Efficient Transaction Processing in SAP HANA Database – The End of a Column Store Myth

Vishal Sikka
Franz Färber
Wolfgang Lehner
Sang Kyun Cha
Thomas Peh
Christof Bornhövd

Presented By
-Richa Desai

Abstract

- The overall goal of the SAP HANA database is to provide a generic but powerful system for different query scenarios, both transactional and analytical.
- Main features that differentiate the SAP HANA database from classical relational database engines and the concept of record life cycle management to use different storage formats for the different stages of a record.

Introduction

- Data management is the most challenging topics in today's software industry.
- On the system side, data management scenarios have become extremely complex and complicated to manage.
- An efficient, flexible, robust, and cost-effective data management layer is the core for a number of different application scenarios essential in today's business environments.

- Initially, classical ERP (Enterprise Resource Planning) systems were implemented as the information processing backbone that handles such application scenarios.
- Column-organized data structures gained more and more attention in the analytical domain to avoid projection of queried columns and exploit significantly better data compression rates.

- Thus , we make the following observations:

1)Usage Perspective

- SQL is no longer considered as the only appropriate interaction model for business applications.
- Need to optimally support an application layer with a tight coupling mechanism.
- Need for scripting languages with built-in database features for specific application domains like Pig to work on Hadoop installations.
- Need for a comprehensive support of domain-specific and proprietary query languages like SAP's FOX for financial planning scenarios

2) Cost awareness

- Provide a lower TCO (Total Cost of Ownership) solution for the complete data management stack ranging from hardware to setup costs to operational and maintenance costs.

3) Performance

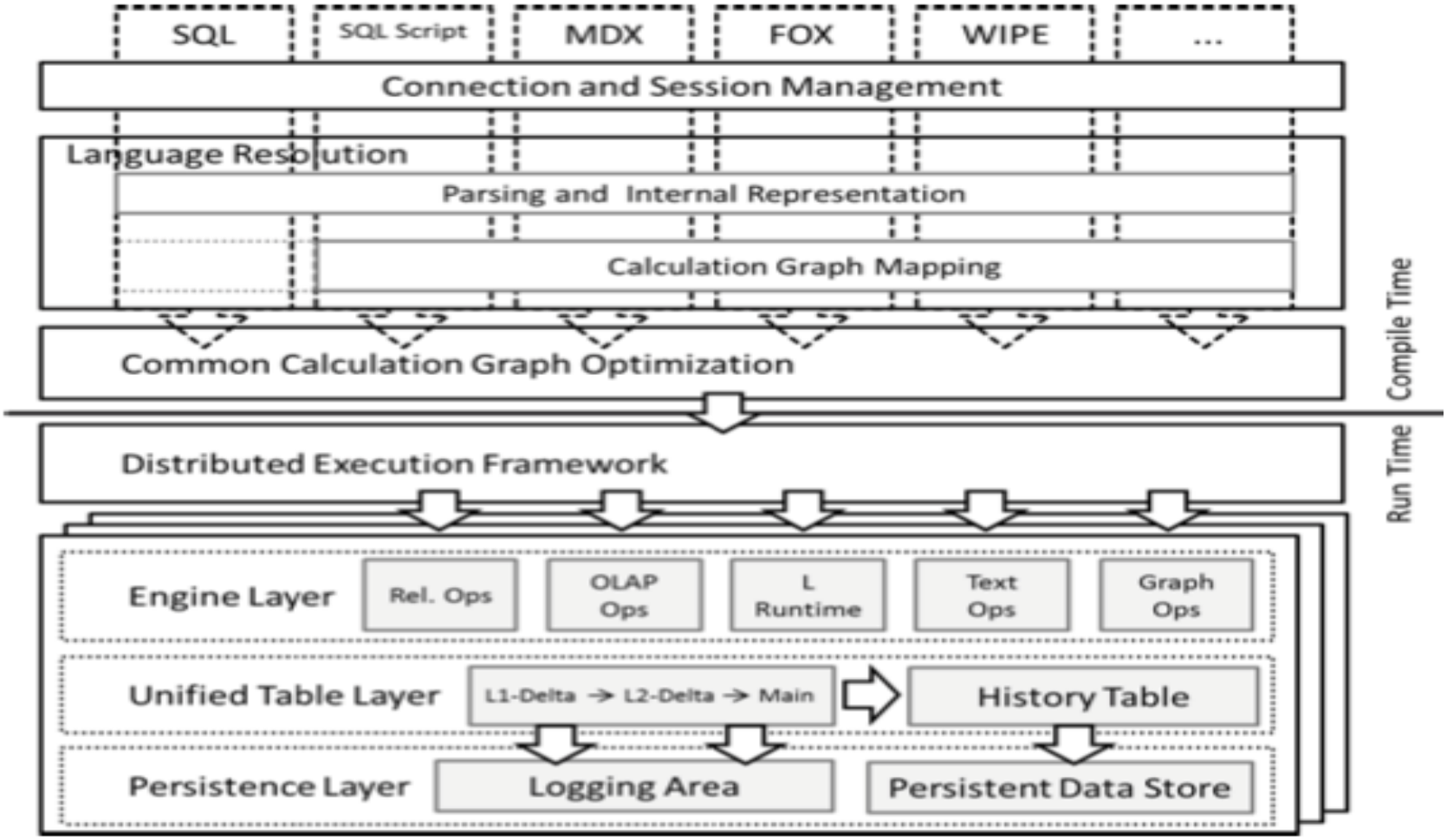
- It is still the main reason to use specialized systems.
- The challenge is to provide a flexible solution with the ability to use specialized operators or data structures whenever possible and needed.

Contribution and Outline:

- The main distinguishing features of the SAP HANA database for the scope of typical business applications are as follows:
  - The HANA database comprises a multi-engine query processing environment that offers different data abstractions supporting data of different degrees of structure.
  - It supports the representation of application-specific business objects (like OLAP cubes) and logic (domain specific function libraries) directly inside the database engine.

- HANA database is optimized to efficiently communicate between the data management and the application layer.
- The SAP HANA database supports the efficient processing of both transactional and analytical workloads on the same physical database.

Layered Architecture Of The Sap Hana Database:



- Different query languages can enter the system via a common connection and session management layer performing all infrastructural tasks with the outside world.
- In a first step, a query string is translated into an internal optimized representation (similar to an abstract syntax tree), which is local for every domain-specific language.
- In a second step, the query expression is mapped to a ”Calculation Graph”.

Calculation Graph Model:

- The “Calculation Graph Model” follows the classical data flow graph principle.
- Source nodes represent either persistent table structures or the outcome of other calc graphs.
- Inner nodes reflect logical operators consuming either one or multiple incoming data flows.

- The set of calc graph operators can be split into two groups of operator types.

1) The calc model defines a set of intrinsic operators

- For example: aggregation, projection, joins, union etc.
  - SQL for example can be completely mapped to this class of operators.

2) The calc model provides operators which implement core business algorithms like currency conversion or calendar functionality.

- In addition the calc model supports the following types of operators:
  - **Dynamic SQL nodes**

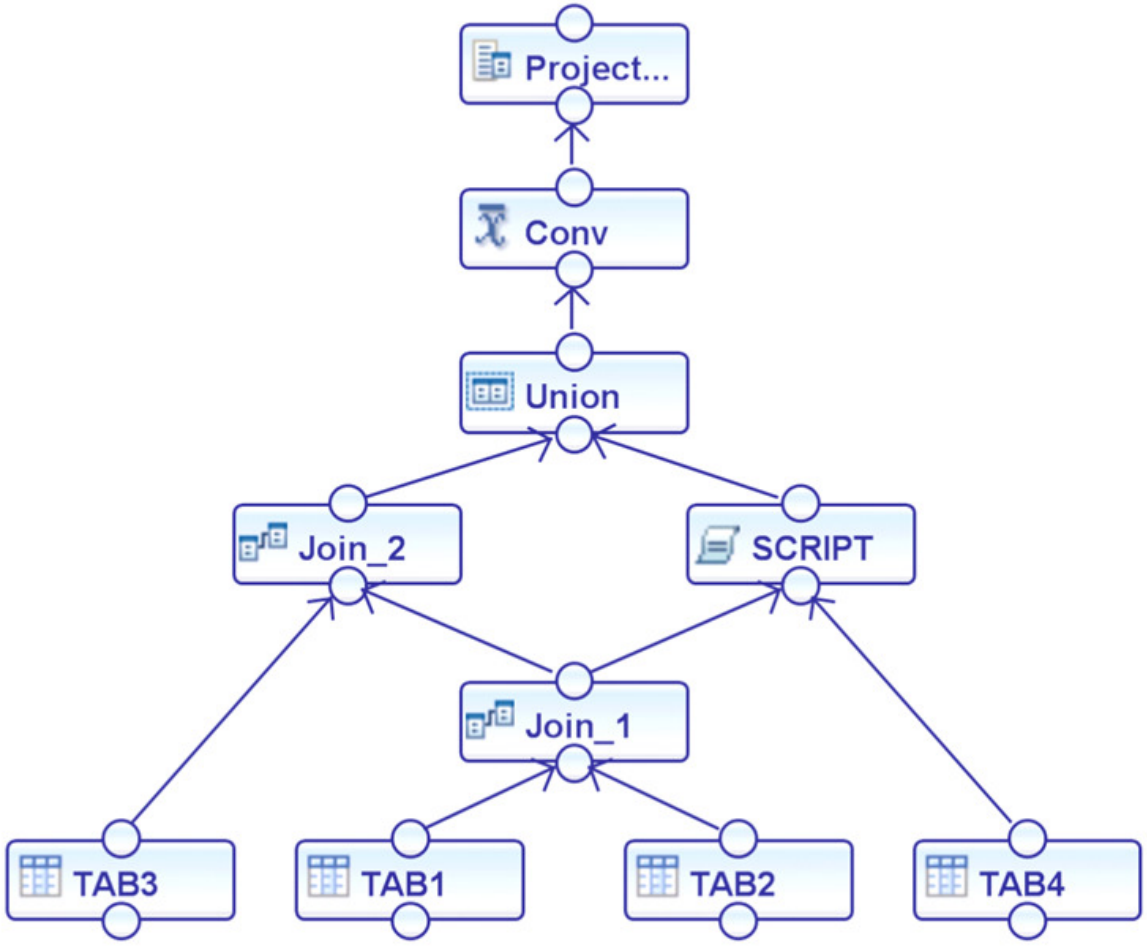
    A calc model operator may execute a complete SQL statement on the incoming data flow. The statement can be a parameter and compiled and executed at runtime of the calc graph.
  - **Custom nodes**

    A custom node may be used to implement domain-specific operators
  - **R nodes**

    An R node can be used to forward incoming data sets to an R execution environment.
  - **L nodes**

    The language L represents the internal runtime of the SAP HANA database.
  - The calc model provides “split” and “combine” operators which dynamically defines and re-distributes partitions of data flows as a base construct.

Example of SAP HANA Calc Model Graph:



Calc Graph Compilation and Execution:

- Once the query scripts are mapped to a data flow graph in the calc model, the optimizer runs optimization procedures to restructure and transform the logical plan into a physical plan.

- During optimization, the fragments of the data-flow graph are mapped to physical operators provided by the “Engine Layer”.

- The Engine layer itself consists of a collection of different physical operators.

- SAP HANA database provides the following set of operators:

  - **Relational Operators**
  - Handles classic relational query graph processing

  - **OLAP operators**
  - They are optimized for star-join scenarios with fact and dimension tables.

  - **L runtime**
  - Reflects the building block to execute L code, represents the given calc graph as L nodes.

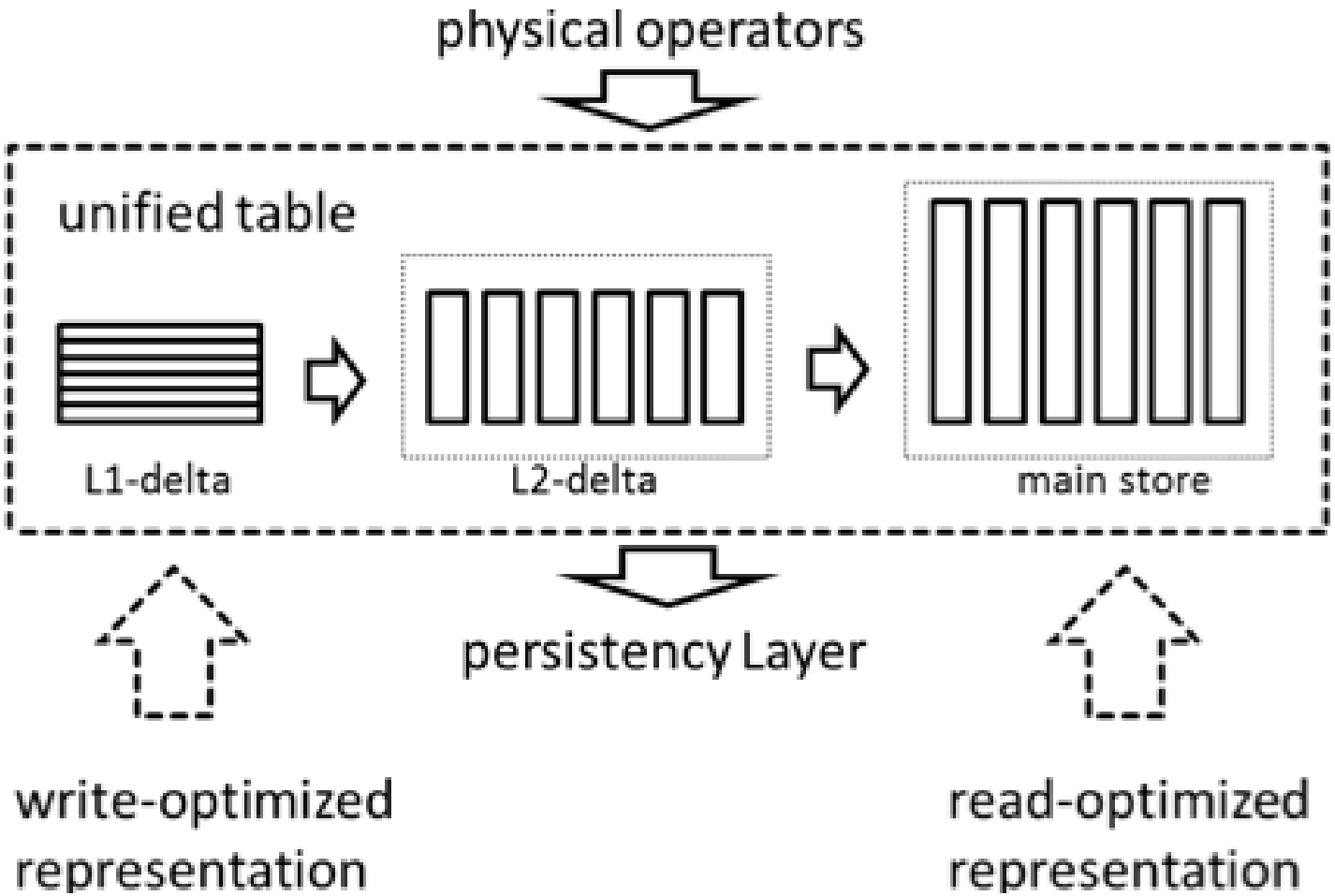
  - **Text operators**
  - Comprises the set of functionality already available in SAP Enterprise Search to deliver comprehensive text analysis features.

  - **Graph operators**
  - Provides support for graph-based algorithms, to efficiently implement complex resource planning scenarios or social network analysis tasks.

Lifecycle Management of Database Records :

- SAP HANA database uses ‘unified table structure’, to provide data access for all applicable physical operators.
- This is a key differentiator to classical database architectures.
- SAP HANA conceptually propagates records through different stages of a physical representation.

Overview of the Unified Table Concept:

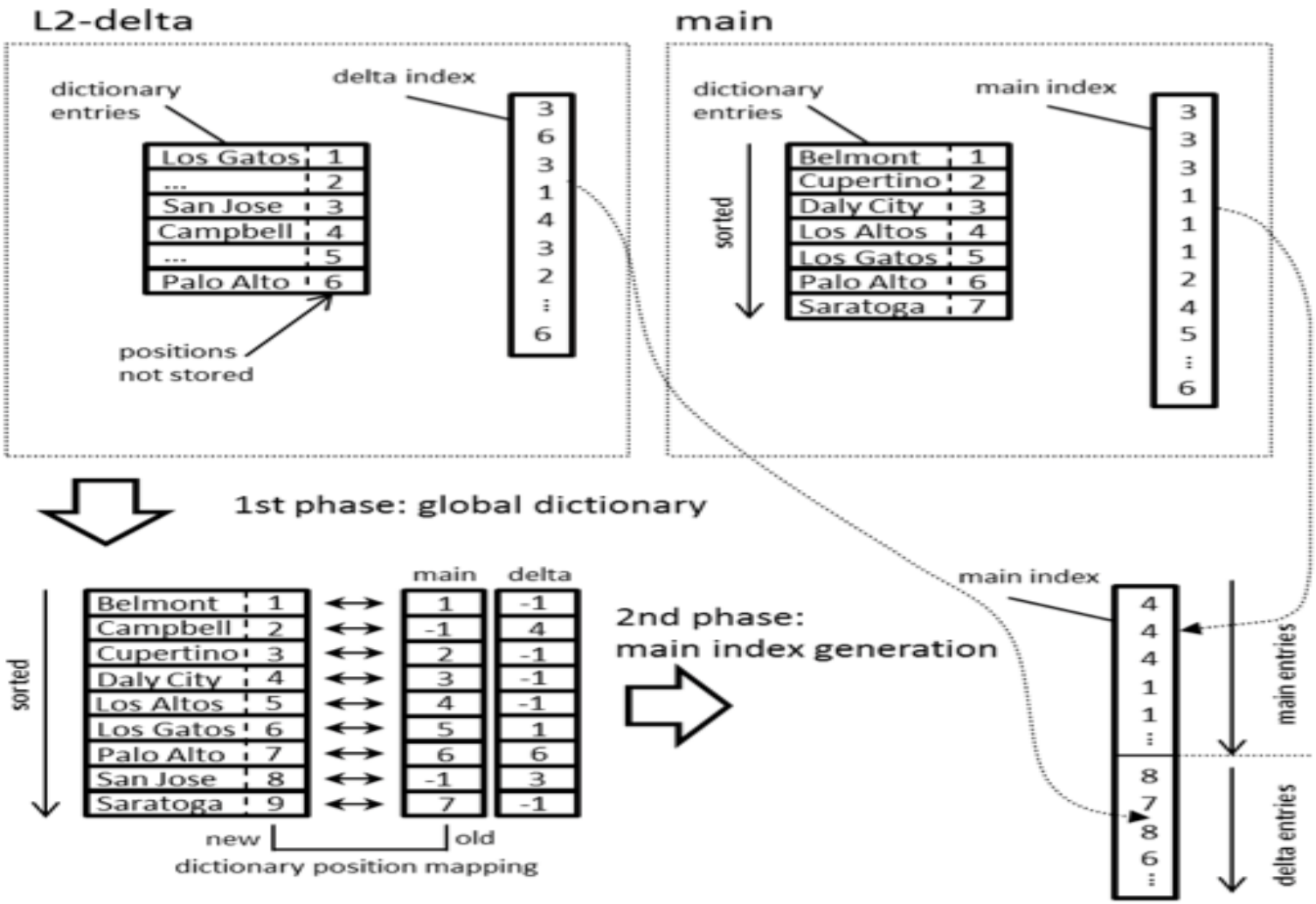


Unified Table Access:

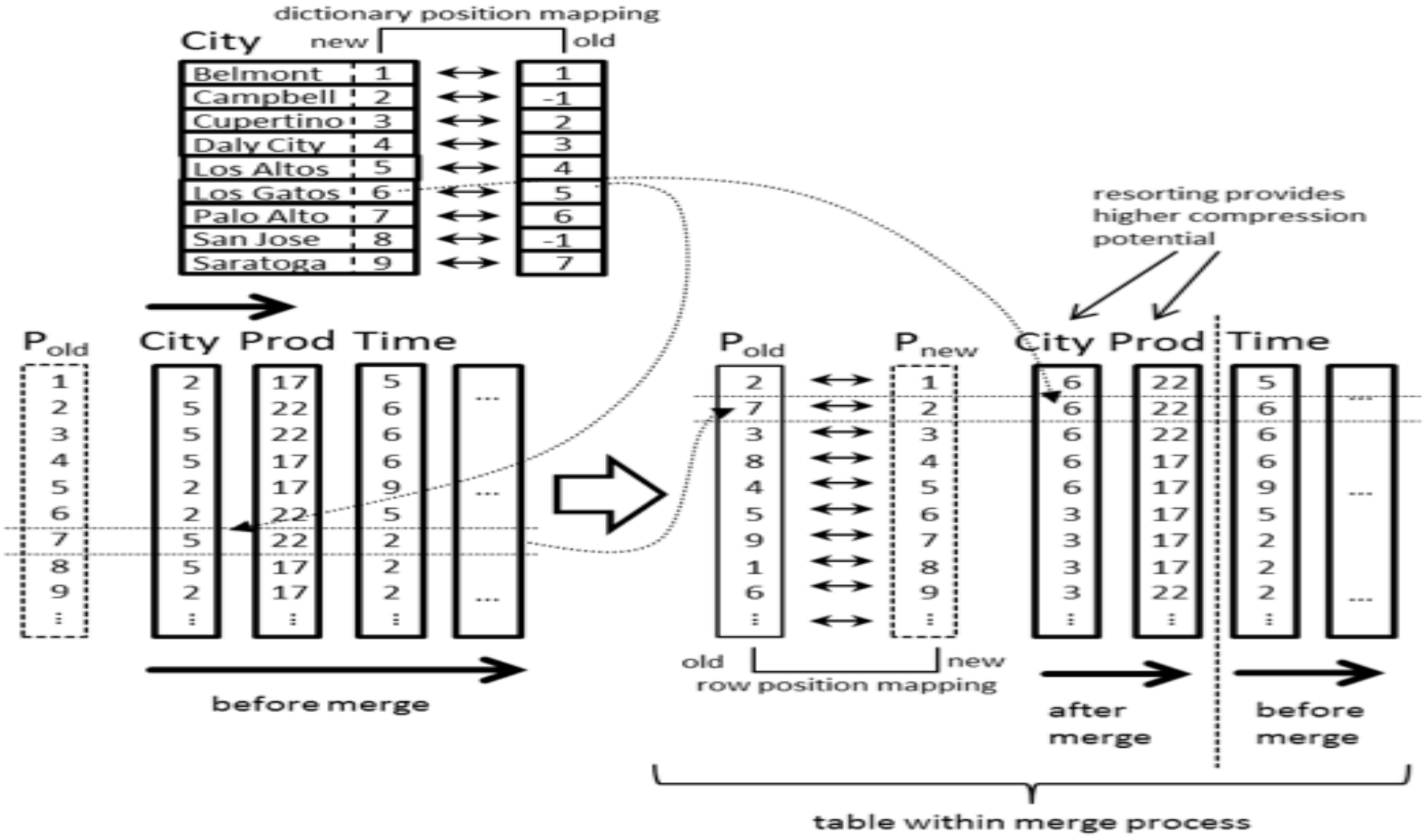
- L1-to-L2-delta Merge:
  - Rows of the L1-delta are split into their corresponding columnar values and column-by-column inserted into the L2-delta.Hence it is the main step.
  - The corresponding column values are added to the value vector using the dictionary encodings
  - Then, the propagated entries are removed from the L1-delta.

- L2-delta-to-main Merge:
  - The current L2-delta is closed for updates and a new empty L2-delta structure is created serving as the new target for the L1-to-L2-delta merge.
  - If a merge fails, the system still operates with the new L2-delta and retries the merge with the previous versions of L2-delta and existing main.

The Classic Merge:



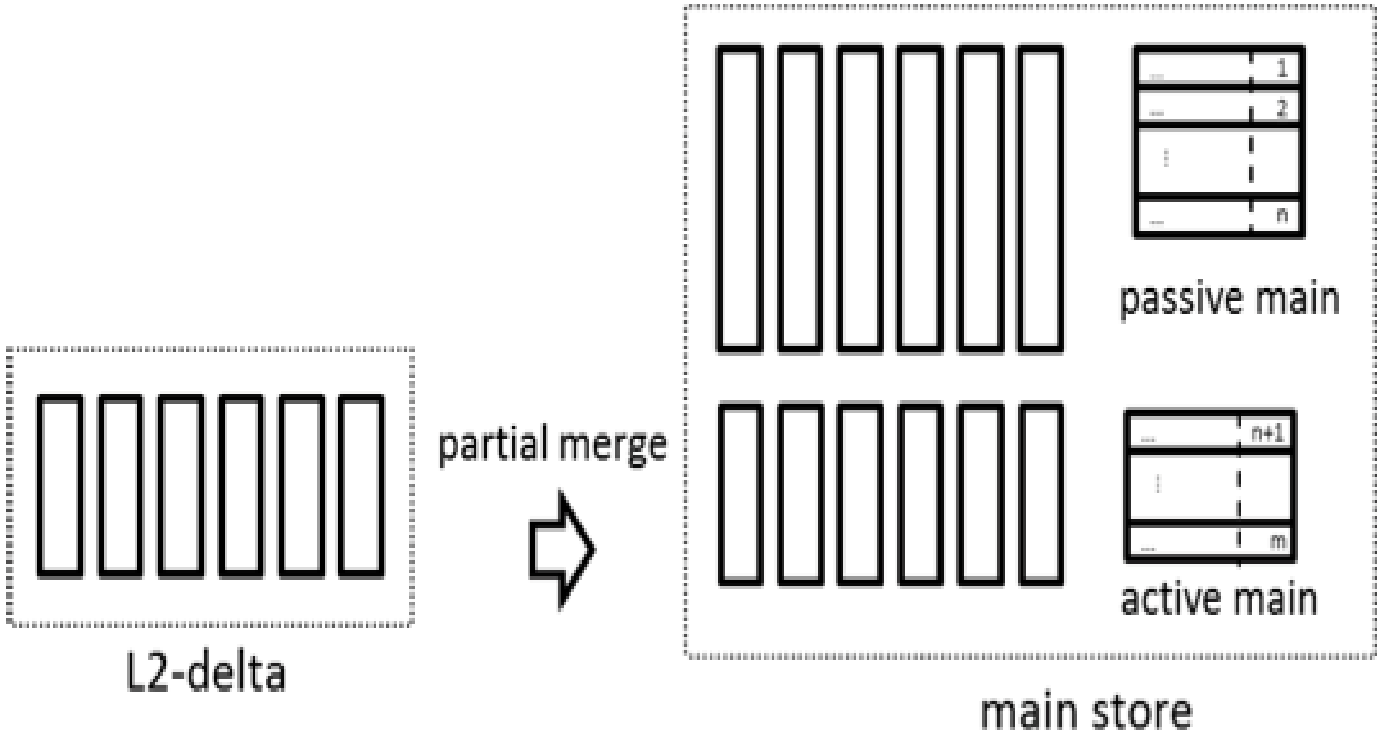
Re-sorting Merge:



Partial Merge:

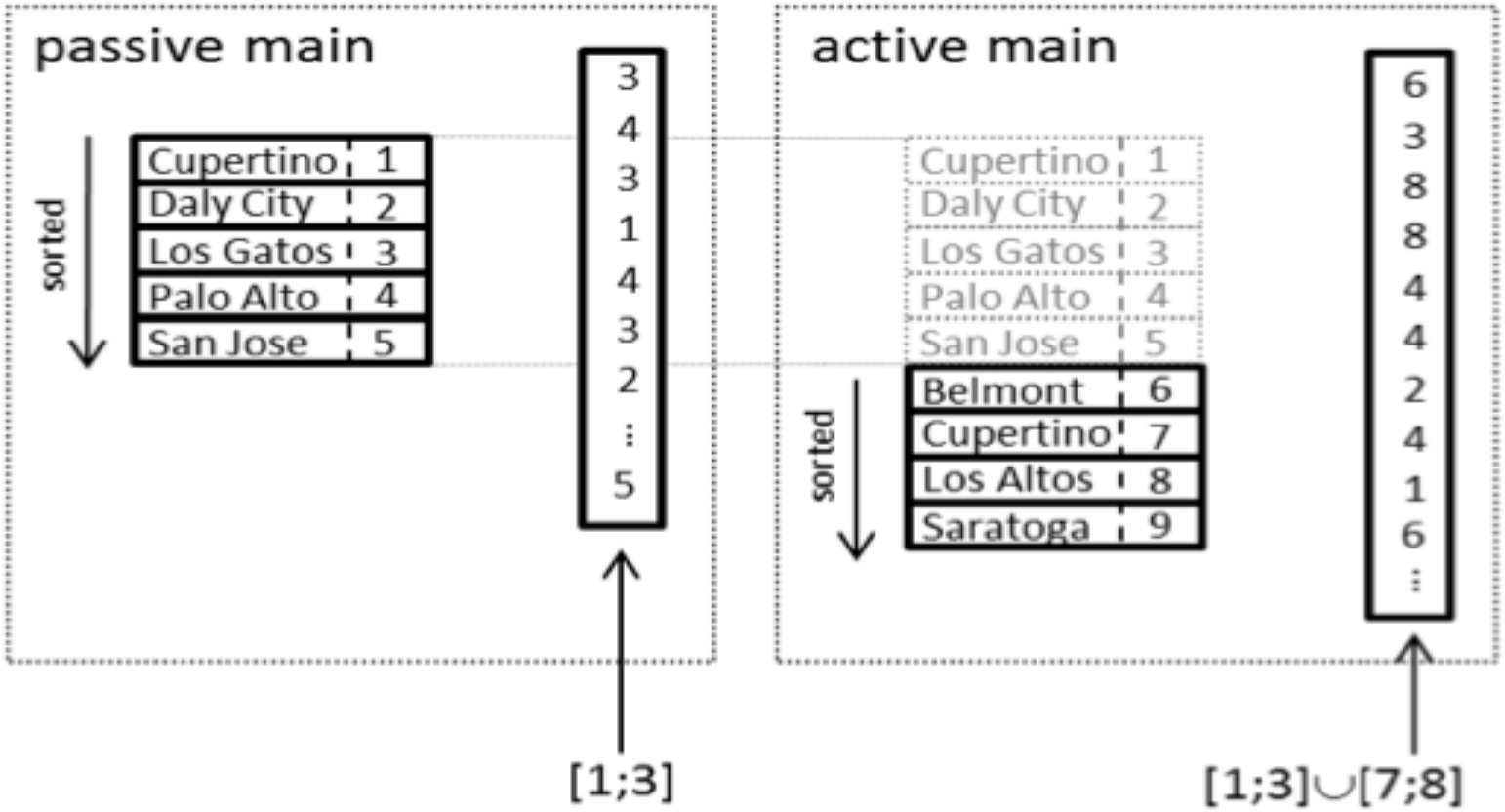
- Partial merge is to split the main into two (or even more) independent main structures.
- **Passive main**: The passive main reflects a stable part of the main store which is in general not part of the merge process.
- **Active main**: The active main is the part of the column which grows/shrinks dynamically and takes part of the merge process with the L2-delta.
- Merge interval within the partial merge strategy starts with an empty active main.
- Passive main reflects the regular main structure.
- Whenever a merge operation is scheduled, the L2-delta merges with the active main, the passive main remains untouched.

Overview of Partial Merge:



- During the search,if the requested value was found, the corresponding position is used as the encoding value for both, the passive and the active main value index
- If the requested value was not found, the dictionary of the active main is consulted.

Range Query Execution for Active and Passive Main



Scan: [between „C%“ and „L%“]

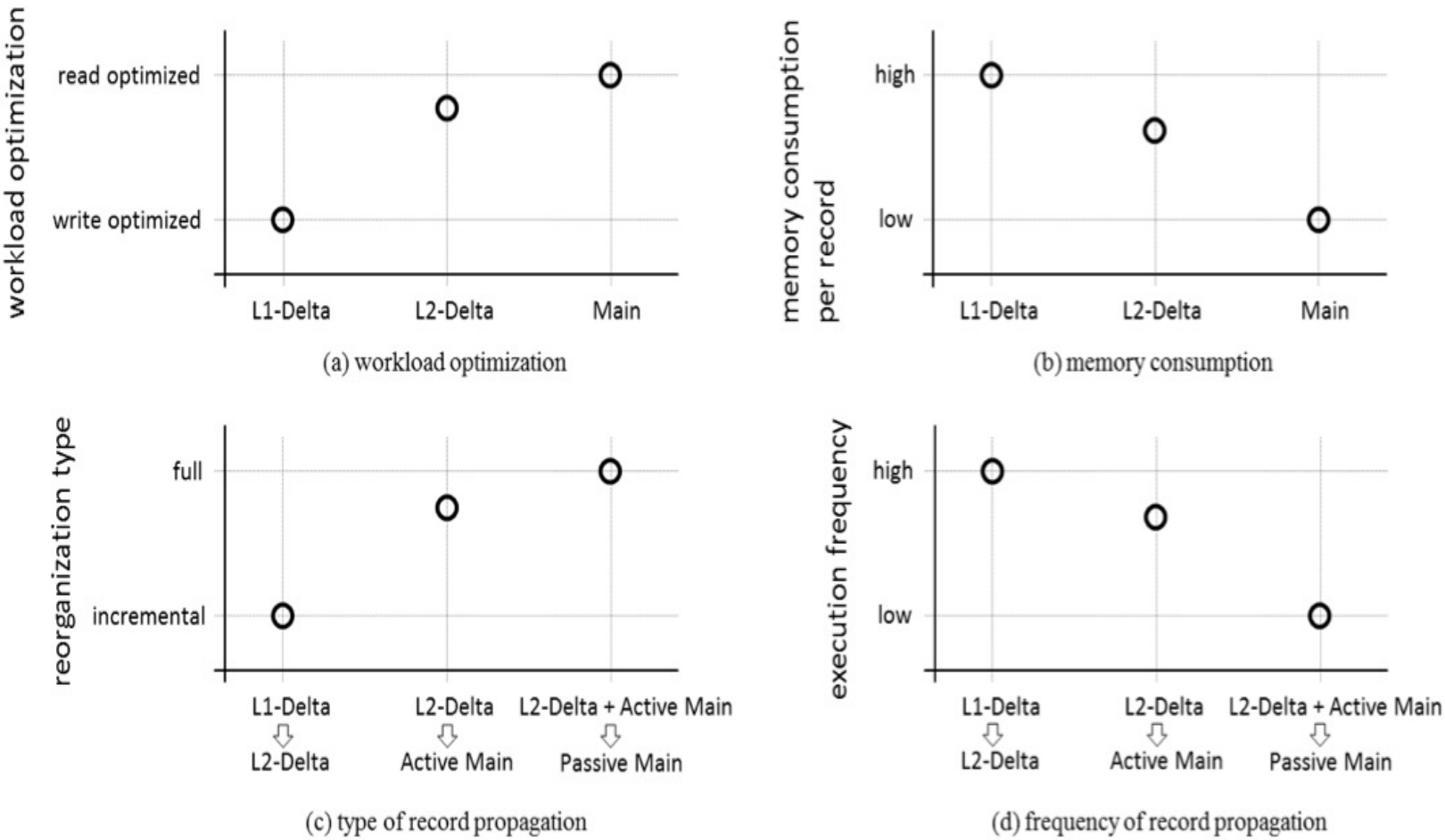


Figure 11: Characteristics of the SAP HANA database record life cycle

Conclusion:

- Column store systems are well known to provide superb performance for OLAP-style workload.
- The overall goal is to demonstrate some of the optimizations which are implemented within the SAP HANA database to make a column store suitable for high-scale transactional processing and ending the myth to use columnar technique only for OLAP-style workloads.