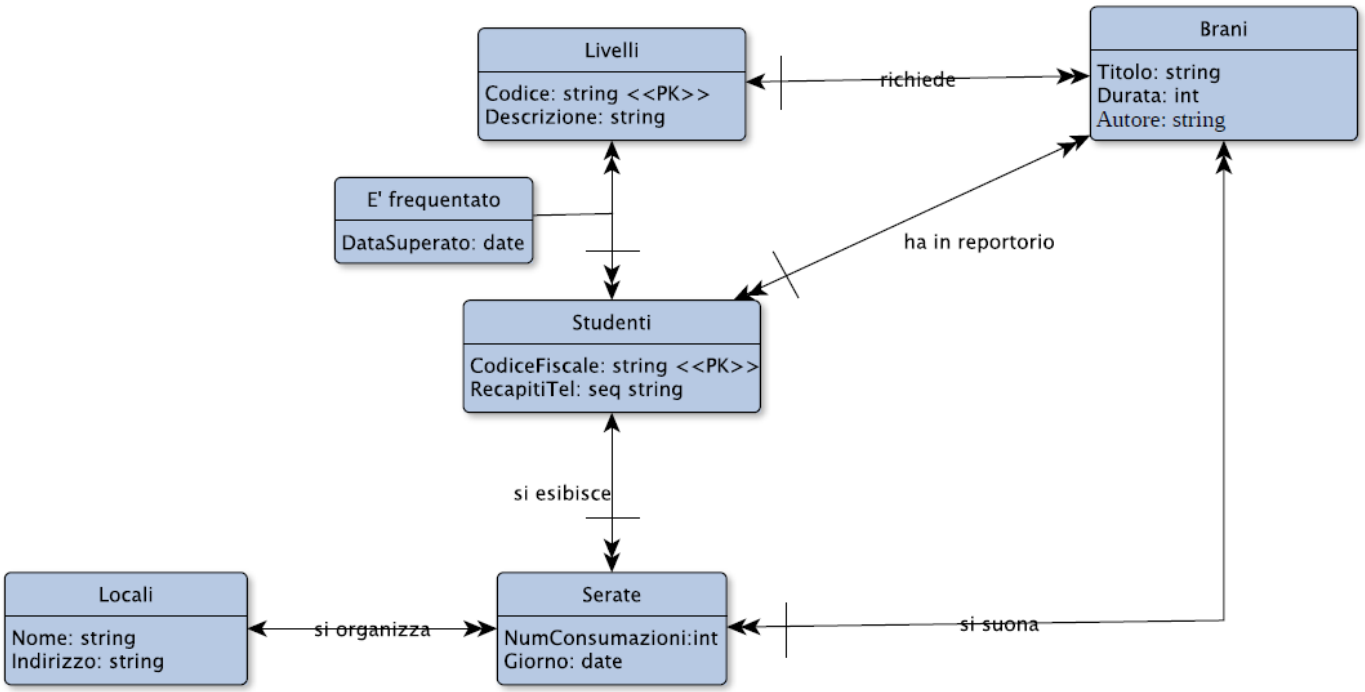
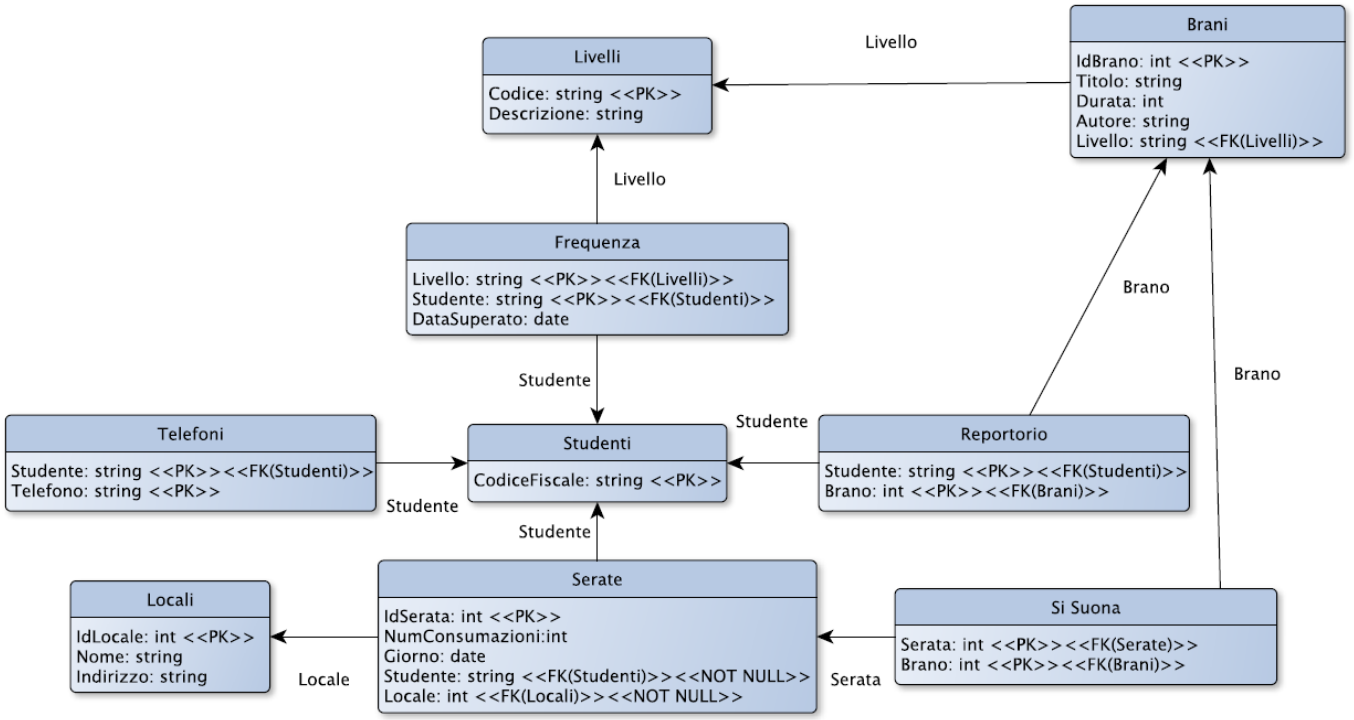
Basi di dati MOD 1 - Modellazione

Nel MOD 1 si progettano lo schema grafico a oggetti e lo schema relazionale.
Esempio schema ad oggetti:



Che andrà convertito in uno schema relazionale:



Noi modelliamo seguendo il concetto di entità e le sue proprietà.
Un'entità è un oggetto concreto, un oggetto astratto oppure un evento. Es:

- Un attore
- Un cantante
- Un libro
- Un telefono
- Una lampada
- Un utente

Le proprietà di un'entità sono coloro che la caratterizzano.Es:

- Un utente ha come proprietà: nome, cognome, indirizzo, telefono.

Una proprietà è una coppia <nome attributo, valore>. Es:

- Utente: <nome, "Andrea">,<cognome, "Rossi">

Le proprietà vengono classificate nei seguenti modi:

- Atomica o Strutturata. Es:
  - Residenza: strutturata, fatta da più elementi (Via, n° civico, CAP, Città, Provincia)
  - Nome: atomica, non è composta da più parti il nome
- Univoca o Multivalore. Es:
  - Cognome: Univoco, una persona non ha più di un cognome.
  - Recapiti telefonici: Multivalore, una persona può averne più di uno!
- Totale o Parziale. Es:
  - Totale: Deve esserci per forza
  - Parziale: Può non esserci

Ogni entità ha un suo tipo, che ne identifica la natura. Es:

- Antonio ha tipo Persona con proprietà:
  - Nome: tipo string
  - Indirizzo: tipo string

Una collezione (classe) è un insieme di entità omogenee (dello stesso tipo). Es:

- Studenti è una collezione, è l'insieme di tutte le entità del dominio del discorso.

Altri esempi di Entità con le relative proprietà:

- Studente: Nome, AnnoNascita, Matricola, Email,...
- Esame: Materia, Candidato, Voto, Lode,...
- Auto: Modello, Colore, Cilindrata, Targa,...

Spesso le collezioni vengono specializzate o generalizzate (sottoclassi e superclassi). Es:

- In una BD Biblioteca possiamo concepire l'entità Utenti come generalizzazione di Studenti e Docenti.

La specializzazione e la generalizzazione hanno due importanti caratteristiche:

1. Ereditarietà delle proprietà
2. Inclusione delle entità della sottoclasse nella superclasse. Es: Se la collezione C1 specializza C2, allora C1 è un sottoinsieme di C2.

Un associazione tra due entità è un fatto che le correla, stabilendo un legame tra di loro. Es:

- Mario legge il libro

- Andrea guida la macchina
- Luigi compra il computer

Le associazioni tra le entità si scrivono con R(X,Y) e hanno due proprietà strutturali:
- Molteplicità
- Totalità

Un'associazione è univoca da X a Y se per ogni istanza di X esiste al massimo un'istanza di Y, altrimenti è multivalore.

Molteplicità:
- R(X,Y) è (1:N) se è multivalore da X a Y e univoca da Y a X.
- R(X,Y) è (N:1) se è univoca da X a Y e multivalore da Y a X.
- R(X,Y) è (M:N) se è multivalore da X a Y e multivalore da Y a X.
- R(X,Y) è (1:1) se è univoca da X a Y e univoca da Y a X.

Vediamo alcuni esempi:
- Frequenta (Studenti, Corsi) ha molteplicità M:N
- Insegna (Professori, Corsi) ha molteplicità 1:N
- SuperatoDa (Esami, Studenti) ha molteplicità N:1
- Dirige(Professori, Dipartimenti) ha molteplicità 1:1

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. Es:
- Insegna (Professori, Corsi) ha molteplicità 1:N, è totale su Corsi in quanto non può esistere un corso senza Professore, parziale su Professori perché un professore può non tenere corsi
- Nata_a (Persone, Città) ha molteplicità N:1, è parziale su Città, perché una Città può avere persone non nate lì e totale su Persone perchè una persona non può non avere una città di nascita
- Ha_visitato (Persone, Città) ha molteplicità M:N, è parziale su Città e su Persone
- è_sindaco_di (Persone, Città) ha molteplicità 1:1, è è parziale su Persone e Città (può esserci città senza sindaco e una persona può non essere sindaco)

Vincoli di integrità: sono di due tipi, statici o dinamici. Li vedremo nel secondo modulo…

Noi modelliamo uno schema, chiamato schema ad oggetti, che è una variante dello schema E-R dove:
- Ogni entità diventa un oggetto
- Il tipo dell'entità diventerà il tipo dell'oggetto
- Una collezione diventa una classe (OOP style fioi...)
- L'associazione rimane un'associazione o diventa una relazione

Un oggetto ha stato, comportamento e identità:
- Stato: un insieme di costanti e variabili con valori di qualsiasi complessità
- Comportamento: è determinato da delle procedure chiamate metodi

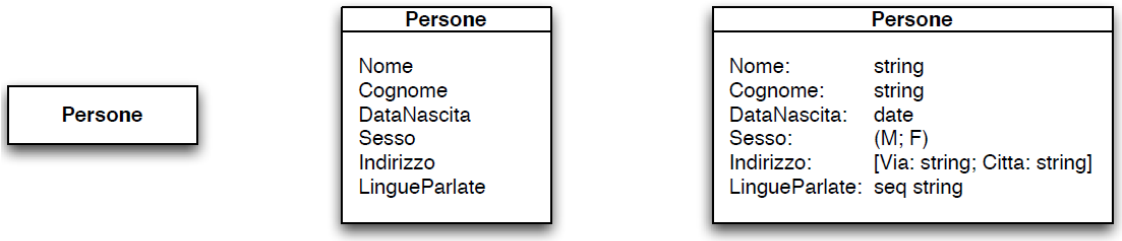
- Identità: è associata all'oggetto alla creazione ed è immutabile

Un oggetto può rispondere a certe richieste, dette messaggi, restituendo valori dello stato o calcolandoli con una procedura locale.

Come prima cosa nella costruzione del modello bisogna classificare le entità e definire i tipi degli oggetti, cosa che ne determinerà i comportamenti (metodi) e gli attributi (proprietà).
Quindi l'oggetto sarà formato da:

- Identità
- Stato

I metodi li lasciamo stare per il momento…
Es classe: Persone



Una classe `Persone` a diversi livelli di specifica

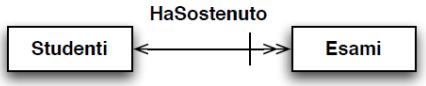
Gli attributi possono essere di due tipi:

- primitivi (int, real, bool, date, string)
- non primitivi

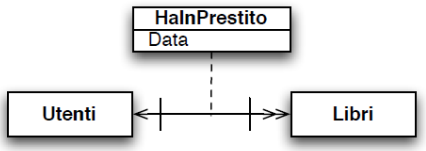
Es di attributi non primitivi, ottenuti applicando i seguenti operatori ad altri tipi:

- tipo record
  [A1:T1; ..., An:Tn]
- tipo enumerazione
  (Val1; ...; Valn)
- tipo sequenza
  seq T

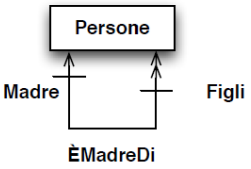
- Le associazioni si modellano con un costrutto apposito



- Le associazioni possono avere delle proprietà



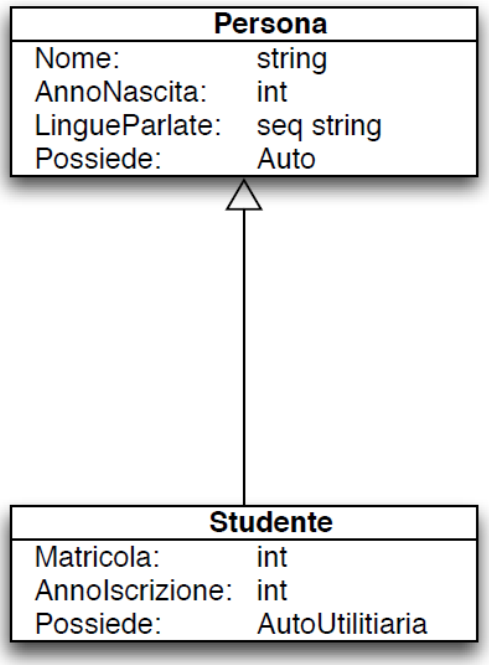
- Le associazioni possono essere ricorsive
- Associazioni n-arie



Nella modellazione possiamo usare l'ereditarietà, cioè creare un oggetto "derivandolo" da un altro e aggiungendo degli attributi, oppure ridefinendo alcuni attributi. Normalmente usiamo l'ereditarietà per definire sottotipi e si fanno due cose in particolare:

- Aggiunta di attributi
- Ridefinizione attributi specializzandone il tipo

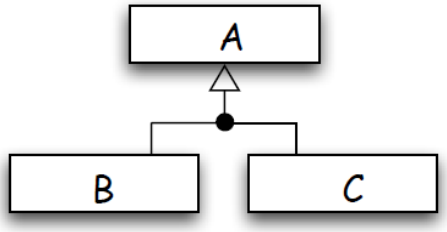
Es: da Persona creo Studente per ereditarietà



Sulle sottoclassi si possono mettere dei vincoli:

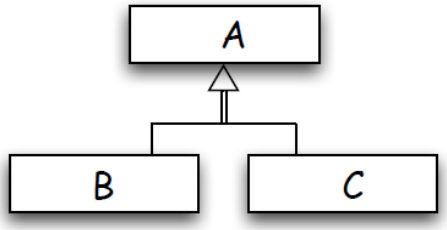
Vincolo di disgiunzione

$B \cap C = \varnothing$

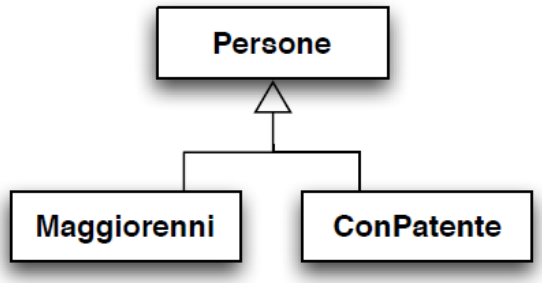


Vincolo di copertura

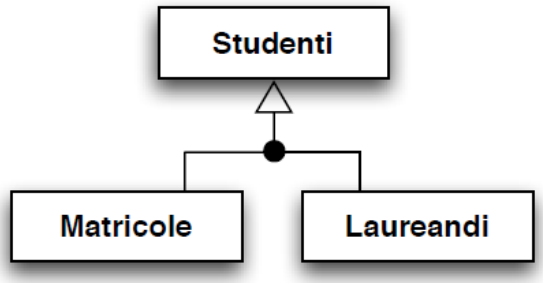
$B \cup C = A$



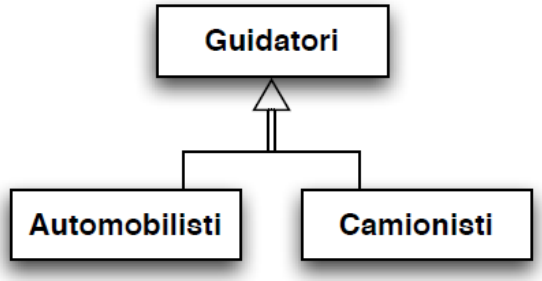
Il pallino va messo dove le due sottoclassi hanno in comune solo gli attributi della classe "madre", mentre la copertura (freccia con linee doppie) va messa quando le due sottoclassi con i loro attributi, combinate, creano la classe madre.
Es:



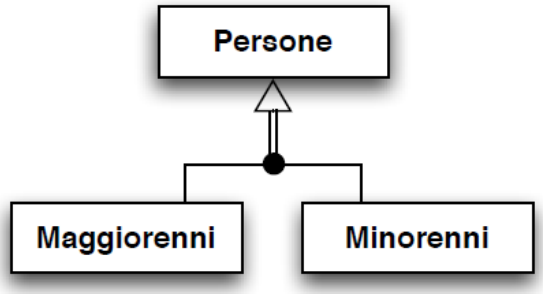
Sottoclassi scorrelate



Sottoclassi disgiunte

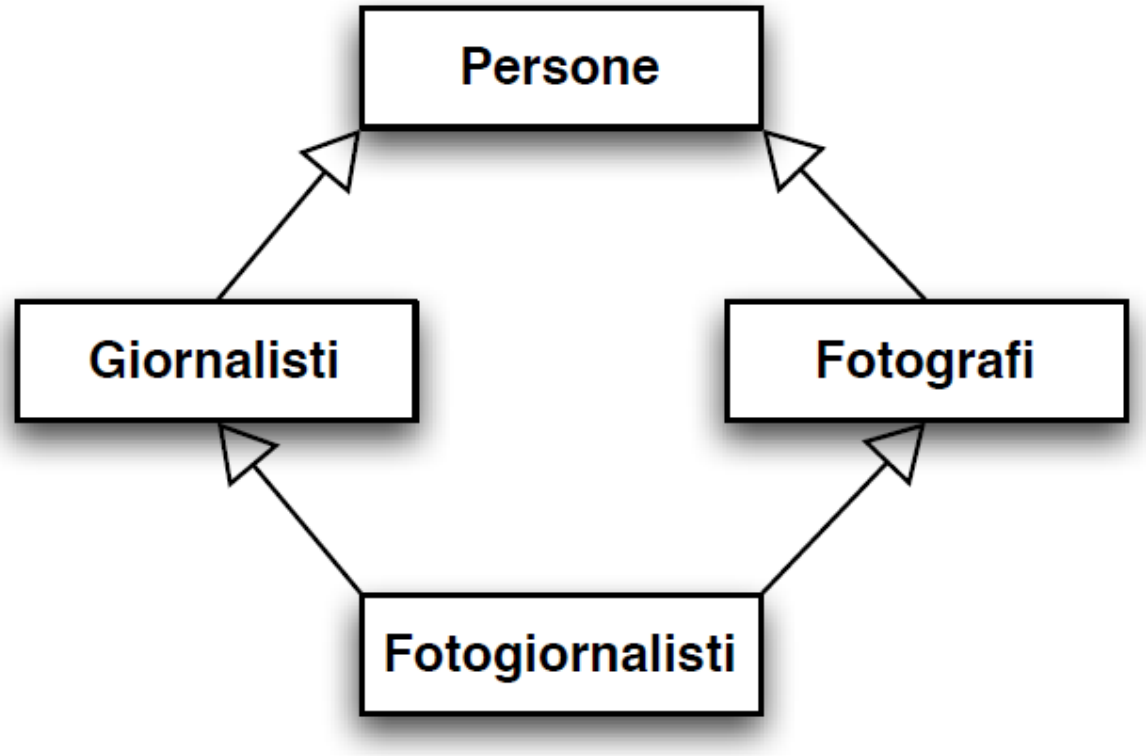


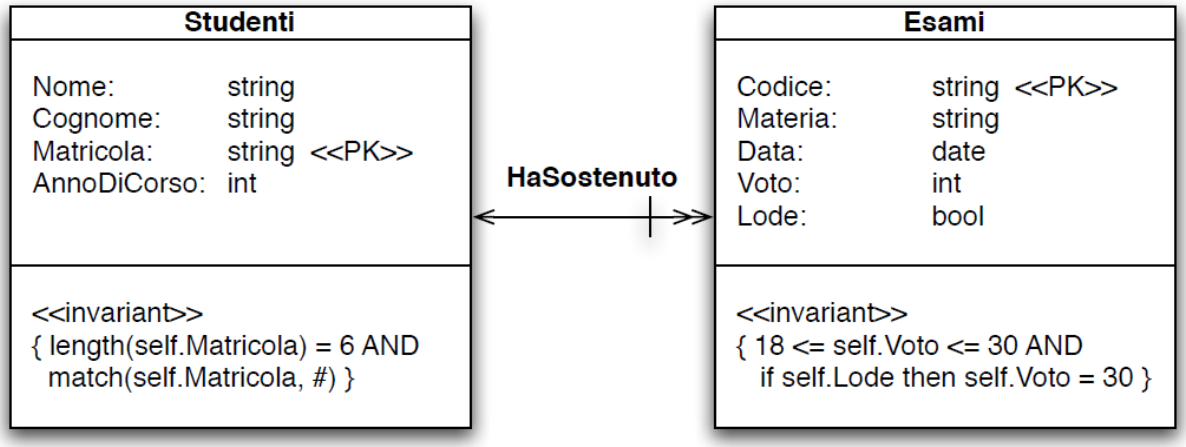
Sottoclassi copertura



Sottoclassi partizione

La gerarchia tra classi può anche essere multipla:



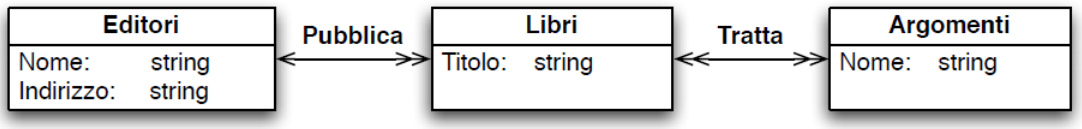


PK è la chiave primaria, che identifica ciascuna istanza dell'oggetto. Può essere composta da più parti!
FK è un attributo che fa riferimento alla chiave primaria di un'altra entità.
Nel modello concettuale la PK va messa solo se esplicitamente richiesto! Sennò va messa solo nel modello ad oggetti!

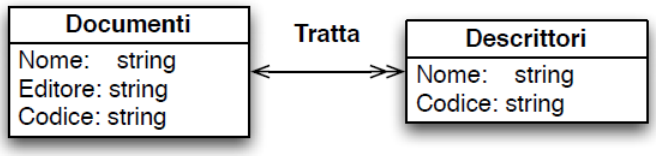
Per grandi BD può essere necessario integrare dei sottoschemi, quando lo si fa dobbiamo fare:

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

Es:
Nella BD di una biblioteca, dobbiamo integrare lo schema B nello schema A



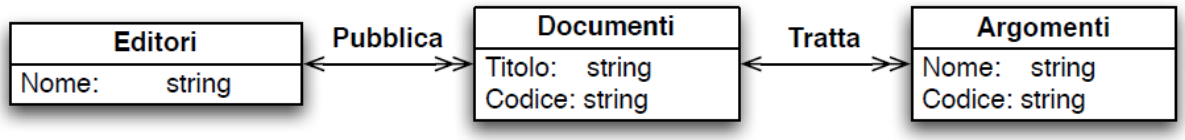
A



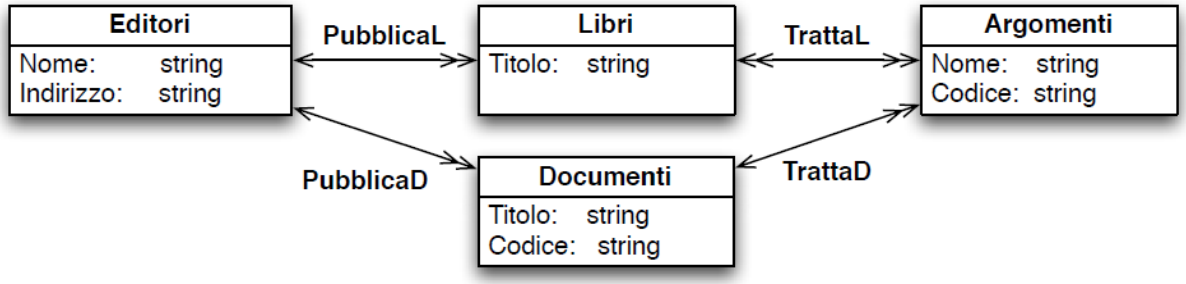
B

Risoluzione dei conflitti di nome, di tipo e di vincoli di integrità uniformando i due schemi:

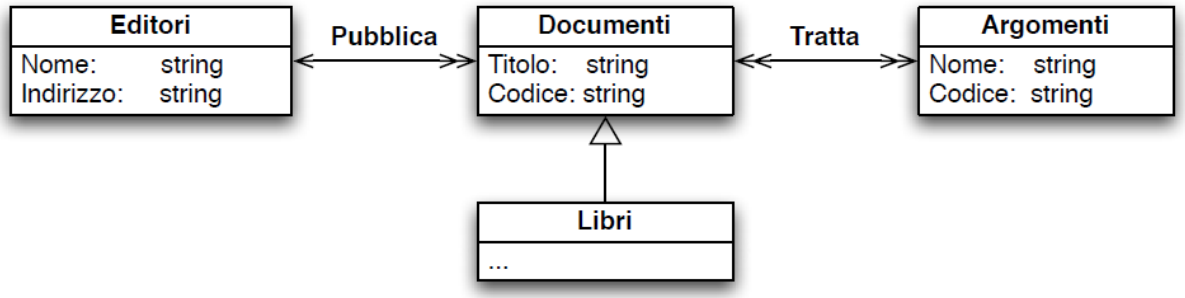
- Nello schema B Editore diventa un'entità, Nome diventa Titolo, Descrittori diventa Argomenti



Alla fine la fusione dei due schemi ci darà:



Oppure potevamo integrare i due schemi in questo modo:



Esempio di modellazione preso da un testo d'esame:

Si vuole costruire una base di dati per la gestione di una scuola di canto. La scuola di canto da la possibilità di certificare i livelli di preparazione per ciascun studente. Ogni livello è identificato da un codice e possiede una descrizione che specifica gli obiettivi didattici che devono essere raggiunti. Per ogni livello ci sono dei brani musicali che lo studente dovrà interpretare per poter ricevere il certificato che attesta di aver raggiunto un certo livello di preparazione. Per ogni studente della scuola, identificato dal codice scale, e necessario sapere quando e stato superato un certo livello di preparazione. Ogni studente ha dei recapiti telefonici per contattarlo e un proprio repertorio di brani, che non necessariamente devono coincidere con quelli dei livelli di preparazione. Ogni brano deve avere il titolo, la durata e l'autore che l'ha composto. Gli studenti della scuola si possono esibire in alcuni locali, di cui si vuole memorizzare il nome, l'indirizzo e le serate organizzate. Per ogni studente si vuole registrare la serata in cui si e esibito e le canzoni che ha cantato. Per ogni serata si vuole sapere il numero di consumazioni vendute per poter permettere al locale di richiamare i cantanti che attirano più pubblico.
Si dia uno schema grafico a oggetti (secondo la notazione del libro di testo) della base di dati e si trasformi nello schema relazionale mostrandone la rappresentazione graca (anche questa secondo la notazione del libro di testo, indicando la chiave primaria e le chiavi esterne). Sia per lo schema a oggetti che per lo schema relazionale si devono specificare, rispettivamente, i nomi e i tipi degli attributi di ciascuna classe e relazione.

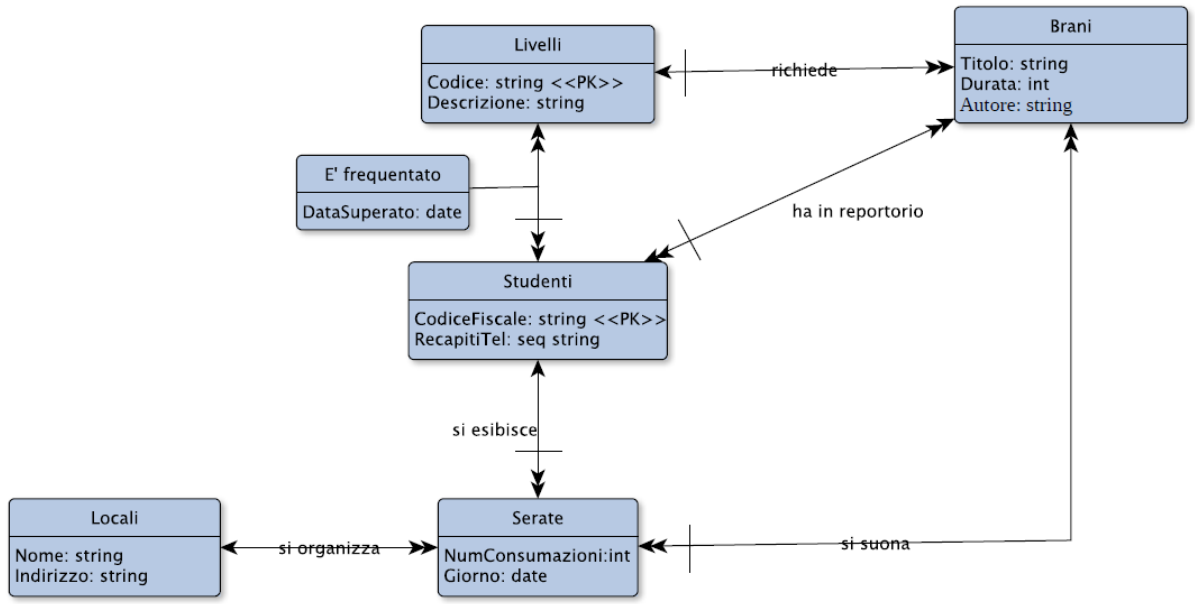
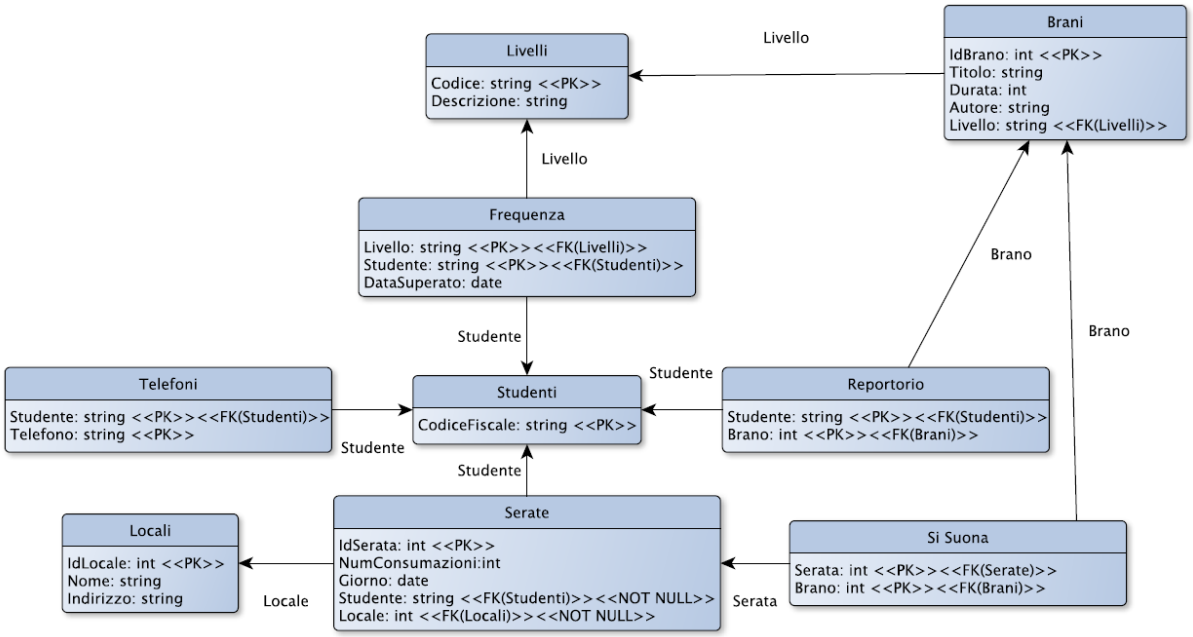


Figura 1: Schema a oggetti


Livelli
Codice: string <<PK>>
Descrizione: string
Livello
Brani
IdBrano: int <<PK>>
Titolo: string
Durata: int
Autore: string
Livello: string <<FK(Livelli)>>
Livello
Frequenza
Livello: string <<PK>><<FK(Livelli)>>
Studente: string <<PK>><<FK(Studenti)>>
DataSuperato: date
Brano
Brano
Studente
Telefoni
Studente: string <<PK>><<FK(Studenti)>>
Telefono: string <<PK>>
Studente
Studenti
CodiceFiscale: string <<PK>>
Studente
Reportorio
Studente: string <<PK>><<FK(Studenti)>>
Brano: int <<PK>><<FK(Brani)>>
Studente
Locali
IdLocale: int <<PK>>
Nome: string
Indirizzo: string
Locale
Serate
IdSerata: int <<PK>>
NumConsumazioni:int
Giorno: date
Studente: string <<FK(Studenti)>><<NOT NULL>>
Locale: int <<FK(Locali)>><<NOT NULL>>
Serata
Si Suona
Serata: int <<PK>><<FK(Serate)>>
Brano: int <<PK>><<FK(Brani)>>