SQL Server 2016

New Feature Preview
2016-10-01
@RileyMajor

PASS

PASS SQL saturday
#557 | MINNESOTA 2016

SQL Saturday #557

passmn
SQL Server User Group

- Thank you Sponsors!
  - Please visit the sponsors and enter their end-of-day raffles.
- Event After Party
  - Sky Deck Sports Grille and Lanes at the Mall of America at 7 PM.
- Want More Free Training?
  - PASSMN meets the 3rd Tuesday of every month. https://mnssug.org/

Lunch Sponsor - Dell EMC

For those who paid for lunch already, we will refund you via PayPal. If you wish to donate to Rebecca CoderDojo, please drop your ticket in the bucket at registration.

You Rock Sponsor - Pyramid Analytics

- Gold Sponsors
  - IDERA
  - Pragmatic Works
  - VMWare
  - GNet
  - Tail Wind
  - Microsoft
  - Dell Software

Other Sponsors

- Silver Sponsors
  - Improving
  - Experts Exchange
  - Pure Storage
- Bronze Sponsors
  - SQL Sentry
  - COZYROC
- PASS
- Blog Sponsors
  - SQLVariant

SQL Saturday #557

- Twitter
  - @PASSMN
  - #SQLSatMN

Presentation Overview

- Brief History of SQL Server
- New Features in 2016
  - List
  - Demos
- Getting Started with 2016
- Future Thoughts
- Bio

Caveats: I talk mostly about the database engine (as opposed to business intelligence), and from the perspective of a developer at a smaller shop.

Misc links:
http://sqlturbo.com/sql-server-2016-new-features/
http://www.databasejournal.com/features/mssql/slideshows/10-new-features-worth-exploring-in-sql-server-2016.html

SQL Server is Born

- 1989 - Version 1 (Sybase / Ashton Tate)
- 1993 - Version 4.2 (Microsoft)
- 1995 - Version 6
- 1996 - Version 6.5
- 1998 - Version 7

My first databases were built with dBase IV. I made a database of states and people in my class. I even had a spot for pictures.

I started working with SQL Server version 7 with the Enterprise Manager and Query Analyzer

http://blogs.msdn.com/b/euanga/archive/2006/01/19/sql-mythbusters-sql-server-is-really-a-sybase-product-not-a-microsoft-one.aspx

https://en.wikipedia.org/wiki/Microsoft_SQL_Server
https://en.wikipedia.org/wiki/Ashton-Tate
https://en.wikipedia.org/wiki/Borland
https://en.wikipedia.org/wiki/Sybase

SQL Server Grows Up

- 2000 - SQL Server 2000
  - User-Defined Functions, Reporting & Analysis Services
- 2005 - SQL Server 2005
  - “completely rewritten”, SSMS, XML, TRY/CATCH, APPLY, CTEs, SSIS, CLR
- 2008 - SQL Server 2008
  - Date and Time, Extended Events, Geography, Compression, Filtered Indexes, Resource Governor
- 2010 - SQL Server 2008 R2
  - BI Tooling

SQL Server Recent History

- SQL Server 2012
  - TRY_PARSE, LAG/LEAD, Columnstore, Tabular Model
- SQL Server 2014
  - Backup to Azure, Updatable Columnstore, In-Memory OLTP (Hekaton)

The New and Shiny

- SQL Server 2016
  - Available Now
  - Cumulative Update 2
  - Free SSMS / Separate Download
  - Free Developer Version
  - New T-SQL
  - New Engine Features
  - New Admin Features

Dan English - What's new in SQL Server 2016 for Business Intelligence?
http://www.sqlsaturday.com/557/Sessions/Details.aspx?sid=49946

Brian Beswick - SSAS 2016 Tabular - Diving into the new features
http://www.sqlsaturday.com/557/Sessions/Details.aspx?sid=50000

SSMS 2016 Download
https://msdn.microsoft.com/en-us/library/mt238290.aspx

The SQL Server 2016 bits are "free", but you have to jump through some hoops to get them.
https://blogs.sqlsentry.com/aaronbertrand/get-developer-edition/

SQL Server 2016 CU 1
https://support.microsoft.com/en-us/kb/3164674

SQL Server 2016 CU 2
https://support.microsoft.com/en-us/kb/3182270

SQL Server 2016 Downloads
https://blogs.sqlsentry.com/team-posts/latest-builds-sql-server-2016/

SELECT All the Features

- T-SQL Improvements
  - TRUNCATE TABLE with PARTITION
  - DROP IF EXISTS
  - STRING_SPLIT()
  - DATEDIFF_BIG
  - AT TIME ZONE
  - COMPRESS / DECOMPRESS
  - JSON

T-SQL Enhancements
https://www.mssqltips.com/sqlservertip/4108/tsql-enhancements-in-sql-server-2016/

DATEDIFF_BIG
https://msdn.microsoft.com/en-us/library/mt628058.aspx

AT TIME ZONE
https://msdn.microsoft.com/en-us/library/mt612795.aspx

Generating Table of Numbers with JSON
https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/11/03/generate-serie-of-numbers-in-sql-server-2016-using-openjson/

It slices. It dices.

- SELECT STRING_SPLIT('a,b,c',',')
- Results:
  - a
  - b
  - c

Performance Surprises and Assumptions : STRING_SPLIT() by Aaron Betrand
https://sqlperformance.com/2016/03/sql-server-2016/string-split

Squish.

- COMPRESS / DECOMPRESS
  - T-SQL Functions
  - GZIP Compression
  - Applied to data (text, binary).
  - Results in smaller binary data.
  - Alternative to full table compression.
  - Standard Edition!

COMPRESS / DECOMPRESS in Standard Edition
https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/12/08/built-in-functions-for-compressiondecompression-in-sql-server-2016/

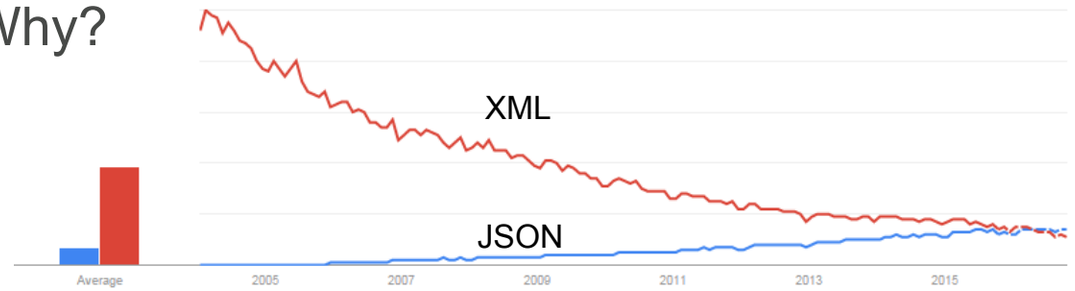
COMPRESS
https://msdn.microsoft.com/en-us/library/mt622775.aspx

DECOMPRESS
https://msdn.microsoft.com/en-us/library/mt622776.aspx

Whole Table Compression (2012 and onward) - Enterprise Only
https://msdn.microsoft.com/en-us/library/cc280449.aspx

JSON

- Why?



- What?

```
[
        {
                "MyField1": "MyValue",
                "MyField2": 123.45
        },
        {
                "MyField1": "MyValue2",
                "MyField2": 345.67
        }
]
```


Google Search Trend
https://www.google.com/trends/explore#q=json%2C%20xml&cmpt=q&tz=Etc%2FGMT%2B5

Look at the timing. SQL Server 2005's XML support came at the apex of XML popularity. SQL Server 2016's JSON support comes just as JSON surpasses XML.

JSON's data format is rooted in JavaScript syntax. It's an array of objects. Each object has properties.

References:

See also Jim Dorame's Talk: JSON for the DBA
http://www.sqlsaturday.com/557/Sessions/Details.aspx?sid=53668

JSON Data (SQL Server)
https://msdn.microsoft.com/en-us/library/dn921897.aspx

Format Query Results as JSON with FOR JSON (SQL Server)
https://msdn.microsoft.com/en-us/library/dn921882.aspx

SQL Server 2016 : JSON Support
By Aaron Bertrand on May 11, 2015 in T-SQL
http://blogs.sqlsentry.com/aaronbertrand/sql-server-2016-json-support/

How the New JSON Support Will Work in SQL Server 2016
https://visualstudiomagazine.com/blogs/data-driver/2015/05/sql-server-json-support.aspx

JSON vs XML – Sample Data

```
DECLARE @Orders TABLE
(
	OrderID bigint IDENTITY,
	OrderDate datetime
);
```

```
DECLARE @OrderDetails TABLE
(
	OrderDetailsID bigint IDENTITY,
	OrderID bigint,
	ProductID varchar(50),
	Qty int
);
```

OrderID	OrderDate	ProductID	Qty
1	2015-10-10	Bike	2
1	2015-10-10	Helmet	2
1	2015-10-10	Wheels	4
2	2015-10-09	Ball	10


```
DECLARE @Orders TABLE
(
			OrderID bigint IDENTITY,
			OrderDate datetime
);
DECLARE @OrderDetails TABLE
(
			OrderDetailsID bigint IDENTITY,
			OrderID bigint,
			ProductID varchar(50),
			Qty int
);
INSERT INTO @Orders
(
			OrderDate
)
VALUES
			('2015-10-10'),
			('2015-10-09');
INSERT INTO @OrderDetails
(
			OrderID,
			ProductID,
			Qty
)
VALUES
			(1,'Bike',2),
			(1,'Helmet',2),
			(1,'Wheels',4),
			(2,'Ball',10);

SELECT
			Orders.OrderID,
			Orders.OrderDate,
			OrderDetails.ProductID,
			OrderDetails.Qty
FROM			@Orders AS Orders
JOIN			@OrderDetails AS OrderDetails
ON				Orders.OrderID = OrderDetails.OrderID;
```

JSON vs XML – Production (Path)

JSON

```
SELECT
      Orders.OrderID,
      Orders.OrderDate,
      OrderDetails.ProductID,
      OrderDetails.Qty
FROM    @Orders AS
            Orders
JOIN    @OrderDetails AS
            OrderDetails
ON      Orders.OrderID =
            OrderDetails.OrderID
FOR     JSON
            PATH;
```

XML

```
SELECT
      Orders.OrderID,
      Orders.OrderDate,
      OrderDetails.ProductID,
      OrderDetails.Qty
FROM    @Orders AS
            Orders
JOIN    @OrderDetails AS
            OrderDetails
ON      Orders.OrderID =
            OrderDetails.OrderID
FOR     XML
            PATH;
```

JSON vs XML – Production (Path)

JSON

```
[
	{
		"OrderID":1,
		"OrderDate":"2015-10-10T00:00:00",
		"ProductID":"Bike",
		"Qty":2
	},
	{
		"OrderID":1,
		"OrderDate":"2015-10-10T00:00:00",
		"ProductID":"Helmet",
		"Qty":2
	},
	{
		"OrderID":1,
		"OrderDate":"2015-10-10T00:00:00",
		"ProductID":"Wheels",
		"Qty":4
	},
	{
		"OrderID":2,
		"OrderDate":"2015-10-09T00:00:00",
		"ProductID":"Ball",
		"Qty":10
	}
]
```

XML

```
<row>
	<OrderID>1</OrderID>
	<OrderDate>2015-10-10T00:00:00</OrderDate>
	<ProductID>Bike</ProductID>
	<Qty>2</Qty>
</row>
<row>
	<OrderID>1</OrderID>
	<OrderDate>2015-10-10T00:00:00</OrderDate>
	<ProductID>Helmet</ProductID>
	<Qty>2</Qty>
</row>
<row>
	<OrderID>1</OrderID>
	<OrderDate>2015-10-10T00:00:00</OrderDate>
	<ProductID>Wheels</ProductID>
	<Qty>4</Qty>
</row>
<row>
	<OrderID>2</OrderID>
	<OrderDate>2015-10-09T00:00:00</OrderDate>
	<ProductID>Ball</ProductID>
	<Qty>10</Qty>
</row>
```


Formatters:

JSON
https://jsonformatter.curiousconcept.com/

XML
http://www.freeformatter.com/xml-formatter.html#ad-output

JSON vs XML – Production (Auto)

JSON

```
SELECT
      Orders.OrderID,
      Orders.OrderDate,
      OrderDetails.ProductID,
      OrderDetails.Qty
FROM    @Orders AS
            Orders
JOIN    @OrderDetails AS
            OrderDetails
ON      Orders.OrderID =
            OrderDetails.OrderID
FOR     JSON
            AUTO;
```

XML

```
SELECT
      Orders.OrderID,
      Orders.OrderDate,
      OrderDetails.ProductID,
      OrderDetails.Qty
FROM    @Orders AS
            Orders
JOIN    @OrderDetails AS
            OrderDetails
ON      Orders.OrderID =
            OrderDetails.OrderID
FOR     XML
            AUTO;
```

JSON vs XML – Production (Auto)

JSON

```
[
{
        "OrderID":1,
        "OrderDate":"2015-10-10T00:00:00",
        "OrderDetails":
        [
                { "ProductID":"Bike", "Qty":2 },
                { "ProductID":"Helmet", "Qty":2 },
                { "ProductID":"Wheels",  "Qty":4 }
        ]
},
{
        "OrderID":2,
        "OrderDate":"2015-10-09T00:00:00",
        "OrderDetails":
        [
                { "ProductID":"Ball", "Qty":10 }
        ]
}
]
```

XML

```
<Orders OrderID="1" OrderDate="2015-10-10T00:00:00">
        <OrderDetails ProductID="Bike" Qty="2" />
        <OrderDetails ProductID="Helmet" Qty="2" />
        <OrderDetails ProductID="Wheels" Qty="4" />
</Orders>
<Orders OrderID="2" OrderDate="2015-10-09T00:00:00">
        <OrderDetails ProductID="Ball" Qty="10" />
</Orders>
```

JSON vs XML – Path with Nesting

JSON

```
SELECT
    Orders.OrderID,
    Orders.OrderDate,
    (
        SELECT
            OrderDetails.ProductID,
            OrderDetails.Qty
        FROM    @OrderDetails AS
                OrderDetails
        WHERE   Orders.OrderID =
                OrderDetails.OrderID
        FOR JSON PATH
    ) AS OrderDetails
FROM        @Orders Orders
FOR         JSON PATH,
            ROOT('Orders');
```

XML

```
SELECT
    Orders.OrderID,
    Orders.OrderDate,
    (
        SELECT
            OrderDetails.ProductID,
            OrderDetails.Qty
        FROM    @OrderDetails AS
                OrderDetails
        WHERE   Orders.OrderID =
                OrderDetails.OrderID
        FOR XML PATH('OrderDetail'), TYPE
    ) AS OrderDetails
FROM        @Orders Orders
FOR         XML PATH('Order'),
            ROOT('Orders');
```

JSON vs XML – Path with Nesting

JSON

```
{
"Orders":
[
        {
                "OrderID":1,
                "OrderDate":"2015-10-10T00:00:00",
                "OrderDetails":
                [
                        {"ProductID":"Bike","Qty":2},
                        {"ProductID":"Helmet","Qty":2},
                        {"ProductID":"Wheels","Qty":4}
                ]
        },
        …
]
}
```

XML

```
<Orders>
        <Order>
                <OrderID>1</OrderID>
                <OrderDate>2015-10-10T00:00:00</OrderDate>
                <OrderDetails>
                        <OrderDetail>
                                <ProductID>Bike</ProductID>
                                <Qty>2</Qty>
                        </OrderDetail>
                        <OrderDetail>
                                <ProductID>Helmet</ProductID>
                                <Qty>2</Qty>
                        </OrderDetail>
                        <OrderDetail>
                                <ProductID>Wheels</ProductID>
                                <Qty>4</Qty>
                        </OrderDetail>
                </OrderDetails>
        </Order>
        …
</Orders>
```

Unholy Unions

XML in JSON

```
{
    "UnholyUnion":
        "<DataList
DataElement=\"Yes, you can put
XML in JSON!\"\/><DataList
DataElement=\"But why would you
do this?\"\/>"
}
```

JSON in XML

```
<row>
    <UnholyUnion>
        [{"DataElement":"Yes, you
can put JSON in XML!"},
{"DataElement":"But why would
you do this?"}]
    </UnholyUnion>
</row>
```



```
SELECT
                (
                                SELECT          *
                                FROM            (VALUES
                                                                ('Yes, you can put XML in JSON!'),
                                                                ('But why would you do this?')) AS DataList(DataElement)
                                FOR XML AUTO
                ) AS UnholyUnion
FOR JSON PATH;

SELECT
                (
                                SELECT          *
                                FROM            (VALUES
                                                                ('Yes, you can put JSON in XML!'),
                                                                ('But why would you do this?')) AS DataList(DataElement)
                                FOR JSON AUTO
                ) AS UnholyUnion
FOR XML PATH;
```

Look ma, no tags!

XML

```
SELECT
      'Test'
FOR XML PATH('');
```

Results:

```
Test
```

JSON

```
SELECT
      'Test'
FOR JSON PATH;
```

Results:

```
Msg 13605, Level 16, State 1, Line 1
Unnamed tables cannot be used as JSON identifiers as well as unnamed columns cannot be used as key names. Add alias to the unnamed column/table.
```

You’re just not my type.

- XML is a data type.
- JSON is *not* a data type. Use NVARCHAR.
  - Already being stored as text.
    - But so was XML.
    - And so what? Convert over time. Convert on the fly.
  - Don’t have to update other SQL Server tools.
    - Boo hoo. Ok for now, but convert over time.
  - Client apps can handle native XML but not JSON.
    - Wait, what?
    - And so what if it’s text to the outside world; what about in-database performance?

JSON Support in SQL Server 2016 - Jovan Popovic (MSFT) 16 May 2015 7:17 AM
http://blogs.msdn.com/b/jocapc/archive/2015/05/16/json-support-in-sql-server-2016.aspx

MSSQL Server 2016 coming with JSON support (not really)
http://www.itworld.com/article/2925117/enterprise-software/mssql-server-2016-coming-with-json-support-not-really.html

Nettlesome Nesting

XML

```
SELECT
      CONVERT(xml,
            '<TextXML>I typed this.</
TextXML>'
      ) AS 'OuterTag'
FOR XML PATH('');
```

Results:

```
<OuterTag>
<TextXML>I typed this.</TextXML>
</OuterTag>
```

JSON

```
SELECT
      '{"TextJSON":"I typed this."}'
AS 'OuterTag'
FOR JSON PATH;
```

Results:

```
{"OuterTag":"{\"TextJSON\":\"I
typed this.\"}"}
```

Nettlesome Nesting - Workaround

```
SELECT
    (
        SELECT
            'I typed this.' AS TextJSON
        FOR JSON PATH
    ) AS 'OuterTag'
FOR JSON PATH;
```

Results:

```
{"OuterTag":{"TextJSON":"I typed this."}}
```

Well is it or isn't it?

- Without JSON type, can't use TRY_CONVERT() to validate.
- Use ISJSON() instead.
- Can use in CHECK constraint to ensure text field has valid JSON.
- Can then safely create calculated field based off JSON contents.

Is it “rows” or “records”?

OPENXML

```
DECLARE
    @i int,
    @x xml = '<x><a>1</
a><a>2</a></x>';

EXEC sp_xml_preparedocument
@i OUTPUT, @x;

SELECT * FROM
    OPENXML (@i, '/x/a', 2)
WITH (a varchar(10) '.');
```

Nodes

```
DECLARE
    @x xml = '<x><a>1</
a><a>2</a></x>';

SELECT
a.value('.','varchar(10)')
FROM  @x.nodes('/x/a') AS x(a);
```

But I haven’t prepared!

- There is no nodes-style syntax for JSON.
- OPENJSON has similar syntax to OPENXML.
- No prepare statement is needed.
  - Work in user-defined function?
  - Multiple in single SQL statement?
  - Performance?

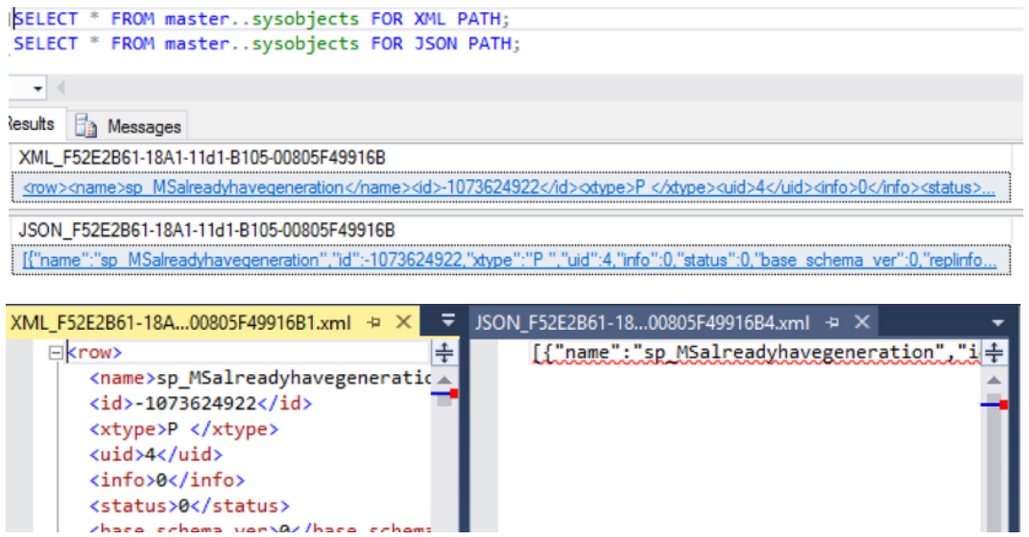
OPENJSON

```
DECLARE @j nvarchar(max) = '{"Orders": [
{"OrderID":1, "OrderDate": "2015-10-10"},
{"OrderID":2, "OrderDate": "2015-10-09"}]}';

SELECT
    OrderID, OrderDate
FROM OPENJSON (@j, '$.Orders')
WITH
(
    OrderID bigint,
    OrderDate datetime
) AS OrdersArray;
```

Jason who?

- SSMS sees JSON as special but doesn't deliver.



Yuck.

Results | Messages

	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	[{"name":"sysrscols","object_id":3,"schema_id":4,"parent_object_id":0,"type":"S ","type_desc":"SYSTEM_TABLE","create_date":"2015-09-20T03:...
2	ped":true,"is_published":false,"is_schema_published":false},{"name":"sysdbfrag","object_id":18,"schema_id":4,"parent_object_id":0,"type":"S ","ty...
3	odify_date":"2015-09-20T03:35:03.367","is_ms_shipped":true,"is_published":false,"is_schema_published":false},{"name":"sysowners","object_id":2...
4	pe_desc":"SYSTEM_TABLE","create_date":"2015-09-20T03:35:04.470","modify_date":"2015-09-20T03:35:04.480","is_ms_shipped":true,"is_publi...
5	ct_id":46,"schema_id":4,"parent_object_id":0,"type":"S ","type_desc":"SYSTEM_TABLE","create_date":"2009-04-13T12:59:08.217","modify_date...
6	":false,"is_schema_published":false},{"name":"sysendpts","object_id":56,"schema_id":4,"parent_object_id":0,"type":"S ","type_desc":"SYSTEM_T...
7	fy_date":"2015-09-20T03:35:04.390","is_ms_shipped":true,"is_published":false,"is_schema_published":false},{"name":"sysclsobjs","object_id":64,"s...
8	":"SYSTEM_TABLE","create_date":"2009-04-13T12:59:08.047","modify_date":"2009-04-13T12:59:08.057","is_ms_shipped":true,"is_published":fal...
9	dedrecoveryforks","object_id":81,"schema_id":4,"parent_object_id":0,"type":"S ","type_desc":"SYSTEM_TABLE","create_date":"2015-09-20T03:...
10	7","is_ms_shipped":true,"is_published":false,"is_schema_published":false},{"name":"sysqnames","object_id":90,"schema_id":4,"parent_object_id":0...
11	create_date":"2009-04-13T12:59:07.577","modify_date":"2009-04-13T12:59:07.603","is_ms_shipped":true,"is_published":false,"is_schema_publish...
12	_ms_shipped":true,"is_published":false,"is_schema_published":false},{"name":"queue_messages_1035150733","object_id":1051150790,"schema_i...
13	":false},{"name":"sqlagent_jobsteps_logs","object_id":1339151816,"schema_id":4,"parent_object_id":0,"type":"IT","type_desc":"INTERNAL_TABL...
14	t_id":1467152272,"schema_id":1,"parent_object_id":0,"type":"V ","type_desc":"VIEW","create_date":"2015-09-20T03:37:24.107","modify_date":"...

JSON Summary

- FOR JSON
  - Works pretty much like FOR XML.
- OPENJSON
  - Similar to OPENXML, but no need for separate “preparation” step (sp_xml_preparedocument).
- ISJSON(@JSON)
  - Similar to TRY_CONVERT(xml, '<x>xml</x>')
- JSON_VALUE(@JSON, '$.Order.Qty')
  - Similar to @XML.value

Vroom, vroom!

- Engine Improvements
  - Temporal Tables
  - Query Store
  - Dynamic Data Masking
  - Live Query Statistics
  - In-Memory OLTP Improvements
  - R

Dynamic Data Masking
https://msdn.microsoft.com/en-us/library/mt130841.aspx

Flux Capacitor

- Temporal Tables
  - Like Apple's Time Machine, but for your tables.
  - Can operate entirely transparently to existing applications.
  - Main table gets two extra time stamp fields.
  - History table:
    - Has same schema and stores old information.
    - Uses compression by default.
    - Can be queried directly.
    - Microsoft suggests using Azure stretch table.

Temporal Tables – Basic Syntax

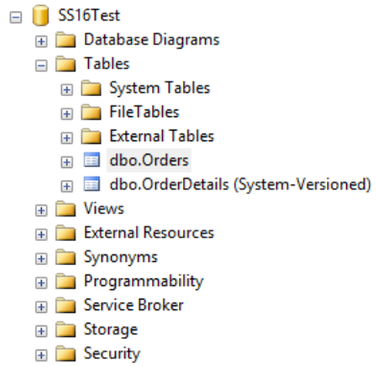
```
CREATE TABLE OrderDetails
    (
        OrderDetailID bigint IDENTITY PRIMARY KEY,
        OrderID bigint,
        ProductID varchar(50),
        Qty int,
        EffectiveStart datetime2
            GENERATED ALWAYS AS ROW START NOT NULL,
        EffectiveStop datetime2
            GENERATED ALWAYS AS ROW END NOT NULL,
        PERIOD FOR SYSTEM_TIME (EffectiveStart, EffectiveStop)
    )
    WITH (SYSTEM_VERSIONING =
        ON (HISTORY_TABLE = dbo.OrderDetailsHistory));
```


Temporal Tables
https://msdn.microsoft.com/en-us/library/dn935015.aspx

CREATE TABLE (Transact-SQL)
https://msdn.microsoft.com/en-us/library/ms174979.aspx

Temporal Tables – Time Travelling

```
SELECT   *
FROM     OrderDetails
   FOR
   SYSTEM_TIME
   AS OF @dt
ORDER BY   OrderDetailID;
```



Cha-ching!

- Query Store
  - Store ALL the query plans!
  - Your plan cache is only so big, but your disks are bigger.
  - Why recompile when you can reload?
  - Reboot? Would you like me to warm that cache up for you?
  - Don’t like this new plan? Return it for a full refund!

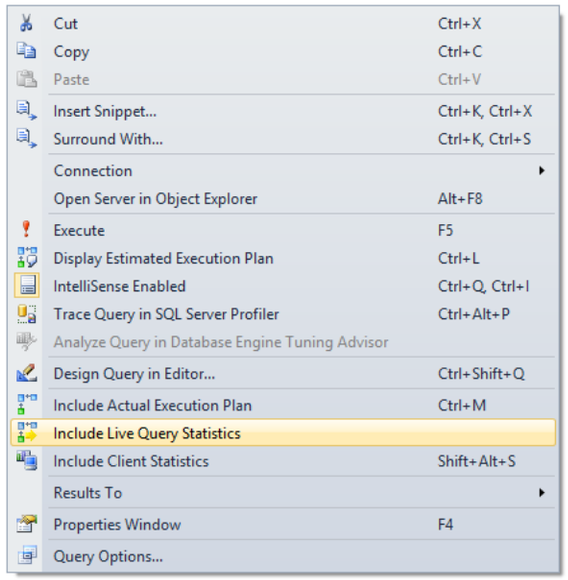
<REDACTED>

- Dynamic Data Masking
  - Hides sensitive information (such as personally-identifiable information– PII) from normal users.
  - There are a variety of masking functions which turn text into XXXX, dates into 1900-01-01, etc.
  - UNMASK permissions are required to see real data.

Dynamic Data Masking
https://msdn.microsoft.com/en-us/library/mt130841.aspx

Live Query Statistics

- Boom!
- Wow.
- “The performance impact of turning this on, on a production server, could be significant.” – Russ Thomas
- ☹

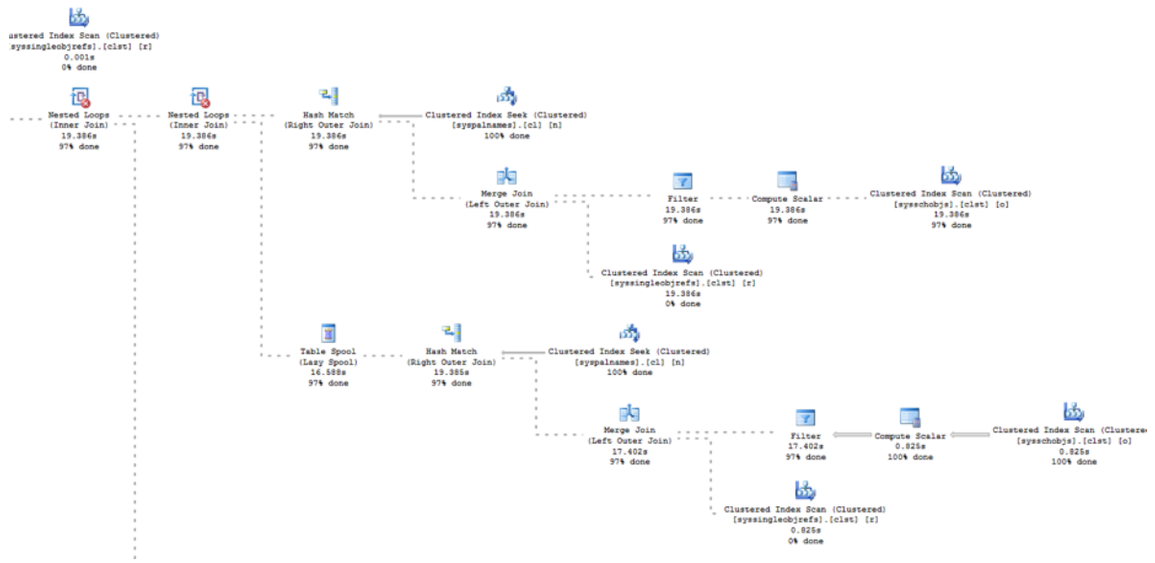


You have to enable this in advance. You can’t just start monitoring an existing long-running query.

Coming Soon SQL 2016 Live Query Statistics (LQS)
https://sqljudo.wordpress.com/2015/06/02/coming-soon-sql-2016-live-query-statistics-lqs/

Live Query Statistics
https://msdn.microsoft.com/en-us/library/dn831878.aspx

Feast your eyes…



Coming Soon SQL 2016 Live Query Statistics (LQS)
https://sqljudo.wordpress.com/2015/06/02/coming-soon-sql-2016-live-query-statistics-lqs/

But wait, there's more…

- Administrative Features
  - Always Encrypted
  - Stretch Database (Auto-Archive to Azure)
  - Row-Level Security
  - Multiple Secondary Readers for Load Balancing
  - Automatic TempDB Optimization
  - Polybase (T-SQL for Hadoop)

Row-Level Security
http://www.databasejournal.com/features/mssql/row-level-security-with-sql-server-2016.html

Hold this but don’t look inside.

- Always Encrypted
  - Defined for Specific Columns
  - DBA Can’t See the Data
  - Useful for Hosting Environments
  - Keys Stored on Client or in Key Store
  - Encrypted/Decrypted by Drivers
  - No Special App Coding
  - Choose Repeatable Encryption for Increased Speed but Decreased Security
  - Enterprise Only

See also Ed Leighton-Dick’s

To infinity, and beyond!

- Stretch Database
  - Configured on individual tables.
  - Define criteria for “cold” data.
  - SQL Server manages migration of data.
  - SQL Server seamlessly merges cold and hot data.
  - Can use full Azure SQL Database instance or special stretch database instance with unlimited storage.
  - Local backups don’t include remotely stored data.

SQL Server Stretch Database
https://azure.microsoft.com/en-us/services/sql-server-stretch-database/

MSDN Stretch Database
https://msdn.microsoft.com/en-us/library/dn935011.aspx

Paul Timmerman Talk
http://www.sqlsaturday.com/557/Sessions/Details.aspx?sid=53602

Kinda Sorta

- You can approximate many features in previous versions with a lot of effort.
  - Data Masking, Row-Level Security -> Views
  - STRING_SPLIT() -> Variety of Methods
  - TRUNCATE PARTITION -> Re-create
  - Stretch Database -> Jobs, Linked Servers, & Views
  - DROP IF EXISTS -> Extra IF
  - Temporal Tables -> Triggers & Views

Pay to Play

- Standard vs Enterprise
  - Most new features available in Standard.
  - Enterprise:
    - Polybase
    - Always Encrypted
    - Multiple Secondary Readers

Hey, hey, hey… goodbye!

- SQL Server 2016 didn't discontinue any features.
- Can no longer act as SQL Server 2005.
- Can't use MD5 / SHA1 in current compatibility mode.

Discontinued Features in 2016
https://msdn.microsoft.com/en-US/library/ms144262.aspx

Still Kicking

- Killed in Next Version:
  - SET ROWCOUNT for Modification
    - (Use TOP Instead)
  - Result sets from triggers.
  - Remote Servers
    - (Use Linked Servers Instead)
- Killed Eventually:
  - Omitting Semicolons
  - Database Mirroring

Deprecated Database Engine Features in SQL Server 2016
https://msdn.microsoft.com/en-us/library/ms143729.aspx

The People Have Spoken

- JSON Support is #1 Connect Item
  - 1070 Up-Votes
- But…
  - CREATE OR REPLACE
  - 7th Most Popular (442 Up-Votes)
  - Created in March of 2005

Add native support for JSON to SQL Server, a la XML (as in, FOR JSON or FROM OPENJSON) By - bret_m_lowery
https://connect.microsoft.com/SQLServer/Feedback/Details/673824

In the year 2000…

- SQL Server 2018?
- Brent Ozar: VMs mean 2005 & 2008 4eva
- Other popular Connect items:
  - Better Error Info for “String… truncated”
  - Error Table for Big Insert/Update Failures
  - Inline Scalar UDFs
  - Full Regex for LIKE, etc.
  - Built-in Tally Table (Table of Numbers)
  - Read/Write Table-Valued Parameters

Most of the popular Connect items are T-SQL or developer-centric improvements but they haven’t been getting a lot of love recently. It’s all about Big Data and big speed.

Databases Five Years from Today
http://www.brentozar.com/archive/2013/03/databases-five-years-from-today/

Connect items:

Please fix the "String or binary data would be truncated" message to give the column name By – Dwalker
https://connect.microsoft.com/SQLServer/Feedback/Details/339410

new virtual table: errors. It would analogous to the deleted and inserted tables By - danholmes
https://connect.microsoft.com/SQLServer/Feedback/Details/774754

The Scalar Expression function would speed performance while keeping the benefits of functions. - by Andrew Novick
https://connect.microsoft.com/SQLServer/Feedback/Details/273443

Regex functionality in pattern matching By - Simon Sabin
https://connect.microsoft.com/SQLServer/Feedback/Details/261342

Add a built-in table of numbers By - Erland Sommarskog
https://connect.microsoft.com/SQLServer/Feedback/Details/258733

Relax restriction that table parameters must be readonly when SPs call each other. By - Erland Sommarskog
https://connect.microsoft.com/SQLServer/Feedback/Details/299296

Shut Up and Take My Money

"While this has also been hotly requested, I'd caution you to **be careful what you ask for**. Users demanded XML support inside SQL Server, and then proceeded to use SQL Server as an XML query engine, sending CPU through the roof. **SQL Server is one of the world's most expensive application servers.**"

– Brent Ozar, 2015-05-04

Reading the SQL Server 2016 Data Sheet
http://www.brentozar.com/archive/2015/05/reading-the-sql-server-2016-data-sheet/

Download Now

- SQL Server 2016 Developer Version
- SQL Server Management Studio 2016
- They are *free*.
- You can install as a separate instance and application.
- Play with Azure SQL Database.

Evaluations

- Please fill them out.
- In person and online.
- Sessions and event.
- We can’t improve without your feedback.

Riley Major

- @RileyMajor | PASSMN@RileyMajor.com
- Scribnasium.com
- Enterprise Architect
- Manna Freight Systems, Inc.
- Worked with SQL Server since May of 2000
- PASSMN Board – Director of Technology
- Conference speaker
- Father of three girls