Oracle Database In-Memory

A Survey of In-Memory Databases from SAP, Microsoft, IBM, MemSQL and Oracle

ORACLE WHITE PAPER | JUNE 2016

Introduction

In-Memory Database (IMDB) technology is one of the most active data management software categories in recent past. In December 2013 a major technology research firm predicted that “Memory Optimized (“In-Memory”) Database Technology is taking over Enterprise Databases”.[1] In support of this prediction, between June 2013 and July 2014, IBM, Microsoft and Oracle released memory-optimized database technology as part of their mainstream relational databases, joining SAP’s HANA in-memory database released in late 2010. Thus, all major database vendors and many startups had embraced IMDB technology as of mid-2014.

Databases have used in-memory data structures for many years to avoid the relatively slow performance of reading and writing directly to disk drives. As a result, many startup databases or caching products today claim to be part of the “in-memory” category and it would be near impossible to compare this broad spectrum of implementations. For the purposes of this paper, IMDB technologies are those that go further than just accessing data in memory. They apply formats and algorithms that take advantage of memory and modern processors and aren’t burdened by compatibility with disk-based approaches. The IMDB products from Oracle, IBM, Microsoft, MemSQL and SAP generally meet this definition, though to different degrees.

Performance is the rationale behind IMDB technology. More specifically, high-performance analytics is the most common objective, although faster OLTP might also be targeted. Analytics includes reporting, business intelligence and ad-hoc queries, often against a Data Warehouse and increasingly against transactional (OLTP) databases if possible. In the case of all five vendors, high-performance analytics is supported through data stores organized by columns rather than the traditional row format. The technologies and techniques employed are similar across all five vendors, such that the performance differences may not be substantially different. The major differences lay in other areas, such as scalability, application transparency and mixed workload support.

[1] IDC Worldwide Application Development & Deployment Predictions 2014, by Carl Olofson – Dec 5, 2013

SAP HANA

Summary: The *SAP HANA* In-Memory Database has been available since November, 2010. There have been 12 subsequent HANA releases at 2-3 per year through May 2016. HANA is marketed as a way to perform real-time OLTP and OLAP against a column-format data store. SAP is the only vendor that promotes the use of a column-format data store for all database workloads.

SAP refers to HANA as a “platform” for real-time applications, rather than an in-memory database. However, the functional components of HANA are similar to those within a database such as Oracle, supporting many types of data, multiple ways to access and search for information, and a variety of programming interfaces and tools. The HANA programming interface is built on SQL but, like every database, has many unique constructs that render existing database applications (that use Oracle, IBM and Microsoft databases) incompatible with HANA without modifications. On closer inspection, one can also challenge the degree to which HANA can compare with mature enterprise databases, particularly in areas such as scale out, availability, backup and disaster recovery. HANA has made some advancement in these areas, but it was only with the latest release, SPS12, that support for HANA database recovery using a combination of storage snapshot and delta backups (incremental and differential backups) was added. Extending the HA/DR capabilities to a certain point was overdue and has been a major barrier in its adoption.

SAP has nurtured a community of third-party software vendors developing applications on HANA and has modified many SAP applications to use HANA, such as SAP Business Warehouse (BW) and SAP Business Suite, and the newly released SAP S/4 HANA last year. Organizations can deploy HANA on-premise after purchasing an SAP-certified hardware appliance, or on a public cloud from SAP, Amazon, Microsoft, IBM, and several other cloud providers. According to SAP, there are thousands of SAP HANA customers.[2] SAP has energized the IMDB category by being the first major vendor to emphasize the technology. Although SAP is early in the development cycle of a full-fledged database system, and has yet to verify some of their claims, they have validated the ability to power real-time analytics with an IMDB and have attracted a respectable following considering the immaturity of the product. SAP has gone strategically “all in” on in-memory computing with HANA and positioning HANA as a platform.

Technology: SAP HANA has traditionally been a "pure" IMDB, which limited HANA to databases that could be fully loaded into memory. This meant that HANA could not support very large databases

[2] http://global.sap.com/corporate-en/investors/newsandreports/news.epx?articleID=24021&category=45

where the size exceeded the amount of memory available. This has been a source of a large number of complaints from HANA customers over the need for a smooth transition within HANA to non-volatile storage. SAP partially addressed this problem with the dynamic tiering option and support for "extended tables" released in late 2014. Extended tables are disk-based structures that allow HANA to access data on disk as well as in memory but currently there is no concept of partitioning, and extended tables are discrete objects from in-memory tables, which must be managed by applications. More recently, their Data Lifecycle Management (DLM) tool can be used for moving data between memory, disk and Hadoop/SybaseIQ based on specified rules and policies. This solved the large database problem, but introduced other issues. Besides the issue that extended tables will be very slow to access, use of extended tables is also not transparent to the application. The logic to manage data tiering and access for extended tables may need to be added to every application that accesses data in a large database.

HANA does have support for row based tables, but the performance emphasis is on columnar based tables and this is the format that all of HANA's performance features operate on. This is a significant issue in HANA, as a table is either row based or it is column based; it cannot be both simultaneously. So although row based tables are supported, HANA is primarily a columnar database and SAP emphasizes that most tables should be columnar based. This implies that a single data format is suitable for both OLTP and analytics.

SAP has yet to validate that a column-format data store can process high-volume OLTP. It is intuitive that a single row insert into a row-format table is more efficient than inserting values into potentially hundreds of columns in a column-format table. Not surprisingly then, although SAP claims over 6,400 HANA customers[3] (as of June 2015), no references detailing OLTP performance results can be found.

"Hana stores after a merge (reorganization) all data in a main store organized by columns. After that all new entries are stored in a delta store which is also organized by columns."

Hasso Plattner comment to: HANA, Columns, and OLTP Aug 22, 2013
http://www.saphana.com/community/blogs/blog/2013/08/21/hana-columns-and-OLTP

While the concept may sound appealing, computer science fundamentals do not support SAP's claims for HANA performing high-volume OLTP from a columnar data store. Even SAP researchers have

[3] http://www.businessinsider.com/sap-shares-hana-customer-numbers-2015-6

admitted to the challenge and offered scant solutions.[4] The two big areas of concern are high volume DML operations (insert, update and deletes) and HANA's ability to merge these changes into the main column store without delays, and the ability to process high volumes of "row" based queries where all or most of the columns of a row are requested. Both of these workloads are typical of high volume OLTP databases. While HANA may be suitable for SAP-designed applications, it's not clear whether HANA will provide a good general purpose database solution, and it is precisely the mixed workload environment for which SAP has said HANA is well suited.[5]

SAP also makes claims about HANA's ability to scale out, but HANA uses a shared-nothing architecture that is unproven for large clusters and has never been demonstrated for OLTP workloads. It requires deep knowledge about data and workload to achieve scaling, such that application partitioning and data co-location can be achieved. Hence, scale out to multiple nodes is possible only for certain applications like SAP-BW. Others such as SAP ERP and S/4 HANA are restricted to one node With the latest release, SAP has increased the memory limits for HANA to be 12TB (or 4TB on a virtual instance), beyond which data needs to be managed using dynamic tiering on disk.

Perhaps another sign of expediency is revealed when one looks under the covers of HANA. Far from being a grounds-up "pure" development, HANA is the combination of several database and search products that SAP acquired or developed; MaxDB, P*Time, SybaseIQ and TREX.[6] SAP has not disclosed the manner in which these separate databases have been integrated into HANA, though it stands to reason that considerable complexity exists.

Strategy: Although SAP has traditionally been known as an applications company, its corporate strategy has shifted significantly toward building the HANA platform and integrating current and new SAP and third-party applications on top of HANA, preferably in the cloud.[7] Yet the reshaping of an enterprise as large as SAP is by no means certain, particularly given the technical complexities of an enterprise database platform.

Bending to the reality that a "pure in-memory" database store is generally impractical for all but the smallest environments, SAP has attempted to address the issue in recent HANA releases. First a near-line storage solution based upon the old Sybase IQ platform for archived data was introduced

[4] http://sites.computer.org/debull/A12mar/hana.pdf
[5] http://blogs.saphana.com/2013/01/03/oltp-bi-in-a-single-hana-instance/
[6] http://en.wikipedia.org/wiki/SAP_HANA
[7] http://www.sap.com/corporate-en/factsheet

and more recently policy driven dynamic tiering option that enables the administrator to create disk based tables to contain older, less frequently accessed data. For SAP's existing customers, this "bet the farm" strategic shift may be viewed as adding risk and diverting resources from applications, and imposing frequent product updates for early adopters.

Validation: SAP has successfully evangelized HANA and in-memory computing and described new categories of analytics applications that take advantage of HANA. Many hardware vendors have introduced HANA appliances in support of SAP's agenda, and a community of ISVs is slowly forming. SAP HANA implementation is often highly complex, time consuming and a disruptive task. This is due to the fact that such a large scale project often involves fundamental changes to the organizations IT architecture. Testing is required, processes have to be altered, and users will need additional training to come to grips with the new technology. SAP's main challenge is to complete the development of HANA and catch up to decades of database development and innovation. The architectural choices that underlie HANA may also need rethinking in order to fulfill SAP's vision of a single platform for OLTP and analytics, with all data resident in a column-format data store.

Microsoft SQL Server 2016

Summary: Microsoft claims in-memory support in *SQL Server 2016* through combination of multiple features that consist primarily of *In-Memory OLTP* for speeding up transactional workloads and *Columnstore* for read-only or read-mostly Data Warehouses.

In-memory OLTP is a memory-optimized database engine integrated into the SQL Server engine, targeted for OLTP workloads. The OLTP engine allows you to create memory optimized tables within your existing database and was designed to improve SQL Server scalability for high-volume transactional workloads by using latch-free and lock-free data structures, compiled stored procedures and multi-version concurrency control to support high concurrency rates. Similar techniques have been available for years in other database systems such as Oracle Database, for which scalability is not a weakness, nor are tables required to fit entirely into memory - a requirement of Microsoft's In-Memory OLTP feature.

"Memory-optimized" tables (as opposed to standard, "disk-based" tables) reside completely in-memory and store data in an uncompressed row format, which is periodically checkpointed to disk based log files. There are two main types of in-memory-optimized OLTP tables: SCHEMA_AND_DATA (i.e. durable tables) and SCHEMA_ONLY (i.e. non-durable tables). The first provides full durability guarantee just like disk-based tables for your OLTP workload, whereas for the second only the schema is written to the disk logs and is aimed at scenarios where data persistence is not required.

Columnstore on the other hand is actually disk-based structures that are organized into compressed columns but still use the regular buffer cache for processing. It consists of Clustered Columnstore indexes (which replace the traditional disk based row-major tables) and Non-Clustered Columnstore indexes (additional columnar structures on the traditional disk based row-major tables). As the columnar structures are not primarily in memory, the use of memory is optimized but the approach is not fully "in-memory". This makes the "in-memory" label somewhat misleading. On the other hand, this approach is not limited by the amount of memory available, as is the case with a pure in-memory database.

In-Memory OLTP and Columnstore indexes were implemented as independent projects, each with a significant list of restrictions and minimal integration across these features. With SQL Server 2016, one columnstore index can be created on the in-memory OLTP tables allowing for analytics on the transactional data, but Microsoft has not yet released any performance, space and logging requirement numbers or benchmarks for such columnar structures in OLTP heavy environments.

Additionally, the columnstore index on in-memory tables must include all the columns and the queries do not take advantage of in-memory native processing for processing such data.

Although these features may improve the performance of specific use cases, broad adoption will likely require the removal of numerous restrictions on their usage as well as enabling the features to be used together. Microsoft has worked towards removing some of these restrictions with SQL Server 2016, most noticeable of which are updatability of Non-Clustered Columnar Indexes and increasing the recommended total memory size for Memory-Optimized tables to 2TB (as opposed to 256GB with SQL Server 2014), but all of the data in every memory-optimized table must fit in memory - a requirement of Microsoft's In-Memory OLTP feature.

> ***"Use the information in Estimate Memory Requirements for Memory-Optimized Tables to estimate the in-memory size of the database's durable memory-optimized tables. Once you determine the size, you need to provide disk space that is four times the size of durable, in-memory tables."***
>
> ***Microsoft SQL Server 2016 Documentation https://msdn.microsoft.com/en-us/library/dn688968.aspx***

Microsoft has made some improvements with the supported functionality of In-Memory OLTP, but the list of restrictions is still significant. This makes the existing applications incompatible with the In-Memory OLTP engine and adds to application complexity. There is no support for partitioning, no support for replication and restrictions around the supported data types with memory optimized tables. In-Memory OLTP doesn't support computed (virtual) columns and doesn't support all index types.. More fundamental restrictions exist around referential integrity constraints across in-memory and on-disk tables, updatability of primary keys and support for DDL triggers must be enforced in application logic and could require significant code changes.

Some other notable limitations that have been relaxed with SQL Server 2016 include the ability to generate parallel processing plans for accessing memory-optimized tables, encryption of sensitive data on disk using Transparent Data Encryption (TDE) for memory-optimized tables, ability to modify the schema of the table after creation and multiple log reader threads for both recovery and checkpoint. The fact that Microsoft had to relax these fundamental restrictions is yet another indicator that Microsoft's in-memory engine is not an integral part of the SQL Server engine and Microsoft has to make various database technologies work with their in-memory and Columnstore engines.

SQL Server 2016 does allow for querying memory-optimized tables on the readable secondary instance, which allows read-only access to all it's databases, with their AlwaysOn technology such that the data is always consistent with the primary.

Technology: In-Memory OLTP (code-named Hekaton) was a multi-year project at Microsoft culminating in the first release within SQL Server 2014. The In-Memory OLTP feature includes memory-optimized tables and native compilation of Transact-SQL stored procedures for accessing those tables. The overall objective of In-Memory OLTP is to maximize the performance of transactional workloads that access tables that are entirely resident in memory and running on modern multi-core processors. This required new data structures, optimistic locking algorithms and multi-version concurrency control. Microsoft claims performance improvements up to 100x compared to SQL Server performance without In-Memory OLTP.

The constraints placed upon the use of In-Memory OLTP in SQL Server 2016 can be substantial. For instance, Microsoft recommends that a database should not use more than 2 TB of In-Memory OLTP data, no support for compression for In-Memory OLTP tables (which can lead to excessive memory requirements) and scaling up to only a 4-socket machine, a modest-sized system in today's world.[8] When using compiled stored procedures, Microsoft has made some improvements with the supported functionality (such as, support for Left and Right outer join, OR and NOT operators, sub-queries and nested stored procedures), but the list of restrictions is still significant, including no triggers or constraints, such as foreign key or unique constraints.[9] This means referential integrity must be enforced in application logic and could require significant code changes.

> ***"Unlike disk-based tables, storage for memory-optimized tables is not compressed. When migrating a compressed (ROW or PAGE) disk-based table to memory-optimized table, you will need to account for the change in size."***
>
> ***Microsoft SQL Server 2016 Documentation https://msdn.microsoft.com/en-us/library/dn133174.aspx***

Strategy: Microsoft's strategy for IMDB technology has been erratic. The company first introduced in-memory technology into SQL Server 2008 R2, by way of the VertiPaq columnstore engine, which later became the xVelocity in-memory analysis engine and then simply a feature called Columnstore

8 http://msdn.microsoft.com/en-us/library/dn170449(v=sql.120).aspx
9 http://msdn.microsoft.com/en-us/library/dn246937.aspx

Indexes in SQL Server 2012. That evolved into Clustered Columnstore Indexes in SQL Server 2014 and a completely separate but parallel effort for OLTP, called In-Memory OLTP aka In-Memory Optimization.[10] The appearance is of separate teams with different agendas and no common strategy. With SQL Server 2016, Microsoft has tried to remove various restrictions on their in-memory features, but all these features don't seem to be well integrated.

Validation: Microsoft's Columnstore index technology has been available since the SQL Server 2012 release supporting analytics. On the other hand, In-Memory OLTP technology was first released with SQL Server 2014. In recent releases, these two features have been integrated to work together and can now support analytics for transactional systems, but significant restrictions still exist which can make the existing SQL Server applications incompatible and require rewrite. As a result there is a scarcity of customer reference material on the use of Columnstore Indexes along with In-Memory OLTP in transactional and mixed workload deployments.

[10] http://searchsqlserver.techtarget.com/feature/In-memory-from-xVelocity-to-the-memory-optimized-columnstore-index

IBM DB2 10.5 with BLU Acceleration

Summary: IBM released *DB2 10.5* in June, 2013. A major feature of that release was *BLU Acceleration* (DB2 BLU), implemented as *column-organized tables* for high-performance analytics. Similar to Microsoft, column-organized tables are actually disk-based and not strictly "in-memory", but can thus exceed the capacity of memory. Column-organized tables are updatable, but intended for read-mostly environments and employ the main technologies associated with in-memory databases.

In August, 2014, IBM added functionality to DB2 BLU (release "*Cancun*") that enabled a table to be represented in both row and column formats, so that the same data could be used for both OLTP and analytics workloads.[11] IBM's data replication product is used to transfer changes from the row format to the column format. The release also added support for a variety of data types and DB2 features not previously supported, though a number of restrictions still remain.[12] Notably, clustering via IBM pureScale is still not supported in DB2 BLU, so you cannot spread a column store across more than one server.

Overall, IBM's approach with DB2 BLU covers many of the IMDB bases. It employs the principal in-memory technologies, supports both row and column formats for mixed workloads, and isn't constrained by the size of memory. However, there are still many incompatibilities with existing DB2 features plus the use of replication to synchronize row and column stores adds contradictory performance overhead and excessive administration.

> ***"To implement shadow tables in your environment, you must set up and configure all software components and replication, create the shadow tables, start replication to populate the shadow tables with data, and enable query routing to shadow tables."***
>
> ***Bringing BLU Acceleration performance to OLTP environments with Shadow Tables: IBM Documentation***
> ***http://public.dhe.ibm.com/ps/products/db2/info/vr105/pdf/en_US/shadowtablesdb21050.pdf***

Technology: The key in-memory concepts that are embodied in DB2 BLU are column-organized tables, compression, SIMD vector processing and data skipping. Similar approaches are used in all major IMDB products for high-performance analytics. What separates DB2 BLU from SAP HANA and Microsoft SQL Server 2016 is the ability to represent the same table in both row and column formats

[11] http://www-01.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.wn.doc/doc/c0061179.html
[12] DB2 for LUW 10.5.0 MQT Restrictions

within a common database. DB2 BLU does this with a “shadow table”; a new type of column-organized table that is derived from and synchronized with a corresponding row table. Shadow tables combine the pre-existing Materialized Query Table (MQT) with new column-organized table technology.

This approach is conceptually similar to that of Oracle’s Database In-Memory option, with its “dual-format” architecture. In the case of DB2 BLU, “shadow tables” are columnar representations of a row format table that are updated via IBM’s InfoSphere® Data Replication product. Shadow tables may contain a subset of the columns from the row format. Using a replication tool to synchronize the two formats can add significant latency, making it more likely that stale data could be returned from the column store. Users can set a latency threshold, above which a query is directed to the row store instead of the column store version. This architecture, though workable in some situations, is not sufficient to support real-time analytics against a high-volume OLTP database if the queries must return current data. Additionally, it could result in application performance degradation due to shadow table replication lag.

Strategy: DB2 BLU is not the first attempt at an IMDB from IBM. IBM Smart Analytics Optimizer for DB2 z/OS (2010) and Informix Warehouse Accelerator (2011) incorporated in-memory database features. Such features evolved from a project code-named Blink, and DB2 BLU is reputed to be a further evolution of the same project.

IBM Smart Analytics Optimizer has since been renamed DB2 Analytics Accelerator for z/OS (IDAA). It combines DB2 mainframe features with the Netezza data warehouse appliance (renamed IBM PureData System for Analytics) that uses a modified version of the open source database PostgreSQL. Evidence (or lack thereof) suggests IDAA has had minimal acceptance by IBM DB2 mainframe customers and could be replaced by DB2 BLU technology, or some variant, in the future. Data type incompatibilities between DB2 z/OS and the Netezza PostgreSQL database are notable issues. IBM has yet to rationalize its Netezza acquisition for Data Warehousing, as compared to the use of DB2 LUW (Linux, Unix and Windows), and DB2 on the mainframe.

Despite multiple attempts at an IMDB strategy, DB2 BLU is clearly IBM’s way forward for DB2 LUW, and may be the unifying approach across IBM’s DB2 variations in the future. Future releases of DB2 BLU are likely to focus on closing the remaining compatibility gaps with DB2, with the most notable issue being the lack of clustering support. For now it appears that Netezza (PureData System for Analytics) will not embrace IMDB technology in the near term and Informix databases will continue on a separate path.

Validation: IBM appears to be putting significant effort into popularizing the use of DB2 BLU and building an active user community. A reasonable amount of customer validation is beginning to appear for DB2 BLU and this critical mass may help to unify the many (sometimes competing) data management agendas at IBM.

MemSQL

Summary: MemSQL Inc. launched its database in June 2012 as a distributed, in-memory SQL database system. There have been 11 subsequent MemSQL releases (5 major) through March 2016. It is a relational system which complies with all ACID properties and is targeted for simultaneous transactions and analytics at scale by combining lock-free data structures and Just-In-Time compilation (JIT) to process highly volatile workloads.

Overall, MemSQL's approach covers many of the IMDB bases and it employs the principal in-memory technologies, including support for both row and columnar representation for mixed workloads. However, the approach that MemSQL took, row-major in memory and column-major on disk, is the opposite of what other IMDB vendors have chosen. In a way, MemSQL's approach is similar to Microsoft SQL Server 2016 where Columnstore Indexes are actually disk-based structures that are organized into compressed columns. But unlike Microsoft, with MemSQL there is no regular buffer cache (or buffer pool) to cache the columnar on-disk data in memory. Data Warehouse applications should experience performance improvement by using compressed columnar disk structures, which reduce the amount of data accessed for analytic queries, but without the use of memory the performance improvement will be limited. For MemSQL the primary use of memory is to hold more recent transactional data in row major representation.

MemSQL requires users to define the tables as reference tables, stored in row format in memory, or regular tables stored in columnar format on disk. There is no guidance provided to the end user on one or the other. In a typical analytic workload there will always be joins across different format tables and as a result in-memory join optimizations cannot be used. This is a major drawback of MemSQL's architecture. Even though MemSQL supports partitioning, all of the partitions of a table need to be in the same format; either in-memory row format or disk-based column format. It doesn't allow for mixed partition level settings, making their approach very restrictive.

Technology: The key concepts embodied in MemSQL are a distributed, scalable in-memory database that combines lock-free data structures and Just-In-Time compilation (JIT) to process highly volatile workloads. More specifically, MemSQL implements lock-free hash tables and lock-free skip lists in memory for fast random access to data.

MemSQL supports storing and processing data using either a completely in-memory row store or a completely disk-backed column store. A given object (table) can either be defined to be in row store in memory or column store on disk, but is not available in both representations simultaneously. While the in-memory row store is designed for optimum performance in transactional workloads, the column store provides cost-effective data storage of large amounts of historical and archival data. A combination of MemSQL row store and column store engines allow merging of real-time and historical data in single query, but this raises a question if various in-memory technologies, such as SIMD vector processing, can be used efficiently to process data at memory bandwidth in this architecture. Hence the performance benefits may be limited when accessing the on-disk columnar structures.

According to MemSQL documentation, the in-memory row store provides low latency and highly concurrent reads and writes for individual rows as well as analytical queries, which are looked up using primary and unique keys. The in-memory row store is periodically checkpointed to disk, and hence has a longer recovery time (the entire table needs to be loaded into memory from the checkpoint and write ahead logs). On the other hand, the on-disk column store works best for analytic workloads and allows for tables larger than the amount of memory available in the cluster. Ironically, compression is supported only on the column store on disk and not in memory.

MemSQL requires you to make a tradeoff between performance and durability. This is a user controllable attribute, and can be tuned all the way from synchronous durability (every write is recorded on disk before the transaction completes) to purely in-memory durability (maximum throughput on writes at **risk of losing recent transactions**).

> ***"You can get full durability at the cost of increased query latency."***
>
> ***MemSQL Documentation http://docs.memsql.com/4.0/faq/#can-i-configure-memsql-to-be-fully-durable***

Even though MemSQL supports a mixed architecture, it doesn't seem to be completely transparent to the application. Applications need to be aware of the tables to be stored in the in-memory row store and the tables that need to be stored in the disk based column store, and there is no simultaneous representation in both these formats. In fact, if the amount of memory used by the row store tables is greater than the memory provisioned, MemSQL will not be able to start any new write transactions and will return a user error. Additionally, if a running query runs out of memory it will be rolled back and result in a user error.

> ***"If the amount of memory used by row store tables is greater than the maximum_table_memory global variable, MemSQL will refuse to start new write queries. If a current running query runs out of memory it will rollback and notify the client of the error."***
>
> ***MemSQL Documentation http://docs.memsql.com/4.0/faq/#can-i-configure-memsql-to-be-fully-durable***

MemSQL supports both distribution of a table on multiple nodes as well as replication of a given table on every node (in case of reference tables, etc.). It makes use of hash-based partitioning to distribute the data. MemSQL is designed to be a scalable architecture, where new nodes can be added to the cluster at any time to increase storage capacity and processing power. Data distribution is done automatically, and the cluster rebalances the data and workload distribution. As data can be replicated across the cluster, a node can go down with negligible effect on performance but at an expense of increased network traffic since MemSQL replication works by transferring snapshot and log files from master to slaves. Nodes periodically commit transactions to disk based logs, along with periodic full backups of the entire in-memory row store. If a node goes down, it can restart using one of these snapshots.

Strategy: MemSQL Database was first released in June, 2012 as an in-memory, SQL database management system. Over years and across many major and minor releases, MemSQL has added capabilities such as replication, columnar data store, distributed architecture, support for JSON and Geospatial datatypes. It ships with an installation, management and monitoring tool called MemSQL Ops, which can be used to set up a distributed MemSQL database across machines, and provide metrics about the overall system.

> ***"MemSQL is not designed to be a blob store or data lake. MemSQL is not designed to run on micro instances or other low-powered computers. It is designed to run on servers with at least 4 cores and 8GB or RAM."***
>
> ***MemSQL Documentation http://docs.memsql.com/4.0/faq/#what-is-memsql-not-for***

Validation: Even though MemSQL Database was first released 2012, the volume of customers and references is very small. MemSQL's architecture makes it a direct competitor for Microsoft SQL Server and the performance gains described are significant but its architecture suffers some serious drawbacks, such as cross format joins, lack of support for different formats at the partition level and tradeoffs between performance and durability.

Oracle Database In-Memory

Summary: Oracle's mainstream IMDB product was released in July, 2014 as the *Oracle Database In-Memory* option (In-Memory). Like other IMDB implementations, It features a column-format in-memory data store with compression, data skipping and SIMD vector processing to support real-time analytics. Additional features, such as in-memory aggregation and in-memory fault tolerance, distinguish Oracle Database In-Memory. Unlike other IMDB implementations, 100% of the Oracle SQL and data types are supported, allowing all existing applications to use Oracle Database In-Memory without any changes. In addition, all the functionality in the Oracle Database is inherited by the implementation of Oracle Database In-Memory, such as logging and recovery, data replication, security and clustering. Oracle Database In-Memory is available as a separately-priced option to Oracle Database 12c Enterprise Edition.[13]

Oracle Database In-Memory features a "dual-format" architecture to represent the same data in both row and column format, allowing the optimizer to determine the best format to satisfy each SQL request. Only specific columns from specific tables or just partitions of tables that require real-time analytics need to be defined for column-format representation. Real-time analytics can run directly against current production data in the column store while transactions continue to use the row store. Changes to the row store are consistently applied to the column store to ensure analytics queries are accessing current data. In addition, the in-memory store allows analytical indexes to be removed from the row store, speeding OLTP by removing the overhead of those index updates.

Oracle Database In-Memory works with any size database, regardless of how much data will fit in memory. Data larger than one server's memory can span a storage hierarchy of memory, flash and disk, and can also be spread across a cluster of servers, each with its own memory and storage hierarchy. When used on an Oracle Engineered System platform, such as Oracle Exadata or Oracle SuperCluster, a unique fault tolerant feature automatically duplicates in-memory data on more than one server in the cluster for continual access to in-memory data should a cluster node fail.

Although Oracle Database In-Memory is compatible with all existing applications, yielding "out of the box" performance gains, even greater performance is possible when applications take full advantage of the potential of IMDB technology. For example, applications that perform set processing and parallelization can exploit in-memory technologies to the fullest. To that end, Oracle has announced a series of new "in-memory" modules for performance-critical functions in applications such as Oracle E-

[13] http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

Business Suite, Oracle PeopleSoft, Oracle JD Edwards, and Oracle Siebel.[14] End user and partner experiences with Oracle Database In-Memory were also described at the June 2014 launch of the product.[15]

In addition to Oracle Database In-Memory, Oracle TimesTen In-Memory Database has been available since 2005 for embedded real-time OLTP applications. Such applications are typically used inside communications networks, financial trading and risk management systems. A key differentiator for TimesTen is the elimination of network latency by embedding the IMDB within the application. By doing so, response times for applications using TimesTen can be measured in microseconds, whereas the addition of a network adds a millisecond or more to the response time.

Technology: Oracle Database In-Memory is pure IMDB technology seamlessly integrated into Oracle Database 12c using Oracle's "dual-format" architecture.[16] The Oracle optimizer decides when to use the in-memory column store and when to use the traditional Oracle row store. Changes to the data are applied to the row store and immediately reflected in the column store via lightweight mechanisms. The in-memory column store also implements in-memory compression, data skipping via Oracle in-memory storage indexes, in-memory scans, joins (via bloom filters) and aggregation, and SIMD vector processing. In short, the latest technologies and techniques for real-time analytics are contained in Oracle Database In-Memory.

The dual-format architecture enables the same production database to service both high-volume transactions and real-time analytics, avoiding the need to copy the production database to another server. The time and overhead of creating the copy is eliminated and the reports and analytic queries access the very latest production data. Since the in-memory column store now services the analytical requests, it's possible to drop previous analytic indexes from the row store and improve OLTP performance as well. All of this is accomplished with no application changes.

Strategy: Oracle has been in the IMDB business longer than any major database vendor by virtue of the TimesTen In-Memory Database acquired by Oracle in 2005. It can also be argued that the Oracle Database In-Memory option is the first mainstream IMDB product that is a complete database implementation. As was detailed, the other implementations have significant restrictions, incompatibilities or incomplete functionality, relative to a mature database system. The strategy Oracle

14 https://www.oracle.com/corporate/pressrelease/real-time-enterprise-093014.html
15 http://www.oracle.com/us/corporate/events/dbim/index.html
16 http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-in-memory-2245633.html

has taken, making ease of use and 100% compatibility a priority, should result in rapid and broad adoption across Oracle's large customer base.

Validation: A number of customers and partners joined in the launch of Oracle Database In-Memory in June, 2014, after an extensive beta test period. Oracle applications teams have already announced new in-memory application modules and the Oracle PartnerNetwork's Oracle Database Ready certification now includes Oracle Database In-Memory.[17] End-user deployment references will soon appear.

For Oracle Database customers, testing the impact of Oracle Database In-Memory is trivial. Install Oracle Database 12c, enable in-memory data and identify the data to be populated in the column store. Existing production databases can be tested for real-time analytics and concurrent OLTP. On Oracle Engineered Systems platforms, such as Oracle Exadata, in-memory fault tolerance is available to ensure uninterrupted real-time processing.

[17] http://www.oracle.com/us/corporate/press/2215799

IMDB Product Comparison Table

	Oracle Database In-Memory 12.1.0.2	SAP HANA SPS11	Microsoft SQL Server 2016	IBM DB2 BLU 10.5	MemSQL 5.0
Release date	July 2014	November 2015	June 2016 (TBD)	June 2013	March 2016
Columnar format	✓	✓	✓	✓	✓
Compression	✓	✓	✓	✓	✓
SIMD vector processing	✓	✓	✓	✓	✓
Data skipping	✓	✓	✓	✓	✓
In-memory aggregation	✓				
In-Memory OLTP optimizations			✓		✓
Size not limited by DRAM	✓	✓[4]	✓	✓	✓
Cluster (scale-out) support	✓	✓			✓
Dual-format in one database	✓			✓	
Consistent updates	✓	✓	✓		
100% application transparency	✓				
Persistence of Column Store		✓	✓	✓	✓
Query on Secondary Replica			✓		✓
Materialized View with Column Store	✓				
Integration with R	✓	✓	✓	✓	✓
Memory-Only columns		✓		✓	

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

blogs.oracle.com/in-memory
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together