In Memory Database (IMDB)

Introduction

- An **IMDB** also called **Main memory Database(MMDB)** is a database whose primary data store is main memory. That means in **IMDB** the primary copy lives permanently in memory.

History

- Is this a new idea? NO!!!

- Why now so important?

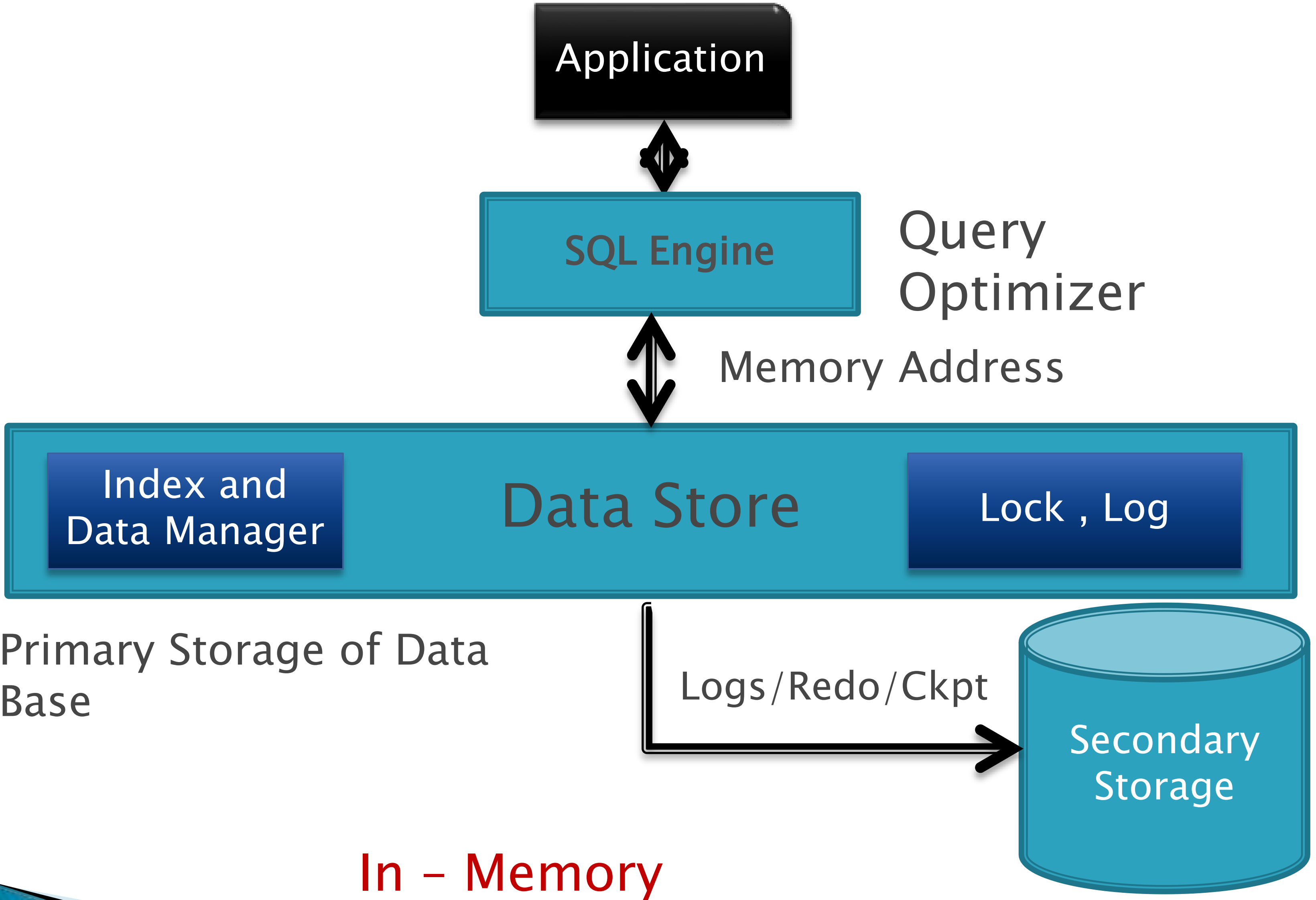
  Due to 4 factors

Factors:

- **LOWER ING COSTS & GROWING SIZE (RAM):**
  - In early 2000, the cost of 64 MB RAM @ $71
  - But now , 8GB DDR3 @ $69.99
- **MULTICORE PROCESSORS**
  - parallel and faster computation
- **64 bit Computing**
  - multiple GB of main memory
- **Faster responses to queries**

In-memory	Traditional
Data stored in main memory	Data stored in disk
Data may be persistent or volatile	Data is always persistent
Size is less or limited because of less main memory	Size is large
Extra memory for cache	No extra memory
Optimized for specialized workloads	Support very broad set of workloads

Architecture



In – Memory Database

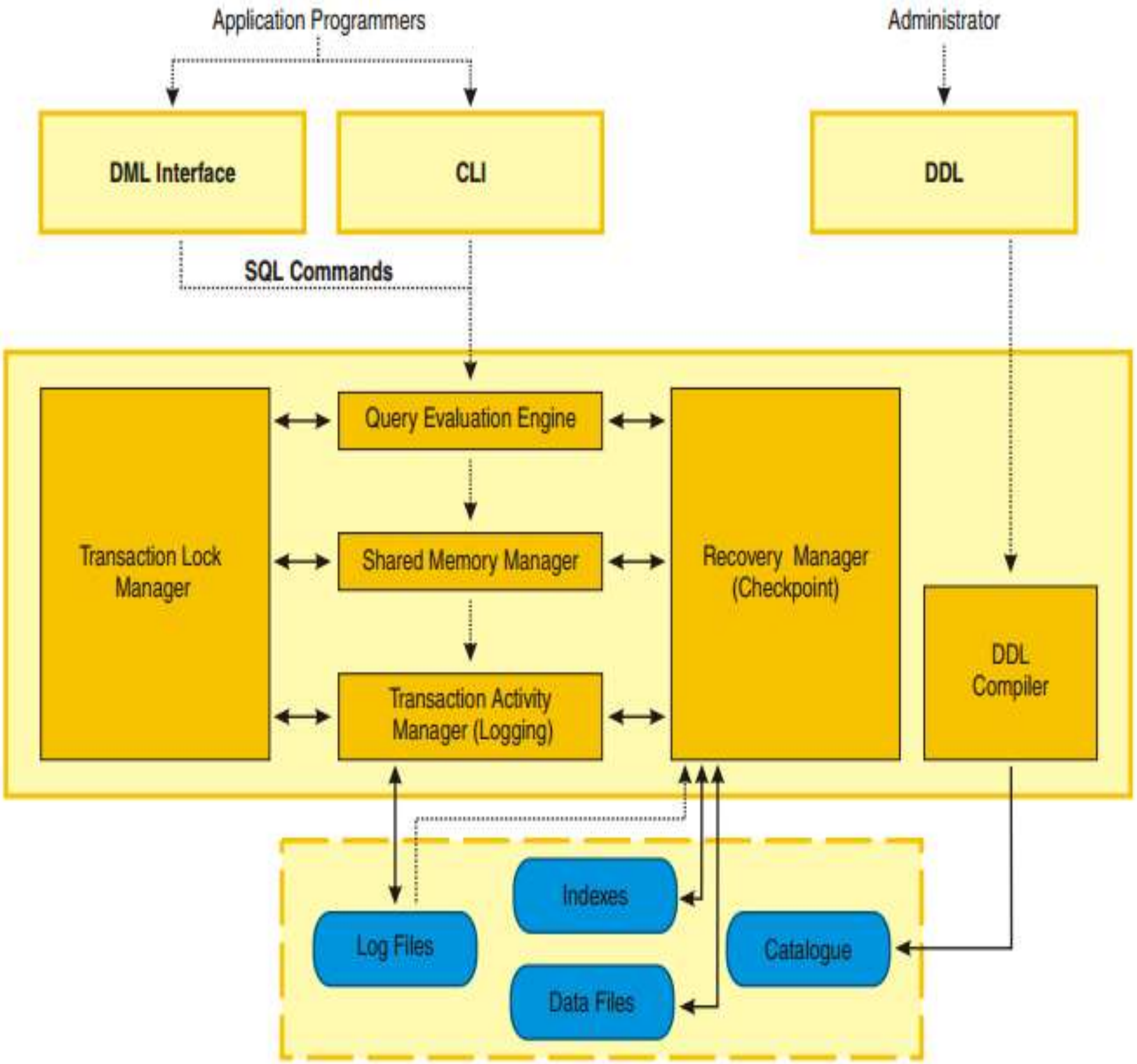
Recovery Purpose

Architecture

- 1.IMDB eliminates disk access by storing and manipulating entire database in main memory.
- 2. The access time for main memory is orders of magnitude less than for disk storage
- 3. Disks have a high, fixed cost per access that does not depend on the amount of data that is retrieved during the access. For this reason, disks are block-oriented storage devices. Main memory is not block oriented.
- 4. The layout of data on a disk is much more critical than the layout of data in main memory, since sequential access to a disk is faster than random access. Sequential access is not as important in main memories.
- 5. Buffer pool management totally disappears, number of machine instructions are reduced the structure and size of index pages is simplified, consequently the design becomes simple and more compact and most importantly requests are executed faster.
- 6.You can see there is a secondary storage used for writing logs , checkpoints etc.

Architecture of In-Memory Database

Most of the In-Memory Database is based on the architecture shown below. The components shown in the yellow space are all in-memory components.



Practical Application

- Applications that demand very fast data access, storage and manipulation
- In real-time embedded systems
- Music databases in MP3 players
- Programming data in set-top boxes
- e-commerce and social networking sites
- financial services and many more…

IMDB vs. DRDB(Disk Resident DB)

Disk Resident Data Base	In-Memory Data Base
Carries File I/O burden	No file I/O burden
Extra memory For Cache	No extra memory
Algorithm optimized for disk	Algorithms optimized for memory
More CPU cycles	Less CPU cycles
Assumes Memory is abundant	Uses memory more efficiently

Challenges in IMDB

- Durability
- Query optimization
- Size of Data Base

Impact of IMDB:

- Data Representation,
- Concurrency control,
- Data Access Methods,
- Query Processing,
- ACID Properties,
- Recovery.

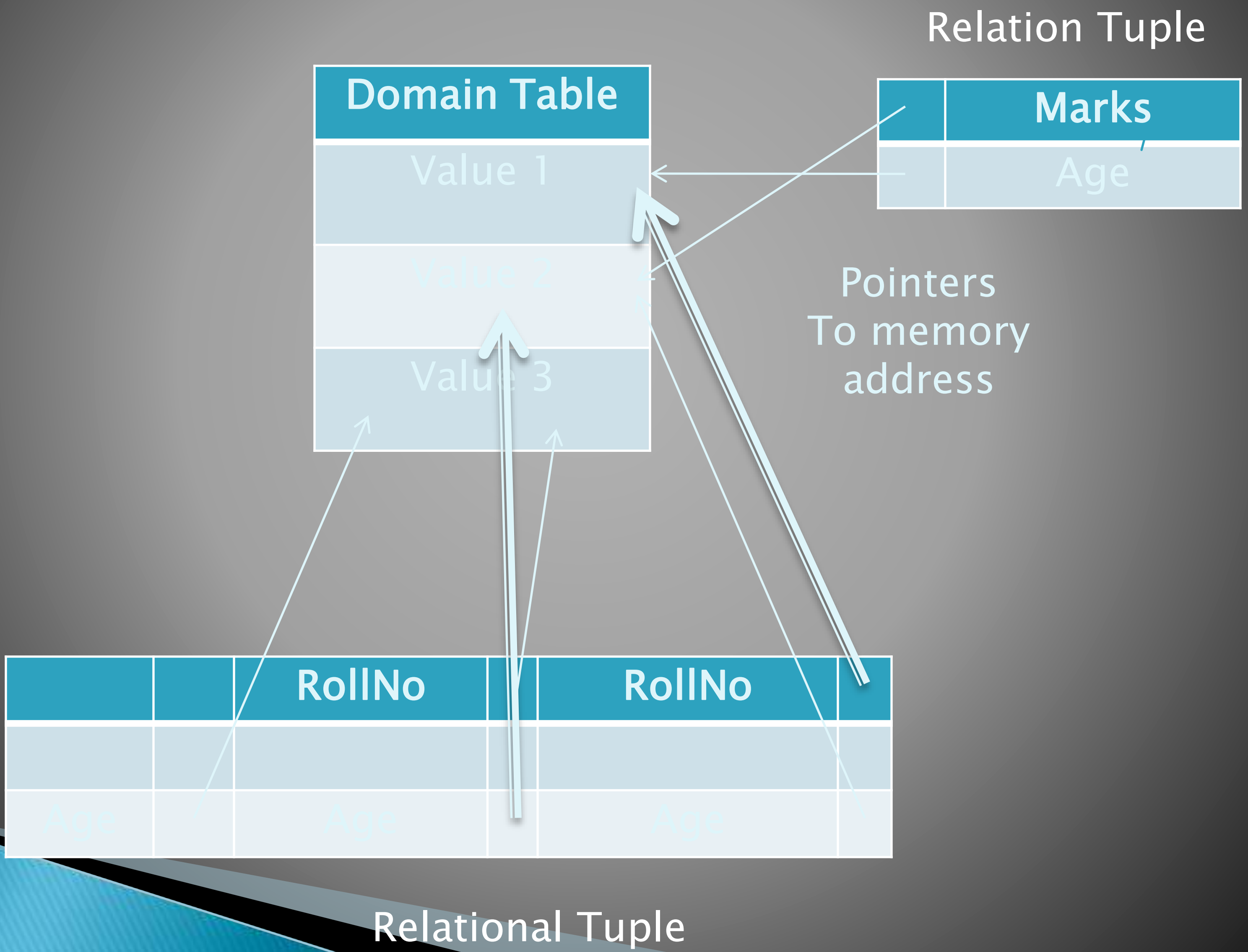
Data Representation

- In Disk Resident DB, we use flat files and sequential access.
- In IMDB, Relational tuples with direct pointers.
  - Space efficient.
  - Shared between columns and relations.

Data Representation

- Main memory databases can also take advantage of efficient pointer following for data representation. Relational tuples can be represented as a set of pointers to data values. The use of pointers is space efficient when large values appear multiple times in the database, since the actual value needs to only be stored once.
- Relational data are usually represented as flat files. Tuples are stored sequentially. Enumerated types larger than the pointer size are stored in the tuple as pointers to the domain table values, domain tables can be shared among different columns and even among different relations.

Data Representation Diagram



Concurrency Control(lock based)

- In DRDB , locking granules are low level.
  - To reduce contention
- In IMDB, due to fast processing it create coarser locks.
  - locking granules like a relation or entire database.

Data Access Methods

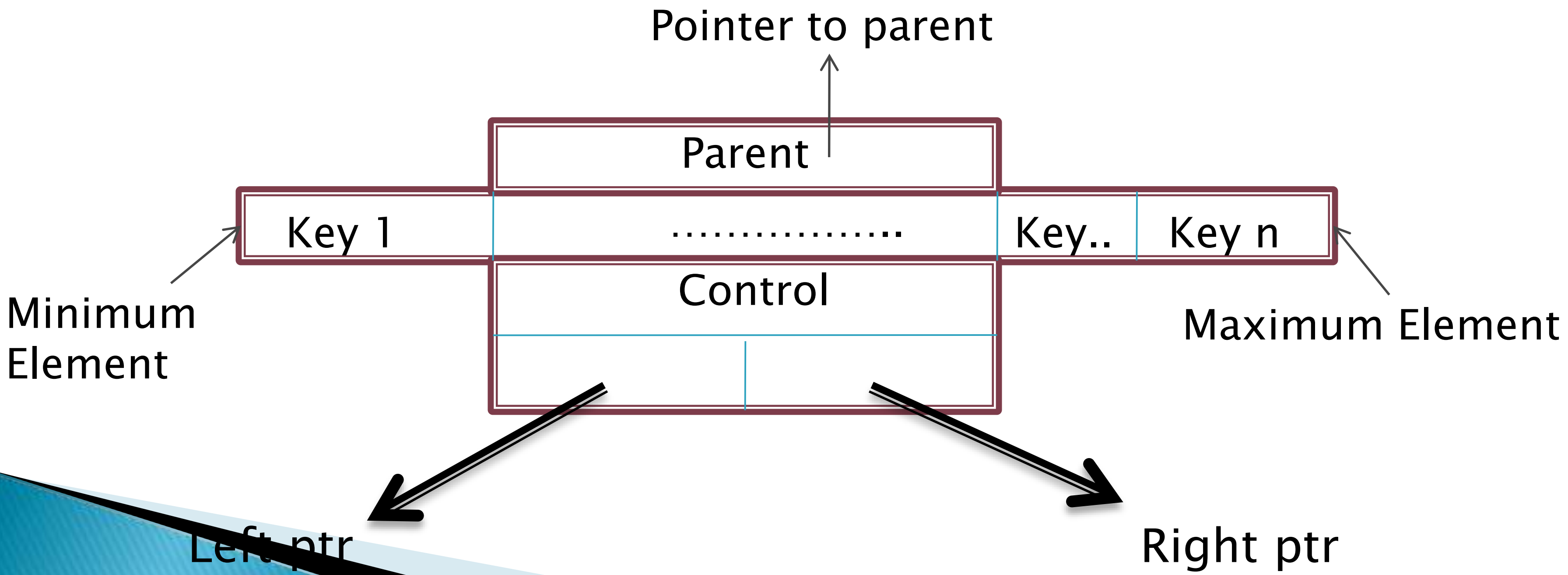
- In DRDB, B–tree index structure is used.
  - ranges , exact match queries
  - lies in hard disk

- In IMDB, T–tree index structure is explicitly designed

  A T–tree is a balanced index tree data structure optimized for cases where both the index and the actual data are fully kept in memory. T–trees do not keep copies of the indexed data fields within the index tree nodes themselves. Instead, they take advantage of the fact that the actual data is always in main memory together with the index so that they just contain pointers to the actual data fields.

T-tree

- T-tree node consists
  - ordered elements in the range min and max values
  - two pointers to the left and right nodes



T-trees uphold the fact that the actual data is always in main memory collectively with the index, hence it do not keep copies of actual attribute values within the index tree nodes. Instead it just contains pointers to the actual data fields . It is an ordered structure like an AVL tree having multiple keys per node. It is an Ideal index structure for ordered search over data.

Other index structure supported by MMDB is heap file for handling a large number of fixed-length data items. Hash file supports unordered scan of data items as well as locking of data item that are obtained transparently when items are inserted, deleted, updated or scanned. Use of T-trees dramatically reduces the CPU processing required to access data and completely eliminates the index value compression and expansion found in B-trees.

Query Processing

- In DRDB, main focus is on processing costs, and attempt to minimize disk access.

- In IMDB, main factors are
  - Cardinality of table
  - Presence of index
  - Any ORDER BY clause
  - Predicate evaluation

ACID Properties

- IMDBs can be said to lack support for the durability portion of the ACID

- Many MMDBs have added durability via the following mechanisms:
  - Checkpoints
  - Transaction logging
  - NVRAM(Non-Volatile RAM)

Recovery

- Mechanisms for recovery are :
  - Logging
  - Checkpoints
  - Reloading

- transactional durability is kept, by keeping two separate but synchronized copies of the database at all times as well as storing log files on-disk.

Challenges in IMDB

- Durability
- Query optimization
- Size of Data Base

IMDB Open Sources:

- CSQL
- HyperSQL
- VoltDB
- Mcobject
- MonetDB