Elementi di Informatica LB

Basi di Dati

Anno accademico 2007/2008

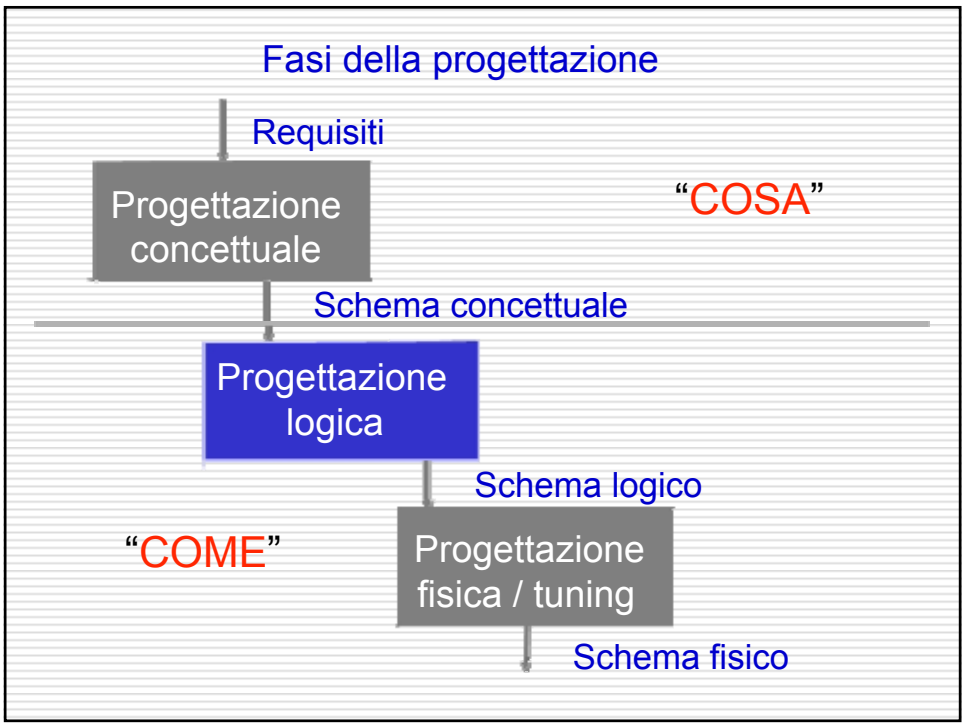
Prof. Stefano Contadini

Elementi di Informatica LB

La progettazione logica

titolo:

1. Introduzione alla progettazione logica



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

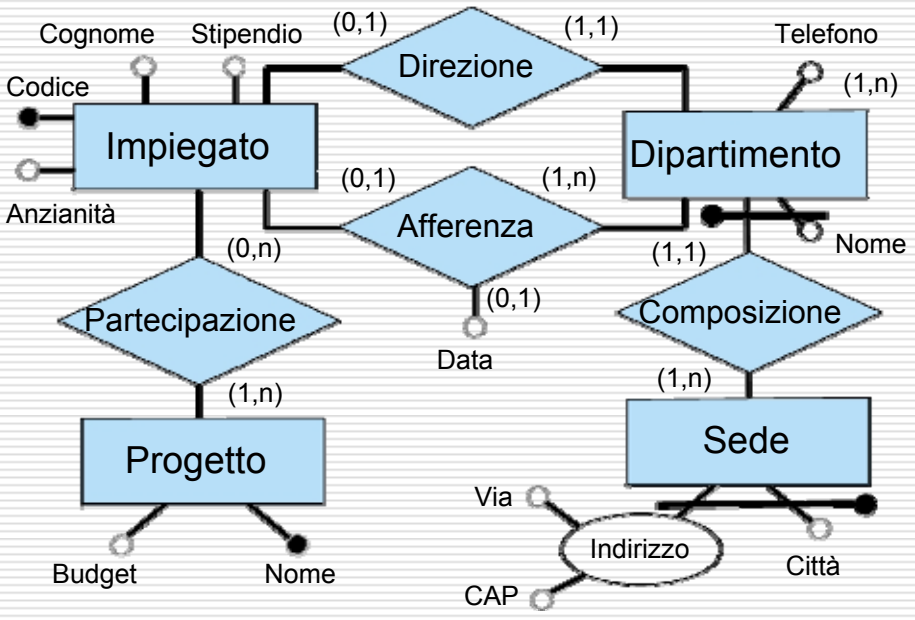
Consideriamo degli "indicatori" dei parametri che regolano le prestazioni:
- tempo di esecuzione delle operazioni di principale interesse: numero di istanze (di entità e relazioni) mediamente accedute durante l'esecuzione dell'operazione (accessi)

-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

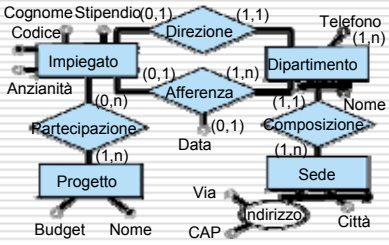
Si noti che i valori relativi al numero di istanze di entità e relazioni nella tabella dei volumi sono influenzati:

- 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).

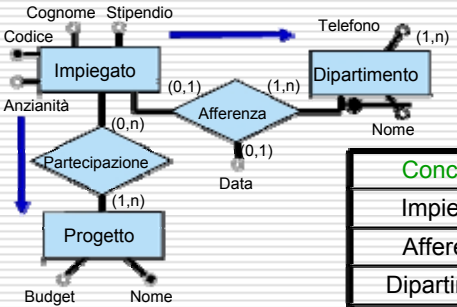


Tabella degli accessi

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 primari 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
   - 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)

Elementi di Informatica LB

La progettazione logica

titolo:

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

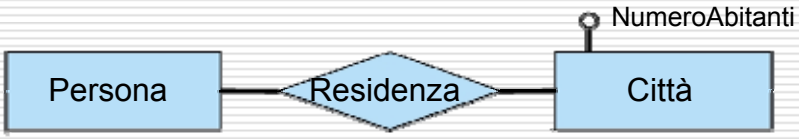


Tabella dei volumi

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: 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

Valutazione dei costi: 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

Ristrutturazione
fase 2: eliminazione di attributi multivalore

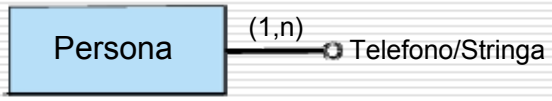
- Un attributo multivalore (ovvero un attributo con cardinalità massima maggiore di 1) non può essere tradotto direttamente nel modello relazionale senza introdurre delle ridondanze nelle relazioni ottenute.
- Dobbiamo quindi eliminare tutti gli attributi multivalore.
- L'eliminazione di un attributo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

Nota:
se trasformiamo una relazione R in un'entità, e R stava in ISA o in una gerarchia con altre relazioni, allora anche queste devono essere trasformate in entità

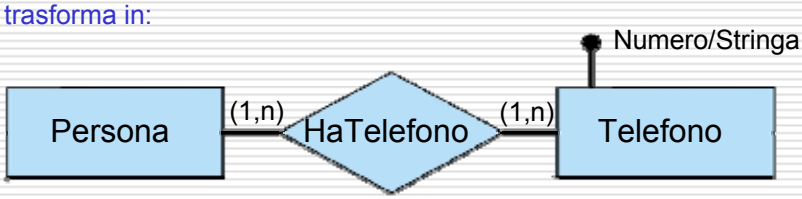
Eliminazione di attributi multivalore di entità

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

Esempio:

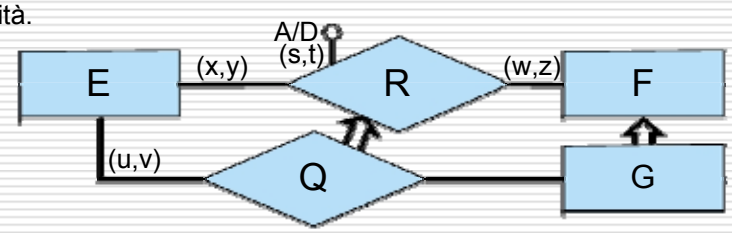


Si trasforma in:

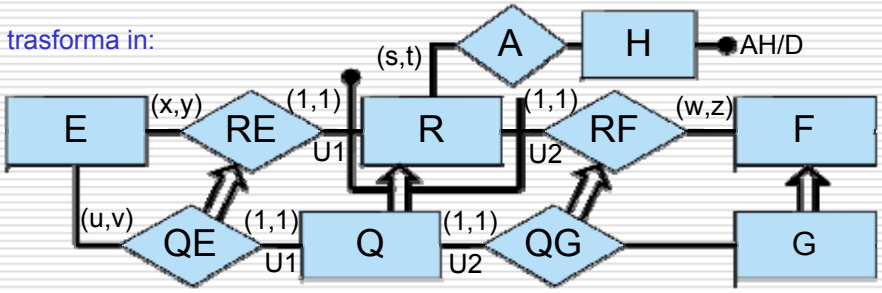


Eliminazione di attributi multivalore di relazioni

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).

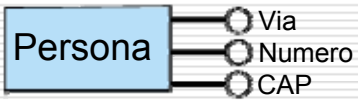
- Se l'attributo ha cardinalita` (1,1), si associano direttamente gli attributi componenti all'entità (o alla relazione).
- Se l'attributo ha cardinalità (0,1) si può
  - procedere come per gli attributi (1,1), ma con l'avvertenza che l'opzionalità diventa un vincolo esterno
  - oppure trasformare l'attributo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

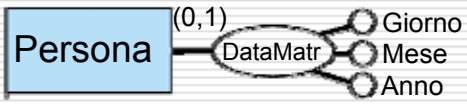
Si associano direttamente gli attributi componenti all'entità (o alla relazione) a cui è associato l'attributo:

Esempio:

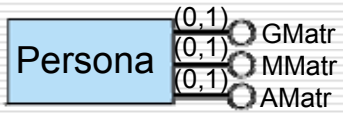
viene trasformato in:



Esempio:



viene trasformato in:

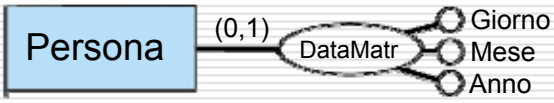


... con il vincolo esterno: per ogni istanza di Persona, ciascun attributo tra GMatr, MMatr e AMatr è definito se e solo se lo sono anche gli atri due.

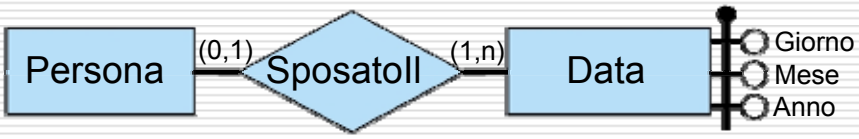
Eliminazione di attributi composti: alternativa 2

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

Esempio:



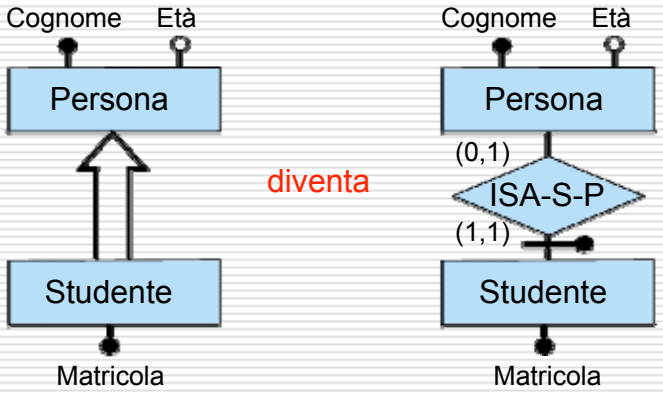
Viene trasformato in:



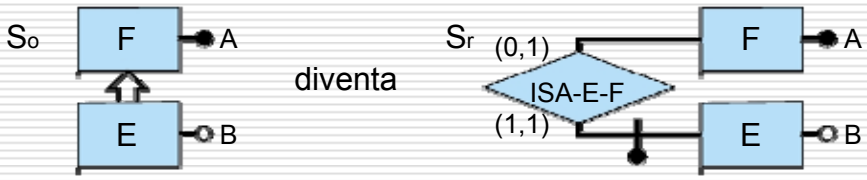
Ristrutturazione
fase 4: eliminazione di ISA tra entità

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) }

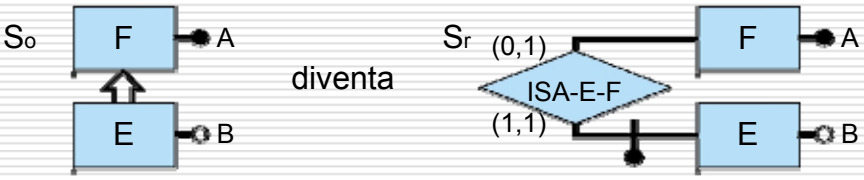
Istanza di Sr:
istanze(F) = { f_1, f_2, f_3 }
istanze(E) = { e_1, e_2 }
istanze(A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(B) = { (e_1,b_1), (e_2 ,b_2) }
istanze(ISA-E-F) = { (E:e_1,F:f_1), (E:e_2 ,F:f_2) }

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.

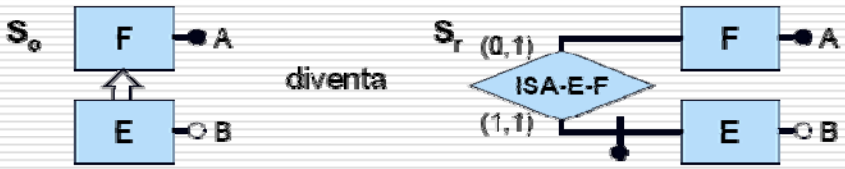
Mostriamo questa proprietà tramite un esempio:



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

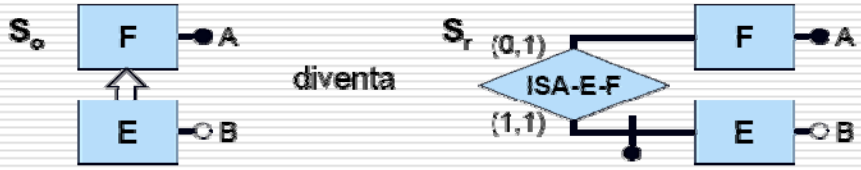


Definiamo la funzione g: istanze(S_o) → istanze(S_r) in modo che, ad un'istanza I_o di S_o, g assegni un'istanza I_r di S_r definita al seguente modo:

- *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 $x \in$ *istanze*(I_o,E), un nuovo oggetto $g_E(x)$, e definiamo
  istanze(I_r,E) = { $g_E(x)$ | $x \in$ *istanze*(I_o,E) }
- *istanze*(I_r,ISA-E-F) = { ($g_E(x)$,x) | $x \in$ *istanze*(I_o,E) }
- *istanze*(I_r,B) = { ($g_E(x)$,b) | (x,b) $\in$ *istanze*(I_o,B) }

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

Corrispondenza tra istanze – funzione h

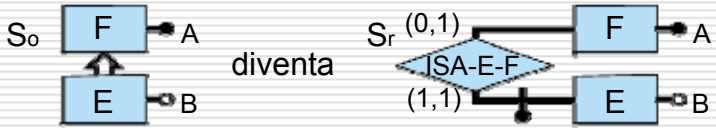


Definiamo la funzione h: istanze(S_r) → istanze(S_o) in modo che, ad un'istanza I_r di S_r, h assegni l'istanza I_o di S_o definita al seguente modo:

- *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.

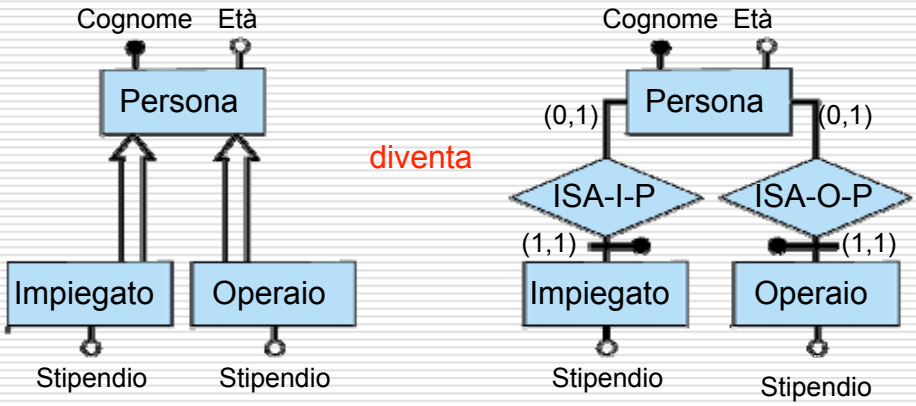
Come si concilia questa osservazione con il fatto che nello schema originario ciò non era vero? La risposta sta nelle funzioni g ed h illustrate precedentemente. Tramite queste funzioni è possibile infatti stabilire se due qualunque istanze di entità nello schema risultante corrispondono ad un'unica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

Dobbiamo quindi imporre che le due funzioni, quando applicate a due oggetti e, f dello schema risultante che corrispondono allo stesso oggetto dello schema originario, assegnino ad e ed f lo stesso valore. Ciò viene fatto con un vincolo esterno.

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)

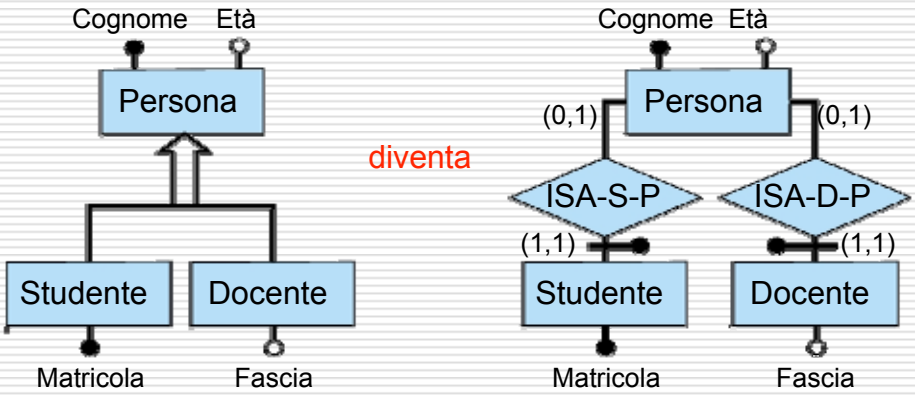
Ristrutturazione – fase 4: eliminazione di generalizzazioni tra entità

- 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 $ISA\text{-}E_1\text{-}F, \ldots, ISA\text{-}E_n\text{-}F$.
- 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 $ISA\text{-}E_1\text{-}F, \ldots, ISA\text{-}E_n\text{-}F$
  - 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* $ISA\text{-}E_1\text{-}F, \ldots, ISA\text{-}E_n\text{-}F$

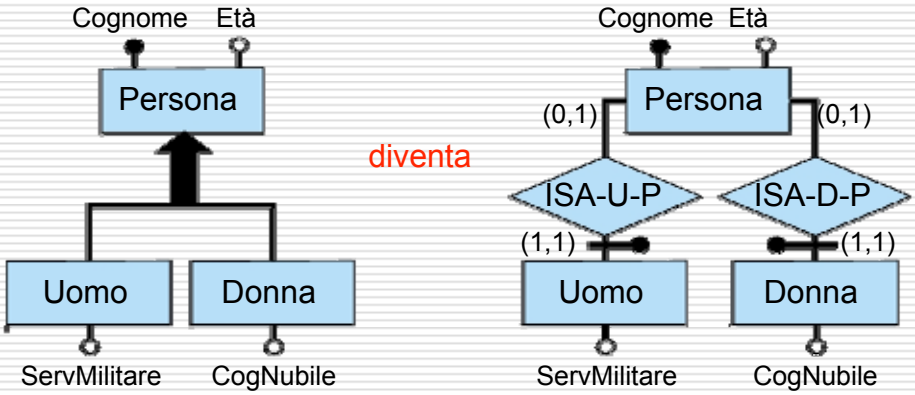
Eliminazione di generalizzazioni tra entità



Vincoli di generalizzazione:

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

Eliminazione di generalizzazioni complete tra entità



Vincoli di generalizzazione:

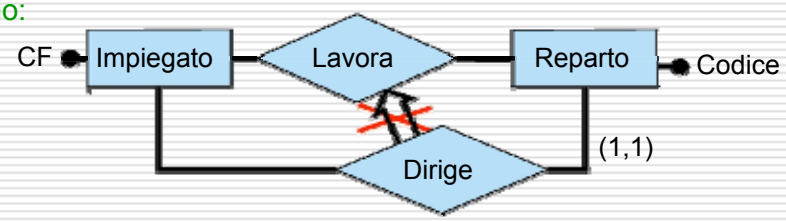
ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

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) insistono su esattamente le stesse entità, è immediato esprimere il vincolo esterno.

Esempio:

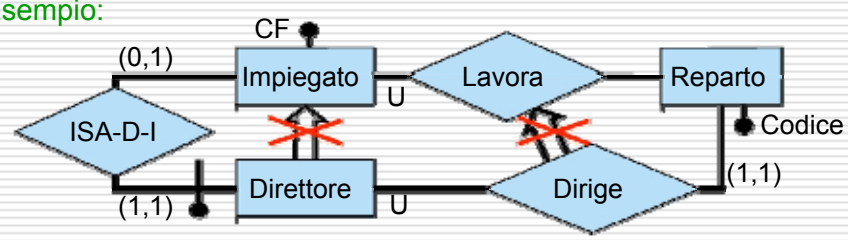


Vincolo esterno: ogni istanza di Dirige è anche un'istanza di Lavora.

Eliminazione di ISA e generalizzazioni tra relazioni

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

Esempio:



Vincolo esterno: per ogni istanza (d,r) di Dirige, sia i l'istanza di Impiegato tale che (d,i) è un'istanza di ISA-D-I. Si noti che i esiste sempre ed è unica. Allora (i,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 con pochi campi)
- 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

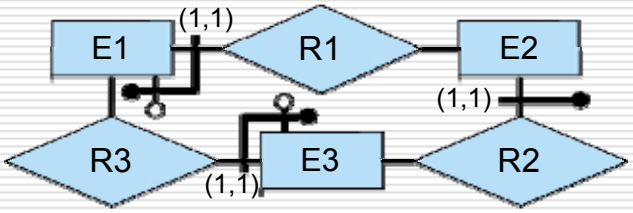
Nella scelta degli identificatori principali è necessario fare attenzione a non introdurre cicli di identificazione esterna.

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

- ad ogni entità del diagramma corrisponde un nodo
- c'e un arco dall'entità E all'entità F se E partecipa ad un identificatore (principale) esterno di F.

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

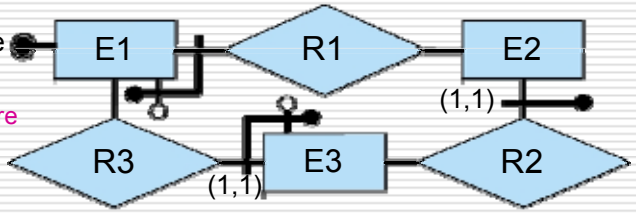
Cicli di identificazione esterna



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. (1,1)

Codice

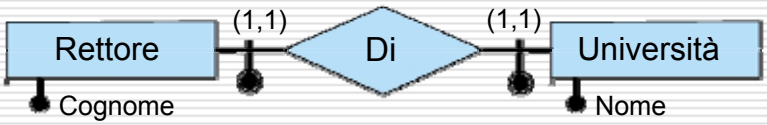
Codice è l'identificatore principale.



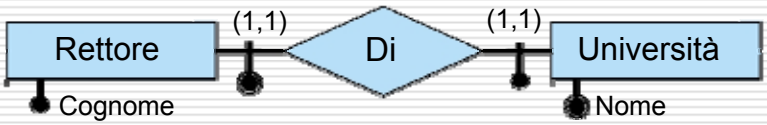
Cicli di identificazione esterna: esempio

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

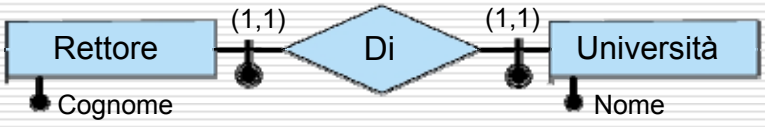
Esempio:

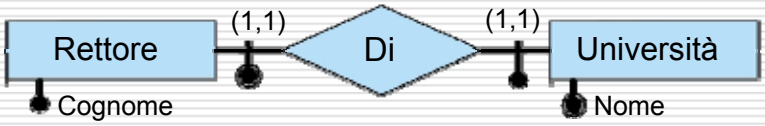


Abbiamo un ciclo di identificazione esterna, che deve essere spezzato. Una possibilità è la seguente:



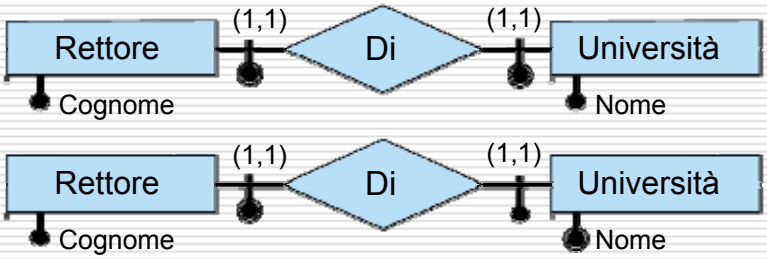
Esercizio 2: cicli di identificazione esterna





- 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 la cui chiave è data dagli identificatori di tutte le entità che partecipano ad R. In presenza di cicli di identificazione esterna questo non è possibile.

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)

- 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)

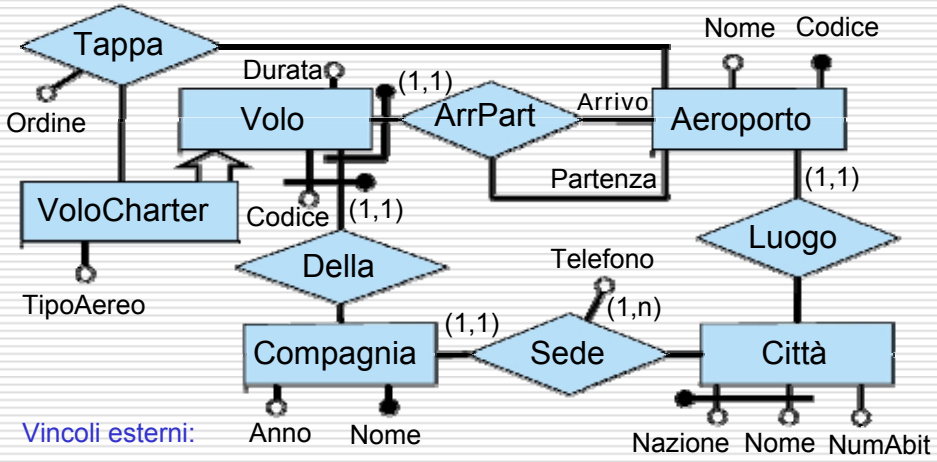
Ristrutturazione - fase 7: riformulazione di operazioni e carico applicativo

- 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 vincolo esterno)
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 - altrimenti 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

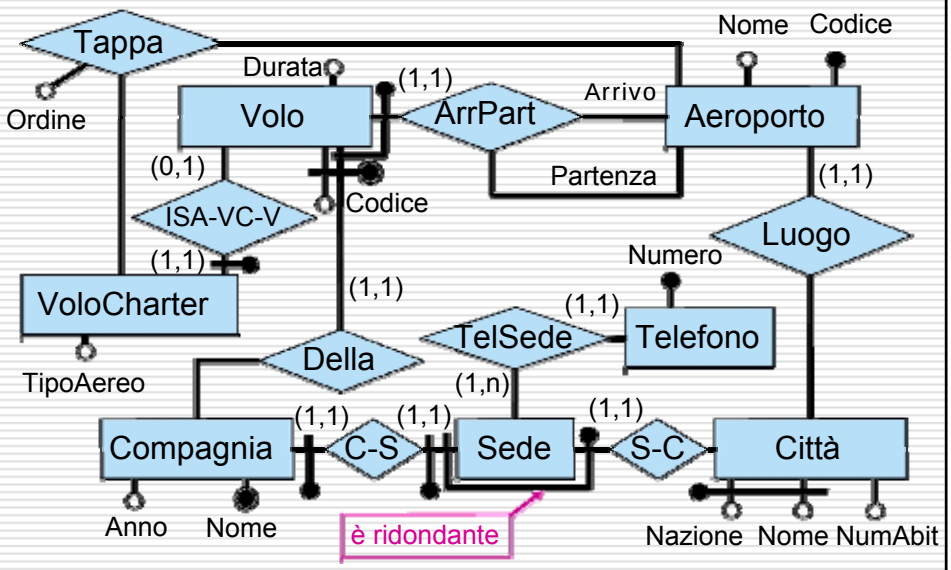
Esercizio 3: ristrutturare il seguente schema



Vincoli esterni:

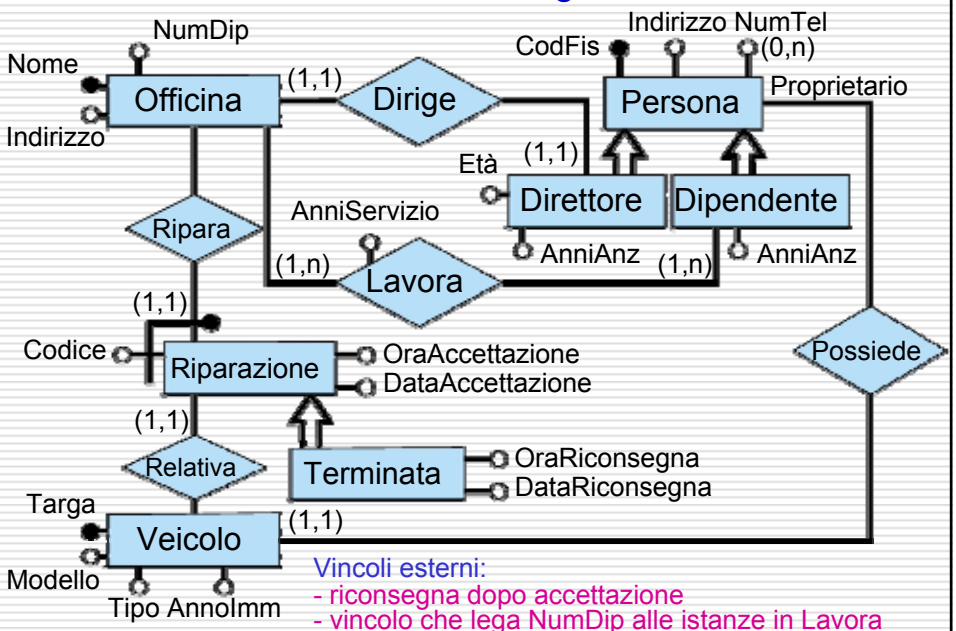
1) per ogni v in VoloCharter, se $(v,a_1),...,(v,a_n)$ sono tutte le coppie in Tappa alle quali partecipa v, e se $o_1,...,o_n$ sono i valori assegnati a tali coppie dall'attributo Ordine, allora per i=1,...,n esiste un o_j tale che o_j=i.
2) Un telefono è di una sola sede.

Esercizio 3: soluzione

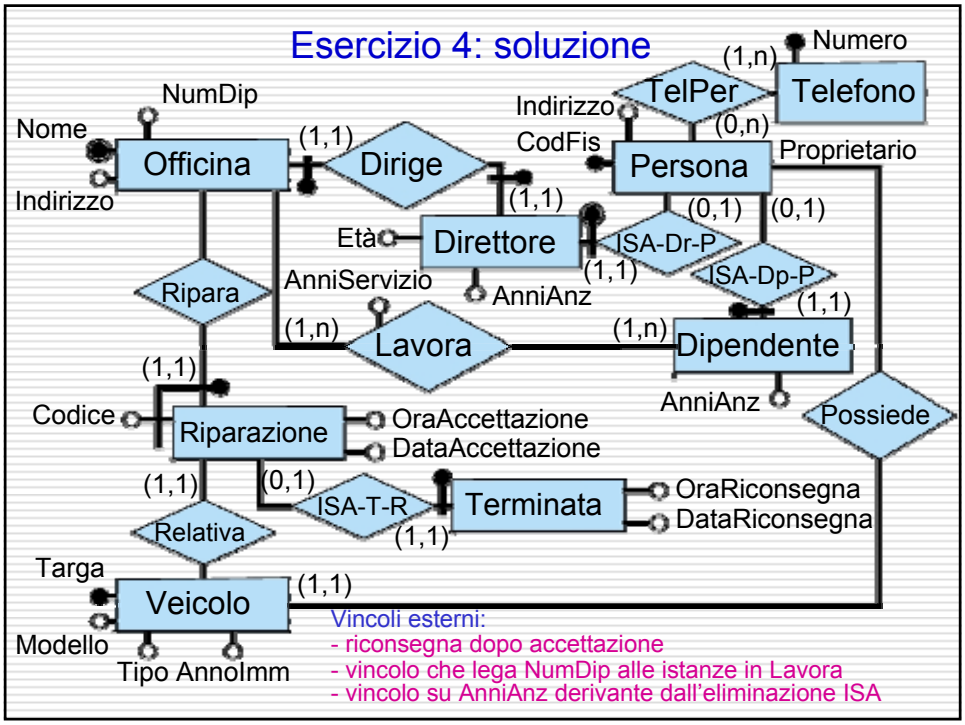


Vincolo esterno: 1) vincolo su ordine in Tappa (2 è diventato interno allo schema)

Esercizio 4: ristrutturare il seguente schema



Esercizio 4: soluzione



Elementi di Informatica LB

La progettazione logica

titolo:

3. Traduzione diretta del modello relazionale

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
- 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
- non ci sono cicli di identificazione esterna

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 generalmente richiede successive ristrutturazioni.

Consiste delle seguenti attività:

- traduzione delle entità in relazioni dello schema logico
- traduzione delle relazioni dello schema ER in relazioni dello schema logico
- traduzione dei vincoli esterni
- riformulazione di operazioni e specifiche sul carico applicativo in termini dello schema logico

Nota: quando è necessario fare distinzione tra le due accezioni di relazione, useremo il termine ER-relazione per denotare le relazioni dello schema ER.

Traduzione di entità: regole generali

- Un'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 RE quando:
  - un ruolo di Q partecipa 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);
  - E è l'unica entità che partecipa a Q con cardinalità (1,1).
- La chiave primaria di RE è determinata in base all'identificatore principale di E (attributi di E e/o attributi derivanti dall'identificazione esterna).
- Agli altri identificatori di E corrispondo dei vincoli di chiave su RE.

Traduzione di ER-relazioni: regole generali

- Una 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 con le entità partecipanti divengono in R_Q vincoli di foreign key verso le relazioni che corrispondono alle entità partecipanti.

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)
- 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 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.

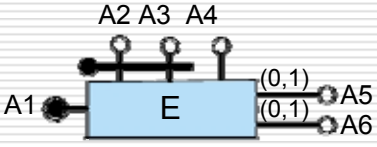
Riformulazione di operazioni e carico applicativo: regole generali

- 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 un'entità senza accorpamenti

Consideriamo per ora un'entità per cui non si effettuano accorpamenti di relazioni (ad esempio, se partecipa a relazioni solo con cardinalità massima n).

- 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 (tali attributi sono indicati con * nello schema logico).
  - altrimenti l'attributo non può assumere valore nullo
- L'identificatore principale dell'entità si traduce nella chiave primaria della relazione.
- Gli altri identificatori si traducono in chiavi della relazione.
- Ricordarsi dei vincoli esterni per identificatori opzionali correlati (derivanti da atttributi composti opzionali).

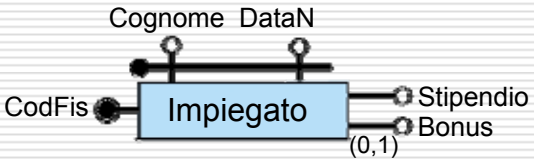


E(A1, A2, A3, A4, A5*, A6*)

chiave: A2, A3

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

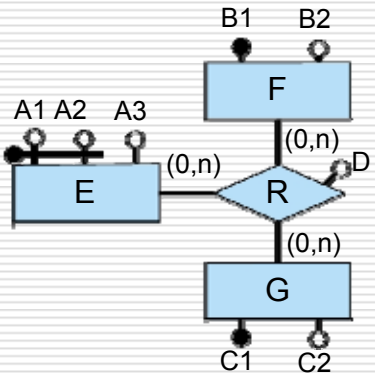
Traduzione di un'entità: esempio



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

Traduzione di una relazione non accorpata

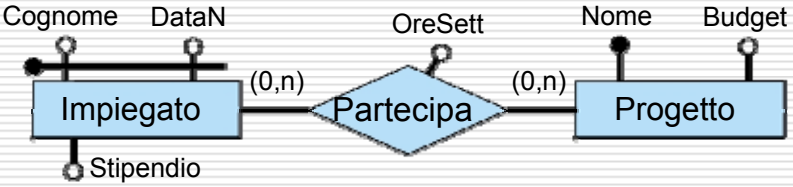
- Una ER-relazione che non è stata accorpata a nessuna entità si traduce in una relazione.
- Gli attributi della relazione sono quelli della ER-relazione, più le chiavi primarie delle relazioni corrispondenti alle entità partecipanti.
- Se nessuna cardinalità massima è 1, allora la chiave primaria della relazione è data dalle chiavi primarie delle entità partecipanti.
- Abbiamo vincoli di foreign key dalla relazione verso le entità partecipanti.



E(A1, A2, A3)
F(B1, B2)
G(C1, C2)
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 una 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).

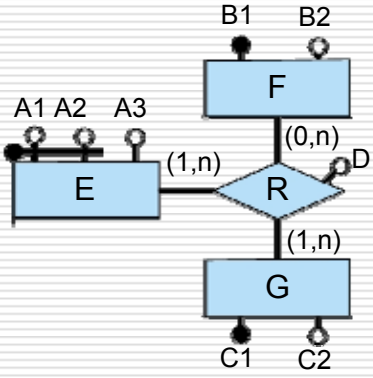


Impiegato(Cognome, DataN, Stipendio)
Progetto(Nome, Budget)
Partecipa(Cognome, DataN, Progetto, OreSett)

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

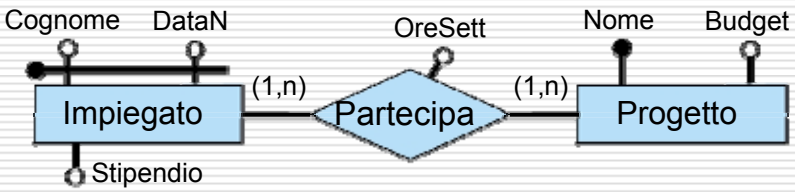
Relazioni con vincoli di cardinalità minima 1

- Un vincolo di cardinalità minima 1 per la partecipazione di un'entità ad una relazione (non accorpata nell'entità) si traduce in un vincolo di inclusione dall'entità verso la relazione.
- Si noti che questo vincolo di inclusione non è in generale un vincolo di 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]

Relazioni con vincoli di cardinalità (1,n): esempio



Impiegato(Cognome, DataN, Stipendio)

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

Progetto(Nome, Budget)

inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

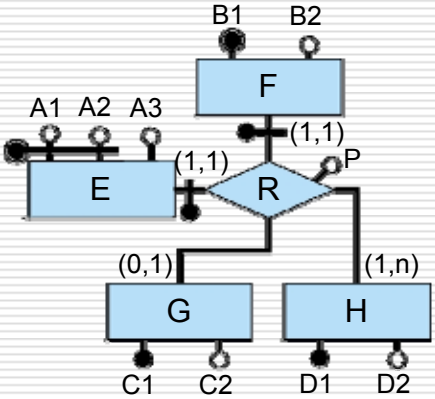
Partecipa(Cognome, DataN, Progetto, OreSett)

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

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

Relazioni con vincoli di cardinalità massima 1

- Se per una ER-relazione una entità partecipa con cardinalità massima 1 (e la relazione non è stata accorpata nell'entità), 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[A1,A2]

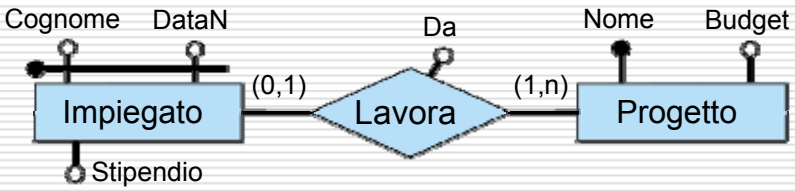
F(B1, B2)
foreign key: F[B1] ⊆ R[B1]

G(C1, C2)

H(D1, D2)
inclusione: H[D1] ⊆ R[D1]

R(A1, A2, B1, C1, D1, P)
foreign key: R[A1,A2] ⊆ E[A1,A2]
foreign key: R[B1] ⊆ F[B1]
foreign key: R[C1] ⊆ G[C1]
foreign key: R[D1] ⊆ H[D1]
chiave: A1, A2 chiave: B1

Relazioni con vincoli di cardinalità (0,1): esempio



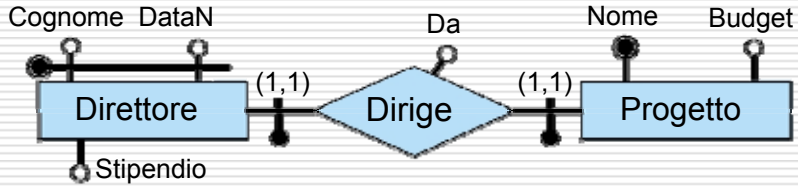
Impiegato(Cognome, DataN, Stipendio)
Progetto(Nome, Budget)

inclusione: Progetto[Nome] ⊆ Lavora[Progetto]

Lavora(Cognome, DataN, Progetto, Da)

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

Relazioni con vincoli di 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]

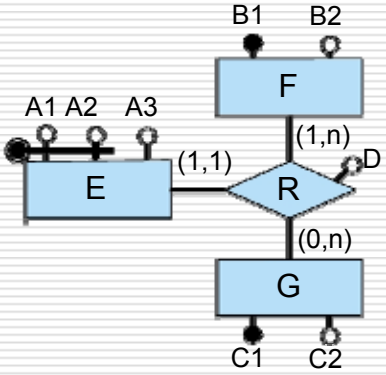
In alternativa:

Dirige(Cognome, DataN, Progetto, Da)

chiave: Progetto foreign key: ...

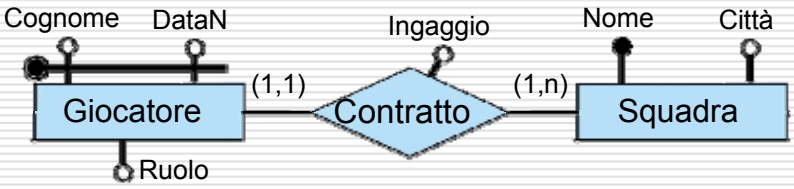
Accorpamento di relazione in entità: caso 1

- Se una ER-relazione ha un'unica entità che vi partecipa con cardinalità (1,1), la ER-relazione viene accorpata nell'entità. Questo significa che tutti gli attributi della ER-relazione e le chiavi primarie delle altre entità partecipanti diventano attributi della relazione che corrisponde all'entità.
- Devono venire aggiunti vincoli di foreign key dalla relazione accorpata verso le altre entità che partecipano alla relazione.
- Come per le relazioni non accorpate, un vincolo di cardinalità minima di 1 su un'altra entità che partecipa alla relazione si traduce in un vincolo di inclusione.



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

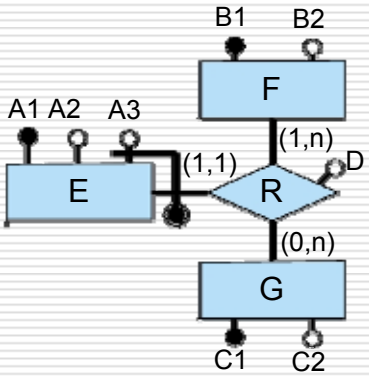
Accorpamento di relazione in entità (1): esempio



Giocatore(Cognome, DataN, Ruolo, Ingaggio, Squadra)
foreign key: Giocatore[Squadra] ⊆ Squadra[Nome]
Squadra(Nome, Città)
inclusione: Squadra[Nome] ⊆ Giocatore[Squadra]

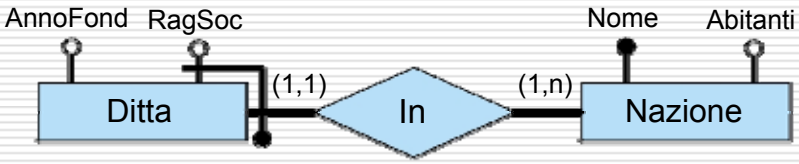
Accorpamento di relazione in entità: caso 2

- Se una ER-relazione è parte dell'identificatore principale esterno di un'entità, la ER-relazione viene accorpata nell'entità. Si noti che, per l'assenza di cicli di identificazione esterna, ci può essere al più un'entità per cui la ER-relazione è parte dell'identificatore principale esterno.
- 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, B1, C1, D)
foreign key: E[B1] ⊆ F[B1]
foreign key: E[C1] ⊆ G[C1]
chiave: A1
F(B1, B2)
inclusione: F[B1] ⊆ E[B1]
G(C1, C2)

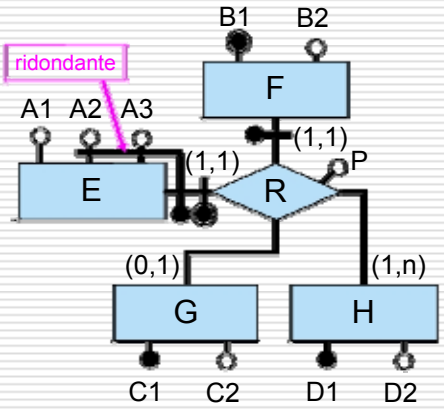
Accorpamento di relazione in entità (2): esempio



Ditta(RagSoc, Nazione, AnnoFond)
foreign key: Ditta[Nazione] ⊆ Nazione[Nome]
Nazione(Nome, Abitanti)
inclusione: Nazione[Nome] ⊆ Ditta[Nazione]

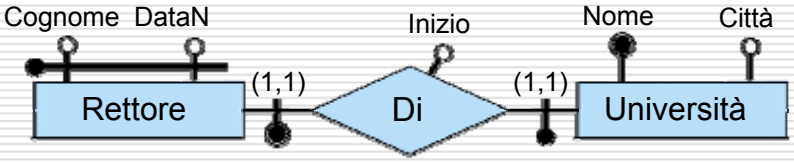
Accorpamento di relazione in entità: caso 2 bis

- Se una ER-relazione R è parte dell'identificatore principale esterno di un'entità E, allora R viene accorpata in E, anche se vi sono altre entità che partecipano ad R con cardinalità massima 1 (ad esempio F o G).
- In presenza di tali entità, le istanze di E sono identificate dalle istanze di R, e la chiave principale di E deve essere scelta tra le chiavi principali di tali entità (ad esempio quelle di F o G).



E(A1, A2, A3, B1, C1, D1, P)
foreign key: E[B1] ⊆ F[B1]
foreign key: E[C1] ⊆ G[C1]
foreign key: E[D1] ⊆ H[D1]
chiave: C1
F(B1, B2)
foreign key: F[B1] ⊆ E[B1]
G(C1, C2)
H(D1, D2)
inclusione: H[D1] ⊆ E[D1]

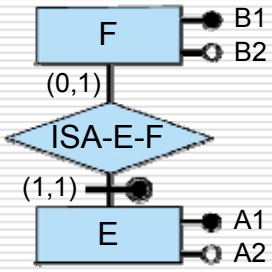
Accorpamento di relazione in entità (2): esempio



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 parte dell'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, e a rendere tali attributi anche chiave primaria di E (nel caso in cui per E si scelga come identificatore principale quello esterno sulla relazione ISA). Il vincolo derivante dall'ISA dello schema ER originario diventa quindi un vincolo di foreign key dello schema logico.

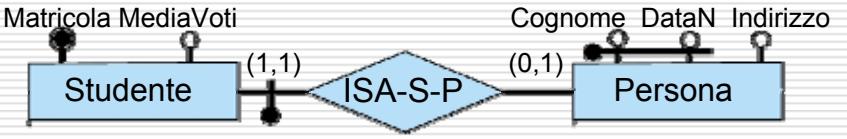
Relazione derivante da ISA: esempio

Studente(Matricola, MediaVoti, Cognome, DataN)

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

chiave: Matricola

Persona(Cognome, DataN, Indirizzo)



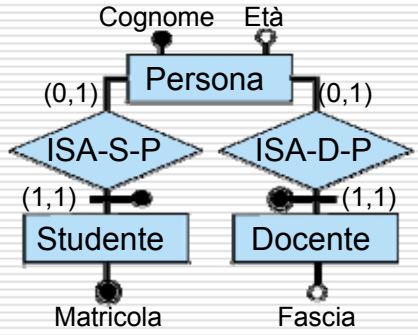
Studente(Matricola, MediaVoti, Cognome, DataN)

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

chiave: Cognome, DataN

Persona(Cognome, DataN, Indirizzo)

Traduzione di vincoli di generalizzazione

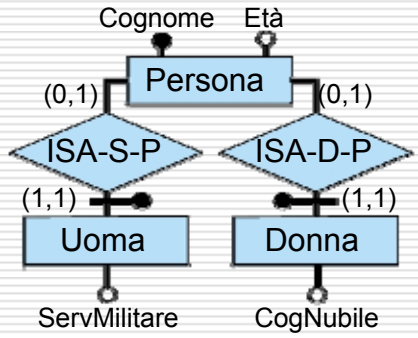


Vincolo di generalizzazione:
nessuna istanza di Persona partecipa sia a ISA-S-P sia a ISA-D-P

Diventa sullo schema logico:
Studente[Cognome] ∩ Docente[Cognome] = ∅

Persona(Cognome, Età)
Studente(Matricola, Cognome)
foreign key: Studente[Cognome] ⊆ Persona[Cognome]
chiave: Cognome
Docente(Cognome, Fascia)
foreign key: Docente[Cognome] ⊆ Persona[Cognome]

Traduzione di vincoli di generalizzazione completa



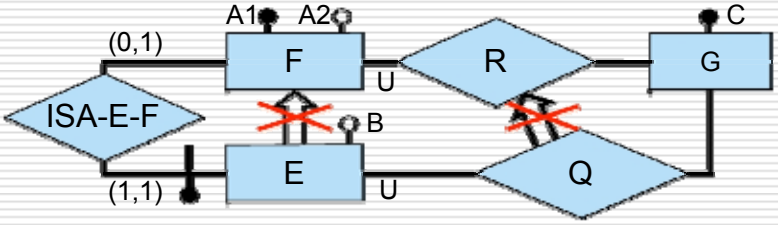
Vincolo di generalizzazione:
ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

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 vincoli derivanti da ISA tra relazioni

La ristrutturazione di un'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] ⊆ F[A1]

F(A1, A2) G(C)

R(A1, C) foreign key: R[A1] ⊆ F[A1], R[C] ⊆ G[C]

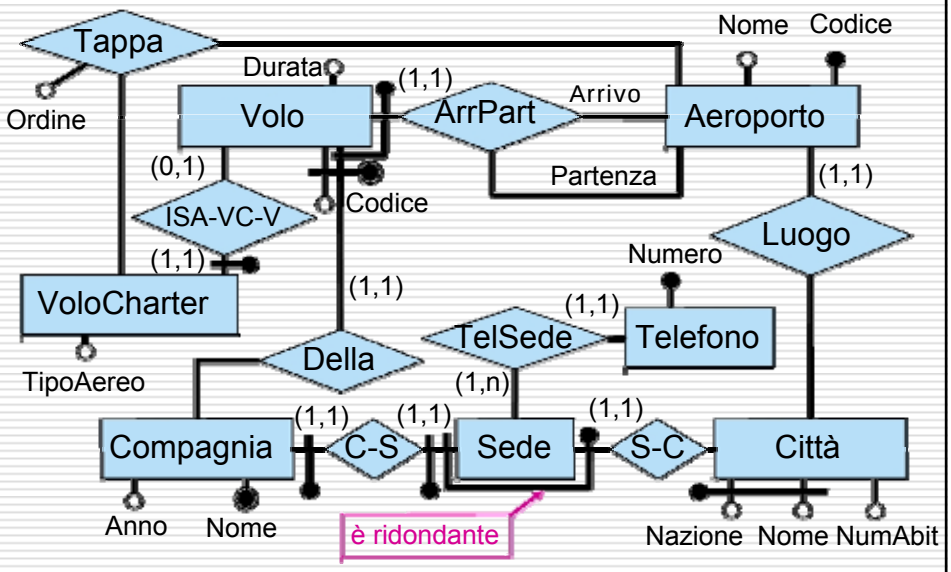
Q(A1, C) foreign key: Q[A1] ⊆ E[A1], Q[A1, C] ⊆ R[A1, C]

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 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 esterna, la relazione non può avere altre entità per le quali la relazione è parte di identificatore esterno)
   - come prima, per le relazioni per cui l'entità è l'unica con cardinalità (1,1)
2. Traduzione di ogni ER-relazione (non ancora accorpata al punto 1) in relazione, 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 (formulati come vincoli insiemistici)
   - 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

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

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

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

Città(Nome, Nazione, NumAbitanti)

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

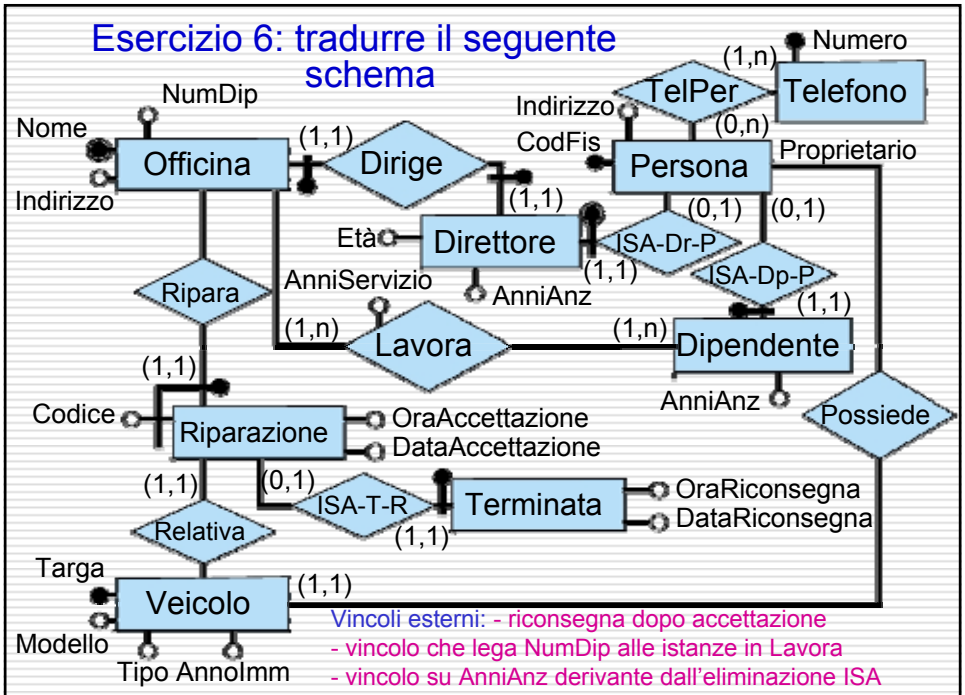
SedeCompagnia(Comp, NomeCittà, NazCittà)
- foreign key: SedeCompagnia[Comp] ⊆ Compagnia[Nome]
- foreign key: SedeCompagnia[NomeCittà, NazCittà] ⊆ Città[Nome, Nazione]
- inclusione: SedeCompagnia[Comp] ⊆ Telefono[Comp]

Telefono(Numero, Comp)
- foreign key: Telefono[Comp] ⊆ SedeCompagnia[Comp]

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



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, Eta, 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]

Esercizio 6: soluzione (parte 2)

TelPer(CodFis, Telefono)
foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
foreign key: TelPer[Telefono] ⊆ Telefono[Numero]
Telefono(Numero)
inclusione: Telefono[Numero] ⊆ TelPer[Telefono]
Veicolo(Targa, Modello, Tipo, AnnoImm, Proprietario)
foreign key : Veicolo[Proprietario] ⊆ Persona[CodFis]
Riparazione(Codice, Officina, OraAcc, DataAcc, Veicolo)
foreign key: Riparazione[Officina] ⊆ Officina[Nome]
foreign key: Riparazione[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 Direttore[AnniAnz] e Dipendente[AnniAnz] derivante dall'eliminazione ISA

Elementi di Informatica LB

La progettazione logica

titolo:

4. Ristrutturazione dello schema logico

Cosa sappiamo dopo la traduzione

- Abbiamo rispettato la modularizzazione concettuale (tranne che per l'accorpamento in unica entità con cardinalità (1,1)).
- Si possono presentare potenziali problemi di 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 $R_1[K_1] \equiv R_2[K_2]$
  - ridondanze lasciate

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

Modello di costo

- Il numero di accessi a 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
- $N_T(R)$ il numero di tuple di R
- $N_P(R)$ il numero di pagine in memoria secondaria occupate da R

Costo delle operazioni:
- proiezione di una relazione R: costo pari a $N_P(R)$
- join di R con Q: si basa su un doppio ciclo:

  per ogni tupla r di R
  per ogni tupla q di Q
  se r e q sono in join, metti la tupla nel risultato
  quando Q ha un indice sull'attributo di join il costo è $N_P(R) + N_T(R)$

Criteri generali per individuare potenziali problemi

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

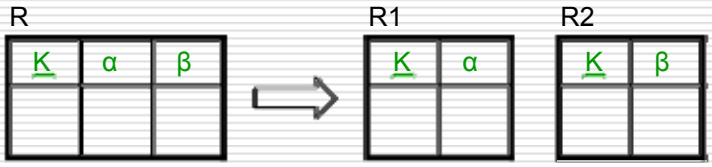
Ristrutturazioni dello schema logico

- decomposizione
  - verticale: sempre sulla chiave
    - per facilitare l'accesso (con selezioni e proiezioni)
    - per normalizzazione
  - orizzontale
    - per evitare valori nulli (se numerosi o interdipendenti)
    - per facilitare l'accesso
  - mista
    - per evitare valori nulli
- accorpamento: per facilitare l'accesso
  - forte
  - debole

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

Nota: le ristrutturazioni si applicano in presenza di determinati attributi che sono chiave (eventualmente non primaria) di relazioni. Sulle slide indicheremo tali attributi con K, intendendo che K è una chiave (primaria o non) della relazione corrispondente.

Decomposizione verticale



Vincoli dello schema ristrutturato:

foreign key: R1[K] ⊆ R2[K]

foreign key: R2[K] ⊆ R1[K]

- vincoli di inclusione da e a R si suddividono su R1 e R2
- tutti gli altri vincoli che coinvolgono R vanno riformulati

- Si applica quando gli accessi ad R avvengono separatamente sugli attributi di α e su quelli di β.
- La relazione R può venire ricostruita attraverso una vista che calcola il join tra R1ed R2 su K.

Decomposizione orizzontale per evitare valori nulli

R

K	α*	β
k1	a	b1
k2	NULL	b2

R1

K	α	β
k1	a	b1

R2

K	β
k2	b2

Vincoli dello schema ristrutturato:

- ciascuna chiave di R (in particolare K) diventa chiave di R1 e R2, e i suoi valori in R1 e R2 sono disgiunti. Per K: $R1[K] \cap R2[K] = \varnothing$
- 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 vi sono molti valori nulli, possibilmente interdipendenti.
- La relazione R può venire ricostruita attraverso una vista che calcola l'unione di R1ed R2.

Decomposizione orizzontale per facilitare l'accesso

Q

K2	β
k1	b

R

K1	α
k1	a1
k2	a2

foreign key: $Q[K2] \subseteq R[K1]$

Q

K2	β
k1	b

R1

K1	α
k1	a1

$Q[K2] \subseteq R1[K1]$

$R1[K1] \subseteq Q[K2]$

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

R2

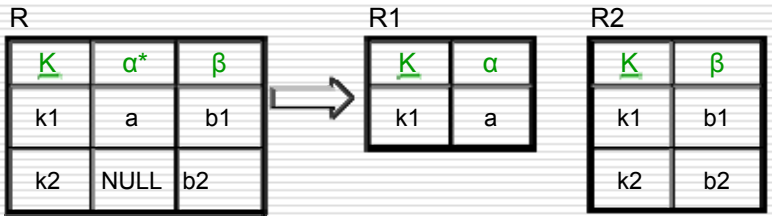
K1	α
k2	a2

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 che corrispondono a tuple di Q avvengono separatamente dagli accessi alle altre tuple di R.
- La relazione R può venire ricostruita attraverso una vista che calcola l'unione di R1ed R2.

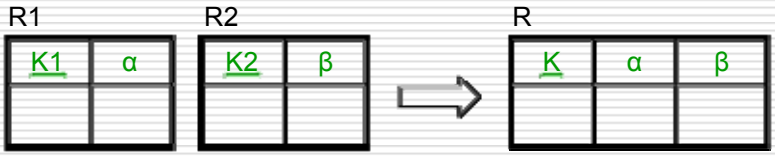
Decomposizione mista



Vincoli dello schema ristrutturato:
- foreign key: R1[K] ⊆ R2[K]
- vincoli di inclusione da e a R diventano inclusioni da e a R2
- tutti gli altri vincoli che coinvolgono R diventano vincoli che coinvolgono R2

- La relazione R può venire ricostruita attraverso una vista che calcola il join esterno tra R1ed R2 su K.

Accorpamento forte



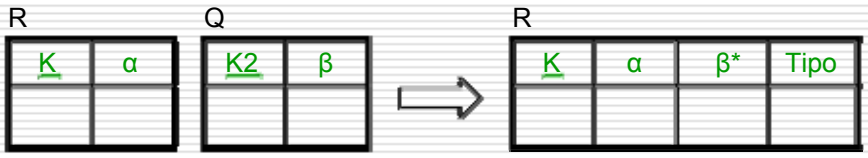
foreign key: R1[K1] ⊆ R2[K2]
foreign key: R2[K2] ⊆ R1[K1]

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.

- Le relazioni R1 e R2 possono venire ricostruite attraverso due viste che calcolano rispettivamente le proiezioni di R su (K, α) e su (K, β).

Accorpamento debole - caso 1



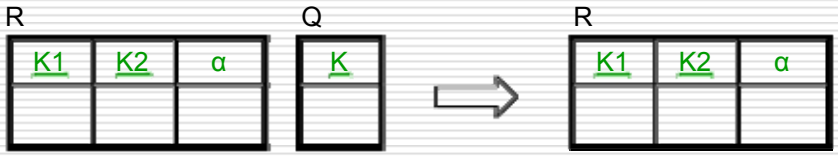
foreign key: Q[K2] ⊆ R[K]

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono R o Q vanno riformulati su R

- Si applica per facilitare gli accessi a Q quando questi avvengono prevalentemente insieme ad accessi a R e richiedono di calcolare il join tra R e Q con K1=K2.
- L'attributo Tipo serve per distinguere le tuple di R che corrisponderebbero a tuple di Q dalle altre. Può essere omesso se β non può assumere valori nulli.
- Le relazioni R e Q originarie possono venire ricostruite attraverso due viste che effettuano prima una opportuna selezione su R in base a Tipo, e proiettano poi il risultato rispettivamente su (K, α) e su (K, β).

Accorpamento debole - caso 2



foreign key: R[K1] ⊆ Q[K]
inclusione: Q[K] ⊆ R[K1]

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono Q vanno riformulati su R

- Si applica quando Q è composta solo dalla chiave, per ottimizzare l'occupazione di spazio ed eliminare una relazione ricostruibile.
- Si noti che il vincolo di inclusione Q[K] ⊆ R[K1] non è un vincolo di foreign key in quanto K1 è solo parte di chiave.
- Q non è necessaria in quanto può essere ricostruita attraverso una proiezione di R su K1.

Traduzione dei vincoli in SQL

- 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: ristrutturare il seguente schema ...

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

Vincolo esterno: vincolo su ordine in Tappa.

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

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

Città(Nome, Nazione, NumAbitanti)

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

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

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

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

... tenendo conto delle seguenti specifiche

- Non ci devono essere valori nulli.
- Si accede spesso per conoscere tutte le proprietà di un volo charter.
- Quando si accede alla compagnia si accede anche ai dati relativi alla sua sede.

La relazione Tappa deve essere accorpata in VoloCharter? Perché?

Esercizio 7: soluzione - ristrutturazioni

- Non ci devono essere valori nulli. ->
  - non si possono fare accorpamenti deboli perché in questo caso introdurrebbero valori nulli
- Si accede spesso per conoscere tutte le proprietà di un volo charter. ->
  - decomposizione orizzontale di Volo in
    - VoloNonCharter
    - DatiVoloCharter
  - accorpamento forte di DatiVoloCharter e VoloCharter
- Quando si accede alla compagnia si accede anche ai dati relativi alla sua sede. ->
  - accorpamento forte di Compagnia e SedeCompagnia

La metodologia non consente di accorpare Tappa in VoloCharter. Sarebbe infatti sbagliato, perché porterebbe ad uno schema con ridondanze estensionali.

Esercizio 7: 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, NomeCittà, NazCittà)
foreign key: Aeroporto[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] ⊆ Telefono[Comp]

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

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

Esercizio 7: 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 esterno: vincolo su ordine in Tappa

Viste per ricostruire le relazioni dello schema originario:

view Volo = $PROJ_{Codice,Comp,Durata,Arrivo,Partenza}$(VoloNonCharter) ∪
$PROJ_{Codice,Comp,Durata,Arrivo,Partenza}$(VoloCharter)

view SedeCompagnia = $PROJ_{Nome, NomeCittà, NazCittà}$ (Compagnia)
view CompagniaOrig = $PROJ_{Nome, AnnoFond}$ (Compagnia)

Esercizio 8: ristrutturare il seguente schema ...

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

Esercizio 8: schema (cont.) ...

TelPer(CodFis, Telefono)
foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
foreign key: TelPer[Telefono] ⊆ Telefono[Numero]
Telefono(Numero)
inclusione: Telefono[Numero] ⊆ TelPer[Telefono]
Veicolo(Targa, Modello, Tipo, AnnoImm, Proprietario)
foreign key : Veicolo[Proprietario] ⊆ Persona[CodFis]
Riparazione(Codice, Officina, OraAcc, DataAcc, Veicolo)
foreign key: Riparazione[Officina] ⊆ Officina[Nome]
foreign key: Riparazione[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 Direttore[AnniAnz] e Dipendente[AnniAnz] 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 tutti i dati relativi al loro direttore.
- Solitamente non interessano i dati anagrafici dei direttori.
- Quando si accede agli impiegati interessano anche i dati anagrafici.
- Un'operazione frequente è la stampa dell'elenco di tutte le riparazioni (terminate e non), con officina, autoveicolo e ora e data di accettazione ed eventuale riconsegna.

Esercizio 8: 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 forte di Direttore, Dirige e Officina
- Solitamente non interessano invece i dati anagrafici dei direttori. ->
  - non c'è partizionamento orizzontale di Persona in direttori e non
- Quando si accede agli impiegati interessano anche i dati anagrafici.
  - partizionamento orizzontale di Persona in dipendenti e non
  - accorpamento forte tra l'entità risultante e Dipendente
- Un'operazione frequente è la stampa dell'elenco di tutte le riparazioni (terminate e non), con officina, autoveicolo e ora e data di accettazione ed eventuale ora e data di riconsegna. ->
  - accorpamento debole di Terminata in Riparazione
- Accorpamento debole di Telefono in TelPer allo scopo di eliminare una relazione.

Esercizio 8: soluzione - schema ristrutturato

Officina(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] ⊆ Officina[Nome]
foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]
TelPer(CodFis, Telefono)
foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
Veicolo(Targa, Modello, Tipo, AnnoImm, Proprietario)
Riparazione(Codice, Officina, OraAcc, DataAcc, Veicolo, OraRic*, DataRic*)
foreign key: Riparazione[Officina] ⊆ Officina[Nome]
foreign key: Riparazione[Veicolo] ⊆ Veicolo[Targa]

Esercizio 8: soluzione - vincoli e viste

Vincoli:

- PersonaNonDip e Dipendente sono disgiunti:
  - PersonaNonDip[CodFis] ∩ Dipendente[CodFis] = ∅
- Vincoli risultanti dai vincolo di foreign key verso Persona
  - Officina[Direttore] ⊆ PersonaNonDip[CodFis] ∪ Dipendente[CodFis]
  - Veicolo[Proprietario] ⊆ PersonaNonDip[CodFis] ∪ Dipendente[CodFis]
  - TelPer[CodFis] ⊆ PersonaNonDip[CodFis] ∪ Dipendente[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 = PersonaNonDip ∪ $PROJ_{CodFis,Indirizzo}$(Dipendente)
view OfficinaOrig = $PROJ_{Nome, NumDip, Indirizzo}$(Officina)
view Direttore = $PROJ_{Direttore, EtaDir, AnniAnzDir}$ (Officina)
view Dirige = $PROJ_{Nome, Direttore}$(Officina)
view Terminata = $PROJ_{Codice, Officina, OraRic, DataRic}$(SEL $_{OraRic\ NOT\ NULL}$(Riparazione))
view RiparazioneOrig = $PROJ_{Codice, Officina, OraAcc, DataAcc, Veicolo}$(Riparazione)
view Telefono = $PROJ_{Telefono}$(TelPer)