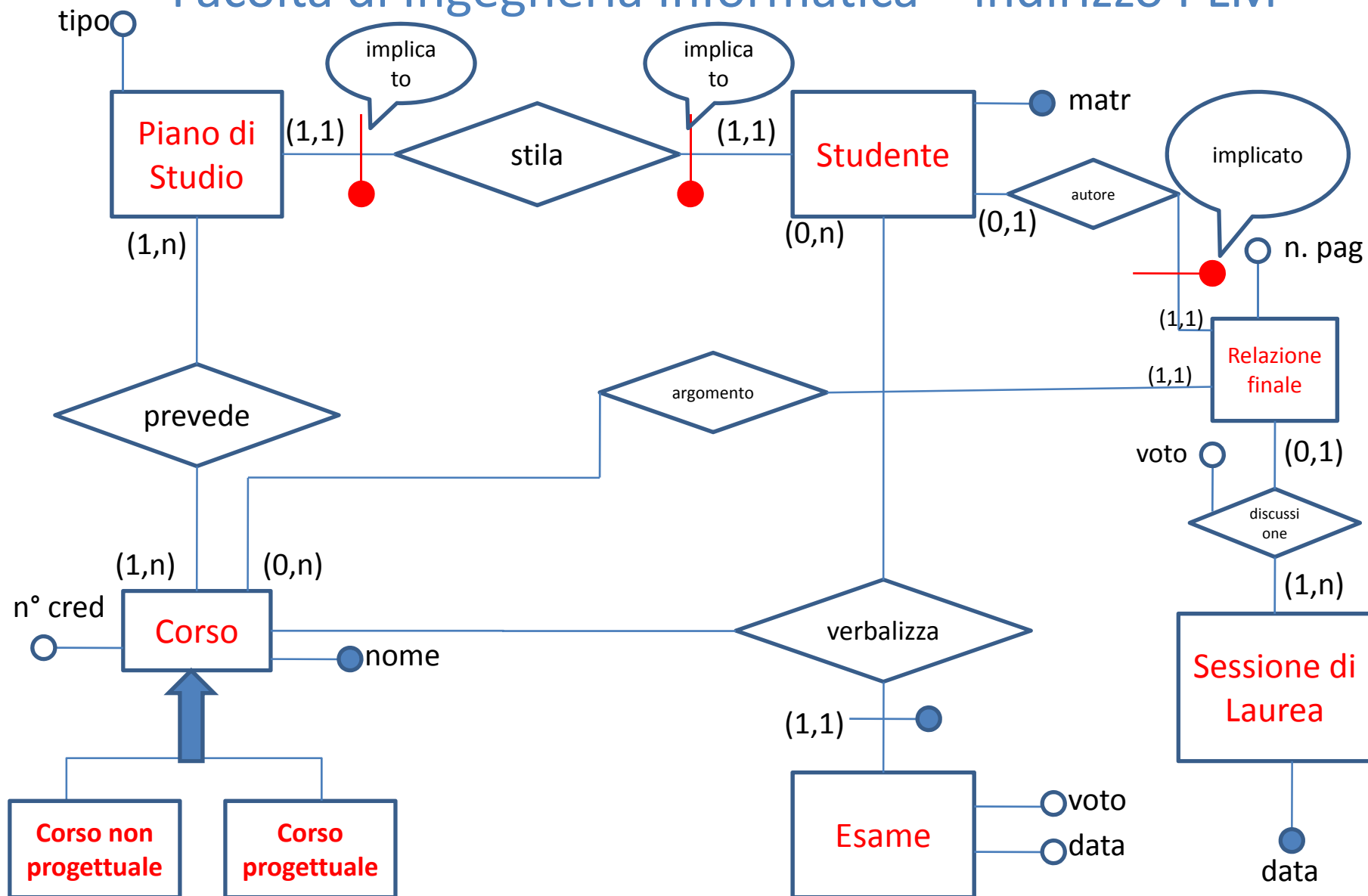


Esercitazione 2

Vincoli dichiarativi, stored routine e
trigger in MySQL








Progettazione concettuale: schema ER SAPIENZA – Facoltà di Ingegneria Informatica – indirizzo PLM



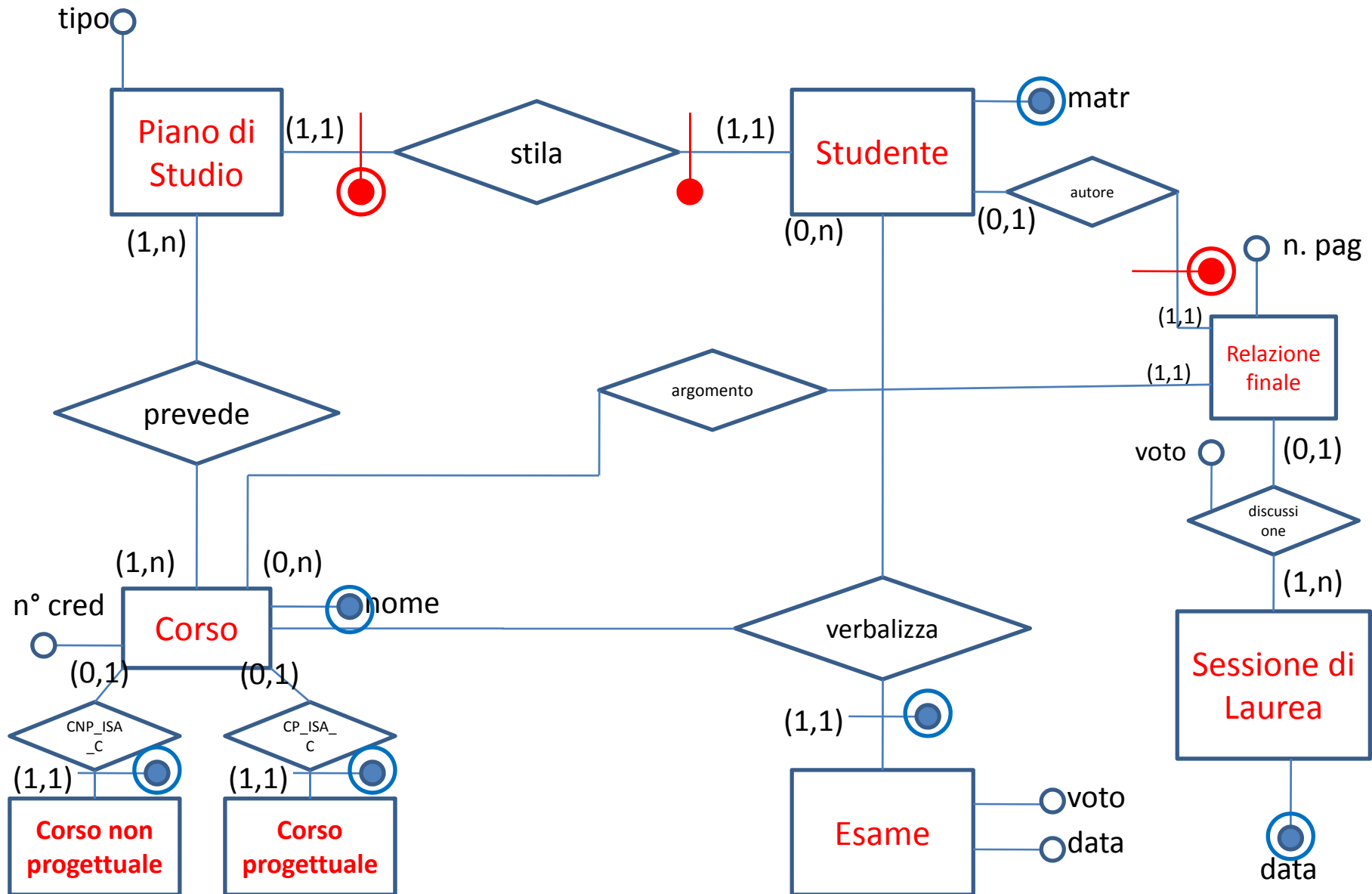
Vincoli di integrità esterni

1. Nel caso dei corsi progettuali, l'attributo nome può assumere soltanto i valori che appartengono all'insieme { 'Progetto di applicazioni software', 'Progetto di reti di calcolatori e sistemi informatici' }
2. Uno studente può discutere la sua tesi di laurea in una certa sessione solo se tutti gli esami previsti dal suo piano di studio sono stati verbalizzati (con voto maggiore o uguale a 18)
3. Esistono solamente 2 tipi di piani di studio:
tipo \in { 'PLM', 'Sistemi Informatici' }
4. Se uno studente ha un piano di studi di tipo PLM, allora deve produrre una relazione finale che verta su un corso di tipo progettuale non compreso nel suo piano di studio
5. La somma totale dei crediti previsti dai corsi di un qualunque piano di studio deve essere pari a 174
6. Un voto di laurea può essere compreso tra 70 e 110

Ristrutturazione schema ER: fasi

1. analisi delle ridondanze 
2. eliminazione degli attributi multivalore 
3. eliminazione degli attributi composti 
4. eliminazione delle ISA e delle generalizzazioni 
5. scelta degli identificatori principali 
6. specifica degli ulteriori vincoli esterni 
7. riformulazione delle operazioni e delle specifiche sul carico applicativo in termini dello schema ristrutturato 

Ristrutturazione: fasi 4 e 5



Ristrutturazione: fase 6

5.

6. Vincolo di generalizzazione: ogni istanza di Corso partecipa o a CP_ISA_C o a CNP_ISA_C ma non ad entrambi

Traduzione diretta nel modello relazionale: entità senza accorpamento

1. Corso(nome, n°_crediti)
2. Studente(matricola)
3. Sessione_di_laurea(data)

Traduzione diretta nel modello relazionale: entità con accorpamento

1. Piano di studio (accorpamento della relazione stila):
 - `PianoDiStudio(studente, tipo)`
 - Foreign key: `PianoDiStudio[studente] ⊆ Studente[matricola]`
 - Foreign key: `Studente[matricola] ⊆ PianoDiStudio [studente]`
2. Relazione Finale (accorpamento della relazione autore):
 - `RelazioneFinale(studente, n°pag)`
 - Foreign key: `RelazioneFinale [studente] ⊆ Studente[matricola]`
3. Esame (accorpamento della relazione verbalizza):
 - `Esame(studente, corso, voto, data)`
 - Foreign key: `Esame [studente] ⊆ Studente[matricola]`
 - Foreign key: `Esame[corso] ⊆ Corso[nome]`
4. Corso non progettuale (accorpamento della relazione CNP_ISA_C):
 - `CorsoNonProgettuale(nome)`
 - Foreign key: `CorsoNonProgettuale[nome] ⊆ Corso [nome]`
5. Corso progettuale (accorpamento della relazione CP_ISA_C): :
 - `CorsoProgettuale(nome)`
 - Foreign key: `CorsoProgettuale[nome] ⊆ Corso [nome]`

Traduzione diretta nel modello relazionale: relazioni non accorpate

1. prevede:

- Prevede(pianoDiStudio, corso)
- Foreign key: `prevede[pianoDiStudio] ⊆ PianoDiStudio [studente]`
- Foreign key: `prevede[corso] ⊆ Corso [nome]`
- inclusione: `Corso[corso] ⊆ prevede [corso]`
- inclusione: `PianoDiStudio[studente] ⊆ prevede [pianoDiStudio]`

2. argomento:

- Argomento(relazione, corso)
- Foreign key: `argomento[relazione] ⊆ RelazioneFinale[studente]`
- Foreign key: `argomento[corso] ⊆ Corso[nome]`
- Foreign key: `RelazioneFinale[studente] ⊆ argomento[relazione]`

3. discussione:

- Discussione(relazioneFinale, sessioneDiLaurea, voto)
- Foreign key: `Discussione[relazioneFinale] ⊆ RelazioneFinale[studente]`
- Foreign key: `Discussione[sessioneDiLaurea] ⊆ SessioneDiLaurea[data]`
- inclusione: `SessioneDiLaurea[data] ⊆ discussione[sessioneDiLaurea]`

Schema relazionale completo

Corso(nome, n°_crediti)

inclusione: **Corso**[nome] \subseteq prevede [corso]

Studente(matricola)

Foreign key: **Studente**[matricola] \subseteq PianoDiStudio [studente]

Sessione_di_laurea(data)

inclusione: **SessioneDiLaurea**[data] \subseteq discussione[sessioneDiLaurea]

PianoDiStudio(studente, tipo)

Foreign key: **PianoDiStudio**[studente] \subseteq **Studente**[matricola]

inclusione: **PianoDiStudio**[studente] \subseteq prevede [pianoDiStudio]

RelazioneFinale(studente, n°pag)

Foreign key: **RelazioneFinale** [studente] \subseteq **Studente**[matricola]

Foreign key: **RelazioneFinale**[studente] \subseteq argomento[relazione]

Esame(studente, corso, voto, data)

Foreign key: **Esame** [studente] \subseteq **Studente**[matricola]

Foreign key: **Esame**[corso] \subseteq **Corso**[nome]

CorsoNonProgettuale(nome)

Foreign key: **CorsoNonProgettuale**[nome] \subseteq **Corso** [nome]

CorsoProgettuale(nome)

Foreign key: **CorsoProgettuale**[nome] \subseteq **Corso** [nome]

Prevede(pianoDiStudio, corso)

Foreign key: **prevede**[pianoDiStudio] \subseteq **PianoDiStudio** [studente]

Foreign key: **prevede**[corso] \subseteq **Corso** [nome]

Argomento(relazione, corso)

Foreign key: **argomento**[relazione] \subseteq **RelazioneFinale**[studente]

Foreign key: **argomento**[corso] \subseteq **Corso**[nome]

Discussione(relazioneFinale, sessioneDiLaurea, voto)

Foreign key: **Discussione**[relazioneFinale] \subseteq **RelazioneFinale**[studente]

Foreign key: **Discussione**[sessioneDiLaurea] \subseteq **SessioneDiLaurea**[data]

Vincoli (esterni) sullo schema relazionale

1. **CorsoProgettuale[nome] \in { 'Progetto di applicazioni software', 'Progetto di reti di calcolatori e sistemi informatici' }** (Vincolo di tupla, di stato)
2. Uno studente discute la sua tesi di laurea in una certa sessione solo se tutti gli esami previsti dal suo piano di studio sono stati verbalizzati (con voto maggiore o uguale a 18) (Vincolo inter-tabella, di stato)
3. **PianoDiStudio[tipo] \in { 'PLM', 'Sistemi Informatici' }** (Vincolo di tupla, di stato)
4. Se uno studente ha un piano di studi di tipo PLM, allora deve produrre una relazione finale che verta su un corso di tipo progettuale non compreso nel suo piano di studio (Vincolo inter-tabella, di stato)
5. La somma totale dei crediti previsti dai corsi di un qualunque piano di studio deve essere pari a 174 (Vincolo inter-tabella, di stato)
6. **$70 \leq$ Discussione[voto] ≤ 110** (Vincolo di tupla, di stato)
7. ogni istanza di Corso partecipa o a CP_ISA_C o a CNP_ISA_C ma non ad entrambi (Vincolo inter-tabella, di stato) :
 1. **Corso[nome] \subseteq CorsoProgettuale[nome] \cup CorsoNonProgettuale[nome]** (Completezza)
 2. **CorsoProgettuale[nome] \cap CorsoNonProgettuale[nome] = \emptyset** (Disgiuntezza)

Ristrutturazione dello schema logico

- Assente in questo esempio
- Richiesta nel progetto

Obiettivo 1 (\approx 20 min.)

- Tradurre lo schema logico su un database MySQL*
 - Tralasciare, per ora, tutti i vincoli che non possono essere implementati in maniera dichiarativa nel database MySQL
- *il carattere di backtick [= Alt + 96] permette di creare delle tabelle identificate da qualsiasi stringa, che può essere anche una parola chiave del db

Obiettivo 2.1

1. Creare e testare (20'):

- Una user defined function che, dato un voto di laurea in input (con tipo di dato coerente alla definizione della tabella *Discussione*), restituisca true se il voto di laurea è compreso fra 70 e 110, false altrimenti
- Un trigger che, prima di aggiungere una tupla nella tabella *Discussione*, verifichi se il voto di laurea sia compreso fra 70 e 110, sfruttando la funzione creata precedentemente

2. Creare e testare (20'):

- Una stored procedure con un parametro di input e uno di output: il parametro di input rappresenta un piano di studio (con tipo di dato coerente alla definizione della tabella *PianoDiStudio*), il parametro di output è un booleano che rappresenta il risultato della computazione della stored procedure, che è la seguente: se il piano di studio compare almeno una volta nella tabella *Prevede*, allora il valore che viene scritto nel parametro di output è true, altrimenti viene scritto false
- Un trigger che, prima di aggiungere una tupla nella tabella *PianoDiStudio*, controlli, attraverso la stored procedure creata precedentemente, se il piano di studio compaia almeno una volta nella tabella *Prevede*

3. Creare e testare (20'):

- Una user defined function che, dato l'identificatore di una relazione finale in input (con tipo di dato coerente alla definizione della tabella *RelazioneFinale*), restituisca true se lo studente autore di quella relazione finale può laurearsi, ovvero se per ogni corso previsto dal piano di studi di quello studente esiste un esame, relativo a quel corso, verbalizzato da quello studente, con voto ≥ 18 , false altrimenti
- Un trigger che, prima di inserire una tupla nella tabella *Discussione*, verifichi se la relazione finale può essere discussa, sfruttando la funzione creata precedentemente

Obiettivo 2.2: a casa

1. Creare una store procedure (o una funzione) per ognuno dei vincoli esterni 1,4,5,7 che decida se quel vincolo è rispettato o meno, sulla base degli esercizi svolti per raggiungere l'obiettivo 2.1
2. Creare inoltre dei trigger che si attivino opportunamente per controllare se il vincolo sia soddisfatto o meno

Approfondimento: presenza di vincoli ciclici nello schema relazionale

Corso(nome, n°_crediti)

inclusione: $\text{Corso}[\text{nome}] \subseteq \text{prevede}[\text{corso}]$

Studente(matricola)

Foreign key: $\text{Studente}[\text{matricola}] \subseteq \text{PianoDiStudio}[\text{studente}]$

Sessione_di_laurea(data)

inclusione: $\text{SessioneDiLaurea}[\text{data}] \subseteq \text{discussione}[\text{sessioneDiLaurea}]$

PianoDiStudio(studente, tipo)

Foreign key: $\text{PianoDiStudio}[\text{studente}] \subseteq \text{Studente}[\text{matricola}]$

inclusione: $\text{PianoDiStudio}[\text{studente}] \subseteq \text{prevede}[\text{pianoDiStudio}]$

RelazioneFinale(studente, n°pag)

Foreign key: $\text{RelazioneFinale}[\text{studente}] \subseteq \text{Studente}[\text{matricola}]$

Foreign key: $\text{RelazioneFinale}[\text{studente}] \subseteq \text{argomento}[\text{relazione}]$

Esame(studente, corso, voto, data)

Foreign key: $\text{Esame}[\text{studente}] \subseteq \text{Studente}[\text{matricola}]$

Foreign key: $\text{Esame}[\text{corso}] \subseteq \text{Corso}[\text{nome}]$

CorsoNonProgettuale(nome)

Foreign key: $\text{CorsoNonProgettuale}[\text{nome}] \subseteq \text{Corso}[\text{nome}]$

CorsoProgettuale(nome)

Foreign key: $\text{CorsoProgettuale}[\text{nome}] \subseteq \text{Corso}[\text{nome}]$

Prevede(pianoDiStudio, corso)

Foreign key: $\text{prevede}[\text{pianoDiStudio}] \subseteq \text{PianoDiStudio}[\text{studente}]$

Foreign key: $\text{prevede}[\text{corso}] \subseteq \text{Corso}[\text{nome}]$

Argomento(relazione, corso)

Foreign key: $\text{argomento}[\text{relazione}] \subseteq \text{RelazioneFinale}[\text{studente}]$

Discussione(relazioneFinale, sessioneDiLaurea, voto)

Foreign key: $\text{Discussione}[\text{relazioneFinale}] \subseteq \text{RelazioneFinale}[\text{studente}]$

↔ = ciclo tra foreign key

↔ = ciclo tra inclusioni/
foreign key

Gestione dei vincoli ciclici

- Come popolare uno schema in cui compaiono dei vincoli ciclici?
- Una soluzione, per ora, è quella di disabilitare momentaneamente la coppia di vincoli ciclici, effettuare gli inserimenti necessari, e riabilitare quei vincoli.
- Per quanto riguarda i trigger, MySQL non permette in modo nativo di disabilitare temporaneamente un trigger.
Workaround: si può utilizzare una variabile globale `@DISABLE_TRIGGERS` nel body del trigger. Se quella variabile non è NULL, allora il trigger viene eseguito, altrimenti non fa nulla
- Diverso è il caso delle foreign key...

Vincoli ciclici in MySQL: foreign key

- Uso del parametro *foreign_key_checks*
- If set to 1 (the default), foreign key constraints for InnoDB tables are checked. If set to 0, they are ignored. Disabling foreign key checking can be useful for reloading InnoDB tables in an order different from that required by their parent/child relationships. [See [Section 13.6.4.4, “FOREIGN KEY Constraints”](#)].
- Setting [foreign_key_checks](#) to 0 also affects data definition statements: [DROP SCHEMA](#) drops a schema even if it contains tables that have foreign keys that are referred to by tables outside the schema, and [DROP TABLE](#) drops tables that have foreign keys that are referred to by other tables.
- **NOTE:** Setting [foreign_key_checks](#) to 1 does not trigger a scan of the existing table data. Therefore, rows added to the table while [foreign_key_checks = 0](#) will not be verified for consistency.