A SQL Server Love Story

How 22 years at match.com prepared me for church work

Jeff McClure

Lakepointe Church, Rockwall TX

jeff.mcclure@lakepointe.church

https://github.com/mcdude8675309/RX2024_SQL

SQL Server - Prerequisites

- Coming into this, you should be familiar with:
    - Basic T-SQL Syntax
    - Basic Query patterns and Joins
    - Basic SQL Server Objects
    - SSMS or Azure Data Studio

    NOTE: There are multiple ways to do almost everything. Nothing I'm presenting is the gospel truth, just my perspective from my experience.

Agenda - Next 6-8 hours:

- Background - My SQL Server Love story
- Lakepointe Environment
- Tables
- Queries
- Indexes
- Performance
- Best Practices
- Demo
- Q & A

Background / History

How I got here:

- ☐ Skill
- ☐ Charisma / Charm
- ☐ Brains
- ☐ Luck
- ☐ Having all the answers
- ☐ Fate
- ✅ None of the above

But God…

"but GOD" is a holy conjunction that appears 48 times in the ESV translation of the Bible in both the Old and New Testaments. This phrase always appears after some statement of defeat or red alert emergency.

However, there was a problem…

Let's Dive In

Lakepointe Environment

- Production:
  - 1 Burstable Standard_B20ms VM
  - 1 Azure SQL Instance running between Premium P4 and P11
- Development:
  - 3 Separate Environments RockDev, RockBeta, RockTrain
    - VM: Standard B2MS
    - SQL: Standard S4 - 200 DTU
- Runbooks:
  - Automatic Scaling (Prod)
  - Automatic Startup/Shutdown (Dev)

SQL Server Generalities

- RDBMS - Relational Database Management System
- Data is relational - meaning that data in one table is often related to other tables through relationships called Foreign Keys.*
- Data is usually normalized to 3rd normal form
- Generally configured as case insensitive
- Amazingly fast and scalable
- Expensive

SQL Server On-Prem vs AzureSQL

- Benefits of going to AzureSQL:
  - Licensing cost is included
  - Always running the latest version
    - You can still control the compatibility mode
  - Enterprise edition
    - High-availability and Reliability
    - Online Indexing!
    - Automatic Tuning available (use caution)
    - Globally Redundant
  - Quickly Scalable - pay for what you need
  - Secure
  - Backup/Restore service is excellent and easy to manage
  - Integrated Performance tools
  - Read-Only Replica at no additional cost in Premium Tiers (demo)

SQL Server On-Prem vs AzureSQL

- Same Engine
- However Not all features available on-prem are available in AzureSQL like:

SQL Server Agent (Jobs / Chron Engine / Replication)	SQL CLR Integration
Extended Events - Exists with some limitations	Full Text Search Services
Cross-Database Queries / 3 Part Naming (i.e. rock.dbo.history)	Linked Servers
Windows Authentication	In-Memory OLTP (available in some tiers with limitations)
Resource Governor	Service Broker
SSRS - SQL Server Reporting Services	Database Mail

Tables

Tables and Data Storage

Basic data containers - like workbooks in a spreadsheet

- Rows and Columns
- Various keys (Primary, Foreign, and Unique)
- Since Rock is built on Entity Framework, most table records are basically Rock objects or entities with an identity (Id - PK) and many properties (columns)
- Stored in 8K data pages*

Tables - Structures

- Primary Keys
- Foreign Keys
- Indexes
- Heaps

Normalization and Denormalization

- "Normalize until it hurts, denormalize until it works." -Common Nerd Saying
- Here are some examples of denormalization in Rock:
  - The "Analytics" tables are heavily denormalized for read performance. As a result, there is an entire set of code built specifically for updating these tables.
  - Recent Rock denormalizations for performance
    - Person.Age - Birthdate alone requires a function to calculate - efficient to write, expensive to read. Storing Age is easy to write, efficient to read but expensive to maintain and guess what, it changes every year.
    - GroupMember.GroupTypeId - saves a join back to Group - often a large table thus improving read time at the expense of maintenance. Uses a Foreign Key to ensure Ref. Integrity.
    - AttributeValue - Persisted Values….
      - What if I manually update the value column with a date of 8/15/2024 but forget to update the ValueAsDateTime to the same?
      - Rock Cleanup Job - complexity necessary to keep data clean

TIP - Table Size / Data Size

- Exec sp_spaceused <tablename>
- Exec sp_spaceused Interaction


```
EXEC sp_spaceused Interaction
```

119 %

Results | Messages

	name	rows	reserved	data	index_size	unused
1	Interaction	56203572	33110736 KB	8067792 KB	21935904 KB	3107040 KB

Indexes

Indexes - What are they?

- Mini-tables attached to your base tables that the SQL can see, but you can't
- B-Tree structures
- Key to "Read Performance"
- An index key can contain up to 16 columns but can't exceed 900 bytes.

Example Script:

```
Create NonClustered Index IX_Person_Age_JMac_20240812
On dbo.Person (IsDeceased, Gender, Age Descending)
With(Online=On) On [Primary];
```

Indexes - Types

- Clustered - only 1 per table
- Nonclustered
- Unique - Can be clustered or nonclustered
- Covering - Not an actual type of index

Example of Covering Index:

```
Create NonClustered Index IXI_Person_Age_JMac_20240812
On dbo.Person (IsDeceased, Gender, Age Descending)
Include(Birthdate, NickName, LastName) With(Online=On)
On [Primary];
```

Indexes - Operations

- Seek
- Scan
- Key Lookup

```
Create NonClustered Index IXI_Person_Age_JMac_20240812
On dbo.Person (IsDeceased, Gender, Age Descending)
Include(Birthdate, NickName, LastName) With(Online=On)
On [Primary];
```

Simple Table / Index Example

Employee Table - Clustered PK on Id

Id	FirstName	LastName	Gender	DeskPhoneNum	BirthDate	Title
1	Alisha	Marble	Female	2614	7/1/1974	Chief Mugwump
2	Joe	James	Male	5477	8/4/1988	Driver
3	Sally	Hammer	Female	9874	6/2/1981	VP Operations
4	Fred	FlintStone	Male	8547	2/2/1968	Pin Setter
5	Barney	Rubble	Male	4958	1/8/1969	Cook
6	Wilma	FlintStone	Female	1255	6/1/1967	Engineer
7	Astro	TheDog	Male	NULL	NULL	Mascot

IX_Employee_Birthdate

Birthdate	Id
6/1/1967	6
2/2/1968	4
1/8/1969	5
7/1/1974	1
6/2/1981	3
8/4/1988	2
NULL	7

IX_Employee_LastName_FirstName

LastName	FirstName	Id
FlintStone	Fred	4
FlintStone	Wilma	6
Hammer	Sally	3
James	Joe	2
Marble	Alisha	1
Rubble	Barney	5
TheDog	Astro	7

IXI_Employee_Title_Incl_FirstName_LastName

Title	Id	FirstName	LastName
Chief Mugwump	1	Jeff	McClure
Cook	5	Barney	Rubble
Driver	2	Joe	James
Engineer	6	Wilma	FlintStone
Mascot	7	Astro	TheDog
Pin Setter	4	Fred	FlintStone
VP Operations	3	Sally	Hammer

Legend

Legend
KEY
POINTER
LEAF/DATA

Index - Compression

- Save Space - Disk AND Buffer
- Improves IO
- Cost: CPU
- Best for data with a lot of duplication
- Each index must be compressed independently
- Two Types: Row and Page

```
CREATE NONCLUSTERED INDEX IX_CreatedByPersonAliasId
ON dbo.Interaction (CreatedByPersonAliasId)
WITH (FILLFACTOR=90, ONLINE=ON, DATA_COMPRESSION=PAGE)
ON [PRIMARY];
```

Indexes - Why not index every column?

- Slower Writes
- 20 Indexes = 20 Inserts
- Disk Space / Memory Space
- Fragmentation

Indexes - The Best Part

- They have the biggest potential for solving a query performance problem - IF you know where to put them

- They can be added without affecting core code!

Query Performance

Query Performance

- "Query Optimizer" - Evaluates things like data distribution, indexes available, and system resources available to generate a query plan. That plan is then stored in the plan cache (for potential later use) and then sent to query engine to do the actual DML (Data Manipulation Language) work of Selecting, Inserting, Updating, Deleting, Merging, etc.
  - Can be Unpredictable or flat out wrong
  - Behavior can and often does change over time.
  - This is because your data evolves over time.
- Hints

Query Performance

"This report worked fine yesterday, but today it is timing out!"

- You're not crazy
- When this happens, your query plan has likely changed.
  - If there isn't a plan in the Procedure Cache, a new one gets created based on the currently available statistics.
- It could be a really gnarly EF query for which the optimizer just gave up.

Demo - The good stuff

- Execution Plans - A graphical representation of the query (Estimated or actual)
- Set Statistics IO On
- Timing a Query
- Index Dig - what indexes are on my table and are they being used?
- Joins
- Test for Read Only Status
- AttributeValue Performance Hack
- Windowing Function Example
- Scalar Function Demo
- Chunky Data Processing
- Explicit Transactions

Resources:

- The Rock Community! RocketChat
- AI Tools - super helpful for asking questions when no-one else seems to know.
- Training: https://www.iamtimcorey.com/
  - Tim actually attends Lakepointe with his family, and our very own [Rockstar] Jon Corey happens to know him really well.
  - Tons of free youtube content as well.
  - Limited Time Promo Code for 15% off the DevPass full access subscription for Rock Conference Attendees:
- Red-Gate Tools. https://www.red-gate.com/products/sql-search/
  - SqlSearch is Free
  - SQL Prompt is a great coding helper.
- https://ola.hallengren.com/
- https://www.brentozar.com/
- https://glennsqlperformance.com/

THE END - THANK YOU!

Slides and scripts can be downloaded here:

https://github.com/mcdude8675309/RX2024_SQL

Contact: jeff.mcclure@lakepointe.church

Best Practices and other Tips and Tricks Below

Along with all the things we didn't have time to discuss in the session

Tips - Performance Specific

- Use CTE's, #Temp Tables. There are some perf issues with Temp table Variables that are addressed in later versions of SQL Server.
- Use Indexed Columns in the Where clause and Joins
- Use your read-only replica to offload compute for reporting and analytics
  - Connection String includes: "ApplicationIntent=ReadOnly"
- Avoid "Select *" in production code.   Count(*) is okay!
  - Best used to see contents of a table or for basic troubleshooting
    - Limit to Top 100 or Top 1000:
      Select Top 100 *
      From dbo.Person
      Order by Id Desc

Tips - Performance Specific

- [Always] use Order By with a Top clause
  - As query plans change (and they do), the rows selected to be returned by a Top query may vary unless you specify the Order
- Parameterize your queries - this allows you to control the type of data that can be sent into and executed on your server. In other words, don't expose inputs that allow people (or bots) to pass in variable length strings and then concatenate that input into a query. ALWAYS scrub you inputs!
  - Rock Workloads seem like they would benefit from Forced Parameterization. I have turned this on but haven't noticed any significant differences, but it is hard to measure.

Tips - General

- Use Joins rather than Subqueries when possible.
  - They generally perform better, and can be easier to read and understand.
- Match DataTypes - esp. in the where clause. If the left column is an INT and the right column is an NVARCHAR, use explicit language to match them up. Same with variables. Yes, this may require wrapping a CAST or CONVERT function around a column, but sometimes we have to do it. Always cast from text to numbers rather than the other way around. It takes less compute to compare integers than letters. If you don't do it, SQL Server will implicitly convert for you.

Tips - General

- Comment your code. You likely won't be the last person to touch your code. Unless a 3rd grader could tell what you were doing, leave comments to help the next person out. Some code is super complex and difficult to read. Be kind to your future self and your future developers by helping them understand why you wrote that weird piece of logic.
  - /* This is a comment block and can span multiple lines */
  - --This is an inline comment and only affects things on the line that follow it
- Name your disposable tables in a way that you remember to clean them up! ie. `Person_Backup_MAYDELETE_AFTER_20241231`

Tips - General

- Stop Using *SQL Server Profiler*. Learn to use Extended Events instead. This is a great tool to watch calls coming into your server including the parameter values so you can run it yourself. This is my #1 goto for finding and troubleshooting a real-time performance issue. As with most things, use caution. (Let me know if you would like a session on just this at a future event.) Profiler doesn't work with AzureSQL anyway /shrug
- Other Useful perf tools:
  - Azure Monitor Tools
  - Dynamic Management Views (DMVs) (Glenn Barry)
  - Third Party Tools (Expensive)
  - APM Tools like NewRelic who offers 5 Free licenses for Nonprofits.

Tips - Save yourself some grief

- Use Explicit Transactions - esp when updating or deleting data manually

BEGIN TRAN

Do some work

Validate results - i.e. Row count

COMMIT or ROLLBACK

See Demo

Tips - General

- Adhere to the Rock Style Guide
  - https://triumph.slab.com/public/posts/%F0%9F%9A%A9-developer-codex-current-orig-y3t qutbv#sql-formatting
  - https://poorsql.com/
- Seek out good tools - if you spend a good deal of your time in SQL Server, you may benefit from a tool like Red-Gate SQL Prompt
- Only allow qualified people into your database. Don't give the fox access to the hen-house; even if he says he'll be good.

Tips - General

- AttributeValue Table (aka: the junk drawer)
  - Requires occasional cleanup - we purged about 15M records (15GB data) last year.
  - Use ValueCheckSum as performance aid.
    - Demo
- Study the "Greats"
  - Brent Ozar
  - Ola Hallagren
  - Jonathan Keyhais
  - Glen Barry
  - Many Others

Tips - General

- Stay Humble. Socialize with the community. Help where you can. Learn something new every day if possible.
- Test like crazy. You should always test your code yourself. Take responsibility for what you write and ensure that you have put it through its paces. Ask others to test with you. Automate unit testing were possible.

Tips - Functions you should know

CONCAT()	String concatenation. Use instead of "+". Gracefully handles NULL values
STRING_AGG()	Generates a delimited list of items and leaves off the pesky tailing delimiter
QUOTENAME()	Wraps output in square brackets
TRY_CAST()	Attempts to cast to your datatype of choice and returns NULL if not possible - super helpful for joins between AttributeValue and DefinedValue for GUIDS
ROWNUMBER() OVER()	Windowing function that makes getting Numbering output with a group.
sp_who2 [active]	Show all actively running SPIDs on your SQL Server

Keyboard Shortcuts - My Faves

F5, Alt-X, Ctrl-E	All run the selection or the whole script if nothing selected
Ctrl-R	Hide or Unhide the Results Pane
Ctrl-K-C, Ctrl-K-U	Comment/Uncomment (also works for blocks of text)
Shift-Delete	Deletes the current row in the editor and copies it into the clipboard
Alt-mouse-select	Allow duplicate typing on multiple rows

You can also do extensive customization and set useful keyboard shortcuts that execute certain commands or queries that make the mundane manageable.

Tips - Windowing Functions

Allow ranking and preaggregation to help you get the data you need in a simpler way than previously possible. Great video by Triumph here:
https://www.triumph.tech/videos/sql-window-functions

- Sum(), Avg(), Min(), Max()
- RowNumber() and Rank(), NTile() —>DEMO
- Lead(), Lag()

SQL Server - Things to avoid

- Don't wrap scalar functions around table columns in the where clause. This prevents the use of an index seek and instead forces a scan. (DEMO)
- Don't Overuse Triggers - They have their place, but can be easily abused by obscuring or hiding business logic that will leave you puzzled and frustrated trying to find out why something is happening.
  - Great for Auditing or enforcing something that you can't do any other way (Staff Email Address)
  - Can be confusing and scary
- Avoid Hardcoding values in queries - easy to do, hard to maintain (*guilty as charged*)
  - Use Parameters and Variables where possible
- Never ever Auto Shrink your database. There is a time and place for shrinking a database, but it should always be manual, and always be followed by a rebuild of all indexes because they will likely all be shredded with fragmentation.

SQL Server - Things to avoid

- Avoid modifying core database objects.
  - If you do, you may lose your [forgotten] changes in a future release and have to spend time troubleshooting. Worse yet, you have moved on and the person that took your place will have to figure it out.

Debatable…The Use of Cursors

- A cursor is a SQL Server construct with control logic for traversing and processing records from a dataset one row at a time. What you've heard: That's RBAR and RBAR is BAD. Use Set based queries instead. True for reading data, *not always true for updating/deleting*

- Cursors have their place and there are many types of them. When used, I almost always use an INSENSITIVE CURSOR which has very little overhead

- More often than not, I would load the data I want to process into a traditional #temp table, then process that data one record at a time with lightweight looping and control logic.

Debatable…The Use of Cursors Cont.

- The problem with large set based operations is that when you are modifying data either through inserts, updates, or deletes - all of it must be written to the transaction log in one big operation (slow) and you will likely end up blocking others from reading/writing from that table during the operation, and it will consume a lot of log space.

- Alternatives:
  - General Looping Logic (See Demo)
  - Example of an Insensitive cursor in the spDBARebuildIndexes stored proc

Stuff we don't have enough time to cover goes below this slide. Maybe next time…

SQL Server Best Practices - if self hosted:

- Use enterprise-grade hardware if possible
- Redundant NVMEs or SSDs
- 128GB RAM Minimum - go as big as you can afford after the Disks
  - If you are on-prem, max out your system RAM - minimum of 128GB (Std. Ed.). Enterprise and higher editions have a much higher limit. We had multiple servers running 2 TB of RAM at Match.
- Take Regular backups
  - Ola Hallaengren provides FREE masterful tools for this if self hosted
  - https://ola.hallengren.com/
- Test your backups (regardless of whether you are self-hosted)

Topics each worthy of a Conf. Session

- Extended Events
- Query Store
- Azure Performance Tools
- Database Tuning Advisor
- SQL Server Configuration and Database Options
- Encryption (esp. TDE)
- Forced Parameterization
- HA/DR
- How I accidentally brought down the match.com website by truncating a table in Production

Indexes - Suggested Naming

- Prefix + TableName + ColumnNames + Whodunnit + Date
  - Including a name and date will clearly distinguish it from core indexes
  - When indexes are created SQL Server doesn't record the date! Why?!?!
- Use underscore as delimiter for readability
- Prefixes:
  - Primary Key - PK
  - Nonclustered Index - IX
  - Unique Clustered Index - UCX
  - Unique NonClustered Index - UX
  - Nonclustered with Includes - IXI
  - Filtered Index - FIX
    *Note: Just try to be consistent ;)

Indexes - Less Common Types

- Filtered - NonClustered Only - Rare: Special Cases only.
  - A where clause on an index that limits it not only to certain columns but also certain rows.
- Columnstore Index - Not a B-Tree
- Spatial Index - Typically used for geographical spatial queries
- XML Index
- Full Text Index - Not available in AzureSQL