

Basi di Dati

Alex Narder

October 26, 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
- Amministrazione di una base 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 decisioni, data mining ...

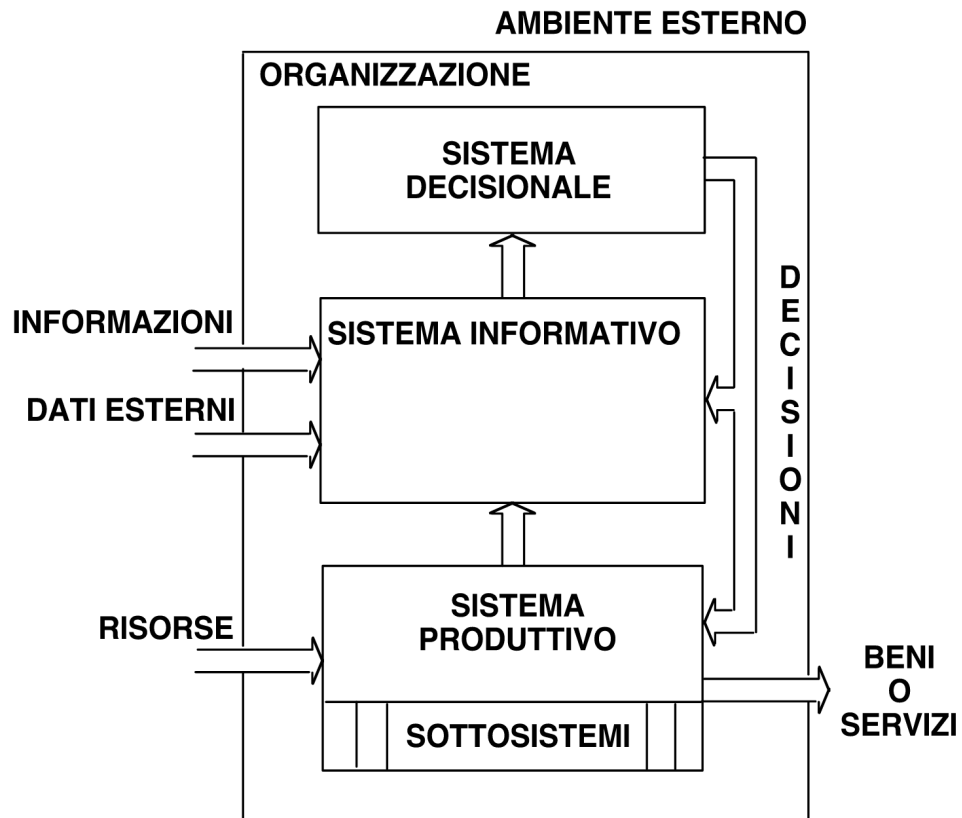
Area di sintesi

- 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 Organizzazioni 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:

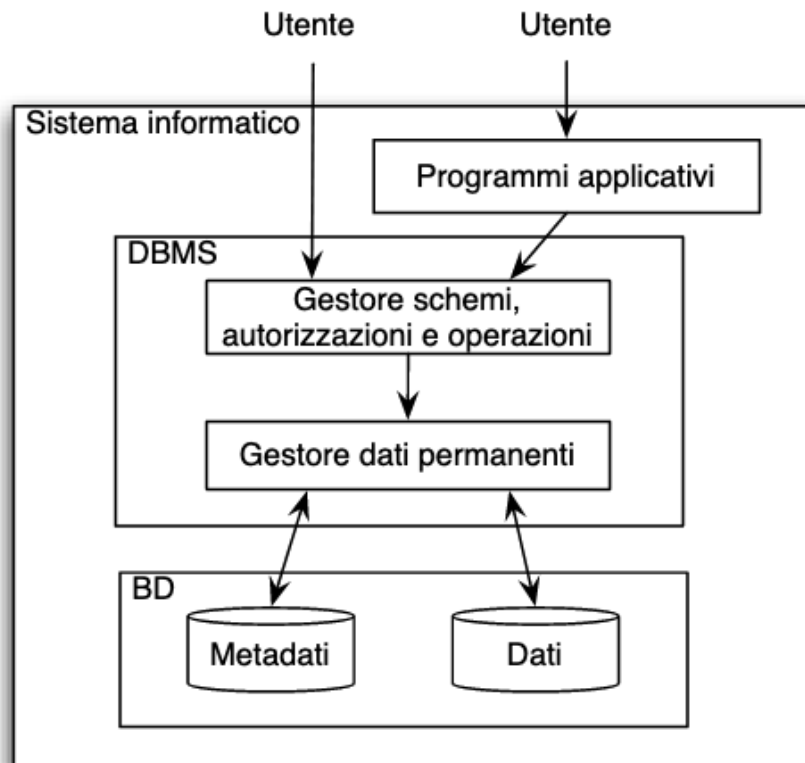
- **Metadati:** definiscono lo schema della BD, che descrive:
 - struttura dei dati
 - restrizioni sui valori ammissibili, quindi i **vincoli di integrità**
 - utenti autorizzati
- **Dati:** le rappresentazioni di certi fatti conformi alle definizioni dello schema, con le seguenti caratteristiche:
 - 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 hardware 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 struttura 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'accesso ai **soli utenti autorizzati**
- **limitazione delle operazioni eseguibili**
- **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 inizi
- scheduling dei singoli passi di ciascuna transazione in T_1, \dots, T_n che garantisca che l'effetto complessivo sarebbe ottenibile eseguendo le transazioni isolatamente in qualche ordine

Es:

T1 leggi(SALDO) $SALDO := SALDO + 100$ scrivi(SALDO)	T2 leggi(SALDO) $SALDO := SALDO - 100$ 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 le 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'amministrazione

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*

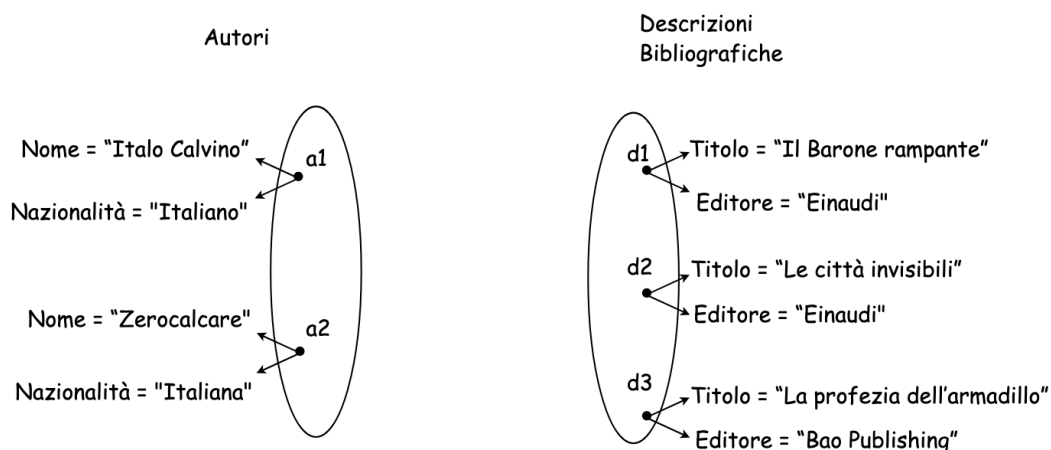
es2:

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 entità 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.

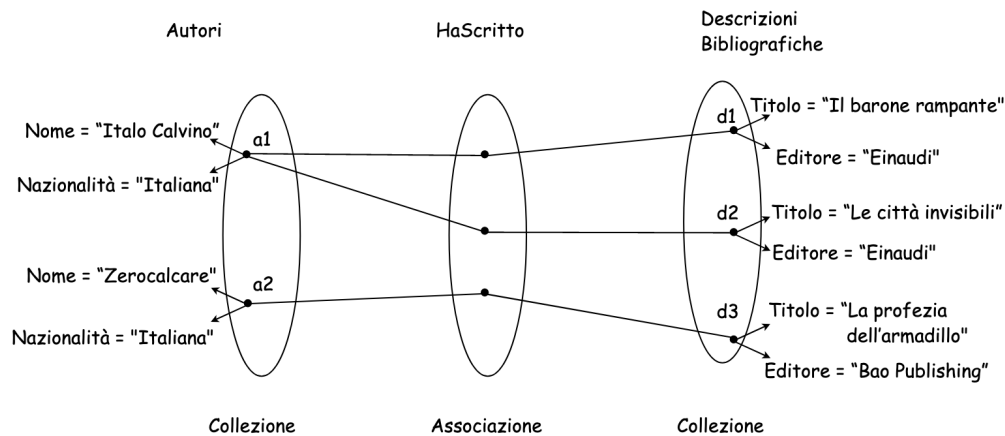
Problema: scelta delle sottoclassi →

- 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

Fatti generali che descrivono:

- la **struttura della conoscenza concreta**
 - **collezioni**: nomi, tipo degli elementi (nome, dominio, caratteristiche delle proprietà)
 - **associazioni**: nomi, collezioni correlate, proprietà strutturali
- restrizioni sui **valori possibili** della conoscenza concreta e sui modi in cui esse possono **evolvere** nel tempo (vincoli di integrità)
- regole per **derivare** nuovi fatti da altri noti

16.2.1 Vincoli di Integrità

Ce ne sono due tipi:

- **Vincoli di integrità statici**: definiscono delle condizioni sui valori della conoscenza concreta che devono essere soddisfatte indipendentemente da come evolve l'universo del discorso

Es: *Stipendio* deve essere positivo, *Matricola* è una chiave, ...

- **Vincoli di integrità dinamici**: definiscono delle condizioni sul modo in cui la conoscenza concreta può evolvere nel tempo

Es: *DataNascita* non può cambiare, uno studente iscritto ad un corso di laurea non può iscriversi nuovamente, ...

16.2.2 Fatti derivabili

Fatti derivabili:

- L'età di una persona, ricavabile per differenza fra l'anno attuale e il suo anno di nascita
- La media dei voti degli esami superati da uno studente, derivabile attraverso dei calcoli

17 Come si modella?

17.1 Modello dei dati a oggetti

- Un **modello dei dati** è un insieme di meccanismi di astrazione per descrivere la struttura della conoscenza concreta (**schema**)
- Uno schema verrà dato usando una notazione grafica, variante dei cosiddetti **diagrammi ER** (Entità-Relazione)
- Nozioni fondamentali:
 - Oggetto, Tipo di oggetto, Classe
 - Ereditarietà, Gerarchia fra tipi, Gerarchia fra classi

17.2 Questioni terminologiche

dominio del discorso	modello informatico
entità	oggetto (entity instance)
tipo entità	tipo oggetto (entity type)
collezione	classe (entity)
associazione	associazione o relazione

17.3 Oggetto

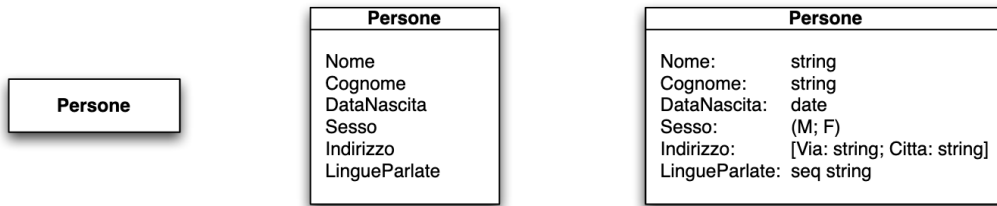
- Ad ogni **entità** del dominio corrisponde un **oggetto** del modello informatico
- Un oggetto è un'entità software con **stato, comportamento e identità**
 - Lo **stato** è modellato da un insieme di **costanti o variabili** con valori di qualsiasi complessità
 - Il **comportamento** è modellato da un insieme di procedure locali (con parametri) chiamate **metodi**
 - L'**identità** è associata all'oggetto dalla creazione e non viene modificata da aggiornamenti dello stato
- Un oggetto può rispondere a richieste, dette **messaggi**, restituendo valori memorizzati nello stato o calcolati con una procedura locale

17.4 Tipo Oggetto

- Il primo passo nella costruzione di un modello consiste nella **classificazione delle entità del dominio** con la identificazione di collezioni omogenee e la definizione dei **tipi degli oggetti** che le rappresentano
- Un **tipo oggetto** definisce:
 - componenti dello stato
 - metodi
- Noi useremo solo un paio di concetti elementari del modello a oggetti
- Oggetto:
 - identità
 - stato (insieme di attributi)
- no metodi, no incapsulamento, ecc.

17.5 Classi

Una classe è un insieme di oggetti dello stesso tipo, modificabile con operatori per includere o estrarre elementi dall'insieme, associabile a vincoli di integrità.

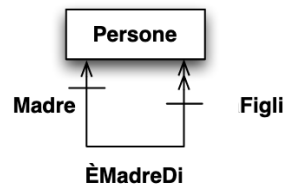
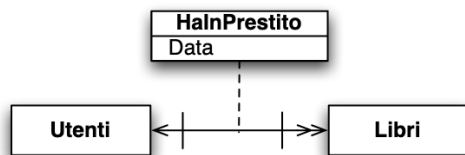
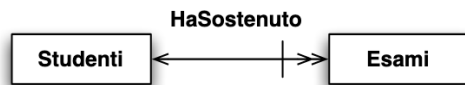


Una classe *Persone* a diversi livelli di specifica

- I tipi degli attributi possono essere:
 - **primitivi** (int, bool, string, date, real)
 - **non primitivi**
- **Tipi non primitivi:** ottenuti applicando i seguenti operatori ad altri tipi:
 - **tipo record;** [A1:T1; ... , An:Tn]
 - **tipo enumerazione;** [Val1; ... ; Valn]
 - **tipo sequenza;** seq T

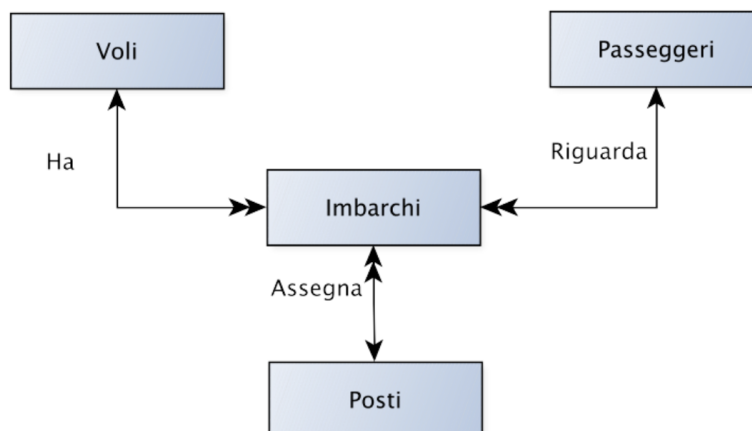
17.6 Associazioni

- Le associazioni si modellano con un costrutto apposito
- Le associazioni possono avere delle **proprietà**
- Le associazioni possono essere **ricorsive**
- Associazioni n-arie



17.7 Associazioni n-arie

- **Esempio:** Si vuole rappresentare l'associazione tra Voli, Passeggeri e Posti. Per ogni volo, al momento dell'imbarco, viene assegnato un posto a ciascun passeggero.



17.8 Descrizione di un caso

Si vogliono modellare alcuni fatti riguardanti una biblioteca universitaria:

- le **descrizioni bibliografiche** dei libri, opere con un solo volume ...
- i **termini del thesaurus** (parole chiave)
- le **copie dei libri** disponibili che corrispondono ad una descrizione bibliografica
- gli **autori** dei libri
- gli **utenti** della biblioteca
- i **prestiti** in corso

Le Descrizioni Bibliografiche dei libri a volume unico, sia già disponibili che quelli ordinati ma non ancora consegnati alla biblioteca, sono caratterizzati dal Codice ISBN (International Standard Book Number) che le identifica, titolo del libro, autori, editore, anno di pubblicazione e termini che le descrivono.

Degli autori dei libri interessano il codice fiscale, il nome e cognome, la nazionalità e la data di nascita.

I libri, disponibili in una o più copie ognuna identificata da un Codice, una stringa con la loro collocazione e numero.

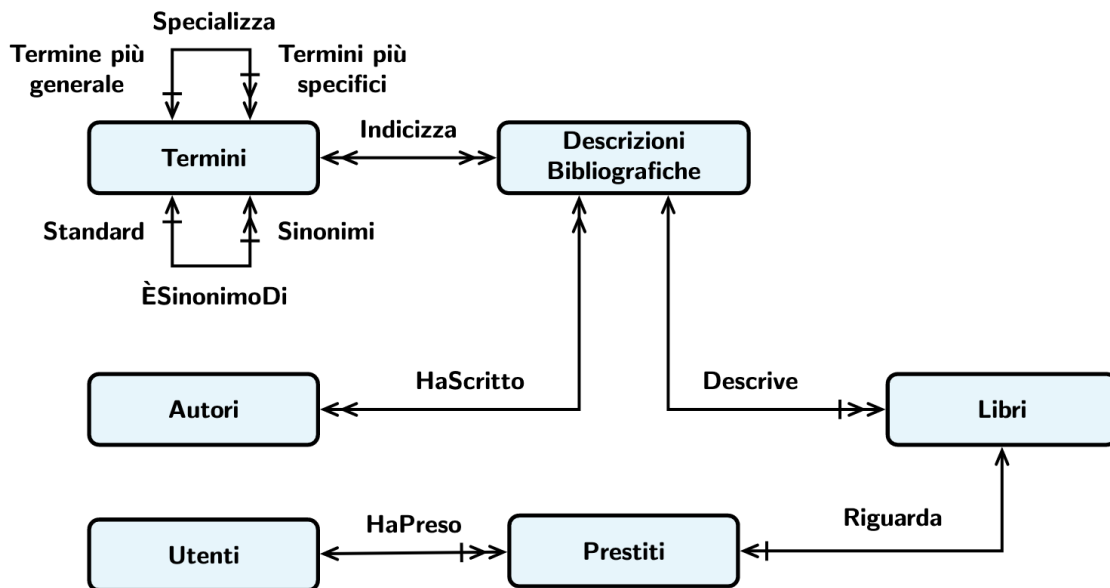
Quando un utente prende un libro in prestito, si registrano i dati dell'utente, se non sono già presenti, la data del prestito e la data di restituzione. Di un utente interessano il codice fiscale, il nome, il cognome, l'indirizzo e i recapiti telefonici. Un utente può avere più libri in prestito. I dati su un prestito interessano fino al momento della restituzione del libro.

Il thesaurus è un insieme di termini, e di associazioni fra di loro, che costituiscono il lessico specialistico da usare per descrivere il contenuto dei libri. Fra i termini del thesaurus interessano le seguenti relazioni, fra le tante possibili:

- **Preferenza**, per rimandi da termini standard a termini non standard e viceversa. Per esempio:

- Elaboratore **Standard** (vedi) Calcolatore;
- Calcolatore **Sinonimi** (UsatoPer) Elaboratore, Calcolatore, Stazione di lavoro
- **Gerarchia**, per mettere in evidenza il rapporto specificità-generalità tra due termini. Per esempio:
 - Felino **PiùSpecifico** Gatto Leone Tigre;
 - Gatto **PiùGenerale** Felino

Esempio di Biblioteca:



17.9 Gerarchia tra tipi oggetto

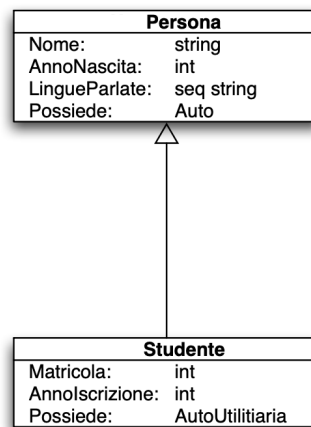
- Relazione di **sottotipo** \models , tra i tipi oggetto
- Se **t** è sottotipo di **t'** ($t \models t'$)
 - gli elementi di tipo **t** possono essere usati in ogni contesto in cui possano apparire elementi di tipo **t'** (**sostituibilità**)
- In particolare:

- gli elementi di t hanno tutti gli attributi degli elementi di t'
- per ogni attributo A in t' , il suo tipo in t è sottotipo del suo tipo in t'

17.10 Ereditarietà

- **Ereditarietà (inheritance)**: permette di definire un tipo oggetto a partire da un altro "per differenza"
 - aggiunta di attributi
 - ridefinizione di attributi esistenti
- Normalmente l'ereditarietà tra tipo si usa solo per definire sottotipi (**ereditarietà stretta**); in questo caso:
 - gli attributi possono essere aggiunti
 - gli attributi possono essere ridefiniti solo specializzandone il tipo

tipi definiti per ereditarietà:

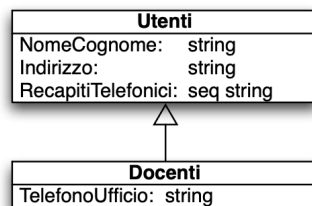


17.11 Gerarchia tra classi

Fra classi può essere definita una relazione di **sottoclasse** (talvolta della sottotipoinsieme), con le seguenti proprietà:

- riflessiva, antisimmetrica e transitiva (ordine parziale).
- Se C è sottoinsieme di B , allora le entità in C sono un sottoinsieme delle entità in B (**vincolo estensionale**).
- Se C è sottoclasse di B , allora il tipo delle entità in C è sottotipi del tipo degli elementi in B (**vincolo estensionale**).

esempio:

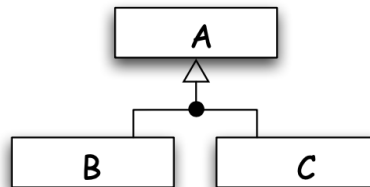


17.12 Vincoli su sottoclassi

Ci sono diversi tipi di vincoli:

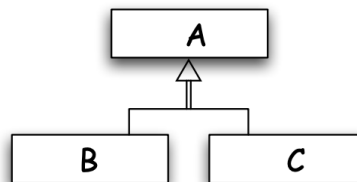
- **Vincolo di disgiunzione:**

$$B \cap C = \emptyset$$

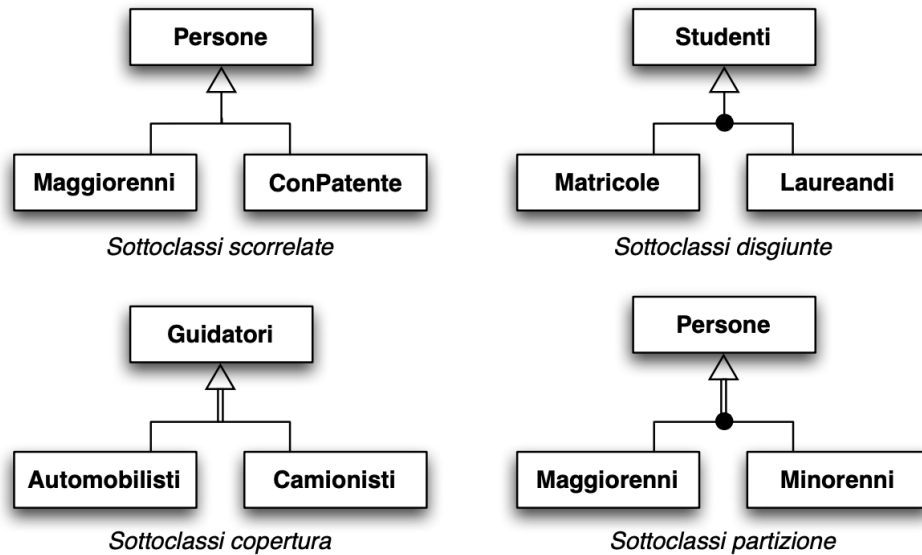


- **Vincolo di copertura**

$$B \cup C = A$$



esempi:



17.13 Come funzionano le frecce

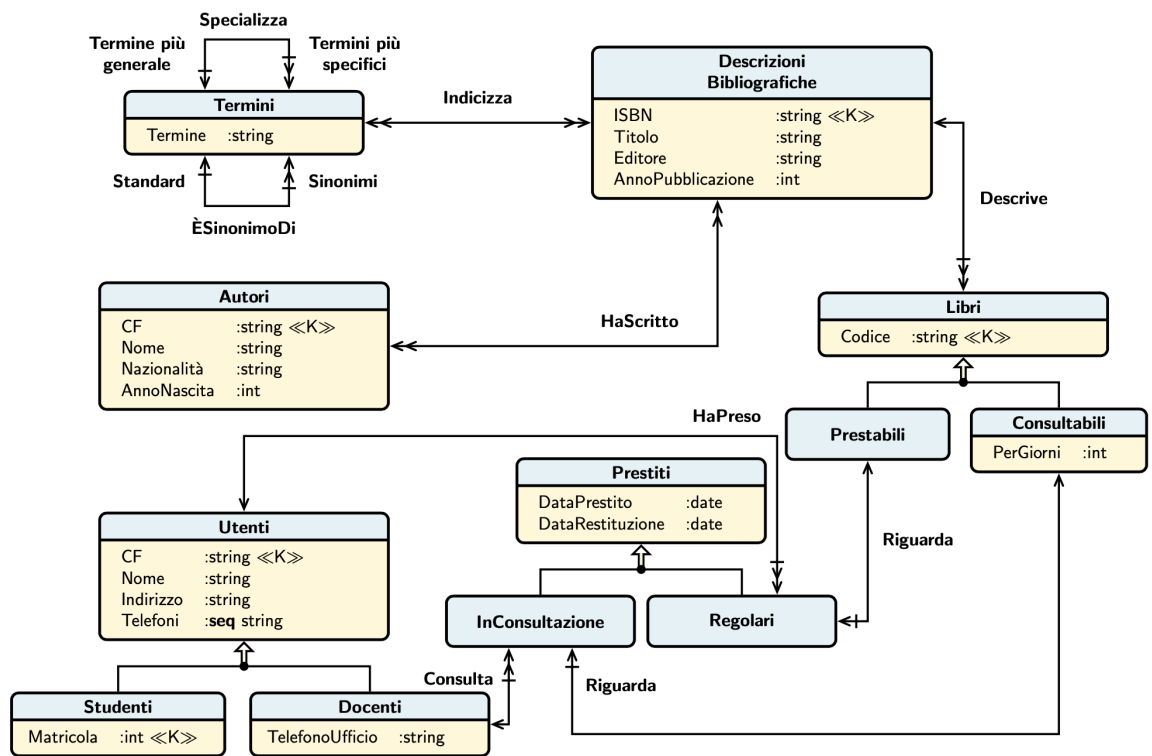
- **Freccia singola;** una corrispondenza.
- **Freccia multipla;** più corrispondenze.
- **Barretta prima delle frecce;** potrebbe non esserci corrispondenza.

Quando le frecce sono da entrambi i lati vuol dire che da entrambe le vie ci possono essere più corrispondenze. Ad esempio un conto corrente può avere più intestatari, e una persona può avere più conti corrente.

Per corrispondenza si intende che ci ha a che fare.

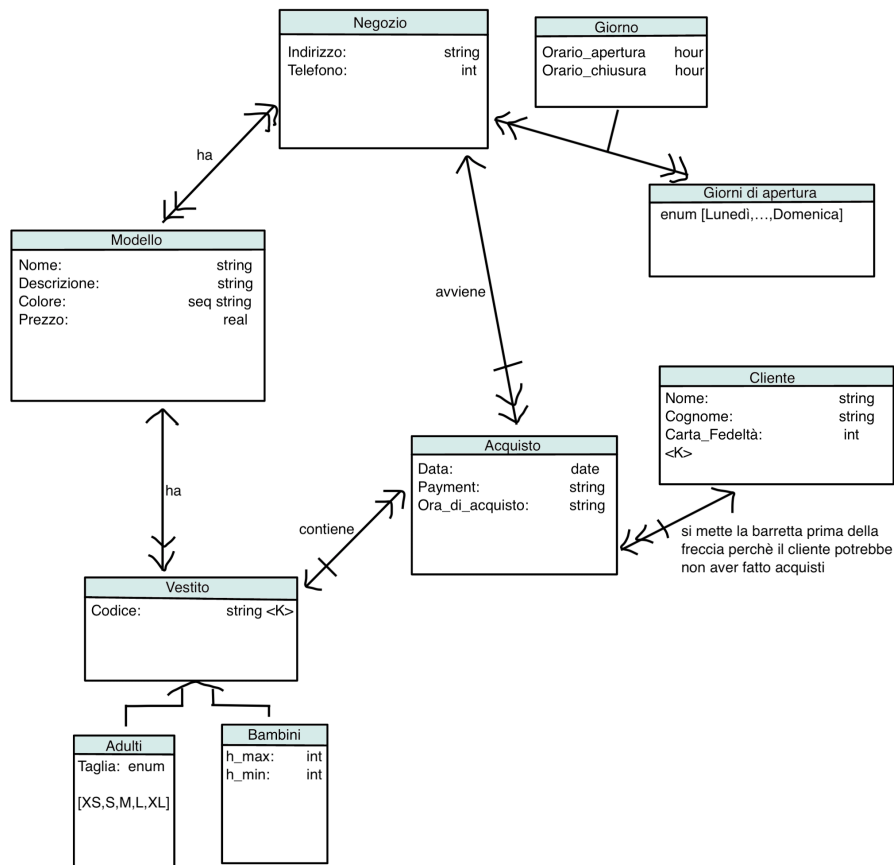
17.14 Modello concettuale

- Gli utenti possono essere studenti o docenti. Di uno studente interessa anche la matricola e di un docente anche il telefono dell'ufficio.
- Alcuni libri sono per la sola consultazione e possono essere presi in prestito per un numero prefissato di giorni solo dagli utenti che sono docenti.

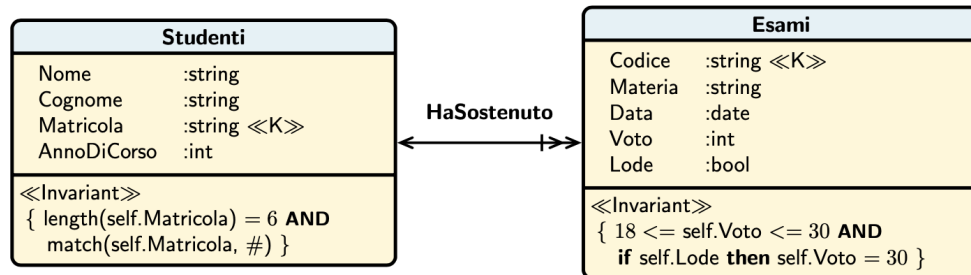


17.14.1 Esempi di esercizi:

- 1) Si vogliono rappresentare informazioni relative a una catena di negozi di abbigliamento. Un negozio ha un indirizzo, un numero di telefono e l'orario di apertura costituito dai giorni della settimana e dalle ore di apertura e chiusura. I vestiti sono identificati da un codice e hanno un modello e una taglia che è espressa con XS, S, M, L o XL se è un vestito per adulti, dall'altezza minima e massima se è per bambini. I modelli sono caratterizzati da un nome, una descrizione, un insieme di colori, un prezzo e dai negozi in cui questi modelli sono disponibili. Inoltre si vogliono memorizzare informazioni relative ai clienti: il nome, il cognome e il numero della carta fedeltà. I clienti possono fare diversi acquisti e si vuole conoscere i vestiti acquistati, il giorno, l'ora, la modalità di pagamento dell'acquisto e il negozio in cui si è effettuato l'acquisto.



17.15 Descrittore di classe con vincoli

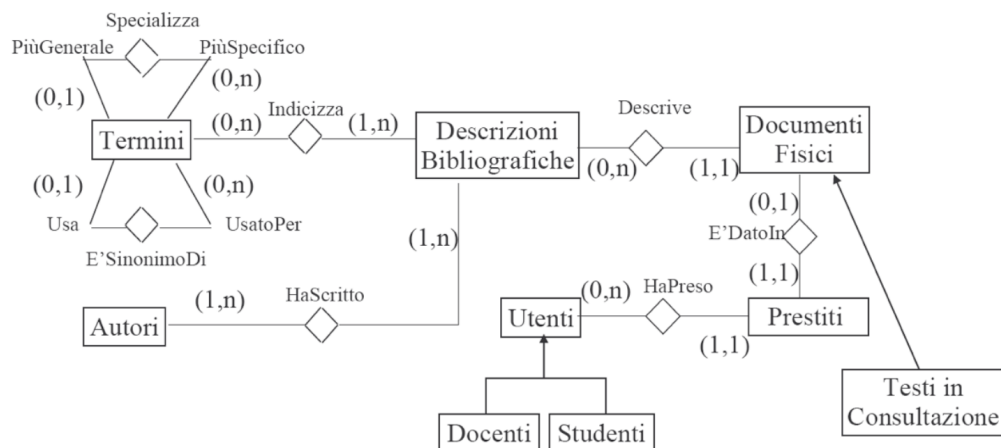


<<K>> **chiave**: sottoinsieme minimale di attributi che identifica l'oggetto
 <<NOT NULL>> **totalità**

self.Nome = attributo *Nome* dell'oggetto stesso

In *Esami* potremmo usare *self.HaSostenuto.Matricola*

17.16 Formalismi grafici



18 Progettazione di Data Base

18.1 Progettazione concettuale di schemi

- Identificare le classi
- Identificare le associazioni con relative proprietà strutturali
- Identificare gli attributi delle classi e associazioni e i loro tipi
- Individuare le sottoclassi
- Individuare le generalizzazioni
- Vincoli di integrità

18.2 Esercizio: Segreteria studenti

La segreteria si occupa di pratiche relative agli studenti del corso di laurea in informatica, e in particolare delle richieste di trasferimento da altri corsi di laurea.

Una **Domanda di trasferimento** viene trasmessa al CCS e determina l'istruzione di una pratica di trasferimento. La pratica viene infine evasa da una delibera che indica quali esami sono stati convalidati. Un esame di un corso esterno viene convalidato come esame interno, e la convalida di un esame può essere piena o previo colloquio integrativo.

Di una domanda di trasferimento interessano il numero di protocollo (che la identifica), il nome e recapito dello studente che la presenta, Università, Facoltà, Corso di laurea di provenienza, data di presentazione, elenco dei corsi sostenuti.

Di una pratica di trasferimento interessano la domanda di trasferimento a cui si riferisce, il numero progressivo che la identifica, annotazioni eventuali, l'eventuale delibera relativa.

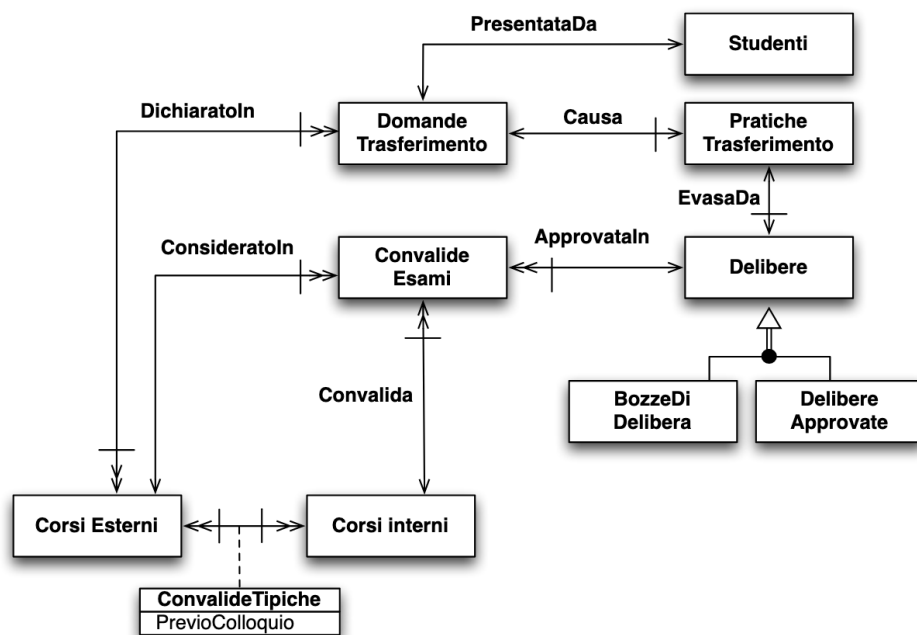
Di una delibera interessano la pratica relativa, gli esami convalidati, annotazioni eventuali, data e numero del verbale del CCS (assenti quando la delibera è in stato di bozza).

Di un corso esterno interessano nome, Università, Facoltà, Corso di laurea (non necessariamente quelli di provenienza dello studente), Anno Accademico di superamento.

Di un corso interno interessano nome e numero di crediti.

Di una convalida d'esame interessa corso interno, corso esterno e se l'esame è convalidato in modo pieno o previo colloquio. Per alcuni corsi esterni esistono convalide tipiche, che indicano corso interno corrispondente e modalità di convalida.

Disegniamo lo schema iniziale con sottocalssi e studenti:



Tipi degli attributi:

Studenti	
Nome:	string
Recapito:	string
UniversitàDiProvenienza:	string
FacoltàDiProvenienza:	string
CorsoDiLaureaDiProvenienza:	string

DomandeTrasferimento	
NumProtocollo:	int
DataPresentazione:	date

PraticheTrasferimento	
Annotazioni:	string
NumeroProgressivo:	int

Delibere	
Annotazioni:	string

BozzeDiDelibere	
-----------------	--

DelibereApprovate	
DataVerbale:	date
NumVerbale:	int

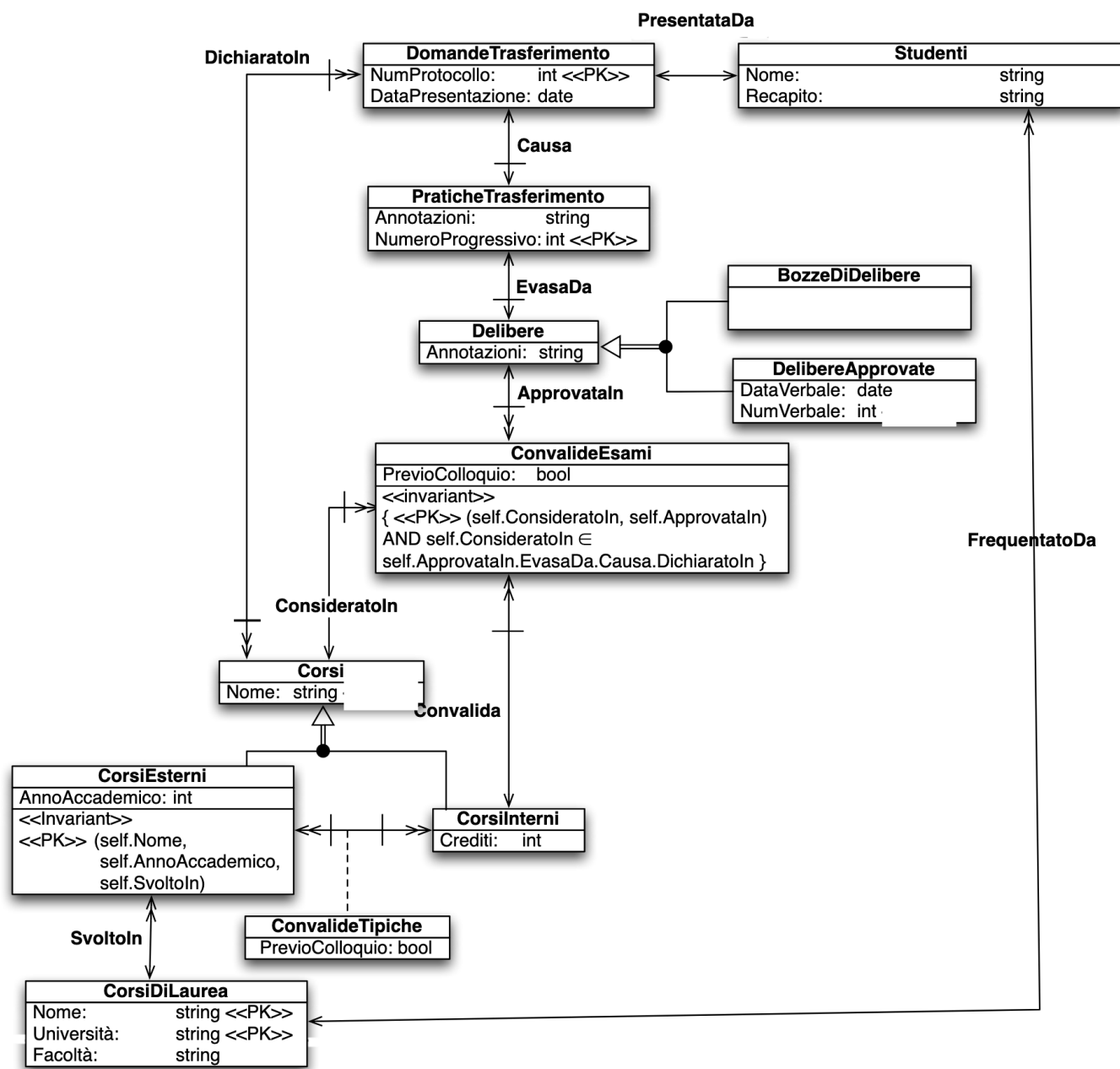
CorsiEsterni	
Nome:	string
Università:	string
Facoltà:	string
CorsoDiLaurea:	string
AnnoAccademico:	int

CorsiInterni	
Nome:	string
Crediti:	int

ConvalideEsami	
PrevioColloquio:	bool

Ora che abbiamo un'idea generale di come debba essere fatto lo schema con vincoli di integrità possiamo andare a farlo:

Schema finale: vincoli di integrità



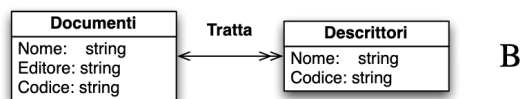
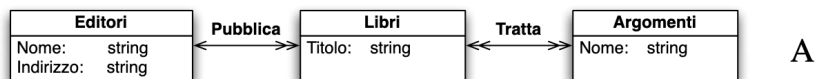
18.3 Integrazione di schemi

Per i grandi database accade che si costruiscano sottoschemi settoriali che vanno poi integrati.

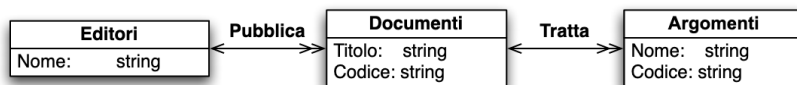
- **Soluzione dei conflitti** (nome, tipo, proprietà strutturali, vincoli di integrità)
- **Fusione degli schemi**
- **Analisi delle proprietà interschema**

esempio:

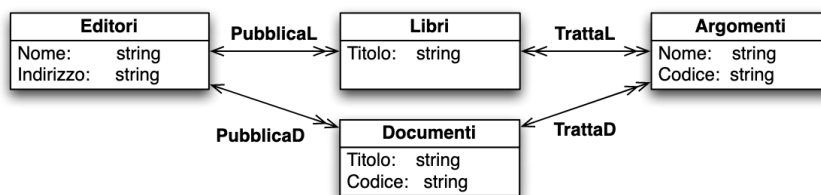
- In una biblioteca:



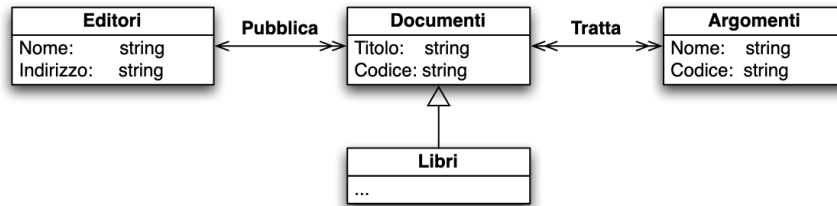
- Risoluzione dei conflitti di nome, di tipo e di vincoli di i.
dobbiamo uniformare gli schemi, nello schema B *Editore* diventa una entità, *Nome* diventa *Titolo*, *Descrittori* diventa *Argomenti*



- Fondere gli schemi:



- Proprietà interschema



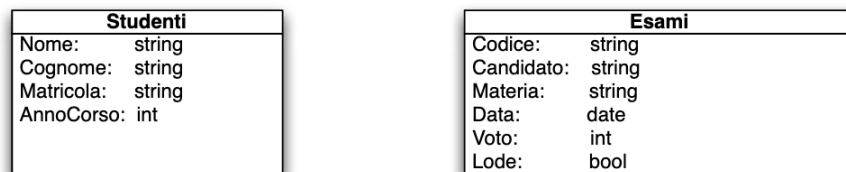
19 Progettazione Logica: Il Modello Relazionale

- Il modello dei dati relazionale
- Trasformazione dal modello concettuale ad oggetti al modello logico relazionale
- Algebra relazionale

20 Il Modello Relazionale

20.1 Intuizione

- Collezioni come relazioni (tabelle)



- Associazioni tramite chiavi

