Built-in JSON Support

Leonard Lobel
CTO, SLEEK TECHNOLOGIES

@lennilobel

JSON and SQL Server

Capabilities

- Format and export JSON from relational queries
- Store and query JSON inside the database

Conceptually similar to XML support

- Simpler model
- No native “json” data type; uses nvarchar(max)

Why no native type?

- Easier migration to leave json columns as ordinary string types
- Cross-feature compatibility (e.g., Hekaton, temporal)

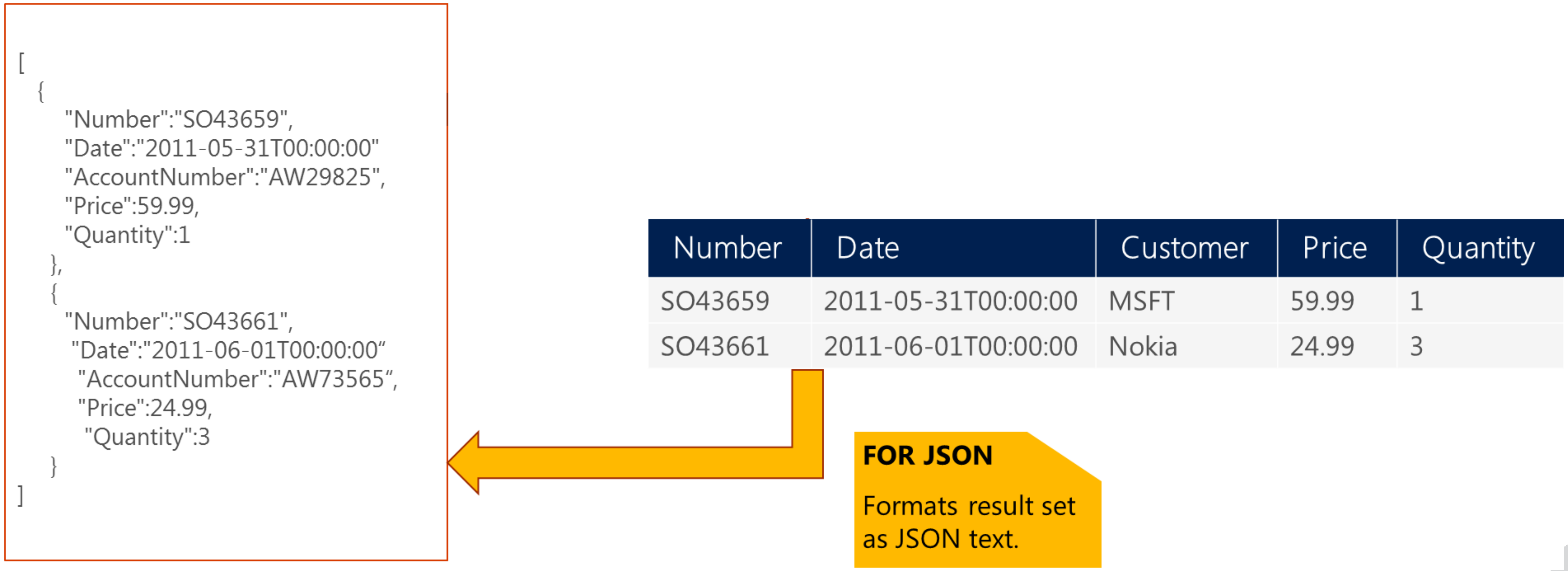
No custom JSON indexes

- Optimize JSON queries using standard indexes
- Create computed columns over desired properties, and then index the computed columns

Bidirectional JSON Transformation

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

Bidirectional JSON Transformation



```
[
  {
     "Number":"SO43659",
     "Date":"2011-05-31T00:00:00"
     "AccountNumber":"AW29825",
     "Price":59.99,
     "Quantity":1
   },
   {
     "Number":"SO43661",
      "Date":"2011-06-01T00:00:00“
       "AccountNumber":"AW73565“,
       "Price":24.99,
        "Quantity":3
   }
]
```

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

FOR JSON

Formats result set as JSON text.

Bidirectional JSON Transformation

Built-in functions
ISJSON
JSON_VALUE
JSON_QUERY

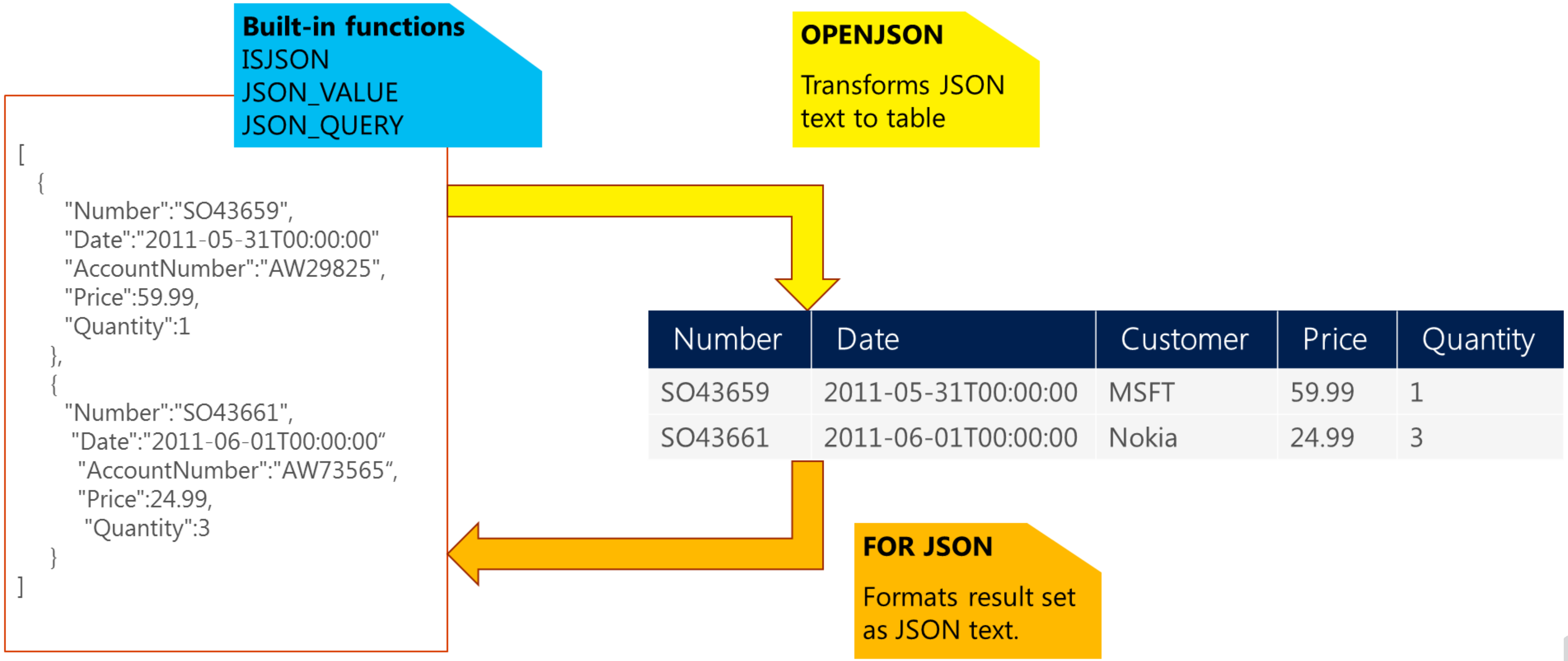
```
[
  {
     "Number":"SO43659",
     "Date":"2011-05-31T00:00:00"
     "AccountNumber":"AW29825",
     "Price":59.99,
     "Quantity":1
   },
   {
     "Number":"SO43661",
      "Date":"2011-06-01T00:00:00“
       "AccountNumber":"AW73565“,
       "Price":24.99,
        "Quantity":3
   }
]
```

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

FOR JSON

Formats result set as JSON text.

Bidirectional JSON Transformation



Built-in functions
ISJSON
JSON_VALUE
JSON_QUERY

```
[
  {
     "Number":"SO43659",
     "Date":"2011-05-31T00:00:00"
     "AccountNumber":"AW29825",
     "Price":59.99,
     "Quantity":1
   },
   {
     "Number":"SO43661",
      "Date":"2011-06-01T00:00:00“
       "AccountNumber":"AW73565“,
       "Price":24.99,
        "Quantity":3
   }
]
```

OPENJSON

Transforms JSON text to table

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

FOR JSON

Formats result set as JSON text.

FOR JSON Clause

Append to SELECT statements to generate results in JSON format

- Example:

```
SELECT * FROM Customer FOR JSON AUTO
```

FOR JSON AUTO

- Creates nested structure based on table hierarchy

FOR JSON PATH

- Creates nested structure based on column aliases

FOR JSON Formatting Options

WITHOUT_ARRAY_WRAPPER

- Don’t generate [] syntax (single JSON object)

ROOT

- Generate single root wrapper object around the results

INCLUDE_NULL_VALUES

- Generate properties for NULL columns

Demo

Querying with FOR JSON AUTO

Demo

Querying with FOR JSON PATH

Built-in JSON Functions

ISJSON

- Validates for well-formed JSON
- Use in check constraints for NVARCHAR columns containing JSON

JSON_QUERY

- Queries by path expression and returns a nested object/array
- Similar to *xml.query*

JSON_VALUE

- Queries by path expression and returns a scalar value
- Similar to *xml.value*

No JSON "DML"

- Cannot directly modify JSON content
- No equivalent to *xml.modify*

JSON Path Expressions

Reference JSON properties using a JavaScript-like syntax

Syntax	Description
$	References the entire JSON object

JSON Path Expressions

Reference JSON properties using a JavaScript-like syntax

Syntax	Description
`$`	References the entire JSON object
`$.prop1`	References a top-level property in the JSON object

JSON Path Expressions

Reference JSON properties using a JavaScript-like syntax

Syntax	Description
`$`	References the entire JSON object
`$.prop1`	References a top-level property in the JSON object
`$.prop1[5]`	References the sixth element in the array contained inside a top-level property in the JSON object

JSON Path Expressions

Reference JSON properties using a JavaScript-like syntax

Syntax	Description
`$`	References the entire JSON object
`$.prop1`	References a top-level property in the JSON object
`$.prop1[5]`	References the sixth element in the array contained inside a top-level property in the JSON object
`$.prop1.prop2 .prop3[5].prop4 .prop5[15].prop6`	References a property nested deeply within the JSON object

JSON Query Example

```
SELECT
 Id,
 OrderNumber,
 OrderDate,
 JSON_VALUE(OrderDetails, '$.Order.ShipDate')
FROM
 SalesOrderRecord
WHERE
 ISJSON(OrderDetails) = 1 AND
 JSON_VALUE(OrderDetails, '$.Order.Type') = 'C'
```

JSON Query Example

```
SELECT
 Id,
 OrderNumber,
 OrderDate,
 JSON_VALUE(OrderDetails, '$.Order.ShipDate')
FROM
 SalesOrderRecord
WHERE
 ISJSON(OrderDetails) = 1 AND
 JSON_VALUE(OrderDetails, '$.Order.Type') = 'C'
```

JSON Query Example

```
SELECT
 Id,
 OrderNumber,
 OrderDate,
 JSON_VALUE(OrderDetails, '$.Order.ShipDate')
FROM
 SalesOrderRecord
WHERE
 ISJSON(OrderDetails) = 1 AND
 JSON_VALUE(OrderDetails, '$.Order.Type') = 'C'
```

JSON Query Example

```
SELECT
 Id,
 OrderNumber,
 OrderDate,
 JSON_VALUE(OrderDetails, '$.Order.ShipDate')
FROM
 SalesOrderRecord
WHERE
 ISJSON(OrderDetails) = 1 AND
 JSON_VALUE(OrderDetails, '$.Order.Type') = 'C'
```

Demo

Storing JSON

Demo

Querying JSON

Shredding JSON

OPENJSON table-valued function (TVF)

- Provides a rowset view over a JSON document
- Shreds single JSON document into multiple rows

What does it do?

- Iterates through objects (if JSON array) or properties (if JSON object)
- Generates a row for each object/property with key, value, and type

Discoverable schema

- Key, value, and type columns

Explicit schema

- Include columns, data types, and property-to-column mapping rules

Demo

Getting started with OPENJSON

Demo

Parsing parent/child objects with OPENJSON

Demo

Using OPENJSON with an explicit schema

Summary

FOR JSON AUTO

FOR JSON PATH

ISJSON

JSON_VALUE

JSON_QUERY

OPENJSON