Corso di Laurea in Ingegneria Gestionale
SAPIENZA Università di Roma
Esercitazioni del corso di Basi di Dati
Prof.ssa Catarci e Prof.ssa Scannapieco

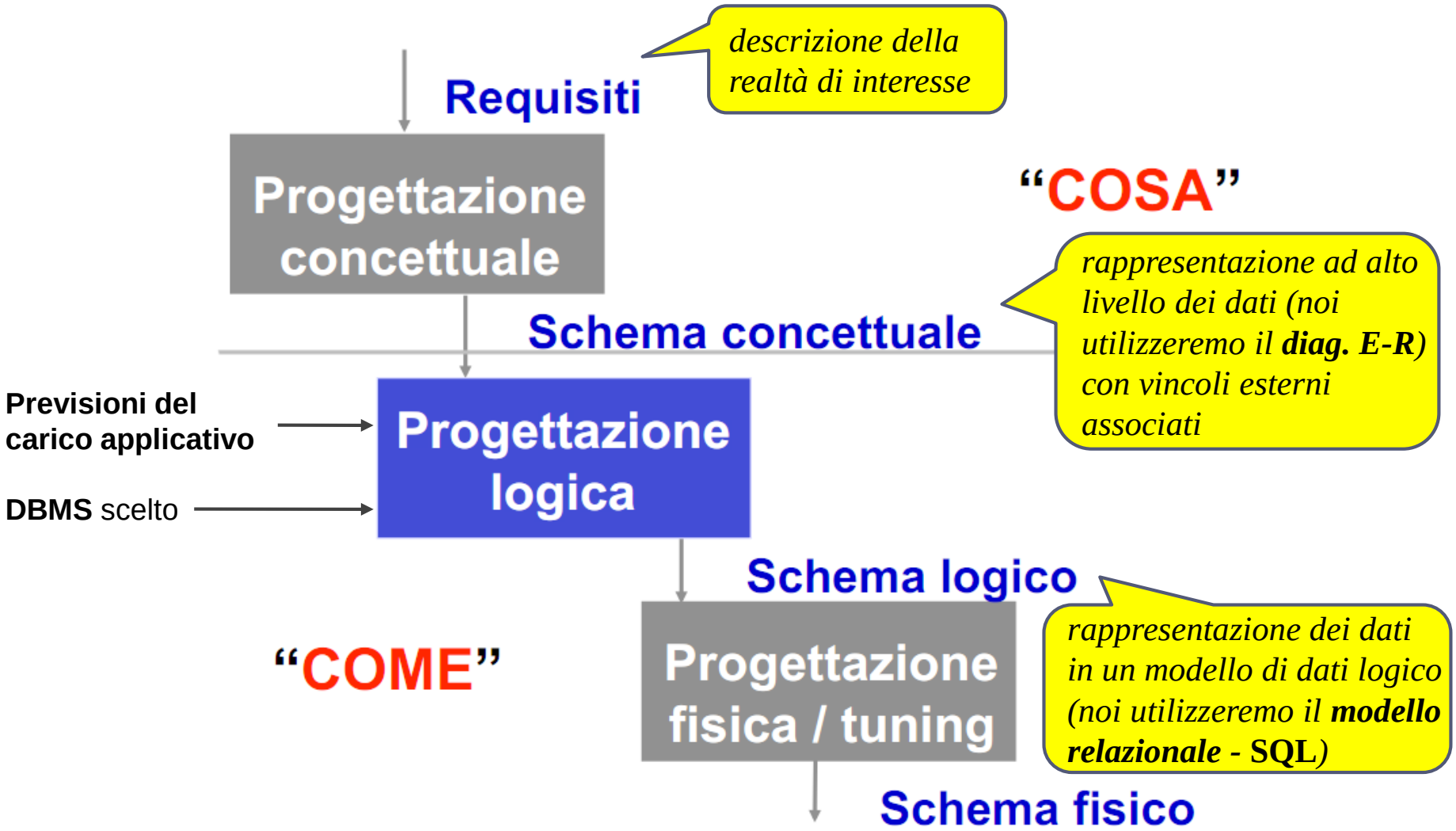
Anno Accademico 2012/2013

11 - Progettazione Logica

Francesco Leotta

Ultimo aggiornamento : 16/05/2013

Progetto di una base di dati - lo scenario



Obiettivo della progettazione logica

- L'obiettivo della progettazione logica è quello di costruire uno schema logico in grado di descrivere, in maniera corretta ed efficiente, tutte le informazioni contenute nello schema Entità-Relazione prodotto nella fase di progettazione concettuale.
- **Non si tratta di una semplice traduzione**.
- 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.
- La progettazione logica costituisce la base per l'effettiva realizzazione dell'applicazione e deve tener conto, per quanto possibile, anche delle sue **prestazioni**.
  - La ristrutturazione dello schema concettuale deve essere attuata con l'ottica di rendere più efficiente l'esecuzione delle operazioni previste.

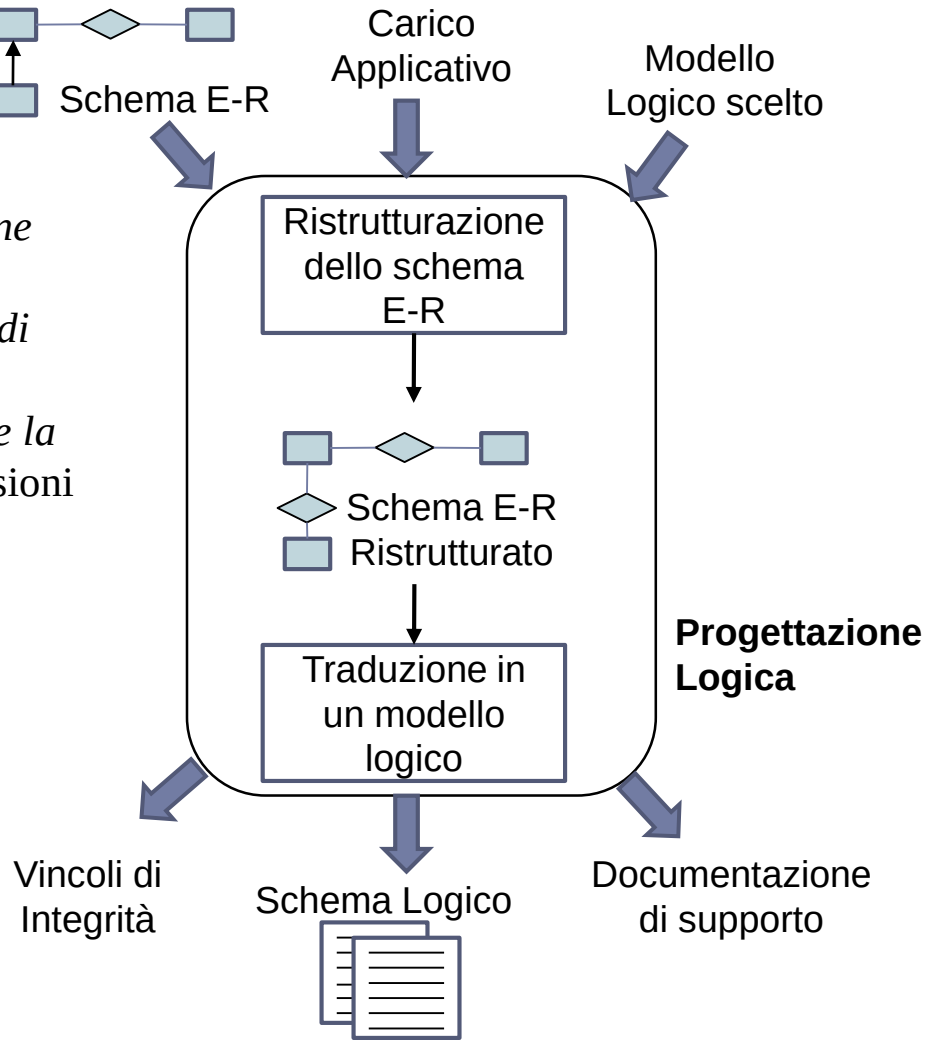
Dallo schema concettuale al logico

Input:

- Schema concettuale *(prodotto dalla progettazione concettuale).*
- DBMS scelto *(per sapere quale modello logico di dati adottare).*
- Previsioni del carico applicativo *(per ottimizzare la base di dati rispetto ad esso)*, in termini di dimensioni dei dati e caratteristiche delle operazioni.

Output:

- Schema logico *(rappresentazione dei dati in un modello di dati logico, eventualmente con vincoli complessi).*
- Vincoli di integrità.
- Documentazione di supporto.



Dallo schema concettuale al logico

- La progettazione logica si può dividere in due fasi principali :
  - **1) Ristrutturazione dello schema ER:**
    - eliminazione dei costrutti non direttamente traducibili nel modello relazionale.
    - scelta degli identificatori principali delle entità.
    - considerazione di aspetti relativi all'**efficienza**.
    - **Osservazione :** uno schema ER ristrutturato **non è più propriamente uno schema concettuale**, in quanto costituisce una rappresentazione dei dati che tiene conto degli aspetti realizzativi.
  - **2) Traduzione diretta dello schema ER ristrutturato nel modello relazionale:**
    - la traduzione è **diretta, nel senso che non richiede (quasi)** scelte da parte del progettista.
    - lo schema relazionale prodotto **non contiene ridondanze**, se non quelle volute.
    - la traduzione diretta tiene conto delle scelte fatte in fase di progettazione concettuale.

1. Ristrutturazione dello schema E-R

- **Fase 1:** analisi delle ridondanze
- **Fase 2:** eliminazione di attributi multivalore
- **Fase 3:** eliminazione di ISA tra entità
- **Fase 4:** eliminazione di generalizzazioni tra entità
- **Fase 5:** eliminazione di ISA fra relazioni
- **Fase 6:** scelta degli identificatori principali
- **Fase 7:** specifica degli ulteriori vincoli esterni
- **Fase 8:** riformulazione di operazioni e carico applicativo

Fase 1: analisi delle ridondanze

- Una **ridondanza** in uno schema ER è un' informazione significativa ma derivabile da altre.
- In questa fase si decide se eliminare le ridondanze eventualmente presenti o mantenerle. In alcuni casi potrebbe essere necessario introdurne delle nuove.
- La valutazione viene effettuata in funzione del **carico applicativo previsto** (aggiornamenti, interrogazioni, occupazione di spazio).
  - **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.

Rappresentazione del carico applicativo

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

nella tabella dei volumi vengono riportati tutti i concetti dello schema (entità e relazioni) con il numero di istanze previste a regime.

Tabella delle operazioni

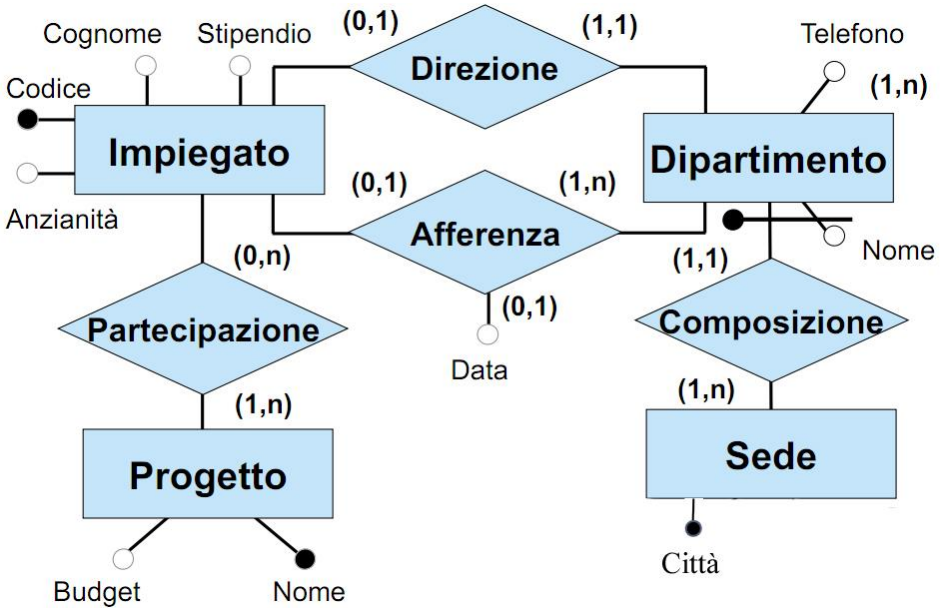
Op.	Tipo	Frequenza
1	Interattiva	50 / giorno
2	Interattiva	100 / giorno
3	Interattiva	10 / giorno
4	Batch	2 / sett.

nella tabella delle operazioni vengono riportate, per ogni operazione, la frequenza prevista e la tipologia.

Tabella dei volumi

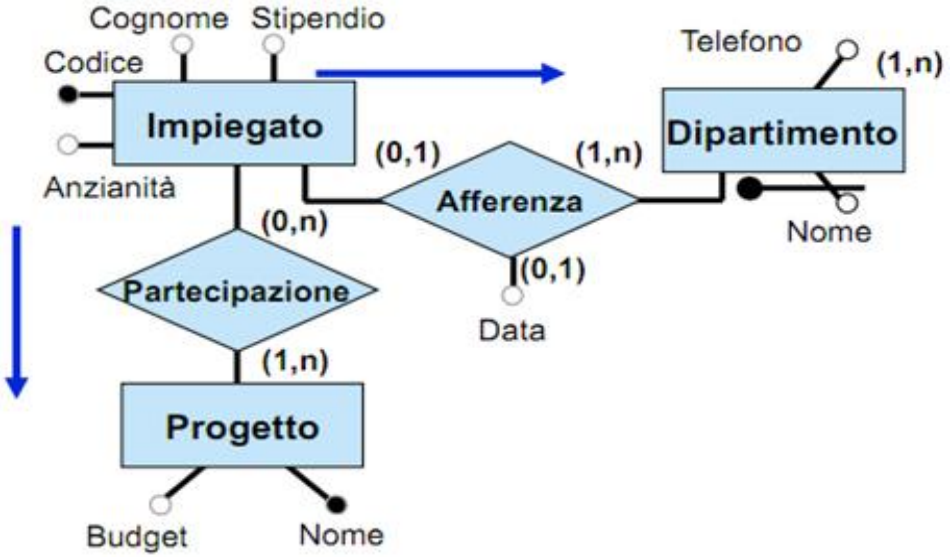
- Si noti che i valori relativi al numero di istanze di entità e relazioni nella tabella dei volumi sono influenzati:
  - dalle cardinalità nello schema.
  - dal numero medio di volte che le istanze delle entità partecipano alle relazioni.
- ***Esempio :***
  - vol(Composizione) = vol(Dipartimento)
  - vol(Direzione) = vol(Dipartimento)
  - vol(Afferenza) ≤ vol(Impiegato)
  - se ogni impiegato partecipa in media a 3 progetti (si capisce dalla tabella dei Volumi a pagina precedente):

    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.*



- per ogni operazione si possono descrivere graficamente i dati coinvolti con un frammento di schema E-R interessato dall'operazione, sul quale viene disegnato il "cammino logico" da percorrere per accedere alle informazioni di interesse.
- ad esempio, per ottenere le informazioni di interesse su un impiegato si parte dall' entità **Impiegato** per accedere, attraverso la relazione **Afferenza**, al suo dipartimento, e attraverso la relazione **Partecipazione**, ai progetti ai quali partecipa.

- Disponendo di queste informazioni è possibile fare una stima del costo di un'operazione sulla base di dati contando il numero di accessi alle istanze di entità e associazioni necessario per eseguire l'operazione .

Valutazione di costo

- ***Esempio :*** *trova tutti i dati di un impiegato, del dipartimento nel quale lavora e dei progetti ai quali partecipa.*

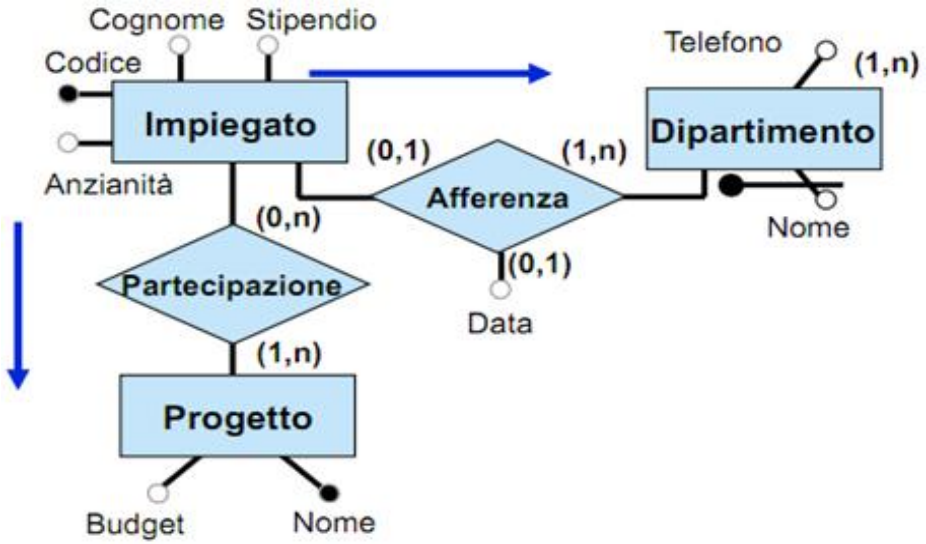


Tabella degli accessi dell'operazione 2

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

- per eseguire l'operazione, bisogna prima accedere ad un'istanza dell'entità **Impiegato** per accedere poi ad un'istanza della relazione **Afferenza** (infatti ogni impiegato asserisce al più ad un dipartimento) e, attraverso questa, ad un'istanza dell'entità **Dipartimento**.
- per conoscere i dati dei progetti a cui un impiegato lavora, si deve accedere a 3 istanze dell'associazione **Partecipazione** (abbiamo detto che in media un impiegato lavora su 3 progetti) e, attraverso queste, a 3 istanze dell'entità **Progetto**.

- Tutto questo può essere riassunto in una tavola degli accessi (**L** : accesso in lettura; **S** : accesso in scrittura – **più costoso**).

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 schema 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.

Esempio di valutazione dei costi

L'attributo ***NumeroAbitanti*** è ridondante; infatti questa informazione si potrebbe ricavare contando il numero di relazioni legate a ciascuna entità **Città**.

Supponiamo che si prevedano le seguenti operazioni e si abbia a disposizione la seguente tabella dei volumi:

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

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

Tabella dei volumi

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

Conviene mantenere la ridondanza? **(contando doppi gli accessi in scrittura, dato che hanno costo maggiore). La decisione di mantenere/eliminare la ridondanza va presa confrontando il costo di esecuzione delle operazioni che coinvolgono il dato ridondante e la relativa occupazione di memoria nei casi di presenza/assenza di ridondanza.**

Valutazione dei costi in assenza di ridondanza

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 singola città, incluso il numero di abitanti (2 volte al giorno)

Tabella degli accessi operazione 1

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

(scrittura dati su Persona)
(associazione città di residenza)

Tabella degli accessi operazione 2

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

(lettura dati di una Città)
(conteggio del numero di relazioni ***Residenza*** legate a **Città**)

Costi:

– **operazione 1:** 1000 accessi in scrittura al giorno – **COSTO = 2000** (contiamo doppi gli accessi in scrittura)
– **operazione 2:** 10002 accessi in lettura al giorno – **COSTO = 10000**

totale di *12000* accessi al giorno

Valutazione dei costi in presenza di ridondanza

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)

Tabella degli accessi operazione 1

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

Tabella degli accessi operazione 2

Concetto	Costrutto	Accessi	Tipo	
Città	Entità	1	L	(lettura dati di una Città, incluso il numero di abitanti)

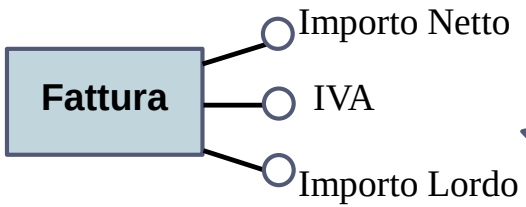
Costi:

– **operazione 1:** 1500 accessi in scrittura e 500 in lettura al giorno
– **operazione 2:** 2 accessi in lettura al giorno
Se contiamo doppi gli accessi in scrittura (hanno costo maggiore) :

totale di 3502 accessi al giorno

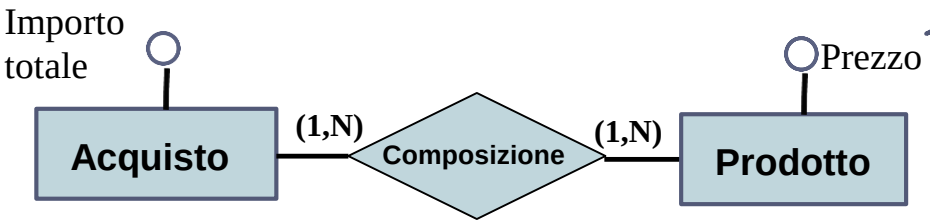
CONCLUSIONE :
in questo esempio
manteniamo la ridondanza

Esempi di Ridondanze



Attributi derivabili
L'attributo ***ImportoNetto*** è deducibile dagli altri attraverso un'operazione matematica.

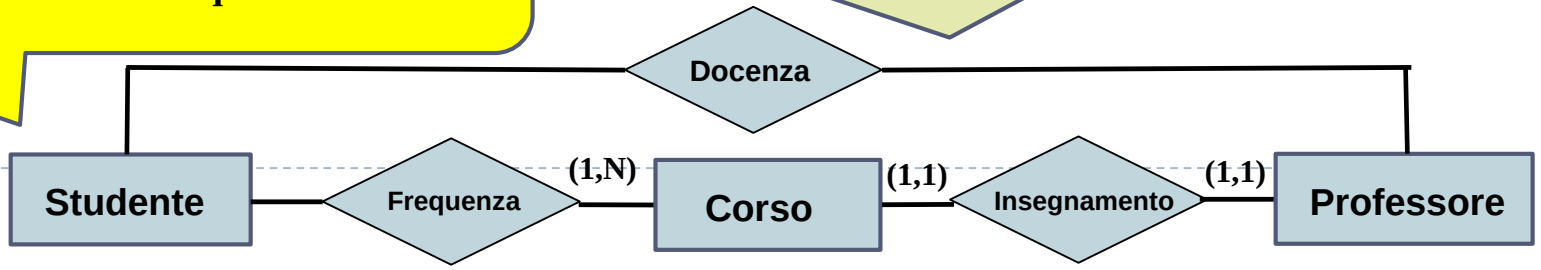
Attributi derivabili da attributi di altre entità o associazioni
L'entità **Acquisto** ha un attributo ***ImportoTotale*** che si può derivare, attraverso la relazione **Composizione,** dall'attributo ***Prezzo*** dell'entità **Prodotto**, sommando i prezzi dei prodotti di cui un acquisto è composto.



Attributi derivabili da operazioni di conteggio di istanze
L'attributo ***NumeroAbitanti*** di una città può essere derivato contando le istanze della relazione **Residenza**, a cui tale città partecipa.

Va però precisato che la presenza di cicli non genera necessariamente ridondanze. Se al posto di docenza ci fosse stata una relazione **Tesi** rappresentante il legame tra studenti e relatori (concetto indipendente dal fatto che un professore è un docente dello studente), allora lo schema non sarebbe stato ridondante.

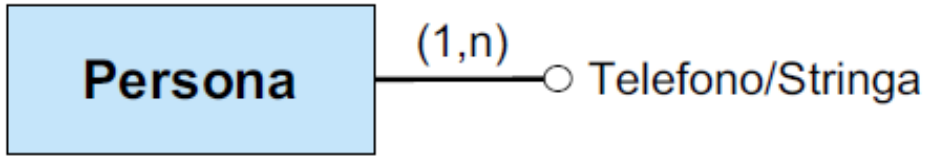
Attributi derivabili dalla composizione di altre associazioni in presenza di cicli
L'associazione **Docenza** tra Studenti e professori può essere derivata dalle relazioni **Frequenza** e **Insegnamento**.



Fase 2: eliminazione di attributi multivalore su entità

- Un'attributo multivalore 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

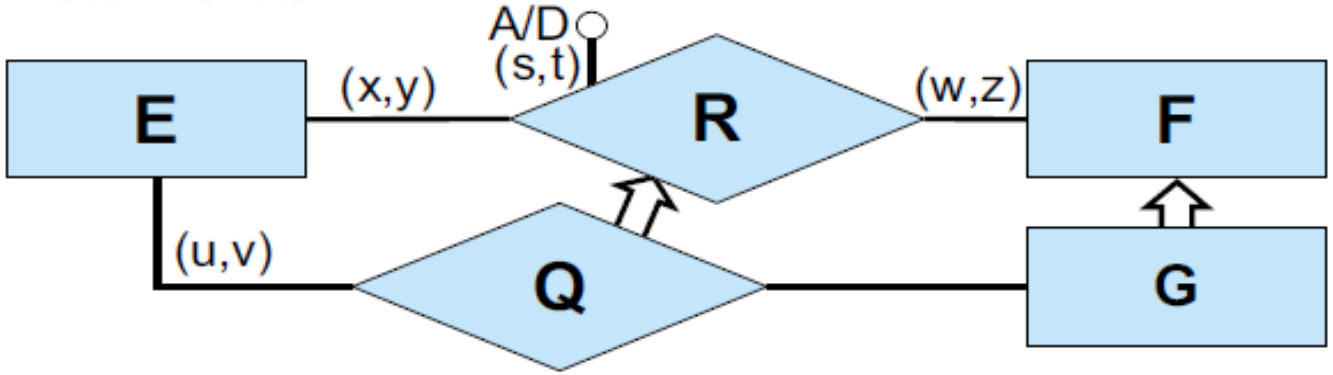
Esempio:



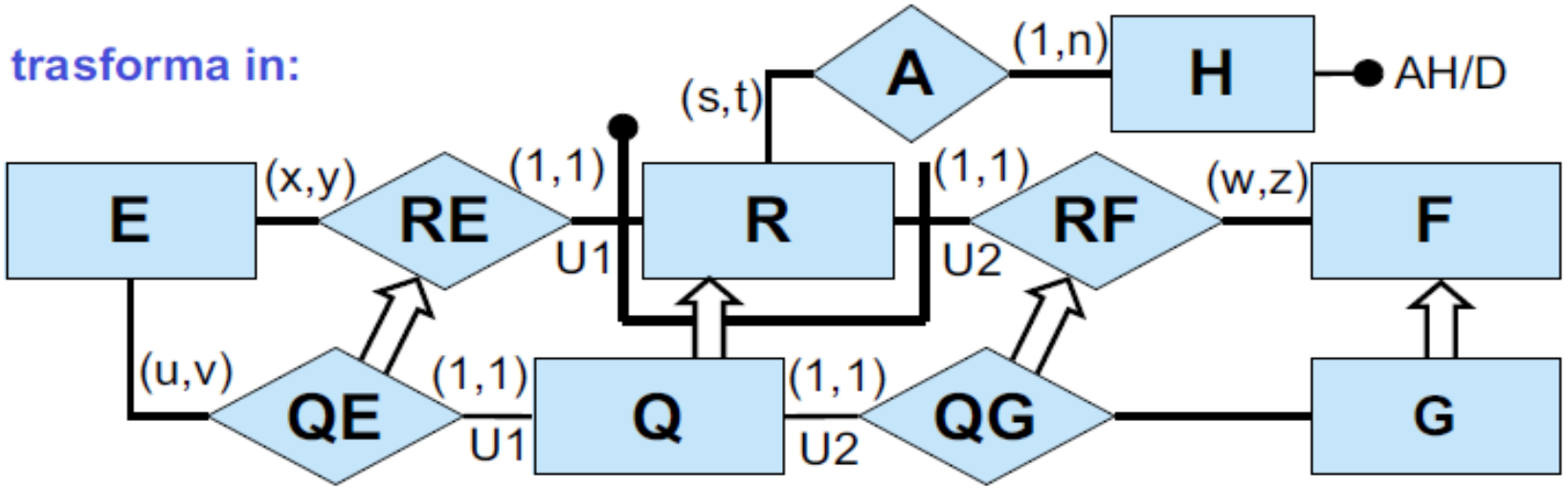
Si trasforma in:

Fase 2: eliminazione di attributi multivalore su 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.

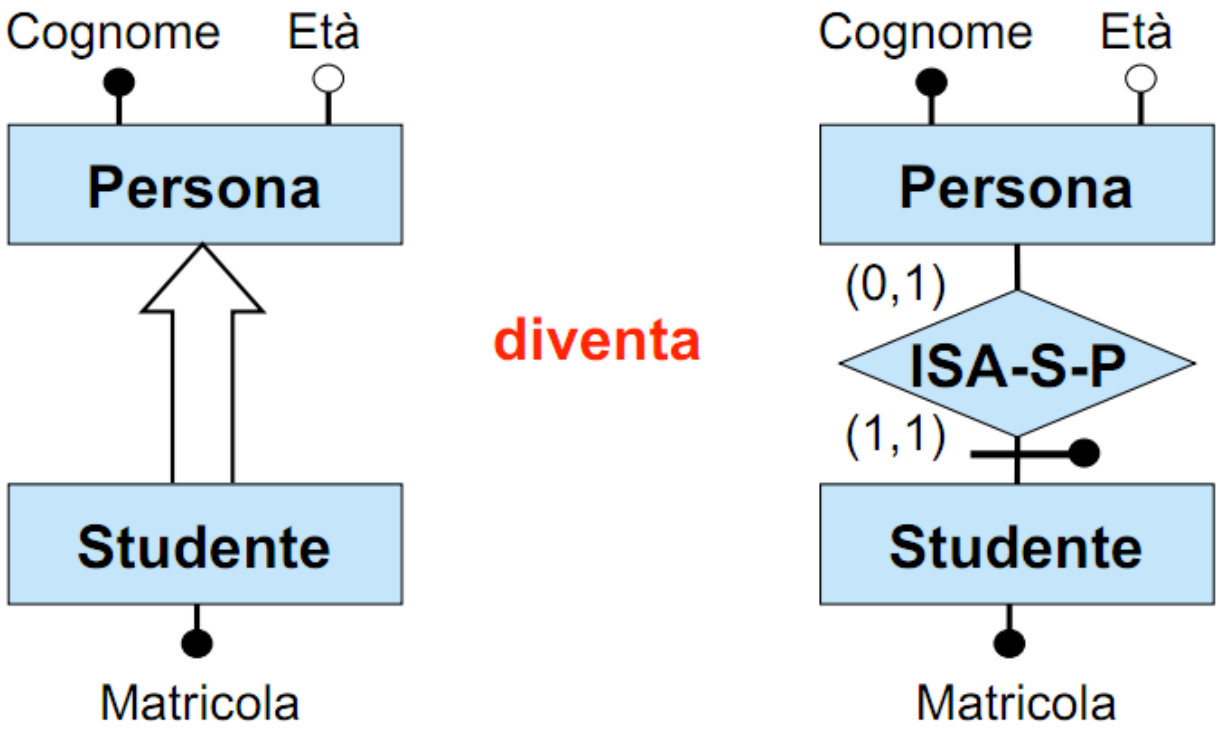


Si trasforma in:



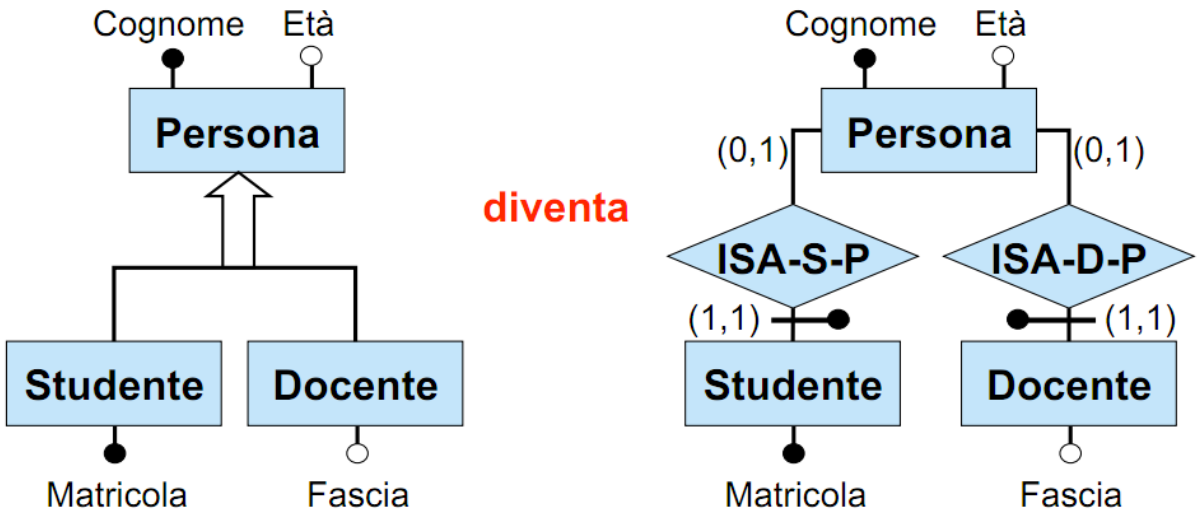
Fase 3: eliminazione di ISA tra entità

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



Fase 4: eliminazione di generalizzazioni tra entità

- Una generalizzazione **non completa** tra un'entità padre **F** e le sottoentità **E1,E2,…,En,** viene trattata come n relazioni **E1 ISA F,…, En ISA F,** introducendo n relazioni binarie **ISA-E1-F,…, ISA-En-F.**
- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni, detti **vincoli di generalizzazione:**



VANTAGGI

- Viene praticamente lasciata immutata la struttura dello schema

SVANTAGGI

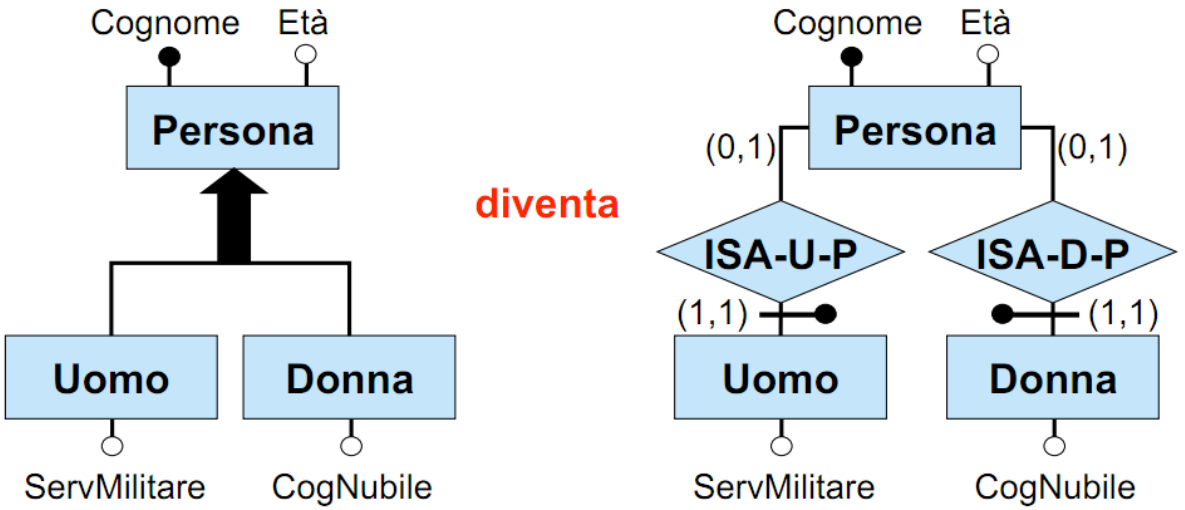
- Ridondanza (vengono lasciate tutte le entità)
- Operazioni di aggiornamento molto complesse
- Spreco di memoria

Vincoli di generalizzazione:

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

Fase 4: eliminazione di generalizzazioni tra entità

- Una generalizzazione **completa** tra un'entità padre **F** e le sottoentità **E1,E2,…,En,** viene trattata come n relazioni **E1 ISA F,…, En ISA F,** introducendo n relazioni binarie **ISA-E1-F,…, ISA-En-F.**
- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni, detti **vincoli di generalizzazione:**

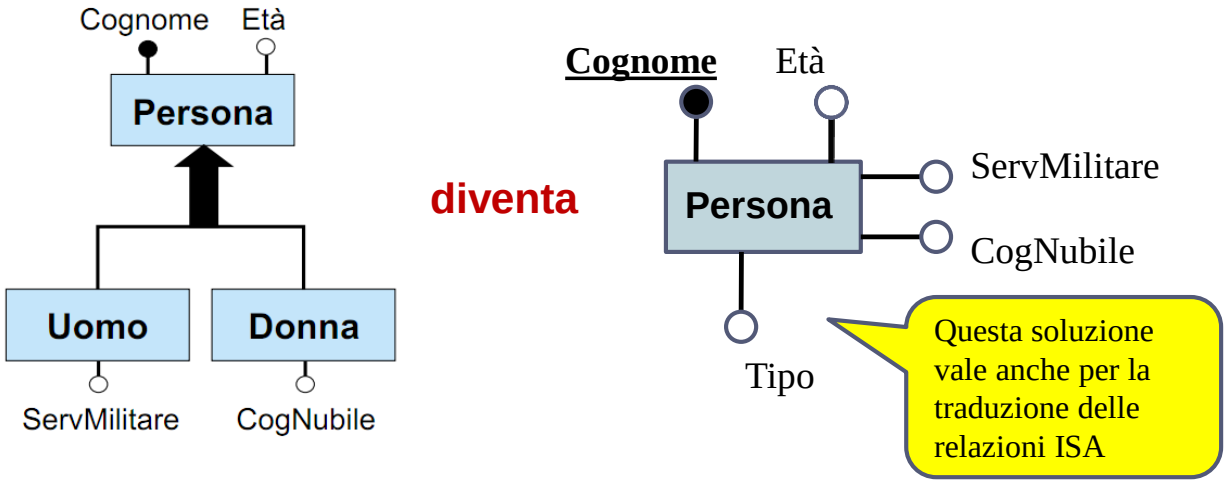


Vincoli di generalizzazione:

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

Fase 4: eliminazione di generalizzazioni tra entità – *soluzione 2 : fusione*

- Una generalizzazione (completa o non completa) tra un'entità padre **F** e le sottoentità **E1,E2,…,En,** può venire trattata come un'unica entità **F** in cui vengano riportati tutti gli attributi di F e tutti gli attributi di **E1,E2,...,En**



VANTAGGI

- Informazioni raggruppate in un'unica entità

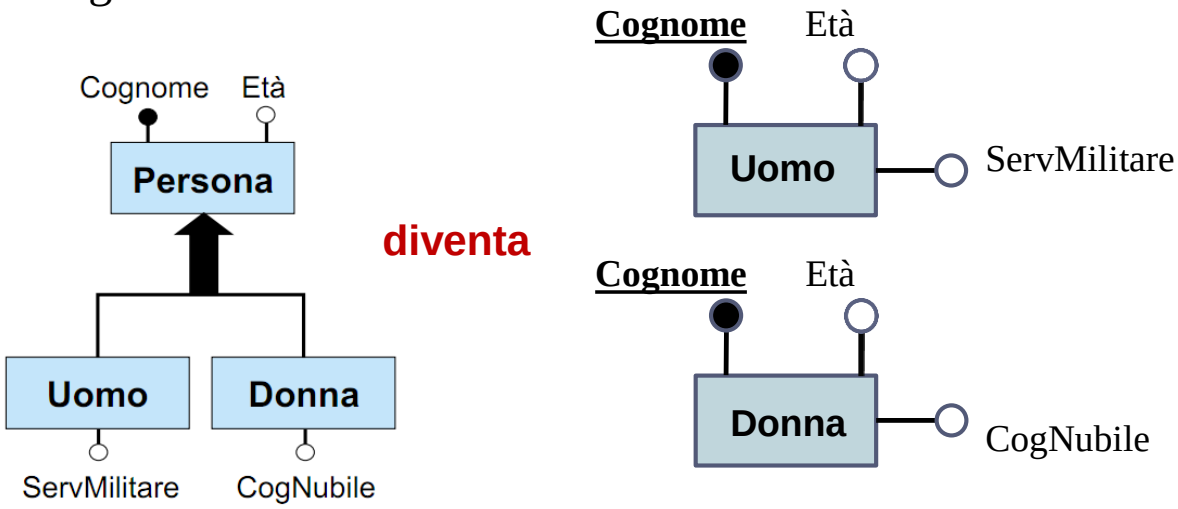
SVANTAGGI

- Grande occupazione di memoria quando le sottoentità hanno molti attributi
- Inutile se l'entità principale ha pochi attributi e/o relazioni
- Valori ***null***
- Necessità di procedure per il controllo dell'integrità

- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni :
  - *Persona.Tipo = Uomo* oppure *Persona.Tipo = Donna* (**solo se** la generalizzazione è **completa**)
  - se *Persona.Tipo = Uomo* , allora *ServMilitare* IS NOT NULL e *CogNubile* IS NULL
  - se *Persona.Tipo = Donna* , allora *CogNubile* IS NOT NULL e *ServMilitare* IS NULL

Fase 4: eliminazione di generalizzazioni tra entità – *soluzione 3 : divisione*

- Una generalizzazione **completa** tra un'entità padre **F** e le sottoentità **E1,E2,…,En,** può venire trattata come n entità **E1,E2,...,En** in cui ciascuna entità **Ei** riporta tutti gli attributi di **F**



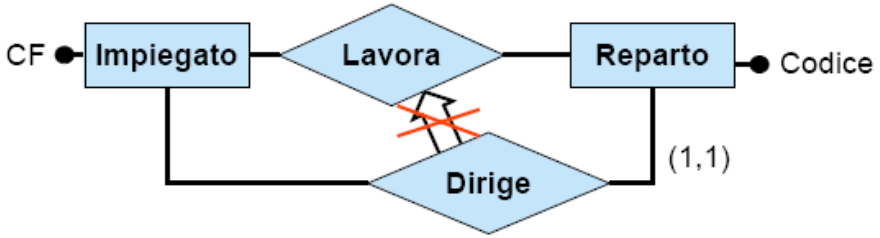
VANTAGGI
- Nessuno spreco di memoria

SVANTAGGI
- Necessità di procedure per il controllo dell'integrità

- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni :
  - Uomo(Cognome) ∩ Donna(Cognome) = $\emptyset$

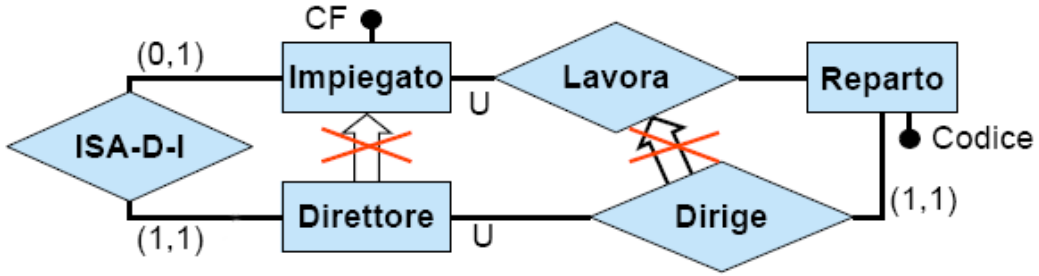
Fase 5: eliminazione ISA tra relazioni

- Nel caso in cui le relazioni in ISA insistano su esattamente le stesse entità per tutti i ruoli, è immediato esprimere il vincolo esterno



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

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



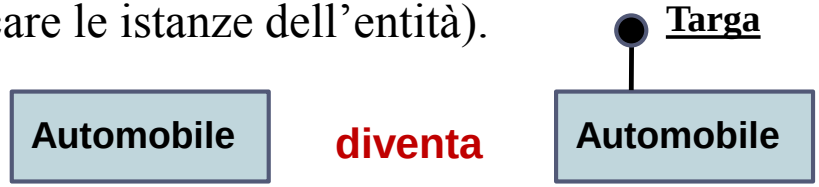
Vincolo esterno: per ogni istanza (U:*d*,Reparto:*r*) di Dirige, detta *i* l'istanza di Impiegato tale che (Direttore:*d*,Impiegato:*i*) è un'istanza di ISA-D-I (si noti che *i* esiste sempre ed è unica), si ha che (U:*i*,Reparto:*r*) è un'istanza di Lavora.

Fase 6: scelta degli identificatori principali

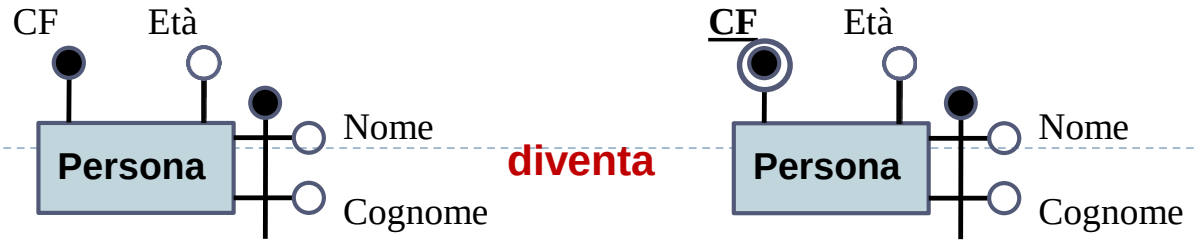
- Per **ogni entità è necessario:**
  - individuare almeno un identificatore
  - scegliere tra gli identificatori dell'entità un **identificatore principale.**
- **Criteri per la scelta dell'identificatore principale:**
  - semplicità (cioè formato da pochi attributi)
  - preferenza per gli identificatori interni
- 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.

Fase 6: scelta degli identificatori principali

- Per **ogni entità è necessario:**
  - individuare almeno un identificatore
  - scegliere tra gli identificatori dell'entità un **identificatore principale.**
- **Criteri per la scelta dell'identificatore principale:**
  - semplicità (cioè formato da pochi attributi)
  - preferenza per gli identificatori interni
- se per un'entità nessuno degli identificatori soddisfa tali requisiti, è possibile introdurre un ulteriore attributo dell'entità (un **codice,** i cui valori sono speciali ed hanno l'unico scopo di identificare le istanze dell'entità).



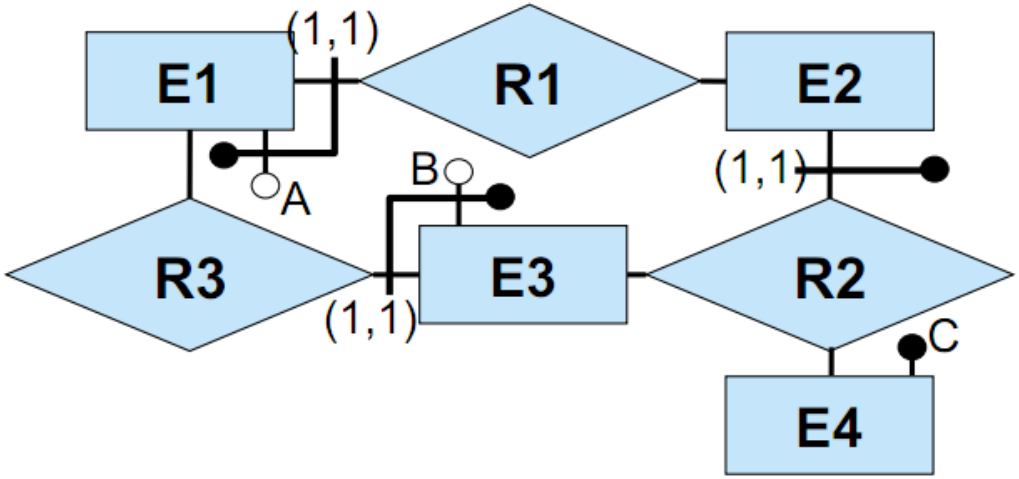
- **In una entità con più identificatori**, quello principale viene indicato nella documentazione associata allo schema ristrutturato. Sulle slide, in presenza di più identificatori per un'entità, denoteremo quello principale con un cerchio addizionale.

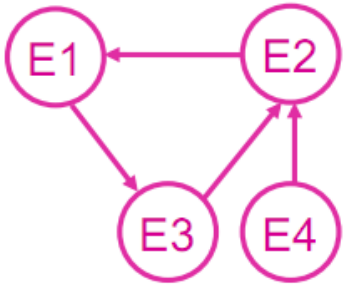


Cicli di Identificazione esterna

- Nella scelta degli identificatori principali è necessario fare attenzione a non introdurre cicli tra gli identificatori principali esterni.
- Definiamo il grafo degli identificatori (principali) esterni nel seguente modo:
  - ad ogni entità del diagramma corrisponde un nodo
  - c'è un arco dall'entità E all'entità F se e solo se E partecipa ad una relazione che è parte dell'identificatore (o che è identificatore) principale esterno di F.
  - **Si ha un ciclo di identificazione esterna quando il grafo degli identificatori principali esterni contiene un ciclo.**

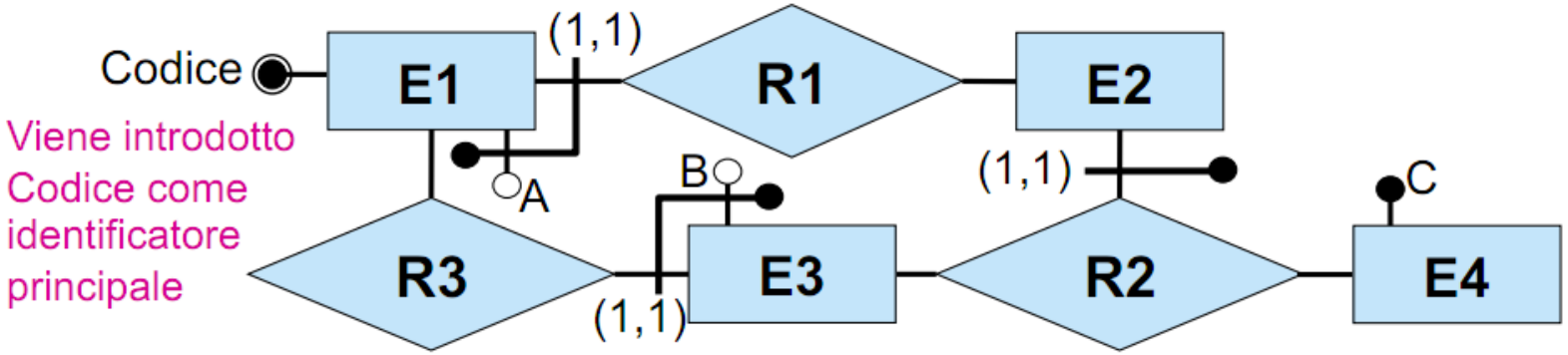
Cicli di Identificazione esterna





grafo degli identificatori principali esterni

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



Fase 7: specifica degli ulteriori vincoli esterni

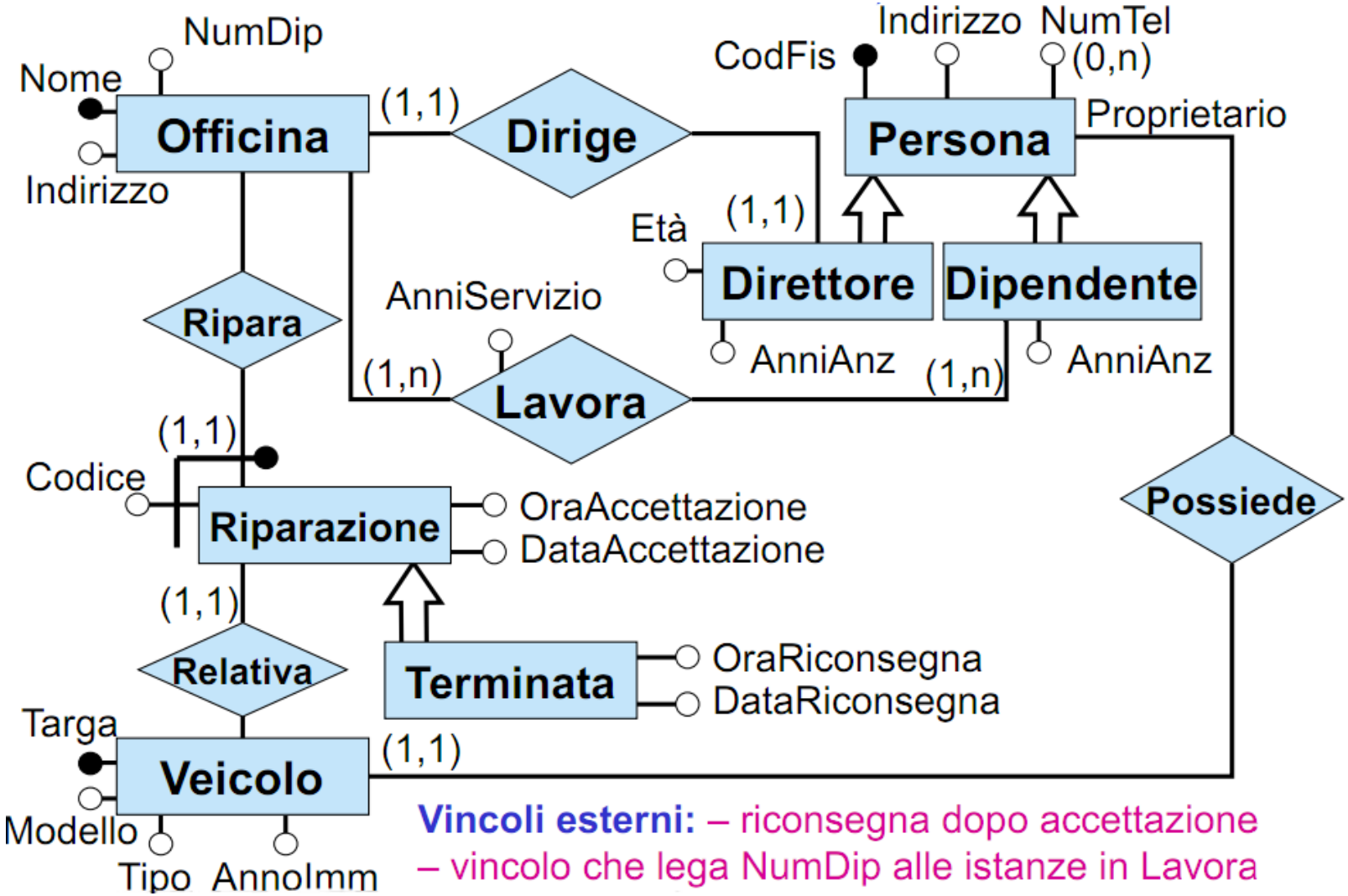
- **E' inoltre necessario** *riformulare* tutti i vincoli esterni dello schema originario in termini dello schema ristrutturato ed *aggiungere* i vincoli derivanti dalla ristrutturazione.

  - 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

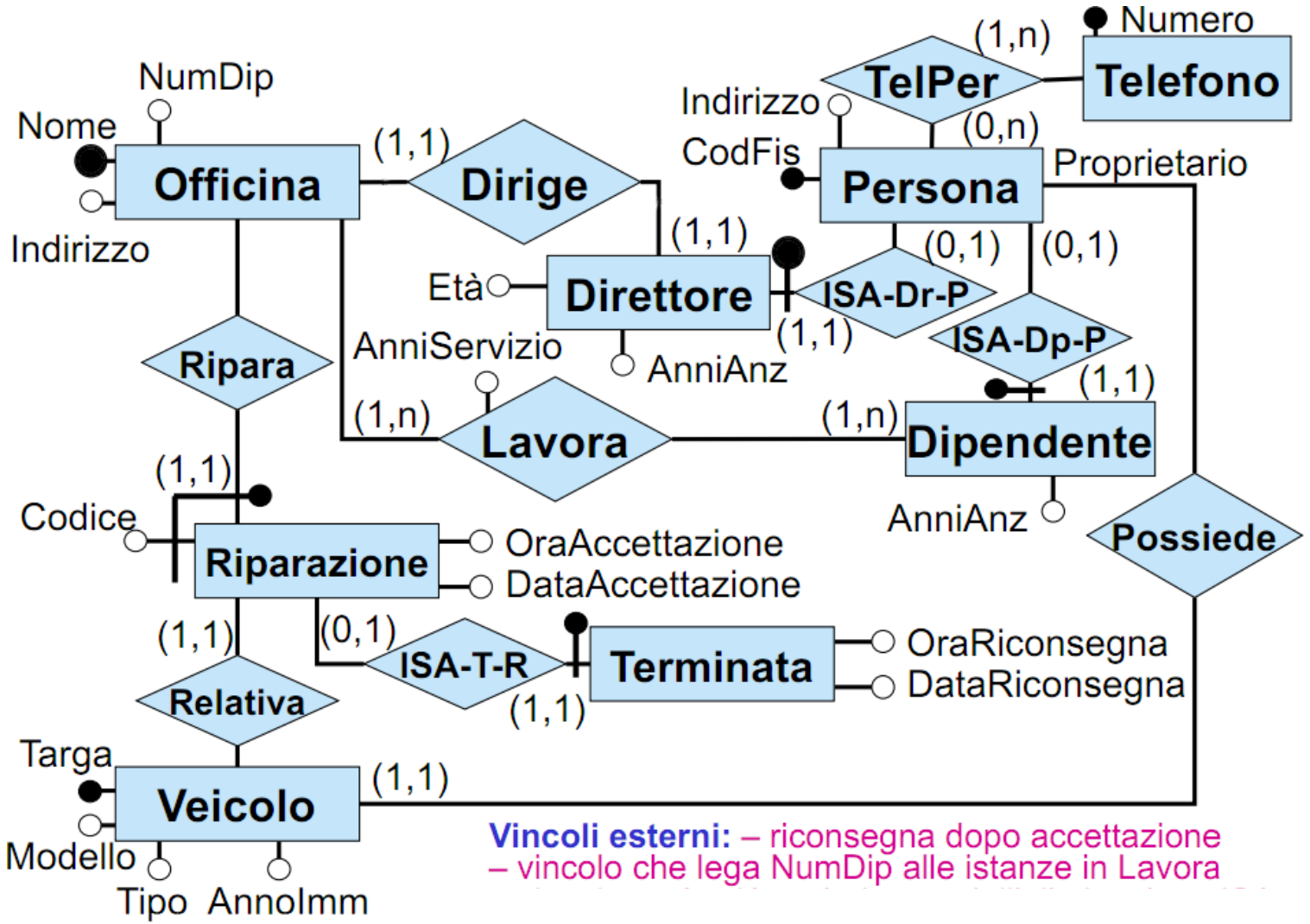
Fase 8: riformulazione di operazioni e carico applicativo

- Infine, **è opportuno** *riformulare* tutte le operazioni e le specifiche sul carico applicativo in termini dello schema ristrutturato

Esercizio – Ristrutturare il seguente schema



Soluzione Esercizio



2. Traduzione nel modello relazionale

- La fase di ristrutturazione ha prodotto uno schema ER ristrutturato con le seguenti proprietà:
  - preserva la semantica dello schema originale. Intuitivamente, esiste una funzione che associa ad ogni istanza dello schema originale un'opportuna istanza dello schema ristrutturato e viceversa.
  - può contenere delle ridondanze, ma sono volute per motivi di efficienza e sono comunque documentate.
  - non contiene attributi multivalore.
  - non contiene ISA o generalizzazione (nè tra entità, nè tra relazioni); quindi tutte le entità sono disgiunte a coppie.
  - tutte le entità hanno un **unico identificatore principale**.
- Lo schema E-R ristrutturato è il punto di partenza per la traduzione nel modello relazionale. La traduzione consiste delle seguenti **attività :**
  - traduzione delle **entità in relazioni dello schema logico,** con relativi vincoli.
  - traduzione delle **relazioni dello schema ER in relazioni dello schema logico**, con relativi vincoli.
  - traduzione dei **vincoli esterni**.
  - riformulazione di **operazioni e specifiche** sul carico applicativo in termini dello schema logico.

Notazione

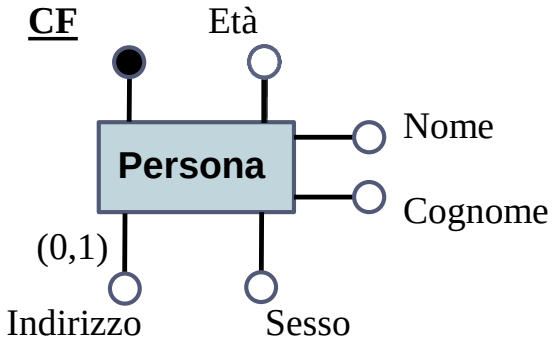
- Nella fase di traduzione diretta ed in quella di ristrutturazione dello schema logico, esprimeremo gli schemi relazionali mediante una notazione che prevede di descrivere le relazioni con nome e attributi, ed i vincoli ad esse associati in forma testuale.
- ***Esempio:***

- Ovviamente, questa notazione si può tradurre **senza difficoltà** in termini di "*create table*" in SQL.

Traduzione di entità – caso generale

- Consideriamo per ora un'entità singola.
  - Un'entità si traduce in una **relazione** dello schema relazionale.
  - Gli attributi della relazione corrispondente all'entità sono gli stessi dell'entità.
  - Se un attributo è opzionale (0,1) , allora viene tradotto in un attributo della relazione che può assumere valore nullo (nella nostra notazione per lo schema logico, tali attributi sono indicati con *); in caso contrario l'attributo non potrà assumere valore nullo.
  - L'identificatore principale dell'entità si traduce nella **chiave primaria** della relazione.
  - Gli altri identificatori interni si traducono in **chiavi della relazione**.

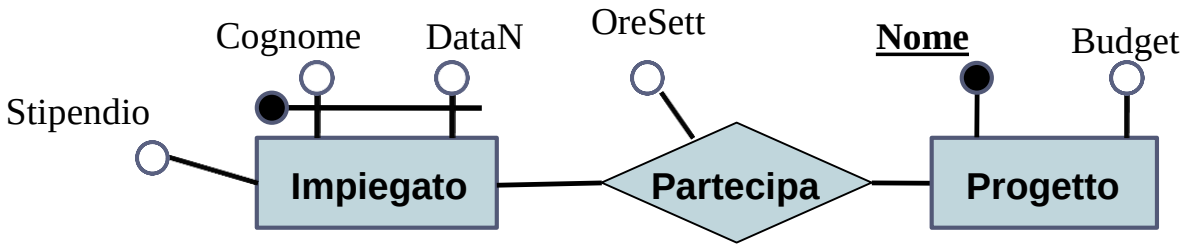
Persona(**CF**,Età,Nome,Cognome,Sesso,Indirizzo*)




```
CREATE TABLE Persona(
CF CHAR(10) PRIMARY KEY
Nome VARCHAR(20) NOT NULL,
Cognome VARCHAR(20) NOT NULL,
Età INT NOT NULL,
Sesso CHAR NOT NULL,
Indirizzo VARCHAR(20)
)
```

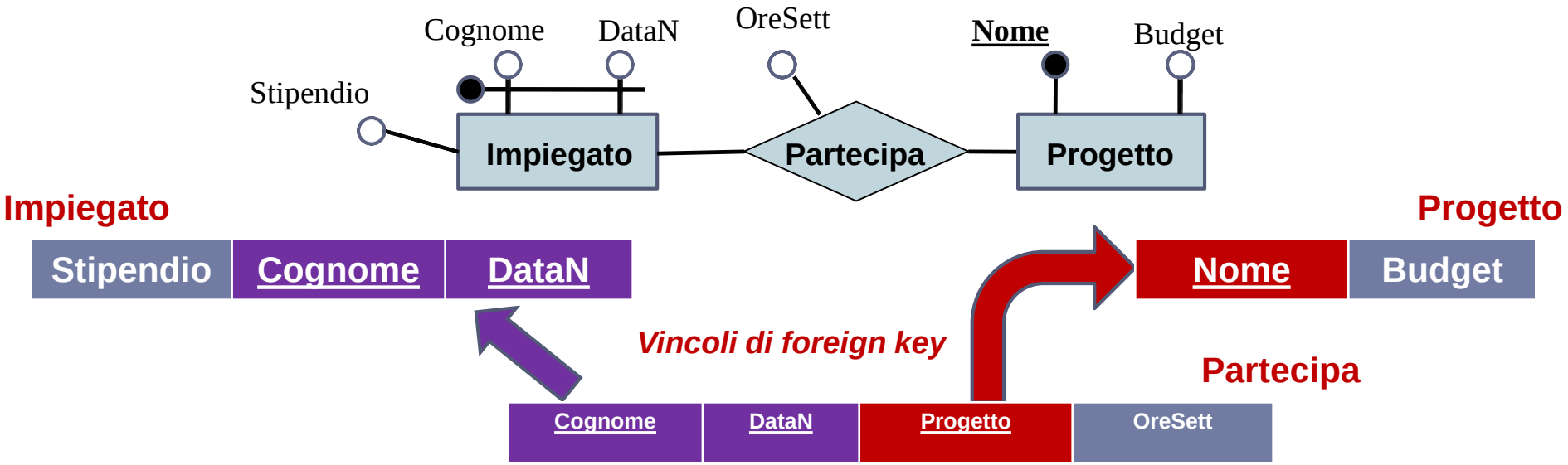
Traduzione di relazione con cardinalità (0,N)

- Consideriamo il caso di ER-relazione in cui tutte le cardinalità siano di tipo (0,n). La traduzione nel modello relazionale prevede :
  - per ogni entità, una **relazione** con lo stesso nome avente per attributi i medesimi attributi dell'entità e per chiave il suo identificatore.
  - per l'ER-relazione, una **relazione** con lo stesso nome avente per attributi gli attributi dell'ER-relazione e gli identificatori delle entità coinvolte; **tali identificatori formano la chiave della relazione.**
  - Si definiscono vincoli di *foreign key* dalla relazione relativa all'ER-relazione verso le entità partecipanti per dar conto dei vincoli di tipizzazione nello schema ER.



Partecipa(**Cognome,DataN,Progetto**,OreSett)
foreign key : Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] ⊆ Progetto[Nome]
Impiegato(**Cognome,DataN,**Stipendio)
Progetto(**Nome**,Budget)

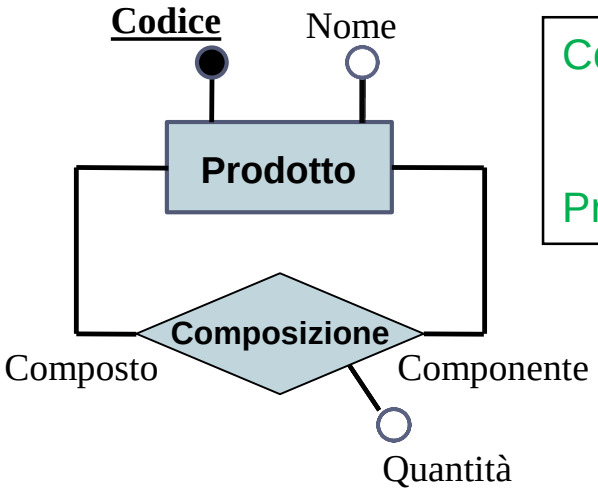
Traduzione di relazione con cardinalità (0,N)



- Per rendere più comprensibile il significato di uno schema relazionale, è conveniente effettuare alcune **ridenominazioni**; nel nostro esempio si può chiarire il contenuto della relazione **Partecipazione** definendo un attributo di nome '*Progetto*' (anziché chiamarlo '*Nome*').
- L'attributo *Progetto* di **Partecipazione** non è altro che un insieme di nomi di progetti, i cui valori saranno un sottoinsieme dei valori contenuti nell'attributo *Nome* di **Progetto**.

Partecipa(**Cognome,DataN,Progetto**,OreSett)
foreign key : Partecipa[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] $\subseteq$ Progetto[Nome]
Impiegato(**Cognome,DataN,**Stipendio)
Progetto(**Nome**,Budget)

Esempio di traduzione con cardinalità (0,N)



Composizione(**Composto,Componente**,Quantità)
foreign key : Composizione[Composto] ⊆ Prodotto[Codice]
foreign key : Composizione[Componente] ⊆ Prodotto [Codice]
Prodotto(**Codice**,Nome)

In questo caso la ridenominazione è essenziale. Entrambi gli attributi ***Composto*** e ***Componente*** contengono ***Codici*** di prodotti;il primo dei due ha il secondo come componente.

Prodotto

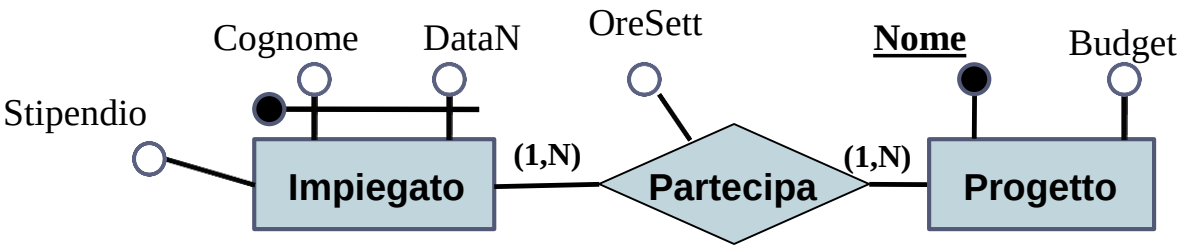
Codice	Nome
A	Seme di Cacao
B	Acqua
C	Zucchero
D	Latte
E	Cioccolata

Composizione

Composto	Componente	Quantità
A	B	2
A	C	3
E	A	100
E	D	2

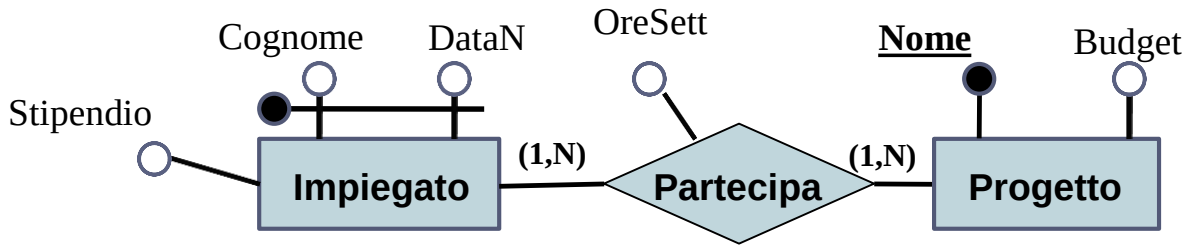
Traduzione di relazione con cardinalità minima 1

- Consideriamo il caso di un'ER-relazione in cui tutte le cardinalità massime siano N e le cardinalità minime siano pari ad 1.
- La traduzione avviene come nel caso della slide precedente, con una piccola aggiunta :
  - Un vincolo di cardinalità minima 1 per la partecipazione di un'entità (in un ruolo) alla relazione si traduce in un **vincolo di inclusione dall'entità verso l'attributo** (o gli attributi) **corrispondenti a quel ruolo nella relazione**.



Partecipa(**Cognome,DataN,Progetto**,OreSett)
foreign key : Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] ⊆ Progetto[Nome]
Impiegato(**Cognome,DataN,**Stipendio)
inclusione: Impiegato[Cognome,DataN] ⊆ Partecipa[Cognome,DataN]
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

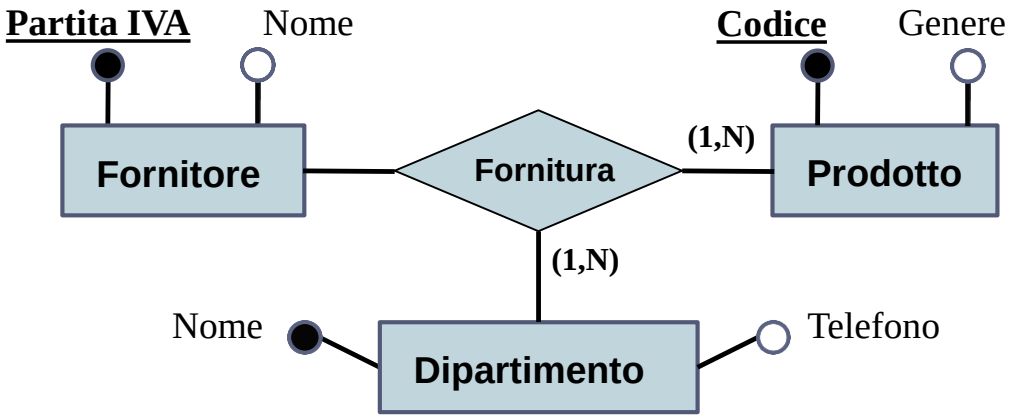
Esempio di vincolo di inclusione in SQL



Partecipa(**Cognome,DataN,Progetto**,OreSett)
foreign key : Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] ⊆ Progetto[Nome]
Impiegato(**Cognome,DataN,**Stipendio)
inclusione: Impiegato[Cognome,DataN] ⊆ Partecipa[Cognome,DataN]
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

```
CREATE TABLE Progetto(
Nome CHAR PRIMARY KEY,
Budget INT NOT NULL,
check ( Nome in SELECT Progetto FROM Partecipa)
)
```

Esempio di traduzione con relazione ternaria



Le relazioni con più di due entità partecipanti si traducono **in maniera analoga** a quanto detto per le associazioni binarie.

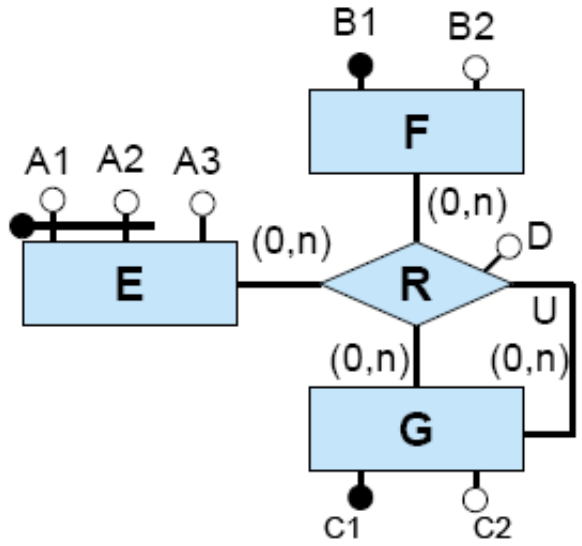
Fornitura(**Fornitore, Prodotto, Dipartimento**)
- foreign key : Fornitura[Fornitore] ⊆ Fornitore[Partita IVA]
- foreign key : Fornitura[Prodotto] ⊆ Prodotto[Codice]
- foreign key : Fornitura[Dipartimento] ⊆ Dipartimento [Nome]

Prodotto(**Codice**,Genere)
- inclusione : Prodotto[Codice] ⊆Fornitura[Prodotto]

Dipartimento(**Nome**, Telefono)
- inclusione : Dipartimento[Nome] ⊆Fornitura[Dipartimento]

Fornitore(**Partita IVA**, Nome)

Ulteriori esempi di traduzione



E(A1, A2, A3)

F(B1, B2)

G(C1, C2)

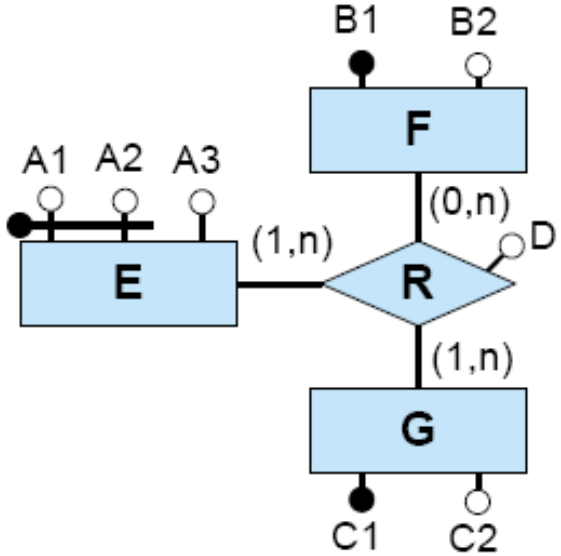
R(EA1, EA2, F, G, U, D)

foreign key: R[EA1,EA2] ⊆ E[A1,A2]

foreign key: R[F] ⊆ F[B1]

foreign key: R[G] ⊆ G[C1]

foreign key: R[U] ⊆ G[C1]



E(A1, A2, A3)

inclusione: E[A1,A2] ⊆ R[A1,A2]

F(B1, B2)

G(C1, C2)

inclusione: G[C1] ⊆ R[C1]

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

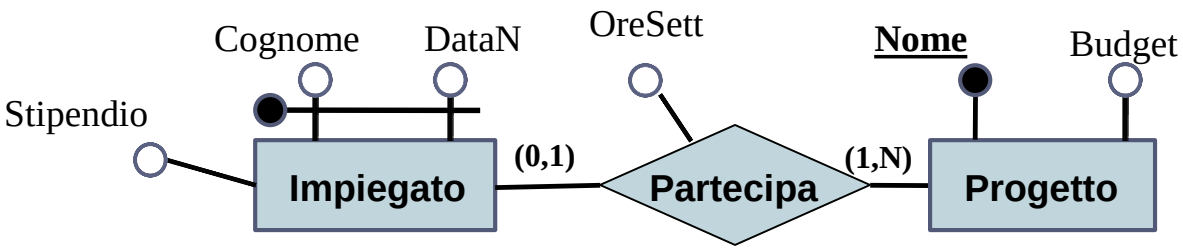
foreign key: R[A1,A2] ⊆ E[A1,A2]

foreign key: R[B1] ⊆ F[B1]

foreign key: R[C1] ⊆ G[C1]

Traduzione di relazione con cardinalità massima 1

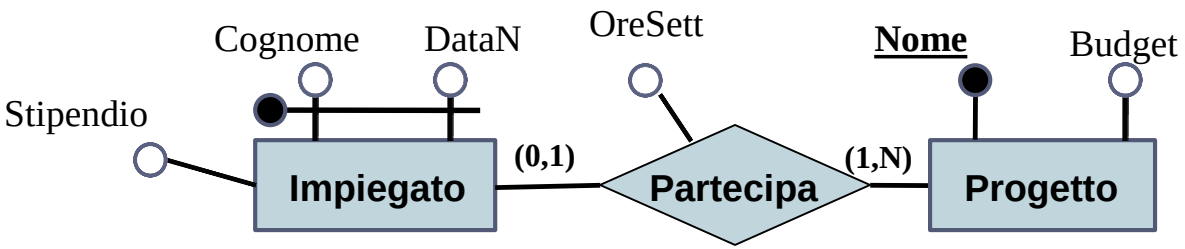
- Consideriamo il caso di ER-relazione in cui **almeno una** cardinalità massima sia 1.
- In questo caso, **si può fissare la chiave primaria della relazione relativa all'ER-relazione come l'unione delle chiavi primarie delle entità partecipanti**. In aggiunta, ciascuna chiave primaria ereditata da quelle entità legate alla relazione *con vincolo di cardinalità massimo* 1 **dovrà essere anche chiave della relazione.**



Partecipa(**Cognome,DataN,Progetto**,OreSett)
foreign key : Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] ⊆ Progetto[Nome]
chiave : Cognome, DataN
Impiegato(**Cognome,DataN,**Stipendio)
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

Traduzione di relazione con cardinalità massima 1 - **ALTERNATIVA**

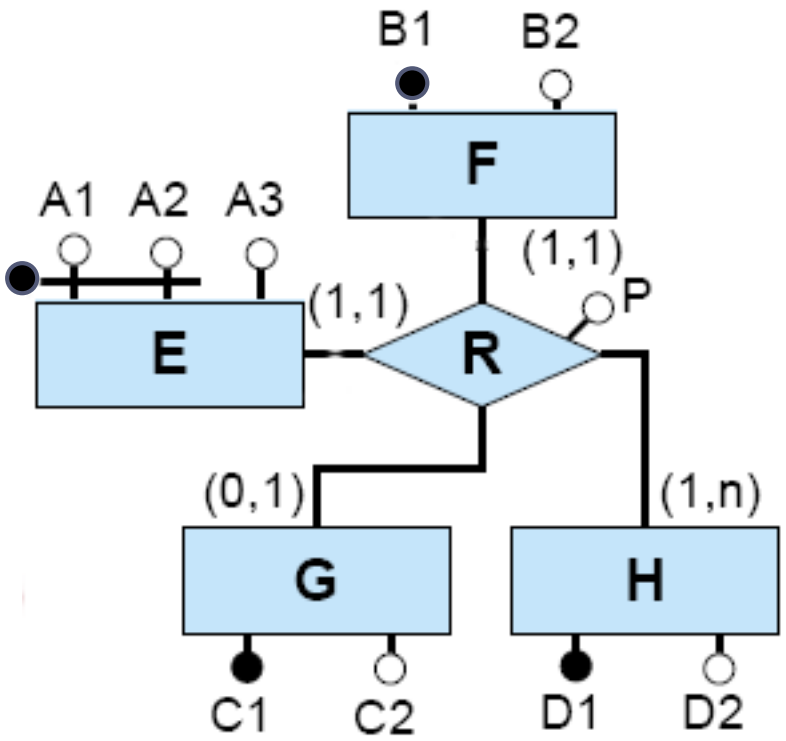
- Consideriamo una traduzione alternativa al caso precedente.
- Se vi è più di un'entità con cardinalità massima 1, **bisogna scegliere la chiave primaria della ER-relazione tra le chiavi primarie delle entità legate con vincolo di cardinalità massima 1 alle ER-relazione stessa**. Le chiavi primarie delle entità diverse da quella scelta per la ER-relazione si **traducono in semplici vincoli di chiave per la relazione**.



Partecipa(**Cognome,DataN,**Progetto,OreSett)
foreign key : Partecipa[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
foreign key : Partecipa[Progetto] ⊆ Progetto[Nome]
Impiegato(**Cognome,DataN,**Stipendio)
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] ⊆ Partecipa[Progetto]

Esempio di traduzione

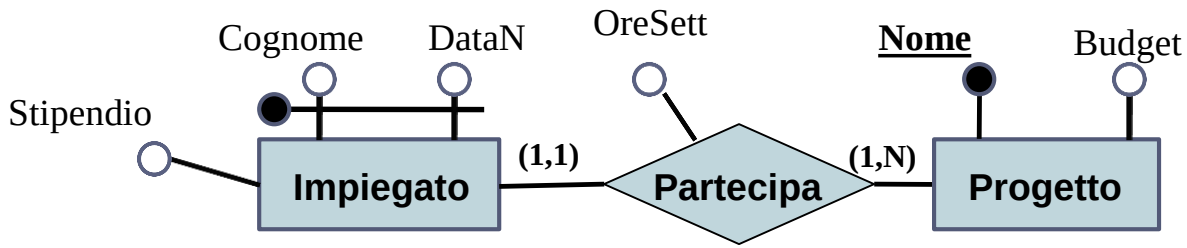
Solitamente, le E-R relazioni n-arie si traducono automaticamente in relazioni SQL, senza effettuare alcun accorpamento



E(A1, A2, A3)
inclusione: E[A1,A2] ⊆ R[EA1,EA2]
F(B1, B2)
inclusione: F[B1] ⊆ R[F]
G(C1, C2)
H(D1, D2)
inclusione: H[D1] ⊆ R[H]
R(EA1, EA2, F, G, H, P)
foreign key: R[EA1,EA2] ⊆ E[A1,A2]
foreign key: R[F] ⊆ F[B1]
foreign key: R[G] ⊆ G[C1]
foreign key: R[H] ⊆ H[D1]
chiave: EA1, EA2 chiave: F

Traduzione di relazione con cardinalità (1,1)

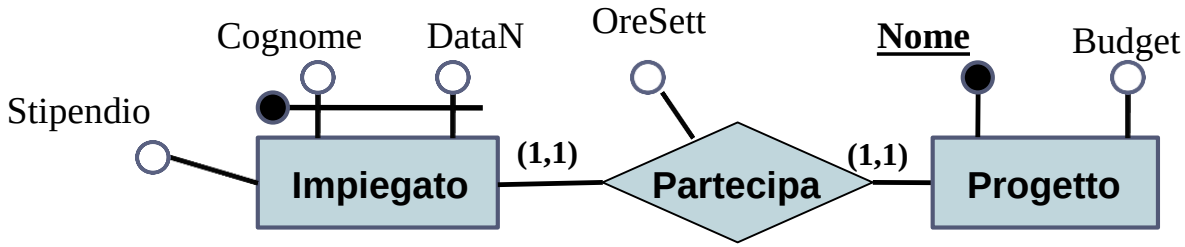
- Consideriamo il caso di ER-relazione binaria in cui **una** cardinalità sia (1,1)
- In questo caso, nell'effettuare la traduzione, la E-R relazione viene **accorpata** nell'entità che partecipa ad essa con cardinalità (1,1)



Impiegato(**Cognome,DataN,**Stipendio,Progetto,OreSett)
foreign key : Impiegato[Progetto] $\subseteq$ Progetto[Nome]
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] $\subseteq$ Impiegato[Progetto]

Traduzione di relazione con cardinalità (1,1)

- Consideriamo il caso di ER-relazione binaria in cui **entrambe** le cardinalità sono (1,1)
- In questo caso, nell'effettuare la traduzione, la E-R relazione può essere **accorpata** in **una** delle entità che partecipano ad essa con cardinalità (1,1). Nel caso dell'esempio, l'accorpamento può essere effettuato o dentro **Impiegato** oppure dentro **Progetto**.



Impiegato(**Cognome,DataN,**Stipendio,Progetto,OreSett)
foreign key : Impiegato[Progetto] ⊆ Progetto[Nome]
chiave : Progetto
Progetto(**Nome**,Budget)
inclusione: Progetto[Nome] ⊆ Impiegato[Progetto]

Un'istanza di **Progetto** partecipa **al massimo una volta** alla relazione "Partecipa" con un'istanza di **Impiegato**. Ciò significa che l'attributo *Progetto* dentro **Impiegato** deve essere posto a **chiave**

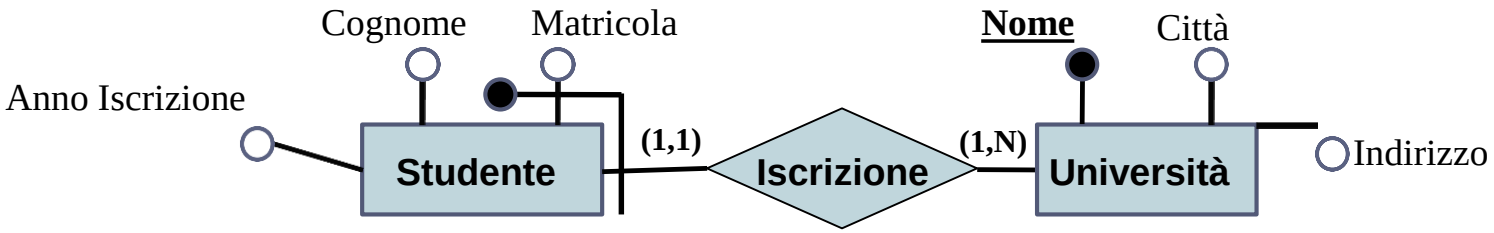
Un'istanza di **Progetto** partecipa **almeno una volta** alla relazione "Partecipa" con un'istanza di **Impiegato**. Ciò significa che ogni valore dell'attributo *Nome* dentro **Progetto** deve essere presente (da qui deriva l'**inclusione)** tra i valori dell'attributo *Progetto* dentro **Impiegato**

OPPURE

Impiegato(**Cognome,DataN,**Stipendio)
inclusione: Impiegato[Conome,DataN] ⊆ Progetto[Cognome,DataN]
Progetto(**Nome**,Cognome,DataN,Budget,OreSett)
foreign key: Progetto[Cognome,DataN] ⊆ Impiegato[Cognome,DataN]
chiave: Cognome, DataN

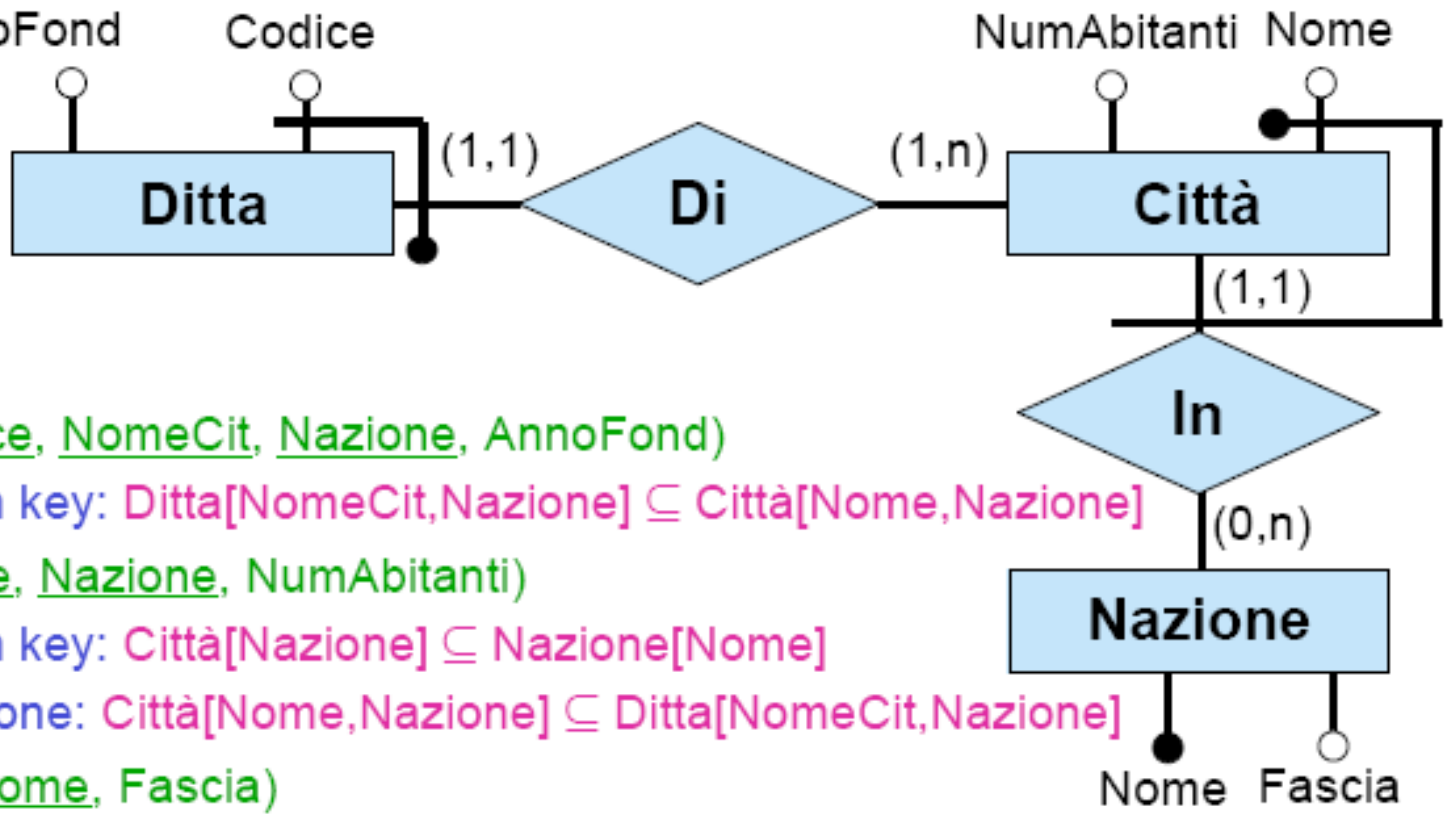
Traduzione di relazioni con entità debole

- Consideriamo il caso di ER-relazione binaria in cui **un'entità E è debole**
- La ER-relazione **viene accorpata nell'entità debole E.** Questo significa che tutti gli attributi della ER-relazione e le chiavi primarie delle altre entità partecipanti alla ER-relazione diventano attributi della relazione che corrisponde all'entità **E.** Tali chiavi primarie **fanno parte** della chiave primaria della relazione che corrisponde all'entità E.



Studente(**Matricola,Università**,Cognome,AnnoIscrizione)
foreign key : Studente[Università] ⊆ Università[Nome]
Università(**Nome**,Città,Indirizzo)
inclusione: Università[Nome] ⊆ Studente[Università]

Traduzione di relazioni con entità debole



Ditta(Codice, NomeCit, Nazione, AnnoFond)

foreign key: Ditta[NomeCit,Nazione] ⊆ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

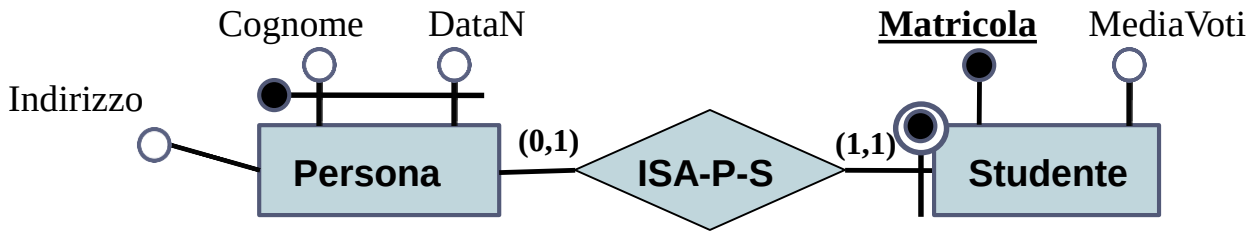
foreign key: Città[Nazione] ⊆ Nazione[Nome]

inclusione: Città[Nome,Nazione] ⊆ Ditta[NomeCit,Nazione]

Nazione(Nome, Fascia)

Traduzione di ISA

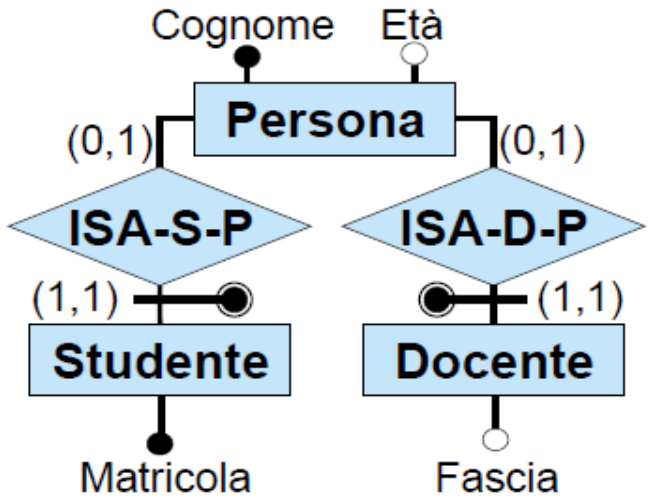
- Consideriamo il caso di ER-relazione binaria **R** derivante dalla ristrutturazione di un'ISA tra due relazioni nello schema originale
- Si noti come la traduzione della parte di schema ER che si ottiene dalla ristrutturazione di **E ISA F** corrisponda ad aggiungere agli attributi di E la chiave primaria di F**,** ed a rendere tali attributi anche chiave primaria di E
- Il vincolo derivante dall'ISA dello schema ER originario diventa quindi un vincolo di foreign key dello schema logico



Studente(Cognome, DataN, Matricola, MediaVoti)
foreign key: Studente[Cognome,DataN] ⊆ Persona[Cognome,DataN]
chiave: Matricola
Persona(Cognome, DataN, Indirizzo)

Traduzione di Generalizzazione non completa

- Consideriamo il caso di ER-relazione binaria **R** derivante da una generalizzazione **non completa** nello schema originale con padre E e figli F1 ed F2
- I vincoli di generalizzazione **devono essere espressi** in forma insiemistica



Vincolo di generalizzazione:

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

Diventa sullo schema logico:

Studente[Cognome] ∩ Docente[Cognome] = ∅

Persona(Cognome, Età)

Studente(Cognome, Matricola)

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

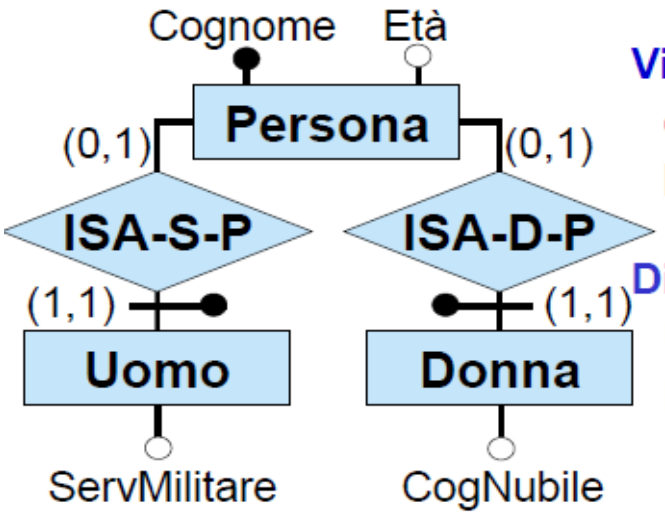
chiave: Matricola

Docente(Cognome, Fascia)

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

Traduzione di Generalizzazione completa

- Consideriamo il caso di ER-relazione binaria **R** derivante da una generalizzazione **completa** nello schema originale con padre E e figli F1 ed F2
- I vincoli di generalizzazione **devono essere espressi** in forma insiemistica



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:

$Uomo[Cognome] \cap Donna[Cognome] = \varnothing$

$Persona[Cognome] \subseteq Uomo[Cognome] \cup Donna[Cognome]$

Persona(Cognome, Età)

Uomo(Cognome, ServMilitare)

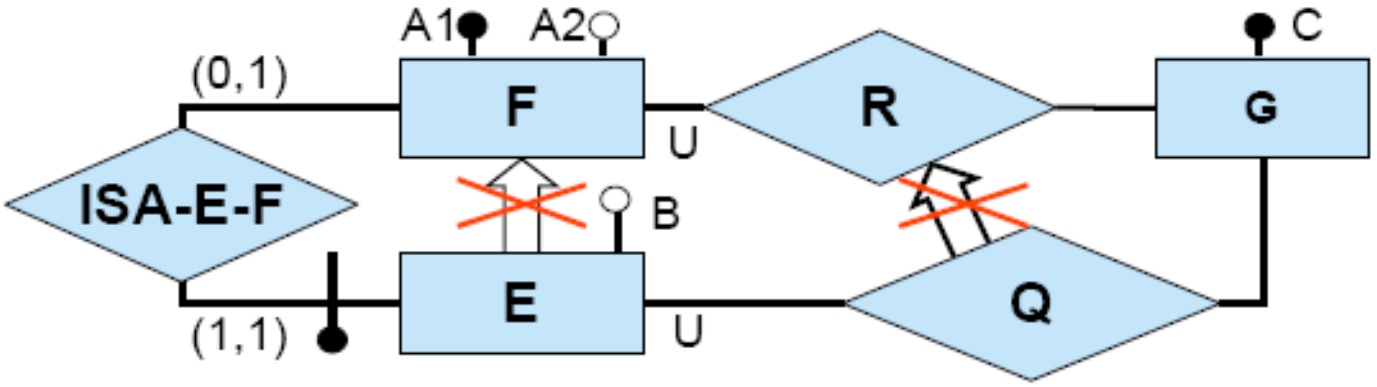
foreign key: $Uomo[Cognome] \subseteq Persona[Cognome]$

Donna(Cognome, CogNubile)

foreign key: $Donna[Cognome] \subseteq Persona[Cognome]$

Traduzione di vincoli derivanti da ISA tra relazioni

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



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

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

E(A1, B) foreign key: E[A1] ⊆ F[A1]

F(A1, A2)

G(C)

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

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

ESERCIZIO D'ESAME

1) Si effettui la progettazione concettuale della base di dati secondo la specifica che segue fornendo un diagramma ER.

Si vuole costruire un'applicazione che memorizzi i dati relativi allo svolgimento del campionato di calcio di Serie A. Per ogni partita in programma, si vogliono descrivere la giornata in cui si è svolta, il risultato finale, il numero progressivo della partita nella giornata (es. prima partita, seconda partita, ecc), la data ed il numero della giornata . Si suppone che le partite relative ad una giornata siano giocate tutte nella stessa data. Inoltre, per ogni partita, si vogliono conoscere

- il nome, il cognome e la città di nascita dell'arbitro della partita ;
- le squadre coinvolte nella partita (distinguendo se la squadra ha giocato in casa o in trasferta), con nome, città della squadra e allenatore.

Si memorizzi, per ogni giornata, quanti punti ha ogni squadra. Si devono poi distinguere le partite giocate regolarmente da quelle rinviate. Per quelle rinviate, rappresentare la data in cui si sono effettivamente giocate ed evidenziare se tali partite vengono giocate in una città diversa da quella della squadra ospitante; per queste ultime si vuole rappresentare la città in cui si svolgono, nonché il motivo della variazione di sede.

Infine si vogliono conoscere i giocatori che giocano in ogni squadra con i loro nomi e cognomi, la loro data e città di nascita e il loro ruolo principale. Si vuole sapere, per ogni partita, i giocatori che hanno giocato, i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita). Si vogliono rappresentare anche i seguenti vincoli : 1) non ci possono essere più di 5 giocatori in una squadra che giocano nello stesso ruolo e 2) se una squadra gioca in casa una partita, allora è ospite nella partita successiva

ESERCIZIO D'ESAME

2) Si effettui la progettazione logica del diagramma ER realizzato con riferimento alla domanda 1

3) Si fornisca il codice DDL di creazione delle tabelle.

Come tradurre le specifiche in un diagramma E-R?

Un primo passo che si può mettere in atto per migliorare la comprensione della specifica consiste nel riorganizzare in gruppi omogenei i dati mostrati nella specifica stessa:

- ***FRASI RELATIVE ALLA PARTITA E ALLA GIORNATA***

Per ogni partita in programma, si vogliono descrivere la giornata in cui si è svolta, il risultato finale, il numero progressivo della partita nella giornata (es. prima partita, seconda partita, ecc), la data ed il numero della giornata . Si suppone che le partite relative ad una giornata siano giocate tutte nella stessa data. Si devono poi distinguere le partite giocate regolarmente da quelle rinviate. Per quelle rinviate, rappresentare la data in cui si sono effettivamente giocate ed evidenziare se tali partite vengono giocate in una città diversa da quella della squadra ospitante; per queste ultime si vuole rappresentare la città in cui si svolgono, nonché il motivo della variazione di sede

- ***FRASI RELATIVE ALL'ARBITRO***

Per ogni partita, si vogliono conoscere il nome, cognome e città di nascita dell'arbitro della partita

Come tradurre le specifiche in un diagramma E-R?

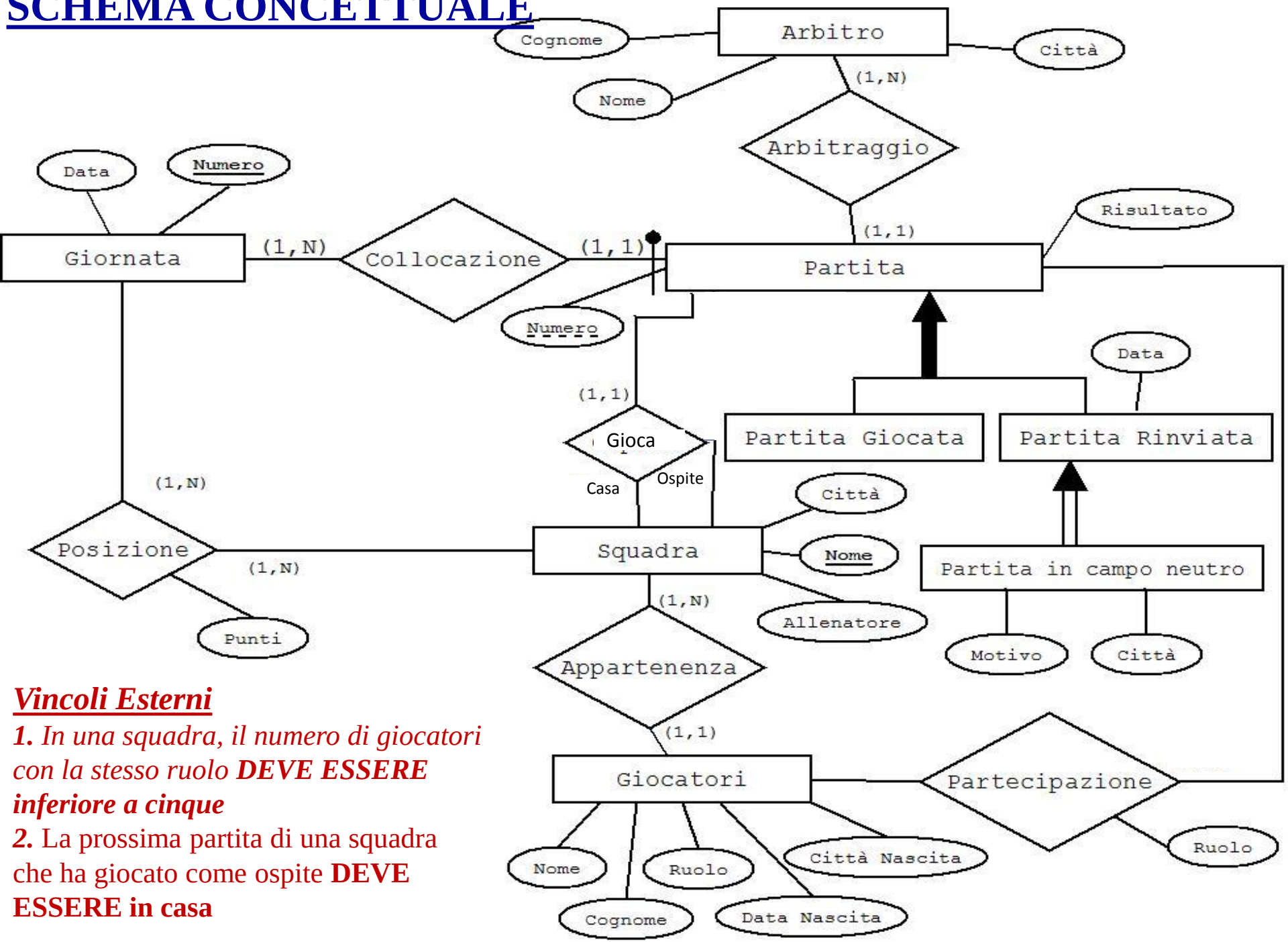
- ***FRASI RELATIVE ALLE SQUADRE***

Per ogni partita, si vogliono conoscere le squadre coinvolte nella partita (distinguendo se la squadra ha giocato in casa o in trasferta), con nome, città della squadra e allenatore. Si memorizzi, per ogni giornata, quanti punti ha ogni squadra. non ci possono essere più di 5 giocatori in una squadra che giocano nello stesso ruolo. Se una squadra gioca in casa una partita, allora è ospite nella partita successiva

- ***FRASI RELATIVE AI GIOCATORI***

Infine si vogliono conoscere i giocatori che giocano in ogni squadra con i loro nomi e cognomi, la loro data e città di nascita e il loro ruolo principale. Si vuole sapere, per ogni partita, i giocatori che hanno giocato ed i ruoli di ogni giocatore (i ruoli dei giocatori possono cambiare di partita in partita)

SCHEMA CONCETTUALE

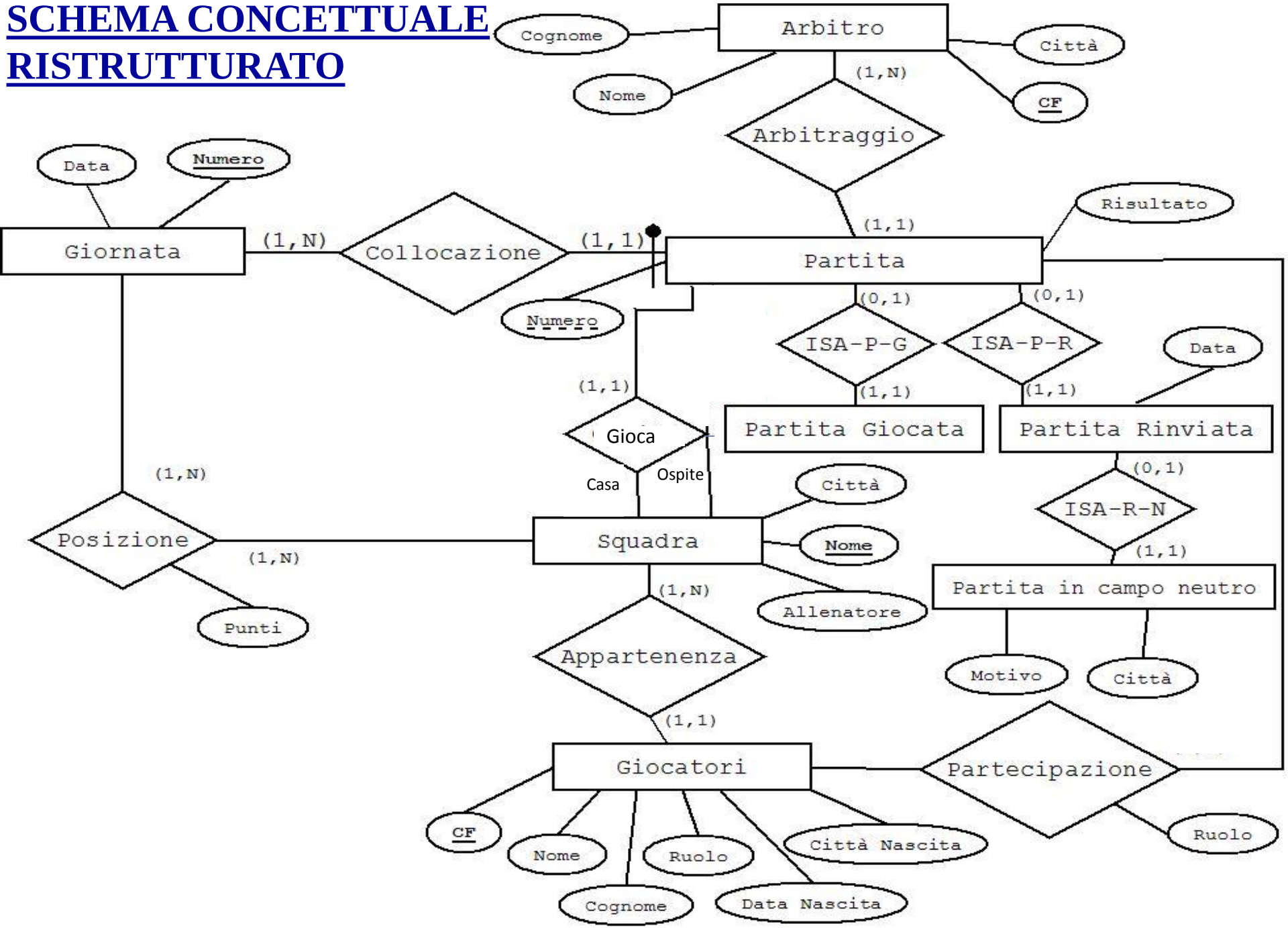


Vincoli Esterni

1. *In una squadra, il numero di giocatori con la stesso ruolo* ***DEVE ESSERE inferiore a cinque***

2. La prossima partita di una squadra che ha giocato come ospite **DEVE ESSERE in casa**

SCHEMA CONCETTUALE RISTRUTTURATO



Vincoli sullo schema ristrutturato

- In una squadra, il numero di giocatori con la stesso ruolo DEVE ESSERE inferiore a cinque
- La prossima partita di una squadra che ha giocato come ospite DEVE ESSERE in casa
- Ogni istanza di Partita partecipa ad ISA-P-G oppure ad ISA-P-R ma non ad entrambe (dovuto all'eliminazione della generalizzazione)

Schema Logico – Parte 1

ARBITRO(**<u>CF</u>**, Cognome, Nome, Città)

inclusione : Arbitro[CF] $\subseteq$ Partita[Arbitro]

PARTITA(**<u>NumeroPartita, NumeroGiornata</u>,** Risultato, Arbitro, Casa, Ospite)

foreign key : Partita[NumeroGiornata] $\subseteq$ Giornata[Numero]

foreign key : Partita[Arbitro] $\subseteq$ Arbitro[CF]

foreign key : Partita[Casa] $\subseteq$ Squadra[Nome]

foreign key : Partita[Ospite] $\subseteq$ Squadra[Nome]

GIORNATA(**<u>Numero</u>**, Data)

inclusione : Giornata[Numero] $\subseteq$ Partita[NumeroGiornata]

inclusione : Giornata[Numero] $\subseteq$ Posizione[Giornata]

SQUADRA(**<u>Nome</u>**, Città, Allenatore)

inclusione : Squadra[Nome] $\subseteq$ Giocatori[Squadra]

inclusione : Squadra[Nome] $\subseteq$ Posizione[Squadra]

Schema Logico – Parte 2

GIOCATORI(**CF**, Cognome, Nome, Ruolo, CittàNascita, DataNascita, Squadra)
foreign key : Giocatori[Squadra] $\subseteq$ Squadra[Nome]

PARTECIPAZIONE(**Giocatori, Partita, Giornata**, Ruolo)
foreign key : Partecipazione[Giocatori] $\subseteq$ Giocatori[CF]
foreign key : Partecipazione [Partita,Giornata]
$\subseteq$ Partita[NumeroPartita,NumeroGiornata]

POSIZIONE(**Squadra, Giornata**, Punteggio)
foreign key : Posizione[Squadra] $\subseteq$ Squadra[Nome]
foreign key : Posizione[Giornata] $\subseteq$ Giornata [Numero]

PARTITAGIOCATA(**Partita, Giornata**)
foreign key : PartitaGiocata[Partita,Giornata] $\subseteq$ Partita[NumeroPartita, NumeroGiornata]

Schema Logico – Parte 3

PARTITARINVIATA(<u>Partita, Giornata</u>,Data)

foreign key : PartitaRinviata[Partita,Giornata] $\subseteq$ Partita[NumeroPartita, NumeroGiornata]

PARTITAINCAMPONEUTRO(<u>Partita, Giornata</u>, Motivo, Città)

foreign key : PartitaInCampoNeutro[Partita,Giornata] $\subseteq$ PartitaRinviata[NumeroPartita, NumeroGiornata]

<u>Vincoli Esterni</u>

1. In una squadra, il numero di giocatori con la stesso ruolo **DEVE ESSERE inferiore a cinque**

2. La prossima partita di una squadra che ha giocato come ospite **DEVE ESSERE in casa**

<u>Vincoli di Generalizzazione</u>

3. PartitaRinviata[Partita,Giornata] $\cap$ PartitaGiocata[Partita,Giornata] = $\emptyset$

4. Partita[Partita,Giornata] = PartitaRinviata[Partita,Giornata] $\cup$ PartitaGiocata[Partita,Giornata]

Schema Logico – Traduzione in SQL

ARBITRO(**CF**, Cognome, Nome, Città)
inclusione : Arbitro[CF] $\subseteq$ Partita[Arbitro]

```
CREATE TABLE Arbitro(
CF CHAR(10) PRIMARY KEY
Nome CHAR(20) NOT NULL,
Cognome CHAR(20) NOT NULL,
Città CHAR(20) NOT NULL,
check ( CF in SELECT Arbitro
                FROM Partita)
)
```

Schema Logico – Traduzione in SQL

PARTITA(<u>NumeroPartita, NumeroGiornata</u>, Risultato, Arbitro, Casa, Ospite)

foreign key : Partita[NumeroGiornata] ⊆ Giornata[Numero]

foreign key : Partita[Arbitro] ⊆ Arbitro[CF]

foreign key : Partita[Casa] ⊆ Squadra[Nome]

foreign key : Partita[Ospite] ⊆ Squadra[Nome]

```
CREATE TABLE Partita(
NumeroPartita INT,
NumeroGiornata INT,
Risultato CHAR(4) NOT NULL,
Arbitro CHAR(10) NOT NULL,
Casa CHAR(20) NOT NULL,
Ospite CHAR(20) NOT NULL,
PRIMARY KEY (NumeroPartita,NumeroGiornata),
FOREIGN KEY (NumeroGiornata) REFERENCES Giornata(Numero)
FOREIGN KEY (Arbitro) REFERENCES Arbitro(CF)
FOREIGN KEY (Casa) REFERENCES Squadra(Nome)
FOREIGN KEY (Ospite) REFERENCES Squadra(Nome))
```

Schema Logico – Traduzione in SQL

GIORNATA(<u>Numero</u>, Data)

inclusione : Giornata[Numero] $\subseteq$ Partita[NumeroGiornata]

inclusione : Giornata[Numero] $\subseteq$ Posizione[Giornata]

```
CREATE TABLE Giornata(
Numero INT PRIMARY KEY,
Data DATE NOT NULL,
check ( Numero in SELECT NumeroGiornata FROM Partita),
check ( Numero in SELECT Giornata FROM Posizione)
)
```

Schema Logico – Traduzione in SQL

SQUADRA(**Nome**, Città, Allenatore)
inclusione : Squadra[Nome] ⊆ Giocatori[Squadra]
inclusione : Squadra[Nome] ⊆ Posizione[Squadra]

```
CREATE TABLE Squadra(
Nome CHAR(20) PRIMARY KEY,
Città CHAR(20) NOT NULL,
Allenatore CHAR(20) NOT NULL,
check ( Nome in SELECT Squadra FROM Giocatori),
check ( Nome in SELECT Squadra FROM Posizione)
)
```

Schema Logico – Traduzione in SQL

GIOCATORI(**CF**, Cognome, Nome, Ruolo, CittàNascita, DataNascita, Squadra)
foreign key : Giocatori[Squadra] $\subseteq$ Squadra[Nome]

```
CREATE TABLE Giocatori(
CF CHAR(10) PRIMARY KEY,
Nome CHAR(20) NOT NULL,
Cognome CHAR(20) NOT NULL,
Ruolo CHAR(20) NOT NULL,
CittàNascita CHAR(20) NOT NULL,
DataNascita DATE NOT NULL,
Squadra CHAR(20) NOT NULL,
FOREIGN KEY (Squadra) REFERENCES Squadra(Nome))
```

Schema Logico – Traduzione in SQL

PARTECIPAZIONE(<u>Giocatore, Partita, Giornata</u>, Ruolo)
foreign key : Partecipazione[Giocatori] $\subseteq$ Giocatori[CF]
foreign key : Partecipazione [Partita,Giornata] $\subseteq$ Partita[NumeroPartita,NumeroGiornata]

```
CREATE TABLE Partecipazione(
Giocatore CHAR(20),
Partita INT,
Giornata INT,
Ruolo CHAR(20) NOT NULL,
PRIMARY KEY (Giocatore,Partita,Giornata),
FOREIGN KEY (Giocatori) REFERENCES Giocatori(CF),
FOREIGN KEY (Partita,Giornata) REFERENCES
                    Partita(NumeroPartita,NumeroGiornata))
```

Schema Logico – Traduzione in SQL

POSIZIONE(<u>Squadra, Giornata</u>, Punteggio)
foreign key : Posizione[Squadra] ⊆ Squadra[Nome]
foreign key : Posizione[Giornata] ⊆ Giornata [Numero]

```
CREATE TABLE Posizione(
Squadra CHAR(20),
Giornata INT,
Punteggio INT NOT NULL,
PRIMARY KEY (Squadra,Giornata),
FOREIGN KEY (Squadra) REFERENCES Squadra(Nome),
FOREIGN KEY (Giornata) REFERENCES
                        Giornata(Numero))
```

Schema Logico – Traduzione in SQL

PARTITAGIOCATA(<u>Partita, Giornata</u>)

foreign key : PartitaGiocata[Partita,Giornata] $\subseteq$ Partita[NumeroPartita,NumeroGiornata]

```
CREATE TABLE PartitaGiocata(
Partita INT,
Giornata INT,
PRIMARY KEY (Partita,Giornata),
FOREIGN KEY (Partita,Giornata) REFERENCES
                    Partita(NumeroPartita,NumeroGiornata))
```

Schema Logico – Traduzione in SQL

PARTITARINVIATA(<u>Partita, Giornata</u>,Data)

foreign key : PartitaRinviata[Partita,Giornata] $\subseteq$ Partita[NumeroPartita, NumeroGiornata]

```
CREATE TABLE PartitaRinviata(
Partita INT,
Giornata INT,
Data DATE NOT NULL,
PRIMARY KEY (Partita,Giornata),
FOREIGN KEY (Partita,Giornata) REFERENCES
                    Partita(NumeroPartita,NumeroGiornata))
```

Schema Logico – Traduzione in SQL

PARTITAINCAMPONEUTRO(<u>Partita, Giornata</u>, Motivo, Città)
foreign key : PartitaInCampoNeutro[Partita,Giornata] ⊆
PartitaRinviata[Partita, Giornata]

```
CREATE TABLE PartitaInCampoNeutro(
Partita INT,
Giornata INT,
Motivo CHAR(20) NOT NULL,
Città CHAR(20) NOT NULL,
PRIMARY KEY (Partita,Giornata),
FOREIGN KEY (Partita,Giornata) REFERENCES
                    PartitaRinviata(Partita, Giornata))
```

Schema Logico – Traduzione in SQL

PartitaRinviata[Partita,Giornata] ∩ PartitaGiocata[Partita,Giornata] = Ø

```
CREATE ASSERTION
PARTITARINVIATA_DISJOINT_PARTITAGIOCATA(
check ( NOT EXISTS (
        SELECT Partita,Giornata
        FROM PartitaRinviata
        INTERSECT
        SELECT Partita,Giornata
        FROM PartitaGiocata )
)
```

Schema Logico – Traduzione in SQL

Partita[Partita,Giornata] = PartitaRinviata[Partita,Giornata] ∪
PartitaGiocata[Partita,Giornata]

```
CREATE ASSERTION PG_PR_COMPLETE_Partita (
check (NOT EXISTS (
        SELECT * FROM Partita P WHERE
                0 = (SELECT count(*) FROM PartitaRinviata PR
WHERE PR.Partita = P.NumeroPartita AND PR.Giornata =
P.Giornata) AND
                0 = (SELECT count(*) FROM PartitaGiocata PG
WHERE PG.Partita = P.NumeroPartita AND PG.Giornata =
P.Giornata);
```

Schema Logico – Traduzione in SQL

In una squadra, il numero di giocatori con la stesso ruolo **DEVE ESSERE inferiore a cinque**

```
CREATE ASSERTION CINQUE_GIOCATORI (
check (
    5 >= all SELECT count(Ruolo)
             FROM Giocatori
             GROUP BY Squadra,Ruolo
        )
)
```

Schema Logico – Traduzione in SQL

La prossima partita di una squadra che ha giocato come ospite **DEVE ESSERE in casa**

```
CREATE ASSERTION OSPITE_CASA (
check ( NOT EXISTS (
        SELECT NumeroPartita,NumeroGiornata
        FROM Partita P
        WHERE Ospite in (SELECT P1.Ospite
                         FROM Partita P1
                         WHERE P1.Giornata = P.Giornata + 1))
))
```