Agile Data Warehousing: Leveraging Vertical Partitioning for Enhanced Model Refactoring

Introduction

Vertical partitioning is an agile approach to data warehousing that offers benefits in terms of flexibility, scalability, and adaptability.

It allows for incremental growth of the data warehouse while minimizing disruption to existing solutions and accommodating changing requirements.

This approach can also be used to change Domain Data Models and gracefully adapt and extend them, with features "promoted" from derived Data Products. Consider the decision to upgrade some KPIs or other relevant attributes, defined at the Data Product level, to the Domain level because the business owners regarded them significant to the Domain and suitable for investigation in other Data Products.

This chapter provides an overview of the vertical partitioning approach.

CONSIDER PRESENT A CONCRETE CASE or provide images that clarify the procedure

Overview:

Vertical partitioning involves dividing a data warehouse into smaller, more manageable units based on specific criteria or attributes. Instead of a monolithic structure, the data warehouse is split vertically into separate "partitions", each containing a subset of columns or attributes from the fact and dimension tables.

Motivation and Benefits:

The key motivation behind vertical partitioning is to provide a more agile way to grow the data warehouse while minimizing the impact on existing solutions.

Some of the benefits of vertical partitioning include:

- Flexibility:
    - With vertical partitioning, new attributes or columns can be added to the data warehouse without impacting existing solutions.
    - This flexibility allows for easier adaptation to changing business requirements and the integration of new data sources.
- Scalability:
    - By dividing the data warehouse into smaller partitions, vertical partitioning improves scalability.
    - It enables the efficient management of large volumes of data and supports distributed processing across different partitions, leading to improved performance, although this is not as significant with a Column Oriented database like BigQuery.
- Refactoring Support:
    - Vertical partitioning facilitates the refactoring of the database structure by allowing for changes to be made incrementally and in a more controlled way
    - Existing partitions can remain unchanged while new partitions are introduced or modified, reducing the complexity and risk associated with large-scale database refactoring efforts.
- Performance Optimization:
    - Vertical partitioning can improve query performance by reducing the amount of data accessed during a query.
    - By selecting only the relevant partitions for a specific query, the overall response time can be significantly improved, although this is not as significant with a Column Oriented database like BigQuery.
- Improve data management and better change (and historical data) management in a SCD Type 2 table.
    - Some attributes of a given Data Object (for example, Customer) can be grouped by "velocity" of change and isolated on dedicated tables with comparable behaviour attributes (similare to the approach to generate different "Sattelites" in Data Vault Methodology)

Implementation:

The implementation of vertical partitioning involves identifying the key attributes or columns that can be logically grouped together into separate partitions.

These "partitions" can be based on factors such as data volatility, access patterns, or business requirements.

Two immediate solutions can be used depending on the granularity of the resulting segregation and the corresponding attributes:

- If the "new" Data Object has the same granularity as the original Data Object from which it was derived, one can simply use the same key – this should be the case for Fact tables.
    - scenarios:
        - add new attributes from the sources or from other derived objects, such as a Data Product
        - group less utilized attributes in a large table
- If the granularity is substantial less, one can apply a distinct on such characteristics, generate a key for it, and use that key to refer back to the original Data Object - this should be the case with Dimension tables.
    - In such a case, concatenating all existing attributes (including a separator character to improve safety and limit the likelihood of key collision) and applying a hash algorithm to the resulting concatenation is a rapid and safe technique to generate a key.
    - This technique can be applied in several possible scenarios, as:

        - Similar characteristics in an existing table that experience varying rates of change (velocity of change) from the others
        - isolate a group of flags or minor attributes on a Data Object (ver o nome q o kimbal deu an isto outright...)
        - Isolate Data Object's frequent and important features, that can be used to classify or describe data on other Data Models.
            - For instance, social-economic factors can be used to categorize customer-related data in a Fact table.
        - group up less-used characteristics in a big table.

Considerations:

When adopting vertical partitioning, it's important to consider the following:

- Partitioning Criteria:
    - Selecting appropriate partitioning criteria is crucial to ensure efficient data organization and access.
    - The criteria should align with the specific requirements and usage patterns of the data warehouse.
- Data Distribution:
    - Distributing the data across partitions should be done in a way that balances the workload and optimizes query performance.
    - Several techniques can be used to achieve an optimal distribution.
- Maintenance and Administration:
    - Vertical partitioning introduces additional management and administration overhead, as each partition needs to be maintained separately.
    - It requires careful monitoring and ongoing maintenance to ensure data consistency and performance.
- Query Optimization (depends on the type of technology used - in the case of BigQuery, is not very impactfull):
    - Query optimization is essential when working with vertically partitioned data.
    - Techniques such as query pruning, predicate pushdown, and intelligent query routing can be employed to optimize query execution and minimize data movement between partitions.

Conclusion:

Vertical partitioning offers an agile approach to growing a data warehouse, allowing for incremental changes and better adaptability to evolving requirements.

By dividing the data warehouse vertically into partitions, it provides flexibility, scalability, and the ability to refactor the database without impacting existing solutions. However, it requires careful planning, consideration of partitioning criteria, and ongoing maintenance to ensure optimal performance and data integrity. Vertical partitioning is a valuable technique for organizations seeking a more agile and scalable approach to data warehousing.