

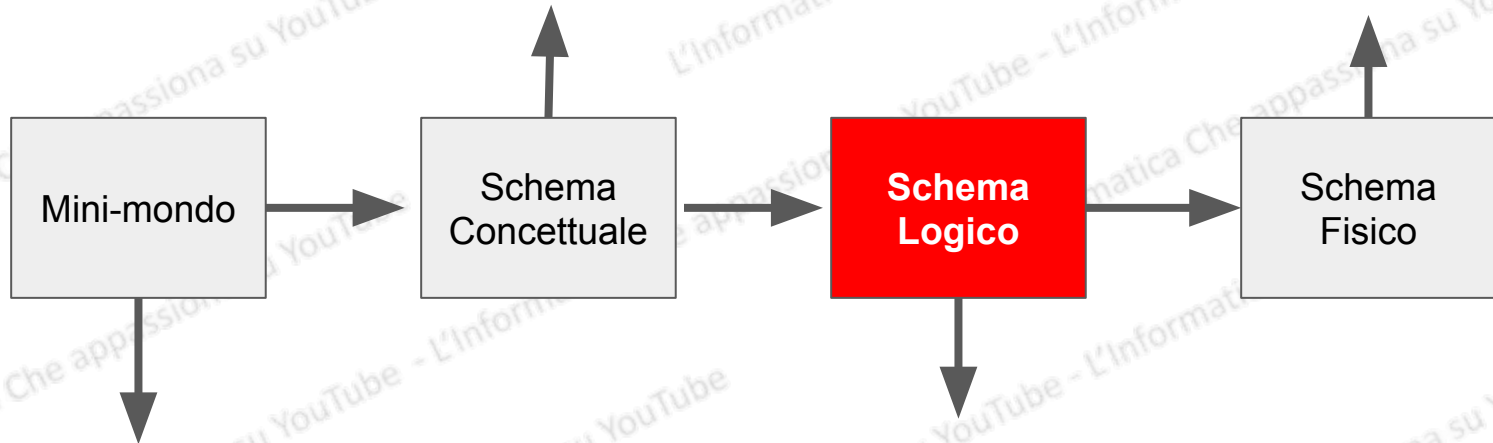
Passo 3 - Schema logico

- Cos'è lo schema logico
- Come indicarlo
- Esercizio svolto

Step per la progettazione di un database

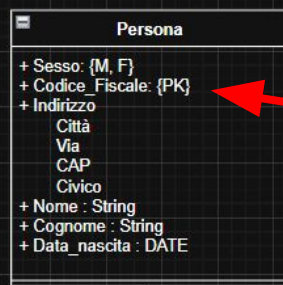
Mediante gli strumenti appositi
Cerchiamo di capire come
dovrebbe venire fuori il DB

Scriviamo il codice
SQL per creare il
nostro database



Ascoltiamo il cliente cosa
Vorrebbe e annotiamo
quante più cose possibili

Esplicitiamo le relazioni per via
del modello relazionale

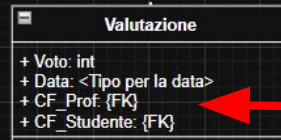
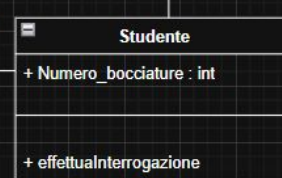
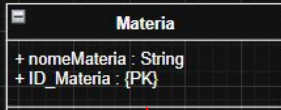


{ totale, disgiunto }

1..*

+ Assegna voto

1..*



Dobbiamo semplicemente riportare le chiavi FK e PK di ciascuna tabella

Un esempio

Studente(ID_Studente, Nome, Cognome)

Professore(ID_Prof, Nome, Cognome)

Votazione(ID_Prof, ID_Studente, Voto, Data)

Chiavi primarie:

Studente.ID_Studente

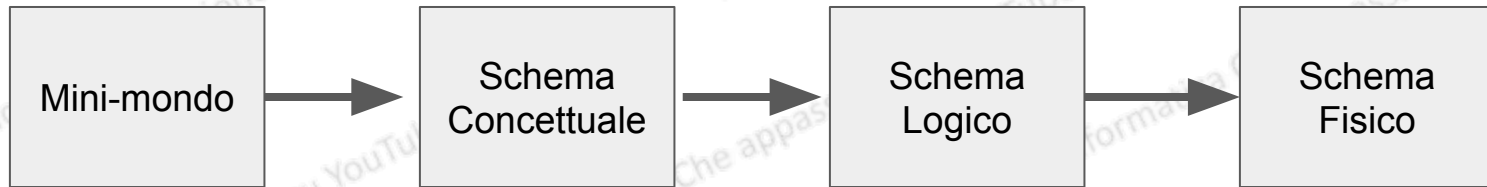
Professore.ID_Prof

Chiavi secondarie:

Votazione.ID_Prof -> Professore.ID_Prof

Votazione.ID_Studente -> Studente.ID_Studente

Facciamo un esercizio completo



Partiamo da qui

Registro elettronico - Traccia

Una scuola deve creare il proprio registro elettronico.

In particolare, ha la necessità di immagazzinare informazioni riguardo i loro studenti, le informazioni che dovrà immagazzinare sono: nome, cognome, data di nascita mentre per i professori vogliamo nome, cognome, data di nascita e la materia che insegnano (ciascun professore potrebbe insegnare più materie).

Come qualunque registro elettronico che si rispetti, si vuole immagazzinare anche informazioni riguardo i voti che gli studenti ricevono e in particolare oltre al voto numerico è importante avere la data di quando quest'ultimo è stato inserito.

Inoltre, si vuole memorizzare anche le classi a cui sono associati gli studenti. Ogni classe può contenere un massimo di 25 studenti (secondo le indicazioni ministeriali).

Regole per la lettura

- Rosso = Entità
- Sottolineatura = Attributi
- Grassetto = Cardinalità e Partecipazione

Registro elettronico - 1° Passo

Una scuola deve creare il proprio registro elettronico.

In particolare, ha la necessità di immagazzinare informazioni riguardo i loro **studenti**, le informazioni che dovrà immagazzinare sono: nome, cognome, data di nascita mentre per i **professori** vogliamo nome, cognome, data di nascita e la materia che insegnano (ciascun professore **potrebbe insegnare più materie**).

Come qualunque registro elettronico che si rispetti, si vuole immagazzinare anche **informazioni riguardo i voti** che gli studenti ricevono e in particolare oltre al voto numerico è importante avere la data di quando quest'ultimo è stato inserito.

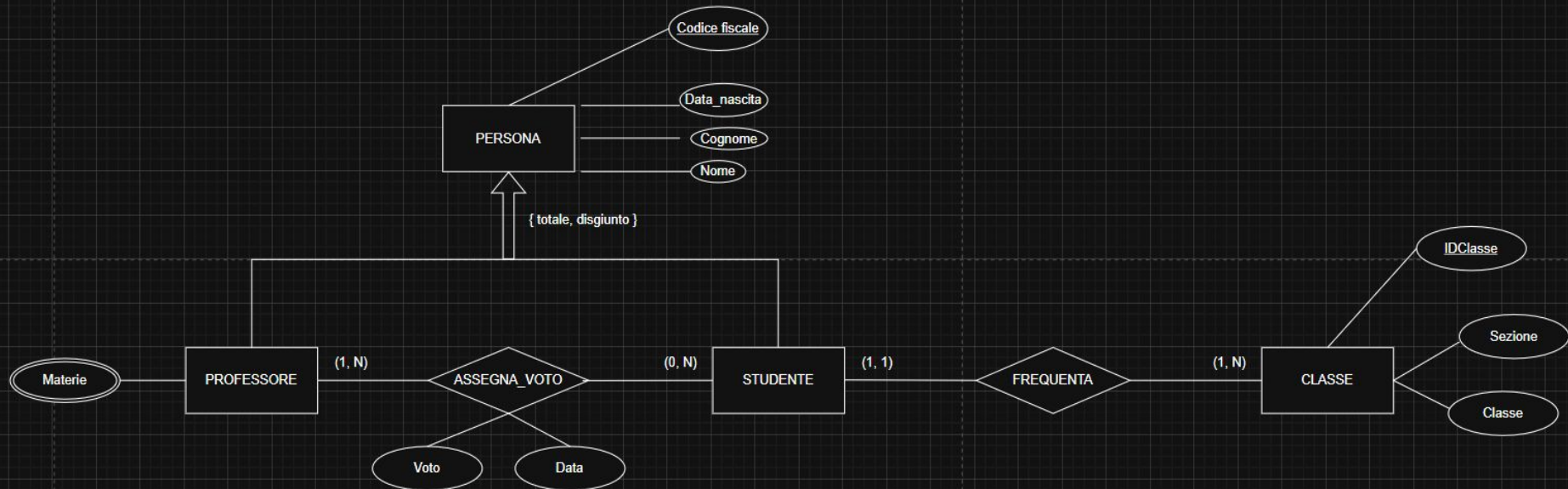
Inoltre, si vuole memorizzare anche le **classi** a cui sono associati gli studenti. Ogni classe può **contenere un massimo di 25 studenti**

Approfondimento - 1° Passo

- Abbiamo domande per il professore?
- C'è qualcosa per noi di ambiguo?
- Il cliente ci ha fornito poche informazioni?
- Potrebbero esserci errori nella traccia?

Dobbiamo farci quante più domande possibili


Registro elettronico - 2° Passo



Abbiamo eseguito una prima bozza del diagramma, dobbiamo ristrutturarlo **sennò non possiamo fare lo schema logico**

Ristrutturazione - 2° Passo

Consiglio: Parti sempre dalla generalizzazione e interpreta il suo significato

Opzione	Quando usarlo	Cosa fare	Vantaggi	Svantaggi	
8A Relazioni multiple – superclasse e sottoclasse	Qualsiasi tipo di specializzazione (totale/parziale, disgiunta/sovrapposta)	Creare una relazione per la superclasse e una per ciascuna sottoclasse	Flessibile per tutti i tipi di specializzazione	Redondanza e necessità per recuperare dati completi	Copia tabella
8B Relazioni multiple – solo sottoclassi	Specializzazione totale e preferibilmente disgiunta	Creare solo una relazione per ciascuna sottoclasse (niente superclasse)	Eliminazione della relazione della superclasse Risparmio spazio	Ridondanza se sovrapposta (duplicazione) Perdita info se la specializzazione è parziale	
8C Singola relazione con attributo tipo	Specializzazione disgiunta	Un'unica relazione con tutti gli attributi e un attributo tipo per indicare la sottoclasse	Schema compatto Nessun join necessario	Non adatta a specializzazioni sovrapposte Molti NULL se sottoclassi hanno attributi molto diversi	
8D Singola relazione con molti attributi tipo	Specializzazione sovrapposta	Un'unica relazione con tutti gli attributi + un attributo booleano per ogni sottoclasse	Supporta specializzazioni sovrapposte Nessun join necessario	Molti attributi booleani e molti valori NULL	

Ristrutturazione - 2° Passo

Se dovessi scegliere:

- 8A: Voglio mantenere una distinzione tra tutti **però dovrei fare più query**
- 8B: **Duplicherei** i dati della superclasse
- 8C: Potrei **cambiare il significato** della generalizzazione

8A: Vantaggi vs Svantaggi

Vantaggi 8A:

- Mantengo una distinzione molto chiara tra: Persona, Studente e Docente
- Non appesantisco un'entità con molti valori NULL, ogni riga ha il suo

Svantaggi 8A:

- Sto usando più spazio, perché ho creato una tabella in più
- Creo maggiore complessità, lo schema diventa più grande

8A

Relazioni multiple –
superclasse e sottoclasse

Qualsiasi tipo di
specializzazione
(totale/parziale,
disgiunta/sovrapposta)

Creare una relazione per la
superclasse e una per
ciascuna sottoclasse

Flessibile per tutti i tipi di
specializzazione

Redondanza e necessità
per recuperare dati completi

[Copia tabella](#)

8B: Vantaggi vs Svantaggi

Vantaggi 8B:

- Riduco la complessità dello schema
- Non appesantisco un'entità con molti valori NULL, ogni riga ha il suo

Svantaggi 8B:

- Sto duplicando gli attributi comuni per ciascuna sottoclasse
- Se voglio solo un certo tipo di dato di Professore devo per forza visualizzare le sue generalità mentre vorrei solo le sue materie

8B

Relazioni multiple – solo sottoclassi

Specializzazione **totale** e preferibilmente **disgiunta**

Creare solo una relazione per ciascuna sottoclasse (niente superclasse)

Eliminazione della relazione della superclasse
Risparmio spazio

Ridondanza se sovrapposta (duplicazione)
Perdita info se la specializzazione è parziale

8C

Specializzazione **disgiunta**

Un'unica relazione con tutti

Schema compatto

Non adatta a specializzazioni

8C: Vantaggi vs Svantaggi

Vantaggi 8C:

- Risparmio molto più spazio

Svantaggi 8C:

- Potrei avere molti NULL
- Potrebbe avere un significato diverso (?)

8C

Singola relazione con attributo
tipo

Specializzazione **disgiunta**

Un'unica relazione con tutti
gli attributi e un attributo tipo
per indicare la sottoclasse

Schema compatto
Nessun join necessario

e parziale

Non adatta a specializzazioni
sovrapposte
Molti NULL se sottoclassi hanno
attributi molto diversi

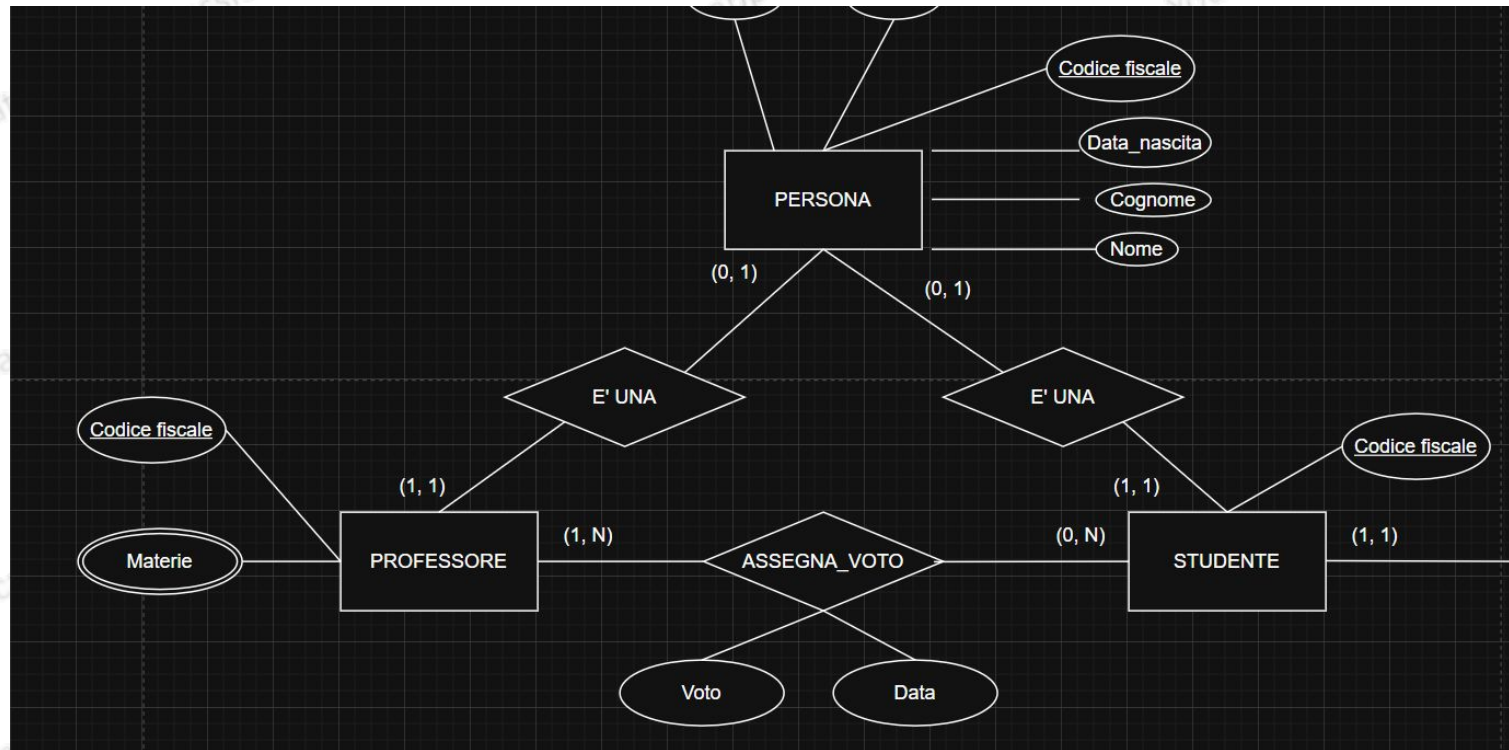
Cosa scegliamo?

Dipende!

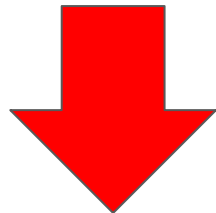
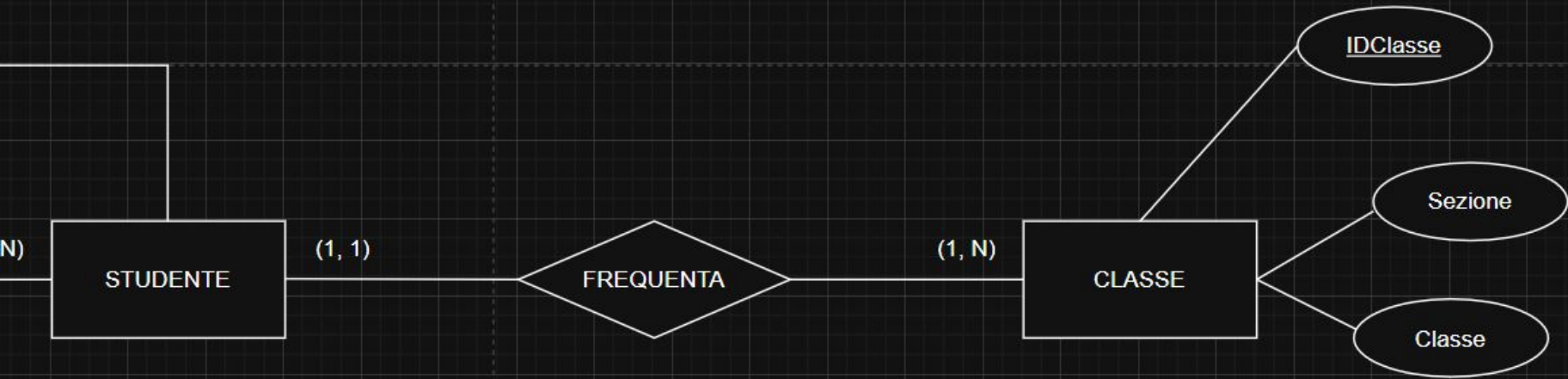
- Il cliente deve andare a risparmio?
- Ha delle esigenze particolari?
- Dobbiamo mantenere quanto più possibile separate le logiche?
- ...

Personalmente scelgo la 8A, mantengo tutto separato. Non c'è nulla di male

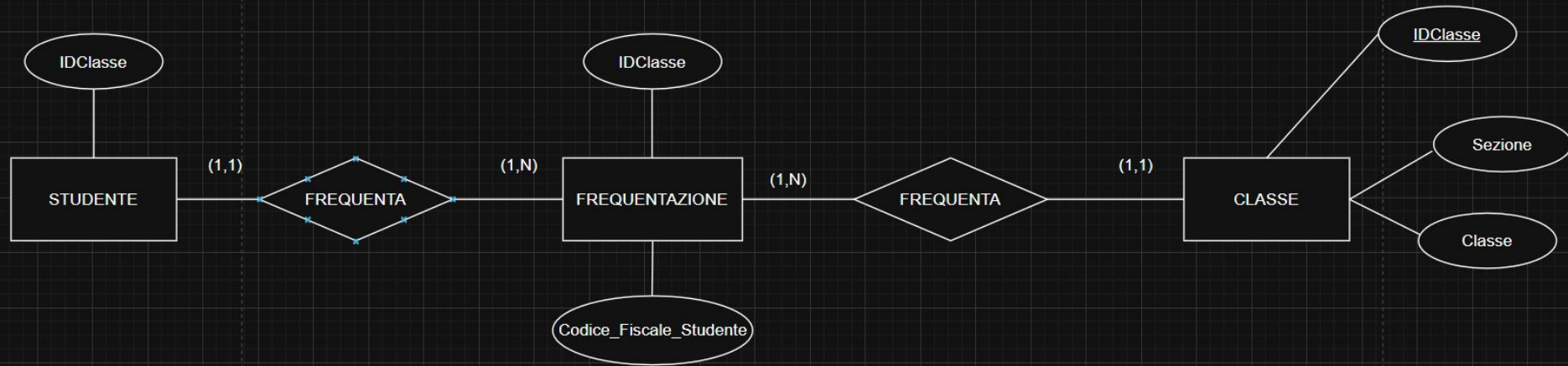
Generalizzazione Ristrutturata



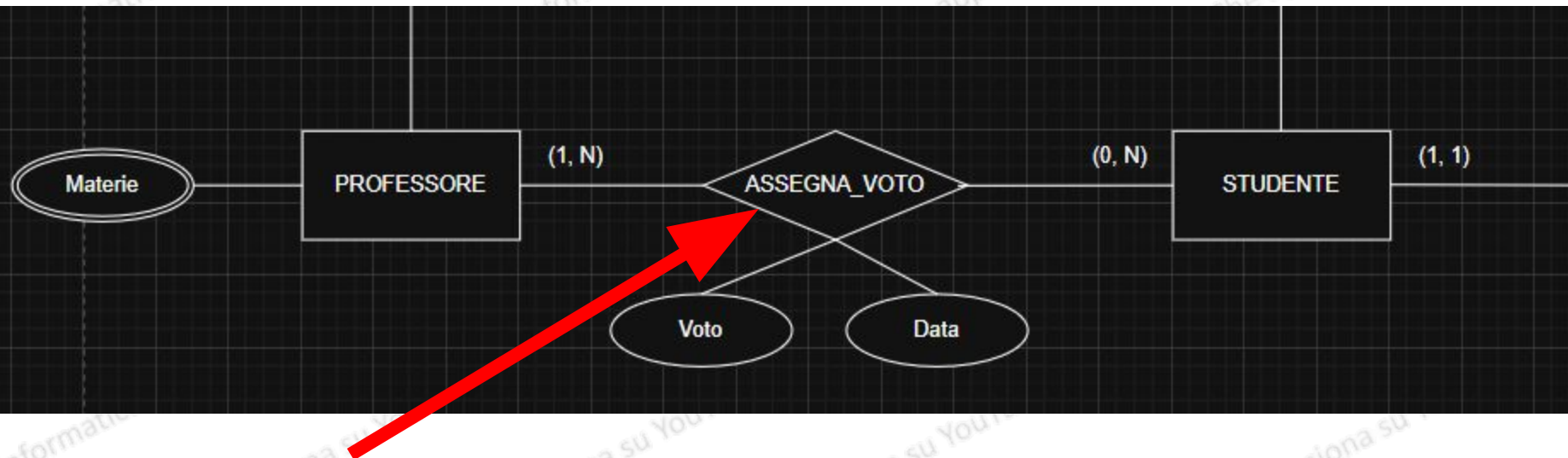
Studente - Classe?



Ristrutturazione Studente-Classe

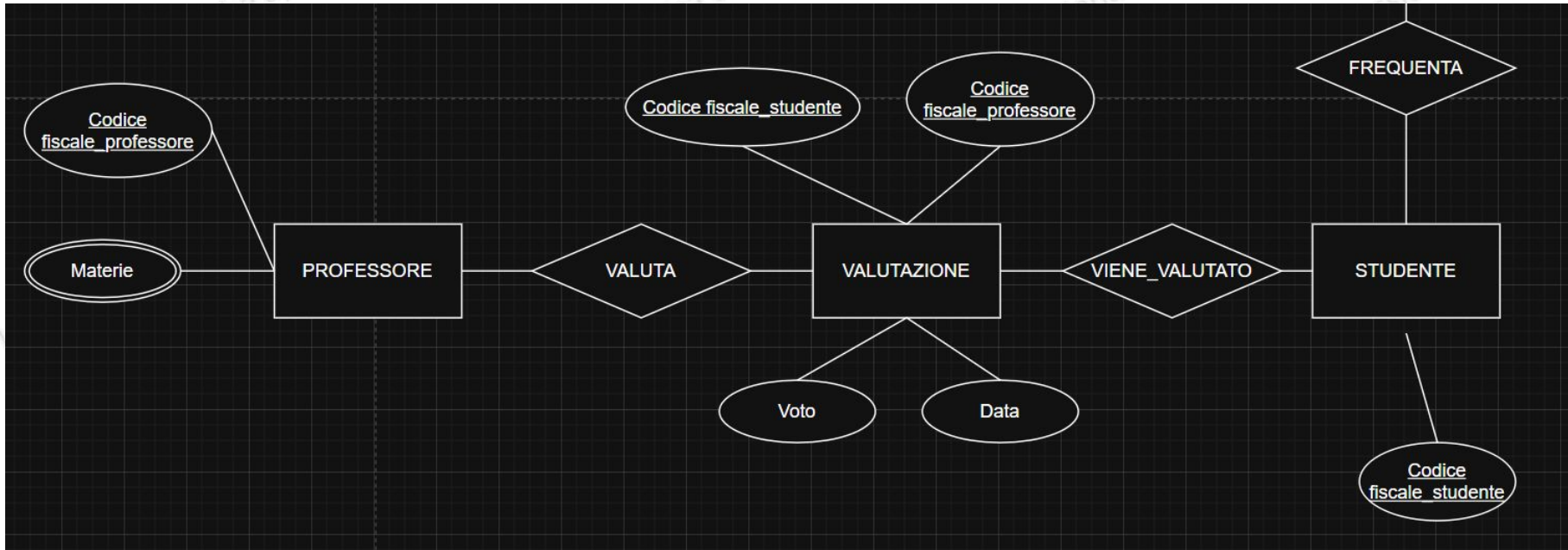


Professore - Studente?



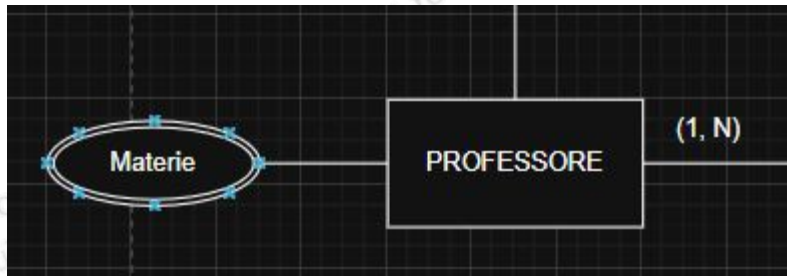
Attenzione alla relazione che contiene attributi

Professore - Studente?

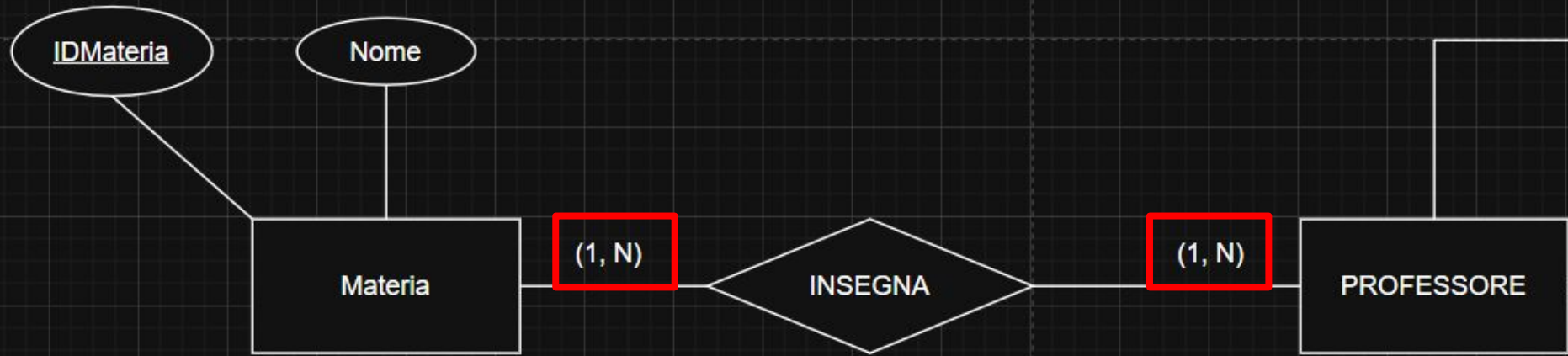


Attributi strani ?

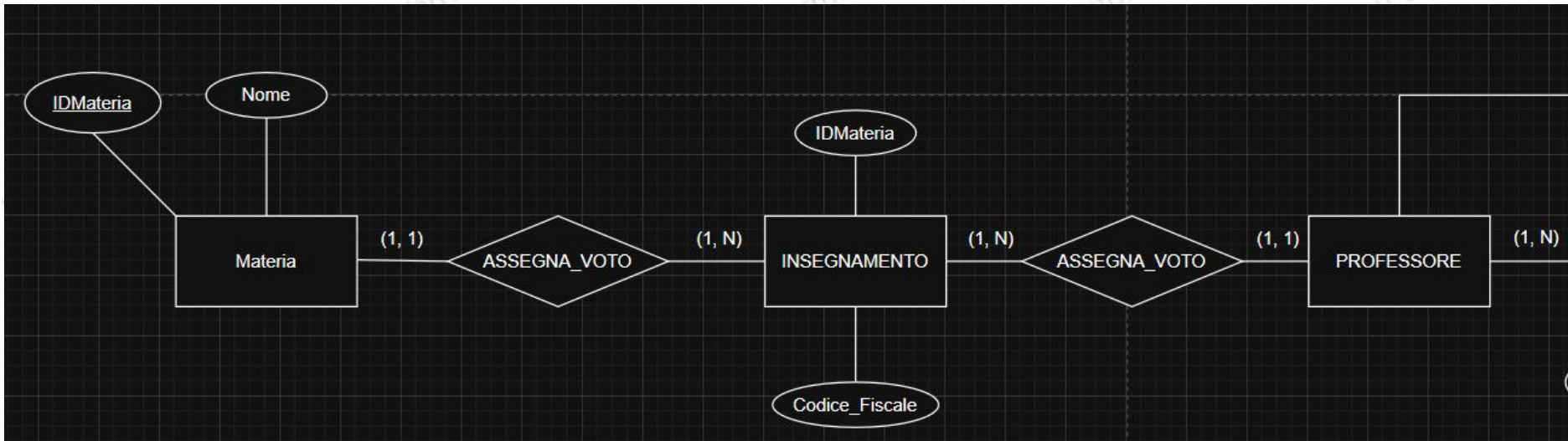
- Ristrutturiamo attributi multivalore, composti e derivati



Bozza della ristrutturazione di Materie



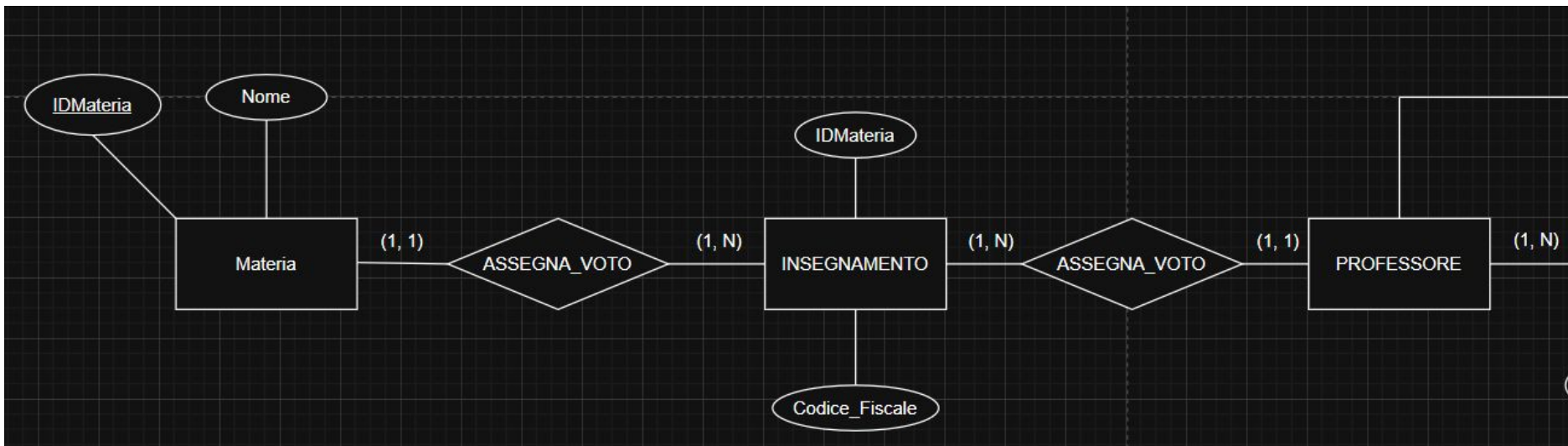
Ristrutturazione Materia



Schema logico - 3° passo

- Cerchiamo di mettere tutti gli schemi finali insieme, partiamo dalla **Materia che già è un'entità**

Per ovvie ragioni ci saranno alcuni attributi che poi leveremo da alcuni schemi logici



Materia(IDMateria, Nome)

Professore (Codice_Fiscale) - **Lo definiamo dopo**

Insegnamento(IDMateria, Codice_Fiscale)

Chiavi primarie:

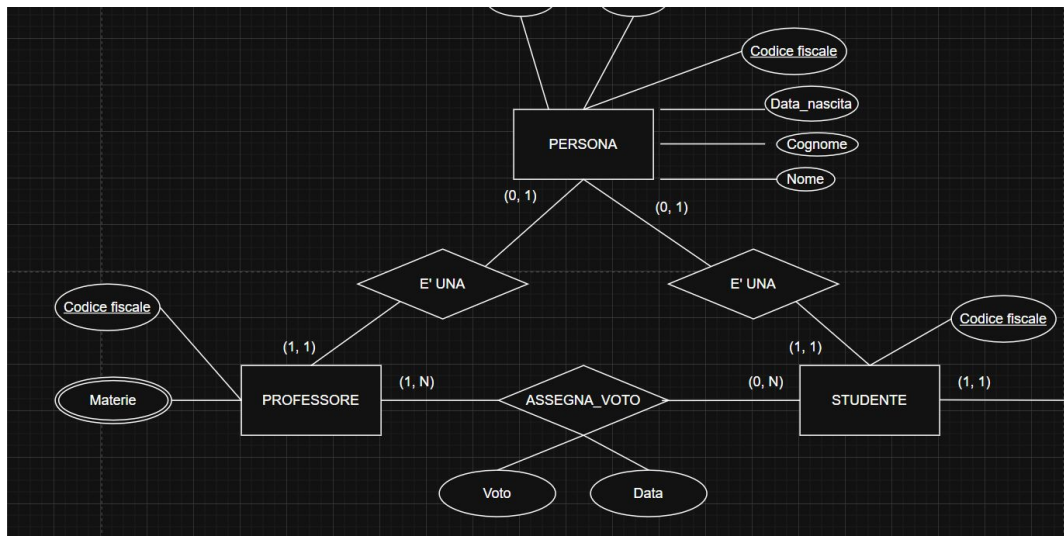
Materia.IDMateria

Chiavi secondarie:

Insegnamento.Codice_Fiscale -> Professore.Codice_Fiscale

Insegnamento.IDMateria -> Materia.IDMateria

**Gestione dell'attributo
multivalore Materie**



Gestione della generalizzazione

Persona(Codice_Fiscale, Data_nascita, Cognome, Nome)

Professore(Codice_Fiscale)

Studiante(Codice_Fiscale)

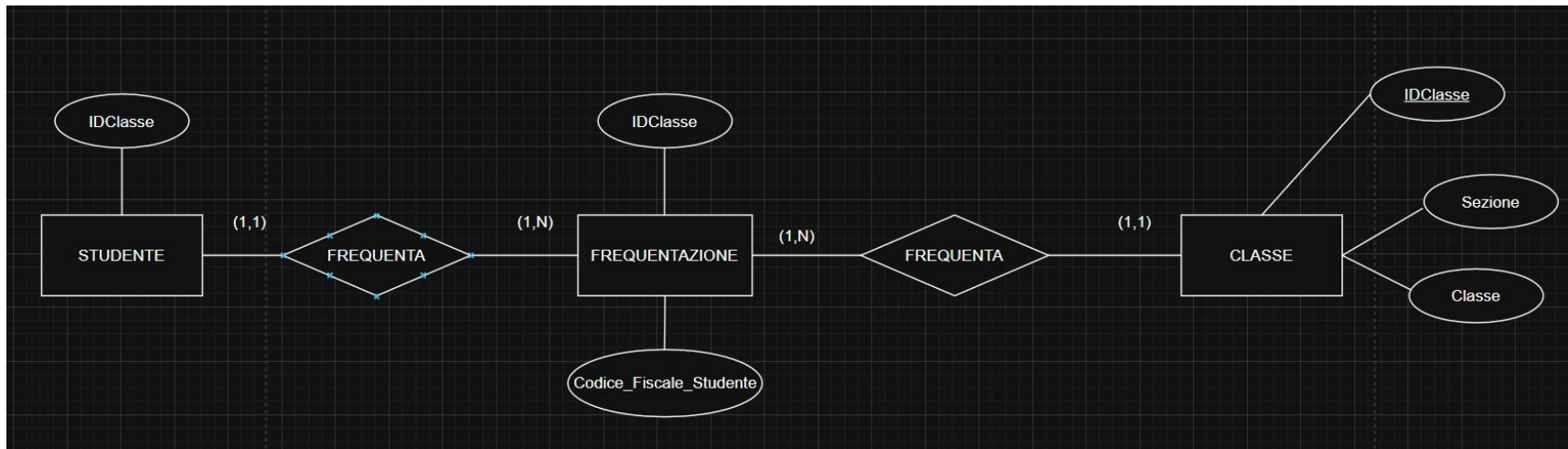
Chiavi primarie:

Persona.Codice_Fiscale

Chiavi secondarie:

Professore.Codice_Fiscale -> Persona.Codice_Fiscale

Studiante.Codice_Fiscale -> Persona.Codice_Fiscale



Studente(IDClasse, Codice_Fiscale_Studente)

Classe(IDClasse, Sezione, Classe)

Frequenzazione(IDClasse, Codice_Fiscale_Studente)

Chiavi primarie:

Classe.IDClasse

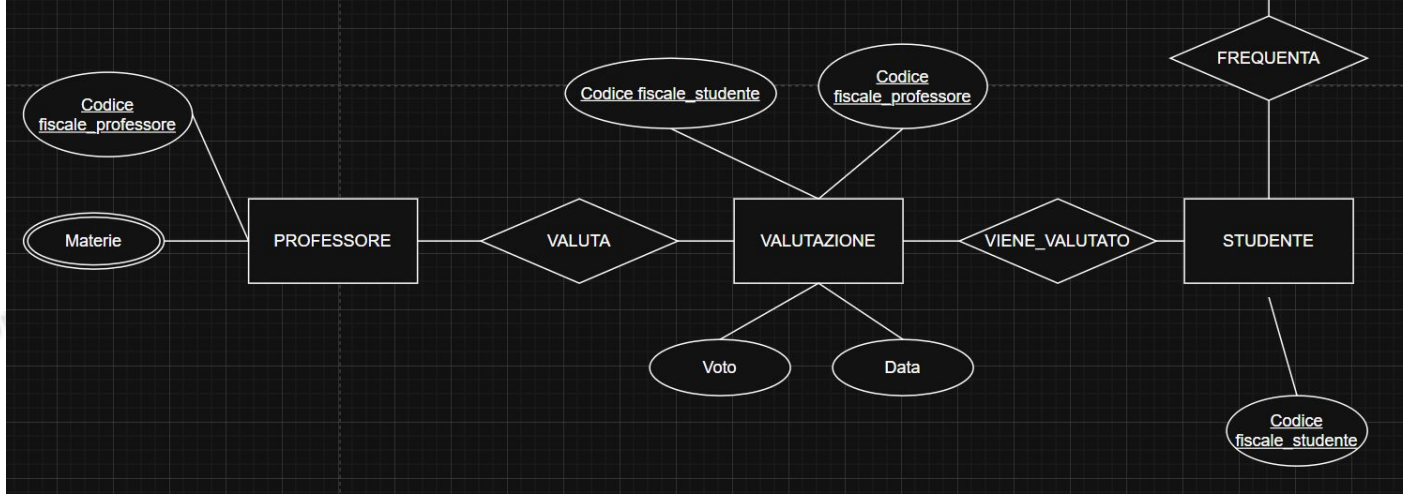
Chiavi secondarie:

Studente.IDClasse -> Classe.IDClasse

Frequenzazione.IDClasse -> Classe.IDClasse

Frequenzazione.Codice_Fiscale_Studente -> Studente.Codice_Fiscale

Gestione della relazione Studente-Classe



Studente e Professore – Già definiti

Valutazione(CF_Stud, CF_Prof, Voto, Data)

Chiavi secondarie:

Valutazione.CF_Stud = Studente.CF_Stud

Valutazione.CF_Prof = Professore.CF_Prof

**Gestione della relazione
Studente-Professore**

Entità:

Professore (Codice_Fiscale)

Persona(Codice_Fiscale, Data_nascita, Cognome, Nome)

Studente(Codice_Fiscale, IDClasse)

Materia(IDMateria, Nome)

Insegnamento(IDMateria, Codice_Fiscale)

Classe(IDClasse, Sezione, Classe)

Frequenzazione(IDClasse, Codice_Fiscale_Studente)

Valutazione(CF_Stud, CF_Prof, Voto, Data)

Chiavi primarie:

Materia.IDMateria

Persona.Codice_Fiscale

Classe.IDClasse

Chiavi secondarie:

Insegnamento.Codice_Fiscale -> Professore.Codice_Fiscale

Insegnamento.IDMateria -> Materia.IDMateria

Professore.Codice_Fiscale -> Persona.Codice_Fiscale

Studente.Codice_Fiscale -> Persona.Codice_Fiscale

Studente.IDClasse -> Classe.IDClasse

Frequenzazione.IDClasse -> Classe.IDClasse

Frequenzazione.Codice_Fiscale_Studente -> Studente.Codice_Fiscale

Valutazione.CF_Stud = Studente.CF_Stud

Valutazione.CF_Prof = Professore.CF_Prof

Schema logico finito