SQLintersection

Wednesday {Part 1 : 1:45 - 2:45PM, Part 2: 3:00 - 4:00PM}

Query Processing Innovations

Joe Sack, Principal Program Manager, Microsoft
Joe.Sack@Microsoft.com

Speaker: Joe Sack

Program Manager on Query Processing in the Azure SQL Database engineering team

Working as a SQL professional for 21 years

Overview

In this two-part session we'll learn more about query processing improvements added in Azure SQL Database, SQL Server 2017 and SQL Server 2019

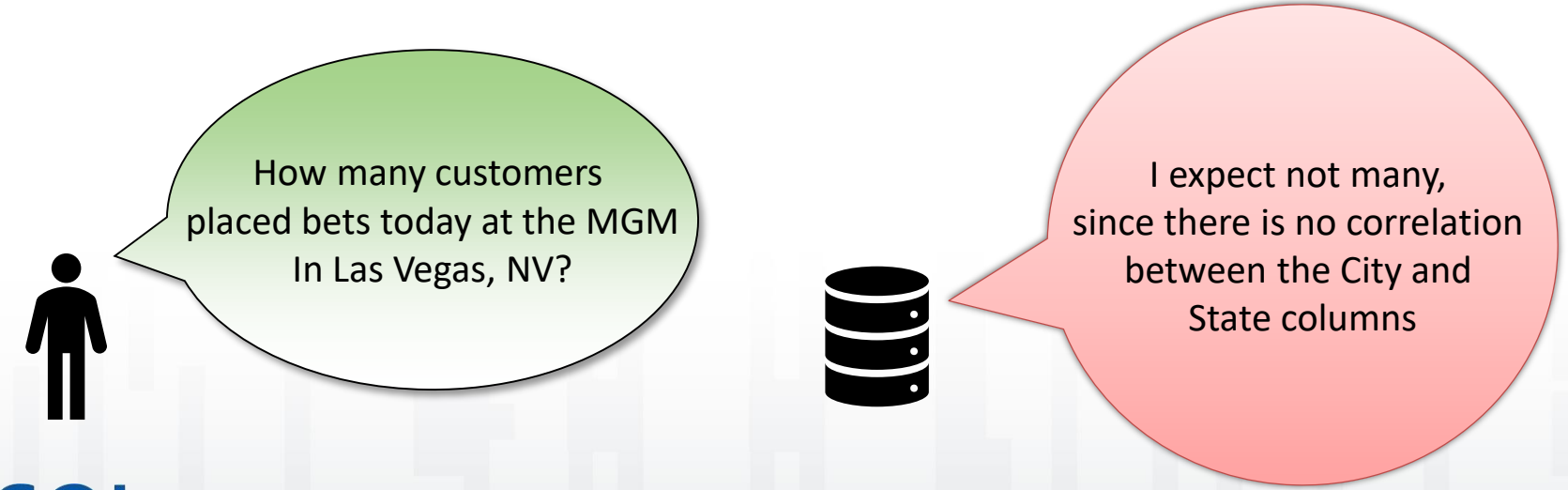
SQL intersection

QP North Star

Generate a good-enough query execution plan *quickly* that results in optimal execution time

Query Processing Problem Spaces

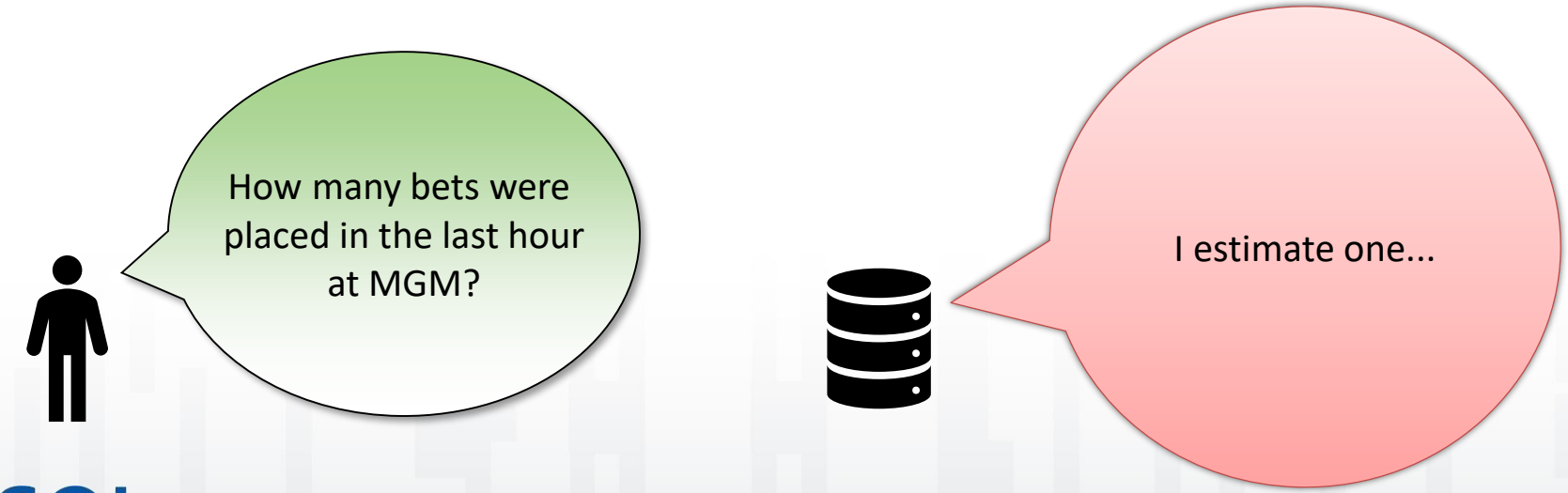
Our assumptions may not match the reality of the actual data distributions, correlation, containment, and inclusion



SQL intersection

Query Processing Problem Spaces

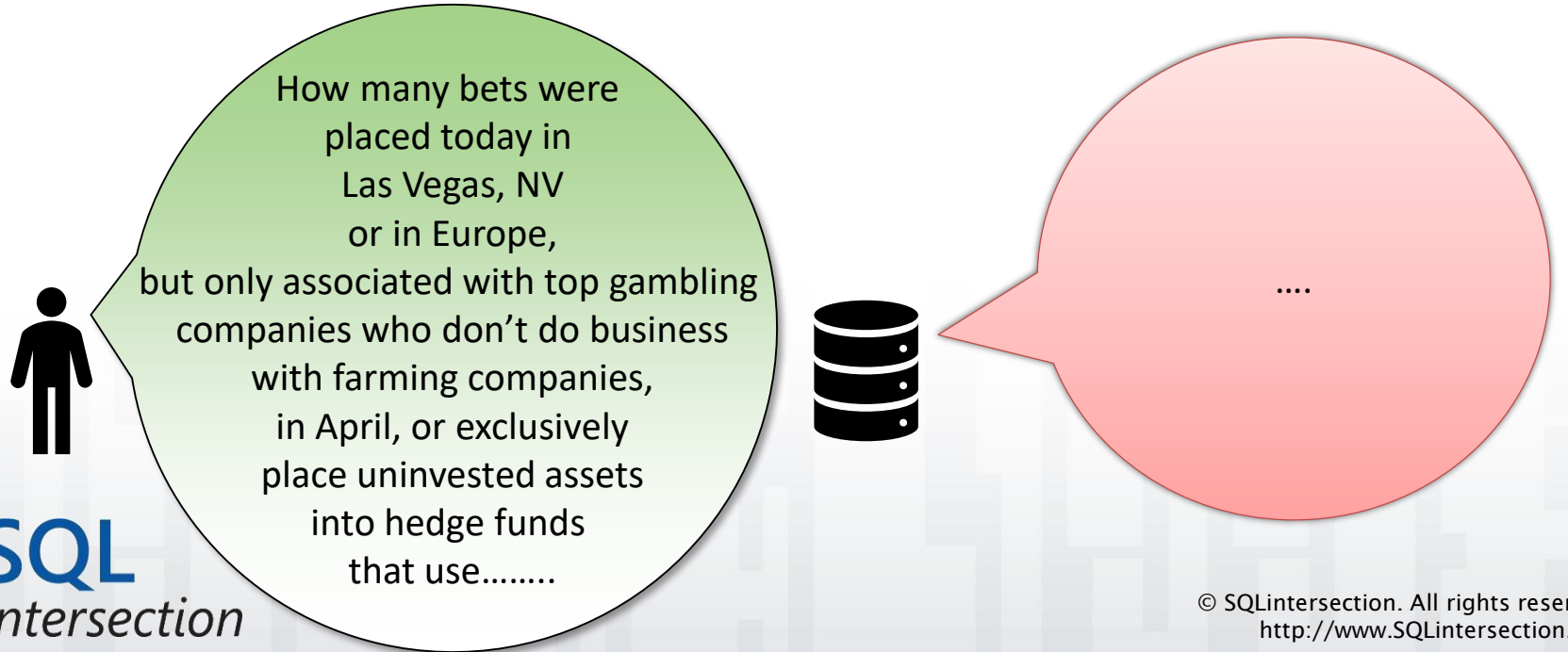
Statistics that don't reflect reality result in poor decisions



SQL intersection

Query Processing Problem Spaces

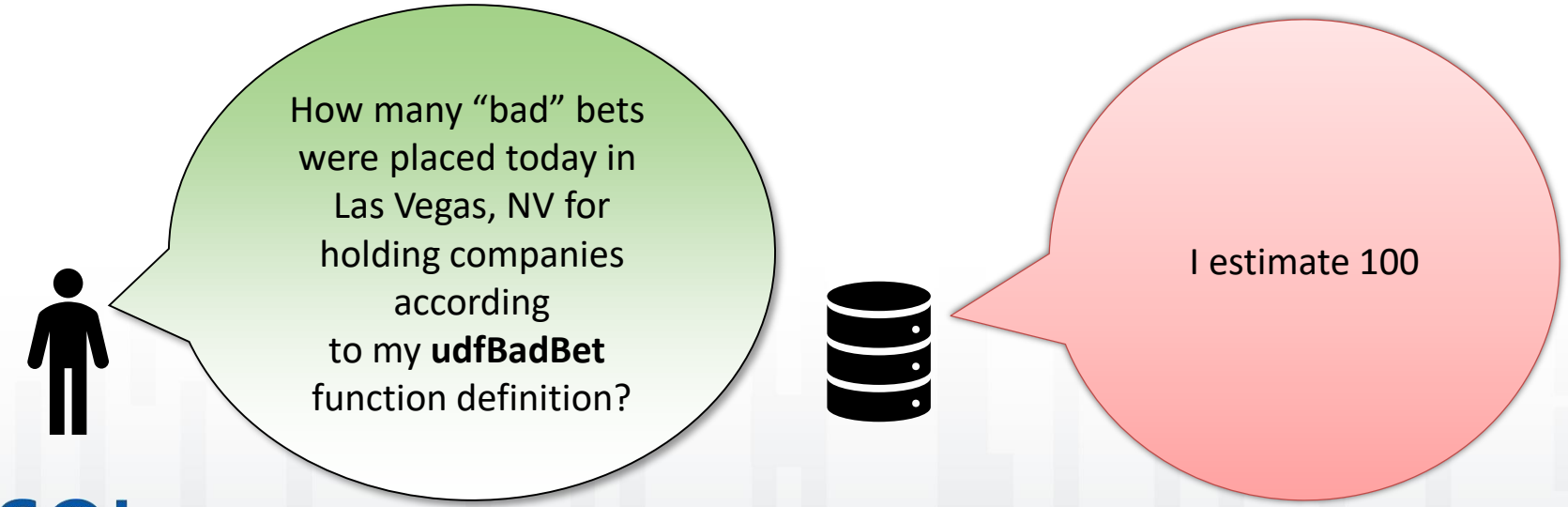
Complexity of the query can significantly increase “search space”, but we cannot afford to search for an unlimited amount of time



SQL intersection

Query Processing Problem Spaces

Some constructs are “black box” for the query optimizer, so we must guess



SQL intersection

TPC_H Query 14 - Promotion Effect

Percentage of the revenue derived from promotional parts

```
SELECT 100.00 * SUM(    CASE
                            WHEN P_TYPE LIKE 'PROMO%' THEN
                                L_EXTENDEDPRICE * (1 - L_DISCOUNT)
                            ELSE
                                0
                        END
                    ) / SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS
                PROMO_REVENUE
FROM LINEITEM
    INNER JOIN PART
        ON L_PARTKEY = P_PARTKEY
WHERE L_SHIPDATE >= '1997-06-01'
      AND L_SHIPDATE < DATEADD(mm, 1, '1997-06-01');
```

Source: www.tpc.org

TPC-H is a decision support benchmark

Search space…

Structures to be used (CI, NCCI)	Lineitem seek or scan?	Part seek or scan?	Lookups needed?
Hash Join, Nested Loop, Merge, Adaptive?	Outer input (LineItem or Part)?	Serial or parallel?	Batch or row execution mode?

Compilation vs. Execution time (1/2)

Default behavior in SQL Server 2019 CTP 2.2, compatibility level 150

Compile (strategy) time 11 ms, Execution time 321 ms

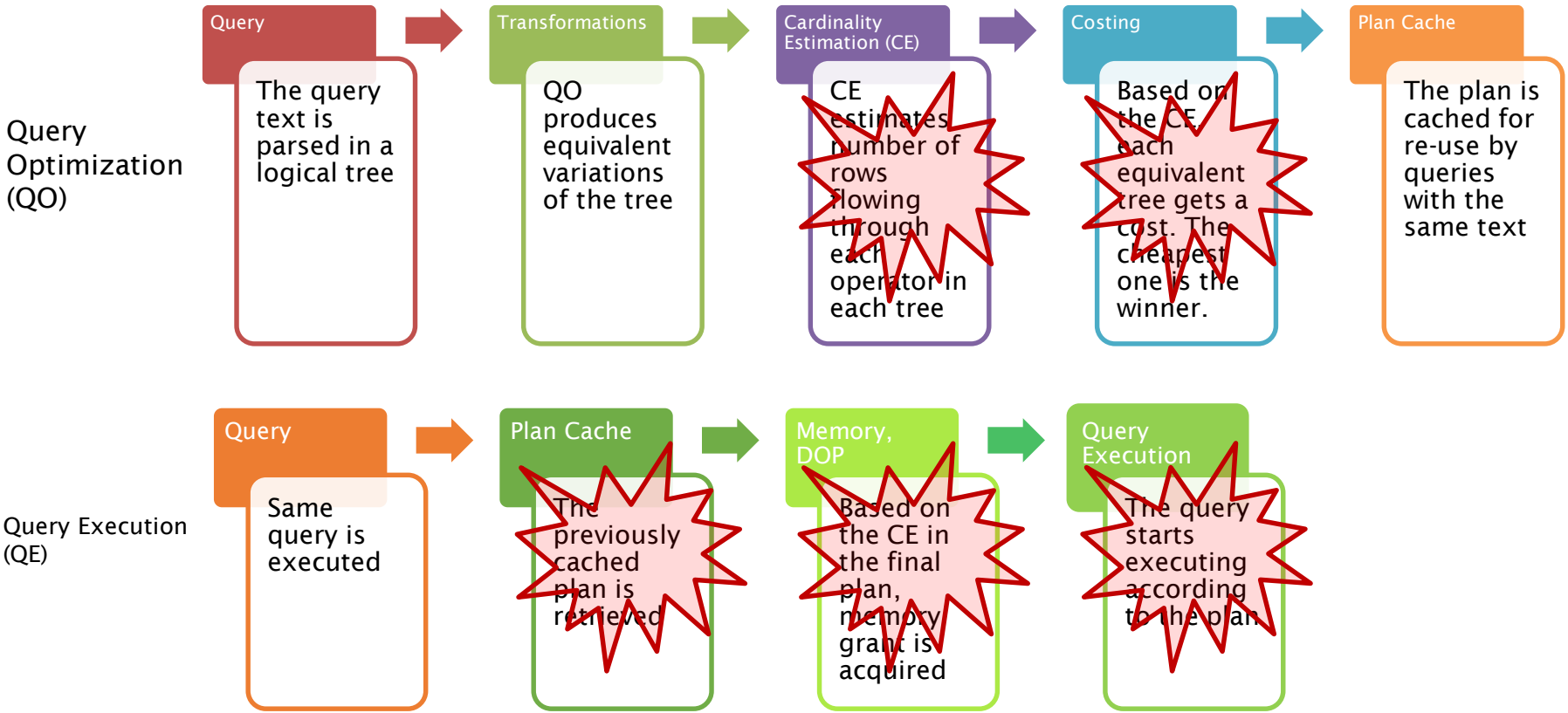
Clustered index seek on LineItem	Adaptive join to Part table	Clustered index scan on Part OR index seek on Part
Hash Match Aggregate	Parallel operators for almost everything	Batch mode on rowstore for both scan paths

Compilation vs. Execution time (2/2)

Consider the following scenarios for the same query:

Compile Time	Execution Time	Total Time
11ms	321ms	332ms
100ms	220ms	320ms – maybe worth it?
1,000ms	10ms	1,010ms – NOT worth it (at least for ad hoc, single execs)

What could go wrong?



Common reasons for incorrect estimates

Missing statistics

Stale statistics

Inadequate statistics sample rate

Bad parameter sniffing scenarios

Out-of-model query constructs

- E.g. Multi-Statement TVFs, table variables, XQuery

Assumptions not aligned with data being queried

- E.g. independence vs. correlation

Cost of incorrect estimates

Slow query response time due to inefficient plans

Excessive resource utilization (CPU, Memory, IO)

Spills to disk

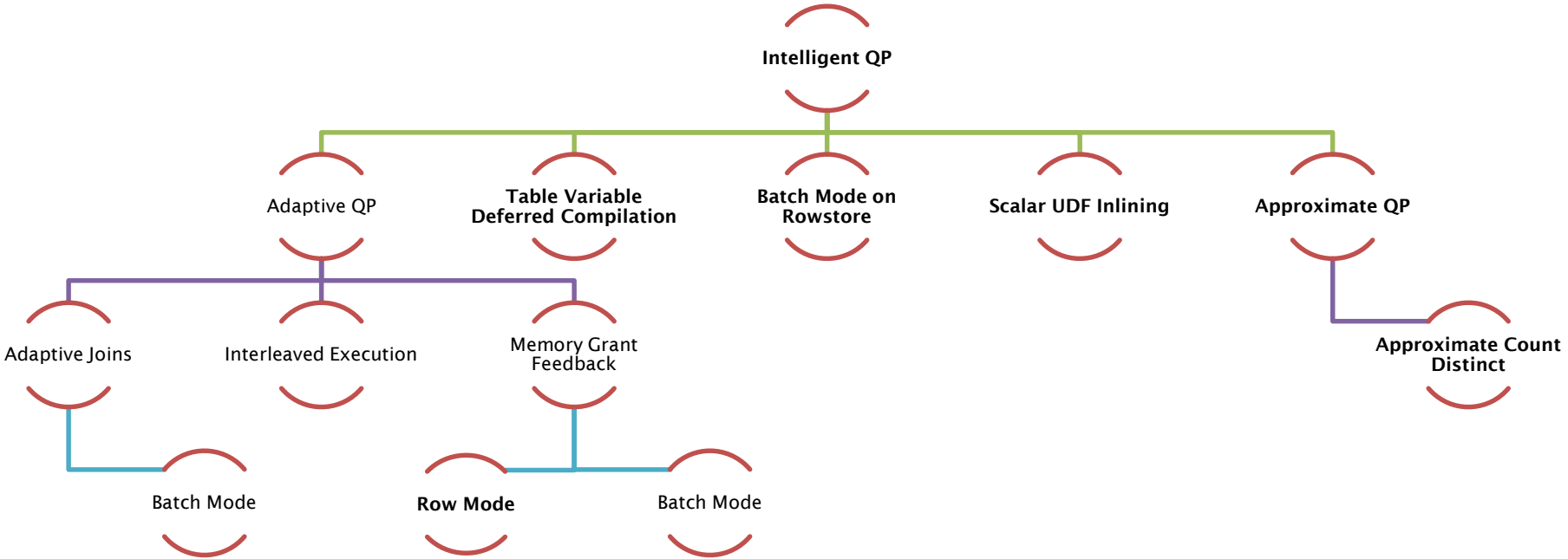
Reduced throughput and concurrency

T-SQL refactoring to work around off-model statements

QP feature roadmap strategy

- React to issues during execution
- React to misestimates during compilation
- Address old limitations (prioritized for big markets)
- Learn via feedback
- Add intelligence to common operations

Intelligent Query Processing



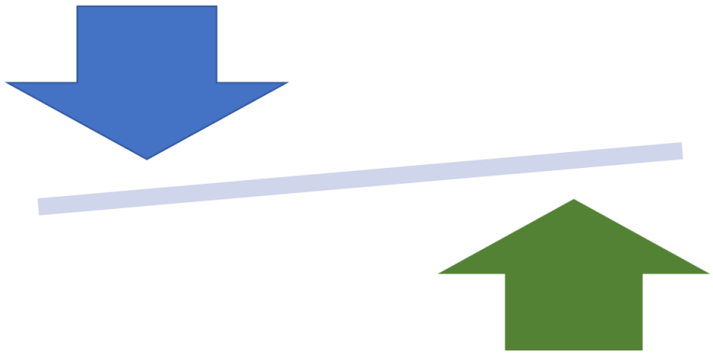
The **intelligent query processing** feature family includes features with broad impact that improve the performance of existing workloads with minimal implementation effort.

Batch Mode Memory Grant Feedback (MGF)

Problem: Queries may spill to disk or take too much memory based on poor cardinality estimates

MGF will adjust memory grants based on execution feedback

MGF will remove spills and improve concurrency for repeating queries



Row Mode Memory Grant Feedback

Same as batch mode MGF, but enabled for row mode

New query execution plan attributes to understand the state of memory grant feedback

MemoryGrantInfo	
DesiredMemory	13992
GrantedMemory	13992
GrantWaitTime	0
IsMemoryGrantFeedbackAdjusted	YesStable
LastRequestedMemory	13992
MaxQueryMemory	1497128
MaxUsedMemory	3744

150 database compatibility level

MGF and parameter sensitive scenarios

Different parameter values may also require different query plans in order to remain optimal

This type of query is defined as "parameter-sensitive."

For parameter-sensitive plans, memory grant feedback will disable itself on a query if it has unstable memory requirements

The plan is disabled after several repeated runs of the query and this can be observed by monitoring the memory_grant_feedback_loop_disabled XEvent

MGF grant feedback caching

Feedback can be stored in the cached plan for a single execution

It is the consecutive executions of that statement, however, that benefit from the memory grant feedback adjustments

Memory grant feedback will change only the cached plan

Changes are currently not captured in the Query Store. Feedback is not persisted if the plan is evicted from cache. Feedback will also be lost if there is a failover.

Disabling Memory Grant Feedback

Set database compatibility level less than 140 (batch mode MGF), less than 150 (row mode MGF)

(2017) ALTER DATABASE SCOPED CONFIGURATION SET
DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK = OFF;

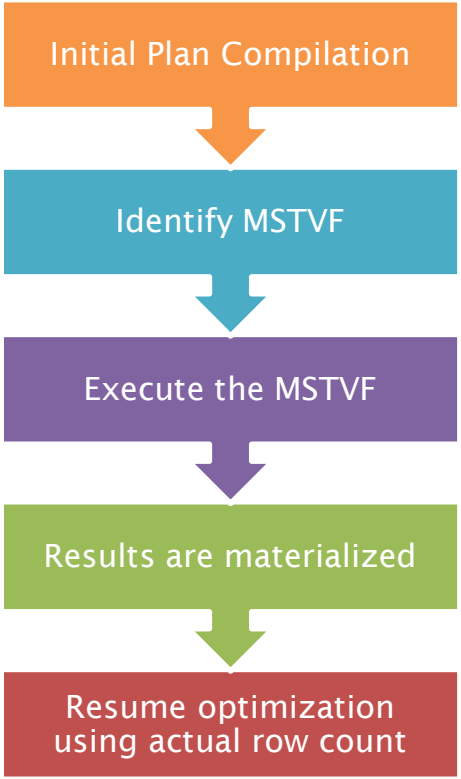
(2019) ALTER DATABASE SCOPED CONFIGURATION SET
BATCH_MODE_MEMORY_GRANT_FEEDBACK = OFF;

ALTER DATABASE SCOPED CONFIGURATION SET **ROW_MODE_MEMORY_GRANT_FEEDBACK** = OFF;

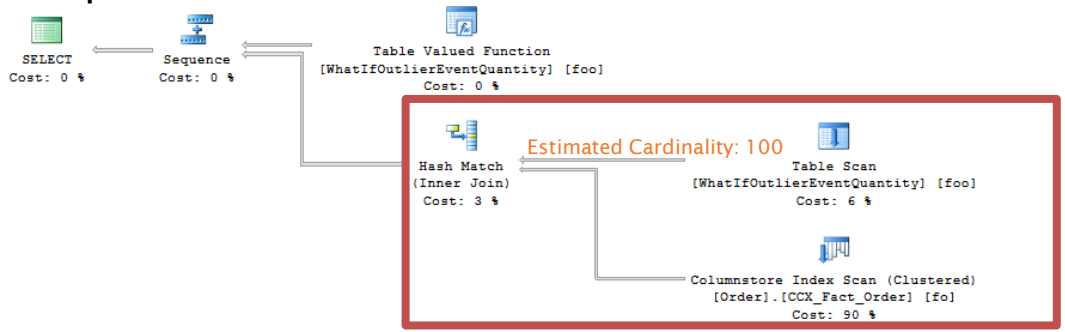
OPTION (USE HINT('**DISABLE_BATCH_MODE_MEMORY_GRANT_FEEDBACK**'))

OPTION (USE HINT ('**DISABLE_ROW_MODE_MEMORY_GRANT_FEEDBACK**'))

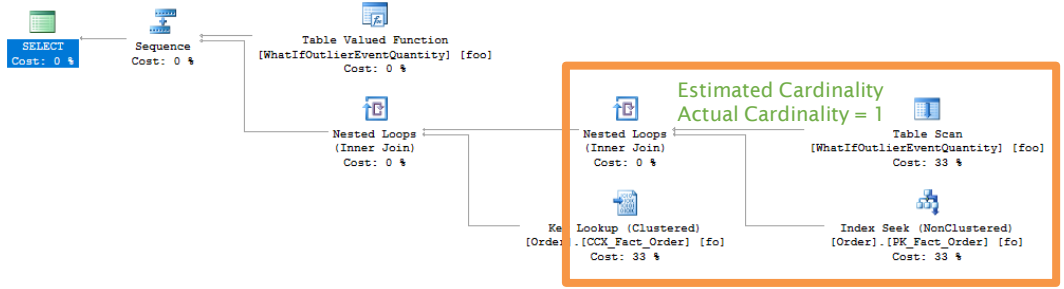
Interleaved Execution for MSTVFs



Old plan



New Plan with Interleaved Execution



Interleaved Execution Benefit

In general, interleaved execution benefits queries where:

1. There is a large skew between the estimated vs. actual number of rows for the intermediate result set (in this case, the MSTVF)
2. And the overall query is sensitive to a change in the size of the intermediate result. This typically happens when there is a complex tree above that subtree in the query plan.

A simple SELECT * from an MSTVF will not benefit from interleaved execution.

Interleaved Execution Overhead

The overhead should be minimal-to-none. MSTVFs were already being materialized prior to the introduction of interleaved execution

As with any plan affecting changes, some plans could change such that with better cardinality for the subtree we get a worse plan for the query, overall

Once an interleaved execution plan is cached, the plan with the revised estimates on the first execution is used for consecutive executions without re-instantiating interleaved execution

Disabling Interleaved Execution

Set database compatibility level less than 140

(2017) ALTER DATABASE SCOPED CONFIGURATION SET **DISABLE_INTERLEAVED_EXECUTION_TVF** = OFF;

(2019+) ALTER DATABASE SCOPED CONFIGURATION SET **INTERLEAVED_EXECUTION_TVF** = OFF;

OPTION(USE HINT('**DISABLE_INTERLEAVED_EXECUTION_TVF**'));

Table Variable Deferred Compilation

Legacy behavior

Area	Temporary Tables	Table Variables
Manual stats creation and update	Yes	No
Indexes	Yes	Only inline index definitions allowed.
Constraints	Yes	Only PK, uniqueness and check constraints.
Automatic stats creation	Yes	No
Creating and using a temporary object in a single batch	Compilation of a statement that references a temp table that doesn't exist is deferred until the first execution of the statement	A statement that references a table variable is compiled along with all other statements before any statement that populates the TV is executed, so compilation sees it as "1".

Table Variable Deferred Compilation

Azure SQL Database and SQL Server 2019 behavior

Area	Temporary Tables	Table Variables
Manual stats creation / update	Yes	No
Indexes	Yes	Only inline index definitions allowed.
Constraints	Yes	Only PK, uniqueness and check constraints.
Automatic stats creation	Yes	No
Creating and using a temporary object in a single batch	Compilation of a statement that references a temp table that doesn't exist is deferred until the first execution of the statement	Compilation of a statement that references a table variable that doesn't exist is deferred until the first execution of the statement

150 database compatibility level

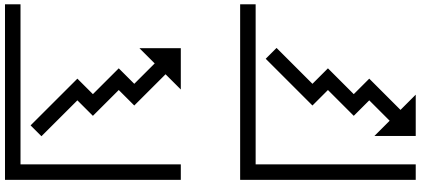
Disabling Table Variable Deferred Compilation

Set database compatibility level less than 150

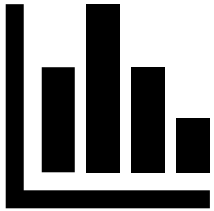
ALTER DATABASE SCOPED CONFIGURATION SET
DEFERRED_COMPILATION_TV = OFF;

OPTION (USE HINT('**DISABLE_DEFERRED_COMPILATION_TV**'))

When approximate is good enough…



Dashboard scenarios against big data sets with many distinct values (for example, distinct orders counts over a time period) – and many concurrent users



Data science big data set exploration. Need to understand data distributions quickly and exact values are not paramount

APPROX_COUNT_DISTINCT

Fast data exploration, dashboard applications, trend analysis where exact values are not necessary

Provide approximate COUNT DISTINCT for big data scenarios with the benefit of high performance and a (very) low memory footprint

Syntax

SQL Copy

```
-- Syntax for Azure SQL Database, Azure SQL Data Warehouse and Parallel Data Warehouse

APPROX_COUNT_DISTINCT ( expression )
```

Arguments

expression

An expression of any type, except **image**, **sql_variant**, **ntext**, or **text**.

APPROX_COUNT_DISTINCT Candidate Workloads

Not banking applications or anywhere an exact value is required! ☺

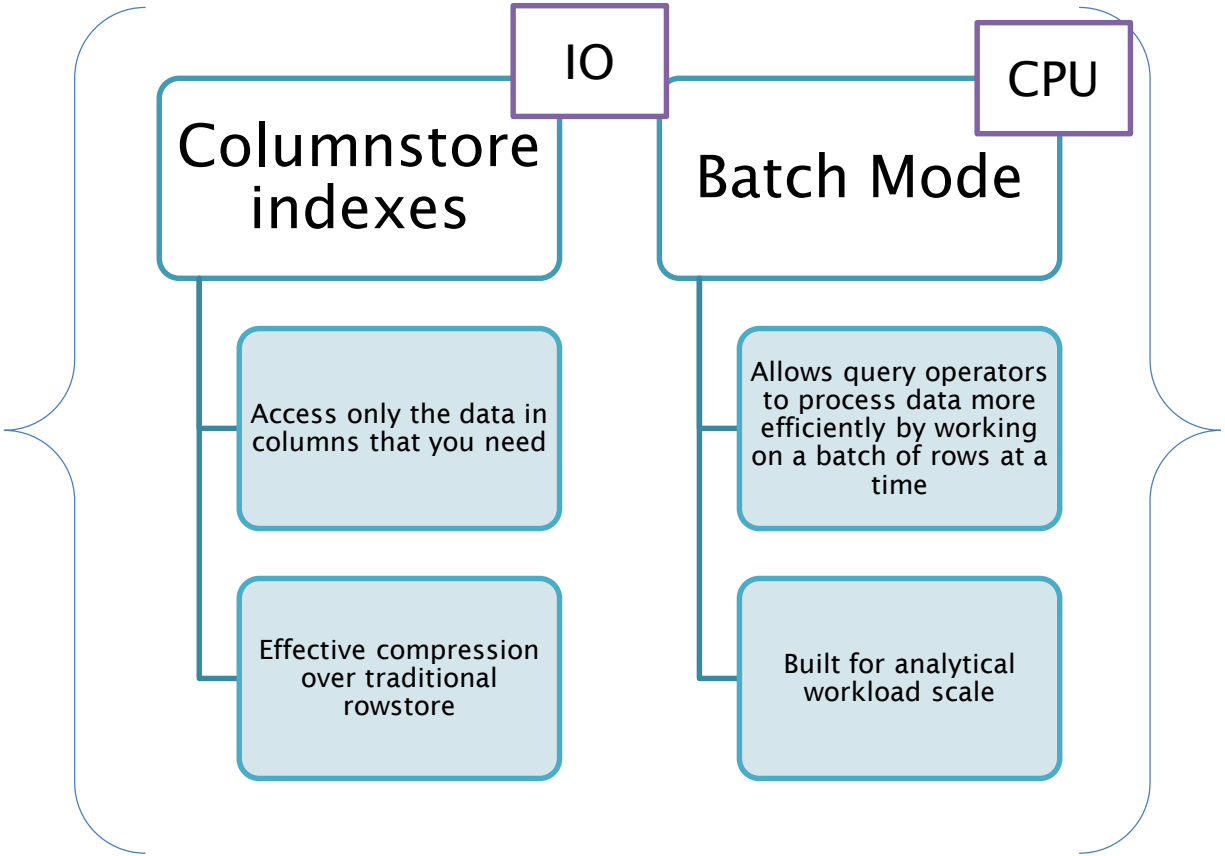
APPROX_COUNT_DISTINCT is designed for use in big data scenarios and is optimized for the following conditions:

- Access of data sets that are millions of rows or higher and
- Column or columns that have many distinct values

The function implementation guarantees up to a 2% error rate within a 97% probability.

Batch Mode and Columnstore

Since SQL Server 2012 – we've bound these two features together



Can’t use Columnstore?

Sometimes Columnstore isn’t an option:

- OLTP-sensitive workloads
- Vendor support

Columnstore interoperability limitations with your data tier design?

- For example – CCIs -> no triggers, cursors, persisted computed columns

Batch Mode on Rowstore

The query optimizer has (until now) considered batch-mode processing only for queries that involve at least one table with a Columnstore index

Batch mode on rowstore:

- Support for batch mode with on-disk heaps and B-tree indexes
  - Scan can evaluate batch mode bitmap filters
- Support for *existing* supported batch mode operators

Batch Mode on Rowstore Heuristics

New checks:

- ✓ A rough initial check involving table sizes, operators used, and estimated cardinalities in the input query
- ✓ Additional checkpoints, as the optimizer discovers new, cheaper plans for the query
- ✓ If plans do not make significant use of batch mode, the optimizer will stop exploring batch mode alternatives

150 database compatibility level

Batch Mode on Rowstore Candidate Workloads

A significant part of the workload consists of analytical queries AND

The workload is CPU bound AND

- Creating a columnstore index adds too much overhead to the transactional part of your workload OR
- Creating a columnstore index is not feasible because your application depends on a feature that is not yet supported with columnstore indexes OR
- You depend on a feature not supported with columnstore (for example, triggers)

Disabling Batch Mode on Rowstore

Set database compatibility level less than 150

ALTER DATABASE SCOPED CONFIGURATION SET
BATCH_MODE_ON_ROWSTORE = OFF;

OPTION(USE HINT('**DISALLOW_BATCH_MODE**'));

Batch Mode on Rowstore Considerations (1/2)

There is no guarantee that query plans will use batch mode.

No guarantee that if you get a row mode plan, it will be the same as the plan you get in a lower compatibility level

No guarantee that if you get a batch mode plan, it will be the same as the plan you'd get with a columnstore index

Plans may also change in subtle ways for queries that mix columnstore and rowstore indexes, because of the new batch mode rowstore scan

Batch Mode on Rowstore Considerations (2/2)

Current limitations for the new batch mode on rowstore scan: It will not kick in for in-memory OLTP tables, or for any index other than on-disk heaps and B-trees.

It will also not kick in if a LOB column is fetched or filtered. This limitation includes sparse column sets, and XML columns.

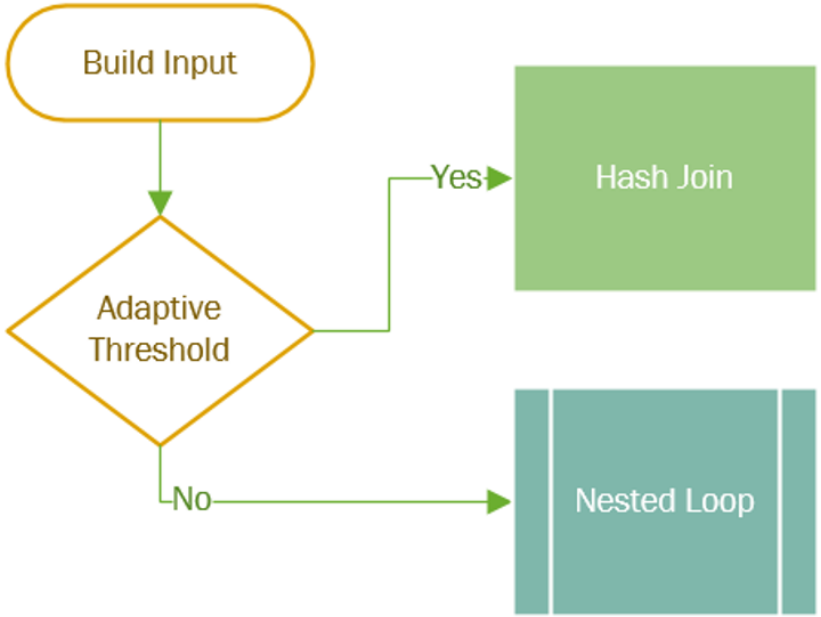
There are queries for which batch mode is not used even with columnstore indexes (for example queries involving cursors), and these same exclusions extend to batch mode on rowstore as well.

Batch Mode Adaptive Joins (AJ)

If cardinality estimates are skewed, we may choose an inappropriate join algorithm

AJ will defer the choice of hash join or nested loop until after the first join input has been scanned

AJ uses nested loop for small inputs, hash joins for large inputs



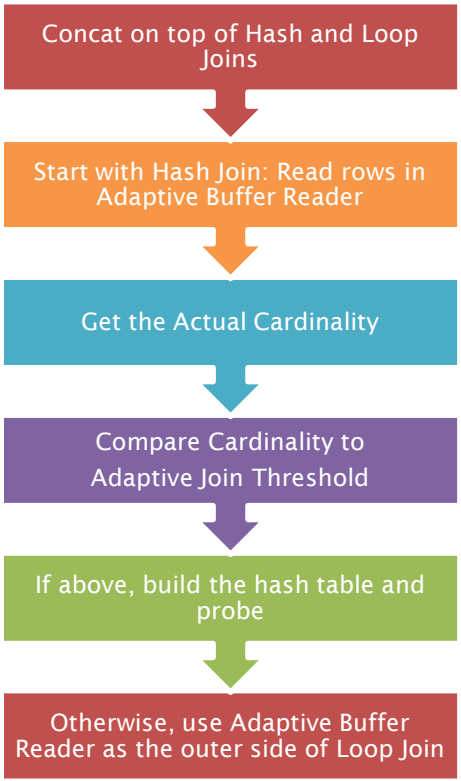
Batch Mode Adaptive Join Eligibility

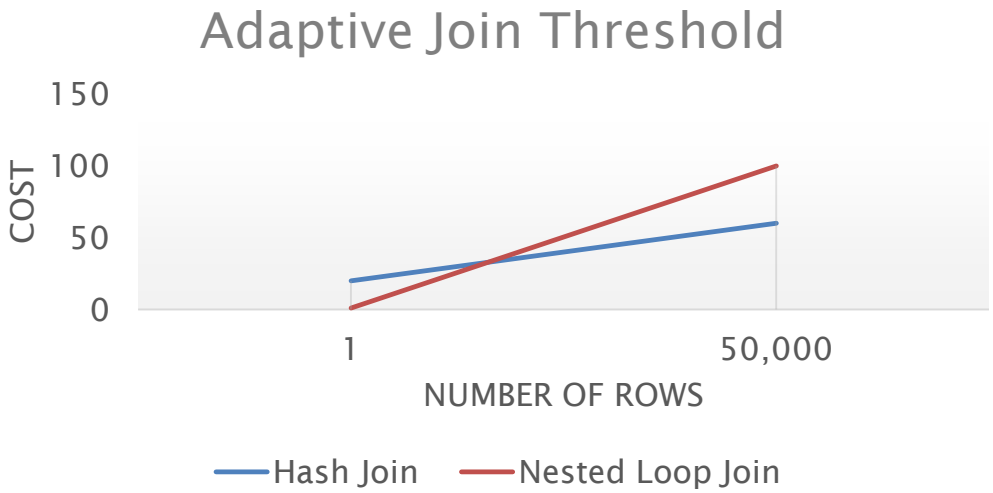
The join is eligible to be executed both by an **indexed nested loop** join and a **hash join** physical algorithm.

The hash join uses batch mode – either through the presence of a Columnstore index in the query overall or a Columnstore indexed table being referenced directly by the join.

New: if batch mode on rowstore enables a batch mode hash join – this becomes an eligible component!

Adaptive Joins





Adaptive Join Cost vs. Benefit

Workloads with frequent oscillations between small and large join input scans will benefit most from this feature

Adaptive joins introduce a higher memory requirement than an indexed Nested Loops Join equivalent plan

Batch mode Adaptive Joins work for the initial execution of a statement, and once compiled, consecutive executions will remain adaptive based on the compiled Adaptive Join threshold

Disabling Adaptive Joins

Set database compatibility level less than 140

(2017) ALTER DATABASE SCOPED CONFIGURATION SET **DISABLE_BATCH_MODE_ADAPTIVE_JOINS** = OFF;

(2019+) ALTER DATABASE SCOPED CONFIGURATION SET **BATCH_MODE_ADAPTIVE_JOINS**= OFF;

OPTION(USE HINT('**DISABLE_BATCH_MODE_ADAPTIVE_JOINS**'));

T-SQL Scalar User-Defined Functions

User-Defined Functions that are implemented in Transact-SQL and return a single data value are referred to as T-SQL Scalar User-Defined Functions

T-SQL UDFs are an elegant way to achieve code reuse and modularity across SQL queries

Some computations (such as complex business rules) are easier to express in imperative UDF form

UDFs help in building up complex logic without requiring expertise in writing complex SQL queries

T-SQL Scalar UDF performance issues!

Iterative invocation: Invoked once per qualifying row. Repeated context switching – and even worse for UDFs that execute SQL queries in their definition

Lack of costing: Scalar operators are not costed (realistically)

Interpreted execution: Each statement itself is compiled, and the compiled plan is cached. No cross-statement optimizations are carried out.

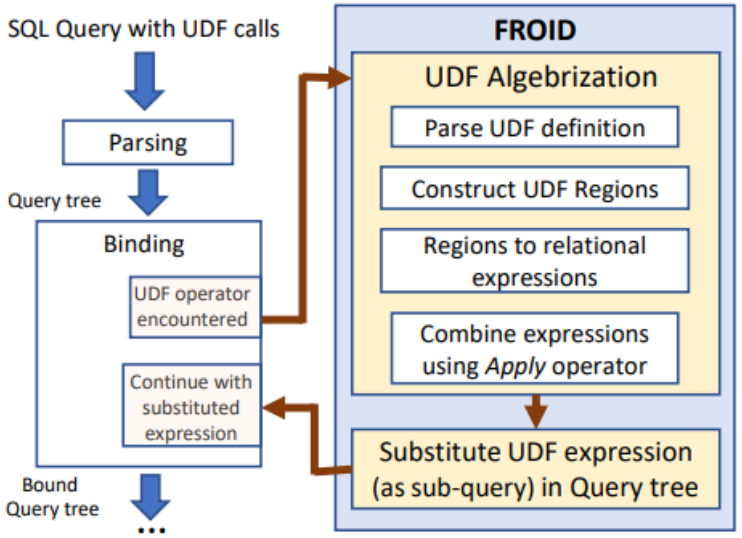
Serial execution: SQL Server does not allow intra-query parallelism in queries that invoke UDFs

Scalar UDF Inlining

Enable the benefits of UDFs without the performance penalty

Goal of the Scalar UDF Inlining feature is to improve performance for queries that invoke scalar UDFs where UDF execution is the main bottleneck

Using query rewriting techniques, UDFs are transformed into equivalent relational expressions that are "inlined" into the calling query



150 database compatibility level

Source: "Froid: Optimization of Imperative Programs in a Relational Database"

Scalar UDF Inlining

Table 1: Relational algebraic expressions for imperative statements (using standard T-SQL notation from [33])

Imperative Statement (T-SQL)	Relational expression (T-SQL)
DECLARE {@*var* *data_type* [= *expr*]}[, . . . *n*];	SELECT {*expr*|*null* AS *var*}[, . . . *n*];
SET {@*var* = *expr*}[, . . . *n*];	SELECT {*expr* AS *var*}[, . . . *n*];
SELECT {@*var*1 = *prj_expr*1}[, . . . *n*] FROM *sql_expr*;	{SELECT *prj_expr*1 AS *var*1 FROM *sql_expr*}; [, . . . *n*]
IF (*pred_expr*) {*t_stmt*; [. . . *n*]} ELSE {*f_stmt*; [, . . . *n*]}	SELECT CASE WHEN *pred_expr* THEN 1 ELSE 0 END AS *pred_val*; {SELECT CASE WHEN *pred_val* = 1 THEN *t_stmt* ELSE *f_stmt*; }[. . . *n*]
RETURN expr;	SELECT *expr* AS *returnVal*;

Source: “Froid: Optimization of Imperative Programs in a Relational Database”

Scalar UDF Inlining

Candidate UDFs

Existing UDFs that are candidates will be inlined during compilation. No need to make changes.

First version candidates:

- DECLARE, SET: Variable declaration and assignments.
- SELECT: SQL query with multiple variable assignments.
- IF/ELSE: Branching with arbitrary levels of nesting.
- RETURN: Single or multiple return statements.
- UDF: Nested/recursive function calls.
- Others: Relational operations such as EXISTS, ISNULL.

Non-inlineable constructs

- Invoking any intrinsic function that is either **time-dependent** (such as GETDATE()) or has **side effects** (such as NEWSEQUENTIALID())
- Referencing table variables or table-valued parameters
- Referencing scalar UDF call in its **GROUP BY clause**
- Natively compiled (interop is supported)
- **Used in a computed column** or a check constraint definition (*we've received lots of feedback on this scenario for this one*)
- References user-defined types
- Used in a partition function

To inline, or not to inline

See **sys.sql_modules** catalog view includes a property called **is_inlineable**

- 1 indicates that it is inlineable, and 0 indicates otherwise
- Value of 1 for all inline TVFs as well

If a scalar UDF is inlineable, *it does not imply that it will always be inlined*

- SQL Server will decide (on a per-query, per-UDF basis) whether to inline a UDF or not
- Definition runs into thousands of lines of code
- UDF in a GROUP BY clause

Decision is made when the query referencing a scalar UDF is compiled

Disabling Scalar UDF Inlining

Set database compatibility level less than 150

ALTER DATABASE SCOPED CONFIGURATION SET **TSQL_SCALAR_UDF_INLINING** = OFF;

OPTION (USE HINT('**DISABLE_TSQL_SCALAR_UDF_INLINING**'))

When defining the function:

```
CREATE OR ALTER FUNCTION dbo.discount_price(@price DECIMAL(12,2), @discount DECIMAL(12,2))
RETURNS DECIMAL (12,2)
WITH INLINE = OFF
AS
BEGIN
    RETURN @price * (1 - @discount);
END
```

Feedback requests

Regressions due to a feature?

Situations where something didn't kick off and you think it *should* have?

These features are in public preview – and we want your feedback!

Please email IntelligentQP@Microsoft.com

FAQ: Open problems (not a roadmap slide)

CE models – one-size-fits all? Or do we need feedback system?

Parameter sensitivity / optional parameter

Stats quality, richness, breadth

Intelligent QP resources…

Questions?

Don't forget to complete an online evaluation!

Query Processing Innovations

Your evaluation helps organizers build better conferences
and helps speakers improve their sessions.

Thank you!

Reminder: Intersect with Speakers and Attendees

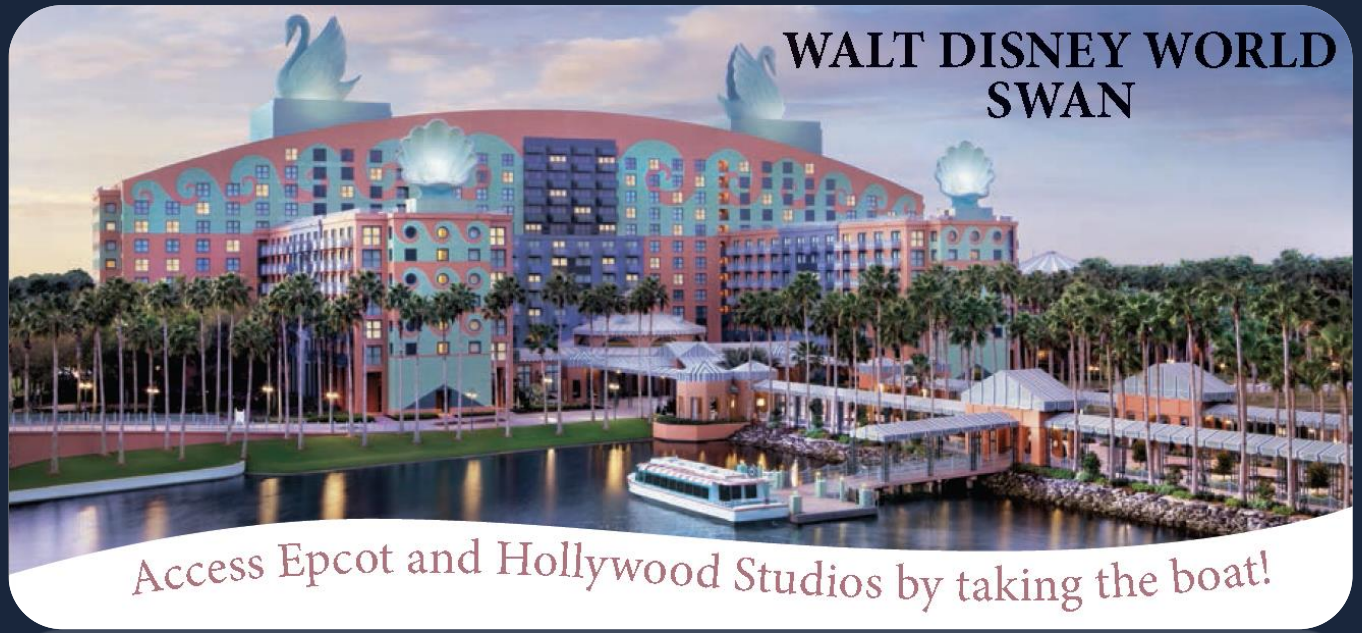
- **Tweet *tips and tricks* that you learn and follow tweets posted by your peers!**
  - Follow: #SQLintersection and/or #DEVintersection
- **Join us – Wednesday Evening – for SQLafterDark**
  - Doors open at **7:00 pm**
  - Trivia game starts at **7:30 pm**
    Winning team receives something fun!
  - Raffle at the end of the night
    Lots of great items to win including a seat in a five-day SQLskills Immersion Event!
  - The first round of drinks is sponsored by **SentryOne** and **SQLskills**

SQL intersection

SQLskills immerse yourself in sql server & SentryOne™

Save the Date!

www.SQLintersection.com



2019

June 9-14

We're back in Orlando!

SQL intersection

Leave the every day behind and enter a world of wonder and enchantment at the Walt Disney World® Resort. Located in the heart of the most magical place on earth, the Walt Disney World Swan and Dolphin Resort provides a truly extraordinary backdrop for our event! Beautiful tropical landscaping, tranquil waterways, and classic art and architecture work together to create a stunning landmark!