Basi di Dati

Alex Narder

October 15, 2022

1 Contenuti del corso

- Sistemi per Basi di Dati
- Modello dei dati
- Progettazione di Basi di dati
- Modello relazionale
- SQL per la definizione, la manipolazione e la consultazione dei DB
- Teoria della normalizzazione
- Amministazione di una basi di dati
- Programmazione di applicazioni che utilizzano basi di dati
- Applicazione: Flask + SQLAlchemy
- NoSQL: Modelli di rappresentazione dell'informazione diversi da quello relazionale

2 Introduzione

Le **basi di dati** sono ovunque, le troviamo alla base di sistemi informativi aziendali, sistemi informativi territoriali, applicazioni internet, basi di dati distribuite, sistemi di supporto alle decisiomi, data mining ...

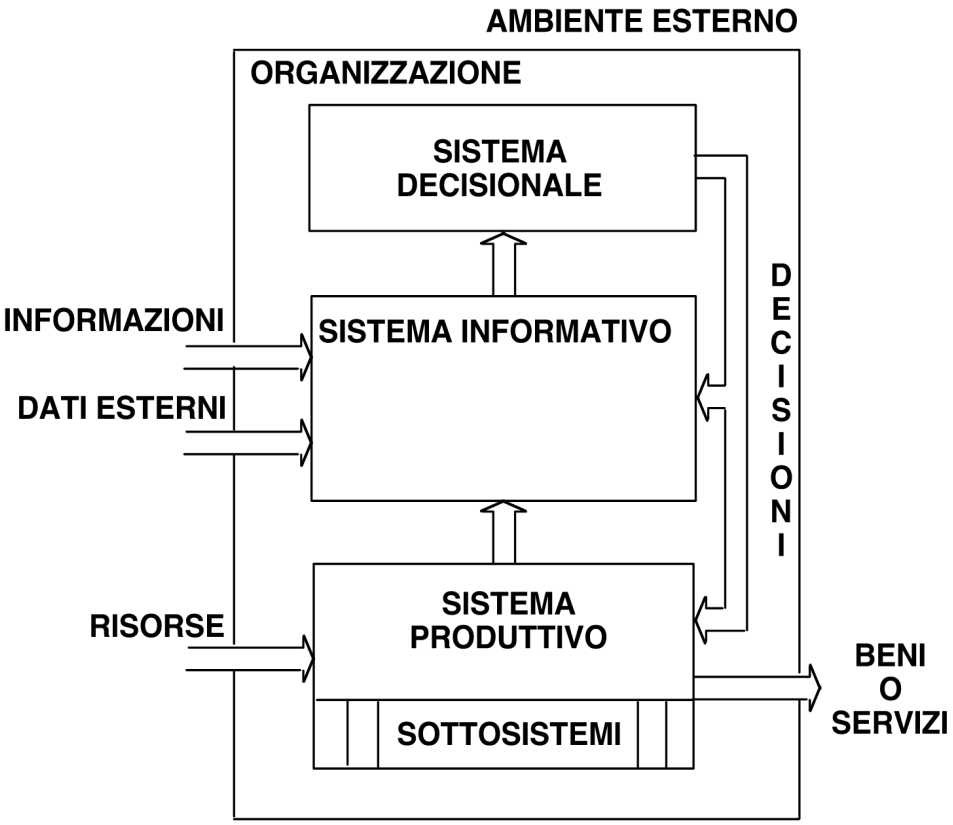
Area di sintesi

- Area di sintesi di competenze
  - linguaggi
  - ingegneria del software
  - algoritmi e strutture dati
  - reti
  - intelligenza artificiale
- Presenta aspetti modellistici, ingegneristici, teorici
- Pone interessanti problemi di ricerca

3 Oranizzazioni e sistemi informativi

• Le organizzazioni hanno bisogno di gestire le informazioni per realizzare le proprie attività.
• Un sistema informativo di un'organizzazione serve per la raccolta e acquisizione, l'archiviazione, conservazione l'elaborazione, trasformazione, produzione la distribuzione, comunicazione e lo scambio delle informazioni necessarie alle attività dell'organizzazione.

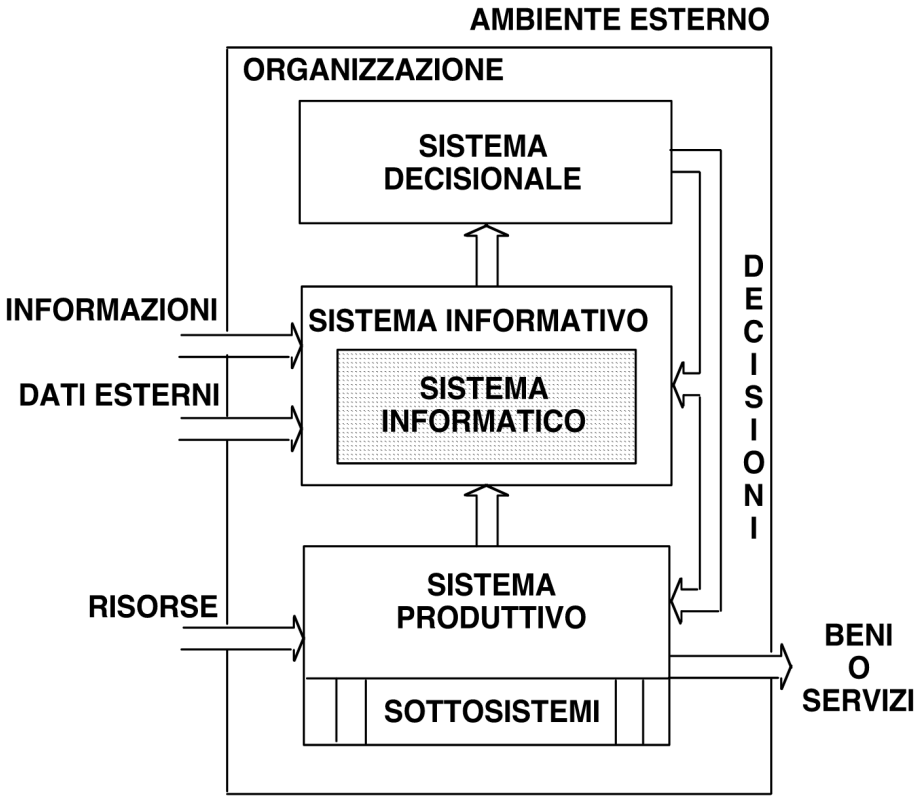
Sistema informativo nelle organizzazioni:



4 Sistemi Informatici

• Il **sistema informatico** è l'insieme delle tecnologie informatiche e della comunicazione (Information and Communication Technologies, ICT) a supporto delle attività di un'organizzazione.
• Il **sistema informativo automatizzato** è quella parte del sistema informativo in cui le informazioni sono raccolte, elaborate, archiviate e scambiate usando un sistema informatico.

Sistema informatico nelle organizzazioni:

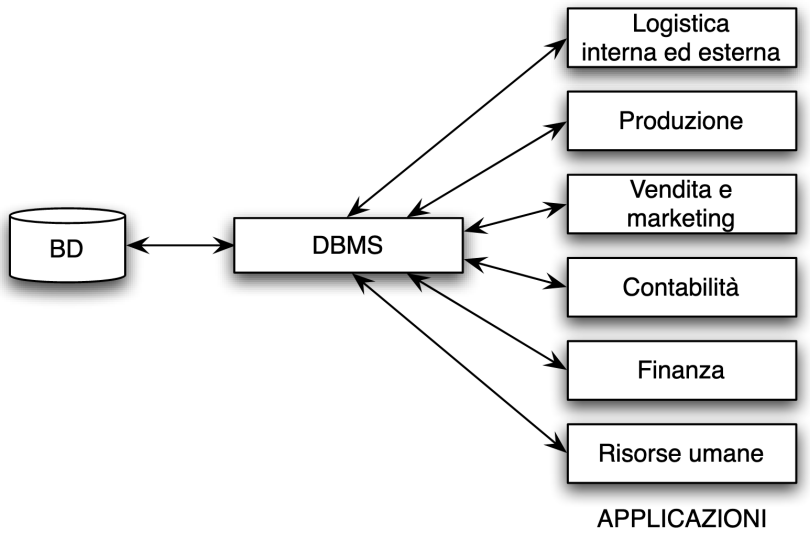


5 Informazioni e dati

• Nei sistemi informatici le **informazioni** vengono rappresentate in modo essenziale attraverso i dati.
• **Informazione**: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.
• **Dato**: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elementi di informazione costituiti da simboli che debbono essere elaborati.

6 Classificazione dei Sistemi Informatici

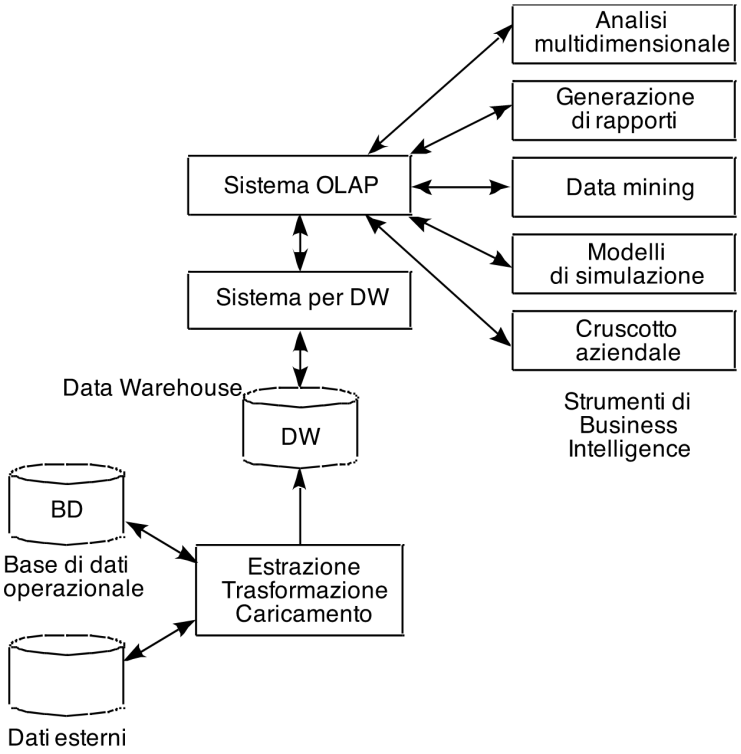
- **Sistemi Informatici Operativi**:
  - I dati sono organizzati in BD
  - Le applicazioni si usano per svolgere le classiche **attività strutturate e ripetitive** dell'azienda nelle aree amministrativa e finanziaria, vendite, produzione, risorse umane ecc. (calcolo paghe, emissione fatture, magazzino, ...)



Elaborazioni su BD: OLTP

- **OLTP**: Acronimo di **On-Line Transaction Processing**
- Tradizionale elaborazione di transazioni, che realizzano i processi operativi per il funzionamento di organizzazioni:
  - Operazioni predefinite e relativamente semplici
  - Ogni operazione coinvolge "pochi" dati
  - Dati di dettaglio, aggiornati
- Uso principale dei DBMS

- **Sistemi Informatici Direzionali**:
  - La direzione intermedia e alta necessitano di: **analisi storiche** e **produzione interattiva** di rapporti di sintesi
  - Le basi di dati operative risultano inadeguate: contengono solo **dati recenti** e le operazioni coinvolgono grandi quantità di dati o sono molto complesse e quindi rallenterebbero in modo inaccettabile le funzioni operative
  - I dati sono organizzati in **Data Warehouse (DW)**
  - Sono gestiti da un opportuno **sistema per analisi interattive dei dati**
  - Usano **applicazioni di business intelligente** come strumenti di supporto ai processi di controllo delle presentazioni aziendali e di decisione manageriale



Elaborazioni su DW: OLAP

- **OLAP**: Acronimo di **On-Line Analytical Processing**
- Uso principale dei **data warehouse**
- Caratteristiche
  - Operazioni complesse e casuali
  - Ogni operazione può coinvolgere moltissimi dati
  - I dati sono letti, ma non modificati
  - Dati aggregati, storici, anche non attualissimi

7 Sistemi per la gestione di Basi di Dati (DBMS)

7.1 Baso di dati (gestite da DBMS)

In generale una qualsiasi raccolta di informazioni permanenti gestite tramite un elaboratore elettronico è considerata una base di dati, ma per noi:
Una **Base di dati** è una suddivisa in due categorie:

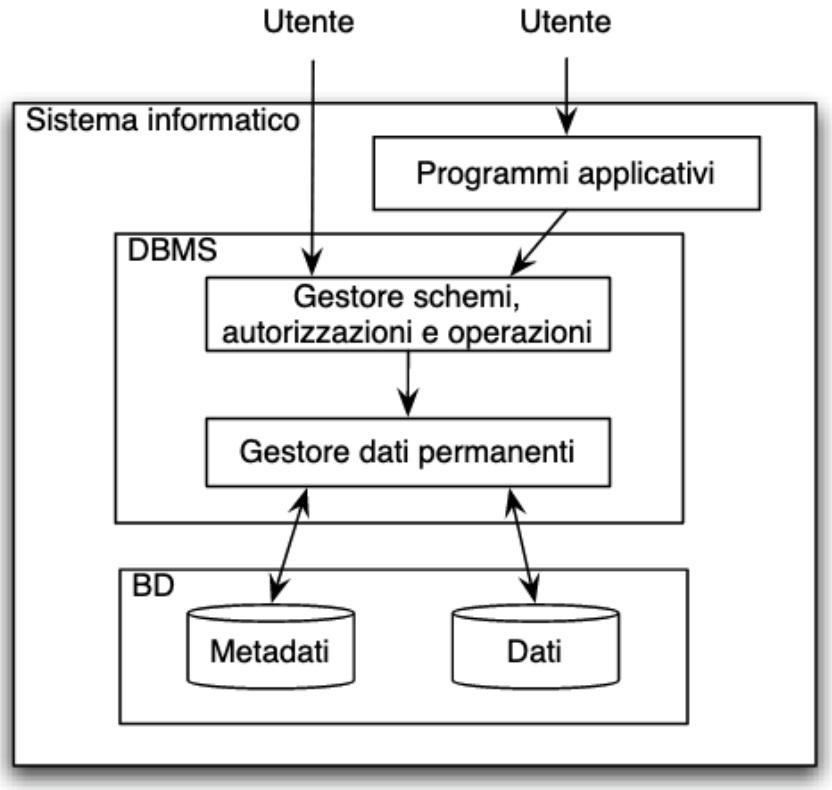
- **Metadati**: definiscono lo schema della BD, che descrive:
  - struttura dei dati
  - restrizioni sui valori ammissibili, quindi i **vincoli di integrità**
  - utenti autorizzati
- **Dati**: le rappresentazioni di certi fatti conformi alle definizioni dello schema, con le seguenti caratteristiche:
  - sono organizzati in **insiemi omogenei**, fra i quali sono definite delle **relazioni**. La loro struttura e le relazioni sono descritte usando il **modello dei dati** adottato. Per esempio una collezione di dati che contiene studenti, una che contiene classi, ecc. ecc.
  - sono **molti** rispetto ai metadati, e non possono essere gestiti in memoria temporanea
  - sono **permanenti**, continuano a esistere fino a quando non vengono rimossi
  - sono **utilizzabili contemporaneamente** da diversi utenti
  - sono **protetti** sia da accessi non autorizzati che da corruzione dovuta a guasti di harware o software
  - sono accessibili tramite **transazioni**, unità di lavoro atomiche che non possono avere effetti parziali, quindi devono andare tutte a buon esito, altrimenti falliscono tutte. Come se fosse una singola operazione, anche se in realtà sono più di una.

7.2 Sistemi per Basi di Dati -> DBMS

definizione: Un **DBMS (data Base Management System)** è un sistema centralizzato o distribuito che offre opportuni linguaggi e strumenti per:

- **definire lo schema** del DB (che va definito prima di creare dati, definito attraverso il modello dei dati del DBMS, interrogabile con le stesse modalità previste per i dati)
- scegliere le **strutture dati** per la memorizzazione dei Dati
- **memorizzare** i dati rispettando i vincoli definiti nello schema
- **recuperare e modificare** i dati interattivamente (tramite un linguaggio di interrogazione, anche detto **query language**)

Architettura dei DBMS centralizzati:



Esempio di Sessione con un DBMS Relazionale

- Il modello relazionale dei dati è il più diffuso fra i DBMS commerciali.

- Il meccanismo di astrazione fondamentale è la **relazione** (**tabella**)
  ~ insieme di record con campi di tipo elementare;

Nome	Matricola	Citta	AnnoNascita
Verdi	71523	Padova	1987
Rossi	76366	Dolo	1988
Zeri	71347	Venezia	1988

Studenti

- Lo schema specifica le tabelle
  - nome
  - struttura degli elementi (nome e tipo degli attributi).

- Definizione base di dati (schema vuoto)
  - `CREATE DATABASE EsempioEsami;`

- Definizione schema:
  -
```
CREATE TABLE Studenti (
        Nome char(8),
        Matricola int NOT NULL,
        Citta char(10),
        AnnoNascita int,
        PRIMARY KEY (Matricola) );
```
  -
```
CREATE TABLE ProveEsami (
        Materia char(5),
        Matricola int,
        Data char(6),
        Voto int,
        Lode char(1),
        PRIMARY KEY (Materia,Matricola) );
```

Nome	Matricola	Citta	AnnoNascita
Verdi	71523	Padova	1987
Rossi	76366	Dolo	1988
Zeri	71347	Venezia	1988

Studenti

ProveEsami

Materia	Matricola	Data	Voto	Lode
CN	71523	08.07.06	27	N
FIS	76366	08.07.07	26	N
BD	71523	28.12.06	30	S

- Inserzione dati:
  - ```
    INSERT INTO ProveEsami
    VALUES ('BD', 71523,'28.12.06',30, 'S');
    ```
- Interrogazione:

```
SELECT Matricola
FROM   ProveEsami
WHERE  Materia = 'BD' AND Voto = 30;

Matricola

71523
```

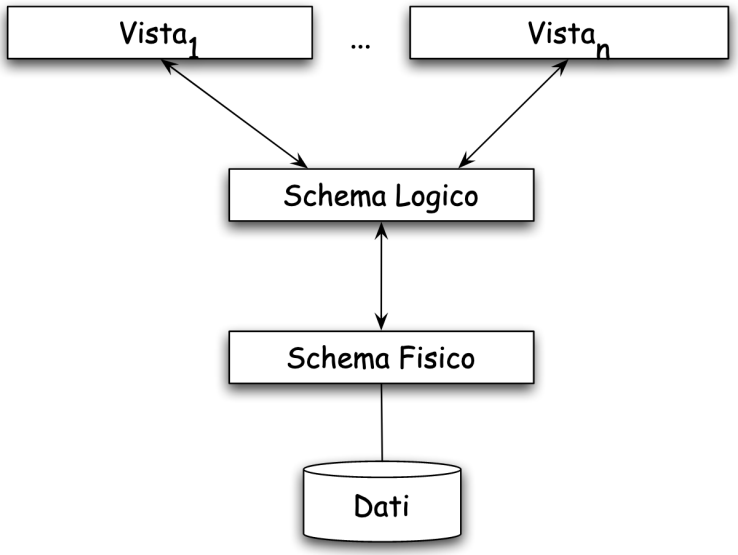
8 Funzionalità dei DBMS

- Linguaggio per la **definizione dei DB (DDL)**, **data definition language**
- Linguaggi per l'**uso dei dati (DML)**, **data manipulation language**
- Meccanismi per il **controllo dei dati**
- Strumenti per il **responsabile del DB**
- Strumento per lo **sviluppo delle applicazioni**
```

9 DDL: Definizione dei DB

Per definire la base di dati si usa il DDL, la descrizione del DB è indipendente dalle applicazioni che la usano.
Vogliamo che il DB abbia una sua vita indipendente nonostante le applicazioni che vanno ad usarla.

Ci sono 3 livelli di descrizione dei dati;
- livello di vista logica
- livello logico, qui si descrive la struttuta e i vincoli della base di dati (sono raccolti nello schema logico)
- livello fisico



9.1 Livello vista logica

Descrive come deve apparire la struttura della base di dati ad una certa applicazione (**schema esterno o vista**).

- Esempio:
  - `InfCorsi(IdeC char(8), Titolo char(20), NumEsami int)`

```
CREATE VIEW InfCorsi (IdeC, Titolo, NumEsami) AS
            SELECT IdeC,
                   Titolo,
                   COUNT(*)
            FROM Corsi NATURAL JOIN Esami
            GROUP BY IdeC, Titolo;
```

9.2 Livello logico

Schema logico: Descrive la **struttura degli insiemi di dati e delle relazioni** fra questi, secondo un certo modello dei dati, senza nessun riferimento alla loro organizzazione fisica nella memoria permanente.

- Esempio:

```
Studenti(Matricola int, Nome char(20), Login char(8),
         AnnoNascita int, Reddito real )

Corsi(IdeC char(8), Titolo char(20), Credito int )

Esami(Matricola int, IdeC char(8), Voto int )
```

... realizzata in SQL con **`CREATE TABLE`**.

9.3 Livello fisico

Descrive lo **schema fisico** o **interno**: come vanno organizzati fisicamente i dati nelle memorie permanenti, strutture dati ausiliarie per l'uso (es. indici). Un indice è quella "cosa" che mi permette di recuperare in maniera veloce un'informazione.

- Esempio:
  - `Corsi` e `Esami` organizzate in modo seriale
  - `Studenti` organizzata in modo sequenziale con indice `Indice` su `Matricola`

```
CREATE INDEX Indice ON Studenti(Matricola);
```

9.4 Indipendenza fisica e indipendenza logica:

- **Indipendenza fisica**: i programmi applicativi non devono essere modificati in seguito a modifiche dell'organizzazione fisica dei dati. Quindi se io vado a cambiare qualcosa non devo modificare i programmi applicativi.
  - **Esempio**: Se si deve risalire spesso agli studenti che hanno sostenuto un particolare esame:

```
CREATE INDEX IndiceIdeC ON Esami(IdeC);
```

- **Indipendenza logica**: i programmi applicativi non devono essere modificati in seguito a modifiche dello schema logico.
  - Esempio: per suddividere la collezione degli studenti in part-time e full-time:

```
CREATE TABLE StudentiFull (...);
CREATE TABLE StudentiPart (...);

CREATE VIEW Studenti AS
   SELECT * FROM StudentiFull
   UNION
   SELECT * FROM StudentiPart;
```

10 DML: linguaggi per l'uso dei dato

Un DBMS prevede varie modalità d'uso per soddisfare le esigenze delle diverse **categorie di utenti** che possono accedere alla base di dati:

- **Utenti delle applicazioni**, nessun tipo di accesso elaborato
- **Utenti non programmatori**, hanno bisogno di agire in modo più evoluto rispetto agli utenti delle app.
- **Programmatori delle applicazioni**, utenti che hanno bisogno di operare in maniera intensiva nei DB.

- **Utenti non programmatori**:
  - Interfaccia grafica per accedere ai dati in modo semplice.
  - Linguaggio per interrogazione
- **Utenti programmatori**:
  - Linguaggio integrato (dati e DML): Linguaggio pensato per SQL, i comandi SQL sono controllati staticamente dal traduttore ed eseguiti dal DBMS
  - Linguaggio convenzionale + funzioni di libreria predefinita: Linguaggio convenzionale che usa delle funzioni di una libreria predefinita per usare SQL. I comandi SQL sono stringhe passate come parametri alle funzioni che poi vengono controllate dinamicamente dal DBMS prima di eseguirle.
  - Linguaggio che ospita l'SQL (SQL embedded): Linguaggio convenzionale esteso con un nuovo costrutto per marcare i comandi SQL. Occorre un **pre-compilatore** che controlla i comandi SQL, li sostituisce con chiamate a funzioni predefinite e genera un programma nel linguaggio convenzionale + funzioni di libreria

11 Controllo dei Dati

11.1 DBMS: controllo dei dati

Sono dei meccanismi offerti per garantire le seguenti proprietà:

- **Integrità**, mantenimento delle proprietà specificate in modo dichiarativo nello schema (vincoli d'integrità)
- **Sicurezza**, protezione dei dati da usi non autorizzato:

- restrizione dell'accasso ai **soli utenti autorizzati**
- **limitazione delle operazioni esegiubili**

- **Affidabilità**, protezione dei dati da:
  - **interferenze** indesiderate dovute all'accesso concorrente ai dati da parte di più utenti
  - **malfunzionamenti hardware o software** (fallimenti di transazione, fallimenti di sistema, disastri)

11.1.1 Protezione da interferenze indesiderate tra accessi concorrenti ai dati

- Basterebbe impedire l'inizio di una transazione prima che un'altra inizio
- scheduling dei singoli passi di ciascuna transazione in T1, ..., Tn che garantisca che l'effetto complessivo sarebbe ottenibile eseguendo le transazioni isolatamente in qualche ordine

Es:

T1	T2
leggi(SALDO)	leggi(SALDO)
SALDO := SALDO + 100	SALDO := SALDO - 100
scrivi(SALDO)	scrivi(SALDO)

11.1.2 Protezione da malfunzionamenti hardware o software

- **fallimenti di transazione**: dovuta a una situazione prevista dall'applicazione o a eventi imprevisti, come la violazione di vincoli di integrità o accessi non autorizzati
- **fallimenti di sistema**: dovuti ad un'anomalia HW o SW dell'unità centrale o di una periferica, che determina l'interruzione di tutte la transazioni attive e la perdita del contenuto della memoria temporanea
- **disastri**: danni alla memoria permanente

11.2 DBMS: Controllo dei dati: Transazioni

Una **transazione** è una sequenza di azioni di lettura e scrittura in memoria permanente e di elaborazioni di dati in memoria temporanea, con le seguenti proprietà:

- **Atomicità**, le transazioni che terminano prematuramente sono trattate dal sistema come se non fossero mai iniziate, quindi eventuali effetti sul DB sono annullati
- **Serializzabilità**, nel caso di esecuzioni concorrenti di più transazioni, l'effetto complessivo è quello di una esecuzione seriale
- **Persistenza**, Le modifiche sulla base di dati di una transazione terminata normalmente sono permanenti, cioè non sono alterabili da eventuali malfunzionamenti

12 Strumenti per l'amministazione

12.1 DBMS: strumenti per l'amministrazione

- Linguaggio per la definizione e la modifica degli schemi della base di dati - logico, esterno, e fisico
- Strumenti per il controllo e messa a punto del funzionamento del sistema.
- Strumenti per stabilire i diritti di accesso ai dati
- Strumenti per ripristinare la base di dati in caso di malfunzionamenti di sistemi o disastri

12.2 riepilogo dei vantaggi dei DBMS

- indipendenza fisica e logica
- gestione efficiente dei dati
- integrità e sicurezza dei dati
- accessi interattivi, concorrenti e protetti da malfunzionamenti

- amministrazione dei dati
- riduzione dei tempi di sviluppo delle applicazioni

12.3 riepilogo degli svantaggi dei DBMS

- Possono essere costosi e complessi da installare e mantenere in esercizio.
- Richiedono personale qualificato (se si tratta di personale esterno, aumenta la dipendenza da ditte esterne)
- Le applicazioni sviluppate possono essere trasferite con difficoltà su sistemi diversi se vengono usati linguaggi troppo "legati" al DBMS usato
- La riduzione dei costi della tecnologia e i possibili tipi di DBMS disponibili sul mercato facilitano la loro diffusione

13 Progettazione e Modellazione

- Progettare una base di dati significa progettare la:
  - struttura dei dati
  - applicazioni
- La progettazione della struttura dei dati è l'attività fondamentale
- Richiede di specificare un modello della realtà di interesse (**universo del discorso**) quanto più possibile fedele
- Per questo ci concentreremo sulla modellazione:
  - cosa significa definire un modello?
  - cosa si modella?
  - come si modella (quale formalismo)?

14 Modello dei Dati e Progettazione

14.1 Modelli Informatici

Un **modello astratto** è la rappresentazione formale di idee e conoscenze relative a un fenomeno.

- Aspetti di un modello:
  - il modello è la **rappresentazione di certi fatti**;
  - la rappresentazione è data con un **linguaggio formale**;
  - il modello è il risultato di un **processo di interpretazione**, guidato dalle idee e conoscenze possedute dal soggetto che interpreta.

14.2 Progettazione di un DB

Ogni una di queste fasi è incentrata sulla modellazione.

15 Modellazione concettuale

15.1 Aspetti del problema

- Quale conoscenza del dominio nel discorso si rappresenta?
  ()aspetto ontologico)
- Con quali meccanismi di astrazione si modella?
  (aspetto logico)
- Con quale linguaggio formale si definisce il modello?
  (aspetto linguistico)
- Come si procede per costruire un modello?
  (aspetto pragmatico)

16 Cosa si modella?

- **Conoscenza concreta**: I fatti
- **Conoscenza astratta**: Struttura e vincoli sulla conoscenza concreta
- **Conoscenza procedurale**: Le operazioni di base e le operazioni degli utenti
- **Comunicazioni**: Come si comunicherà con il sistema informatico

16.1 Conoscenza concreta

• Fatti specifici che si vogliono rappresentare: - **le entità** con le loro **proprietà** - le **collezioni** di identità omogenee - le **associazioni** fra entità

16.1.1 entità e proprietà

• Le **entità** sono ciò di cui interessa rappresentare alcuni fatti (**o proprietà**): oggetti concreti, oggetti astratti ed eventi.
Come ad esempio un libro, una descrizione bibliografica o un prestito.

• Le **proprietà** si distinguono dalle entità poichè sono fatti che interessano solo perchè descrivono caratteristiche di determinata identità

come ad esempio: indirizzo che interessa solo in quanto indirizzo di un utente.

nota: un'entità non coincide con i valori delle sue proprietà.

• Una **proprietà** è una coppia <Attributo, valore di un certo tipo>.

• Classificazione delle proprietà:
- **atomica o strutturata**
- **univoca / multivalore**
- **totale / parziale**

• Esempi:
- nome (atomica, univoca, totale)
- residenza = [indirizzo, cap, città] (strutturata)
- recapiti telefonici (multivalore, parziale)

16.1.2 Collezioni di entità

• **Tipi di entità**: ogni entità ha un **tipo** che ne specifica la natura (identifica caratteristiche: proprietà e dominio relativo).

es: Antonio ha tipo **Persona** con proprietà:
– $Nome$: $string$
– $Indirizzo$: $string$

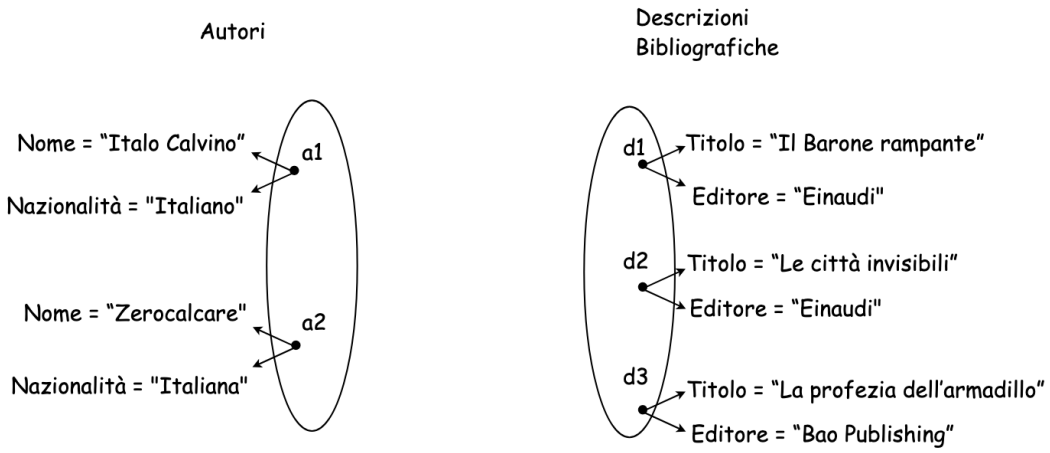
es2:

Tipo Entità	Proprietà
Studente	Nome, AnnoNascita, Matricola, e-mail, ...
Esame	Materia, Candidato, Voto, Lode, ...
Auto	Modello, Colore, Cilindrata, Targa, ...
Descrizione bibliografica	Autori, Titolo, Editore, Anno, ...

• **Collezione (classe)**: un insieme variabile nel tempo di entità omogenee, ovvero dello stesso tipo.

es: **Studenti**: insieme di tutti gli studenti nel dominio del discorso

es2:



16.1.3 Scelta delle entità e delle proprietà

Certi fatti possono essere interpretati come proprietà in certi contesti e come entità in altri, per esempio:

• **Descrizione bibliografica** con proprietà; **Autori, Titolo, Editore, Anno**

oppure

• **Autore** con proprietà **Nome, Nazionalità, AnnoNascita,** ...

• **Editore** con proprietà **Nome, Indirizzo, e-mail,** ...

• **Descrizione** biblioteca con proprietà **Titolo, Anno, ...**

16.1.4 Gerarchie

• Spesso le collezioni di enrità sono organizzate in una gerarchia di **specializzazione/generalizzazione** (si parla anche di **sottoclassi** e **superclassi**).

• Es: nel DB della biblioteca la collezione degli *Utenti* può essere considerata una generalizzazione di *Studenti* e *Docenti*

• Due importanti caratteristiche delle gerarchie:
- **ereditarietà** proprietà
- **inclusione**: se la collezione C1 specializza C2, gli elementi di C1 sono un sottoinsieme degli elementi di C2.

Problema: scelta delle sottoclassi →

- la classe degli **studenti** universitari è una generalizzazione delle classi:
  - **matricole** e **laureandi**,
  - **studenti in corso** e **studenti fuori corso**,
  - **studenti veneziani** e **studenti fuori sede**,

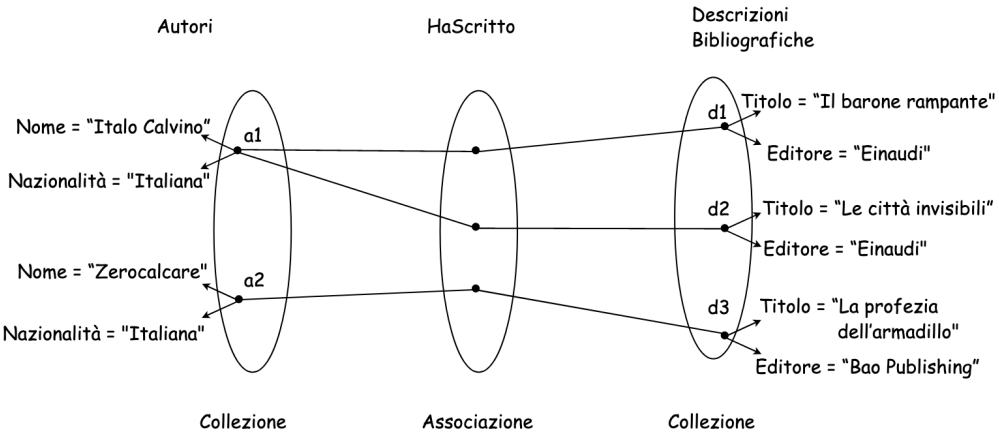
- **Attenzione**:
  "un laureando **è uno** (is-a) studente"
  è diverso da
  "Mario **è uno** studente!"

16.1.5 Associazioni

- Un'**istanza di associazione** è un fatto che correla due o più entità, stabilendo un legame logico tra di loro.
  - la descrizione bibliografica con titolo "*Basi di Dati*" riguarda il documento con collocazione "$D3 - 55 - 2$"
  - l'utente "*Tizio*" ha in prestito una copia della "*DivinaCommedia*"

- Un'**associazione** R(X,Y) fra due collezioni di entità X e Y è un insieme di istanze di associazione tra elementi di X e Y, che varia in generale nel tempo. Il prodotto cartesiano (X * Y) è detto **dominio** dell'associazione.

- esempio:



- Tipi di associazione: Un'associazione è caratterizzata dalle seguenti proprietà strutturali:
  - **molteplicità** (o cardinalità)
  - **totalità**

16.1.6 Tipi di associazione: Molteplicità

- **Vincolo di univocità:**
  - Un'associazione R(X,Y) è **univoca da X a Y** se per ogni elemento x di X esiste al più un elemento di Y che è associato ad x; se non vale questo vincolo, l'associazione è **multivalore da X a Y**.

- **Cardinalità:**
  - **R(X,Y) è (1:N)** se essa è multivalore da X a Y ed univoca da Y a X
  - **R(X,Y) è (N:1)** se essa è univoca da X a Y e multivalore da Y a X

- **R(X,Y) è (N:M)** se essa è multivalore da X a Y e multivalore da Y a X
- **R(X,Y) è (1:1)** se essa è univoca su da X a Y e univoca da Y a X

- esempi:
  - `Frequenta(Studenti, Corsi)`

    ha molteplicità **`(N:M)`**,
  - `Insegna(Professori, Corsi)`

    ha molteplicità **`(1:N)`**,
  - `SuperatoDa(Esami, Studenti)`

    ha molteplicità **`(N:1)`**,
  - `Dirige(Professori, Dipartimenti)`

    ha molteplicità **`(1:1)`**.

16.1.7 Tipi di associazione: Totalità

- **Vincolo di totalità**: Un'associazione R(X,Y) è **totale** da X a Y se per ogni elemento x di X esiste almeno un elemento di Y che è associato ad x; se non vale questo vincolo, l'associazione è **parziale** da X a Y.

  Esempio: $Insegna(Professori,\ Corsi)$ è totale su $Corsi$ in quanto non può esistere un corso del piano di studi senza il corrispondente docente che lo tiene, parziale su $Professori$, in quanto un professore potrebbe non tenere corsi.

- esempi:

Tipi di associazioni fra Persone e Città:

`NataA(Persone, Città)`

ha cardinalità `(N:1)`, totale su Persone e parziale su Città

`HaVisitato(Persone, Città)`

ha cardinalità `(N:M)`, parziale su `Persone` e `Città`

`ÈSindacoDi(Persone, Città)`

ha cardinalità `(1:1)`, parziale su `Persone` e `Città`

16.2 Conoscenza Astratta

16.2.1 Vincoli di Integrità

16.2.2 Fatti derivabili

17 Come si modella?