Data Modeler's Guide to Implement Kimball Dimensional modeling on Amazon Redshift

Suhas Hanumanthaiah

suhas.h@hotmail.com

Abstract:
This paper presents a comprehensive guide for implementing Kimball's dimensional modeling methodology within Amazon Redshift, a fully managed cloud data warehouse platform. As enterprises increasingly adopt cloud-based analytics solutions, adapting proven data warehousing techniques to the unique architectural constraints and capabilities of Redshift becomes essential. The paper begins by outlining the foundational principles of Kimball dimensional modeling, including star schema design, the use of fact and dimension tables, surrogate keys, conformed dimensions, and slowly changing dimensions. It then explores how these concepts can be practically applied within Redshift's distributed architecture using best practices in table design, such as the optimal use of sort keys, distribution styles, and compression encoding. Specific attention is given to leveraging Redshift's features—like columnar storage, SUPER data types for handling semi-structured data, and materialized views—to enhance query performance and support modern BI requirements. The guide also addresses common challenges, such as the lack of enforced referential integrity and how to overcome them with robust ETL/ELT processes. By combining theoretical modeling concepts with actionable Redshift-specific strategies, this research empowers data modelers, architects, and engineers to build scalable, performant, and maintainable data warehouse solutions. Ultimately, it bridges the gap between classical BI methodologies and modern cloud-native data infrastructure.

Keywords: Dimensional Modeling, Amazon Redshift, Star Schema, Data Warehouse Architecture, Business Intelligence.

1. INTRODUCTION

In today's data-driven landscape, the ability to deliver fast, reliable, and scalable analytics is critical for business success. Dimensional modeling—particularly the approach advocated by Ralph Kimball—has become a cornerstone of modern data warehousing and business intelligence (BI) solutions [1]. Kimball's methodology emphasizes simplicity, understandability, and performance, allowing organizations to construct data models that align closely with business processes and are accessible to non-technical stakeholders.

As cloud adoption accelerates, enterprises are increasingly turning to platforms like Amazon Redshift to support large-scale analytics and reporting needs. Amazon Redshift, a fully managed, cloud-native, petabyte-scale data warehouse service, brings together the scalability and elasticity of cloud infrastructure with the power of Massively Parallel Processing (MPP). However, Redshift introduces unique design considerations that must be addressed when implementing traditional modeling techniques such as the star schema.

This guide bridges the gap between classical dimensional modeling and its implementation on Amazon Redshift. It explores how Kimball's principles can be adapted to leverage Redshift's columnar architecture, distribution and sort key strategies, compression encoding, and support for semi-structured data via SUPER data types [9]. Additionally, it outlines how to handle practical challenges, such as the lack of enforced referential integrity and the need for efficient ETL/ELT pipelines.

By providing practical insights, design recommendations, and Redshift-specific implementations of dimensional components—such as fact and dimension tables, conformed dimensions, and slowly changing dimensions—this paper serves as a valuable reference for data modelers seeking to design performant, scalable, and maintainable data warehouses in the cloud.

2. KIMBALL DIMENSIONAL MODELING

Ralph Kimball's approach to dimensional modeling is a methodology for designing data warehouses and business intelligence systems, focusing on creating simple, understandable, and performant data structures for analytical purposes. Dimensional models focus on process measuring events, dividing data into either measurements (Fact table) or the "who, what, where, when, why, and how" descriptive context (Dimension) [1].

Dimensional models can be instantiated in both relational databases, referred to as star schemas, or multidimensional databases, known as online analytical processing (OLAP) cubes. Star schemas characteristically consist of fact tables linked to associated dimension tables via primary/foreign key relationships. OLAP cubes can be equivalent in content to, or more often derived from, a relational star schema. An OLAP cube contains dimensional attributes and facts, but it is accessed via languages with more analytic capabilities than SQL, such as XMLA. OLAP cubes are included in this list of basic techniques because a cube is often the final deployment step of a dimensional DW/BI system, or may exist as an aggregate structure based on a more atomic relational star schema [1].

Here's are the key aspects of Kimball's dimensional modeling:

2.1. Focus on Business Processes

- The Kimball approach emphasizes modeling data around business processes, like sales or inventory, which aligns with how business users think about their operations.
- It's a bottom-up approach, focusing on implementing individual business processes first and then integrating them.

2.2. Emphasis on Simplicity and Performance

- Kimball's approach prioritizes simplicity and ease of use for business users, making the data model intuitive and easy to understand.
- It aims for fast query performance by using denormalized structures (like the star schema) and minimizing complex joins.

2.3. Core Components

- Fact Tables: Store quantitative data (measures) that represent the outcomes of business processes. Examples include sales revenue, quantity sold, or profit margins.
- Dimension Tables: Provide descriptive context (attributes) to the facts, answering questions like "who, what, where, when, and why". Examples include customer, product, and time information.
- Star Schema: The most common structure, where a central fact table is directly linked to multiple dimension tables. This offers simplicity and faster querying.
- Grain: The grain establishes exactly what a single fact table row represents. The grain declaration becomes a binding contract on the design. The grain must be declared before choosing dimensions or facts because every candidate dimension or fact must be consistent with the grain. This consistency enforces a uniformity on all dimensional designs that is critical to BI application performance and ease of use.
- Surrogate Key: A unique auto number field that represents the Grain. All dimension tables are built with surrogate key as primary key. Dimension's surrogate key is referenced through Foreign Key relationship in Fact table for efficient query plans.

2.4. Conformed Dimensions:

- Conformed dimensions are a crucial concept, ensuring data consistency across different fact tables and allowing for integrated analysis.
- A conformed dimension means that a single dimension table, like a customer dimension, is shared across multiple star schemas, guaranteeing that the same data item is used consistently.

In essence, Kimball dimensional modeling focuses on building data warehouses that are easy to use and understand for business users, while also optimizing for quick query performance.

3. OVERVIEW OF AMAZON REDSHIFT

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift Serverless lets you access and analyze data without the usual configurations of a provisioned data warehouse. Resources are automatically provisioned and data warehouse capacity is intelligently scaled to deliver fast performance for even the most demanding and unpredictable workloads.

When designing table in Amazon Redshift, following design options will need to be considered:

3.1. Sort Key

Amazon Redshift stores data on disk in sorted order according to the sort key. Sort Key helps query optimizer choose optimal query plan. Default is AUTO where Redshift choose the appropriate sort order. Custom Sort Key needs to be explicitly defined during table creation. One or more columns can be selected for sort key.[7] There are two types of sort keys:

- *Compound Sort Keys:* Default Sort Type where the table is sorted primarily by the first column in the sort key, then by the second and so on. They are suitable when queries frequently filter on the leading columns of the sort key.
- *Interleaved Sort Keys:* Interleaved sort keys provide equal weight to all columns in the sort key. It is suitable when queries frequently filter on different columns in sort key, and the order of filtering varies.

The data type of a sort key column can be: BOOLEAN, REAL, DOUBLE PRECISION, SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIME, TIMETZ, TIMESTAMP, or TIMESTAMPTZ, CHAR, or VARCHAR.

3.2. Data Distribution

An Amazon Redshift cluster is a set of nodes as shown in Fig 1.1. Each node in the cluster has its own operating system, dedicated memory, and dedicated disk storage. One node is the leader node, which manages the distribution of data and query processing tasks to the compute nodes. The compute nodes provide resources to do those tasks. [3]

The disk storage for a compute node is divided into a number of slices. The number of slices per node depends on the node size of the cluster. The nodes all participate in running parallel queries, working on data that is distributed as evenly as possible across the slices. This in essence is the Massively Parallel processing (MPP) architecture that supports parallel execution of queries.

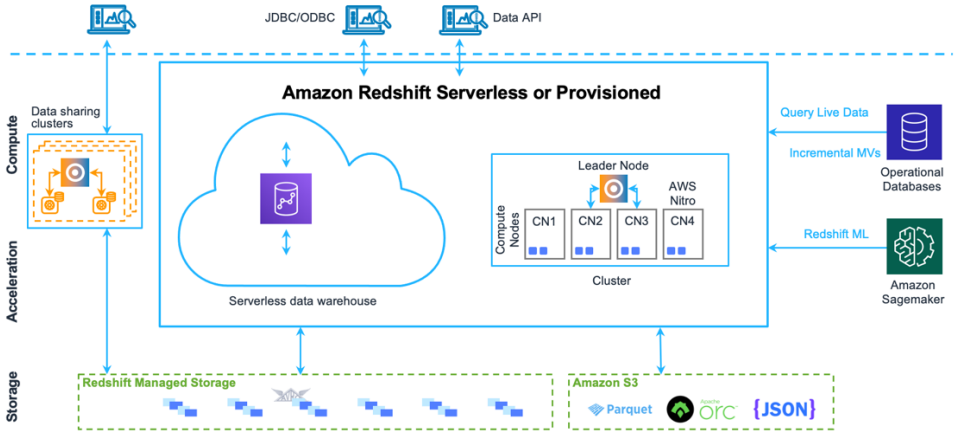

Fig 1.1: Data warehouse system architecture [2]

When you load data into a table, Amazon Redshift distributes the rows of the table to each of the node slices according to the table's distribution style [3]. As part of a query plan, the optimizer determines where blocks of data must be located to best run the query. The data is then physically moved, or redistributed, while the query runs. Redistribution might involve either sending specific rows to nodes for joining or broadcasting an entire table to all of the nodes.

Data redistribution can account for a substantial portion of the cost of a query plan, and the network traffic it generates can affect other database operations and slow overall system performance. To the extent that you anticipate where best to locate data initially, you can minimize the impact of data redistribution.

Amazon Redshift supports following Distribution Styles [3]:

- ALL distribution – A copy of entire table is distributed to every node
- EVEN distribution – The leader node distributes the rows across the slices in a round-robin fashion, regardless of the data inside any column.
- KEY distribution – The rows are distributed according to the values in one column. The leader node places matching values on the same node slice. If you distribute a pair of tables on the joining keys, the leader node collocates the rows on the slices according to the values in the joining columns. This way,

matching values from the common columns are physically stored together. DISTKEY must be defined when creating table to use KEY distribution. DISTKEY is set for only one column per table.

- AUTO distribution – Default distribution. Redshift assigns optimal distribution style based on the size of the table data.
  - For small tables, Redshift chooses ALL distribution
  - For large tables, Redshift chooses KEY distribution based on Primary Key if defined
  - For large tables with no primary key defined and none of the columns qualify for key then Redshift chooses EVEN distribution

3.3. Data Types

In table design data type of a column plays crucial role in performance measures. It is recommended to limit the width of the column to finite size based on requirement. In addition to basic data types, Redshift supports SUPER data type – It supports storage of semi-structured data or documents as values. PartiQL can be used to access data points from SUPER data type column which stores data as schema-less arrays and structures that contain Amazon Redshift scalars and possibly nested arrays and structures.[9][10]

3.4. Collation

Amazon Redshift supports two collation settings of CASE_SENSITIVE or CASE_INSENSITIVE at individual column level or database level. Amazon Redshift doesn't support locale-specific or user-defined collation sequences. In general, the results of any predicate in any context could be affected by the lack of locale-specific rules for sorting and comparing data values. For example, ORDER BY expressions and functions such as MIN, MAX, and RANK return results based on binary UTF8 ordering of the data that does not take locale-specific characters into account.

3.5. Compression Encoding

Compression encoding specifies the type of compression that is applied to a column of data values as rows are added to a table. Compression reduces the column size drastically which improves query performance since moving data between nodes will be reduced. With default ENCODE AUTO settings on table, Redshift assigns following compression encoding [4]:

- Columns that are defined as sort keys are assigned RAW compression.
- Columns that are defined as BOOLEAN, REAL, or DOUBLE PRECISION data types are assigned RAW compression.
- Columns that are defined as SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIMESTAMP, or TIMESTAMPTZ data types are assigned AZ64 compression.
- Columns that are defined as CHAR or VARCHAR data types are assigned LZO compression.

Best practices for Comparing and choosing Encoding:

- Use ANALYZE COMPRESSION tool for recommendations of storage space savings as compared to RAW for all columns. To apply recommendation, ALTER TABLE script should be used.
- Consider ZSTD for Most Data Types: Zstandard (zstd) is a powerful, open-source compression algorithm that generally offers high compression ratios and good performance across diverse datasets and data types. It's often recommended by ANALYZE COMPRESSION.
- Consider LZ0 for Long string values
- Consider AZ64 for numeric values
- Avoid Compressing on Sort Key Columns: Compressing on sort key columns is not recommended, particularly the first column in a compound sort key. High compression on sort keys can lead to inefficient range-restricted scans and degrade query performance.

4. KIMBALL DIMENSIONAL MODELING ON AMAZON REDSHIFT

Amazon Redshift uses columnar storage to store tabular data efficiently. With advanced datatypes in Redshift and their support in downstream BI applications like PowerBI, semi-structured data can be modeled using Kimball Dimensional modeling (further referred as Star Schema). We are going to discuss different characteristics of Star Schema and how to build it in Amazon Redshift below:

4.1. Dimension Table

Characteristics	Implementation in Amazon Redshift
Surrogate Key	Redshift support IDENTITY [8] which can populate auto number to build Surrogate Key
Primary Key	Although Amazon Redshift allows declaration of Primary Key, it is not enforced. So, the Data Engineering Tool has to enforce the Referential Integrity
Natural Key	Natural Keys are required fields. This is enforced by added constraint of NOT NULL
Multiple Hierarchy	For efficient query execution, *Interleaved Sort Keys* need to be defined for columns that constitute the Hierarchy. If only a single defined Hierarchy exists, then conventional *Compound Sort Keys* would perform

Table 1: General Dimension characteristics and how to achieve it in Amazon Redshift

4.1.1. Conformed Dimension

Dimension tables conform when attributes in separate dimension tables have the same column names and domain contents. Information from separate fact tables can be combined in a single report by using conformed dimension attributes that are associated with each fact table. When a conformed attribute is used as the row header (that is, the grouping column in the SQL query), the results from the separate fact tables can be aligned on the same rows in a drill-across report. This is the essence of integration in an enterprise DW/ BI system. Conformed dimensions, defined once in collaboration with the business's data governance representatives, are reused across fact tables; they deliver both analytic consistency and reduced future development costs because the wheel is not repeatedly recreated. [5]

Characteristics	Implementation in Amazon Redshift
Surrogate Key	• Redshift support IDENTITY [8] which can populate auto number to build Surrogate Key for primary conformed dimension. • For other dimensions in similar grain, the surrogate key is looked up to primary dimension by Natural Key
Primary Key	Although Amazon Redshift allows declaration of Primary Key, it is not enforced. So, the Data Engineering Tool has to enforce the Referential Integrity. Additionally, for other dimension with similar grain, ETL/ELT tool must populate all required data from primary conformed dimension
Sort Key	Sort Keys are recommended for columns that go through range filtering or equality filtering
Distribution Style	Use AUTO distribution for large/wide dimension tables Use ALL distribution for small dimension that is frequently used. Example, Date dimension or Junk dimensions
Compression Encoding	RAW for Sort Key columns LZ0 for columns with long string values ZSTD for other columns
Ragged Hierarchy	Use Redshift's Super data type to model Hierarchy data as an array object

Table 2: Conformed Dimension characteristics and how to achieve them in Amazon Redshift

4.1.2. Slowly Changing Dimension (SCD)

Based on how frequently the attributes in dimension changes, there are following types of Slowly Changing Dimension:

- Type 0 – The dimension attribute value never changes. Type 0 is appropriate for attributes labeled "original". Example Social security number, original credit score, most date dimension attributes
- Type 1 – the old attribute value is overwritten with new value. So, no history is saved. Type 1 is appropriate for attributes labeled "last". Although this approach is easy to implement and does not create additional dimension rows, it complicates aggregated Fact tables which may be required to re-calculate.

- Type 2 – The old attribute value's validity is updated by maintaining effective begin date and effective end date. New data is added as new row with new effective begin date.
- Type 3 – type 3 changes add a new attribute in the dimension to preserve the old attribute value; the new value overwrites the main attribute as in a type 1 change.

Characteristics	Implementation in Amazon Redshift
Surrogate Key	• Redshift support IDENTITY [8] which can populate auto number • For Type 2, Surrogate Key provides a unique way of mapping Dimension attributes as of date from Fact table • Surrogate Key should be defined as the Primary Key of the table
Natural Key	For Type 2, Natural Key plays critical role in identifying changes in ETL/ELT process.
Sort Key	Natural Key and effective dates (begin date and end date) should be used as Interleaved Sort Keys.
Distribution Style	Use KEY distribution for large/wide dimension (Type 2) tables based on low cardinality column Use ALL distribution for small dimension (Type 0, 1, 3) that is frequently used. Example, Date dimension or Junk dimensions
Compression Encoding	RAW for Sort Key columns LZ0 for columns with long string values ZSTD for other columns
Late Arriving Dimension [12]	Create placeholder entry on Dimension table if Natural Key is not found. This can be accomplished through ETL/ELT process or Batch process in Amazon Redshift
Dynamic Value Banding [13]	Amazon Redshift supports creation of View to build dynamic value banding. Materialized View [11] supports better performance at the cost of additional storage.

Table 3: SCD Dimension characteristics and how to achieve them in Amazon Redshift

4.2. Fact Table

A fact table contains the numeric measures produced by an operational measurement event in the real world. At the lowest grain, a fact table row corresponds to a measurement event and vice versa. The numeric measures in a fact table fall into three categories. The most flexible and useful facts are fully *additive*; additive measures can be summed across any of the dimensions associated with the fact table. *Semi-additive* measures can be summed across some dimensions, but not all; balance amounts are common semi-additive facts because they are additive across all dimensions except time. Finally, some measures are completely *non-additive*, such as ratios.[6]

Characteristics	Implementation in Amazon Redshift
Surrogate Key	Redshift support IDENTITY [8] which can populate auto number to build Surrogate Key. Surrogate Key should be defined as the Primary Key of the table
Sort Key	Timestamp column that represents when event occurred for Measures and frequently used columns for joining should be used as *Interleaved Sort Keys*.
Distribution Style	Use *KEY* distribution for Fact tables based on low cardinality column which is frequently used to join Dimension to avoid skewing
Transactional/Snapshot	Generally, Fact tables are built on transactional data or Snapshot data. Amazon Redshift recommends to migrate datetime fields to one of datetime data types instead of storing as one of string data types for better performance.

Compression Encoding	RAW for Sort Key columns LZ0 for columns with long string values ZSTD for other columns

Tabel 4: General Fact characteristics and how to achieve it in Amazon Redshift

5. CONCLUSION

Implementing the Kimball dimensional modeling methodology on Amazon Redshift presents both significant opportunities and unique challenges. While the platform's distributed architecture and performance capabilities make it an excellent choice for data warehousing, the lack of enforced referential integrity constraints requires careful planning and robust ETL processes.

Key findings from this research include:

1. **Architectural Alignment**: Redshift's distributed architecture aligns well with dimensional modeling principles, particularly for star schema implementations.
2. **Performance Benefits**: Proper implementation of distribution and sort keys can yield significant performance improvements for analytical queries.
3. **Integrity Challenges**: The absence of enforced constraints necessitates alternative approaches to maintaining data quality and referential integrity.

The success of dimensional modeling implementations on Redshift depends heavily on understanding the platform's unique characteristics and adapting traditional methodologies to leverage its strengths while mitigating its limitations. As cloud-based data warehousing continues to evolve, ongoing research and practical experience will further refine these approaches and identify new optimization opportunities.

REFERENCES:

1. R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.
2. Amazon Web Services, “Amazon Redshift Architecture,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_architecture.html
3. Amazon Web Services, “Data Distribution Styles,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/t_Distributing_data.html#t_data_distribution_concepts
4. Amazon Web Services, “Compression Encodings,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/c_Compression_encodings.html#compression-encoding-list
5. The Kimball Group, “Conformed Dimension,” [Online]. Available: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/conformed-dimension/
6. R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013, ch. 6, pp. 145–160
7. Amazon Web Services, “Sort Keys,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/t_Sorting_data.html
8. Amazon Web Services, “Using IDENTITY Columns in Redshift,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/r_CREATE_TABLE_NEW.html
9. Amazon Web Services, “SUPER Data Type,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/super-data-type.html
10. Amazon Web Services, “PartiQL for Querying Semi-structured Data,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/querying-semistructured-data.html
11. Amazon Web Services, “Materialized Views,” [Online]. Available: https://docs.aws.amazon.com/redshift/latest/dg/materialized-view-overview.html
12. R. Kimball and M. Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd ed. Hoboken, NJ, USA: Wiley, 2013, ch. 17, pp. 453–456.
13. A. Gupta, V. Harinarayan, N. Rajaraman, and J. D. Ullman, "Index Selection for Dynamic Value Banding in Data Warehouses," in *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, Toronto, Canada, 2004, pp. 536–547. doi: 10.5555/1041410.1041468.

7. ABBREVIATIONS

Abbreviation	Full Form
BI	Business Intelligence
DW	Data Warehouse
OLAP	Online Analytical Processing
SQL	Structured Query Language
XMLA	XML for Analysis
MPP	Massively Parallel Processing
ETL	Extract, Transform, Load
ELT	Extract, Load, Transform
SCD	Slowly Changing Dimension
IDENTITY	Auto-increment field in Amazon Redshift
DISTKEY	Distribution Key
ENCODE	Compression Encoding
UTF8	Unicode Transformation Format 8-bit
ZSTD	Zstandard Compression
AZ64	Amazon Redshift 64-bit Compression
LZO	Lempel–Ziv–Oberhumer Compression
SUPER	Semi-structured Universal Persistent Entity Representation
PartiQL	SQL-compatible query language for semi-structured data