL Server)

(https://docs.microsoft.com/en-us/sql/relational-databases/xml/xml-data-sql-server)

JSON data in SQL Server

(https://docs.microsoft.com/en-us/sql/relational-databases/json/json-data-sql-server)

CLR User-Defined Types

(https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/clr-user-defined-types)

Homework

Sparse Columns

Create additional filtered indexes

XML Documents

Leverage computed columns vs. selective XML indexes to optimize queries

JSON

Rewrite all queries using `OPENJSON()`

CLR UDT

Implement additional deterministic data access methods