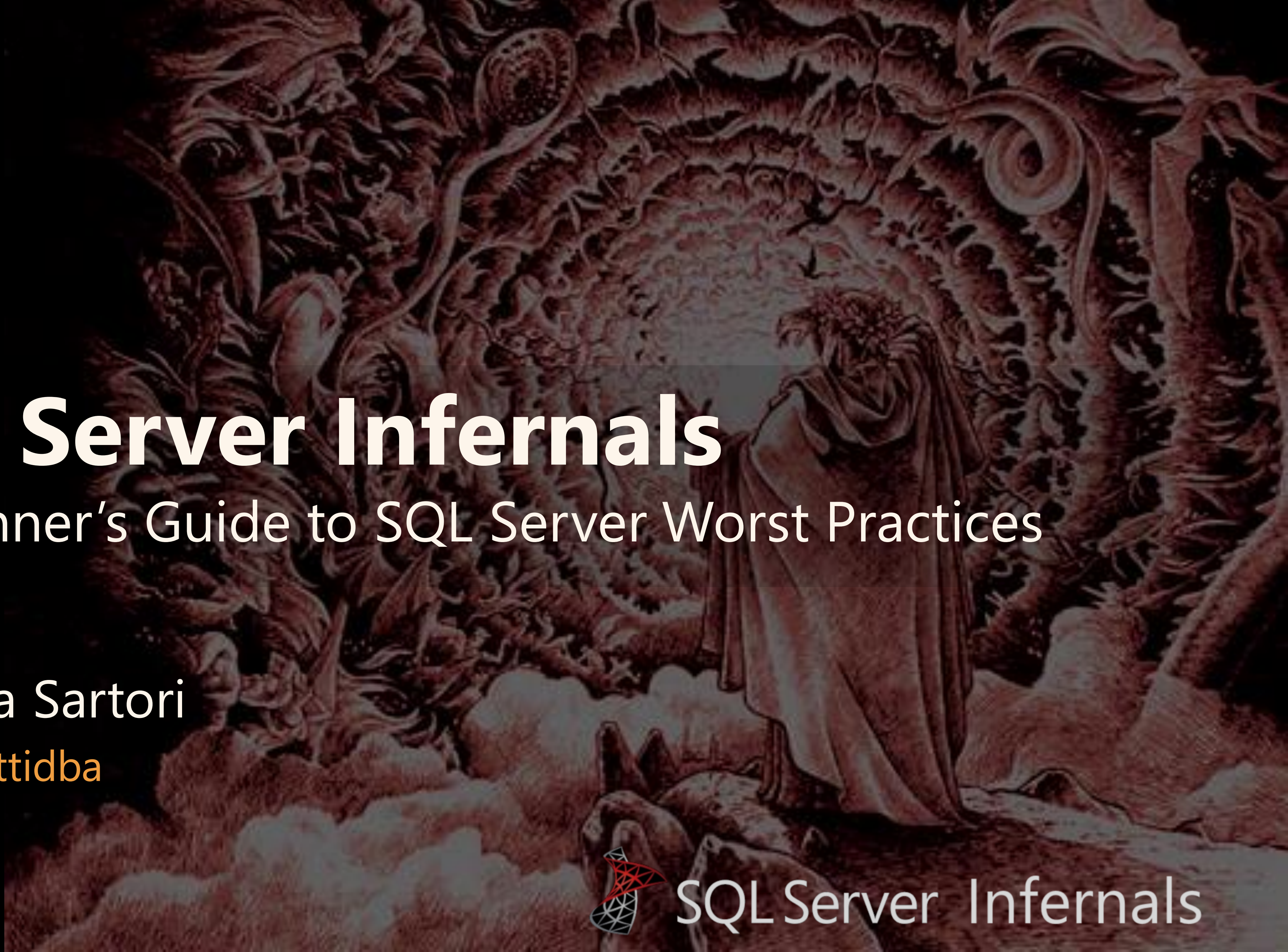


SQL Server Infernals

A Beginner’s Guide to SQL Server Worst Practices

Gianluca Sartori

@spaghettidba

SQL Server Infernals

Gianluca Sartori

Independent SQL Server consultant

Data Platform MVP, MCTS, MCITP, MCT

Works with SQL Server since version 7

DBA @ Scuderia Ferrari

Blog:	spaghettidba.com
Twitter:	@spaghettidba

Agenda

- Best practices or Worst practices?

- What can go wrong?
  - Design
  - Development
  - Installation
  - Administration

Disclaimer:

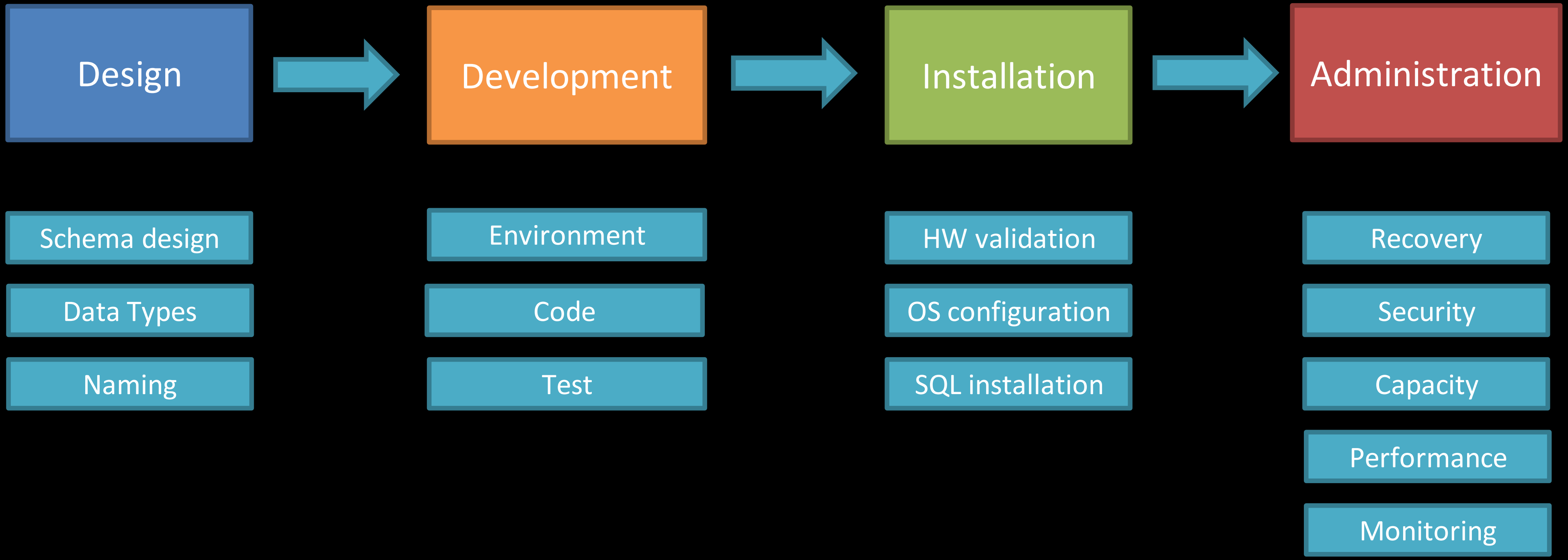
- Not everything is black or white
- «It depends» is the most likely answer

There are edge cases when some of these worst practices are the only possible solution, or not such a bad idea…

Best Practices vs. Worst Practices

- Why Best Practices are not enough
  - Too many
  - No time
  - Lack of experience
  - Not always clear what happens if we don’t follow them
- Why Worst Practices help
  - They show the mistakes to avoid
  - We can learn from someone else’s mistakes

Worst Practices Areas



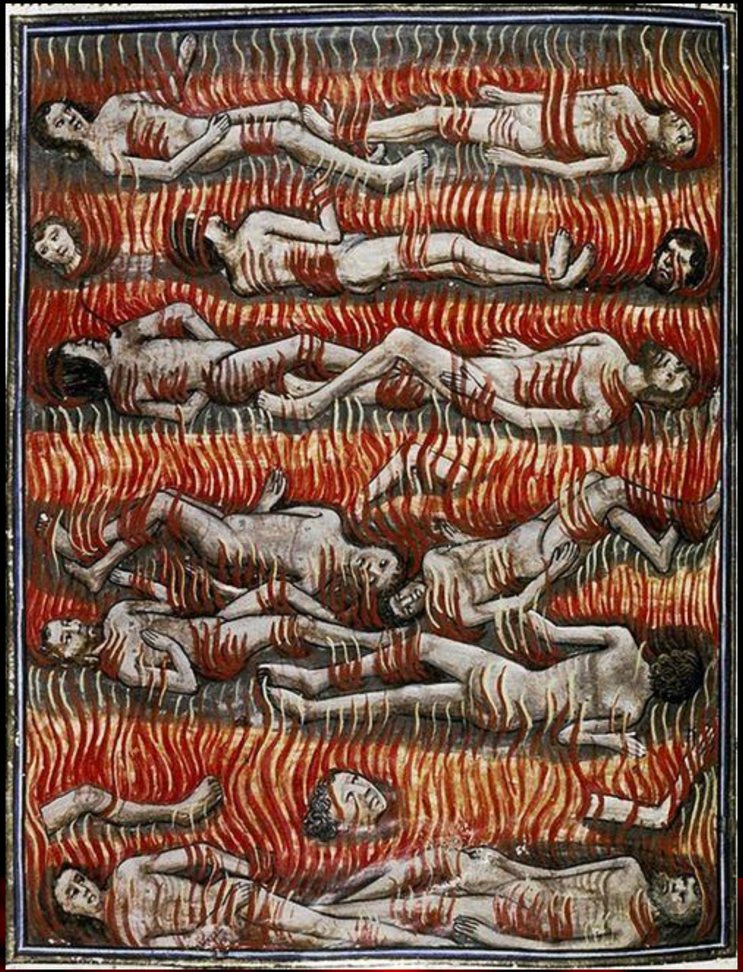
SQL Server Infernals

- Worst Practices are sins that will put you in the SQL Server hell!!
- I will guide you through the circles, as Virgil did with Dante

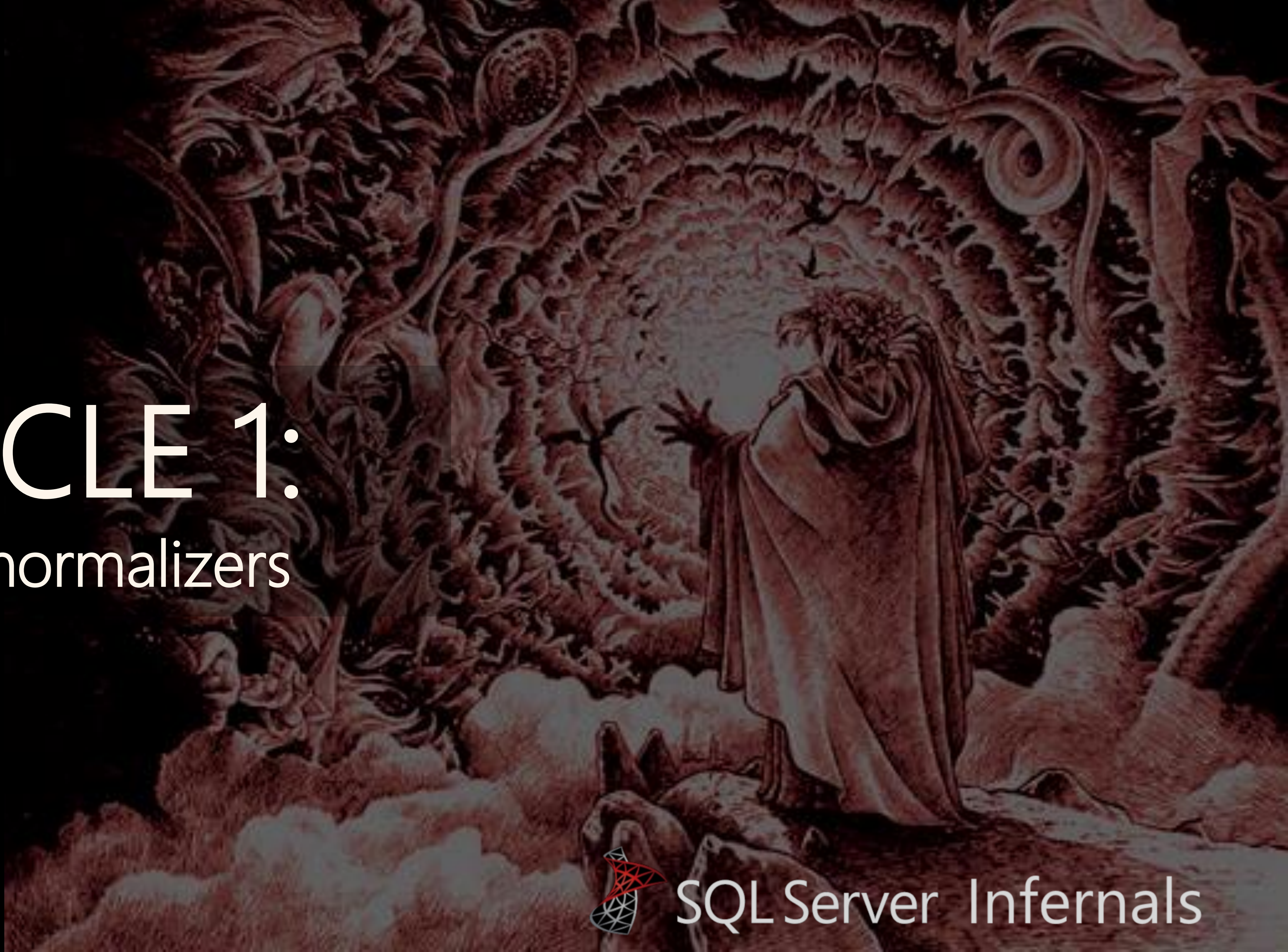
SQL Server Infernals

SQL Server Infernals BINGO!

- Check your sins in the your SQL Server Infernals BINGO card!
- Download it from https://git.io/v5oHj
- Special treats for worst sinners!



SQL Server Infernals



CIRCLE 1:

Undernormalizers

SQL Server Infernals

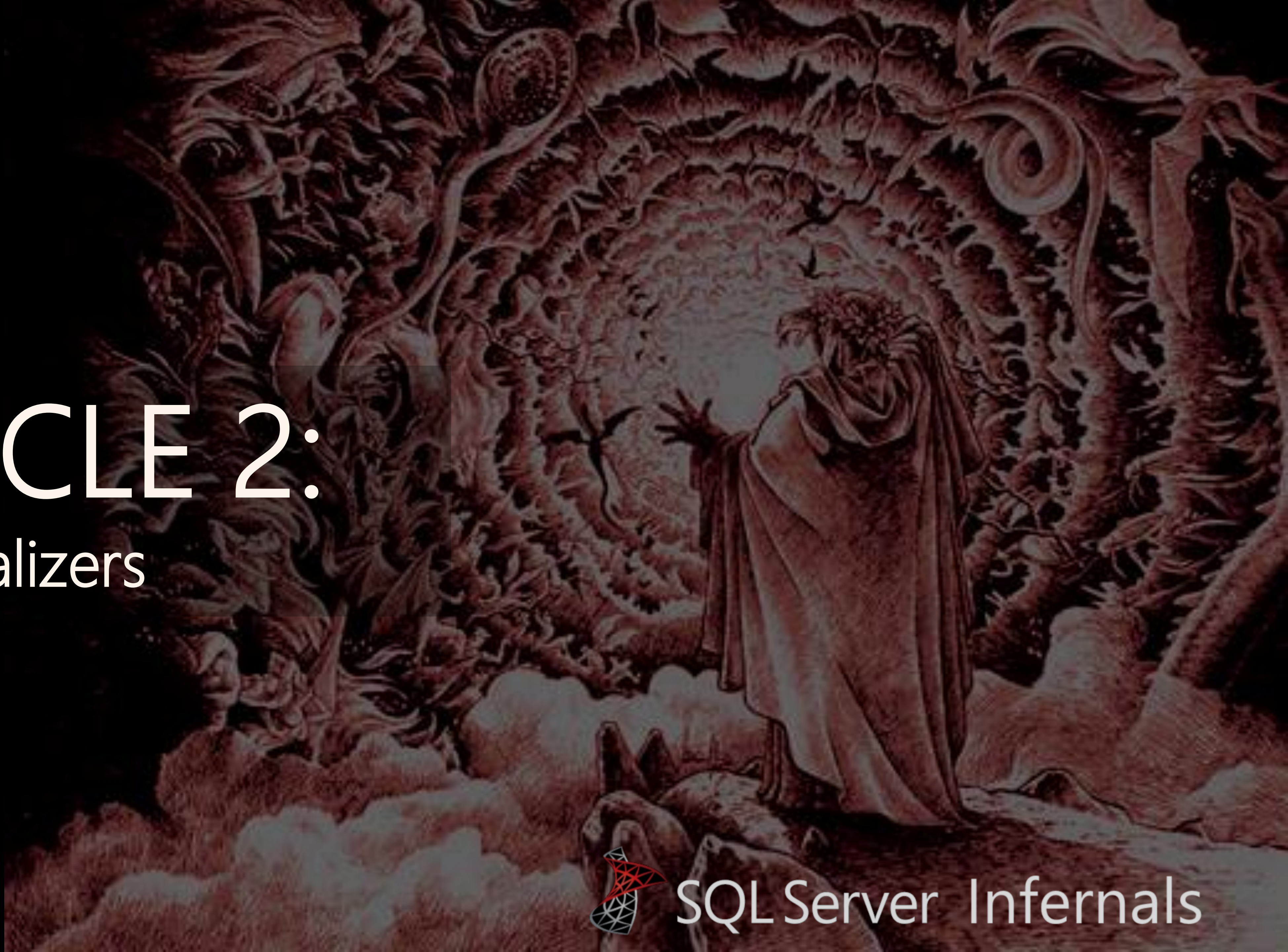
Schema Design

- Not normalizing the schema
  - 1NF:
    A primary key, atomic attributes only
  - 2NF:
    Every attribute depends on **the whole** key
  - 3NF:
    Every attribute depends **only** on the key

*«The key, the whole key, nothing but the key,
so help me Codd»*

Clues of denormalization

- Repeating data ← *redundancies*
- Inconsistent data between tables ← *anomalies*
- Data separated by «,»
  - *Ex: john@gmail.com, john@business.com*
- Structured data in «notes» columns
- Columns with a numeric suffix
  - *Ex: Zone1, Zone2, Zone3 …*

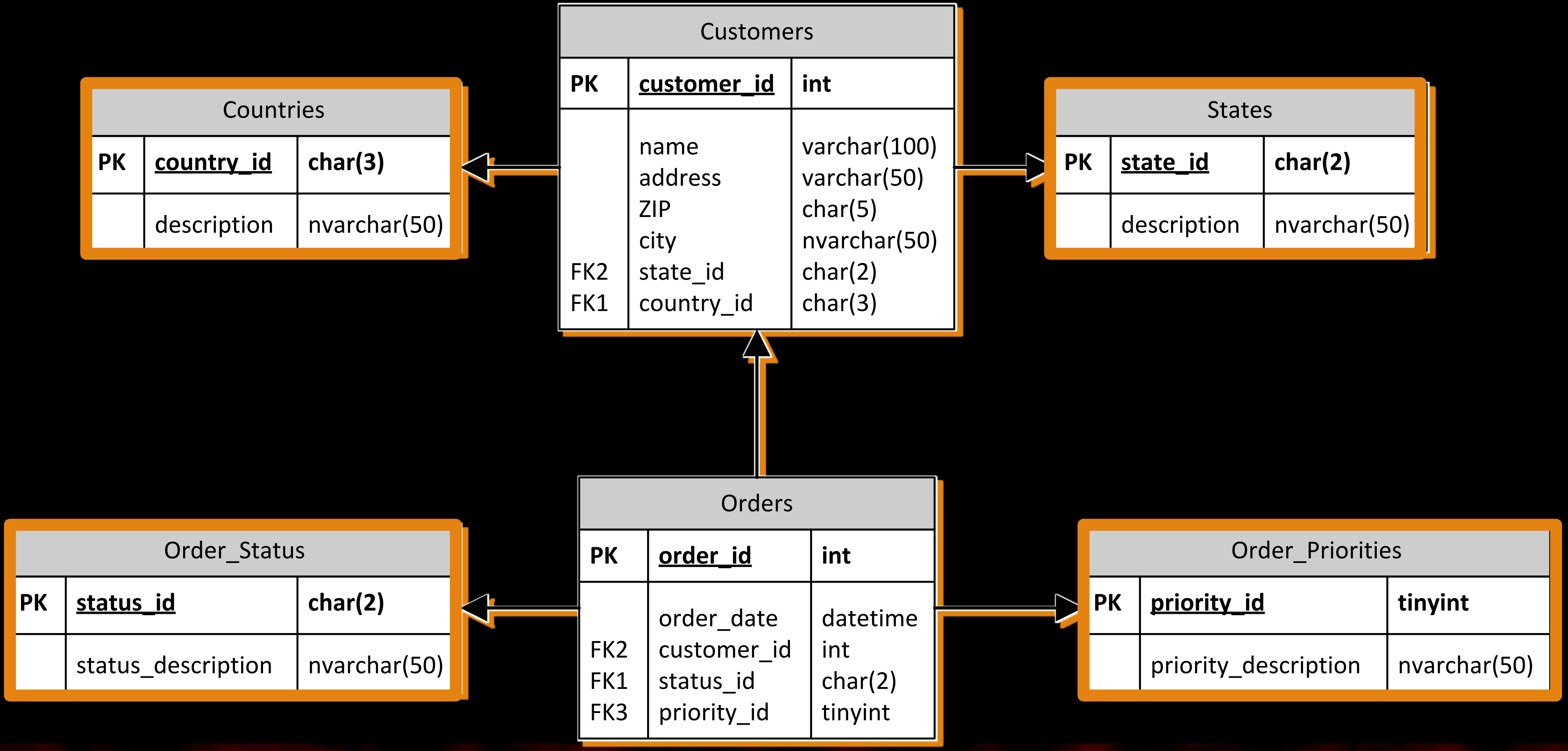


CIRCLE 2:

Generalizers

SQL Server Infernals

Lookup Tables



Customers		
PK	**customer_id**	**int**
	name	varchar(100)
	address	varchar(50)
	ZIP	char(5)
	city	nvarchar(50)
FK2	state_id	char(2)
FK1	country_id	char(3)

Countries		
PK	**country_id**	**char(3)**
	description	nvarchar(50)

States		
PK	**state_id**	**char(2)**
	description	nvarchar(50)

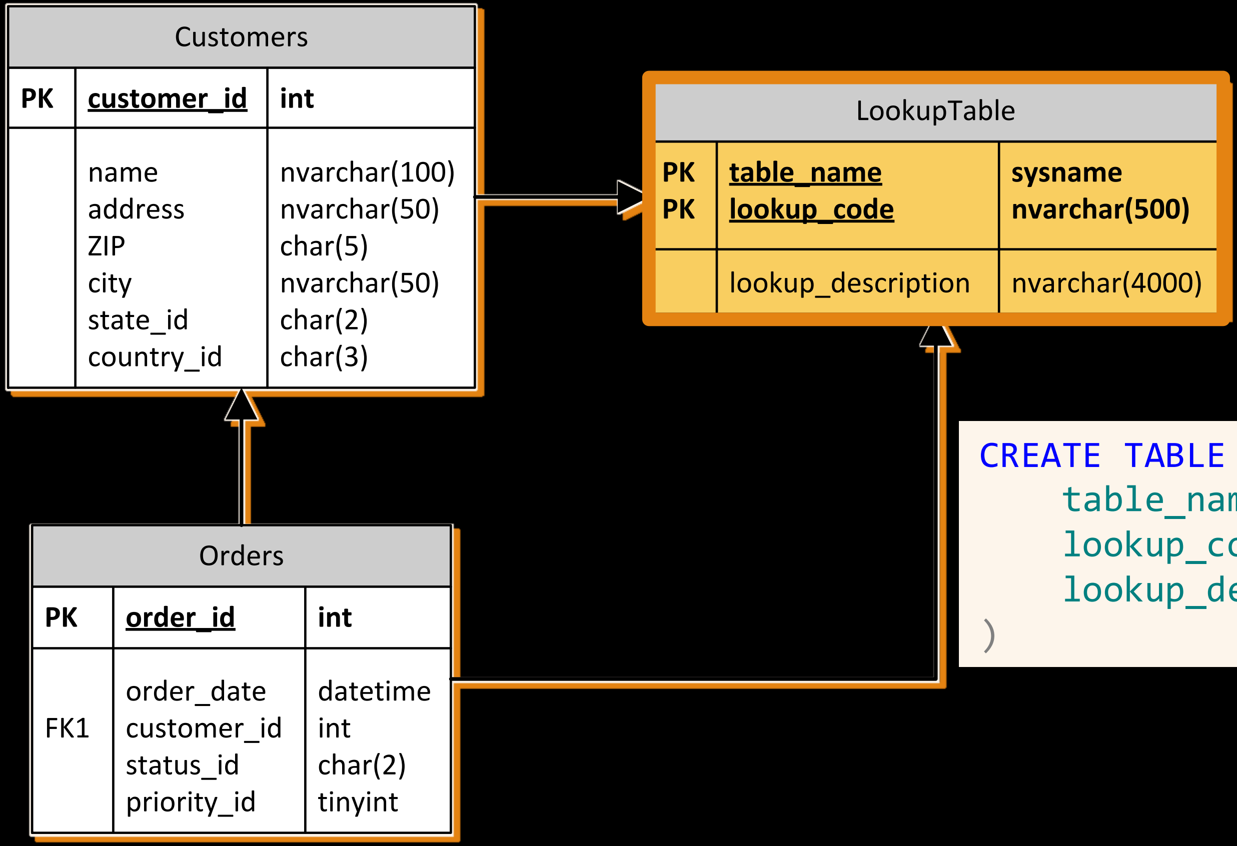
Orders		
PK	**order_id**	**int**
	order_date	datetime
FK2	customer_id	int
FK1	status_id	char(2)
FK3	priority_id	tinyint

Order_Status		
PK	**status_id**	**char(2)**
	status_description	nvarchar(50)

Order_Priorities		
PK	**priority_id**	**tinyint**
	priority_description	nvarchar(50)

One lookup table for each attribute

OTLT: One True Lookup Table



```
CREATE TABLE LookupTable (
    table_name sysname,
    lookup_code nvarchar(500),
    lookup_description nvarchar(4000)
)
```

One lookup table for all attributes

OTLT: One True Lookup Table

- No Foreign Keys
- Generic data types → nvarchar(SomeHighNumber)

  Implicit Conversions, Incorrect Data, Huge memory grants…
- CHECK constraints may help to a point…

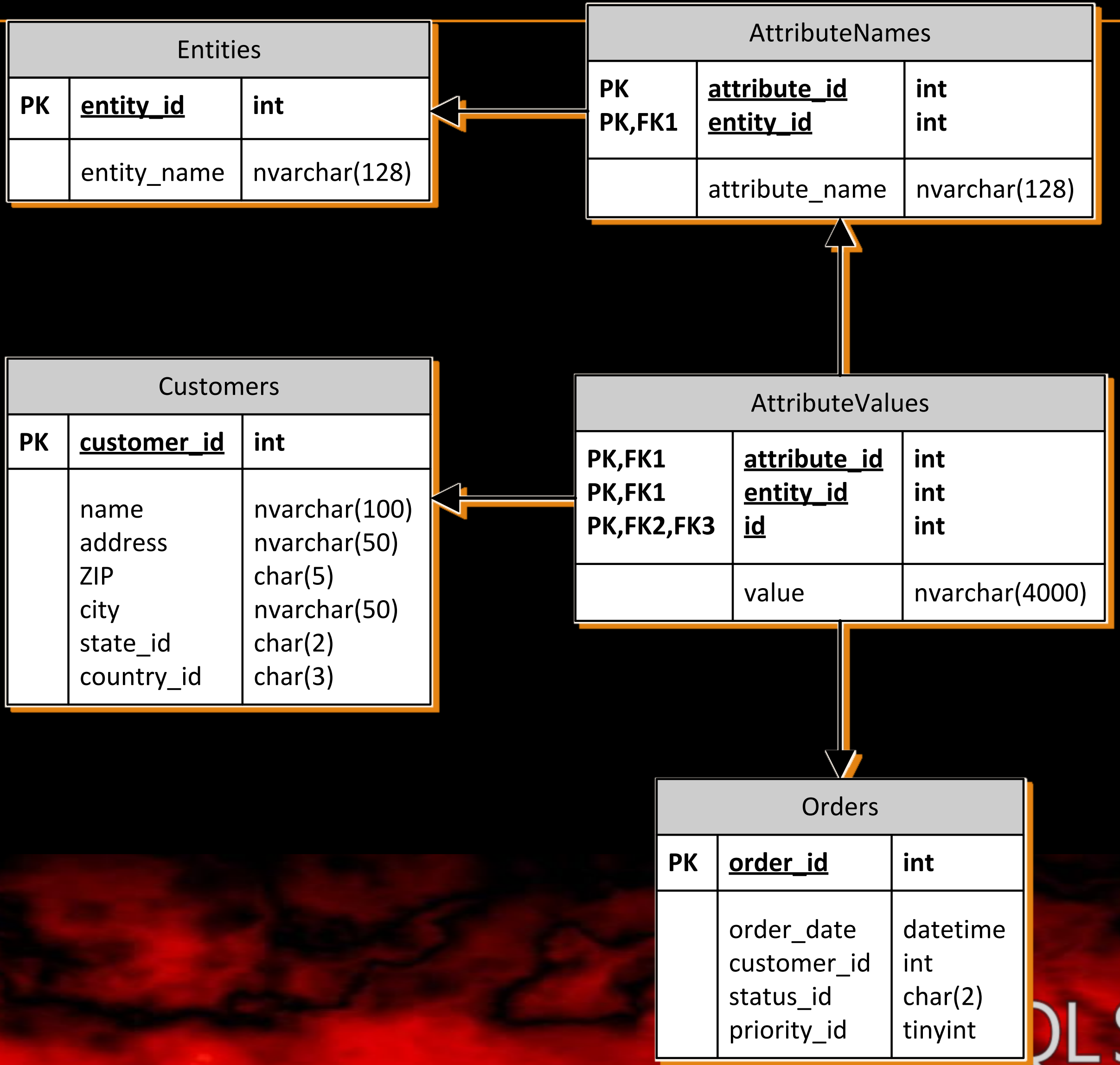
```
CHECK(
    CASE
        WHEN lookup_code = 'states'     AND lookup_code LIKE '[A-Z][A-Z]'      THEN 1
        WHEN lookup_code = 'priorities' AND lookup_code LIKE '[0-9]'           THEN 1
        WHEN lookup_code = 'countries'  AND lookup_code LIKE '[0-9][0-9][0-9]' THEN 1
        WHEN lookup_code = 'status'     AND lookup_code LIKE '[A-Z][A-Z]'      THEN 1
        ELSE 0
    END = 1
)
```

- Locking

SQL Server Internals

EAV: Entity, Attribute, Value

INSTANT DAMNATION!



Entities

PK	**entity_id**	int
	entity_name	nvarchar(128)

AttributeNames

PK	**attribute_id**	int
PK,FK1	**entity_id**	int
	attribute_name	nvarchar(128)

Customers

PK	**customer_id**	int
	name	nvarchar(100)
	address	nvarchar(50)
	ZIP	char(5)
	city	nvarchar(50)
	state_id	char(2)
	country_id	char(3)

AttributeValues

PK,FK1	**attribute_id**	int
PK,FK1	**entity_id**	int
PK,FK2,FK3	**id**	int
	value	nvarchar(4000)

Orders

PK	**order_id**	int
	order_date	datetime
	customer_id	int
	status_id	char(2)
	priority_id	tinyint

EAV: Entity, Attribute, Value

INSTANT DAMNATION!

Disadvantages:

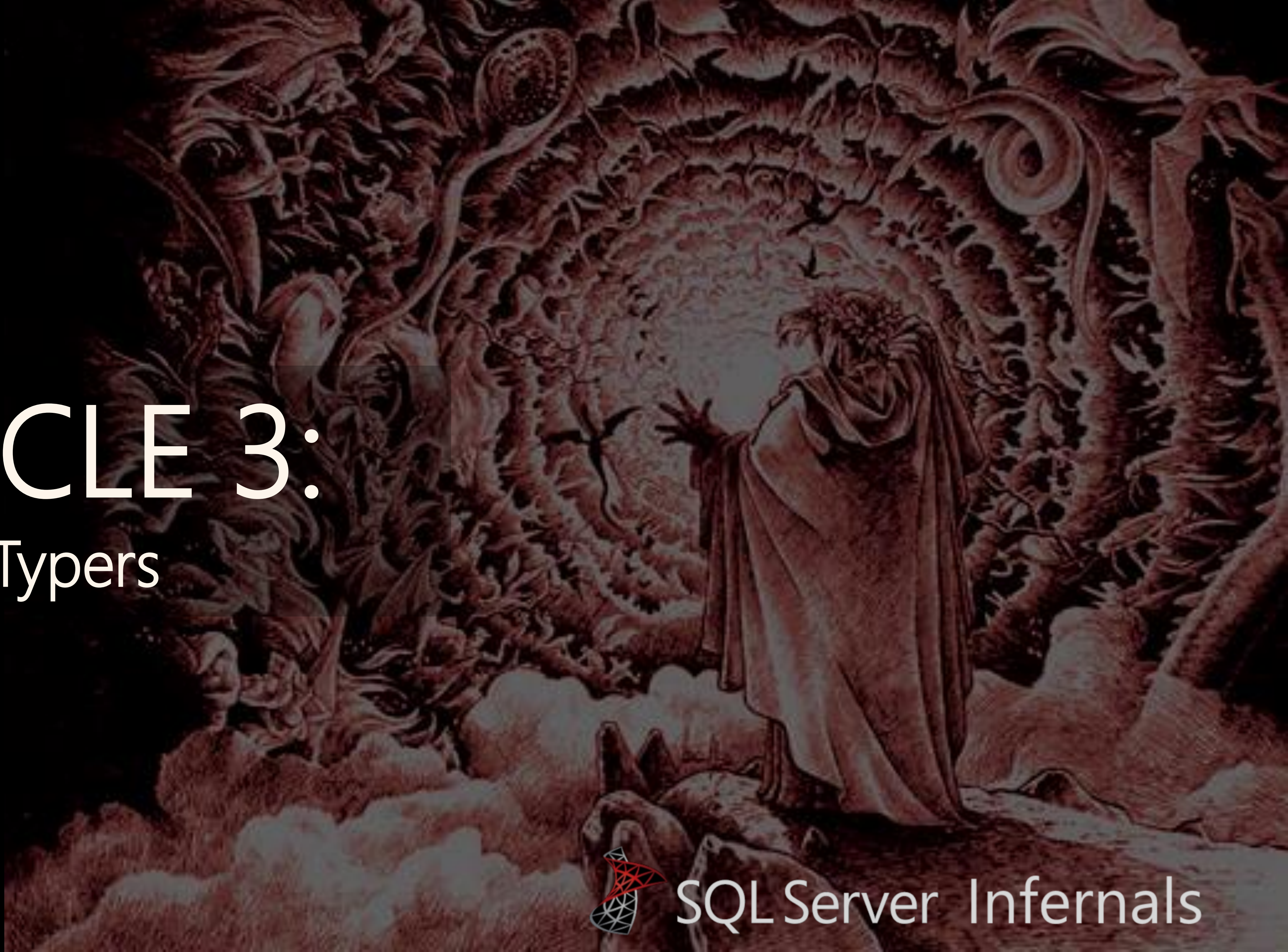
- Generic data types → Ex: varchar(4000)
- No Foreign Keys
- No CHECK constraints
- Multiple accesses to the same table
  - *One access per attribute*

Advantages

- Dynamic schema: no need to alter the database
  - *Replication, distributed environments*

EAV: Entity, Attribute, Value

- Reporting is insanely hard.
- Writing to the EAV schema is a mess
- Workaround:
  - Reads: PIVOT / Crosstab
  - Writes: View + INSTEAD OF triggers
- Alternatives:
  - SPARSE columns
  - XML/JSON
  - Key-value store databases
  - Document-oriented databases



CIRCLE 3:

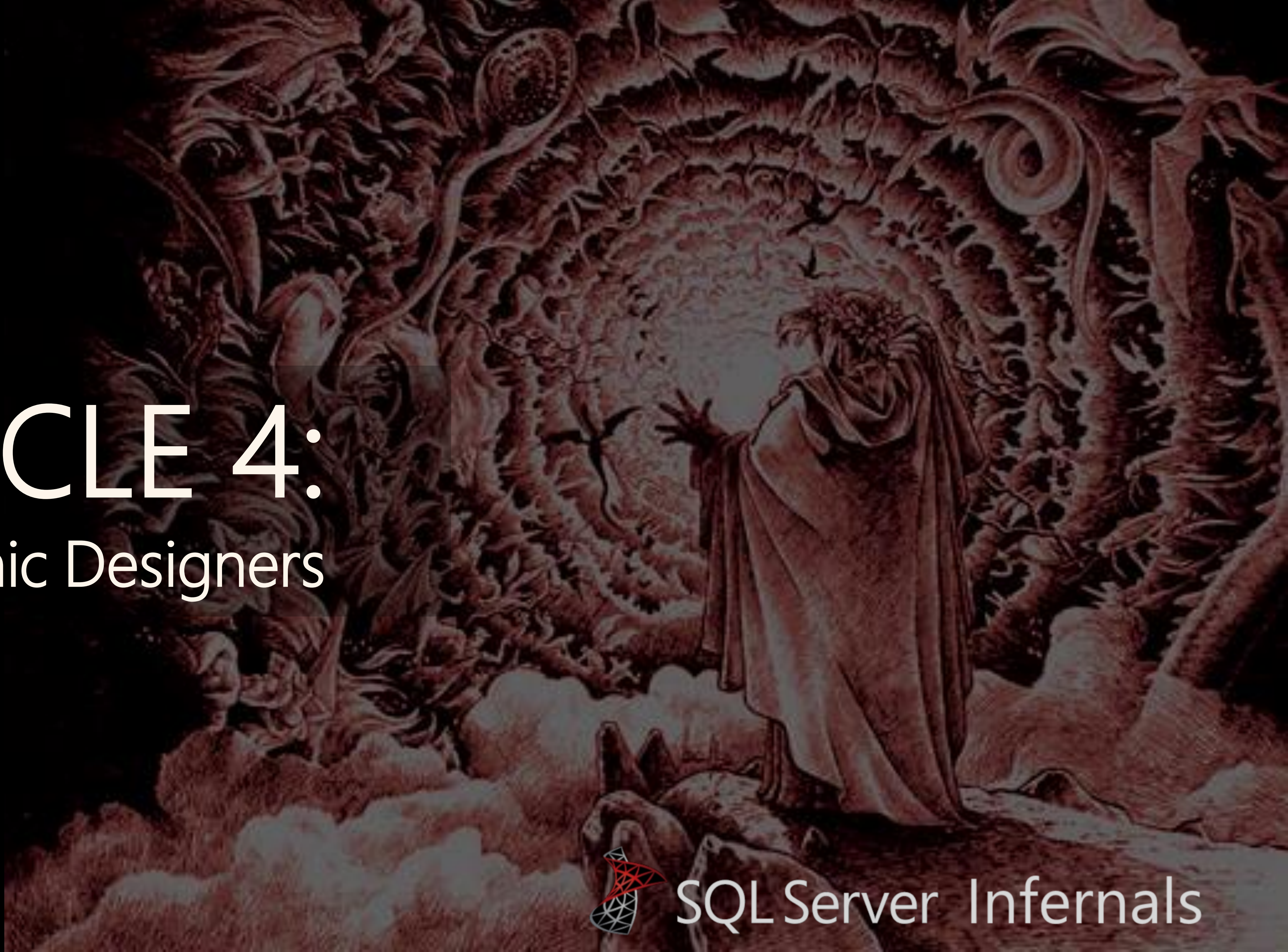
Shaky Typers

SQL Server Infernals

Data type Worst Practices

- Numeric data types for non-numeric data
- Storing data as their human-readable representation
- Using deprecated data types
- Using larger data types “just in case”
- Using variable length data types for fixed size data
- Storing durations in date/datetime columns
- Getting Unicode wrong
- Using different data types for the same data in different tables

INSTANT DAMNATION!



CIRCLE 4:

Anarchic Designers

SQL Server Infernals

Chaos Belongs to Hell

- No Primary Key o surrogate keys only

  *«identity» is not **the** only possible key!*
- No Foreign Keys

  They're «awkward»
- No CHECK constraint

  The application will guarantee consistency…
- Wrong data types
  - Data type is the 1° constraint on the data
- Use of NULL where not appropriate
- Use of «dummy» data (ex: '' , 0)

CIRCLE 5:

Inconsistent Baptists

SQL Server Infernals

Damnation by Namification

- Hungarian Notation (AKA «tibbing»)
- Insanely short names
- Insanely long names
- Mixing languages
- Using the «sp_» prefix
- Using reserved words or illegal characters
- Using system generated constraint names
- No naming convention or multiple naming conventions

Hungary is a nice str_country

CIRCLE 6:

Environment Pollutors

SQL Server Infernals

Pollutors will be prosecuted

- Developing in production
- Using the test environment for development
- Using a shared database for development
- No source control
- Developing with sysadmin privileges
- Developing on a different version/edition from production
  (less problematic after 2016 SP1)

CIRCLE 7:

Overly Optimistic Testers

SQL Server Infernals

Pessimists are Optimists with Experience

- Not testing all the code

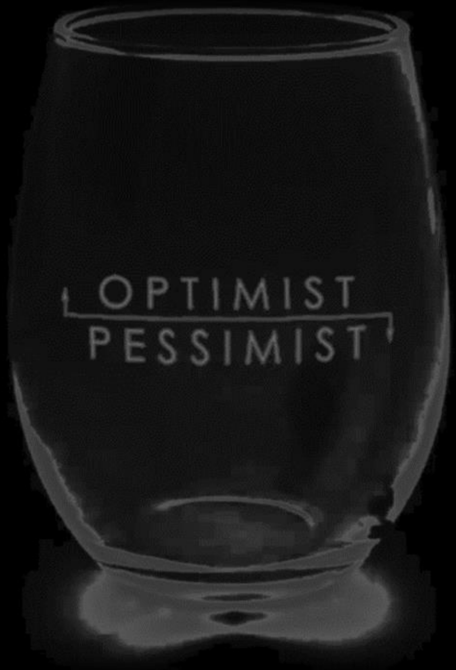
  Use meaningful data volumes
- Testing in production

  Can alter production data

  Interferes with production users
- Testing in development environment

  Useful at most for unit tests

INSTANT DAMNATION!



CIRCLE 8:

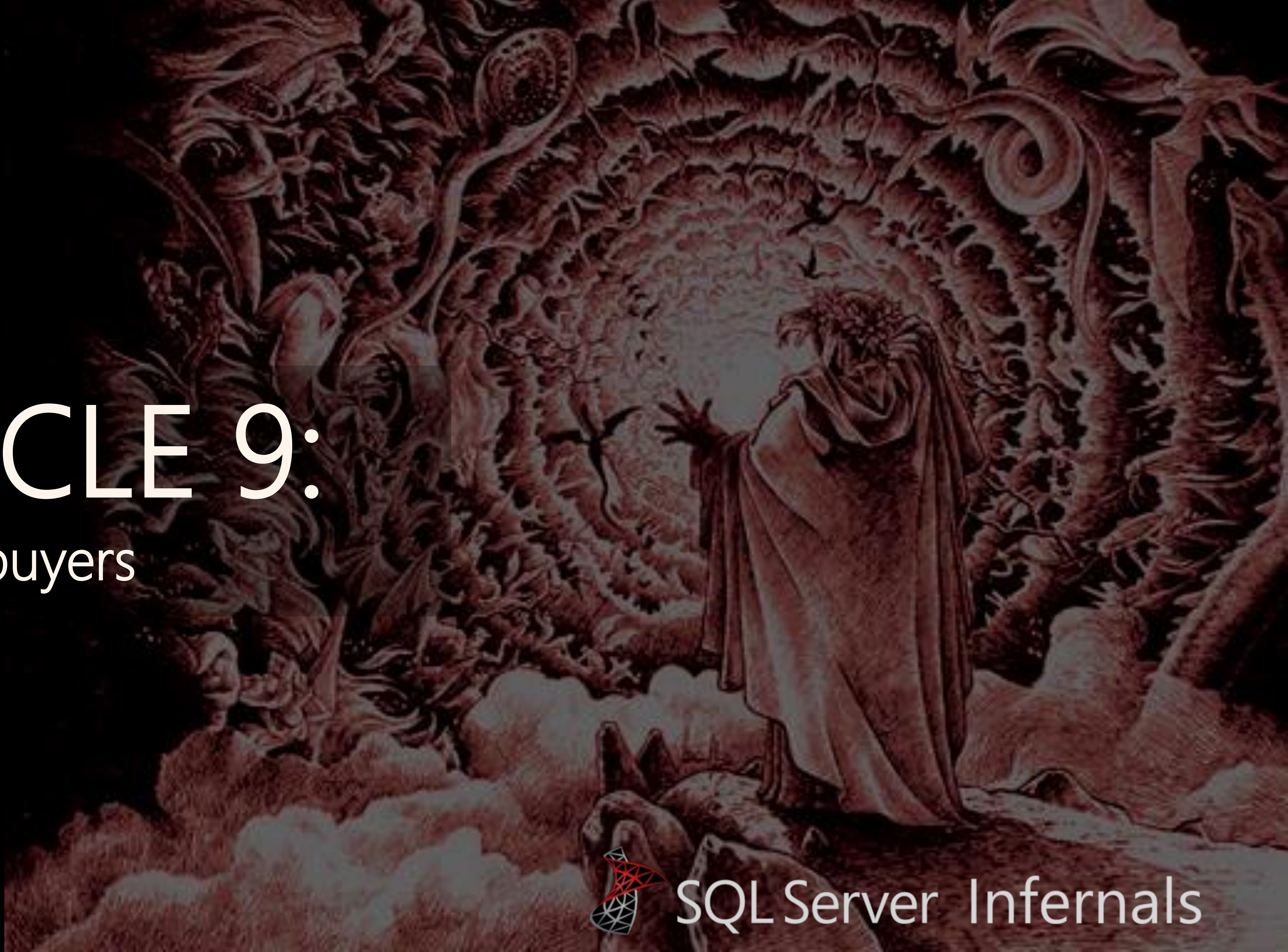
Indolent developers

SQL Server Infernals

Development Worst Practices

- No transactions
- No error handling
  @@ERROR is a thing of the past!
- Wrong isolation levels
  NOLOCK = no consistency!
- SELECT *
- Dynamic SQL with concatenated params
- Code vulnerable to *SQL injection*
- No abstraction layer
  Views, Functions, Stored Procedures

It’s all about laziness



CIRCLE 9:

Stingy buyers

SQL Server Infernals

HW Worst Practices

INSTANT DAMNATION!

- Using inadequate or unbalanced HW
- Reusing decommissioned servers for new installations
  - Slower CPUs (license costs the same on fast CPUs)
  - Less RAM supported
- Planning storage with capacity in mind
  - Choosing the wrong RAID level

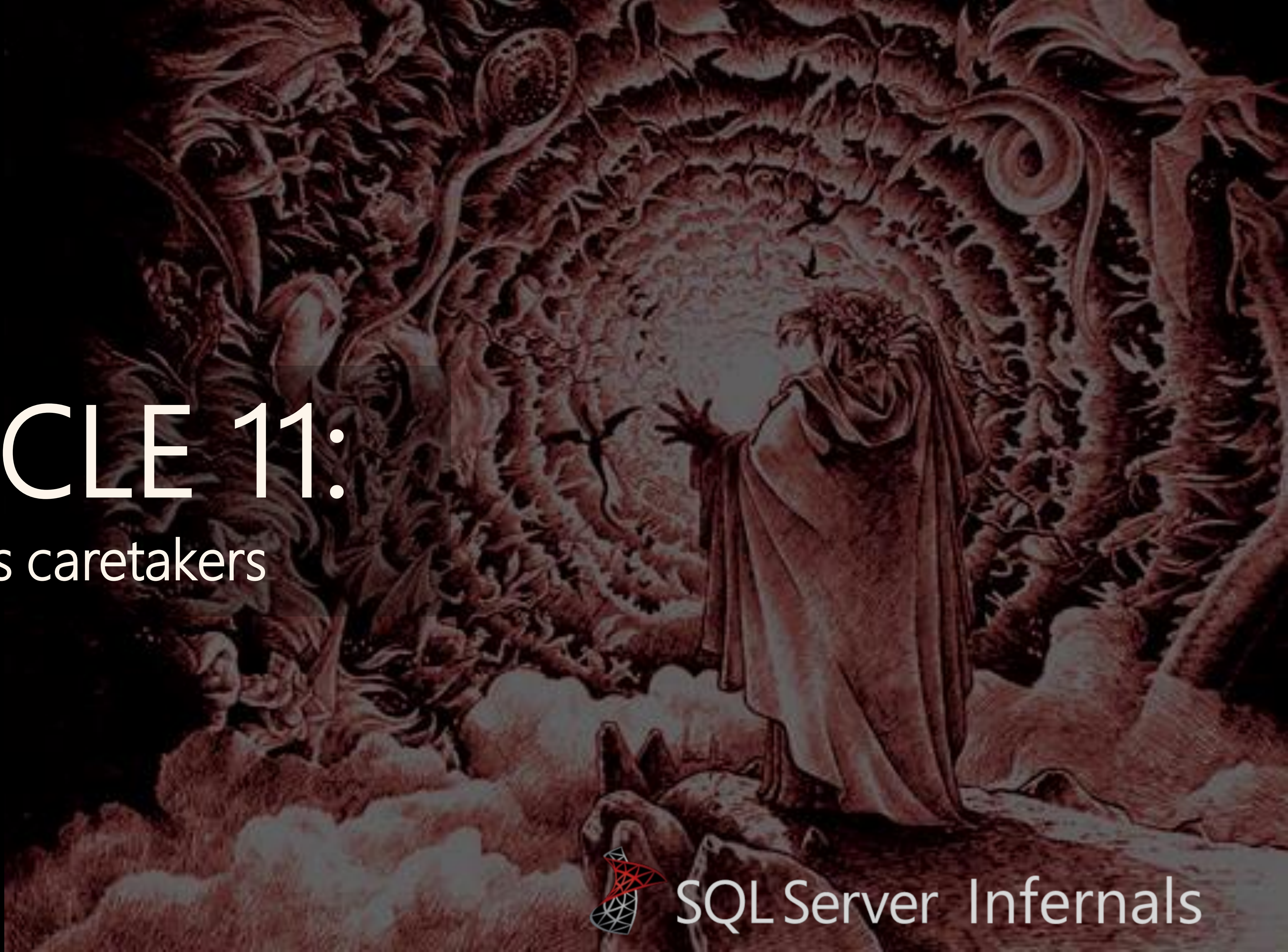
CIRCLE 10:

Next next finish installers

SQL Server Infernals

Installation Worst Practices

- Installing accepting all the defaults
  - *Data files on the system drive*
  - *MAXDOP = 0*
  - *Max Server Memory = +∞*
- Installing unused components
- Installing multiple services on the same machine
- Giving up easy wins on I/O
  - Partition misalignment
  - Using the default allocation unit (4Kb)

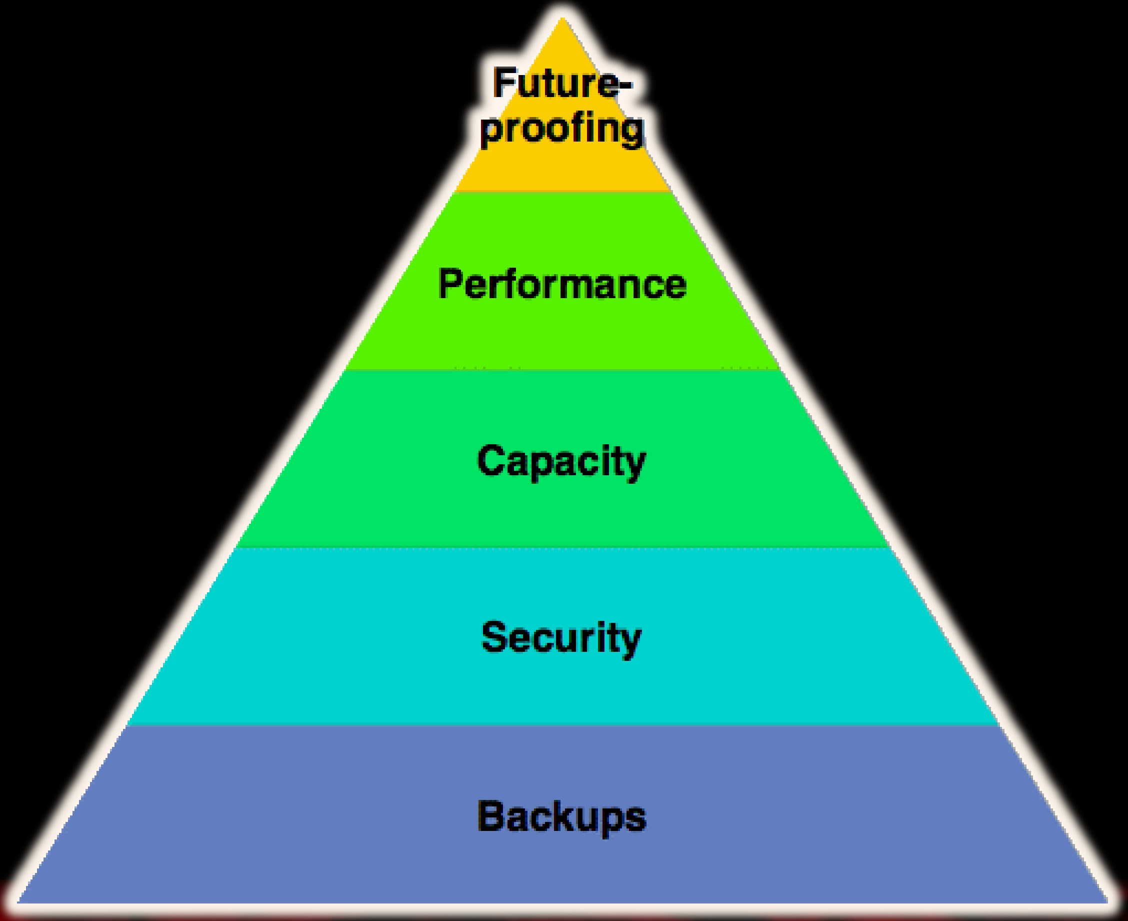


CIRCLE 11:

Careless caretakers

SQL Server Infernals

What does a database need?



Backup and Recovery Worst Practices

- No backup
  - *With FULL recovery it's a timebomb*
  - *Ignoring RPO and RTO (it's not your decision!)*
- No test restores
- No consistency checks
  - *DBCC REPAIR_ALLOW_DATA_LOSS as default response to corruption*

INSTANT DAMNATION!

Our responsibility is to perform restores, not backups!

Security Worst Practices

- Too many sysadmins
- Everyone authenticating as `‘sa’`
- Using SQL Authentication
  - *Weak passwords*
    - *123*
    - *P4$$w0rd*
    - *Same as username*
- No auditing on sensitive data

INSTANT DAMNATION!

Capacity Management Worst Practices

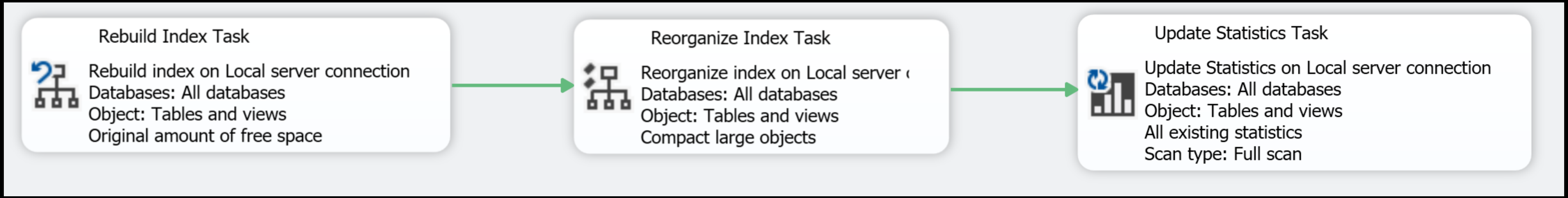
- Not checking disk space
  - *No space left = database halted!*
  - *FULL recovery and no log backups?*
- Relying 100% on autogrowth
- Autoshrink
- Autoclose
- Not presizing tempdb

  Different file size = latching (and striping) penalty

INSTANT DAMNATION!

Maintenance Worst Practices

- Not maintaining indexes and statistics
- Obsessing over maintaining indexes and statistics
- Using catch-all maintenance plans



Rebuild Index Task

Rebuild index on Local server connection
Databases: All databases
Object: Tables and views
Original amount of free space

Reorganize Index Task

Reorganize index on Local server (
Databases: All databases
Object: Tables and views
Compact large objects

Update Statistics Task

Update Statistics on Local server connection
Databases: All databases
Object: Tables and views
All existing statistics
Scan type: Full scan

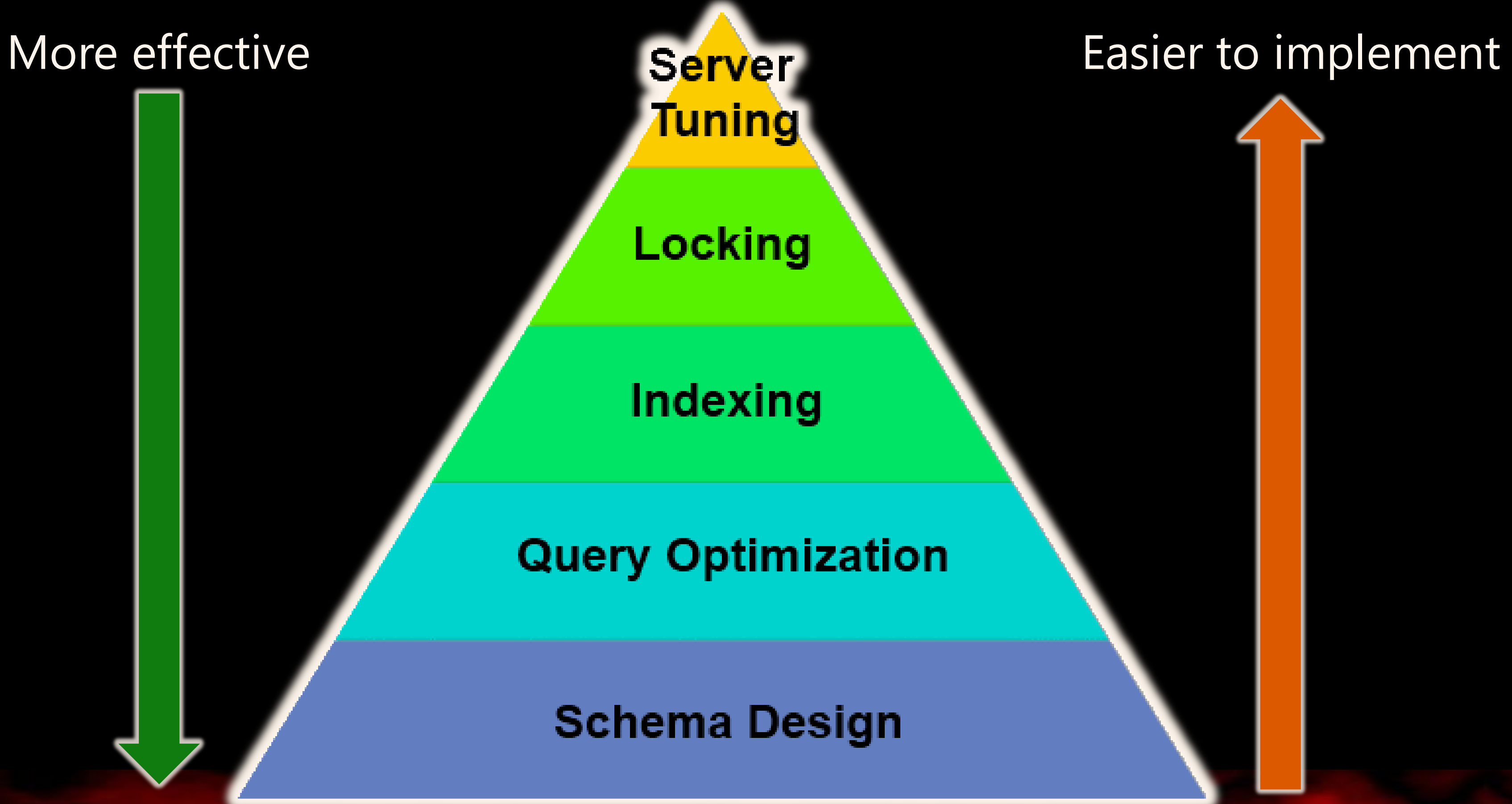
SQL Server Infernals

CIRCLE 12:

Performance Killers

SQL Server Infernals

Performance Tuning



Query Optimization Worst Practices

RBAR: Row By Agonizing Row

- *Cursors*
- *WHILE loops*
- *App-side cursors*
- *Scalar and multi-statement functions*

Query Optimization Worst Practices

Views on views on views…

Might look like a brilliant idea at first (code re-use FTW!)

- You can end up losing control
- Unneeded multiple accesses to the same tables
- Unnecessary JOINs

Query Optimization Worst Practices

- One query to rule them all
  - The optimizer is good, not perfect
  - «divide et impera» delivers better performance
- DISTINCT in all queries
  - … because “who wants stinkin’ duplicates?”
- Query HINTs all over the place
  - Especially index hints

Indexing Worst Practices

- Accepting all suggestions from Tuning Advisor
- Duplicate indexes
- An index for each column
  - *Indexes are not for free!*
- Suboptimal Clustered Index
  - Unique
  - Small
  - Unchanging
  - Ever increasing or decreasing

$\longrightarrow$

```
NEWSEQUENTIALID()
NEWID()
```

Server Tuning Worst Practices

- «*Throwing HW*» at the problem
  - *A 2x faster machine might make RBAR code 2x faster*
  - *Using set-based code might make it 500x faster*
- Using «advanced» options without testing
  - *NT Fibers (lightweight pooling)*
  - *Priority Boost*

Resources

Detailed blog posts on spaghettidba.com

One post for each circle:

https://spaghettidba.com/category/sql-server/sql-server-infernals/

SQL Server Infernals

Resources

Free Tool:

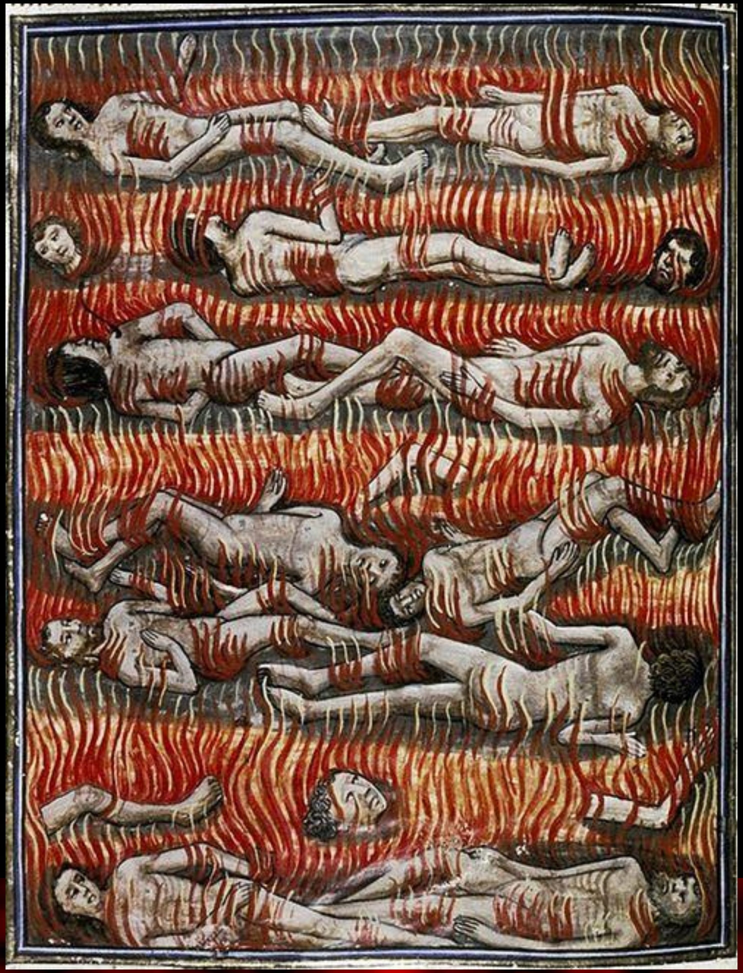
Best Practices Analyzer

- Highlights configuration parameters that don't comply with best practices
- Highlights potential problems
- Offers recommendations

http://www.microsoft.com/en-us/download/details.aspx?id=15289

SQL Server Infernals BINGO!

- Tweet your score with the **#GroupBy** hashtag
- Post your score on Slack in the **#groupby** channel
- You win nothing, but it's fun ☺



SQL Server Infernals

A Beginner's Guide to SQL Server Worst Practices

https://groupby.org/conference-session-abstracts/sql-server-infernals/

Contact:

spaghettidba@sqlconsulting.it

More infernal stuff:

https://spaghettidba.com/category/sql-server/sql-server-infernals/

SQL Server Infernals