DBMS e Transazioni

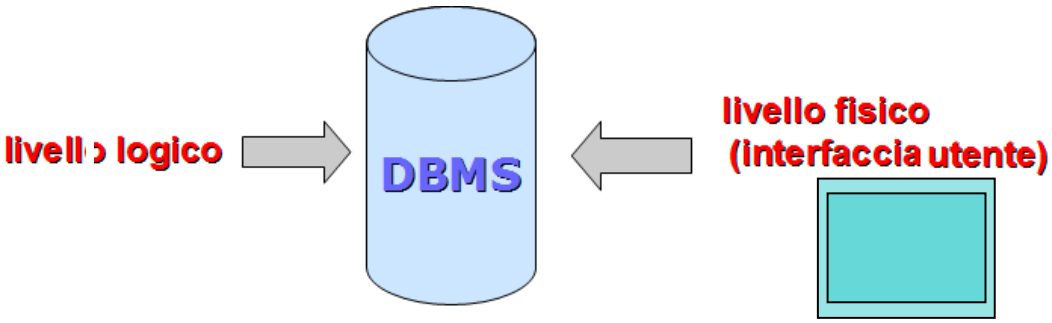
Un DBMS *(Data Base Management System*) è un sistema centralizzato (programmi coordinati) o distribuito (rete) che permette di memorizzare, modificare ed estrarre ***informazioni*** da un database (DB), realizzando l'indipendenza del SW dall'organizzazione fisica e logica delle strutture dati.

Un DBMS:

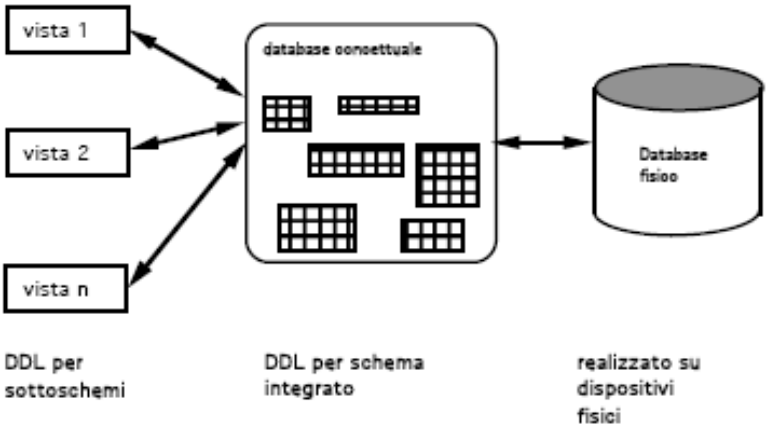
- Garantisce *l'integrità dei dati* (unica raccolta di dati anziché copie distinte scoordinate che potrebbero causare duplicazioni, ridondanze) assicurando **consistenza** cioè coerenza (specie sicurezza negli aggiornamenti) evitando contraddizioni tra i dati archiviati.
- Organizza le informazioni del database secondo la struttura di un database **gerarchico**, di un database **di rete** o di un database **relazionale** o ad **oggetti** gestendo **grandi moli di dati *non volatili*** in un **ambiente multiutente**, consentendo **elaborazione concorrente**
- Garantisce l'accesso **corretto** e **concorrente** alle informazioni (con gestione di *transazioni*) aumentando la sicurezza - intesa come riservatezza - sia a livello logico (solo a persone **autorizzate** p.e. tramite una password) sia a livello fisico (si possono impostare *privilegi diversi*)
- Un DBMS oltre a stabilire schemi organizzativi e di **controllo**, rende le informazioni **accessibili** agli utenti, tramite query (senza necessità di conoscere dettagli implementativi) ed in **modo efficiente**

Il **Data Base Management System** mette in relazione il livello logico e quello fisico, nel progetto di database, ne è il *collante* e su di esso si basano applicazioni come Microsoft Access.

Esempi: Oracle, Microsoft SQL server, MySQL, PostgreSQL, InterBase.



Il **livello fisico**: strutture di memoria di massa usate per conservare i dati e per **accedervi** in modo efficiente.
Nascosto all'utente (*l'interfaccia utente* è ciò che l'utente visualizza a video)



Il DBMS mantiene un modello astratto dei dati, l'utente **vede** quindi i dati in base a ciò che, per esso, rappresentano. Esisteranno dunque diversi *livelli di astrazione*, sia per l'istanza che per lo schema.

Tra tali livelli di astrazione, il **livello esterno**, o **vista** è quello più vicino all'utente 'finale' del database, che corrisponde al **modo di vedere i dati** da parte dell'utente stesso.

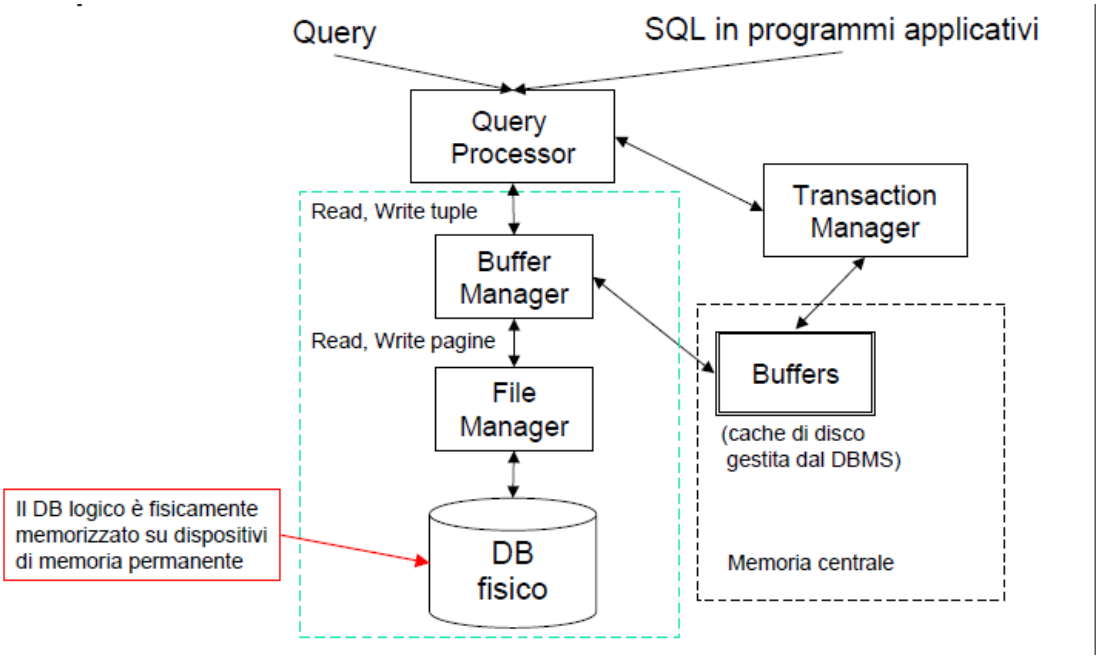
(*DDL*: **l**inguaggio di **d**efinizione dei **d**ati che permette di agire sullo schema del database, creando, modificando o eliminando oggetti)

Livello fisico o **interno** è il più vicino all'elaboratore, che corrisponde al modo secondo il quale i dati sono effettivamente memorizzati. È necessario distinguere tra:

- I **dati** veri e propri
- Le **strutture** che li contengono e che ci permettono di **accedere** ai medesimi

Il livello fisico è, per l'utente, del tutto trasparente: egli, infatti, non si preoccupa affatto di come i dati vengano registrati sui supporti, tale funzione è compito esclusivo del DBMS. L'**utente** si occuperà principalmente ***del cosa*** vi è registrato: quali sono i dati e in quale relazione si trovano tra loro.

Essendo la gestione dei file realizzata dal sistema, l'utente non deve interessarsi della forma assunta dai dati in memoria di massa e di come accedervi; non gli resta dunque che occuparsi del valore informativo dei dati.



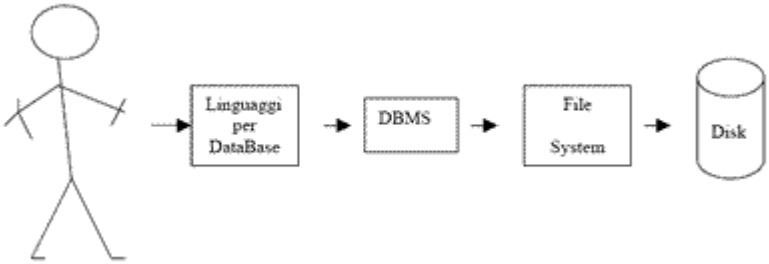
Le **strutture fisiche di accesso** descrivono il modo in cui vengono organizzati i dati per garantire operazioni di ricerca e di modifica efficienti da parte dei programmi applicativi. In genere, ciascun DBMS ha a disposizione un numero abbastanza limitato di tipi di strutture di accesso; ad esempio, nei sistemi relazionali sono disponibili **semplici indici**, che vengono definiti dal progettista tramite istruzioni *DDL* (con possibilità di eliminare tale indicizzazione in modalità diversa a seconda del DBMS).

Differenze tra DBMS e file system

Il file system è un nucleo, costituito da programmi, presente in ogni sistema operativo. La sua funzione è quella di gestire le varie operazioni sui file; questa gestione non è visibile all'utente, infatti il file system opera direttamente al servizio di altri programmi o utility.

Il DBMS è per l'appunto un software che poggiando sul sistema operativo utilizza il file system di quest'ultimo. consentendo **indipendenza del SW** dall'organizzazione logica e fisica.

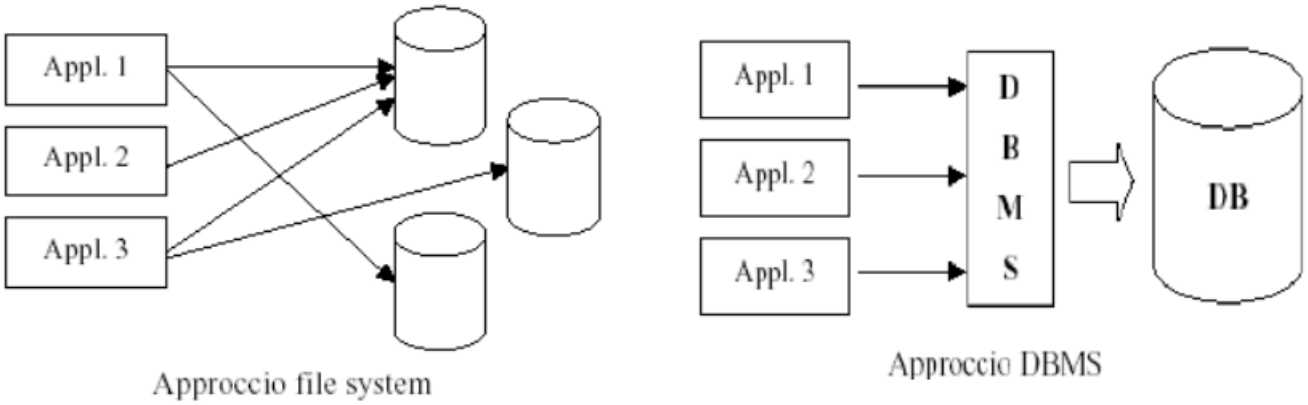
In questo caso dunque, la maggiore 'distanza' del DBMS dall'hardware, rispetto al file system, permette un grado di iterazione maggiore. Questo significa che l'utente (programmatore, amministratore di sistema, ecc.) non dovrà più avere a che fare con record e file, bensì con entità astratte che rappresentano la realtà.



Una suddivisione semplificata (quindi parziale), utile a comprendere per linee generali il comportamento di un DBMS, potrebbe essere questa:

1. Gestore delle interrogazioni
2. Gestore dei ***metodi di accesso*** (ad esempio ***accessi sequenziali indicizzati*** ***I**ndexed **S**equential **A**ccess **M**ethod o ISAM*)
3. Gestore del buffer (***Buffer manager***)

*Per un confronto: la tradizionale gestione mediante archivi **vs.** l'attuale gestione con DBMS*



Prima dello sviluppo dei DBMS l'approccio che veniva applicato al problema dell'archiviazione prevedeva l'uso diretto delle strutture del file system. In tale soluzione, le applicazioni accedono direttamente agli archivi, quindi ognuna deve conoscere la struttura interna degli archivi e le relazioni tra i dati e deve evitare la duplicazione degli stessi. Inoltre la non volatilità dei dati e la gestione degli accessi contemporanei di più applicazioni agli archivi viene relegata a strati software non specializzati per tali compiti, quali il sistema operativo.

La caratteristica saliente che differenzia un sistema per la gestione di database è la presenza di un componente specializzato a tale ruolo. In tale soluzione, le applicazioni rivolgono al DBMS le proprie **richieste di accesso alla base di dati**, il quale gestisce i dati svincolando le applicazioni da tale onere.

Quindi il DBMS è un modulo, specializzato nella gestione del DB a cui tutte le applicazioni si rivolgono per accedere ai dati. Si ottiene così un triplice scopo: da una parte le funzionalità di gestione del database sono raggruppate in un unico insieme, dall'altra le applicazioni risultano alleggerite e quindi più veloci da realizzare e, soprattutto, **nessuna potrà effettuare operazioni scorrette sul database**.

(*guida alla lettura di appunti sui Database e DBMS*)

I DBMS – non ad uso didattico – hanno anche degli aspetti negativi:

· I DBMS sono prodotti ***costosi, complessi*** e abbastanza diversi da molti altri strumenti informatici. La loro introduzione comporta quindi notevoli investimenti, diretti (acquisto del prodotto) e indiretti (acquisizione delle risorse hardware e software necessarie, conversione delle applicazioni, formazione del personale).

· I DBMS forniscono, in forma integrata, una serie di servizi, che sono necessariamente associati ad un costo. Nei casi in cui questi servizi non siano tutti necessari, è ***difficile scorporare i servizi*** effettivamente richiesti dagli altri, e ciò può comportare una riduzione di prestazioni.

Efficace controllo, per evitare **anomalie**, si realizza impostando – a livello logico – l'**integrità referenziale**:

> *" insieme di regole usate per assicurare che le relazioni tra i record delle tabelle correlate siano valide e che non vengano eliminati o modificati per errore i dati correlati."*

Si può impostare se:

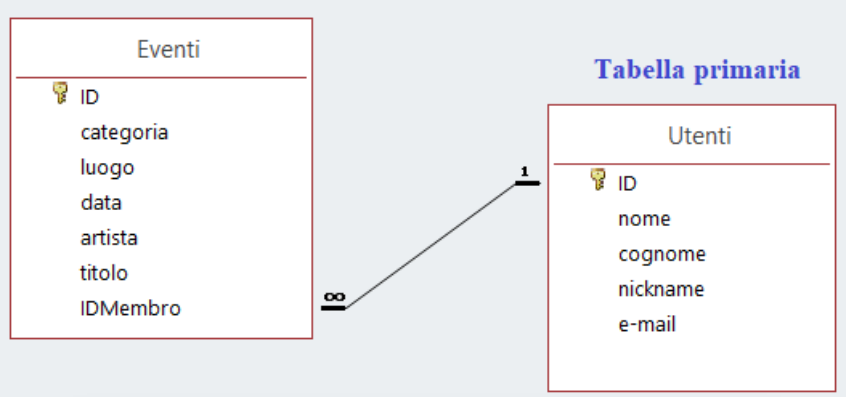
- il campo della tabella primaria è una chiave primaria
- i campi correlati contengono lo stesso tipo di dati

Si distinguono tre tipi di **anomalia**:

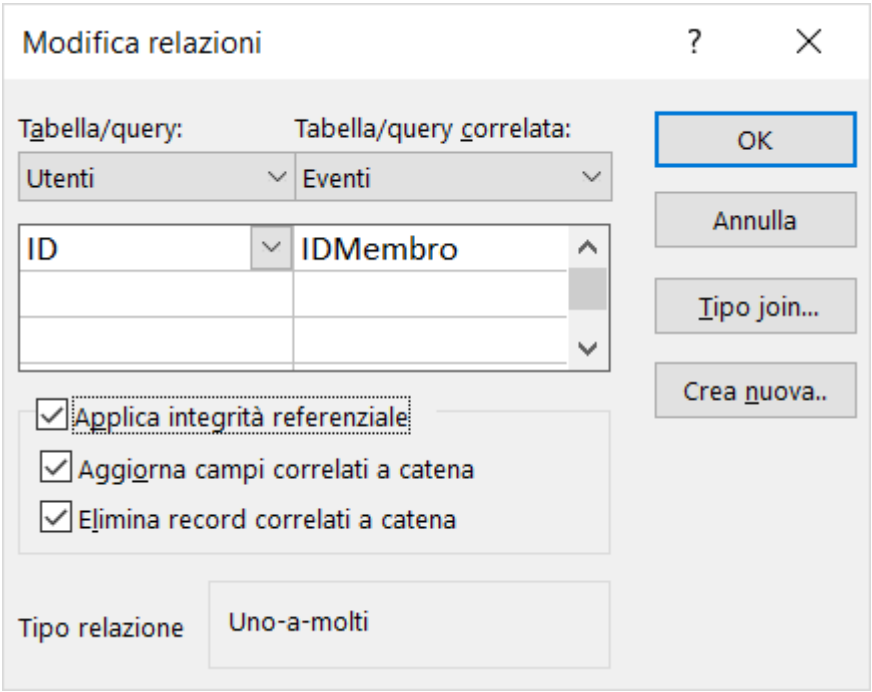
1. *Anomalia di inserimento*. Se nell'inserire un nuovo record in una tabella si è costretti a inserire informazioni già presenti nel DB.

2. *Anomalia di cancellazione*. Se nel cancellare un record si è costretti a cancellare informazioni che possono essere ancora utili nel DB.

3. *Anomalia di aggiornamento*. Se per aggiornare un record si è costretti ad aggiornarne molti altri.



Nb:
sia IDMembro (***chiave esterna***) sia ID (***chiave primaria*** della tabella primaria) sono di **tipo numerico**



Glossario

DB: un insieme di dati logicamente connessi, **strutturati** in accordo ad un preciso **schema** (*modello dei dati concettuale* - E/R - o *logico* - rappresentazione tabellare - con espliciti vincoli), memorizzati su supporto permanente in cui è possibile effettuare operazioni (ricerche e modifiche) caratterizzate dalle proprietà **ACID** (**Atomicity**, **Consistency**, **Isolation** e **Durability**).

Sottolineando il diverso approccio rispetto al gestire file indipendenti (**archivi**), i dati sono usabili per diverse applicazioni, anche in *concorrenza tra loro* (per questo le operazioni sui dati devono apparire **I**solate), utenti diversi possono interessarsi ad un sottoinsieme dei dati presenti (vista o "*view*") ed esiste un'*integrazione* che rende minima la ridondanza (evita incontrollata duplicazione). Esistono, infine, meccanismi di *sicurezza* (**C**onsistenza cioè integrità dei dati) e *ripristino* (**D**urevolezza cioè persistenza).

Le operazioni fondamentali che si possono eseguire su un database (**ricerca e modifica: inserimento, aggiornamento** e **cancellazione**) sono ***procedure indivisibili*** (**A**tomicità).

Transazione: sequenza di azioni di una procedura che si propone di modificare od osservare la base di dati con la proprietà di **atomicità**: o vengono eseguite del tutto come un'unica azione elementare e indivisibile, o non vengono eseguite affatto ed eventuali effetti parziali vengono annullati.

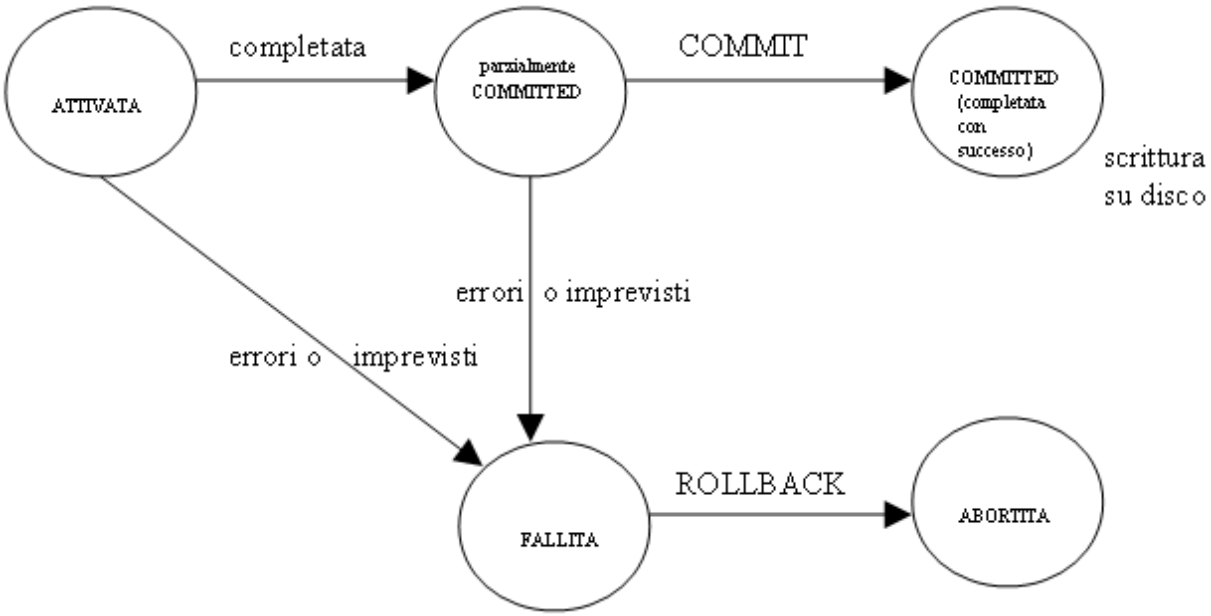


Diagramma degli stati di avanzamento di una transazione

ACID: deriva dall'acronimo inglese **Atomicity**, **Consistency**, **Isolation** e **Durability** (**A**tomicità, **C**oerenza o consistenza, **I**solamento e **D**urabilità) cioè le proprietà dei meccanismi che implementano transazioni che operino in modo corretto sui dati.

- **atomicità**: la transazione è indivisibile nella sua esecuzione e la sua esecuzione deve essere o totale o nulla, non sono ammesse esecuzioni parziali;
- **coerenza** o consistenza: quando inizia una transazione il database si trova in uno stato coerente e quando la transazione termina il database deve essere in uno stato coerente, ovvero non deve violare eventuali vincoli di integrità, quindi non devono verificarsi contraddizioni (*inconsistency*) tra i dati archiviati nel DB;
- **isolamento**: ogni transazione deve essere eseguita in modo isolato e indipendente dalle altre transazioni, l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione;
- **durabilità (**durevolezza o durata**)**: detta anche **persistenza**, si riferisce al fatto che una volta che una transazione abbia richiesto un *commit work*, i cambiamenti apportati non dovranno essere più persi. Per evitare che nel lasso di tempo fra il momento in cui la base di dati si impegna a scrivere le modifiche e quello in cui li scrive effettivamente si verifichino perdite di dati dovuti a malfunzionamenti, vengono tenuti dei registri di log dove sono annotate tutte le operazioni sul DB.