



Database Objects Under the Hood



Foreign Key Constraints

Usually benefit the system

- Ensure that data is valid and help catching the bugs
- Help Query Optimizer

Introduce some overhead during referential integrity check

- Usually not an issue unless system is very busy

Incompatible with some SQL Server features

- Partition switch

Create an index on referencing column(s)



Foreign Key Constraints

Demo



Unique and Check Constraints

Unique constraints

- Enforce uniqueness of the values in the column(s)
- Implemented as unique nonclustered indexes
- Do not allow to add INCLUDE columns

Check constraints

- Enforce domain integrity and quality of the data
- Help Query Optimizer
- Introduce overhead during data modifications



Check Constraints

Demo



Trusted Constraints

CHECK/FK constraints are either trusted or untrusted

- ALTER TABLE ADD CONSTRAINT WITH **CHECK/NOCHECK**

Query Optimizer does not use untrusted constraints

- Some exceptions in DW environment

SQL Server scans and locks the table (SCH-M lock) when you create trusted constraint



Regular Views

Metadata only

Helps to abstract database schema but..



Views: "Joins Hell"
Demo



Indexed Views

Data has been materialized on disk

Behavior is Edition-specific

- Non-Enterprise Editions – (NOEXPAND) hint is required
- Enterprise Edition – view can be used even when not referenced

Use-Cases

- Aggregations (DW), Join optimization (OLTP + DW), Optimization of 3rd party code

Consider update overhead in OLTP systems



Optimizing Code with Indexed Views

[Demo](#)



DML Triggers

Triggers degrade performance

ALTER UPDATE/DELETE triggers introduce index fragmentation

Avoid time consuming operations and external access

- Transactions are active and all locks are held
- Exceptions can roll back the transaction

CONTEXT_INFO / SESSION_CONTEXT (SS 2016) allows to pass parameter(s) to the triggers



Working with Triggers

Demo



DDL and Logon Triggers

DDL Triggers

- Use-cases:
 - Audit
 - Prevention of metadata changes in production
- Fired AFTER event

Logon Triggers

- Use-cases:
 - Limiting number of session per user
 - Custom Authentication
- Be very careful with external access from within Logon triggers



Functions

Code Reuse is very *questionable* pattern in T-SQL

Avoid multi-statement functions at all cost!

- Use schema binding when you cannot avoid them

For multi-statement table-valued functions SQL Server always estimates just one row in the result set

- Can introduce very bad execution plans (demo later)



Functions: The Good, the Bad and the Ugly

Demo

Replace CTE function



XML Data Type

Stores data in internal format (compressed UTF-16)

Untyped XML – any valid XML Data

- Less storage space
- Better insert/update performance
- Slower querying performance (harder to write efficient queries)

Typed XML – bound by XML Schema

- More storage space
- Schema validation overhead during insert/update
- Better querying performance (easier to write efficient queries)



XML Indexes

Primary XML Index

- SQL Server shreds XML to (very large) relational table
- Always used to query the data

Secondary XML Indexes

- Nonclustered indexes on Primary XML index table

XML Storage Space

```
<Order OrderId="42" OrderTotal="49.96" CustomerId="123"  
      OrderNum="10025" OrderDate="2013-07-15T10:05:20">  
  <OrderLineItem ArticleId="250" Quantity="3" Price="9.99"/>  
  <OrderLineItem ArticleId="404" Quantity="1" Price="19.99"/>  
</Order>
```

XML Storage Space

```
<Order>
  <OrderId>42</OrderId>
  <OrderTotal>49.96</OrderTotal>
  <CustomerId>123</CustomerId>
  <OrderNum>10025</OrderNum>
  <OrderDate>2013-07-15T10:05:20</OrderDate>
  <OrderLineItems>
    <OrderLineItem>
      <ArticleId>250</ArticleId>
      <Quantity>3</Quantity>
      <Price>9.99</Price>
    </OrderLineItem>
    <OrderLineItem>
      <ArticleId>404</ArticleId>
      <Quantity>1</Quantity>
      <Price>19.99</Price>
    </OrderLineItem>
  </OrderLineItems>
</Order>
```

XML Storage Space (65,536 rows)

	Clustered Index (KB)	Primary XML Index (KB)	Total Size (KB)
Untyped element-centric	28,906	90,956	119,862
Untyped attribute-centric	26,021	57,390	83,411
Typed element-centric	45,760	52,595	99,355
Typed attribute-centric	36,338	54,105	90,443

Untyped attribute-centric XML is easier to work with than element-centric XML



Working with XML Data Demo

Temporary Tables

```
create table #StageTbl(...);  
  
insert into #StageTbl(...)  
select ..  
from UserTbl  
where ..
```

**Temp Table Creation:
System Objects and
Allocation Maps changes**

**Writing Log Records to
tempdb log file**

**Logical or Physical Reads
from UserTbl**

**Writing Data Pages to
tempdb data file(s)**

**Extents and Pages
allocations for #StageTbl
data**

**Temp Table Deletion:
System Objects and
Allocation Maps changes**



Temporary Tables

Staging large amount of data is expensive

Good use-cases:

- Query simplification
- Improving cardinality estimations (especially with UDFs)
- Reducing locking time



Temporary Tables

Demo



Table Variables

Table Variables still use tempdb ☺

SQL Server does not keep statistics on TV and always estimates one row

- Could produce extremely inefficient plans with the large # of rows and joins
- Statement-level recompile can help to mitigate it to some degree
- T2453 (SS 2012 SP2, SS 2014 CU3) – tracks / recompile queries based on # of rows changed (still no histogram)

Table variables are not “transaction-aware”



Table Variables Demo



Temporary Table Caching

SQL Server truncates temporary tables/table variables keeping IAM and 1 data pages pre-allocated

- It significantly reduces the load on allocation map pages

Requirements

- No schema changes after creation (DROP TABLE is OK)
- No named constraints
- Less than 8MB in size



Temporary Table Caching

Demo



Table-Valued Parameters

TVP == Table Variable as T-SQL SP/Function parameter

One of the fastest ways to save the batch of the rows from the client

- Memory-Optimized TVPs are even faster!

Development challenges

- Client definition must match TVP schema
- No sql_variant support



Table-Valued Parameters

Demo



TempDB Best Practices

Reduce tempdb load

Put tempdb to the fastest disk array

Use –T1118 (disables Mixed Extents Allocations)

Maybe use –T1117 (uniform file growth)

Create multiple data files in tempdb

- <=8 logical CPUs: 1 file per CPU
- >8 logical CPUs: 8 files and add files in groups of 4 when needed
- All files should have the same size with auto-growth in MB

Utilize temporary object caching



CLR

T-SQL is better for

- Direct data access and manipulation in the database
- Set-based logic

CLR is better for

- Complex string manipulation
- Regular Expressions
- Advanced Math
- File Access
- Cryptography
- Access to external resources
- Logging outside of DB



CLR Performance: UDF

UDF – “Is integer even?”

- CLR without data access context: 167ms
- CLR with data access context: 246ms
- T-SQL scalar multi-statement UDF: 675ms
- T-SQL inline table-valued function: 18ms

UDF – “Distance between two points”

- CLR without data access context: 347ms
- T-SQL scalar multi-statement UDF: 1,955ms
- T-SQL inline table-valued function: 721ms



CLR Data Access Performance

Stored Procedure with 50,000 individual selects

- CLR stored procedure: 2,330ms
- T-SQL stored procedure: 410ms

Cursor vs. Data Reader (50,000 rows)

- CLR stored procedure: 116ms
- T-SQL stored procedure: 556ms

CLR Performance: Building CSV List

	T-SQL variable	CLR Aggregate	FOR XML
1,000 rows	1ms	3ms	<1ms
10,000 rows	129ms	12ms	3ms
25,000 rows	840ms	33ms	6ms
100,000 rows	535,040ms	146ms	43ms

Demos are available for download



CLR In General

32 bit OS: Uses memory in the process space (not AWE)

Security

- Breaks ownership chaining
- Can access external resources with appropriate rights
- Must be enabled on the target system – not really good for off-the-shelf software

Make sure CLR code “yields