What’s In YOUR DB Wallet?

Kevin Feasel

SQL Saturday #217, Columbus

2013-06-08

Who Am I? What Am I Doing Here?

- Database Administrator
  - SQL Server DBA
  - SSIS developer
  - Currently working for Aetna
    - Standard employer disclaimer
  - Catallaxy Services
    - http://www.catallaxyservices.com
- Security Nut
- Cyclist
- Occasional world traveler

Setting Goals and Expectations

- This is a **beginner’s level** talk
  - Emphasis on design and brainstorming ideas
  - Very little code; mostly examples
  - Audience participation is paramount
    - Please share your experiences
    - I’m here to learn as well
- My goal: you can take this back to the office and start work on a new administrative DB (or improve an existing model)

Today’s Outline In Question Form

- **What** is an administrative database?
- **Why** should I have one?
- **For whom** should this exist?
- **Where** should I put the database?
- **When** should I update information?
- **How** should I design and implement it?
- **What else** could I use as an alternative?

What?

- Central, non-business database
- Breakdown by task:
  - **Analytics**: real-time or time-series health metrics
  - **Auditing**: tracking system changes
  - **Configuration**: central service configuration
  - **Maintenance**: regular processes to improve health
  - **Research**: scratch area for DBAs to analyze problems
  - **Reporting**: easy-to-understand presentation of Analytics, Auditing, and Maintenance processes
  - **Utilities**: useful tools and objects

What?

- Be sure to tailor this to your environment!
  - Audit trails for key applications
    - Especially if the app doesn't do this already!
  - Focus on high-value activities
  - Integrate with existing data sources

Why?

- Proof of Work
- Sandbox for DBAs
- Generate reports on system health
- Move from react-after-crash to predict-before-crash
- Collect an audit trail to prove compliance
- Save important “temporary” data
- Centralize frequently-used processes

For Whom?

- DBAs and database managers
- Developers
  - Utilities
  - Configuration
- Auditors
  - Audit trails for relevant activities
- High-level IT and business managers
  - System health dashboards

Where?

- Three primary options available
  - Central DB
  - Distributed databases
  - Central “collection” database with distributed “nodes”

Where? -- One Central DB

- Pros
  - Easy to report against
  - Easy to differentiate environments
  - Easy to maintain: simply update one database
- Cons
  - Some process necessary to link central DB to other instances
  - May need to re-write code to "launch" from central DB
  - If the central DB goes down, the entire monitoring system goes down
    - Who monitors the monitors?

Where? -- Distributed DBs

- Pros
  - Easier to develop: only need to care about local instance
  - Can write primarily in T-SQL without resorting to linked servers or OPENQUERY statements
- Cons
  - Reporting in SSRS more difficult
    - Dynamic expressions for connection strings
    - Difficult to string together multiple instances
  - Need to update N instances

Where? -- Central With Nodes

- Pros
  - Easier to develop: for distributed nodes, focus on local instance
  - Can develop distributed nodes using T-SQL without use of linked servers or OPENQUERY
  - Reporting easy in SSRS: report from central DB
- Cons
  - Some process necessary for ETL to central DB
  - More difficult to differentiate environments
  - Need to update N+1 databases (N nodes plus central DB)

Where? – Factors

- Developer good with SSIS → Central DB with nodes
- Linked servers already exist → Central DB
- Heavy customization per instance → Distributed DBs
- More instances → Central DB (or with nodes)
- Multi-instance reporting required → Central DB (or with nodes)

When?

- Push versus Pull
  - Push data into administrative database as part of normal operations
    - Database triggers
    - Stored procedures
    - Service Broker
    - Policy-Based Management
  - Pull data from other sources for later analysis
    - SQL Agent jobs
    - Analytics should typically be “pull” events

When? -- How Often To Pull?

- It depends!
- Factors which affect this answer:
  - Frequency of event
  - Importance of instance / application
  - Expected amount of data
  - Expected frequency of change
  - SLAs or other business agreements
  - Other applications which already handle certain metrics

When?

- Beware the observer cost
  - Gathering information isn’t free
    - CPU, RAM, and I/O costs for additional work
    - Gathering data may even cause problems with your line of business code
  - It is typically pretty cheap, though: reading metadata from DMVs and system tables are very fast and use few resources
  - Slower processes tend to be obvious

How?

- Basic design ideas:
  - Don’t act inframarginally
    - Find your needs and fill them first
  - Don’t re-invent the wheel
  - Use tools you already know
  - Design iteratively

How?

- Quick reminder of themes:
  - Analytics
  - Auditing
  - Configuration
  - Maintenance
  - Research
  - Reporting
  - Utilities

How? -- Analytics

- Wait stats
- Active sessions (real-time)
- Data and log file growth rates
- Virtual Log File (VLF) counts
- Service runs and failures
- Long-running jobs
- Backup optimization
- Deadlocks

How? -- Auditing

- Database triggers to track DDL changes
- Trace file readers looking for activity
- Policy-Based Management feeding into auditing tables
- Defining current permissions (or permission changes)

How? -- Configuration

- Centralized configuration for services (e.g., SSIS pre-2012)

How? -- Maintenance

- DBCC CHECKDB runs (and errors!)
- Backups, restorations, and restoration tests
- Index / statistics updates
- Environmental comparisons
  - Databases
  - Tables
  - Indexes
  - Triggers

How? – Reporting

- Daily status reports
- Alerts which don’t get picked up by SQL Agent alerting
- Management dashboards
- Frequency and type of errors on each instance
- Time elapsed for SQL Agent jobs
- Long-running jobs
- Space provisioning
- Frequency of events (e.g., autogrowth)
- SLA validation rules

Short version: WHATEVER THE BUSINESS REQUIRES!

How? -- Utilities

- Numbers/tally table: http://www.sqlservercentral.com/articles/T-SQL/62867/
- Date table
- Useful developer functions: T-SQL or CLR
  - Complex business calculations
  - Oft-used conversions

How? -- Re-Inventing The Wheel

- Don't (unless there's good reason to!)
- Analytics
  - Adam Machanic's Sp_WhoIsActive: http://tinyurl.com/spWhoIsActive
  - Devin Knight and Jorge Segarra's long-running Agent jobs: http://tinyurl.com/LongRunningAgentJobs
  - Nic Cain's backup optimization: http://sirsql.net/blog/2012/12/12/automated-backup-tuning
  - Paul Randal's wait stats: http://tinyurl.com/SQLWaitStats

How? -- Reinventing The Wheel

- Maintenance
  - Olla Hallengren's maintenance scripts: http://ola.hallengren.com/
  - Michelle Ufford's index defragmentation script: http://sqlfool.com/2011/06/index-defrag-script-v4-1/

How? -- Reinventing The Wheel

- Utilities
  - Aaron Bertrand’s sp_foreachdb: http://tinyurl.com/spForEachDB
  - Kim Tripp’s duplicate index check: http://www.sqlskills.com/blogs/kimberly/removing-duplicate-indexes/
- Also, embrace and extend
  - Modify existing code to fit your environment
  - Build on what others have done

How? Tools To Use

- T-SQL
  - System stored procedures (sp_spaceused)
  - Dynamic Management Views (wait stats)
  - Database and table listings
  - Permissions (sys.fn_my_permissions)
- Extended events: read resulting XML & insert into tables
- SSIS
- Powershell (via SQLPSX)
- Service Broker
  - Line of business apps feed data into administrative DB

What Else?

- Analytics
  - SQL Server’s built-in reports: free, but limited
  - Activity monitor: OK GUI and free, but limited presentation capabilities
  - Third-party monitors: typically very feature-rich, but can be expensive and sometimes difficult to get data
  - Management Data Warehouse: free and fairly easy to understand

What Else?

- Auditing
  - Third-party logging: GUI for auditing and can be feature-rich, but typically fairly costly per instance
- Maintenance
  - Maintenance plans: too easy to do stupid things (e.g., regular DB shrinking)
  - SSIS-based maintenance: maintenance plans on steroids
- Utilities
  - Put in master DB: easy to call (especially if prefixed with sp_) but can clutter master

Fin