

DATABASE

MODELLO RELAZIONALE

Una relazione matematica sugli insiemi D_1, D_2, \dots, D_n è un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \dots \times D_n$, quindi:

- Il numero di n-uple corrisponde alla cardinalità della relazione
- Essendo un insieme non vi è ordinamento ed ogni n-upla è diversa dall'altra, il sistema però è posizionale

Nel modello relazionale ogni relazione è una tabella nella quale ogni componente rappresenta un attributo, ognuno dei quali è rappresentato da un nome ed un insieme di valori (detto dominio dell'attributo).

- Esempio:

Casa	Fuori	RetiCasa	RetiFuori
Juve	Lazio	3	1
Lazio	Milan	2	0
Juve	Roma	1	2
Roma	Milan	0	1

notazione:

$t[A]$ (oppure $t.A$) => valore della tupla t in corrispondenza dell'attributo A

OSS non esistono distinzioni tra i tipi di valori NULL, possono indicare: valori sconosciuti, inesistenti o la semplice mancanza di informazioni

Vincoli di integrità

Un vincolo di integrità è una condizione che si esprime a livello di schema e si intende debba essere soddisfatta da tutte le istanze della base di dati, poiché individua una condizione necessaria per tutte quelle istanze della base di dati che rappresentano situazioni corrette per l'applicazione

Vincoli intrarelazionali

- **Vincolo di tupla** => esprimono condizioni sui valori di ciascuna tupla di una relazione, indipendentemente dalle altre tuple.
- **Vincoli di chiave** => specifica che un insieme non vuoto di attributi è chiave della relazione

Le chiavi

Una chiave di una relazione è un insieme non vuoto di attributi che identificano le ennupla di una relazione, può essere:

- **superchiave** => insieme di attributi che formano una chiave per la relazione, usata nel caso in cui ogni attributo abbia delle ripetizioni (ci possono essere più superchiavi nella stessa relazione)
- **Chiave** => superchiave minimale (la superchiave con il minor numero di attributi)
- **Chiave primaria** => chiave senza valori NULL

Vincoli interrelazionali

- vincoli di integrità referenziale (foreign key) => informazioni in relazioni differenti sono correlate attraverso valori comuni, in particolare, attraverso valori delle chiavi, ovvero:

Una foreign key fra una sequenza non vuota X di n attributi di una relazione R₁ ed una sequenza Y di n attributi che formano una chiave di una relazione R₂ impone che ogni combinazione di valori su X presenti in R₁ compaia come combinazione di valori su Y in R₂

OSS generalizzato diventa vincolo di inclusione

Infrazioni				
Codice	Data	Vigile	Prov	Numero
34321	1/2/95	3987	MI	39548K
53524	4/3/95	3295	TO	E39548
64521	5/4/96	3295	PR	839548
73321	5/2/98	9345	PR	839548

Auto	Prov	Numero	Cognome	Nome
	MI	39548K	Rossi	Mario
	TO	E39548	Rossi	Mario
	PR	839548	Neri	Luca

In pratica ad ogni valore della tabella in un determinato attributo corrisponde lo stesso valore nell'altra tabella, se il valore non è presente si ha violazione del vincolo

Algebra relazionale

Insieme di valori ed operazioni chiusi nell'insieme

Operatori insiemistici

A livello estensionale le relazioni sono insiemi di tuple, quindi si possono usare:

- **UNIONE** \cup => tabella risultante ha tutte le tuple delle relazioni, tenendo conto che, essendo un insieme, le tuple in comune si prendono una volta sola
- **INTERSEZIONE** \cap => tabella risultante ha solo le tuple in comune tra le relazioni
- **DIFFERENZA** — => tabella risultante ha solo le tuple della prima relazione che non sono contenute nella seconda relazione

ATTENZIONE applicabili solo con relazioni con definizioni uguali (stessi attributi)

Unione			Intersezione			Differenza																																						
Laureato			Quadro			Laureato																																						
Laureato	Matricola	Nome	Età	Matricola	Nome	Età	Matricola	Nome																																				
7274	Rossi	42	9297	Neri	33	7274	Rossi	42																																				
7432	Neri	54	7432	Neri	54	7432	Neri	54																																				
9824	Verdi	45	9824	Verdi	45	9824	Verdi	45																																				
Laureato \cup Quadro			Laureato \cap Quadro			Laureato – Quadro																																						
<table border="1"> <tr><td>Matricola</td><td>Nome</td><td>Età</td></tr> <tr><td>7274</td><td>Rossi</td><td>42</td></tr> <tr><td>7432</td><td>Neri</td><td>54</td></tr> <tr><td>9824</td><td>Verdi</td><td>45</td></tr> <tr><td>9297</td><td>Neri</td><td>33</td></tr> </table>			Matricola	Nome	Età	7274	Rossi	42	7432	Neri	54	9824	Verdi	45	9297	Neri	33	<table border="1"> <tr><td>Matricola</td><td>Nome</td><td>Età</td></tr> <tr><td>7432</td><td>Neri</td><td>54</td></tr> <tr><td>9824</td><td>Verdi</td><td>45</td></tr> </table>			Matricola	Nome	Età	7432	Neri	54	9824	Verdi	45	<table border="1"> <tr><td>Matricola</td><td>Nome</td><td>Età</td></tr> <tr><td>7274</td><td>Rossi</td><td>42</td></tr> <tr><td>7432</td><td>Neri</td><td>54</td></tr> <tr><td>9824</td><td>Verdi</td><td>45</td></tr> </table>			Matricola	Nome	Età	7274	Rossi	42	7432	Neri	54	9824	Verdi	45
Matricola	Nome	Età																																										
7274	Rossi	42																																										
7432	Neri	54																																										
9824	Verdi	45																																										
9297	Neri	33																																										
Matricola	Nome	Età																																										
7432	Neri	54																																										
9824	Verdi	45																																										
Matricola	Nome	Età																																										
7274	Rossi	42																																										
7432	Neri	54																																										
9824	Verdi	45																																										

Ridenominazione

Lo schema della relazione “operando” viene modificato sostituendo il nome A_i di ogni attributo con il corrispettivo nuovo nome B_i .

OSS permette quindi operazioni insiemistiche su schemi apparentemente diversi.

Sintassi

$\text{REN}_{A_1 \leftarrow B_1, A_2 \leftarrow B_2, \dots, A_n \leftarrow B_n} (\text{Operando})$

o anche

$\text{REN}_{A_1, A_2, \dots, A_n \leftarrow B_1, B_2, \dots, B_n} (\text{Operando})$

Impiegati	Cognome	Ufficio	Stipendio
	Rossi	Roma	55
	Neri	Milano	64

Operai	Cognome	Fabbrica	Salario
	Bruni	Monza	45
	Verdi	Latina	55

$\text{REN}_{\text{Sede}, \text{Retribuzione} \leftarrow \text{Ufficio}, \text{Stipendio}} (\text{Impiegati})$

\cup

$\text{REN}_{\text{Sede}, \text{Retribuzione} \leftarrow \text{Fabbrica}, \text{Salario}} (\text{Operai})$

Cognome	Sede	Retribuzione
Rossi	Roma	55
Neri	Milano	64
Bruni	Monza	45
Verdi	Latina	55

Selezione

Permette di selezionare solo le tuple di “operando” che rispettano “condizione”, condizione corrisponde ad un’espressione booleana.

Sintassi:

$\text{SEL}_{\text{Condizione}} (\text{Operando})$

Impiegato			
Matricola	Cognome	Filiale	Stipendio
7309	Rossi	Roma	55
5998	Neri	Milano	64
5698	Neri	Napoli	64

Impiegato			
Matricola	Cognome	Filiale	Stipendio
5998	Neri	Milano	64

$\text{SEL}_{\text{Stipendio} > 50 \text{ AND } \text{Filiale} = \text{'Milano'}} (\text{Impiegato})$

Proiezione

Permette di proiettare solo gli attributi di “operando” contenuti in “listaAttributi”, il numero di tuple rimane inalterato.

Sintassi

$\text{PROJ}_{\text{ListaAttributi}} (\text{Operando})$

Impiegato			
Matricola	Cognome	Filiale	Stipendio
7309	Neri	Napoli	55
5998	Neri	Milano	64
9553	Rossi	Roma	44
5698	Rossi	Roma	64

Impiegato	
Cognome	Filiale
Neri	Napoli
Neri	Milano
Rossi	Roma

$\text{PROJ}_{\text{Cognome}, \text{Filiale}} (\text{Impiegato})$

OSS selezione e proiezioni sono operazioni ortogonali

Join

Permette di correlare dati tra relazioni differenti, può essere:

- **JOIN NATURALE** => la tabella risultante contiene le tuple aventi i valori di uno o più attributi in comune nelle 2 relazioni, ovvero:

R₁ JOIN R₂ è una relazione su X₁X₂ (X₁UX₂) il cui insieme di ennupla è:

$$\{ t \text{ su } X_1X_2 | \text{ esistono } t_1 \in R_1 \text{ e } t_2 \in R_2 \text{ tali che } t[X_1] = t_1 \text{ e } t[X_2] = t_2 \}$$

OSS cardinalità => $0 \leq |R_1 \text{ JOIN } R_2| \leq |R_1| \times |R_2|$

Sintassi

R₁ JOIN R₂

Docente	Corso	Corso	Ling
1	BD	BD	SQL
2	PS	BD	Java
3	Reti	PS	Java
1	PS	PS	UML

Docente	Corso	Ling
1	BD	SQL
1	BD	Java
1	PS	Java
1	PS	UML
2	PS	Java
2	PS	UML

definizioni:

- join completo => ogni ennupla di entrambe le relazioni contribuisce al risultato
- Join non completo => una o più ennupla non contribuiscono al risultato
- Join vuoto => nessuna ennupla contribuisce al risultato

ATTENZIONE se ho tutti attributi diversi nelle 2 relazioni il risultato è prodotto cartesiano

- **THETA JOIN** => combinazione del JOIN naturale e dell'operazione di selezione, indispensabile per relazioni non aventi attributi in comune
- OSS con condizione di uguaglianza “=” prende il nome di EQUI-JOIN

Sintassi

R₁ JOIN_{condizione} R₂

Impiegato		Reparto	
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B	B	Bruni

Impiegato JOIN _{Reparto=Codice} Reparto			
Impiegato	Reparto	Codice	Capo
Rossi	A	A	Mori
Neri	B	B	Bruni
Bianchi	B	B	Bruni

- **JOIN ESTERNO** => permette di mantenere ennuple che verrebbero scartate in caso di JOIN estendendole con valori nulli, può essere:
 - Sinistro => mantiene solo ennuple in più dell'operando sinistro
 - Destro => mantiene solo ennuple in più dell'operando destro
 - Completo => mantiene le ennuple in più di entrambi gli operandi

Join esterno sinistro		Join esterno destro	
Impiegato	Reparto	Impiegato	Reparto
Rossi	A	Rossi	A
Neri	B	Neri	B
Bianchi	B	Bianchi	B

Impiegato JOIN _{LEFT} Reparto		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL

Impiegato JOIN _{RIGHT} Reparto		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni

Join esterno completo			
Impiegato	Reparto	Reparto	Capo
Rossi	A	B	Mori
Neri	B	C	Bruni
Bianchi	B		

Impiegato JOIN _{FULL} Reparto		
Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL
NULL	C	Bruni

Extra NULL

Nell'algebra relazionale le condizioni valgono solo per i valori non nulli, quindi nel caso di una condizione su un attributo avente valori null, questi verranno sicuramente scartati.

Si ha quindi:

$$SEL_{attributo > numero}(tabella) \cup SEL_{attributo \leq numero}(tabella) \neq tabella$$

Per riferirsi ad i valori null ed eventualmente includerli si usano le condizioni:

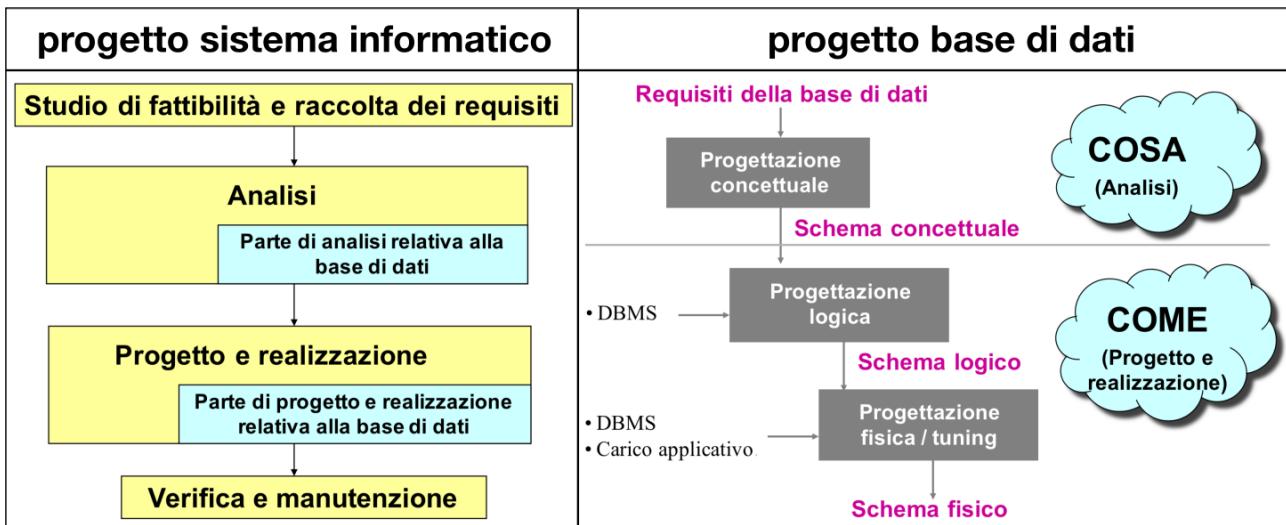
- **IS NULL**
- **IS NOT NULL**

Impiegato				Impiegato			
Matricola	Cognome	Filiale	Età	Matricola	Cognome	Filiale	Età
7309	Rossi	Roma	32				
5998	Neri	Milano	45				
9553	Bruni	Milano	NULL				

$SEL_{Età > 40} (\text{Impiegato})$	$SEL_{(Età > 40) \text{ OR } (Età IS NULL)} (\text{Impiegato})$
-------------------------------------	-----------------------------------------------------------------

OSS due espressioni si dicono equivalenti se producono lo stesso risultato qualunque sia l'istanza della base di dati sulla quale vengono valutate

PROGETTAZIONE CONCETTUALE



Modello entità-relazione (ER)

Modello concettuale di dati utilizzato per i database, diviso in:

- **Livello intensionale** => schema concettuale, ne descrive la struttura (tabelle)
- **Livello estensionale** => istanze dello schema (tuple che compongono la tabella)

Entità

Una entità è una classe di oggetti con esistenza autonoma e proprietà comuni.

Semantica: Dato uno schema S in cui è definita un entità E, in ogni istanza I dello schema S è associato un insieme di oggetti {e1, e2, e3, ... } detto estensione di E nella istanza I dello schema S (*istanze(I,E)*)

Sintassi:

NomeEntità

Impiegato

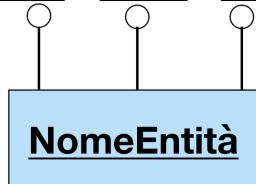
Attributo di entità

Un attributo di entità è una proprietà locale dell'entità, associa ad ogni sua istanza un valore appartenente ad un insieme, detto dominio dell'attributo.

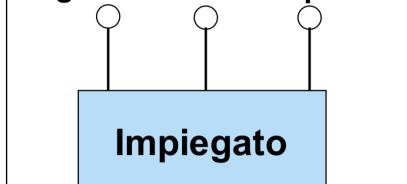
Semantica: data l'istanza di uno schema S dove A è un attributo dell'entità E su un dominio D, *istanze(I,A)*, è un insieme di coppie (x,y) tali che x è in *istanze(I,E)*, y è in D, ed esiste una ed una sola coppia in *istanze(I,A)* per ogni x in *istanze(I,E)*.

Ovvero $A : \text{istanze}(I, E) \rightarrow D$

Sintassi: Attr. 1 Attr. 2 Attr. 3

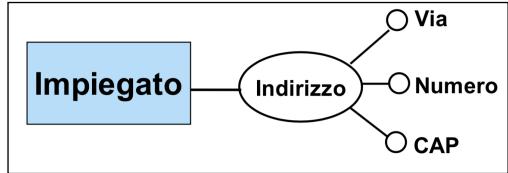


Cognome Età Stipendio



ATTENZIONE attributo è una funzione totale, quindi ad ogni entità deve essere associato 1 ed un solo valore del dominio

OSS l'attributo può anche essere definito su un dominio complesso e quindi diventare composto



Relazione

Rappresenta un legame tra due o più entità su cui è definita, il numero di entità coinvolte nella relazione ne determina il grado.

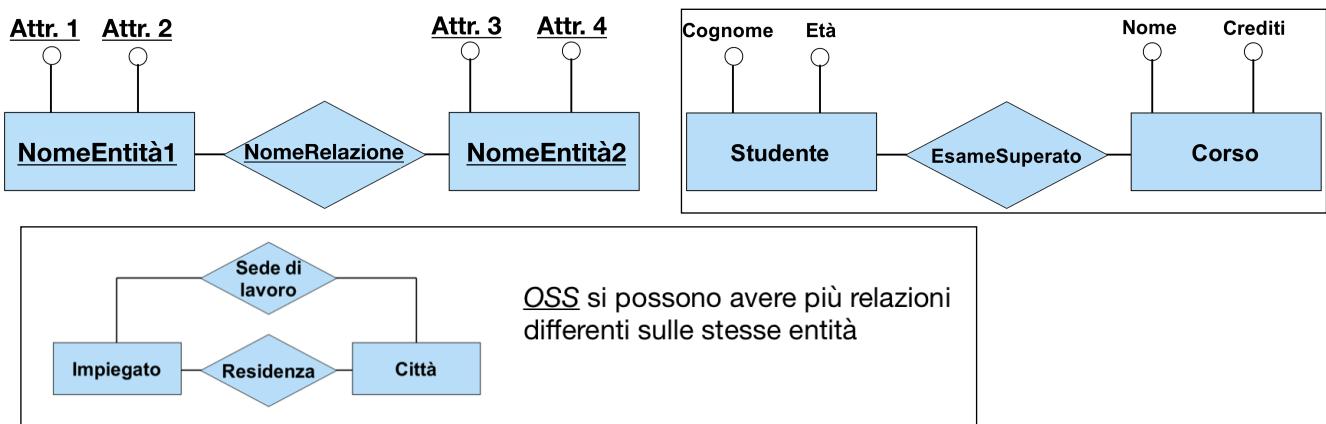
Semantica: Dato uno schema S in cui è definita una relazione R sulle entità E e F, in ogni istanza I di S, alla relazione R è associato un insieme di coppie $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots\}$ detto estensione di R nella istanza I dello schema S $istanze(I, R)$.

Ovvero: $istanze(I, R) \subseteq istanze(I, E) \times istanze(I, F)$

Nella versione n-aria diventa: $istanze(I, R) \subseteq istanze(I, E_1) \times \dots \times istanze(I, E_n)$

ATTENZIONE quindi non possono esistere 2 istanze di R che coinvolgono le stesse istanze di entità (tutte le coppie riferite alla stessa relazione sono diverse)

Sintassi:



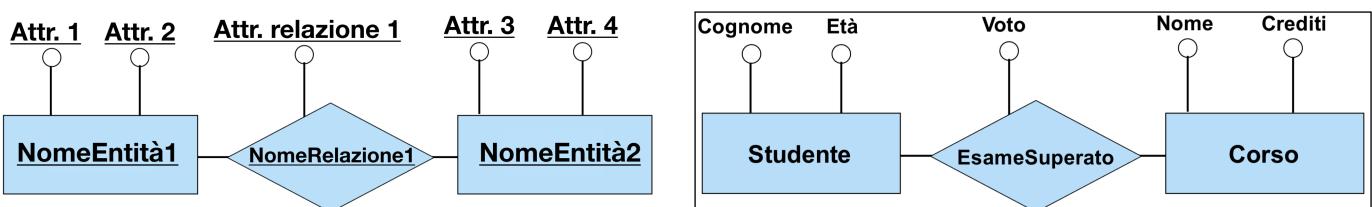
Attributo di relazione

Un attributo di relazione è una proprietà locale della relazione, ovvero rappresenta un attributo appartenente al legame tra le relazioni ma non alle relazioni stesse, questo avviene associando ad ogni istanza di relazione un valore appartenente ad un insieme detto dominio dell'attributo.

Semantica: data l'istanza dello schema S dove A è un attributo della relazione R sul dominio D, $istanze(I, A)$ è un insieme di coppie (x, y) tale che x è in $istanze(I, R)$, y è in D, ed esiste una ed una sola coppia in $istanze(I, A)$ per ogni x in $istanze(I, R)$.

Ovvero $A : istanze(I, R) \rightarrow D$

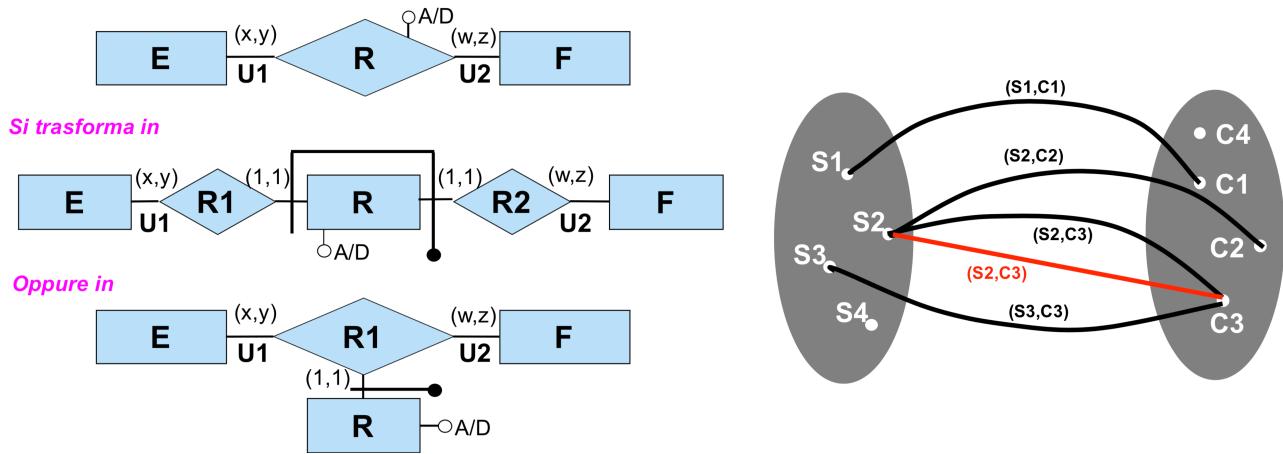
Sintassi:



ATTENZIONE l'attributo non è parte intrinseca della natura della relazione e quindi non può distinguerne le istanze (serve reificazione)

La reificazione

Consiste nel trasformare una relazione in una entità, così facendo è possibile trasformare il semplice attributo di relazione in un elemento in grado di distinguere le istanze. Così non si hanno più 2 coppie uguali ma una tripla comprendente l'attributo reificato.

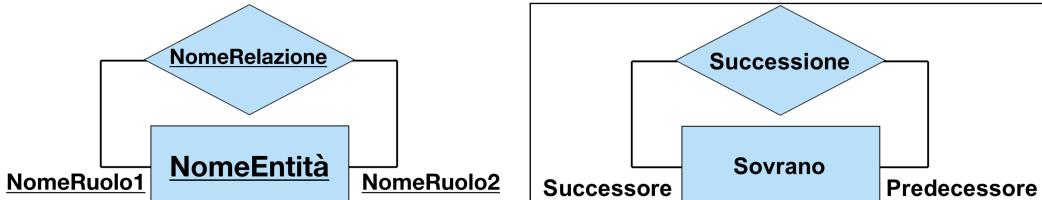


Il ruolo

Serve nel caso di relazione che coinvolge più volte le stesse entità, permette di specificare quale ruolo ha ogni istanza nella relazione.

Semantica: In ogni istanza I dello schema S, una relazione R tra le entità E₁, E₂, ..., E_n (non necessariamente tutte distinte) con rispettivi ruoli U₁, U₂, ..., U_n (tutti distinti) è costituita da un insieme *istanze*(I,R) in cui ogni elemento è una n-pla etichettata (U₁:x₁, U₂:x₂, ..., U_n:x_n) tale che ogni x_i appartiene a *istanze*(I,E_i), con i da 1 a n.

Sintassi:



Relazione ISA

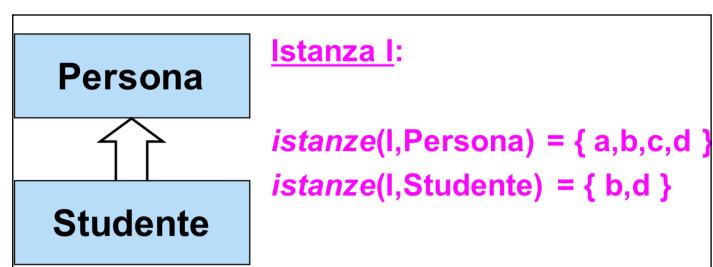
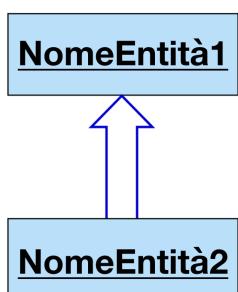
Relazione di sottoinsieme, cioè si ha un “entità figlia” nella quale ogni istanza è anche istanza di “entità padre”, essa è quindi un sottoinsieme del padre ed eredita tutte le sue proprietà: attributi, partecipazioni in relazioni e vincoli.

Entità figlia inoltre aggiunge a queste istanze nuovi elementi.

ATTENZIONE ogni entità può avere al massimo un entità padre (no ereditarietà multipla), al contrario però ogni entità può avere più entità figlie, le quali possono avere anche istanze in comune in comune. Inoltre vale proprietà di trasitività.

Semantica: in ogni istanza I dello schema S con E₁ ISA E₂: *istanze*(I,E₁) ⊆ *istanze*(I,E₂)

Sintassi:



Generalizzazioni

Altra relazione di sottoinsieme, questa volta però generalizza diverse sottoentità rispetto un unico criterio, quindi le sottoentità hanno insieme di istanze disgiunti a coppie.

Similmente ad ISA esistono quindi un “entità padre” e più “entità figlia” sottoinsieme di essa che eredita tutte le istanze con rispettivi attributi, partecipazioni in relazioni e vincoli.

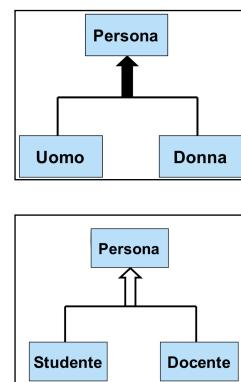
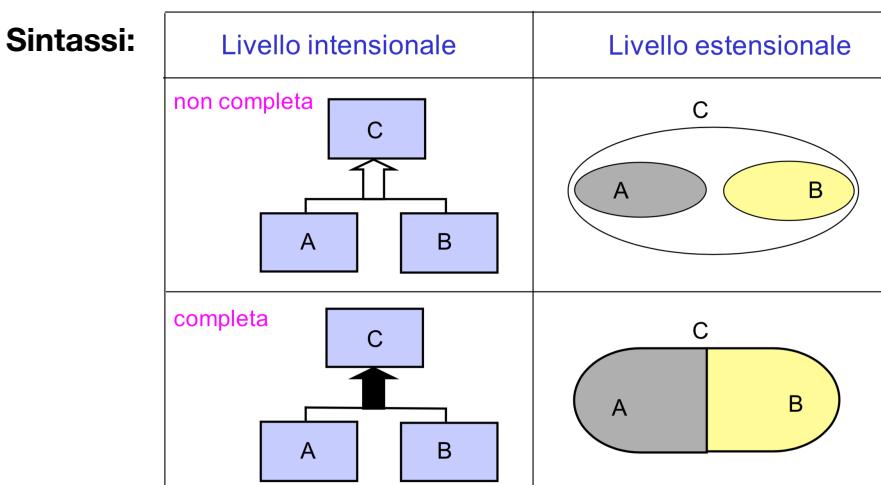
ATTENZIONE anche in questo caso ogni entità può avere al massimo un entità padre (no ereditarietà multipla)

Possono essere di 2 tipi:

- **Generalizzazione completa** => l'unione dell'insieme delle sottoentità è uguale all'insieme delle istanze dell'entità padre
- **Generalizzazione non completa** => esistono istanze che non appartengono a nessuna delle sottoentità

Semantica: in uno schema S in cui è definita una generalizzazione tra un entità padre F e le sottoentità E₁, E₂, ..., E_n, in ogni istanza I di S si ha: $istanze(I, E_i) \subseteq istanze(I, F)$ e inoltre $istanze(I, E_i) \cap istanze(I, E_k) = \emptyset$, per ogni $1 \leq i, k \geq n, i \neq k$.

Con generalizzazione completa vale anche: $istanze(I, E_1) \cup \dots \cup istanze(I, E_n) = istanze(I, F)$



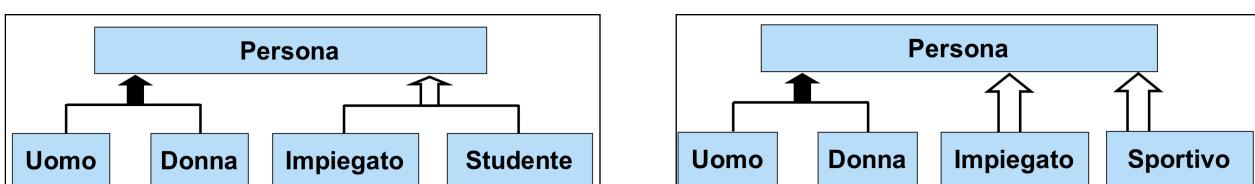
OSS quindi la differenza con le ISA sta nel fatto che nelle generalizzazioni non si possono avere istanze in comune tra le sottoentità.

Extra

1. Uso corretto di generalizzazione al posto di ISA



2. Differenza tra entità basate su stesso criterio ed indipendenti

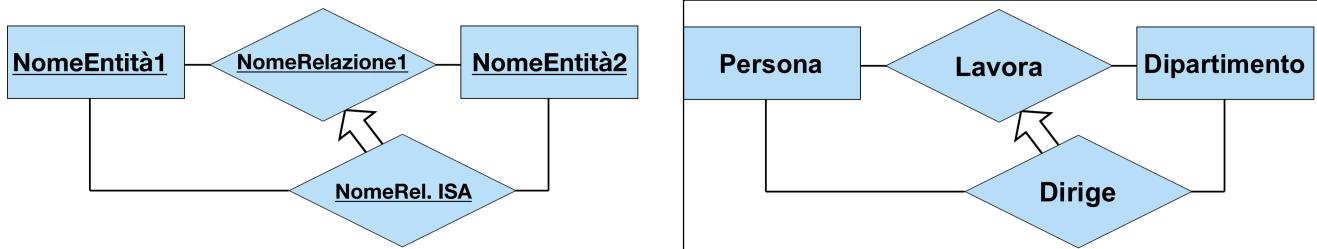


ISA e generalizzazioni tra relazioni

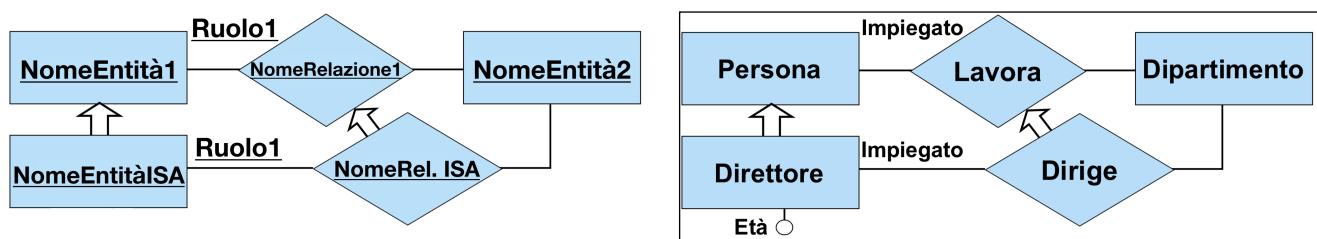
Gli stessi concetti si possono applicare anche sulle relazioni, nel caso ISA o generalizzazione tra una relazione figlia R ed una relazione padre Q bisogna però tener conto di alcune condizioni:

- Entrambe le relazioni hanno lo stesso grado
- Entrambe le relazioni hanno gli stessi ruoli
- Per ogni ruolo U, l'entità corrispondente ad U in R è un entità figlia dell'entità corrispondente ad U in quel ruolo

Sintassi v1 (relazioni in ISA su stesse entità):

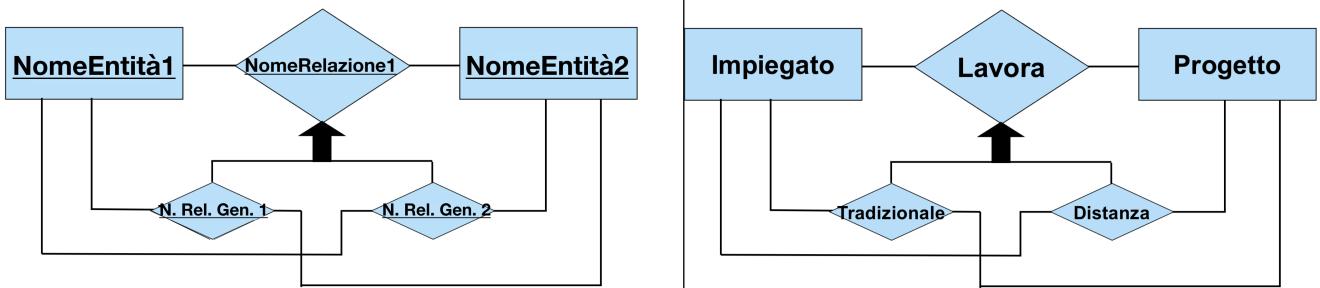


Sintassi v2 (relazioni in ISA su diverse entità):



ATTENZIONE la relazione in ISA può definirsi solo su entità che a loro volta sono in ISA con entità in relazione con la sua relazione padre, inoltre è importante specificare che entrambe hanno lo stesso ruolo di partecipazione

Sintassi v3 (generalizzazione di entità):



Vincoli di integrità modello ER

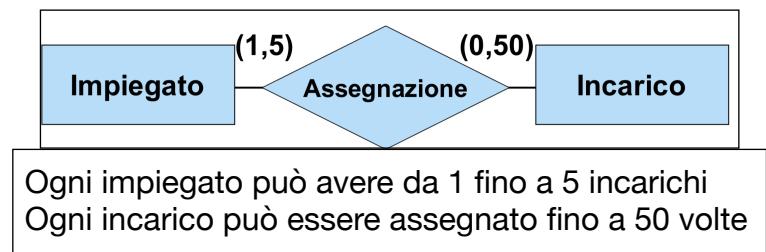
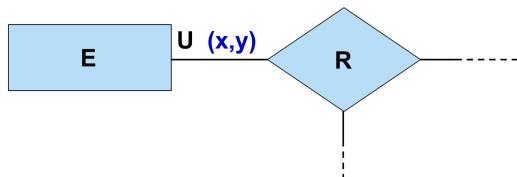
Regole che si esprimono nello schema e devono essere rispettate da ogni istanza

Vincoli di cardinalità sulle relazioni

Determina quante volte ogni istanza di un entità può partecipare ad una relazione in un determinato ruolo, ovvero quante volte l'istanza dell'entità può comparire nella stessa istanza della relazione.

Semantica: dato uno schema S in cui è definito un vincolo di cardinalità (x,y) associato ad un ruolo U (corrispondente ad un entità E) in una relazione R, allora in ogni istanza di I nello schema S, per ogni i in $istanze(I,E)$, il numero di istanze di R che in I hanno i come componente nel ruolo U è: $x \leq \text{numero istanze} \leq y$

Sintassi:



OSS mancanza del vincolo esplicito implica (0,n)

ATTENZIONE nel caso di ISA tra entità l'entità figlia eredita anche il vincolo, nel caso si voglia modificarlo questo può essere solo più ristretto.

Vincoli di cardinalità sugli attributi

Di base gli attributi hanno cardinalità implicita (1,1), è possibile modificarla per avere:

- **Opzionalità** (valore può non essere presente, NULL)
- **Attributi multivale**

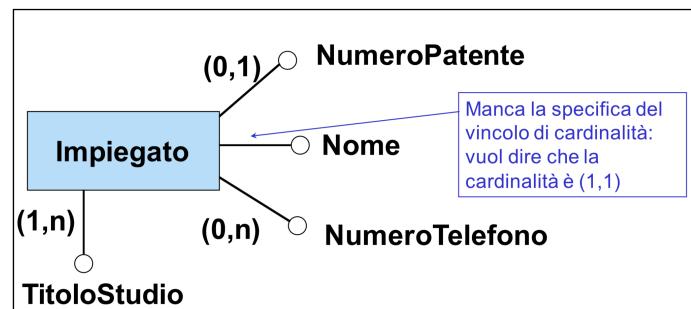
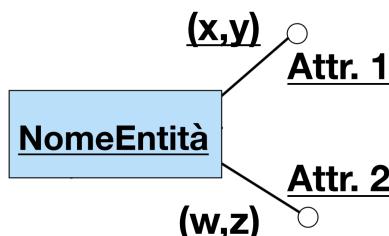
Semantica: dato uno schema S su cui è definito un attributo A in n entità E_1, E_2, \dots, E_n , rispettivamente con domini D_1, D_2, \dots, D_n , e con cardinalità $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$, in ogni istanza I di S, A è una relazione del tipo:

$$istanze(I,A) \subseteq (istanze(I,E_1) \cup \dots \cup istanze(I,E_n)) \times D_1, (D_2 \cup \dots \cup D_n)$$

e tale che, per ogni i vale: se $a \in istanze(I,E_i)$, allora il numero di coppie (a,b) in $istanze(I,A)$ è soggetto ad i vincoli di cardinalità (x_i, y_i) , e per ogni $(a,b) \in istanze(I,A)$, si ha che $b \in D_i$

OSS nel caso in cui due entità dovessero avere lo stesso attributo questo farebbe parte dello stesso dominio, quindi se non sono disgiunte deve corrispondere in entrambe

Sintassi:



Vincoli di identificazione di identità

Insieme di proprietà (attributi e/o relazioni) che permettono di identificare univocamente le istanze di un entità, ovvero non esistono 2 istanze dell'entità che hanno lo stesso valore per tutte le proprietà che compongono l'identificatore, possono essere di 2 tipi:

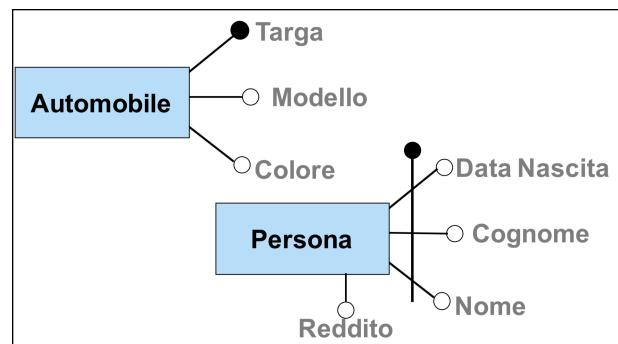
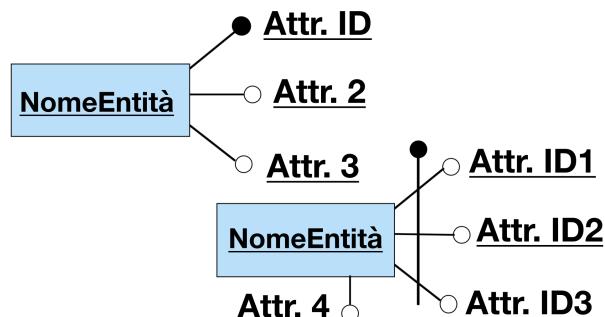
- **Interno** => formato solo da attributi dell'entità
- **Esterno** => formato da ruoli nelle relazioni ed eventuali attributi (id debole)

ATTENZIONE ogni attributo e ruolo deve avere cardinalità (1,1)

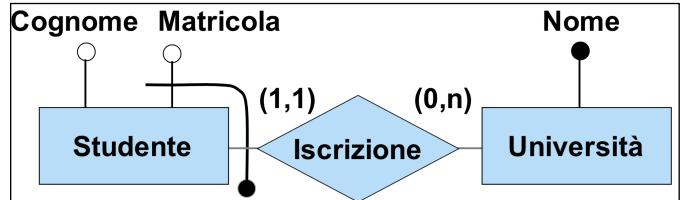
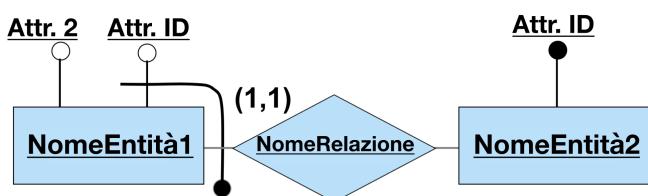
Semantica: dato uno schema S in cui è definito un vincolo di identificazione che specifica un identificatore per l'entità E formato dagli attributi A_1, A_2, \dots, A_n e/o dalle relazioni relazioni $R_1(U_1), R_2(U_2), \dots, R_n(U_n)$ (tutti con cardinalità (1,1)), allora in ogni istanza I di S, prese due istanze qualunque e_1 ed e_2 in $\text{istanze}(I, E)$, esse differiscono almeno in un valore A_i o almeno nella partecipazione con ruolo U_i in una R_i

OSS gli identificatori di una sola entità possono essere molteplici.

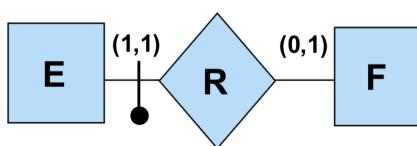
Sintassi v1 (vincoli interni):



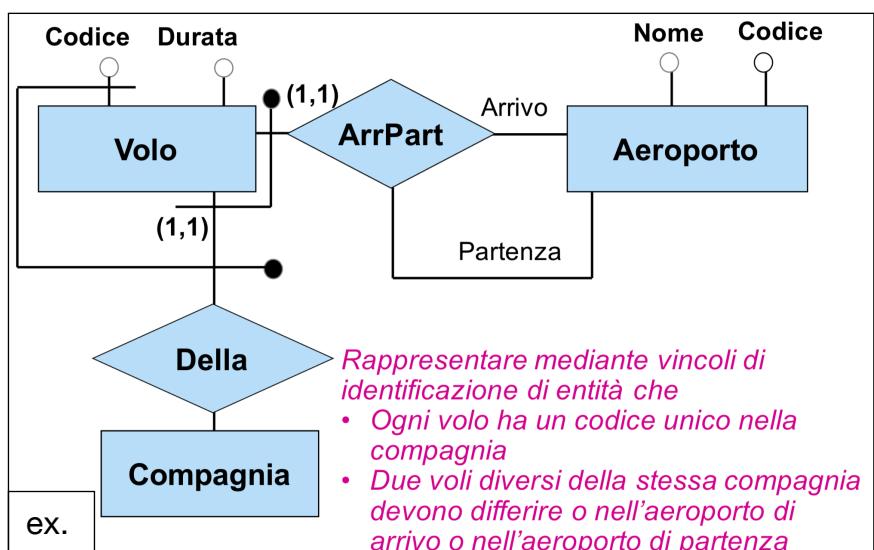
Sintassi v2 (vincoli esterni):



Extra



Se un entità E ha un identificatore (1,1) su una relazione ed almeno un'altra entità ha (0,1) allora l'identificatore esterno è implicito, non vale il contrario poiché valori differenti possono essere associati ad E



Rappresentare mediante vincoli di identificazione di entità che

- Ogni volo ha un codice unico nella compagnia
- Due voli diversi della stessa compagnia devono differire o nell'aeroporto di arrivo o nell'aeroporto di partenza

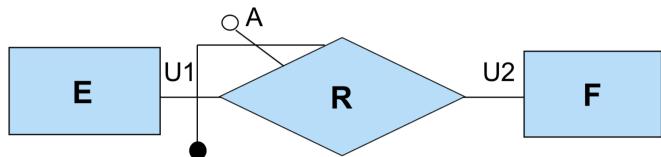
Vincoli di identificazione di relazioni

Permette di identificare univocamente le istanze di una relazione, non possono quindi esistere 2 istanze della relazione aventi gli stessi valori per tutte le proprietà che la compongono.

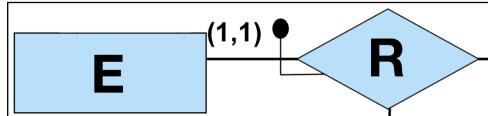
Semantica: dato uno schema S in cui è definito un vincolo di identificazione che specifica un identificatore per una relazione R formato da dagli attributi A₁, A₂, ..., A_n (tutti con cardinalità (1,1)) e/o dalle relazioni relazioni R₁(U₁), R₂(U₂), ..., R_n(U_n), allora in ogni istanza I di S, prese 2 istanze qualunque r₁ ed r₂ in *istanze(I,R)*, esse differiscono almeno in un valore A_i o almeno nella partecipazione con ruolo U_i in una R_i

OSS nelle relazioni esiste un vincolo implicito dato da tutti i ruoli che partecipano ad essa

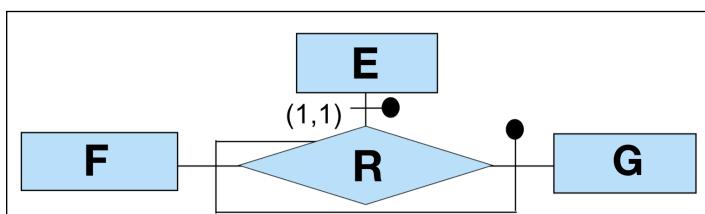
Sintassi:



Extra



Vincolo (1,1) o (0,1) implica identificatore di relazione implicito sull'entità



Considerando che un vincolo (1,1) implica identificatore per l'entità allora anche le altre entità che partecipano alla relazione, potendo essere accoppiate una sola volta con essa hanno un identificatore implicito corrispondente nella relazione

Vincoli esterni

Contenuti nel dizionario dei dati, permettono di specificare intrrelazioni non direttamente esprimibili tramite il diagramma.

Si possono esprimere in tabelle esterne allo schema tramite:

- formalismi opportuni (ex. Logica matematica)
- Linguaggio naturale

PROGETTAZIONE LOGICA

Serve per tradurre lo schema concettuale in uno schema logico che rappresenta gli stessi dati utilizzando il modello logico del DBMS e tiene conto del carico applicativo per l'ottimizzazione delle prestazioni (tempo esecuzione / spazio in memoria).

Il carico applicativo è dato da:

- volume dei dati
- Caratteristiche delle operazioni => batch/interattiva e frequenza operazioni

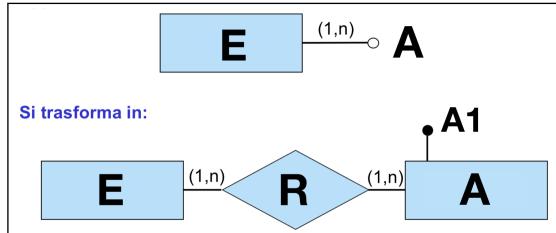
1. Ristrutturazione schema ER

Serve per rendere tutti i costrutti traducibili nel modello logico del DBMS (ex. relazionale), si divide in più fasi:

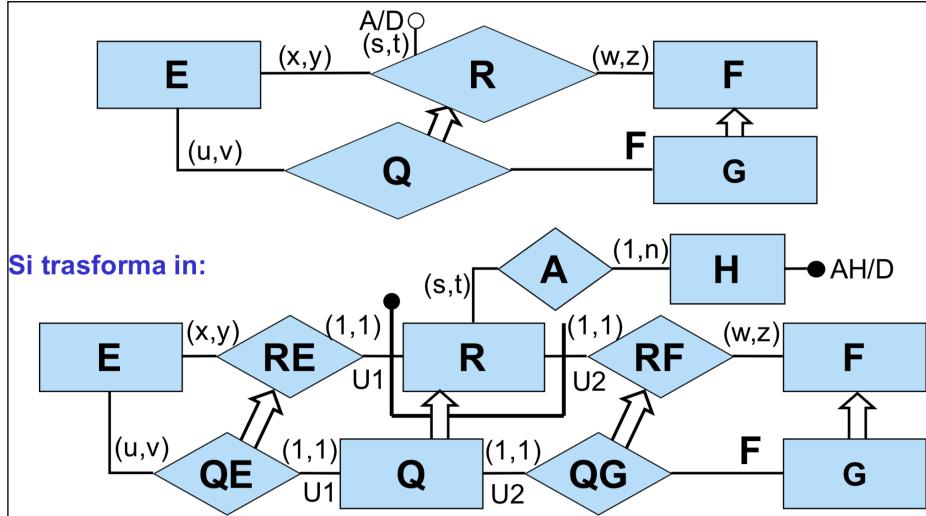
0. **Preparazione dello schema** => si riportano tutti i vincoli, anche quelli impliciti
1. **Analisi delle ridondanze** => tenendo conto delle informazioni sul carico applicativo si sceglie quali ridondanze utili tenere e quali eliminare per favorire efficienza
2. **Eliminazione attributi multivalore** => indispensabile poiché modello relazionale accetta un solo valore per attributo, vengono sostituiti da relazioni binarie.
3. **Eliminazione degli attributi composti** => come per punto 2
4. **Eliminazione di ISA e generalizzazioni** => poiché questi due concetti non sono previsti nel modello relazionale si trasformano anch'esse:
 - nel caso di entità si trasformano in relazioni binarie, con aggiunta di vincoli nel caso di generalizzazioni
 - nel caso di relazioni basta eliminare l'ISA ed aggiungere un vincolo esterno
OSS bisogna fare attenzione al fatto che nel caso di ISA non si hanno più istanze in comune tra padre e figlio ma entità disgiunte a coppie come per le generalizzazioni, questo implica inoltre l'aggiunta di altri vincoli esterni per eventuali attributi in comune
5. **Scelta identificatori principali** => sia nel caso di entità che di relazioni si possono avere più identificatori, bisogna quindi sceglierne uno secondo i criteri:
 - Essenzialità
 - Semplicità (formato da pochi elementi)
 - Utilizzo più frequente
 - (solo entità) preferenza id interni
 - (solo entità) Nel caso di id esterni si privilegia quello proveniente da ISA
 - (solo entità) Possibile aggiungere eventuale codice
OSS è importante tener conto della formazione di cicli se si scelgono id esterni
 - (solo relazioni) nel caso di ISA si opta per ruolo entità figlia
6. **Riformulazione vincoli esterni**
7. **Riformulazione di operazioni e carico applicativo**

Trasformazione attributi multivalore (2)

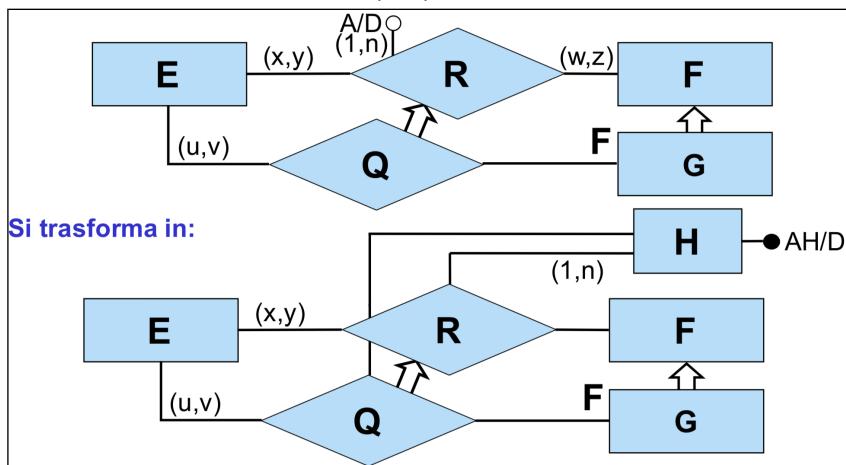
Entità:



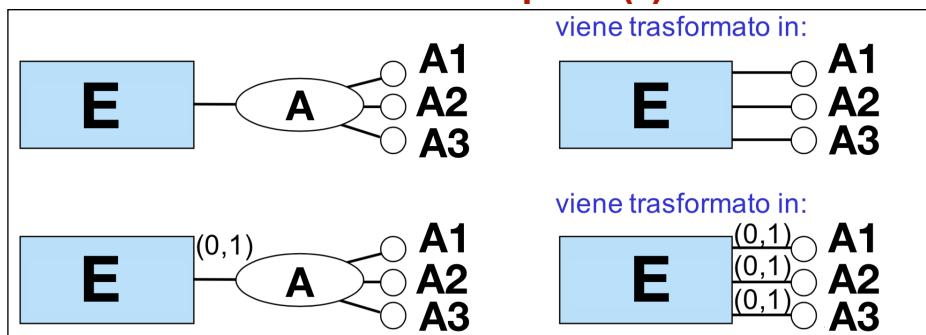
Relazione, caso generale:



Relazione, caso attributo $(1,n)$:



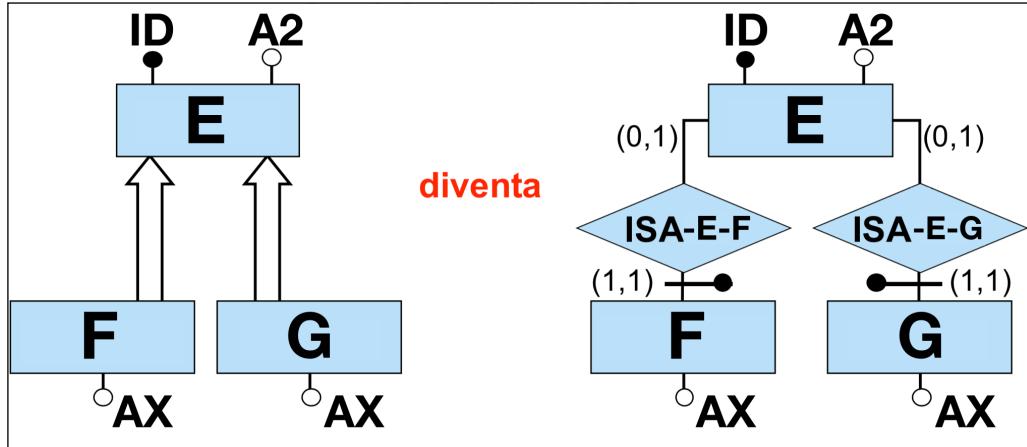
Trasformazione attributi composti (3)



OSS si può realizzare anche come entità come per gli attributi multivalore

vincolo esterno: ciascun attributo è definito solo se sono definiti anche tutti gli altri

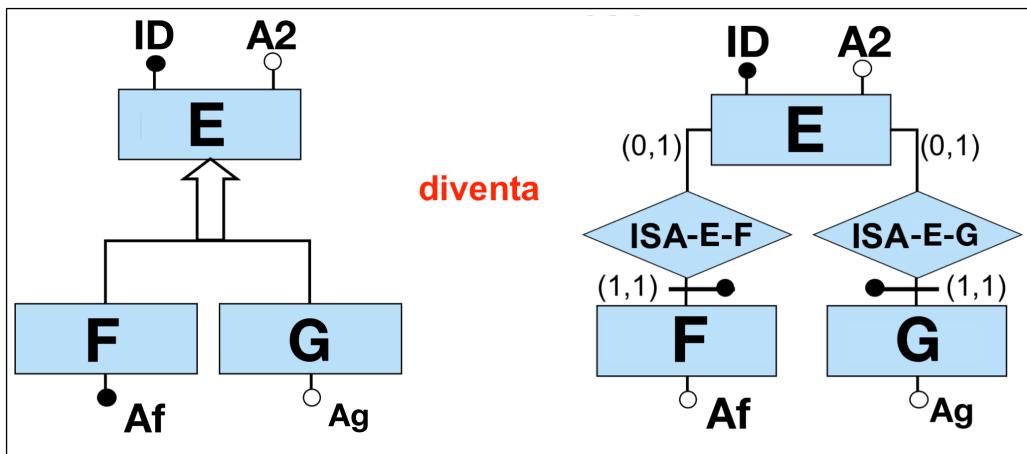
Trasformazione ISA entità (4)



so in presenza di attributi in comune si ha vincolo esterno:

{vincolo esterno: per ogni istanza I dello schema S , per ogni $e \in \text{istanze}(I, E)$, se esistono $f \in \text{istanze}(I, F)$, e $g \in \text{istanze}(I, G)$ tali che $(f, e) \in \text{istanze}(I, \text{ISA}-E-F)$ e $(g, e) \in \text{istanze}(I, \text{ISA}-E-G)$, allora esiste ax tale che $\langle f, ax \rangle \in \text{istanze}(I, AX)$ e $\langle g, ax \rangle \in \text{istanze}(I, AX)$ }

Trasformazione generalizzazione entità (4)

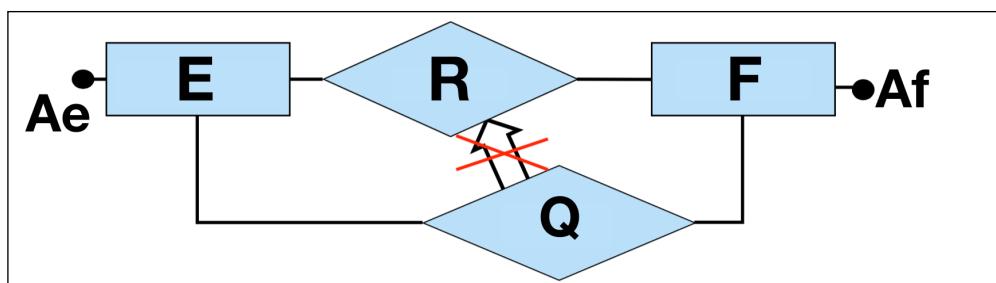


vincolo esterno:

generalizzazione completa => ogni istanza di **E** partecipa a **ISA-E-F** oppure a **ISA-E-G**, ma non a entrambi

Generalizzazione non completa => nessuna istanza di **E** partecipa sia a **ISA-E-F** che a **ISA-E-G**

Trasformazione generalizzazione relazione semplice (4)



vincolo esterno: ogni istanza di **Q** è anche istanza di **R**

2. Traduzione diretta

Traduce automatica dello schema ER ristrutturato in un schema relazionale che può essere utilizzato come schema logico massima, può richiedere altre ristrutturazioni

Notazione

NomeEntità (attributo1, attributo2, attributo3*...)

foreign key: NomeEntità[attributo1, ...] \subseteq NomeAltraEntità[attributo1, ...]

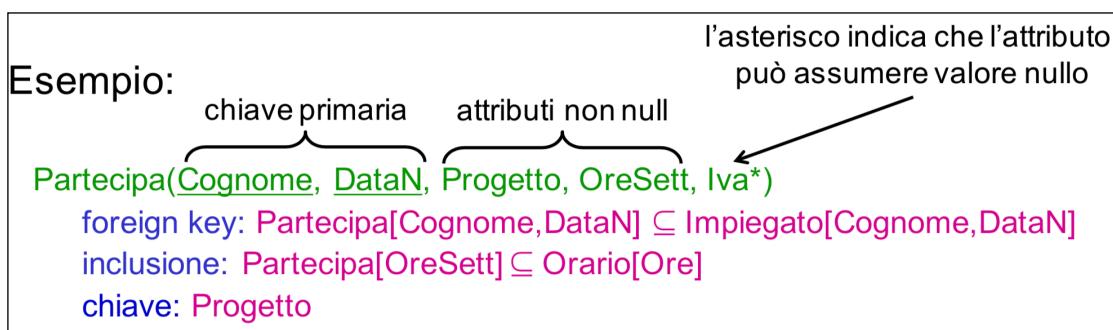
inclusione: NomeEntità[attributo1, ...] \subseteq NomeAltraEntità[attributo1, ...]

chiave: altri attributi degnati con identificatore (non chiave primaria)

vincolo: vincoli esterni non traducibili

Sottolineato => chiave primaria

Asterisco => attributo può essere NULL



Traduzione di entità

Ogni entità ER viene tradotta in una tabella (relazione) dello schema relazionale, in generale si seguono le regole:

- Ogni attributo dell'entità diventa attributo della tabella
- L'identificatore principale diventa la chiave primaria, altri semplici vincoli di chiave
- Gli attributi (0,1) vengono asteriscati

Inoltre in caso di accorpamento (identificatore esterno):

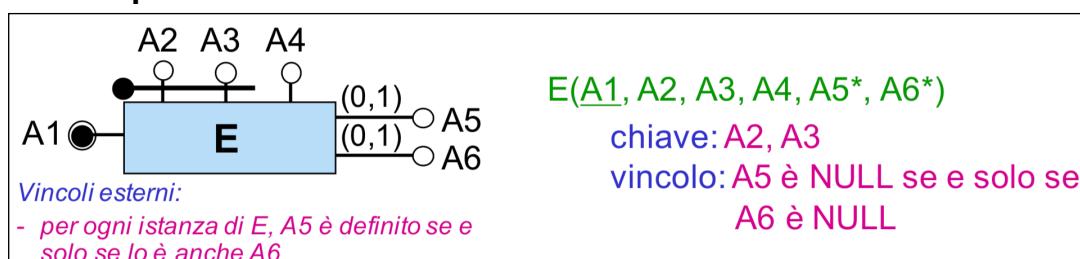
- Ogni ruolo delle entità collegate alla ER-relazione diventa attributo della tabella
- Per ogni ruolo inserito nella tabella si crea un vincolo di foreign key su attributi che ne compongono la chiave primaria

OSS Inoltre se il ruolo ha cardinalità (1,1) si aggiunge nella sua tabella un vincolo di foreign key, nel caso di (1,n) si aggiunge un vincolo di inclusione

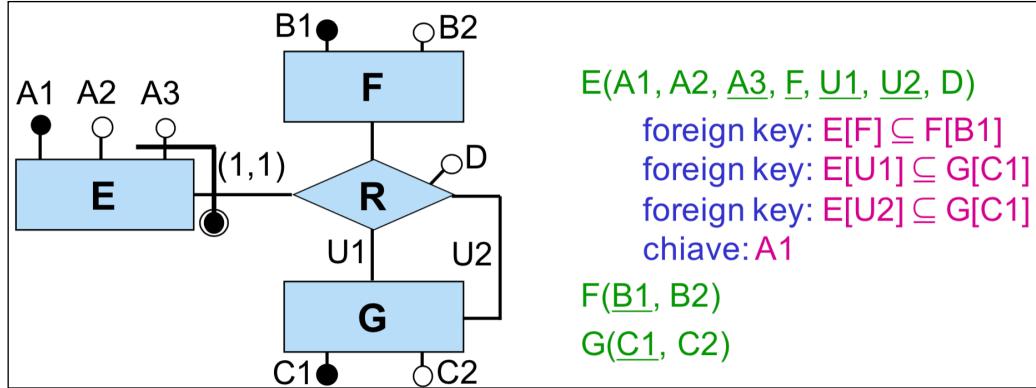
- Gli attributi della ER-relazione diventano attributi della tabella, inoltre nel caso sia definito un identificatore della ER-relazione su di essi (ed eventuali altre entità), questi diventano ulteriori chiavi per la tabella dell'entità

Esempi:

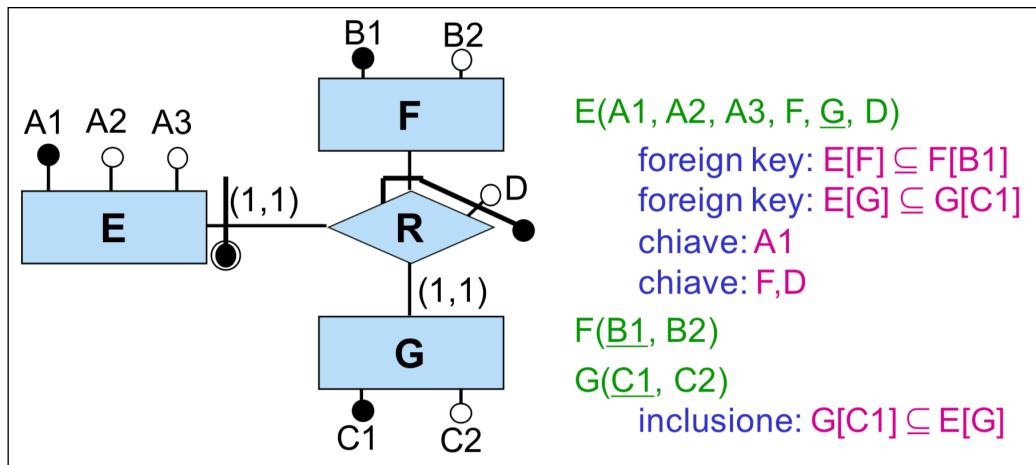
- **Senza accorpamento**



- Con accorpamento (caso base)

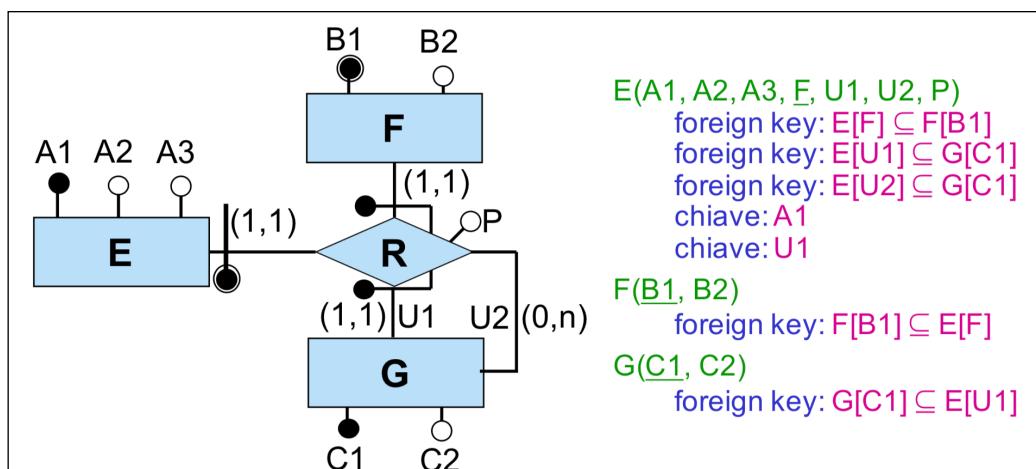


- Con accorpamento (chiave completamente esterna)



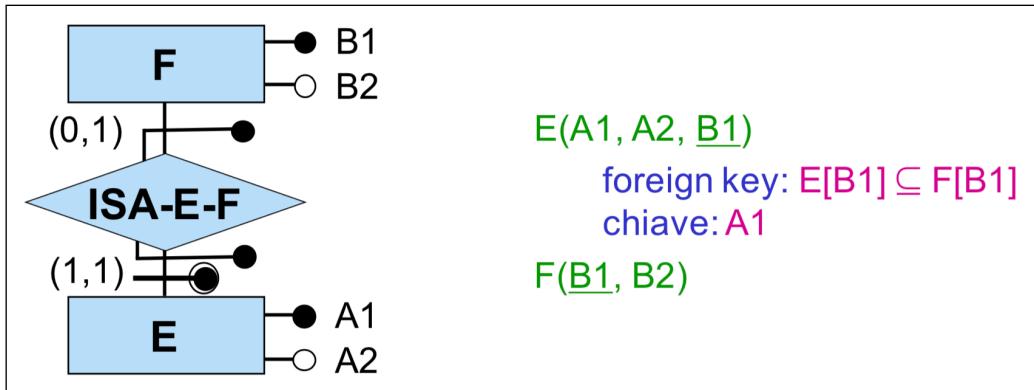
OSS Notare presenza vincolo di inclusione su G
e chiave data da D su E

- Con accorpamento (chiave completamente esterna e vincoli (1,1))



OSS Notare come nel caso in cui l'identificatore sia completamente esterno e tutte le altre relazioni abbiano un vincolo (1,1) è possibile utilizzare uno solo dei ruoli come chiave primaria per la tabella

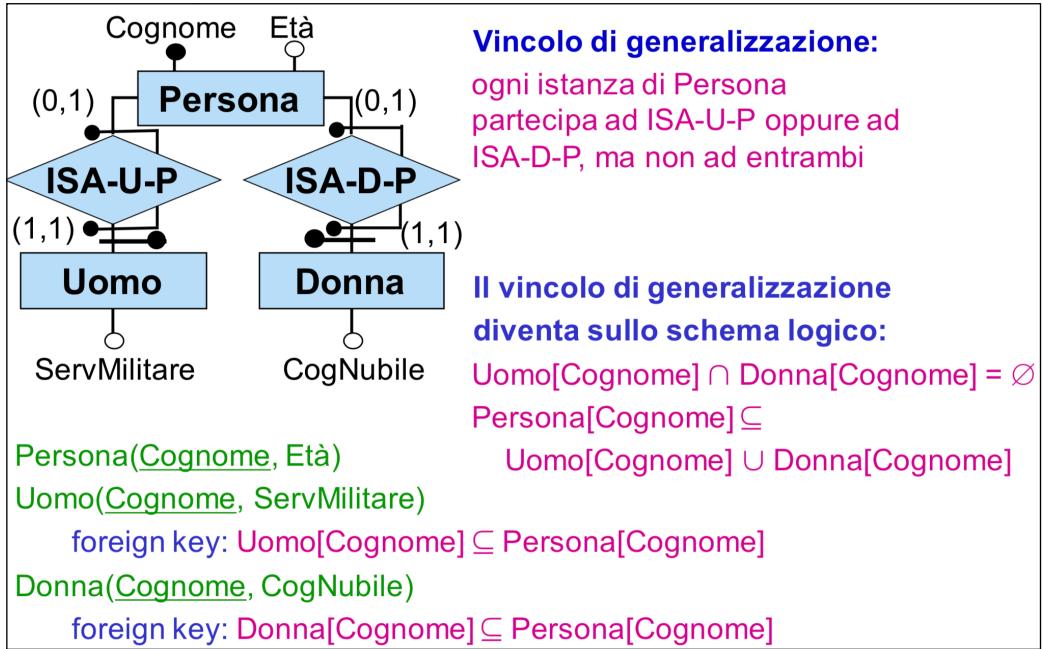
- Accorpamento derivante da ISA



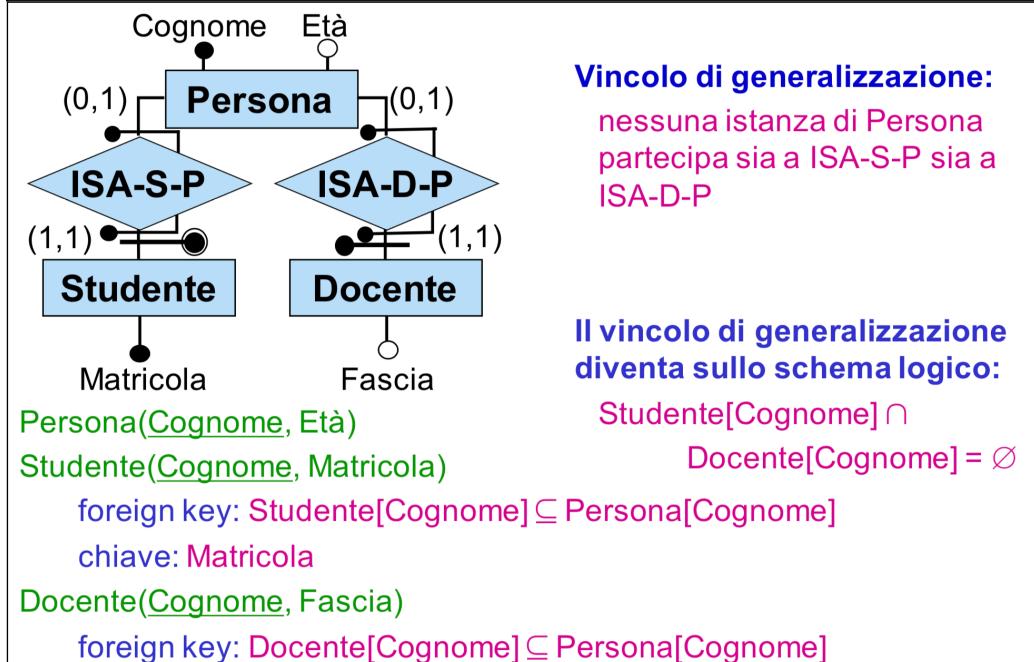
OSS Notare come l'accorpamento dell'ISA si effettua nella relazione figlia e chiave primaria rimane del padre

- Accorpamento derivante da generalizzazione

Completa



Non completa



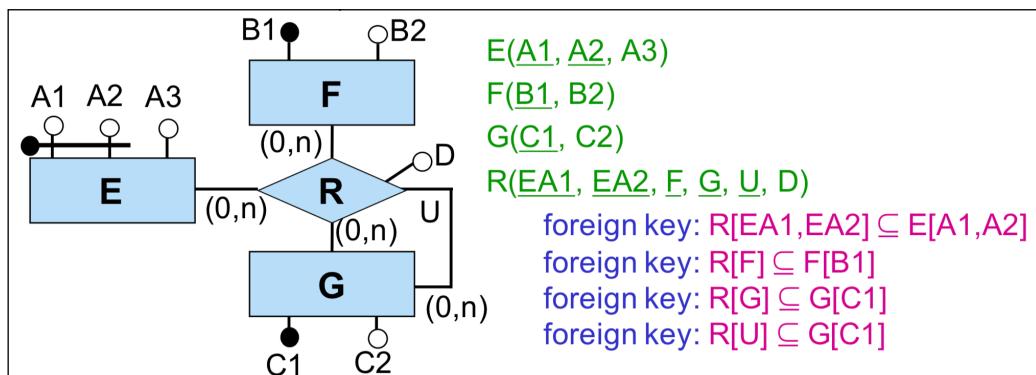
Traduzione di ER-relazione

Da effettuare solo su ER-relazioni che non sono state accorpate nelle tabelle delle entità, secondo le regole:

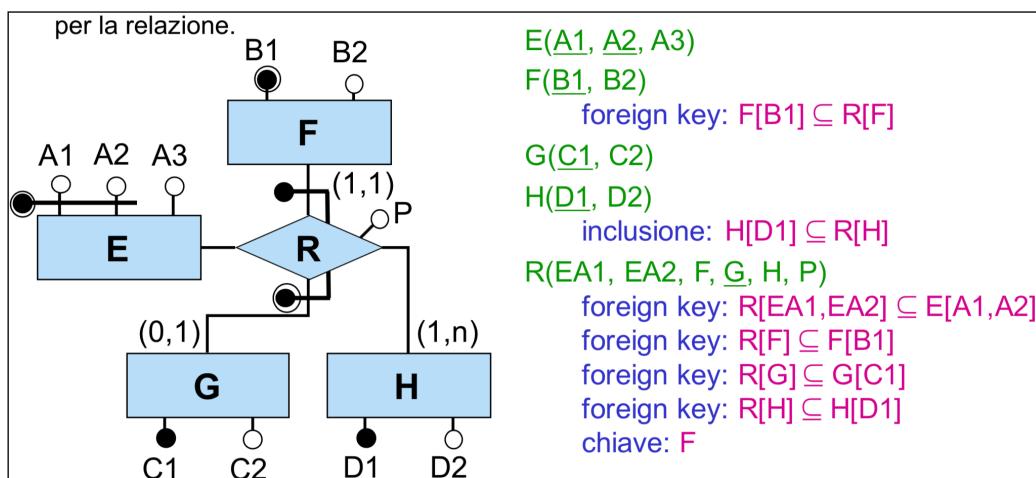
- Ogni attributo della ER-relazione diventa attributo della tabella
- Ogni ruolo associato alla ER-relazione diventa attributo della tabella
- Per ogni ruolo associato alla ER-relazione si crea un vincolo di foreign key
- L'identificatore principale diventa la chiave primaria, altri semplici vincoli di chiave OSS se non specificato si prende tutto
- Gli attributi (0,1) vengono asteriscati

Esempi:

- Caso base



- Caso con identificatore esplicito e vincoli di inclusione



3. ristrutturazione schema logico

Il modello logico risultante dalla traduzione diretta può comportare problemi di efficienza, i quali possono essere rispetto allo spazio:

- valori nulli
 - due relazioni R_1 ed R_2 con chiavi k_1 e k_2 si può avere una doppia inclusione, ovvero sia $R_1[k_1] \subseteq R_2[k_2]$ che $R_2[k_2] \subseteq R_1[k_1]$
 - Ridondanze non essenziali
- Oppure rispetto al tempo di esecuzione di query e updates:
- Numero e struttura delle relazioni
 - Ridondanze non essenziali

Modello di costo

Il problema principale nell'uso dei database sta nel fatto che, avendo grandi quantità di dati, si necessita il salvataggio della struttura nella memoria secondaria, potendo caricare solo alcune pagine alla volta nella memoria primaria.

Usando le notazioni:

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

Si può ricavare il seguente **costo** di massima:

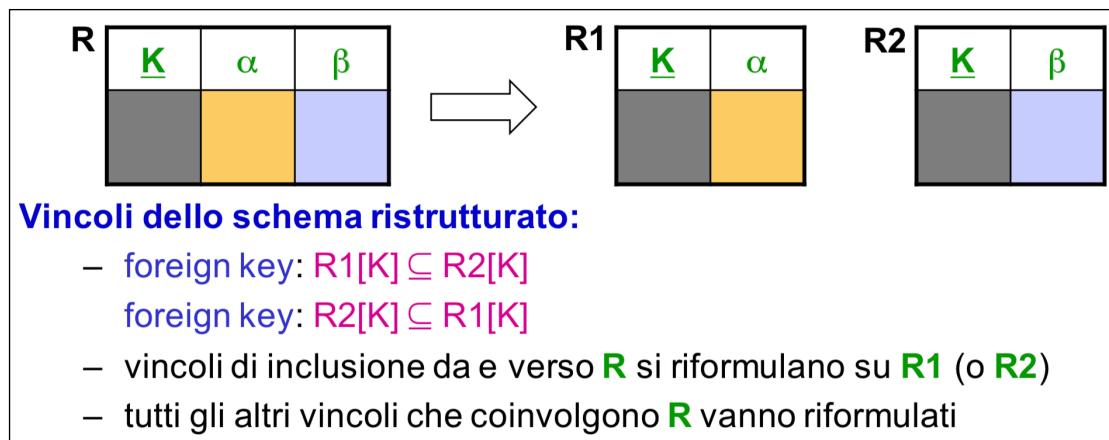
- Selezione su R => $N_P(R)$
- Proiezione su R => $N_P(R)$
OSS costosa in pagine con poche tuple
- Join tra R e Q
 - Senza indici => $N_P(R) \times N_P(Q)$
 - Con indici molto selettivi => $N_P(R) + N_P(R) \times N_{TP}(R)$

Per aumentare l'efficienza quindi bisogna cercare di tener basso il numero di pagine in memoria $N_P(R)$, nel caso, anche aumentando il numero di tuple per pagina $N_{TP}(R)$.

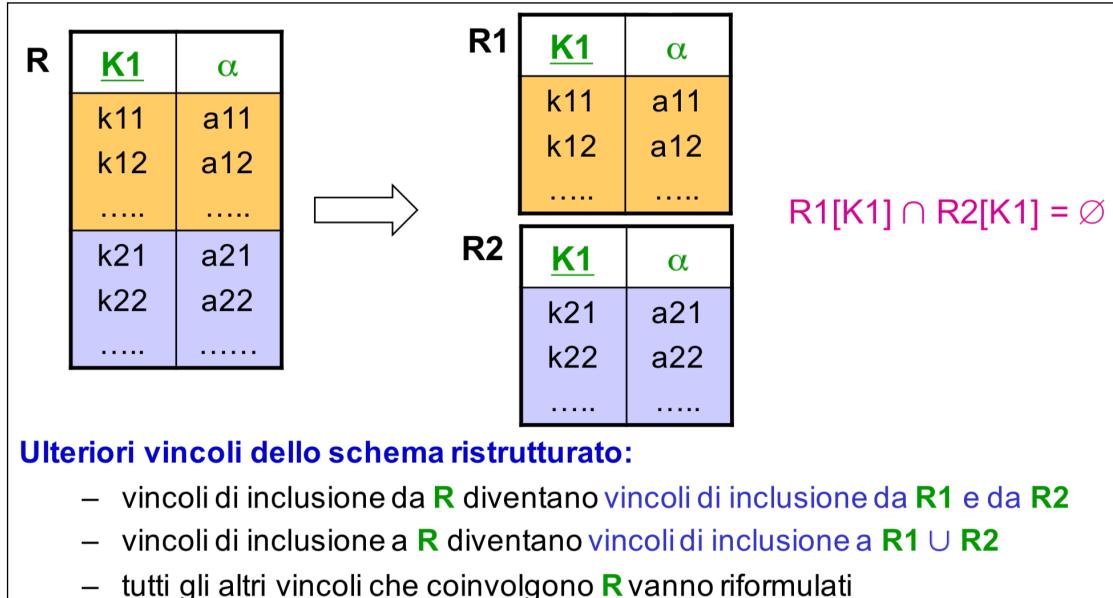
Ristrutturazioni dello schema

Decomposizione => per aumentare $N_{TP}(R)$:

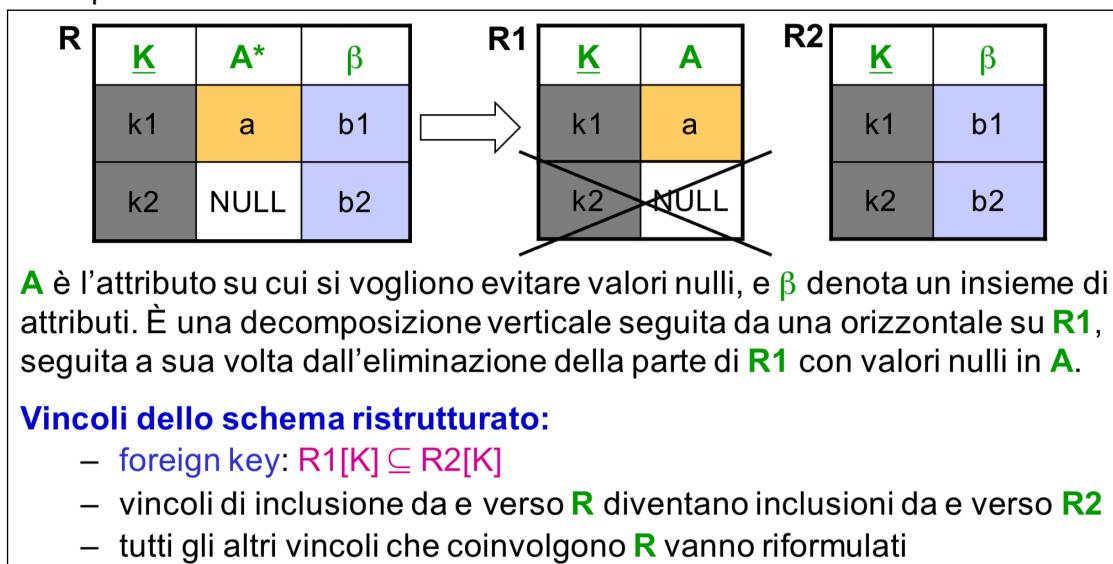
- Verticale => facilita accesso (selezione / proiezione) e permette normalizzazione, utile quando non si hanno molti accessi ad entrambi gli attributi



- Orizzontale => facilita accesso (selezione), utile per accesso a tuple a fasce (indici)

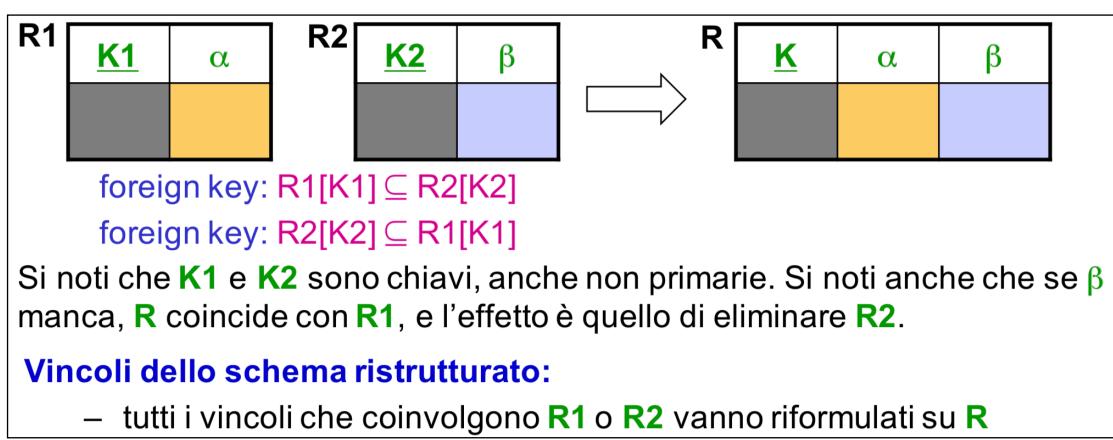


- Mista => permette di evitare valori nulli

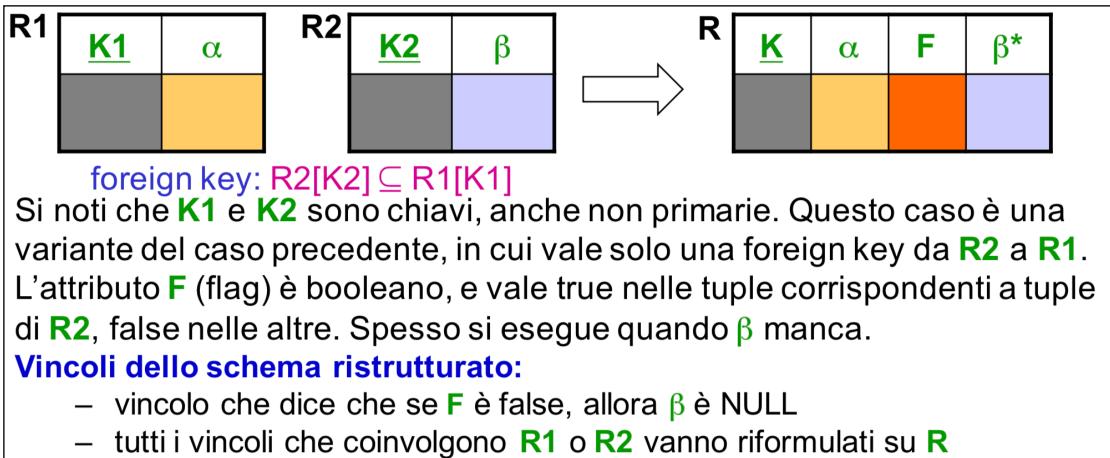


Accorpamento => facilita il join ed elimina relazioni ed attributi superflui:

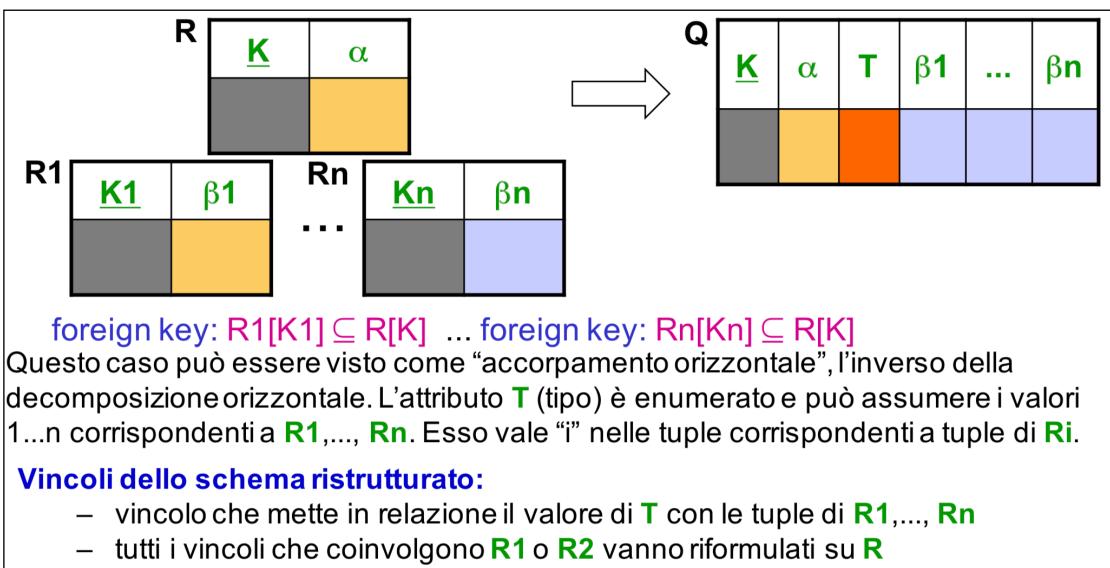
- caso 1



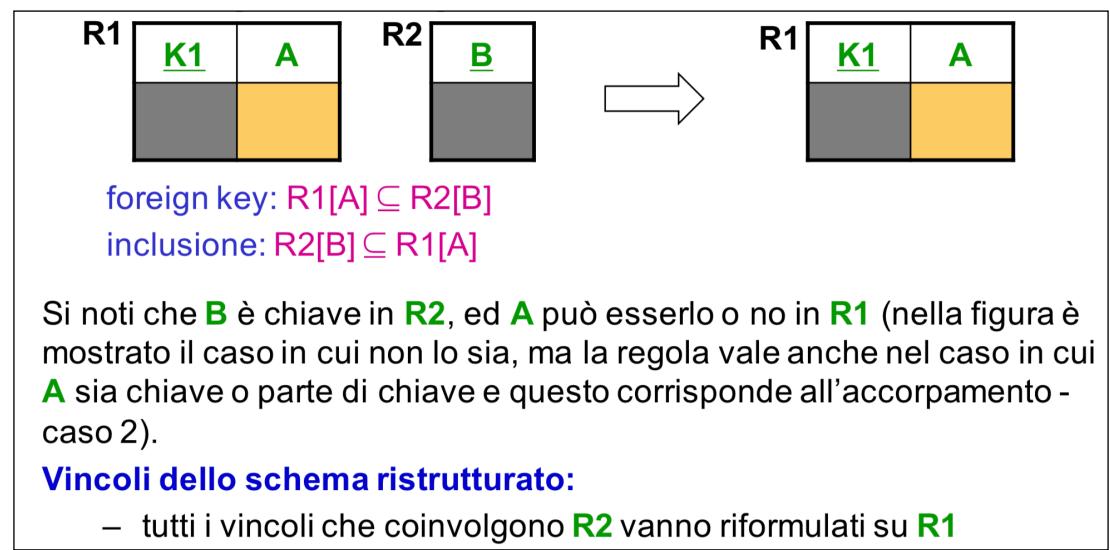
- caso 2 => utilizzabile quando sono richiesti molti join tra le due tabelle R1 e R2 con la condizione $k_1=k_2$



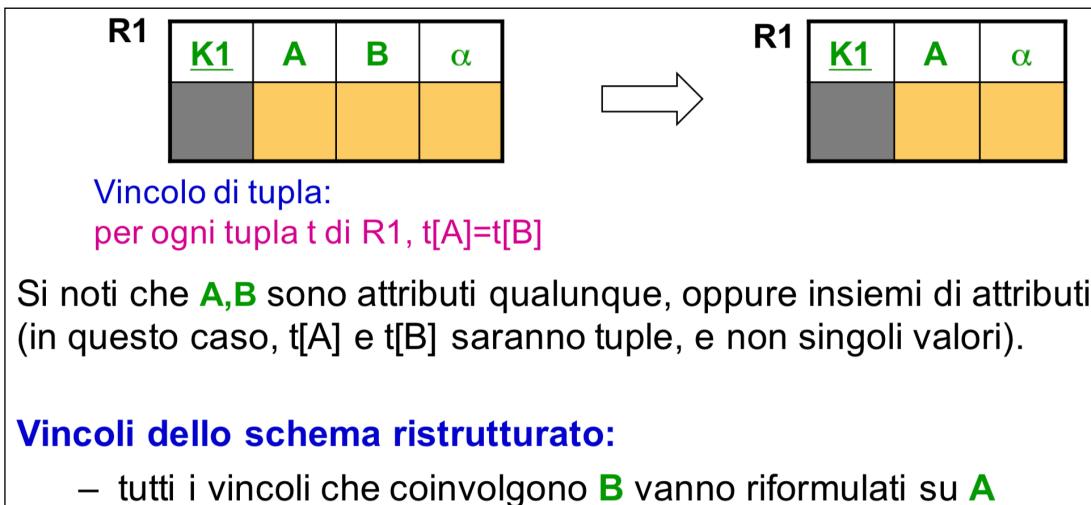
- caso 3 => versione con più tabelle del caso 2



- accorpamento per eliminare relazioni inutili, poiché tutti gli attributi sono già contenuti in un'altra relazione



- accorpamento per eliminare attributi inutili



OSS le tabelle iniziali possono essere recuperate tramite l'uso di views

EQUIVALENTI 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