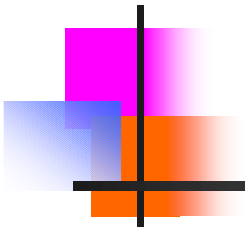
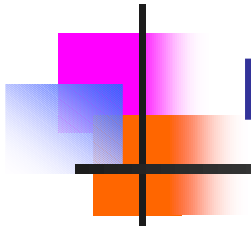


Le funzionalità di un DBMS

Sistemi Informativi L-A

Home Page del corso:
http://www-db.deis.unibo.it/courses/SIL-A/

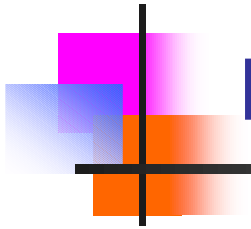
Versione elettronica: DBMS.pdf



DBMS: principali funzionalità

- Le caratteristiche fondamentali di un DBMS sono 3, riassumibili dicendo che:

  un DBMS è un sistema software che gestisce grandi quantità di dati persistenti e condivisi
- La gestione di grandi quantità di dati richiede particolare attenzione ai problemi di efficienza (ottimizzazione delle richieste, ma non solo!)
- La persistenza e la condivisione richiedono che un DBMS fornisca dei meccanismi per garantire l'affidabilità dei dati (fault tolerance), per il controllo degli accessi e per il controllo della concorrenza
- Diverse altre funzionalità vengono messe a disposizione per motivi di efficacia, ovvero per semplificare la descrizione dei dati, lo sviluppo delle applicazioni, l'amministrazione di un DB, ecc.



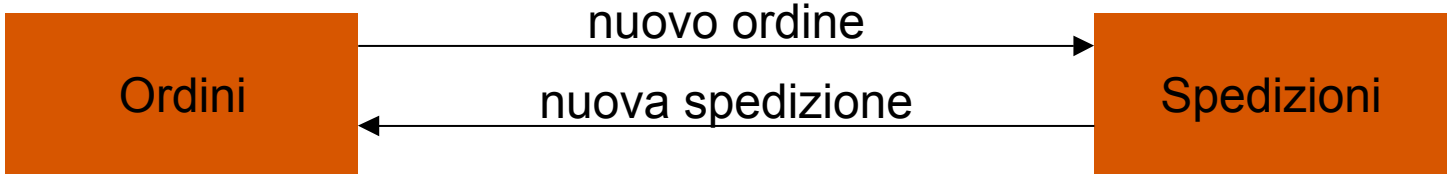
Perché non i file system?

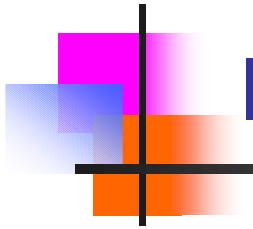
- Per gestire grandi quantità di dati in modo persistente e condiviso, sarebbe anche possibile fare uso dei file system, ma ciò ha una serie di inconvenienti, tra cui:
  - Non sono disponibili i servizi aggiuntivi offerti da un DBMS
  - I meccanismi di condivisione sono limitati, in particolare il livello di granularità è quello del file

    Due utenti non possono modificare contemporaneamente parti (record) diverse di uno stesso file
  - L'accesso a file condivisi richiede una descrizione degli stessi nel codice delle applicazioni, con rischi di descrizioni errate e quindi inconsistenti
- Per contro, la gestione dei dati mediante file system può risultare più efficiente che con un DBMS, proprio per la maggiore semplicità dei primi

Perché condividere i dati?

- I diversi settori in cui si articola una (grande) organizzazione e/o le diverse applicazioni hanno in comune vari dati di interesse. Pertanto la gestione integrata e la condivisione di tali dati permettono di evitare ripetizioni (ridondanza dovuta a copie multiple dello stesso dato), e quindi inutile spreco di risorse (memoria)
- Inoltre, la ridondanza può dar luogo a problemi di inconsistenza delle copie e, in ogni caso, comporta la necessità di propagare le modifiche, con ulteriore spreco di risorse (CPU e rete)
  - Il settore Ordini di un'azienda manifatturiera memorizza i propri dati in un file, non condiviso con gli altri settori aziendali. Ogni volta che arriva un ordine, i dati relativi devono essere trasmessi al settore Spedizioni, affinché l'ordine possa essere evaso. A spedizione eseguita, i dati relativi devono essere ritrasmessi al settore Ordini





Il modello dei dati

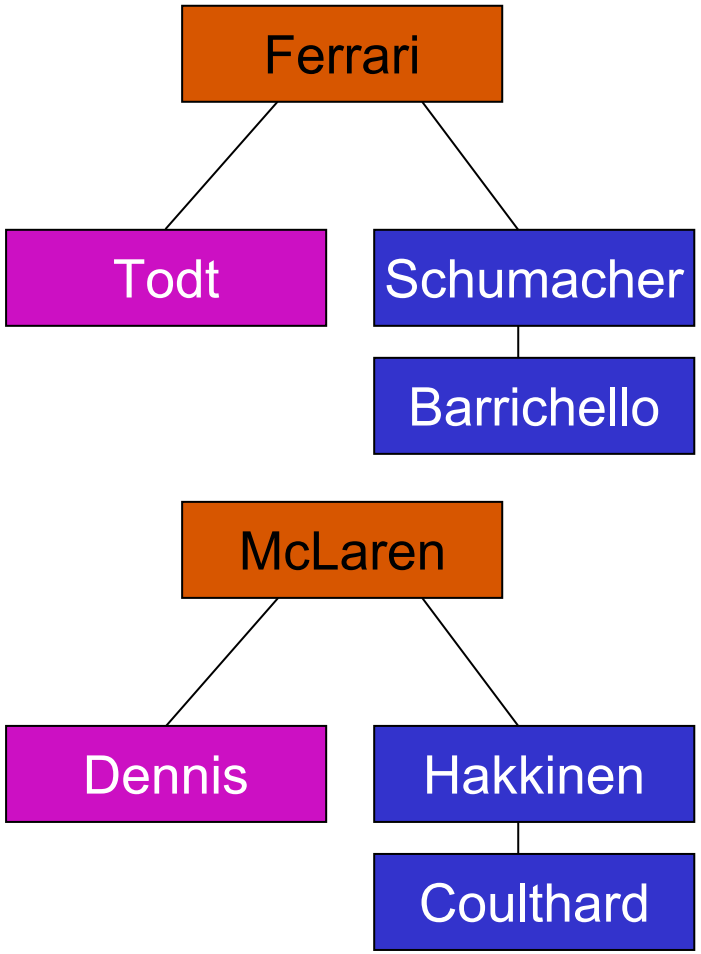
- Dal punto di vista utente un DB è visto come una collezione di dati che modellano una certa porzione della realtà di interesse
- L'astrazione logica con cui i dati vengono resi disponibili all'utente definisce un modello dei dati; più precisamente:

  un modello dei dati è una collezione di concetti che vengono utilizzati per descrivere i dati, le loro associazioni, e i vincoli che questi devono rispettare

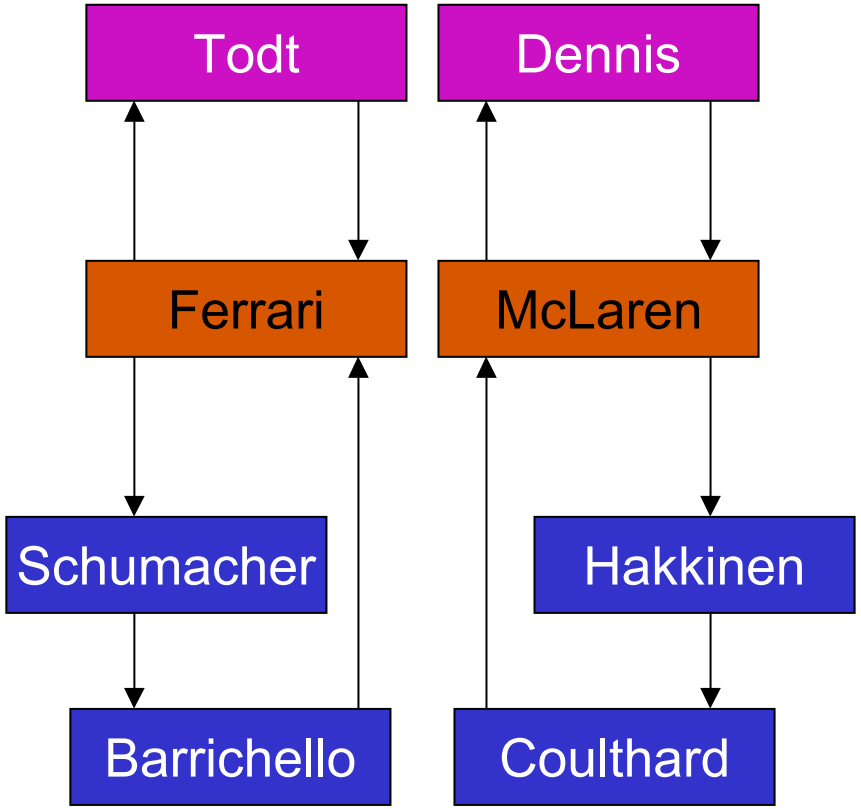
- Un ruolo di primaria importanza nella definizione di un modello dei dati è svolto dai meccanismi che possono essere usati per strutturare i dati (cfr. i costruttori di tipo in un linguaggio di programmazione)
  - Ad es. esistono modelli in cui i dati sono descritti (solo) sotto forma di alberi (modello gerarchico), di grafi (modello reticolare) e di oggetti complessi (modello a oggetti)

Esempio: scuderie e piloti (1)

- …nel modello gerarchico



- …nel modello reticolare



Esempio: scuderie e piloti (2)

- …e nel modello relazionale

Ferrari	Todt
McLaren	Dennis

Ferrari	Schumacher
Ferrari	Barrichello
McLaren	Hakkinen
McLaren	Coulthard

ATTENZIONE: le tabelle sono solo “un modo” (conveniente) di rappresentare le relazioni

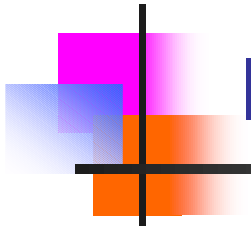
Schemi e istanze

- In ogni DB si hanno due componenti:
  - Lo schema, che descrive la struttura dei dati (la cosiddetta parte intensionale del DB)
  - L'istanza, ovvero i dati veri e propri (la parte estensionale del DB)
- Lo schema permette di fatto di interpretare i dati dell'istanza

Scuderia	D_S
Ferrari	Todt
McLaren	Dennis

Scuderia	Pilota
Ferrari	Schumacher
Ferrari	Barrichello
McLaren	Hakkinen
McLaren	Coulthard

  - In generale, mentre un'istanza varia nel tempo, lo schema tende a restare invariato, e variazioni dello schema comportano una ristrutturazione più o meno semplice del DB



Indipendenza fisica e logica

- Tra gli obiettivi di un DBMS vi sono quelli di fornire caratteristiche di:

Indipendenza fisica

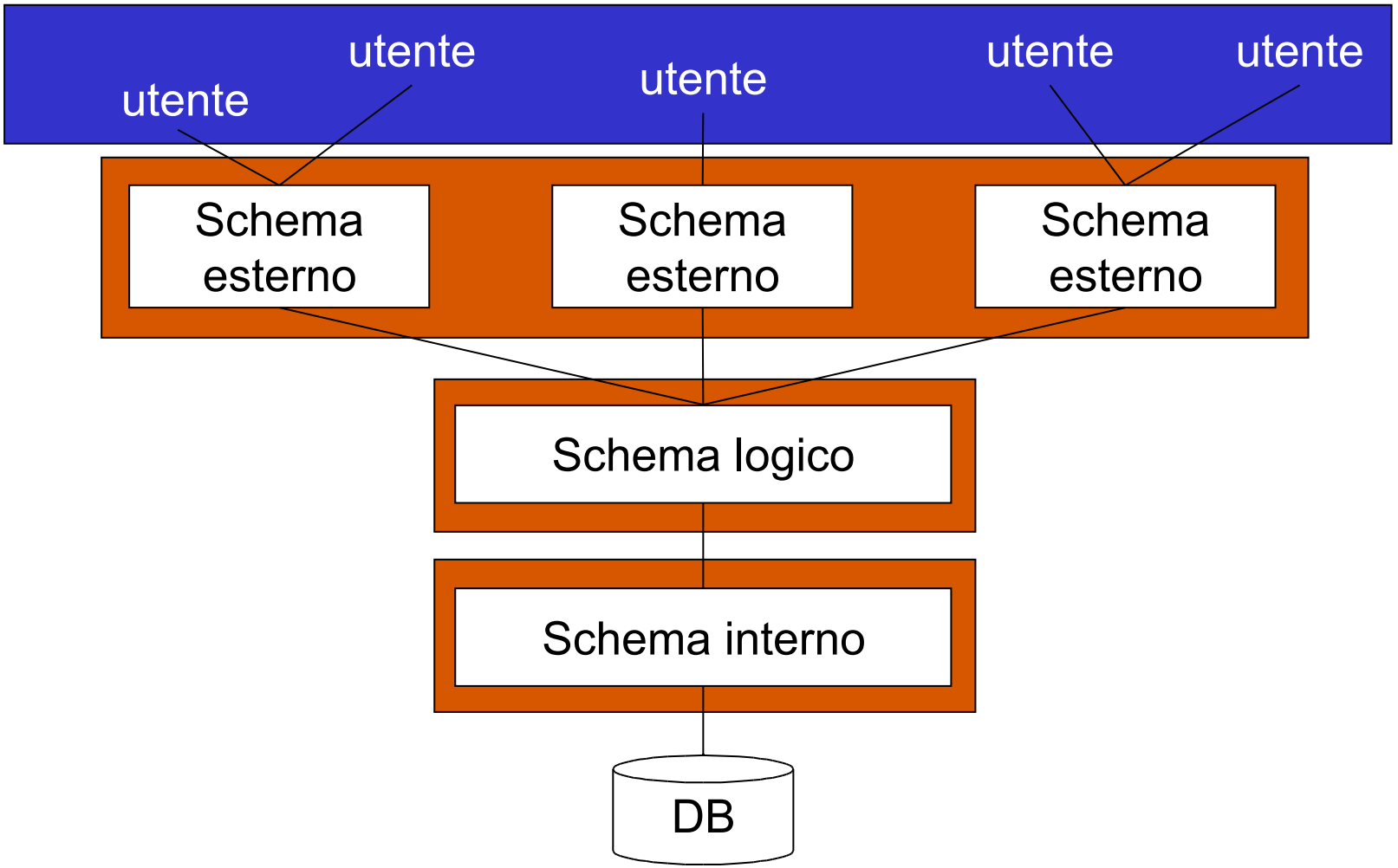
- L'organizzazione fisica dei dati dipende da considerazioni legate all'efficienza delle organizzazioni adottate. La riorganizzazione fisica dei dati non deve comportare effetti collaterali sui programmi applicativi

Indipendenza logica

- Pur in presenza di uno schema logico integrato non è utile o conveniente che ogni utente ne abbia una visione uniforme

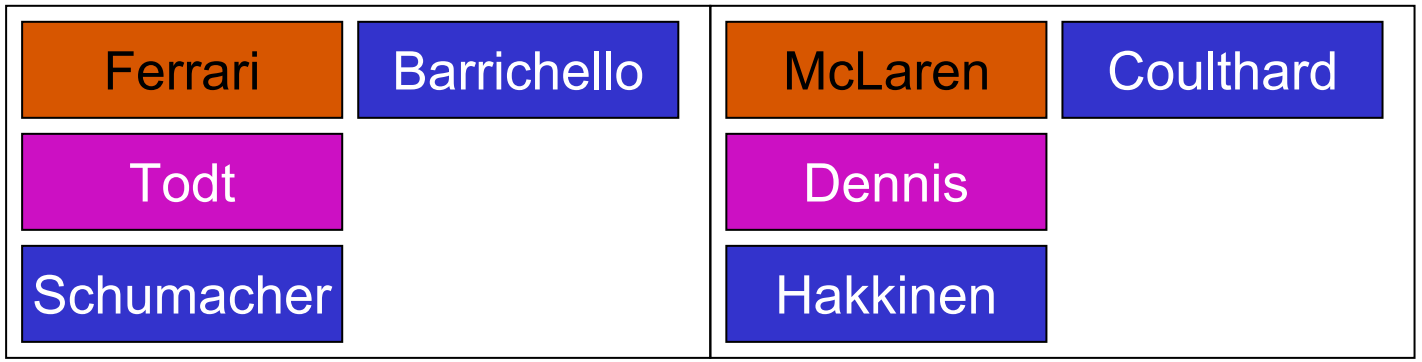
- La soluzione porta a quella che è comunemente nota come…

Architettura a 3 livelli di un DBMS



Il livello fisico (o interno)

- Il "DB fisico" consiste di una serie di file, residenti su memorie di massa (dischi, nastri,...) che contengono dati, indici e altro
- Lo schema fisico descrive come il DB logico è rappresentato a livello fisico (ad es.: in quale/i file è memorizzata una relazione)



File: **`c://apps/DB/data/scuderie`**

- La gestione del DB fisico è a carico di chi amministra il DBMS e non degli utenti, che quindi possono concentrarsi su aspetti di più alto livello per lo sviluppo delle loro applicazioni

Il livello delle viste (o esterno)

- Il livello esterno viene costruito a partire dallo schema logico integrato mediante la definizione di viste *ad hoc*, che descrivono parte dello schema logico secondo le esigenze dei diversi utenti
  - Ad es. e' possibile definire una vista che combina i dati di più relazioni:

Pilota	D_S
Schumacher	Todt
Barrichello	Todt
Hakkinen	Dennis
Coulthard	Dennis

La distinzione tra livello esterno e logico può, in molti casi, risultare trasparente agli utenti, che, ad es., in un RDBMS vedono semplicemente un insieme di tabelle

A cosa servono le viste?

- Oltre a fornire una visione “personalizzata” del DB, le viste possono svolgere un ruolo importante anche per diversi altri motivi:
  - Una ristrutturazione dello schema integrato può, in alcuni casi, essere opportunamente “mascherata” facendo uso di viste
  - Mediante le viste è possibile regolare meglio il controllo degli accessi al DB, ad es. mascherando dati riservati
  - Le viste possono essere usate per “calcolare” dinamicamente nuovi dati a partire da quelli memorizzati nel DB, senza per questo introdurre ridondanza

Pilota	DataNascita
Schumacher	03/01/1969
Hakkinen	28/09/1968

Il 28/09/2001

Pilota	Età
Schumacher	32
Hakkinen	33

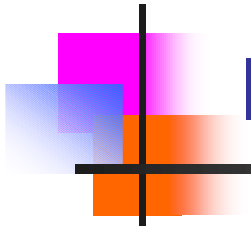
I 3 livelli: una semplice analogia

- Per fissare meglio le idee, si considerino due classi Java, Pilota e Scuderia:
  - Il livello logico corrisponde (in un certo senso) all'interfaccia delle classi
  - Il livello interno corrisponde alla loro implementazione concreta
  - Il livello esterno può includere (un metodo di) una classe che usa i servizi delle classi Pilota e Scuderia

```
public class Pilota {
private String nome;
private Date datanascita; …
public Date getDataNascita() {…} …}
```

```
public int Eta(Pilota unPilota) {
int eta =
      diffYears(new Data(),unPilota.getDataNascita());
return eta;
}
```

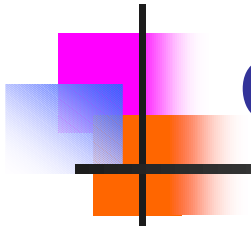
```
public class Scuderia {
private Pilota[ ] iPiloti;
…
public Scuderia() {
iPiloti  = new Pilota[2];
…}

 public LinkedList getPiloti() {
…}
…}
```



I linguaggi dei DBMS

- Un DBMS mette a disposizione diversi linguaggi per interagire con le BD. Il livello di astrazione di tali linguaggi dipende fortemente dal modello dei dati cui ci si riferisce
- Una comune distinzione classifica i linguaggi sulla base delle funzioni svolte:
  - DDL (Data Definition Language)
    - Serve per definire gli schemi (logici, esterni, interni)
  - DML (Data Manipulation Language)
    - Serve per interrogare e modificare le istanze delle BD
  - DCL (Data Control Language)
    - Include comandi di vario tipo, ad es. per il controllo degli accessi

SQL riunisce in sé istruzioni di tutte le tre tipologie
(per cui si parla del DDL di SQL, del DML di SQL e del DCL di SQL)



Condivisione: regolamentare gli accessi

- Gli utenti di un DB sono naturalmente suddivisibili in diverse tipologie, a cui vanno pertanto associate autorizzazioni distinte
  - Uno studente può leggere i propri dati, ma non quelli di altri studenti; inoltre non può modificare l'elenco degli esami sostenuti ;-)
  - Un docente può leggere i dati dei soli studenti del proprio corso; non può modificare l'elenco degli esami già sostenuti da uno studente, ma può registrare esami del proprio corso
  - La segreteria può leggere i dati di tutti e può registrare nuovi studenti

- La gestione delle autorizzazioni può essere oltremodo complessa, per questo motivo sono previste specifiche figure di Data Base Administrator (DBA) che conferiscono agli utenti i "giusti" privilegi
- Mediante il DCL di SQL la concessione dei privilegi è molto semplificata, e inoltre permette, oltre ad assegnare un privilegio ad un utente (ad es. leggere i dati di una tabella), anche di consentirgli di "passarlo" ad altri

Condivisione: gestire la concorrenza (1)

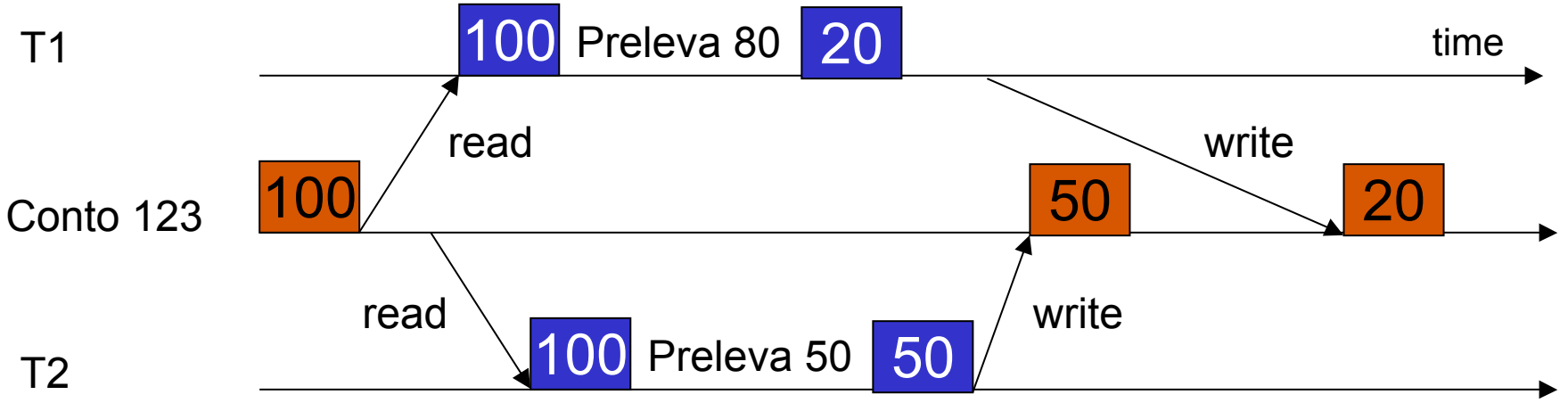
- Un DBMS deve garantire che gli accessi ai dati, da parte di diverse applicazioni, non interferiscano tra loro. Al fine di conservare l'integrità dei dati è pertanto necessario far ricorso a opportuni meccanismi di controllo della concorrenza

Es.: si considerino due richieste, T1 e T2, che cercano di eseguire in contemporanea un'operazione di prelievo, rispettivamente di 80 e 50 €, dal conto corrente numero 123 che ha una disponibilità iniziale di 100 €:

```
CC = read(C_Correnti; 123);
if CC.disponibile >= Q then {
        CC.disponibile = CC.disponibile - Q;
        eroga la somma Q;
        write(C_Correnti;CC)}
                else print("operazione non possibile");
```

Condivisione: gestire la concorrenza (2)

- Una possibile esecuzione (non corretta!) è la seguente:

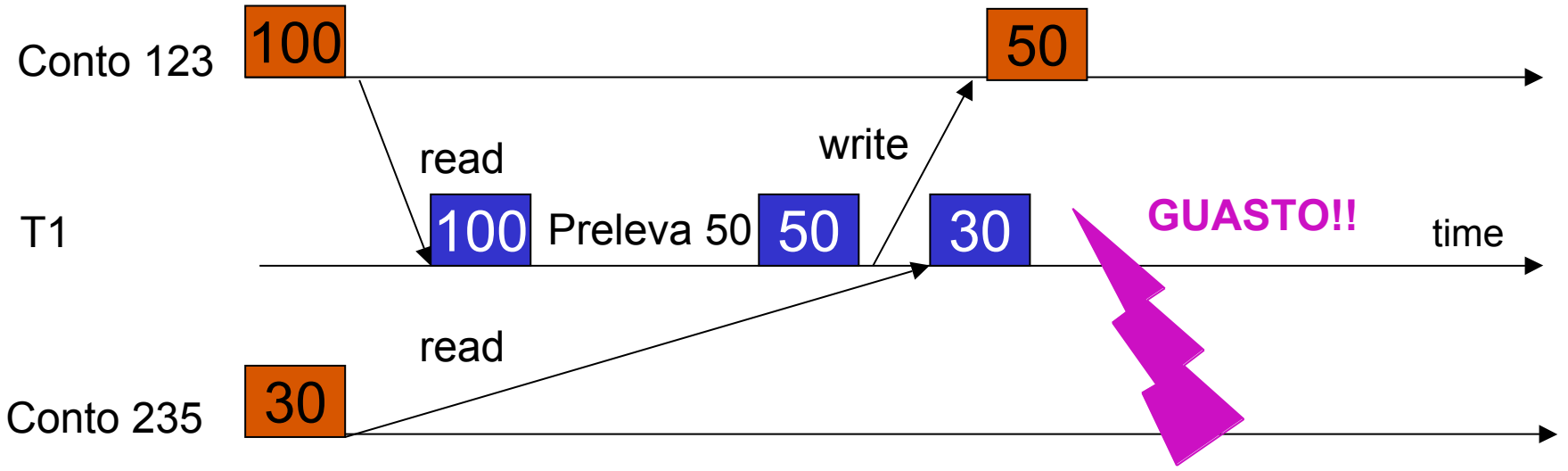


- È compito del DBMS evitare tali esecuzioni non corrette
  - ad es. impedendo a T1 di scrivere il nuovo saldo (20) e forzando la rilettura del saldo aggiornato (50)

Persistenza: protezione dai guasti

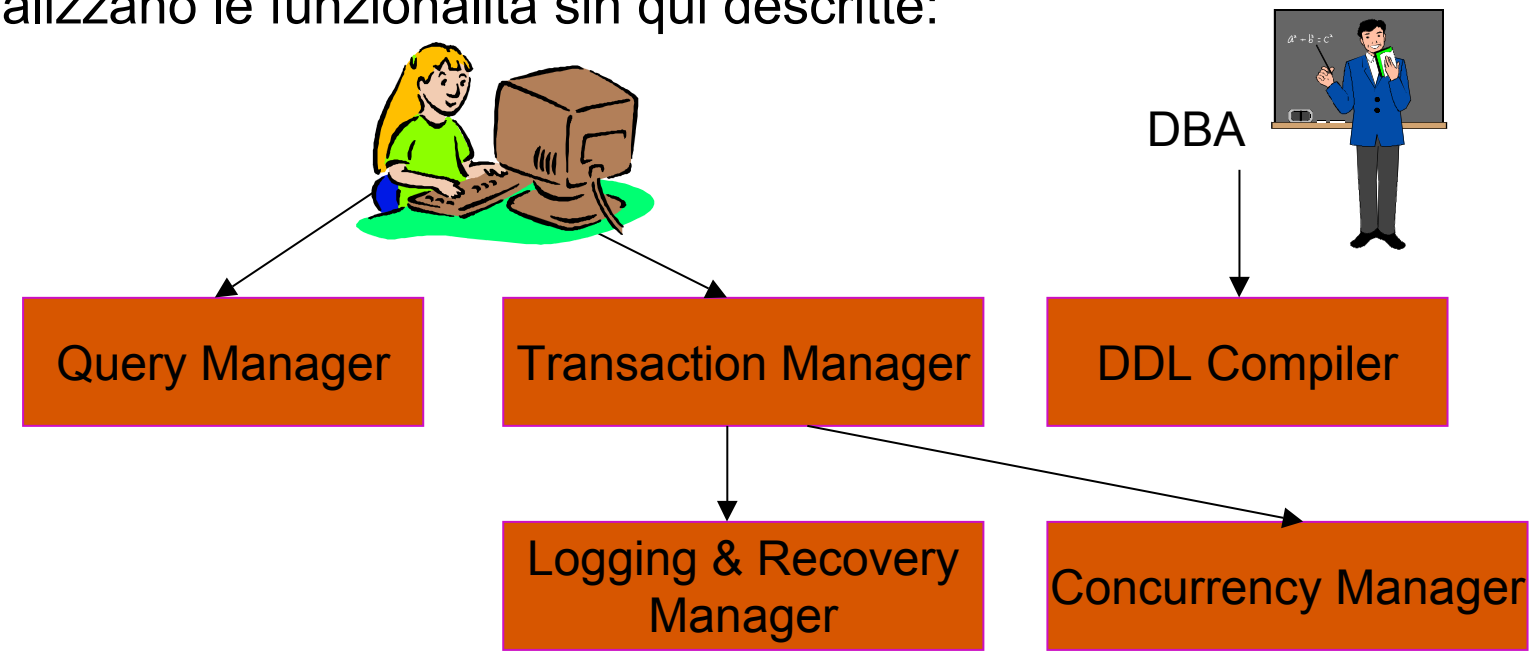
- Quando un'applicazione deve operare più operazioni di modifica, è possibile che per qualche motivo (guasti, intervento dell'utente, ecc.) solo una parte di queste venga effettivamente eseguita
- In questo caso, per garantire l'integrità dei dati, il DBMS deve provvedere ad annullare tali modifiche

Es.: se T1 volesse trasferire 50 € dal conto 123 (saldo 100 €) al conto 235 (saldo 30 €), si potrebbe verificare la seguente situazione (indesiderata!):

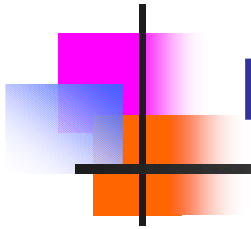


Moduli di un DBMS

- Per quanto visto, e senza entrare troppo nei dettagli architetturali, si possono già individuare alcuni moduli fondamentali di un DBMS, che realizzano le funzionalità sin qui descritte:



QM: analizza, autorizza, ottimizza ed esegue le richieste
TM: coordina le richieste
L&RM: protegge dai guasti
CM: gestisce accessi concorrenti



Riassumiamo:

- Un DBMS è un sistema software (complesso!) che gestisce grandi quantità di dati persistenti e condivisi, garantendone l'integrità e la sicurezza mediante l'esecuzione coordinata delle richieste, la protezione da malfunzionamenti e la realizzazione di una politica degli accessi
- Un modello dei dati è una collezione di concetti che vengono utilizzati per descrivere i dati, le loro associazioni, e i vincoli che questi devono rispettare
- Una base di dati si compone di uno schema, che ne descrive la struttura logica, e di un'istanza, che consiste dei dati memorizzati
- Mediante un'organizzazione a 3 livelli, un DBMS permette di ottenere gradi di indipendenza fisica e logica dei dati