Ivan Donev

Real world DevOps challenges with SQL Server and SQL DB


Microsoft


avast


Joyful
Craftsmen


Quest™


J&T BANKA


ORIGAM


PASS


MYIA


PASS
SQLSATURDAY
PRAGUE | 21 OCT 2017

Session objectives and takeaways

At the end of this session, you should be better able to use the SQLDOM parser and the DACFX libraries to:

- Minimize downtime due to database schema changes
- Deploy at scale across hundreds or thousands of databases
- Ensure end-to-end security at the DB layer regardless of the type of the query

Working with Transact-SQL

Have you ever wanted easily to:

- Limit SQL Injection attacks?
- Inspect T-SQL code for best (worse) practices
- Easily parameterize your queries?
  - i.e. use Always Encrypted without having to rewrite your application.
- Format (Pretty Print) your Transact-SQL code?
- Perform static Transact-SQL code analysis?

Scenario: Unparameterized Transact-SQL

Customer Scenario

Vendor A had a solution for the Oil and Gas Industry that had over 10,000 places where dynamic Transact-SQL was generated

Core Issue

A potential new customer wanted all code to be parameterized to reduce the likelihood of SQL Injection

Fix It!

SqlCommandFilters to the rescue!

```
using (SqlCommand cmd = new SqlCommand(
"SELECT Name FROM Patients WHERE SSN =
@SSN”
 , conn))
{
 cmd.Parameters.Add(new SqlParameter(
  "@SSN", SqlDbType.VarChar, 11).Value =
   "111-22-3333");
 SqlDataReader reader =
  cmd.ExecuteReader();
```

CEK

Enhanced ADO.NET

```
exec sp_describe_parameter_encryption
@params = N'@SSN VARCHAR(11)'
, @tsql = N'SELECT * FROM Patients WHERE SSN = @SSN'
```

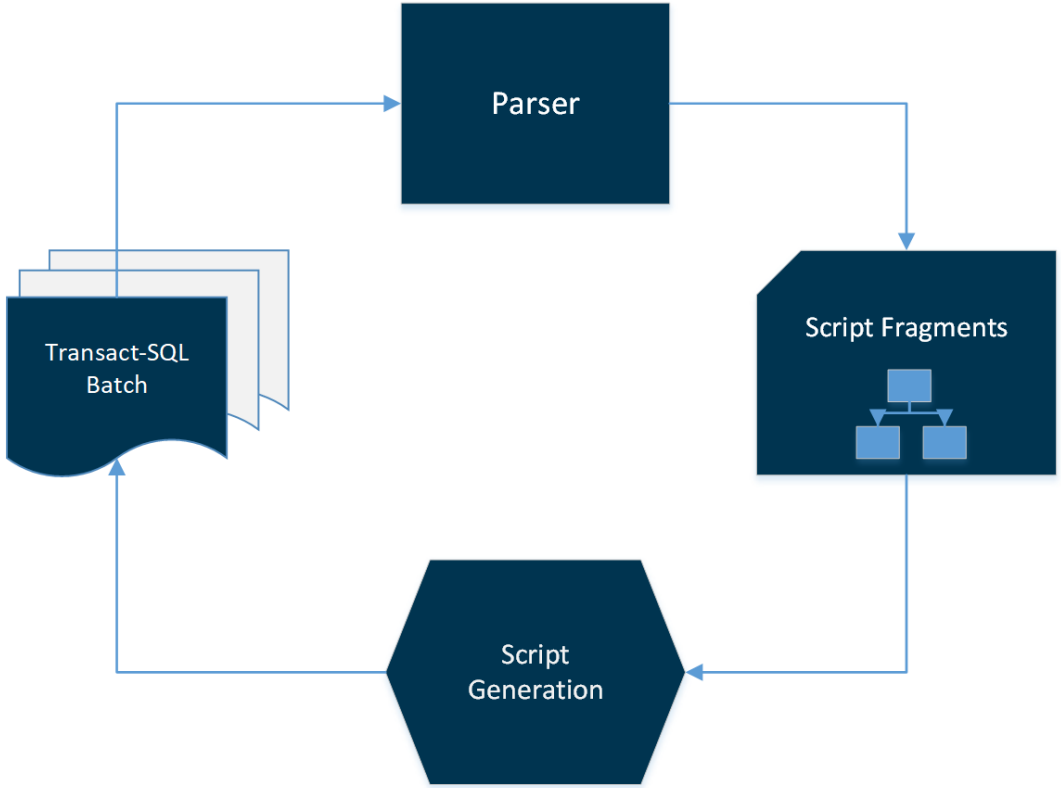
Param	Encrypted CEK Value	CMK Store Provider Name	CMK Path
@SSN		CERTIFICATE _STORE	Current User/ My/f2260...

```
EXEC sp_execute_sql
N'SELECT * FROM Patients WHERE SSN = @SSN'
, @params = N'@SSN VARCHAR(11)', @SSN=0x7ff654ae6d
```

Name	SSN
0x19ca706...	0x7ff654ae...
0xfbd9ae...	0x654ae6...

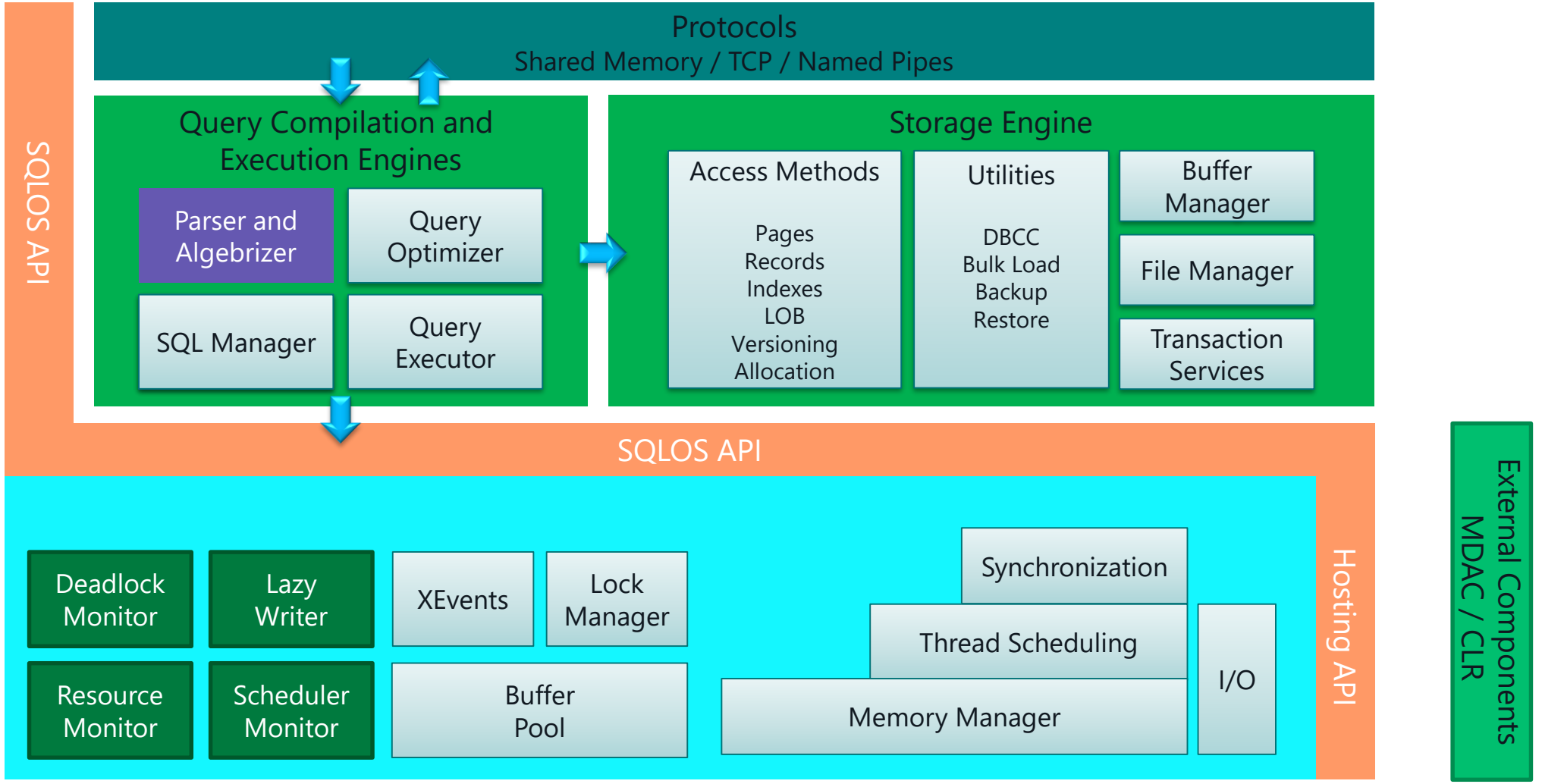
Name
Jim Gray

The ‘Crown Jewels’



Part of the SQL Server 2016 Feature Pack

SQL Server Generic Architecture



= thread, SQLOS actually does not know what these are

Getting started

Obtaining the parsers and DACFX (download locations)

- SSMS
- DACFX redist (GAC considerations)
- SQLServer PowerShell module

'Use specific version' consideration in .NET

SSDT: Always ensure you are updated! (https://docs.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt)

In a nutshell

```
// Get the parse tree
TSqlFragment tree = parser.Parse(rdr, out errors);

//Use the vistor pattern to examine the parse tree
TsqlBatchVisitor visit = new TsqlBatchVisitor(
    cmd, reparse);

// Now walk the tree and do our work
tree.Accept(visit);
```

How to use SqlCommandFilters?

```
con.Open();
SqlCommand cmd = new SqlCommand();
cmd.Connection = con;
cmd.CommandText = “your code”
SqlCommandFilters.Parameters.Parameterize(ref cmd);
// cmd is now parameterized!
```

Demo: ExplicitVisitor and CommandFilters

Scenario: Project Database Settings vs. SQL defaults

Customer Scenario

Customer used a T-SQL script to change compatibility level of 400 databases in an elastic pool to compatibility 130. Some weeks later, they found the compatibility level magically changed back to 120

Core Issue

Making changes directly to the database is not a great idea! Changing compatibility level directly in the DB and not in the SSDT project will cause changes to be 'rolled back' on subsequent DACPAC deployments

Fix It!

DevOps with DACPACs is all about declaratively changing schema / code in the SSDT project and then 'deploying that forward' into the DB. Always make changes to the SSDT project first, and be aware of certain non-default settings!

Database Settings

Common | Operational | Miscellaneous

Default Filegroup

Default filegroup: PRIMARY

Default filestream filegroup:

Automatic

- [ ] Auto close
- [ ] Auto shrink
- [x] Auto create statistics
- [x] Auto update statistics
- [ ] Auto create incremental
- [ ] Auto update statistics asynchronously

Cursor

- [ ] Close cursor on commit enabled

Default cursor: LOCAL

Recovery

Recovery: FULL

Target recovery time (seconds): 0

Page verify: NONE

Snapshot Options

- [ ] Allow snapshot isolation
- [ ] Read committed snapshot
- [ ] Memory optimized elevate to snapshot

Transactions

Delayed durability:

Query Store

Operation mode: Off

Query capture mode: All

Stale query threshold (days): 367

Data flush interval (seconds): 900

Max storage size (MB): 100

OK | Cancel

Scenario: Importing a ‘bad’ BACPAC

Customer Scenario

A large SaaS ISV was using BACPACs to provide an ‘offline backup’ for their customer databases (Azure SQL DB). One day, they found a particular BACPAC would not import successfully

Core Issue

The database had a set of default constraints which had names conflicting with auto-generated default constraint names (demo will help you visualize this!)

Fix It!

We were able to use a custom deployment contributor to auto-name unnamed default constraints, thereby avoiding the name conflict

Scenario: Online ALTER COLUMN

Customer Scenario

A large SaaS ISV was forced to take planned downtime (hundreds of DBs) during their DACPAC deployments due to database blocking issues

Core Issue

DACPAC deployment internally produces a script which uses ALTER TABLE… ALTER COLUMN in some cases. These take out restrictive schema modification locks, causing blocking

Fix It!

Azure SQL DB supports online ALTER COLUMN operations, but for DACPAC deployment to take advantage of this, we needed to implement a custom deployment contributor

Deployment Contributors

What are they?

Where can they be used?

- Import / Export BACPAC
- Extract / Deploy DACPAC

What is already available?

Syntax

- SQLPackage
- SSDT deploy
- DACFX

Demo: Importing a ‘bad’ BACPAC and Online ALTER COLUMN

In review: session objectives and takeaways

- Minimize downtime due to database schema changes?
- Deploy at scale across hundreds or thousands of databases?
- Ensure end-to-end security at the DB layer regardless of the type of the query?
- By using the SQLDOM parser and the DACFX libraries to accomplish these objectives

Session resources

Sample Code Repositories

https://github.com/Microsoft/DACExtensions/

https://github.com/arvindshmicrosoft/SQLScriptDomSamples

https://github.com/sqlbobt/SqlCommandFilters

https://github.com/GoEddie/ScriptDomVisualizer/tree/master/release

More on SQLDOM

https://www.youtube.com/watch?v=CciVxRFXgH8

https://blogs.msdn.microsoft.com/arvindsh/tag/sqldom/

Slidedeck and forked demos

https://github.com/ikdonev/SQLSatPrague689


Microsoft


avast


Joyful
Craftsmen


Quest™


J&T BANKA


ORIGAM


PASS


MYIA


PASS
SQLSATURDAY
PRAGUE | 21 OCT 2017