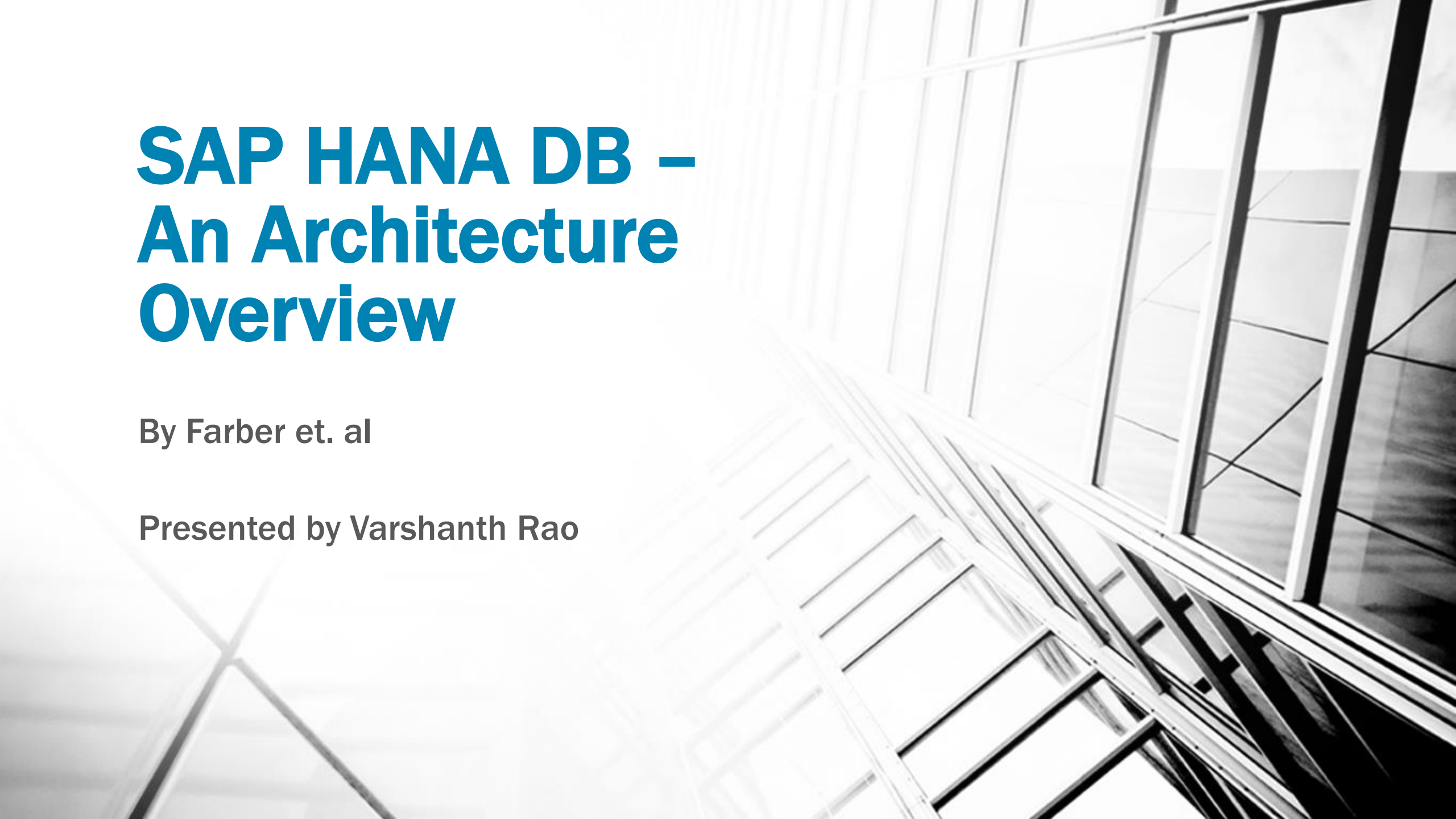
SAP HANA DB – An Architecture Overview

By Farber et. al

Presented by Varshanth Rao



Agenda

- OLTP vs OLAP
- ERP vs BI
- Why do we need Main Memory Systems?
- Goals of SAP HANA DB
- SAP HANA DB Architecture
- Query Processing
  - Analytical QP
  - Transactional QP

	OLTP System Online Transaction Processing (Operational System)	OLAP System Online Analytical Processing (Data Warehouse)
Source of data	Operational data; OLTPs are the original source of the data.	Consolidation data; OLAP data comes from the various OLTP Databases
Purpose of data	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
What the data	Reveals a snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and Updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries Returning relatively few records	Often complex queries involving aggregations
Processing Speed	Typically very fast	Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes
Space Requirements	Can be relatively small if historical data is archived	Larger due to the existence of aggregation structures and history data; requires more indexes than OLTP
Database Design	Highly normalized with many tables	Typically de-normalized with fewer tables; use of star and/or snowflake schemas
Backup and Recovery	Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method

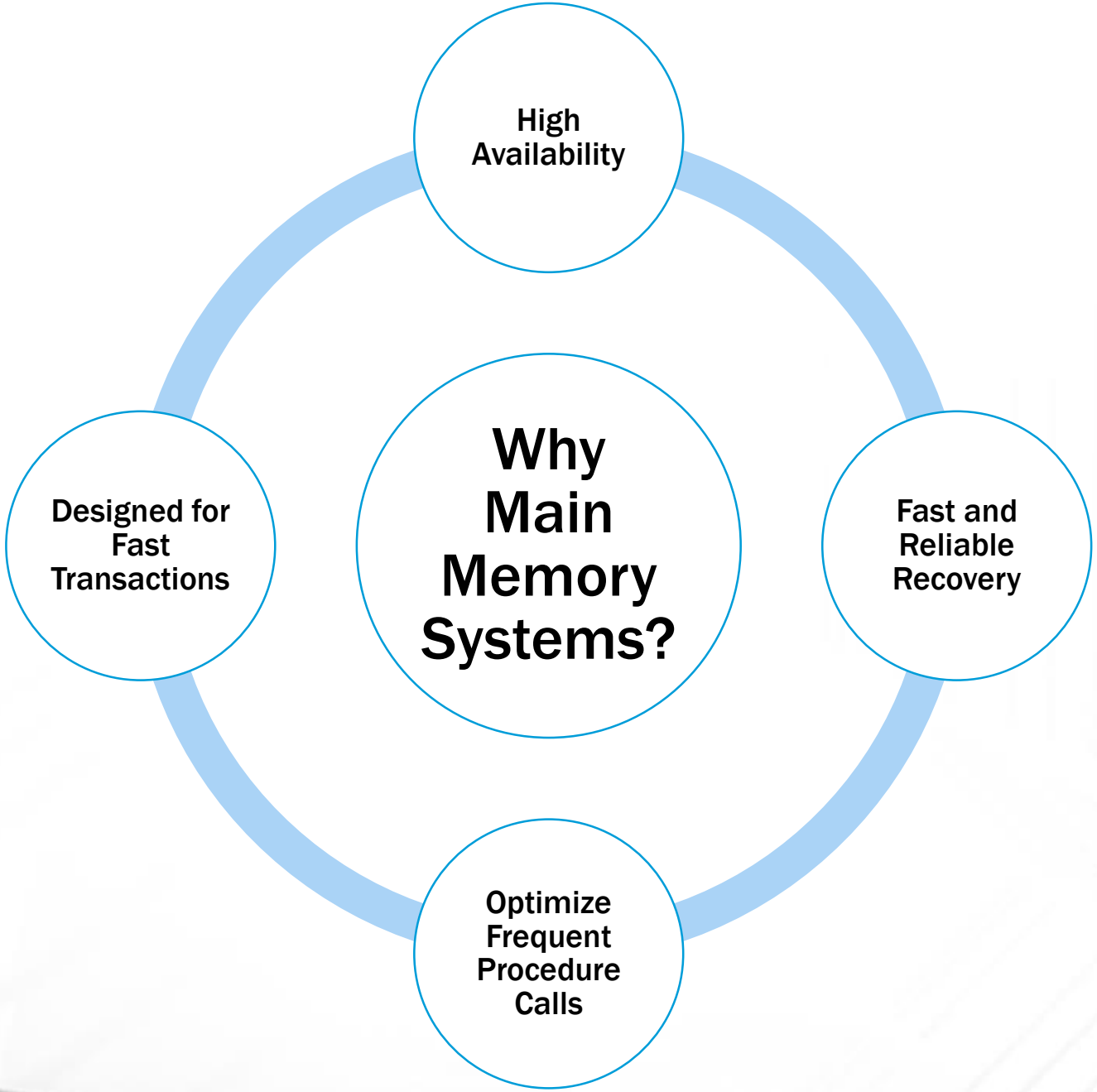
source: www.rainmakerworks.com

ERP vs BI: What is ERP

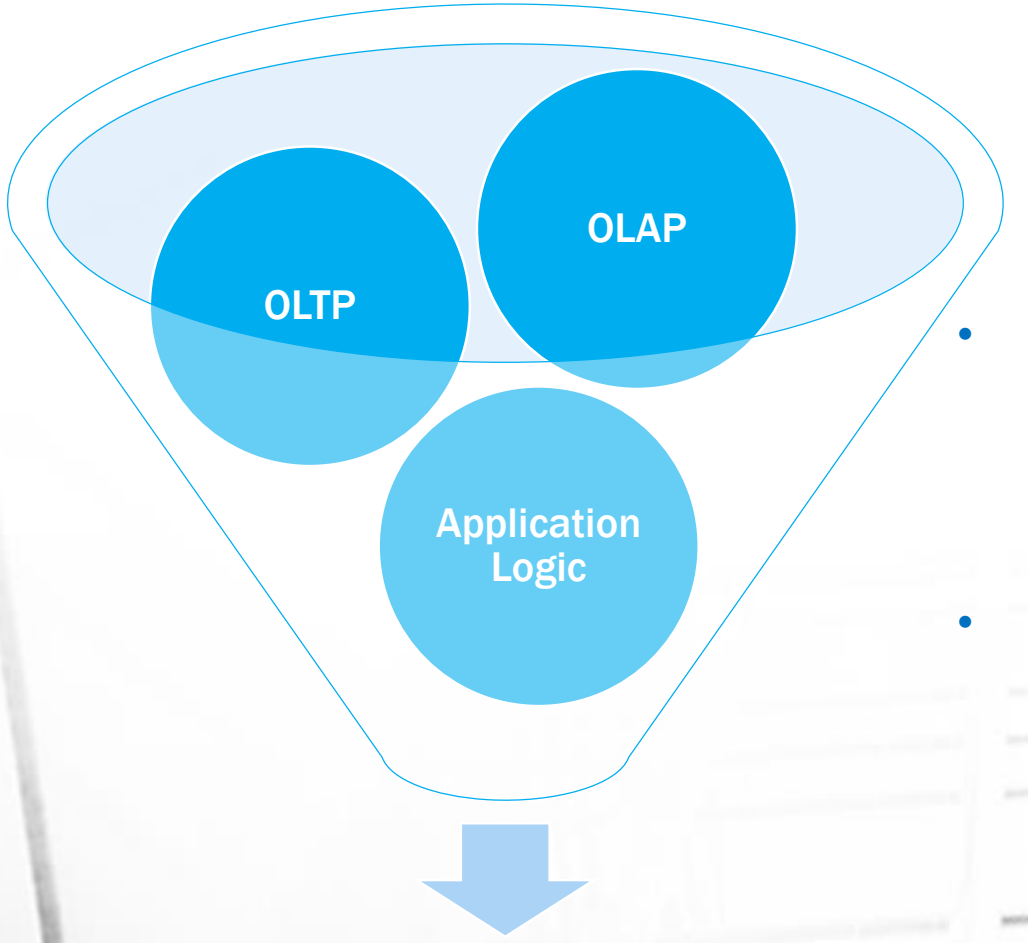
Gartner states, “ERP tools share a common process and data model, covering broad and deep operational end-to-end processes such as those found in finance, HR, distribution, manufacturing, service and the supply chain”

A BI tool accesses all of the data in your data warehouse, both strategic (revenue, profit and growth), and operational (daily sales performance). BI tools enable you to conduct in-depth analyses to generate comprehensive information that can deliver high-level insights.

ERP, on the other hand, is an operational system full of operational and transactional data. It will give you an exact view of your business from an operational perspective, but it is not built to perform trend analyses or give you high-level overviews. It is a tool centered around delivering operational insights.


High Availability
Designed for Fast Transactions
Why Main Memory Systems?
Fast and Reliable Recovery
Optimize Frequent Procedure Calls

SAP HANA DB Goals



- In Memory Columnar Store -> OLAP <- SAP TREX Text Engine + SAP BI Accelerator
- In Memory Row Store -> OLTP <- P*Time RDBMS

Master of All Trades

Jack of None

SAP HANA DB Goals

1. Main Memory Centric
2. Support for Pure SQL for Traditional Applications
3. Support for expressive interaction model for SAP applications
4. Achieve parallelization within and across distributed setups

SAP HANA DB Architecture

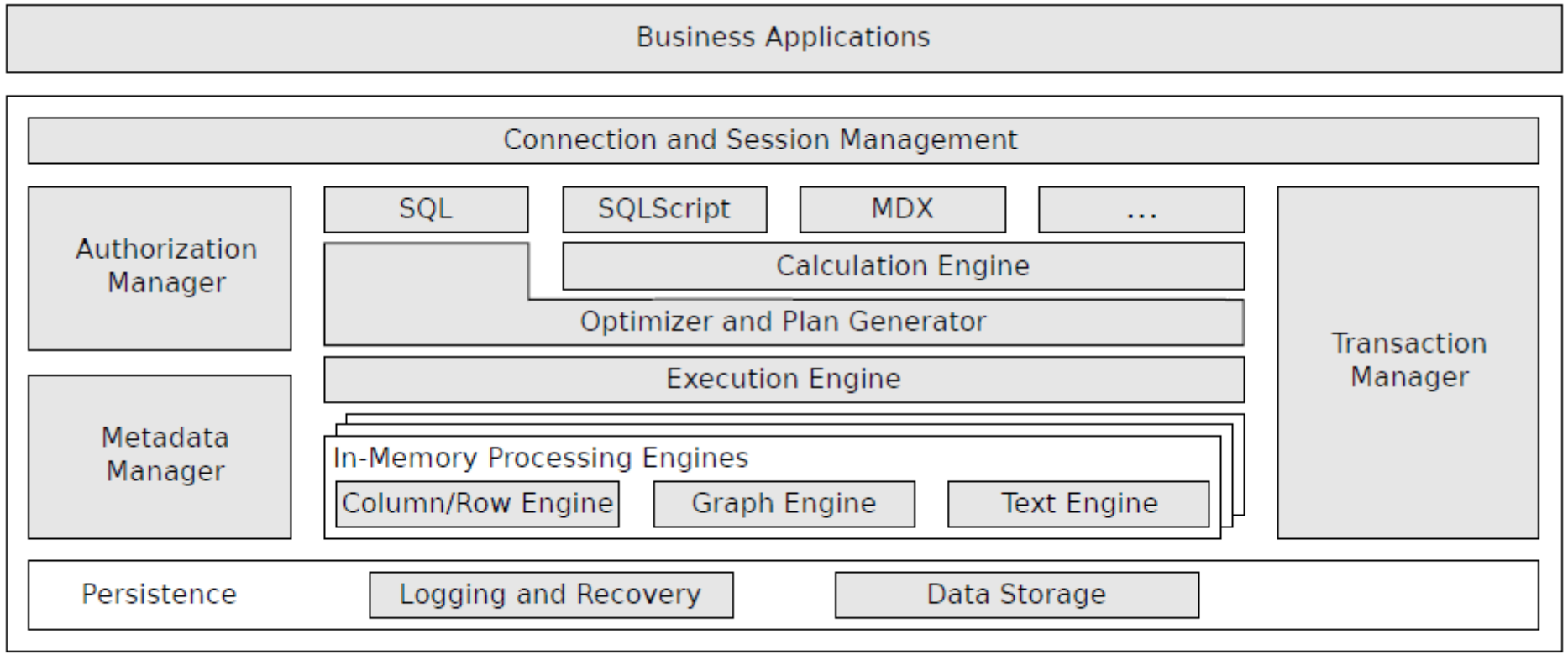
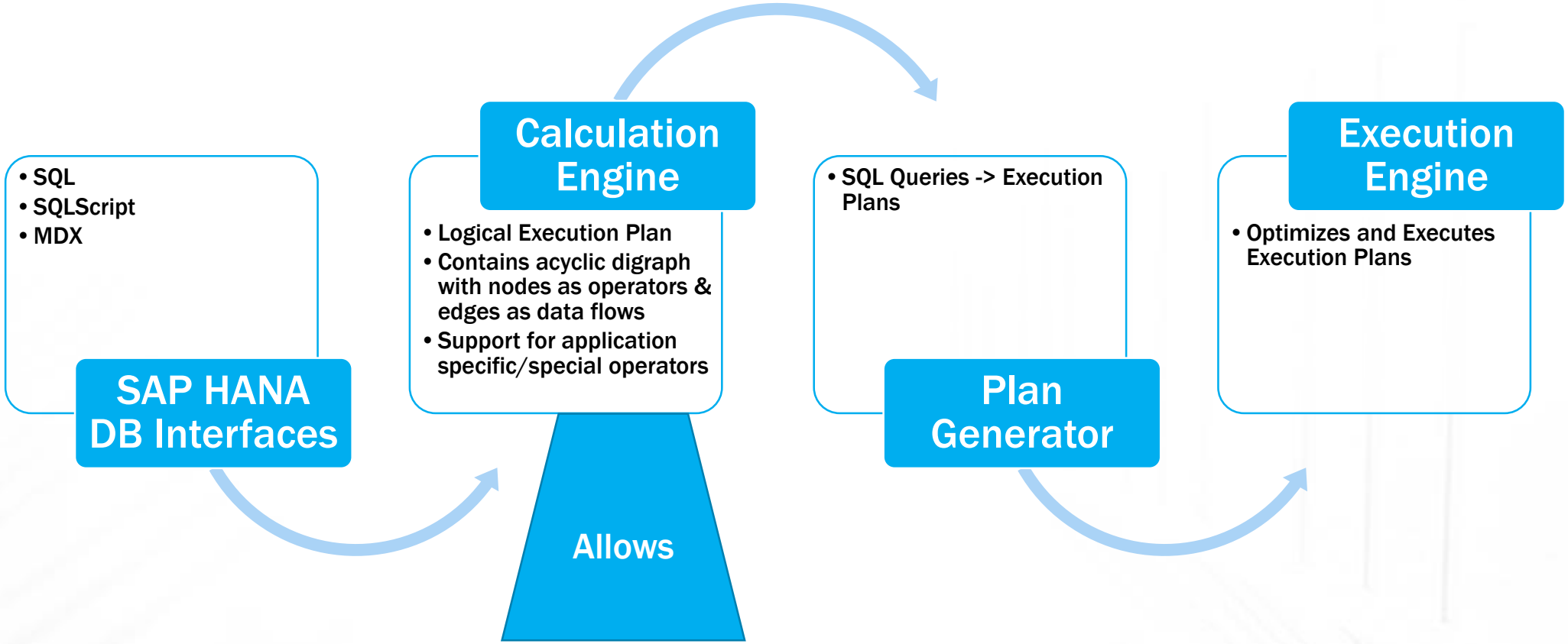


Figure 1: Overview of the SAP HANA DB architecture

SAP HANA DB Architecture

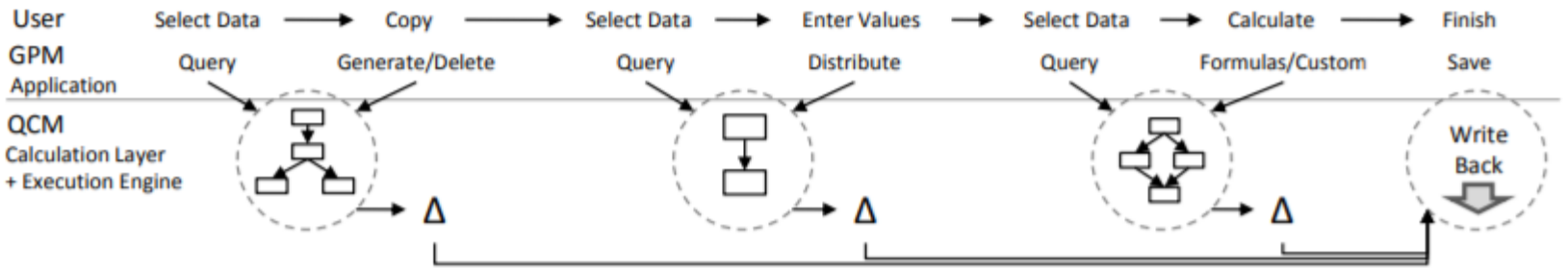
- All engines keep data in main memory as long as there is enough space available
- Data Structures are optimized for cache efficiency not disk efficiency
- Various compression schemes used by engines
- When available main memory is reached, all data objects are unloaded from main memory and reloaded into main memory when required

SAP HANA DB Architecture



- Interweaving of statistical algorithms like time series analysis, clustering, classification etc. with database operations
- Execute complex planning scenarios using disaggregation and custom formulas
- Construction of application specific operators like currency conversion

SAP HANA DB Architecture



Example Planning Process with General Planning Model and Query Calculation Model

Source: A Plan for OLAP by Jaeksch et al.

Query Processing: Analytical QP

- Column Stores are well suited for analytical queries on massive amounts of data : Allows columns to be sorted so that efficient compression schemes like Run Length Encoding can be applied
- Compression performed by using a representing values as integers (valueID) which can be compressed (dictionary compression)
- Compression allows more data to be stored on single node and faster query processing

Attribute Table

DocId	ValueId
1	24
2	3
3	7
4	17
5	3
6	…

Dictionary

ValueId	Value
1	IBM
2	Microsoft
…	
17	SAP
…	

Index

ValueId	DocIdList
1	…
2	…
3	2, 5
4	…
…	
17	4

Source: Data Mining with the SAP NetWeaver BI Accelerator by Legler et. al

Query Processing: Analytical QP

- Delta storage aids in reducing expense of single updates (delta storage loads rows incrementally according to update access and uses Row Level Versioning (RLV) to allow concurrent transactions without locks
- Dictionary compression used in delta storage as well but dictionary is stored in Cache Sensitive B+- Tree: variant of B+-Trees that stores all the child nodes of any given node contiguously, and keeps only the address of the first child in each node. The rest of the children can be found by adding an offset to that address. Since only one child pointer is stored explicitly, the utilization of a cache line is high
- Delta storage is merged periodically into main data storage (dictionary merging between delta store (CSB+-Tree)) and old main data storage
- Write operations are redirected to new delta storage when delta merge starts and read ops access new, old delta and old main storage
- Intra operator parallelism (example grouping ops) can be performed across cores and nodes to speed up execution
- Large tables can be partitioned into parts/complete tables and can be assigned to different nodes.
- Execution Engine schedules ops in parallel & executes them on the node containing the data (if possible)

Query Processing: Analytical QP – InfoCube Basics

- Infocube is data storage area in which we maintain data which we are extracting from source system physically. An InfoCube can function as both a data target and an InfoProvider

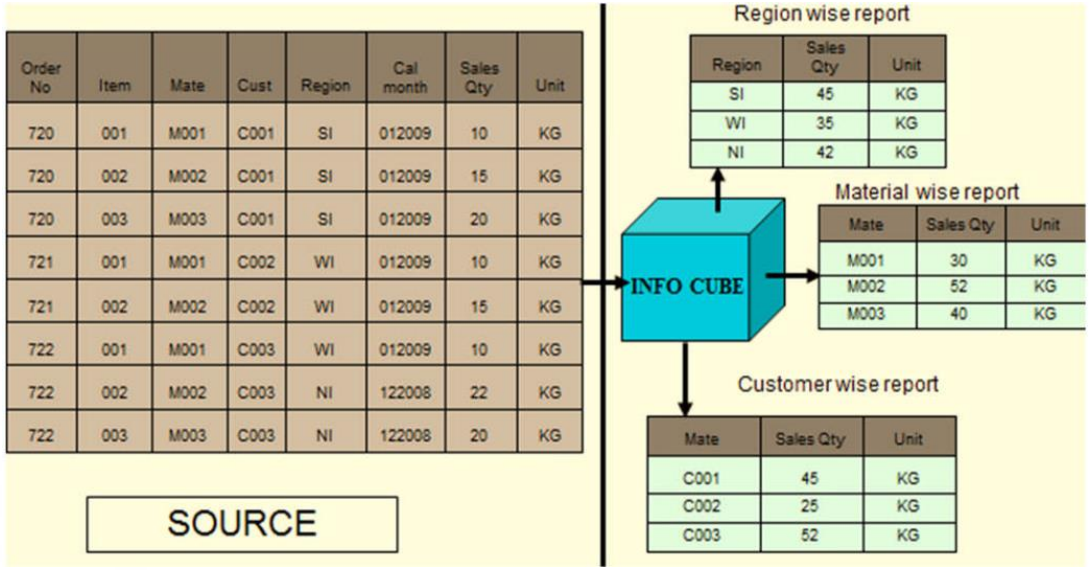
- An Infocube follows the Extended Star Schema.

- It is a self-enclosed data set encompassing one or more related business processes. It is used to store summarized / aggregated data for long periods of time. Infocubes consist of precisely one fact table surrounded by dimensional tables.

- A fact table consists of facts of a particular business process e.g., sales revenue by month by product. Facts are also known as measurements or metrics. A fact table record captures a measurement or a metric.

- The accelerator index structures were designed to support such read-only or infrequent-write scenarios
- Designed to handle updates efficiently in the following two respects:
  - Updates to an InfoCube are visible with minimal delay in the accelerator for query processing (Delta Index Mechanism)
  - Accumulated updates do not degrade response times significantly } Background jobs (Delta Merge Mechanism)

Query Processing: Analytical QP – InfoCube Basics



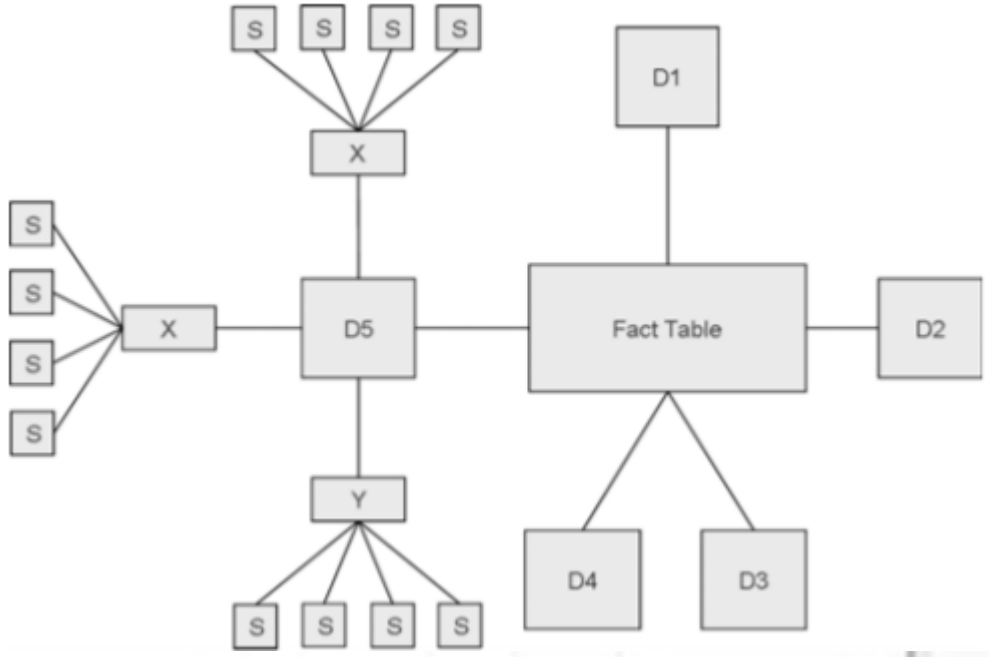
Order No	Item	Mate	Cust	Region	Cal month	Sales Qty	Unit
720	001	M001	C001	SI	012009	10	KG
720	002	M002	C001	SI	012009	15	KG
720	003	M003	C001	SI	012009	20	KG
721	001	M001	C002	WI	012009	10	KG
721	002	M002	C002	WI	012009	15	KG
722	001	M001	C003	WI	012009	10	KG
722	002	M002	C003	NI	122008	22	KG
722	003	M003	C003	NI	122008	20	KG

Region	Sales Qty	Unit
SI	45	KG
WI	35	KG
NI	42	KG

Mate	Sales Qty	Unit
M001	30	KG
M002	52	KG
M003	40	KG

Mate	Sales Qty	Unit
C001	45	KG
C002	25	KG
C003	52	KG

Source: https://www.guru99.com/what-is-an-infocube-how-to-create-one.html

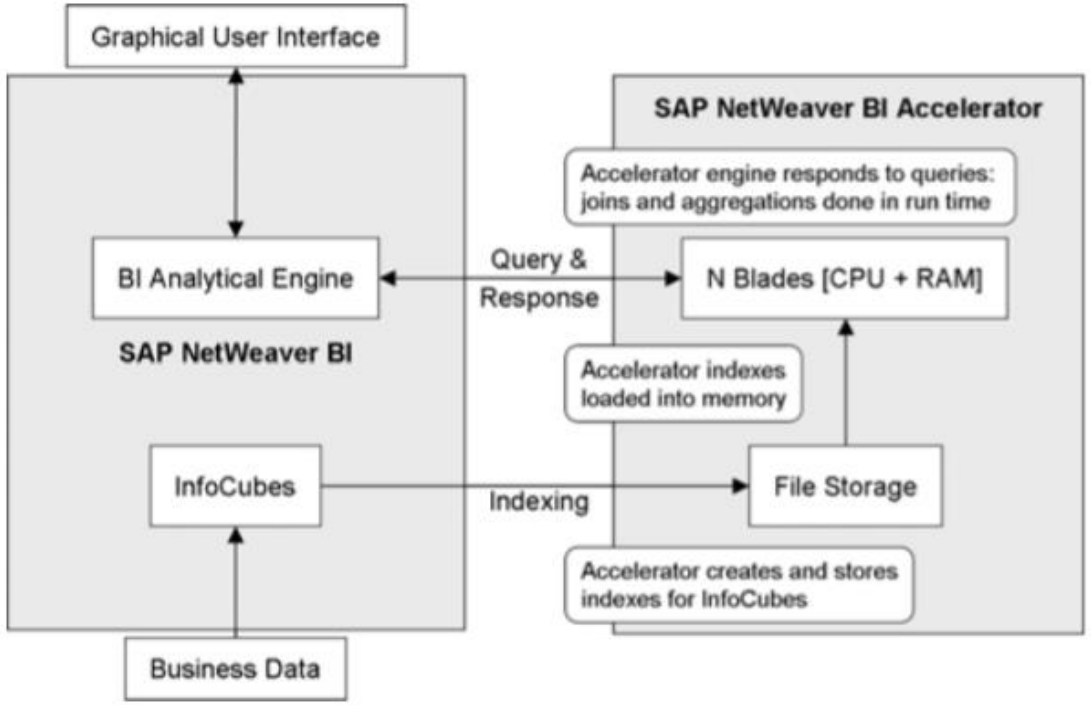


Source: Data Mining with SAP NetWeaver BI Accelerator by Legler et. al

Query Processing: Analytical QP – Processing In Memory

- All data needed to answer a query is copied to memory -> Execution Plan is constructed -> Join Paths created -> Performs required aggregations -> Merges Result -> Return to User

- Column indices are written into memory and cached as flat files -> Reduced memory and less I/O flows between CPUs



Source: Data Mining with SAP NetWeaver BI Accelerator by Legler et. al

Query Processing: Transactional QP

- Advantages of using column store:

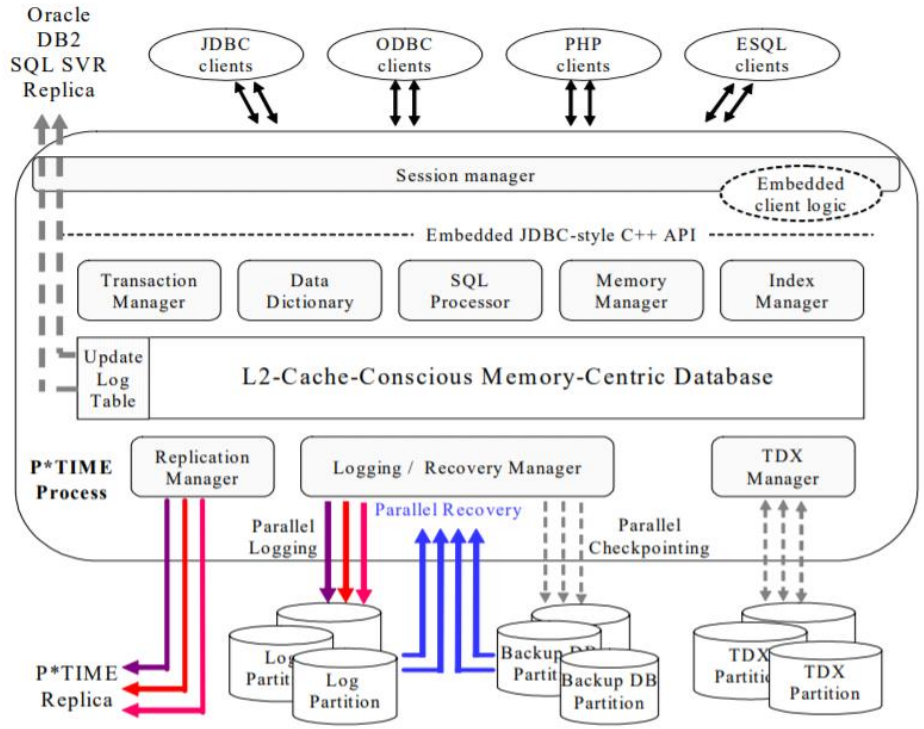
  1. Unused columns/May contain only default values/status flags = High compression opportunity
  2. Real world transactional workload has larger read ops than benchmarks
  3. Append scheme for updates much simpler than in-place updates
  4. Avoiding use of indices (use only for primary keys) as scan performance is good enough

- Challenges of using column store:

  1. Performance overhead to allocate memory per column
  2. Frequent updates = new entities in delta storage = frequent merges } CPU Intensive Operation

Query Processing: Transactional QP – P*Time RDBMS

- C++ with templating was chosen for P*Time DB Codebase
- L2 Cache Conscious In-Memory DB -> Tables are stored (column wise partitions) in 1 or more containers which can hold multiple pages for easy swap-in-out. B+- Tree used for indexing
- Fine grained parallel logging and recovery (using separate log partitions to store differential (XOR) logs for parallelization and fine grained fuzzy checkpointing [record or column level])
- Concurrency control through optimistic, latch-free index traversal (OLFIT) scheme (bottom up latching instead of top down latching) on B+- trees
- Access to SQL processor using embedded C++ APIs



Source: P*TIME: Highly Scalable OLTP DBMS for Managing Update-Intensive Stream Workload by Cha et. al

Hekaton vs SAP HANA DB

Criteria / In Memory DB	Hekaton	SAP HANA DB
Target	OLTP	OLTP + OLAP
Storage Layout	Column Store	Interchangable Layout (Row <-> Column)
Codebase	TSQL translated into machine code using C as intermediate	C++
Data Structure	Mixed Abstract Tree (Data + Metadata) -> Pure Imperative Tree (Simplified Data)	SAP BI Infocubes Converted to Internal Tree Representations
Concurrency Control Mechanism	Latch Free Bw Trees Optimistic Multi Version Concurrency Control	L2 Cache Sensitive B+- Trees OLFIT
Checkpointing	Delta File Maintenance & Merging	Fuzzy Checkpointing
Garbage Collection	Hekaton GC – Single Process Multi Threaded by Piggybacking on Hekaton Transaction Workers	Separate Process, Multi Threaded using Dedicated GC Workers. Hybrid Version Proposed

Conclusion/Review

- ERP vs BI:
  - ERP popularly uses OLTP
  - BI popularly uses OLAP
- Main memory systems
- Review of OLTP vs OLAP
  - OLTP: Update & Write heavy, Row Store Preferred
  - OLAP: Read heavy, Column Store Preferred
- SAP HANA Goals: OLTP+OLAP In 1 Suite
- SAP HANA DB Architecture
- Analytical Query Processing
  - SAP BIA Data Structures & In-Memory Architecture
  - Infocube Basics
- Transactional Query Processing
  - Advantages & Challenges of using Column Store
  - P*TIME RDBS for OLTP
- Hekaton vs SAP HANA DB

Critique/Discussion

- Write operations are redirected to new delta storage when delta merge starts and read ops access new, old delta and old main storage [author refers to Fast Updates on Read-Optimized Databases Using Multi-Core CPUs by Krueger et. al, but does not explain consistent read ops are managed]

- Intra operator parallelism (example grouping ops) can be performed across cores and nodes to speed up execution [How does the result aggregation work when ops are performed across nodes? How are failures handled i.e. does the aggregator wait till it succeeds or is the task given to another core/node]

- SAP BIA uses CSB+- tree for Indexing into OLAP targeted DBs -> Cache optimized delta index dictionaries, P*TIME uses B+- and justifies using B+- trees over CSB+- tree claiming better performance

- Fuzzy checkpointing is reliable only in case of high bandwidth availability of disk since frequent flushing to disk is required

- In memory row store and column store conversions allowed -> How long does the conversion take? Are there mechanisms to smartly store/retrieve the tables in the frequently used layouts?

- Authors follow partitioning schemes & indexes within partitions. Previous work shows indexing within partitions is only useful when they workload is partitionable

Thank You!