Basi di dati e programmazione web

Lezione 4

Prof. Paolo Ciaccia
paolo.ciaccia@unibo.it

DEIS – Università di Bologna

Argomenti della lezione

- Progettazione di DB
  - Concettuale
    - modello E/R
    - metodologia di progettazione
  - Logica
    - traduzione da E/R a relazionale
    - mismatch di espressività tra E/R e il DDL di SQL
    - normalizzazione di schemi relazionali

Progettazione di DB

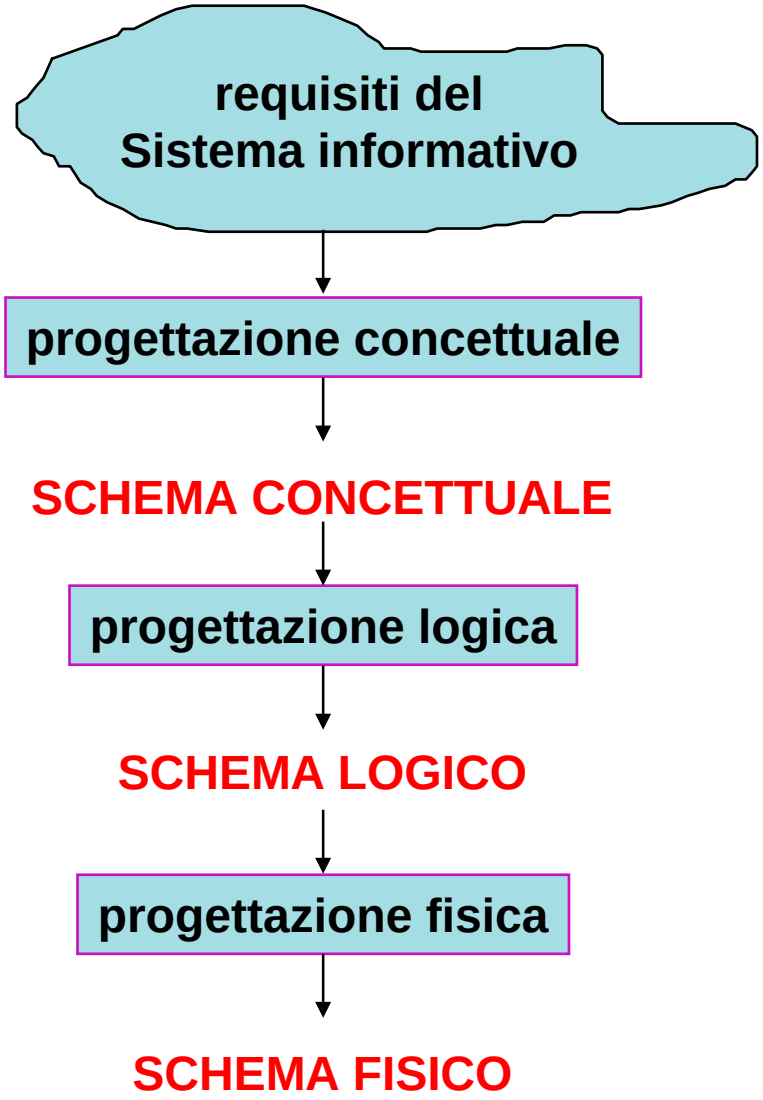
Progettazione “guidata dai dati”

- Progettare un’applicazione basata su DB richiede attività di:
  - Progettazione dei dati
  - Progettazione delle applicazioni

- Il ruolo primario viene svolto dai dati, in quanto:
  - Strutturalmente più stabili
  - Condivisibili da più applicazioni
  - Minore dipendenza dalla tecnologia utilizzata

- Approccio comune: si progetta innanzitutto la base di dati e successivamente le applicazioni

Progettazione della base di dati



Modelli dei dati: logici vs concettuali

Un modello dei dati è una collezione di concetti
che vengono utilizzati per descrivere i dati,
le loro associazioni, e i vincoli che questi devono rispettare

Modelli logici: utilizzati nei DBMS per l'organizzazione dei dati

Modelli concettuali: permettono di rappresentare i dati in modo indipendente dal sistema, lo scopo è cercare di descrivere al meglio i concetti del mondo reale che si devono rappresentare

- modelli semantici, RM/T, ... [inizio anni '70]
- Entity-Relationship (E/R) [entità-associazione] [Chen 1976]
- IDEF1X [standard adottato dagli uffici governativi USA]
- UML (Universal Modelling Language) [1999]

Tutto quello che avreste voluto sapere sul modello E/R…

Modello Entity-Relationship

- Standard *de facto* per la progettazione concettuale
- Ma NON esiste uno standard (molti dialetti, molte rappresentazioni di tipo grafico degli stessi concetti)

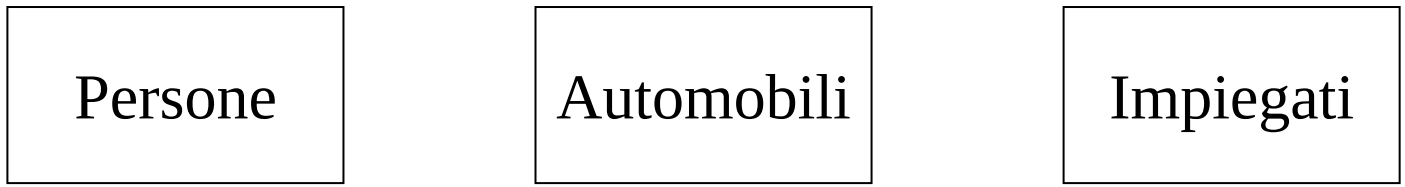
- Concetti fondamentali:

  - Entità (entity)
  - Associazione (relationship)
  - Attributo
  - Identificatore

  - Vincolo di cardinalità
  - Gerarchia

Modello E/R: Entità

- Insieme (classe) di oggetti della realtà di interesse che possiedono caratteristiche comuni e che hanno esistenza “autonoma”
- L’istanza (elemento) di un’entità è uno specifico oggetto appartenente a quella entità

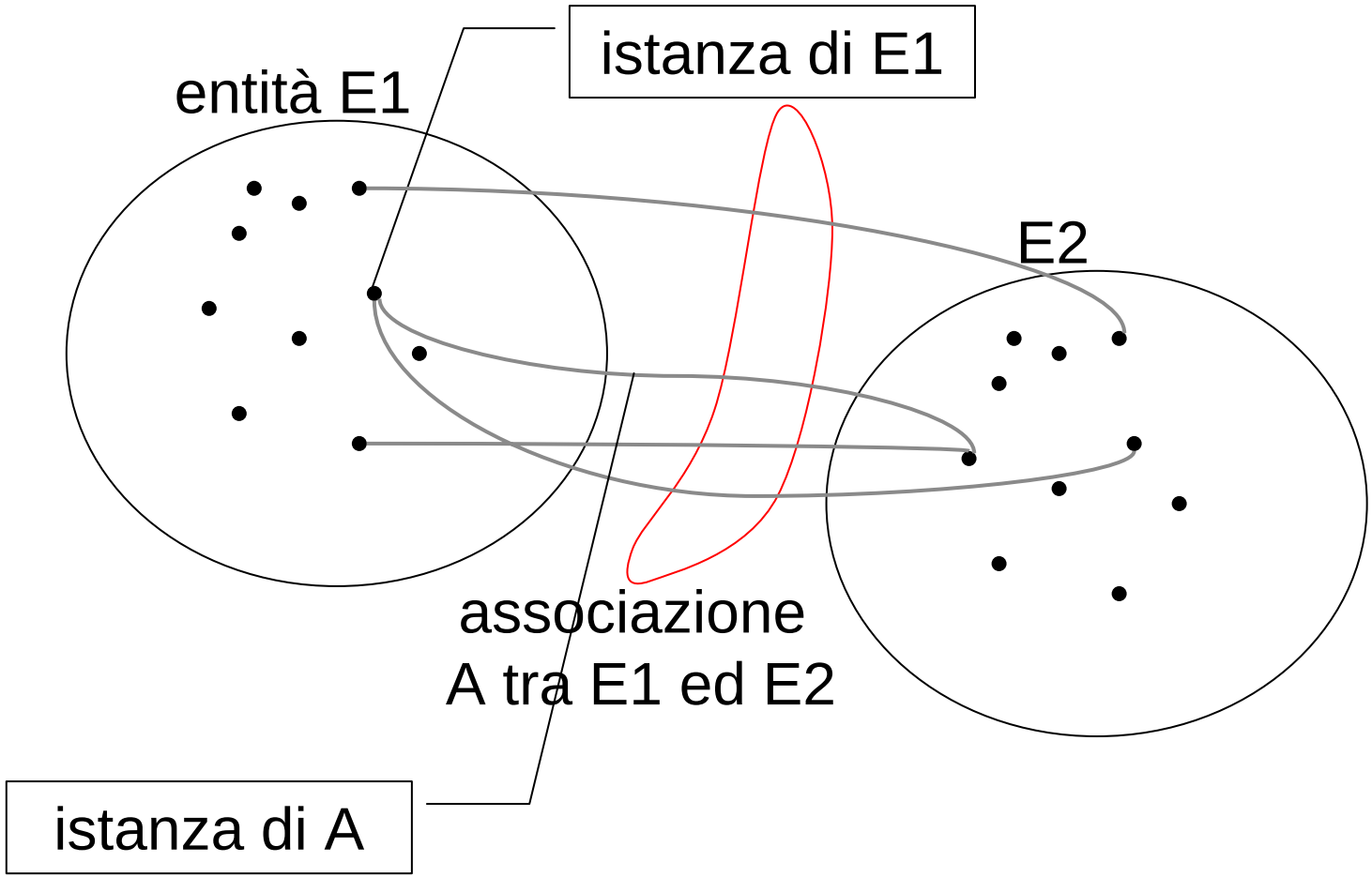


Modello E/R: Associazione

- Rappresenta un legame logico tra entità, rilevante nella realtà che si sta considerando
- Istanza di associazione: combinazione (aggregazione) di istanze delle entità che prendono parte all'associazione
- Graficamente:

- Se p è un'istanza di Persone e c è un'istanza di Città,
  la coppia (p, c) è un'istanza dell'associazione Risiedono

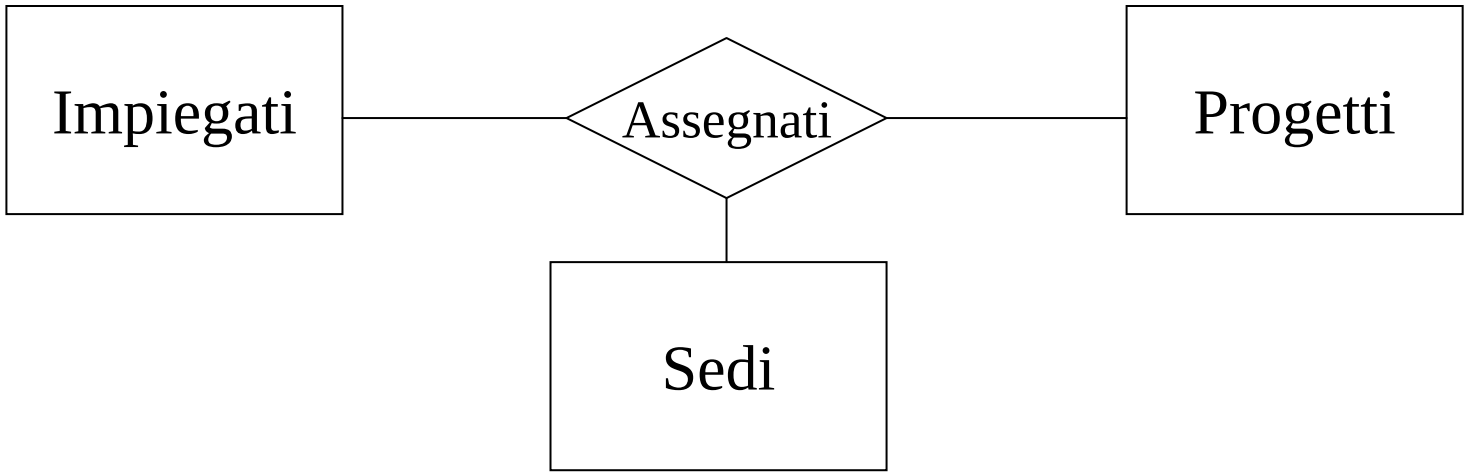
A livello di istanze…



Grado delle associazioni

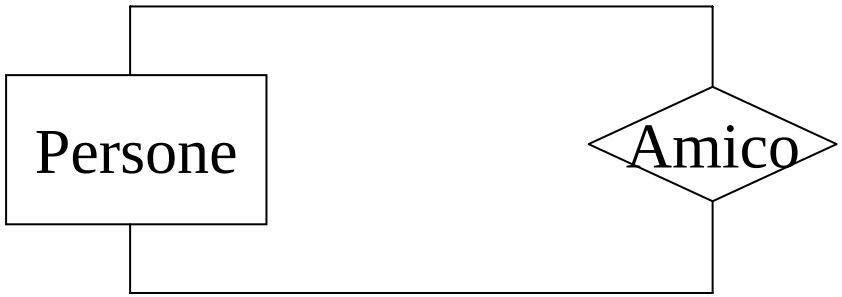
- È il numero di istanze di entità che sono coinvolte in un'istanza dell'associazione
- associazione binaria: grado = 2

- associazione ternaria: grado = 3

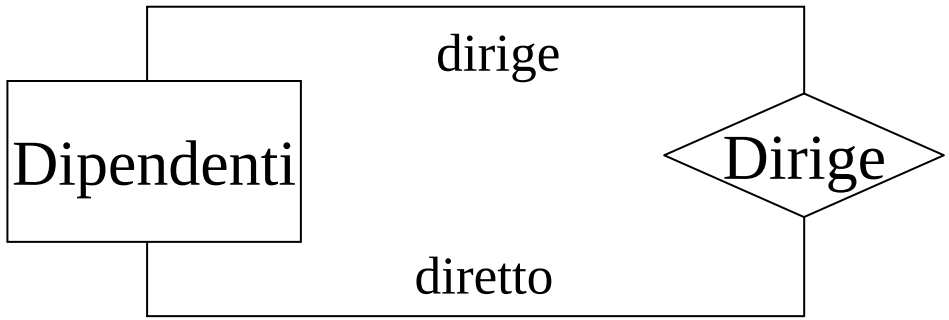


Associazioni ad anello (1)

- Un’associazione ad anello coinvolge più volte la stessa entità

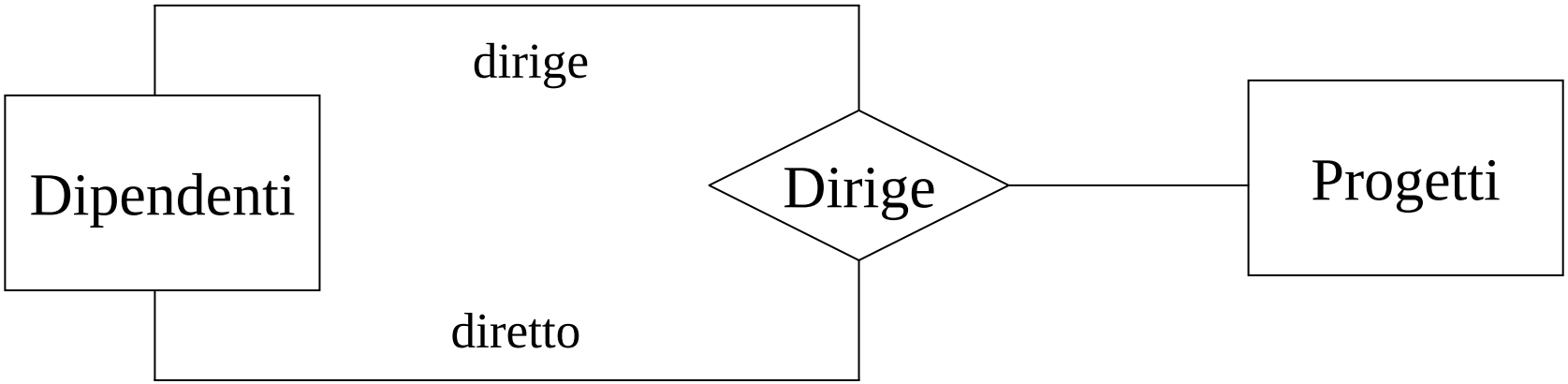


- E’ necessario specificare, per ogni ramo dell’associazione, il relativo ruolo



Associazioni ad anello (2)

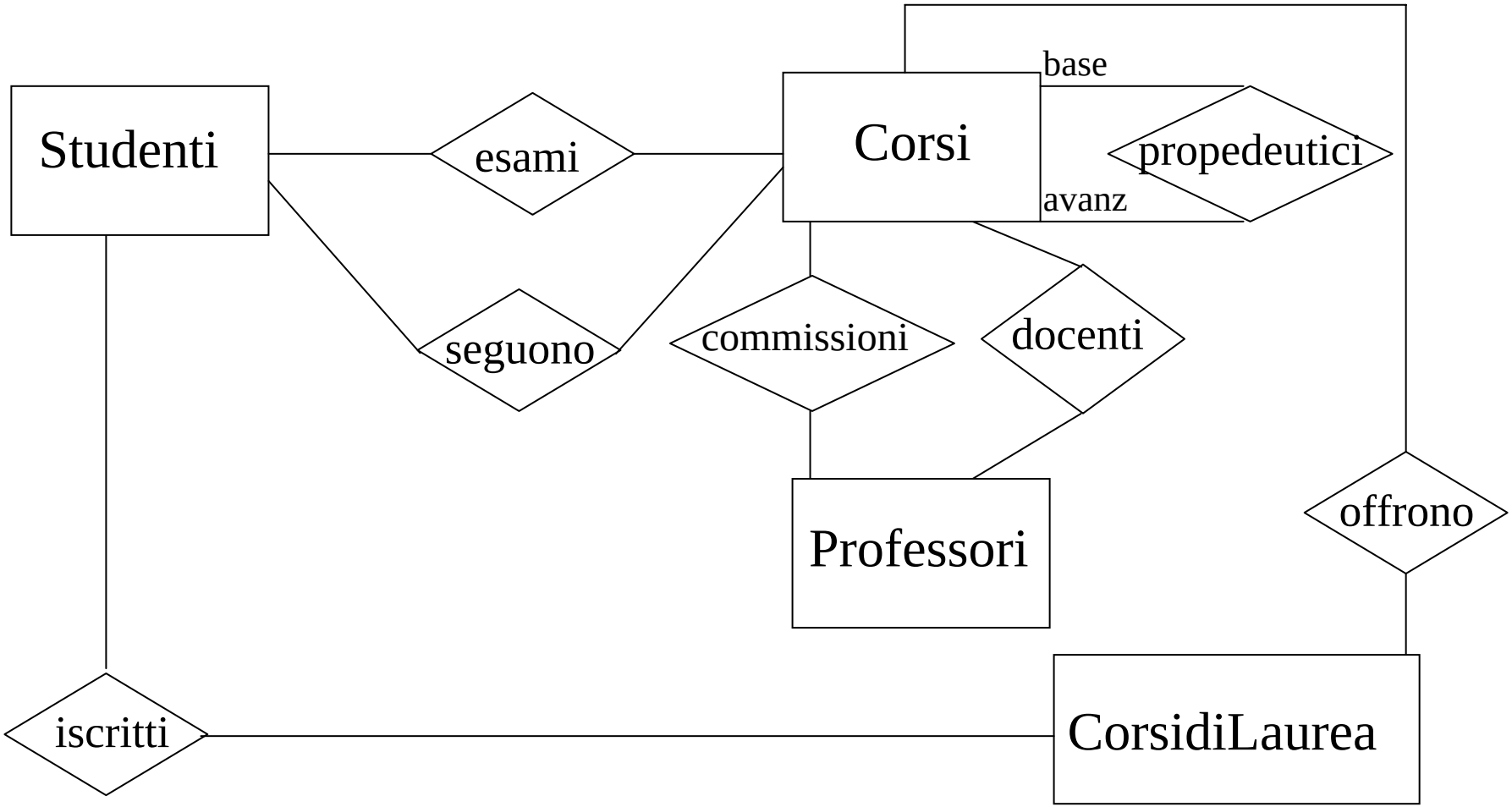
- È possibile avere anelli anche in relazioni n-arie generiche (n > 2)



- L'interpretazione di un'istanza (d1,d2,p) è:

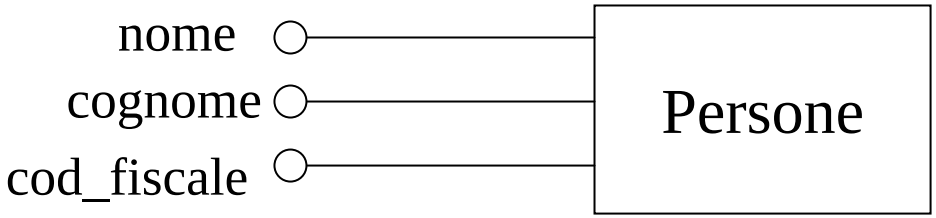
il dipendente d1 *dirige il dipendente* d2 *all'interno del progetto* p

Un semplice schema E/R (incompleto)



Attributi

- Un atttributo è una proprietà elementare di un'entità o di un'associazione
- Graficamente:

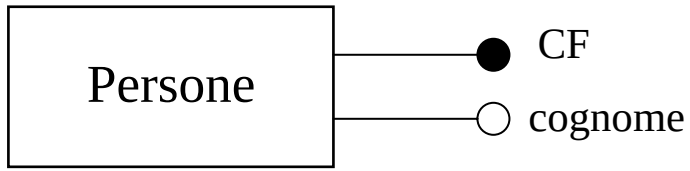


Identificatori

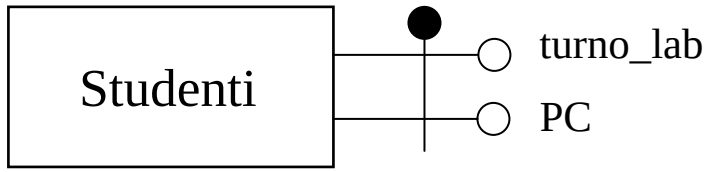
- Un identificatore permette l'individuazione univoca delle istanze di un'entità; deve valere anche la minimalità: nessun sottoinsieme proprio dell'identificatore deve a sua volta essere un identificatore
  - Corrisponde al concetto di chiave del modello relazionale

- Per definire un identificatore per un'entità E si hanno due possibilità:
  - Identificatore interno: si usano uno o più attributi di E
  - Identificatore esterno: si usano altre (una o più) entità, collegate a E da associazioni, più eventuali attributi di E
    - Talvolta quando l'identificatore usa sia altre entità che attributi propri si parla di identificatore misto

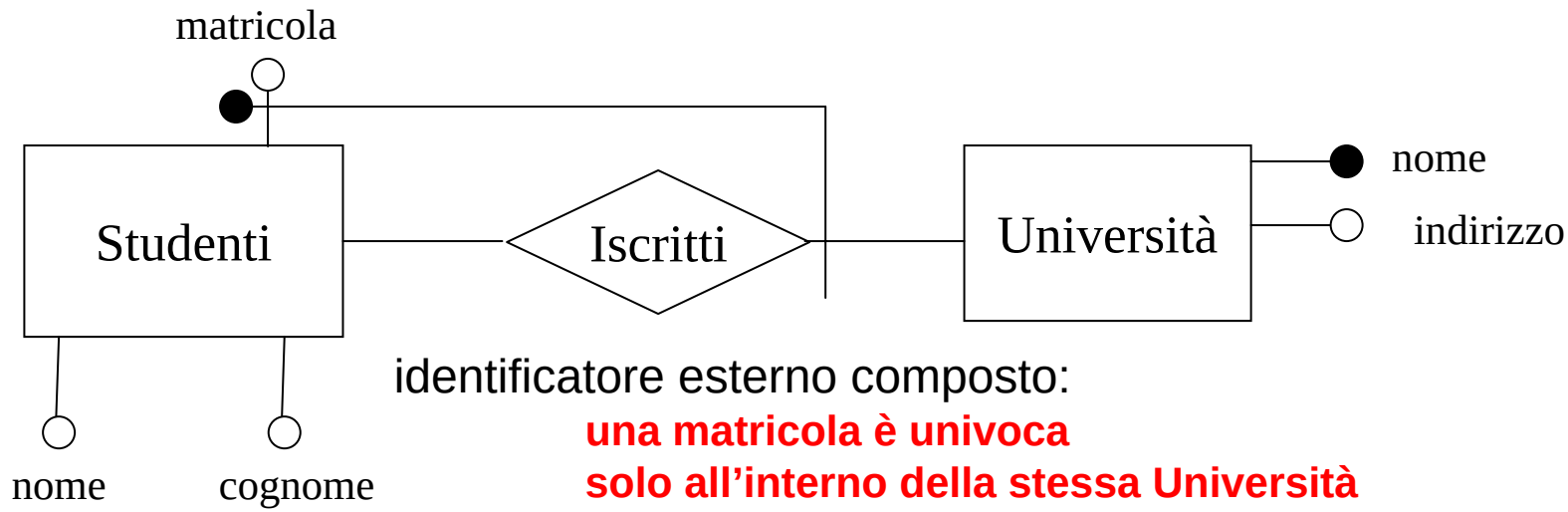
Identificatori interni ed esterni



identificatore interno semplice



identificatore interno composto



identificatore esterno composto:

una matricola è univoca

solo all'interno della stessa Università

Differenze con il modello relazionale (1)

- Nel modello relazionale abbiamo, per ogni relazione:
  - Una o più chiavi
  - Una chiave primaria
- La chiave primaria viene poi “esportata”, definendo così delle foreign keys
- Quindi: per definire una foreign key dobbiamo aver prima definito qual è la chiave primaria della relazione che vogliamo referenziare

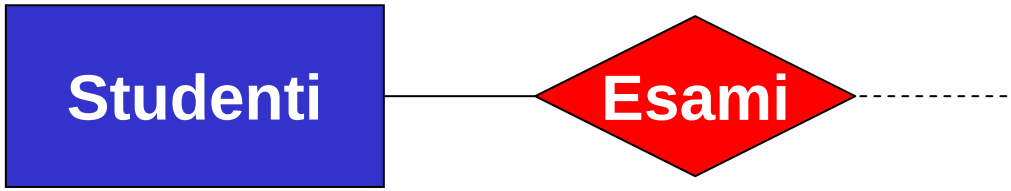
Studenti

Matricola	CodiceFiscale	Cognome	Nome	DataNascita
29323	BNCGRG78F21A	Bianchi	Giorgio	21/06/1978
35467	RSSNNA78D13A	Rossi	Anna	13/04/1978
39654	VRDMRC79I20A	Verdi	Marco	20/09/1979
42132	VRDMRC79I20B	Verdi	Marco	20/09/1979

Se in Esami vogliamo referenziare la primary key di Studenti dobbiamo prima scegliere se è Matricola o CodiceFiscale!

Differenze con il modello relazionale (2)

- Nel modello E/R il “riferimento” di un’associazione a un’entità è esplicito nello schema, anche quando non è stato ancora definito alcun identificatore



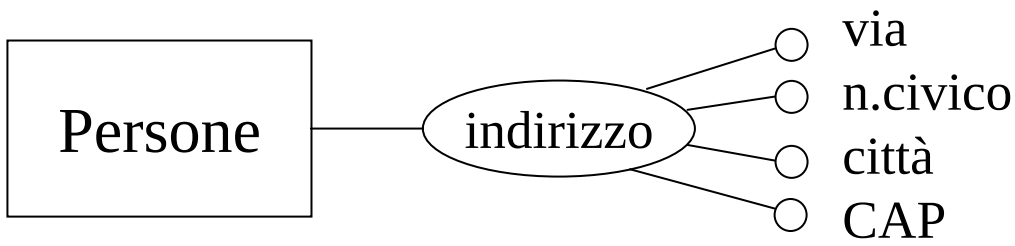
- Lo schema mi dice già che

  ogni istanza di Esami referenzia una specifica istanza di Studenti

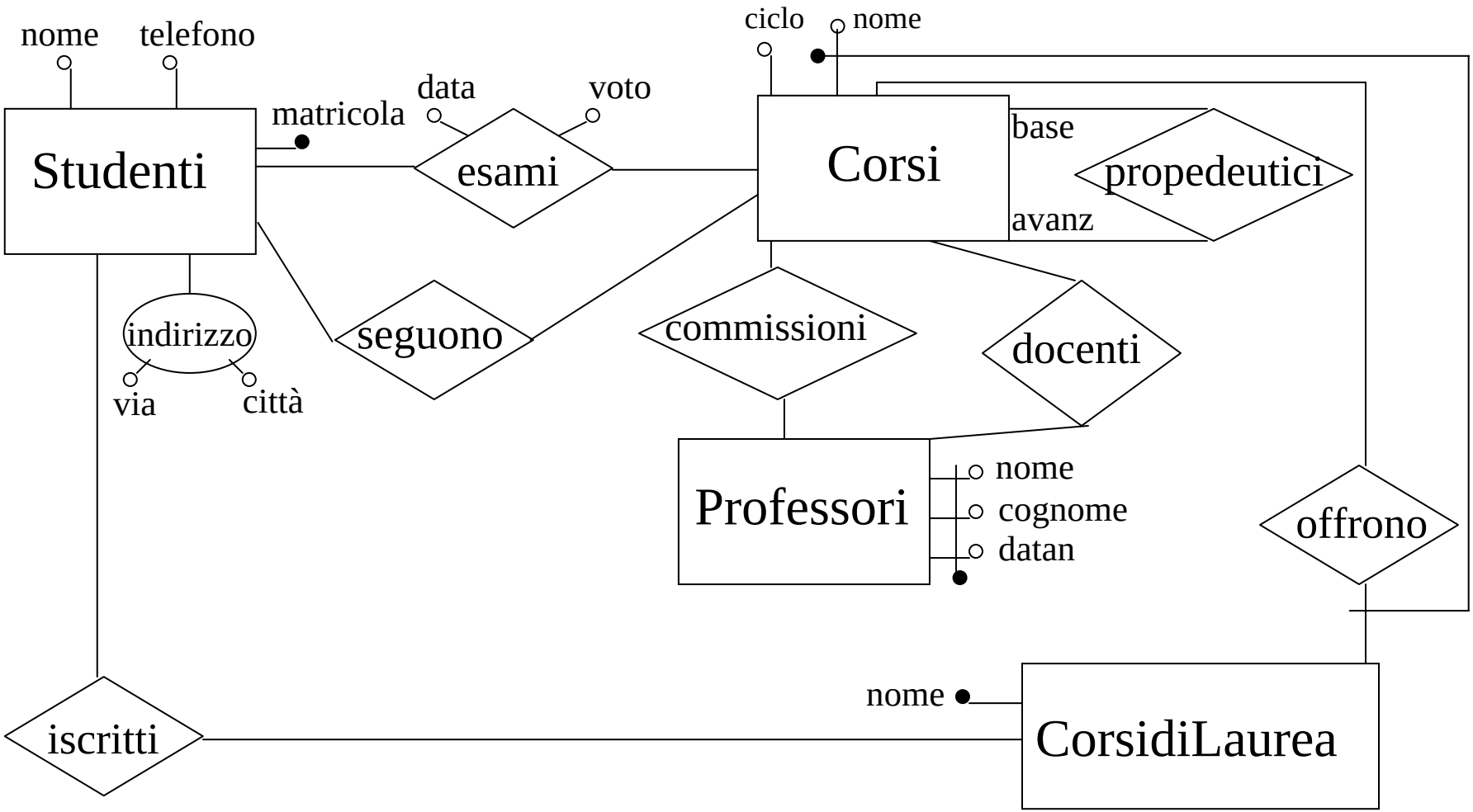
- Come? A questo livello di dettaglio non è necessario saperlo,
  si può stabilire succesivamente

Attributi composti

- Sono attributi che si ottengono aggregando altri (sotto-)attributi, i quali presentano una forte affinità nel loro uso e significato



Uno schema E/R (ancora incompleto)



Vincoli nel modello Entity-Relationship

- In ogni schema E/R sono presenti dei vincoli

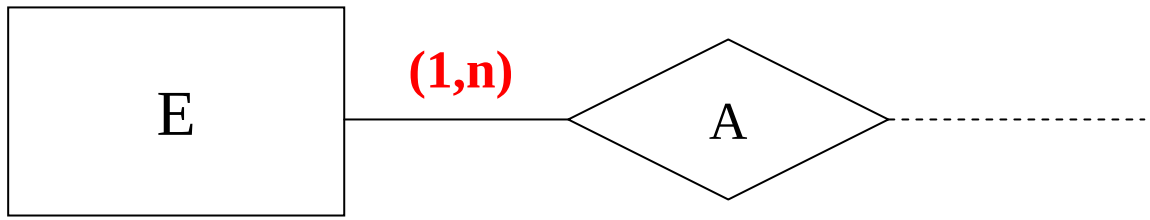
- Alcuni sono impliciti, in quanto dipendono dalla semantica stessa dei costrutti del modello:
  - ogni istanza di associazione deve riferirsi ad istanze di entità
  - istanze diverse della stessa associazione devono riferirsi a differenti combinazioni di istanze delle entità partecipanti all'associazione
  - …

- Altri vincoli sono espliciti, e vengono definiti da chi progetta lo schema E/R sulla base della conoscenza della realtà che si sta modellando
  - vincoli di cardinalità
  - vincoli di identificazione

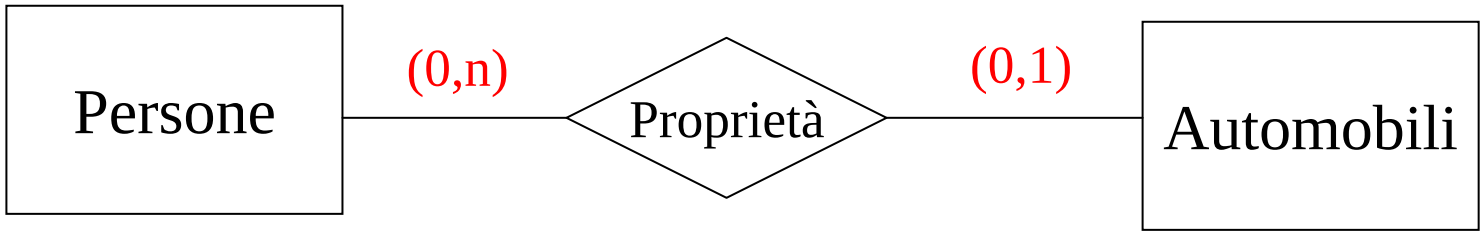
Associazioni: vincoli di cardinalità

- Sono coppie di valori (min-card,max-card) associati a ogni entità che partecipa a un'associazione, che specificano il numero minimo e massimo di istanze dell'associazione a cui un'istanza dell'entità può partecipare

- Ad esempio, se i vincoli di cardinalità per un'entità E relativamente a un'associazione A sono (1,n) questo significa che:
  - ogni istanza di E partecipa almeno ad una istanza di A
    - min-card = 1
  - ogni istanza di E può partecipare a più istanze di A (senza limiti)
    - max-card = n



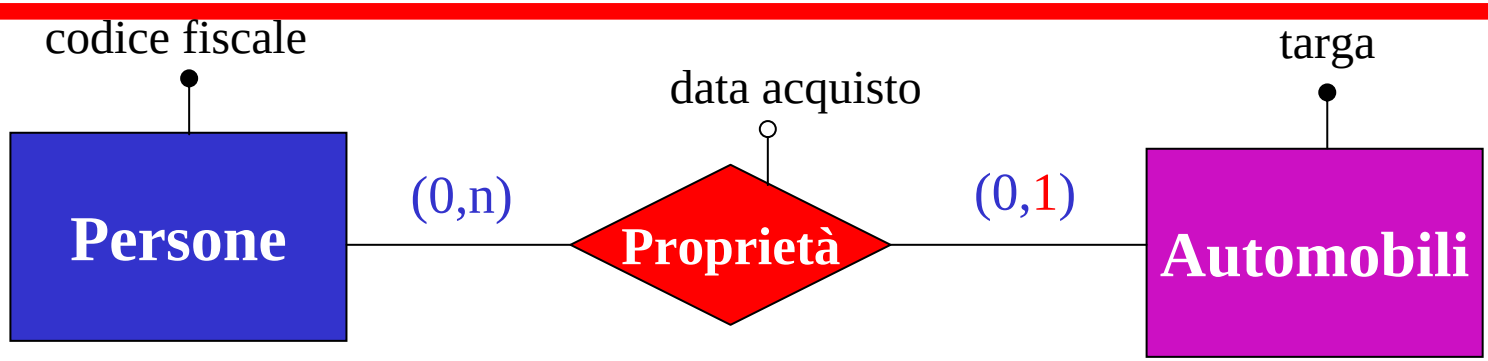
Vincoli di cardinalità: un esempio



- min-card(Automobili,Proprietà) = 0: esistono automobili non possedute da alcuna persona
- max-card(Automobili,Proprietà) = 1: ogni automobile può avere al più un proprietario
- min-card(Persone,Proprietà) = 0: esistono persone che non posseggono alcuna automobile
- max-card(Persone,Proprietà) = n: ogni persona può essere proprietaria di un numero arbitrario di automobili

Si noti che i vincoli si possono stabilire correttamente solo se è ben chiaro cosa rappresentano le diverse entità (analisi delle specifiche)

Perché sono importanti?



- Un possibile modo di tradurre Proprietà:

Proprietà

CF	DataAcquisto	Targa
BLGSTR71B22	12/08/2004	CT 001 MJ
BLGSTR71B22	15/07/2003	CM 415 EF
FDLNNR66M45	12/06/2003	CL 217 HK
...	...	...

- Un'automobile ha al massimo un proprietario
- Quindi non esistono valori ripetuti di Targa in Proprietà
- Quindi Targa è chiave di Proprietà!

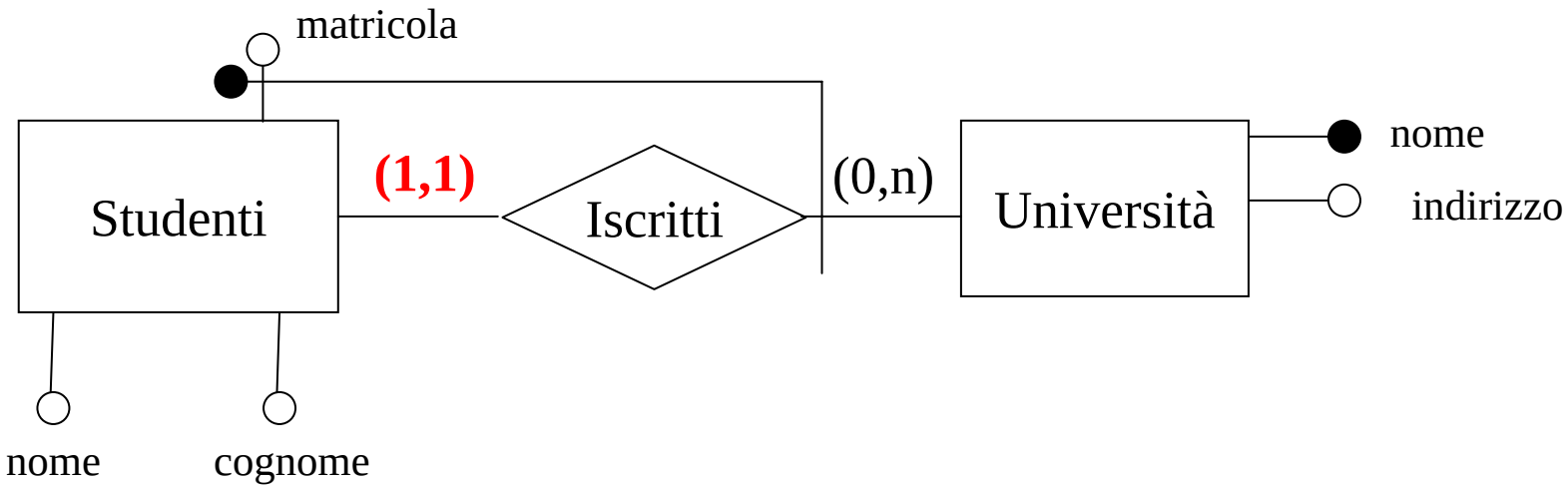
Tipi di associazione: terminologia

- Nel caso di un'associazione binaria A tra due entità E1 ed E2 (non necessariamente distinte), si dice che:
  - A è uno a uno se le cardinalità massime di entrambe le entità rispetto ad A sono 1
  - A è uno a molti se max -card(E1,A) = 1 e max-card(E2,A) = n, o viceversa
  - A è molti a molti se max-card(E1,A) = n e max-card(E2,A) = n

- Si dice inoltre che:
  - La partecipazione di E1 in A è opzionale se min-card(E1,A) = 0
  - La partecipazione di E1 in A è obbligatoria (o totale) se min-card(E1,A) = 1

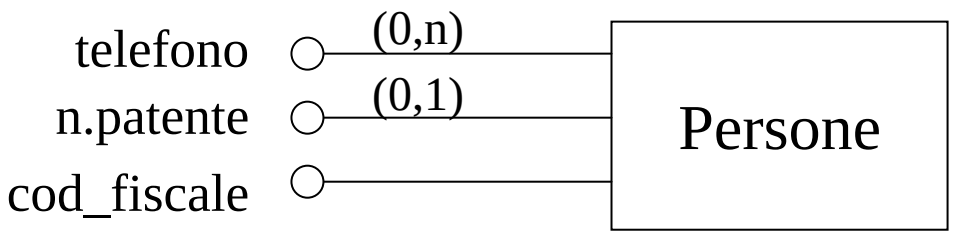
Vincoli di cardinalità e identificatori esterni

- Se E è identificata esternamente attraverso l'associazione A, allora si ha, sempre, min-card(E,A) = max-card(E,A) =1



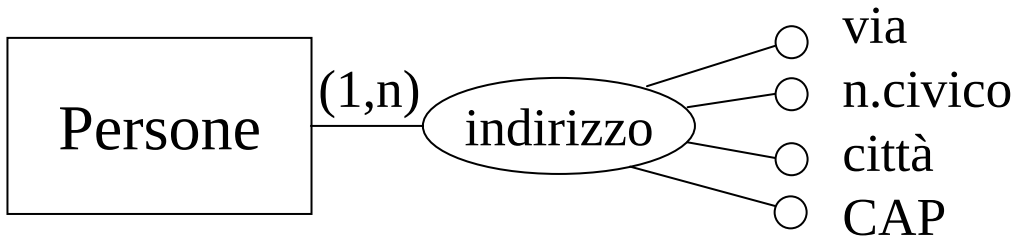
Attributi: vincoli di cardinalità

- Anche per gli attributi è possibile specificare il numero minimo e massimo di valori dell'attributo che possono essere associati ad un'istanza della corrispondente associazione o entità
- Si parla di attributi:
  - opzionali: se la cardinalità minima è 0 (es. n. patente)
  - monovalore: se la cardinalità massima è 1 (es. cod_fiscale)
  - multivalore (o ripetuti): se la cardinalità massima è n (es. telefono)

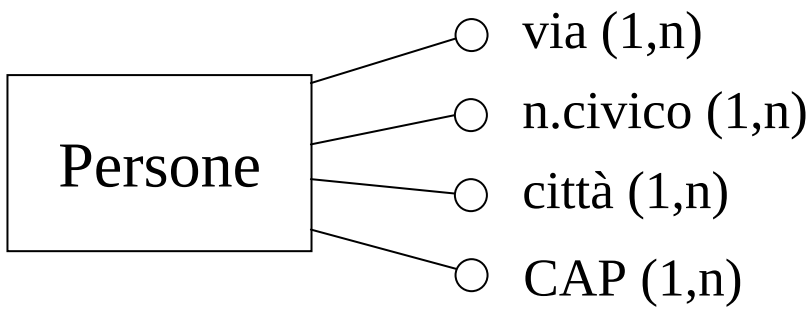


Attributi ripetuti e composti

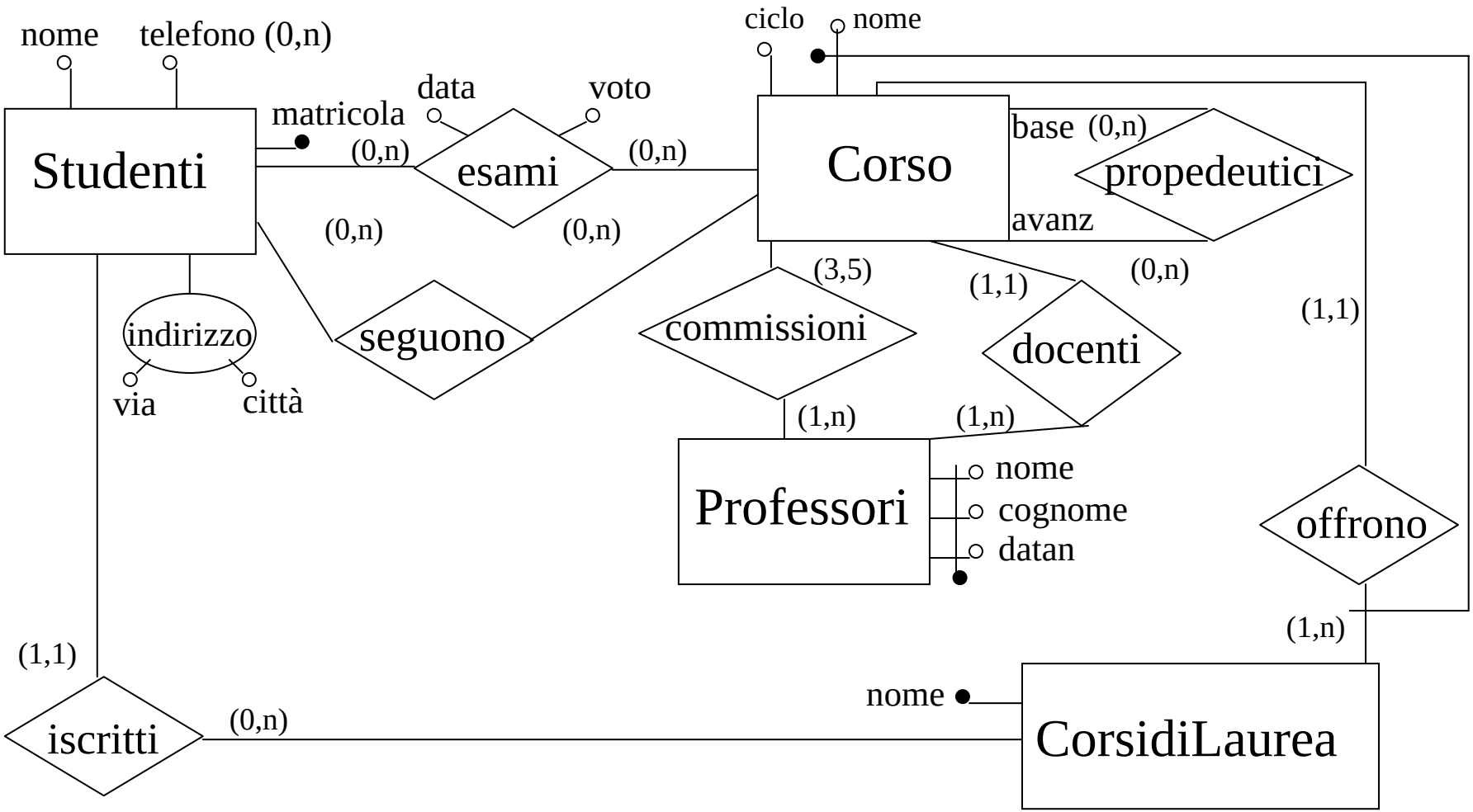
- Nel caso di presenza di più attributi multivalore, la creazione di un attributo composto può rendersi necessaria per evitare ambiguità
- Ad esempio, se una persona ha più indirizzi…



…non si può rappresentarlo così:

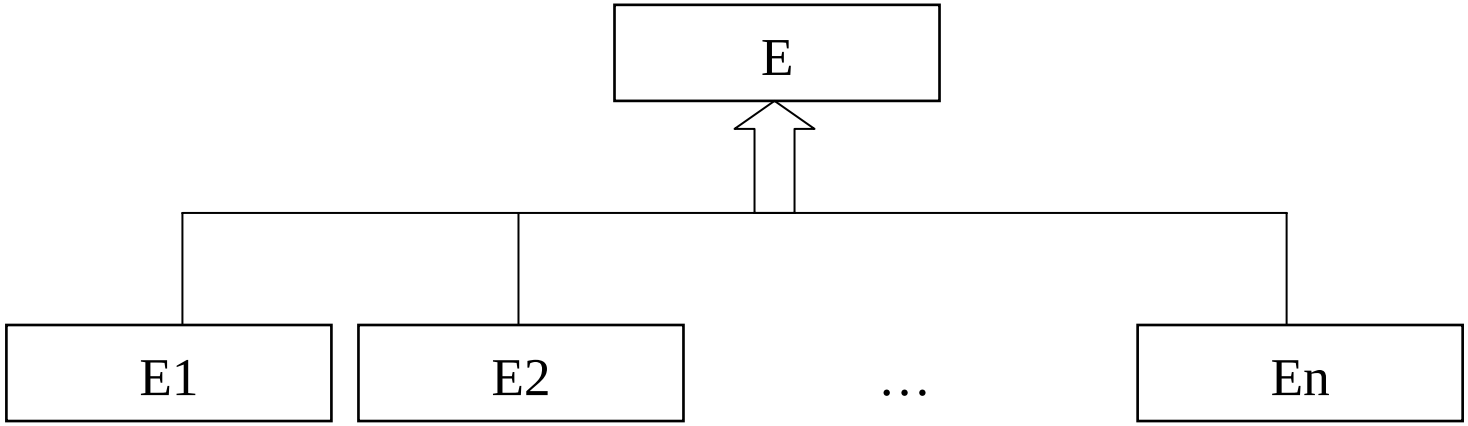


Uno schema E/R completo



Gerarchie di generalizzazione

- Un'entità E è una generalizzazione di un gruppo di entità E1, E2, ..., En se ogni istanza di E1, E2, ..., En è anche un'istanza di E
- Le entità E1, E2, ... En sono dette specializzazioni di E

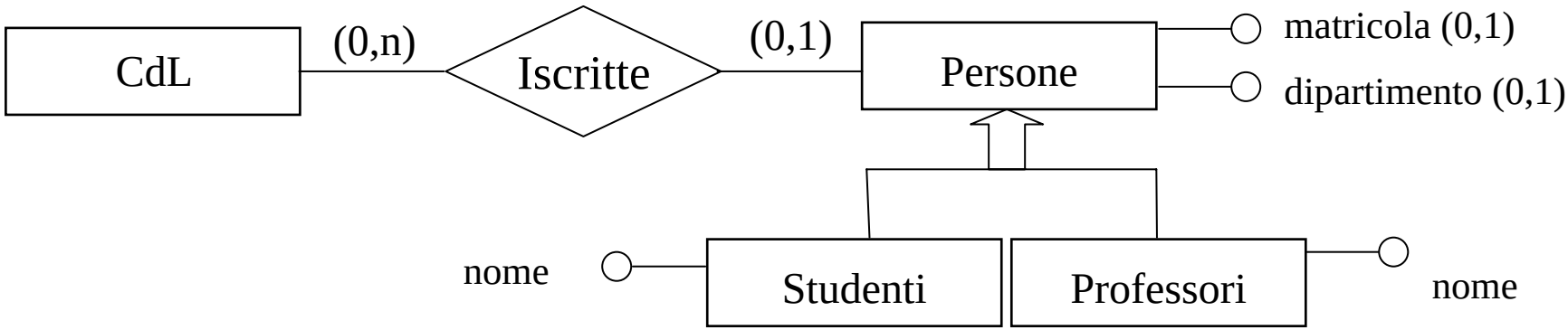


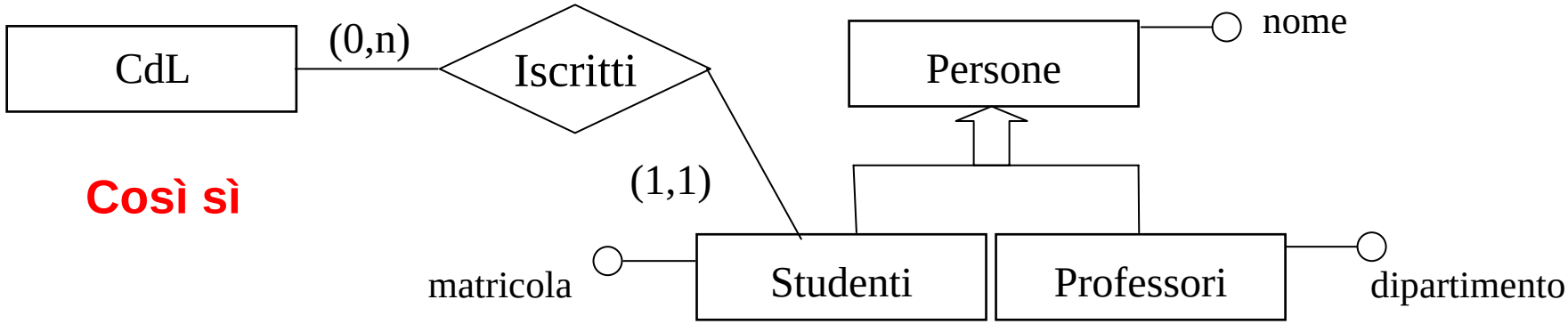
- Le proprietà di E sono ereditate da E1, E2, ..., En: ogni Ei ha gli attributi di E e partecipa alle associazioni definite per E

Ereditarietà delle proprietà

- Gli attributi vanno riferiti all'entità più generica in cui sono presenti obbligatoriamente; analogamente per le associazioni

Quindi così non va bene:





Così sì

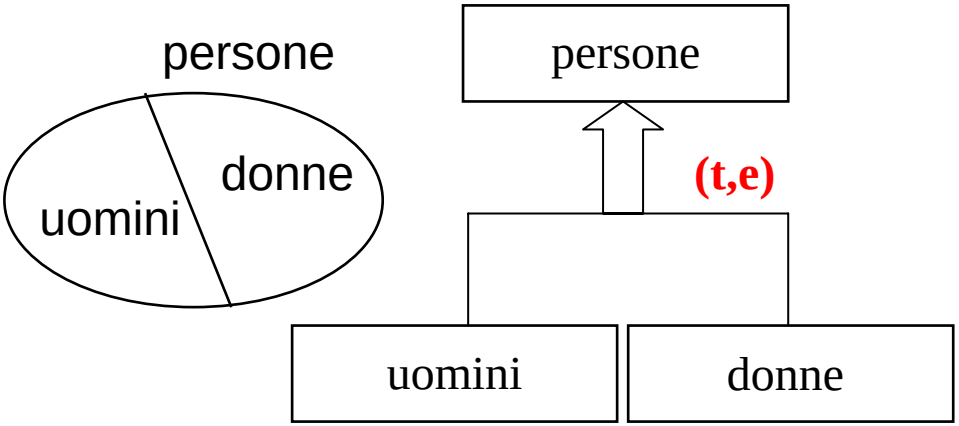
Copertura delle generalizzazioni

- Le generalizzazioni si caratterizzano per due dimensioni indipendenti

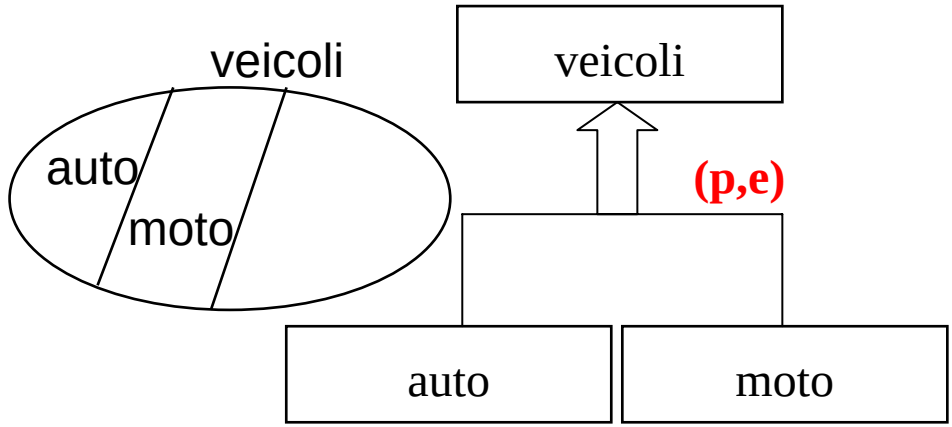
- Confronto fra unione delle specializzazioni e classe generalizzata
  - **t**otale se la classe generalizzata è l'unione delle specializzazioni
  - **p**arziale se la classe generalizzata contiene l'unione delle specializzazioni
- Confronto fra le classi specializzate
  - **e**sclusiva se le specializzazioni sono fra loro disgiunte
  - sovrapposta (**o**verlapped) se può esistere una intersezione non vuota fra le specializzazioni
- Sono ovviamente possibili quattro combinazioni:

  (t,e) (p,e) (t,o) (p,o)

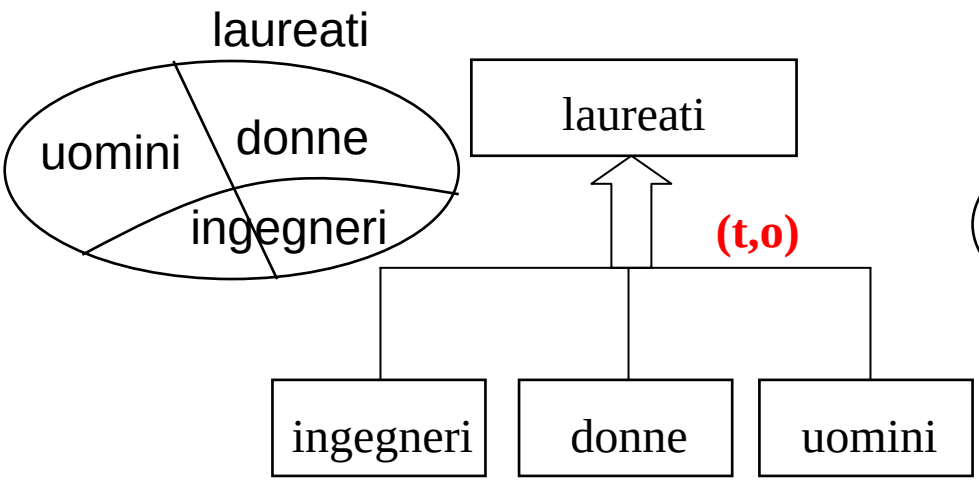
Proprietà di copertura - esempi



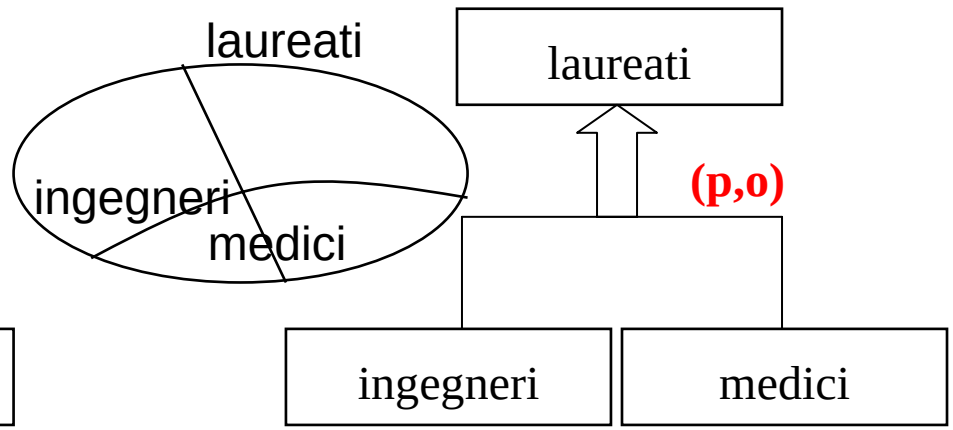
Totale esclusiva



Parziale esclusiva

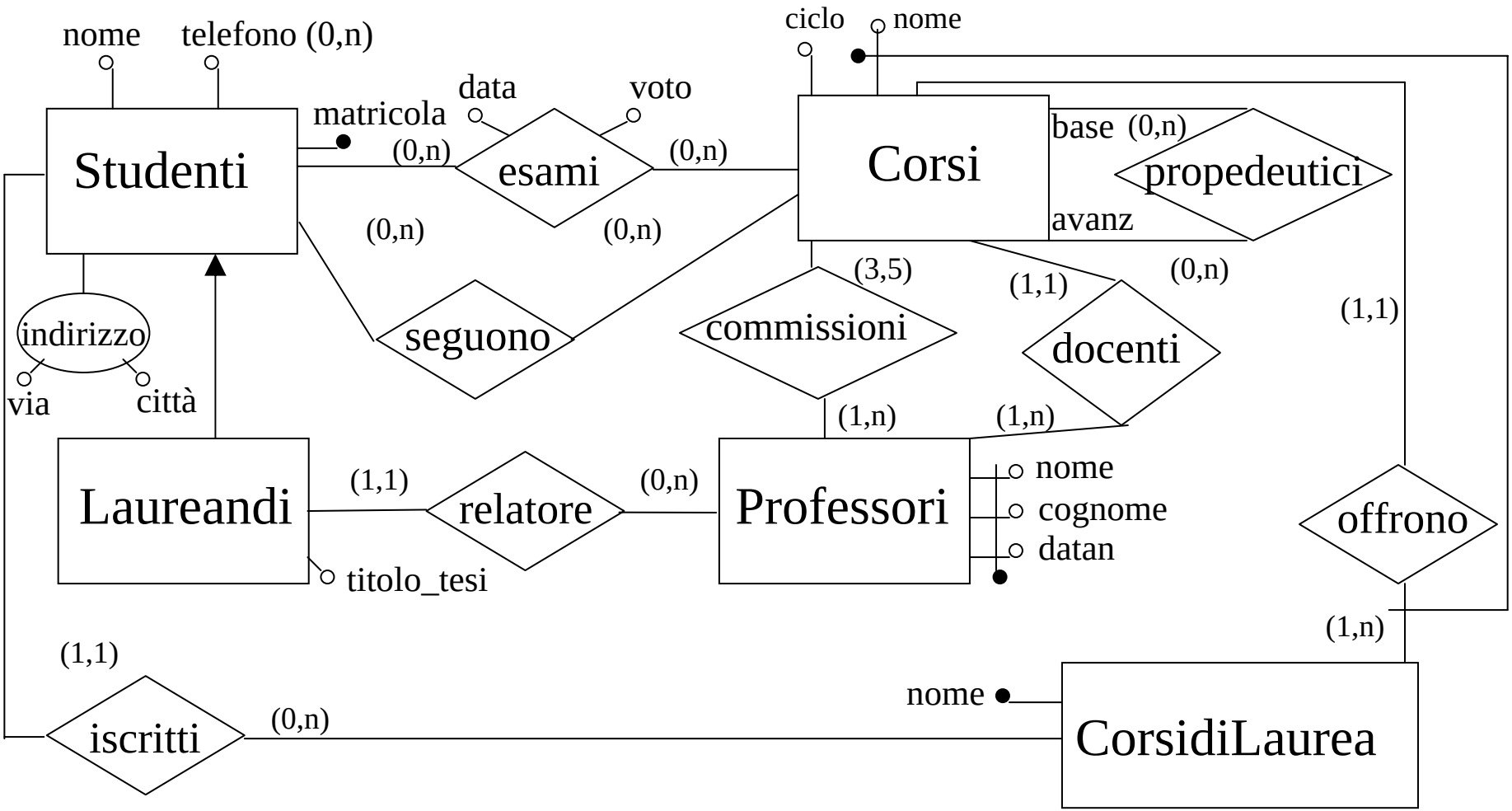


Totale sovrapposta

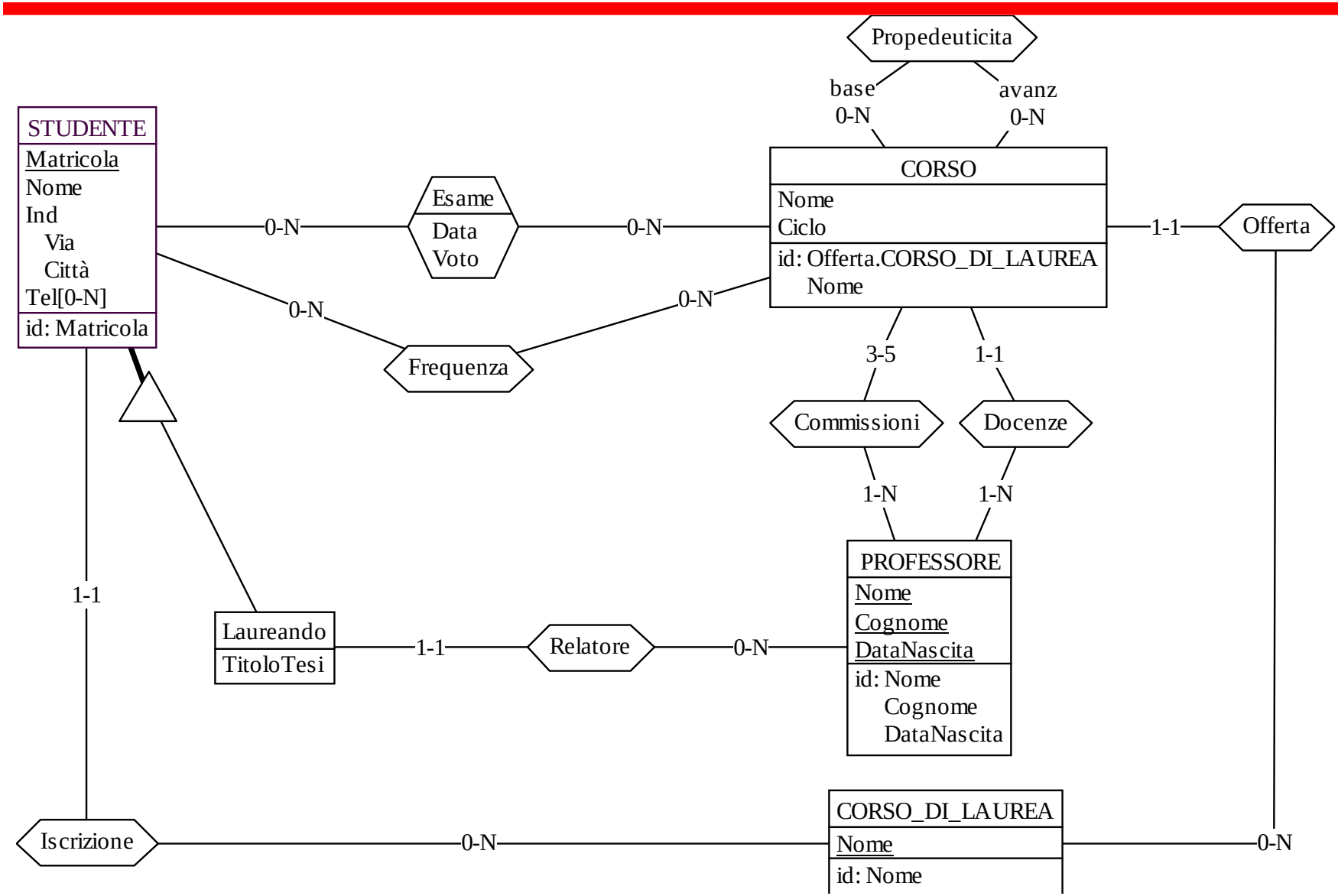


Parziale sovrapposta

Uno schema E/R con gerarchie



Lo stesso schema E/R con DB-Main



DB-Main http://www.db-main.eu

DB-MAIN: a data-architecture tool

DB-MAIN is **a data-modeling and data-architecture tool**. It is designed to help developers and analysts in most data engineering processes, including:

- *Design processes*: requirement analysis, conceptual design, normalisation, schema integration, logical design, physical design, schema optimisation, code generation.
- *Transformations*: schema transformation, model transformation, ETL.
- *Reverse engineering and program understanding*: schema analysis, code analysis, data and data flow reverse engineering, impact analysis.
- *Maintenance, evolution and integration*: database migration, database evolution, database integration and federation, data wrapper design and generation.
- *And many other domains* like temporal and active databases, datawarehouse, XML engineering, ...

DB-MAIN also includes **meta-modeling components** that allow its users to develop new functions and extend its repository.

DB-MAIN is free of charge (without object and functionality limit).

History

DB-MAIN was first conceived as part of a research, development and technology transfer project initiated in 1991 by the LIBD Laboratory (University of Namur).

DB-MAIN is based on ORGA, the first CASE tool marketed back in 1986. It is the result of years of research, and studies are still ongoing. All PHD theses at laboratory are based on DB-MAIN. In the past they have all helped improve the tool kernel or plugins.

Since January 2004, DB-MAIN is developed and marketed by REVER S.A.

- Sviluppato in C++
- Attesa a breve una versione per Linux

Nota

- I concetti che definiscono il modello E/R possono essere modellati mediante uno schema E/R (!?)
  - Ad esempio:
    - Ogni entità ha almeno un identificatore (interno o esterno)
    - Ogni associazione, in base al suo grado, è collegata a n entità
    - Ogni attributo ha un nome (univoco all'interno dell'entità o associazione cui si riferisce)
  - …e la stessa cosa si può fare per i concetti del modello relazionale!
- Utile se si deve creare un DB per memorizzare schemi E/R o relazionali

Utilità del modello E/R

- Uno schema E/R è più espressivo di uno schema relazionale, inoltre può essere utilizzato con successo per alcuni compiti diversi dalla progettazione, ad esempio:
- Documentazione:
  - La simbologia grafica del modello E/R può essere facilmente compresa anche dai non “addetti ai lavori”
- Reverse engineering:
  - A partire da un DB esistente si può fornirne una descrizione in E/R allo scopo di meglio analizzarlo ed eventualmente reingegnerizzarlo
- Integrazione di sistemi:
  - Essendo indipendente dal modello logico dei dati, è possibile usare il modello E/R come “linguaggio comune” in cui rappresentare DB eterogenei, allo scopo di integrarli

Limiti del modello E/R

- Per contro, per quanto più espressivo di uno schema relazionale, uno schema E/R non è sufficiente a rappresentare tutti gli aspetti di interesse
- I limiti sono di due tipi:
  - i nomi dei vari concetti possono non essere sufficienti per comprenderne il significato
  - non tutti i vincoli di integrità sono esprimibili in uno schema E/R
  - Ad esempio:
    - per sostenere un esame è necessario avere sostenuto tutti gli esami propedeutici
    - un laureando deve aver sostenuto almeno tutti gli esami dei primi 2 anni

- In fase di progettazione bisogna quindi “corredare” lo schema con una documentazione appropriata e successivamente prendere delle misure per far rispettare tali vincoli (business rules)

Progettazione concettuale

Regole guida

- Non ragionare in termini “relazionali”
  - No performance, no tables, no queries
- Occuparsi solamente di essere il più possibile precisi nel rappresentare i requisiti
- Criteri di qualità:
  - Correttezza
  - Completezza
  - Leggibilità
  - Minimalità/ridondanza

Strategie di progettazione

- Per affrontare progetti complessi è opportuno adottare uno specifico modo di procedere, ovvero una strategia di progettazione
- Casi notevoli (limite):
  - Top-down
    - Schema iniziale astratto ma completo, man mano raffinato
  - Bottom-up
    - Integrazione progressiva di schemi parziali
  - Inside-out
    - Sviluppa “a macchia d’olio”, partendo dai concetti più importanti

Strategie: pro e contro

Top-down

+ non è inizialmente necessario specificare i dettagli
- richiede sin dall'inizio una visione globale del problema, non sempre ottenibile in casi complessi

Bottom-up

+ permette una ripartizione delle attività
- richiede una fase di integrazione

Inside-out

+ non richiede passi di integrazione
- richiede ad ogni passo di esaminare tutte le specifiche per trovare i concetti non ancora rappresentati.

Un approccio “misto”

- Nella pratica si fa spesso uso di una strategia ibrida, nella quale:

1. si individuano i concetti principali e si realizza uno schema scheletro, che contiene solamente i concetti più importanti (ossatura su cui basare decomposizione e integrazione)
2. sulla base di questo si può decomporre
3. poi si raffina, si espande, si integra

Glossario dei termini, omonimi e sinonimi

- Raramente i requisisti espressi in linguaggio naturale sono privi di ambiguità. È infatti frequente il caso di

  Omonimi: lo stesso termine viene usato per descrivere concetti differenti

  Sinonimi: termini diversi vengono usati per descrivere lo stesso concetto
- Un modo conveniente per rappresentare sinteticamente i concetti più rilevanti emersi dall'analisi è il glossario dei termini, il cui scopo è fornire per ogni concetto rilevante:
  - Una breve descrizione del concetto
  - Eventuali sinonimi
  - Relazioni con altri concetti del glossario stesso
- Il glossario può essere direttamente utilizzato per costruire lo schema scheletro

Dai concetti allo schema E/R

- Va sempre ricordato che un concetto non è di per sé un'entità, un'associazione, un attributo, o altro

DIPENDE DAL CONTESTO!

- Come regole guida, un concetto verrà rappresentato come
  - Entità
    - se ha proprietà significative e descrive oggetti con esistenza autonoma
  - Attributo
    - se è semplice e non ha proprietà
  - Associazione
    - se correla due o più concetti
  - Generalizzazione/specializzazione
    - se è caso più generale/particolare di un altro

Esempio: società di formazione

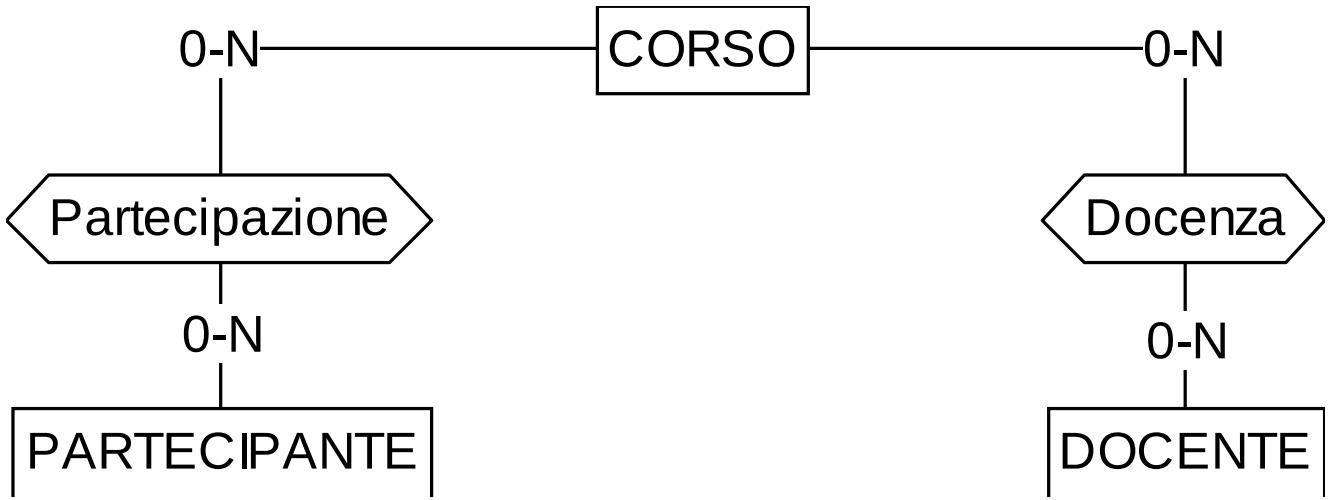
- Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei partecipanti ai corsi e dei **docenti**.
- Per gli studenti (circa 5000), identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, il **luogo** di nascita, il nome dei loro attuali datori di lavoro, i *posti* dove hanno lavorato in precedenza insieme al periodo, l'indirizzo e il numero di telefono, i corsi che hanno frequentato (i corsi sono in tutto circa 200) e il giudizio finale.
- Rappresentiamo anche i seminari che stanno attualmente frequentando e, per ogni giorno, i **luoghi** e le ore dove sono tenute le lezioni.
- I corsi hanno un codice, un titolo e possono avere varie edizioni con date di inizio e fine e numero di partecipanti.
- Se gli studenti sono liberi professionisti, vogliamo conoscere l'area di interesse e, se lo possiedono, il titolo. Per quelli che *lavorano alle dipendenze di altri*, vogliamo conoscere invece il loro livello e la posizione ricoperta.
- Per gli **insegnanti** (circa 300), rappresentiamo il cognome, l'età, il *posto* dove sono nati, il nome del corso che insegnano, quelli che hanno insegnato nel passato e quelli che possono insegnare. Rappresentiamo anche tutti i loro recapiti telefonici. I docenti possono essere dipendenti interni della società o collaboratori esterni.

Requisiti ristrutturati

- Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei partecipanti ai corsi e dei docenti.
- Per i partecipanti, identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, il comune di nascita, il nome dei loro attuali datori di lavoro e di quelli precedenti (insieme alle date di inizio e fine rapporto), le edizioni dei corsi che stanno attualmente frequentando e quelli che hanno frequentato nel passato, con la relativa votazione finale.
- Per ogni corso si rappresentano luoghi, date e orario delle lezioni.
- I corsi hanno un codice, un titolo e possono avere varie edizioni con date di inizio e fine e numero di partecipanti.
- Se gli studenti sono liberi professionisti, vogliamo conoscere l'area di interesse e, se lo possiedono, il titolo di studio. Per i dipendenti vogliamo conoscere invece il loro livello e la posizione ricoperta.
- Per i docenti rappresentiamo il cognome, data e comune di nascita, il nome del corso che insegnano, quelli che hanno insegnato nel passato e quelli che possono insegnare. Rappresentiamo anche tutti i loro recapiti telefonici. I docenti possono essere dipendenti della società o collaboratori esterni.

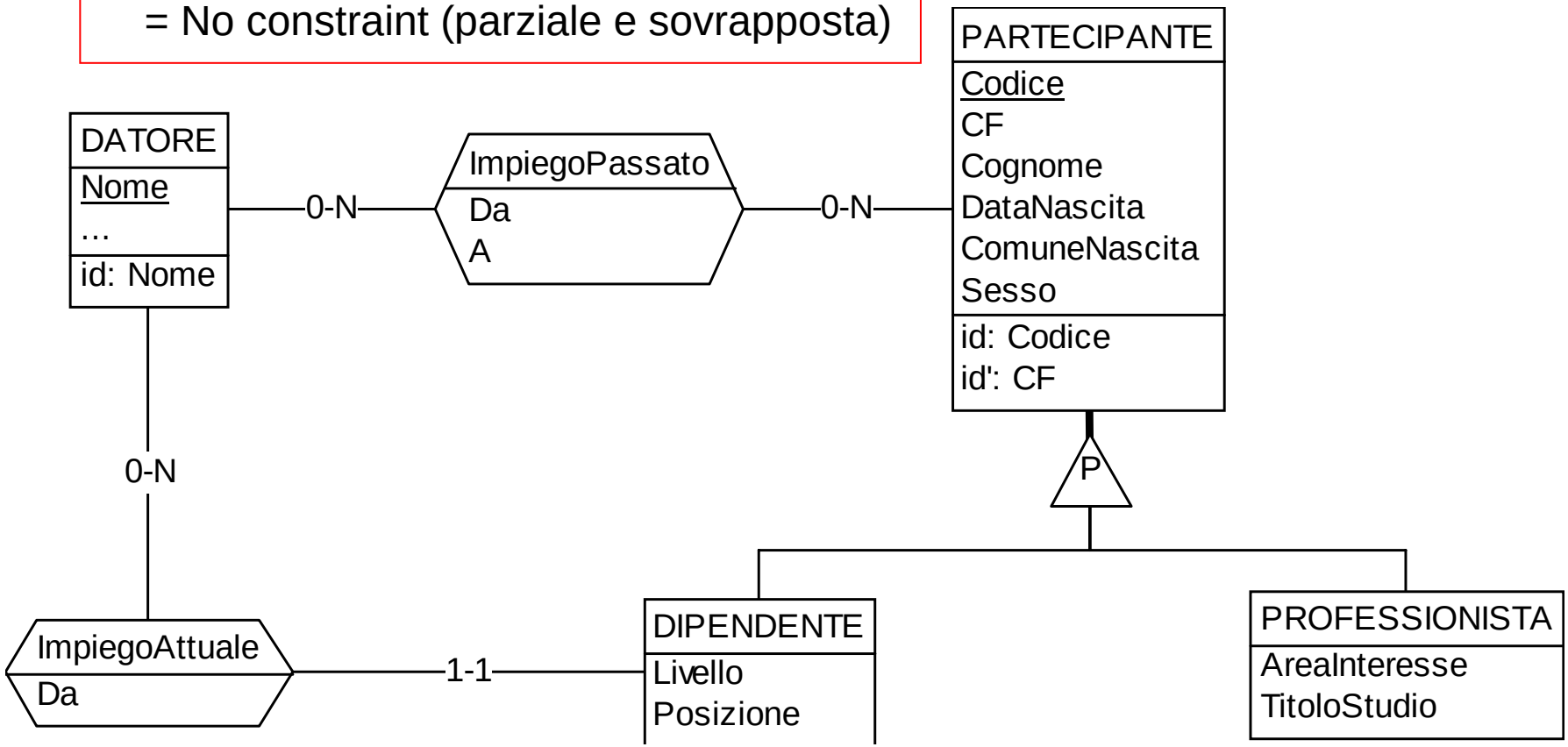
Glossario dei termini e schema scheletro

Termine	Descrizione	Sinonimi	Collegamenti
Partecipante	Persona che partecipa ai corsi. Può essere un dipendente o un professionista	Studente	Corso
Docente	Docente dei corsi. Può essere un collaboratore esterno	Insegnante	Corso
Corso	Corso organizzato dalla società. Può avere più edizioni	Seminario	Docente, Partecipante

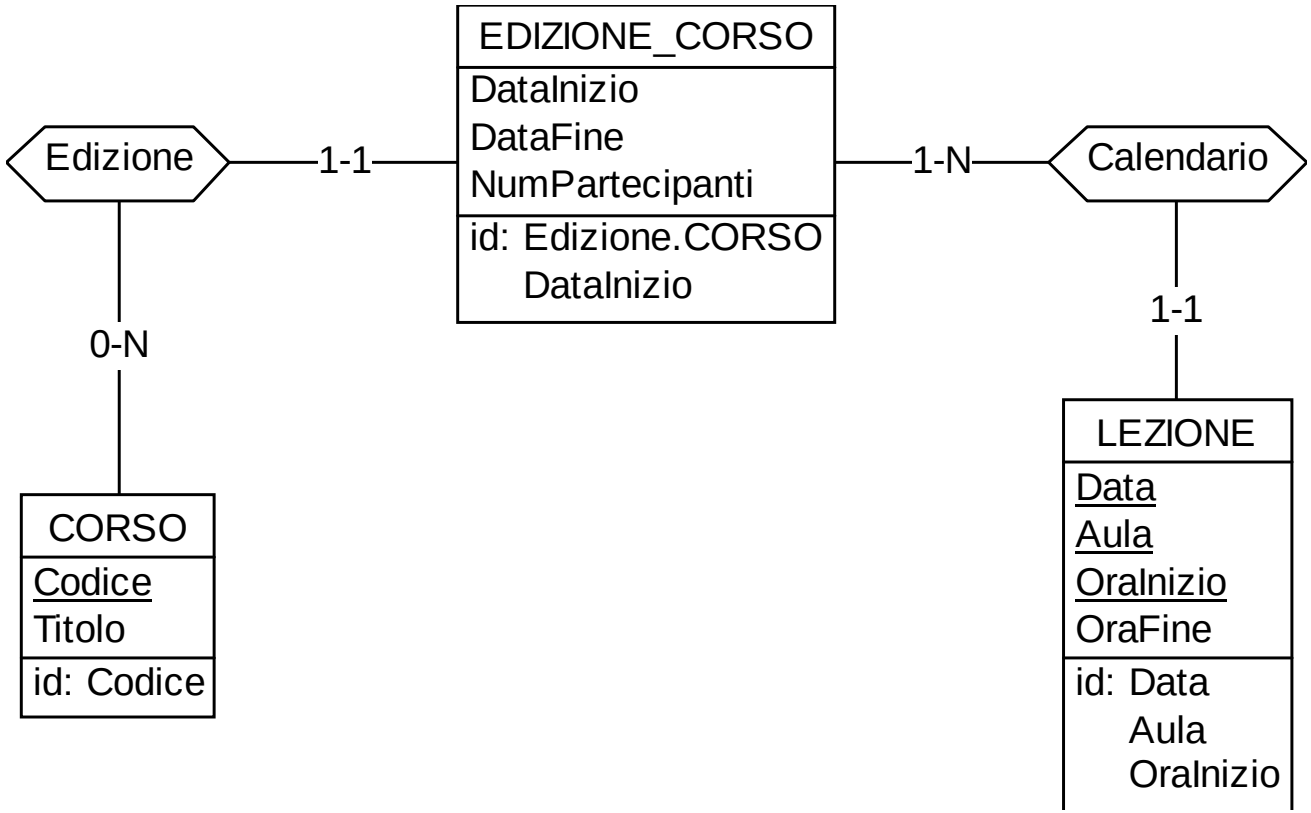


Raffinamento di Partecipante

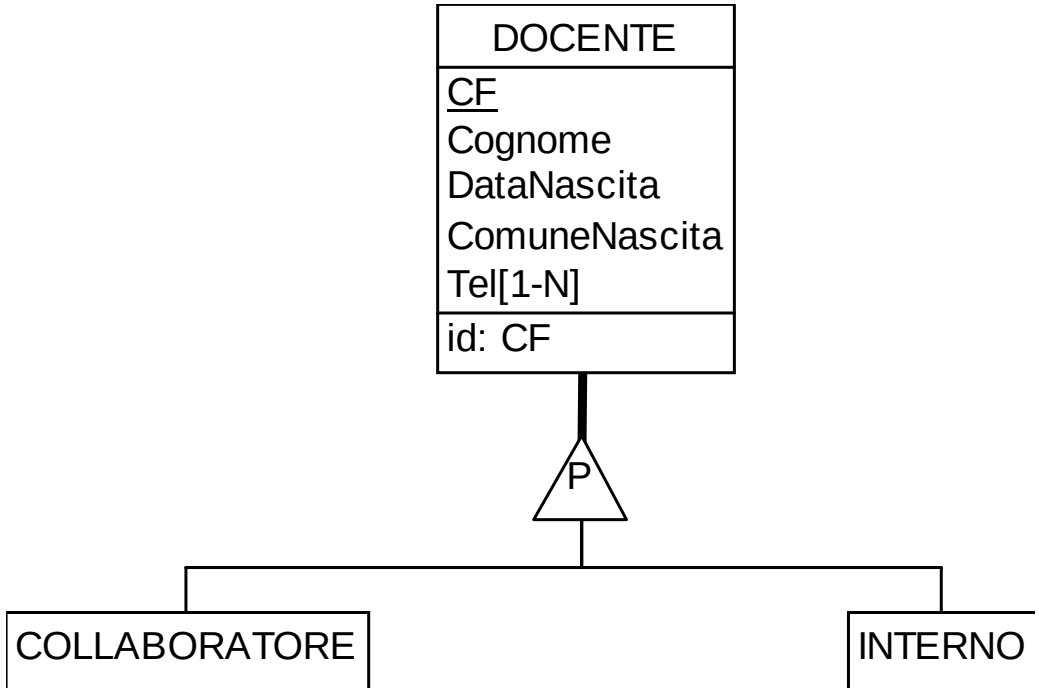
T = Total (totale e sovrapposta)
P = Partition (totale ed esclusiva)
D = Disjoint (parziale ed esclusiva)
= No constraint (parziale e sovrapposta)



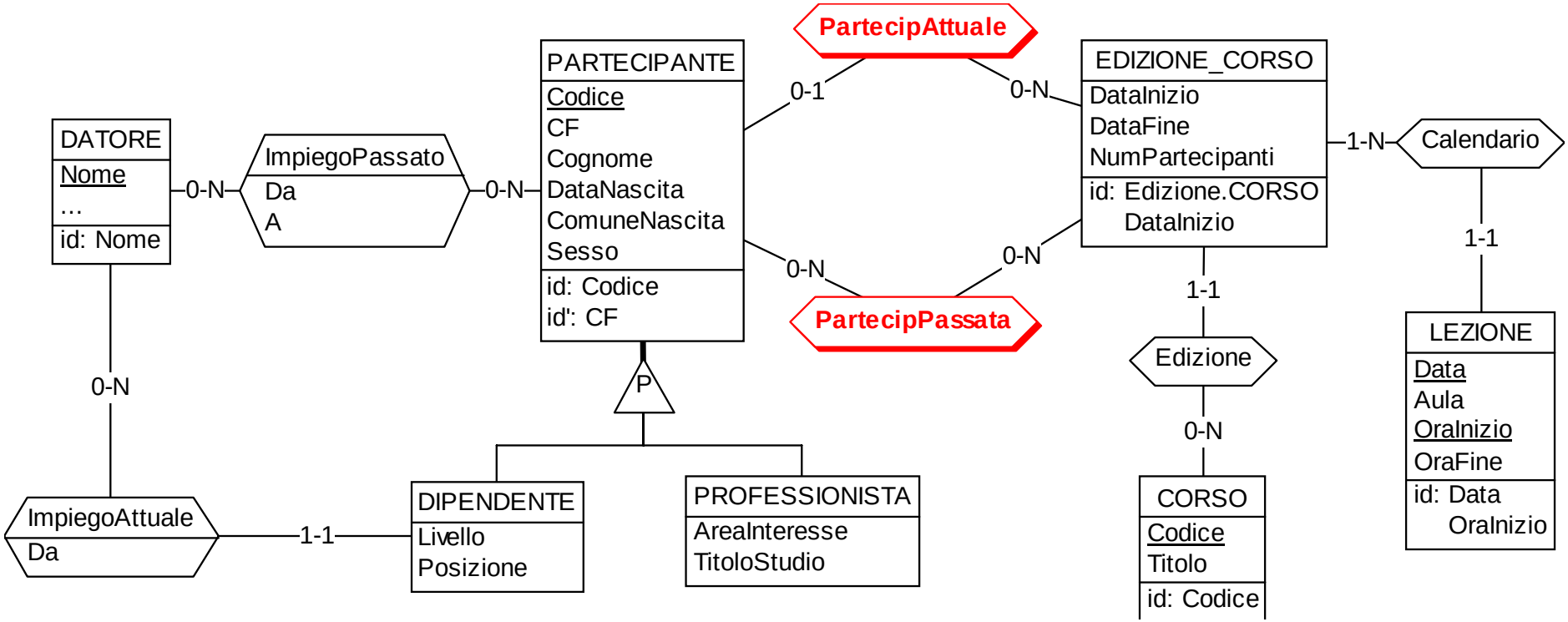
Raffinamento di Corso



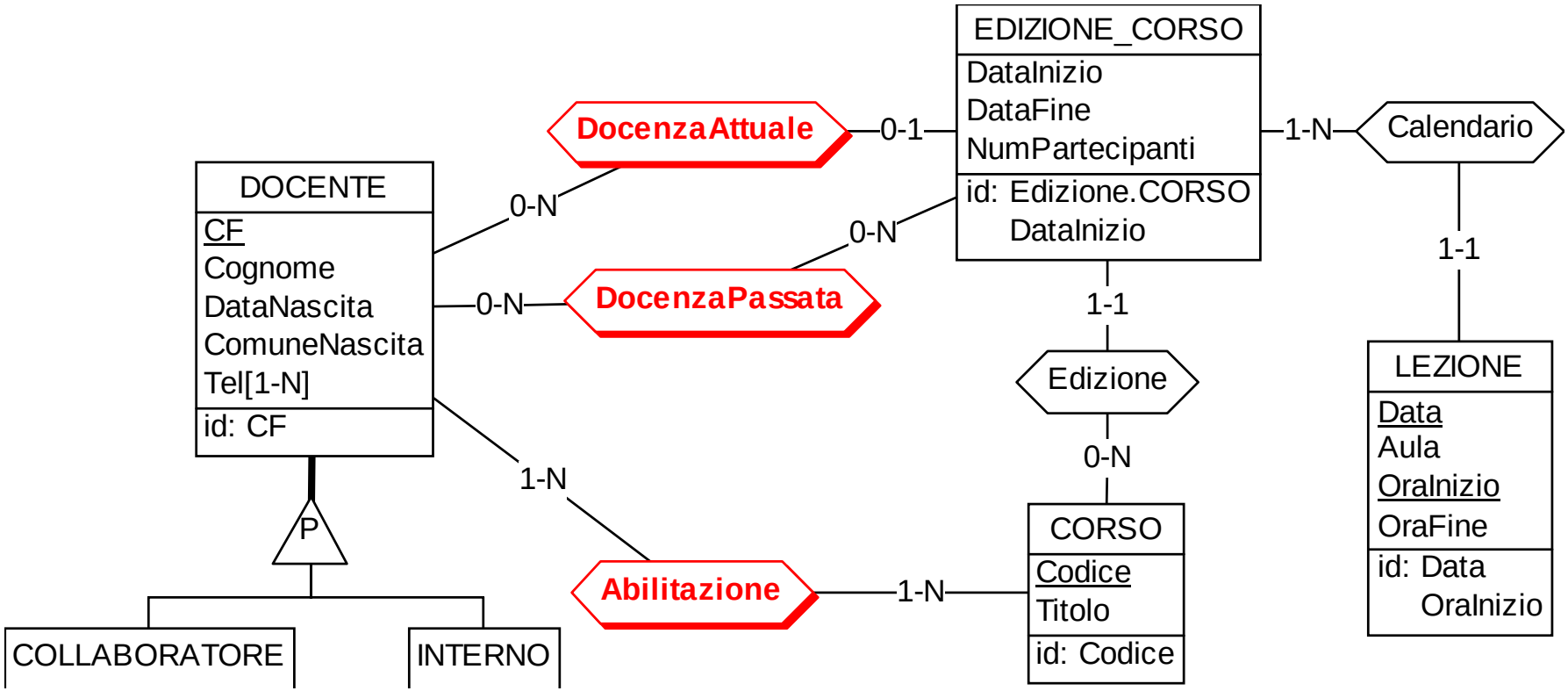
Raffinamento di Docente



Integrazione: Partecipante e Corso



Integrazione: Docente e Corso



Progettazione logica

Progettazione logica

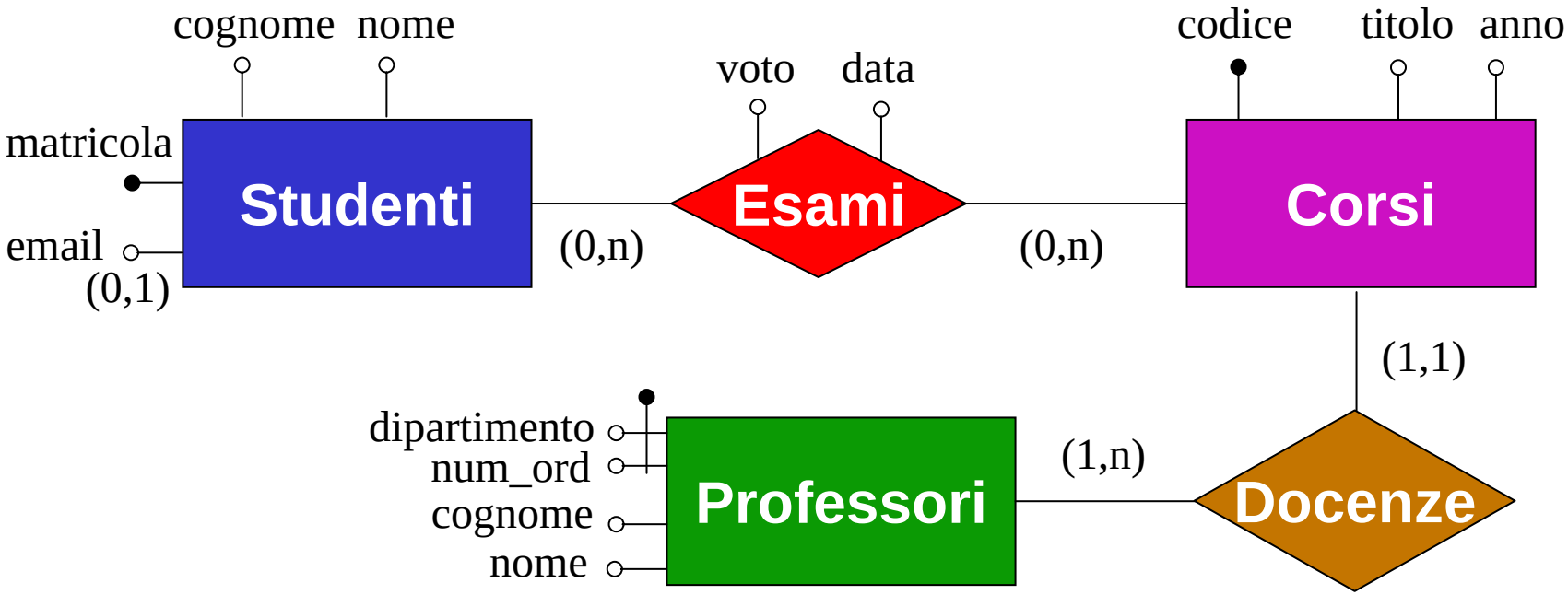
- Obiettivo: pervenire, a partire dallo schema concettuale, a uno schema logico che lo rappresenti in modo fedele e che sia, al tempo stesso, "efficiente"
- L'efficienza è legata alle prestazioni, ma poiché queste non sono valutabili precisamente a livello concettuale e logico si ricorre a degli indicatori semplificati, basati su considerazioni (spesso solo qualitative) di spazio (numero di istanze previste) e tempo (numero istanze visitate)
- In generale, per confrontare tra loro diverse alternative di traduzione bisogna conoscere, almeno in maniera approssimativa, il "carico di lavoro", ovvero:
  - Le principali operazioni che la BD dovrà supportare
  - I volumi dei dati in gioco

Fasi della progettazione logica

- La progettazione logica può articolarsi in due fasi principali:
  - Ristrutturazione: eliminazione dallo schema E/R di tutti i costrutti che non possono essere direttamente rappresentati nel modello logico target (relazionale nel nostro caso):
    - Eliminazione degli attributi multivalore
    - Eliminazione delle generalizzazioni
    - Partizionamento/accorpamento di entità e associazioni
    - Scelta degli identificatori principali
  - Traduzione: si mappano i costrutti residui in elementi del modello relazionale

Traduzione di schemi E/R “semplici”

- Caso più semplice:
  - Abbiamo entità e associazioni, ma non gerarchie
  - Ogni entità ha un singolo identificatore, ed è interno
  - Non abbiamo attributi ripetuti



Traduzione di base: entità

- Ogni entità è tradotta con una relazione con gli stessi attributi
  - La chiave primaria coincide con l'identificatore dell'entità
  - Se un attributo è opzionale permettiamo la presenza di valori nulli (l'asterisco (*) indica tale possibilità)

Studenti

Matricola	Cognome	Nome	Email*
29323	Bianchi	Giorgio	gbianchi@alma.unibo.it
35467	Rossi	Anna	
39654	Verdi	Marco	
42132	Neri	Lucia	lucia78@cs.ucsd.edu

Corsi

Codice	Titolo	Anno
483	Analisi	1
729	Analisi	1
815	Elettronica	2
913	Sistemi Informativi	3

Professori

Dip	Num_ord	Cognome	Nome
DEIS	12	Bolzano	Bernhard
DEIS	15	Codd	Ted
DIE	12	Marconi	Guglielmo

Traduzione di base: associazioni

- Ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega
  - gli identificatori delle entità collegate costituiscono una superchiave
  - la chiave primaria dipende dalle cardinalità massime delle entità nell'associazione

Esami

Matricola	Codice	Voto	Data
29323	483	28	12/06/2003
39654	729	30	15/07/2003
29323	913	26	12/06/2003
35467	913	30	20/09/2004

Docenze

Codice	Dip	Num_ord
483	DEIS	12
729	DEIS	12
815	DIE	12
913	DEIS	15

Associazioni molti a molti

- La chiave primaria coincide con l'unione degli identificatori delle entità collegate

  Esami(Matricola, CodCorso, Voto, Data)

  FK: Matricola REFERENCES Studenti

  FK: CodCorso REFERENCES Corsi

- Per le associazioni molti a molti la traduzione presentata, ovvero con una relazione a sé, è la sola alternativa possibile

Associazioni uno a molti

- La chiave primaria coincide con l'identificatore dell'entità che partecipa con cardinalità massima 1

Docenze(CodCorso, Dip, Num_ordine)

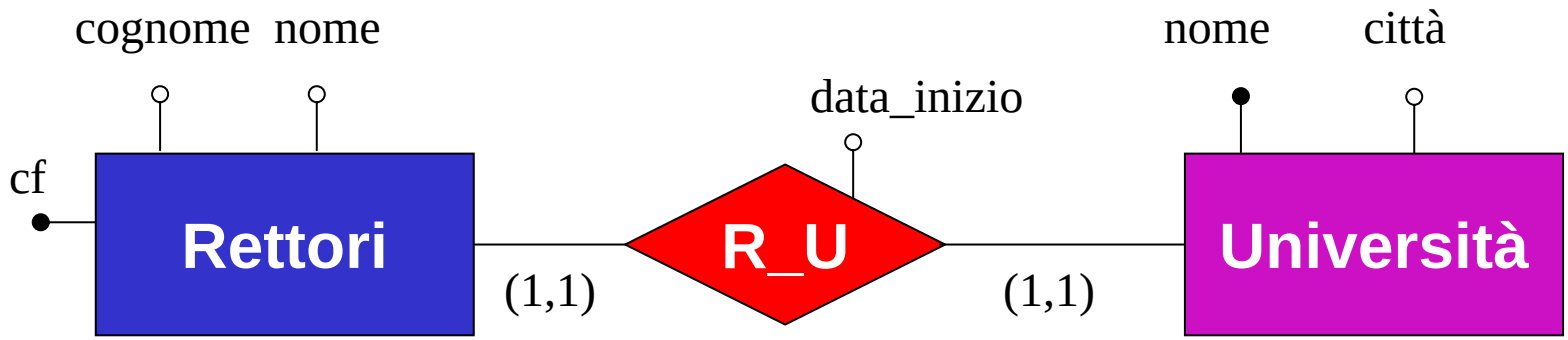
FK: CodCorso REFERENCES Corsi

FK: (Dip, Num_ordine) REFERENCES Professori(Dip, Num_ord)

- Poiché ogni corso c ha al massimo 1 docente, allora c può comparire al massimo una volta in Docenze

Associazioni uno a uno

- La chiave primaria è uno dei due identificatori, l'altro diventa una chiave alternativa



R_U(CF, NomeUniversità)

FK: CF REFERENCES Rettori

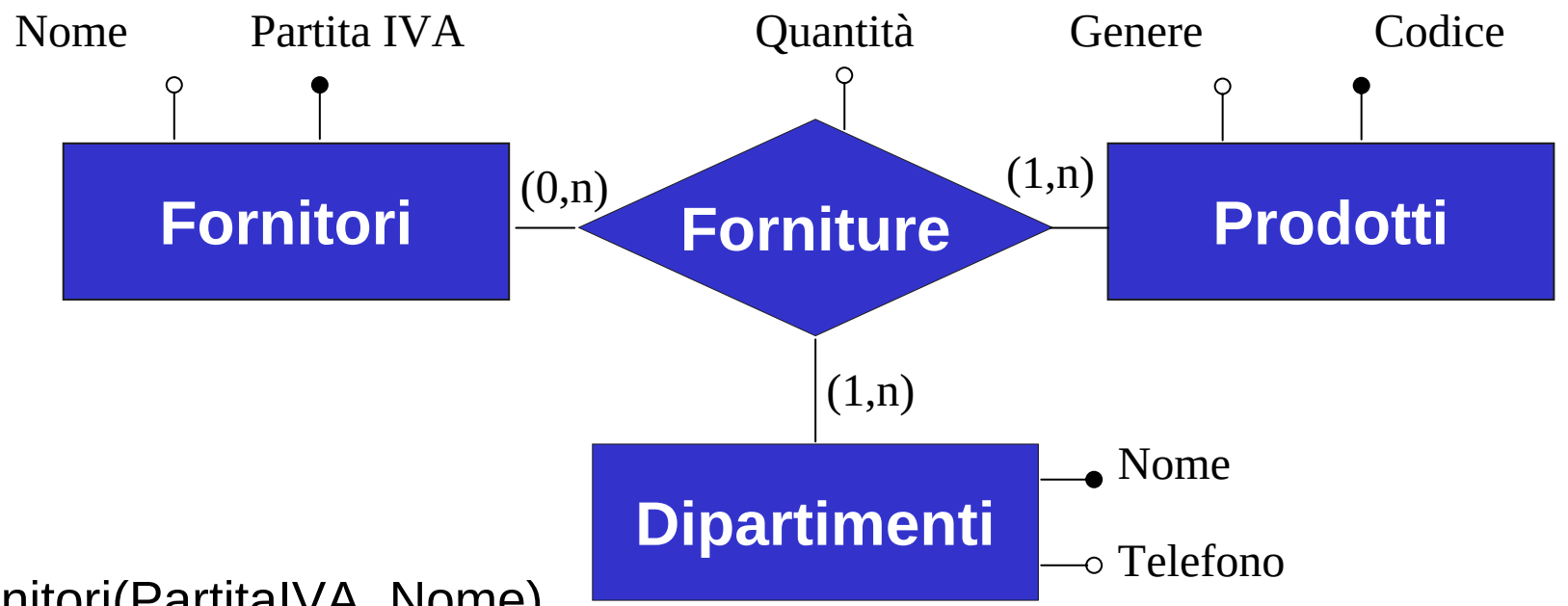
FK: NomeUniversità REFERENCES Università

Unique(NomeUniversità)

La scelta dipende dall'importanza relativa delle chiavi

Associazioni n-arie

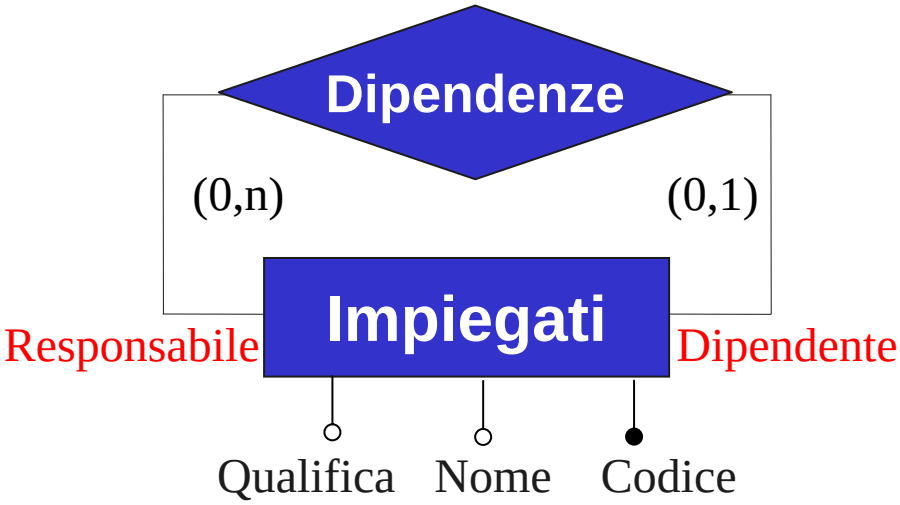
- Valgono le considerazioni già fatte, ad esempio:

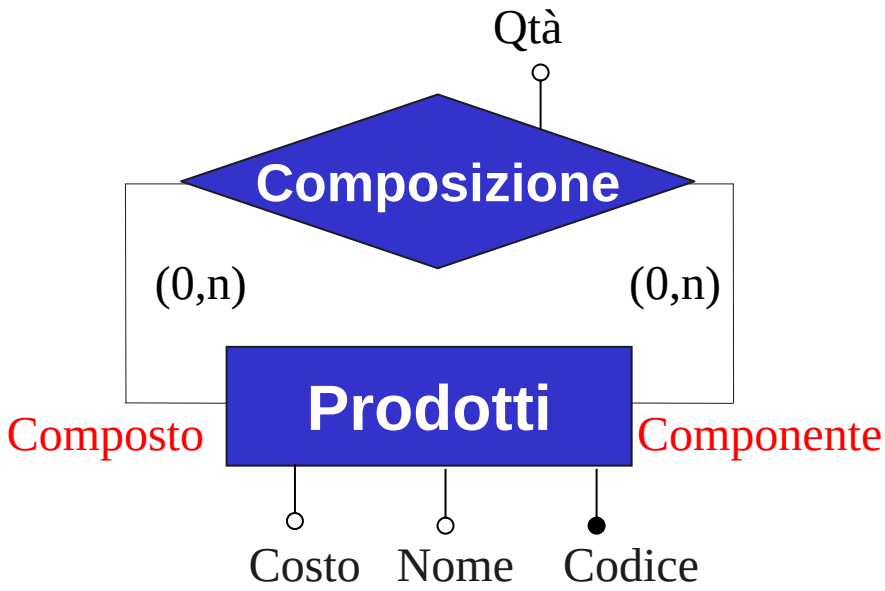


Fornitori(PartitaIVA, Nome)
Prodotti(Codice, Genere)
Dipartimenti(Nome, Telefono)
Forniture(Fornitore, Prodotto, Dipartimento, Quantità)
FK: …

Associazioni ad anello

- In questo caso i nomi delle foreign key si possono derivare dai ruoli presenti nei rami dell'associazione





Impiegati(Codice, Nome, Qualifica) Dipendenze(Dipendente, Responsabile) FK: Dipendente REFERENCES Impiegati FK: Responsabile REFERENCES Impiegati	Prodotti(Codice, Nome, Costo) Composizione(Composto, Componente, Qtà) FK: Composto REFERENCES Prodotti FK: Componente REFERENCES Prodotti

Associazioni uno a molti: alternative (1)

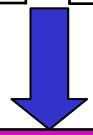
- Per le associazioni uno a molti è possibile considerare anche una traduzione più compatta, che ingloba l'associazione nella relazione che traduce l'entità partecipante con cardinalità massima 1

Corsi

Codice	Titolo	Anno
483	Analisi	1
729	Analisi	1
815	Elettronica	2
913	Sistemi Informativi	3

Docenze

Codice	Dip	Num_ordine
483	DEIS	12
729	DEIS	12
815	DIE	12
913	DEIS	15



Corsi

Codice	Titolo	Anno	Dip_docente	Num_ordine_docente
483	Analisi	1	DEIS	12
729	Analisi	1	DEIS	12
815	Elettronica	2	DIE	12
913	Sistemi Informativi	3	DEIS	15

Associazioni uno a molti: alternative (2)

- Il vantaggio che se ne trae si apprezza in fase di interrogazione:
  - Si semplifica la scrittura di query SQL e, soprattutto,
  - Si riduce il numero di join da eseguire (migliori prestazioni!)

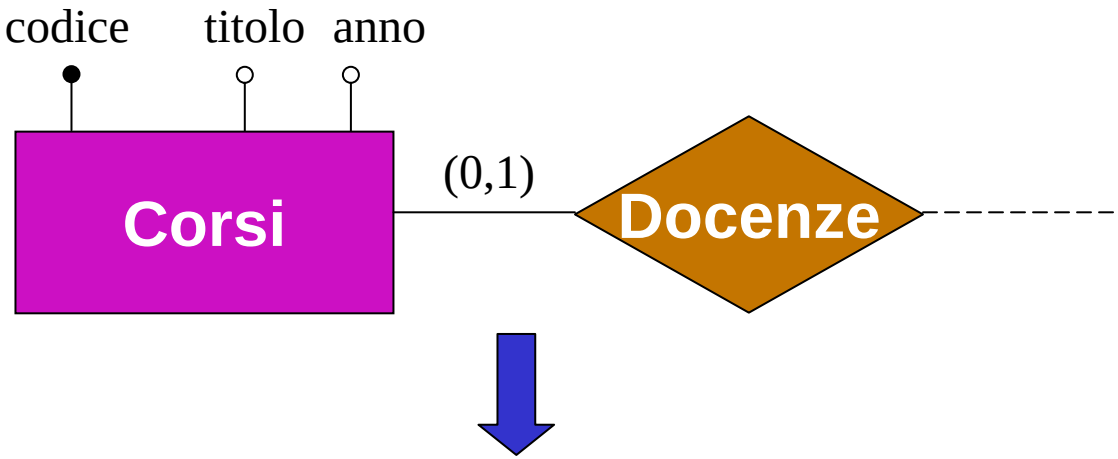
"Nome e cognome degli insegnanti di Analisi"

```
SELECT  P.Nome, P.Cognome
FROM    Corsi C, Docenze D, Professori P
WHERE   C.Codice = D.CodCorso
AND     D.Dip = P.Dip AND D.Num_ordine = P.Num_ord
AND     C.Titolo = 'Analisi'
```

```
SELECT  P.Nome, P.Cognome
FROM    Corsi C, Professori P
WHERE   C.Dip_docente = P.Dip
AND     C.Num_ordine_docente = P.Num_ord
AND     C.Titolo = 'Analisi'
```

Associazioni uno a molti: alternative (3)

- Per contro, bisogna stare attenti se l'entità che ingloba l'associazione partecipa con cardinalità minima 0
- Supponiamo che un corso possa anche non avere un docente…



Corsi

Codice	Titolo	Anno	Dip_docente*	Num_ordine_docente*
483	Analisi	1	DEIS	12
729	Analisi	1	DEIS	12
815	Elettronica	2		
913	Sistemi Informativi	3	DEIS	15

Associazioni uno a molti: alternative (4)

- La presenza dei valori nulli può essere tollerata se questi sono relativamente pochi (dipende dai volumi dei dati in gioco), altrimenti si ha uno spreco inutile di spazio
- Bisogna ovviamente garantire che i valori della foreign key siano o tutti definiti o tutti nulli

Corsi(Codice, Titolo, Anno, Dip_docente*, Num_ordine_docente*)

FK: (Dip_docente, Num_ordine_docente)
REFERENCES Professori(Dip, Num_ord)

CHECK (((Dip_docente IS NOT NULL) AND (Num_ordine_docente IS NOT NULL))
OR ((Dip_docente IS NULL) AND (Num_ordine_docente IS NULL)))

Nota: clausola CHECK

- Nello standard SQL la clausola CHECK ammette qualsiasi espressione booleana come argomento, ad es:

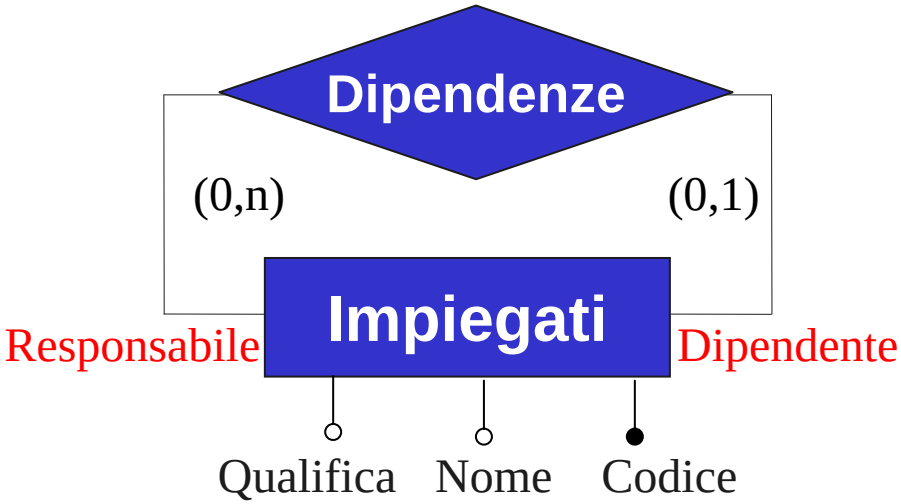
```
CHECK ((SELECT D.Stipendio FROM DIPENDENTI D
        WHERE D.CodDip = CodMgr) > Stipendio)
```

  verifica che lo stipendio di un dipendente sia sempre minore di quello del suo manager

- In pratica, nessun DBMS consente clausole CHECK arbitrariamente complesse
- Caso più comune: l'espressione si riferisce ad una singola tupla della relazione su cui la clausola è definita
  - E' il caso di PostgreSQL
  - MySQL accetta, per ragioni di compatibilità, clausole CHECK ma le ignora!

Associazioni uno a molti ad anello

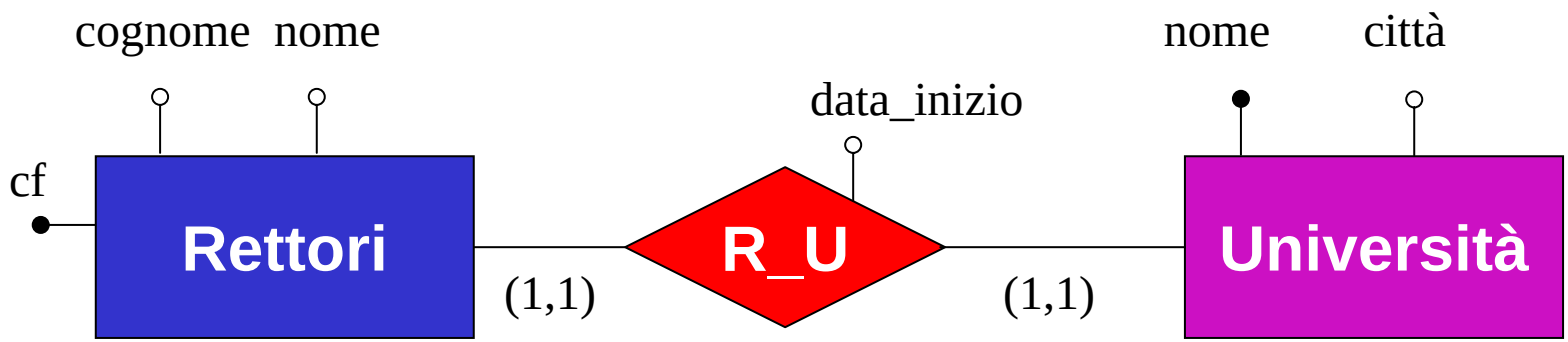
- Quanto visto si applica anche alle associazioni ad anello, ad esempio:



Impiegati(<u>Codice</u>, Nome, Qualifica, CodiceResponsabile*)
FK: CodiceResponsabile REFERENCES Impiegati

Associazioni uno a uno: alternative (1)

- Si hanno a disposizione varie possibilità:



Rettori(CF, Cognome, Stipendio, NomeUniversità, DataInizio)

FK: NomeUniversità REFERENCES Università

Unique(NomeUniversità)

oppure

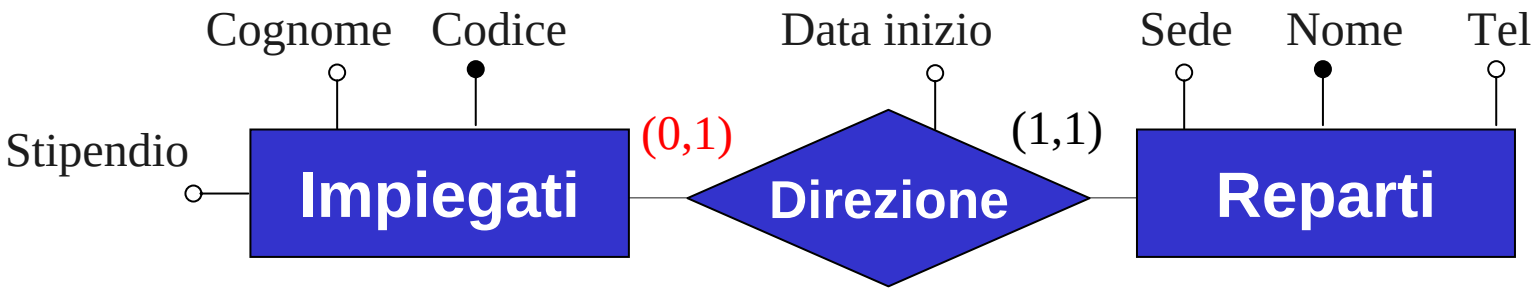
Università(Nome, Città, CFRettore, DataInizio)

FK: CFRettore REFERENCES Rettori

Unique(CFRettori)

Associazioni uno a uno: alternative (2)

- Se min-card(E,R) = 0, tradurre l'associazione R inglobandola in E non è in generale una buona scelta (ancora, dipende dai volumi dei dati in gioco)



Impiegati(Codice, Cognome, Stipendio, Reparto*, DataInizio*)

FK: Reparto REFERENCES Reparti

Unique(Reparto)

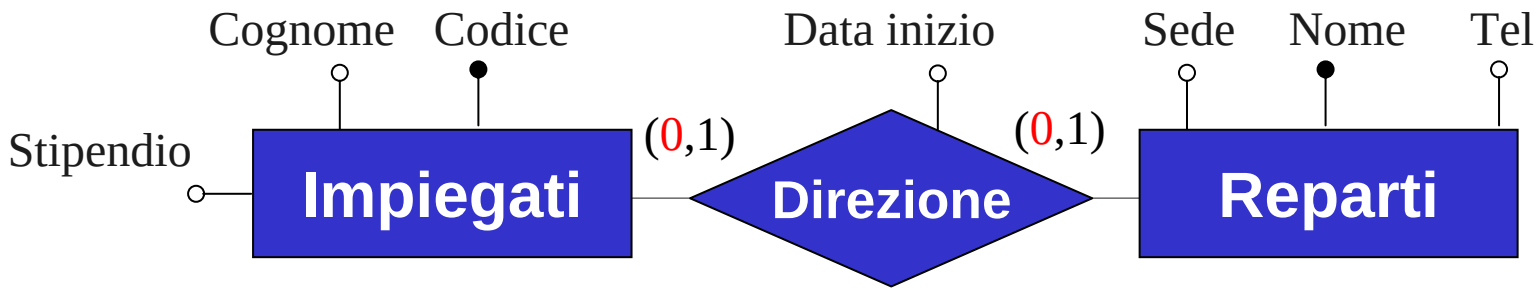
CHECK (((Reparto IS NOT NULL) AND (DataInizio IS NOT NULL)) OR
((Reparto IS NULL) AND (DataInizio IS NULL)))

Reparto(Nome, Sede, Telefono)

TROPPI VALORI NULLI!!

Associazioni uno a uno: alternative (3)

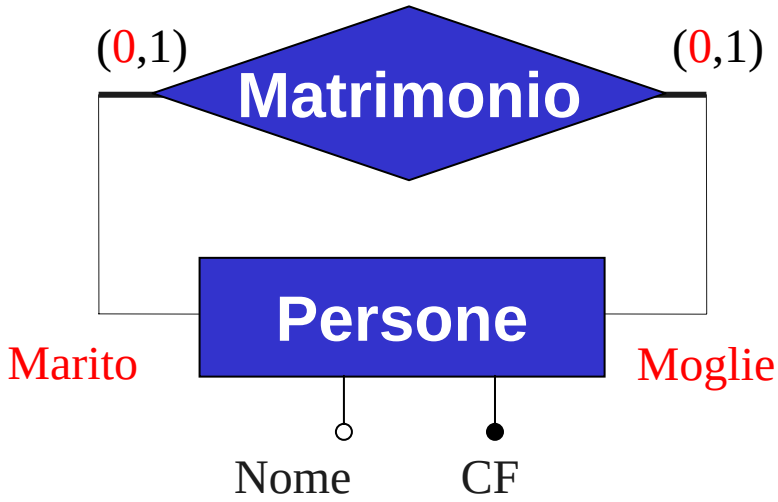
- E' possibile anche operare una traduzione in cui le entità vengono "accorpate" (questo andrebbe meglio considerato in fase di ristrutturazione). In ogni caso:
  - Se min-card(E1,R) = min-card(E2,R) = 1 si avranno due chiavi, entrambe senza valori nulli (la chiave primaria è "la più importante")
  - Se min-card(E1,R) = 0 e min-card(E2,R) =1 la chiave derivante da E2 ammetterà valori nulli, e la chiave primaria si ottiene da E1
  - Se min-card(E1,R) = min-card(E2,R) = 0 entrambe le chiavi hanno valori nulli, quindi si rende necessario introdurre un codice



ImpRep(<u>CodiceImpDip</u>, CodiceImp*, …, Reparto*, …, DataInizio*)

Associazioni uno a uno ad anello

- In questo caso è possibile ancora inglobare l'associazione nell'entità
- La traduzione è ancora problematica se entrambe le partecipazioni sono opzionali



1 relazione:

Persone(Codice, CFUomo*, NomeUomo*, CFDonna*, NomeDonna*)

2 relazioni:

Persone(CF, Nome)

Matrimoni(Marito, Moglie)

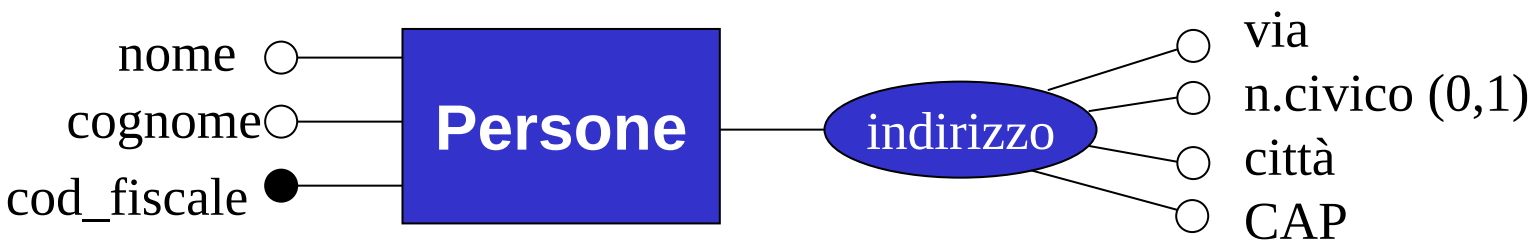
FK: Marito REFERENCES Persone

FK: Moglie REFERENCES Persone

Unique (Moglie)

Attributi composti

- Gli attributi composti vengono tradotti suddividendoli ricorsivamente nelle loro componenti
- In alternativa si possono mappare in un singolo attributo della relazione, il cui dominio va opportunamente definito, ma questa scelta ovviamente porta a un impoverimento della rappresentazione
- E' consigliabile usare un prefisso comune per gli attributi che si ottengono

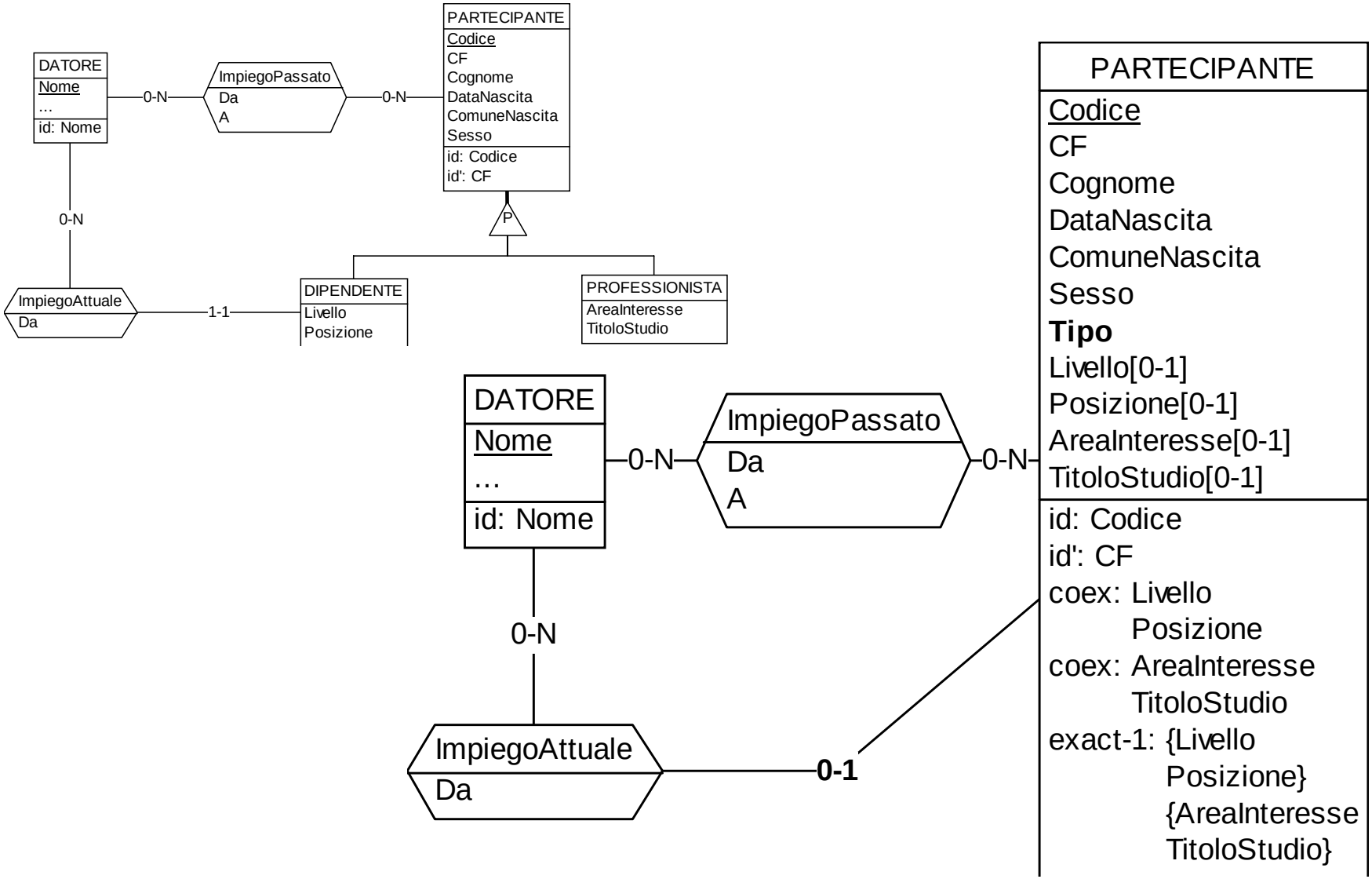


Persone(CF, Cognome, Nome, Ind_Via, Ind_NCivico*, Ind_Città, Ind_CAP)

Ristrutturazione: eliminazione delle gerarchie

- Il modello relazionale non può rappresentare direttamente le generalizzazioni
- Si eliminano perciò le gerarchie, sostituendole con entità e associazioni
- Vi sono 3 possibilità di base (più altre soluzioni intermedie):
  - Accorpare le entità figlie nel genitore (collasso verso l'alto)
  - Accorpare il genitore nelle entità figlie (collasso verso il basso)
  - Sostituire la generalizzazione con associazioni

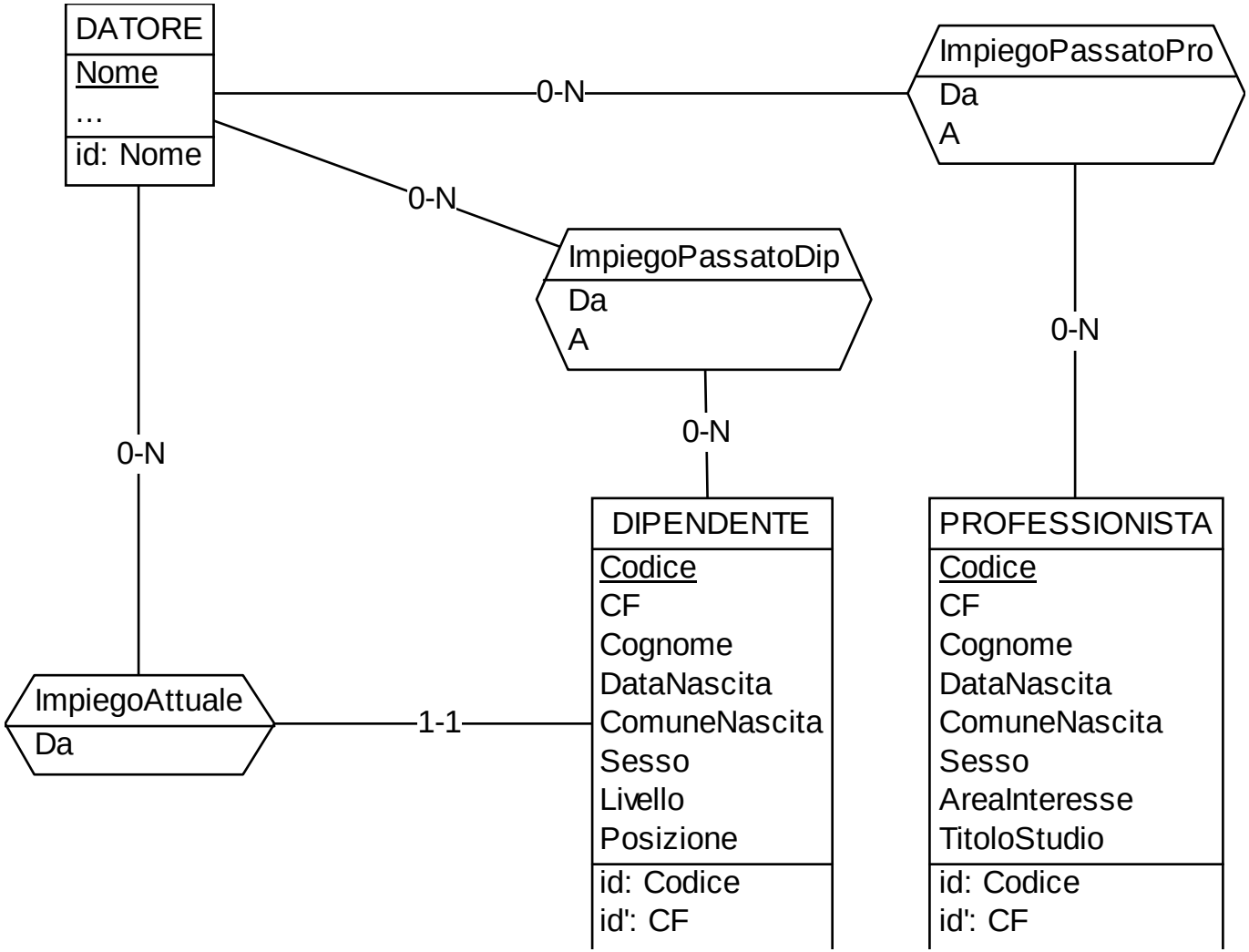
1. Collasso verso l'alto



Collasso verso l'alto: osservazioni

- "Tipo" è un attributo selettore che specifica se una singola istanza dell'entità più generica E appartiene a una delle N sottoentità
- Copertura
  - totale esclusiva: Tipo ha N valori, tanti quanti le sottoentità
  - parziale esclusiva: Tipo ha N+1 valori
  - sovrapposta: tanti selettori quante sono le sottoentità, ciascuno a valore booleano
- Le eventuali associazioni connesse alle sottoentità si trasportano su E, le cardinalità minime diventano 0

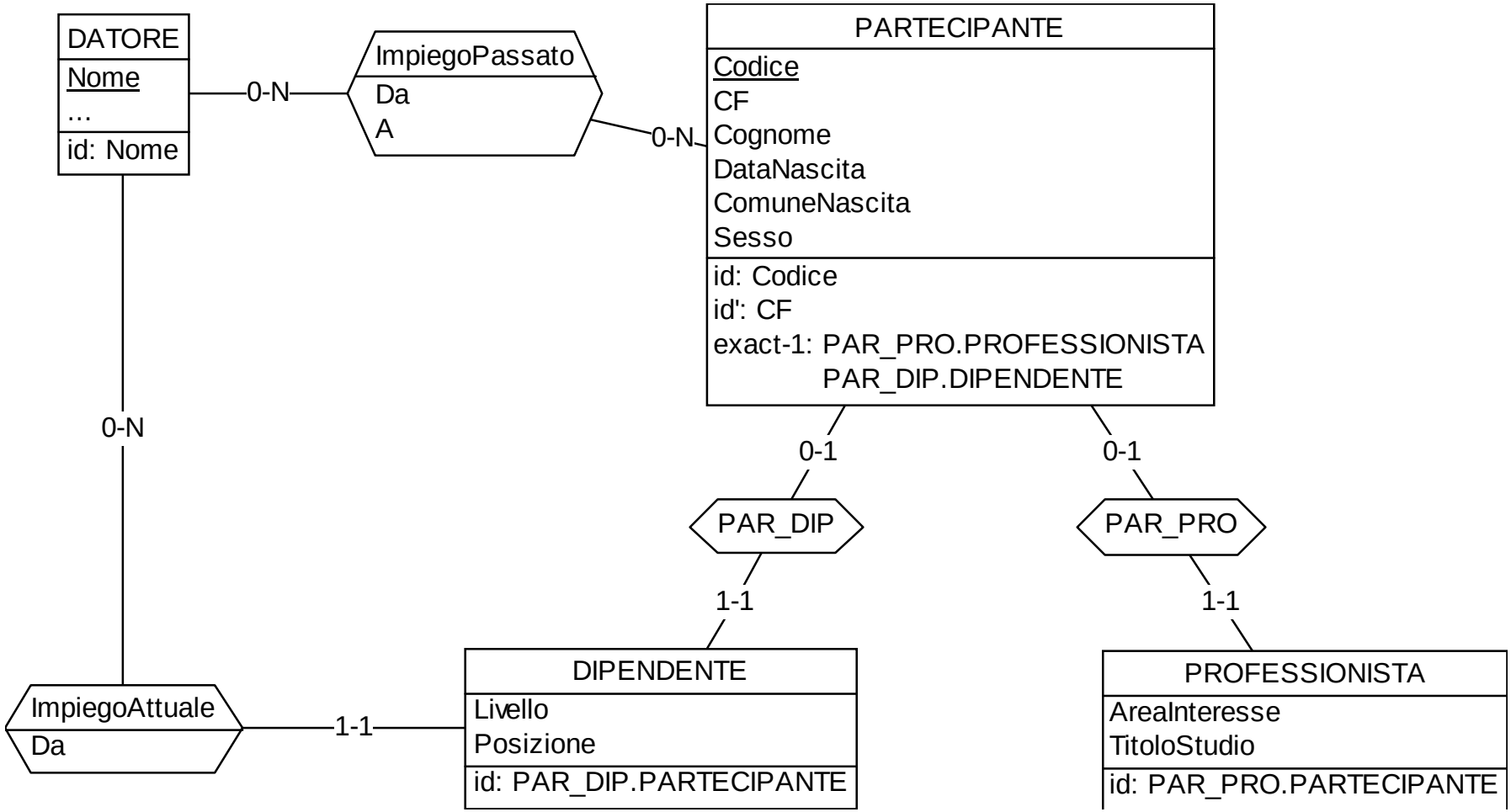
2. Collasso verso il basso



Collasso verso il basso: osservazioni

- Se la copertura non è completa non si può fare
  - non si saprebbe dove mappare le istanze di E che non sono anche istanze di una sottoentità
- Se la copertura non è esclusiva introduce ridondanza
  - una certa istanza può essere parte di due sottoentità E1 ed E2, e quindi si rappresentano due volte gli attributi comuni che provengono da E

3. Sostituire con associazioni



Cosa conviene fare?

- È possibile seguire alcune semplici regole generali, che si riconducono al principio:

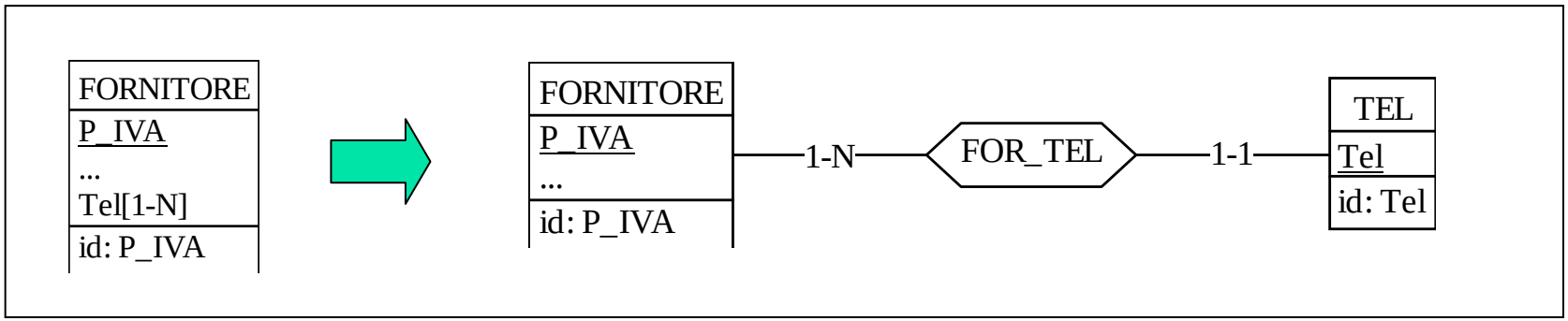
  mantieni insieme ciò che viene usato insieme

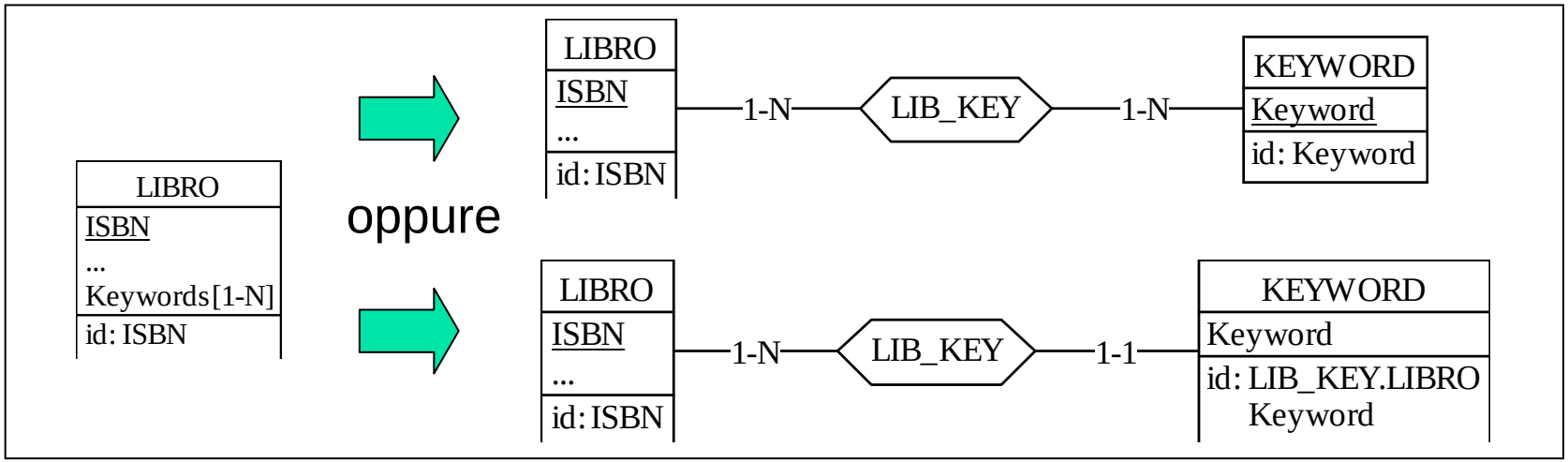
  1. conviene se gli accessi al genitore e alle figlie sono contestuali
  2. conviene se gli accessi alle figlie sono distinti
  3. conviene se gli accessi alle entità figlie sono separati dagli accessi al genitore

- Sono anche possibili soluzioni “ibride”, soprattutto in gerarchie a più livelli

Eliminazione di attributi multivalore

- Si introduce una nuova entità le cui istanze sono identificate dai valori dell'attributo
- L'associazione può essere uno a molti o molti a molti





Partizionamenti e accorpamenti

- È possibile ristrutturare lo schema accorpando o partizionando entità e associazioni
- Tali ristrutturazioni vengono effettuate per rendere più efficienti le operazioni in base al principio già visto, ovvero:
  - separando attributi di un concetto che vengono acceduti separatamente
  - raggruppando attributi di concetti diversi acceduti insieme

- I casi principali sono:
  - partizionamento “verticale” di entità
  - partizionamento “orizzontale” di entità e associazioni
    - es: dati storici vs attuali
  - accorpamenti di entità e associazioni

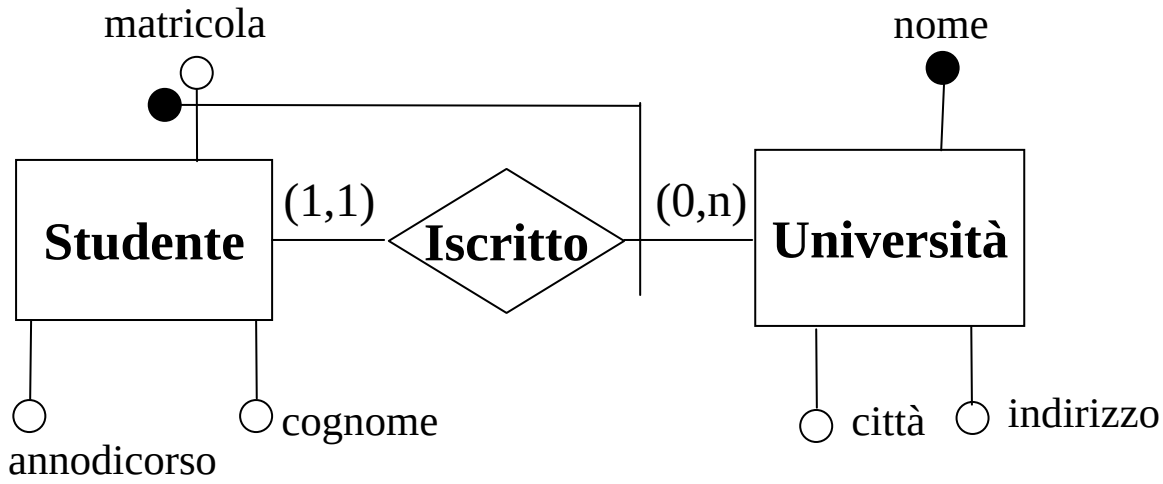
Scelta degli identificatori principali

- È un'operazione indispensabile per la traduzione nel modello relazionale, che corrisponde alla scelta della chiave primaria
- I criteri da adottare sono:
  - assenza di opzionalità (valori NULL)
  - semplicità
  - utilizzo nelle operazioni più frequenti o importanti

- Se nessuno degli identificatori soddisfa i requisiti si introducono dei nuovi attributi (dei "codici") allo scopo

Entità con identificazione esterna

- Nel caso di entità identificata esternamente, si “importa” l’identificatore della/e entità identificante/i.
- L’associazione relativa risulta automaticamente tradotta



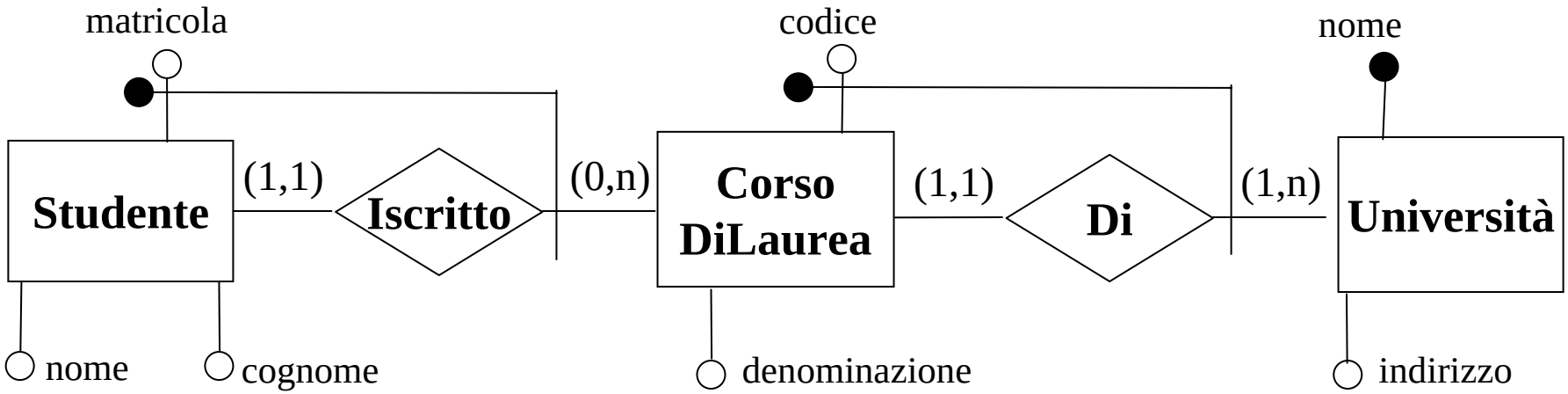
Studente(Matricola, Università, Cognome, AnnoDiCorso)

FK: Università REFERENCES Università

Università(Nome, Città, Indirizzo)

Identificazioni esterne in cascata

- Nel caso generale, si possono avere identificazioni esterne in cascata
- Per operare correttamente occorre partire dalle entità non identificate esternamente e propagare gli identificatori che così si ottengono



Università(Nome, Indirizzo)

CorsoDiLaurea(Università, Codice, Denominazione)

Studente(Università, CodiceCdL, Matricola, Cognome, Nome)

Rispetto dei vincoli (1)

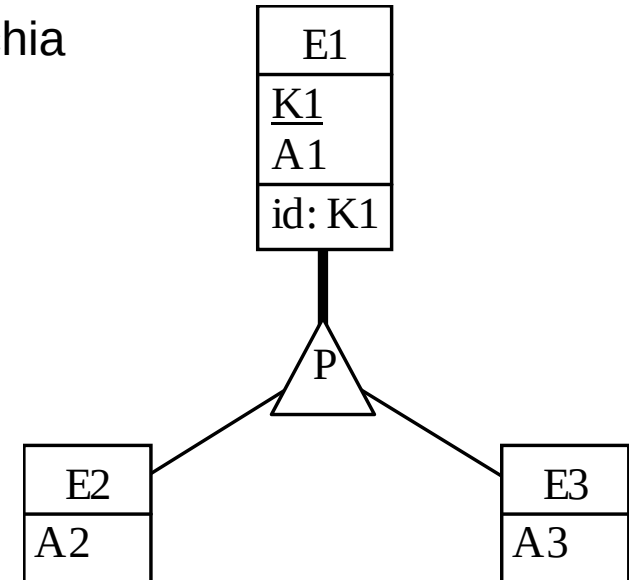
- La traduzione da uno schema E/R a uno schema relazionale non sempre consente di definire schemi di relazioni in grado di imporre tutti i vincoli presenti a livello concettuale

- Gli esempi che seguono, lungi dal coprire tutte le possibilità, mirano ad evidenziare in che modo vincoli apparentemente complessi si originino naturalmente quando si passa da uno schema E/R a uno relazionale

Rispetto dei vincoli (2)

GERARCHIE (T,E):

- Si consideri la traduzione della seguente gerarchia (supponendo per semplicità che tutti gli attributi dello schema siano di tipo INT)



mediante 2 relazioni (collasso verso il basso):

```
Create table E2 (
 K1 int not null primary key,
 A1 int not null,
 A2 int not null      );

Create table E3 (
 K1 int not null primary key,
 A1 int not null,
 A3 int not null );
```

Rispetto dei vincoli (3)

- Occorre garantire che le istanze di E2 ed E3 siano effettivamente disgiunte
- Se fossero ammesse clausole CHECK arbitrariamente complesse si potrebbe aggiungere allo schema di E2 il vincolo (analogamente per E3):

```
Create table E2 (
 K1 int not null primary key,
 A1 int not null,
 A2 int not null,
 constraint exclusive_hierarchy
  check (not exists (select * from E3 where E3.K1 = K1)) );

Create table E3 (
 K1 int not null primary key,
 A1 int not null,
 A3 int not null,
 constraint exclusive_hierarchy
  check (not exists (select * from E2 where E2.K1 = K1)) );
```

Rispetto dei vincoli (4)

Sopperire ai limiti della clausola CHECK

- Il vincolo **`exclusive_hierarchy`** può essere imposto,
  - lato client mediante *query di verifica* da eseguire prima di inserire una tupla in E2 (analogamente per E3), o
  - lato server mediante *before triggers*:

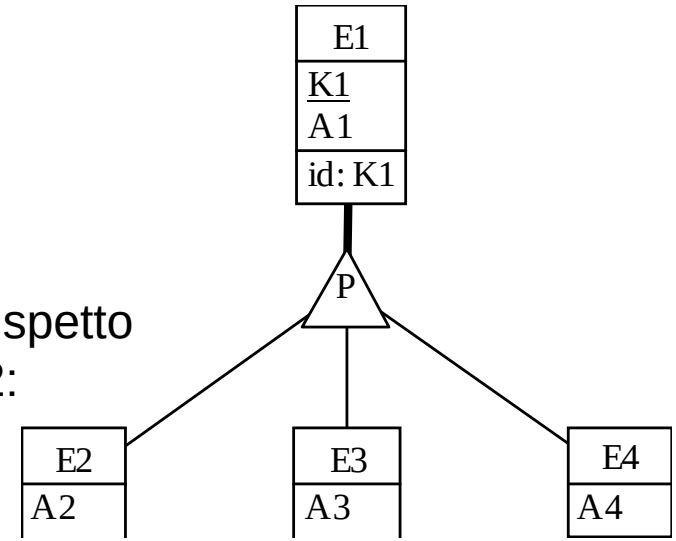
```
-- OK se non restituisce nessuna tupla (not exists)
Select * from E3 where E3.K1 = :K1;


Insert into E2 (K1,A1,A2)
Values (:K1,:A1,:A2);
```

Rispetto dei vincoli (5)

Tre o più entità specializzate:

- Con uno schema del tipo:



bisogna garantire la mutua esclusione rispetto a tutte le altre entità figlie. Ad es. per E2:

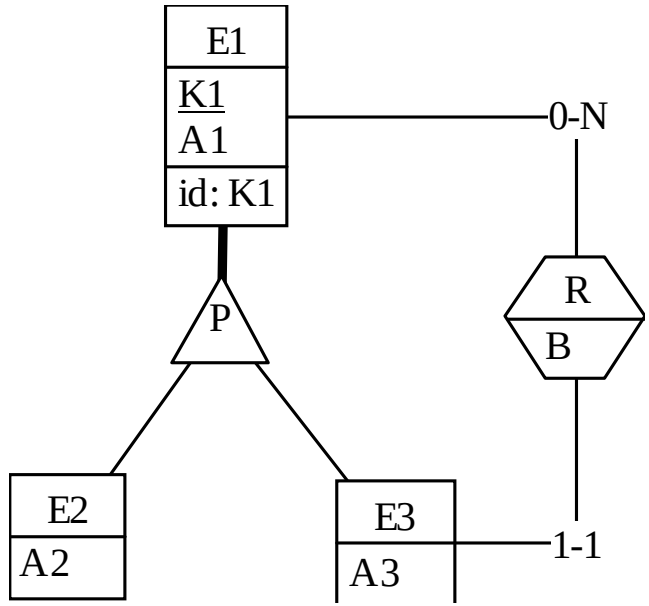
```
-- OK se non restituisce nessuna tupla
Select E3.K1 from E3 where E3.K1 = :K1 -- NB: non Select *
 UNION
Select E4.K1 from E4 where E4.K1 = :K1 ;
```

```
Insert into E2 (K1,A2)
Values (:K1,:A1,:A2);
```

Rispetto dei vincoli (6)

Collasso verso il basso con associazione:

- Lo schema:



viene tradotto collassando verso il basso e inglobando l'associazione R in E3:

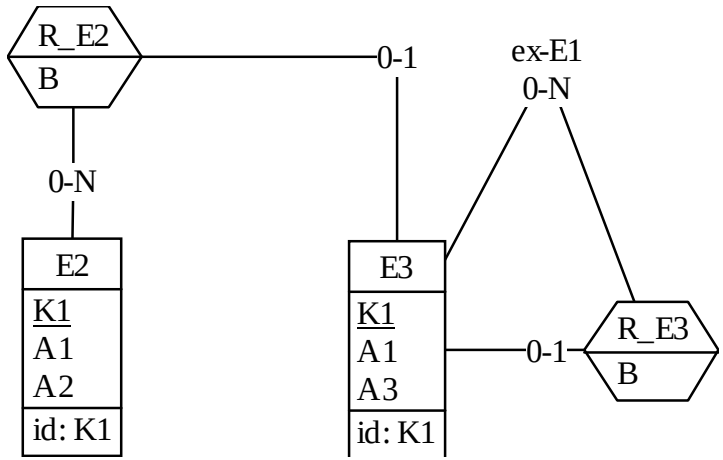
```
Create table E2 (
 K1 int not null primary key,
 A1 int not null,
 A2 int not null );

Create table E3 (
 K1 int not null primary key,
 A1 int not null,
 A3 int not null ); -- manca l'associazione
```

Rispetto dei vincoli (7)

Traduzione dell'associazione:

- Per tradurre correttamente l'associazione, occorre tenere presente che un elemento di E3 ora può referenziare o un elemento di E3 o uno di E2, come se lo schema fosse:



- Si introducono quindi 2 foreign key, che possono anche assumere valore nullo:

```
Create table E3 (
 K1 int not null primary key,
 A1 int not null,
 A3 int not null,
 K1_R_E2 int references E2,
 K1_R_E3 int references E3,
 B int not null );
```

Rispetto dei vincoli (8)

Vincolo sull'associazione:

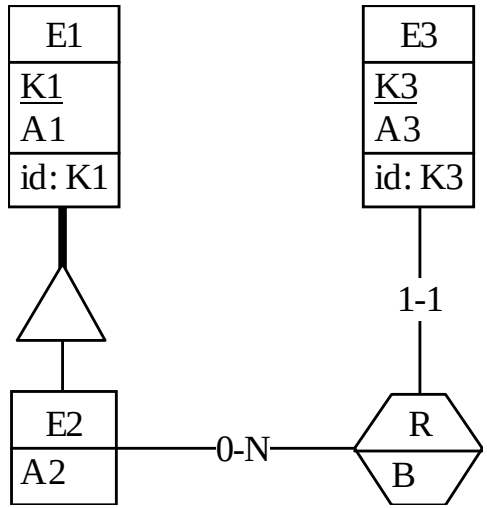
- Esattamente una delle due foreign key deve assumere valore nullo:

```
Create table E3 (
 K1 int not null primary key,
 A1 int not null,
 A3 int not null,
 K1_R_E2 int references E2,
 K1_R_E3 int references E3,
 B int not null,
 constraint R_always_defined
  check ( (K1_R_E2 is null and K1_R_E3 is not null) or
          (K1_R_E3 is null and K1_R_E2 is not null) )    );
```

Rispetto dei vincoli (9)

Associazioni che referenziano entità non più esistenti:

- Si consideri il seguente schema:



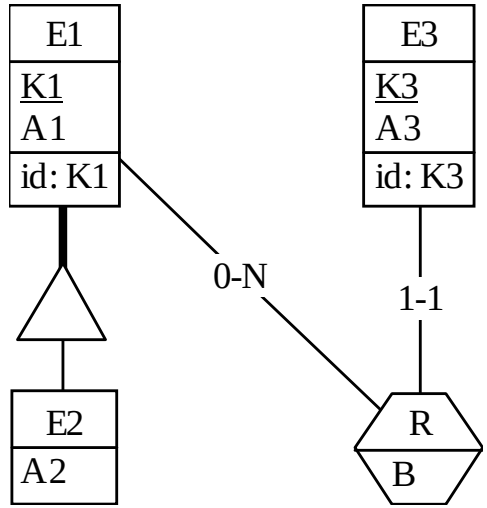
che viene tradotto <u>inglobando E2 in E1</u> e <u>R in E3</u>:

```
Create table E1 (
 K1 int not null primary key,
 A1 int not null,
 sel smallint not null
  check (sel in (1,2)), -- 2 = appartiene a E2
 A2 int,
 constraint E2_instance
  check ( (sel = 1 and A2 is null) or (sel = 2 and A2 is not null)) );

Create table E3 (
 K3 int not null primary key,
 A3 int not null
 K1 int not null references E1,
 B int not null);
```

Rispetto dei vincoli (10)

- Lo schema relazionale è impreciso, in quanto corrispondente allo schema E/R:



- Occorre quindi aggiungere un vincolo opportuno che, se si potesse disporre di clausole CHECK espressive, sarebbe:

```
Create table E3 (
 K3 int not null primary key,
 A3 int not null
 K1 int not null references E1,
 B int not null,
 constraint R_references_E2
  check (exists (select * from E1 where E1.K1 = K1 and sel = 2)) );
```

- Si deve fare quindi uso di query di verifica (o trigger), da eseguire prima di ogni inserimento in E3:

```
-- OK se restituisce una tupla (exists)
Select * from E1 where E1.K1 = :K1 and E1.sel = 2;
```

Rispetto dei vincoli (11)

Esempio di trigger per MySQL:

```
DELIMITER //  -- per usare ; nel trigger

CREATE TRIGGER CHECK_E2 BEFORE INSERT ON E3
FOR EACH ROW
BEGIN
IF NOT EXISTS
 (SELECT * FROM E1 WHERE E1.K1 = NEW.K1 and E1.sel = 2)
THEN SIGNAL SQLSTATE '45000'
       SET MESSAGE_TEXT = 'error message here';
ENDIF;
END; //

DELIMITER ;
```

SIGNAL: dalla versione 6.0.11
'45000': unhandled user-defined exception

Osservazioni finali

- La progettazione logica, pur potendosi avvalere di strumenti CASE, non va svolta “alla cieca”; in presenza di diverse alternative occorre valutare:
  - La presenza o meno di valori nulli, e la loro incidenza, che dipende dal volume dei dati
  - Il modo con cui le operazioni navigano lo schema E/R, che poi si traduce in operazioni di join tra le relazioni che vengono create

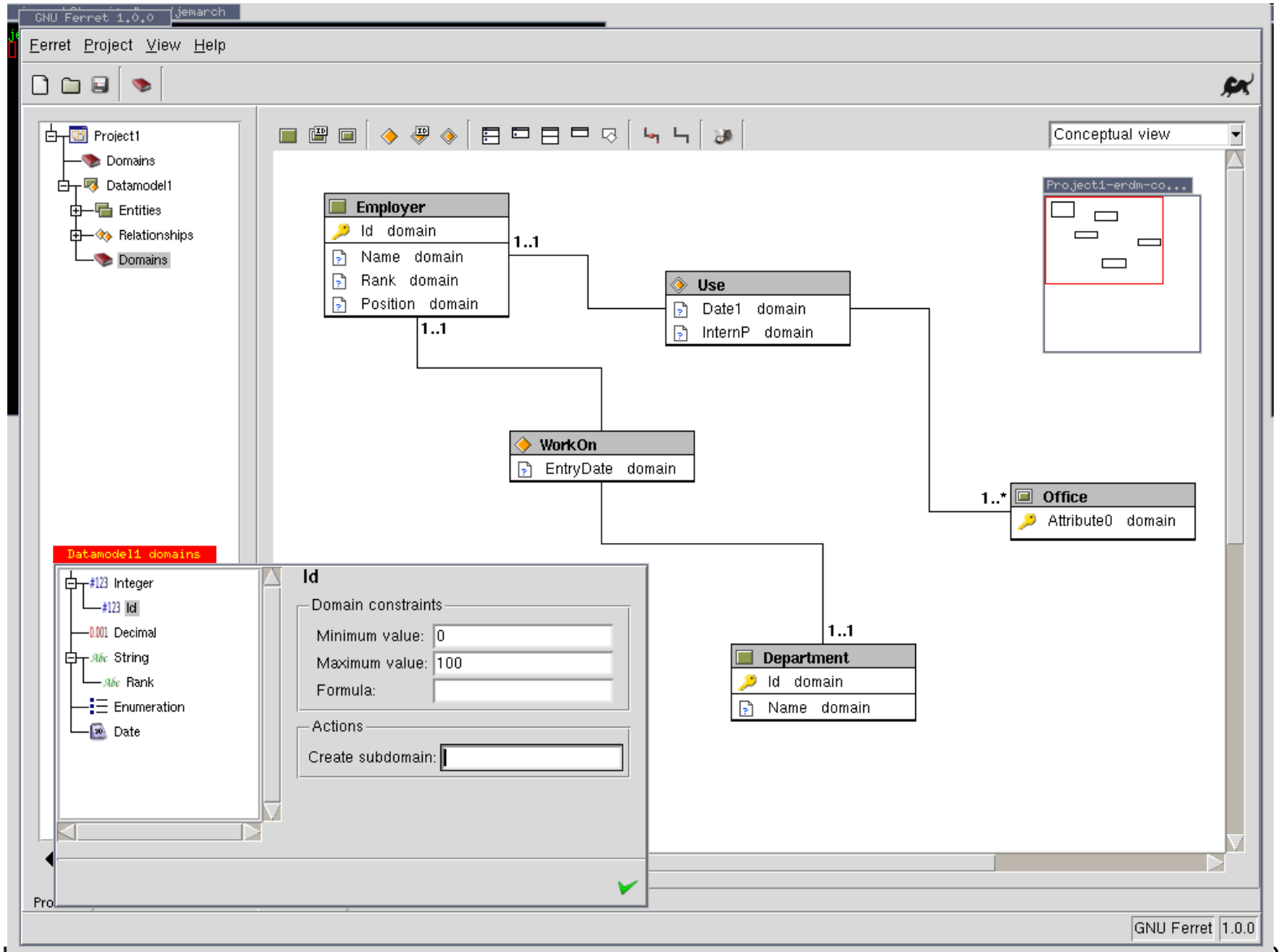
- I casi visti non esauriscono l’argomento e lasciano sempre lo spazio per soluzioni specifiche “ad hoc”
- Ad esempio, associazioni uno a molti con max-card(E2,R) = K, con K “piccolo”, possono al limite essere tradotte con 1 sola relazione, prevedendo K repliche degli attributi di E2 (es. tipico: numeri di telefono)

Tool di progettazione

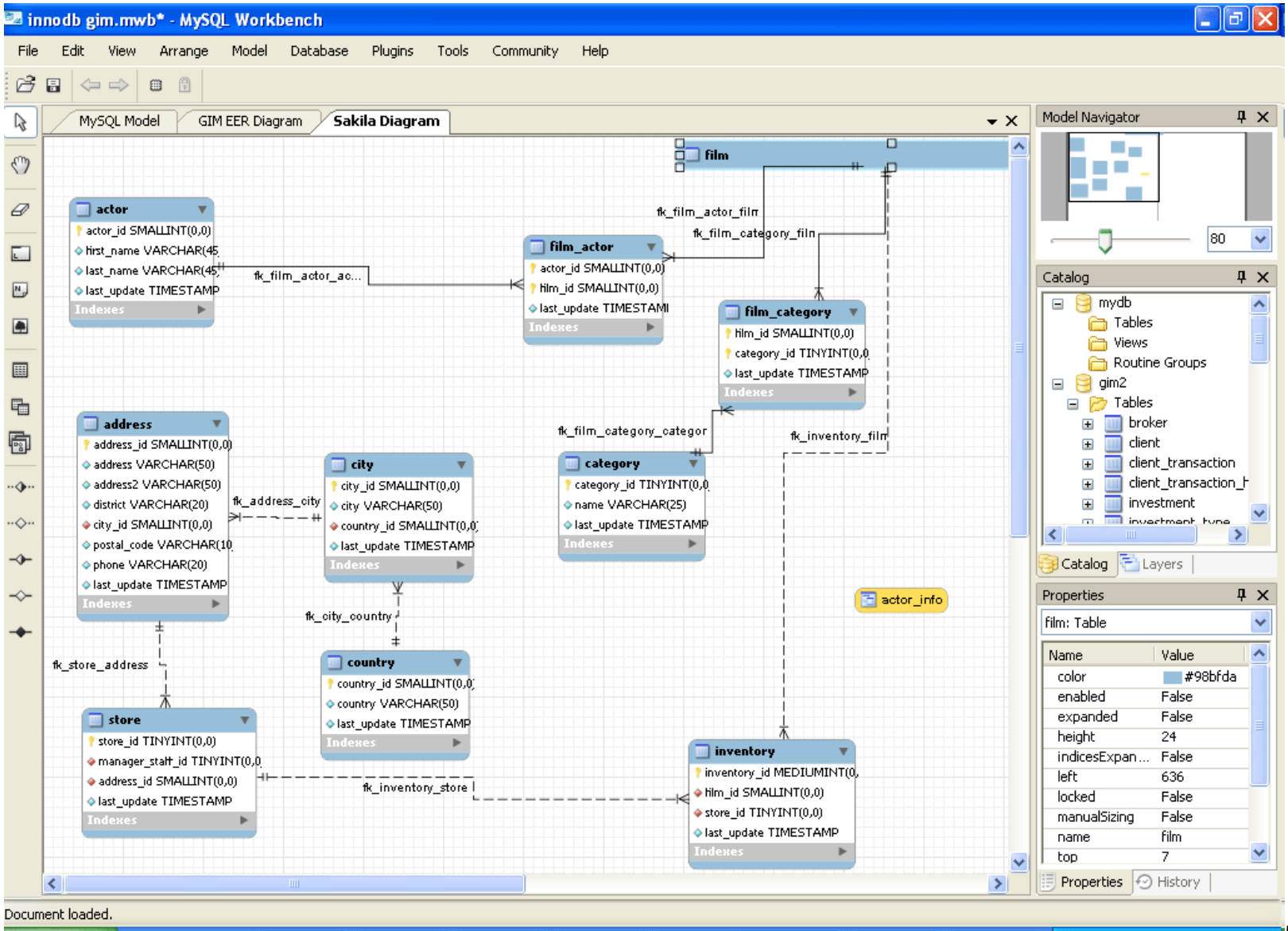
- Spesso: E/R = tables + links FK-PK ☹
- Oppure semplici E/R drawing tools
- Enfasi su generazione automatica del DDL

- Spesso assenti gerarchie, identificatori esterni, vincoli di cardinalità generici, associazioni n-arie

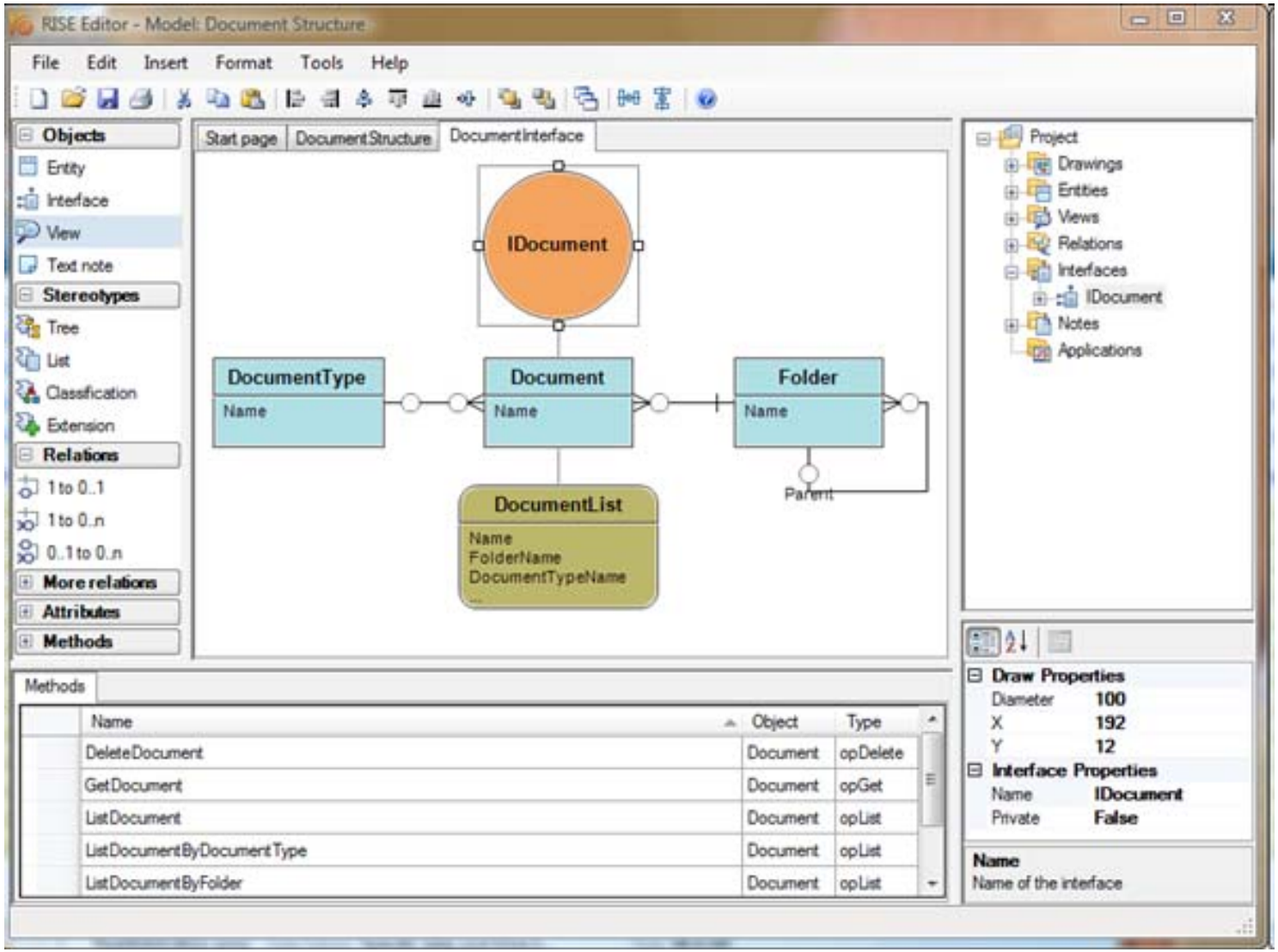
GNU Ferret http://www.gnuferret.org/



MySQL Workbench http://www.mysql.com/products/workbench/



RISE Editor http://www.risetobloome.com/

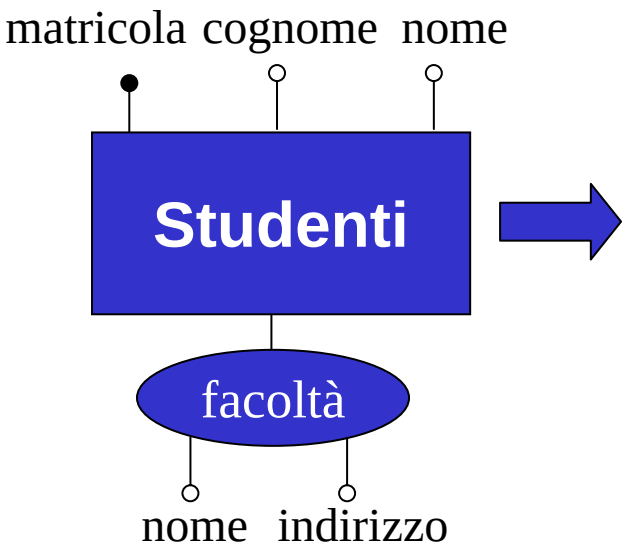


Appendice:
Normalizzazione di schemi relazionali

Forme normali

- Una forma normale è una proprietà di uno schema relazionale che ne garantisce la "qualità", cioè l'assenza di determinati difetti
- Una relazione non normalizzata:
  - presenta ridondanze,
  - esibisce comportamenti poco desiderabili durante gli aggiornamenti
- Le forme normali sono definite sul modello relazionale, ma hanno senso anche in altri contesti, ad esempio nel modello E/R
- L'attività che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale è detta normalizzazione
- La normalizzazione va utilizzata come tecnica di verifica dei risultati della progettazione di una base di dati
  - Non costituisce quindi una metodologia di progettazione
- La teoria relazionale mette a disposizione strumenti teorico-pratici per analizzare schemi e normalizzarli

Una relazione con anomalie



Matricola	Cognome	Nome	Fac_nome	Fac_ind
29323	Bianchi	Giorgio	Ingegneria	Risorgimento 2
35467	Rossi	Anna	Medicina	Massarenti 9
39654	Verdi	Marco	Ingegneria	Risorgimento 2
42132	Neri	Lucia	Agraria	Fanin 50
11274	Gialli	Luca	Agraria	Fanin 50

- L'indirizzo di una facoltà è ripetuto in tutte le tuple dei suoi studenti: ridondanza
- Se l'indirizzo di una facoltà cambia, è necessario modificare il valore in diverse tuple: anomalia di aggiornamento
- Una nuova facoltà senza studenti non può essere inserita: anomalia di inserimento
- …

Un’altra relazione con anomalie

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

Dipendenze funzionali

- Per formalizzare i problemi visti si introduce un nuovo tipo di vincolo, la dipendenza funzionale

- Dati uno schema R(X) e due sottoinsiemi (non vuoti) di attributi Y e Z di X, in R(X) vale la dipendenza funzionale (FD) $Y \rightarrow Z$ (Y determina funzionalmente Z, Z è funzione di Y) se

per ogni istanza r di R(X) e
per ogni coppia di tuple t1 e t2 in r con gli stessi valori su Y,
t1 e t2 hanno gli stessi valori anche su Z

Esempi di FD

- Nella relazione vista si hanno diverse FD, tra cui:

  Impiegato $\rightarrow$ Stipendio

  Progetto $\rightarrow$ Bilancio

  Impiegato, Progetto $\rightarrow$ Funzione

- Altre FD sono meno “interessanti” (“banali”), perché sempre soddisfatte, ad esempio:

  Impiegato, Progetto $\rightarrow$ Progetto

  - Y $\rightarrow$ A è non banale se A non appartiene a Y

FD e (super-)chiavi

- Se su uno schema R(X) vale la FD:

$$K \rightarrow X$$

allora K è un identificatore per R, ovvero una superchiave
(= chiave oppure chiave + altri attributi)

- Infatti se t1.K = t2.K allora t1.X = t2.X, quindi t1 e t2 sono la stessa tupla

$$\text{Matricola} \rightarrow \text{Matricola, Cognome, Nome, Fac\_nome, Fac\_ind}$$

- Quindi le FD generalizzano i vincoli di chiave

Anomalie e FD

- Le anomalie viste si riconducono alla presenza delle FD:

  Impiegato $\rightarrow$ Stipendio

  Progetto $\rightarrow$ Bilancio

- Viceversa la FD

  Impiegato, Progetto $\rightarrow$ Funzione

  non causa problemi
- Motivo:
  - La terza FD ha sulla sinistra una chiave e non causa anomalie
  - Le prime due FD non hanno sulla sinistra una chiave e causano anomalie

- La relazione contiene alcune informazioni legate alla chiave e altre ad attributi che non formano una chiave

La terza forma normale

- Esistono diverse forme normali definibili con riferimento alle FD. La più comunemente utilizzata è la:

Terza Forma Normale (3NF)

Uno schema R(X) è in terza forma normale se, per ogni dipendenza funzionale (non banale) $Y \rightarrow Z$ definita su di esso, Y è una superchiave di R(X) oppure ogni attributo in Z è contenuto in almeno una chiave di R(X)

- Una relazione in 3NF può ancora presentare anomalie (raramente)
- Tuttavia il vantaggio è che è sempre possibile normalizzare, ottenendo schemi in 3NF, che hanno due proprietà interessanti:
  - Si preservano i dati
  - Si preservano le dipendenze

Esempio di decomposizione

- Se uno schema non è in 3NF, la soluzione è “decomporlo”, sulla base delle FD
- L’idea, che funziona per casi semplici, è “estrarre” gli attributi che dipendono da attributi non chiave

Impiegato → Stipendio

Impiegato	Stipendio
Rossi	20
Verdi	35
Neri	55
Mori	48
Bianchi	48

Impiegato, Progetto → Funzione

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Verdi	Giove	progettista
Verdi	Venere	progettista
Neri	Venere	direttore
Neri	Giove	consulente
Neri	Marte	consulente
Mori	Marte	direttore
Mori	Venere	progettista
Bianchi	Venere	progettista
Bianchi	Giove	direttore

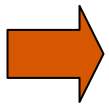
Progetto → Bilancio

Progetto	Bilancio
Marte	2
Giove	15
Venere	15

Attenzione!

- La soluzione non è sempre così semplice, bisogna fare anche altre considerazioni; ad esempio, operando come prima:

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano



Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Impiegato → Sede Progetto → Sede

…se proviamo a tornare indietro (Join su Sede):

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

Diversa dalla relazione di partenza!

Decomposizione senza perdita

- La decomposizione non deve assolutamente alterare il contenuto informativo del DB
- Si introduce pertanto il seguente requisito

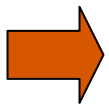
Decomposizione senza perdita *(lossless)*

Uno schema R(X) si decompone senza perdita negli schemi R1(X1) e R2(X2) se, per ogni istanza legale r su R(X), il join naturale delle proiezioni di r su X1 e X2 è uguale a r stessa:

- Una decomposizione con perdita può generare tuple spurie
- Per decomporre senza perdita è necessario e sufficiente che il join naturale sia eseguito su una superchiave di uno dei due sottoschemi, ovvero che valga $X1 \cap X2 \rightarrow X1$ oppure $X1 \cap X2 \rightarrow X2$

Esempio di decomposizione lossless

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano



Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere

… ma i problemi non sono ancora finiti…

Modifichiamo il DB…

- Supponiamo di voler inserire l'informazione che Neri lavora al progetto Marte:

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

- Ricostruendo la relazione otteniamo:

che però viola la FD Progetto → Sede!

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Neri	Marte	Milano

Preservazione delle dipendenze

- Diciamo che una decomposizione preserva le dipendenze se ciascuna delle dipendenze funzionali dello schema originario coinvolge attributi che compaiono tutti insieme in uno degli schemi decomposti
  - Nell'esempio Progetto $\rightarrow$ Sede non è conservata

- Se una FD non si preserva diventa più complicato capire quali sono le modifiche del DB che non violano la FD stessa

- In generale si devono prima eseguire query SQL di verifica o fare uso di trigger

Esempio

- Bisogna verificare che il progetto (Marte) sia presso la stessa sede dell'impiegato (Neri). A tal fine bisogna trovare un impiegato che lavora al progetto Marte

Impiegati

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

ImpProg

Impiegato	Progetto
Rossi	Marte
Verdi	Giove
Verdi	Venere
Neri	Saturno
Neri	Venere
Neri	Marte

```
SELECT *      -- OK se restituisce una tupla
FROM   Impiegati I
WHERE  I.Impiegato = 'Neri'
  AND  I.Sede IN ( SELECT I1.Sede
                   FROM   Impiegati I1, ImpProg IP
                   WHERE  I1.Impiegato = IP.Impiegato
                   AND  IP.Progetto = `Marte')
```

Qualità delle decomposizioni

- Una decomposizione:
  - deve essere senza perdita, per garantire la ricostruzione delle informazioni originarie
  - dovrebbe conservare le dipendenze, per semplificare il mantenimento dei vincoli di integrità originari
- Nel nostro esempio, questo suggerisce di inserire anche lo schema:
  - In questo caso va comunque eseguita una query, ma più semplice:

Progetti

Progetto	Sede
Marte	Roma
Giove	Milano
Venere	Milano
Saturno	Milano

```
SELECT *  -- OK se restituisce una tupla
FROM   Impiegati I, Progetti P
WHERE  I.Impiegato = 'Neri'
  AND  P.Progetto = `Marte'
  AND  I.Sede = P.Sede
```

Decomposizione in 3NF

- L'idea alla base dell'algoritmo che produce una decomposizione in 3NF è creare una relazione per ogni gruppo di FD che hanno lo stesso lato sinistro (determinante) e inserire nello schema corrispondente gli attributi coinvolti in almeno una FD del gruppo

- Per far questo è tuttavia necessario minimizzare l'insieme di FD individuate, altrimenti il risultato corretto non è garantito

Esempio: Se le FD individuate sullo schema R(ABCDEFG) sono:

AB → CDEF, C → F, F → G

si genererebbero gli schemi **R1(ABCDEF), R2(CF), R3(FG)**

Ma **R1** non è in 3NF a causa della FD **C → F!**

Innanzitutto: chiusura di X

- Come prima cosa chiediamoci:

  Se F è un insieme di FD su R(U) e X un insieme di attributi, quali attributi di U dipendono funzionalmente da X?

  Es: Se F include **A** $\rightarrow$ **B e B** $\rightarrow$ **C**, allora è anche vero che **A** $\rightarrow$ **C**.
  Infatti C dipende da B, che a sua volta dipende da A
- Denotiamo con X^+ l'insieme degli attributi di R(U) che dipendono da X
- Calcolare X^+ è semplice:

$X^+ = X$;
Ripeti:
Fine = true;
Per tutte le FD in F = $\{V_i \rightarrow W_i\}$:
Se $V_i \subseteq X^+$ allora: $\{X^+ = X^+ \cup W_i$; Fine = false$\}$
Fino a che Fine = true o $X^+ = U$

Chiusura di X - esempio

- Supponiamo di avere $F = \{A \rightarrow B, BC \rightarrow D, B \rightarrow E, E \rightarrow C\}$
  e calcoliamo A^+, ovvero l'insieme di attributi che dipendono da A

$A^+ = A$

$A^+ = AB$ poiché $A \rightarrow B$ e $A \subseteq A^+$

$A^+ = ABE$ poiché $B \rightarrow E$ e $B \subseteq A^+$

$A^+ = ABEC$ poiché $E \rightarrow C$ e $E \subseteq A^+$

$A^+ = ABECD$ poiché $BC \rightarrow D$ e $BC \subseteq A^+$

- Quindi da A dipendono tutti gli attributi dello schema, ovvero
  A è superchiave (e anche chiave)

Passo 1: FD “semplici”

- Per minimizzare un insieme di FD è innanzitutto necessario scriverle tutte in una forma “standard”, in cui sulla destra c’è sempre un singolo attributo
- Supponiamo di avere $F = \{AB \rightarrow CD, AC \rightarrow DE\}$
- Allora riscriviamo F come

$$F = \{AB \rightarrow C, AB \rightarrow D, AC \rightarrow D, AC \rightarrow E\}$$

Passo 2: attributi “estranei”

- In alcune FD è possibile che sul lato sinistro ci siano degli attributi inutili (“estranei”): vanno eliminati!
- Supponiamo di avere $F = \{AB \rightarrow C, A \rightarrow B\}$ e calcoliamo A^+

  $A^+ = A$

  $A^+ = AB$ poiché $A \rightarrow B$ e $A \subseteq A^+$

  $A^+ = ABC$ poiché $AB \rightarrow C$ e $AB \subseteq A^+$
- Quindi C dipende solo da A, ovvero in $AB \rightarrow C$ l’attributo B è estraneo (perché a sua volta dipendente da A) e possiamo riscrivere l’insieme di FD più semplicemente come: $F' = \{A \rightarrow C, A \rightarrow B\}$

Come facciamo a stabilire che in una FD del tipo $AX \rightarrow B$ l’attributo A è estraneo?

- Calcoliamo X^+ e verifichiamo se include B, ovvero se basta X a determinare B!

Passo 3: FD ridondanti

- Dopo avere eliminato gli attributi estranei si deve verificare se vi sono intere FD inutili ("ridondanti"), ovvero FD che sono implicate da altre
- Supponiamo di avere $F = \{B \rightarrow C, B \rightarrow A, C \rightarrow A\}$
- Si vede che $B \rightarrow A$ è ridondante in quanto bastano le altre due per stabilire che A dipende da B, e quindi possiamo riscrivere l'insieme di FD più semplicemente come: $F' = \{B \rightarrow C, C \rightarrow A\}$

  Come facciamo a stabilire che una FD del tipo $X \rightarrow A$ è ridondante?
- La eliminiamo da F, calcoliamo X^+ e verifichiamo se include A, ovvero se con le FD che restano riusciamo ancora a dimostrare che X determina A!

NB: NON INVERTIRE I PASSI 2 E 3!

- Sia $F = \{AB \rightarrow C, B \rightarrow A, C \rightarrow A\}$. Con il passo 2 scopriamo che A è estraneo in $AB \rightarrow C$, quindi otteniamo $F' = \{B \rightarrow C, B \rightarrow A, C \rightarrow A\}$ e dopo possiamo eliminare $B \rightarrow A$, restando con $F'' = \{B \rightarrow C, C \rightarrow A\}$
- Se invertiamo i passi non riusciamo più a eliminare $B \rightarrow A$!

Creazione degli schemi in 3NF

- Avendo minimizzato l'insieme iniziale di FD si può procedere con la creazione degli schemi in 3NF

Passo 4:

Si raggruppano tutte le FD che hanno lo stesso lato sinistro (determinante) X e si crea uno schema che ha X come chiave

Passo 5:

Se 2 o più determinanti si determinano reciprocamente, si fondono gli schemi (più chiavi alternate per lo stesso schema)

Passo 6:

Alla fine si verifica che esista uno schema la cui chiave è anche chiave dello schema originario (se non esiste lo si crea)

Creazione degli schemi – esempio 1

Sia $F = \{A \rightarrow BC, B \rightarrow C, ABE \rightarrow D\}$ e R(ABCDE)

Passo 1: $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, ABE \rightarrow D\}$

Passo 2: $F' = \{A \rightarrow B, A \rightarrow C, B \rightarrow C, AE \rightarrow D\}$ poiché B dipende da A

Passo 3: $F'' = \{A \rightarrow B, B \rightarrow C, AE \rightarrow D\}$ poiché $A \rightarrow C$ è ridondante

Passo 4: Si generano gli schemi R1(<u>A</u>B), R2(<u>B</u>C) e R3(<u>AE</u>D)

Passi 5 e 6: Le chiusure delle chiavi sono:

$A^+ = ABC$

$B^+ = BC$

$AE^+ = ABCED$ che è quindi anche chiave dello schema non decomposto

Creazione degli schemi – esempio 2

Sia $F = \{A \rightarrow BC, B \rightarrow AG, BE \rightarrow D\}$ e R(ABCDEG)

Passo 1: $F = \{A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow G, BE \rightarrow D\}$

Passo 2: F' = F

Passo 3: F'' = F

Passo 4: Si generano gli schemi R1(<u>A</u>BC), R2(<u>B</u>AG) e R3(<u>BE</u>D)

Passi 5 e 6: Le chiusure delle chiavi sono:

A^+ = ABCG

B^+ = BAGC

BE^+ = ABCEDG che è quindi anche chiave dello schema non decomposto

Tuttavia $A \rightarrow B$ e $B \rightarrow A$, quindi si fondono R1 e R2 in R12(<u>A</u>BCG), con, ad es., A chiave primaria e B chiave alternata

Creazione degli schemi – esempio 3

Sia F = {Imp $\rightarrow$ Stip, Prog $\rightarrow$ Bilancio}

Passo 1: F = {Imp $\rightarrow$ Stip, Prog $\rightarrow$ Bilancio}

Passo 2: F' = F

Passo 3: F'' = F

Passo 4: Si generano gli schemi IMP(Imp,Stip) e PROG(Prog,Bilancio)

Passo 5: Non si fonde nulla

Passo 6: Né Imp né Prog sono chiavi dello schema non decomposto; tuttavia Imp,Prog lo è, quindi si crea lo schema

PARTECIPAZIONI(Imp,Prog)

Normalizzare o no?

- La normalizzazione non va intesa come un obbligo, in quanto in alcune situazioni le anomalie che si riscontrano in schemi non normalizzati sono un male minore rispetto alla situazione che si viene a creare normalizzando
- In particolare, le cose da considerare sono:
  - Normalizzare elimina le anomalie, ma può appesantire l'esecuzione di certe operazioni (join tra gli schemi normalizzati)
  - La frequenza con cui i dati vengono modificati incide su qual è la scelta più opportuna (relazioni "quasi statiche" danno meno problemi se non normalizzate)
  - La ridondanza presente in relazioni non normalizzate va quantificata, per capire quanto incida sull'occupazione di memoria, e sui costi da pagare quando le repliche di una stessa informazione devono essere aggiornate