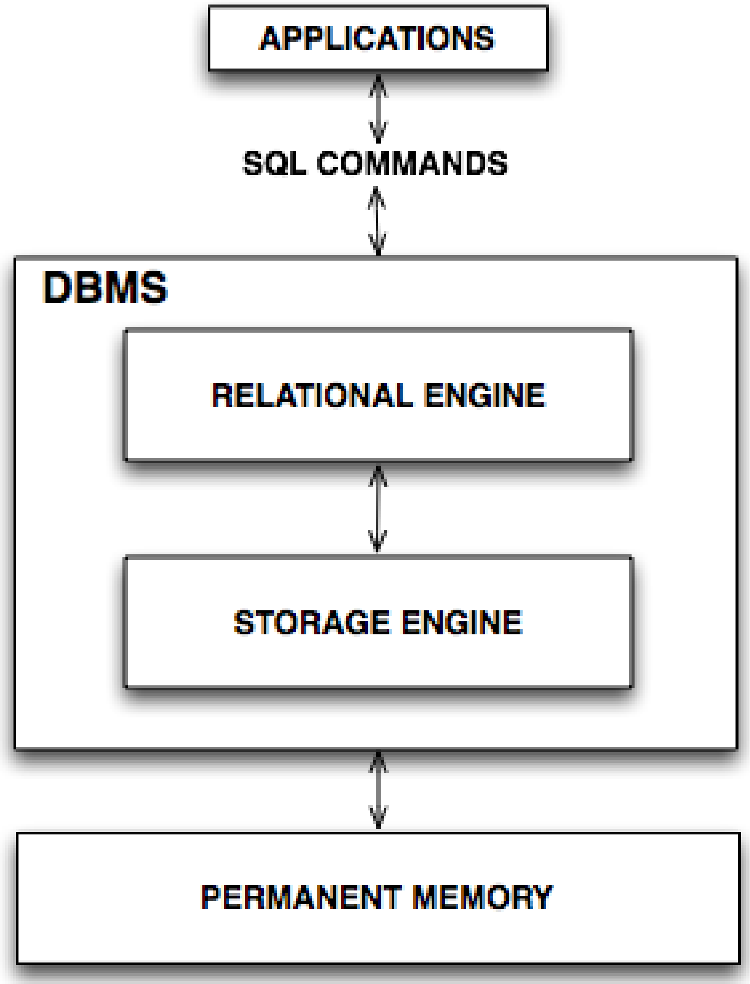
Advanced Database Systems

DBMS Internals

- Data structures and algorithms to implement RDBMS
- Internals of non relational data management systems



Why to take this course?

- To understand the strengths and weaknesses of commercial (ORACLE, DB2, SQLServer,...) and public domain (MySQL, PostgreSQL) DBMS
- To make you a better application designer, or database administrator, or database programmer
- To prepare you as implementer of data intensive systems

Course topics

- Architecture of a DBMS
- Permanent Memory Data Structures
- Query Processing and Optimization
- Transaction Management
- Database Physical Design and Database Tuning
- Internals of NoSQL systems

Before taking this course

- Before taking this course you should be comfortable with
  - Logics (set theory, first order logic, De Morgan rules)
  - Operating systems (persistent memories, buffers, concurrency)
  - Algorithms (hashing, balanced trees)
  - Functional dependencies (normalization)
  - Relational algebra
  - SQL querying

Course material and exams

- Course web site. Lecture Notes: Relational DBMS Internals
- http://www.di.unipi.it/~ghelli/bd2/bd2.eng.html
- Exams:
  - Written and oral exams, in the same session
  - Two optional mid-term tests (*compitini*)
  - Each of them, separately, can be used to forfeit half of the written test, until August
- Office hours: https://www.di.unipi.it/it/didattica/inf-l/commissioni-e-docenti/ricevimento or ask for a date by email
  - (Web site: Education / Master Programme … / Committees and Faculty / Faculty and office hours)

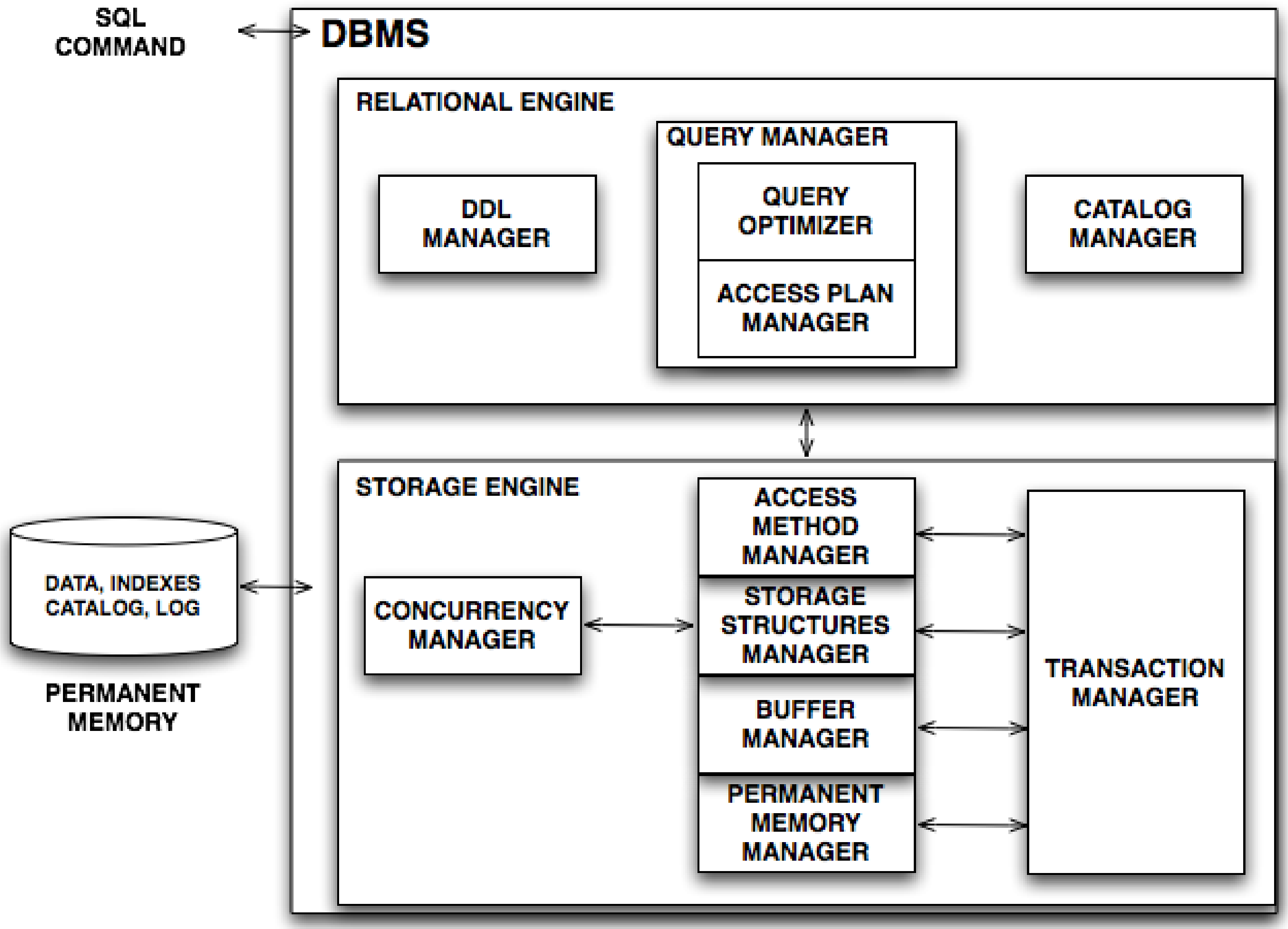
JRS

- A system write and execute SQL queries and access plans
- http://www.di.unipi.it/~albano/JRS/toStart.html

How to use the lecture notes

- Use them. Slides are not enough
- Start reading them now
- If you do not understand anything ask me

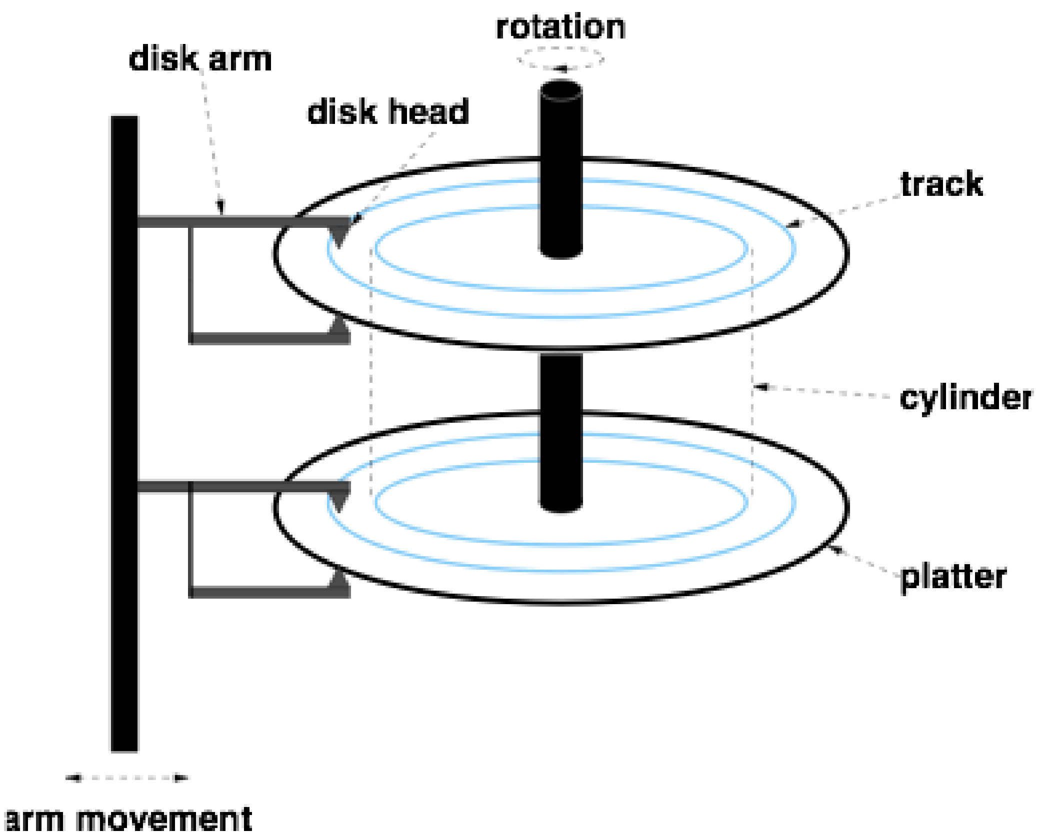
DBMS Architecture



Why not just main memory

- Costs too much. For $1000 the market offers (Jan 2015):
  - ~120 GB of RAM
  - ~3 TB of Solid State Disk (Flash)
  - ~30 TB of Magnetic Disk
- Main memory is volatile

Disks – survival of the mecha-saurs

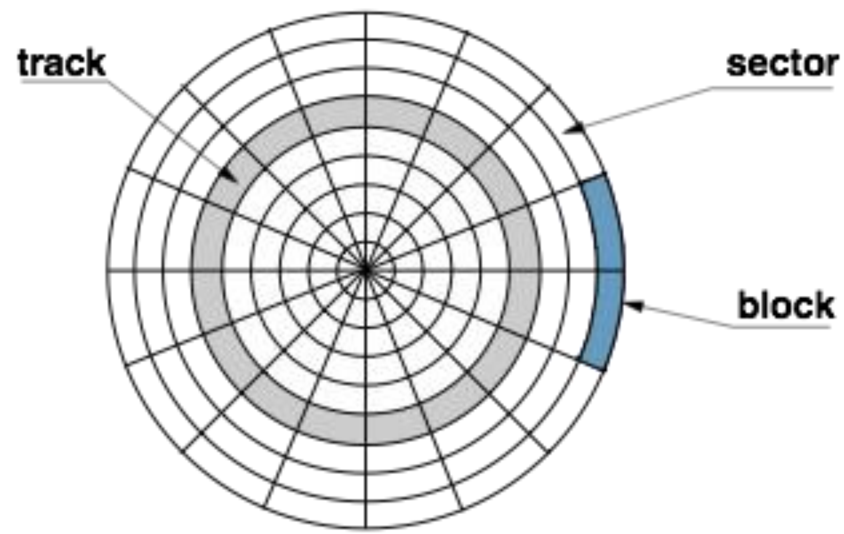


Access Time =

Seek Time (5-20 ms) +

Rotational Delay (0-5 ms) +

Transfer time (.01 ms per 8K)



Evolution of technology

- Disk capacity increases each year of the ~ 50%
- Transfer time decreases each year of the ~ 50%
- Seek time and rotational delay decrease very slowly (~ 10%)

Improving performance: RAID

- RAID: Redundant Array of Independent Disks
  - RAID 0 (striping without parity): performance
  - RAID 1 (mirroring without parity): fault tolerance
  - RAID 5 (striping with distributed parity): performance and tolerance
  - RAID 6: more robust than RAID 5

Persistent memory: flash

Characteristics of the three types of memory

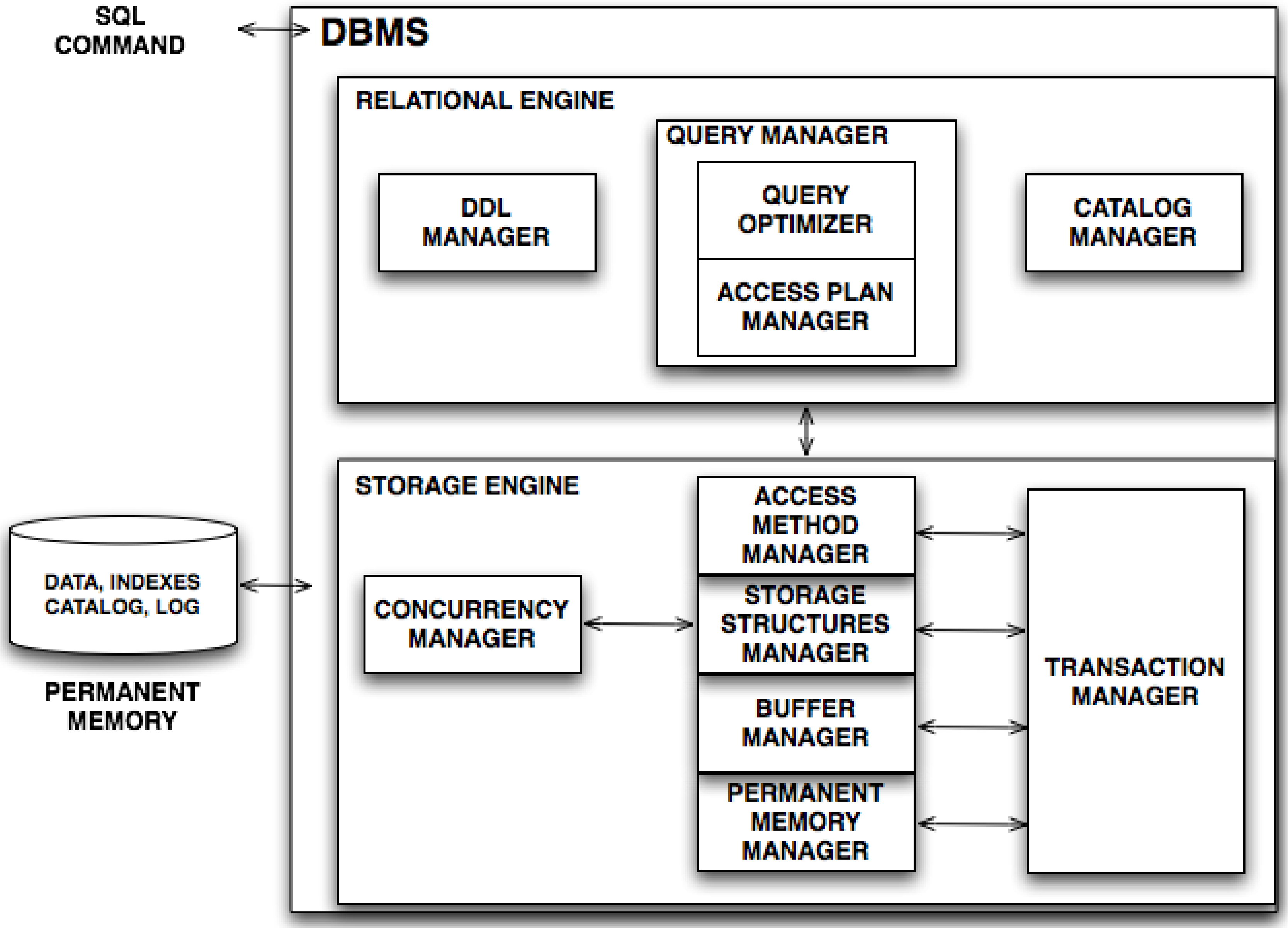
- This table should NOT be taken too seriously

Memory	Read	Write	Erase
HD	12.7 ms (2KB)	13.7 ms (2 KB)	
NAND Flash	80 µs (2 KB)	200 µs (2 KB)	1.5 ms (128 KB)
RAM	22.5 ns	22.5 ns	

Characteristics of the three types of memory

- Seek time:
  - Flash, RAM: little or no seek time
  - Disk: huge
- Transfer rate (do not take this too seriously, depends on MANY things):
  - RAM: 6 Gb/sec
  - Flash: 1 Gb/sec
  - Disc: 140 Mb/sec
- I/O Time Disk = 100 x Flash = 100 000 RAM
- Flash memory operations: Read, Write, Erase
- Capacity and costs quite different
- Lifetime: disk 10 years, Flash: 100 K cycles E/W

DBMS Architecture



Permanent memory manager

- The PMM gives an abstraction of permanent memory as a set of databases, each of them as a set of logical files of physical pages (or blocks), linearly addressed, hiding:
  - The disk characteristics (“disk geometry”)
  - The operating system
- Each file can grow dynamically (but the physical contiguity cannot be assured)
- Each relation (and index) of a database is stored in a logical file

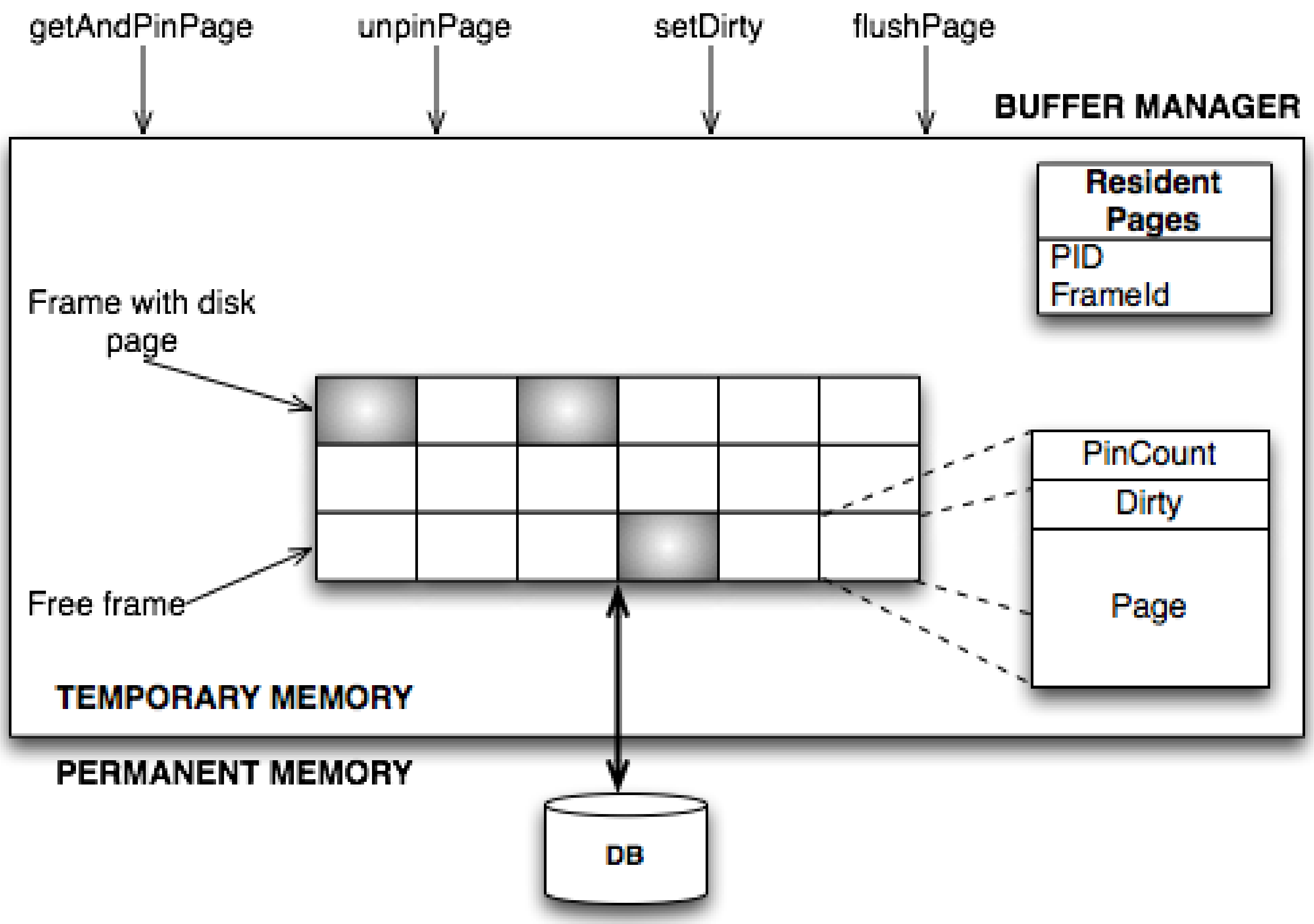
Permanent Memory Manager

- JRS Interface:
  - GM_createDB: Path, DbName -> null (GM_destroyDB)
  - GM_createFile: Path, DbName, FileName -> null (GM_destroyFile)
  - GM_openFile: Path, DbName, FileName -> FileIde (GM_closeFile)
  - GM_newBlock: FileIde, string -> PID
    - PID = (FileIde, NumBlock)
  - GM_readBlock: PID -> string
  - GM_writeBlock: PID, string -> null

Buffer Manager

- It manages the transfer of pages between temporary and permanent memory
- gives the abstraction of permanent memory as a set of pages that can be used in temporary memory
- Buffer Interface (partial)
  - GB_getAndPinPage: PID -> Page
  - GB_setDirty: PID, bool -> null
  - GB_unpinPage: PID -> null

Buffer manager



Buffer manager

Function
GB_getAndPinPage(p):
IF buffer contains **p**
THEN (pinCount(**p**) := pinCount(**p**) + 1
    RETURN address of frame with **p**);
ELSE
  select a frame with page **p'** to be replaced
  IF dirty(**p'**) **THEN** **GM_writeBlock**(**p'**);
  p' := **GM_readBlock(p)**,
  pinCount(**p'**) := 1; dirty(**p'**) := false;
  RETURN address of frame with **p'**;

Buffer replacement policy

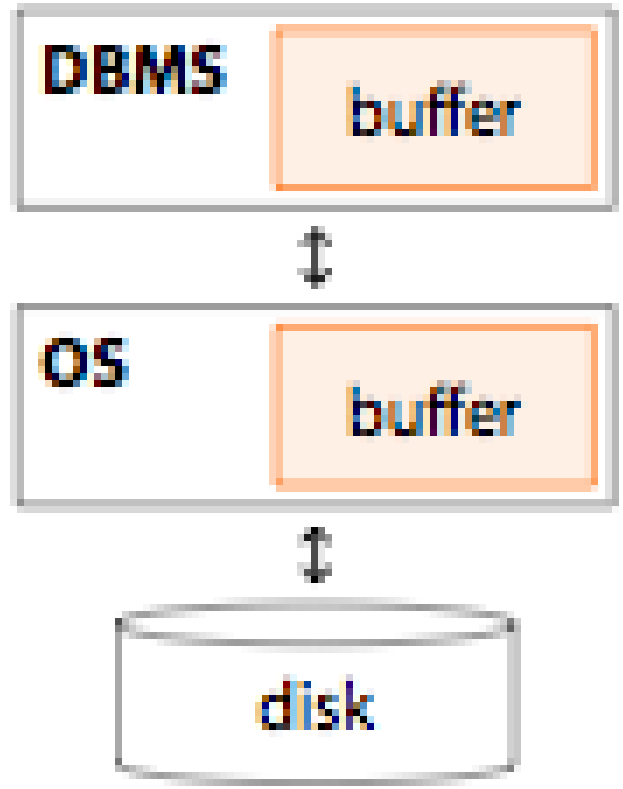
- Very common policy: Least Recently Used (LRU) frame
- Replace the frame which has the earliest unpinned time
- Not always the best:
  - In a join loop, the LRU could be optimal for one table, while for the other the optimal policy is the Most Recently Used (MRU)

Buffer Manager: page release

- What happens when a page p is no longer needed by a transaction?
- If p has not been modified
  - GB_unpinPage(p):
  - pinCount(p) := pinCount(p) - 1 ;
- If p has been modified
  - GB_setDirty(p):dirty(p) := true ;
  - GB_unpinPage(p); ←?
  - GM_writeBlock(p); ←?

Buffer Manager and OS

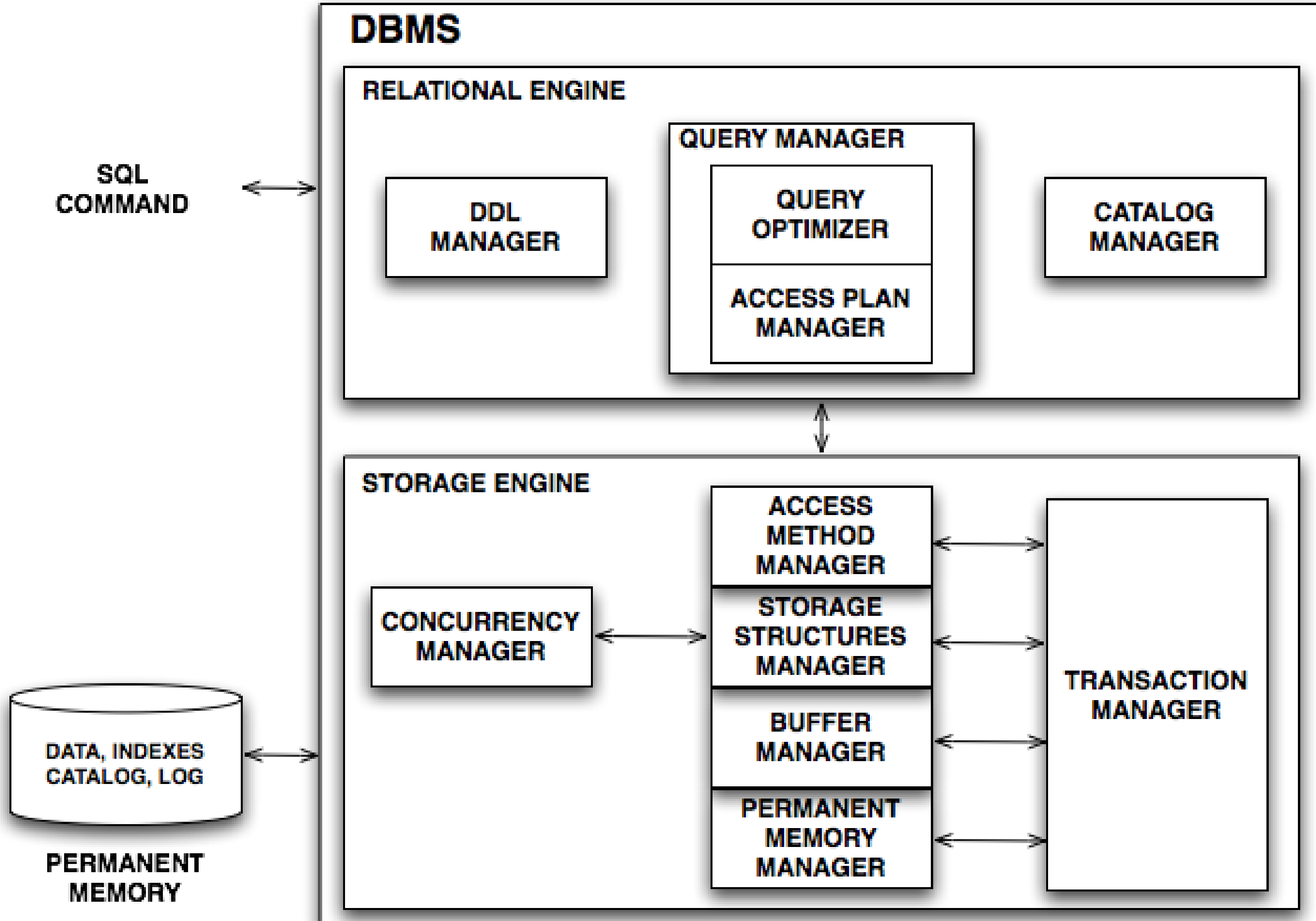
- A disk page is in two buffers
- DBMS try to turn off OS functionality: raw disk access instead of OS files
- May be difficult or impossible



Summary

- Permanent Memory
  - Magnetic disk: cheap, random access, but cost depend on location of page
- Buffer Manager
  - DBMS vs OS VM manager. DBMS need features not found in many OS's, e.g. controlling the order of page writes to disk, forcing page to disk, ability to control pre-fetching and page replacement policy, based on predictable access patterns

Next



DBMS
RELATIONAL ENGINE
SQL
COMMAND
DDL
MANAGER
QUERY MANAGER
QUERY
OPTIMIZER
ACCESS PLAN
MANAGER
CATALOG
MANAGER
STORAGE ENGINE
ACCESS
METHOD
MANAGER
CONCURRENCY
MANAGER
STORAGE
STRUCTURES
MANAGER
BUFFER
MANAGER
PERMANENT
MEMORY
MANAGER
TRANSACTION
MANAGER
DATA, INDEXES
CATALOG, LOG
PERMANENT
MEMORY

Next: storage structures manager

- Data organizations
  - Heap or sequential organizations
  - Primary organizations (hash, tree)
  - Secondary organizations
- Cost model
  - Number of pages (Npag(R))
  - Operations cost (accessed pages):
    - Equality and Range search
    - Update, Insertion, Delete