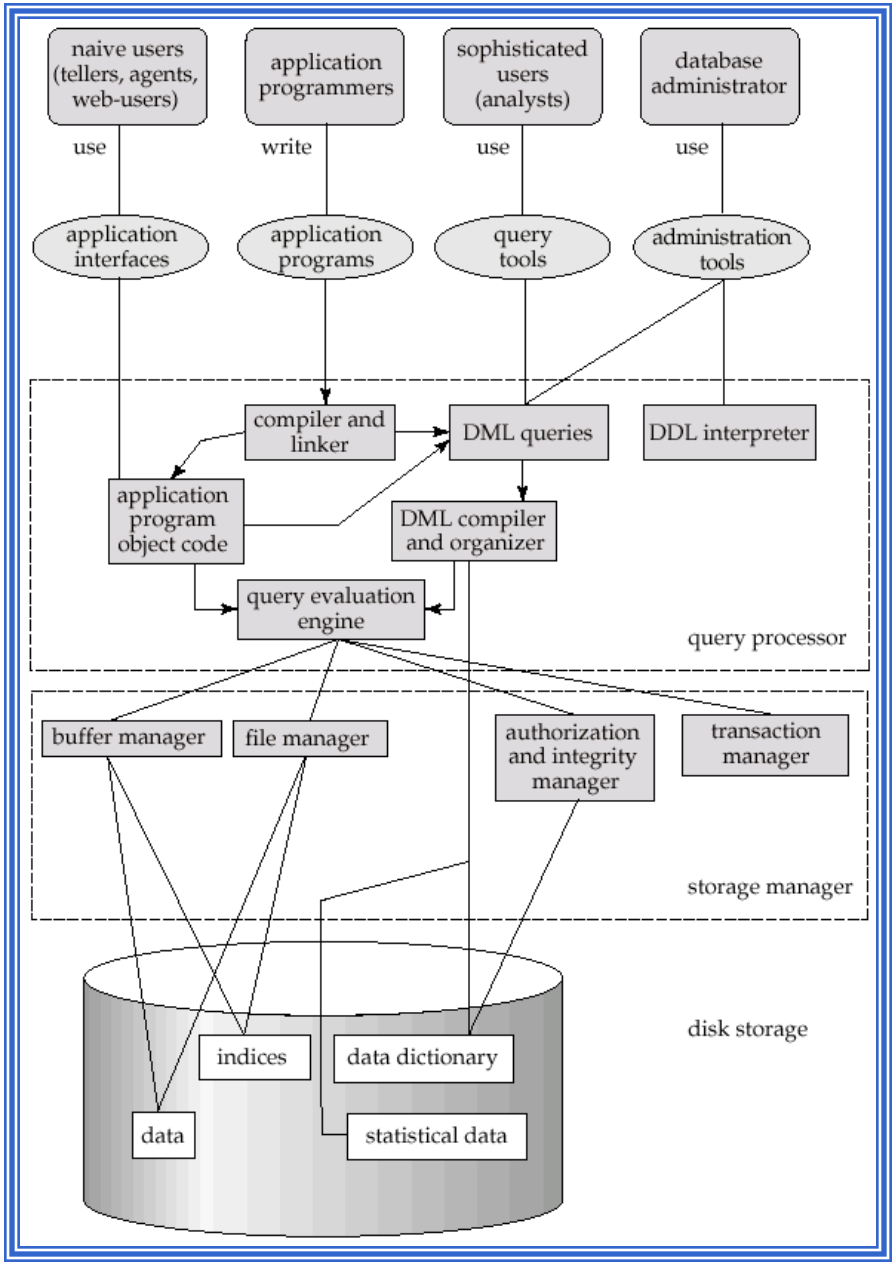
# Storage Manager

Read Sec. 4.2 and 4.3 Riguzzi et al.
Sistemi Informativi

Some slides by Avi Silberschatz, some by Atzeni et al.

# DBMS Architecture



# Storage Manager

- The storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - interaction with the file manager
  - efficient storing, retrieving and updating of data

# Transaction Management

- A *transaction* is a collection of operations that performs a single logical function in a database application
- The transaction manager component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and concurrent transaction interactions.
  - Reliability manager
  - Concurrency manager

# Buffer manager

- A database system seeks to minimize the number of block transfers between the disk and memory. We can reduce the number of disk accesses by keeping as many blocks as possible in main memory.
- **Buffer** – portion of main memory available to store copies of disk blocks.
- **Buffer manager** – subsystem responsible for allocating buffer space in main memory.
- Primitives: ***fix***, ***unfix***, ***setDirty***, ***force***.

# Data structures

- Each block in the memory has
  - A bit, called dirty bit, that is 1 if the page has unwritten changes
  - A counter that keeps track of how many transactions are using the page

# Primitives

- ***fix:*** request of a block, It requires a read only if the page is not in the buffer (increment of the counter associated with the block)
- ***setDirty:*** tells the buffer manager that the block has been modified (set the dirty bit to 1)
- ***unfix:*** indicates that the transaction has finished using the block (decrement of the counter associated to the block)
- ***force***: immediate transfer of a block on secondary memory (by request of the reliability manager, not by the query processor)

# Execution of the fix

- Programs send a fix to the buffer manager to get a block from disk.
  1. If the block is already in the buffer, the requesting program is given the address of the block in main memory

# Execution of the fix

2. If the block is not in the buffer,
    1. Look for an empty block in the buffer (zero counter)
        - If the buffer manager finds an empty block, it reads the block from the disk to the buffer, and passes the address of the block in main memory to requester.
        - Otherwise, two alternatives

          "**steal**": selection of a victim, a used block in the buffer. The data of the victim are written to disk if the Dirty bit is at 1. It reads the required block from the disk to the buffer, and passes the address of the block in main memory to requester.

          "**no-steal**": the fix has to wait
    2. The block counter is incremented

# Buffer-Replacement Policies

- Most operating systems replace the block **least recently used** (LRU strategy)
- Idea behind LRU – use past pattern of block references as a predictor of future references
- Queries have well-defined access patterns (such as sequential scans), and a database system can use the information in a user's query to predict future references

# Types of writes

- The buffer manager writes to disk
  - **Synchronously** when it is asked with a force
  - **Asynchronously** when it thinks it is convenient or necessary. It may decide to anticipate or postpone a write to coordinate and/or exploit different devices

# DBMS and File System

- The File System component of the Operating System manages secondary storage
- DBMS uses the File System for creating and deleting files and for reading and writing single blocks or sequences of blocks

# DBMS and File System

- The DBMS creates large files that uses to store various tables (possibly the entire database)
- It is also possible that the tuples from the same table are stored on different files
- The DBMS manages the blocks in the files as if they were a large secondary storage space and it builds in such a space the physical structures for storing data

# Organization of Records in Files

- Three ways to decide how to organize records in files
  - **Sequential** – records are stored in a sequence
  - **Tree structures** – records are organized in a search tree
  - **Hashing** – a hash function computed on some attribute of each record; the result specifies in which block of the file the record should be placed
- This is also called the **primary structure**

# Sequential Organization

- **Heap** – a record can be placed anywhere in the file where there is space
  - New records are added
    - At the tail (with periodical reorganizations)
    - In place of deleted records
  - Access method: sequential scan, efficient because no empty space

# Sequential Organization

- **Ordered** – store records in order, based on the value of one or more fields (also called pseudo-key or search key) of each record
- Fast output of the ordered table
- Inserting a new record requires re-organizaing the file
- To avoid reorganizations:
  - Empty space left
  - New blocks are added to the file (contiguity is lost)
  - Overflow chains are created

# Sequential Organization

- **Array***:* the position of the record depends on the value of a key
- File seen as an array
- A position is reserved for each value in the range of the key
- E.g. key range=[1,100], 100 positions, file length=100
- The record is put in the position indicated by the key
  - 1st value in the range of the key-> 1st position in the array
  - ….

# Tree Structures

- Records are organized in a tree-like structure (index) to speed the search

# Nomenclature

- Clustered-Unclustered Organization:
  - Clustered structures: ordered, array, tree, hash
  - Unclustered: heap
- Clustering Organization:
  - Mixing records of different tables in the same block (or file)
  - good for queries involving joins
  - bad for queries involving only one table
  - results in variable size records