Beginning Admin
The Care and Feeding of SQL Server

Jennifer McCown

MICROSOFT CERTIFIED MASTERS

FREE TRAINING VIDEOS AT MIDNIGHTDBA.COM

MIDNIGHTSQL CONSULTING, LLC

MINIONWARE, LLC

SQL SERVER MANAGEMENT SOFTWARE

FREE INDEX MAINTENANCE

FREE SQL SERVER BACKUP

SEAN

JEN

Outline – The Big Five

- Backups
- Integrity Checks
- Index Maintenance
- Disk Management
- Alerting

But what about “performance”?

First, a little context for the Big Five

- These are critical, yet overlooked
- **Anything** is better than nothing
- But, let’s put in some effort
- (Maintenance plans suck)

90
IEC II / TYPE II HIGH POSITION
SUPER AVILYN CASSETTE SUPER HIGH RESOLUTION
B:
SUPER PRECISION ANTI-RESONANCE CASSETTE MECHANISM

Backups

Why back up?

- **Backups make a copy of the data or log records**
- Good for disasters
- And other recovery
- And making copies for testing

Databases have…

- One or more **data files**
- A **transaction log file**
- A **recovery mode** (we’ll stick with Full today)

Different Kinds of Backups

Backup Type	What it Does
Full	Backs up data (and active part of the log)
Differential	Backs up all changes since last full backup
Log	Backs up everything that happened in the transaction log *since the last log backup*

Backup Syntax

Backing Up a Whole Database BACKUP DATABASE { database_name | @database_name_var } TO <backup_device> [,...n] [<MIRROR TO clause>] [next-mirror-to] [WITH { DIFFERENTIAL -- Not supporterd in SQL Database Managed Instance | <general_WITH_options> [,...n] }] [;] Backing Up Specific Files or Filegroups BACKUP DATABASE { database_name | @database_name_var } <file_or_filegroup> [,...n] TO <backup_device> [,...n] [<MIRROR TO clause>] [next-mirror-to] [WITH { DIFFERENTIAL | <general_WITH_options> [,...n] }] [;] Creating a Partial Backup BACKUP DATABASE { database_name | @database_name_var } READ_WRITE_FILEGROUPS [, <read_only_filegroup> [,...n]] TO <backup_device> [,...n] [<MIRROR TO clause>] [next-mirror-to] [WITH { DIFFERENTIAL | <general_WITH_options> [,...n] }] [;] Backing Up the Transaction Log (full and bulk-logged recovery models) BACKUP LOG -- Not supported in SQL Database Managed Instance { database_name | @database_name_var } TO <backup_device> [,...n] [<MIRROR TO clause>] [next-mirror-to] [WITH { <general_WITH_options> | \<log-specific_optionspec> } [,...n]] [;] <backup_device>::= { { logical_device_name | @logical_device_name_var } | { DISK -- Not supported in SQL Database Managed Instance | TAPE -- Not supported in SQL Database Managed Instance | URL } = { 'physical_device_name' | @physical_device_name_var | 'NUL' } } <MIRROR TO clause>::= MIRROR TO <backup_device> [,...n] <file_or_filegroup>::= { FILE = { logical_file_name | @logical_file_name_var } | FILEGROUP = { logical_filegroup_name | @logical_filegroup_name_var } } <read_only_filegroup>::= FILEGROUP = { logical_filegroup_name | @logical_filegroup_name_var } <general_WITH_options> [,...n]::= --Backup Set Options COPY_ONLY -- Only backup set option supported by SQL Database Managed Instance | { COMPRESSION | NO_COMPRESSION } | DESCRIPTION = { 'text' | @text_variable } | NAME = { backup_set_name | @backup_set_name_var } | CREDENTIAL | ENCRYPTION | FILE_SNAPSHOT --Not supported in SQL Database Managed Instance | { EXPIREDATE = { 'date' | @date_var } | RETAINDAYS = { days | @days_var } } --Media Set Options { NOINIT | INIT } | { NOSKIP | SKIP } | { NOFORMAT | FORMAT } | MEDIADESCRIPTION = { 'text' | @text_variable } | MEDIANAME = { media_name | @media_name_variable } | BLOCKSIZE = { blocksize | @blocksize_variable } --Data Transfer Options BUFFERCOUNT = { buffercount | @buffercount_variable } | MAXTRANSFERSIZE = { maxtransfersize | @maxtransfersize_variable } --Error Management Options { NO_CHECKSUM | CHECKSUM } | { STOP_ON_ERROR | CONTINUE_AFTER_ERROR } --Compatibility Options RESTART --Monitoring Options STATS [= percentage] --Tape Options. These are not supported in SQL Database Managed Instance { REWIND | NOREWIND } | { UNLOAD | NOUNLOAD } --Log-specific Options. These are not supported in SQL Database Managed Instance { NORECOVERY | STANDBY = undo_file_name } | NO_TRUNCATE --Encryption Options ENCRYPTION (ALGORITHM = { AES_128 | AES_192 | AES_256 | TRIPLE_DES_3KEY } , encryptor_options) <encryptor_options> ::= SERVER CERTIFICATE = Encryptor_Name | SERVER ASYMMETRIC KEY = Encryptor_Name

Common Backup Syntax

```
BACKUP DATABASE [MyDB]
TO DISK = 'D:\SQLBackups\MyDB.bak'
WITH INIT, FORMAT;
```

```
BACKUP DATABASE [MyDB]
TO DISK = 'D:\SQLBackups\MyDB_DIFF.bak'
WITH INIT, FORMAT, DIFFERENTIAL;
```

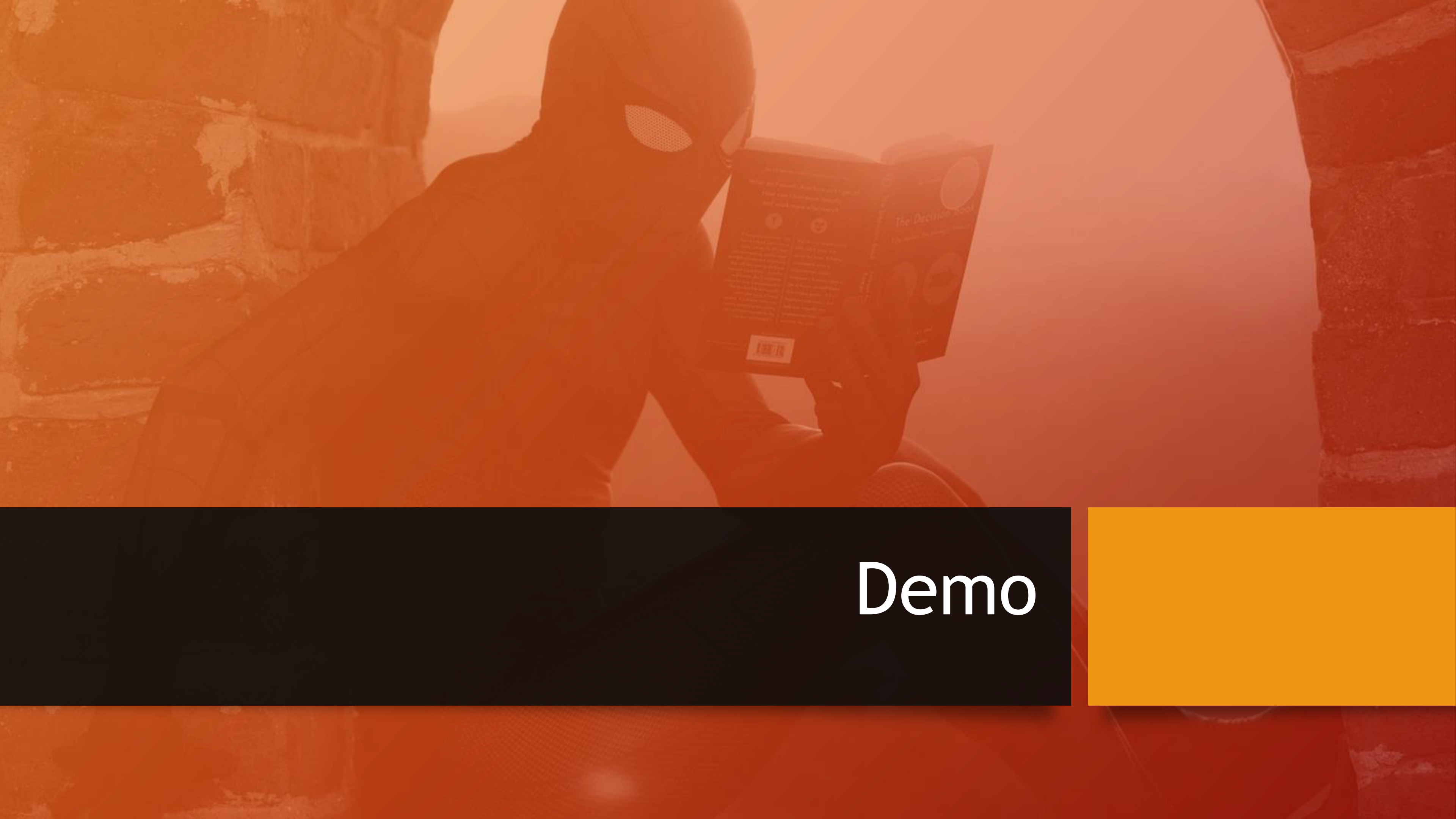
```
BACKUP LOG [MyDB]
TO DISK = 'D:\SQLBackups\MyDB_1.trn';
```

Restoring a Backup

```
RESTORE DATABASE [MyDB]
FROM DISK =
'D:\SQLBackups\MyDB.bak’;
```

```
RESTORE DATABASE [MyDB]
FROM DISK =
'D:\SQLBackups\MyDB_DIFF.bak’;
```

```
RESTORE LOG [MyDB]
FROM DISK =
'D:\SQLBackups\MyDB_1.trn';
```

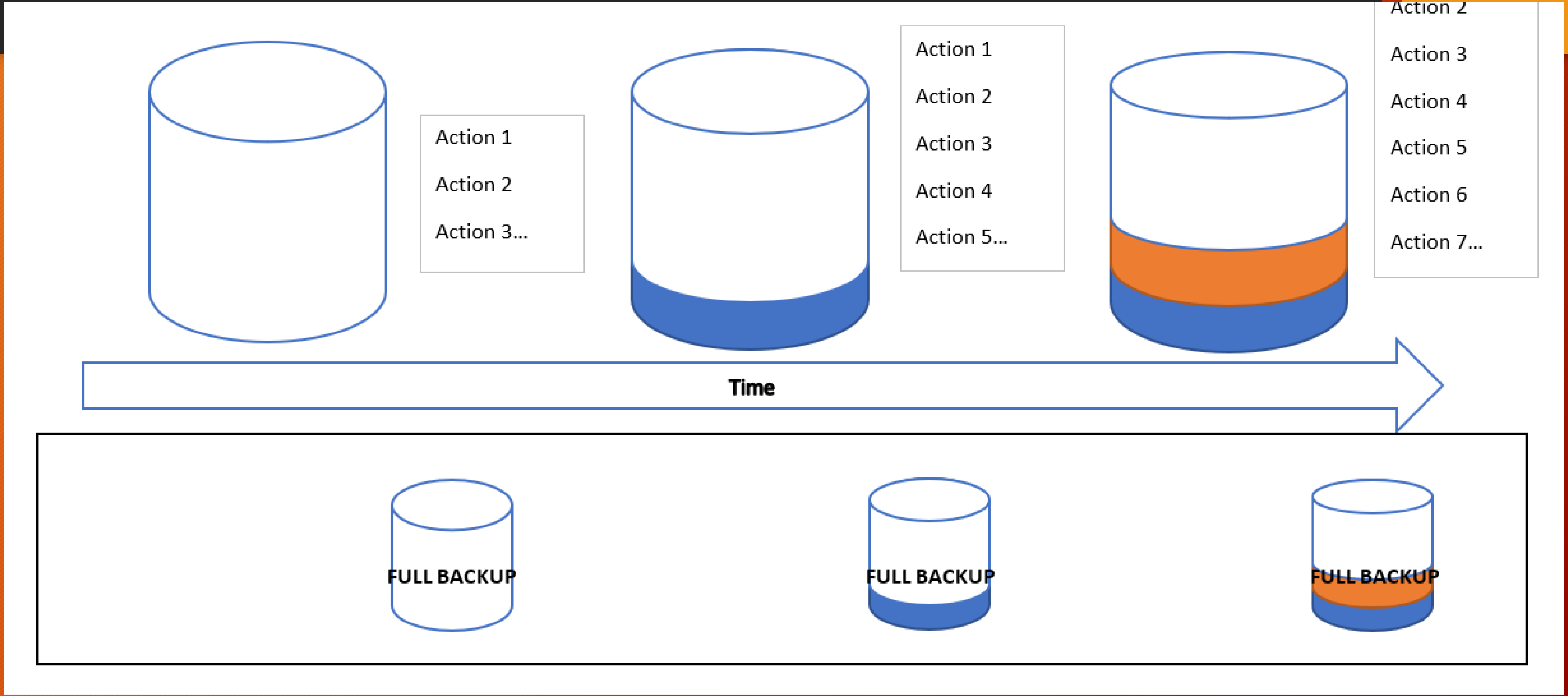


Demo

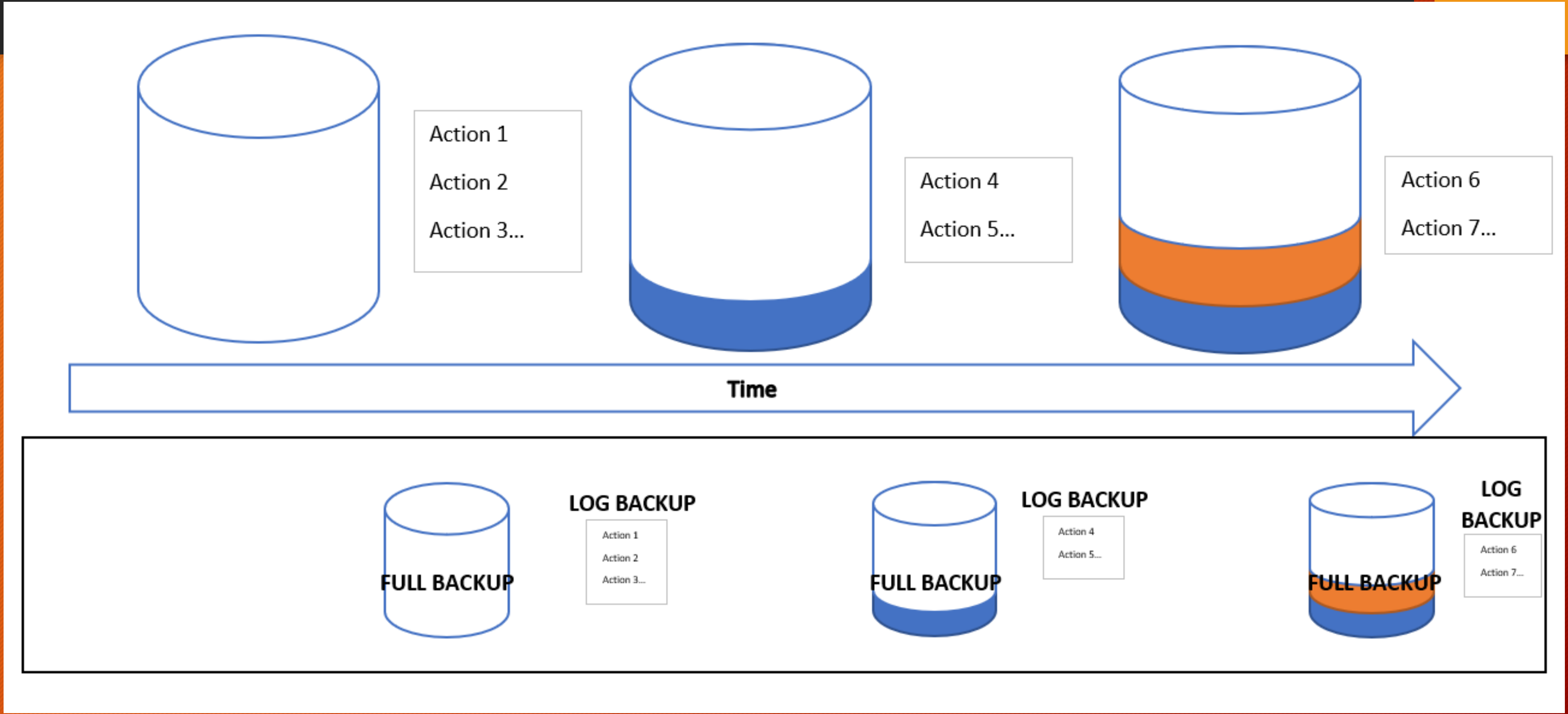
Log Backups: Important to Know

- A log backup saves off transactions since the last log backup
- A log backup allows the log file to truncate old transactions (so it doesn’t grow forever)
- ***The log file itself does not care about full backups***

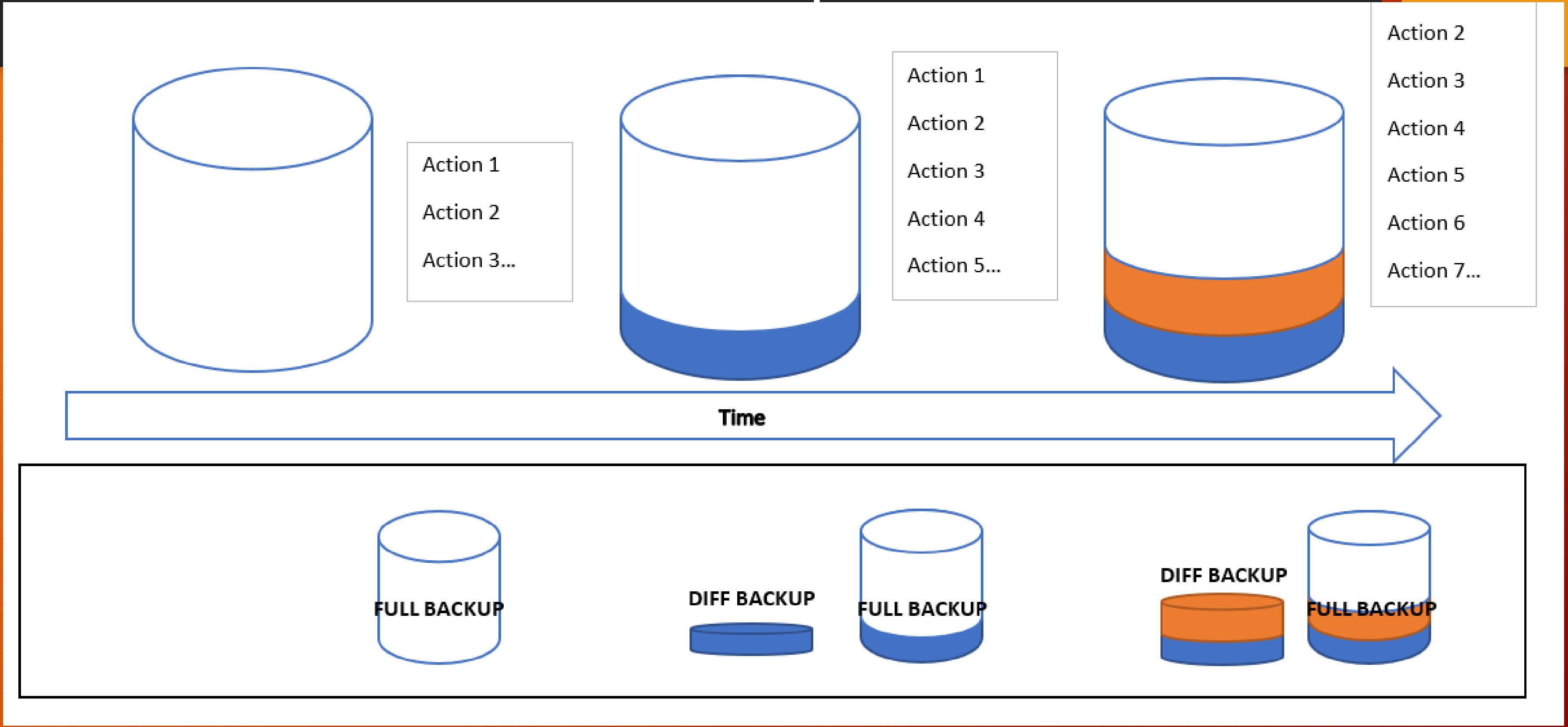
Full Backups



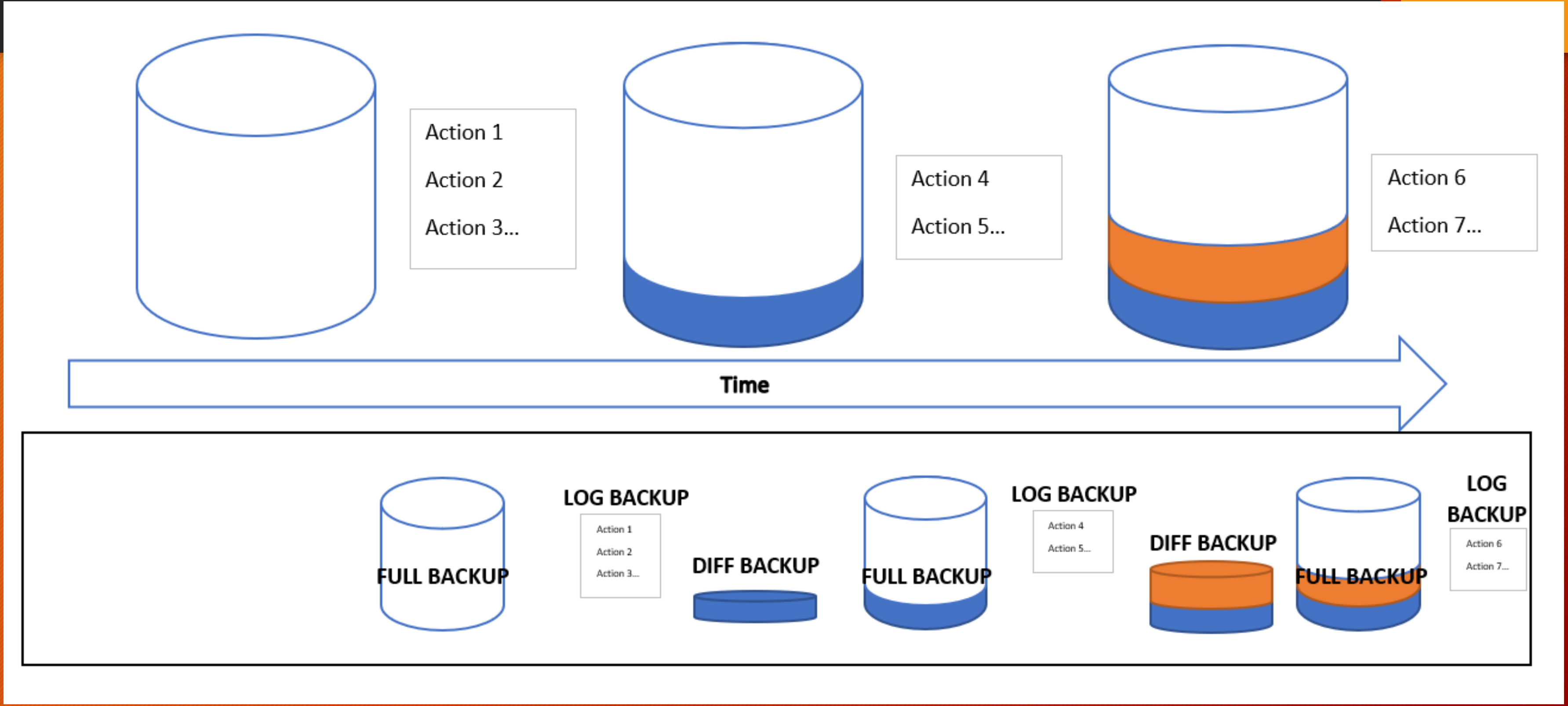
Full and Log Backups



Full and Differential Backups



Full, Log, and Differential Backups



One more thing: Recovery Model

Recovery Model	Backups
Full	Full and log backups required
Bulk-logged	Full and log backups required
Simple	Full backups required **Log backups not possible!**

Bottom Line: Backup Advice

- Schedule backups – full AND log!*
- Use native SQL Server backups
- Log backup activity
- Age out old backup files
- Alert for missing backups

*For databases in Full mode.

Integrity Checks

Why check database integrity?

- Data can be written incorrectly, or mixed up once on disk
- That’s *corruption*
- Corruption gives no warning
- **Check often, or you will lose data**

Corruption Scenarios

Regular integrity checks with alerts, mild corruption 👍

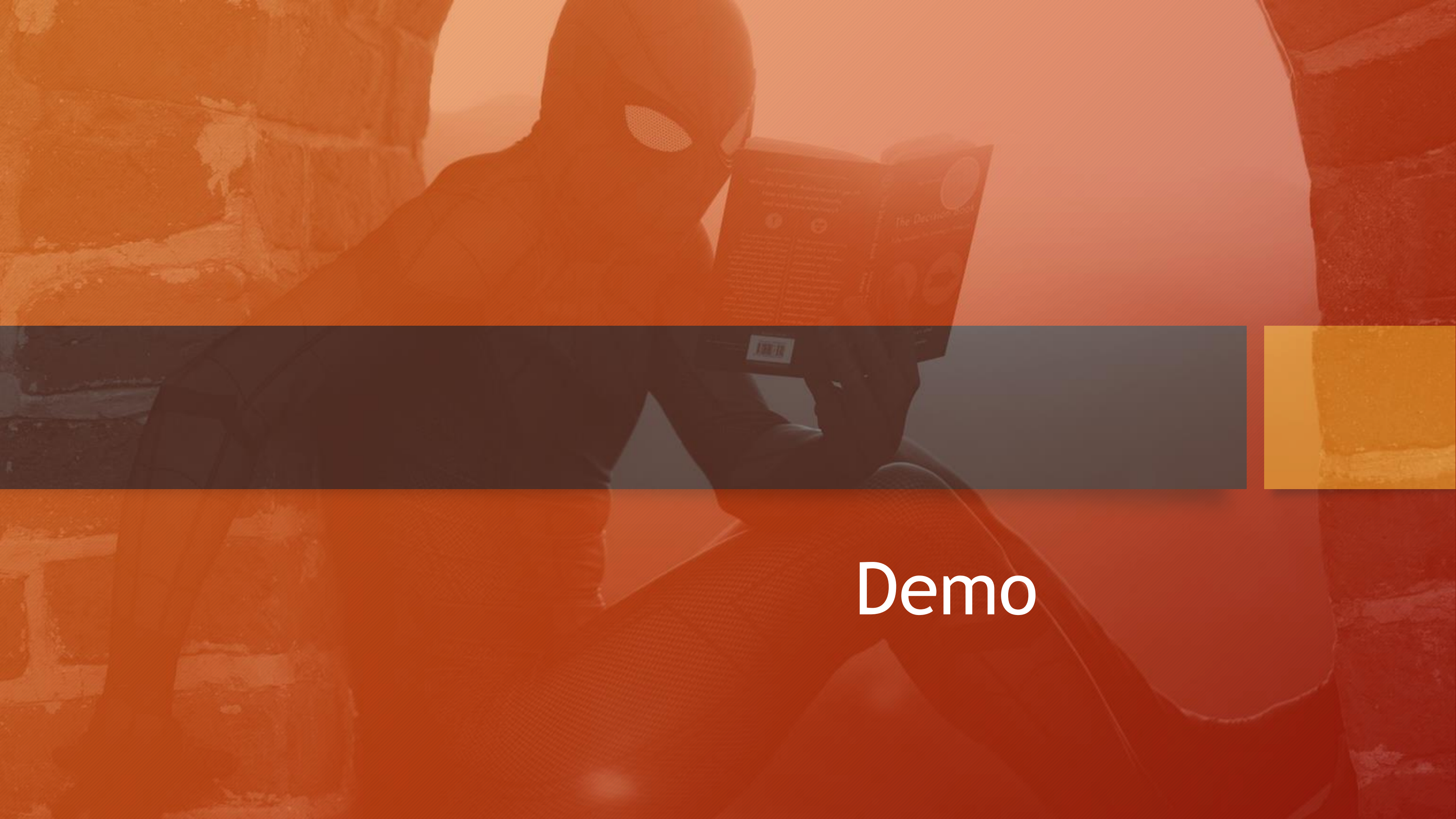
Regular integrity checks with alerts, *bad* corruption 👎

No integrity checks

Recovering from Corruption

Options in order of preference:

1. **Index Corruption only** – Drop and recreate the index
2. **Restore from backup** – Highly recommended by Microsoft
3. **REPAIR_REBUILD** – CHECKDB option does not allow data loss
4. **REPAIR_ALLOW_DATA_LOSS** – The very last resort!



Demo

Bottom Line: Integrity Check Advice

- Schedule frequent checks
- Log results in SQL
- ***Alert on results***
- Make sure you have regular backups

Index Maintenance

Why maintain indexes?

- Indexes naturally get out of order on disk
- They “fragment”

“Heavily fragmented indexes can degrade query performance and cause your application to respond slowly.”

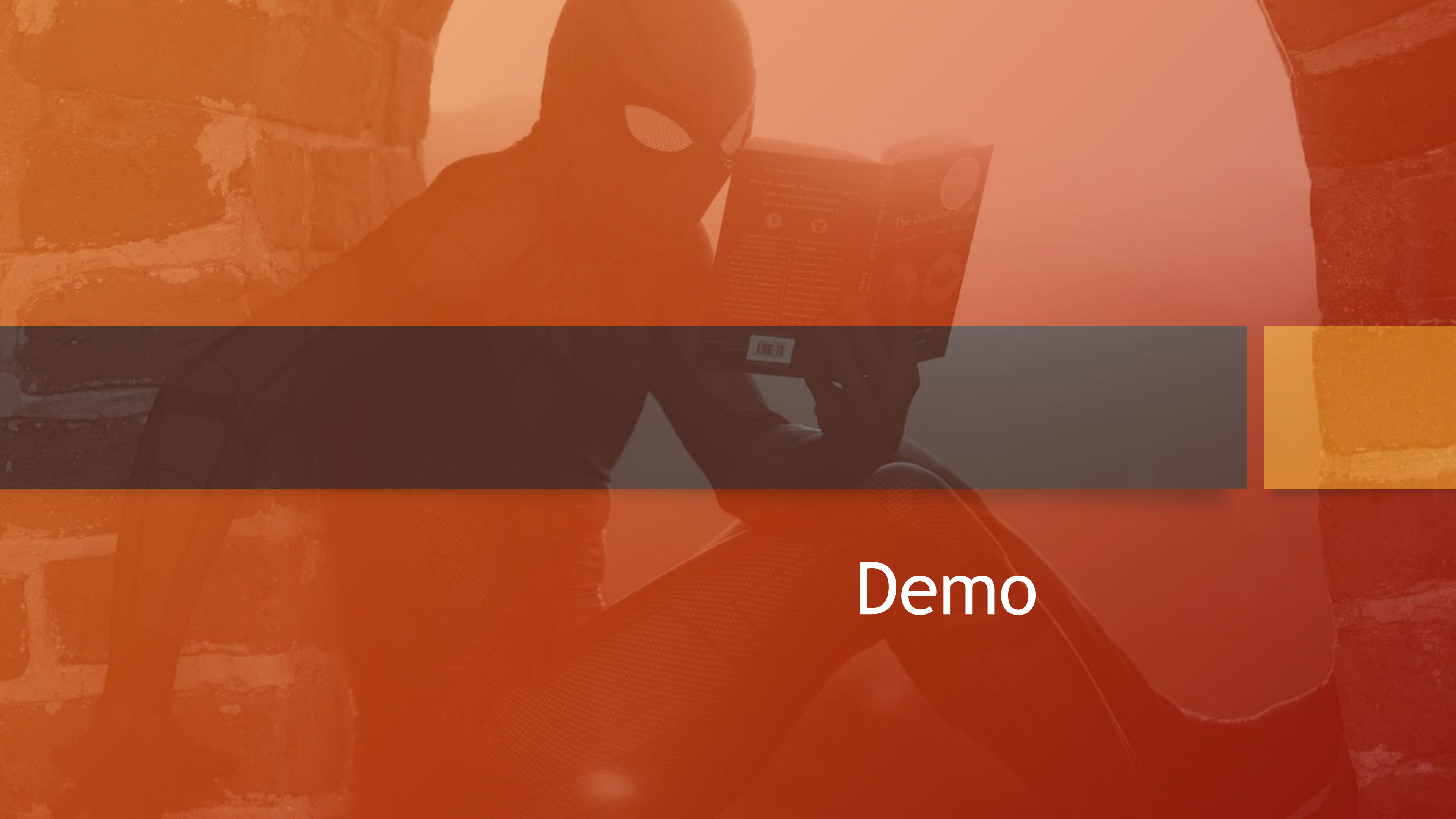
-*Reorganize and Rebuild Indexes*, *Microsoft.com*

Indexes fragment…

- Data is stored on 8 KB pages
- When a page fills, it splits
- And gets placed….somewhere

Index Maintenance Concepts

- Find fragmentation levels
- Prevent with FILLFACTOR and PAD_INDEX
- REORGANIZE – lightweight, online operation for light frag
- REBUILD – heavier, *often* offline operation for heavy frag



Demo

Reorganize vs Rebuild

- You can’t REORGANIZE a heap
- Use ALTER INDEX ALL to defragment all indexes on a table
- Very small indexes don’t see much benefit from maintenance
- Index maintenance on FULL recovery mode databases can use up a lot of transaction log space

Bottom Line: Index Maintenance Advice

- Schedule regular maintenance
- Log results in SQL

Disk Management

Why manage disks?

They get full

(of database files)

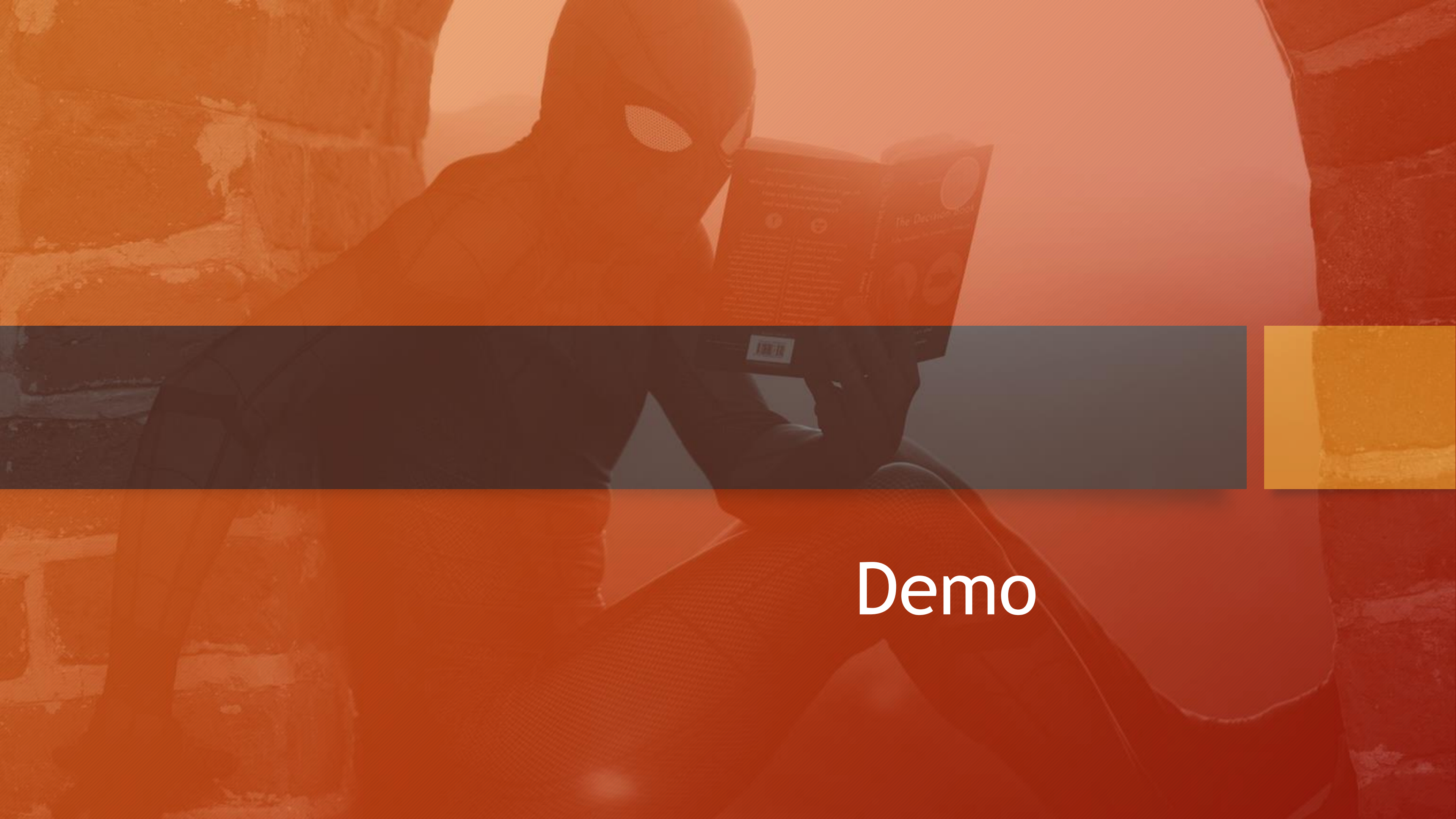
(and backup files)

It’s your responsibility!

Disk Management options

- Create your own solution
- Get the company to buy one*

(Yes, I can recommend one. ☺)



Demo

Bottom Line: Disk Management Advice

- Automate
- Log results in SQL
- ***Alert on results***
- Project disk full date
- Delete aged-out data and files

Alerting

Why alert?

- You’re kidding, right?
- Alerts tell you when something goes wrong
- Don’t alert storm!

On Alerting…

- Set up Database Mail – see Template Browser
- Alert for:
  - **Backups**: missed or failed
  - **Integrity checks**: missed, or found corruption
  - **Index maintenance**: missed
  - **Disk management**: missed collection, or drive nearing full
  - More!

Bottom Line: Alerting

- Automate
- Log alert results in SQL, too!
- Alerts should be *actionable*
- Alerts should have a broad scope
- Alerting depend on SQL Agent
- Alert for all SQL Server instances!

Remember!

- The Big Five are critical, yet overlooked
- **Anything** is better than nothing
- But, put in some effort
- (Maintenance plans suck)

Thanks to Unsplash.com for the art

- Gears - Chester Alvarez
- Tiny alarm clock - Lukas Blazek
- Carnival ride - Filip Mroz
- Cassette tape - Markus Spiske
- Spider-Man, book - Raj Eiamworakul
- Broken windows - Matt Artz
- Binders - Samuel Zeller
- Disks - Florian Pérennès
- Moons - Gianni Zanato
- Sky loudspeaker - Jeremy Yap
- Spill - Tyler Nix

Read Up!

This session is based on my chapter in "Voices from the Data Platform".

Get your copy online!