

Database Management System
(CSE-2004)

In-Memory Database

Presentation by Vibhu Kumar Singh(19BCE0215)
Submitted to Nancy Victor Ma'am





Today's Discussion

Outline of Topics

- 01 Introduction to In-Memory Database
- 02 What is In-Memory Database?
- 03 Why Use an In-Memory Database?
- 04 How Does an In-Memory Database Work?
- 05 How to Implement an In-Memory Database?
- 06 Uses and softwares that make use of IMDB



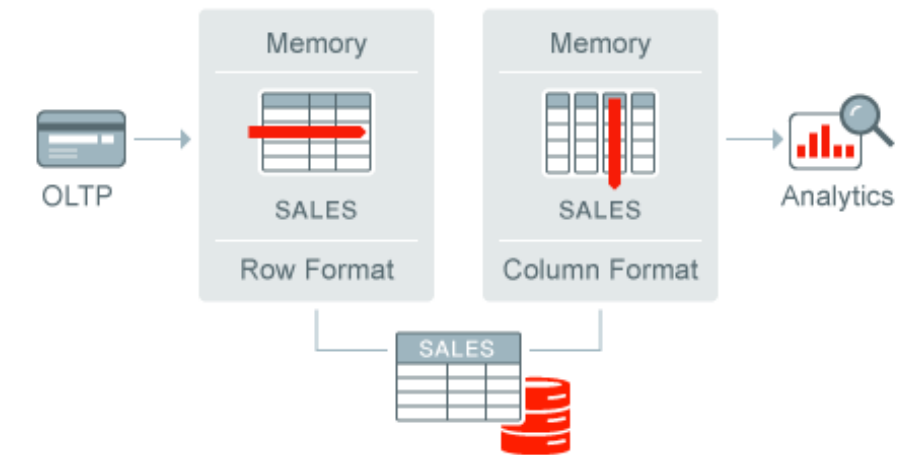
INTRODUCTION

A Brief Introduction

An in-memory database (IMDB) stores computer data in a computer's main memory instead of a disk drive to produce quicker response times. Accessing data stored in memory eliminates the time needed to query data from a disk. In-memory databases are used by applications that depend on rapid response times and real-time data management. Industries that benefit from in-memory databases include telecommunications, banking, travel and gaming. An in-memory database is also referred to as a main memory database (MMDB), real-time database (RTDB) or in-memory database system (IMDS).



What Is an In Memory Database?



Advantage

An in-memory database keeps all its data in the random access memory (RAM) of a computer. Only the main memory is accessed when querying data. This allows for faster access of that data than a disk-based system.

Downside

The downside is the volatility of RAM. The data is lost when an in-memory database crashes. The development of non-volatile random access memory (NVRAM) can help in-memory databases maintain data after a power loss or crash. Flash is one example, but a major drawback is the limit to how many times flash memory can be erased and rewritten. NVRAM chips are being developed that provide a more persistent memory than flash.



What is the difference between an IMDB and simply storing data in shared memory segments?

The biggest difference between in-memory databases and storing data in shared memory segments deals with a structured approach to data access. In-memory databases maintain components of the “ACID” attributes of data storage engines. The ACID properties include Atomicity, Consistency, Isolation, and Durability. While in-memory databases may relax the durability property based on non-persistent storage, they commonly support the other properties:

- Atomicity – multiple changes are committed to the database as an all or nothing operation
- Consistency – structural rules and relationships are maintained for all users
- Isolation – transactional changes cannot be seen by other users until they are committed.

On-Disk Databases

- All data is stored on disk, disk I/O needed to move data into main memory when needed.
- Data is always persisted to disk.
- Traditional data structures like B-Trees designed to store tables and indices efficiently on disk.
- Support a very broad set of workloads, for example, OLTP, data warehousing, mixed workloads etc.

In-Memory Databases

- All data stored in main memory, no need to perform disk I/O to query or update data.
- Data is persistent or volatile depending on the in-memory database product.
- Specialized data structures and index structures assume data is always in main memory.
- Optimized for high-performance.





Why Use an In-Memory Database?

Applications that manage vast quantities of data and require rapid response times can benefit from in-memory database architecture. The data analytics industry increasingly relies on in-memory database systems. The benefits of an in-memory database include:

- Faster transactions
- No translation
- Multi-user concurrency

Advantages

- In-memory databases, by definition, handle the data they are processing in main memory.
- There is no need to deal with secondary storage which can be orders of magnitude slower than accessing data held in main memory.
- Eliminating the requirement of accessing the slower secondary storage allows for the use of algorithms in an in-memory database that would not be feasible for a disk-based database.
- As an example, a disk-based database commonly uses a b-tree based index to limit the number of disk access required to locate a row.
- An in-memory database can use an AVL tree instead of a b-tree which reduces (or eliminates) the need to duplicate data but increases the number of rows accessed during traversal.

Disadvantages

- Typically, the type of memory used with an in-memory database system is not persistent. When an application closes, whether cleanly or unexpectedly, the data stored in an in-memory database will be lost.
- Certain application domains do not require data persistence between runs, but others may. Those applications that require a high-degree of persistence may not be suitable for using an in-memory database.
- An in-memory database stores all data in main memory which can severely limit the amount of data that can be stored. Most database systems can handle allocating memory ad-hoc for storing database objects or can be given a chunk of memory to use as storage.
- Either way, the volume of data that can be stored in an in-memory database tends to be much smaller than that stored in a disk-based system.

How do you cope with these disadvantages?

An application developer needs to understand that data loss is a possibility when working with an in-memory database. There are several approaches, such as persistence and replication, used to mitigate data loss scenarios, but data loss is still a possibility.

Persistence

- Volatile – the database is empty when first opened and all contents are discarded when the database is closed
- Persistent – on open the database is populated from content on secondary storage, on close changed data (inserts, updates, deletes) is written out to secondary storage

If a database is opened using the persistent in-memory mode changed content will automatically be written to secondary storage when the database is closed. In addition, the developer can persist change to secondary storage on demand. Using persistence can limit data loss to what has happened since the last persistence operation.

Replication

Many database systems support replicating changes to another database instance (or another database system). Using replication allows data that may have been lost from the in-memory copy to be recovered from the replicated copy.



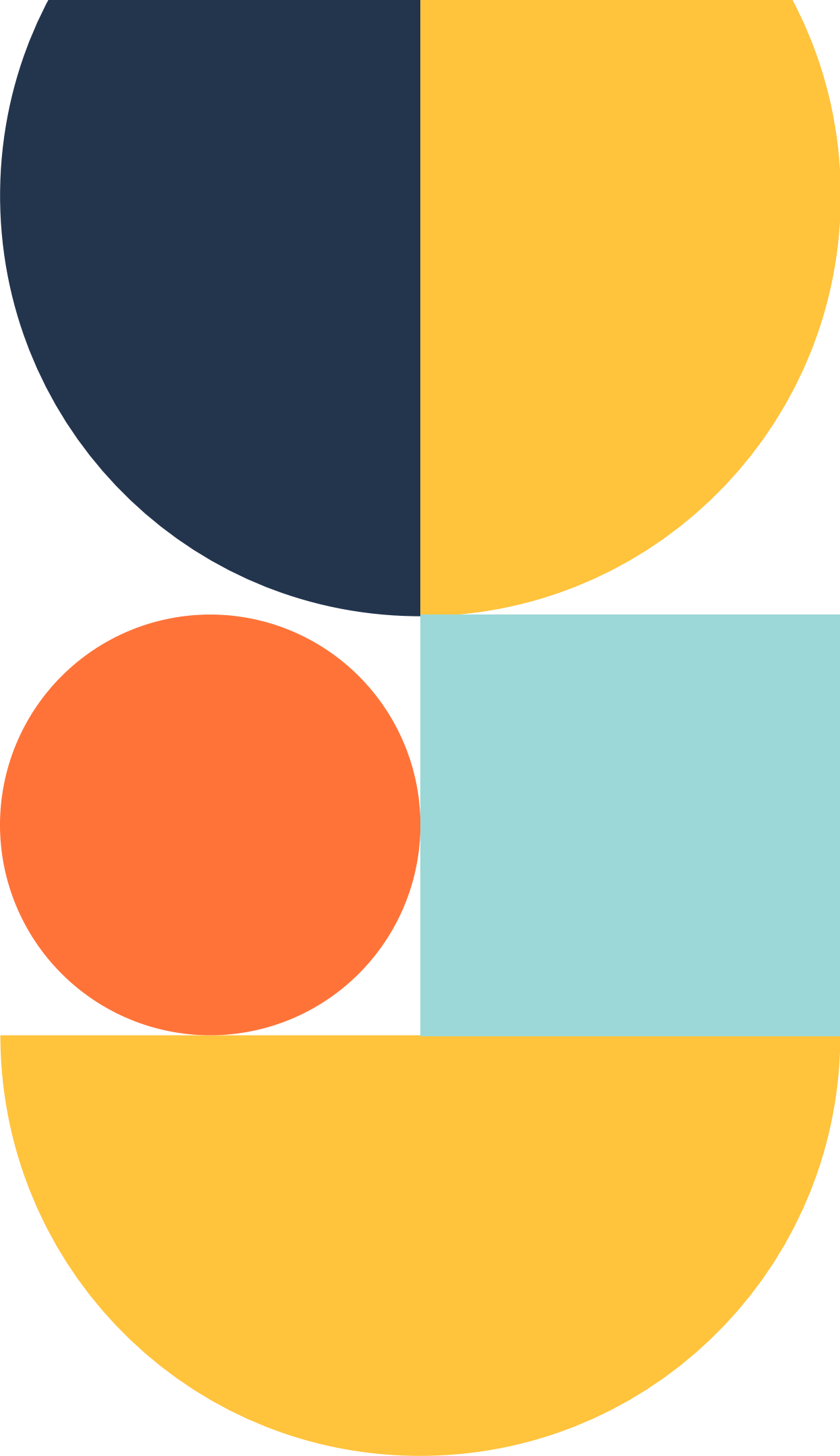
How Does an In-Memory Database Work?

Data storage in an in-memory database relies on a computer's random access memory (RAM) or main memory instead of traditional disk drives. Data is loaded into an in-memory database in a compressed and non-relational format. The data is in a directly usable format without the barrier of compression or encryption. It allows for direct navigation from index to row or column and is a read-only system.

The speed of an in-memory database is made possible by lack of translation and caching. The data is used in the same form as the application that contains it. Data access is managed by an in-memory database management system.

An in-memory database system can also act as an read-only analytic database that stores historical data on metrics for business intelligence (BI) applications. This eliminates data indexing, which can reduce IT costs. Multi-core servers, 64-bit computing and lower RAM prices have made in-memory analytics more common.





How is data accessed and changed in an in-memory database management system?

Data in an In-memory database is “ready to go.” Where data in a disk block may be compressed, encrypted, flattened or encoded, data in memory is in a directly usable format. It also has structure that is independent of disk blocking issues, allowing direct navigation from column to column, row to row or index to row, allowing changes to be implemented by allocating memory blocks and rearranging pointers.

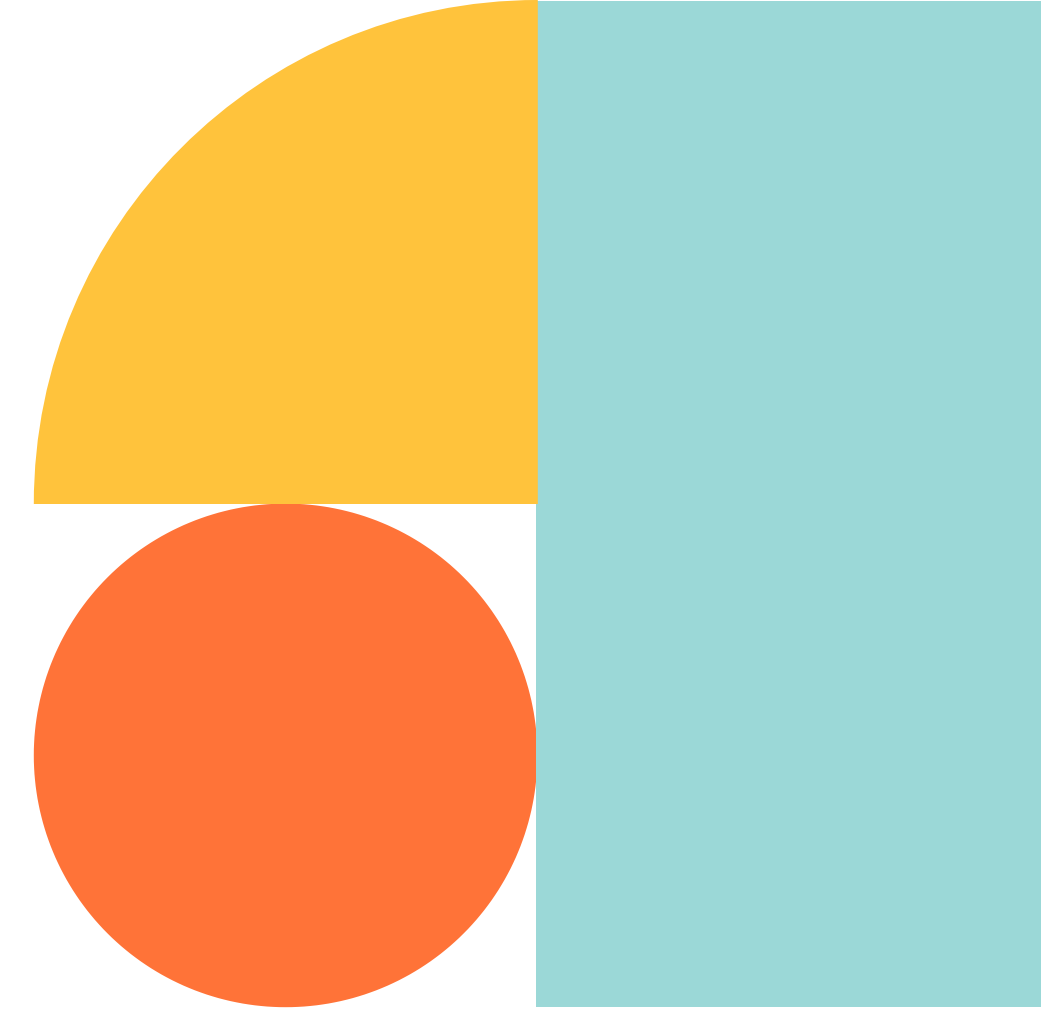
How is in-memory data shared among multiple tasks?

The answer is twofold: fast and faster. For those familiar with the Transactional File Server (TFS), which can be used to share a database among user processes in the same computer, in the office LAN or across the world, the database can be managed in-memory by the TFS.

This is faster than on-disk databases managed by the TFS. But the fastest way to share one database among multiple tasks is to run a single process with multiple threads, where each task is running in its own thread. This form of an application runs on one computer and allows each thread to access the database directly in local heap storage. It also circumvents the need to read unchanged database objects into a local cache, because the original object can be viewed directly from the database memory. This is a very fast way to share a database in a multi-user mode.



Are all embedded databases also in-memory databases?



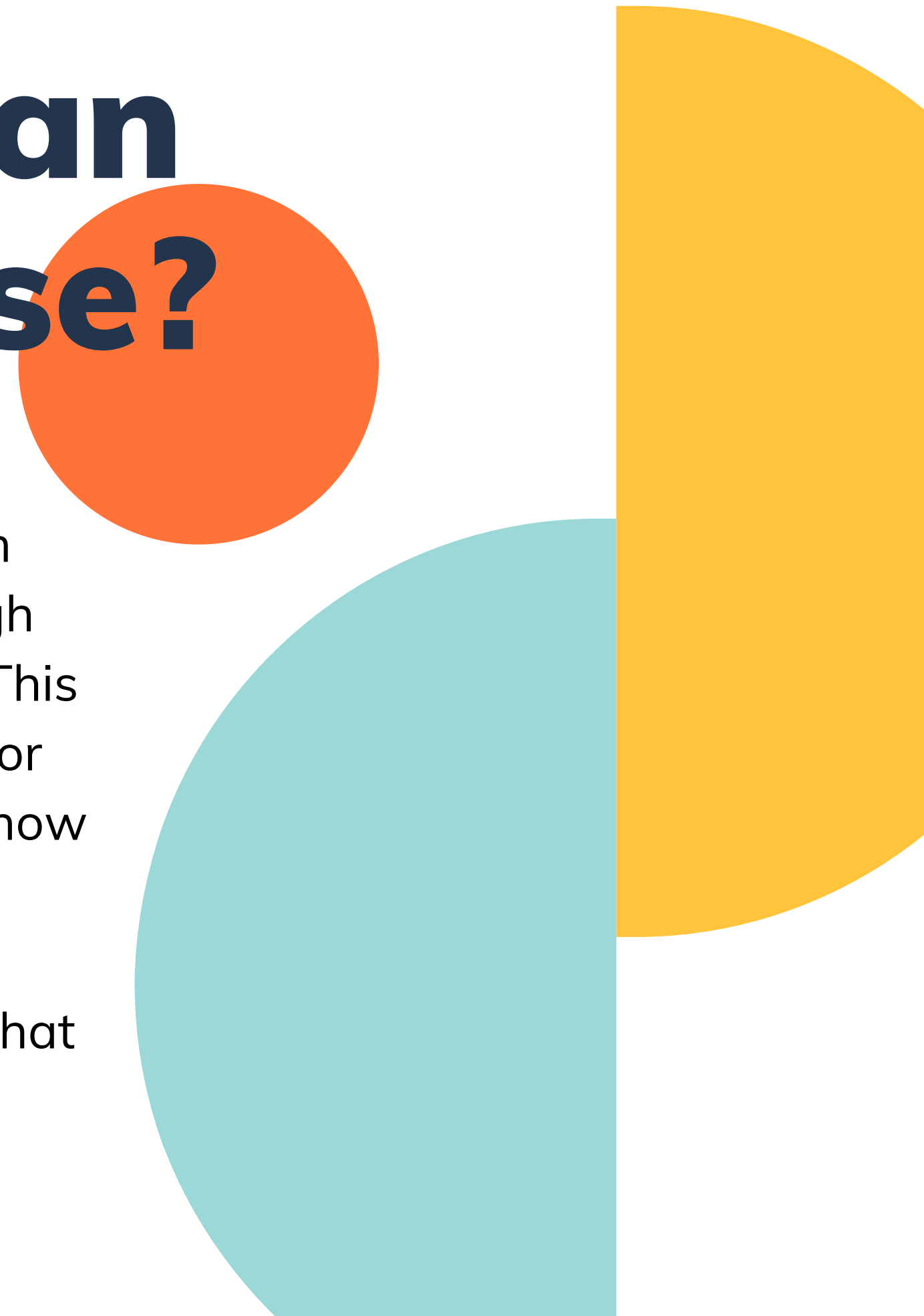
An embedded database can be defined as database engine that runs within an application and does not require other processes to be installed, configured, or accessed. The storage media for the data managed by the engine is implementation dependent.

Originally most database engines used a hard-disk for data storage as the amount of memory available would not allow for sufficient data volume for useful data sets. As the size of memory increase, many vendors began adding in-memory capabilities. Today many database engines, embedded or traditional, have some support for in- memory.

How to Implement an In-Memory Database?

To avoid the risk of losing data in a power outage or a computer crash, enhance an in-memory database with non-volatile random access memory (NVRAM). Flash is commonly used, despite its high cost and limitation in number of times memory can be rewritten. This can help in-memory databases maintain data after a power loss or crash. Flash is one example, but a major drawback is the limit to how many times flash memory can be erased and rewritten.

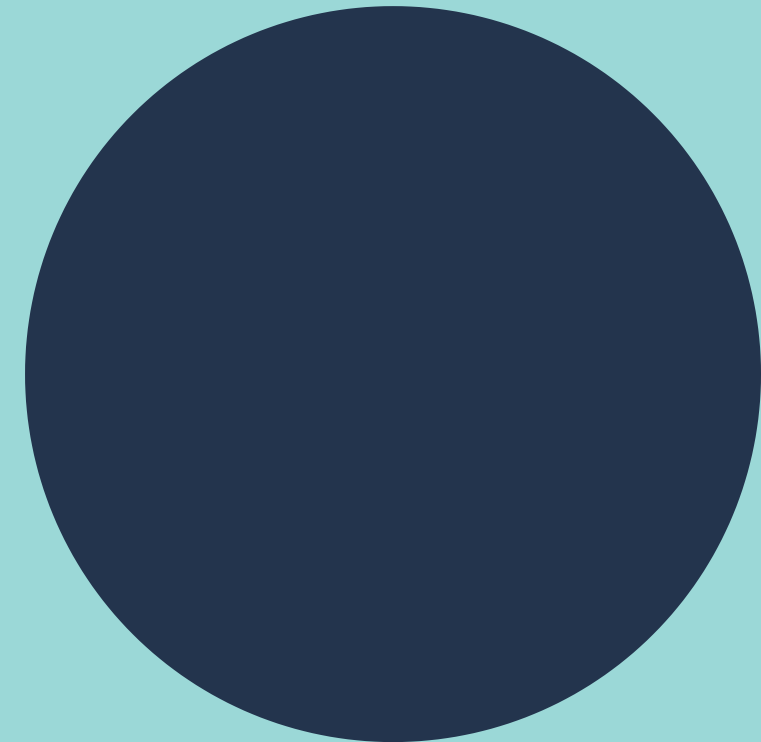
Reading a database from flash is faster than using a disk drive. That is why NVRAM chips are being developed that provide a more persistent memory than flash.



Uses of IMDB

In-memory databases are commonly used for:

- Real-time banking, retail, advertising, medical device analytics, machine learning and billing/subscriber applications
- Online interactive gaming
- Geospatial/GIS processing
- Processing of streaming sensor data
- Developing embedded software systems
- Applications in transport systems, network switches and routers
- Fulfilling the requirements of e-commerce applications



System softwares that use IMDB

List of softwares:



Aerospike DB

Flash-optimized in-memory open source NoSQL database.



Altibase HDB

"Hybrid DBMS" that combines an in-memory database with a conventional disk-resident database in a single unified engine.



Apache Ignite

Apache Ignite is an in-memory computing platform that is durable, strongly consistent, and highly available with powerful SQL, key-value and processing APIs.



Arango DB

Written in C++ and optimized for in-memory computing. In addition ArangoDB integrated RocksDB for persistent storage.

In-Memory Database

by Omni-Sci

All you need to know
about IMDB

by raima.com

Wikipedia.com

References

BOOKS AND WEBSITES



THANK YOU

The Video Presentation link is:

<https://youtu.be/l-H2Mu9s-1w>

