The 3-Level Database Architectural Design for OLAP and OLTP Ops

Vaheedbasha Shaik[1]

[1]*Senior Oracle Database Administrator, IT Department, CenturyLink Technologies Pvt. Ltd.*

Abstract - *The Idea behind this Paper is proposing 3-Level Database design for Real-Time Analysis benefits and for Greater access speed. The high seeking time is a very disturbing value for the Database access on the Disk. The only option to keep up the speed with low seeking time is Memory. Nowadays, the Database software vendors also improved their software functionalities to utilize the Memory component in very effective Manner.*

The 3-Level Database design is applicable for all types of Database software binaries unless they are still didn't adopted to In-Memory Option. This design has been proposed to accommodate the both Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) capabilities to keep up with the Real time analysis and Processing as well.

The Benefits of this Design:

- *The Latency of SQL-Query Execution will be very low.*
- *The Access speed will be improved.*
- *The SQL-Query execution will be completed in very less time.*
- *The seeking time could be 10 Nano Seconds or Lower.*
- *The DML Operations will be execution time will be Low.*
- *No Indexes Required except Primary keys or Constraints.*
- *The Storage space will be saved at Greater Level.*
- *The Remarkable Concurrent users' requests count improved per Second.*
- *The Fragmentation Issues will be Resolved.*
- *The Analytics queries performance will be Improved.*

Key Words:OLTP,OLAP,DML,IMDS,RTOS,DBMS,NVRAM, RAM, FeRAM, MRAM,PRAM

1. INTRODUCTION

The 3-Level Database design would be depending upon the characteristics of the tables. The Tables can be categorized depending upon the Operations which are being performed on them. Here, we are considering below factors to categorize them.

- High DML Operational tables.
- Frequently Used Small Tables.
- Frequently Used Summarized/Inventory Tables.
- Mediumly updated/accessed Tables.
- The Archive Data which is not Allowed to be Updated (like… Transaction History).

We are going to categorize all the tables according to these factors to place them at appropriate level of the Database structure. Nowadays, all most all the Database software Binaries have the 3 features Commonly. Those are i) Tables Partition, ii) Compression iii) In-Memory techniques. If your Database Software doesn't support any of these techniques, then it's time for your Databases to Migrate higher version or to other Technologies. We are using these 2 techniques to reorganize our Database objects.

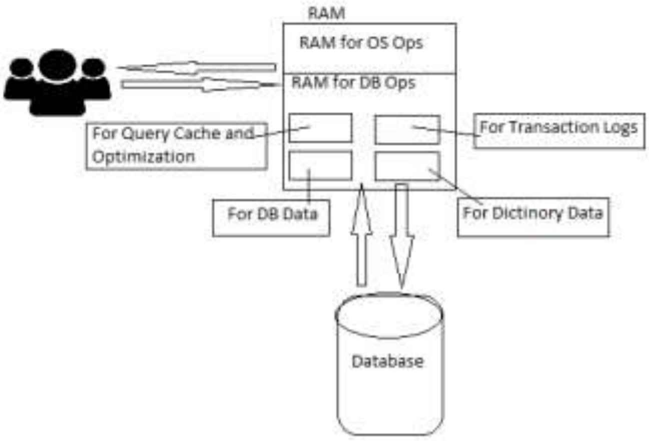


Figure 1

In the Database connectivity, the users will be used their login credentials to establish secure connection with the DB upon authentication. The connection will be attached to the instance in RAM. The Instance will be created with some collection of Memory buffers. These are important to manage the data and to carry out some important functionalities of Databases. The functionality of updating of data on the disk is very uncomfortable to DB. To skip this, we are using the buffers in the RAM. The data will be loaded into these buffers before updating any data or dictionary in the Database.

The main memory buffers have defined in the above Figure 1. The DB Data buffers used to load the data from disk to RAM and returns to the disk once transaction is successful upon certain conditions. The dictionary data buffer will be used to load the objects' dictionary data and object access related information from the DB. The Transaction logs will be used to record the transaction in case DB needs to be Restored or Recovery from failures. The query cache and Optimization will be used to store the old unaltered data and execution plans temporarily in the RAM. In case the same query is fired again then the data can be returned from RAM instead of going back to Disk.

2. EXISTING DESIGN

The Database is a collection of objects and treated as a unit or Set. The purpose of a database is to store and retrieve data related information whenever user sent the processing request to it. In general, a database manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. All this is accomplished while delivering high performance. A database/server also prevents unauthorized access and provides efficient solutions for failure recovery. All Databases usually have 2 types of faces i) Physical Structure and ii) Logical Structure.

The physical structure defines the storing of data in organized manner so that you can access very easily next time when asked for it. The Logical Structure defines under whom ownership you want to keep the data, Authorization for access and Control of it.

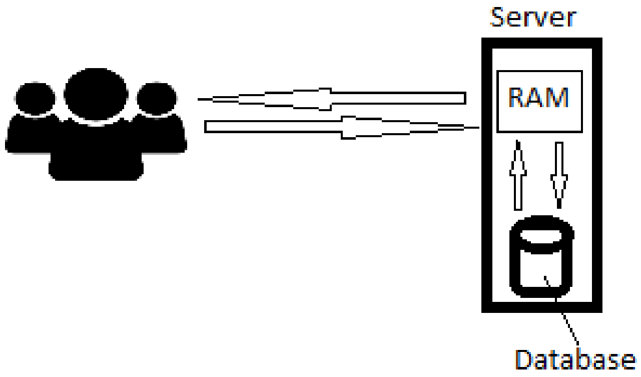


Figure 2

First, Users will be used connection string to establish connection with database which called session with Their authentication methods. Once connection is established the required data from the tables will be loaded into RAM. Some of the RAM would be taken to form an Instance which used to keep the data in the form of buffers from the Tables. From there this data could be transferred to the end user via Network Link in Secure mode. If another user requests data from other table, Then the existing data will be moved to virtual RAM or flushed out completely to make space for new data.

In Existing Setup, the tables used to be stored the data at disk level. In Data retrieval, The RAM will be first verified for the existence of the data whenever users' requests for it. If the data not available in the RAM, then it will be loaded into RAM from disk before initiating the actual updates. To make enough space for the new data loading from Disk the existing data from RAM must be moved out using the Last Recently Used Algorithm of Operating system. Whenever we want to do OLTP and OLAP operations together The OLTP going to suffer because the OLAP operations going to occupy more RAM than OLTP, moreover the OLTP doesn't have fencing mechanism to restrict this.

The Same process will be applicable to all objects in the database irrespective of their importance. The HOT objects which will get frequent requests and Cold Objects which will get very less requests are using the same mechanism. For Real Time Business applications, the OLTP tables should be accessed with more priority than OLAP. Otherwise, The OLAP objects get over the OLTP which can lead to business Impact by slowness.

Hence, The Categorization must be implemented in the Database and Should be prioritized according to their importance to the Current Business.

3. PROPOSED DESIGN TECHNIQUES

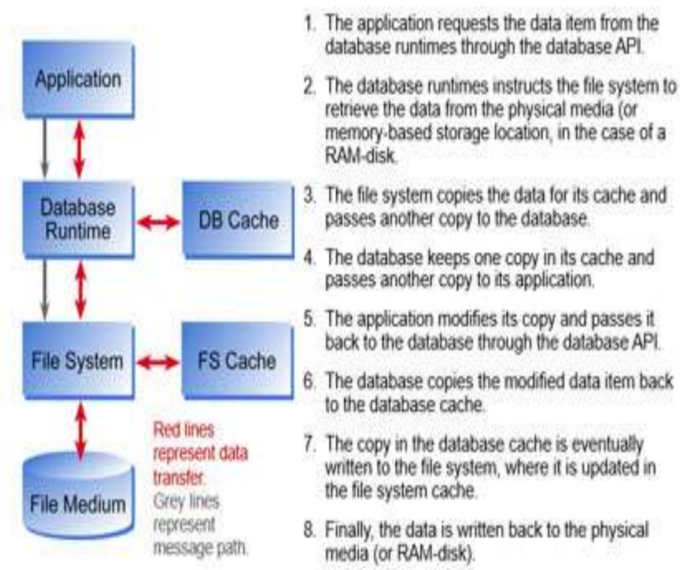
We are mainly using 3 Database techniques to implement 3-Level Database Design. Those are

- In-Memory
- Compression
- Table Partition

3.1 In-Memory Technique

In-Memory databases are mostly used in applications that demand very fast data access, storage and manipulation, and in systems that don't typically have a disk but nevertheless must manage appreciable quantities of data. An important use for in-memory database systems is in real-time embedded systems. IMDSs running on real-time operating systems (RTOSs) provide the responsiveness needed in applications including IP network routing, telecom switching, and industrial control. IMDSs manage music databases in MP3 players and handle program data in set-top boxes. In-memory databases' typically small memory and CPU footprint make them ideal because most embedded systems are highly resource-constrained.

In-Memory database systems (IMDS) are a growing sub-set of database management system (DBMS) software. In-memory databases emerged in response to new application goals, system requirements, and operating environments. This feature is used to store the data in memory rather than on disk. Because working with data in-memory is much faster than writing to and reading from a file system, IMDSs can perform applications' data management functions an order of magnitude faster. Caching is the process whereby on-disk databases keep frequently accessed records in memory, for faster access. However, caching only speeds up retrieval of information, or "database reads." Any database writes – that is, an update to a record or creation of a new record – must still be written through the cache, to disk. So, the performance benefit only applies to a subset of database tasks. To Improve these insert operations, speed up and easy management, we are going to apply partitions.


Application
Database Runtime
DB Cache
File System
FS Cache
File Medium
Red lines represent data transfer
Grey lines represent message path

1. The application requests the data item from the database runtimes through the database API.
2. The database runtimes instructs the file system to retrieve the data from the physical media (or memory-based storage location, in the case of a RAM-disk.
3. The file system copies the data for its cache and passes another copy to the database.
4. The database keeps one copy in its cache and passes another copy to its application.
5. The application modifies its copy and passes it back to the database through the database API.
6. The database copies the modified data item back to the database cache.
7. The copy in the database cache is eventually written to the file system, where it is updated in the file system cache.
8. Finally, the data is written back to the physical media (or RAM-disk).

Figure 3

It needn't be. Most in-memory database systems offer features for adding persistence, or the ability survive disruption of their hardware or software environment. One important tool is transaction logging, in which periodic snapshots of the in-memory database (called "save points") are written to non-volatile media. If the system fails and must be restarted, the database either "rolls back" to the last completed transaction, or "rolls forward" to complete any transaction that was in progress when the system went down (depending on the particular IMDS's implementation of transaction logging).

Non-volatile RAM or NVRAM provides another means of in-memory database persistence. One type of NVRAM, called battery-RAM, is backed up by a battery so that even if a device is turned off or loses its power source, the memory content—including the database—remains. The latest generations replace the battery with a super-capacitor for the same effect. Newer types of NVRAM, including ferroelectric RAM (FeRAM), magneto resistive RAM (MRAM) and phase change RAM (PRAM) are designed to maintain information when power is turned off, and offer similar persistence options.

3.2 Compression Technique

In the hands of experienced DBAs, data compression can go a long way in saving space and increasing the overall efficiency of SQL databases. Further, an oversized database can cause several issues which compression prevents from occurring.

It is prudent for a DBA to consider which form of compression he or she should use and minutely evaluate objects before going ahead with a compress operation.

Despite your best efforts and even after enforcing compression, the database can continue to grow. In such scenarios, you would need to consider alternate solutions like using a third-party tool to deal with any contingencies. The backups also will take less space. The benefits of this feature are

- High Volume of Data read operations can be benefited by compression.
- Archive data can be moved to high level compression technique so that we can save more space.
- Advantage for OLAP operations since high volume of data can be collected from Disk in one io cycle.

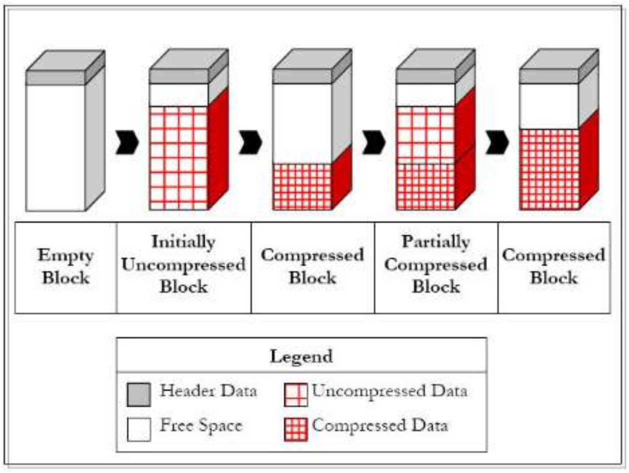


Figure 4

As mentioned in the above diagram the compression will be helpful. By this method we can get most of the data by just accessing small amount of compressed data.

3.3 Table Partition

Table partitioning is a way to divide a large table into smaller, more manageable parts without having to create separate tables for each part. Data in a partitioned table is physically stored in groups of rows called partitions and each partition can be accessed and maintained separately. Partitioning is not visible to end users; a partitioned table behaves like one logical table when queried.

We are going to use the table partition using date as key. For example, the table contains data from every day in 2012, 2013, 2014 and 2015, and there is one partition per year.

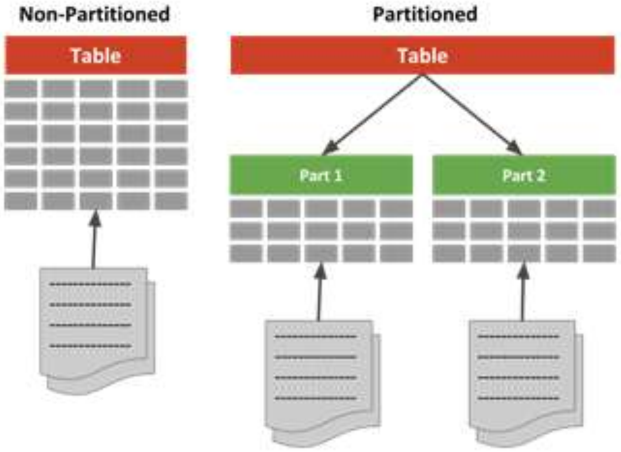


Figure 5

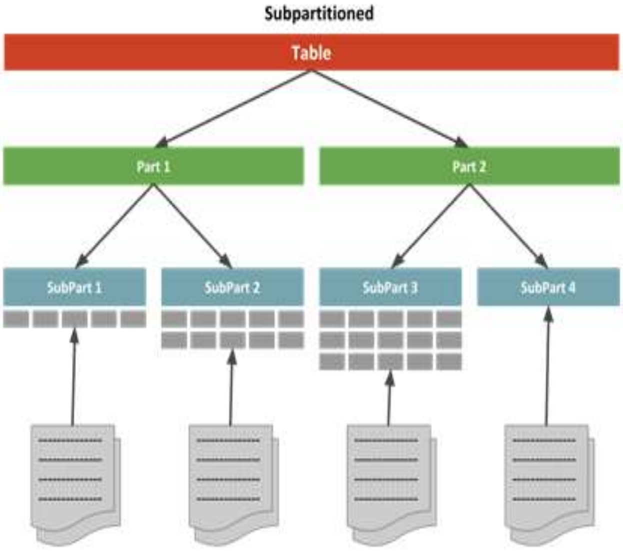


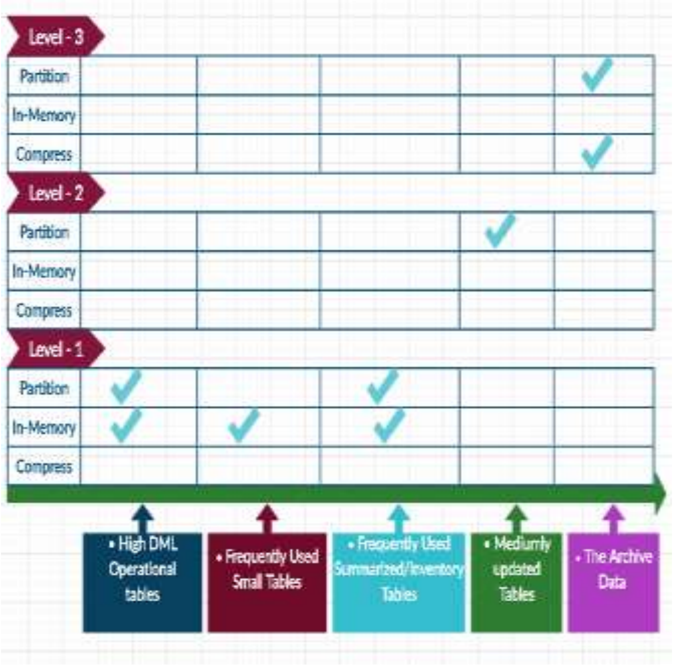
Figure 6

As shown in the above diagram the tables can partitioned or sub partitioned tables.

4. THE 3-LEVEL DATABASE DESIGN CONSTRUCTION BY 3 TECHNIQUES

The proposed system will help us to categorize each table and place them in perfect priority list for better performance of the Database. We are the blow listed factors to categorize the tables.

- High DML Operational tables.
- Frequently Used Small Tables.
- Frequently Used Summarized/Inventory Tables.
- Mediumly updated Tables.
- The Archive Data which is not Allowed to be Updated (like… Transaction History).



	• High DML Operational tables	• Frequently Used Small Tables	• Frequently Used Summarized/Inventory Tables	• Mediumly updated Tables	• The Archive Data
Level - 3					
Partition					✓
In-Memory					
Compress					✓
Level - 2					
Partition				✓	
In-Memory					
Compress					
Level - 1					
Partition	✓		✓		
In-Memory	✓	✓	✓		
Compress					

Figure 7

4.1 High DML Operational tables

We need to trace all the tables which are highly influenced by DML operations. The tracing will be possible by looking at the audits of the tables and displayed them by the order of high DML executions. We are going to implement Partition and In-memory Technique for these tables. For highly influenced tables by DML operations need to be partitioned by using the Date as key. If the volume of data operations is very high, then every day a new partitioned need to be created. Otherwise, every week at least one partition needs to be created. At the end of the Year, The Old data must be moved to archived tables once it's became 365 days older data.

Once partition was successful, we going to implement another technique called In-Memory option. The newly created partitions should be moved to In-memory location by executing the simple SQL command in every software like "alter table table_name inmemory;". The old partition which was already there in memory should be moved out of it once new partition got created.

Thus, we can get the benefit of Partition and In-Memory for highly influenced DML operational tables.

4.2 Frequently Used Small Tables

According to this factor we need to trace all small tables which are influence by select command or DML Operations or Both. Here, we are not going to use Partition technique. But, In-Memory. The tracing of these tables can possible by enabling auditing on the tables. Once you found the all tables list, force them all to stay in memory permanently. It can achieve by executing simple SQL commands in every software like “alter table table_name inmemory;”.

4.3 Frequently Used Summarized/Inventory Tables

With this factor we need to consider two possible options. In case of small tables, the tables should be pushed to memory by using In-Memory technique. In case of big tables only partition technique needs to be applied. Please skip the In-Memory technique for Big tables. The frequent access audit information can be retrieved by enabling the auditing for the tables.

4.4 Mediumly updated Tables

For mediumly updated tables implement partition technique unless they are not big enough in size to create partitions.

4.5 The Archive Data

Please move all old partitions which were created before 365 days to these tables periodically. Create all archive tables with compression technique henceforth all the inserted data in these tables automatically compressed. The compression technique will boost the OLAP operations more effectively. We need not to worry about DML operations on these tables since they are archived data.

REFERENCES

[1] https://app.creately.com/diagram/3ZNc27nHGyk/edit

[2] Oracle online Documents on Partition, in memory and Compress.

[3] Mysql Online Documents on Partition and Compress.

BIOGRAPHIES

Vaheedbasha Shaik has received Master of Technology (M. TECH) in Specialization of Database systems From SRM University and Currently working as Senior Oracle Database Administrator for the IT Department in CenturyLink Technologies Pvt. Ltd.