PERVASIVE DATA MANAGEMENT

MAIN MEMORY DATABASES (MMDB)

Prof. Fabio A. Schreiber
Dipartimento di Elettronica e Informazione
Politecnico di Milano

MAIN MEMORY (MM) DATABASES Vs. DISK RESIDENT (DR) DATABASES

- **THE PRIMARY COPY OF DATA LIVES PERMANENTLY IN MAIN MEMORY**
  - **THE PRIMARY COPY OF DATA IS PERMANENTLY DISK RESIDENT**
- **THERE CAN BE A BACKUP COPY RESIDENT ON DISK**
  - **DATA CAN BE TEMPORARILY CACHED IN MAIN MEMORY FOR ACCESS SPEED-UP**

MAIN MEMORY Vs. DISK STORAGE

1. ACCESS TIME OF MM ORDERS OF MAGNITUDE LESS THAN FOR DISKS (10^2 nsec vs. 10 msec)
2. MMDBMS FOOTPRINTS RANGE BETWEEN 200 KB AND 2 MB
3. MM IS *NORMALLY* VOLATILE; PERMANENT MM STILL EXPENSIVE
4. DISKS HAVE HIGH FIXED COST PER ACCESS INDEPENDENT OF THE AMOUNT OF RETRIEVED DATA (BLOCK-ORIENTED)
5. MM DOES NOT CARE OF SEQUENTIAL ACCESS
6. MM DATA ARE MORE VULNERABLE TO SOFTWARE ERRORS SINCE THEY CAN BE DIRECTLY ACCESSED BY THE PROCESSOR

MAIN MEMORY Vs. DISK STORAGE RELIABILITY

EVEN IF SPECIAL HARDWARE CAN ENHANCE MM RELIABILITY, PERIODIC BACKUP IS NECESSARY

- **MM CONTENT LOST IF SYSTEM CRASHES**
- **IF A SINGLE MEMORY BOARD FAILS THE ENTIRE MACHINE MUST BE POWERED DOWN LOOSING ALL THE DATA**
- **WHATEVER POWER BACKUP FOR MM IS, IN TURN, LESS RELIABLE THAN PASSIVE MAGNETIC MEDIA**

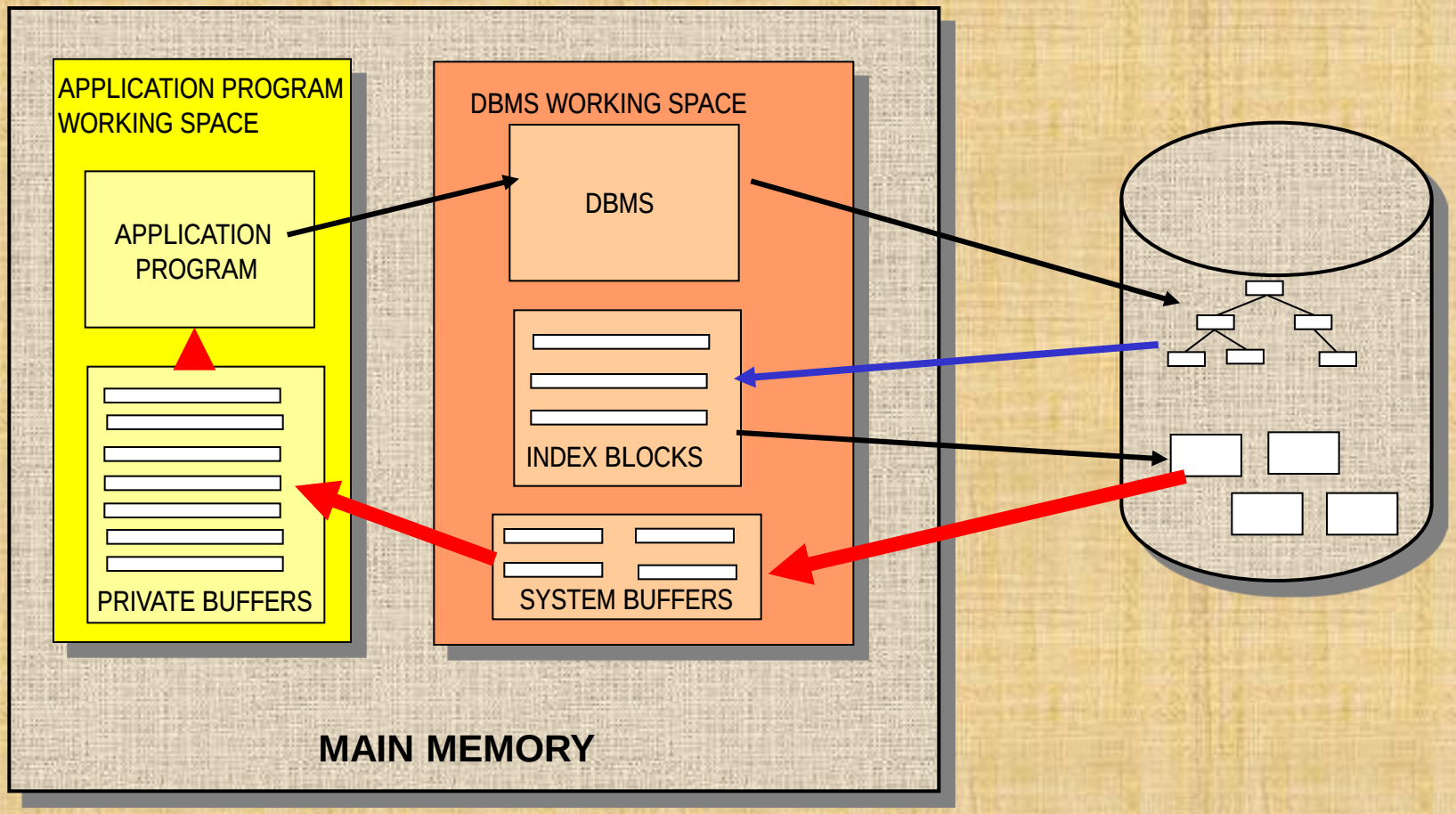
MAIN MEMORY Vs. DISK STORAGE DATA STRUCTURES

MMDB ARE NOT DRDB WITH A VERY LARGE CACHE

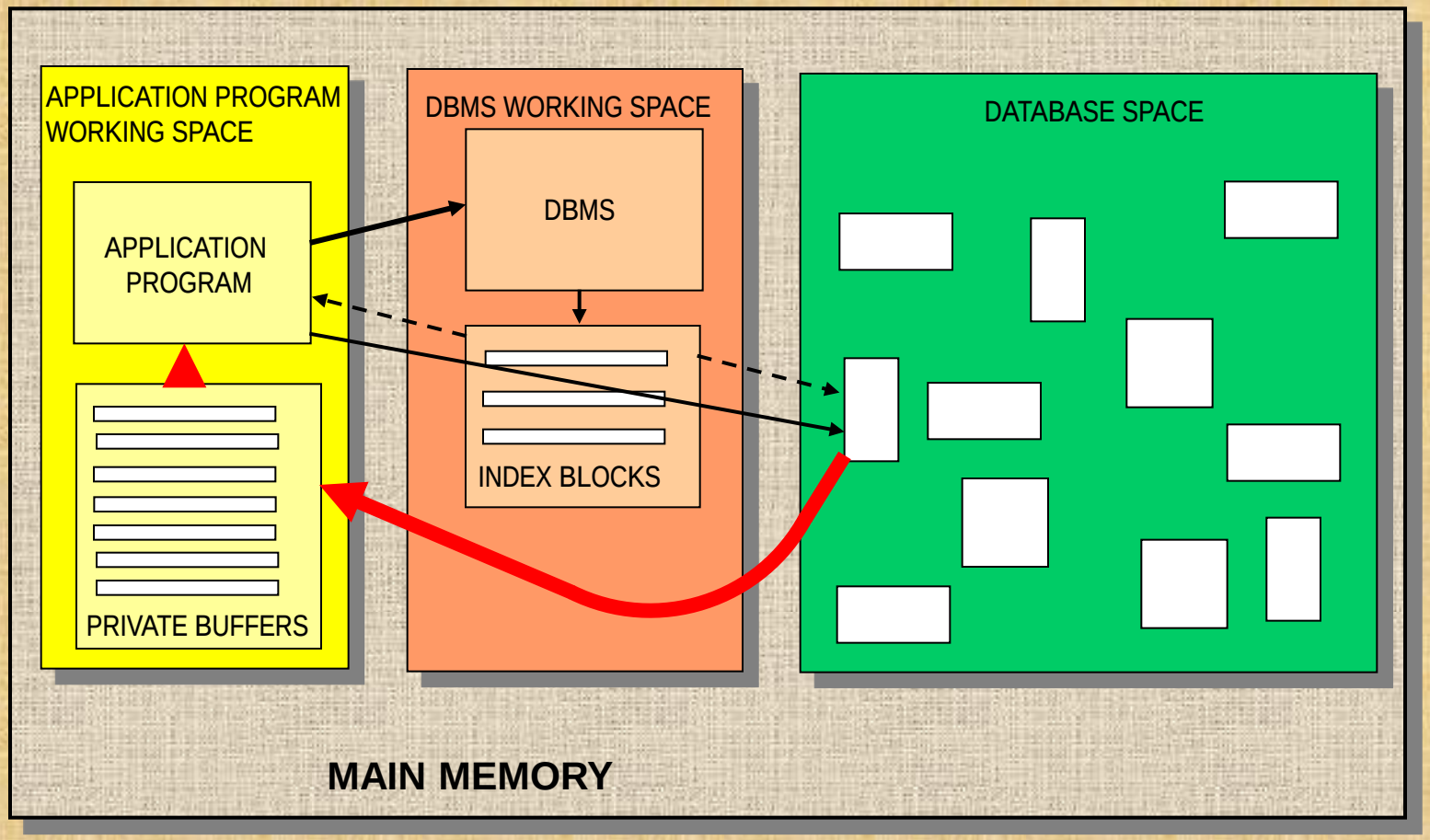
- **CACHED DATA ARE ACCESSED THROUGH INDEXES DESIGNED FOR DISK ACCESS**
- **ACCESS IS MADE THROUGH A BUFFER MANAGER WHICH, GIVEN THE DISK ADDRESS, CHEKS IF THE RELEVANT BLOCK IS IN MM-CACHE AND THEN COPIES IT TO THE MM APPLICATION WORKING AREA**

IN MMDB DATA ARE ACCESSED BY DIRECTLY REFERRING TO THEIR MEMORY ADDRESS

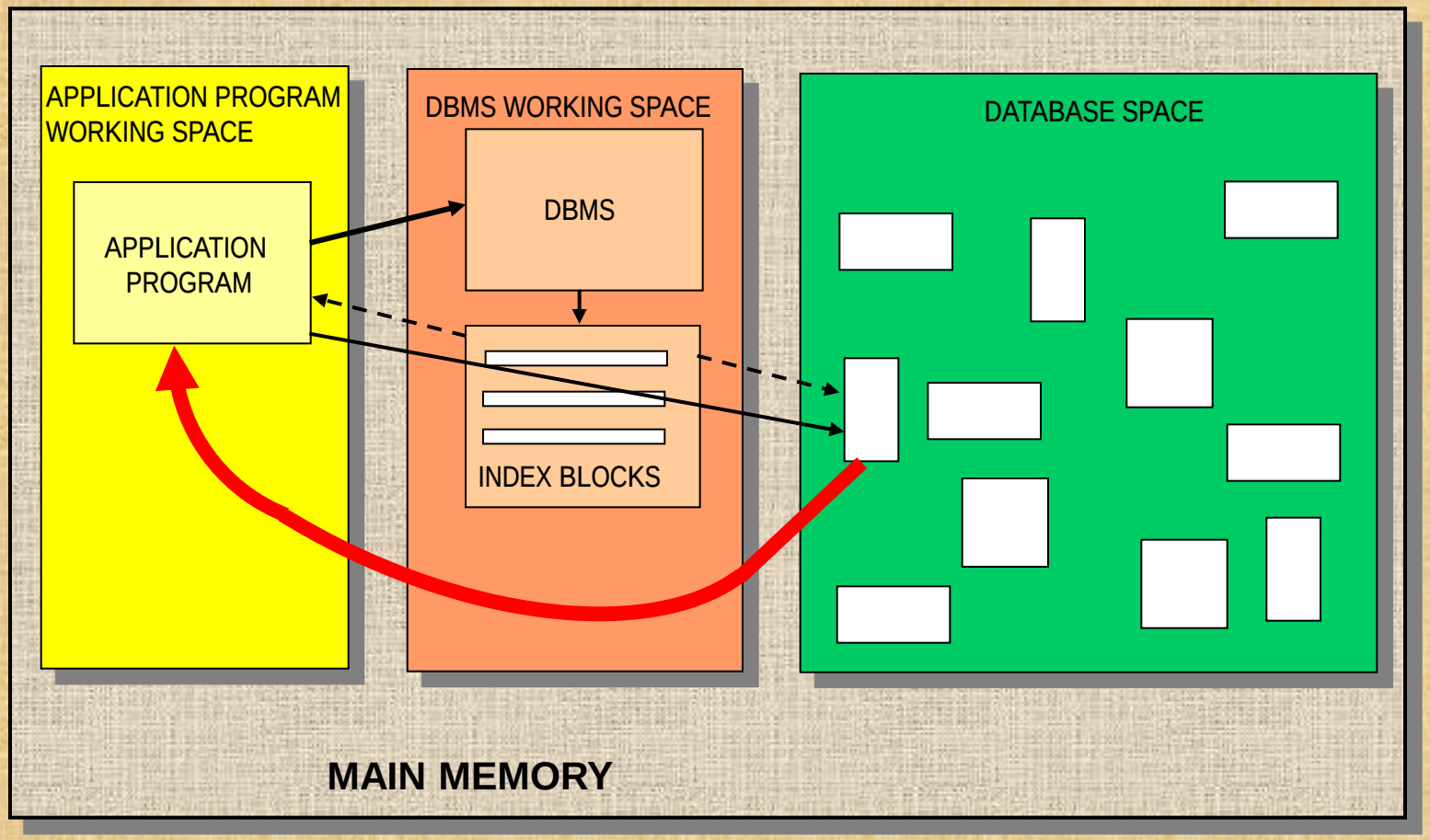
APPLICATION PROGRAM INTERFACE FOR DRDB



APPLICATION PROGRAM INTERFACE FOR MMDB (1)



APPLICATION PROGRAM INTERFACE FOR MMDB (2)



HYBRID MM-DR DATABASE SYSTEMS

- **SOME DB ARE SO LARGE THEY WILL NEVER FIT IN MM**
- **DATA CAN BELONG TO DIFFERENT CLASSES**
  - **HOT: FREQUENTLY ACCESSED, LOW VOLUME, TIMING SENSITIVE (e.g. bank account records)**
  - **COLD: RARELY ACCESSED, VOLUMINOUS, NON TIME CRITICAL (e.g. bank customers records , historical records)**
- **HAVE A COLLECTION OF DATABASES SOME MM OTHERS DR**
- **OBJECTS CAN MIGRATE AMONG THE DBMS, CHANGING THEIR STRUCTURE ACCORDINGLY (e.g. IBM IMS Fast Path)**

ISSUES IN A MMDB

- CONCURRENCY CONTROL
- COMMIT PROCESSING
- DATA REPRESENTATION
- ACCESS METHODS
- QUERY PROCESSING
- RECOVERY
- OBJECTS MIGRATION

MMDBMS CONCURRENCY CONTROL

- **LOCK DURATION IS SHORT**
  - **REDUCED CONTENTION**
  - **LARGE GRANULES (UP TO THE ENTIRE DATABASE)**

- **SERIAL TRANSACTION PROCESSING**
  - **ALMOST ELIMINATES THE NEED OF CC**
  - **HIGHLY REDUCE CACHE FLUSHES**
- **CC STILL NECESSARY WHEN**
  - **MIXED LENGTH TRANSACTIONS COEXIST**
  - **A MULTIPROCESSOR SYSTEM SHARES THE DB AMONG THE DIFFERENT UNITS**

MMDBMS CONCURRENCY CONTROL

TRADITIONAL IMPLEMENTATION

- LOCK (HASH) TABLES HOLDING ENTRIES FOR CURRENTLY LOCKED OBJECTS
- NO LOCK INFORMATION ATTACHED TO DATA

MAIN MEMORY IMPLEMENTATION

- STUFF SOME BITS OF LOCKING INFORMATION INTO DATA
  - 1ST BIT IS THE X-LOCK SET BIT
  - 2ND BIT IS THE WAITING FOR BIT
  - IF MORE THAN ONE TRANSACTION IS WAITING (RARE), USE THE LOCK TABLE AND THE WAKE-UP PROCEDURE
- T&S INSTRUCTION NEEDED TO AVOID MULTIPLE SETTING

MMDBMS COMMIT PROCESSING

DURABILITY OF A TRANSACTION ASKS FOR A LOG RECORD TO BE WRITTEN INTO STABLE STORAGE BEFORE COMMITTING

LOGGING AFFECTS RESPONSE TIME AND THROUGHPUT

- **WAITS EXIST FOR THE DISK SERVICE**
- **THE LOG FILE IS A BOTTLENECK**

TYPICAL LOG RECORD LENGTH 400 BYTES

- **40 BYTES FOR BEGIN/END**
- **360 BYTES FOR OLD/NEW VALUES**

MMDBMS COMMIT PROCESSING

- STORE THE LOG TAIL (< 100 PAGES) IN A SMALL AMOUNT OF STABLE MM
  - REDUCE RESPONSE TIME
  - DOESN'T AFFECT BOTTLENECKS
- PRECOMMIT TRANSACTIONS RELEASE LOCKS AS SOON AS THE LOG RECORD HAS BEEN WRITTEN EVEN IF NOT YET PROPAGATED TO DISK. COMMIT IS DONE AFTER DISK WRITING
  - DOESN'T AFFECT SERIALISATION BECAUSE THE LOG IS SEQUENTIAL
  - DOESN'T REDUCE RESPONSE TIME
  - ENHANCE CONCURRENCY (RESPONSE TIME OF OTHERS)

MMDBMS COMMIT PROCESSING

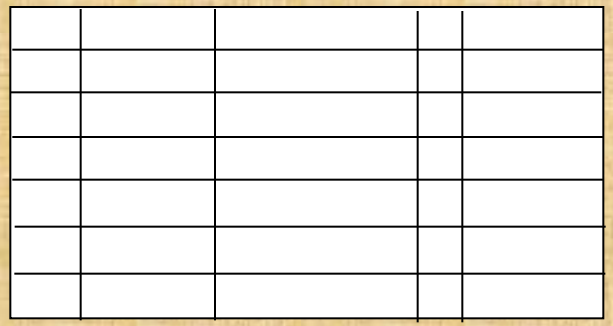
- **GROUP COMMIT ACCUMULATES ENOUGH COMMIT RECORDS TO FILL UP A LOG PAGE AND THEN FLUSHES IT TO DISK**
  - **REDUCES THE TOTAL NUMBER OF DISK ACCESSES**
  - **RELIEVES THE LOG BOTTLENECK**

DATA REPRESENTATION

RELATIONAL DATA ARE USUALLY REPRESENTED AS FLAT FILES (FS)

- **TUPLES ARE STORE SEQUENTIALLY**
- **ATTRIBUTE VALUES ARE EMBEDDED IN THE TUPLES**
- **ACESS IS LOCAL**

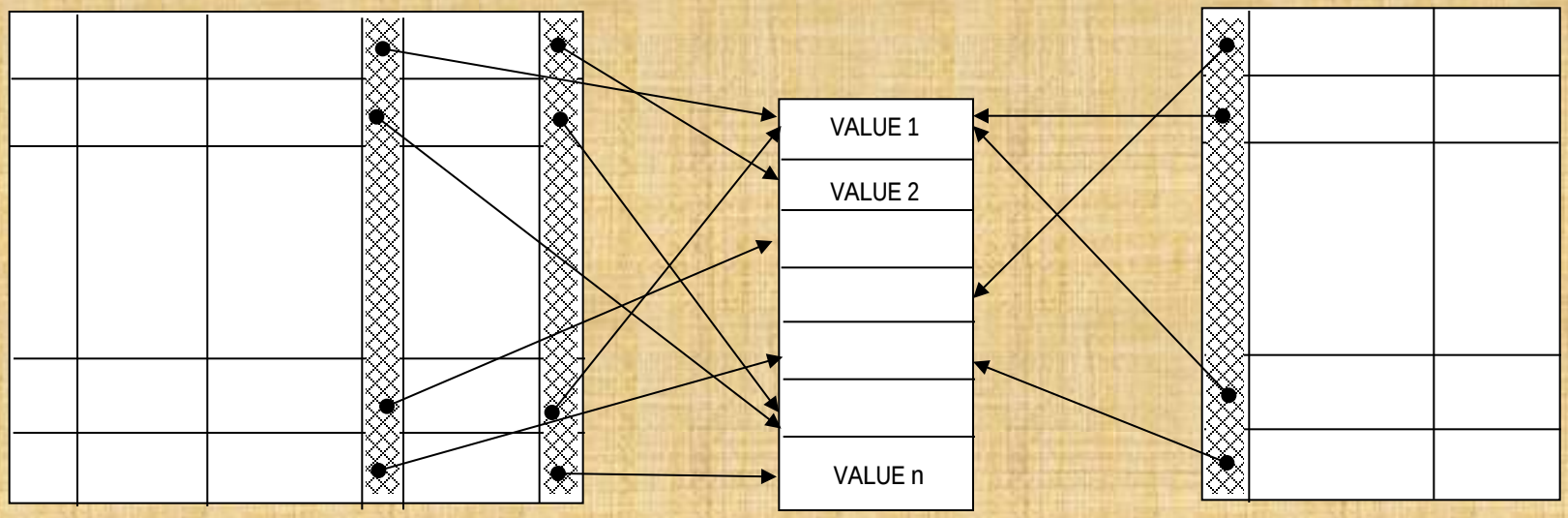
- **SPACE CONSUMING OWING TO DUPLICATE VALUES**
- **INEFFICIENT OWING TO SEQUENTIALITY OF COMPUTATIONS**
- **NEED OF INDEXES**



DATA REPRESENTATION
DOMAIN STORAGE

- ACCESS LOCALITY IS NOT AN ISSUE IN MMDB
- COMPACTNESS IS AN ISSUE FOR BOTH DATA AND INDEXES
- PRECLUDE VALUE DUPLICATION BY GROUPING VALUES IN DOMAINS (DS)
  - 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
  - FIXED SIZE TUPLES

DATA REPRESENTATION
DOMAIN STORAGE



MMDB ACCESS METHODS

GOALS

- **DISK ORIENTED STRUCTURES**
  - **MINIMISE DISK ACCESSES**
  - **MINIMISE STORAGE SPACE**
- **MAIN MEMORY STRUCTURES**
  - **REDUCE OVERALL COMPUTATION TIME**
  - **USE AS LITTLE MEMORY AS POSSIBLE**

ONLY POINTERS TO DATA CAN BE STORED IN THE INDEXING STRUCTURES AND NOT THE DATA VALUES THEMSELVES.

MMDB ACCESS METHODS

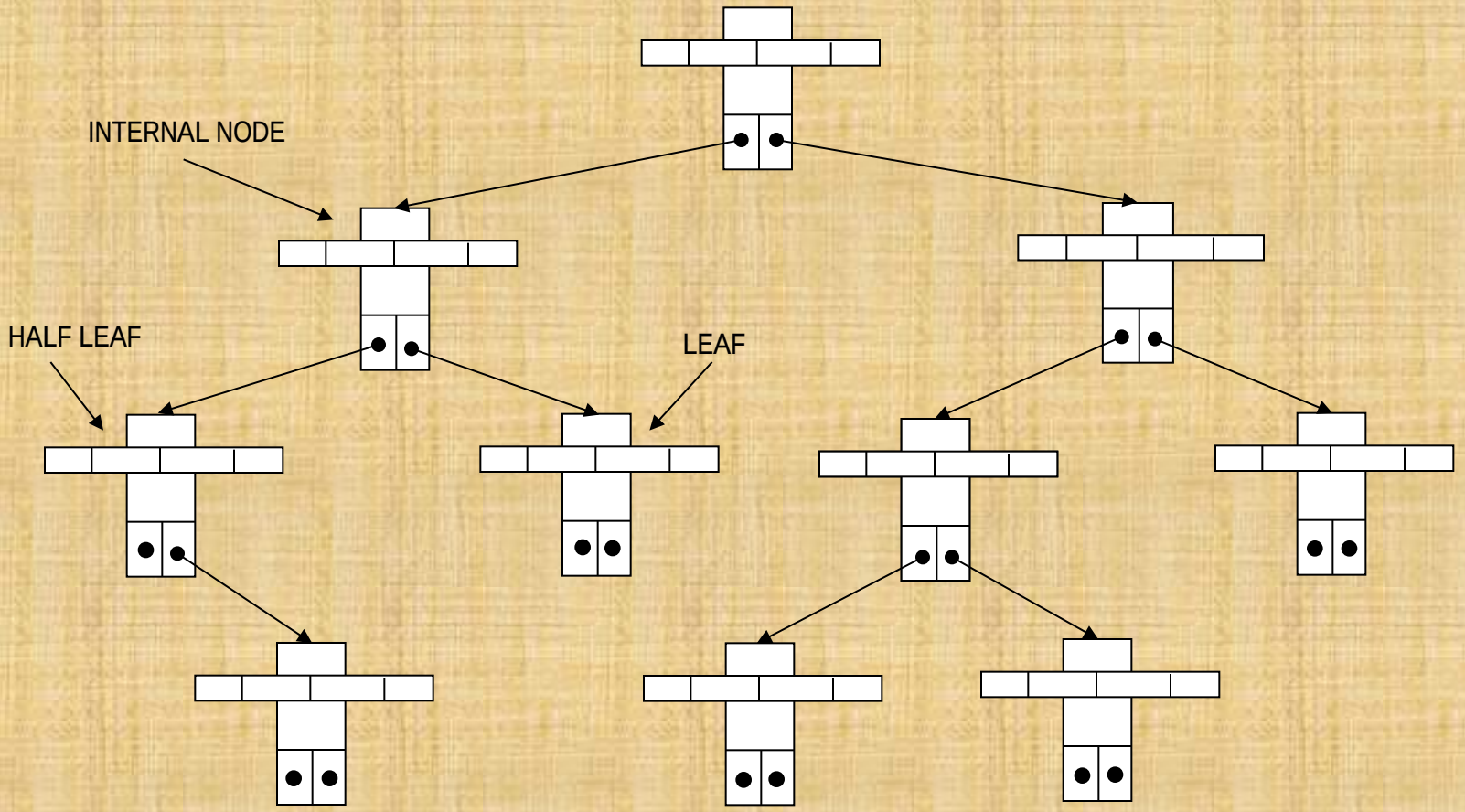
- **HASHING**
  - **FAST LOOKUP AND UPDATING**
  - **NOT SPACE EFFICIENT**
  - **DOESN'T SUPPORT RANGE QUERIES**
- **TREE INDEXING**
  - **WITH A SINGLE POINTER GET ACCESS BOTH TO AN ATTRIBUTE VALUE AND TO THE ENTIRE TUPLE**
  - **POINTERS ARE FIXED (SHORT) LENGTH**
  - **SUITED FOR RANGE QUERIES**

MMDB ACCESS METHODS
THE T-Tree

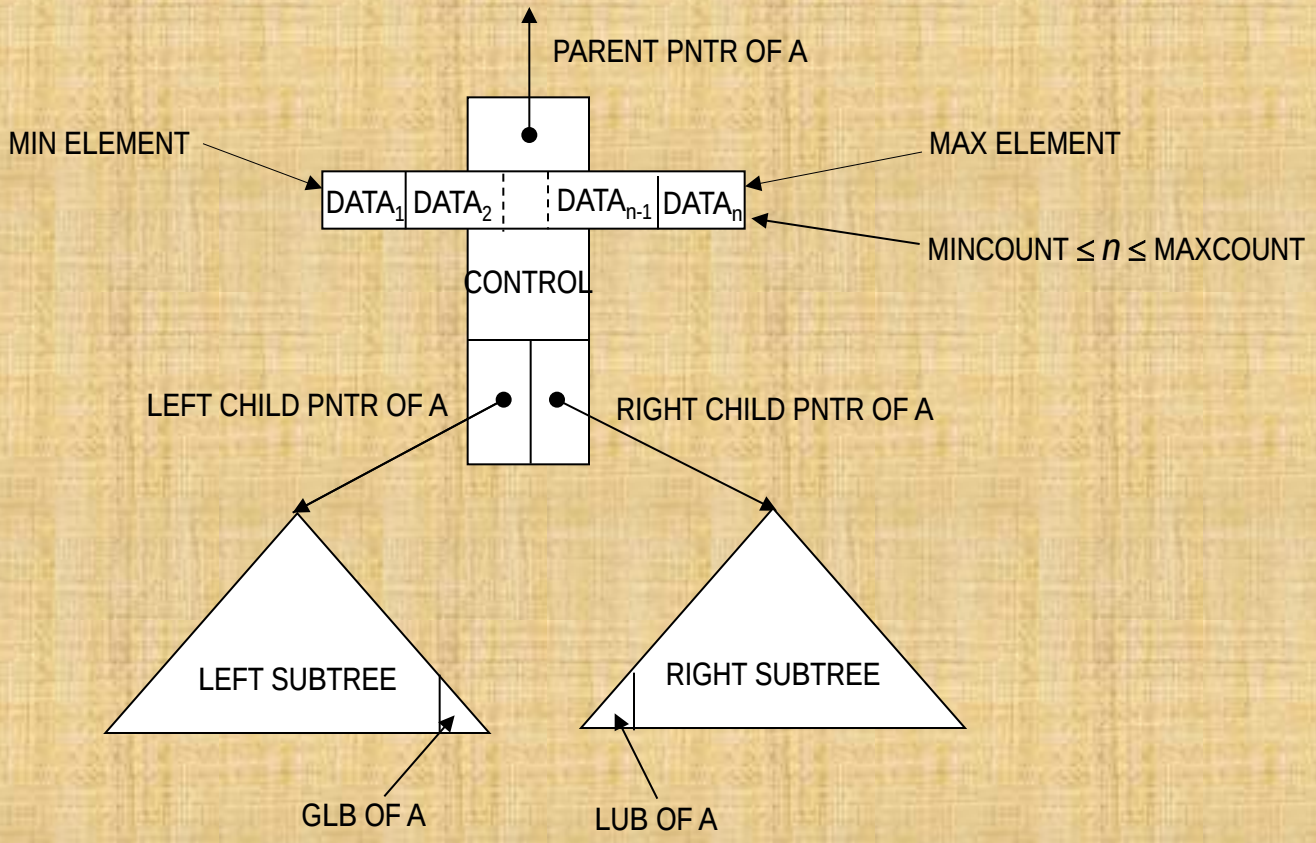
THE T-Tree IS A DATA STRUCTURE WHOSE ANCESTORS ARE B-Trees AND AVL-Trees

- **IT IS BINARY LIKE AVL-Trees**
  - **SEARCH IS ESSENTIALLY BINARY**
- **A T-Node CONTAINS MANY ELEMENTS LIKE B-Trees**
  - **STORAGE AND UPDATE EFFICIENCY**
- **INSERTIONS AND DELETIONS USUALLY MOVE DATA WITHIN A SINGLE NODE (like IN B-Trees)**
- **REBALACING IS DONE BY NODE ROTATION (like in AVL Trees) BUT IS MUCH LESS FREQUENT**

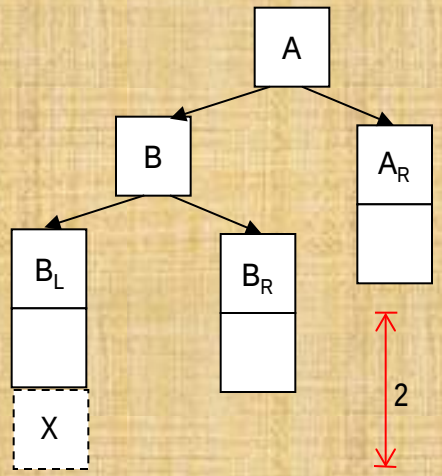
THE T-Tree



T-Tree NODE STRUCTURE

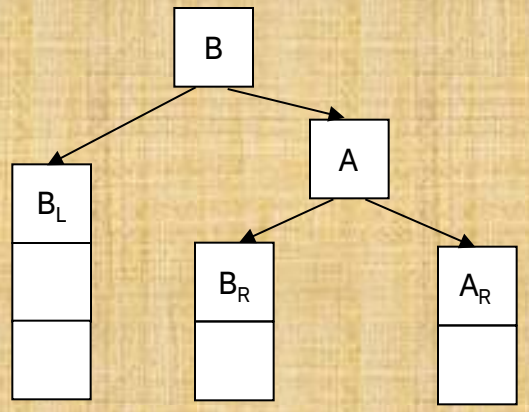


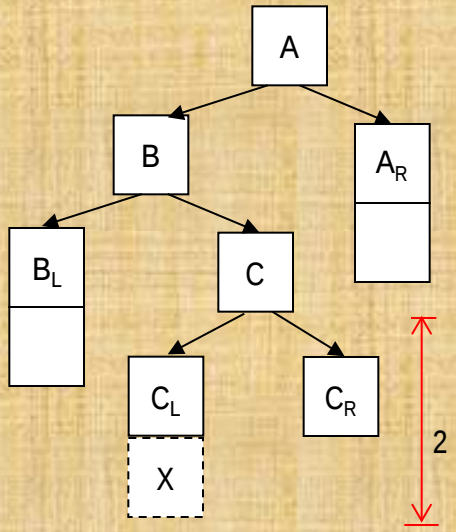
T-Tree REBALANCING



LL (RR) INSERTION

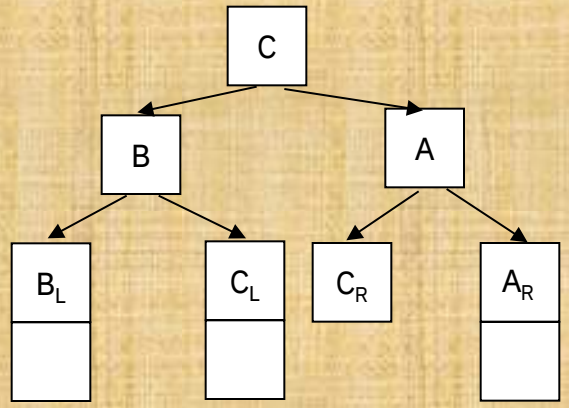
SINGLE ROTATION



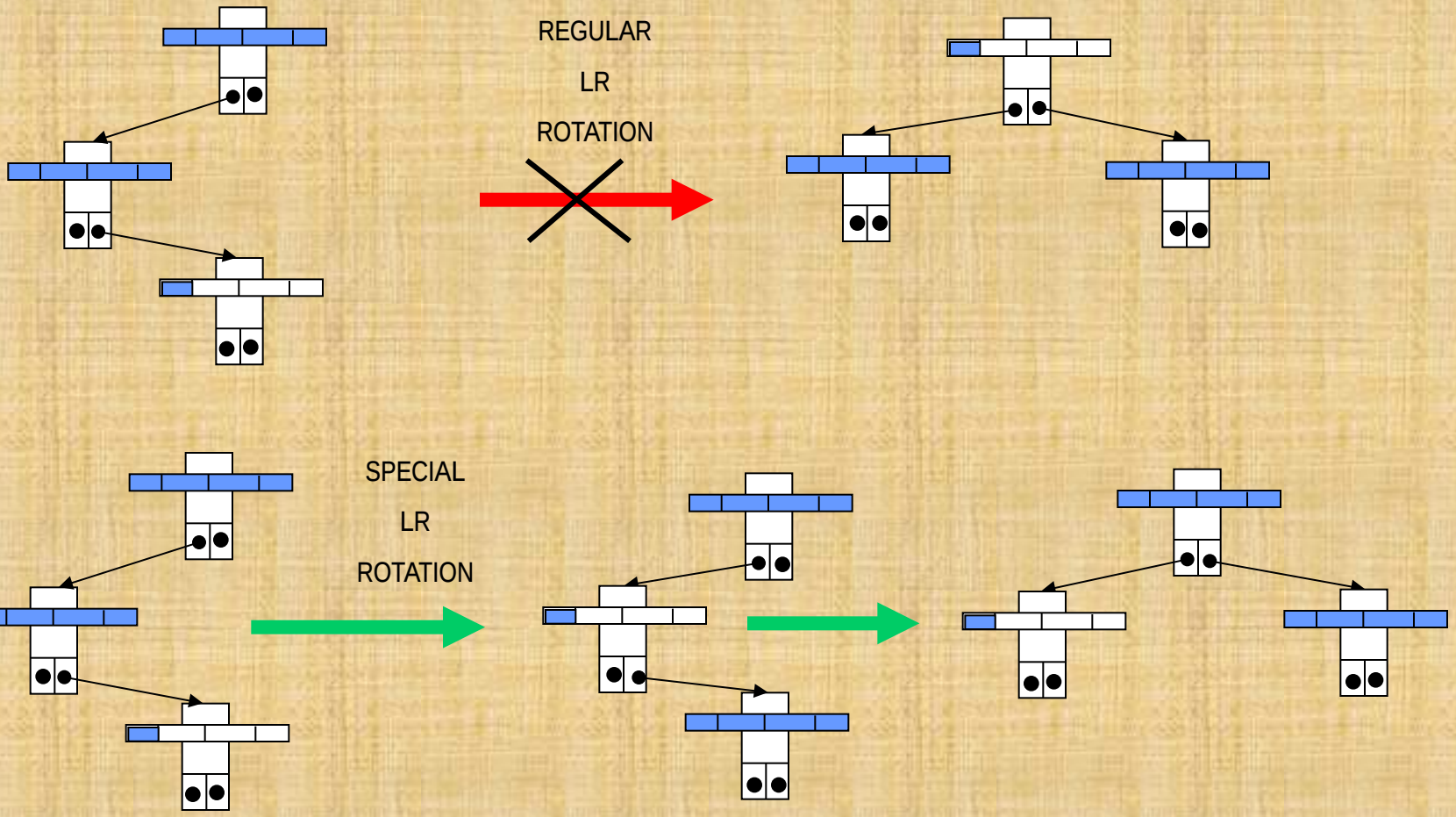


LR (RL) INSERTION

DOUBLE ROTATION



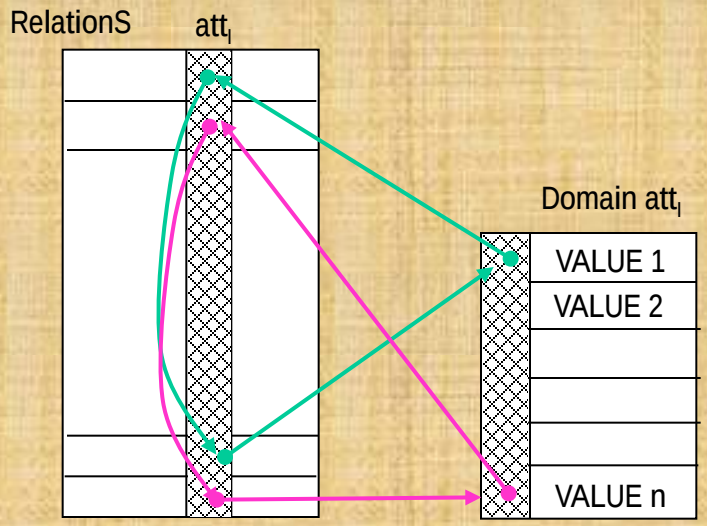
T-Tree REBALANCING



INDEXING WITH DOMAIN STORAGE
RING STORAGE

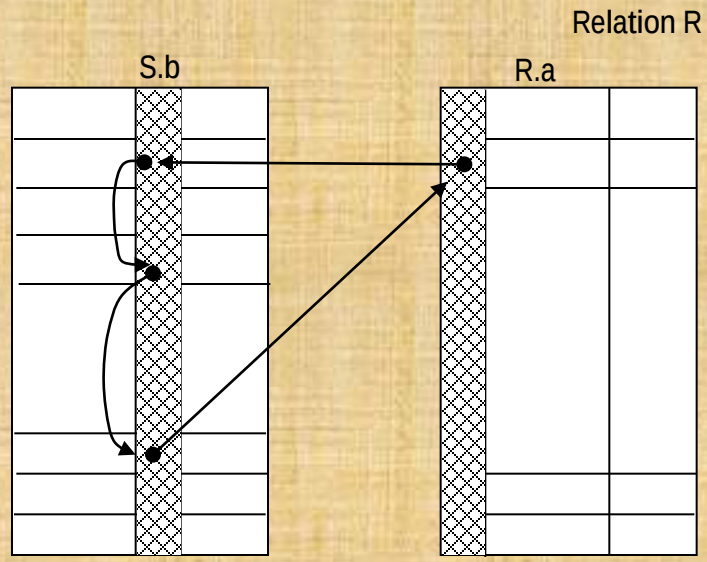
RING SELECT INDEX

VALUE-TO -TUPLE / TUPLE-TO-VALUE



BIDIRECTIONAL RING JOIN INDEX

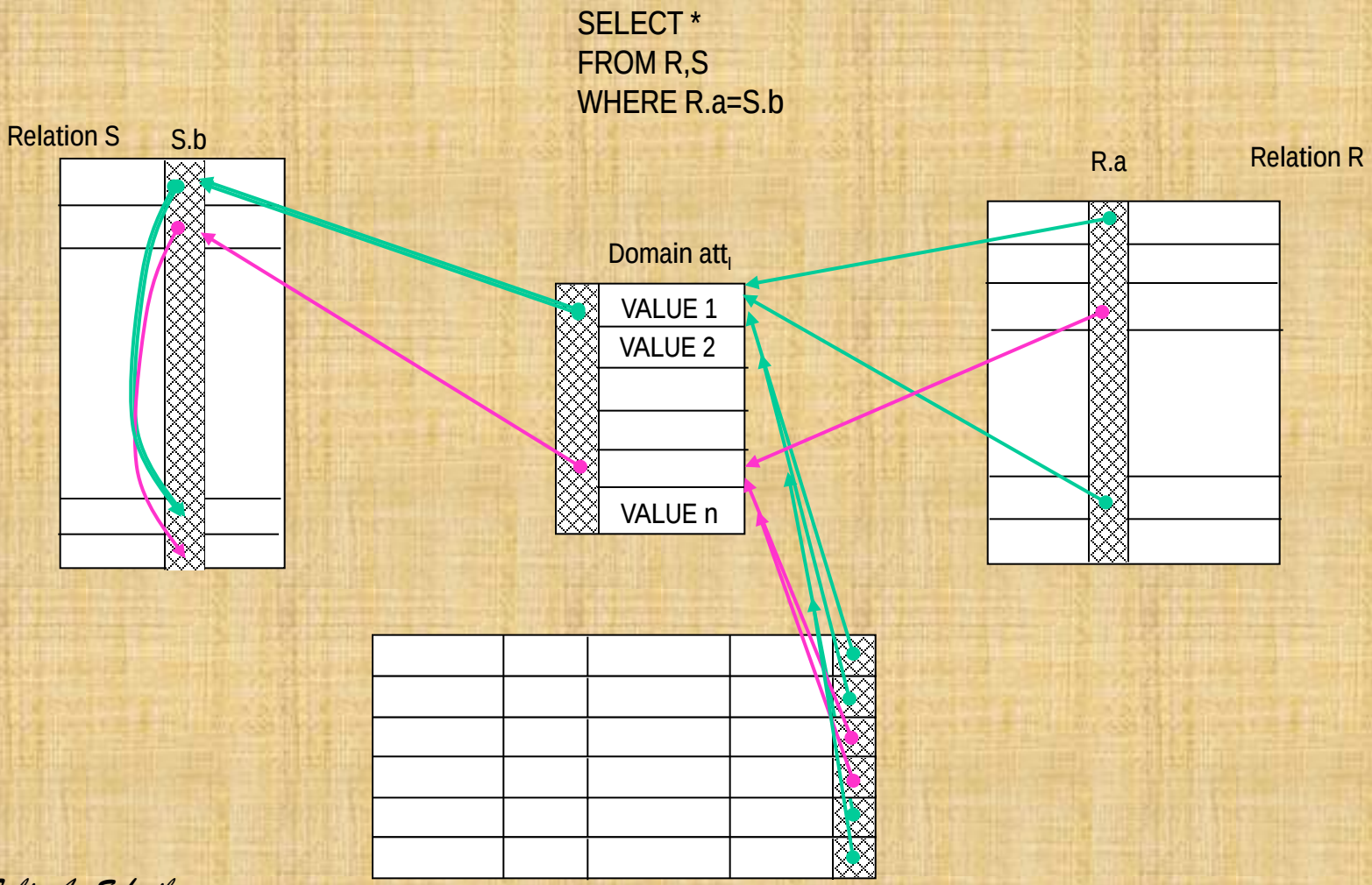
ON A FOREIGN KEY (R.a=S.b)



QUERY PROCESSING

- QUERY PROCESSORS FOR DRDB FOCUS ON REDUCING DISK ACCESS COSTS
- QUERY PROCESSORS FOR MMDB MUST FOCUS ON PROCESSING COSTS
  - OPERATION COSTS VARY FROM SYSTEM TO SYSTEM
  - NO GENERAL OPTIMISATION TECHNIQUE
- IMPLEMENTATION OF RELATIONAL OPERATORS SHOULD BENEFIT OF MM DATA AND INDEX REPRESENTATION
  - NESTED-LOOP JOIN PREFERRED TO SORT-MERGE JOIN

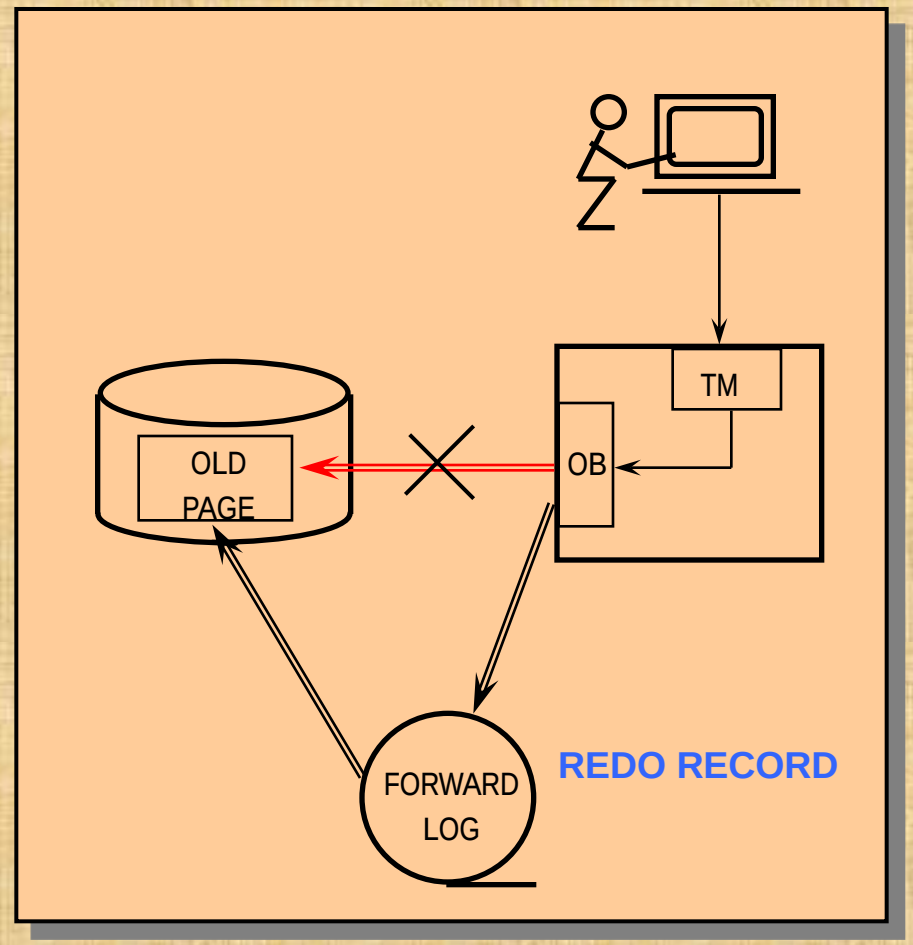
NESTED-LOOP JOIN WITH RING INDEX

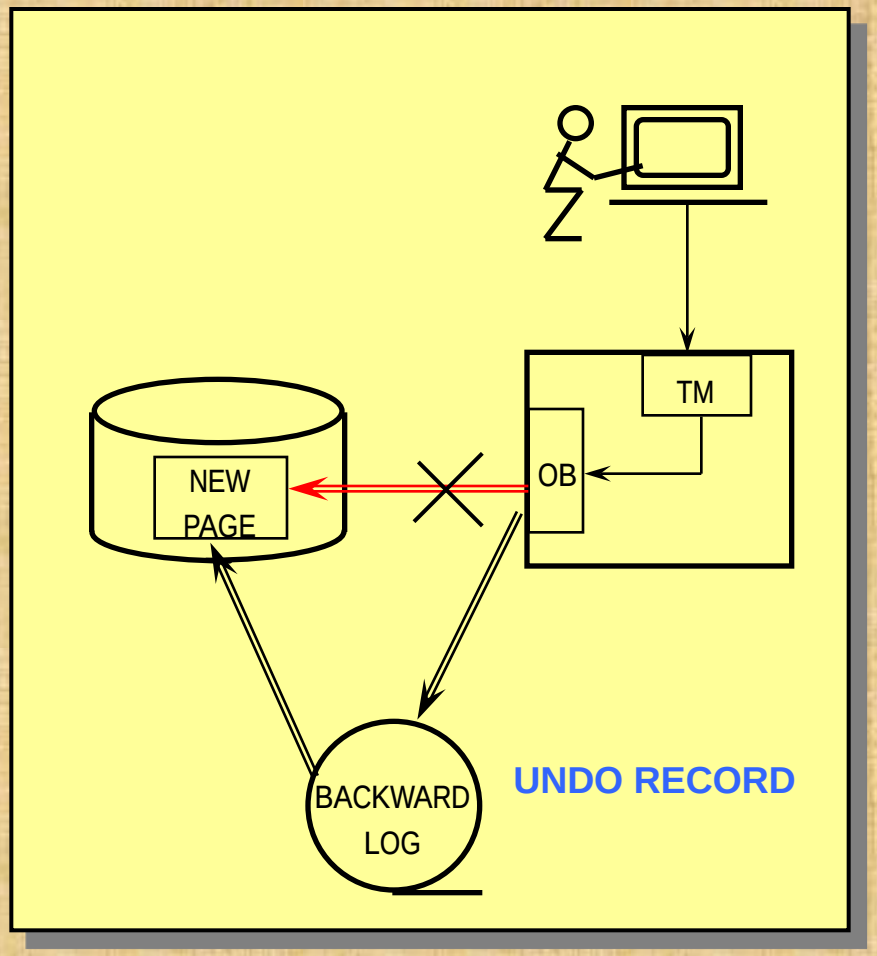


BACKUP AND RECOVERY

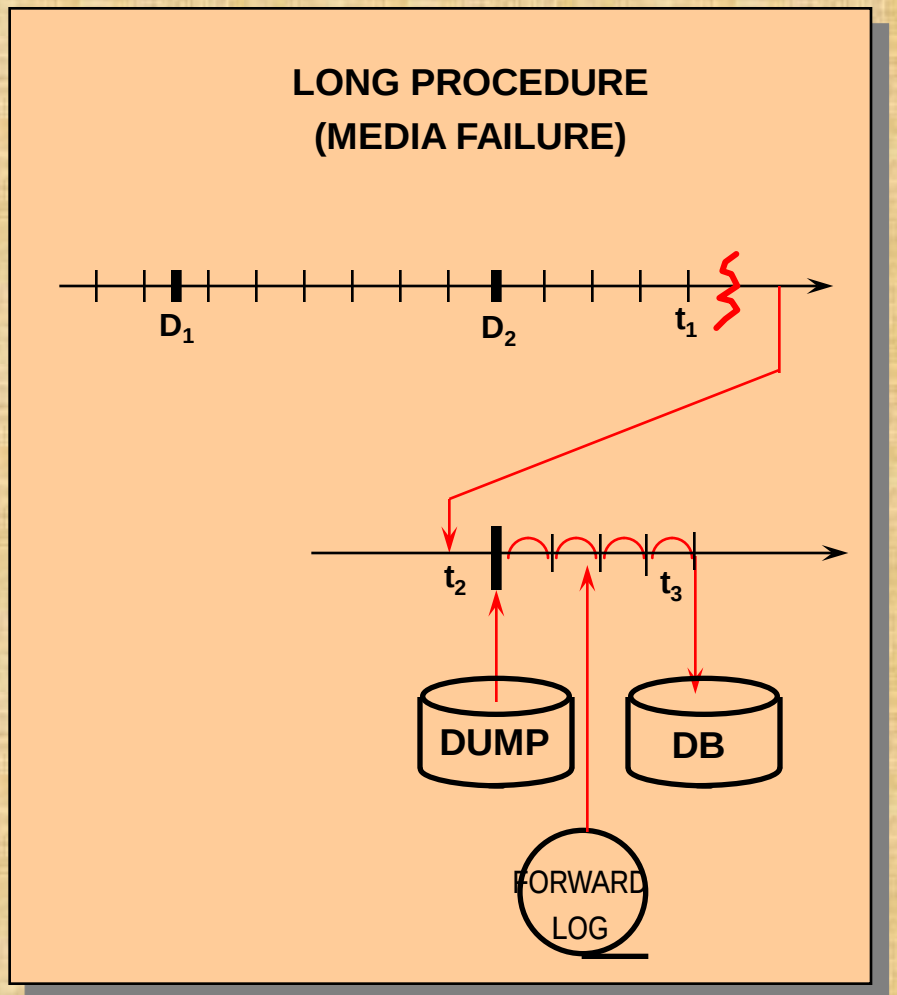
- PERFORM BACKUPS OR CHECKPOINTS TO A DISK DURING NORMAL OPERATION
  - LOG AS MUCH INFORMATION AS POSSIBLE TO PERFORM A FULL AND CONSISTENT RECOVERY
  - KEEP THE OVERHEAD AS SMALL AS POSSIBLE
- RECOVER FROM FAILURES
  - AS FAST AS POSSIBLE

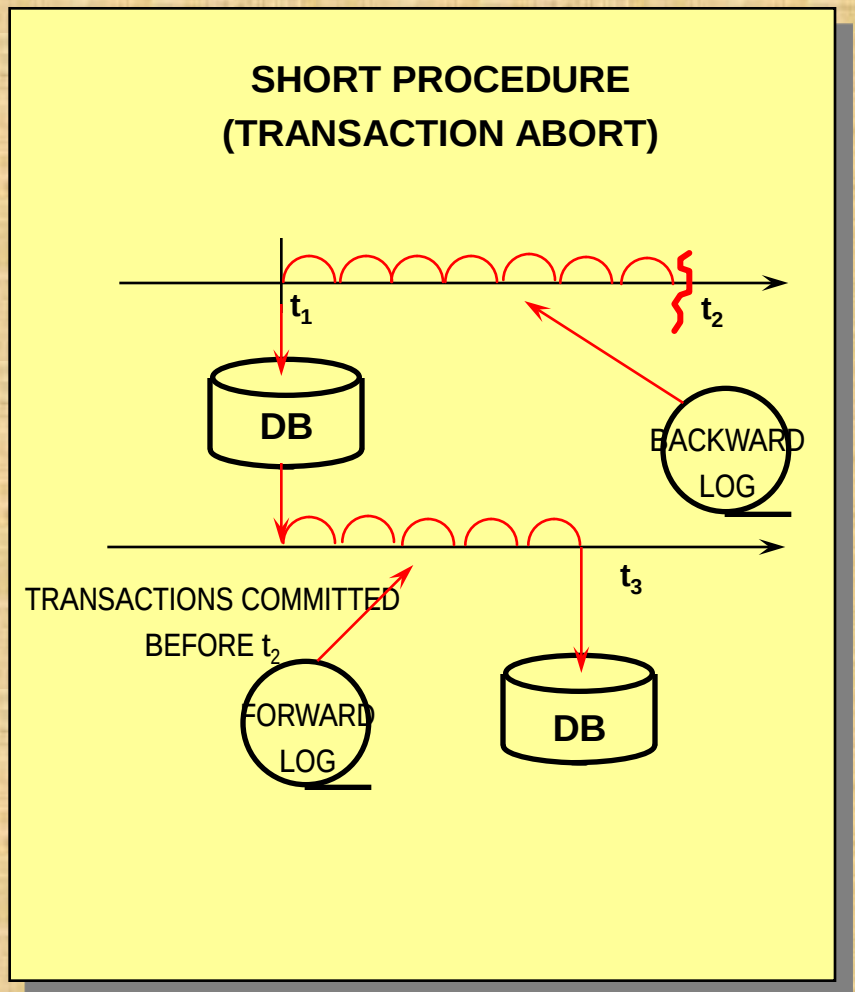
LOGGING TECHNIQUES FOR DRDB





RECOVERY PROCEDURES FOR DRDB





BACKUP AND RECOVERY FOR MMDB

- **PERFORM BACKUPS AND CHECKPOINTS TO A DISK DURING NORMAL OPERATION**
  - **USE VERY LARGE BLOCK SIZES TO ENHANCE EFFICIENCY**
  - **TRANSACTION CONSISTENT OR ACTION-CONSISTENT CHECKPOINTS REQUIRE SYNCHRONIZATION WITH TRANSACTIONS**
- **RECOVER FROM FAILURES**
  - **TRANSFER FROM DISK TAKES A LONG TIME**
    - **TRANSFER BLOCKS ON DEMAND**
    - **USE DISK ARRAYS TO WORK IN PARALLEL**

OBJECT MIGRATION

- IN DRDB RECORDS FROM DIFFERENT RELATIONS ARE OFTEN CLUSTERED IN THE SAME DISK PAGES TO ENHANCE PERFORMANCE

- IN MMDB NO SUCH NEED EXIST
  - TUPLES HAVE OFTEN ONLY POINTERS TO DOMAIN VALUES FOR THE ATTRIBUTES

- WHEN MIGRATION SHOULD OCCUR (e.g. in hybrid systems) DYNAMIC CLUSTERING IS TO BE MADE AND INDEXES REBUILT ACCORDINGLY

BIBLIOGRAPHY

- AA.VV. – Special Issue on Main-Memory Database Systems - IEEE Data Engineering, Vol. 36, n. 2, June 2013
- C. Bobineau et Al. - PicoDBMS: Scaling Down Database Techniques for the Smartcard - Proc. 26th Int. VLDB Conf., 2000, pp.
- D.J. DeWitt et Al. - Implementation techniques for main memory database systems - Proc. *ACM SIGMOD* Conf., June 1984, pp. 1-8
- H. Garcia-Molina, K. Salem - Main Memory Database Systems: An Overview - *IEEE- Transactions KDE*, Vol. 4, n. 6, 1992, pp. 509-516
- T.J. Lehman, M.J. Carey - A Study of Index Structures for Main Memory Database Management Systems - Proc. 12th Int. VLDB Conf., August 1986, pp.294-303