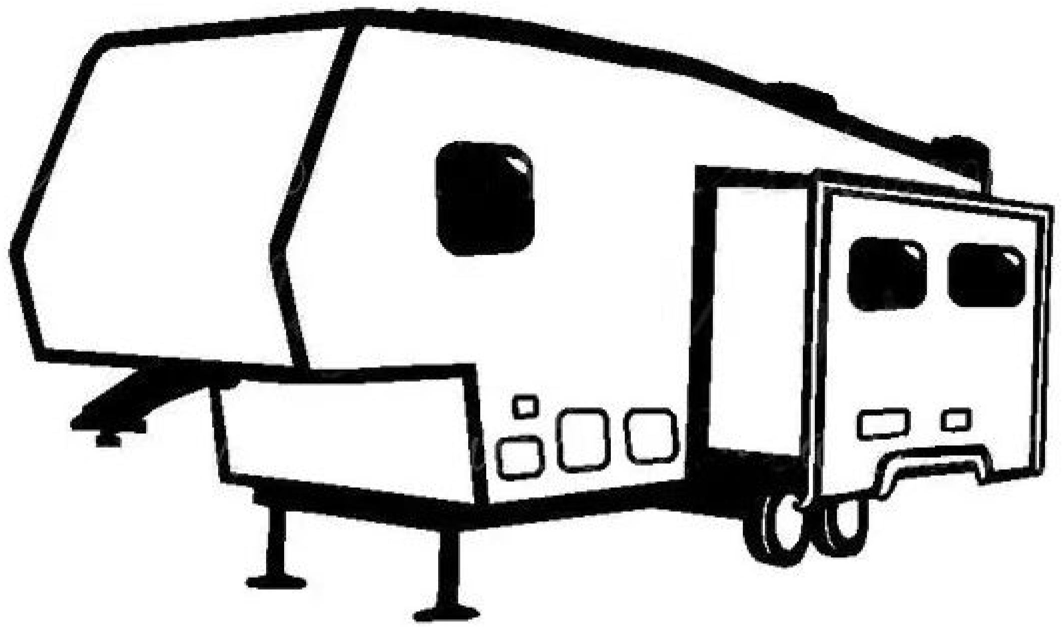
Aaron N. Cutshall, DHA, MSHI, ACHE

SQL RV http://www.sqlrv.com



An Enterprise Data Architecture Framework

Just who is this guy?

SQL RV

Director, Data Services

Since 2023

Speaker – various events

B.S.
Computer
Science

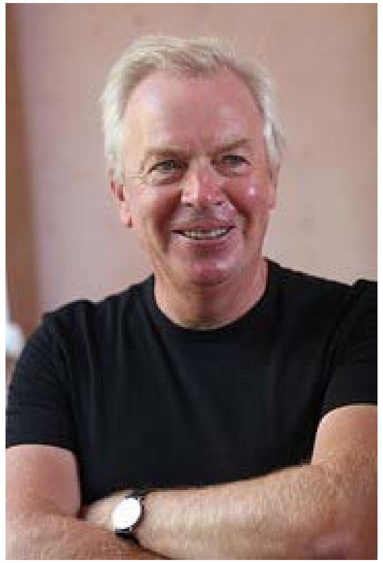
M.S.
Computer
Information
Systems

M.S.
Health
Informatics

Doctor of
Healthcare
Administration

35+
Years

Something to consider…


credit

“The difference between good and bad architecture is the time you spend on it.”

– Sir David Chipperfield, Celebrated British Architect

Introduction

- Data architectural challenges
  - Maintaining consistency
  - Data from disparate sources
  - Confusion on terms
- Enterprise Level Framework
  - Transform data from multiple sources
  - Homogenize into a consistent environment
  - Be the single source of truth

Agenda

- Review some challenges
- Operational Data Store
    - Definition
    - Data Zones
    - Data Migration
    - Data Standards
- Key Management
- Closing Thoughts

Review Some Challenges

- Multiple data sources
  - Various databases
    - RDBMS sources (SQL Server, Postgres, Oracle, etc.)
    - Vendor-specific structures may not meld
    - Potentially limited access to data
  - Flat files from outside sources
    - Governmental agencies (state and federal)
    - Trading partners or third-party agencies
  - Data captured from APIs
    - Usually from closed or protected systems
    - Not always conformant to specifications or complete

Review Some Challenges

- Different data grains
  - Concept easy to understand but difficult to explain
  - Levels at which data is unique
  - Defines the relationship with other data
  - Example – Cars:
    - Make – Unique to a manufacturer (Ex. Chevrolet)
    - Model – Unique within a manufacturer (Ex. Colorado)
    - Trim – Variations of a model (Ex. Z71)
    - Type (vehicle body style) – Classifications used across manufacturers
      - Pickup
      - Crew Cab

Operational Data Store – Definition

- Meant for current operational data, not historical
- Transactional like, but with some denormalization
- Not considered strategic reporting
- Usually stored in an RDBMS (such as Postgres)
- Captures data more frequently than a data warehouse
- Data is scrubbed and resolved for redundancy
- Performs compliance and other business rules
- Provides a cleaner source for the data warehouse

Operational Data Store – Definition

- Multi-source data collection
  - Data acquisition in raw
  - Data conformation in refined
  - Data integration in published
- Data standardization
  - Conforms data to specific data domains
  - Consistent naming standards and data types
  - Results in data homogenization regardless of source
  - Permits queries with data from disparate sources
- Unified source of truth

Operational Data Store – Data Zones

- Source (Data Origination)
  - Data as pulled from various sources (RDBMs, Flat File, API)
  - No transformations other than potentially adding audits
  - Usually stored as files in a Data Lake (such as AWS S3)
- Raw (Data Acquisition)
  - Data loaded into tables structured like the source system
  - Only minimal transformations (such as dates)
  - Allows for later SQL transformations

Operational Data Store – Data Zones

- Refined (Data Conformation)
  - Tables patterned after the published model but source-specific
  - Perform transformations to meet standards
  - Tables and columns fit the standards
- Published (Data Integration)
  - Data is consolidated from refined tables
  - Views created against published tables
  - Data is presented uniformly regardless of the source

Operational Data Store – Data Migration

- Dev – Development
  - May be local or common
  - Nothing reliable – Data is typically old or manufactured
  - Totally in the control of developers
  - Deployment packages prepared for ST
- ST – System Testing
  - Central testing environment more stable than dev
  - Data is relatively static and staged for testing scenarios
  - Controlled by developers but should be treated like RT

Operational Data Store – Data Migration

- RT – Release Testing (UAT – User Acceptance Testing)
  - Central testing environment for Quality Assurance
  - Data may be a mix of staged and production data
  - Treated like prod to test deployments and their effect
  - Deployment, validation, and rollback scripts are all tested
- Prod – Production
  - Deployments are regularly scheduled
  - Performed by separate dedicated release teams
  - Requires validation and rollback scripts

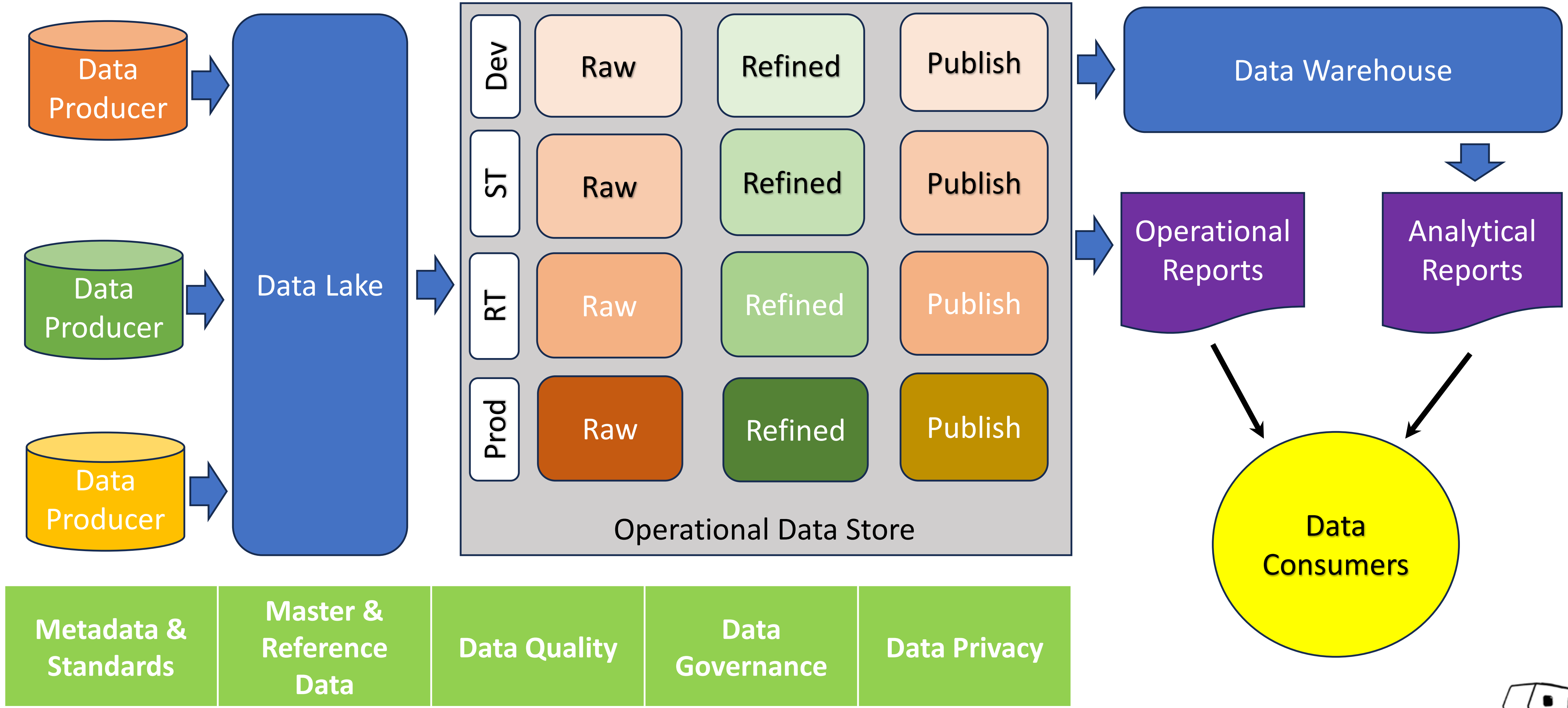
Operational Data Store – Data Standards

- Data Domains for consistency
  - Defined during data architecture
  - Enforced through development
  - Example: Enumeration Codes are all varchar(16)
- Naming Standards
  - Have a dictionary of logical to physical name translations
  - Utilize class names to represent data domain types
  - Define a strict formulation for names
  - Exceptions may be required but should minimal

Operational Data Store – Data Standards

- Have table structure rules
  - Ensure consistency when tables are created
  - Logical placement of columns within a table
  - Example: All FKs follow PK according to dependency
- Apply naming standards regardless of source
  - Each source may have a different name or data type
  - Ensure consistency and improve readability
  - Example: empid from Source A and emplnum from Source B both become employee_id_cd in Published database

Operational Data Store – Review



Key Management – Issues

- Data from multiple sources have differing primary keys
- Combine data from multiple sources into the same table with only one key
- May need to load data in any order or even in parallel
- Build FK references without requiring lookups
- Maintain consistency in key management
- Allow for easier foreign key management

Key Management – Source Key Text

- Pipe Delimited
  - Use vertical pipes to delimit each portion of the key text
  - SYSTEM_CODE|TABLE_FILE_NAME|BUSINESS_KEY
- Part 1: Source System Code
  - Unique code (<=8 characters) Ex. MHS, TRUCARE, EYEMED
  - Data can come from database system tables or flat
- Part 2: Source System Table/File
  - Table name from the source database system
  - Consistent portion of the file name (w/o date/time)

Key Management – Source Key Text

- Part 3: Business or Natural Key(s)
  - One or more data fields that, when combined, uniquely identify the record
  - Each portion is pipe delimited
  - NULL or missing values are treated as empty strings
  - Should be documented in Source-to-Target-Mapping document and in Data Model

Key Management – Source Key Text

- Examples (Authorization):
  - CCMS|EVENT|XC97902B|500014
    - Source System: CCMS
    - Source Table: EVENT
    - Business Key: XC97902B and 500014
  - DAUTH|CCNA|CCNAA189787566
  - OAUTH|EVICORE-ONC|CCNAA202804132
  - TRUCARE|UMAUTHORIZATION|t0lof9q3-3348-2608-6452-57758000034

Key Management – Hash Key Definition

- Hash key is generated from a known text value
- An SHA-256 hash ***always*** generates a char(64) [fixed] result
- It ***always*** produces the same value from the same source string
- Same value produced regardless of database system
- Reduces a natural key of potentially one or more variable length strings to a single fixed-length value

Key Management – Generate Hash Key

- Prepare a value to be hashed (`source_key_text`)
  - Postgres:

    ```
    encode(digest(source_key_txt, 'SHA256'),'hex’)
    ```
  - SQL Server:

    ```
    convert(char(64),hashbytes('sha2_256',source_key_txt),2)
    ```
  - Oracle:

    ```
    dbms.crypto(utl_raw.cast_to_raw(source_key_txt),'SHA256')
    ```
  - SparkSQL and Snowflake:

    ```
    sha2(source_key_txt, 256)
    ```
- All will produce the same result given the same input

Key Management – Hash Key Benefits

- Indexes and PK/FKs work best with fixed-length keys
- FK values are built the same as a reference PK using the same natural key values
- Without FK constraints, records can be loaded in any order or even in parallel
  - Construct an FK using the same rules as a PK for the referenced table
  - No lookups are required to build an FK reference
  - Note: FKs are usually not used in ODS or EDW environments
    - Generally, they reflect data already created
    - Are not typically data creators themselves like a transactional system

Key Management – Hash Key Benefits

- Allows multiple sources of data with varying natural keys to be in the same table with only one key
  - Using natural keys may require variable number and variable length fields for uniqueness
  - Permits the same rules to apply to data regardless of source
- Supports data correlation across different data systems
  - Example: Key generated on SQL Server will be same as on Postgres
- Superior to a traditional surrogate key
  - Sequence numbers are only valid in one DB (Prod Key <> Dev Key)
  - Loading the same data always produces the same key (unlike identity)
  - Supports traceability back to the original system

Key Management – Hash Key Myths

- “Using a Hash or GUID as Primary Key is a bad idea because it causes Index Fragmentation and frequent Page Splits.”
  - This is generally an assumption without good proof
  - See Jeff Moden’s video on the subject
- “Hash values cannot guarantee against collisions.”
  - Small hashes like MD5 may have collisions
  - SHA-256 is sufficiently large enough
  - Odds are greater of having a balanced and efficient national budget
- “Sequence surrogates are smaller, simpler, and just better.”
  - Show me the proof, please.

Review

- Review some challenges
- Operational Data Store
  - Definition
  - Data Zones
  - Data Migration
  - Data Standards
- Key Management – Hash Key

Closing Thoughts

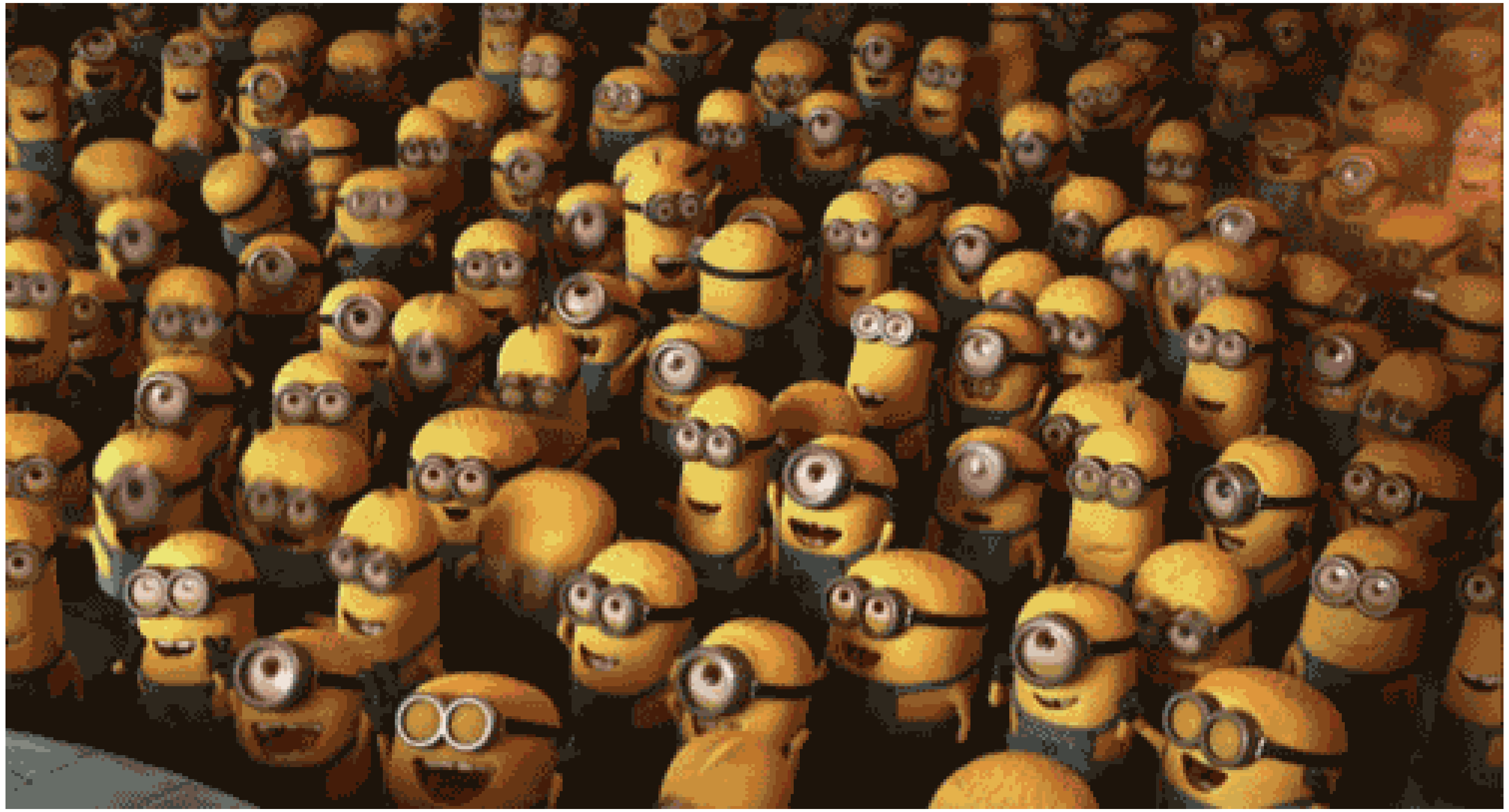
- Data is messy and often needs help to organize it
- An ODS setup consolidates data from multiple systems
- Supports transactional-like reporting with some denormalization for better performance
- Permits reports & analysis on disparate data
- Use of Hash Keys allows efficient consolidation of data
  - Different sources all in the same table
  - Across different servers/environments

References

- Operational Data Store:
  - Operational Data Store (ODS)
  - What is an Operational Data Store: A Complete Guide for 2024
  - Operational Data Store (ODS): An Overview and Use Cases
- Hash Primary Keys:
  - Hash keys, the unsung hero of data warehousing - Part 1
  - Hash keys, the unsung hero of data warehousing - Part 2
  - Potential whitepaper/book opportunity

Thanks to our Sponsors!

Without them, we wouldn't be here!!



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