Efficient Transactions Processing in SAP HANA Database - The End of a Column Store Myth

Vishal Sikka, Franz Farber, Wolfgang Lehner, Sang Kyun Cha, Thomas Peh, Christof Bornhovd

presented by Cong Guo
February 10, 2015

Outline

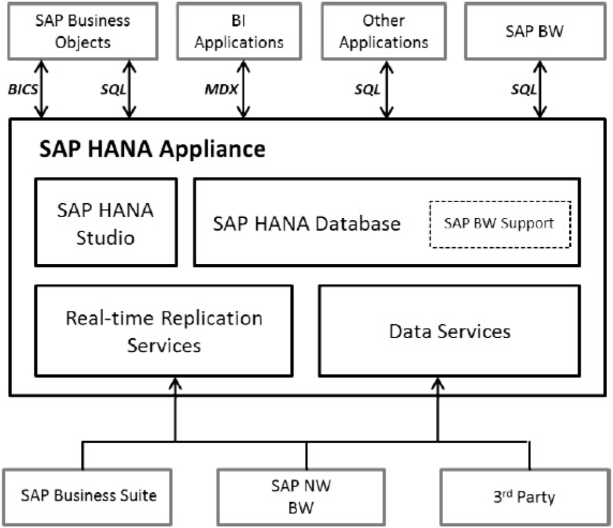
- Motivation
- Architecture of SAP HANA
- Lifecycle Management of Database Records
- Merge Optimization
- Conclusion
- Discussion

Motivation

- Usage perspective -various types of workloads and usage patterns
  - OLTP - high concurrency, frequent updates, and selective point queries
  - OLAP - long transactions, infrequent updates, aggregation queries, and historical data
- Zoo of specialized systems
  - Complex and error-prone
  - High total cost of ownership (TCO)
  - Used for performance

SAP HANA Appliance At a Glance

- Replace the zoo of specialized systems with a flexible platform



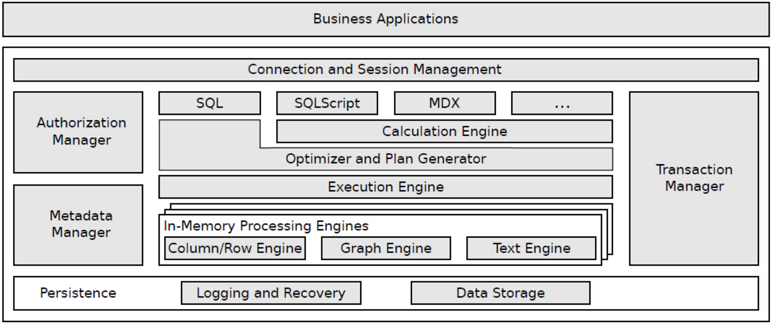
Features of SAP HANA database

- Has a girl's name (Hanna)
- Comprises multiple engines from relational data to graphs to unstructured text data
- Supports application-specific business objects and logic directly
- Communicates with the application layer efficiently
- Supports efficient processing for both OLTP and OLAP workloads

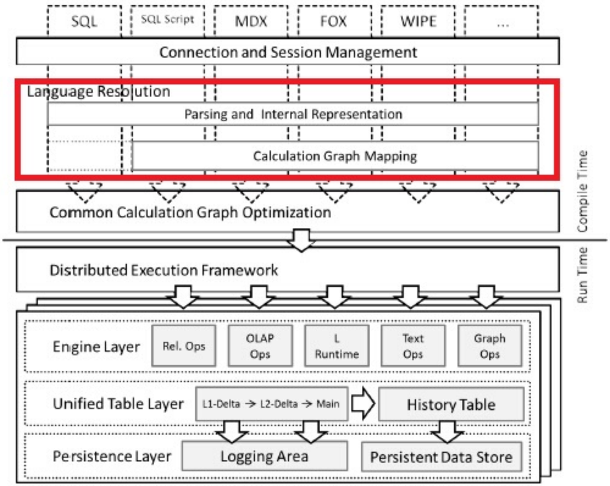
Outline

- Motivation
- Architecture of SAP HANA
- Lifecycle Management of Database Records
- Merge Optimization
- Conclusion
- Discussion

Architecture of SAP HANA



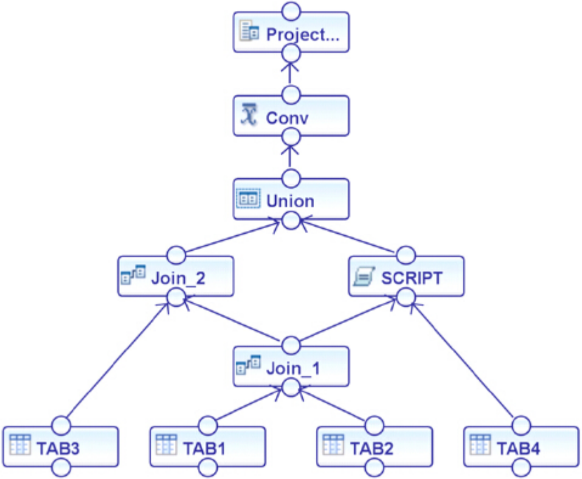
Architecture of SAP HANA



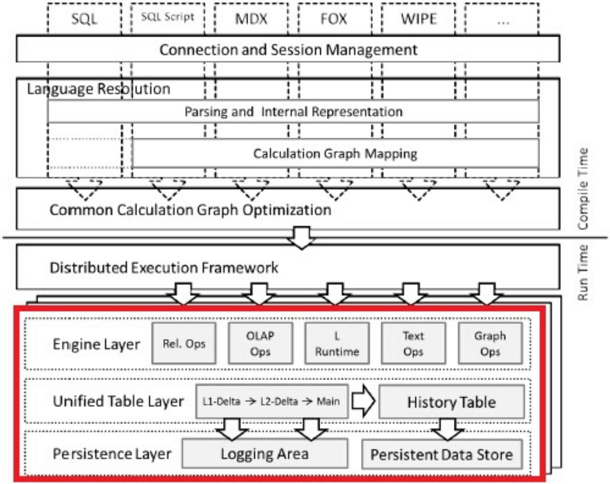
Calculation Graph Model

- An internal representation of query is mapped to a Calculation Graph
- Source nodes - table structures or outcome of other calc graphs
- Inner nodes - logical operators
- Operators
  - Intrinsic operators like projection, joins, union etc
  - Business algorithms like currency conversion
  - Dynamic SQL nodes, custom nodes, R nodes, and L nodes
  - Split and combine

Calculation Graph Model - Example



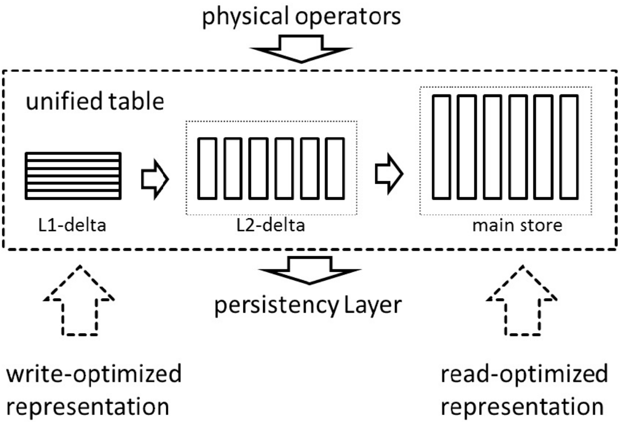
Architecture of SAP HANA



Outline

- Motivation
- Architecture of SAP HANA
- Lifecycle Management of Database Records
- Merge Optimization
- Conclusion
- Discussion

Lifecycle Management of Database Records



L1-delta Storage

- Accepts all incoming data requests
- Stores records in row format (write-optimized)
- No data compression
- Holds 10,000 to 100,000 rows per single-node

L2-delta Storage

- Accepts bulk inserts
- Stores records in column format (an index vector)
- Uses dictionary encoding for better memory usage
  - Unsorted dictionary
  - CSB-Tree based secondary index for point access
- Inverted index mapping value IDs to positions

Main Store

- Stores records in column format
- Employs a sorted dictionary
- Highest compression rate
  - Positions in the dictionary are stored in a bit-packed manner
  - Dictionary is also compressed using RLE and other techniques

Unified Table Access

- A common abstract interface to access different stores
- Records are propagated asynchronously
- Two transformations between stores called merge steps

Merge from L1-delta to L2-delta

- Row format to column format conversion
- Merge Steps
  - Appending new entries to the dictionary (in parallel)
  - Storing column values using the dictionary encodings (in parallel)
  - Removing propagated entries from the L1-delta

Merge from L1-delta to L2-delta

- A straightforward task
- The first two steps can be performed in parallel
- L2-delta data structures are not reconstructed
- Incremental merge
- Minimally invasive to running transactions

Merge from L2-delta to Main

- A resource intensive task
- The old L2-delta is closed for updates
- A new empty L2-delta is created
- A new main structure is created
- The merge is retried on failures

Persistency Mapping

- No fine-grained UNDO mechanisms
- Using REDO logs for new data in L1- and L2-delta and the event of merge
- Propagating pages that contain data structures in L2-delta to persistent storage at next savepoint
- Storing a new version of the main store on the persistent storage

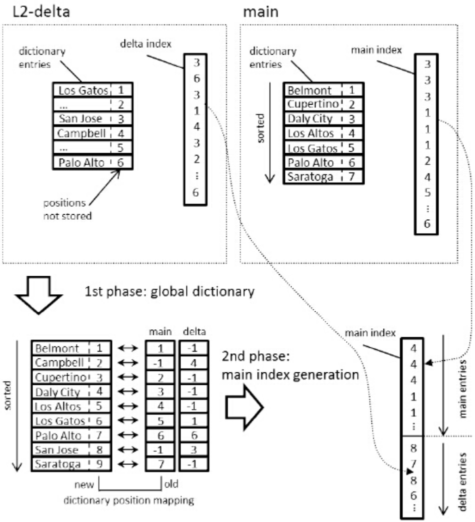
Outline

- Motivation
- Architecture of SAP HANA
- Lifecycle Management of Database Records
- Merge Optimization
- Conclusion
- Discussion

Merge Optimization

- The classic merge needs optimization
  - L2-delta to main merge is resource intensive
  - Main store needs high compression rate
- Re-sorting merge: higher compression rate
- Partial merge: reduce overhead of merge

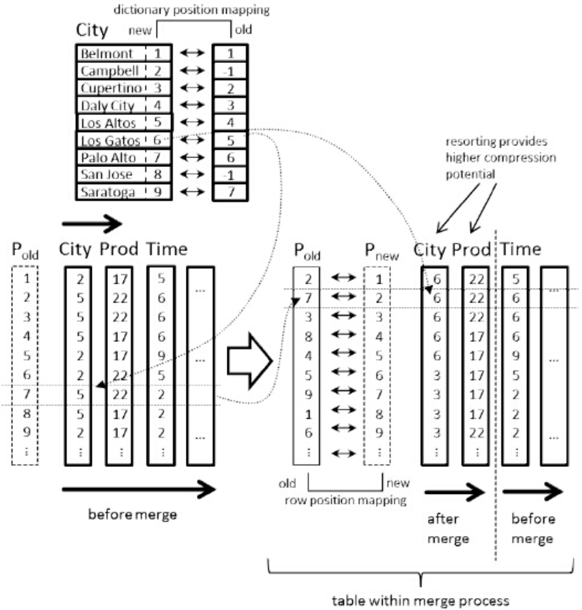
Classic Merge



Re-Sorting Merge

- Individual columns are re-sorted to gain higher compression rate
- A mapping table of row positions is added to reconstruct the row
- Sort order of columns are based on statistics from main and L2-delta

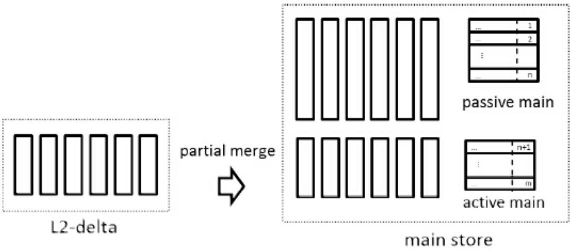
Re-Sorting Merge



Partial Merge

- Reduce merge overhead due to a large table size
- Split the main into two independent structures
  - Passive main
    - not part of the merge process
  - Active main
    - takes part in the merge process with the L2-delta
    - only holds new values not in the passive main
- Accesses are resolved in both dictionaries and parallel scans are performed on both structures

Partial Merge

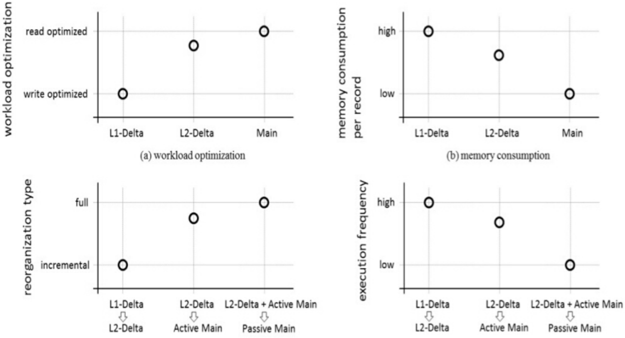


Conclusion

The HANA database is

- the core of SAP application ecosystem
- a main-memory database that efficiently supports both OLTP and OLAP
- consisting of different states of data structures but providing a common interface
- optimized for memory requirements and query processing

Summary



(a) workload optimization

(b) memory consumption

(c) type of record propagation

(d) frequency of record propagation

Discussion

- How does HANA determine when to merge the storages?
  - Currently based on data size
  - L2-delta is used to soften the problem
- Differences between main-memory and disk based DBMSs
  - Cache performance matters
  - The complexity of buffer pool management is reduced
  - Persistency is more challenging
- Differences between column stores and row stores
  - Compression