Basi di dati

Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza"

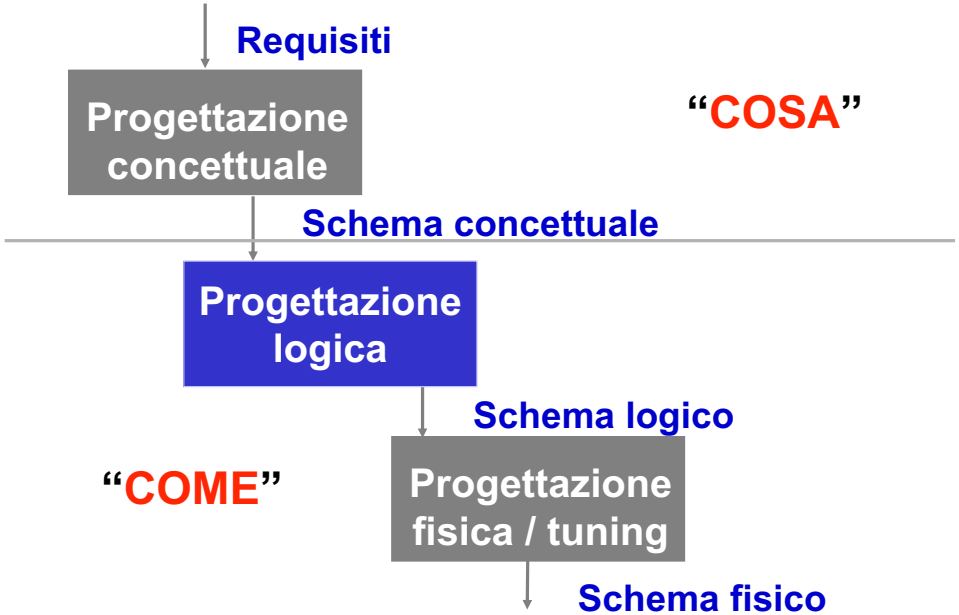
Anno Accademico 2005/2006
Canale M-Z

http://www.dis.uniroma1.it/~degiacomo/didattica/basidati/

5. La progettazione logica

5.1 introduzione alla progettazione logica

Fasi della progettazione



Obiettivo della progettazione logica

Tradurre lo schema concettuale (espresso nel modello ER con vincoli) in uno schema logico che rappresenti gli stessi dati:

- utilizzando il **modello logico** del DBMS scelto
  (nel nostro caso, il modello relazionale con vincoli)
- in maniera **corretta**, **completa** ed **efficiente**

Dati di ingresso e uscita

Ingresso:

- schema concettuale (diagramma ER e dizionario dei dati)
- informazioni sul carico applicativo

Uscita:

- schema logico (nel modello relazionale)
- vincoli aggiuntivi
- documentazione associata

Non si tratta di una pura e semplice traduzione

Motivi:

- Alcuni aspetti dello schema ER possono non essere direttamente rappresentabili nel modello relazionale. È quindi opportuno ristrutturare lo schema ER in modo da renderlo traducibile in modo diretto.
- È necessario porre attenzione alle **prestazioni**.
- Adottiamo un semplice **modello di costo** che permette di fornire una valutazione approssimata delle prestazioni della base di dati in funzione di un certo carico applicativo.
- Le scelte durante in fase di progettazione logica devono essere effettate con l'obiettivo di ottimizzare le prestazioni.

Carico applicativo

Consideriamo degli "indicatori" dei parametri che regolano le prestazioni:

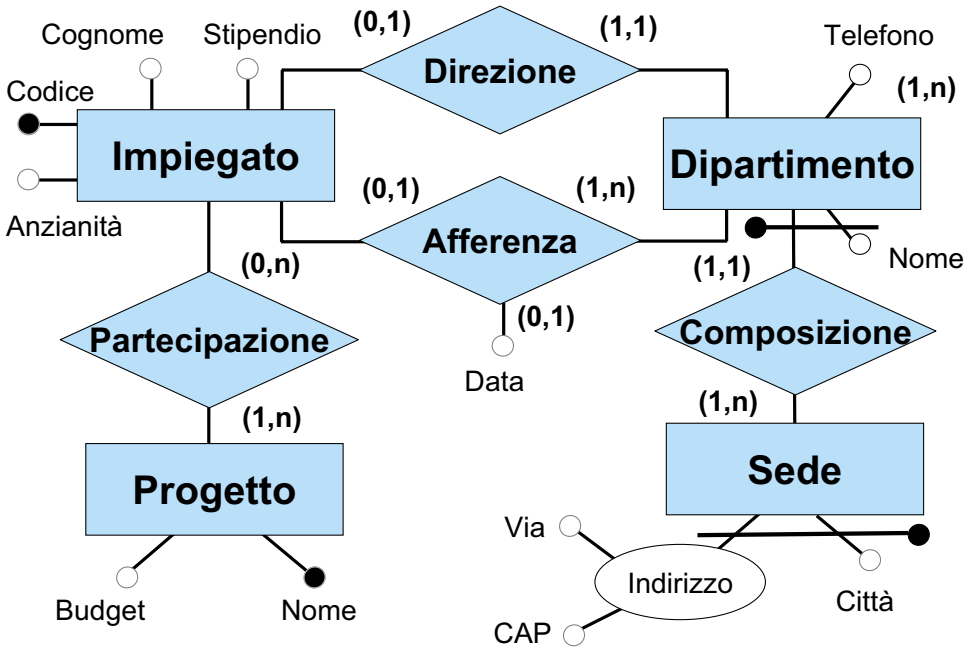
- **tempo di esecuzione** delle operazioni di principale interesse: numero di istanze (di entità e relazioni) mediamente accedute durante l'esecuzione dell'operazione (accessi)
- **spazio di memoria** necessario per memorizzare i dati di interesse

Per valutare questi parametri bisogna conoscere (oltre allo schema):

- **volume dei dati**:
  - numero di istanze previste di entità e relazioni
  - dimensione di ciascun attributo
- **caratteristiche delle operazioni**:
  - tipo: interattiva o batch
  - frequenza: numero medio di esecuzioni in un certo periodo
  - dati coinvolti

Si noti che la valutazione sarà necessariamente approssimata, in quanto le prestazioni effettive della base di dati dipendono anche da parametri fisici, difficilmente prevedibili in questa fase (DBMS utilizzato, indici, ...).

Esempio di carico applicativo: schema



Esempio di carico applicativo: operazioni

Supponiamo che le operazioni di interesse siano:

1. Assegna un impiegato ad un progetto.
2. Trova tutti i dati di un impiegato, del dipartimento nel quale lavora e dei progetti ai quali partecipa.
3. Trova i dati di tutti gli impiegati di un certo dipartimento.
4. Per ogni sede, trova i suoi dipartimenti con il cognome del direttore e l'elenco degli impiegati del dipartimento.

Tabella dei volumi e tavola delle operazioni

Tabella dei volumi

Concetto	Costrutto	Volume
Sede	Entità	10
Dipartimento	Entità	80
Impiegato	Entità	200
Progetto	Entità	500
Composizione	Relazione	80
Afferenza	Relazione	190
Direzione	Relazione	80
Partecipazione	Relazione	600

Tabella delle operazioni

Op.	Tipo	Frequenza
1	Interattiva	50 / giorno
2	Interattiva	100 / giorno
3	Interattiva	10 / giorno
4	Batch	2 / sett.

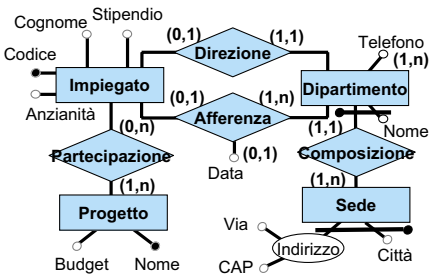
Tabella dei volumi

Si noti che i valori relativi al numero di istanze di entità e relazioni nella tabella dei volumi sono influenzati:

- dalle cardinalità nello schema
- dal numero medio di volte che le istanze delle entità partecipano alle relazioni

Esempio:

- vol(Composizione) = vol(Dipartimento)
- vol(Direzione) = vol(Dipartimento)
- vol(Afferenza) ≤ vol(Impiegato)
- se ogni impiegato partecipa in media a 3 progetti:
  vol(Partecipazione) ≈ 3 × vol(Impiegato)



Valutazione di costo

Per valutare il costo di un'operazione, si costruisce una **tabella degli accessi** basata su uno schema di navigazione associato all'operazione.

Esempio: trova tutti i dati di un impiegato, del dipartimento nel quale lavora e dei progetti ai quali partecipa (operazione 2).

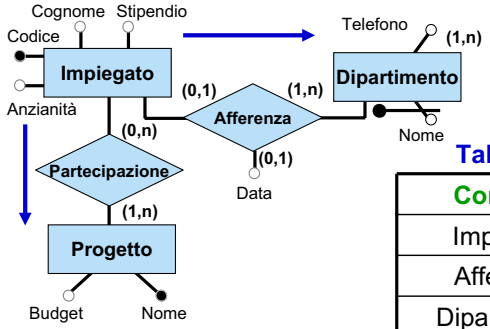


Tabella degli accessi dell'operazione 2

Concetto	Costrutto	Accessi	Tipo
Impiegato	Entità	1	L
Afferenza	Relazione	1	L
Dipartimento	Entità	1	L
Partecipazione	Relazione	3	L
Progetto	Entità	3	L

Valutazione dei costi e traduzione

- La traduzione dello schema concettuale in uno schema relazionale è guidata dall'**obiettivo di ottimizzare le prestazioni**.
- La valutazione delle prestazioni può essere effettuata adottando il modello di costo appena visto.
- Alcune scelte del processo di traduzione sono di fatto fisse e dettate dalla struttura dello scheme ER (e quindi dalle scelte effettuate in fase di progettazione concettuale). In determinati casi invece il progettista deve effettuare delle scelte volte a ottimizzare le prestazioni.
- Vediamo una metodologia di progettazione, articolata in diverse fasi, nella quale i momenti in cui il progettista deve effettuare scelte progettuali sono chiaramente delimitati.

Fasi della progettazione logica

1. **Ristrutturazione dello schema ER**:
   - eliminazione dei costrutti non direttamente traducibili nel modello relazionale
   - scelta degli identificatori principali delle entità
2. **Traduzione diretta** dello schema ER ristrutturato nel modello relazionale:
   - la traduzione è **diretta**, nel senso che non richiede (quasi) scelte da parte del progettista
   - lo schema relazionale prodotto **non** contiene ridondanze, se non quelle volute
   - la traduzione diretta tiene conto delle scelte fatte in fase di progettazione concettuale
3. **Ristrutturazione dello schema relazionale**:
   - richiede delle scelte da parte del progettista, tenendo conto delle prestazioni (carico applicativo)

5. La progettazione logica

5.2 ristrutturazione dello schema ER

Ristrutturazione dello schema ER

Motivazioni:

- **semplificare** la successiva fase di traduzione nel modello relazionale eliminando quei costrutti non direttamente traducibili
- tenere conto di aspetti relativi all'**efficienza**

Osservazione:

- uno schema ER **ristrutturato** è uno schema ER **degradato** dal punti di vista semantico per avvicinarsi al modello relazionale

Attività della ristrutturazione dello schema ER

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 – fase 1: analisi delle ridondanze

- Una **ridondanza** (estensionale) in uno schema ER è una informazione significativa ma derivabile da altre.
- Le ridondanze, se presenti, devono essere documentate (ovvero espresse attraverso **vincoli**).
- In questa fase si decide se eliminare le ridondanze eventualmente presenti o mantenerle, e se introdurne delle nuove.
- **Vantaggi** nel mantenere una ridondanza:
  - potenziale maggiore efficienza nella esecuzione delle interrogazioni
- **Svantaggi** nel mantenere una ridondanza:
  - gestione dei vincoli aggiuntivi
  - appesantimento degli aggiornamenti
  - maggiore occupazione di spazio

Analisi delle ridondanze

Abbiamo visto che le forme di ridondanza estensionale in uno schema ER sono date da:

- attributi derivabili:
  - da altri attributi della stessa entità (o relazione)
  - da attributi di altre entità (o relazioni)
- relazioni derivabili dalla composizione di altre relazioni in presenza di cicli

Per ciascuna ridondanza bisogna valutare, in funzione del carico applicativo previsto (aggiornamenti, interrogazioni, occupazione di spazio) se è opportuno mantenerla oppure eliminarla.

Se si sceglie di mantenere la ridondanza, questa deve essere **documentata** (attraverso opportuni vincoli).

Esempio di analisi delle ridondanze

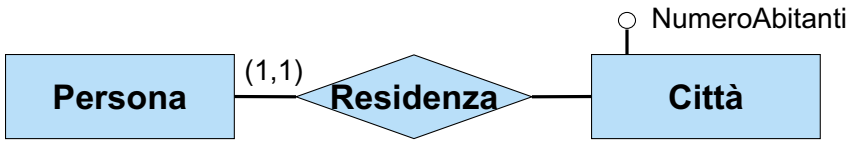


Tabella dei volumi

Concetto	Costrutto	Volume
Città	Entità	200
Persona	Entità	1.000.000
Residenza	Relazione	1.000.000

Operazione 1: memorizza una nuova persona con la relativa città di residenza (500 volte al giorno)

Operazione 2: stampa tutti i dati di una città, incluso il numero di abitanti (2 volte al giorno)

Valutazione dei costi in assenza di ridondanza

Tabella degli accessi operazione 1

Concetto	Costrutto	Accessi	Tipo
Persona	Entità	1	S
Residenza	Relazione	1	S

Tabella degli accessi operazione 2

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L
Residenza	Relazione	5000	L

Costi:

- operazione 1: 1000 accessi in scrittura al giorno
- operazione 2: 10000 accessi in lettura al giorno

Contiamo doppi gli accessi in scrittura: **totale di 12000 accessi al giorno**

Valutazione dei costi in presenza di ridondanza

Tabella degli accessi operazione 1

Concetto	Costrutto	Accessi	Tipo	
Persona	Entità	1	S	(scrittura dati su persona)
Residenza	Relazione	1	S	(associazione città di res.)
Città	Entità	1	L	(lettura numero abitanti)
Città	Entità	1	S	(scrittura nuovo num. abit.)

Tabella degli accessi operazione 2

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L

Costi:

- operazione 1: 1500 accessi in scrittura e 500 in lettura al giorno
- operazione 2: trascurabile

Contiamo doppi gli accessi in scrittura: **totale di 3500 accessi al giorno**

Conclusione: in questo esempio, manteniamo la ridondanza.

Ristrutturazione – fase 2: eliminazione di attributi multivalore

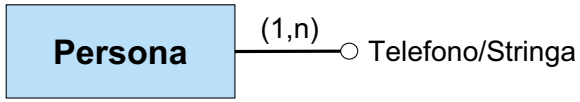
- Un **attributo multivalore** (ovvero un attributo con cardinalità massima maggiore di 1) non può essere tradotto direttamente nel modello relazionale senza introdurre delle ridondanze nelle relazioni ottenute.
- Dobbiamo quindi eliminare tutti gli attributi multivalore.
- L'eliminazione di un **attributo multivalore di un'entità** si effettua trasformando l'attributo in una relazione binaria, ed introducendo un'opportuna entità per il dominio (cfr. parte 4, esercizio 19a).
- L'eliminazione di un **attributo multivalore di una relazione** richiede la preventiva trasformazione della relazione in un'entità (cfr. parte 4, esercizio 19c e 19d).

Nota: se trasformiamo una relazione R in un'entità, ed R era in ISA o in una gerarchia con altre relazioni, allora anche queste devono essere trasformate in entità (cfr. parte 4 esercizio 18c con ISA tra relazioni).

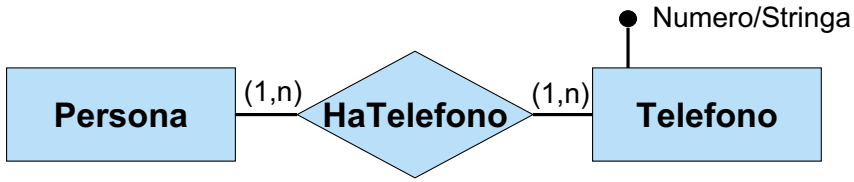
Eliminazione di attributi multivalore di entità

Si trasforma l'attributo multivalore dell'entità in una relazione e il corrispondente dominio in entità.

Esempio:

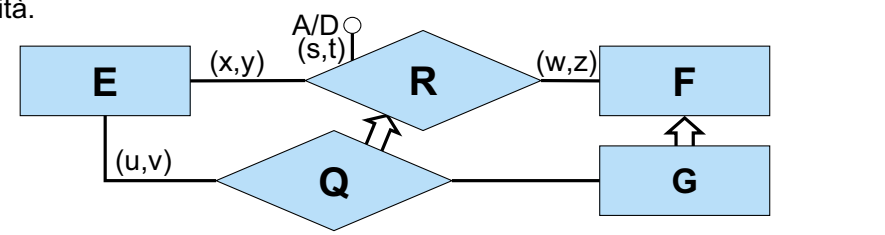


Si trasforma in:

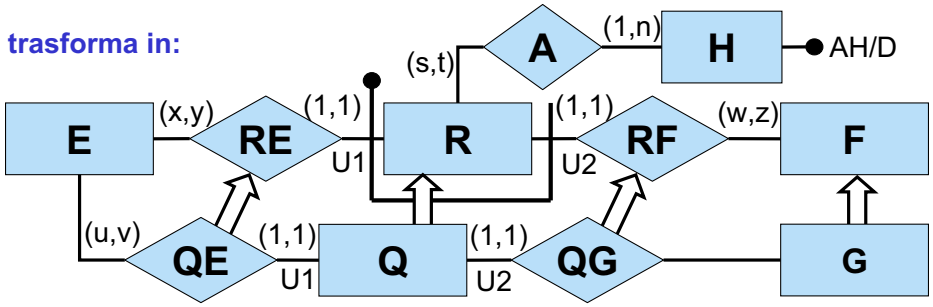


Eliminazione di attributi multivalore di relazioni

Si trasforma la relazione R in entità e l'attributo multivalore di R in una relazione. Anche eventuali relazioni in ISA con R devono essere trasformate in entità.



Si trasforma in:



Ristrutturazione – fase 3: eliminazione di attributi composti

Un **attributo composto** di un'entità (o di una relazione) a questo punto ha cardinalità (1,1) oppure (0,1).

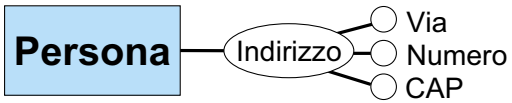
- Se l'attributo ha cardinalità (1,1), si associano direttamente gli attributi componenti all'entità (o alla relazione).
- Se l'attributo ha cardinalità (0,1) si può
  - procedere come per gli attributi (1,1), ma con l'avvertenza che l'opzionalità diventa un vincolo esterno
  - oppure trasformare l'attributo composto in una relazione binaria introducendo una nuova entità, come fatto per gli attributi multivalore (cfr. parte 4 esercizio 19b), mantenendo la cardinalità (0,1) sulla relazione.

    Ovviamente, se l'attibuto composto è di una relazione, è necessario trasformare tale relazione in un'entità.

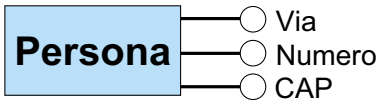
Eliminazione di attributi composti: alternativa 1

Si associano direttamente gli attributi componenti all'entità (o alla relazione) a cui è associato l'attributo:

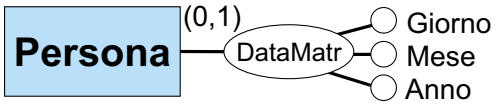
Esempio:



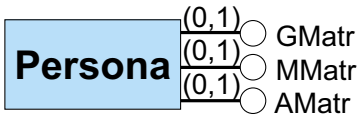
viene trasformato in:



Esempio:



viene trasformato in:

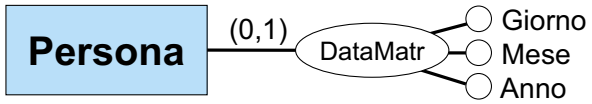


... con il vincolo esterno: per ogni istanza di Persona, ciascun attributo tra GMatr, MMatr e AMatr è definito se e solo se lo sono anche gli altri due.

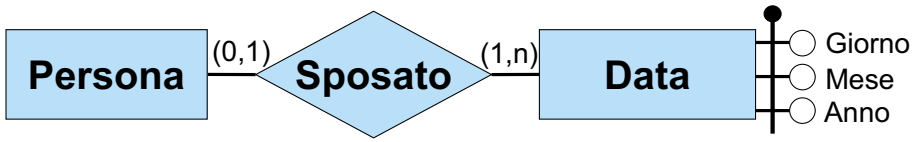
Eliminazione di attributi composti: alternativa 2

Si trasforma l'attributo composto in una relazione binaria e si introduce una nuova entità.

Esempio:



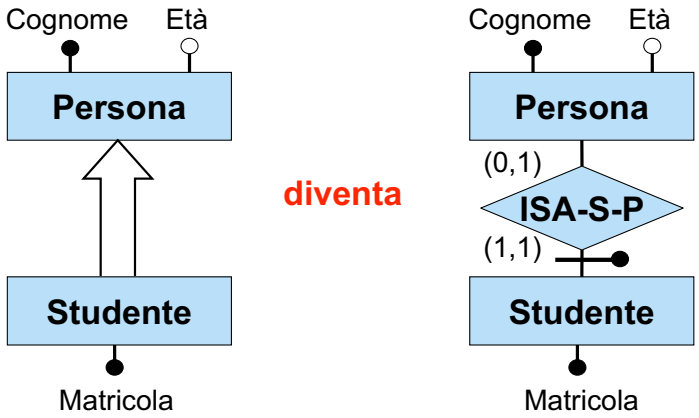
Viene trasformato in:



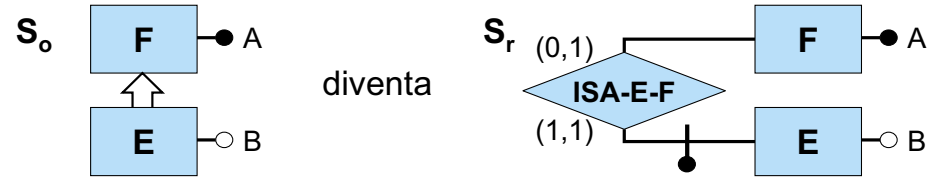
Ristrutturazione – fase 4: eliminazione di ISA tra entità

Una relazione **E ISA F** tra due entità **E** ed **F** viene sostituita da una nuova relazione binaria **ISA-E-F** tra **E** ed **F** a cui **E** partecipa con cardinalità (1,1) e **F** con cardinalità (0,1). Agli eventuali identificatori di **E** viene aggiunto un identificatore esterno dato dalla partecipazione ad **ISA-E-F**.

Esempio:



Eliminazione di ISA tra entità: livello estensionale



Istanza di So:
istanze(F) = { f_1, f_2, f_3 }
istanze(E) = { f_1, f_2 }
istanze(A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(B) = { (f_1,b_1), (f_2 ,b_2) }

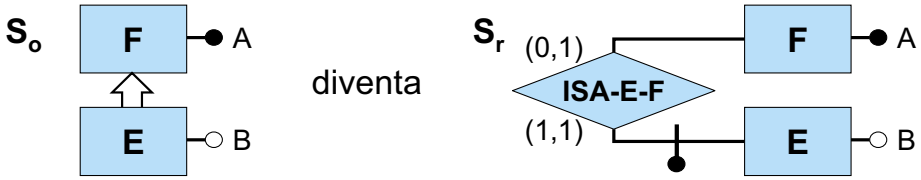
Istanza di Sr:
istanze(F) = { f_1, f_2, f_3 }
istanze(E) = { e_1, e_2 }
istanze(A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(B) = { (e_1,b_1), (e_2 ,b_2) }
istanze(ISA-E-F) = { (E:e_1,F:f_1), (E:e_2,F:f_2) }

Si noti che nello schema S_r risultante dall'eliminazione delle ISA da S_o, **tutte le entità sono disgiunte a coppie**, e quindi non hanno più istanze in comune (su questo torneremo fra breve).

Eliminazione di ISA tra entità: corrispondenza tra istanze

Esiste una stretta corrispondenza tra le istanze di uno schema S_o e le istanze dello schema S_r ottenuto da S_o eliminando le ISA tra entità.

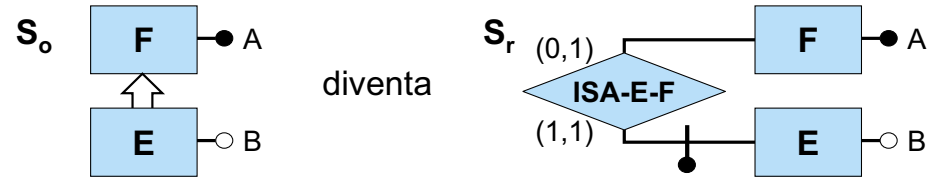
Mostriamo questa proprietà tramite un esempio:



In particolare, mostriamo che esistono due funzioni *g* ed *h* tali che:

- *g* è una funzione totale da *istanze*(S_o) a *istanze*(S_r)
- h è una funzione totale da *istanze*(S_r) a *istanze*(S_o)
- per ogni istanza I_o di S_o, si ha che $h(g(I_o)) = I_o$

Corrispondenza tra istanze – funzione g

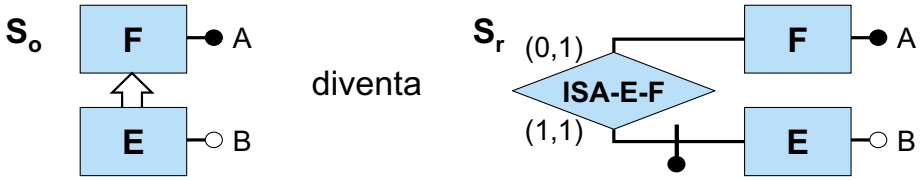


Definiamo la funzione g: istanze(S_o) → istanze(S_r) in modo che, ad un'istanza I_o di S_o, g assegni un'istanza I_r di S_r definita al seguente modo:

- *istanze*(I_r,F) = *istanze*(I_o,F)
- *istanze*(I_r,A) = *istanze*(I_o,A)
- per definire *istanze*(I_r,E), introduciamo in I_r, per ogni x∈*istanze*(I_o,E), un nuovo oggetto $g_E(x)$, e definiamo
  istanze(I_r,E) = { $g_E(x)$ | x∈*istanze*(I_o,E) }
- *istanze*(I_r,ISA-E-F) = { ($g_E(x)$,x) | x∈*istanze*(I_o,E) }
- *istanze*(I_r,B) = { ($g_E(x)$,b) | (x,b)∈*istanze*(I_o,B) }

È facile verificare che I_r così definita è effettivamente un'istanza di S_r.

Corrispondenza tra istanze – funzione h

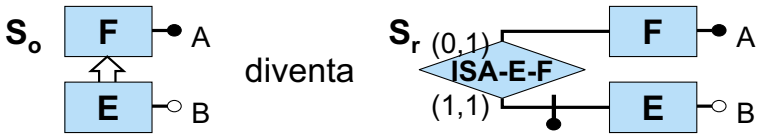


Definiamo la funzione h: istanze(S_r) → istanze(S_o) in modo che, ad un'istanza I_r di S_r, h assegni l'istanza I_o di S_o definita al seguente modo:

- *istanze*(I_o,F) = *istanze*(I_r,F)
- *istanze*(I_o,A) = *istanze*(I_r,A)
- *istanze*(I_o,E) = { x∈*istanze*(I_r,F) | esiste un y∈*istanze*(I_r,E) con (y,x)∈*istanze*(I_r,ISA-E-F) }
- *istanze*(I_o,B) = { (x,b) | x∈*istanze*(I_o,E), (y,x)∈*istanze*(I_r,ISA-E-F) e (y,b)∈*istanze*(I_r,E) }

È facile verificare che I_o così definita è effettivamente un'istanza di S_o, e che inoltre h(g(I_o)) = I_o.

Osservazione sullo schema risultante



Istanza di So:
istanze(F) = { f_1, f_2, f_3 }
istanze(E) = { f_1, f_2 }
istanze(A) = { (f_1,a_1), (f_2,a_2), (f_3,a_3)}
istanze(B) = { (f_1,b_1), (f_2,b_2) }

Istanza di Sr:
istanze(F) = { f_1, f_2, f_3 }
istanze(E) = { e_1, e_2 }
istanze(A) = { (f_1,a_1), (f_2,a_2), (f_3,a_3)}
istanze(B) = { (e_1,b_1), (e_2,b_2) }
istanze(ISA-E-F) = {(E:e_1,F:f_1), (E:e_2,F:f_2) }

Come osservato prima, nello schema S_r risultante dall'eliminazione delle ISA da S_o, **tutte le entità sono disgiunte a coppie**.

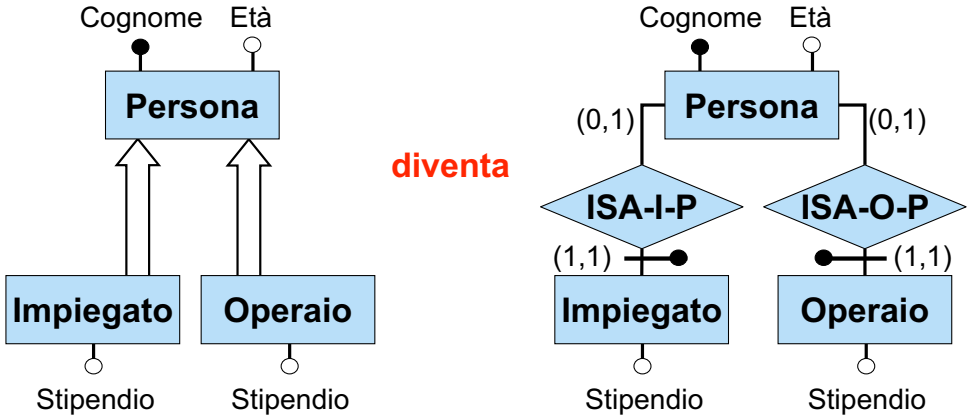
Come si concilia questa osservazione con il fatto che nello schema originario ciò non era vero? La risposta sta nelle funzioni g ed h illustrate precedentemente. Tramite queste funzioni è possibile infatti stabilire se due qualunque istanze di entità nello schema risultante corrispondono ad un'unica istanza di entità nello schema originario (ad es., e_1 ed f_1 nella istanza di S_r corrispondono entrambi ad f_1 nella istanza di S_o).

Attributi in comune nella eliminazione di ISA tra entità

Se due entità non disgiunte nello schema originario hanno un attributo A in comune, applicando la trasformazione per eliminare la relazione ISA, nelle due corrispondenti entità disgiunte dello schema risultante troviamo due attributi di nome A, che di fatto rappresentano due funzioni (o relazioni, se A è multivalore) diverse.

Al fine di mantenere la semantica originaria, dobbiamo quindi imporre che le due funzioni, quando applicate a due oggetti e, f dello schema risultante che corrispondono allo stesso oggetto nello schema di partenza, assegnino ad e ed f lo stesso valore. Ciò viene fatto con un vincolo esterno sullo schema ristrutturato.

Attributi in comune nella eliminazione di ISA tra entità: esempio



con il **vincolo esterno** nello schema risultante:

per ogni *p* ∈ *istanze*(Persona), se esistono *i* ∈ *istanze*(Impiegato), e *o* ∈ *istanze*(Operaio) tali che (*i*,*p*) ∈ *istanze*(ISA-I-P) e (*o*,*p*) ∈ *istanze*(ISA-O-P), allora Stipendio(*i*) = Stipendio(*o*)

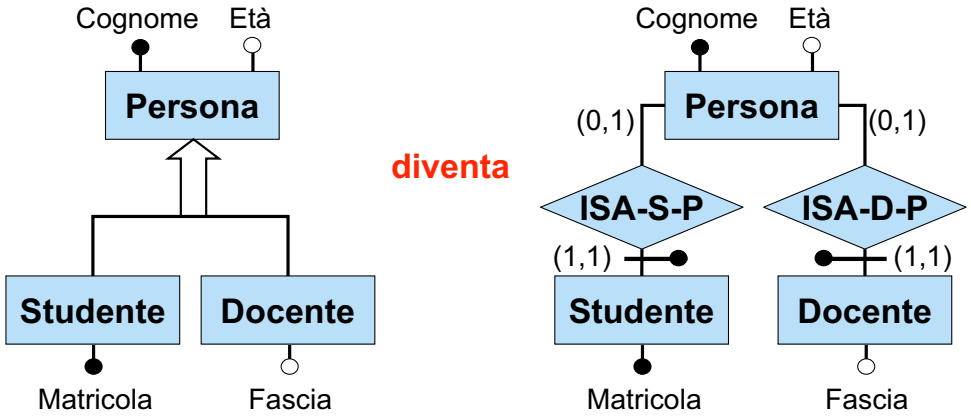
Ristrutturazione – fase 4: eliminazione di generalizzazioni tra entità

- Una generalizzazione tra una entità padre **F** e le sottoentità $E_1, E_2, \ldots, E_n$, viene trattata come n relazioni E_1 **ISA F**,…, E_n **ISA F**, introducendo n relazioni binarie **ISA-E_1-F**,…, **ISA-E_n-F**.
- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni, detti **vincoli di generalizzazione**:
  - la proprietà $istanze(E_1) \cap \ldots \cap istanze(E_n) = \varnothing$ dello schema di partenza corrisponde nello schema ristrutturato al vincolo:

    *ogni istanza di **F** partecipa*
    al più ad una** delle relazioni **ISA-E_1-F,…, ISA-E_n-F
  - se la **generalizzazione** è **completa**, l'ulteriore proprietà $istanze(E_1) \cup \ldots \cup istanze(E_n) = istanze(F)$ dello schema di partenza corrisponde nello schema ristrutturato al vincolo:

    *ogni istanza di **F** partecipa*
    esattamente ad una** delle relazioni **ISA-E_1-F,…, ISA-E_n-F

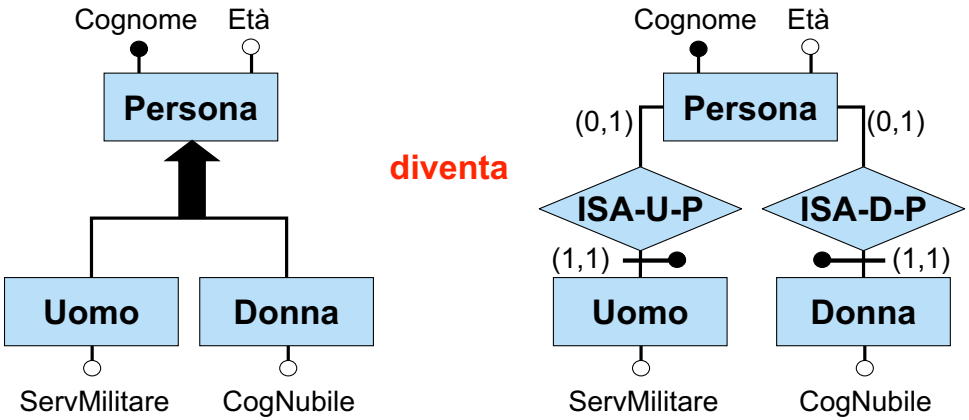
Eliminazione di generalizzazioni tra entità: esempio



Vincoli di generalizzazione:

nessuna istanza di Persona partecipa sia a ISA-S-P sia a ISA-D-P

Eliminazione di generalizzazioni complete: esempio



Vincoli di generalizzazione:

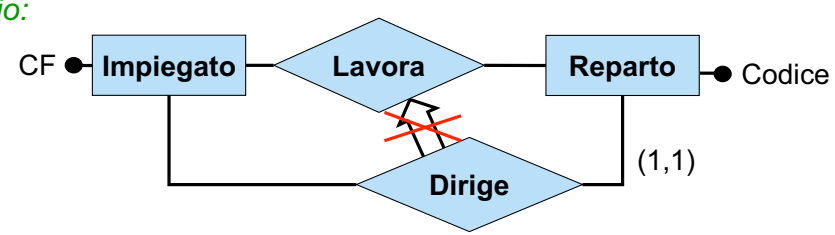
ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

Ristrutturazione – fase 4: eliminazione di ISA e generalizzazioni tra relazioni

Le **relazioni ISA e le generalizzazioni tra relazioni** vengono eliminate dallo schema e vengono espresse tramite opportuni vincoli esterni.

Nel caso in cui le relazioni in ISA (o nella generalizzazione) insistano su esattamente le **stesse entità per tutti i ruoli**, è immediato esprimere il vincolo esterno.

Esempio:

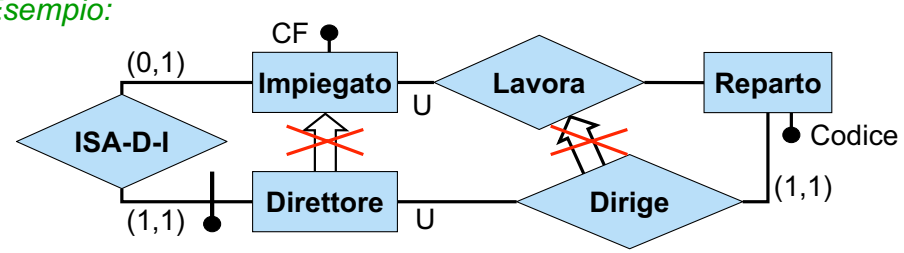


Vincolo esterno: ogni istanza di Dirige è anche un'istanza di Lavora.

Eliminazione di ISA e generalizzazioni tra relazioni

Nel caso in cui le relazioni in ISA (o nella generalizzazione) insistano su **entità diverse in qualche ruolo**, nell'esprimere il vincolo esterno bisogna tenere conto che nello schema ristrutturato entità diverse sono tra loro disgiunte.

Esempio:



Vincolo esterno: per ogni istanza (U:*d*,Reparto:*r*) di Dirige, sia *i* l'istanza di Impiegato tale che (Direttore:*d*,Impiegato:*i*) è un'istanza di ISA-D-I (si noti che *i* esiste sempre ed è unica); allora (U:*i*,Reparto:*r*) deve essere un'istanza di Lavora.

Ristrutturazione – fase 5: scelta degli identificatori principali

Per **ogni entità** è necessario:

- individuare almeno un identificatore
- scegliere tra gli identificatori dell'entità un **identificatore principale**.

Criteri per la scelta dell'identificatore principale:

- semplicità (cioè formato da pochi attributi)
- preferenza per gli identificatori interni
- utilizzo nelle operazioni più frequenti o importanti
- se per un'entità nessuno degli identificatori soddisfa tali requisiti, è possibile introdurre un ulteriore attributo dell'entità (un **codice**, i cui valori sono speciali ed hanno l'unico scopo di identificare le istanze dell'entità).

In una entità con più identificatori, quello principale viene indicato nella documentazione associata allo schema ristrutturato. Sulle slide, in presenza di più identificatori per un'entità, denoteremo quello principale con un cerchio addizionale.

Cicli di identificazione esterna

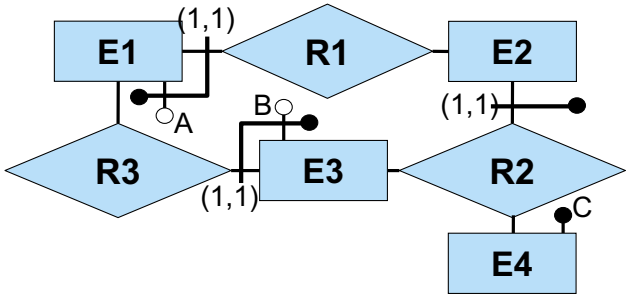
Nella scelta degli identificatori principali è necessario fare attenzione a non introdurre **cicli tra gli identificatori principali esterni**.

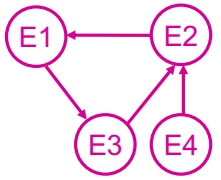
Definiamo il grafo degli identificatori (principali) esterni nel seguente modo:

- ad ogni entità del diagramma corrisponde un nodo
- c'è un arco dall'entità E all'entità F se e solo se E partecipa ad una relazione che è parte dell'identificatore (o che è identificatore) principale esterno di F.

Si ha un ciclo di identificazione esterna quando il grafo degli identificatori principali esterni contiene un ciclo.

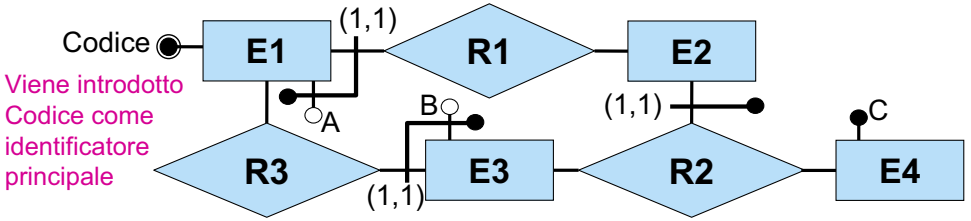
Cicli di identificazione esterna





grafo degli identificatori principali esterni

È necessario **spezzare i cicli di identificazione esterna** scegliendo per almeno una entità nel ciclo un identificatore principale diverso. Se non ci sono alternative, è necessario introdurre un opportuno codice.

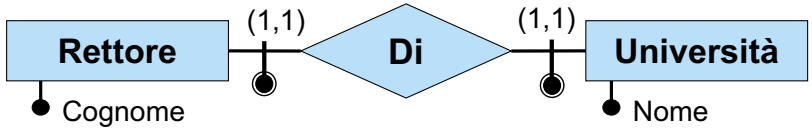


Viene introdotto Codice come identificatore principale

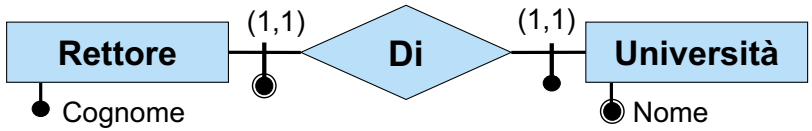
Cicli di identificazione esterna: esempio

Un caso significativo di ciclo di identificazione esterna è dato da due entità che si identificano a vicenda.

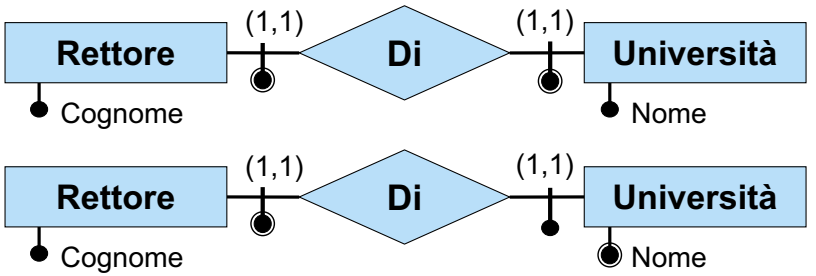
Esempio:



Abbiamo un ciclo di identificazione esterna, che deve essere spezzato. Una possibilità è la seguente:

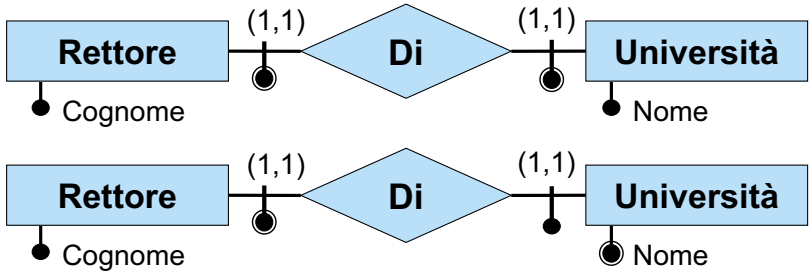


Esercizio 2: cicli di identificazione esterna



- Che differenza c'è tra le istanze dei due schemi?
- Perché è necessario spezzare i cicli di identificazione esterna?

Esercizio 2: soluzione



I due schemi hanno le stesse istanze, in quanto la scelta degli identificatori principali non ha portato all'introduzione di nuovi attributi.

I cicli di identificazione esterna non rappresentano alcun problema per quanto riguarda lo schema concettuale.

È però necessario spezzarli perché renderebbero impossibile la traduzione nel modello relazionale. Vedremo infatti che, se un'entità E ha un identificatore principale esterno su una relazione R, nello schema relazionale prodotto, ad E corrisponderà una relazione in cui viene accorpata la relazione R, con tutti gli identificatori delle entità che partecipano ad R. In presenza di cicli di identificazione esterna questo accorpamento non è fattibile.

Ristrutturazione – fase 6: specifica degli ulteriori vincoli esterni

- È necessario riformulare tutti i vincoli esterni dello schema originario in termini dello schema ristrutturato.
  - mettere in evidenza anche i vincoli impliciti (ovvero che sono conseguenza di altri vincoli)

    Es. vincoli di identificazione esterna per relazioni (1,1) – (1,1) oppure (1,1) – (0,1)
- Si devono aggiungere i vincoli derivanti dalla ristrutturazione:
  - vincoli derivanti da attributi composti opzionali
  - vincoli per due entità che erano in ISA con una stessa entità padre e che hanno attributi in comune
  - vincoli di generalizzazione (disgiuntezza e completezza)
  - vincoli dovuti agli identificatori non principali (se non sono più rappresentati nello schema)

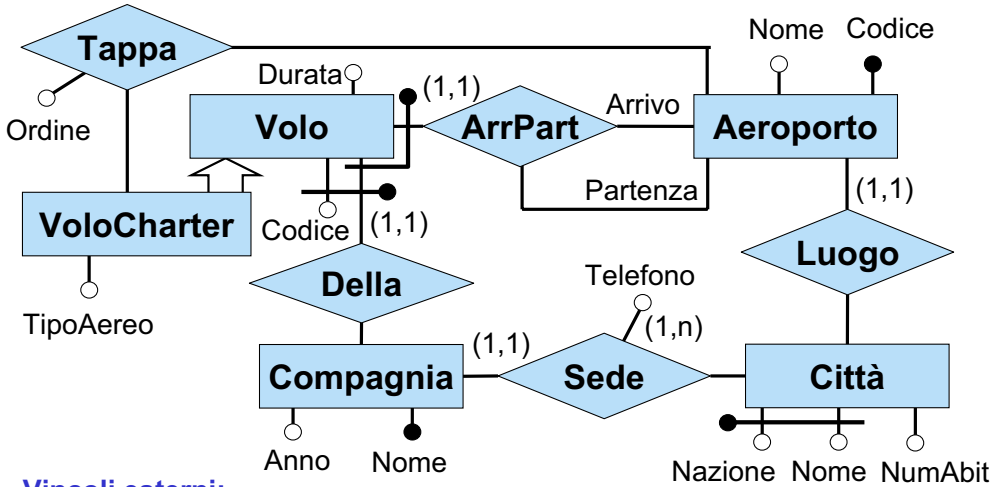
Ristrutturazione – fase 7: riformulazione di operazioni e carico applicativo

- È necessario riformulare le operazioni e i relativi schemi di navigazione in termini dello schema ristrutturato.
- È necessario riformulare le specifiche sul carico applicativo in termini dello schema ristrutturato.

Riassunto sulla ristrutturazione

1. Analisi delle ridondanze (si tiene conto dell'efficienza)
2. Eliminazione degli attributi multivalore
3. Eliminazione degli attributi composti (eventuale vincolo (0,1) diventa un vincolo esterno sugli attributi componenti)
4. Eliminazione delle ISA e delle generalizzazioni
   - vincoli per entità figlie della stessa entità padre con uno stesso attributo
   - vincoli di generalizzazione (disgiuntezza e completezza)
   - si noti che tutte le entità diventano disgiunte
5. Scelta degli identificatori principali
   - tutte le entità devono avere un identificatore (eventualmente, introdurre codice)
   - eliminazione di cicli di identificatori principali esterni
6. Specifica degli ulteriori vincoli esterni
   - vincoli derivanti dalla ristrutturazione
   - riformulazione dei vincoli esterni dello schema originario
7. Riformulazione delle operazioni e delle specifiche sul carico applicativo in termini dello schema ristrutturato

Esercizio 3: ristrutturare il seguente schema

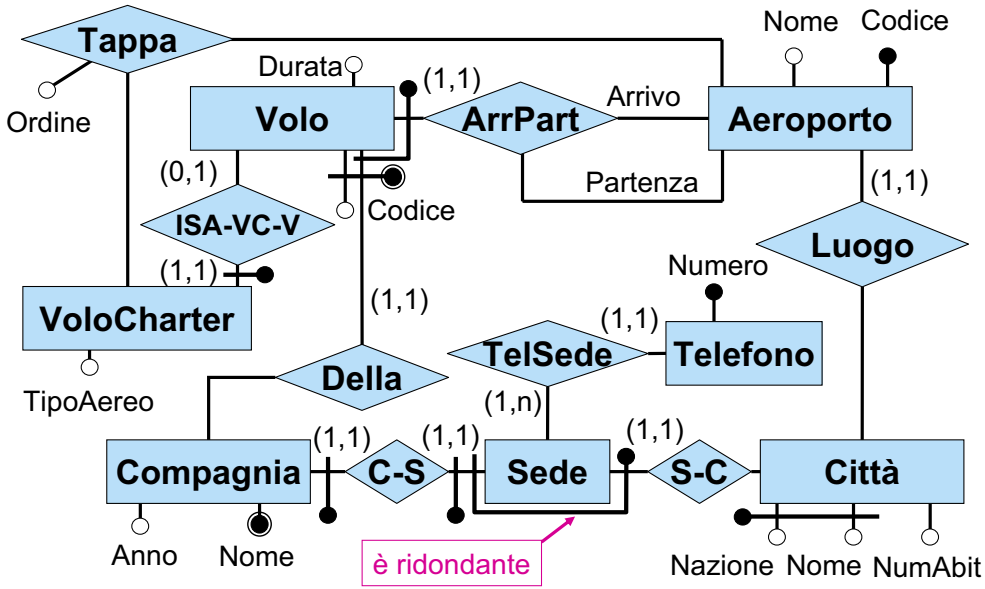


Vincoli esterni:

1) per ogni v in VoloCharter, se $(v,a_1),...,(v,a_n)$ sono tutte le coppie in Tappa alle quali partecipa v, e se $o_1,...,o_n$ sono i valori assegnati a tali coppie dall'attributo Ordine, allora per i=1,...,n esiste un o_j tale che o_j=i.

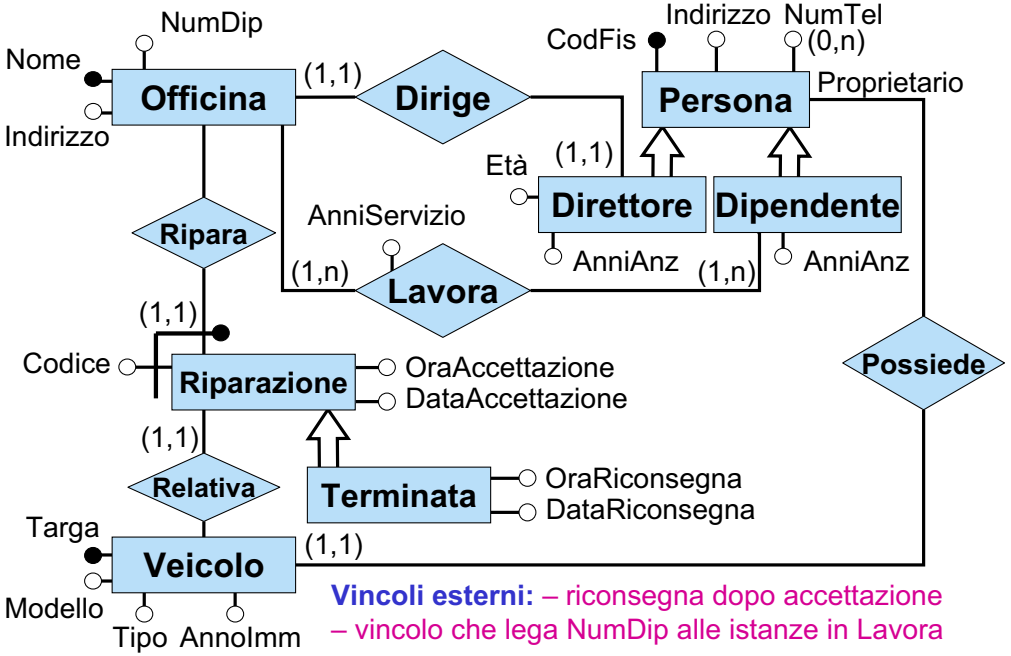
2) Un telefono è di una sola sede.

Esercizio 3: soluzione



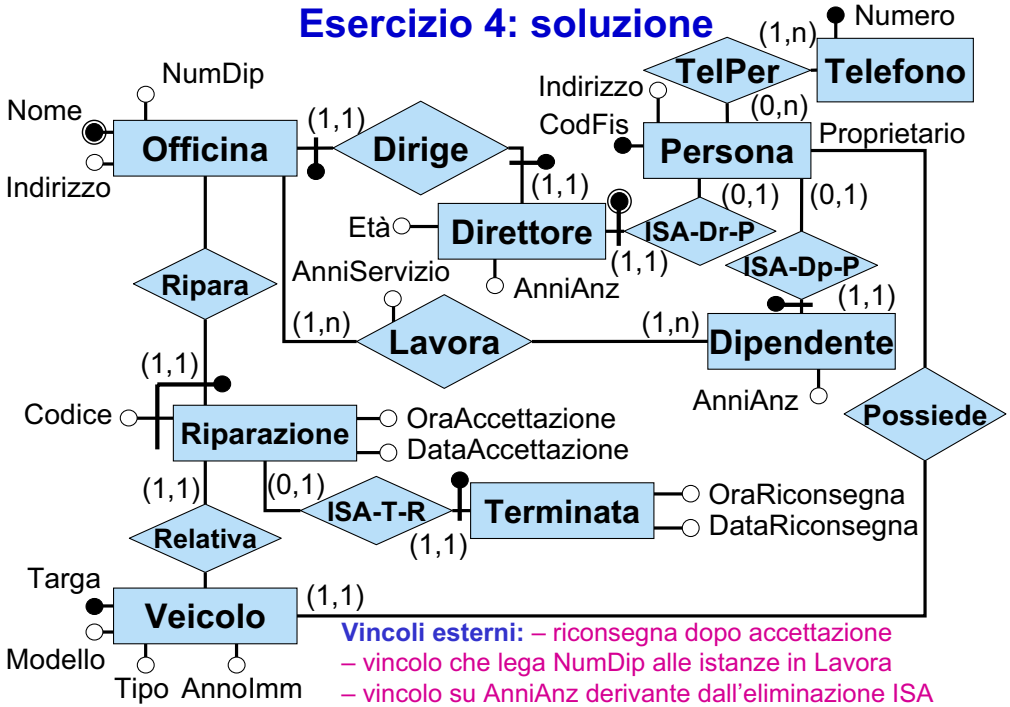
Vincolo esterno: 1) vincolo su Ordine in Tappa (2 è diventato interno allo schema)

Esercizio 4: ristrutturare il seguente schema



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora

Esercizio 4: soluzione



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora
– vincolo su AnniAnz derivante dall'eliminazione ISA

5. La progettazione logica

5.3 traduzione diretta nel modello relazionale

1. introduzione alla progettazione logica
2. ristrutturazione dello schema ER
3. **traduzione diretta nel modello relazionale**
4. ristrutturazione dello schema logico

Proprietà dello schema ristrutturato

La fase di ristrutturazione ha prodotto uno schema ER ristrutturato con le seguenti proprietà:

- preserva la semantica dello schema originale. Intuitivamente, esiste una funzione che associa ad ogni istanza dello schema originale un'opportuna istanza dello schema ristrutturato e viceversa
- può contenere delle ridondanze, ma sono volute per motivi di efficienza e sono comunque documentate
- non contiene attributi multivalore
- non contiene attributi composti
- non contiene ISA o generalizzazione (nè tra entità, nè tra relazioni); quindi tutte le entità sono disgiunte a coppie
- tutte le entità hanno un unico identificatore principale

Lo schema ristrutturato è il punto di partenza per la traduzione nel modello relazionale.

Traduzione diretta

- La traduzione diretta ha lo scopo di tradurre lo schema ER ristrutturato (con vincoli) in uno schema relazionale con vincoli che rappresenti le stesse informazioni
- Non richiede di effettuare scelte (tranne in un caso), in quanto si basa sulle scelte fatte in fase di ristrutturazione
- Produce uno schema logico di massima, che può essere accettabile, ma che può richiedere successive ristrutturazioni
- Consiste delle seguenti **attività**:
  - traduzione delle **entità** in relazioni dello schema logico, con relativi vincoli
  - traduzione delle **relazioni** dello schema ER in relazioni dello schema logico, con relativi vincoli
  - traduzione dei **vincoli esterni**
  - riformulazione di **operazioni** e **specifiche** sul carico applicativo in termini dello schema logico

Per distinguire tra le due accezioni di relazione, useremo il termine ER-relazione per denotare le relazioni dello schema ER

Notazione

- Nella fase di traduzione diretta ed in quella di ristrutturazione dello schema logico, esprimeremo gli schemi relazionali mediante una notazione che prevede di descrivere le relazioni con nome e attributi, ed i vincoli ad esse associati in forma testuale

- Esempio:

chiave primaria — attributi non null

Partecipa(Cognome, DataN, Progetto, OreSett, Iva*)
foreign key: Partecipa[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]
inclusione: Partecipa[OreSett] $\subseteq$ Orario[Ore]
chiave: Progetto

- Ovviamente, questa notazione si può tradurre senza difficoltà in termini di “create table” in SQL. Si lascia allo studente la verifica di come effettuare la traduzione in SQL

Traduzione di entità: regole generali

- Ogni entità **E** dello schema ER viene tradotta in una relazione R_E dello schema relazionale
- Gli **attributi** della relazione R_E sono:
  - gli attributi dell’entità **E** (tutti not null, tranne quelli opzionali)
  - gli attributi derivanti dall’accorpamento di ER-relazioni in R_E – per ogni ER-relazione **Q** accorpata in R_E vengono aggiunti ad R_E come attributi:
    - gli attributi della ER-relazione **Q**
    - le chiavi primarie delle relazioni che corrispondono alle altre entità che partecipano a **Q**
- Una relazione **Q** viene **accorpata** in R_E quando uno o più ruoli di **Q** partecipano all’identificatore principale (esterno) di **E** (si noti che in questo caso **E** partecipa a **Q** con cardinalità (1,1), ed inoltre **E** è l’unica entità per cui un ruolo di **Q** partecipa all’identificatore principale esterno, altrimenti ci sarebbe un ciclo di identificazione principale esterna)
- La **chiave primaria** di R_E è determinata in base all’identificatore principale di **E** (attributi di **E** e/o derivanti dall’identificazione esterna)
- Agli altri identificatori di **E** corrispondono dei **vincoli di chiave** su R_E
- A seconda dei casi, possono essere necessari ulteriori vincoli

Traduzione di ER-relazioni: regole generali

- Ogni relazione **Q** dello schema ER che non è stata accorpata al passo precedente viene tradotta in una relazione R_Q dello schema relazionale.
- Gli **attributi** della relazione R_Q sono:
  - gli attributi della ER-relazione **Q**
  - le chiavi primarie delle entità che partecipano alla ER-relazione **Q**
- Scelta della **chiave primaria** di R_Q:
  - Se nessuna entità partecipa con cardinalità massima 1 a **Q**, allora la chiave primaria di R_Q è costituita dalla combinazione delle chiavi primarie delle entità partecipanti
  - Altrimenti, la chiave primaria di ogni entità che partecipa con cardinalità massima 1 a **Q** è chiave di R_Q, e la chiave primaria di R_Q va scelta tra queste chiavi candidate
- Le chiavi candidate rimanenti divengono **vincoli di chiave** su R_Q
- Le tipizzazioni delle componenti di **Q**, per ogni ruolo, con le entità partecipanti divengono in R_Q vincoli di **foreign key** verso le relazioni che corrispondono alle entità partecipanti
- A seconda dei casi, possono essere necessari ulteriori vincoli

Traduzione di vincoli: regole generali

Questi sono i vincoli da considerare:

- Vincoli **not null** per gli attributi obbligatori
- Vincoli di interdipendenza di valori nulli (provenienti da attributi composti opzionali), formulati come vincoli di tupla
- Vincoli di **chiave** (primarie e non)
- Vincoli di **foreign key** che provengono
  - dalla tipizzazione di relazioni (incluse quelle che sono state accorpate in entità)
  - dai vincoli esterni derivanti dall'ISA di relazioni
- Vincoli **di generalizzazione**, formulati come vincoli insiemistici
- Vincoli di cardinalità:
  - partecipazione obbligatoria (cardinalità minima 1) diventa **vincolo di inclusione** o **foreign key** dalla relazione che corrisponde all'entità verso quella che corrisponde alla ER-relazione
  - funzionalità (cardinalità massima 1) diventa **vincolo di chiave** sulla relazione che corrisponde alla ER-relazione
  - gli altri vincoli di cardinalità diventano vincoli esterni
- Gli altri vincoli esterni vanno opportunamente tradotti

Riformulazione di operazioni e carico applicativo: regole generali

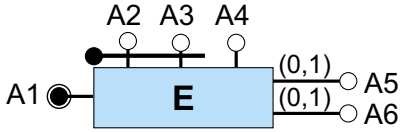
- Le operazioni e le informazioni sul carico applicativo sono state espresse all'inizio della progettazione logica sulla base dello schema concettuale, e poi modificate per renderle coerenti con lo schema concettuale ristrutturato. È ora necessario riformulare le operazioni e le informazioni sul carico applicativo in modo che siano coerenti con lo schema logico

- La riformulazione viene condotta semplicemente tenendo presente come le entità e le relazioni dello schema Entità-Relazione ristrutturato sono state tradotte nello schema relazionale

Traduzione di entità senza accorpamento

Consideriamo per ora **un'entità per cui non si effettua accorpamento di relazioni** (in particolare, un'entità che non ha identificatori esterni).

- L'entità si traduce in una **relazione** dello schema relazionale
- Gli **attributi** della relazione corrispondente all'entità sono quelli dell'entità.
  - se un attributo è opzionale diventa un attributo della relazione che può assumere valore nullo (nella nostra notazione per lo schema logico, tali attributi sono indicati con *)
  - altrimenti l'attributo non può assumere valore nullo
- L'identificatore principale dell'entità si traduce nella **chiave primaria** della relazione
- Gli altri identificatori interni si traducono in **chiavi** della relazione
- Ricordarsi dei vincoli esterni per identificatori opzionali correlati (derivanti da attributi composti opzionali)



Vincoli esterni:

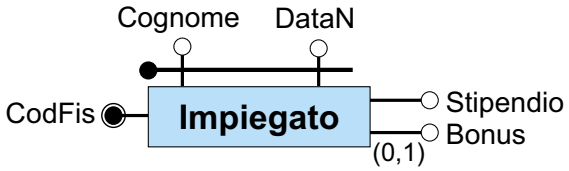
- *per ogni istanza di E, A5 è definito se e solo se lo è anche A6*

E(A1, A2, A3, A4, A5*, A6*)

chiave: A2, A3

vincolo: A5 è NULL se e solo se A6 è NULL

Traduzione di entità senza accorpamento: esempio



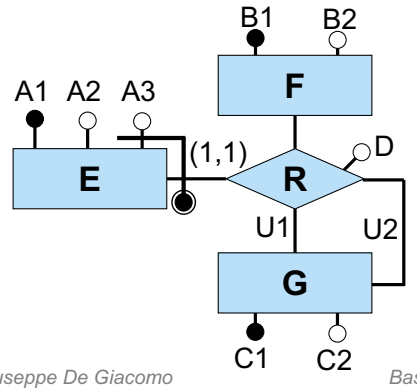
Impiegato(CodFis, Cognome, DataN, Stipendio, Bonus*)

chiave: Cognome, DataN

Traduzione di entità con accorpamento: caso 1

Consideriamo il caso in cui una ER-relazione **R** è **parte dell'identificatore principale esterno di un'entità E con ruolo U**, e in tutti gli altri ruoli di **R** la cardinalità è (0,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E**. Questo significa che tutti gli attributi della ER-relazione e le chiavi primarie delle altre entità partecipanti diventano attributi della relazione RE che corrisponde all'entità **E**. Tali chiavi primarie fanno parte della chiave primaria della relazione RE.
- Si noti che eventuali altri identificatori dell'entità in cui la relazione è stata accorpata si traducono in vincoli di chiave sull'entità.

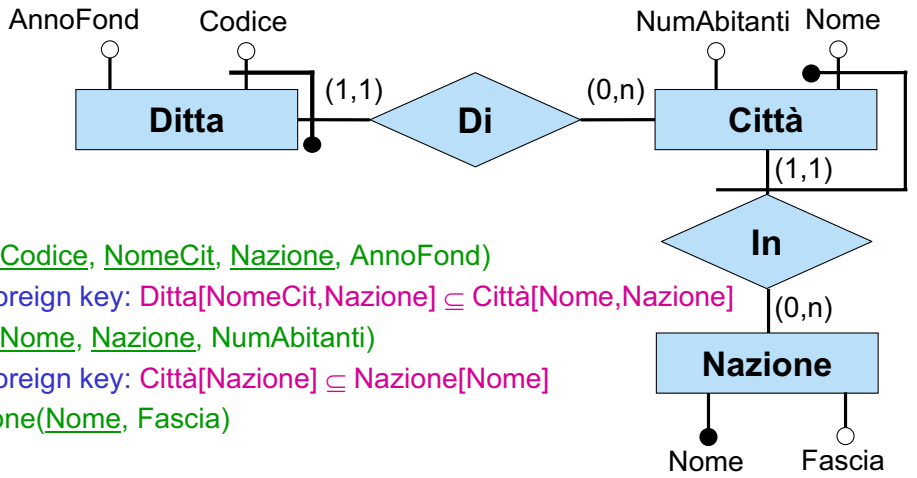


E(A1, A2, A3, F, U1, U2, D)
  foreign key: E[F] ⊆ F[B1]
  foreign key: E[U1] ⊆ G[C1]
  foreign key: E[U2] ⊆ G[C1]
  chiave: A1

F(B1, B2)

G(C1, C2)

Traduzione di entità con accorpamento - caso 1: esempio

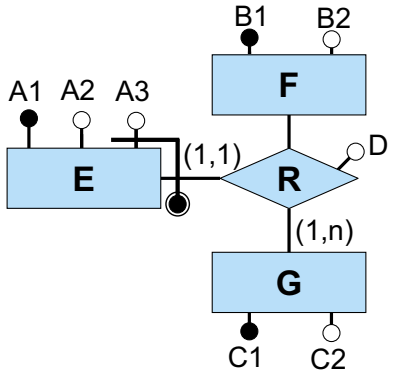


Ditta(Codice, NomeCit, Nazione, AnnoFond)
  foreign key: Ditta[NomeCit,Nazione] ⊆ Città[Nome,Nazione]
Città(Nome, Nazione, NumAbitanti)
  foreign key: Città[Nazione] ⊆ Nazione[Nome]
Nazione(Nome, Fascia)

Traduzione di entità con accorpamento: caso 2

Consideriamo il caso in cui una ER-relazione **R** è **parte dell'identificatore principale esterno di un'entità E con ruolo U**, ed in alcuni degli altri ruoli di **R** la cardinalità è (1,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E** in modo analogo al caso 1
- Ma in questo caso, per ogni ruolo **U** con cardinalità (1,n), si aggiunge un vincolo di inclusione tra la chiave primaria dell'entità corrispondente al ruolo **U** e l'attributo (o gli attributi, se la chiave è composta) di **E** corrispondente ad **U**

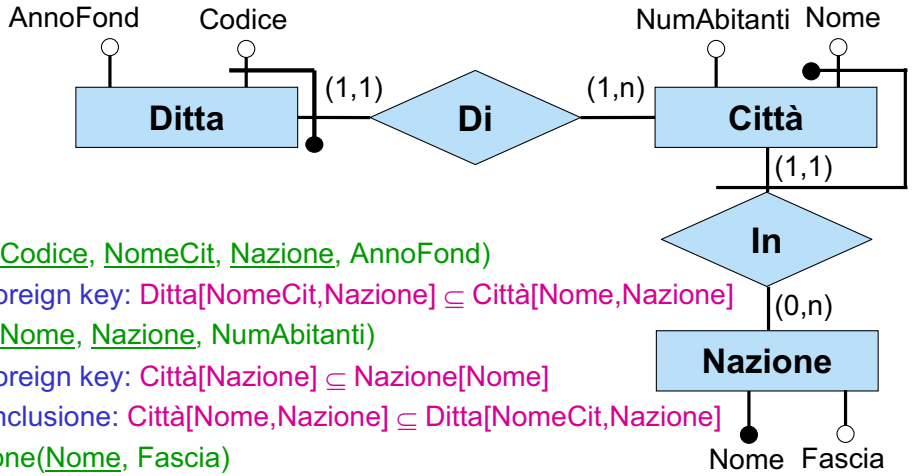


E(A1, A2, A3, F, G, D)
  foreign key: E[F] ⊆ F[B1]
  foreign key: E[G] ⊆ G[C1]
  chiave: A1

F(B1, B2)

G(C1, C2)
  inclusione: G[C1] ⊆ E[G]

Traduzione di entità con accorpamento - caso 2: esempio

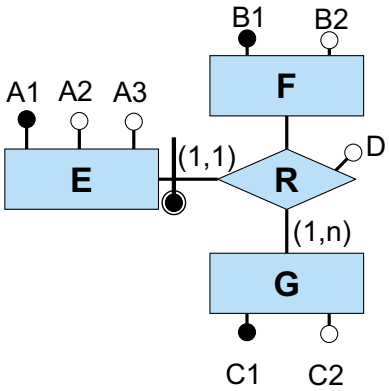


Ditta(Codice, NomeCit, Nazione, AnnoFond)
  foreign key: Ditta[NomeCit,Nazione] ⊆ Città[Nome,Nazione]
Città(Nome, Nazione, NumAbitanti)
  foreign key: Città[Nazione] ⊆ Nazione[Nome]
  inclusione: Città[Nome,Nazione] ⊆ Ditta[NomeCit,Nazione]
Nazione(Nome, Fascia)

Traduzione di entità con accorpamento: caso 3

Consideriamo il caso in cui una ER-relazione **R** è **identificatore principale esterno** (non parte di identificatore) **di un'entità E con ruolo U**, ed in alcuni degli altri ruoli di **R** la cardinalità è (1,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E** in modo analogo ai casi 1 e 2
- Ma in questo caso, la chiave primaria di **E** non comprende attributi di **E**

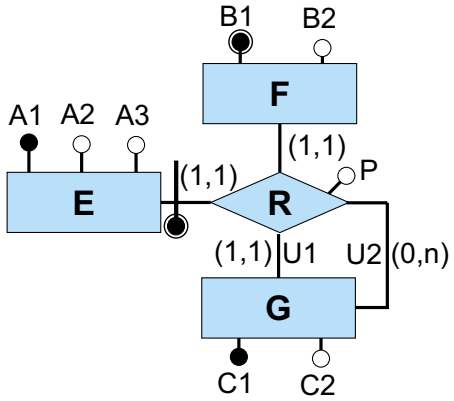


E(A1, A2, A3, F, G, D)
foreign key: E[F] ⊆ F[B1]
foreign key: E[G] ⊆ G[C1]
chiave: A1
F(B1, B2)
G(C1, C2)
inclusione: G[C1] ⊆ E[G]

Traduzione di entità con accorpamento: caso 4

Consideriamo il caso in cui una ER-relazione **R** è **identificatore principale esterno di un'entità E con ruolo U**, e in alcuni degli altri ruoli di **R** la cardinalità massima è 1. Si noti che in questo caso **R** è l'identificatore principale (non parte di identificatore).

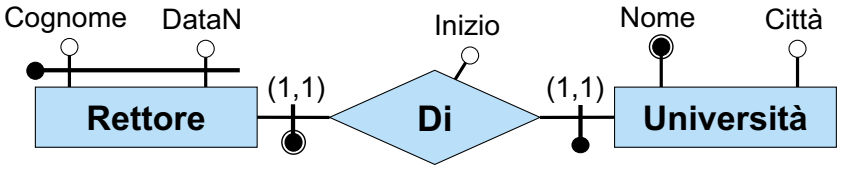
- La ER-relazione **R** viene **accorpata** nell'entità **E** in modo analogo ai casi 1 e 2, ma in questo caso la chiave primaria della relazione che rappresenta **E** deve essere scelta tra le chiavi primarie delle altre entità coinvolte in **R** i cui ruoli hanno cardinalità massima 1



Si sceglie la chiave primaria di F come parte di chiave primaria di E:

E(A1, A2, A3, F, U1, U2, P)
foreign key: E[F] ⊆ F[B1]
foreign key: E[U1] ⊆ G[C1]
foreign key: E[U2] ⊆ G[C1]
chiave: A1
chiave: U1
F(B1, B2)
foreign key: F[B1] ⊆ E[F]
G(C1, C2)
foreign key: G[C1] ⊆ E[U1]

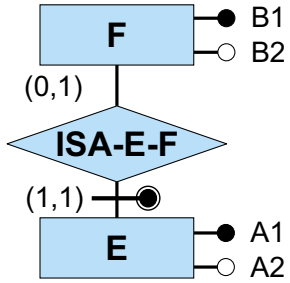
Traduzione di entità con accorpamento - caso 4: esempio



Rettore(Cognome, DataN, Inizio, Università)
foreign key: Rettore[Università] ⊆ Università[Nome]
chiave: Cognome, DataN
Università(Nome, Città)
foreign key: Università[Nome] ⊆ Rettore[Università]

Accorpamento di relazione derivante da ISA

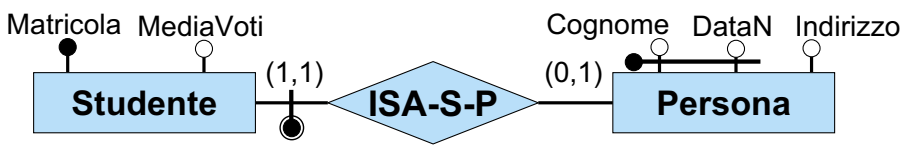
Un caso di ER-relazione che è identificatore principale esterno di un'entità può essere quello derivante dalla ristrutturazione di un'ISA nello schema ER originale.



E(A1, A2, B1)
foreign key: E[B1] ⊆ F[B1]
chiave: A1
F(B1, B2)

Si noti come la traduzione della parte di schema ER che si ottiene dalla ristrutturazione di **E ISA F** corrisponda ad aggiungere agli attributi di **E** la chiave primaria di **F**, ed a rendere tali attributi anche chiave primaria di **E**. Il vincolo derivante dall'ISA dello schema ER originario diventa quindi un vincolo di foreign key dello schema logico.

Accorpamento di relazione derivante da ISA: esempio



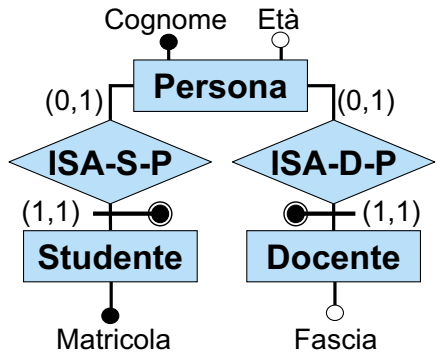
Studente(Cognome, DataN, Matricola, MediaVoti)

foreign key: Studente[Cognome,DataN] ⊆ Persona[Cognome,DataN]

chiave: Matricola

Persona(Cognome, DataN, Indirizzo)

Accorpamento di relazione derivante da generalizzazione



Vincolo di generalizzazione:

nessuna istanza di Persona partecipa sia a ISA-S-P sia a ISA-D-P

Diventa sullo schema logico:

Studente[Cognome] ∩ Docente[Cognome] = ∅

Persona(Cognome, Età)

Studente(Cognome, Matricola)

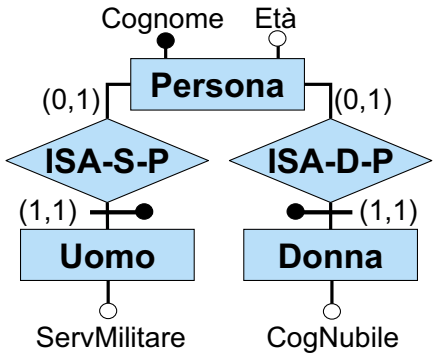
foreign key: Studente[Cognome] ⊆ Persona[Cognome]

chiave: Matricola

Docente(Cognome, Fascia)

foreign key: Docente[Cognome] ⊆ Persona[Cognome]

Accorpamento di relazione derivante da generalizzazione



Vincolo di generalizzazione:

ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

Diventa sullo schema logico:

Uomo[Cognome] ∩ Donna[Cognome] = ∅

Persona[Cognome] ⊆ Uomo[Cognome] ∪ Donna[Cognome]

Persona(Cognome, Età)

Uomo(Cognome, ServMilitare)

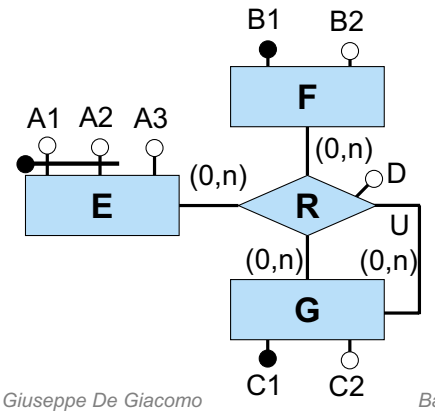
foreign key: Uomo[Cognome] ⊆ Persona[Cognome]

Donna(Cognome, CogNubile)

foreign key: Donna[Cognome] ⊆ Persona[Cognome]

Traduzione di relazione

- Consideriamo il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui tutte le cardinalità siano di tipo (0,n)**
- La ER-relazione si traduce in una relazione
- Gli **attributi** della relazione sono quelli della ER-relazione, più le chiavi primarie delle relazioni corrispondenti alle entità partecipanti (per ogni ruolo)
- Poichè nessuna cardinalità massima è 1, la **chiave primaria** della relazione è data dalle chiavi primarie delle entità partecipanti
- Si definiscono vincoli di foreign key dalla relazione verso le entità partecipanti per dar conto dei vincoli di tipizzazione nello schema ER



E(A1, A2, A3)

F(B1, B2)

G(C1, C2)

R(EA1, EA2, F, G, U, D)

foreign key: R[EA1,EA2] ⊆ E[A1,A2]

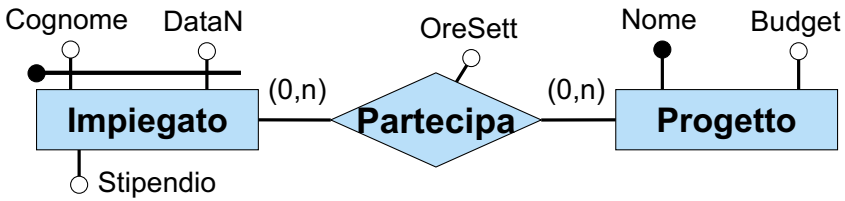
foreign key: R[F] ⊆ F[B1]

foreign key: R[G] ⊆ G[C1]

foreign key: R[U] ⊆ G[C1]

Traduzione di relazione: esempio

Nello scegliere per una relazione il nome di un attributo che rappresenta la chiave primaria di un'entità che partecipa alla relazione, può essere opportuno utilizzare il nome del ruolo con cui l'entità partecipa alla relazione (invece del nome che l'attributo ha per l'entità).



Impiegato(Cognome, DataN, Stipendio)

Progetto(Nome, Budget)

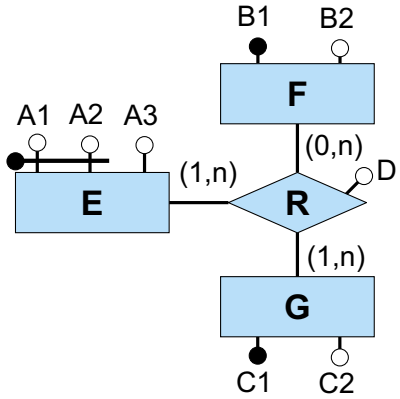
Partecipa(Cognome, DataN, Progetto, OreSett)

foreign key: Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]

foreign key: Partecipa[Progetto] ⊆ Progetto[Nome]

Traduzione di relazione con cardinalità minima 1

- Consideriamo il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui tutte le cardinalità massime siano n**
- Un vincolo di cardinalità minima 1 per la partecipazione di un'entità (in un ruolo) alla relazione si traduce in un **vincolo di inclusione** dall'entità verso l'attributo (o gli attributi) corrispondenti a quel ruolo nella relazione.
- Il vincolo di inclusione non è in generale di foreign key (si vedano in seguito i casi in cui l'inclusione diventa in realtà una foreign key).



E(A1, A2, A3)

inclusione: E[A1,A2] ⊆ R[A1,A2]

F(B1, B2)

G(C1, C2)

inclusione: G[C1] ⊆ R[C1]

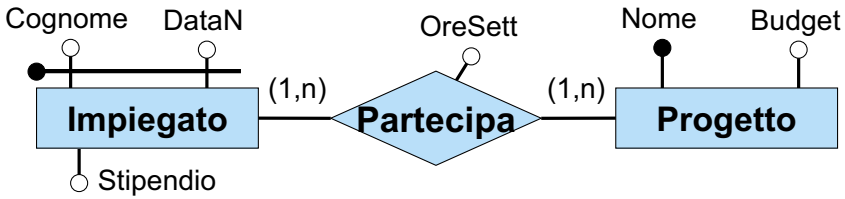
R(A1, A2, B1, C1, D)

foreign key: R[A1,A2] ⊆ E[A1,A2]

foreign key: R[B1] ⊆ F[B1]

foreign key: R[C1] ⊆ G[C1]

Traduzione di relazione con cardinalità minima 1: esempio



Impiegato(Cognome, DataN, Stipendio)

inclusione: Impiegato[Cognome,DataN] ⊆ Partecipa[Cognome,DataN]

Progetto(Nome, Budget)

inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

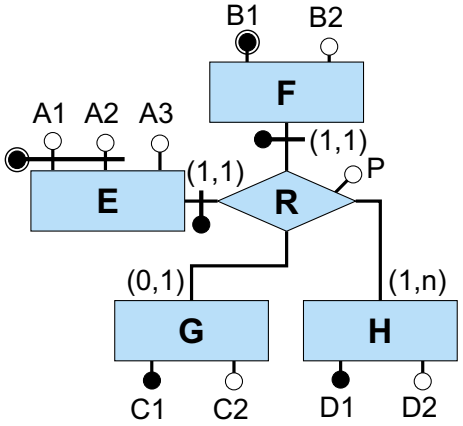
Partecipa(Cognome, DataN, Progetto, OreSett)

foreign key: Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]

foreign key: Partecipa[Progetto] ⊆ Progetto[Nome]

Traduzione di relazione con cardinalità massima 1

- Consideriamo il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui una cardinalità massime sia 1**
- La chiave primaria dell'entità diventa una **chiave della relazione**. Si noti che, se l'entità ha anche cardinalità minima 1, il vincolo di inclusione corrispondente è in realtà un vincolo di foreign key.
- Se vi è più di una di tali entità, **bisogna scegliere la chiave primaria** della relazione tra le chiavi primarie di tali entità. Le chiavi primarie delle entità diverse da quella scelta si traducono in vincoli di chiave per la relazione.



E(A1, A2, A3)

foreign key: E[A1,A2] ⊆ R[EA1,EA2]

F(B1, B2)

foreign key: F[B1] ⊆ R[F]

G(C1, C2)

H(D1, D2)

inclusione: H[D1] ⊆ R[H]

R(EA1, EA2, F, G, H, P)

foreign key: R[EA1,EA2] ⊆ E[A1,A2]

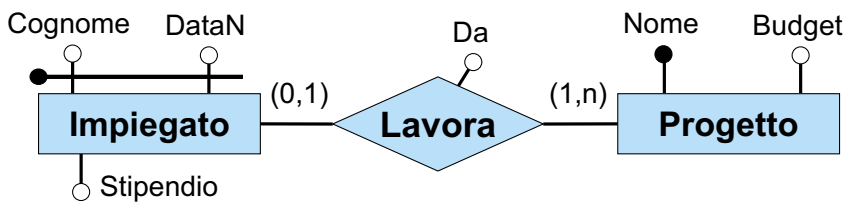
foreign key: R[F] ⊆ F[B1]

foreign key: R[G] ⊆ G[C1]

foreign key: R[H] ⊆ H[D1]

chiave: EA1, EA2 chiave: F

Traduzione di relazione con cardinalità massima 1: esempio



Impiegato(Cognome, DataN, Stipendio)

Progetto(Nome, Budget)

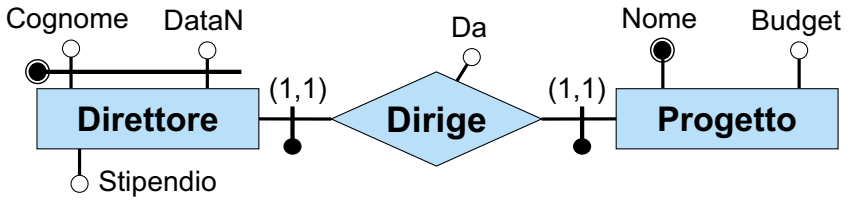
inclusione: Progetto[Nome] ⊆ Lavora[Progetto]

Lavora(Cognome, DataN, Progetto, Da)

foreign key: Lavora[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]

foreign key: Lavora[Progetto] ⊆ Progetto[Nome]

Traduzione di relazione con cardinalità (1,1): esempio



Direttore(Cognome, DataN, Stipendio)

foreign key: Direttore[Cognome,DataN] ⊆ Dirige[Cognome,DataN]

Progetto(Nome, Budget)

foreign key: Progetto[Nome] ⊆ Dirige[Progetto]

Dirige(Cognome, DataN, Progetto, Da)

chiave: Cognome, DataN

foreign key: Dirige[Cognome,DataN] ⊆ Direttore[Cognome,DataN]

foreign key: Dirige[Progetto] ⊆ Progetto[Nome]

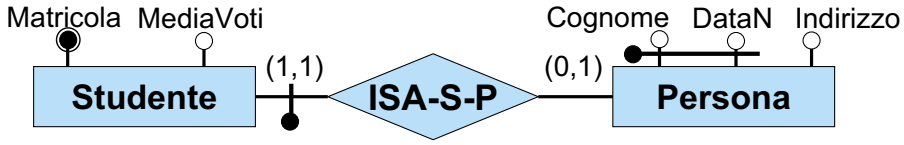
In alternativa:

Dirige(Cognome, DataN, Progetto, Da)

chiave: Progetto foreign key: ...

Traduzione di relazione derivante da ISA: esempio

Una relazione derivante da ISA che non è stata accorpata (perché l'identificatore principale non è quello esterno) si traduce nel modo mostrato in questo esempio.



ISA-S-P(Matricola, Cognome, DataN)

foreign key: ISA-S-P[Cognome,DataN] ⊆ Persona[Cognome,DataN]

foreign key: ISA-S-P[Matricola] ⊆ Studente[Matricola]

chiave: Cognome, DataN

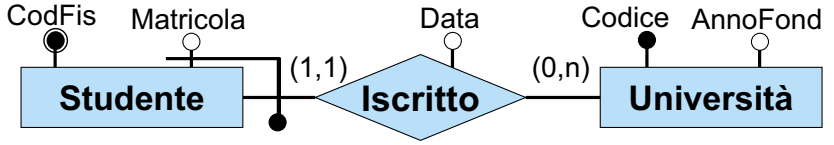
Studente(Matricola, MediaVoti)

foreign key: Studente[Matricola] ⊆ ISA-S-P[Matricola]

Persona(Cognome, DataN, Indirizzo)

Traduzione di entità con identificatore esterno non principale

Un vincolo di identificazione non principale esterna per una entità E diventa un vincolo esterno che riguarda sia l'entita E sia la relazione che partecipa all'identificazione.



Iscritto(Studente, Università, Data)

foreign key: Iscritto[Studente] ⊆ Studente[CodFis]

foreign key: Iscritto[Università] ⊆ Università [Codice]

Studente(CodFis, Matricola)

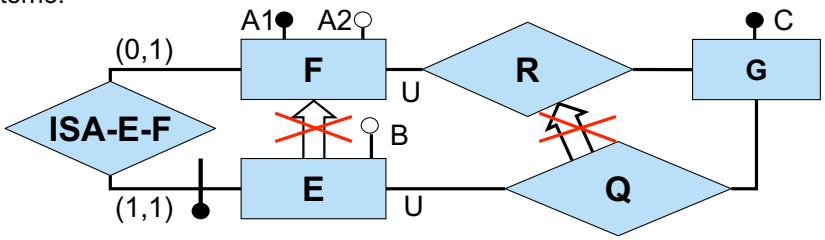
foreign key: Studente[CodFis] ⊆ Iscritto[Studente]

Università(Codice, AnnoFond)

Vincolo: nel join tra Studente e Iscritto sugli attributi CodFis e Studente, gli attributi Matricola e Codice formano una chiave

Traduzione di vincoli derivanti da ISA tra relazioni

Si ricordi che la ristrutturazione di una ISA tra relazioni ha prodotto un vincolo esterno.



Vincolo esterno: per ogni istanza (e,g) di Q, sia f l'istanza di F tale che (e,f) è un'istanza di ISA-E-F (si noti che f esiste sempre ed è unica). Allora (f,g) deve essere un'istanza di R.

Traduzione: il vincolo esterno diventa un vincolo di **foreign key**

E(A1, B) foreign key: E[A1] $\subseteq$ F[A1]

F(A1, A2)

G(C)

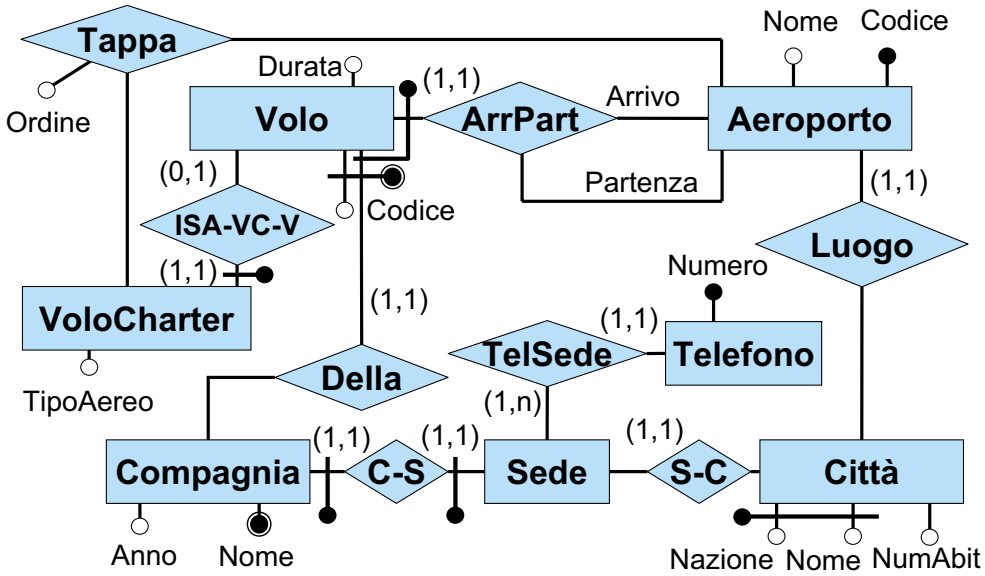
R(A1, C) foreign key: R[A1] $\subseteq$ F[A1] foreign key: R[C] $\subseteq$ G[C]

Q(A1, C) foreign key: Q[A1] $\subseteq$ E[A1] foreign key: Q[A1,C] $\subseteq$ R[A1,C]

Riassunto sulla traduzione diretta

1. Traduzione di ogni entità in una relazione, con i seguenti attributi:
   - gli attributi dell'entità stessa
   - gli attributi delle relazioni che partecipano all'identificazione principale esterna dell'entità, insieme alle chiavi primarie, opportunamente nominate (possibilmente con ruolo) delle entità connesse a tali relazioni (si noti che in questo caso, per l'assenza di cicli sull'identificazione principale esterna, la relazione non può avere altre entità per le quali la relazione è parte di identificatore esterno)
2. Traduzione di ogni ER-relazione (non accorpata al punto 1) in relazione, con opportuna chiave primaria, e con attributi:
   - gli identificatori principali delle entità partecipanti (con opportuno nome)
   - gli attributi della ER-relazione
3. Traduzione di vincoli
   - not null per gli attributi obbligatori
   - chiavi (primarie e non)
   - foreign key che provengono dall'accorpamento (vedi punto 1), da tipizzazione di relazioni, da ISA di relazioni
   - vincoli di generalizzazione
   - vincoli di cardinalità (parte obbligatoria diventa vincolo di inclusione, parte di funzionalità diventa vincolo di chiave)
   - altri vincoli esterni
4. Riformulazione di operazioni e specifiche sul carico applicativo in termini dello schema logico

Esercizio 5: tradurre il seguente schema



Vincolo esterno: vincolo su Ordine in Tappa.

Esercizio 5: soluzione (parte 1)

Volo(Codice, Comp, Durata)
- foreign key: Volo[Comp] $\subseteq$ Compagnia[Nome]
- foreign key: Volo[Codice,Comp] $\subseteq$ ArrPart[Codice,Comp]

ArrPart(Codice, Comp, Arrivo, Partenza)
- foreign key: ArrPart[Arrivo] $\subseteq$ Aeroporto[Codice]
- foreign key: ArrPart[Partenza] $\subseteq$ Aeroporto[Codice]
- foreign key: ArrPart[Codice,Comp] $\subseteq$ Volo[Codice,Comp]
- chiave: Comp, Arrivo, Partenza

Nota: viene dall'identificatore esterno non principale di Volo

VoloCharter(Codice, Comp, TipoAereo)
- foreign key: VoloCharter[Codice,Comp] $\subseteq$ Volo[Codice,Comp]

Aeroporto(Codice, Nome)
- foreign key: Aeroporto[Codice] $\subseteq$ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
- foreign key: LuogoAeroporto[Aeroporto] $\subseteq$ Aeroporto[Codice]
- foreign key: LuogoAeroporto[NomeCittà,NazCittà] $\subseteq$ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

Esercizio 5: soluzione (parte 2)

Compagnia(Nome, AnnoFond)
  foreign key: Compagnia[Nome] ⊆ Sede[Comp]

Sede(Comp)
  foreign key: Sede[Comp] ⊆ Compagnia[Nome]
  foreign key: Sede[Comp] ⊆ CittàSede[Comp]
  inclusione: Sede[Comp] ⊆ TelefonoComp[Comp]

CittàSede(Comp, NomeCittà, NazCittà)
  foreign key: CittàSede[Comp] ⊆ Sede[Comp]
  foreign key: CittàSede[NomeCittà,NazCittà] ⊆ Città[Nome,Nazione]

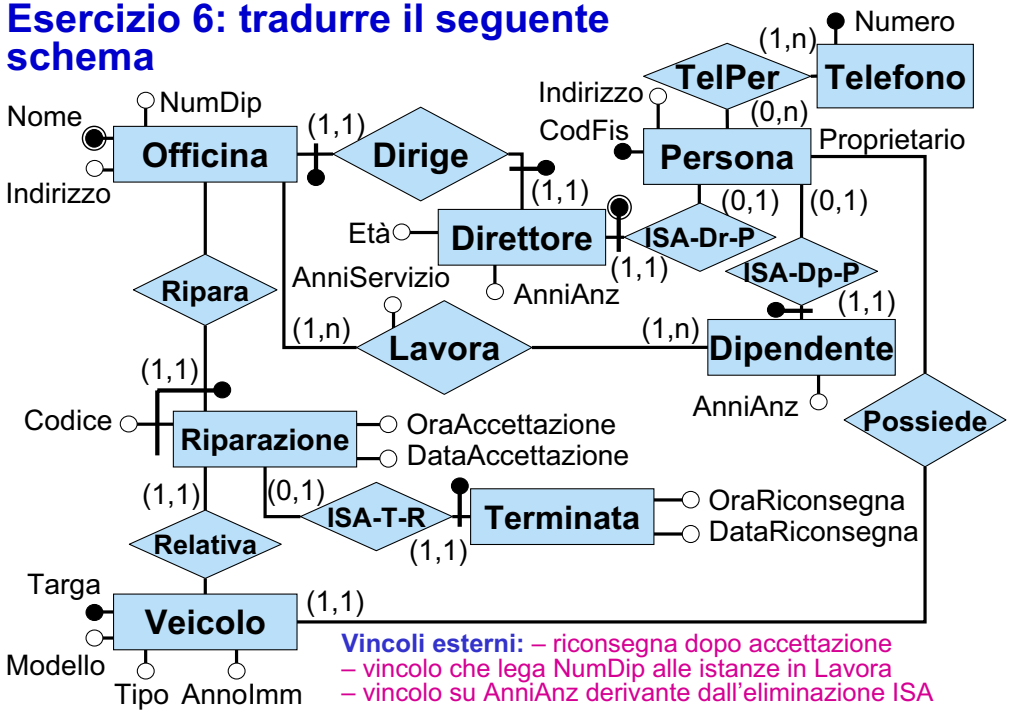
TelefonoComp(Numero, Comp)
  foreign key: TelefonoComp[Comp] ⊆ Sede[Comp]
  foreign key: TelefonoComp[Numero] ⊆ Telefono[Numero]

Telefono(Numero)
  foreign key: Telefono[Numero] ⊆ TelefonoComp[Numero]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
  foreign key: Tappa[CodVoloCharter,Comp] ⊆ VoloCharter[Codice,Comp]
  foreign key: Tappa[Aeroporto] ⊆ Aeroporto[Codice]

Vincolo esterno: vincolo su Ordine in Tappa

Esercizio 6: tradurre il seguente schema



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora
– vincolo su AnniAnz derivante dall'eliminazione ISA

Esercizio 6: soluzione (parte 1)

Officina(Nome, NumDip, Indirizzo)
  foreign key: Officina[Nome] ⊆ Dirige[Officina]
  inclusione: Officina[Nome] ⊆ Lavora[Officina]

Persona(CodFis, Indirizzo)

Direttore(CodFis, Età, AnniAnz)
  foreign key: Direttore[CodFis] ⊆ Persona[CodFis]
  foreign key: Direttore[CodFis] ⊆ Dirige[Direttore]

Dipendente(CodFis, AnniAnz)
  foreign key: Dipendente[CodFis] ⊆ Persona[CodFis]
  inclusione: Dipendente[CodFis] ⊆ Lavora[Dipendente]

Dirige(Officina, Direttore)
  foreign key: Dirige[Officina] ⊆ Officina[Nome]
  foreign key: Dirige[Direttore] ⊆ Direttore[CodFis]
  chiave: Direttore

Lavora(Officina, Dipendente, AnniServizio)
  foreign key: Lavora[Officina] ⊆ Officina[Nome]
  foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]

TelPer(CodFis, Telefono)
  foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
  foreign key: TelPer[Telefono] ⊆ Telefono[Numero]

Esercizio 6: soluzione (parte 2)

Telefono(Numero)
  inclusione: Telefono[Numero] ⊆ TelPer[Telefono]

Veicolo(Targa, Modello, Tipo, AnnoImm)
  foreign key : Veicolo[Targa] ⊆ Possiede[Veicolo]

Possiede(Veicolo, Proprietario)
  foreign key: Possiede[Veicolo] ⊆ Veicolo[Targa]
  foreign key: Possiede[Proprietario] ⊆ Persona[CodFis]

Riparazione(Codice, Officina, OraAcc, DataAcc)
  inclusione: Riparazione[Officina] ⊆ Officina[Nome]
  foreign key: Riparazione[Codice,Officina] ⊆ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
  foreign key: Relativa[Codice,Officina] ⊆ Riparazione[Codice,Officina]
  foreign key: Relativa[Veicolo] ⊆ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
  foreign key: Terminata[Codice,Officina] ⊆ Riparazione[Codice,Officina]

Vincoli esterni:
- riconsegna dopo accettazione
- vincolo che lega Officina[NumDip] alle istanze in Lavora
- vincolo su AnniAnz di Direttore e Dipendente derivante dall'eliminazione ISA

5. La progettazione logica

5.4 ristrutturazione dello schema logico

1. introduzione alla progettazione logica
2. ristrutturazione dello schema ER
3. traduzione diretta nel modello relazionale
4. **ristrutturazione dello schema logico**

Cosa sappiamo dopo la traduzione

- Abbiamo rispettato la modularizzazione concettuale
- Si possono presentare potenziali problemi di efficienza rispetto allo spazio
  - valori nulli (solo quelli dovuti ad attributi opzionali)
  - ci possono essere due relazioni R_1 e R_2 con chiavi K_1 e K_2 tali che valga sia $R_1[K_1] \subseteq R_2[K_2]$ sia $R_2[K_2] \subseteq R_1[K_1]$
  - ridondanze lasciate
- Si possono presentare potenziali problemi di efficienza rispetto al tempo di esecuzione delle query
  - ridondanze lasciate
  - numero e struttura delle relazioni

Ci devono essere dei buoni motivi legati all'**efficienza** per cambiare le scelte fatte

Modello di costo per lo schema logico

- Una relazione occupa un certo numero di pagine di memoria secondaria
- Il numero di accessi alle pagine della memoria secondaria domina l'elaborazione in memoria centrale
- Un accesso a memoria secondaria avviene ad una pagina intera
- Il numero complessivo di pagine accedute dipende in generale:
  - dal tipo di operazione
  - dal numero di tuple delle relazioni coinvolte
  - dal numero di tuple per pagina di memoria secondaria (determina il numero di pagine delle relazioni coinvolte)

Modello di costo: esempi

Per una relazione **R** denotiamo con

- $N_P(R)$ il numero di pagine in memoria secondaria occupate da **R**
- $N_{TP}(R)$ il numero di tuple per ogni pagina di **R**

Costo delle operazioni:

- la selezione su una relazione **R** ha costo pari a $N_P(R)$
- la proiezione su una relazione **R** ha costo pari a $N_P(R)$
- il join di **R** con **Q** si basa su un doppio ciclo
  - se non ci sono indici, allora il costo è $N_P(R) \times N_P(Q)$:
    per ogni tupla **r** di **R**
      per ogni tupla **q** di **Q**
        se **r** e **q** sono in join, metti la tupla nel risultato
  - se **Q** ha un indice molto selettivo sull'attributo di join, allora il costo è $N_P(R) + N_P(R) \times N_{TP}(R)$

Criteri generali per individuare potenziali problemi

- relazione con
  - tante tuple → la relazione occupa molte pagine
  - tanti attributi → una pagina contiene poche tuple
- attributi con tanti valori nulli
  → spreco di spazio → una pagina contiene poche tuple
- la proiezione è costosa (quando una pagina contiene poche tuple)
- il join è costoso (quasi sempre)
- la verifica di vincoli è costosa

Conclusione: per una relazione **R** rilevante (con tante tuple) occorre tentare di tenere basso $N_P(R)$

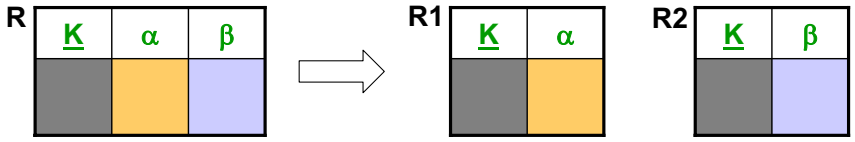
Ristrutturazioni dello schema logico

- **Decomposizione**
  - *Verticale* (sempre sulla chiave)
    - per facilitare l'accesso (con selezioni e proiezioni)
    - per normalizzazione (ignoreremo questo aspetto)
  - *Orizzontale*
    - per facilitare l'accesso (con selezioni)
  - *Mista*
    - per evitare valori nulli
- **Accorpamento**
  - per facilitare l'accesso (evita join)
  - per eliminare relazioni inutili

Le relazioni dello schema originario possono essere ricostruite attraverso la definizione di opportune viste

Nota: le ristrutturazioni si applicano in presenza di determinati attributi che formano una chiave di relazione. Sulle slide indicheremo tali attributi con **K**, intendendo che **K** è una chiave della relazione corrispondente

Decomposizione verticale per facilitare l'accesso



Vincoli dello schema ristrutturato:

- foreign key: R1[K] ⊆ R2[K]
  foreign key: R2[K] ⊆ R1[K]
- vincoli di inclusione da e verso **R** si definiscono su **R1** (o **R2**)
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando gli accessi ad **R** avvengono prevalentemente in modo separato sugli attributi α rispetto agli attributi β
- $N_{TP}(R1)$ e $N_{TP}(R2)$ sono alti rispetto a $N_{TP}(R)$ e quindi $N_P(R1)$ e $N_P(R2)$ sono bassi rispetto a $N_P(R)$
- La relazione **R** può essere ricostruita attraverso una vista che calcola il join tra **R1** ed **R2** su **K**

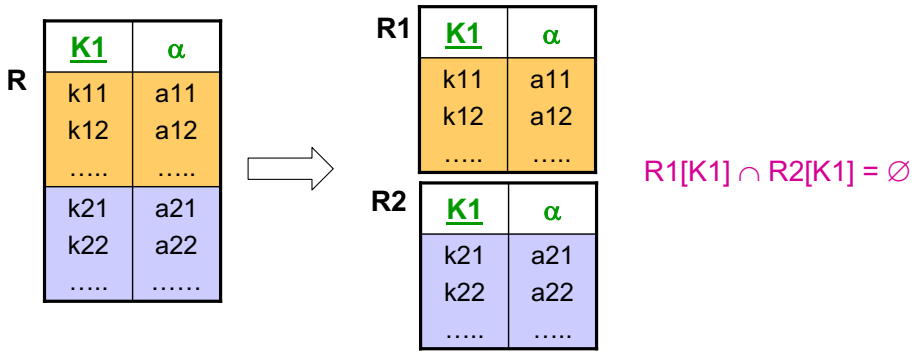
Decomposizione verticale: esempio

Immobile(Codice, Particella, Zona, Indirizzo, Città, Mq, NumVani, Valore)
foreign key: Immobile[Città] ⊆ Città [Cod]
Agenzia(CodiceAg, Immobile)
foreign key: Agenzia[Immobile] ⊆ Immobile[Codice]

Supponiamo che ai dati catastali degli immobili (particella, zona, indirizzo, città) si acceda prevalentemente in modo separato rispetto ai dati commerciali (metri quadri, numero di vani, valore). Applichiamo quindi la decomposizione verticale, ed otteniamo:

ImmobileCatasto(Codice, Particella, Zona, Indirizzo, Città)
foreign key: ImmobileCatasto[Città] ⊆ Città [Cod]
foreign key: ImmobileCatasto[Codice] ⊆ ImmobileComm [Codice]
ImmobileComm(Codice, Mq, NumVani, Valore)
foreign key: ImmobileComm[Codice] ⊆ ImmobileCatasto[Codice]
Agenzia(CodiceAg, Immobile)
foreign key: Agenzia[Immobile] ⊆ ImmobileCatasto[Codice]

Decomposizione orizzontale per facilitare l'accesso



$R1[K1] \cap R2[K1] = \varnothing$

Ulteriori vincoli dello schema ristrutturato:

- vincoli di inclusione da **R** diventano vincoli di inclusione da **R1** e da **R2**
- vincoli di inclusione a **R** diventano vincoli di inclusione a **R1** $\cup$ **R2**
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando gli accessi alle tuple di **R** di una delle due "fasce" avvengono separatamente dagli accessi alle tuple dell'altra fascia
- $N_P(R1)$ e $N_P(R2)$ sono bassi rispetto a $N_P(R)$
- **R** può essere ricostruita attraverso una vista che calcola l'unione di **R1** ed **R2**

Decomposizione orizzontale: esempio

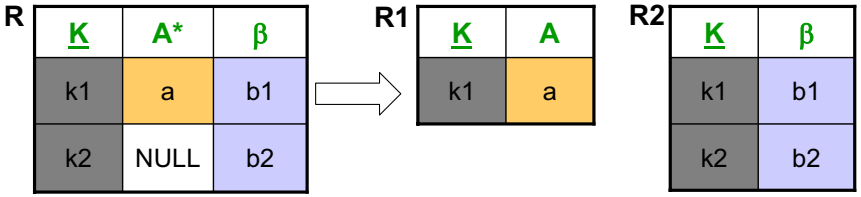
Telefonata(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)
foreign key: Telefonata[UtenzaInvio] $\subseteq$ Utenza[Cod]
foreign key: Telefonata[UtenzaDestinazione] $\subseteq$ Utenza[Cod]
Centrale(Codice, Telefonata)
foreign key: Centrale[Telefonata] $\subseteq$ Telefonata[Codice]

Supponiamo che alle telefonate si acceda per fasce orarie (giorno, sera, notte) determinate sul tempo di inizio. Applichiamo quindi la decomposizione orizzontale (opportuni vincoli detteranno le regole per i valori corretti del campo OrarioInizio delle tabelle e realizzeranno quindi i vincoli di disgiuntezza), ed otteniamo:

TelefonataGiorno(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)
foreign key: TelefonataGiorno[UtenzaInvio] $\subseteq$ Utenza[Cod]
foreign key: TelefonataGiorno[UtenzaDestinazione] $\subseteq$ Utenza[Cod]
TelefonataSera(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)
foreign key: TelefonataSera[UtenzaInvio] $\subseteq$ Utenza[Cod]
foreign key: TelefonataSera[UtenzaDestinazione] $\subseteq$ Utenza[Cod]
TelefonataNotte(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)
foreign key: TelefonataNotte[UtenzaInvio] $\subseteq$ Utenza[Cod]
foreign key: TelefonataNotte[UtenzaDestinazione] $\subseteq$ Utenza[Cod]
Centrale(Codice, Telefonata)

Vincolo esterno:
Centrale[Telefonata] $\subseteq$ TelefonataGiorno[Codice] $\cup$ TelefonataSera[Codice] $\cup$ TelefonataNotte[Codice]

Decomposizione mista per evitare valori nulli



A è l'attributo su cui si vogliono evitare valori nulli, e β denota un insieme di attributi.

Vincoli dello schema ristrutturato:

- foreign key: R1[K] $\subseteq$ R2[K]
- vincoli di inclusione da e verso **R** diventano inclusioni da e verso **R2**
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando la relazione ha tanti valori nulli in A
- $N_{TP}(R1)$, $N_{TP}(R2)$ sono alti rispetto a $N_{TP}(R)$ e quindi $N_P(R1)$, $N_P(R2)$ sono bassi rispetto a e $N_P(R)$
- **R** può essere ricostruita attraverso una vista che calcola il join esterno tra **R1** ed **R2** su **K**.

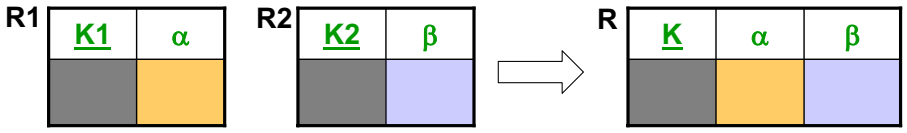
Decomposizione mista: esempio

Persona(CodFiscale, CittàNascita, NumTel*, DataMatrimonio*)
foreign key: Persona[CittàNascita] $\subseteq$ Città[Cod]
Città(Cod, Sindaco)
foreign key: Città[Sindaco] $\subseteq$ Persona[CodFiscale]

Supponiamo di non volere valori nulli. Applichiamo quindi due volte in cascata la decomposizione mista, ed otteniamo:

Persona(CodFiscale, CittàNascita)
foreign key: Persona[CittàNascita] $\subseteq$ Città[Cod]
PersonaConTelefono(CodFiscale, NumTel)
foreign key: PersonaConTelefono[CodFiscale] $\subseteq$ Persona[CodFiscale]
PersonaSposata(CodFiscale, DataMatrimonio)
foreign key: PersonaSposata[CodFiscale] $\subseteq$ Persona[CodFiscale]
Città(Cod, Sindaco)
foreign key: Città[Sindaco] $\subseteq$ Persona[CodFiscale]

Accorpamento per facilitare l'accesso



foreign key: R1[K1] ⊆ R2[K2]

foreign key: R2[K2] ⊆ R1[K1]

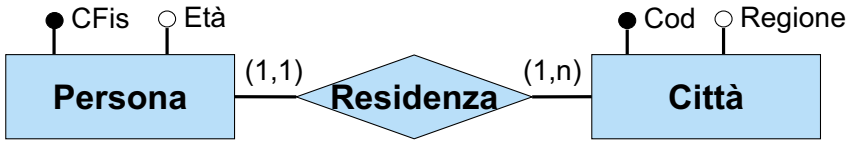
Si noti che **K1** e **K2** sono chiavi, anche non primarie. Si noti anche che se β manca, **R** coincide con **R1**, e l'effetto è quello di eliminare **R2**.

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono **R1** o **R2** vanno riformulati su **R**

- Si applica per facilitare gli accessi a **R1** e **R2** quando questi avvengono prevalentemente insieme e richiedono di calcolare il join tra **R1** e **R2** con **K1=K2**; in altre parole si applica per evitare il join tra **R1** e **R2**
- Le relazioni **R1** e **R2** possono essere ricostruite attraverso due viste che calcolano rispettivamente le proiezioni di **R** su **(K, α)** e su **(K, β)**
- Quando non c'è β, serve ad eliminare una relazione inutile (cioè **R2**)

Accorpamento: esempio 1

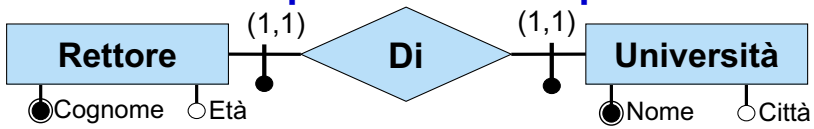


Persona(CFis, Età)
  foreign key: Persona[CFis] ⊆ Residenza[Persona]
Residenza(Persona, Città)
  foreign key: Residenza[Persona] ⊆ Persona[CFis]
  foreign key: Residenza[Città] ⊆ Città[Cod]
Città(Cod, Regione)
  inclusione: Città[Cod] ⊆ Residenza[Città]

Supponiamo che quando si accede alla persone si acceda spesso anche alla sua città di residenza. Applichiamo quindi l'accorpamento, ed otteniamo:

Persona(CFis, Età, Città)
  foreign key: Persona[Città] ⊆ Città[Cod]
Città(Cod, Regione)
  inclusione: Città[Cod] ⊆ Persona[Città]

Accorpamento: esempio 2



Rettore(Cognome, Età)
  foreign key: Rettore[Cognome] ⊆ Di[Rettore]
Di(Rettore, Università)
  foreign key: Di[Rettore] ⊆ Rettore[Cognome]
  foreign key: Di[Università] ⊆ Università[Nome]
  chiave: Rettore
Università(Nome, Città)
  foreign key: Università[Nome] ⊆ Di[Università]

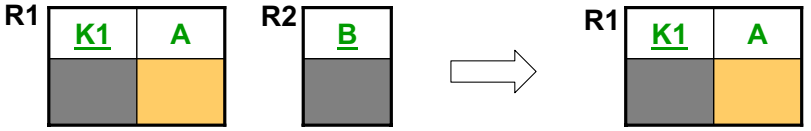
Supponiamo che quando si accede ad una università si acceda anche al suo rettore. Applichiamo l'accorpamento di Università e Di, ed otteniamo

Rettore(Cognome, Età)
  foreign key: Rettore[Cognome] ⊆ Università[Rettore]
Università(Nome, Città, Rettore)
  foreign key: Università[Rettore] ⊆ Rettore[Cognome]

Supponiamo che quando si accede ad un rettore si acceda anche ai dati relativi alla sua università. Applichiamo un ulteriore accorpamento ed otteniamo

UniversitàRettore(Nome, Città, CognomeRettore, EtàRettore)
  chiave: CognomeRettore

Accorpamento per eliminare relazioni inutili



foreign key: R1[A] ⊆ R2[B]

inclusione: R2[B] ⊆ R1[A]

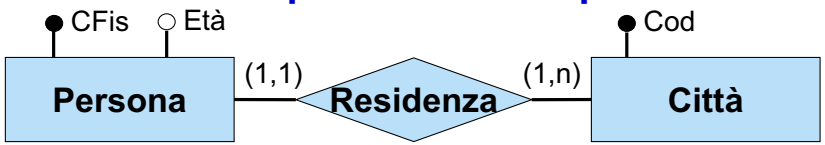
Si noti che **B** è chiave, ma **A** non lo è.

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono **R2** vanno riformulati su **R1**

- Si applica per eliminare la relazione **R2** che è inutile, visto che tutti i valori nell'unico suo attributo si ritrovano nell'attributo **A** della relazione **R1**
- La relazione **R2** può essere ricostruita attraverso la proiezione della relazione **R1** sull'attributo **A**

Accorpamento: esempio 3



Persona(CFis, Età)
  foreign key: Persona[CFis] ⊆ Residenza[Persona]
Residenza(Persona, Città)
  foreign key: Residenza[Persona] ⊆ Persona[CFis]
  foreign key: Residenza[Città] ⊆ Città[Cod]
Città(Cod)
  inclusione: Città[Cod] ⊆ Residenza[Città]

Supponiamo che quando si accede alla persone si acceda spesso anche alla sua città di residenza. Applichiamo quindi l'accorpamento, ed otteniamo:

Persona(CFis, Età, Città)
  foreign key: Persona[Città] ⊆ Città[Cod]
Città(Cod)
  inclusione: Città[Cod] ⊆ Persona[Città]

Applichiamo quindi di nuovo l'accorpamento per eliminare la relazione inutile (Città), ed otteniamo:

Persona(CFis, Età, Città)

Traduzione in SQL

Ogni relazione viene definita tramite la "create table", nel modo ovvio. In particolare, i vincoli vengono tradotti secondo queste indicazioni generali:

- attributi obbligatori: `not null`
- chiave primaria: `primary key`
- chiave: `unique`
- foreign key: `foreign key`
- interdipendenza valori nulli:

```
check ((A is null and B is null) or
       (A is not null and B is not null))
```

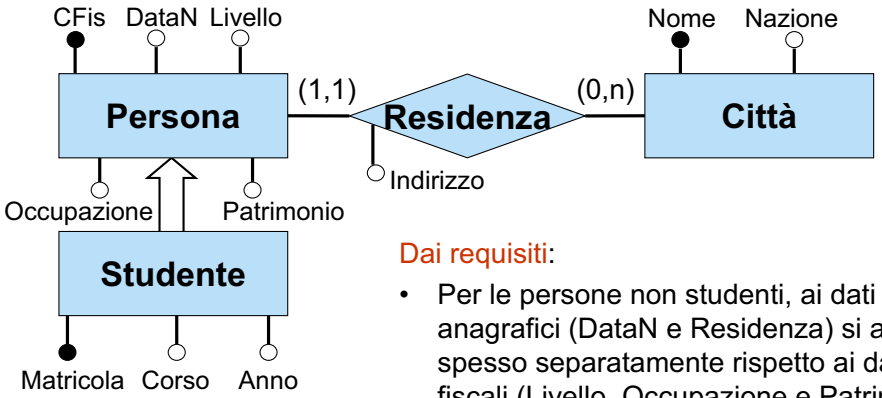
- inclusione: `check (A in (select B from R))`
- disgiunzione: `check (A not in (select B from R))`
- cardinalità (i,j):

```
check ((i ≤ (select count(*) from R where ...) and
       (j ≥ (select count(*) from R where ...))
```

- vincoli esterni: `assertion` o controllati a livello di applicazione

Esercizio 7: effettuare la progettazione logica

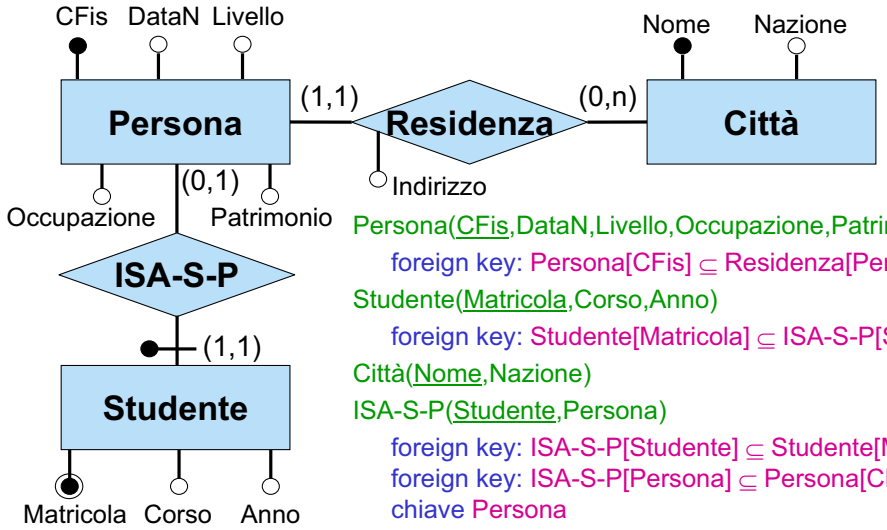
Schema concettuale:



Dai requisiti:

- Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio)
- Agli studenti si accede prevalentemente per matricola, e si accede spesso ai dati universitari (Corso, Anno) insieme alla data di nascita e ai dati fiscali (Livello, Occupazione e Patrimonio)

Esercizio 7: ristrutturazione e traduzione diretta



Persona(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: Persona[CFis] ⊆ Residenza[Persona]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] ⊆ ISA-S-P[Stud]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] ⊆ Studente[Matricola]
  foreign key: ISA-S-P[Persona] ⊆ Persona[CFis]
  chiave Persona
Residenza(Persona,Indirizzo,Città)
  foreign key: Residenza[Persona] ⊆ Persona[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]

Esercizio 7: ristrutturazioni dello schema logico

Per le persone non studenti valgono criteri di accesso diversi rispetto alle persone che sono studenti, anche relativamente ai dati relativi alla residenza:

- **decomposizione orizzontale** di Persona in PersonaNonStudente e PersonaStudente
- **decomposizione orizzontale** di Residenza in ResidenzaNonStudente e ResidenzaStudente

PersonaNonStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaNonStudente[CFis] ⊆ ResidenzaNonStudente[Persona]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaStudente[CFis] ⊆ ISA-S-P[Persona]
  foreign key: PersonaStudente[CFis] ⊆ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] ⊆ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] ⊆ Studente[Matricola]
  foreign key: ISA-S-P[Persona] ⊆ PersonaStudente[CFis] chiave: Persona
ResidenzaNonStudente(Persona,Indirizzo,Città)
  foreign key: Residenza[Persona] ⊆ PersonaNonStudente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] ⊆ PersonaStudente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
Vincolo: PersonaNonStudente[CFis] ∩ PersonaStudente[Cfis] = ∅

Esercizio 7: ristrutturazioni dello schema logico

Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio):

- **decomposizione verticale** di PersonaNonStudente in PersonaDatiAnagrafe e PersonaDatiFisco

PersonaDatiAnagrafe(CFis,DataN)
  foreign key: PersonaDatiAnagrafe[CFis] ⊆ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[CFis] ⊆ ResidenzaNonStudente[Persona]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] ⊆ PersonaDatiAnagrafe[CFis]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaStudente[CFis] ⊆ ISA-S-P[Persona]
  foreign key: PersonaStudente[CFis] ⊆ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] ⊆ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] ⊆ Studente[Matricola]
  foreign key: ISA-S-P[Persona] ⊆ PersonaStudente[CFis] chiave: Persona
ResidenzaNonStudente(Persona,Indirizzo,Città)
  foreign key: Residenza[Persona] ⊆ PersonaDatiAnagrafe[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] ⊆ PersonaStudente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] ∩ PersonaStudente[Cfis] = ∅

Esercizio 7: ristrutturazioni dello schema logico

Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio):

- **accorpamento** tra PersonaDatiAnagrafe e ResidenzaNonStudente

PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)
  foreign key: PersonaDatiAnagrafe[CFis] ⊆ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[Città] ⊆ Città[Nome]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] ⊆ PersonaDatiAnagrafe[CFis]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaStudente[CFis] ⊆ ISA-S-P[Persona]
  foreign key: PersonaStudente[CFis] ⊆ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] ⊆ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] ⊆ Studente[Matricola]
  foreign key: ISA-S-P[Persona] ⊆ PersonaStudente[CFis] chiave: Persona
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] ⊆ PersonaStudente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] ∩ PersonaStudente[Cfis] = ∅

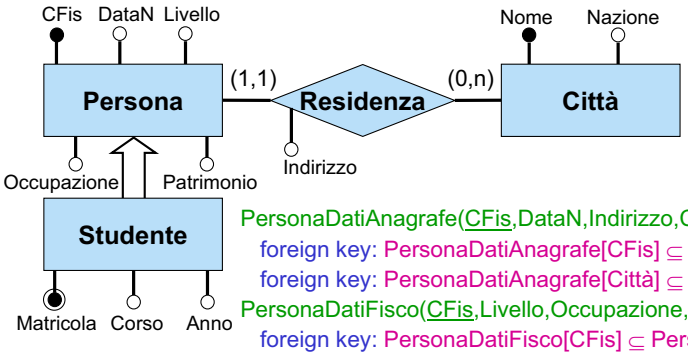
Esercizio 7: ristrutturazioni dello schema logico

Agli studenti si accede prevalentemente per matricola, e si accede spesso ai dati universitari (Corso, Anno) insieme alla data di nascita e ai dati fiscali (Livello, Occupazione e Patrimonio):

- **accorpamento** tra Studente, ISA-S-P e PersonaStudente

PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)
  foreign key: PersonaDatiAnagrafe[CFis] ⊆ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[Città] ⊆ Città[Nome]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] ⊆ PersonaDatiAnagrafe[CFis]
Studente(Matricola,Corso,Anno,CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: Studente[CFis] ⊆ ResidenzaStudente[Studente]
  chiave: CFis
Città(Nome,Nazione)
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] ⊆ Studente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] ∩ Studente[Cfis] = ∅

Esercizio 7: schema logico e confronto col concettuale



PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)
  foreign key: PersonaDatiAnagrafe[CFis] ⊆ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[Città] ⊆ Città[Nome]

PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] ⊆ PersonaDatiAnagrafe[CFis]

Studente(Matricola,Corso,Anno,CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: Studente[CFis] ⊆ ResidenzaStudente[Studente]
  chiave: CFis

Città(Nome,Nazione)

ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] ⊆ Studente[CFis]
  foreign key: Residenza[Città] ⊆ Città[Nome]

Vincolo: PersonaDatiAnagrafe[CFis] ∩ Studente[Cfis] = ∅

Si lascia come esercizio la definizione delle viste per ricostruire le relazioni dello schema logico originario

Esercizio 8: ristrutturare il seguente schema ...

Volo(Codice, Comp, Durata)
  foreign key: Volo[Comp] ⊆ Compagnia[Nome]
  foreign key: Volo[Codice,Comp] ⊆ ArrPart[Codice,Comp]

ArrPart(Codice, Comp, Arrivo, Partenza)
  foreign key: ArrPart[Arrivo] ⊆ Aeroporto[Codice]
  foreign key: ArrPart[Partenza] ⊆ Aeroporto[Codice]
  foreign key: ArrPart[Codice,Comp] ⊆ Volo[Codice,Comp]
  chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo)
  foreign key: VoloCharter[Codice,Comp] ⊆ Volo[Codice,Comp]

Aeroporto(Codice, Nome)
  foreign key: Aeroporto[Codice] ⊆ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
  foreign key: LuogoAeroporto[Aeroporto] ⊆ Aeroporto[Codice]
  foreign key: LuogoAeroporto[NomeCittà,NazCittà] ⊆ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond)
  foreign key: Compagnia[Nome] ⊆ Sede[Comp]

Esercizio 8: ristrutturare il seguente schema...(parte 2)

Sede(Comp)
  foreign key: Sede[Comp] ⊆ Compagnia[Nome]
  foreign key: Sede[Comp] ⊆ CittàSede[Comp]
  inclusione: Sede[Comp] ⊆ TelefonoComp[Comp]

CittàSede(Comp, NomeCittà, NazCittà)
  foreign key: CittàSede[Comp] ⊆ Sede[Comp]
  foreign key: CittàSede[NomeCittà,NazCittà] ⊆ Città[Nome,Nazione]

TelefonoComp(Numero, Comp)
  foreign key: Telefono[Comp] ⊆ Sede[Comp]
  foreign key: TelefonoComp[Numero] ⊆ Telefono[Numero]

Telefono(Numero)
  foreign key: Telefono[Numero] ⊆ TelefonoComp[Numero]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
  foreign key: Tappa[CodVoloCharter,Comp] ⊆ VoloCharter[Codice,Comp]
  foreign key: Tappa[Aeroporto] ⊆ Aeroporto[Codice]

Vincolo esterno: vincolo su Ordine in Tappa

... tenendo conto delle seguenti specifiche

- Si accede spesso all'insieme dei voli charter separatamente dagli altri voli, sia per quanto riguarda la durata, sia per quanto riguarda gli aeroporti di arrivo e partenza
- Quando si accede ad un volo (charter o no) si vuole spesso conoscere tutte le proprietà di tale volo (durata, tipo aereo se volo charter, aeroporto di arrivo e aeroporto di partenza)
- Quando si accede ad una compagnia aerea si accede anche ai dati relativi alla sua sede

Esercizio 8: soluzione – ristrutturazioni

- Si accede spesso all'insieme dei voli charter separatamente dagli altri voli, sia per quanto riguarda la durata, sia per quanto riguarda gli aeroporti di arrivo e partenza:
  - **decomposizione orizzontale** di Volo in
    - VoloNonCharter
    - DatiVoloCharter
  - **decomposizione orizzontale** di ArrPart in
    - ArrPartVoloNonCharter
    - ArrPartVoloCharter
- Quando si accede ad un volo (charter o no) si vuole spesso conoscere tutte le proprietà di tale volo (durata, tipo aereo se volo charter, aeroporto di arrivo e aeroporto di partenza):
  - **accorpamento** di DatiVoloCharter e VoloCharter e, successivamente, della relazione risultante con ArrPartVoloCharter
  - **accorpamento** di VoloNonCharter e ArrPartVoloNonCharter e, successivamente, della relazione risultante con VoloNonCharter
- Quando si accede alla compagnia si accede anche ai dati relativi alla sua sede:
  - **accorpamento** di Compagnia e CittàSede
- Si eliminano le relazioni inutili Sede e Telefono mediante **accorpamento**

Esercizio 8: soluzione – schema ristrutturato

VoloNonCharter(Codice, Comp, Durata, Arrivo, Partenza)
  foreign key: VoloNonCharter[Comp] ⊆ Compagnia[Nome]
  foreign key: VoloNonCharter[Arrivo] ⊆ Aeroporto[Codice]
  foreign key: VoloNonCharter[Partenza] ⊆ Aeroporto[Codice]
  chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo, Durata, Arrivo, Partenza)
  foreign key: VoloCharter[Comp] ⊆ Compagnia[Nome]
  foreign key: VoloCharter[Arrivo] ⊆ Aeroporto[Codice]
  foreign key: VoloCharter[Partenza] ⊆ Aeroporto[Codice]
  chiave: Comp, Arrivo, Partenza

Aeroporto(Codice, Nome)
  foreign key: Aeroporto[Codice] ⊆ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
  foreign key: LuogoAeroporto[Aeroporto] ⊆ Aeroporto[Codice]
  foreign key: LuogoAeroporto[NomeCittà, NazCittà] ⊆ Città[Nome, Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond, NomeCittà, NazCittà)
  foreign key: Compagnia[NomeCittà, NazCittà] ⊆ Città[Nome, Nazione]
  inclusione: Compagnia[Nome] ⊆ TelefonoComp[Comp]

TelefonoCom(Numero, Comp)
  foreign key: TelefonoComp[Comp] ⊆ Compagnia[Nome]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
  foreign key: Tappa[CodVoloCharter, Comp] ⊆ VoloCharter[Codice, Comp]
  foreign key: Tappa[Aeroporto] ⊆ Aeroporto[Codice]

Esercizio 8: soluzione – vincoli e viste

Vincoli:

- VoloNonCharter e VoloCharter sono disgiunti:
  VoloNonCharter[Codice, Comp] ∩ VoloCharter[Codice, Comp] = ∅
  VoloNonCharter[Comp, Arrivo, Partenza] ∩ VoloCharter[Comp, Arrivo, Partenza] = ∅
- vincolo su Ordine in Tappa

Viste per ricostruire le relazioni dello schema originario:

view Volo = $\text{PROJ}_{\text{Codice,Comp,Durata}}$(VoloNonCharter) ∪ $\text{PROJ}_{\text{Codice,Comp,Durata}}$(VoloCharter)

view ArrPart = $\text{PROJ}_{\text{Codice,Comp,Arrivo,Partenza}}$(VoloNonCharter) ∪ $\text{PROJ}_{\text{Codice,Comp,Arrivo,Partenza}}$(VoloCharter)

view Sede = $\text{PROJ}_{\text{Nome}}$(Compagnia)

view CompagniaOriginaria = $\text{PROJ}_{\text{Nome, AnnoFond}}$(Compagnia)

view CittàSede = $\text{PROJ}_{\text{Nome,NomeCittà,NazCittà}}$ (Compagnia)

view Telefono = $\text{PROJ}_{\text{Numero}}$(TelefonoComp)

Esercizio 9: ristrutturare il seguente schema ...

Officina(Nome, NumDip, Indirizzo)
  foreign key: Officina[Nome] ⊆ Dirige[Officina]
  inclusione: Officina[Nome] ⊆ Lavora[Officina]

Persona(CodFis, Indirizzo)

Direttore(CodFis, Età, AnniAnz)
  foreign key: Direttore[CodFis] ⊆ Persona[CodFis]
  foreign key: Direttore[CodFis] ⊆ Dirige[Direttore]

Dipendente(CodFis, AnniAnz)
  foreign key: Dipendente[CodFis] ⊆ Persona[CodFis]
  inclusione: Dipendente[CodFis] ⊆ Lavora[Dipendente]

Dirige(Officina, Direttore)
  foreign key: Dirige[Officina] ⊆ Officina[Nome]
  foreign key: Dirige[Direttore] ⊆ Direttore[CodFis]
  chiave: Direttore

Lavora(Officina, Dipendente, AnniServizio)
  foreign key: Lavora[Officina] ⊆ Officina[Nome]
  foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]

TelPer(CodFis, Telefono)
  foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
  foreign key: TelPer[Telefono] ⊆ Telefono[Numero]

Esercizio 9: ristrutturare il seguente schema...(parte 2)

Telefono(Numero)
  inclusione: Telefono[Numero] ⊆ TelPer[Telefono]
Veicolo(Targa, Modello, Tipo, AnnoImm)
  foreign key : Veicolo[Targa] ⊆ Possiede[Veicolo]
Possiede(Veicolo, Proprietario)
  foreign key: Possiede[Veicolo] ⊆ Veicolo[Targa]
  foreign key: Possiede[Proprietario] ⊆ Persona[CodFis]
Riparazione(Codice, Officina, OraAcc, DataAcc)
  inclusione: Riparazione[Officina] ⊆ Officina[Nome]
  foreign key: Riparazione[Codice,Officina] ⊆ Relativa[Codice,Officina]
Relativa(Codice, Officina, Veicolo)
  foreign key: Relativa[Codice,Officina] ⊆ Riparazione[Codice,Officina]
  foreign key: Relativa[Veicolo] ⊆ Veicolo[Targa]
Terminata(Codice, Officina, OraRic, DataRic)
  foreign key: Terminata[Codice,Officina] ⊆ Riparazione[Codice,Officina]

Vincoli esterni:
- riconsegna dopo accettazione
- vincolo che lega Officina[NumDip] alle istanze in Lavora
- vincolo su AnniAnz di Direttore e Dipendente derivante dall'eliminazione ISA

... tenendo conto delle seguenti specifiche

- Quando si accede ai direttori, interessano anche tutti i dati relativi all'officina che dirigono e viceversa, quando si accede alle officine, interessano anche i dati relativi all'età e alla anzianità del loro direttore
- Ai dipendenti si accede spesso separatamente rispetto ai direttori
- Quando si accede ai dipendenti interessano anche i dati relativi all'indirizzo

Esercizio 9: soluzione – ristrutturazioni

- Quando si accede ai direttori, interessano anche tutti i dati relativi all'officina che dirigono e viceversa quando si accede alle officine, interessano anche tutti i dati relativi al loro direttore:
  - **accorpamento** di Direttore e Dirige e, successivamente, della relazione risultante con Officina
- Ai dipendenti si accede spesso separatamente rispetto ai direttori:
  - **decomposizione orizzontale** di Persona in dipendenti e non
- Quando si accede ai dipendenti interessano anche i loro dati anagrafici:
  - **accorpamento** tra la relazione e Dipendente e PersonaDipendente
- Si elimina la relazione inutile Telefono mediante **accorpamento**

Esercizio 9: soluzione – schema ristrutturato

OfficinaDirettore(Nome, NumDip, Indirizzo, Direttore, EtaDir, AnniAnzDir)
  chiave: Direttore
  inclusione: Officina[Nome] ⊆ Lavora[Officina]
PersonaNonDip(CodFis, Indirizzo)
Dipendente(CodFis, AnniAnz, Indirizzo)
  inclusione: Dipendente[CodFis] ⊆ Lavora[Dipendente]
Lavora(Officina, Dipendente, AnniServizio)
  foreign key: Lavora[Officina] ⊆ OfficinaDirettore[Nome]
  foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]
TelPer(CodFis, Telefono)
Veicolo(Targa, Modello, Tipo, AnnoImm)
Possiede(Veicolo, Proprietario)
  foreign key: Possiede[Veicolo] ⊆ Veicolo[Targa]
Riparazione(Codice, Officina, OraAcc, DataAcc,)
  inclusione: Riparazione[Officina] ⊆ OfficinaDirettore[Nome]
  foreign key: Riparazione[Codice,Officina] ⊆ Relativa[Codice,Officina]
Relativa(Codice, Officina, Veicolo)
  foreign key: Relativa[Codice,Officina] ⊆ Riparazione[Codice,Officina]
  foreign key: Relativa[Veicolo] ⊆ Veicolo[Targa]
Terminata(Codice, Officina, OraRic, DataRic)
  foreign key: Terminata[Codice,Officina] ⊆ Riparazione[Codice,Officina]

Esercizio 9: soluzione – vincoli e viste

Vincoli:

- PersonaNonDip e Dipendente sono disgiunti:
  - $\text{PersonaNonDip}[\text{CodFis}] \cap \text{Dipendente}[\text{CodFis}] = \varnothing$
- Vincoli risultanti dai vincolo di foreign key verso Persona
  - $\text{Officina}[\text{Direttore}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
  - $\text{Veicolo}[\text{Proprietario}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
  - $\text{TelPer}[\text{CodFis}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
- Vincoli esterni:
  - riconsegna dopo accettazione
  - vincolo che lega Officina[NumDip] alle istanze in Lavora
  - vincolo su Officina[AnniAnzDir] e Dipendente[AnniAnz] derivante dall'eliminazione ISA

Viste per ricostruire le relazioni dello schema originario:

view Persona = $\text{PersonaNonDip} \cup \text{PROJ}_{\text{CodFis,Indirizzo}}(\text{Dipendente})$

view OfficinaOrig = $\text{PROJ}_{\text{Nome, NumDip, Indirizzo}}(\text{Officina})$

view Direttore = $\text{PROJ}_{\text{Direttore, EtaDir, AnniAnzDir}}(\text{Officina})$

view Dirige = $\text{PROJ}_{\text{Nome, Direttore}}(\text{Officina})$

view Telefono = $\text{PROJ}_{\text{Telefono}}(\text{TelPer})$