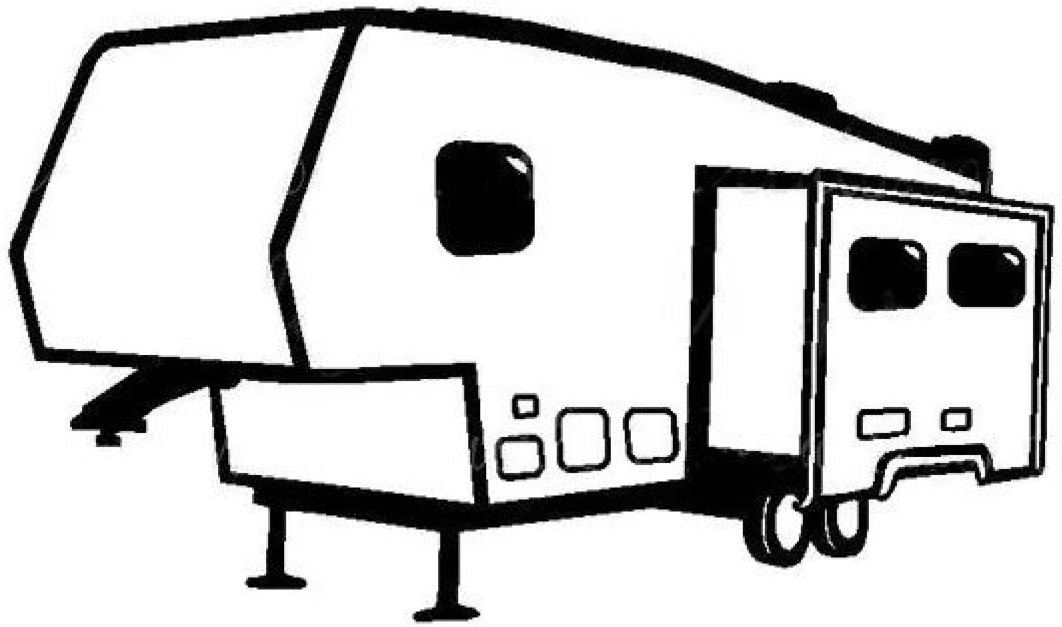
Aaron N. Cutshall, DHA, MSHI

SQL RV http://www.sqlrv.com



Defining What's Normal – The Basics of Database Normalization

Just who is this guy?

SQL RV

healthfirst
Health Insurance for New Yorkers
Enterprise Data Architect

PASS SQLSATURDAY
Speaker – various events

B.S.
Computer
Science

M.S.
Computer
Information
Systems

M.S.
Health
Informatics

VIRGINIA UNIVERSITY OF LYNCHBURG
1886
SIBI AUXILIUM • ET LIBERTAS

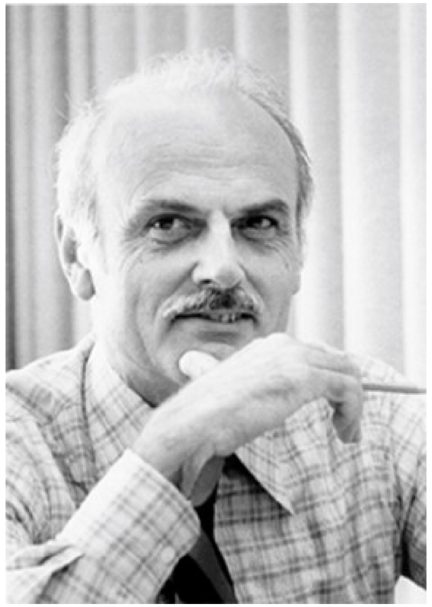
Doctorate in
Healthcare
Administration

35+
Years

Agenda

- What is Database Normalization?
  - Use Cases
  - Goals
- Normal Forms
- Denormalization
  - Reasons
  - Methods
  - Drawbacks
- Closing Thoughts

Something to consider…



credit

“[Every] non-key [attribute] must provide a fact about the key, the whole key, and nothing but the key.”

– Edgar Frank “Ted” Codd 1923-2003

Considered the “Father” of Relational Databases

Database Normalization

- Definition:
  - Reduces data volume for inserts/updates
  - Reduces data anomalies and errors
  - Improves data integrity
- Use Cases
  - For highly transactional systems
  - Situations where space is a consideration
  - Situations where the width of rows is an issue

First Normal Form (1NF)

- Rule 1: Every column in the table must be unique
  - Do not use multiple fields in a single table to store similar data
    - Example: Email1, Email2, etc.
    - Causes problems with additional values
  - This holds true even if the column names indicate their purpose
    - Example: WorkPhone, CellPhone, etc.
    - Does not accommodate new types

First Normal Form (1NF)

- Rule 2: Separate tables must be created for each set of related data
  - Create a separate table to handle multiple-column instances
    - Example: tblContact, tblEmail, tblPhone, etc.
    - Allows for a variable number of values without modifications to table structure or programming
  - Group related columns together
    - Example: EmailAddress, EmailType, etc.
    - Allows for logical grouping of related data

First Normal Form (1NF)

- Rule 3: Each table must be identified by a unique primary key
  - Primary key is an identifier that is unique for each row
  - The primary key may consist of one or more columns of relevant data (Natural Key)
    - Example: LastName, FirstName
  - Create a column of unique values specifically to identify each individual record (Surrogate Key)
    - Example: ContactId

First Normal Form (1NF)

- Rule 4: No rows may be duplicated
  - Each row should be unique
  - If multiple columns repeat data, it may need to be separated into a new table
- Rule 5: No columns may be duplicated
  - Multiple columns should not store the same values
  - Columns that repeat for each row are redundant

First Normal Form (1NF)

- Rule 6: No row/column intersections contain a NULL
  - Some methods do not allow for any NULL values
  - They require default values to indicate no known value
  - Not usually practical is most database implementations
- Rule 7: No row/column intersections contain multivalued fields
  - Only one value per column in a row
  - Do not store multiple values in a single field in a row
    - Example: Email – abc@def.com, ghi@jkl.org
    - Does not allow discrete values of one per field/record

First Normal Form (1NF)

tblContact

FirstName	LastName	WorkPhone	CellPhone	Email1	Email2
Jim	Jones	(555) 123-4567	(555) 123-7654	jjones@abc.com	jim.jones@gmail.com
Sally	Smith	(555) 987-6543	(555) 987-3456	ssmith@def.org	sally123@yahoo.com, sss1999@gmail.com
Bobby	Brown		(555) 321-0987, (555) 321-0123	bobbyb@rr.net	

tblContact

ContactId	FirstName	LastName
1	Jim	Jones
2	Sally	Smith
3	Bobby	Brown

tblPhone

PhoneId	PhoneType	PhoneNbr
1	Work	(555) 123-4567
2	Cell	(555) 123-7654
3	Work	(555) 987-6543
4	Cell	(555) 987-3456
5	Cell	(555) 321-0987
6	Cell	(555) 321-0123

tblEmail

EmailId	EmailType	EmailAddr
1	Work	jjones@abc.com
2	Personal	jim.jones@gmail.com
3	Work	ssmith@def.org
4	Personal	sally123@yahoo.com
5	Personal	sss1999@gmail.com
6	Personal	bobbyb@rr.net

Second Normal Form (2NF)

- Rule 1: Create separate tables for sets of values that apply to multiple records
  - Values that are used in multiple tables should be in separate tables
    - Example: PhoneType, EmailType, etc.
  - Ensures that values are consistent in various uses
  - Maintains data that is identified by a primary key as specific to that key

Second Normal Form (2NF)

- Rule 2: Associate related tables via a foreign key
  - A foreign key is a stored reference to another table’s primary key
    - Example: ContactId in tblPhone references ContactId in tblContact
  - It ensures that rows in one table have corresponding rows in another
  - A foreign key does not have to be unique as multiple rows may reference the same primary key
  - A foreign key may be NULL if it’s not applicable

Second Normal Form (2NF)

tblContact

ContactId	FirstName	LastName
1	Jim	Jones
2	Sally	Smith
3	Bobby	Brown

tblPhone

PhoneId	PhoneType	PhoneNbr
1	Work	(555) 123-4567
2	Cell	(555) 123-7654
3	Work	(555) 987-6543
4	Cell	(555) 987-3456
5	Cell	(555) 321-0987
6	Cell	(555) 321-0123

tblEmail

EmailId	EmailType	EmailAddr
1	Work	jjones@abc.com
2	Personal	jim.jones@gmail.com
3	Work	ssmith@def.org
4	Personal	sally123@yahoo.com
5	Personal	sss1999@gmail.com
6	Personal	bobbyb@rr.net

tblContact

ContactId	FirstName	LastName
1	Jim	Jones
2	Sally	Smith
3	Bobby	Brown

tblPhone

PhoneId	ContactId	PhoneType	PhoneNbr
1	1	Work	(555) 123-4567
2	1	Cell	(555) 123-7654
3	2	Work	(555) 987-6543
4	2	Cell	(555) 987-3456
5	3	Cell	(555) 321-0987
6	3	Cell	(555) 321-0123

tblEmail

EmailId	ContactId	EmailType	EmailAddr
1	1	Work	jjones@abc.com
2	1	Personal	jim.jones@gmail.com
3	2	Work	ssmith@def.org
4	2	Personal	sally123@yahoo.com
5	2	Personal	sss1999@gmail.com
6	3	Personal	bobbyb@rr.net

Third Normal Form (3NF)

- Rule 1: Eliminate fields that do not depend on the primary key
  - Values in a record not part of that record’s key do not belong in the table
  - Do not store data that is better suited in another table
    - Example: PhoneType and EmailType values should be defined in their own tables
  - Caveat: This is where discretion should be used to balance strict adherence with performance issues

Third Normal Form (3NF)

tblContact

ContactId	FirstName	LastName
1	Jim	Jones
2	Sally	Smith
3	Bobby	Brown

tblPhone

PhoneId	ContactId	PhoneType	PhoneNbr
1	1	Work	(555) 123-4567
2	1	Cell	(555) 123-7654
3	2	Work	(555) 987-6543
4	2	Cell	(555) 987-3456
5	3	Cell	(555) 321-0987
6	3	Cell	(555) 321-0123

tblEmail

EmailId	ContactId	EmailType	EmailAddr
1	1	Work	jjones@abc.com
2	1	Personal	jim.jones@gmail.com
3	2	Work	ssmith@def.org
4	2	Personal	sally123@yahoo.com
5	2	Personal	sss1999@gmail.com
6	3	Personal	bobbyb@rr.net

tblContact

ContactId	FirstName	LastName
1	Jim	Jones
2	Sally	Smith
3	Bobby	Brown

tblPhoneType

pTypeId	Description
1	Work
2	Cell

tblPhone

PhoneId	ContactId	pTypeId	PhoneNbr
1	1	1	(555) 123-4567
2	1	2	(555) 123-7654
3	2	1	(555) 987-6543
4	2	2	(555) 987-3456
5	3	2	(555) 321-0987
6	3	2	(555) 321-0123

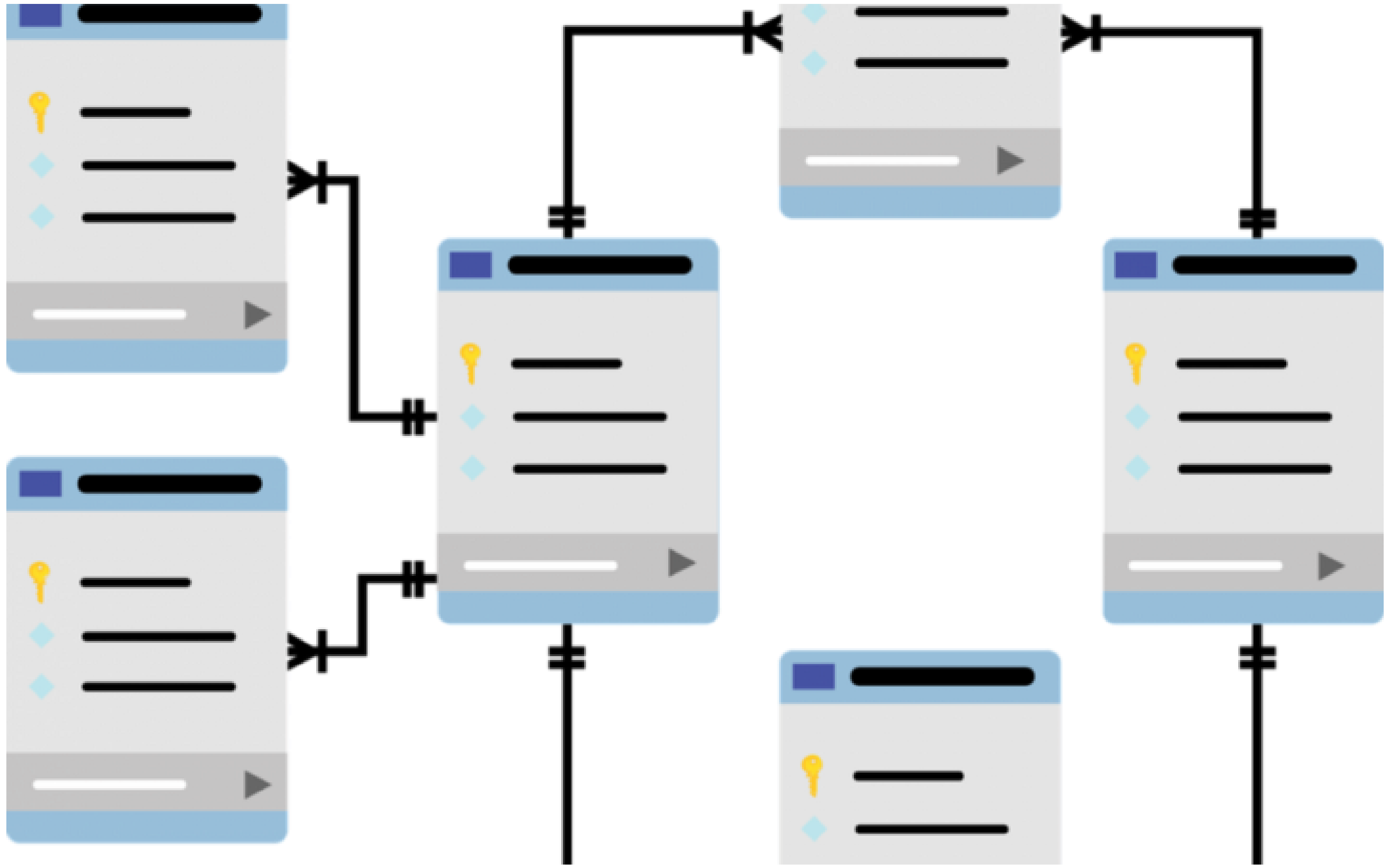
tblEmail

EmailId	ContactId	eTypeId	EmailAddr
1	1	1	jjones@abc.com
2	1	2	jim.jones@gmail.com
3	2	1	ssmith@def.org
4	2	2	sally123@yahoo.com
5	2	2	sss1999@gmail.com
6	3	2	bobbyb@rr.net

Other Normalization Forms

- Boyce-Codd Normal Form (BCNF)
  - Sometimes called 3.5 Normal Form
  - No more than one Candidate Key
- Fourth Normal Form (4NF)
  - No table contains two or more, independent and multivalued data
- Fifth Normal Form (5NF)
  - Cannot be decomposed into any number of smaller tables
- Sixth Normal Form (6NF)
  - Not standardized or well defined at this point

Overview of Database Normalization



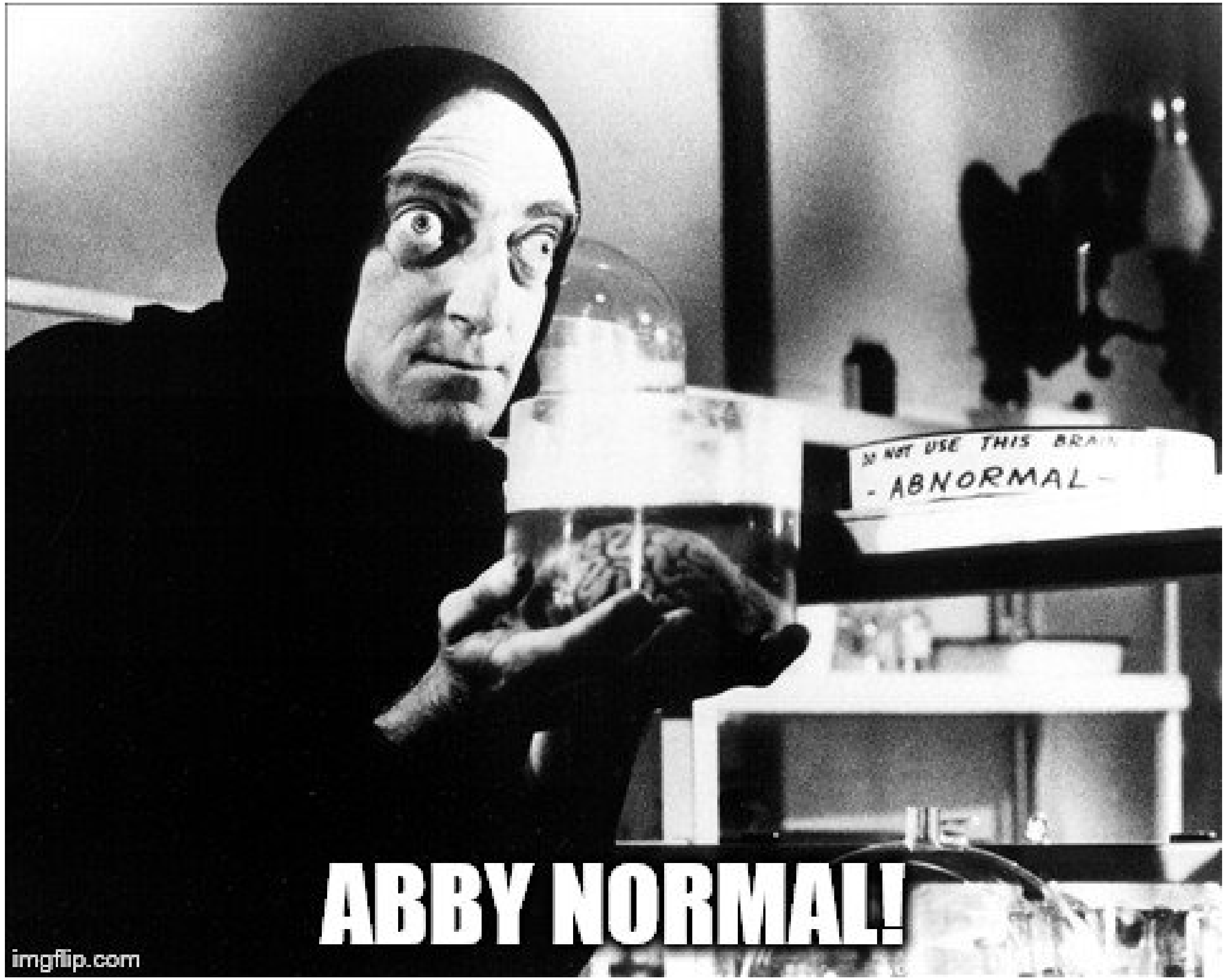
The Key, the whole Key, and nothing but the Key, so help me Codd!

Overview of Database Normalization

- Goals:
  - Reduce data redundancy
  - Reduce data discrepancies
  - Provide a single source of truth
  - Improve data integrity
- Exceptions:
  - Too many small tables can hinder performance
  - Taken too far normalization can increase complexity
  - Keep the application purpose in mind

Something to consider…

Database Denormalization



Database Denormalization

- Denormalization is optimized for data reads
  - Too many tables complicates queries
  - Keeping data duplicates reduces the number of tables in queries
- Normalization is still important
  - Should be followed first
  - Critical for data entry and processing
- Denormalization should only be followed under careful and strict circumstances

Database Denormalization Reasons

1. Maintain history
   - Store historical changes to data
     - Method #1: Multiple time-based records in the same table
     - Method #2: Maintain a separate historical data table
   - Be able to reproduce data as of a given date
2. Enhance query performance
   - Reduce the number of joins used in a query
   - Useful for rarely changing data (ex. Lookups)
   - Combine often used queries into a fixed result set

Database Denormalization Reasons

3. Facilitate and accelerate reporting
    - Pre-aggregate data often used for reports
        - Totals, averages, statistics, etc.
        - Typically refreshed over time as needed when data is updated
    - Reduces impact on source systems
    - Provide data for multiple users with lower costs
4. Simplify database management
    - Pre-calculate values to avoid on-the-fly calculations
    - Avoid logic from being implemented in multiple places
    - Perform complex data manipulations in ETL

Database Denormalization Reasons

5. Store derivable data
    - When derived values are frequently needed
    - When calculations are not updated often
6. Use pre-joined tables
    - Store results of frequently used queries or joins
    - Best when most current data is not required
7. Store lookup or reference values
    - Eliminates the needs for lookup or reference tables
    - Use when source values are not updated often

Database Denormalization Methods

8. Keep details in a master record
    - Best for small number of detail records per master
    - When master-detail records are queried together
    - Store detail record data with master
        - Current record ID, first & last records, etc. for quick reference
        - Store summaries, totals, and aggregates of detail in master
9. Keep master values in detail records
    - Handy for multiple detail records and few master fields
    - Eliminates the needs to read master record

Database Denormalization Drawbacks

1. Extra storage space
   - Additional columns for de-duplicated data
   - Additional rows for historical or lowered grains
   - Most often stored on separate servers to minimize impact to source systems
2. Additional documentation
   - Capture the reasons for the deduplications
   - Describe the additional ETL requirements
   - Data transformations can be difficult to document

Database Denormalization Drawbacks

3. Potential data anomalies
    - Data can become out of sync and impact reliability
    - Errors can result from partial data updates
4. Additional code
    - ETL logic for derived values or aggregates
    - Pre-joined data, lookups and complex transformations
5. Slower inserts/updates
    - ETL operations take more time with more data
    - Data only as current as last update

Common Denormalization Scenarios

1. Name parts
   - Last, First, Middle, Suffix, Prefix, multiple names, etc.
   - Normalization calls for separate table for name parts
   - Most systems use fixed set (first & last)
   - Not intended to fit all scenarios (i.e. multiple names)
2. Address
   - Capture multiple types in multiple instances
   - Sometimes in a separate table (like Amazon)
   - If only one 1 or 2 are needed, often not normalized
   - Address lines often kept at 2-3 with fixed names

What We Covered

- What is Database Normalization?
  - Use Cases
  - Goals
- Normal Forms
- Denormalization
  - Reasons
  - Methods
  - Drawbacks

Closing Thoughts

- Most tables should be at least 2NF
- While desirable, use 3NF when appropriate
  - May not always be practical in application
  - A database heavily normalized may impede performance
  - Strike a balance between normalized and denormalized
- Design with normalization in mind first

Closing Thoughts

- Perform proper analysis on performance
  - Understand the logical design
  - Focus only on parts that need help
  - Analyze how often data is changed
  - Study performance issues and fine-tune queries first
  - Normalize first then denormalize only where necessary
- Experience (trial and error) will be your guide
- *Normalize until it hurts, denormalize until it works!*

References

- Normalization:
  - https://www.itprotoday.com/sql-server/sql-design-why-you-need-database-normalization
  - http://www.informit.com/articles/article.aspx?p=30646
  - http://www.bkent.net/Doc/simple5.htm
  - https://www.guru99.com/database-normalization.html
- Denormalization
  - https://rubygarage.org/blog/database-denormalization-with-examples
  - https://www.vertabelo.com/blog/denormalization-when-why-and-how/

Questions & Comments

BONUS:

A **TON** of free eBooks from Microsoft, RedGate and SentryOne!

PRESENTATION FEEDBACK:

- Your thoughts needed
- Improve presentations
- Make this event even more valuable!!!

Aaron N. Cutshall

@sqlrv

www.linkedin.com/in/sqlrv

aaron@sqlrv.com

www.sqlrv.com

www.youtube.com/@sqlrv