SharePoint for the DBA

Régis Baccaro
Introduction

Régis Baccaro

@regbac

http://Theblobfarm.wordpress.com
http://Thelovefarm.wordpress.com

regis@baccaro.com

• Founder and lead organizer of SQL Saturday Denmark
• Works for IBM
• Passionate about the community
• .Net developer, BI guy, SharePoint fellow and accidental DBA
Why SharePoint at a SQL Server event?

- SharePoint heavily depends on SQL Server
- DBA are from Venus and SharePoint admins from Mars
- You need **coordination** to make it work!
Agenda

- Introduction
- SharePoint Architecture & Design Considerations
- Planning SQL for SharePoint
- Deployment/Configuration/Security
- Maintenance
- High Availability
Introduction

• What is SharePoint?
• Why is SQL Server so important?
Introduction – What is SharePoint?

- Single business productivity platform leading to common
  - End-user Experience
  - Rich Integrated Capabilities
  - Toolset and Development
  - Deployment and Management
SharePoint through the ages
SharePoint

- Very fast
- Broadly adopted
- Very adaptive
Design paradigm

- Single data platform
  - Web Content Management
    - Mostly READ
    - Structured queries and Search
  - Enterprise Content Management
    - 80/20 READ/WRITE
    - Ad-hoc queries

- Upgrade and patch management
  - Requires consistency and integrity

- Application logic expectations on schema

- Enforced integrity and constraints
Deployment

Different options for installing SharePoint:

• Basic Installation – Uses SQL Express
• Wizard Installation – SQL Server but no control over naming
• UI Install – SQL, some control over naming but open to human error
• PowerShell – Significant control over naming and advanced planning
• Pre-Create Databases – Control over naming and database settings
SharePoint Database design

• Types of Database
  • Administration
  • Content
  • Service Applications
• How many databases will be required?
• How large do they need to be?
• How fast will they grow?
Always install an instance of SQL Server that will be dedicated to SharePoint

SQL instance for SharePoint should be a non-default instance

Make sure you use a SQL alias

Max. Degrees of Parallelism must be set to 1

For all SharePoint DB's, set the Default Collation setting to: Latin1_General_CI_AS_KS_WS
System Database Modifications that must be performed

- **TempDB**
  - Increase its initial size. Try starting with 500 MB
  - Set Autogrowth to use MB, not %
  - Autogrowth value should be larger, not smaller. Something like 500 MB should be a good starting value.
  - Put on the fastest available non-system drive. Separate the files – one per core on the server (max 8 files)

- **ModelDB**
  - Initial size should be set to 500 MB as a good start
  - Set to full recovery mode
  - Set Autogrowth to use MB, not %
  - Autogrowth should be set to something like 250 MB
  - Initial log size should be 25% of MDF
  - Log growth should set to something like 70 MB
• Each web application has at least one site collection.
• Each site collection is contained in a single content database.
• Each content database can contain many site collections.
• If the site collection is greater than 100GB, move to its own content database.
• Max size of content collection = 200GB.
• Size of site collection depends on the amount of content.
• Configure temp DB to be at least 10% of total Content DB size.

**Database size = \(((D \times V) \times S) + (10 \text{ KB} \times (L + (V \times D)))\)**

**Database size = \(((200,000 \times 2)) \times 250) + ((10 \text{ KB} \times (600,000 + (200,000 \times 2)))) = 110,000,000 \text{ KB}**
## Management Databases

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Growth</th>
<th>Read/Write</th>
<th>Scaling</th>
<th>Recovery Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharePoint_Config</td>
<td>Small</td>
<td>Slow</td>
<td>90/10</td>
<td>Up</td>
<td>Full</td>
</tr>
<tr>
<td>SharePoint_Admin_Content</td>
<td>Small</td>
<td>Slow</td>
<td>90/10</td>
<td>Up</td>
<td>Full</td>
</tr>
</tbody>
</table>
### Search Databases

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Growth</th>
<th>Read/Write</th>
<th>Scaling</th>
<th>Recovery Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search_Service_Application_DB_&lt;Servername&gt;</td>
<td>10GB</td>
<td>Many factors determine growth</td>
<td>80/20</td>
<td>Mostly Up</td>
<td>Simple</td>
</tr>
<tr>
<td>Search_Service_Application_AnalyticsReportingStoreDB_&lt;Servername&gt;</td>
<td>Medium to Large</td>
<td>Many factors determine growth</td>
<td>90/10</td>
<td>Up</td>
<td>Simple</td>
</tr>
<tr>
<td>Search_Service_Application_CrawlStoreDB_&lt;Servername&gt;</td>
<td>Medium</td>
<td>Depends on the amount of content</td>
<td>90/10</td>
<td>Out</td>
<td>Simple</td>
</tr>
<tr>
<td>Search_Service_Application_LinkStoreDB_&lt;Servername&gt;</td>
<td>Medium to Large</td>
<td>Depends on multiple factors</td>
<td>90/10</td>
<td>Out</td>
<td>Simple</td>
</tr>
</tbody>
</table>
## User Profile Databases

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Growth</th>
<th>Read/Write</th>
<th>Scaling</th>
<th>Recovery Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application_ProfileDB</td>
<td>1MB per profile</td>
<td>Multiple factors</td>
<td>90/10</td>
<td>Up per Service App</td>
<td>Simple</td>
</tr>
<tr>
<td>Application_SyncDB</td>
<td>630KB per user</td>
<td>Multiple factors</td>
<td>50/50</td>
<td>Up per Service App</td>
<td>Simple</td>
</tr>
<tr>
<td>Application_SocialDB</td>
<td>0.009MB per tag</td>
<td>Depends on the amount of activity</td>
<td>80/20</td>
<td>Up per Service App</td>
<td>Simple</td>
</tr>
</tbody>
</table>
## Other Databases

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Growth</th>
<th>Read/Write</th>
<th>Scaling</th>
<th>Recovery Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure_Store_Service_DB</td>
<td>5MB per 1000 credentials</td>
<td>Slow, depends on auditing</td>
<td>80/20</td>
<td>Up</td>
<td>Full</td>
</tr>
<tr>
<td>SessionStateService</td>
<td>1GB</td>
<td>Depends on usage of other services</td>
<td>80/20</td>
<td>Out</td>
<td>Full</td>
</tr>
<tr>
<td>Bdc_Service_DB</td>
<td>Small</td>
<td>Very Slow</td>
<td>90/10</td>
<td>Up</td>
<td>Full</td>
</tr>
<tr>
<td>PerformancePoint Service</td>
<td>1GB</td>
<td>Very slow</td>
<td>80/20</td>
<td>Up per Service App</td>
<td>Full</td>
</tr>
</tbody>
</table>
Sizing and Architecture

SQL Server Best Practices for SharePoint
## Configuration: Typical Deployment Sizes

<table>
<thead>
<tr>
<th>Metric</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content db size</td>
<td>&lt; 50GB</td>
<td>50GB</td>
<td>&gt; 50GB</td>
</tr>
<tr>
<td># of Content dbs</td>
<td>&lt; 20</td>
<td>20</td>
<td>&gt; 20</td>
</tr>
<tr>
<td># of concurrent requests to SQL</td>
<td>&lt; 200</td>
<td>200</td>
<td>&gt; 200</td>
</tr>
<tr>
<td>User</td>
<td>&lt; 1000</td>
<td>1000</td>
<td>&gt; 1000</td>
</tr>
<tr>
<td># of items in regularly accessed list</td>
<td>&lt; 2000</td>
<td>2000</td>
<td>&gt; 2000</td>
</tr>
<tr>
<td># of columns in regularly accessed list</td>
<td>&lt; 20</td>
<td>20</td>
<td>&gt; 20</td>
</tr>
</tbody>
</table>
## Configuration: Recommended (Minimum) Capacities

<table>
<thead>
<tr>
<th>Resource</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minimum</strong> DB server memory</td>
<td>4 GB</td>
<td>8 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>Processor L2 cache</td>
<td>2 MB</td>
<td>&gt; 2 MB</td>
<td>&gt; 2 MB</td>
</tr>
<tr>
<td>Bus bandwidth</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Disks latencies (msec)</td>
<td>&lt; 20</td>
<td>&lt; 10</td>
<td>&lt; 10 (data)</td>
</tr>
<tr>
<td>Network</td>
<td>Gigabit</td>
<td>Gigabit</td>
<td>Gigabit</td>
</tr>
<tr>
<td>Network latency (msec)</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
</tr>
</tbody>
</table>
Configuration : Best practices

• Install latest Service Packs & CUs
• Use dedicated SQL Server for medium/large deployments
• Consider scaling-out Content DBs as deployment grows beyond 4-8 TB (depending on usage model)
  • Some SharePoint databases must be scaled up, example: Config DB
• Use connection alias
  • Simplifies redirecting WFEs to a different database instance
Software Boundaries

• “Hard” Limits.....there are none!
• “Soft Limits”
  • 8 WFEs to 1 SQL Server instance
  • 50,000 site collections per content database
  • 100 GB of data per content database
• Latency < 1ms between Web and database servers
Prioritizing Database Volume

• Recommended database placement priority (fastest to slowest drive)
  1. Temp DB data and t-log files
  2. DB t-log files
  3. Search DB data files
  4. Content DB data files
  5. Profile DB data files

• Place temp DB, Content DB and t-logs on separate LUNs
• Use multiple data files for Content and Search DBs
• Place SharePoint Search crawl & query processing tables on separate spindles
SQL Server TempDB data files

• Data files allocated = number core CPUs in SQL Server (up to 8)
• Data file sizes consistent across all data files
• Data files spread across unique LUNs
  • Separated from Content DB, Search DB, etc.
• Log file separated to unique LUN
- Optimal TempDB data file size formula:

\[
\text{MAX DB SIZE (KB)} \times 0.25 \div \# \text{ CORES} = \text{DATA FILE SIZE (KB)}
\]

- Result (starting size) should be roughly equal to 25% largest content or search DB or 10% of total content DB size

- Use RAID 10; separate LUN from other database objects (content, search, etc.)

- “Autogrow” feature set to a fixed amount; if auto grow occurs, permanently increase size
SQL Server memory

• Set ‘Max Server Memory’

\[
\text{SQL Max Memory} = \text{TotalPhyMem} - (\text{NumOfSQLThreads} \times \text{ThreadStackSize}) - (1\text{GB} \times \text{CEILING}(\text{NumOfCores}/4))
\]

\[
\begin{align*}
\text{NumOfSQLThreads} &= 256 + (\text{NumOfProcessors}^2 - 4) \times 8 \\
\text{ThreadStackSize} &= 1\text{ MB on x86} \\
&\quad 2\text{ MB on 64-bit (x64)} \\
&\quad 4\text{ MB on 64-bit (IA64)}
\end{align*}
\]

• Or the easy way: leave 2-3GB for the OS
**Storage – Recommended I/O Capacities**

- Disk/Sec Transfer
  - Data Files > 10 ms
  - Transaction Logs > 5 ms

<table>
<thead>
<tr>
<th>Type</th>
<th>RAID level</th>
<th>IOPS</th>
<th>SAN Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>tempdb</td>
<td>RAID-10</td>
<td>2 IOPS/GB</td>
<td>Write optimized</td>
</tr>
<tr>
<td>Transaction Logs</td>
<td>RAID-10</td>
<td>2 IOPS/GB</td>
<td>Write optimized</td>
</tr>
<tr>
<td>Search Database</td>
<td>RAID-10</td>
<td>2 IOPS/GB</td>
<td>Read/Write optimized</td>
</tr>
<tr>
<td>Content Databases</td>
<td>RAID-10*</td>
<td>0.75 IOPS/GB</td>
<td>Read optimized</td>
</tr>
</tbody>
</table>

* Raid-5 can be used for static web content
SharePoint container Topology

- SPFarm
  - SPWebApplication (1:n)
    - SPSite (1:n)
      - SPWeb (1:100K)
        - SPList (1:10K)
          - SPFolder (1:10K)
            - SPListItem (1:100K)
SharePoint Data Architecture

- Flexible, user extensible types
  - Announcement, Contacts, Document Types
  - Support 10th of a million types in a single DB
  - A few types that may have 100’s of properties
- Millions of instances of multiple types in a list
- Efficient display of “all items in a folder”
- End-user queries over multiple lists in multiple sites which is mapped onto a single table
### Content DB Architecture

#### Namespace Table

<table>
<thead>
<tr>
<th>Id</th>
<th>StId</th>
<th>DirName</th>
<th>LeafName</th>
<th>WebId</th>
<th>ListId</th>
<th>DocIdRowId</th>
<th>Other Metadata</th>
</tr>
</thead>
<tbody>
<tr>
<td>1..64</td>
<td>1..32</td>
<td>1..8</td>
<td>1..16</td>
<td>1..12</td>
<td>1..8</td>
<td>1..16</td>
<td>~35</td>
</tr>
</tbody>
</table>

#### Userdata Table

<table>
<thead>
<tr>
<th>nvchar</th>
<th>ntext</th>
<th>sql_variant</th>
<th>int</th>
<th>float</th>
<th>datetime</th>
<th>bit</th>
<th>Other metadata</th>
</tr>
</thead>
</table>

#### Sites

<table>
<thead>
<tr>
<th>Id</th>
<th>Quota</th>
<th>Other Metadata</th>
</tr>
</thead>
</table>

#### Container Tables

<table>
<thead>
<tr>
<th>SiteId</th>
<th>Id</th>
<th>Url</th>
<th>Title</th>
<th>ScopeId</th>
<th>Metadata</th>
</tr>
</thead>
</table>

#### Webs

<table>
<thead>
<tr>
<th>SiteId</th>
<th>Id</th>
<th>Url</th>
<th>Title</th>
<th>ScopeId</th>
<th>Metadata</th>
</tr>
</thead>
</table>

#### AllLists

<table>
<thead>
<tr>
<th>WebId</th>
<th>Id</th>
<th>Title</th>
<th>ItemCount</th>
<th>ScopeId</th>
<th>Fields</th>
<th>Metadata</th>
</tr>
</thead>
</table>

#### Userdata table

<table>
<thead>
<tr>
<th>nvchar</th>
<th>ntext</th>
<th>sql_variant</th>
<th>int</th>
<th>float</th>
<th>datetime</th>
<th>bit</th>
<th>Other metadata</th>
</tr>
</thead>
</table>

#### Other Metadata

- 1...64
- 1...32
- 1..8
- 1..16
- 1..12
- 1..8
- 1..16
- ~35
• SharePoint tables are too wide, wraps rows
• SharePoint manages its own indexes
• SharePoint adds force-order, query hints
• Missing indexes for common operations
• Excessive use of Dynamic queries
• No SQL Referential Integrity OR key constraints
• DBCC with data loss
• Use of @table variables
• Lack of consistency checker
• DB Connect failures
• Missing integration of Back-up/Restore
SharePoint Schema

• Demo !
SharePoint maintains its own index

- Multiple types in the same table = untenable SQL Indexing
- Design challenge:
  - How do I put a SQL Index for a given property in all instances of a given type?
  - Do you really suggest 1000+ of index on a table?
- Solution
  - Maintain Name-Value pairs and index NVP table
### SharePoint querying indexed lists

```sql
EXECUTE READER(System.Data.CommandBehavior)

SELECT TOP 101 t1.[Type] AS c0, UserData.[tp_ContentTypeId], UserData.[tp_ID], t1.[TimeCreated] AS c4, UserData.[tp_CopySource], UserData.[float2], UserData.[nvarchar4], UserData.[tp_Created], CASE WHEN DATALENGHT(t1.[DirName]) = 0 THEN t1.[LeafName] WHEN DATALENGHT(t1.[LeafName]) = 0 THEN t1.[DirName] ELSE t1.[DirName] + N'\' t1.[LeafName] END AS c1, UserData.[tp_ModerationStatus], UserData.[tp_Level], UserData.[nvarchar1], UserData.[tp_HasCopyDestinations], UserData.[tp_HasAttachment], t1.[LeafName] AS c3, UserData.[nvarchar2], t1.[DirName] AS c5, UserData.[tp_ContentTypeId], UserData.[int1], t2.[nvarchar1] AS c6, t1.[ScopeId] AS c7, t1.[ScopeId] AS c8, UserData.[tp_UIVersion], UserData.[nvarchar3], UserData.[float1] FROM NameValuePair_Latin1_General_CI AS Nvp WITH (INDEX=NameValuePair_Latin1_General_CI AS_CI) INNER LOOP JOIN UserData WITH(NOLOCK) ON Nvp.SiteId = UserData.tp_SiteId AND Nvp.ListId = UserData.tp_ListID AND Nvp.ItemId = UserData.tp_ID AND Nvp.Level = UserData.tp_Level AND Nvp.SiteId = @L2 AND Nvp.ListId = @L4 AND Nvp.FieldId = @L5 INNER LOOP JOIN Docs AS t1 WITH(NOLOCK) ON (1 = 1 AND UserData.[tp_RowOrdinal] = 0 AND t1.[SiteId] = UserData.tp_SiteId AND t1.[SiteId] = @L2 AND t1.[DirName] = UserData.tp_DirName AND t1.[LeafName] = UserData.tp_LeafName AND t1.[Level] = UserData.tp_Level AND t1.[IsCurrentVersion] = 1 AND (1 = 1)) LEFT OUTER JOIN AllUserData AS t2 WITH(NOLOCK, INDEX=AllUserData_PK) ON (UserData.[int1] = t2.[tp_ID] AND UserData.[tp_RowOrdinal] = 0 AND t2.[tp_RowOrdinal] = 0 AND (t2.[tp_IsCurrent] = 1)) AND t2.[tp_CalculatedVersion] = 0 AND t2.[tp_DeleteTransactionId] = 0x AND t2.[tp_ListId] = @L3 AND UserData.tp_ListId = @L4 WHERE (UserData.tp_IsCurrent = 1) AND UserData.tp_SiteId = @L2 AND (UserData.tp_DirName = @DN) AND UserData.tp_RowOrdinal = 0 AND (Nvp. Value > N'Product 100000 - some content - 9') AND t1.[SiteId] = @L2 AND (t1.[DirName] = @DN)) ORDER BY UserData.[nvarchar3] Asc, UserData.[tp_ID] Asc OPTION (FORCE ORDER)
```
Recap

• We have
  • Wide table with no type/app awareness
  • Row wrapping = multi-row objects
  • Name-Value auxiliary table providing app-level index

• Result
  • SQL Server knows very little about SharePoint App semantics!
  • Can’t afford to rely on QO/QP to do it right
    • Query over little list followed by a large library
    • Query that join from NVP index to content table
Maintenance

SQL Server Best Practices for SharePoint
SharePoint DB Health monitoring

• Monitor SQL Server performance regularly
  • Use SQL Server DMVs
  • Use Recommended Perfmon counters
  • Allocate extra disk space for diagnostics information

• Check integrity of the database routinely

• DBCC CHECKDB
  • Can use REPAIR_REBUILD option to fix errors (not always possible)
  • REPAIR_ALLOW_DATA_LOSS not supported
  • Time consuming operation, run during non-peak hours
SharePoint DB maintenance

• Does SharePoint maintain indexes?
• Use DBCC CheckDB (REPAIR_ALLOW_DATA_LOSS)
• Set of Rules:
  • Databases used by SharePoint have fragmented indices
  • Search – One or more property databases have fragmented indices.
  • Search - One or more crawl databases may have fragmented indices
• Define a maintenance plan
SharePoint Databases maintenance considerations

• Fragmentation occurs by design on SharePoint ;-)  
• Increase space utilization & I/O → degrades performance  
• Content and Search dbs most susceptible  
• Rebuild / Reorganize indexes to eliminate fragmentation  
• Use sys.dm_db_index_physical_stats to measure  
  • More accurate than DBCC SHOWCONTIG, often reports higher fragmentation numbers  
• Use a framework like Ola Halengren’s
SharePoint Databases maintenance considerations

• Do’s
  • Auto-defrag only available for content databases
  • Only shrink content databases, not others
  • Only perform if free space > 50% (after content reorg)
  • Do not perform as part of maintenance plan
  • Perform during off-peak hours (resource intensive)
  • Update statistics – don’t rely on the timer service
  • Use DBCC SHRINKDATABASE or DBCC SHRINKFILE
  • Have reliable backups for all databases before implementing maintenance operations
  • Check for and repair consistency errors by using DBCC CHECKDB
  • Change the server-wide fill factor setting to 70
SharePoint Databases maintenance considerations

• Don'ts
  • Drop and re-create indexes
  • Rebuild indexes or run consistency checks during business hours
  • Set fill factor for individual tables or indexes
  • Shrink any databases other than content databases
  • Auto-shrink databases
  • Shrink databases at all unless you really need to
Schema modifications constraints

- Adding database triggers
- Adding new indexes or changing existing indexes within tables
- Adding, changing, or deleting any primary or foreign key relationships
- Changing or deleting existing stored procedures
- Calling existing stored procedures directly, except as described in the SharePoint Protocols documentation
- Adding new stored procedures
- Adding, changing, or deleting any data in any table of any of the databases for the products that are listed in the "Applies to" section
- Adding, changing, or deleting any columns in any table of any of the databases for the products that are listed in the "Applies to" section
- Making any modification to the database schema
- Adding tables to any of the databases for the products that are listed in the "Applies to" section
- Changing the database collation
- Running DBCC_CHECKDB WITH REPAIR_ALLOW_DATA_LOSS (However, running DBCC_CHECKDB WITH REPAIR_FAST and REPAIR_REBUILD is supported, as these commands only update the indexes of the associated database.)
- Enabling SQL Server change data capture (CDC)
- Enabling SQL Server transactional replication
- Enabling SQL Server merge replication
High Availability

SQL Server Best Practices for SharePoint
High Availability

• High availability options supported:
  • Database mirroring
  • Failover Clustering (local & stretch clusters)
  • Log Shipping
  • Always On

• Database Mirroring is by far the most popular HA/DR solution but Always On is catching up!

• Virtualization of hosts servers
Cluster of clusters
- Shared disks rather than standalone
- Instance redundancy
- Resist to up to 3 machine failures

Cluster with 2 nodes
- Single node failure will result in lost mirroring
Pros And Cons - RPO and RTO comparison based on database technology

<table>
<thead>
<tr>
<th>SQL Server solution</th>
<th>Potential data loss (RPO)</th>
<th>Potential recovery time (RTO)</th>
<th>Automatic failover</th>
<th>Readable secondaries (Not supported by SharePoint)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlwaysOn Availability Group (synchronous-commit)</td>
<td>Zero</td>
<td>Seconds</td>
<td>Yes</td>
<td>0 - 2</td>
</tr>
<tr>
<td>AlwaysOn Availability Group (asynchronous-commit)</td>
<td>Seconds</td>
<td>Minutes</td>
<td>No</td>
<td>0 - 4</td>
</tr>
<tr>
<td>AlwaysOn Failover Cluster Instance</td>
<td>Does not apply An FCI itself does not provide data protection. The amount of data loss depends on the storage system implementation.</td>
<td>Seconds to minutes</td>
<td>Yes</td>
<td>Does not apply</td>
</tr>
<tr>
<td>Database mirroring - High-safety (synchronous mode + witness server)</td>
<td>Zero</td>
<td>Seconds</td>
<td>Yes</td>
<td>Does not apply</td>
</tr>
<tr>
<td>Database mirroring - High-performance (asynchronous mode)</td>
<td>Seconds</td>
<td>Minutes</td>
<td>No</td>
<td>Does not apply</td>
</tr>
<tr>
<td>Backup, copy, restore</td>
<td>Hours or zero if the tail of the log can be accessed after the failure.</td>
<td>Hours to days</td>
<td>No</td>
<td>Not during a restore</td>
</tr>
</tbody>
</table>
Stretched Farm

• Requires:
  • Highly consistent intra-farm latency of <1ms (one way),
  • 99.9% of the time over a period of ten minutes,
  • The bandwidth speed must be at least 1 gigabit per second.
Tack Så Mycket

Régis Baccaro
@regbac
Régis Baccaro

@regbac
http://Theblobfarm.wordpress.com
http://Thelovefarm.wordpress.com
regis@baccaro.com

- Founder and lead organizer of SQL Saturday Denmark
- Works for IBM
- Passionate about the community
- .Net developer, BI guy, SharePoint fellow and accidental DBA