Agenda

- Who am I?
- Defining Zero Downtime for Databases
- Architecting Changes
- Adding Not Null Columns
- Splitting a column
- Rename a field
- Changing a Procedure
- Best Practices

Steve Jones

Advocate, Redgate Software
Editor, SQLServerCentral
he/him

32 years database experience

DBA, developer, manager, writer, speaker in a variety of companies and industries using SQL Server, Oracle, and other database platforms.

Founder, SQL Server Central

Currently the editor in chief, with the goal of helping you learn to be a better data professional every day

15-year Microsoft Data Platform MVP

I have been honored to be recognized by Microsoft for the as a Data Platform MVP working with SQL Server

/in/way0utwest @way0utwest sjones@sqlservercentral.com www.voiceofthedba.com

Visual Studio LIVE!

DEFINING ZERO DOWNTIME FOR DATABASES

Zero Downtime

- Your system is always up
- Or is it?
  - Can we tolerate delays?
  - Disturbances?
  - Retries?
- The goal is that your clients (or manager) doesn't see an interruption
- Architecture for making changes is important
  - Both application and database
  - Coordination is needed

Zero Database Downtime

- Usually, we mean zero downtime from the client's perspective
- There is no zero downtime for the database
  - Locks/blocks
  - Unavailable resources
- Our goal is to minimize the delays/disturbances/blocks
- We want to avoid throwing errors to applications
- We also want applications to retry
  - Avoid throwing errors to the user the first time
  - Be careful with timeouts – give the user a chance to retry

Making better decisions

ARCHITECTING ZERO DOWNTIME

Planning for Zero Downtime

- There is a pattern for improving database deployment impact
  - Expand/Contract model - https://martinfowler.com/bliki/ParallelChange.html
- Trade space for time
  - Use space in the database to store objects
  - Use this time to make changes outside of a deployment window

Use Good Development Practices

- Make everything backwards-compatible
- "Baby steps": If a change is backward-incompatible, split it into multiple backward-compatible steps.
- Ensure the database can rev 1 version without affecting the application
- Never add and delete objects in the same deployment
  - for the same object/structure

General Database Development Guidelines

- All INSERTs need to support new columns (use column list)
- NO RENAMES (use expand / contract for this)
- All procs/functions must only add parameters with defaults
- All new columns need defaults
- Never add and delete objects in the same deployment (for the same database object/structure)

General Database Development Guidelines

- App always calls for columns by names (no SELECT * or 1,2,3 columns)
- Feature flag all items dependent on database schema change
- “Baby steps”: If a change is backward-incompatible, split it into multiple backward-compatible steps.
- Use triggers to keep new and old structures in sync
- If in doubt, save copies of data until the deployment completes

Tracking data changes

ADDING NOT NULL COLUMNS

The Scenario

- Avoiding NULLs is often a good idea
- We want to add a new column to a table that is NOT NULL
- Two ways to do this (platform dependent):
- Two-step process
  - Add a NOT NULL column with a default
  - Fix existing rows
- Three-step process:
  - Add a NULL column
  - Update existing rows with some value
  - Change to NOT NULL

Adding NOT NULL columns

DEMO

Recommendations

- Feature Flag the application
- Use 3 Deployments
  - Add a NULL column
  - Update data/defaults
  - Change to NOT NULL
- Make sure you understand how the app behaves with NULL/default/incomplete data

Working towards normalization

SPLITTING A COLUMN

The Scenario

- We have a CustomerName column in dbo.Customers
- We want this split to two columns:
  - FirstName
  - LastName
- A common occurrence when we realize we've modeled incorrectly
- Multiple types of information are in one column
- We are becoming more normalized
- Useful for customization as well

The Typical Process

- We deploy this:
  - Add columns to the table: FirstName and LastName
  - UPDATE the columns by splitting the data in CustomerName
  - Drop the CustomerName column
- Challenges
  - We might have applications using CustomerName
  - The UPDATE statement might block users while running
  - We will not split some data properly (Georg von Trapp)

Splitting a column

DEMO

The Zero Downtime Process: Splits

- We trade space for time
- We use multiple deployments
  - 1 - Add columns to the table: FirstName and LastName
  - 2 - UPDATE the columns by splitting the data in CustomerName
  - 3 - Drop the CustomerName column
- Advantages
  - We can start this move independently of the application
  - We can batch the update if needed
  - We can delay the final step until ready

Splitting Tables

- The process is similar for splitting tables
- Add a new table, use triggers to sync data
- Remove the columns from the original table later

Merges

- A merge is the reverse of a split
- Can be table or column level
- Similar process
  - Merge data
  - If possible, keep the original values
  - Drops occur later

Don't do this, but if you must…

RENAMING

The Scenario

- Someone doesn't like a column name
- They want to change the name to something else
  - Currently dbo.OrderHeader.OrderDate
  - Change to: dbo.OrderHeader.OrderedbyDate
- Multiple options here (platform dependent):
  - Sp_rename
  - Multi-step process (add new, rename old, rename new, drop old)

Renames

DEMO

The Zero Downtime Process: Renames

- If possible, a quick one-step is preferred
- Test this to verify the code is correct
- Use sp_rename in SQL Server/Azure/Synapse
  - Only if you are sure all apps have rev'd
  - Verify this in testing
- Safer process is:
  - adding a new column/view
  - Use Xevents to check access to the old name
  - Drop the new col/view and rename the old one later

Altering your API

CHANGING A PROCEDURE

The Scenario

- There is a stored procedure in production
- We need to add functionality
  - New parameter
  - New field
- Things Not to Do
  - Remove fields in the result set
  - Remove/rename parameters

Enhancing a Stored Procedure

DEMO

The Zero Downtime Process: Changing Code

- Add in one deployment, remove in another
- Use defaults so old code works
- Verify logical functionality with/without default parameter
  - Without, must return same results are before the deployment
  - Use switching logic inside if necessary
- Log the calls without a parameter somewhere
  - This helps to determine when all clients have rev'd.

The things that work well

BEST PRACTICES

Best Practices for Applications

- Use Feature Toggles
- Use column names in reading result sets
- Ensure INSERTs use column lists
- Parameterize stored procedures and functions
- No SELECT *
- Use good error handling and log db errors for quick resolution

Best Practices for Database Code

- Break deployments up into stages
- Use defaults wherever possible
- No SELECT * in code (views/procs/functions)
- Use INSERT column lists (no INSERT..SELECT)
- Only add parameters in procs
  - Use defaults
- Avoid renames (aliases, synonyms, views can help here)
- Order of columns DOESN'T MATTER

Best Practices for Deployments

- Never add and drop in the same deployment (for related objects)
- Trade space for time, keeping copies of data for a period
- Write the cleanup code with the enhancement code
  - Test this together with the enhancement
  - Make a separate PR with a FUTURE DATE for deployment

Summary

- Understand and apply the DevOps principles to the db
- Learn what impacts your environment and what doesn't
- Use the impact analysis to decide where to split your deployments
- Be patient, make changes across time
- Never add and drop together
- Feature toggles make it easier to separate db from app changes

Session Survey

- Your feedback is very important to us
- Please take a moment to complete the session survey found in the mobile app
- Use the QR code or search for "Converge360 Events" in your app store
- Find this session on the Agenda tab
- Click "Session Evaluation"
- Thank you!

/in/way0utwest @way0utwest sjones@sqlservercentral.com www.voiceofthedba.com

The End


www.voiceofthedba.com


sjones@sqlservercentral.com


@way0utwest


/in/way0utwest