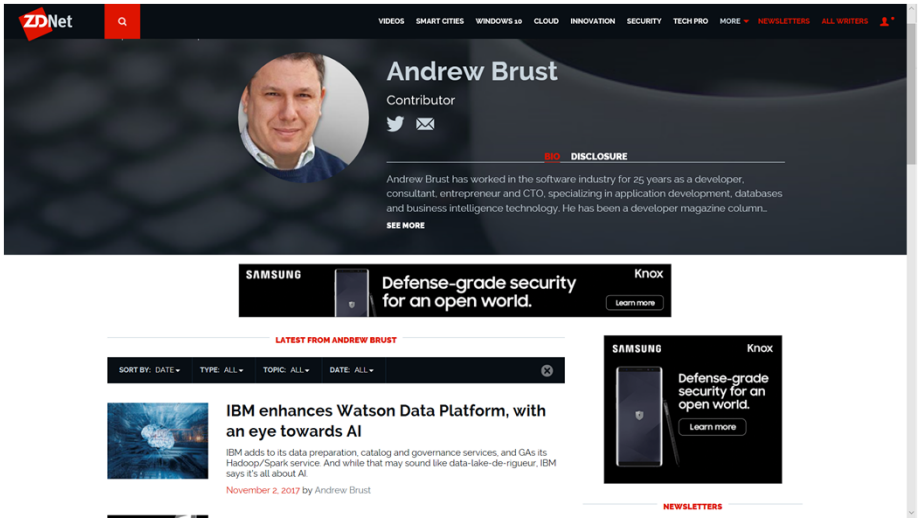
Meet Andrew

- BLUE BADGE INSIGHTS Founder and CEO
- Big Data blogger for ZDNet
- Microsoft Regional Director, MVP
- Co-chair Visual Studio Live!
- Twitter: @andrewbrust

Andrew's Blog (bit.ly/abrustzdnet)



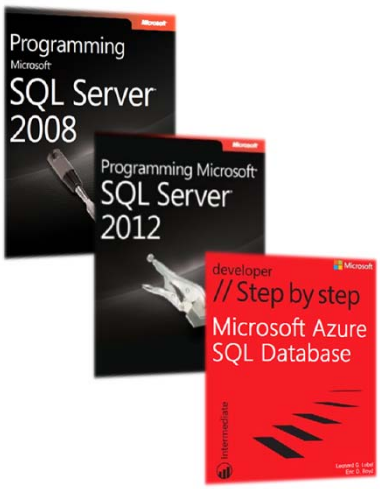
Meet Lenni

Leonard Lobel

- CTO & Co-Founder
  - Sleek Technologies, Inc.
- Principal Consultant
  - Tallan, Inc.
- Microsoft MVP
  - Data Platform
- Co-organizer
  - NYC .NET Developers Group
- Trainer/Speaker/Author
- Programming since 1979

Contact

- Email: lenni.lobel@sleektech.com
- Blog: lennilobel.wordpress.com
- Twitter: @lennilobel



Agenda

- **Introduction and Overview**
- **PART I**
  - New SQL Server 2016 Features
- **PART II**
  - Analytics
- **PART III**
  - Beyond Relational
- **Demos, demos, demos!**
  - **http://bit.ly/vslboston2018_sqlworkshop**

Download Slides and Code

http://bit.ly/vslboston2018_sqlworkshop

(all lower case!)

Schedule

- Workshop Begins
  - 9:00 AM
- Morning Break (15 minutes)
  - 11:00 AM
- Lunch (1 hour)
  - 1:00 PM
- Afternoon Break (15 minutes)
  - 4:00 PM
- Workshop Ends
  - 6:00 PM

Introduction and Overview

SQL Server Versions 2008 – 2016

(Developer Features)

2008

- Table-valued parameters (TVPs)
- New date and time types
- MERGE
- INSERT OVER DML
- GROUPING SETS
- hierarchyid
- FILESTREAM
- Geospatial
- Transparent Data Encryption (TDE)
- Change Data Capture(CDC)
- SQL Audit

2012

- SQL Server Data Tools (SSDT)
- Windowing (OVER) Enhancements
- 14 new general-purpose functions
- 8 new analytic windowing functions
- Server-side paging
- Sequences
- Metadata Discovery
- FileTable
- Columnstore indexes

2016

- DROP IF EXISTS (DIE)
- SESSION_CONTEXT
- Dynamic data masking (DDM)
- Row-level security (RLS)
- Always encrypted
- Stretch database
- JSON
- Temporal
- PolyBase
- QueryStore
- R integration
- Hekaton improvements

2008 R2

- BI Refresh

2014

- Hekaton

SQL Server 2016 New Features for Developers

2008

- Table-valued parameters (TVPs)
- New date and time types
- MERGE
- INSERT OVER DML
- GROUPING SETS
- hierarchyid
- FILESTREAM
- Geospatial
- Transparent Data Encryption (TDE)
- Change Data Capture(CDC)
- SQL Audit

2012

- SQL Server Data Tools (SSDT)
- Windowing (OVER) Enhancements
- 14 new general-purpose functions
- 8 new analytic windowing functions
- Server-side paging
- Sequences
- Metadata Discovery
- FileTable
- Columnstore indexes

2016

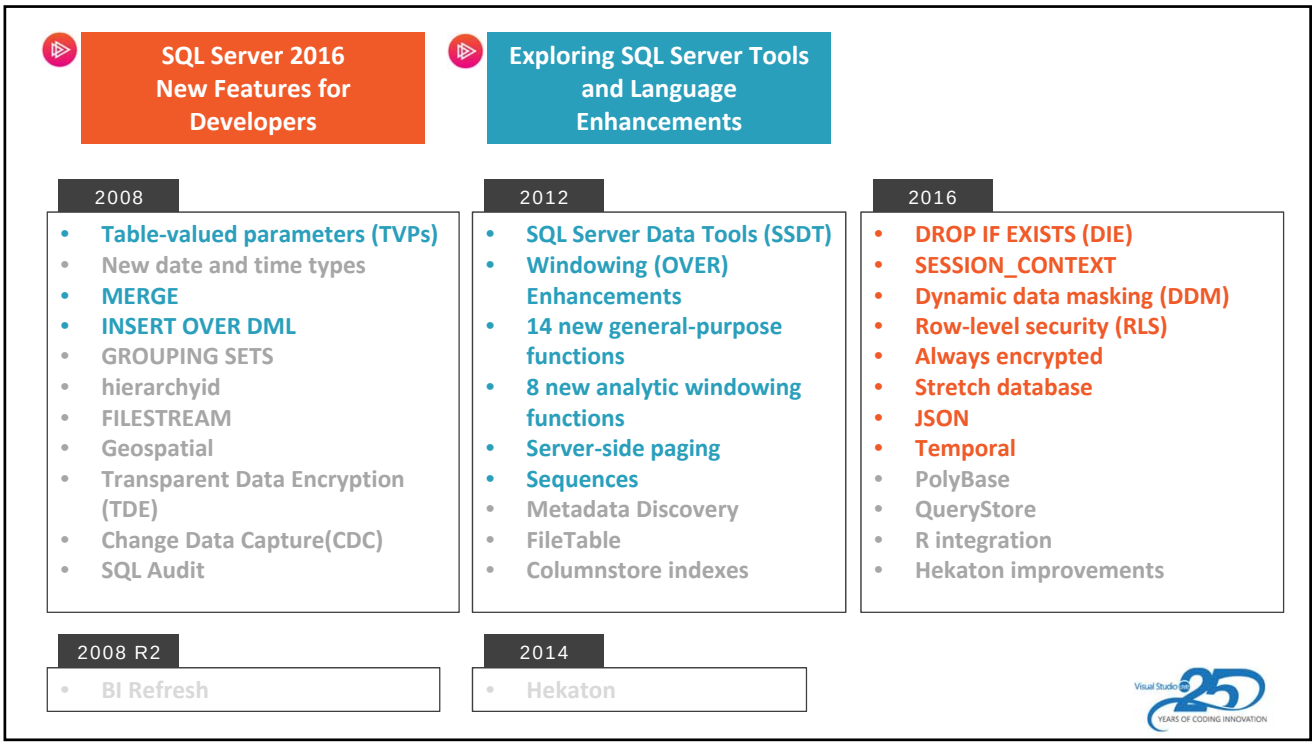
- DROP IF EXISTS (DIE)
- SESSION_CONTEXT
- Dynamic data masking (DDM)
- Row-level security (RLS)
- Always encrypted
- Stretch database
- JSON
- Temporal
- PolyBase
- QueryStore
- R integration
- Hekaton improvements

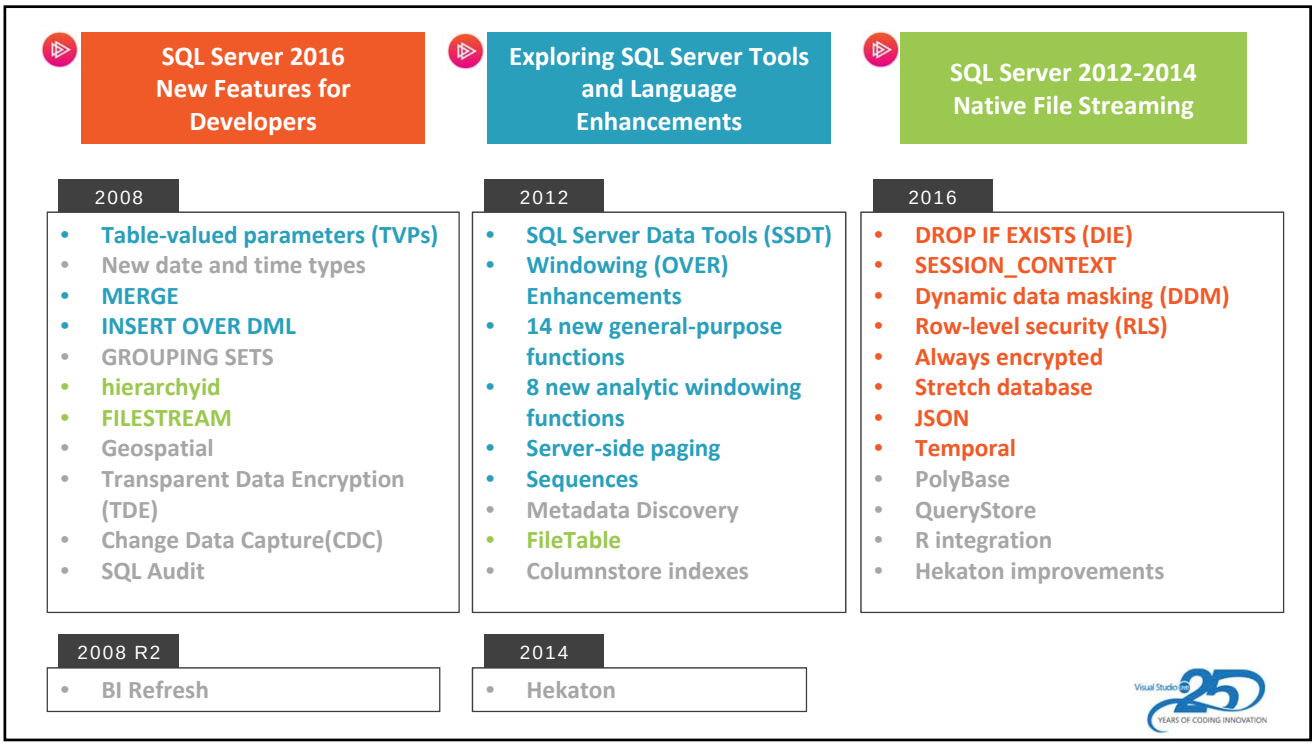
2008 R2

- BI Refresh

2014

- Hekaton


SQL Server 2016 New Features for Developers
Exploring SQL Server Tools and Language Enhancements
2008
Table-valued parameters (TVPs)
New date and time types
MERGE
INSERT OVER DML
GROUPING SETS
hierarchyid
FILESTREAM
Geospatial
Transparent Data Encryption (TDE)
Change Data Capture(CDC)
SQL Audit
2012
SQL Server Data Tools (SSDT)
Windowing (OVER) Enhancements
14 new general-purpose functions
8 new analytic windowing functions
Server-side paging
Sequences
Metadata Discovery
FileTable
Columnstore indexes
2016
DROP IF EXISTS (DIE)
SESSION_CONTEXT
Dynamic data masking (DDM)
Row-level security (RLS)
Always encrypted
Stretch database
JSON
Temporal
PolyBase
QueryStore
R integration
Hekaton improvements
2008 R2
BI Refresh
2014
Hekaton


SQL Server 2016 New Features for Developers
Exploring SQL Server Tools and Language Enhancements
SQL Server 2012-2014 Native File Streaming
2008
Table-valued parameters (TVPs)
New date and time types
MERGE
INSERT OVER DML
GROUPING SETS
hierarchyid
FILESTREAM
Geospatial
Transparent Data Encryption (TDE)
Change Data Capture(CDC)
SQL Audit
2012
SQL Server Data Tools (SSDT)
Windowing (OVER) Enhancements
14 new general-purpose functions
8 new analytic windowing functions
Server-side paging
Sequences
Metadata Discovery
FileTable
Columnstore indexes
2016
DROP IF EXISTS (DIE)
SESSION_CONTEXT
Dynamic data masking (DDM)
Row-level security (RLS)
Always encrypted
Stretch database
JSON
Temporal
PolyBase
QueryStore
R integration
Hekaton improvements
2008 R2
BI Refresh
2014
Hekaton

What's New in SQL Server 2016

- DIE
  - Drop If Exists
- Session context
  - Stateful dictionary
- Dynamic data masking
  - Mask sensitive columns
- Row-level security
  - Filter/block row-level access
- Always encrypted
  - Client-side encryption
- Stretch database
  - Hybrid cloud feature
- Temporal data
  - Point-in-time data access
- Built-in JSON support
  - Store/retrieve/transform JSON
- In-Memory OLTP
  - Improvements over 2014
- Query Store
  - Record & recall query plans
- PolyBase
  - Azure Hadoop/Blob integration
- R Integration
  - Advanced analytics

Data Warehousing

- BI Concepts Dimensional Model, Star Schema
- MPP, APS, Azure SQL DW
- Columnar Databases
- Columnstore indexes
- Vector Processing
- Batch Mode

Big Data and Analytics

- SQL Server Analysis Services
- Power BI
- Big Data, HDInsight, Hadoop, Azure Data Lake Analytics and Spark
- PolyBase
- Other components

Beyond Relational

- XML
  - Native data type for XML storage in the database
- JSON
  - Built-in support for JSON storage in the database
- FILESTREAM
  - Enhance storage of unstructured (BLOB) data in the database
- FileTable
  - Combines FILESTREAM with hierarchyid to expose a logical Win32 file system
- Geospatial
  - Support for planar (flat-earth) and geodetic (round-earth) models

2016

PART I

New SQL Server 2016 Features

2016

Drop If Exists (DIE)

2016

Just DIE (Drop If Exists) Please!

- If you hate this…

```
IF OBJECT_ID('dbo.Product, 'U') IS NOT NULL
 DROP TABLE dbo.Product;
IF EXISTS (SELECT * FROM sys.triggers
 WHERE name = 'trProductInsert')
  DROP TRIGGER trProductInsert
```

- …you're gonna love this!

```
DROP TABLE IF EXISTS dbo.Product
DROP TRIGGER IF EXISTS trProductInsert
```

2016

Objects That Can DIE

- AGGREGATE
- ASSEMBLY
- DATABASE
- DEFAULT
- INDEX
- PROCEDURE
- ROLE
- RULE
- SCHEMA
- SECURITY POLICY
- SEQUENCE
- SYNONYM
- TABLE
- TRIGGER
- TYPE
- VIEW

2016

Dynamic Data Masking (DDM)

2016

Introducing Dynamic Data Masking (DDM)

- Limit exposure to sensitive data by masking
  - Full – entire column is masked
  - Partial – show starting and/or ending characters of the column data, mask the rest with a custom string
  - Email – show the first character of the column data, mask the rest with XXX@XXXX.com
  - Random – entire column is replaced by random values
- Reveals masked data to queries
  - Data in the database is not changed
- Enforced at the database level
  - No impact at the application level

2016

Masking Table Columns

```
CREATE TABLE Customer(
  FirstName varchar(20)
   MASKED WITH (FUNCTION='partial(1, "...", 0)'),
  LastName varchar(20),
  Phone varchar(12)
   MASKED WITH (FUNCTION='default()'),
  Email varchar(200)
   MASKED WITH (FUNCTION='email()'),
  Balance money
   MASKED WITH (FUNCTION='random(1000, 5000)'))

ALTER TABLE Customer
 ALTER COLUMN LastName
 ADD MASKED WITH (FUNCTION='default()')
```

2016

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types

2016

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types
default()	Show **xxxx** mask (strings), or minimum value (other types)	Yes	Yes	Yes	Yes

2016

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types
default()	Show **xxxx** mask (strings), or minimum value (other types)	Yes	Yes	Yes	Yes
partial(*a*, 'x', *b*)	Show first *a* characters, custom mask, and last *b* characters	Yes	No	No	No

2016

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types
default()	Show **xxxx** mask (strings), or minimum value (other types)	Yes	Yes	Yes	Yes
partial(*a*, 'x', *b*)	Show first *a* characters, custom mask, and last *b* characters	Yes	No	No	No
email()	Show first character and **XXX@XXXX.com**	Yes	No	No	No

2016

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types
default()	Show **xxxx** mask (strings), or minimum value (other types)	Yes	Yes	Yes	Yes
partial(*a*, 'x', *b*)	Show first *a* characters, custom mask, and last *b* characters	Yes	No	No	No
email()	Show first character and **XXX@XXXX.com**	Yes	No	No	No
random(*a*, *b*)	Show random value between *a* and *b*	No	Yes	No	No

2016

Discovering Masked Columns

- sys.columns
  - is_masked
  - masking_function
- sys.masked_columns
  - Inherits from sys.columns
  - Filters to show only masked columns
    - WHERE is_masked = 1

2016

Mask Permissions

- DDM is based on user permissions
- Create a table with masked columns
  - No special permission required
- Add, replace, or remove a column mask
  - Requires ALTER ANY MASK permission
- View unmasked data in masked columns
  - Requires UNMASK permission
- Updating data in a masked column
  - No special permission

2016

Dynamic Data Masking (DDM)

demo

2016

DDM Limitations and Considerations

- DDM cannot be used with
  - FILESTREAM columns
  - COLUMN_SET, or a sparse column that's part of a COLUMN_SET
  - Computed columns
    - But will return masked data if it depends on a masked column
  - Key for FULLTEXT index
  - Encrypted columns (Always Encrypted)
- Masking is a one-way street
  - Once masked, the actual data can never be obtained
  - An ETL process from a source with masked columns results in an irreversible data loss when loaded into the target environment

2016

Row-level Security (RLS)

2016

Introducing Row-level Security (RLS)

- Restrict access to individual rows in a table
  - Create predicate functions
  - Write custom logic to control user access to every row
- Security policy
  - Bind the functions to tables as a filter or block predicate
  - SQL Server filters and blocks user access to individual rows
  - Can enable/disable the policy as desired

2016

Filter and Block Predicates

- Filter predicate
  - SELECT, UPDATE, DELETE
    - Can't select, update, or delete rows that violate the predicate
- Block predicate
  - AFTER INSERT, AFTER UPDATE
    - Can't insert or update rows to values that would violate the predicate
  - BEFORE UPDATE, BEFORE DELETE
    - Can't update or delete rows that violate the predicate
    - Implied when combined with filter predicate

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes
AFTER INSERT block	Yes	No	Yes

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes
AFTER INSERT block	Yes	No	Yes
AFTER UPDATE block	Yes	Yes	No

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes
AFTER INSERT block	Yes	No	Yes
AFTER UPDATE block	Yes	Yes	No
BEFORE UPDATE block	No	N/A	N/A

2016

Combining RLS Predicates

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes
AFTER INSERT block	Yes	No	Yes
AFTER UPDATE block	Yes	Yes	No
BEFORE UPDATE block	No	N/A	N/A
BEFORE DELETE block	No	N/A	N/A

2016

Creating Security Predicate Functions

- Write a security predicate function
  - Ordinary inline table-valued function (TVF)
    - Must be schema-bound
  - Accept any parameters of any type
    - Map these parameters to column values
- Implement your own custom logic in T-SQL
  - Examine the row via the columns passed in as parameters
    - Determine if access should be allowed or denied
  - Return a scalar 1 (allow) or nothing at all (deny)
  - Encapsulate logic inside WHERE clause of a single SELECT statement inside the TVF

2016

Creating Security Predicate Functions

```
CREATE FUNCTION sec.fn_MySecurityPredicate(@Parm1 AS int, ...)
 RETURNS TABLE
 WITH SCHEMABINDING
AS
  -- SQL Server passes in column values of each row via parameters

  RETURN
    SELECT 1 AS Result
    WHERE ...
      -- Custom logic here examines the parameters (column values)
      -- passed in, and determines the row's accessibility
```

2016

RLS Security Policy

- Create a security policy
  - Add filter and block predicates to the policy
- Bind each predicate function to a table
  - Map table columns to the TVF parameters
    - SQL Server will call the TVF to determine the accessibility of each row

2016

RLS Security Policy Examples

- With filter predicate

```
CREATE SECURITY POLICY sec.MySecurityPolicy
 ADD FILTER PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable
 WITH (STATE = ON)
```

- With AFTER INSERT and AFTER UPDATE block predicates

```
CREATE SECURITY POLICY sec.MySecurityPolicy
 ADD BLOCK PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable AFTER INSERT,
 ADD BLOCK PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable AFTER UPDATE,
 WITH (STATE = ON)
```

2016

Getting Started with Row-Level Security (RLS)

demo

2016

Identifying Users for RLS

- Credentials supplied for the database connection
  - SQL Server login (username and password)
  - Windows authentication
  - Obtain the username from DATABASE_PRINCIPAL_ID
- Different strategy required for n-tier applications
  - Typically, all users connect to the database using the same service account from the application tier
  - DATABASE_PRINCIPAL_ID is the same for every user
- Solution: Use new SESSION_CONTEXT feature
  - Store the application level user ID as a readonly value in session context

2016

Introducing SESSION_CONTEXT

- What is “session context”?
  - Stateful dictionary (key/value pair) object
    - Key is a Unicode string
    - Value is a sql_variant (any data type)
  - Retains state for the lifetime of the connection
- SESSION_CONTEXT()
  - Built-in function that returns a value from session context by key
- sp_set_session_context
  - System stored procedure that stores a value to session context
  - Specify **@key**, **@value**, and optionally, **@read_only**

2016

Using Row-Level Security (RLS) with SESSION_CONTEXT() for N-Tier Applications

demo

2016

Always Encrypted (AE)

2016

Traditional SQL Server Encryption Features

- Column (cell-level) encryption
  - Uses certificates or symmetric keys
- Database (page-level) and backup encryption
  - Transparent Data Encryption (TDE)
  - Uses TDE certificate with database encryption keys (DEKs)
- Keys and certificates are stored in the database
  - Risk of security breach at the database level
- Data is only encrypted “at rest”
  - Risk of security breach while “in flight”

2016

Introducing Always Encrypted

- Always Encrypted in SQL Server 2016
  - Based on keys managed outside the database
  - Keys are never revealed to SQL Server
- Separating those who own the data from those who manage it
  - Uses client side drivers to encrypt/decrypt on the fly
- SQL server is incapable of decrypting on its own
  - Data is always encrypted in flight
- Enable Always Encrypted
  - Use T-SQL or the Always Encrypted Wizard in SSMS

2016

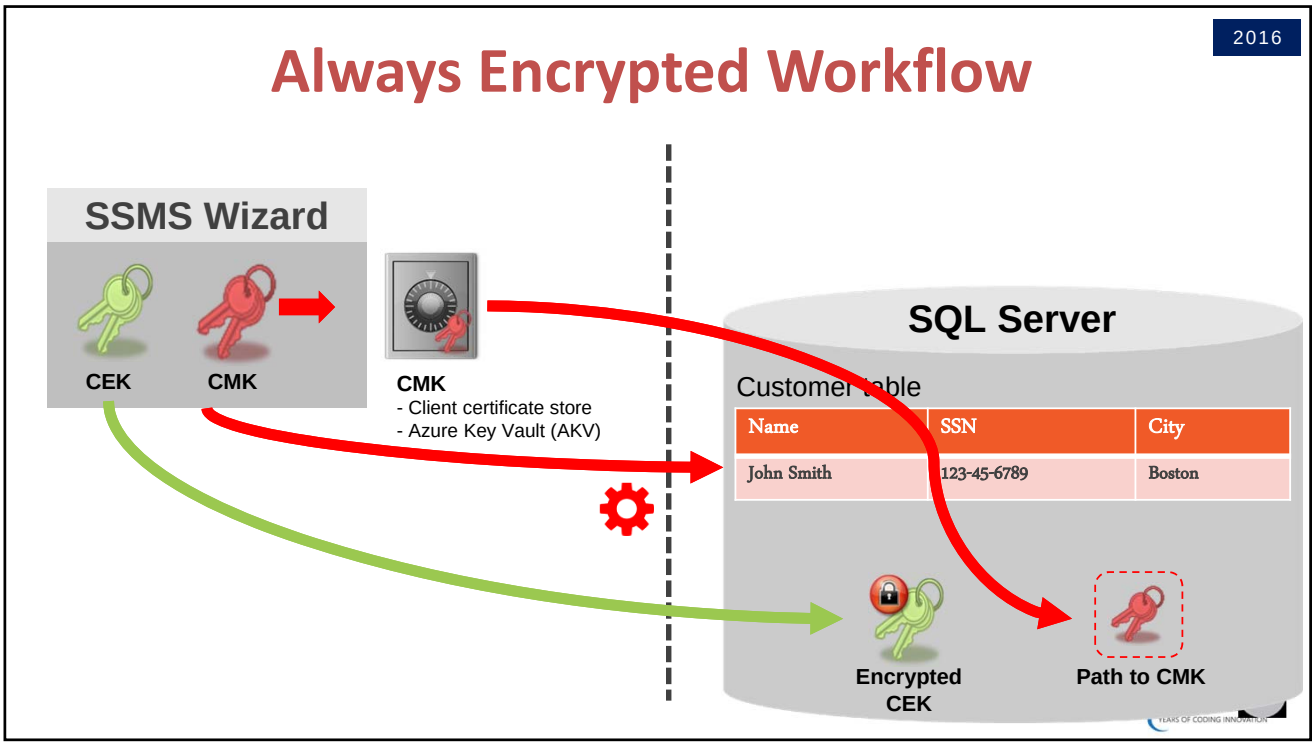
Encryption Types

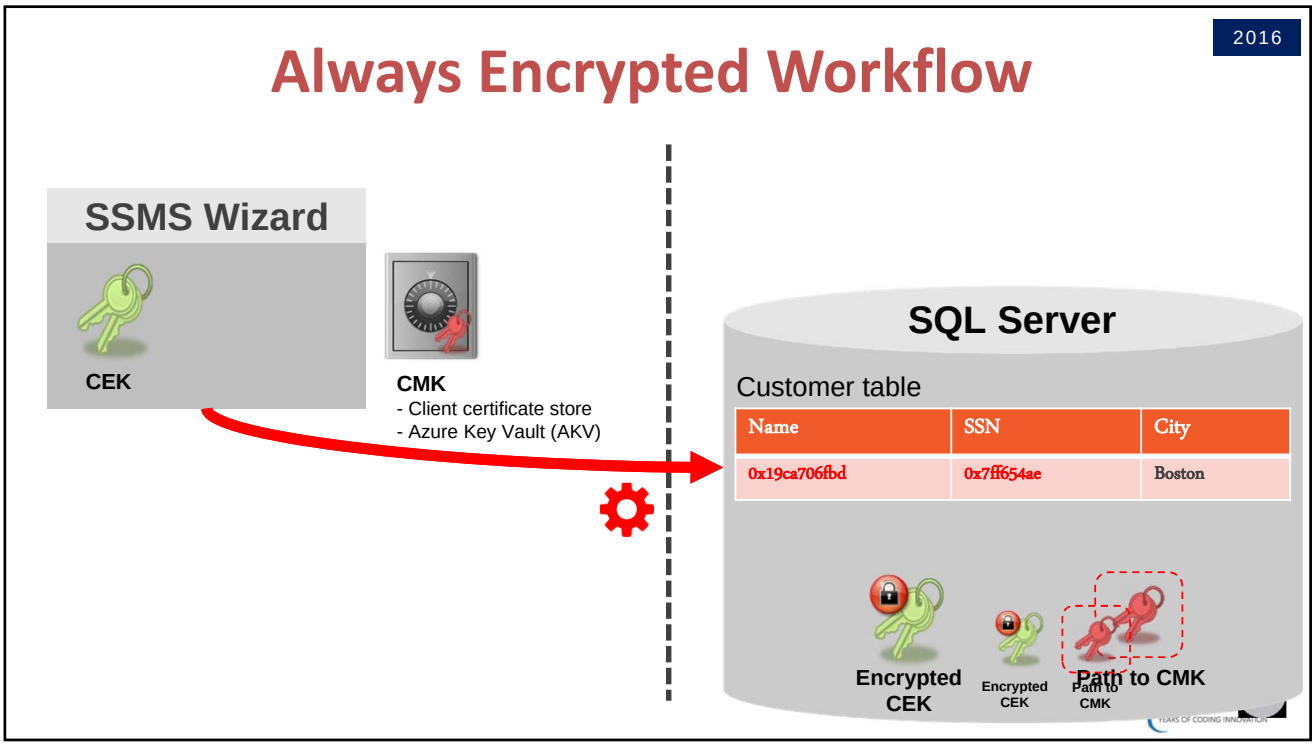
- Randomized
  - Unpredictable, more secure
  - No support for equality searches, joins, grouping, indexing
  - Use for data that is returned but not queried
- Deterministic
  - Predictable, less secure
  - Use for data that must be queried
  - Easier to guess by examining encryption patterns
    - Increased risk for small value sets (e.g., True/False)

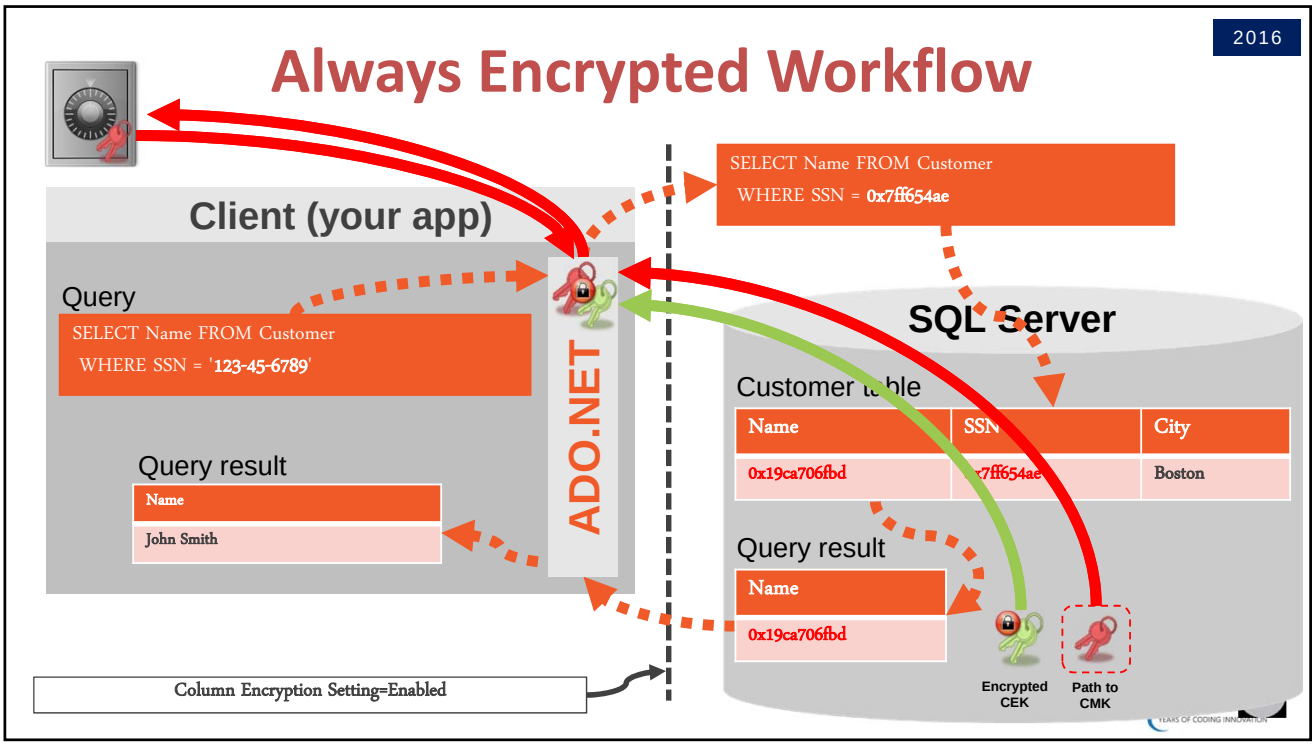
2016

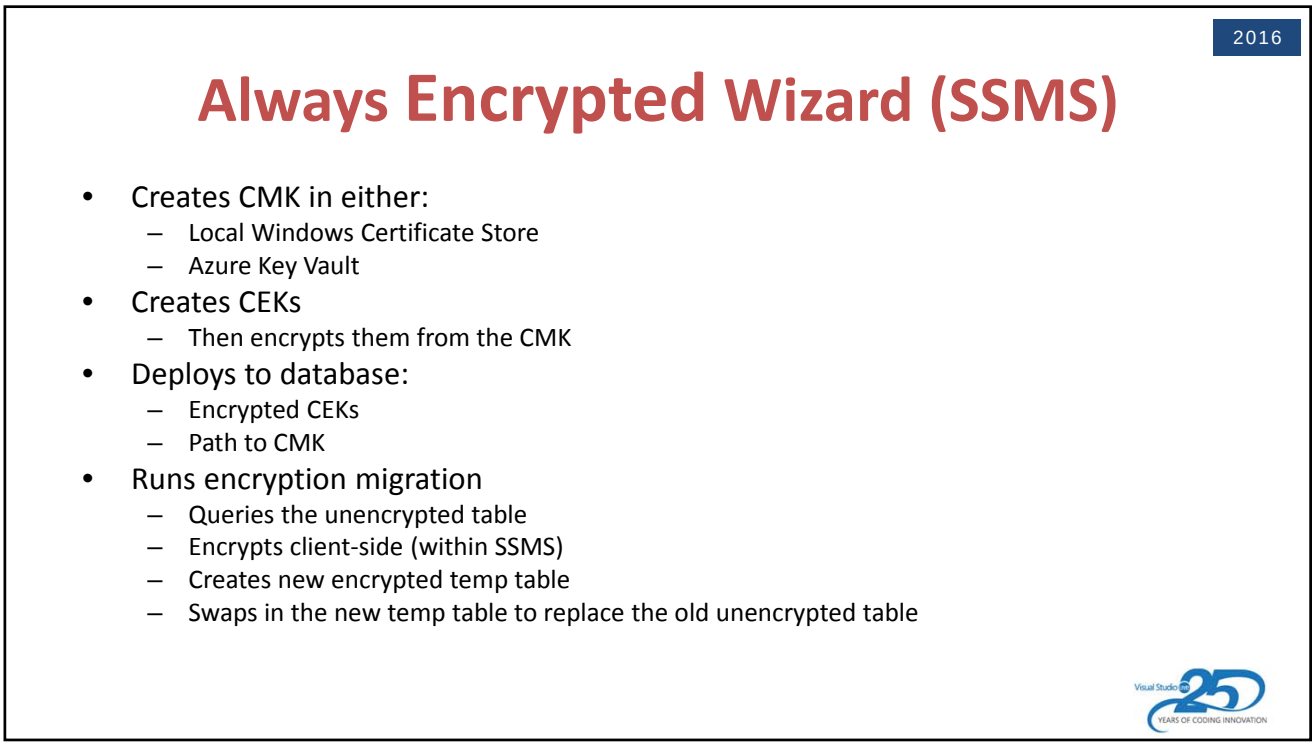
Encryption Keys

- Column Encryption Keys (CEK)
  - Used to encrypt values in specific columns
  - Encrypted versions of each CEK is stored in the database
- Column Master Keys (CMK)
  - Used to encrypt all the CEKs
  - Must be stored externally in a secure key store
    - Key store providers: Azure Key Vault, Certificate store, HSM
- CMK rotation
  - Each CEK can have two encrypted values from two CMKs


2016
Always Encrypted Workflow
SSMS Wizard
CEK
CMK
CMK
- Client certificate store
- Azure Key Vault (AKV)
SQL Server
Customer table
Name
SSN
City
John Smith
123-45-6789
Boston
Encrypted CEK
Path to CMK


2016
Always Encrypted Workflow
SSMS Wizard
CEK
CMK
- Client certificate store
- Azure Key Vault (AKV)
SQL Server
Customer table
Name
SSN
City
0x19ca706fbd
0x7ff654ae
Boston
Encrypted CEK
Encrypted CEK
Path to CMK
Path to CMK


2016
Always Encrypted Workflow
Client (your app)
Query
SELECT Name FROM Customer
WHERE SSN = '123-45-6789'
Query result
Name
John Smith
ADO.NET
SELECT Name FROM Customer
WHERE SSN = 0x7ff654ae
SQL Server
Customer table
Name
SSN
City
0x19ca706fbd
Boston
Query result
Name
0x19ca706fbd
Encrypted CEK
Path to CMK
Column Encryption Setting=Enabled


2016
Always Encrypted Wizard (SSMS)
• Creates CMK in either:
– Local Windows Certificate Store
– Azure Key Vault
• Creates CEKs
– Then encrypts them from the CMK
• Deploys to database:
– Encrypted CEKs
– Path to CMK
• Runs encryption migration
– Queries the unencrypted table
– Encrypts client-side (within SSMS)
– Creates new encrypted temp table
– Swaps in the new temp table to replace the old unencrypted table

2016

Always Encrypted Catalog Views

- **`sys.column_master_keys`**
  - Identifies each CMK
  - Contains external path to CMK location
- **`sys.column_encryption_keys`**
  - Identifies each CEK
- **`sys.column_encryption_key_values`**
  - Contains CMK-encrypted values of each CEK
- **`sys.columns`**
  - New metadata columns to identify encrypted columns

2016

Always Encrypted (AE)

demo

2016

CMK Rotation

- CEKs encrypt all your sensitive data
  - Which is why they are encrypted by a CMK
- The CMK encrypts all your CEKs
  - When the CMK is compromised, all your sensitive data is compromised
- Solution: Rotate the CMK
  - Create a new CMK
  - Re-encrypt the CEKs with the new CMK
  - PowerShell script available at
    - https://blogs.msdn.microsoft.com/sqlsecurity/2015/08/13/always-encrypted-key-rotation-column-master-key-rotation/
  - SQL Server Management Studio has integrated GUI support

2016

AE Limitations and Considerations

- Unsupported data types
  - xml, rowversion, image, ntext, text, sql_variant, hierarchyid, geography, geometry
- Also not supported for
  - FILESTREAM, ROWGUIDCOL, sparse, or partitioning columns
  - Fulltext indexes
  - Columns with default constraints
  - Temporal tables
  - Stretch database

2016

AE Limitations and Considerations (cont.)

- Entity Framework 6 considerations
  - http://blogs.msdn.com/b/sqlsecurity/archive/2015/08/27/using-always-encrypted-with-entity-framework-6.aspx
- Additional management to install certificates on all clients
- And more...
  - http://blogs.sqlsentry.com/aaronbertrand/t-sql-tuesday-69-always-encrypted-limitations/

2016

Stretch Database

2016

Introducing Stretch Database

- Store portions of a database in the cloud
- Remote Data Archive (RDA)
  - Keep “hot” data in local SQL Server database (on-premises)
  - Seamlessly migrate “cold” data to Azure SQL Database
- Stretch Database Advisor
  - Downloadable as part of the SQL Server 2016 Upgrade Advisor
  - Helps identify database and table candidates for stretch
- Enable stretch database
  - Use T-SQL or the Enable Database for Stretch Wizard in SSMS

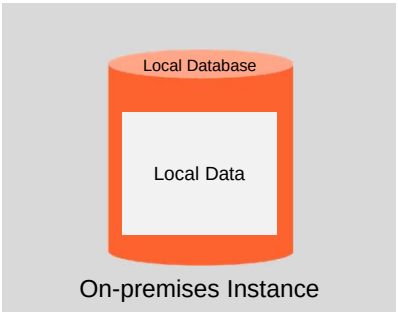
2016

Stretch Database Terminology

- Local database
  - The on-premises SQL Server database being “stretched”
- Eligible data
  - Data in the local database that has not yet been moved to the cloud
- Remote endpoint
  - The Azure SQL Database in the cloud
- Remote data
  - Data that has already been moved from the local database to the cloud

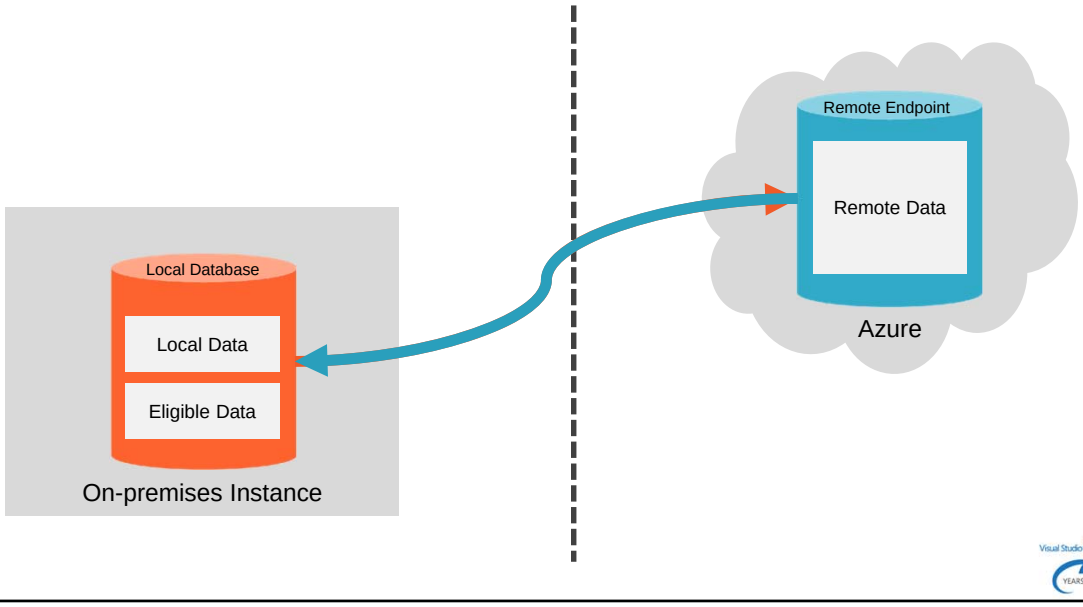
2016

Stretch Database Workflow



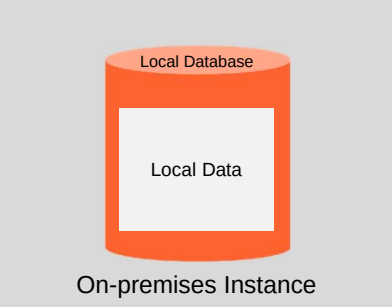
2016

Stretch Database Workflow



2016

Stretch Database Workflow



- sys.sp_rda_deauthorize_db
- sys.sp_rda_reauthorize_db

2016

Stretching a table

demo

2016

Stretching Selected Rows

- The entire table doesn't need to be stretched
  - Write a predicate function to filter the rows to be migrated
  - Easily filter by date, status, or other constant
- Filter predicate restrictions
  - Expression must be deterministic
  - Can't inherently implement a sliding window function
- Update filter predicates in place
  - Signature must not change
  - New function must be less restrictive than current one

2016

Stretch Filter Predicate Functions

```
CREATE FUNCTION sec.fn_MyStretchPredicate(@Parm1 AS int, ...)
 RETURNS TABLE
 WITH SCHEMABINDING
AS
    -- SQL Server passes in column values of each row via parameters

    RETURN
      SELECT 1 AS IsEligible
      WHERE ...
        -- Custom logic here examines the parameters (column values)
        -- passed in, and determines if the row should be migrated
```

2016

Stretch Filter Predicate Functions

```
CREATE FUNCTION sec.fn_MyStretchPredicate(@Parm1 AS int, ...)
 RETURNS TABLE
 WITH SCHEMABINDING
AS
    -- SQL Server passes in column values of each row via parameters

    RETURN
      SELECT 1 AS IsEligible
      WHERE ...
        -- Custom logic here examines the parameters (column values)
        -- passed in, and determines if the row should be migrated
```

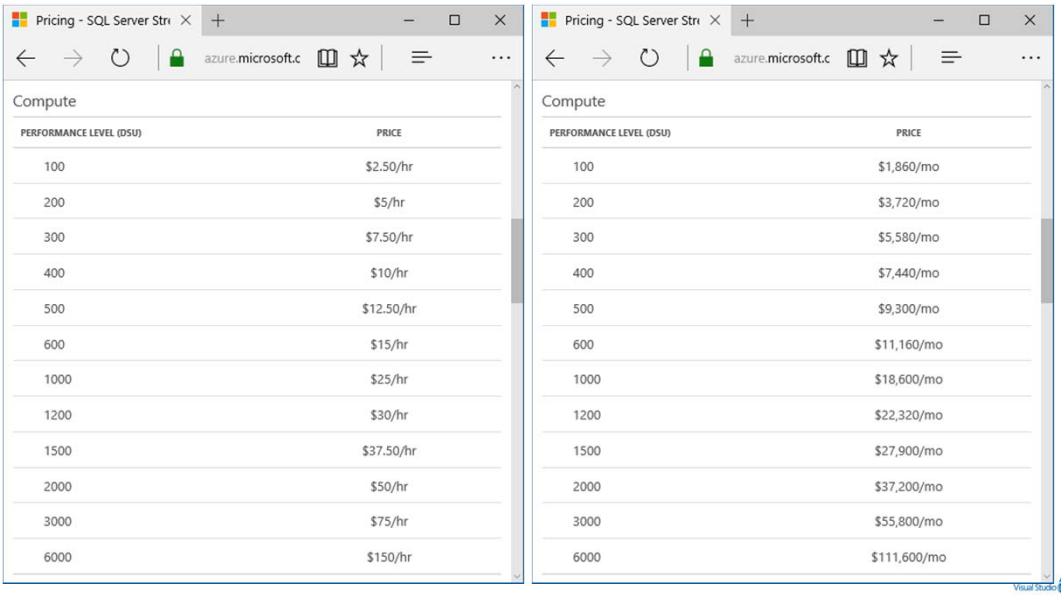
2016

Stretching selected rows

demo

2016

Stretch Pricing



Pricing - SQL Server Stre

azure.microsoft.c

Compute

PERFORMANCE LEVEL (DSU)	PRICE
100	$2.50/hr
200	$5/hr
300	$7.50/hr
400	$10/hr
500	$12.50/hr
600	$15/hr
1000	$25/hr
1200	$30/hr
1500	$37.50/hr
2000	$50/hr
3000	$75/hr
6000	$150/hr

Pricing - SQL Server Stre

azure.microsoft.c

Compute

PERFORMANCE LEVEL (DSU)	PRICE
100	$1,860/mo
200	$3,720/mo
300	$5,580/mo
400	$7,440/mo
500	$9,300/mo
600	$11,160/mo
1000	$18,600/mo
1200	$22,320/mo
1500	$27,900/mo
2000	$37,200/mo
3000	$55,800/mo
6000	$111,600/mo

2016

Stretch Database Limitations and Considerations

- Unsupported table types
  - Memory-optimized tables (Hekaton)
  - Replicated tables
  - FILESTREAM/FileTable
  - Tables enabled for Change Tracking or Change Data Capture (CDC)
  - Tables with more than 1023 column or 998 indexes
- Unsupported data types and column properties
  - timestamp, sql_variant, xml, geography, geometry, hierarchyid, CLR UDTs, COLUMN_SET, computed columns
- Unsupported indexes
  - XML, full-text, spatial, indexed views into the table

2016

Stretch Database Limitations and Considerations (cont.)

- Unsupported constraints
  - Check constraints
  - Default constraints
  - Foreign key constraints out of the table (parent tables)
- Uniqueness not enforced UNIQUE constraints
- UPDATE and DELETE not supported
- ALTER TABLE not supported
- Views over stretched tables
  - Can't create index on the view
  - Can't update or delete from the view (but you can insert into the view)

2016

Temporal Data

2016

Introducing Temporal Tables

- System version tables
- Point-in-time data access
  - Query updated and delete data, not just current data
  - Seamless and transparent
- Four primary use cases
  - Time travel
  - Slowly changing dimensions
  - Auditing
  - Accidental data loss recovery

2016

Using Temporal

- Create an ordinary table
  - Must have primary key column
  - Must have two period (start and end date) **datetime2** columns
- Enable the table for temporal
  - Creates history table with same schema, but without constraints
  - Automatically records updates and deletes to the history table
- Query to point in time
  - Include **FOR SYSTEM_TIME AS OF** in your SELECT statement
- Manage schema changes
  - ALTER TABLE automatically updates the history table
    - Some schema changes (e.g., new IDENTITY or computed columns) require turning temporal off, applying the changes to both tables, and then turning it back on

2016

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.DepartmentHist))
```

2016

Querying a Temporal Table

```
DECLARE @ThirtyDaysAgo datetime2 =
 DATEADD(d, -30, SYSDATETIME())


SELECT *
 FROM Employee
 FOR SYSTEM_TIME AS OF @ThirtyDaysAgo
 ORDER BY EmployeeId
```

2016

Temporal Data

demo

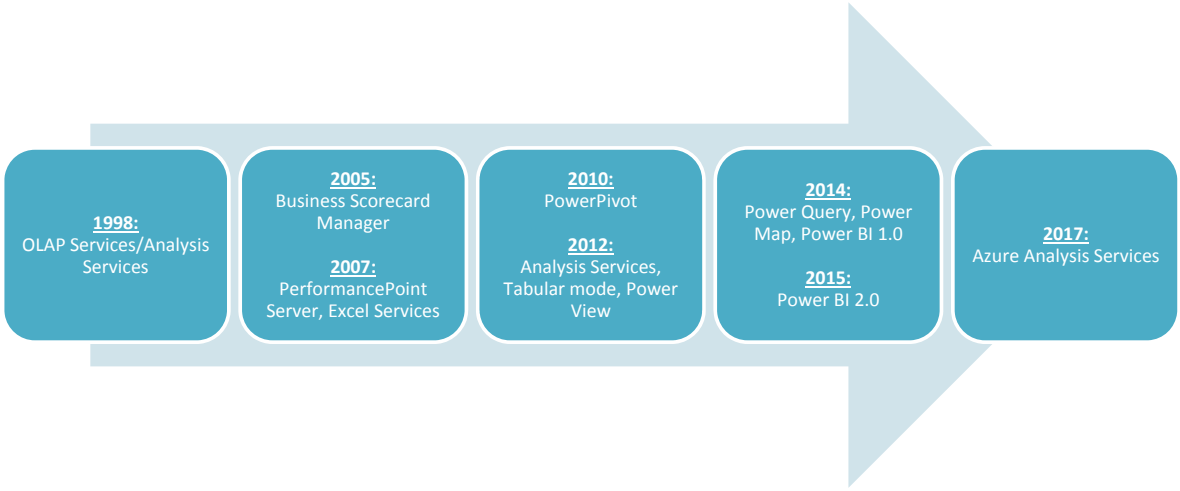
2016

Temporal Limitations and Considerations

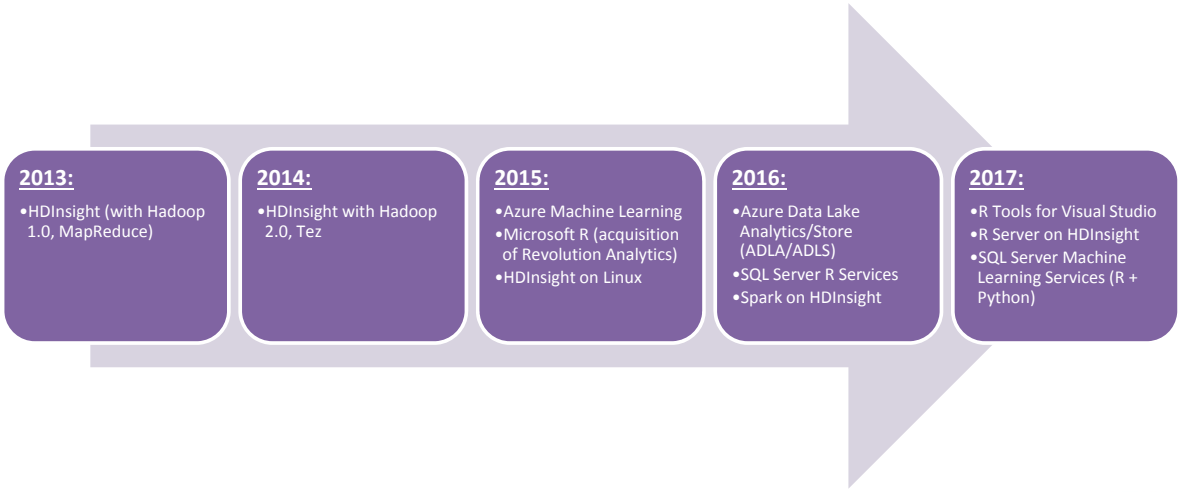
- Triggers
  - INSTEAD OF triggers are unsupported
  - AFTER triggers are supported on the current table only
- Cascading updates and deletes are not supported
- In-memory OLTP (Hekaton) is not supported
- FILESTREAM/FileTable is not supported
- INSERT and UPDATE statements cannot reference the period columns
- Works with other new SQL Server 2016 features
  - DDM, RLS, Always Encrypted, Stretch

PART II
Analytics

Microsoft BI Timeline



Microsoft Big Data Timeline

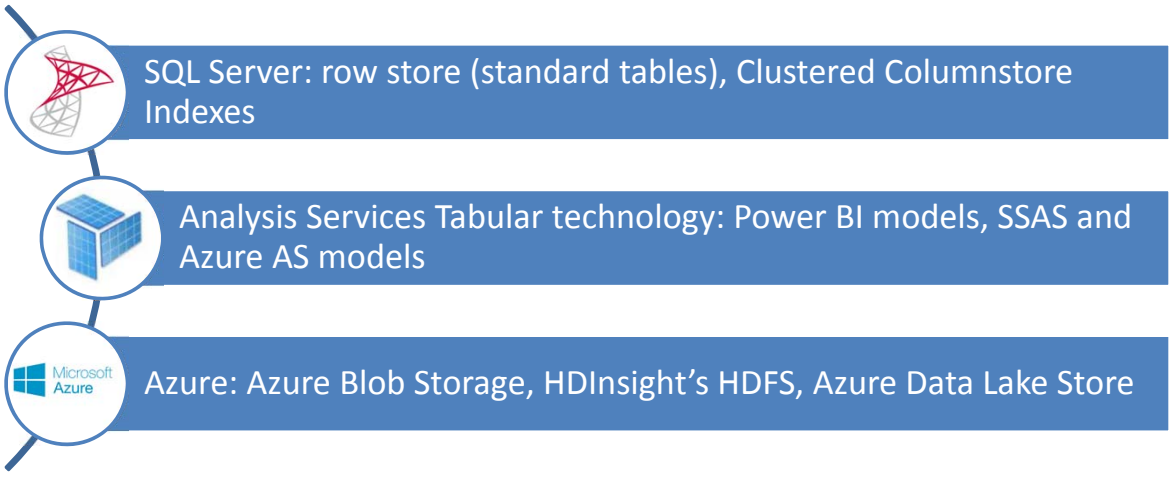


SQL Server Analytics Timeline

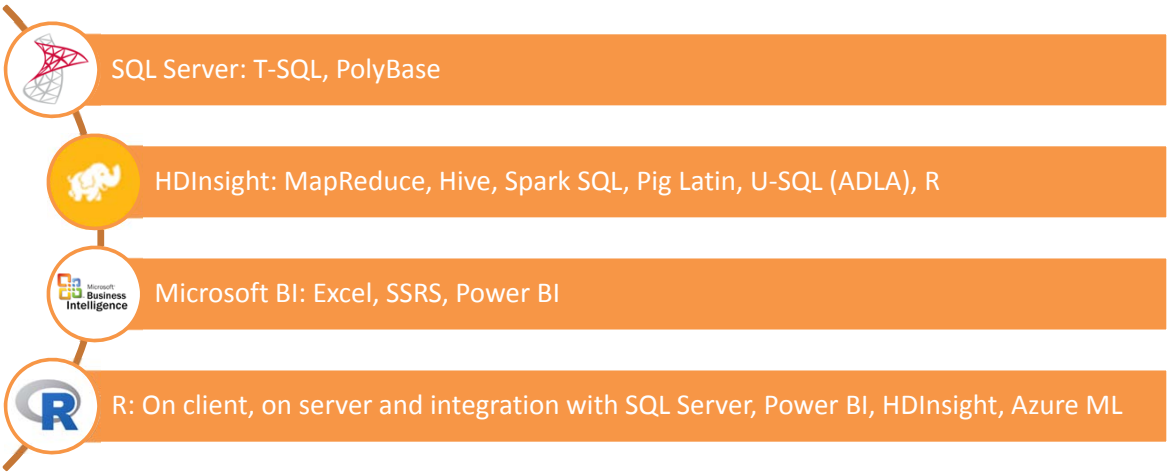
The Result: Lots to Learn

- There are so many components
- Each is rich and complex
- But they all connect
- Let's look at ways to slice this…

Break it down by: Repositories

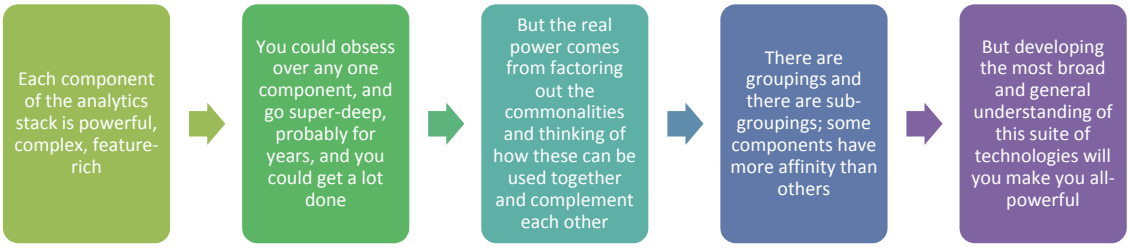


Break it down by: Query front-ends



Break it down by: Client Tools

How to think about it



What We'll Cover

Our Agenda: Preliminaries

- Data warehouse concepts
- Introducing our dataset (NYC 311 service calls)
- Open in Power BI, inspect

Our Agenda: Big Data

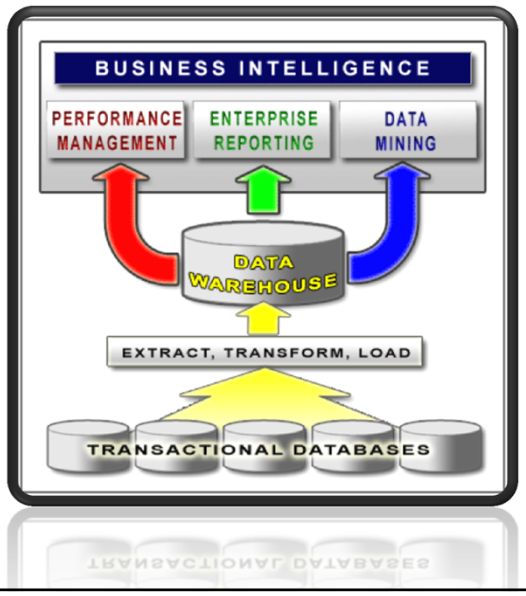
- Discuss HDInsight
- Query and process in Hadoop
  - MapReduce (separate data set)
  - Hive and Pig
- Further process in Azure Data Lake Analytics/U-SQL
- SQL Server PolyBase and Clustered Columnstore Indexes
- Apache Spark

Our Agenda: Business Intelligence

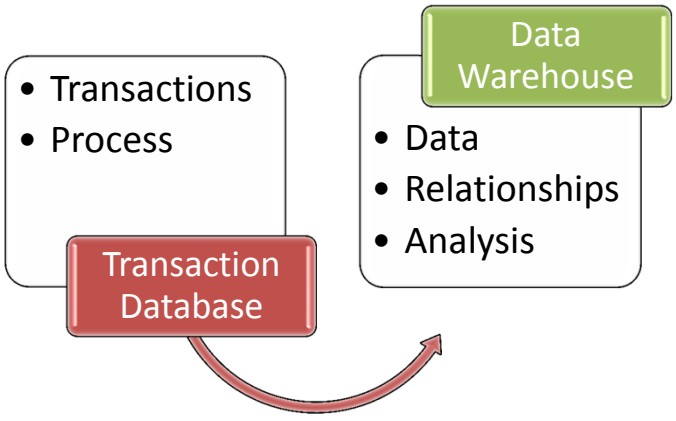
- Power BI – deeper dive
- Analysis Services
- Azure Analysis Services

Data Warehouse Concepts

Business Intelligence

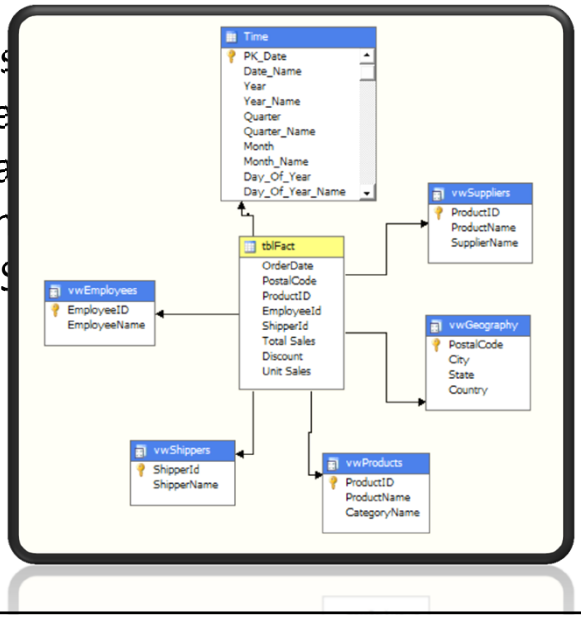


Preparing For Business Intelligence



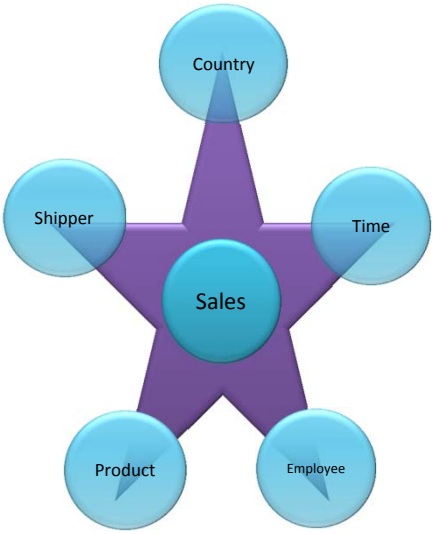
Dimensional Model

- Meas
- Dime
- Hiera
- Grain
- Star S



Star Schemas

- Physical data model
- Central fact table
- Multiple dimension tables
  - Used to constrain fact table queries



Example Data Request

Calendar Year For Country = USA and Calendar Year = 1996

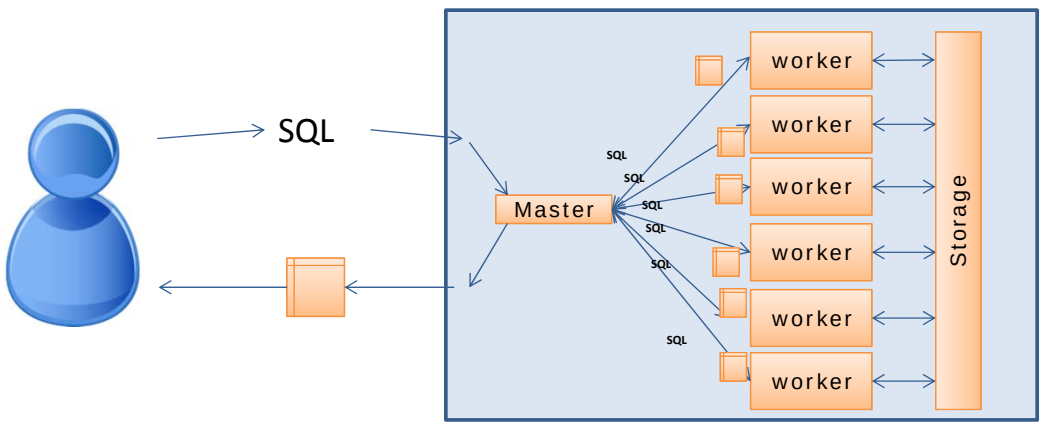
Data Warehouse Query

	STATE	Month_Name	(No column name)
1	NM	August 1996	3343.60
2	WY	August 1996	48.00
3	ID	December 1996	6038.60
4	OR	December 1996	780.00
5	WY	December 1996	3391.20
6	NM	July 1996	624.80
7	WA	July 1996	676.00
8	NM	November 1996	1731.20
9	WA	November 1996	2856.00
10	WY	November 1996	141.60
11	AK	October 1996	934.50

What is MPP?

- Massively Parallel Processing
  - A cluster of individual RDBMS instances (worker nodes)
  - One master node, in front
    - Takes query, delegates parts of it to different worker nodes
    - Combines worker nodes' results, returns as single result set
  - Thus, appears as a single RDBMS
    - Send it one query, get back one result set
    - But query is highly parallelized, so it's fast
    - Perfect for data warehouses
  - Bears some resemblance to MapReduce
  - Examples include Teradata, HP Vertica, IBM Netezza, Pivotal Greenplum

What is MPP?



Column-Oriented Stores

- Imagine, instead of:

Employee ID	Age	Income
1	43	90000
2	38	100000
3	35	100000

- You have:

Employee ID	1	2	3
Age	43	38	35
Income	90000	100000	100000

- Perf: values you wish to aggregate are adjacent
- Efficiency: great compression from identical or nearly-identical values in proximity
- Fast aggregation and high compression means huge volumes of data can be stored and processed, in RAM

MPP + Columnar

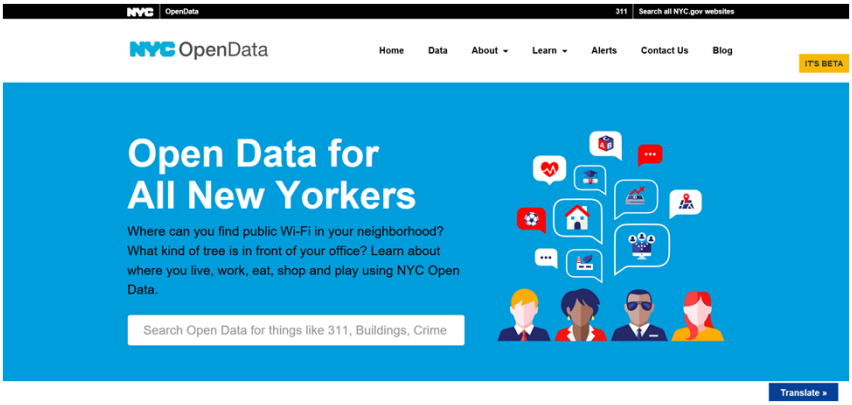
- Together, these greatly accelerate DW performance.
- Far superior to a scaled-up SQL Server Enterprise box
- Most DW platforms combine these two technologies
- Add vector processing and it's a big deal

MPP at Microsoft?

- Yes, resulting from 2008 acquisition of DATAllegro
  - Open source MPP based on Ingres, written in Java, running on Linux
- Project Madison
  - Apply DATAllegro architecture using SQL Server, .NET and Windows
  - Released as SQL Server Parallel Data Warehouse (PDW)
  - Now called Analytics Platform System (APS)

Our Data Set

NYC Open Data



- https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9

NYC Open Data, Power BI Preview

demo

Hadoop and HDInsight

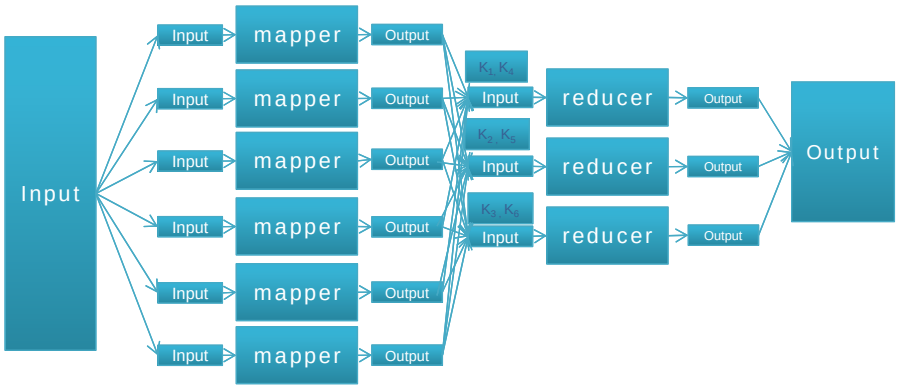
What is Big Data?

- 100s of TB into PB and higher
- Involving data from: financial data, sensors, web logs, social media, etc.
- Parallel processing often involved
  - Hadoop is emblematic, but other technologies are Big Data too
- Processing of data sets too large for transactional databases
  - Analyzing *interactions*, rather than *transactions*
  - The three V's: Volume, Velocity, Variety
- Big Data tech sometimes imposed on small data problems

What's MapReduce?

- "Big" data input accepted in file form
- Data is partitioned and sent to *mappers* (nodes in cluster)
- Mappers pre-process data into KV pairs, then all output for (a) given key(s) goes to a *reducer*
- Reducers aggregate; one line of output per unique key, with one value
- Map and Reduce code natively written as Java functions

MapReduce, in a Diagram



Apache Tez

- Key component added to Hadoop 2.0
- It's a directed acyclic graph (DAG) execution engine that runs on top of YARN (Hadoop 2.0's resource manager)
- Hive and Pig can both run on it
- Shunned by Cloudera

HDFS

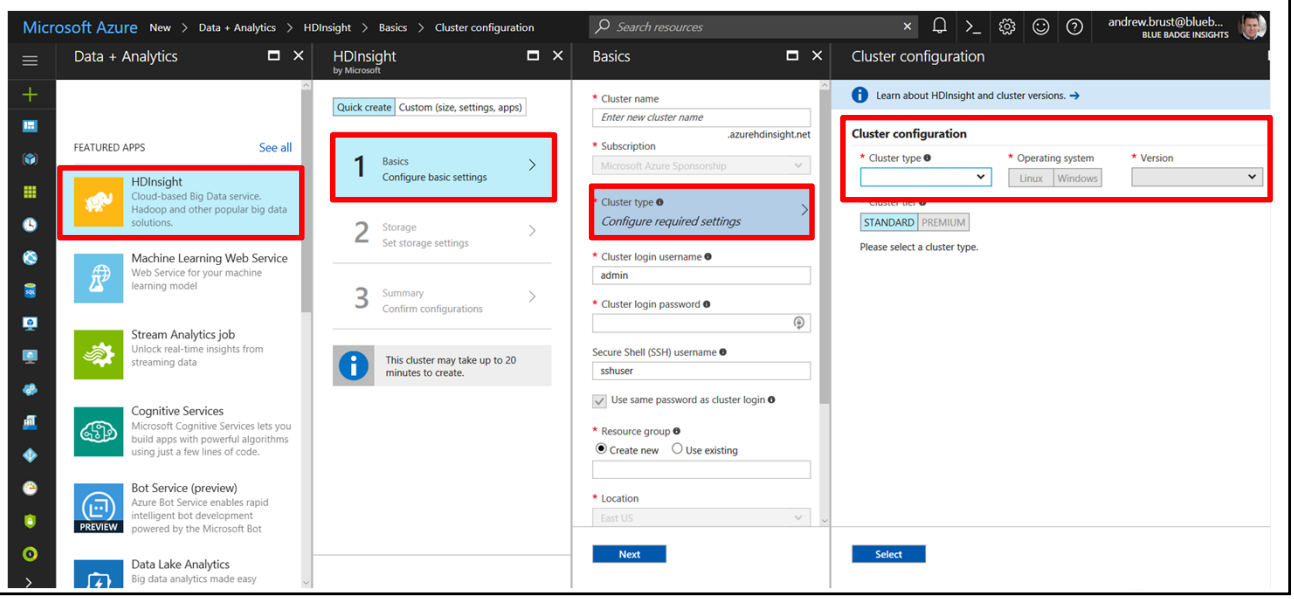
- File system whose data gets distributed over commodity drives on commodity servers
- Data is replicated
- If one box goes down, no data lost
  - "Shared Nothing"
  - Except the name node
- BUT: Immutable
  - Files can only be written to once
  - So updates require drop + re-write (slow)
  - You can append though
  - Like a DVD/CD-ROM

HDINSIGHT

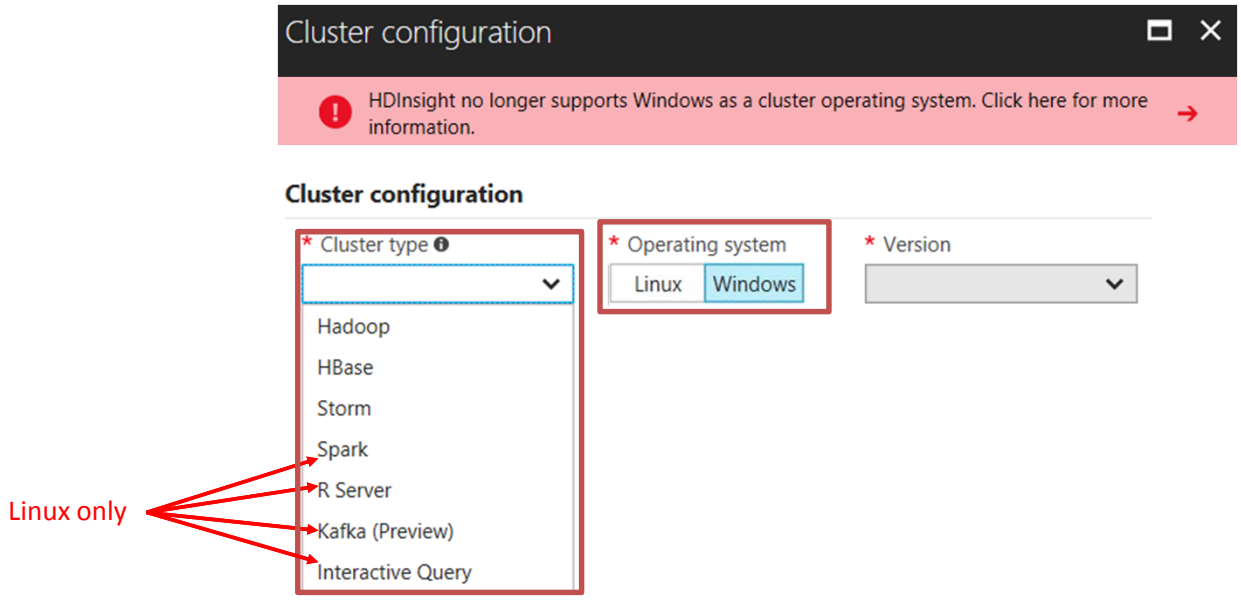
Microsoft HDInsight

- Developed with Hortonworks and incorporates Hortonworks Data Platform (HDP) *for Windows*
- Windows Azure HDInsight and Microsoft HDInsight Server
  - Single node preview runs on Windows client
  - Also Hortonworks HDP for Windows
  - Also HDInsight with Analytics Platform System
- Includes ODBC Drivers for Hive
- All contributed back to open source Apache project

Azure HDInsight Provisioning



Azure HDInsight Provisioning



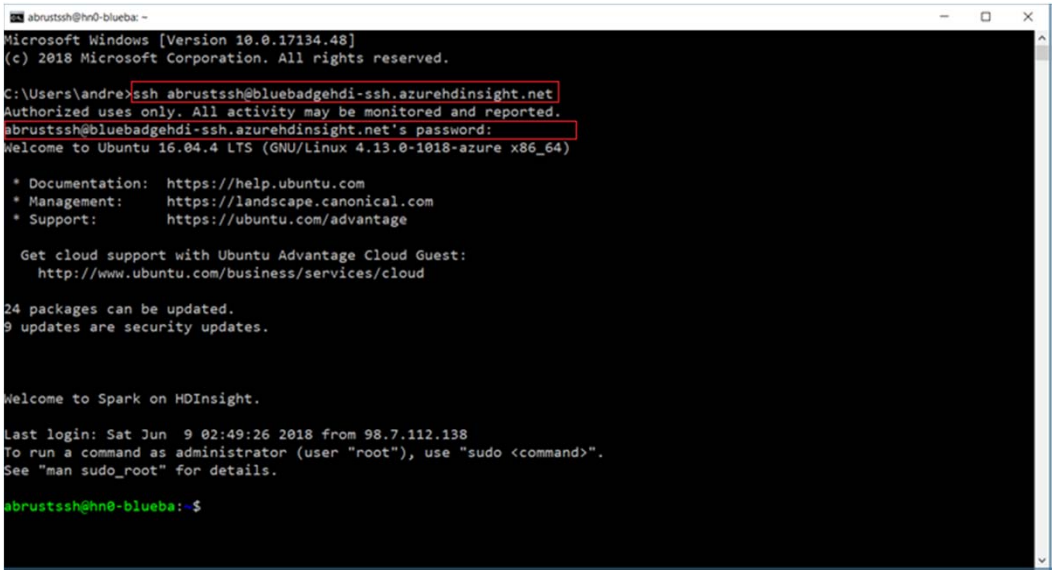
Working with HDInsight

- Apache Ambari
  - For Hive queries and cluster monitoring
- Access via PowerShell and HDInsight cmdlets
  - Need to install PowerShell for Microsoft Azure
  - **Run you PowerShell client as administrator**
- SSH into head node
  - Use PuTTY or new SSH client on Windows 10
  - To ***username*@*clustername***-ssh.azurehdinsight.net

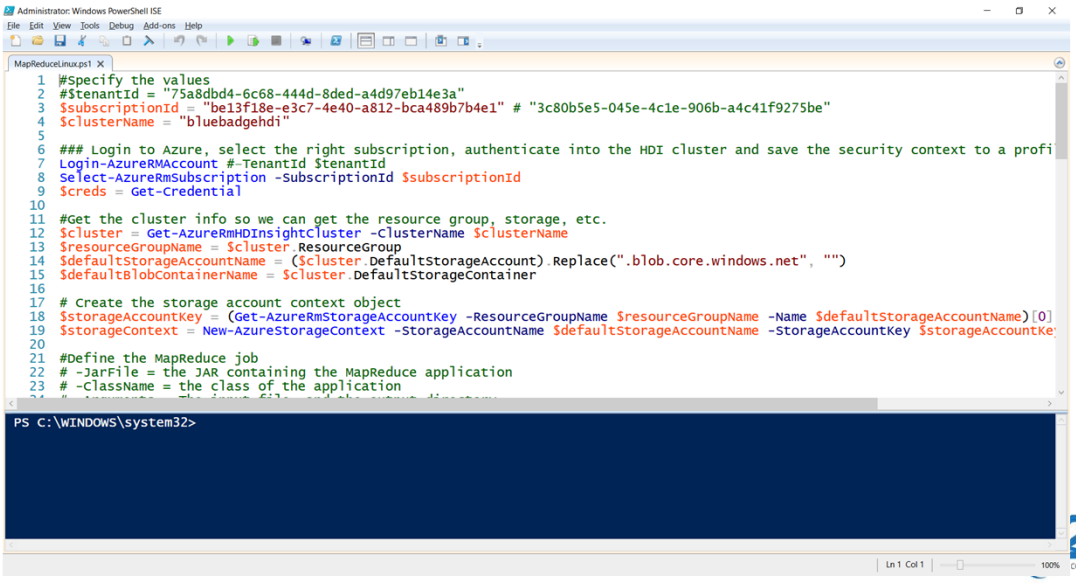
Submitting, Running and Monitoring Jobs

- Upload a JAR
- Run at command line (PowerShell or SSH Command line) passing JAR name and params

Clients Options: Command Line via SSH



Clients Options: PowerShell



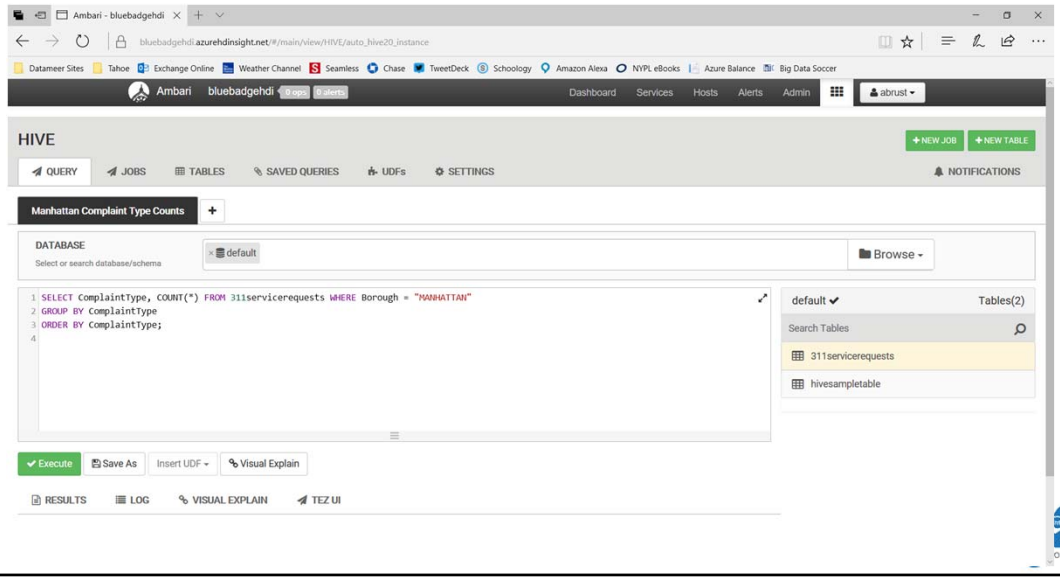
WordCount Code; Running MapReduce Jobs

demo

Hive

- Originally: a SQL abstraction over MapReduce
- Has evolved to run on Tez
  - Along with Project Stinger, achieved 100x improvement over Hive on MR
- Spark SQL is a cousin of Hive
  - More later
- Hive advanced features
  - External tables
  - UDFs

Clients Options: Ambari



Hive

demo

Hive LLAP

- Stands for “Live Long and Process”
- A Hive-on-Tez variant that uses caching heavily for enhanced performance
- In preview on HDInsight as “Interactive Hive” cluster type

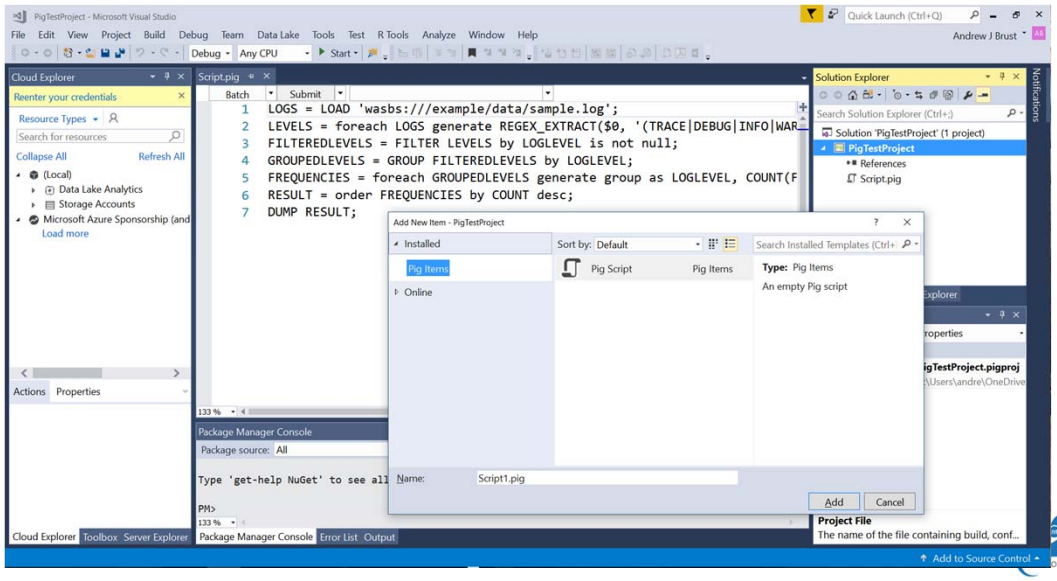
Impala, Hive on Spark

- Hive-compatible MPP engine that works directly against HDFS
- Apache Impala was originally a Cloudera project
- Hive-on-Spark is a Cloudera-led enhancement to Hive that has it run on Spark instead of MR or Tez
- Neither one common on non-Cloudera Hadoop clusters

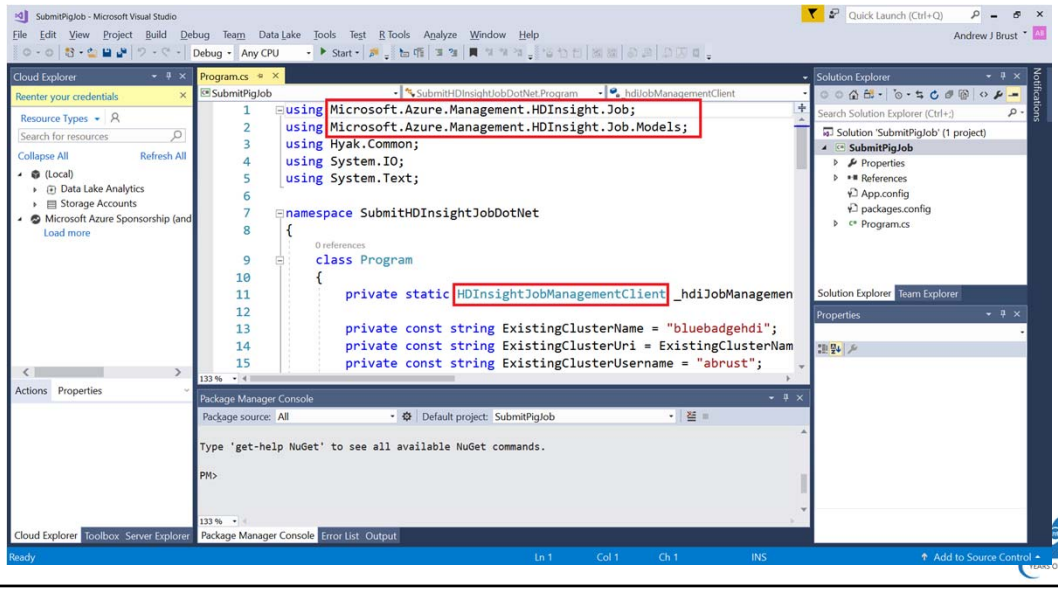
Pig

- Also a programming language abstraction over MapReduce
- Language is called Pig Latin
- Can be used for interactively and for queries
- More often used for data transformation, from scripts

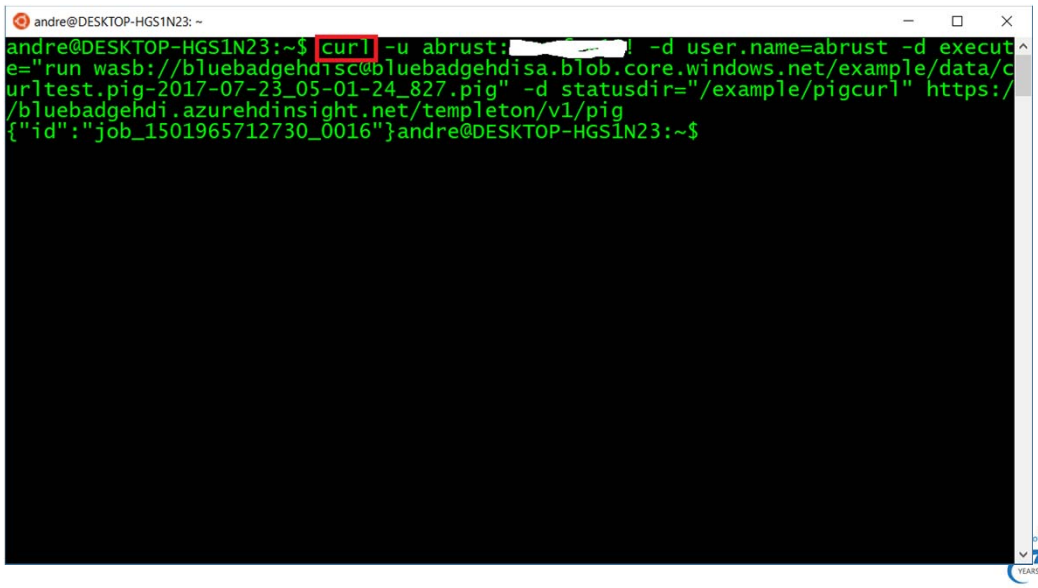
Clients Options: Azure Data Lake Tools in VS



Clients Options: .NET SDK for Hadoop



Clients Options: cURL



Pig

demo

Azure Data Lake

Azure Data Lake Store

- Based on Azure BLOB storage, but…
- No file size limits
- Resources added as needed for scale
- WebHDFS compatible
- Certain HDInsight cluster types can use it instead of Blob storage
- Third parties beginning to support

Azure Data Lake Analytics

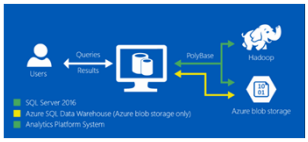
- Lets you do big data analytics on data stored in ADLS
- ADLA jobs run on YARN/HADOOP
- Jobs are run on-demand; no dedicated HDInsight cluster involved
- Right now, only job type supported is U-SQL…

U-SQL

- U-SQL
  - Work with flat files or create databases
    - DBs allow for indexing and partitioning
  - Looks like T-SQL, but allows inclusion of C# code...either for inline expressions, or for UDFs
  - Allows batch operations on whole sets of files using wildcard patterns.
  - Not a business user tool, but an *excellent* abstraction layer on Hadoop for developers
- As part of Azure Data Lake Analytics
  - Runs Hadoop jobs behind the scene – but server-less/cluster-less
  - Native storage is Azure Data Lake Store, but can access data in Azure Blob storage too

Azure Data Lake Analytics/U-SQL

demo

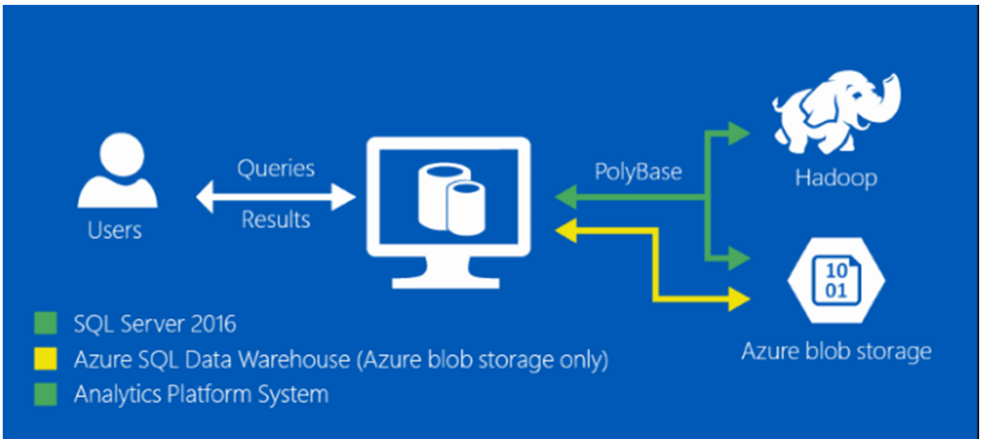


PolyBase

PolyBase

- A “bridging” technology to connect SQL Server to data in Hadoop or Azure Blob Storage
- Makes the Hadoop data look like SQL Server data via “EXTERNAL” tables
- Query as normal; even join with physical tables
- First appeared in Parallel Data Warehouse/APS and Azure SQL DW
- Now included in SQL Server 2016 Enterprise
- Can create physical table with CREATE TABLE…AS SELECT… (CTAS)

PolyBase



Notes

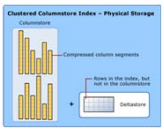
- Data may be moved and processed by SQL Server's engine and optimizer, or may be "pushed down" to Hadoop, or both
- For DW versions of SQL, query is distributed
- Config can be tricky
- Java install is a prerequisite

Relevant T-SQL

- Prepatory:
  - EXEC sp_configure 'hadoop connectivity', *x*
  - RECONFIGURE;
  - CREATE MASTER KEY ENCRYPTION
- Next:
  - CREATE DATABASE SCOPED CREDENTIAL
  - CREATE EXTERNAL DATA SOURCE
  - CREATE EXTERNAL FILE FORMAT
  - CREATE EXTERNAL TABLE

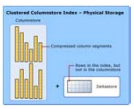
PolyBase

demo



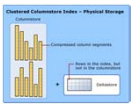
Columnstore Indexes

In Analytics…



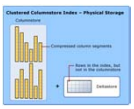
- Geared to reporting and visualization
  - Read frequently, write seldom
- Table scans are expected
- Aggregation (think GROUP BY) is *de riguer*
- Extensive normalization is bad
- You only care about values in a small set of columns…maybe even just one
  - The rest are used with WHERE and HAVING, to filter
- Tables that track location and time are common

A History of Columnstore Indexes



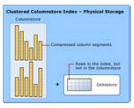
- SQL Server 2012: Nonclustered Columnstore Indexes (NCCIs) added to product
  - Read only
- SQL Server 2014: Clustered Columnstore Indexes (CCIs) added
  - Read/Write
- SQL Server 2016: Numerous enhancements to CCIs

Vector Processing



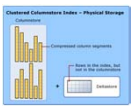
- Intel x86 CPUs have, since supported “single instruction multiple data” (SIMD) operations since the 1990s
- These process data in parallel, handling multiple data points simultaneously
- This is called vector processing
- SQL Server NCCIs and CCIs can take advantage of it

Useful Applications



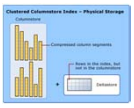
- Data Warehouse/Data Mart scenarios
- In combination with DirectQuery feature in SSAS Tabular and Power BI
- In combination with R Services

Vector Processing and "Batch" Mode



- SQL Server can burst into a vector processing mode
- Instead of iterating through rowsets, one row at a time, it can handle rows in batches
- So it's called "batch mode" and it's *fast*
  - (Not to be confused with batch processing, which can be slow)

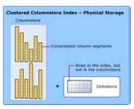
Sanity Check



Columnstore Index Scan (NonClustered)

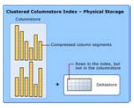
Scan a columnstore index, entirely or only a range.

Physical Operation	Columnstore Index Scan
Logical Operation	Index Scan
Actual Execution Mode	Batch ✓
Estimated Execution Mode	Batch ✓
Storage	ColumnStore



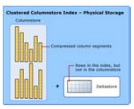
Making Sure it Works

- CIs are fastest when Batch mode kicks in
  - Difference can be negligible otherwise
  - Check Query Plan to make sure
- And meet the prerequisites…



Prerequisites

1. More than one CPU core	2. Maximum Degree of Parallelism (MDOP) set to 0	*Lots* of data
• (Careful on those VMs!)	• Or a value between 2 and 64, if you want to limit it • Use SSMS server properties sheet or sp_configure and RECONFIGURE	• Millions of rows, or don't bother



Columnstore Indexes

demo

Apache Spark

Spark

- Wildly popular open source project, focuses on distributed in-memory processing versus on-disk
- Can use it independently of Hadoop, but most people use it with Hadoop/HDFS
- Very popular component: Spark SQL
  - Allows HiveQL queries against Spark (Power BI can use this)
- Also: Spark Streaming, MLlib, GraphX
- Spark now supported on HDInsight

Spark on HDInsight

- HDInsight Spark clusters include the Jupyter and Zeppelin “notebook” user interface
- Allow interspersal of text, code, and code output, including visualizations
- Supports Python (PySpark) and Scala
- Includes *very* helpful tutorial notebooks

Jupyter Notebooks

- Notebooks combine code, text and data visualization capabilities
- Text and code “cells” are interspersed. Code can be executed in place.
- Jupyter originally called iPython and hosted only Python code; now hosts numerous langauges
- On HDInsight, Jupyter Notebooks can host Python, Scala and R code, running against Spark
- See also: Azure Notebooks

Apache Spark

demo

The Hadoop Stack

Security, governance

Stream processing, analytics

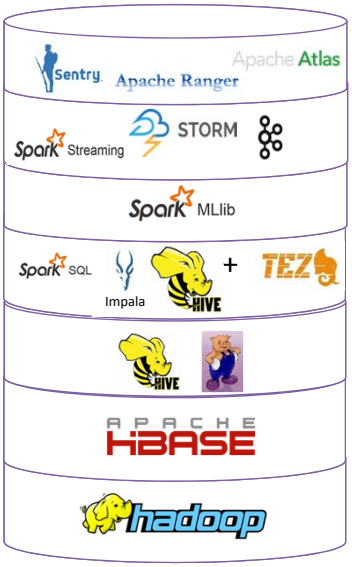
Machine Learning

Interactive SQL

Query: HiveQL and Pig Latin

Database (NoSQL)

HDFS, YARN



Power BI

Power BI

- Based on same columnar, in-memory BI engine as SQL Server Analysis Services Tabular mode
- Free Desktop and Mobile apps
- For individual users, 2 cloud subscription levels: Basic (free) and Pro ($10/month/seat)
- Easy to use, extensible, embeddable, connects to a huge array of conventional and cloud data sources
  - Growing DirectQuery support
- Highly integrated across Microsoft stack

Power BI Ingredients

Power BI Desktop

- Windows Desktop Application
- Has a "main window," akin to the Excel Power View Add-In, for report authoring and some data modeling
  - Report view
  - Data view
  - Relationships view
- Has a Query Editor window, akin to the Excel Power Query Add-In, for data import and transformation
- Can save files (.pbix) locally and an publish them to powerbi.com

Power BI Query Editor: Overview

- Launched with Get Data option (from ribbon or splash page)
- Re-entered using Edit Queries ribbon button
- Use it to import and shape data
- Use Close & Load ribbon button when done
- Try not to confuse this window with the data view in the main window

Power BI Reports Overview

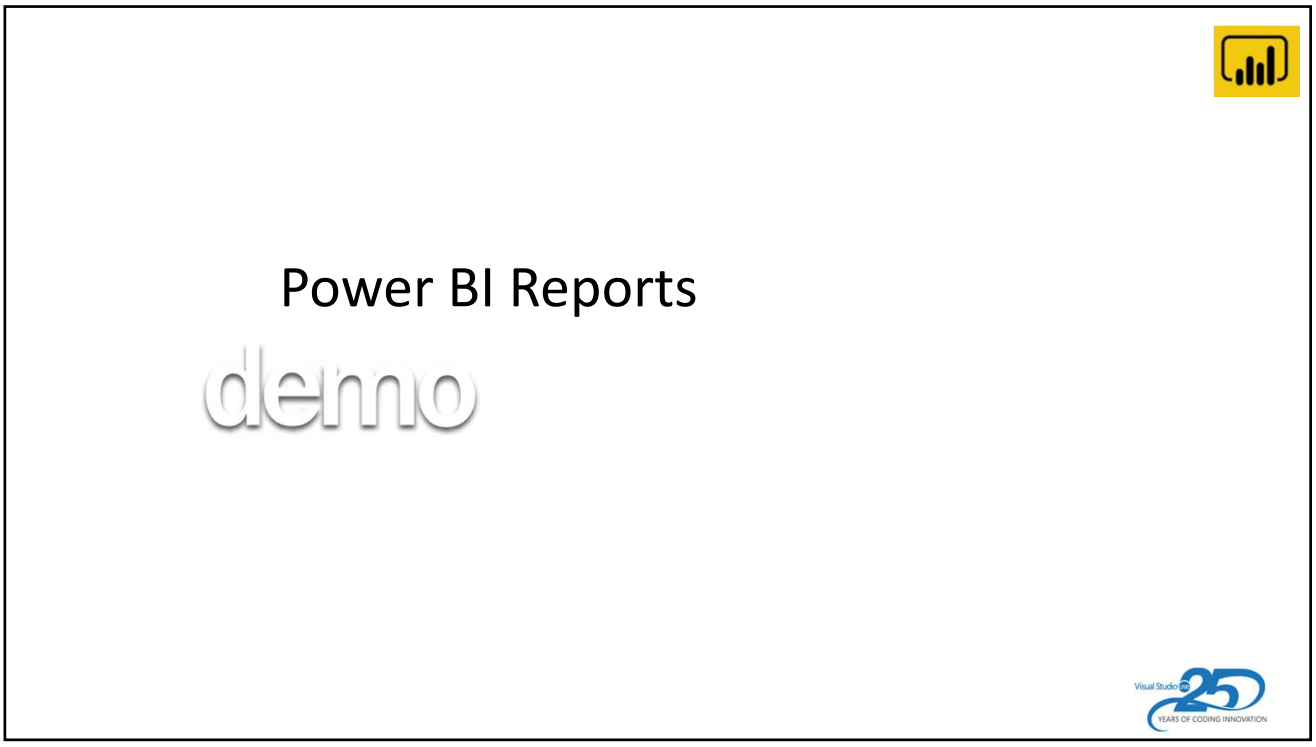
- Data exploration and visualization client
- Visualizations work as filters, too
- Design and view experiences are unified

On-Premises Gateway

- Permits import and scheduled refresh of on-prem data in cloud copy of report
- Personal mode:
  - Runs as app for single user
- Enterprise mode:
  - Runs as service for multiple users
  - "DirectQuery" supported for numerous data sources
  - "Live Connection" supported for SSAS (Tabular or MD)
  - Supports PowerApps, Azure Logic Apps, Microsoft Flow and Azure Analysis Services (preview)

The Views

- Report: the report designer/viewer
- Data: where you can model the data
  - Rename/delete/hide columns and tables
  - Sort by a column (ascending or descending)
  - Add DAX measures and calculated columns
  - Set data types and categories
- Relationships – Where you can view and edit relationships
  - But you must create them with the Manage Relationships dialog



Power BI Cloud Service

- Authoring and consumption tool
- Can create three things
  - Dataset
  - Report
  - Dashboard
- Publish report from PBI Desktop, get link to cloud version
- Also available: “Quick Insights”

Dashboards

- A collection of “pinned” visualizations from existing Power View reports
- Pin entire reports, too!
  - Single visualizations are not interactive
- What you can pin:
  - Web content, images, video, text boxes
  - Visualizations from Quick Insights
  - Excel spreadsheet assets
  - SQL Server Reporting Services assets
  - Camera photos (via iPhone App)

Power BI Service

demo

Q&A

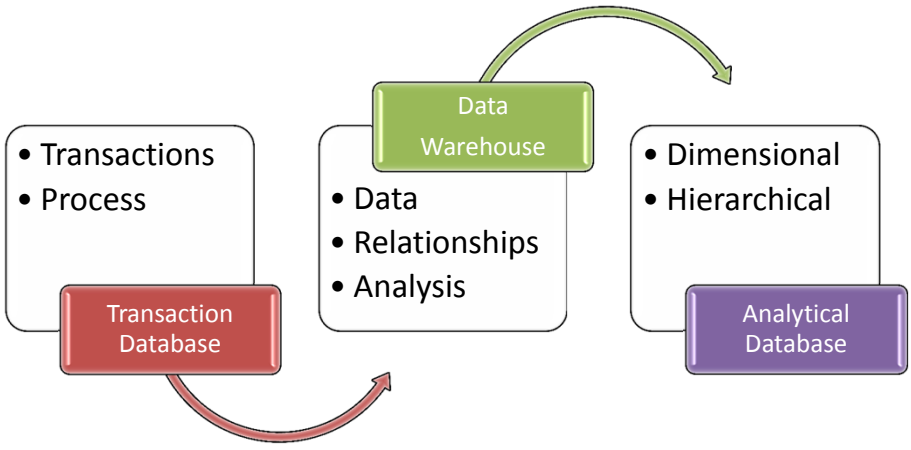
- Natural language query interface to data in underlying model
- Available at top of dashboard
  - Now available in reports too
  - And as authoring tool
- Generates visualization as you type
- Visualization is pin-able

Power BI Premium

- New subscription level for Enterprise use:
  - Unlimited consumption users; Professional subscription still required for each authoring user
  - Dedicated infrastructure; paid for by the number and type of server nodes
  - Starts at $4,995/month for P1 node with 8 cores, 25GB RAM
- Includes on-premises capabilities:
  - Power BI Report Server: Actually a superset of SQL Server Reporting Services. (Available w/o power BI subscription for SQL EE+SA customers.)
  - Licensed for same number of cores included in cloud subscription
  - Reports only; no dashboards

Analysis Services

Data Migration

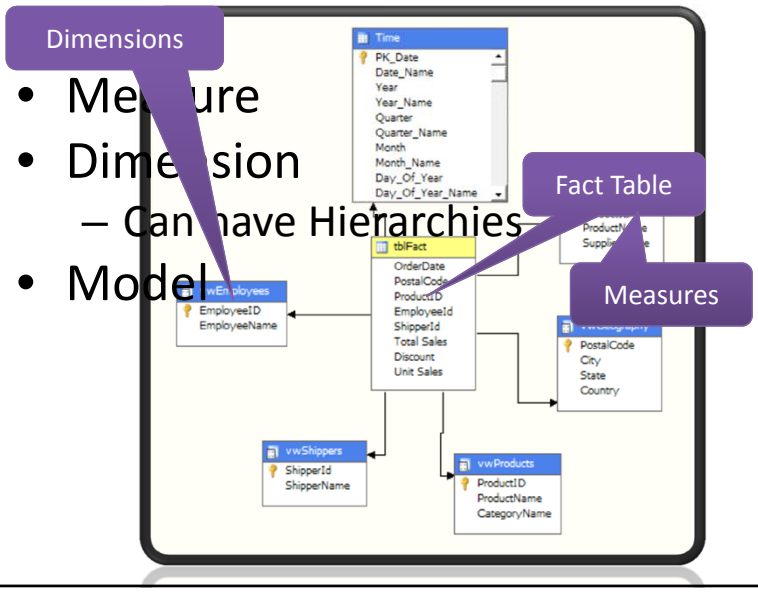


SQL Server Analysis Services

- Built for analysis
- Included with SQL Server Standard, Enterprise
- And you can use the Microsoft stack that you know and love

From Data Warehouse to OLAP

- Measure
- Dimension
  - Can have Hierarchies
- Model



Analysis Services Modes

- **Multidimensional** or **Tabular**
- Tabular is newer, same tech as Excel/PowerPivot data models and Power BI
- Lots of investment in Tabular in SSAS 2016
- We'll look at Tabular today

Analysis Services Tabular Mode

- SSAS Tabular Mode uses a columnar storage engine in place of a multidimensional one
- Must choose mode for SSAS instance at install time
- Can have default instance with one, named instance with the other
- Can create an SSAS Tabular database project by importing an Excel workbook with PowerPivot model
- SSAS tabular models support partitions, roles, translations, display folders

Calculated Columns and DAX

- Formula-based columns may be created
- Formula syntax is called DAX (Data Analysis eXpressions).
  - Not to be confused with MDX or DMX. Or DACs.
- DAX expressions are similar to Excel formulas
  - Work with tables and columns; similar to, but distinct from, worksheets and their columns (and rows)
- =FUNC('table name'[column name])
- =FUNCX('table name', <filter expression>)
- FILTER(Resellers,[ProductLine] = "Mountain")
- RELATED(Products[EnglishProductName])
- DAX expressions can be heavily nested

Analysis Services

demo

Azure Analysis Services

- In preview now
- Platform as a Service offering for Analsis Services Tabular
- Supports Analysis Services 2017 features
- Compatible with Excel, Power BI
- Can use Visual Studio Analysis Services Projects tooling or new browser based tools
- Can use same on-prem gateway as Power BI for refresh of models from on-prem data sources

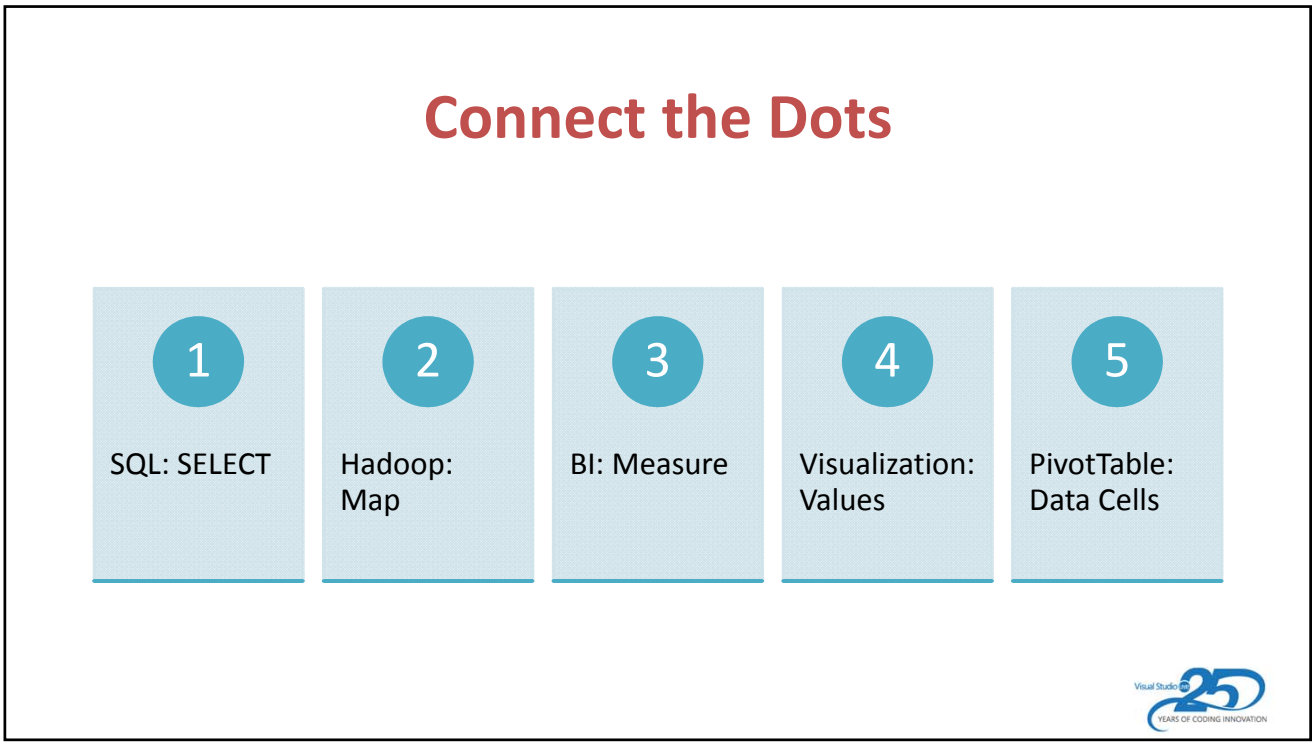
Azure Analysis Services

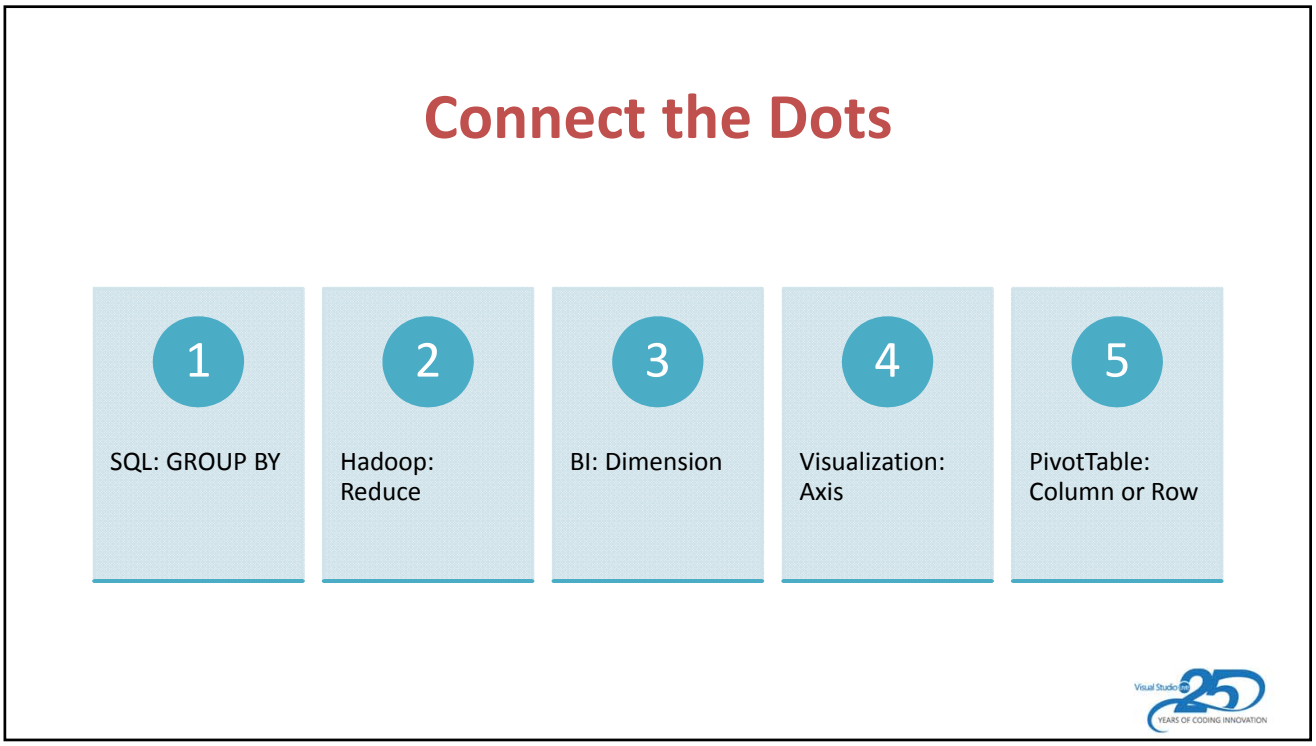
demo

Closing Thoughts

How Do BI and Big Data Relate?

- At the root of each lies the idea of grouping and aggregating
- The Reduce step in MapReduce is all about that
- In the DW/BI side, so is defining dimensions and drilling down by them
- And there is a pretty strong mapping between dimensions/reducer groupings on the one hand and machine learning features on the other
- Think of it this way…


Connect the Dots
1
SQL: SELECT
2
Hadoop: Map
3
BI: Measure
4
Visualization: Values
5
PivotTable: Data Cells


Connect the Dots
1
SQL: GROUP BY
2
Hadoop: Reduce
3
BI: Dimension
4
Visualization: Axis
5
PivotTable: Column or Row

Integration Matrix

	SQL Server RDBMS	HDInsight	Power BI	Analysis Services	Excel	Reporting Services	R	U-SQL	.NET
SQL Server RDBMS		●	●	●	●	●	●		●
HDInsight	●		●	●	●		●	●	●
Power BI	●	●		●	●		●		●
Analysis Services	●	●	●		●	●			●
Excel	●	●	●	●					●
Reporting Services	●	●		●			●		●
R	●	●	●					●	
U-SQL		●					●		●
.NET	●	●	●	●	●	●		●	

PART III

Beyond Relational

2016

JSON

2016

Built-In JSON Support

- 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

2016

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

2016

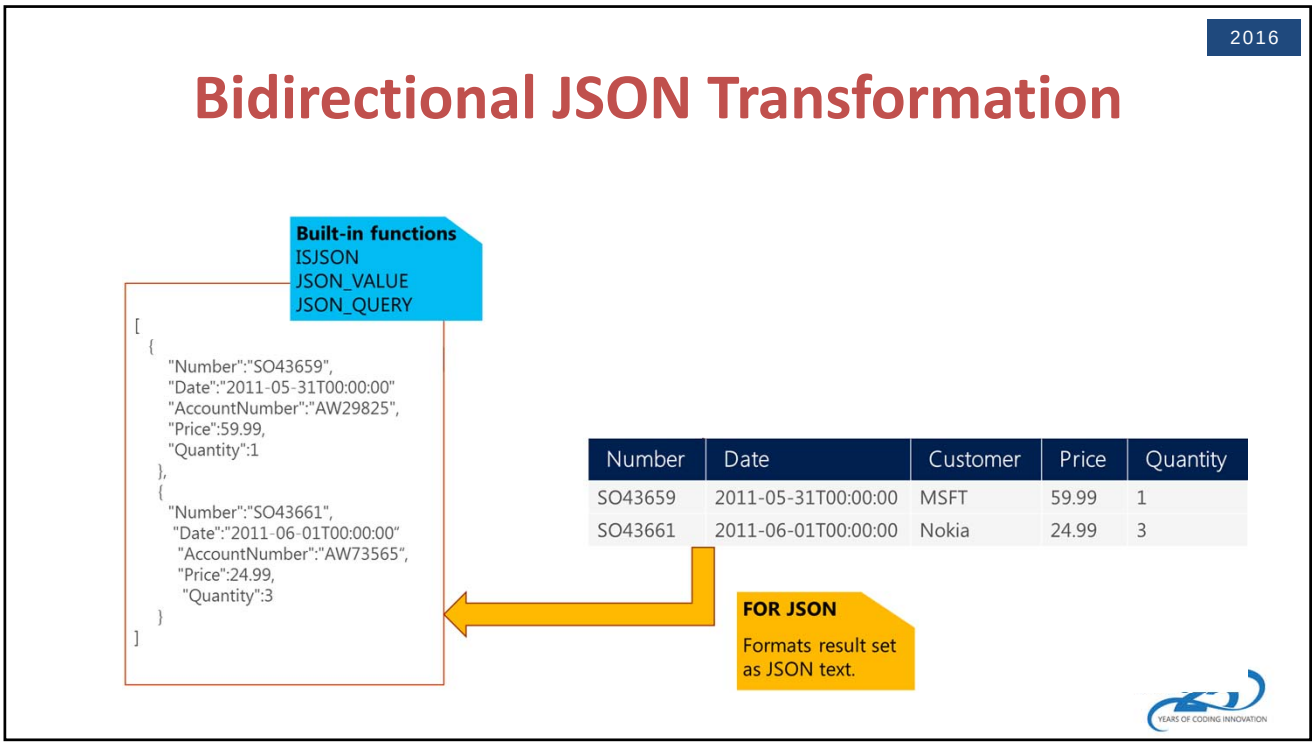
Bidirectional JSON Transformation

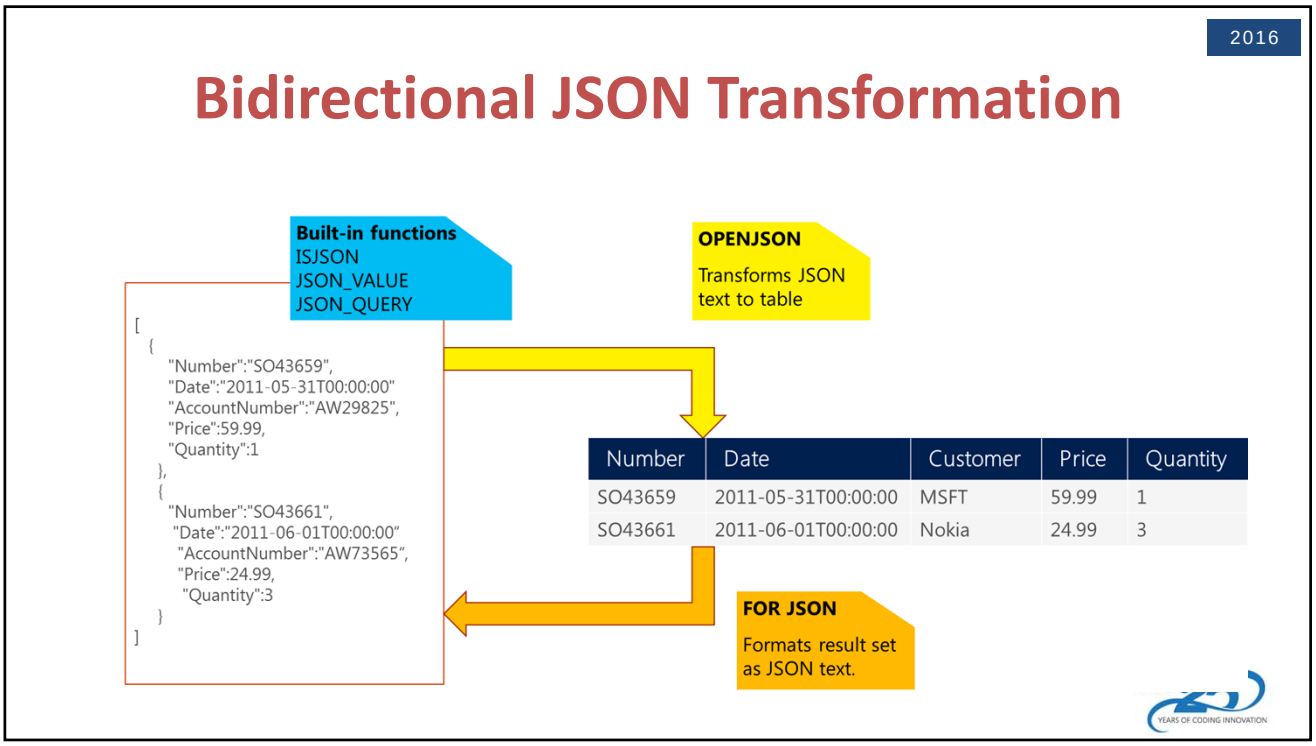
```
[
  {
    "Number":"SO43659",
    "Date":"2011-05-31T00:00:00"
    "AccountNumber":"AW29825",
    "Price":59.99,
    "Quantity":1
  },
  {
    "Number":"SO43661",
     "Date":"2011-06-01T00:00:00"
      "AccountNumber":"AW73565",
      "Price":24.99,
       "Quantity":3
  }
]
```

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.


2016
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.


2016
Bidirectional JSON Transformation
Built-in functions
ISJSON
JSON_VALUE
JSON_QUERY
OPENJSON
Transforms JSON text to table
[
{
"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.

2016

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

2016

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

2016

Generating JSON

demo

2016

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*

2016

JSON Path Expressions

- Reference JSON properties using a JavaScript-like syntax

Syntax	Description
`$`	References the entire JSON object
`$.property1`	References a top-level property in the JSON object
`$[5]`	References the sixth element in the JSON array
`$.property1.property2 .array1[5].property3 .array2[15].property4`	References a complex nested property in the JSON object

2016

JSON Query Example

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

2016

Storing and Querying JSON

demo

2016

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

2016

Shredding JSON

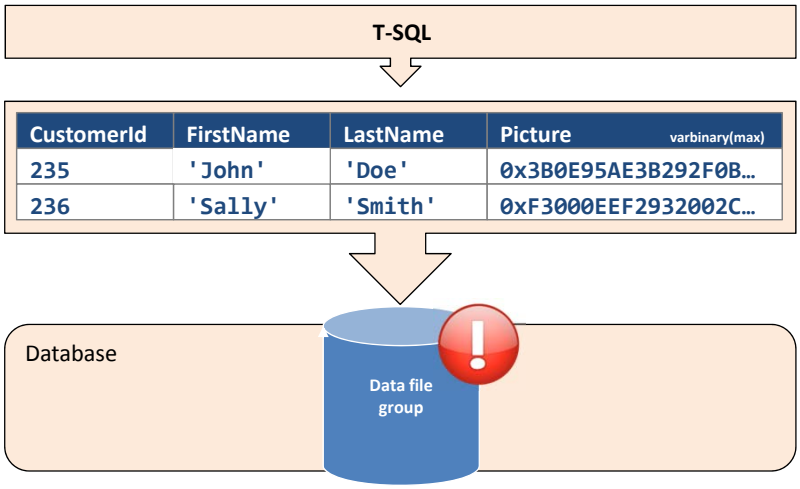
demo

2008+

FILESTREAM

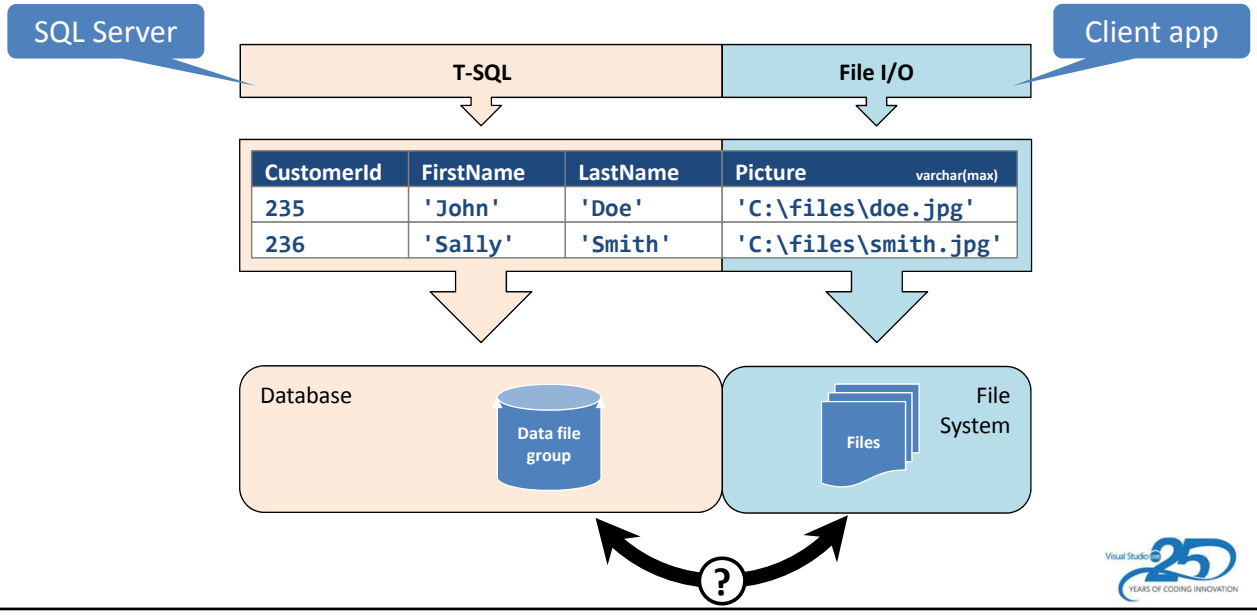
2008+

BLOBs in the Database



2008+

BLOBs Outside the Database



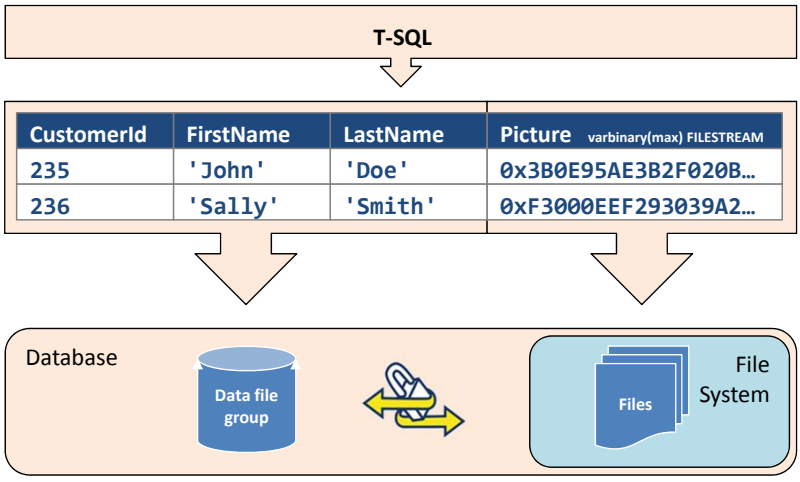
2008+

BLOBs Using FILESTREAM

- Transparently store varbinary(max) data in the file system
  - Declare column as “varbinary(max) FILESTREAM”
  - File system is optimized for storing and streaming BLOBs
- Integrated management
  - BLOBs are logically part of the database (backup, restore, etc.)
  - But they are stored physically separate as a file group mapped to the file system
- Simplified programming
  - Just use T-SQL, or the streaming API
  - SQL Server transparently links rows in relational tables to BLOBs in the file system
- Transactional
  - SQL Server integrates with the NTFS file system
  - Database transactions wrap NTFS transactions

2008+

BLOBs Using FILESTREAM (T-SQL)



2008+

Enabling FILESTREAM

- FILESTREAM is disabled by default
  - Can be enabled for T-SQL or T-SQL + streaming API access
- To use FILESTREAM, it must be enabled twice
  - By the Windows administrator
    - Installs a file system filter driver
  - By the SQL Server administrator
- Set the same access level each time
  - Windows and SQL admins must agree!

2008+

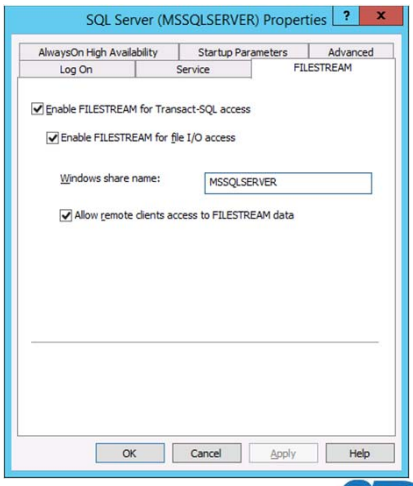
Determining the Access Level

- Disabled
  - FILESTREAM is not enabled
- T-SQL
  - FILESTREAM is enabled for T-SQL access
  - Complete abstraction
- T-SQL + file system I/O (local)
  - FILESTREAM is enabled for both T-SQL access and streamed file system access
  - Streaming API access available only to client applications running on the local server
- T-SQL + file system I/O (remote)
  - Streaming API access available to client applications running anywhere on the network

2008+

Enabling FILESTREAM (Windows)

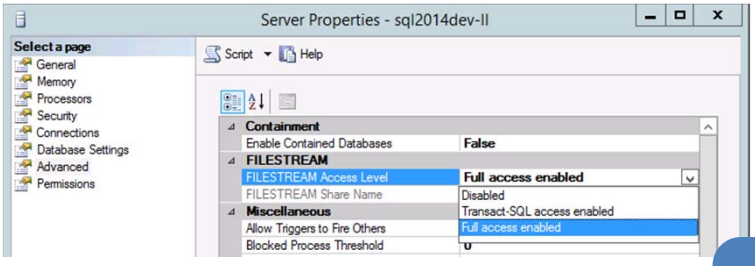
- Enable it either:
  - During setup
  - With SQL Server Configuration Manager
- Set to desired access level
  - Disabled
  - T-SQL only
  - T-SQL + file I/O
    - local only
    - remote
- Can't script with T-SQL



2008+

Enabling FILESTREAM (SQL Server)

- Enable it in SSMS:
  - Server Properties dialog



- Or, enable it with T-SQL
  - `EXEC sp_configure filestream_access_level, n RECONFIGURE`

n =
0: disabled
1: T-SQL only
2: T-SQL + file I/O

2008+

Preparing the Database for FILESTREAM

- Create a FILESTREAM filegroup
  - For a new database, using CREATE DATABASE
  - For an existing database, using ALTER DATABASE
- Add a FILESTREAM container to the filegroup
  - Point to a path on the NTFS file system
  - Contents of each varbinary(max) FILESTREAM column will be stored as a distinct file in this path
- Use multiple filegroups and containers
  - Load balance the BLOB load across multiple disks
  - Can assign different tables to different filegroups

2008+

Creating a FILESTREAM-enabled Database

```
CREATE DATABASE PhotoLibrary
 ON P
  (NAME = PhotoLibrary_data,
   FILENAME = 'C:\DB\PhotoLibrary_data.mdf'),
FILE
 (NAME = PhotoLibrary_photos,
  FILENAME = 'C:\DB\Photos')
 LOG
  (NAME = PhotoLibrary_log,
   FILENAME = 'C:\DB\PhotoLibrary_log.ldf')
```

Must exist

Must not exist

Must exist

Must not exist

Must exist

Must not exist

2008+

Creating FILESTREAM Columns

- Define BLOB columns as “varbinary(max) FILESTREAM”
  - Multiple BLOB columns are permitted per table
- Table requires ROWGUIDCOL column
  - Attribute applied to a uniqueidentifier (GUID) column
  - Must be primary key or have unique constraint
  - Cannot be NULL
- Access using standard T-SQL
  - Back-end NTFS file system storage is completely transparent
  - Inline short BLOBs
  - Import from files using OPENROWSET with SINGLE_BLOB option

2008+

Creating FILESTREAM Columns

```
CREATE TABLE Product(
  ProductId int IDENTITY PRIMARY KEY,
  BlobId uniqueidentifier ROWGUIDCOL NOT NULL UNIQUE
  Summary varchar(max),
  Photo varbinary(max) FILESTREAM)
```

Required ID

BLOB

2008+

Getting Started with FILESTREAM

demo

2008+

Storage vs. Access

- FILESTREAM solves the *storage* scalability problem
  - But what about *access*?
- T-SQL works
  - But is it optimal?

```
SELECT * FROM PhotoAlbum
```

Results | Messages

	PhotoId	RowId	PhotoDescription	Photo
1	1	FB9501B0-E580-E411-80C0-080027729686	Text file	0x31303A333020504D2031322F31302F32303134
2	2	6E12262E-E680-E411-80C0-080027729686	Document icon	0x4749463839610C000E00B30000FFFFFFC6DEC6C0C0C000...
3	3	92629D68-E680-E411-80C0-080027729686	Mountains	0xFFD8FFE000104A4649460001020100480048000FFED0A9...

- SQL Server needs to perform file I/O
  - Must process entire file contents in memory

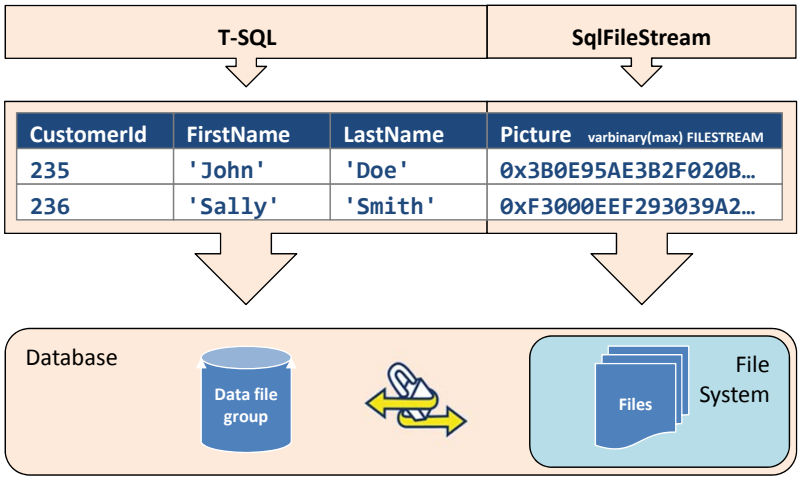
2008+

Introducing SqlFileStream

- Build a .NET streaming client using SqlFileStream
  - Wraps OpenSqlFilestream SQL Server native client API
- Inherits from System.IO.Stream
  - Use standard .NET stream coding patterns
- Included in System.Data.dll
  - No additional references needed
- Use T-SQL to read/write all *non*-BLOB data
  - Use SqlFileStream to read/write all BLOB data
  - Safe and direct streaming access to the SQL Server-controlled file system
- Requires a database transaction
  - Automatically wraps an NTFS file system transaction

2008+

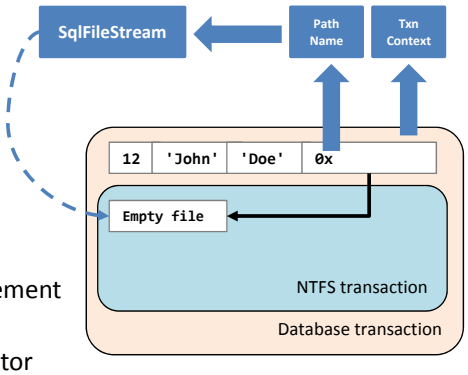
BLOBs Using SqlFileStream



2008+

Storing BLOBs with SqlFileStream

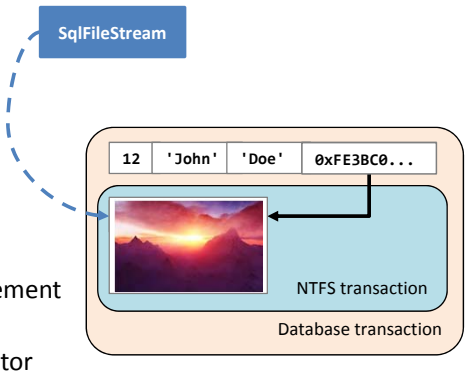
- Start a database transaction
- INSERT new row
  - Store a zero-length binary value for the BLOB column(s)
- Retrieve BLOB path name and transaction context
  - Call **PathName()** function on *varbinary(max) FILESTREAM* column
  - Call **GET_FILESTREAM_TRANSACTION_CONTEXT**
  - Obtain values using an OUTPUT clause in the INSERT statement
- Instantiate SqlFileStream object
  - Pass BLOB path name and transaction context to constructor



2008+

Storing BLOBs with SqlFileStream

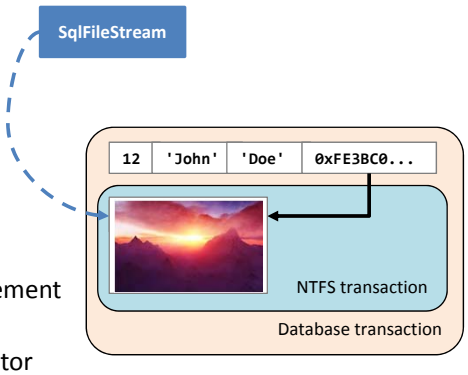
- Start a database transaction
- INSERT new row
  - Store a zero-length binary value for the BLOB column(s)
- Retrieve BLOB path name and transaction context
  - Call **PathName()** function on *varbinary(max) FILESTREAM* column
  - Call **GET_FILESTREAM_TRANSACTION_CONTEXT**
  - Obtain values using an OUTPUT clause in the INSERT statement
- Instantiate SqlFileStream object
  - Pass BLOB path name and transaction context to constructor
- Write to the stream
  - Then close it



2008+

Storing BLOBs with SqlFileStream

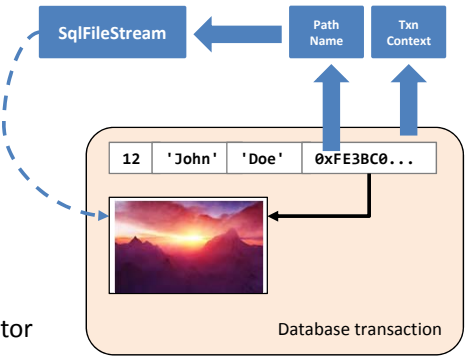
- Start a database transaction
- INSERT new row
  - Store a zero-length binary value for the BLOB column(s)
- Retrieve BLOB path name and transaction context
  - Call **PathName()** function on *varbinary(max) FILESTREAM* column
  - Call **GET_FILESTREAM_TRANSACTION_CONTEXT**
  - Obtain values using an OUTPUT clause in the INSERT statement
- Instantiate SqlFileStream object
  - Pass BLOB path name and transaction context to constructor
- Write to the stream
  - Then close it
- Commit the database transaction
  - Automatically commits the NTFS file system transaction



2008+

Retrieving BLOBs with SqlFileStream

- Start a database transaction
- SELECT existing row
  - Don't include the BLOB column(s)
- Retrieve BLOB path name and transaction context
  - Call **PathName()** function on *varbinary(max) FILESTREAM* column
  - Call **GET_FILESTREAM_TRANSACTION_CONTEXT**
- Instantiate SqlFileStream object
  - Pass BLOB path name and transaction context to constructor
- Read from the stream
  - Then close it
- Commit the database transaction



2008+

Using SqlFileStream

demo

2008+

Improving I/O Scalability

- Distribute FILESTREAM storage by table
  - Each table (or table partition) can be assigned to a different filegroup
    - Use FILESTREAM_ON clause in CREATE TABLE statement
    - Create each filegroup's container on a separate disk volume
- Distribute FILESTREAM storage within a table
  - Partition the table
    - Use FILESTREAM_ON clause to assign a different filegroup to each partition
  - Use multiple containers 2012+
    - Single filegroup for the table will access its containers in round-robin fashion

2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

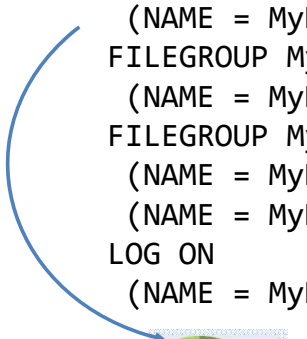
2012+

2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

2012+

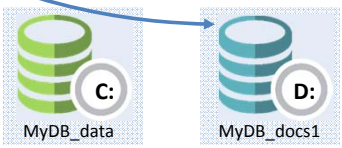


2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

2012+

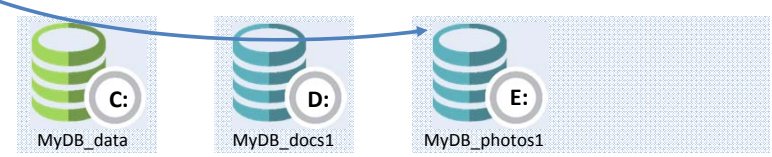


2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

2012+

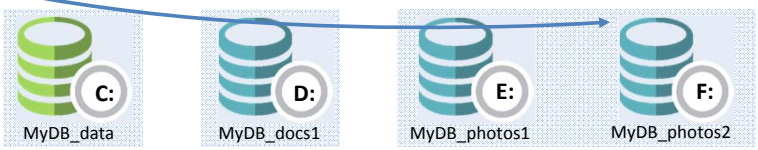


2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

2012+



2008+

Multiple FILESTREAM Filegroups

```
CREATE DATABASE MyDB
 ON PRIMARY
  (NAME = MyDB_data, FILENAME = 'C:\DB\MyDB_data.mdf'),
 FILEGROUP MyDB_docs CONTAINS FILESTREAM DEFAULT
  (NAME = MyDB_docs1, FILENAME = 'D:\DB\MyDB_docs'),
 FILEGROUP MyDB_photos CONTAINS FILESTREAM
  (NAME = MyDB_photos1, FILENAME = 'E:\DB\MyDB_photos'),
  (NAME = MyDB_photos2, FILENAME = 'F:\DB\MyDB_photos'),
 LOG ON
  (NAME = MyDB_log, FILENAME = 'X:\DB\MyDB_log.ldf')
```

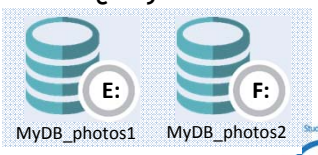
2012+

2008+

Multiple FILESTREAM Filegroups

```
CREATE TABLE Candidate(
  CandidateId int IDENTITY PRIMARY KEY,
  BlobId uniqueidentifier ROWGUIDCOL NOT NULL UNIQUE,
  Position varchar(max),
  Resume varbinary(max) FILESTREAM)
 FILESTREAM_ON MyDB_docs

CREATE TABLE Product(
  ProductId int IDENTITY PRIMARY KEY,
  BlobId uniqueidentifier ROWGUIDCOL NOT NULL UNIQUE,
  ProductDescription varchar(max),
  Photo varbinary(max) FILESTREAM)
 FILESTREAM_ON MyDB_photos
```



2012+

2008+

Limitations and Considerations

- Database Snapshots
  - Not supported for FILESTREAM filegroups
- Integrated security required for streaming API
  - SQL Server authentication not supported
- Transparent Data Encryption (TDE)
  - Supported, but won't encrypt files
- Mirroring
  - Not supported; use AlwaysOn
- NTFS file system
  - SMB network attached storage not supported
- Replication and log shipping
  - All participating servers must be running SQL Server 2008 or higher
  - For replication, the ROWGUIDCOL column must have a NEWSEQUENTIALID or NEWID default
- SQL Server Express edition
  - Fully supported
  - Database size limit (10GB) does not include FILESTREAM data

2008+

Best Practices

- Don't use FILESTREAM with very small BLOBs (< 1 MB)
  - That's just overkill
- For larger BLOBs, use SqlFileStream
  - T-SQL is OK for smaller BLOBs, but SQL Server memory resources take a hit for larger ones
- If reads require only the first few bytes, use T-SQL with SUBSTRING
- Disable short (8.3) filenames
  - `fsutil behavior set disable8dot3 1`
- Disable last access time
  - `fsutil behavior set disablelastaccess 1`
- Tweak cluster size if you have predictable size patterns
  - Tradeoff between disk space (larger clusters) and fragmentation + I/O (smaller clusters)
  - `format F: /FS:NTFS /V:MyFILESTREAMVolume /A:64K`
- Defragment periodically

2008+

More Information

- Pluralsight
  - SQL Server 2012-2014 Native File Streaming
- The Art of SQL Server FILESTREAM
  - http://www.amazon.com/The-Art-SQL-Server-FILESTREAM/dp/1906434891
- Full-text search (FTS)
  - http://msdn.microsoft.com/en-us/library/ms142571.aspx
- Semantic search
  - http://msdn.microsoft.com/en-us/library/gg492075.aspx
- WCF streaming with MTOM
  - http://msdn.microsoft.com/en-us/library/ms733742(v=vs.110).aspx
- OpenSqlFilestream
  - http://msdn.microsoft.com/en-us/library/bb933972.aspx

2008+

hierarchyid

2008+

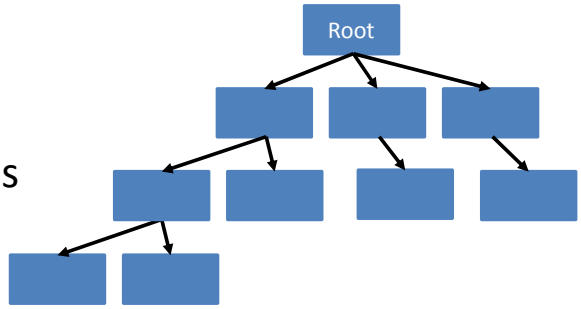
What is hierarchyid?

- System CLR data type
  - Extremely compact variable-length binary format
- Enables a robust hierarchical structure over a self-joining table
  - Each row is a node with a unique hierarchyid value
  - Contains the path in the hierarchy to the node… down to the sibling ordinal position
- Invoke methods in T-SQL
  - Efficiently query the hierarchy
  - Arbitrarily insert, modify, and delete nodes
  - Reparent entire sub-trees with a single update

2008+

Hierarchical Storage Scenarios

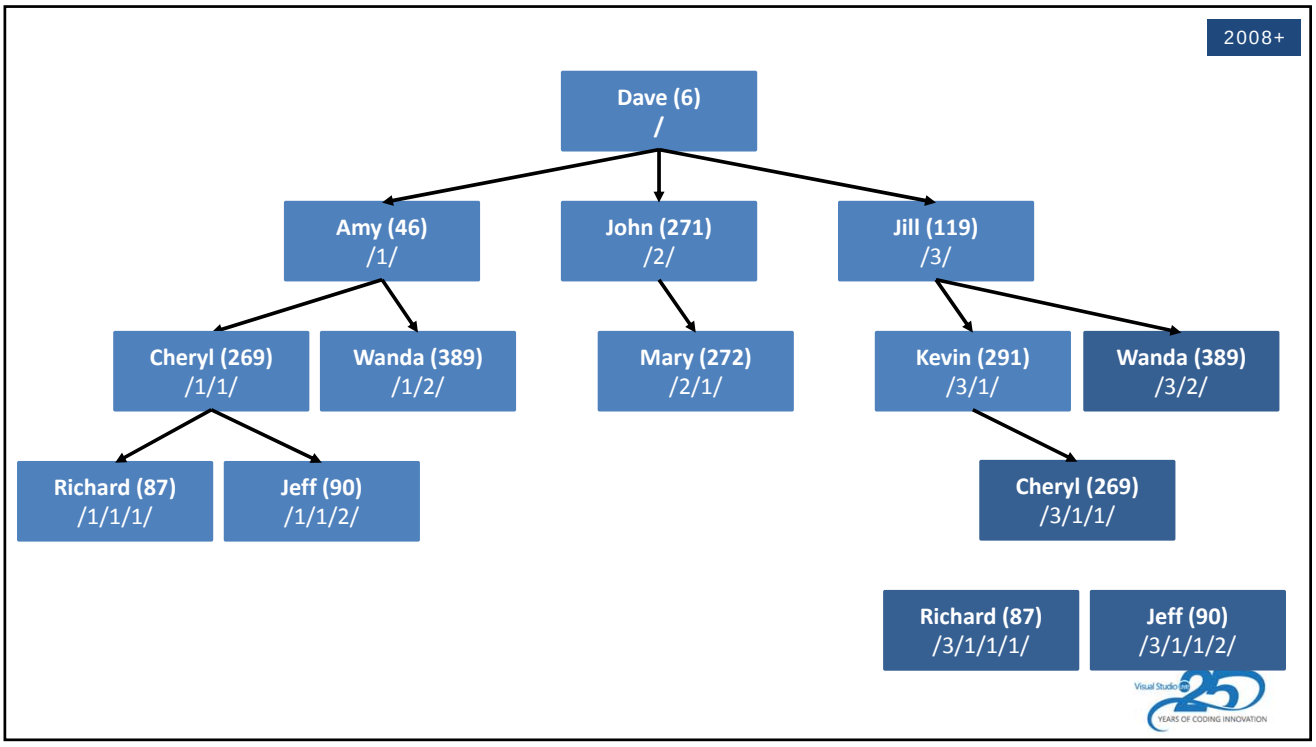
- File system
- Product categories
- Business organization charts
- Forum and mailing list threads
- Content management
- Many more…
  - Unlimited breadth and depth
  - Recursive iteration

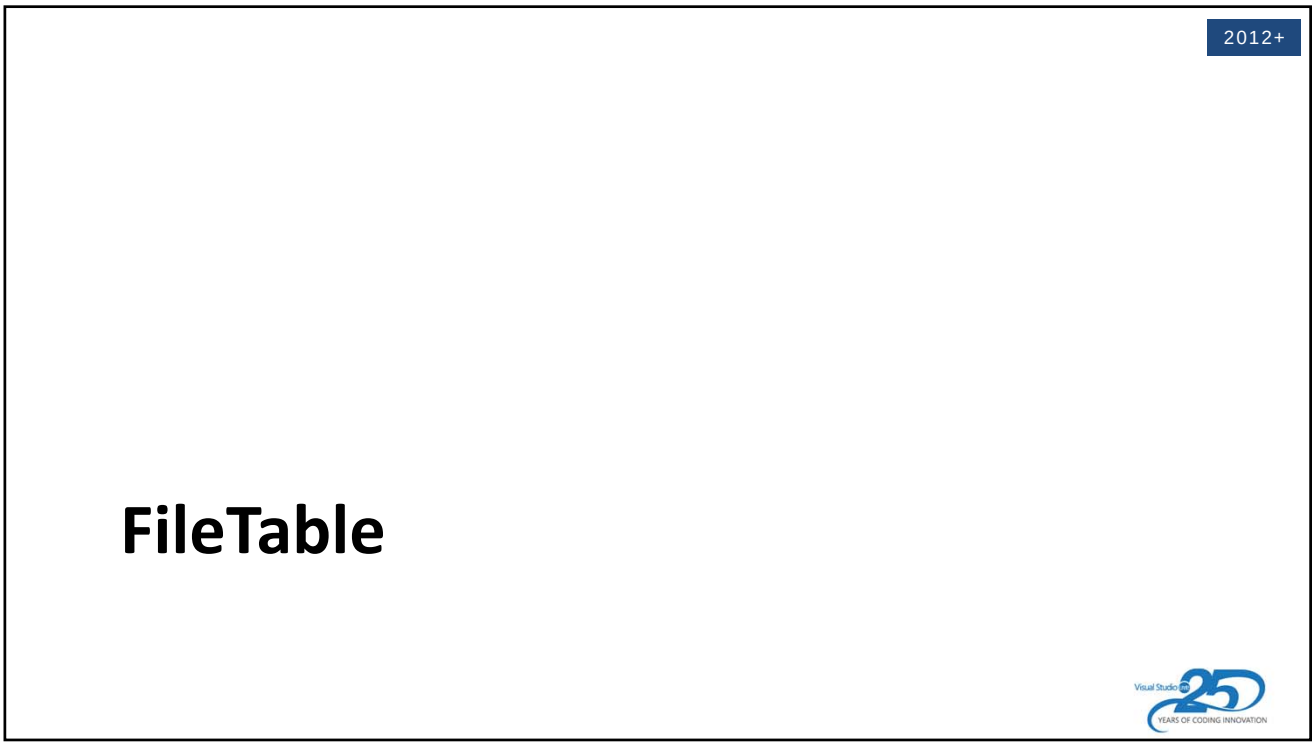


2008+

Common hierarchyid Methods

- GetAncestor
  - Find a node's parent
- GetDescendant
  - Generate a new child node
- GetLevel
  - Return node depth
- GetReparentedValue
  - Move a node
- GetRoot
  - Reference the root node
- IsDescendantOf
  - Return a subtree
- Parse
  - Convert string format to binary value
- ToString
  - Convert binary value to string format


2008+
Dave (6)
/
Amy (46)
/1/
John (271)
/2/
Jill (119)
/3/
Cheryl (269)
/1/1/
Wanda (389)
/1/2/
Mary (272)
/2/1/
Kevin (291)
/3/1/
Wanda (389)
/3/2/
Richard (87)
/1/1/1/
Jeff (90)
/1/1/2/
Cheryl (269)
/3/1/1/
Richard (87)
/3/1/1/1/
Jeff (90)
/3/1/1/2/


2012+
FileTable

2012+

Introducing FileTable

- FILESTREAM is great, but
  - It's *purely* a storage abstraction layer
  - Accessible only to developers and admins
  - The back-end file system is obfuscated and not accessible to users
- Introducing FileTable
  - It's an ordinary table
    - Each row represents either a file or folder
  - It's a functional file system
    - The entire table surfaces as a Windows file share
- FileTable =
  - FILESTREAM + hierarchyid + Windows file system API

2012+

FileTable Schema

Column Name	Data Type	Description
stream_id	uniqueidentifier ROWGUIDCOL	Unique row identifier
file_stream	varbinary(max) FILESTREAM	BLOB content (NULL if directory)
name	nvarchar(255)	Name of file or directory
path_locator	hierarchyid	Location of file or directory within the file system hierarchy
creation_time	datetimeoffset(7)	Created
last_write_time	datetimeoffset(7)	Last modified
last_access_time	datetimeoffset(7)	Last accessed
is_directory	bit	0 = file, 1 = directory
is_offline	bit	Storage attributes
is_hidden	bit	
is_readonly	bit	
is_archive	bit	
is_system	bit	
is_temporary	bit	

2012+

FileTable Access

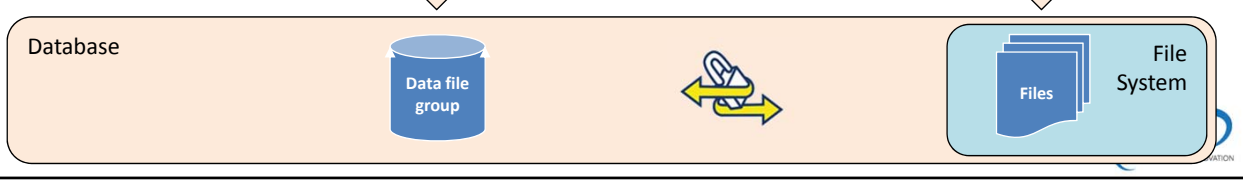
stream_id	name	path_locator	is_directory	...	file_stream
27D8D4AD-D100-39…	'Financials'	0xFF271A3562…	1	…	NULL
78F603CC-0460-73…	'ReadMe.docx'	0xFF59345688…	0	…	0x3B0E956636AE3B2F020B…
207D4A96-E854-01…	'Budget.xlsx'	0xFD0011039A…	0	…	0xF3F359000EEF293039A2…

2012+

FileTable Access via T-SQL

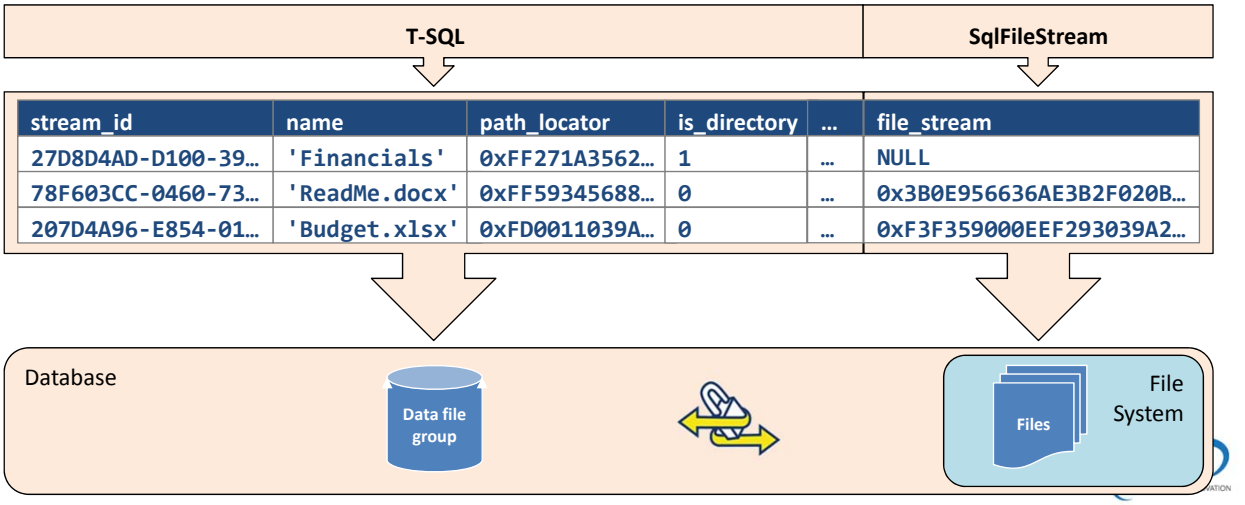
T-SQL

stream_id	name	path_locator	is_directory	...	file_stream
27D8D4AD-D100-39…	'Financials'	0xFF271A3562…	1	…	NULL
78F603CC-0460-73…	'ReadMe.docx'	0xFF59345688…	0	…	0x3B0E956636AE3B2F020B…
207D4A96-E854-01…	'Budget.xlsx'	0xFD0011039A…	0	…	0xF3F359000EEF293039A2…



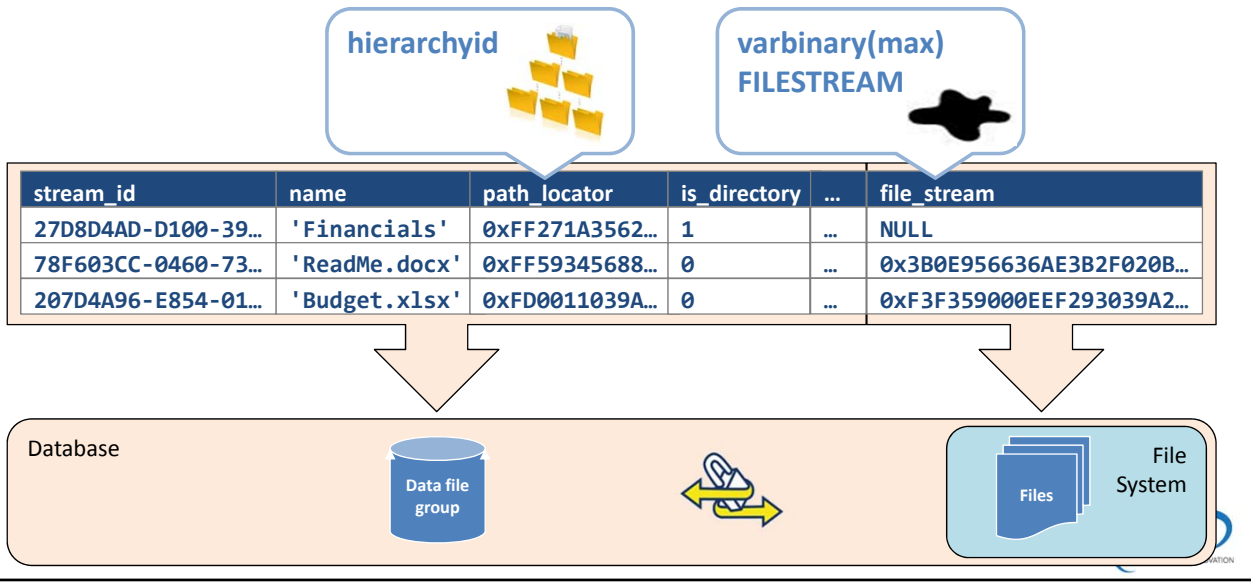
2012+

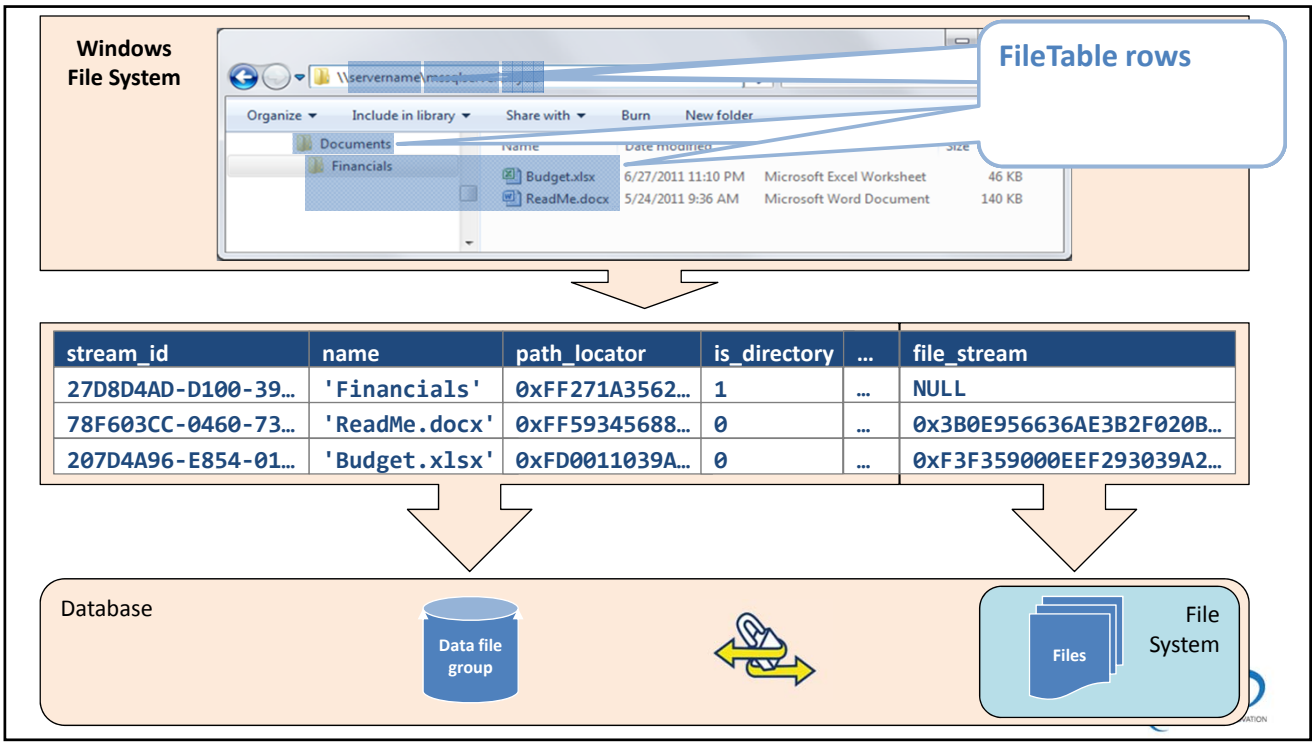
FileTable Access via SqlFileStream



2012+

FileTable Access via Windows File System





2012+

FileTable Prerequisites

- Prerequisites at the instance level
  - FILESTREAM must be enabled for streaming API access
- Prerequisites at the database level
  - Must have a FILESTREAM filegroup and container
  - Set a root directory name for all FileTables in the database
    - Can be different than the database name
    - Surfaces as a folder in the Windows file share for the server instance
  - Enable non-transactional FILESTREAM access for the database
    - Traditional T-SQL and streaming API access is still transactional
    - Can be enabled for readonly or full non-transactional access

2012+

Creating a FileTable-Enabled Database

```
CREATE DATABASE PhotoLibrary
 ON PRIMARY
  (NAME = PhotoLibrary_data,
   FILENAME = 'C:\DB\PhotoLibrary_data.mdf'),
 FILEGROUP FileStreamGroup1 CONTAINS FILESTREAM
  (NAME = PhotoLibrary_photos,
   FILENAME = 'C:\DB\Photos')
 LOG ON
  (NAME = PhotoLibrary_log,
   FILENAME = 'C:\DB\PhotoLibrary_log.ldf')
WITH FILESTREAM
 (DIRECTORY_NAME='PhotoLibrary',
  NON_TRANSACTED_ACCESS=FULL)
```

2012+

FileTable-Enabling an Existing Database

```
ALTER DATABASE PhotoLibrary
SET FILESTREAM
 (DIRECTORY_NAME='PhotoLibrary',
  NON_TRANSACTED_ACCESS=FULL)
```

2012+

Creating a FileTable

- FileTable has a fixed schema
  - You don't (can't) supply a column list
  - You can specify a collation for the name column
    - But it must be a case-insensitive collation
- Exposes a subfolder beneath the database folder in the emulated file system
  - Root directory for the FileTable
  - Named after the table (can be different)
  - Rows in the FileTable surface as files and folders beneath this subfolder

2012+

Creating a FileTable

```
CREATE TABLE FinDoc AS FileTable
  FILESTREAM_ON MyDB_Docs
  WITH (FILETABLE_DIRECTORY = 'Financial Documents')
```

2012+

FileTable Functions

- FileTableRootPath
  - Returns the path for a specified FileTable
- GetFileNamespacePath
  - Returns the path for a specific file_stream instance (folder or file)
- GetPathLocator
  - Returns the hierarchyid for a specific path

2012+

FileTable Namespace

- Refers to a FileTable's hierarchical structure
  - Machine name, instance share name, database name, FileTable name
  - Root path is **\\servername\instance\database\filetable**
- FileTable semantics
  - Uses many constraints, computed columns, and defaults
  - Ensures consistency, which adds overhead
- Improve performance for bulk operations
  - Disabling the namespace disables constraints
  - Temporarily disable the namespace to improve performance for bulk operations

2012+

FileTable Catalog Management Views

- Changed catalog management view
  - **sys.tables**
    - New is_filetable column
- New catalog management views
  - **sys.database_filestream_options**
    - Returns FileTable settings for a database
  - **sys.filetables**
    - Returns information about the FileTables in the database
  - **sys.filetable_system_defined_objects**
    - Returns all the system-generated FileTable defaults and constraints

2012+

Non-Transacted Access DMV/Stored Proc

- New dynamic management view
  - **sys.dm_filestream_non_transacted_handles**
    - Returns currently open non-transactional file and folder handles
- New stored procedure
  - **sp_kill_filestream_non_transacted_handles**
    - Terminates handles to files open for non-transactional access

2012+

FileTable

demo

2012+

FileTable Limitations

- Limited security
  - SQL Server, not the file system, controls security
  - No row-level security = no file permissions
  - Prevent create and delete
    - Deny DELETE and/or INSERT permission on the FileTable
    - Cannot prevent overwriting
- Fixed schema
  - Extend with custom metadata using 1:1-related table
- No support for memory-mapped files

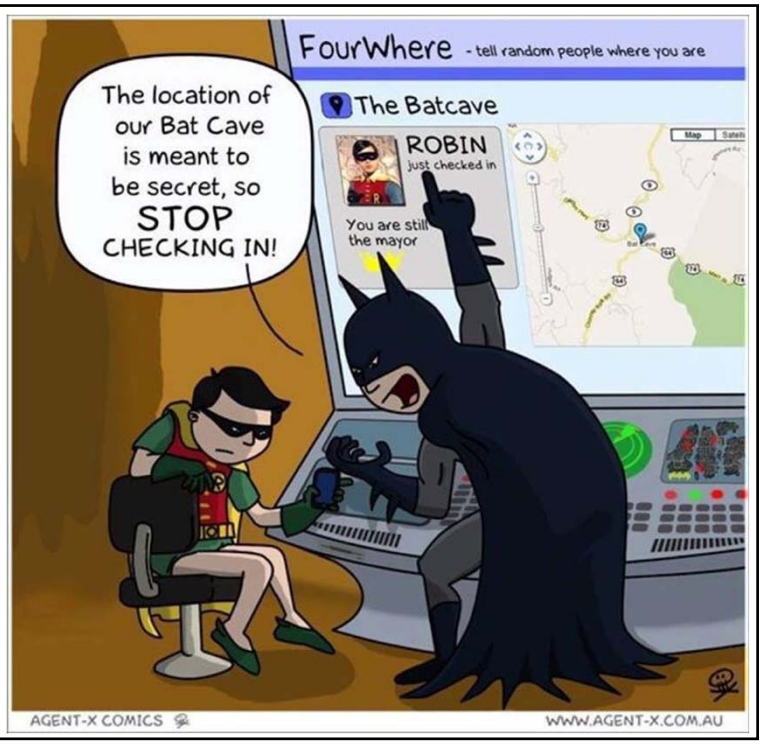
Thank You!

- Contact me
  - lenni.lobel@sleektech.com
- Visit my blog
  - lennilobel.wordpress.com
- Follow me on Twitter
  - @lennilobel
- Thanks for coming! ☺

"Programmers – we do precision guesswork based on unreliable data provided by those of questionable knowledge"

Did you know?

Hey Robin…



"The problem with internet quotes is that you can't always depend on their accuracy"

– Abraham Lincoln

Did you know?

Finding yourself…

"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."

– Martin Golding

Did you know?

Does this describe you too?

"Programmers – we fix problems that you don't know you have, in a way that you don't understand"

Did you know?