# La progettazione logica

Matteo Gorgone
Università degli Studi di Messina

# La progettazione logica

La progettazione logica si articola in due fasi:

- **RISTRUTTURAZIONE DELLO SCHEMA ENTITÀ-ASSOCIAZIONE**: è una fase indipendente dal modello logico scelto e si concentra sui criteri di ottimizzazione dello schema e di semplificazione per la fase successiva;
- **TRADUZIONE NEL MODELLO LOGICO**: si tiene conto di uno specifico modello logico (nel nostro caso quello relazionale) all'interno del quale viene tradotta la realtà rappresentata dallo schema ER.

# Ristrutturazione dello schema ER

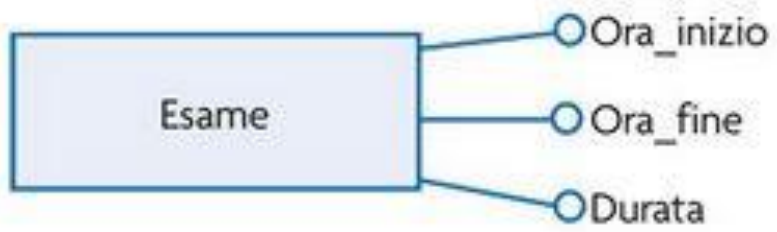
Passi da svolgere nella fase di ristrutturazione dello schema Entità-Associazioni:

1. **Analisi delle ridondanze**;
2. **Eliminazione delle generalizzazioni**;
3. **Partizionamento/accorpamento di entità e associazioni**;
4. **Scelta delle chiavi primarie**.

# Ristrutturazione dello schema ER: *analisi delle ridondanze*

Una ridondanza è un'informazione significativa ma derivabile da altre.

- Attributi derivabili all'interno della stessa entità:

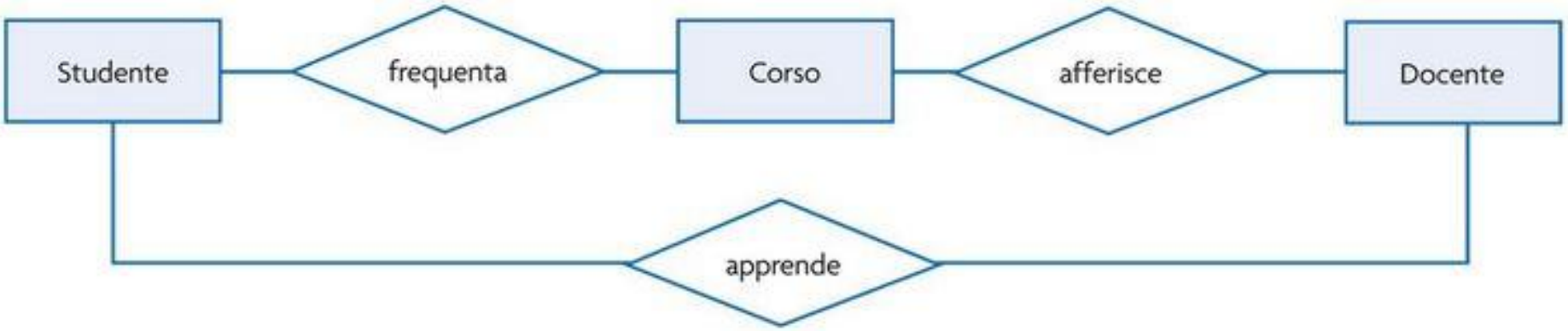



- Attributi derivabili da altre entità:

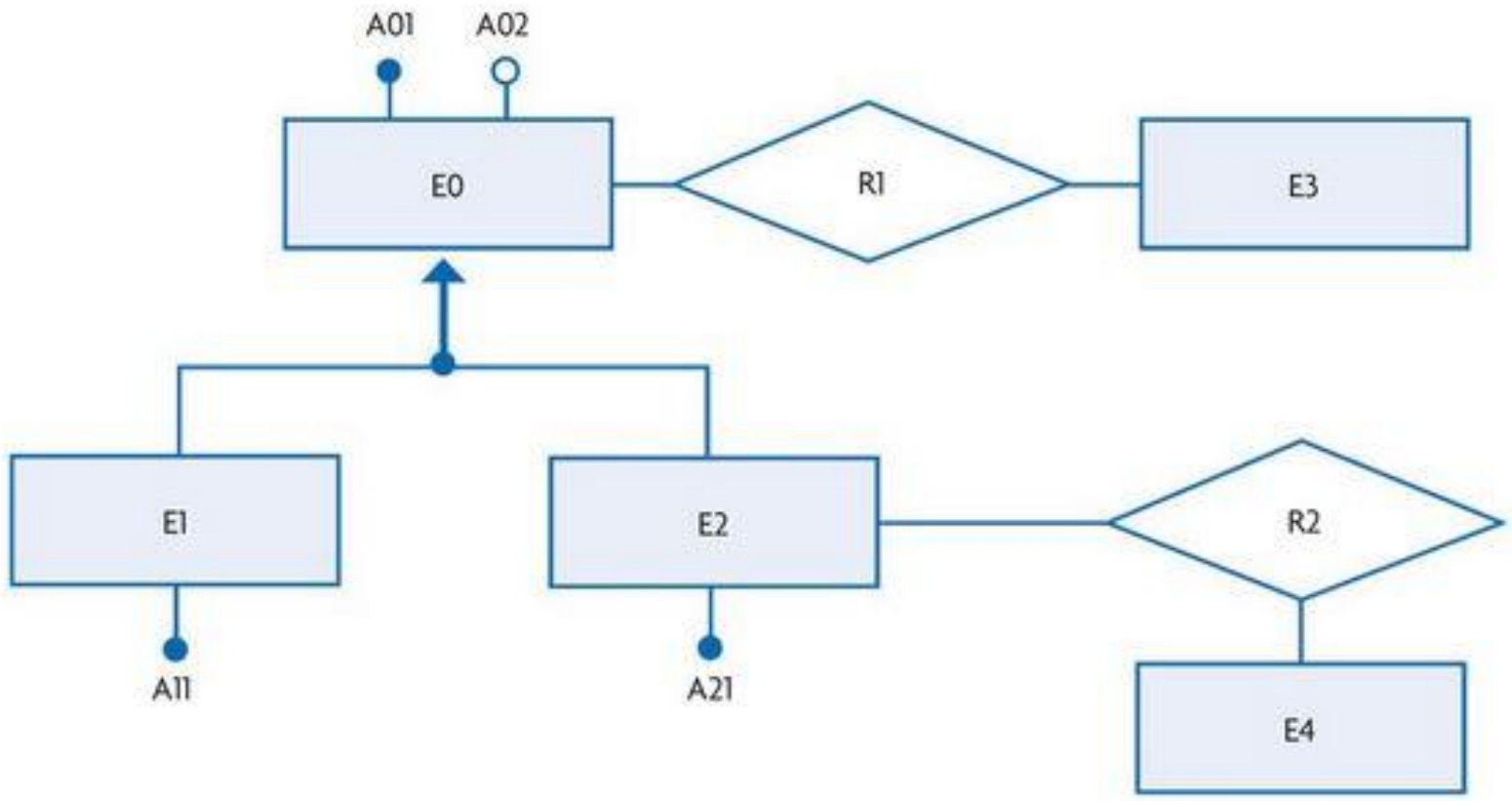
# Ristrutturazione dello schema ER: *analisi delle ridondanze*

- Attributi derivabili da operazioni di conteggio di occorrenze;

- Ridondanza di associazioni in presenza di cicli:

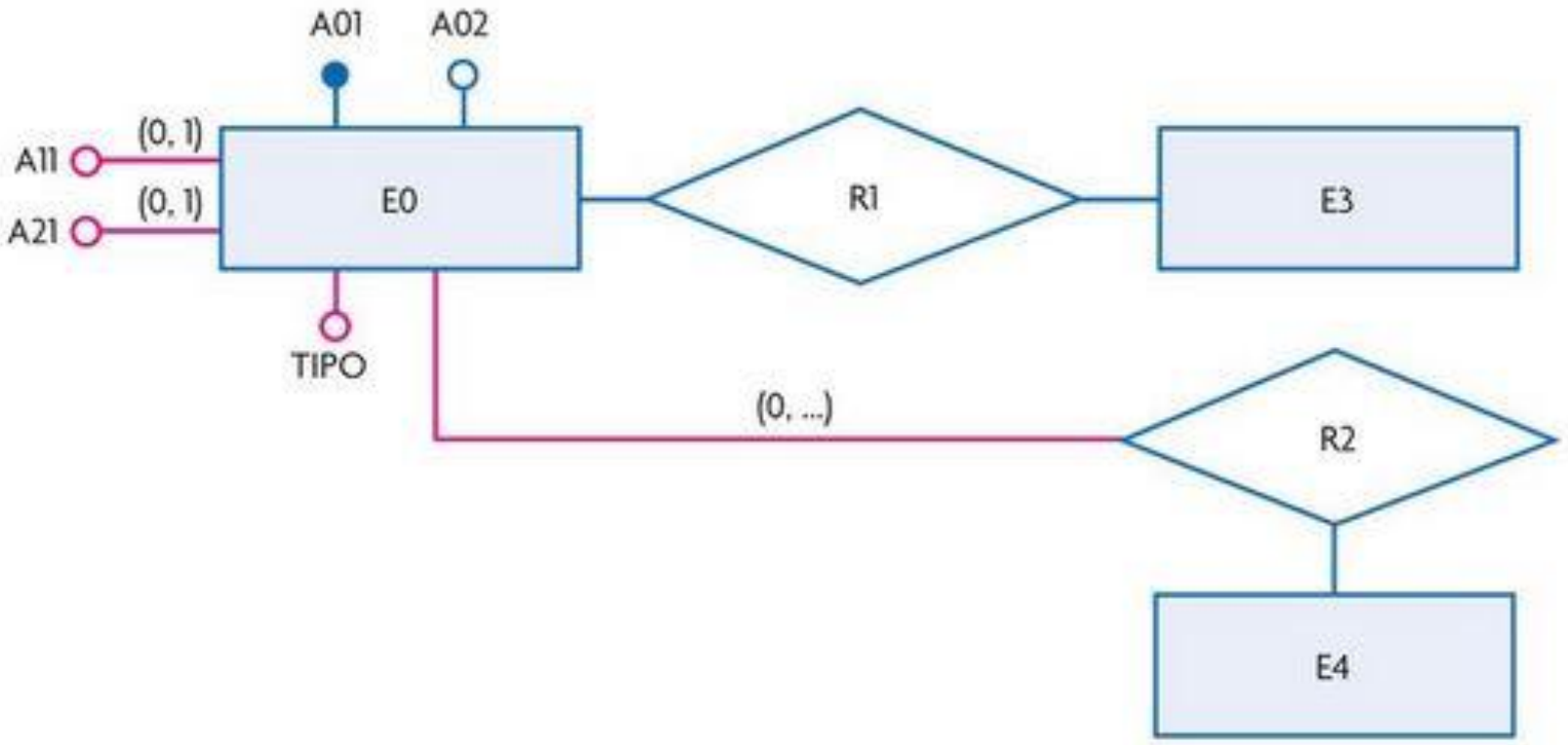


# Ristrutturazione dello schema ER: *eliminazione delle generalizzazioni*



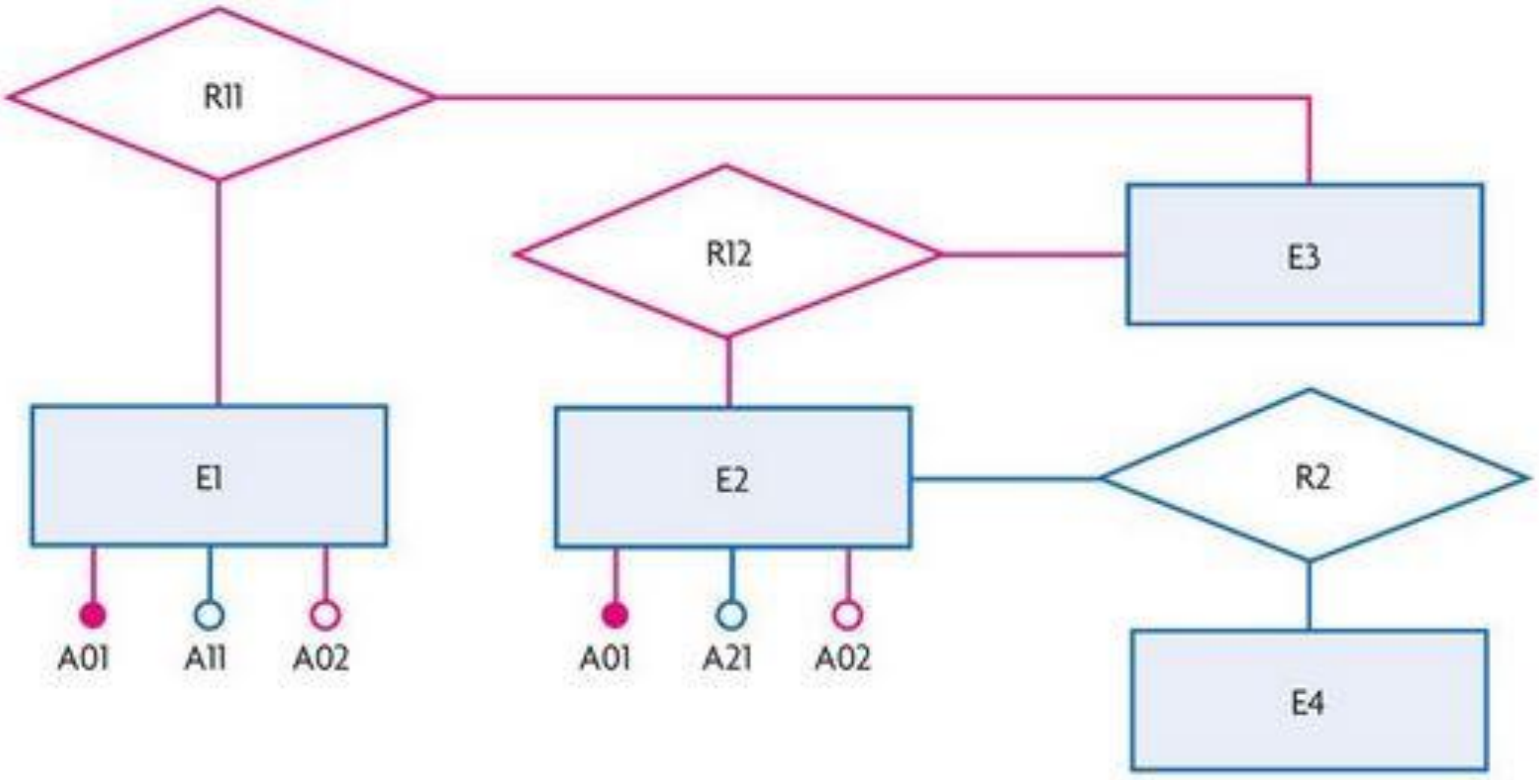
# Ristrutturazione dello schema ER: *eliminazione delle generalizzazioni*

- Accorpamento delle figlie della generalizzazione nel padre:



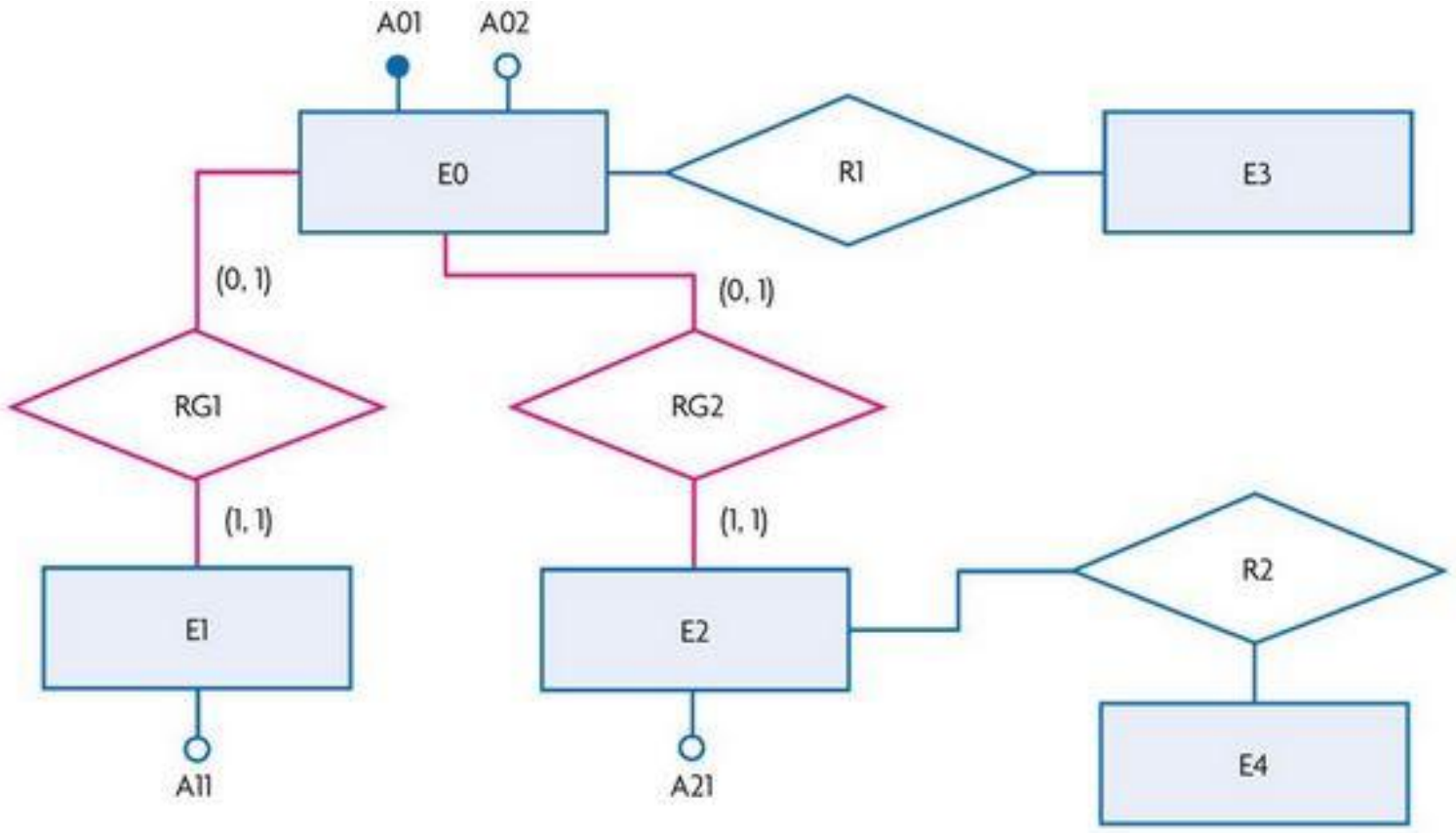
# Ristrutturazione dello schema ER: *eliminazione delle generalizzazioni*

- Accorpamento del padre della generalizzazione nelle figlie:



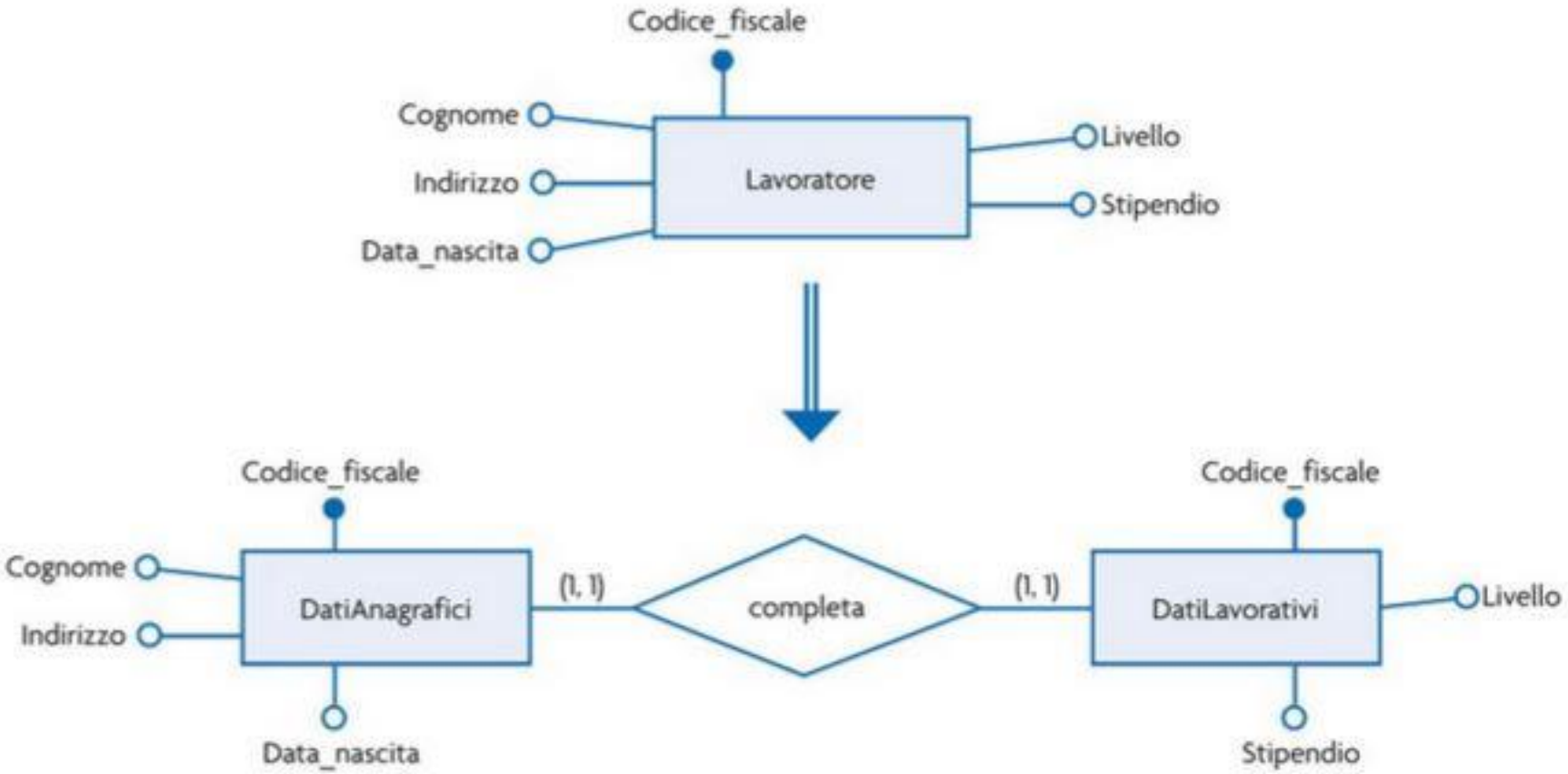
# Ristrutturazione dello schema ER: *eliminazione delle generalizzazioni*

- Sostituzione della generalizzazione con associazioni:



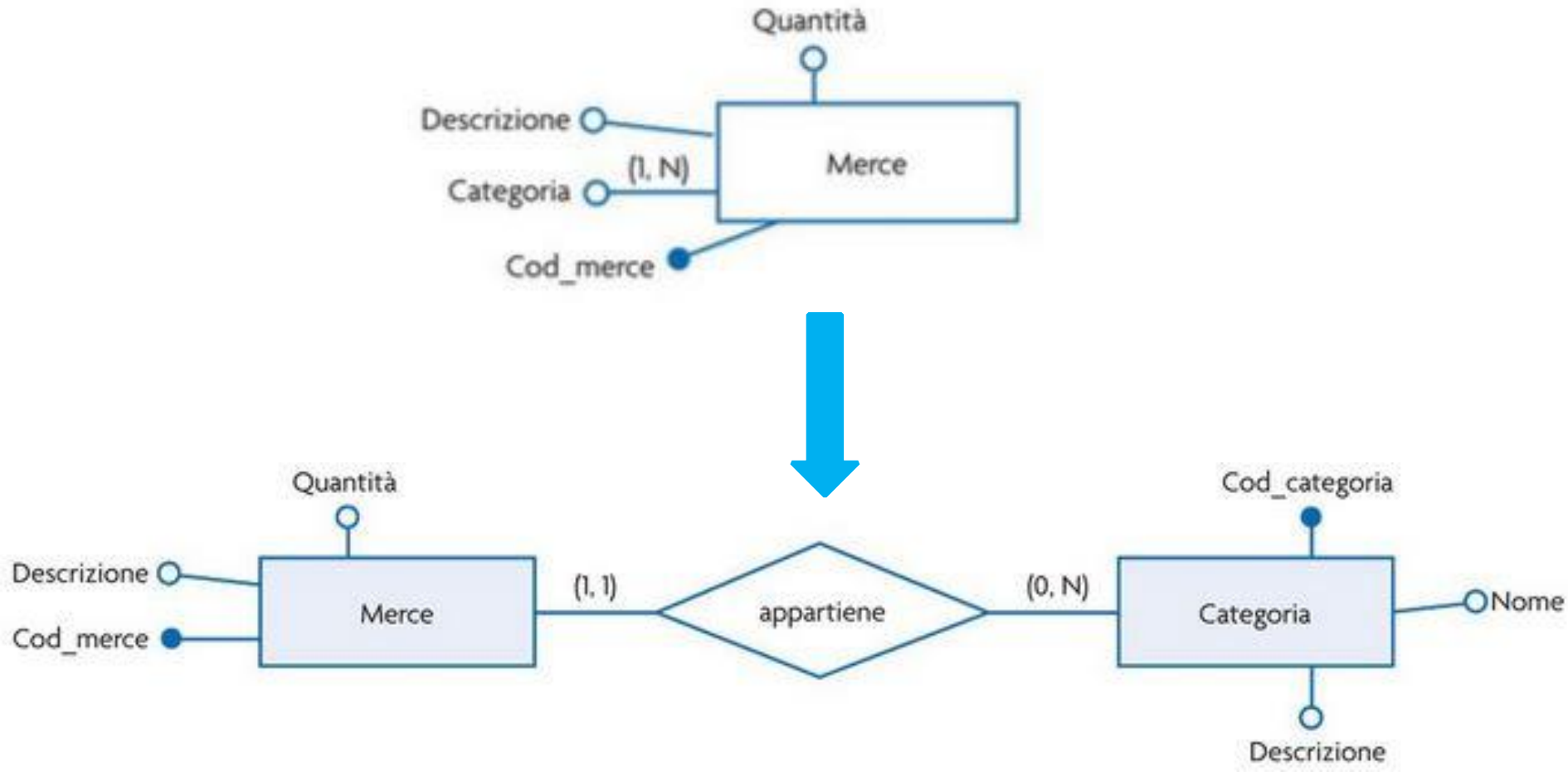
# Ristrutturazione dello schema ER: *partizionamento/accorpamento di entità e associazioni*

- Partizionamento di entità:



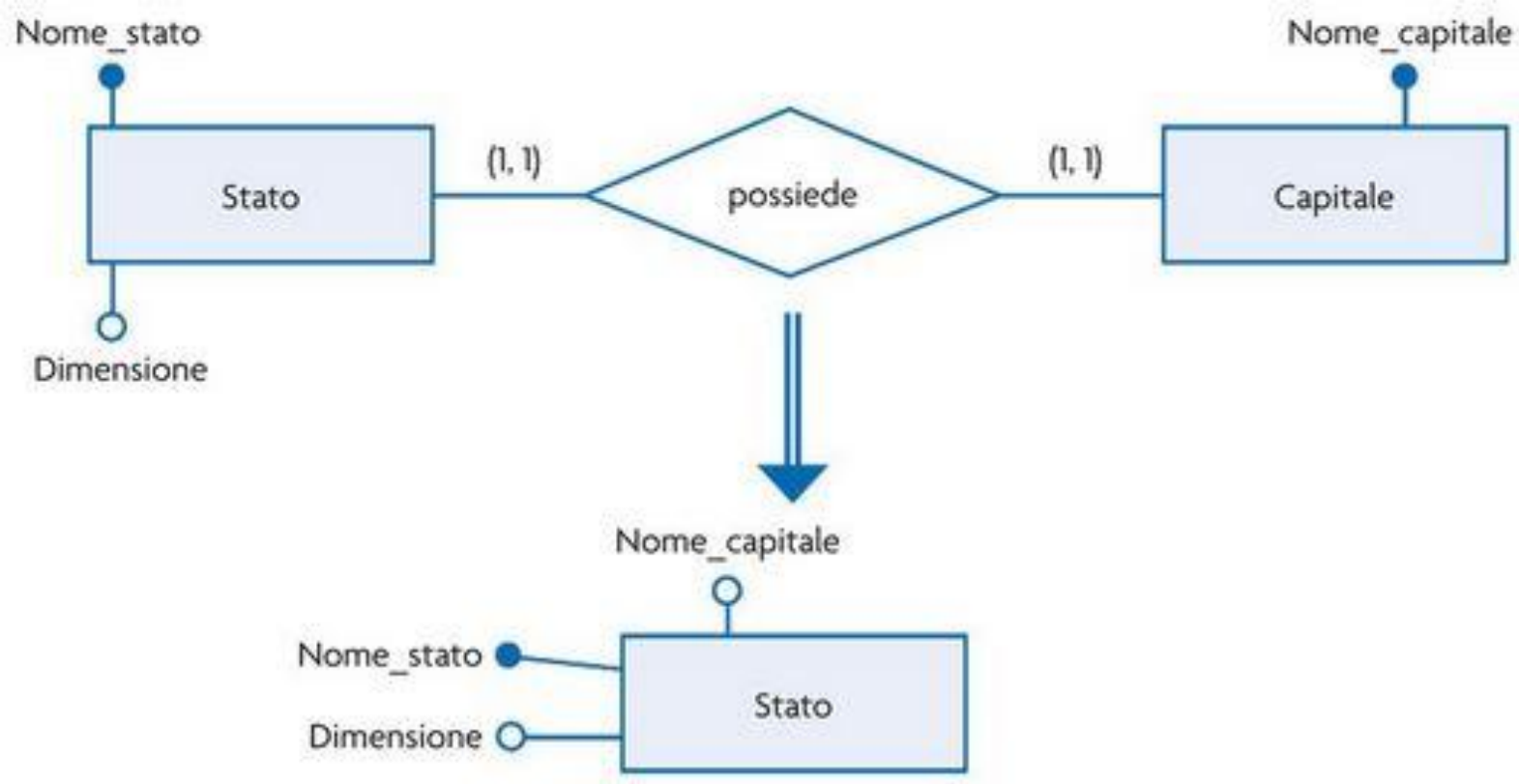
# Ristrutturazione dello schema ER: *partizionamento/accorpamento di entità e associazioni*

- Eliminazione di attributi multivalore:



# Ristrutturazione dello schema ER: *partizionamento/accorpamento di entità e associazioni*

- Accorpamento di entità:



# Ristrutturazione dello schema ER: *scelta delle chiavi primarie*

Criteri:

- Non si possono utilizzare attributi con valori NULLI;
- È preferibile usare identificatori composti da uno o pochi attributi favorendo così la realizzazione dei legami tra le varie relazioni.

OSS.

Se nessuno degli attributi soddisfa i precedenti requisiti si introduce un attributo *(chiave artificiale)* che contiene dei codici generati volutamente per identificare le occorrenze delle entità.

# Progettazione logica

*La progettazione logica relazionale* consiste nel “mapping”, cioè nella conversione del diagramma ER in un insieme di tabelle, detto *schema logico relazionale*, e nella definizione delle operazioni da compiere su di esso.

Una relazione R su una sequenza di insiemi $D_1$, $D_2$,..., $D_n$ (non necessariamente distinti), è un sottoinsieme finito del prodotto cartesiano $D_1 \times D_2 \times ... \times D_n$, che possiamo esprimere con:

$$R \subseteq D_1 \times D_2 \times ... \times D_n,$$

dove:

- Gli insieme D sono detti *Domini* della relazione;
- n è il *grado* della relazione e si indica con Grado(R).

Lo schema di una relazione è il suo nome e la lista dei suoi attributi, racchiusi tra parentesi tonde e separati da virgole:

NOMERELAZIONE(attributo1,…, attributoN)

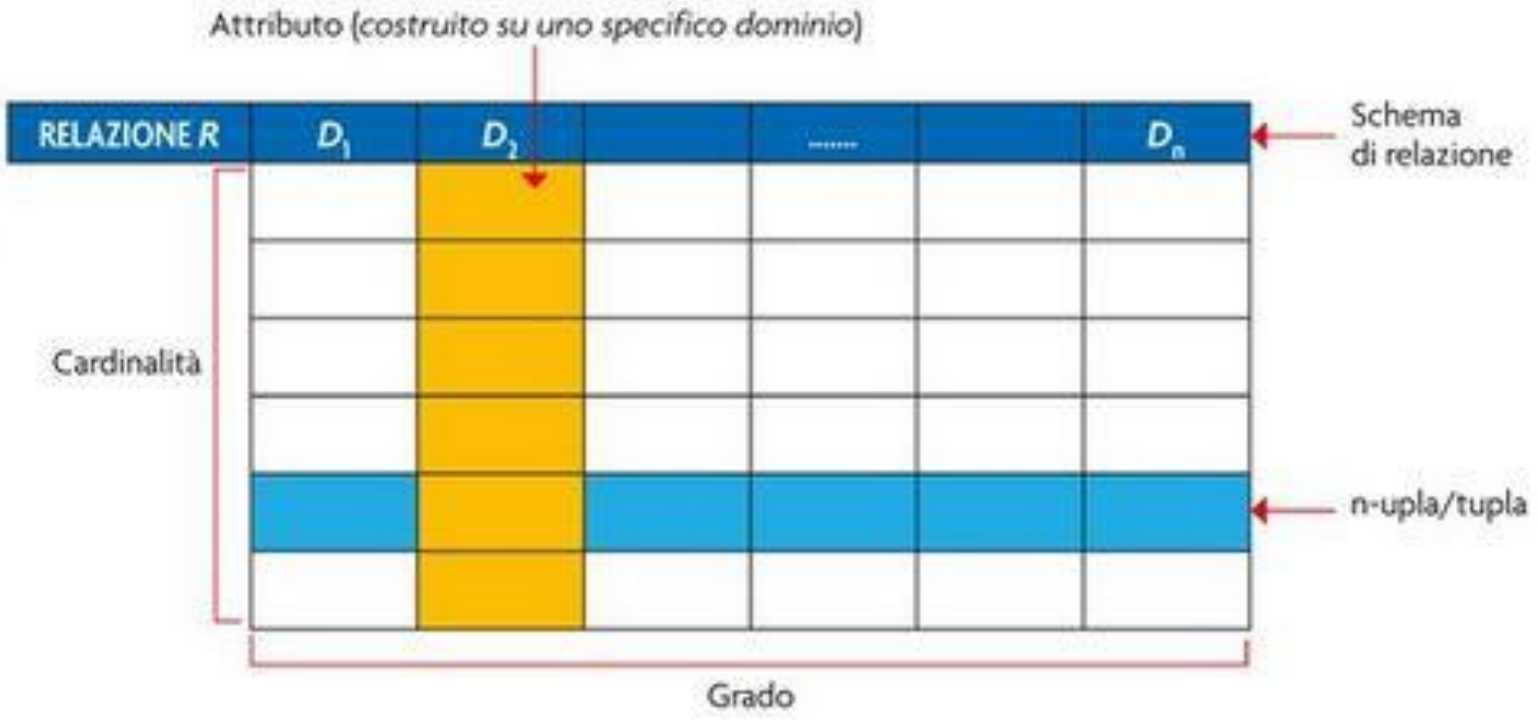
# Progettazione logica

Gli elementi di una relazione sono detti *n-uple* o *tuple* e si indicano con

$$(d_1, d_2, \ldots, d_n) \text{ con } d_i \in D_i$$

L'istanza di una relazione è l'insieme delle sue n-uple in un determinato istante di tempo.

Il numero m di n-uple presenti in un dato istante in una relazione viene chiamato *cardinalità* e si indica con Card(R).



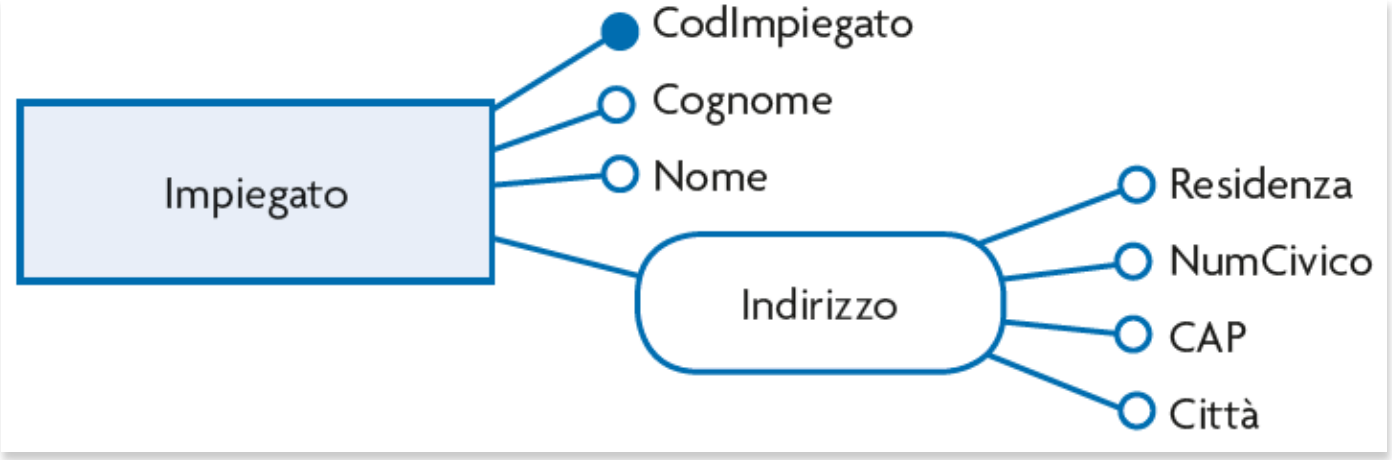
# Mapping di entità e attributi

La rappresentazione di ogni entità, con i suoi attributi, di uno schema ER viene resa graficamente come segue:


- Ogni entità diventa una relazione, rappresentabile mediante una tabella;
- Ogni attributo dell'entità diventa un attributo della relazione, rappresentato con una colonna della tabella;
- L'attributo chiave dell'entità diventa attributo chiave della relazione e viene indicato sottolineandolo;
- Eventuali attributi composti devono essere sostituiti con gli attributi componenti;
- Gli attributi opzionali si indicano facendoli seguire da un asterisco.

# Mapping di entità e attributi

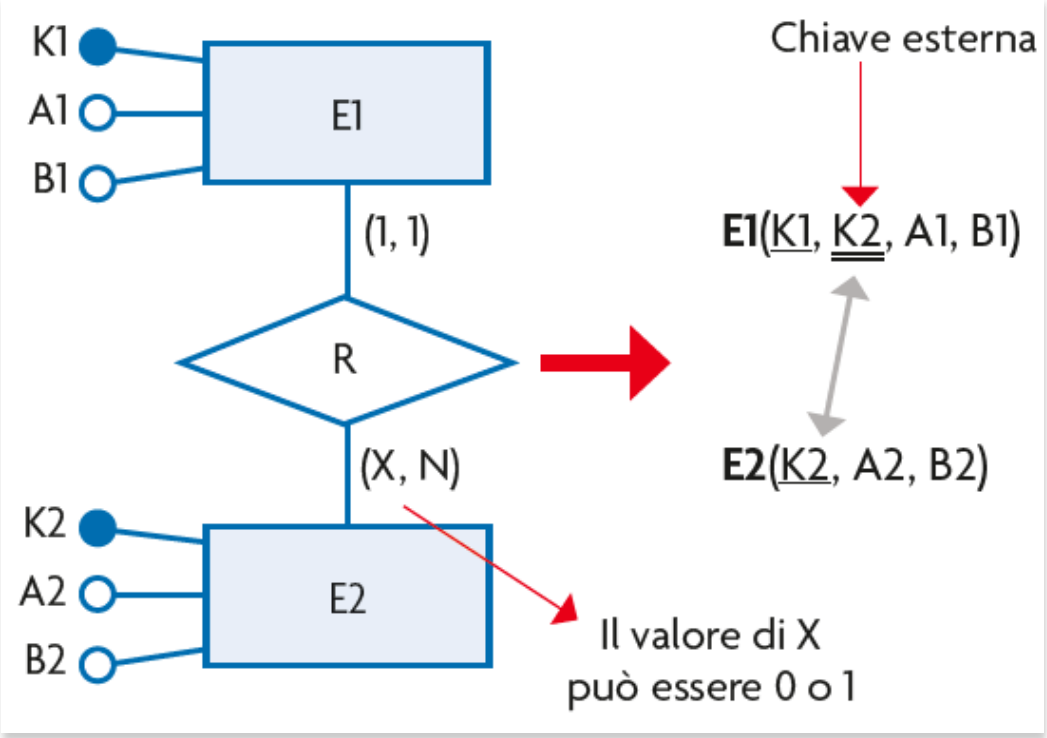
**Esempio:**




IMPIEGATO(CodImpiegato, Cognome, Nome, Residenza, NumCivico, CAP, Città)

# Mapping delle associazioni binarie 1:N

Un'associazione R di tipo **1:N** tra due entità $E_1$ ed $E_2$ è "mappata" aggiungendo alla relazione $E_1$ gli attributi chiave primaria di $E_2$.




Gli attributi chiave primaria di $E_2$ presenti in $E_1$ costituiscono una **chiave esterna** per la relazione $E_1$, che viene rappresentata con una **doppia sottolineatura**.

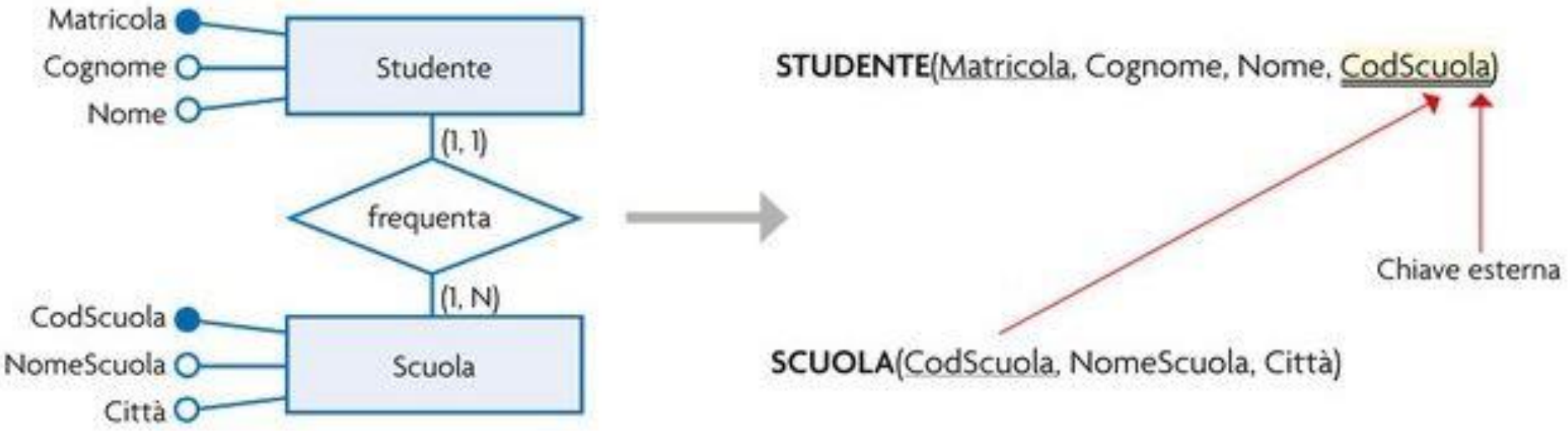
# Mapping delle associazioni binarie 1:N

**Esempio 1:**


**PERSONA**(CodFiscale, Cognome, Nome)

**AUTO**(Targa, Modello, Colore, CodFiscale)

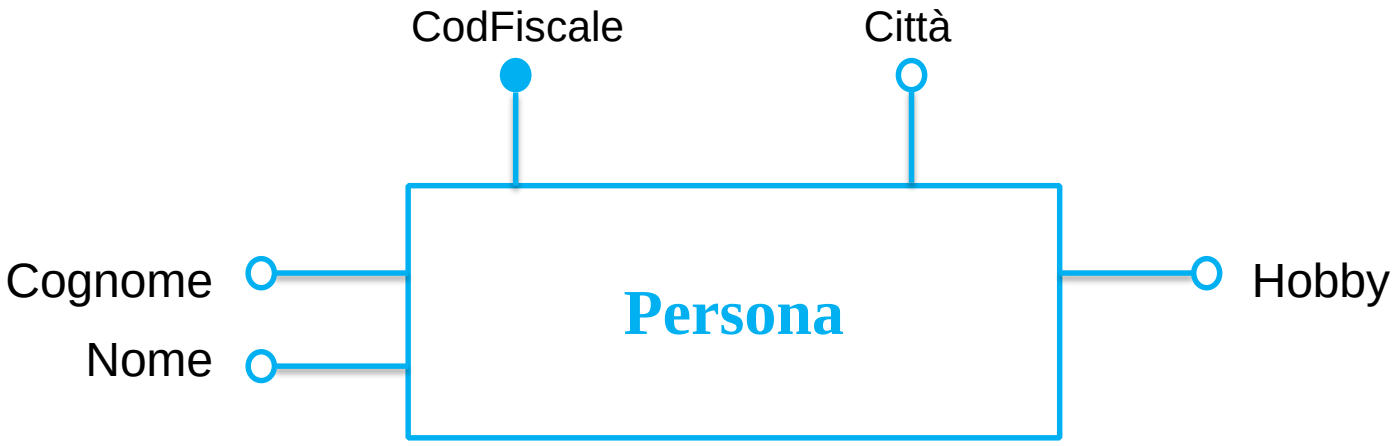
**Esempio 2:**



# Mapping delle associazioni binarie 1:N

**Esempio 3:**

Le associazioni 1:N vengono utilizzate anche per mappare gli attributi multipli.




**PERSONA**(CodFiscale, Cognome, Nome, Città)

**HOBBY**(CodFiscale, NomeHobby)

# Mapping delle associazioni binarie 1:1

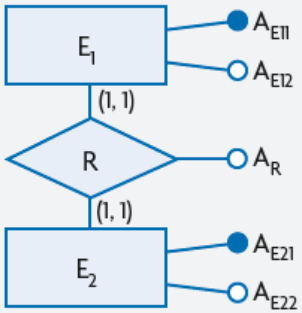
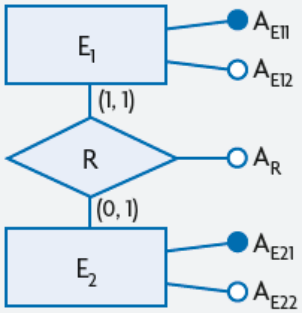
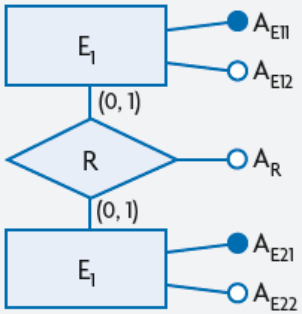
Varie possibilità di mapping:

- Associazione diretta e inversa con partecipazione totale ⟶
  Unica relazione che possiede tutti gli attributi delle due entità; è anche consentita la mappatura 1:N;
- Associazione diretta o inversa con partecipazione facoltativa ⟶
  due relazioni distinte in cui si aggiunge la chiave esterna alla relazione rispetto a cui l'associazione è totale;
- Associazione diretta e inversa con partecipazione facoltativa ⟶
  si prevedono solo relazioni separate e mai la relazione unica.

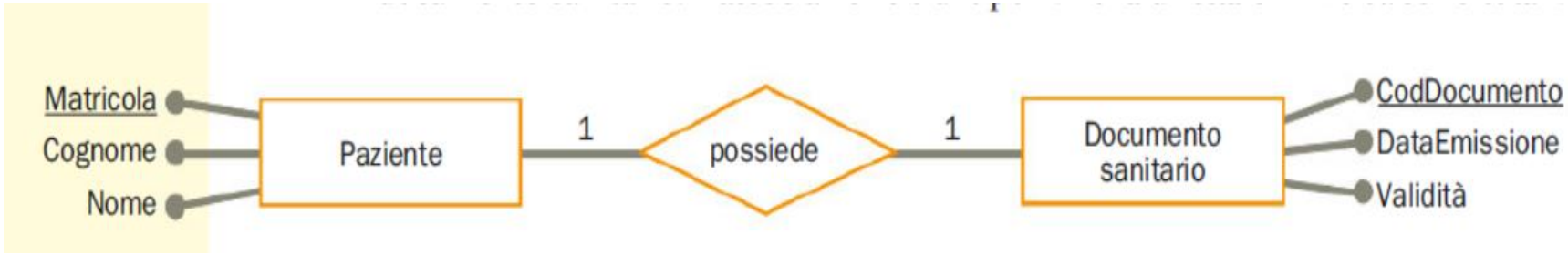
**OSS.**

Gli eventuali attributi dell'associazione vengono inseriti nella relazione a cui si aggiunge la chiave esterna.

# Mapping delle associazioni binarie 1:1

| Tipo di associazione | Schema concettuale | Schema logico |
|---|---|---|
| Associazione uno a uno con partecipazione obbligatoria per entrambe le entità |   | $E_1(\underline{A_{E11}}, A_{E12}, \underline{\underline{A_{E21}}}, A_R)$<br>$E_2(\underline{A_{E21}}, A_{E22})$<br>oppure<br>$E_1(\underline{A_{E11}}, A_{E12})$<br>$E_2(\underline{A_{E21}}, A_{E22}, \underline{\underline{A_{E11}}}, A_R)$ |
| Associazione uno a uno con partecipazione opzionale per una entità |   | $E_1(\underline{A_{E11}}, A_{E12}, \underline{\underline{A_{E21}}}, A_R)$<br>$E_2(\underline{A_{E21}}, A_{E22})$ |
| Associazione uno a uno con partecipazione opzionale per entrambe le entità |   | $E_1(\underline{A_{E11}}, A_{E12})$<br>$E_2(\underline{A_{E21}}, A_{E22}, \underline{\underline{A_{E11}}}, A_R)$<br>oppure<br>$E_1(\underline{A_{E11}}, A_{E12}, \underline{\underline{A_{E21}}}, A_R)$<br>$E_2(\underline{A_{E21}}, A_{E22})$<br>oppure<br>$E_1(\underline{A_{E11}}, A_{E12})$<br>$E_1(\underline{A_{E21}}, A_{E22})$<br>$R(\underline{A_{E11}}, A_{E21}, A_R)$<br>All'interno della relazione R è anche possibile scegliere $A_{E21}$ come chiave primaria piuttosto che $A_{E11}$. |

# Mapping delle associazioni binarie 1:1




Relazione derivata:

**PAZIENTI_ASL**(Matricola, Nome, Cognome, CodDocumento, DataEmissione, Validità)

oppure

**PAZIENTI**(Matricola, Nome, Cognome, CodDocumento)

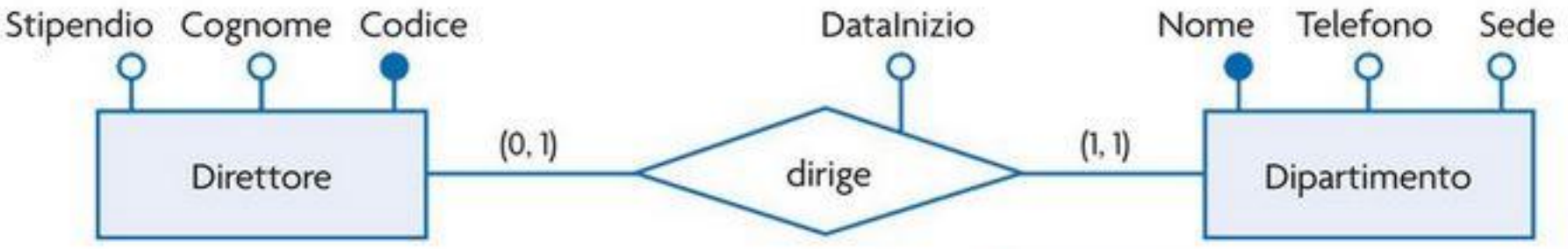
**DOCUMENTO_SANITARIO**(CodDocumento, DataEmissione, Validità)

oppure

**PAZIENTI**(Matricola, Nome, Cognome)

**DOCUMENTO_SANITARIO**(CodDocumento, DataEmissione, Validità, Matricola)

# Mapping delle associazioni binarie 1:1

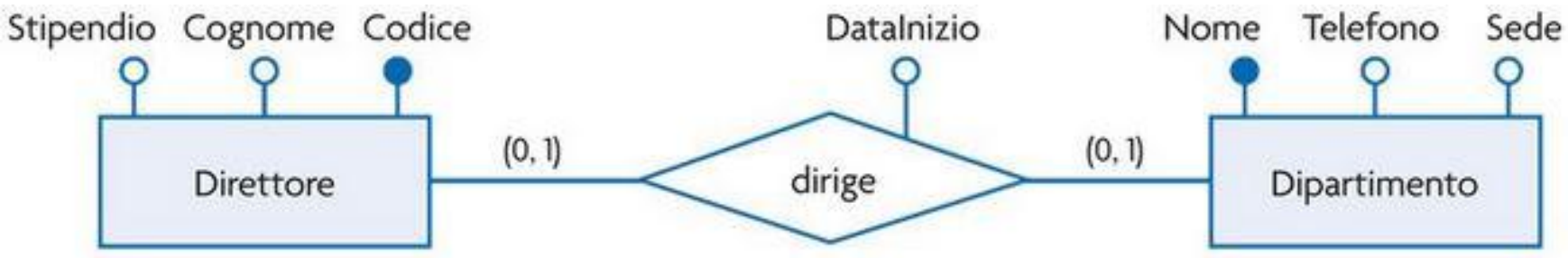



Relazione derivata:

**DIRETTORE**(Codice, Stipendio, Cognome)

**DIPARTIMENTO**(Nome, Telefono, Sede, Codice, DataInizio)

# Mapping delle associazioni binarie 1:1




Relazione derivata:

**DIRETTORE**(Codice, Stipendio, Cognome)

**DIPARTIMENTO**(Nome, Telefono, Sede, Codice, DataInizio)

oppure

**DIRETTORE**(Codice, Stipendio, Cognome, Nome, DataInizio)

**DIPARTIMENTO**(Nome, Telefono, Sede)

oppure

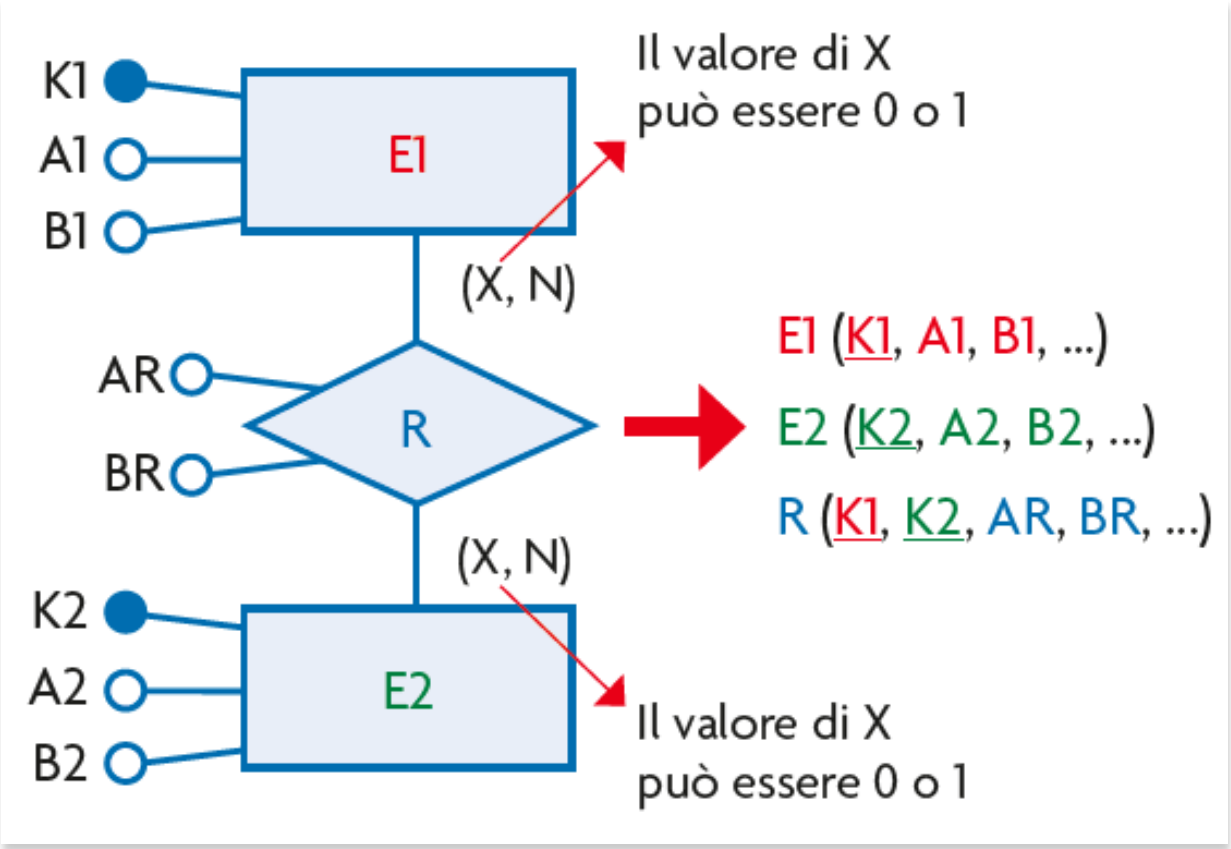
**DIRETTORE**(Codice, Stipendio, Cognome)

**DIPARTIMENTO**(Nome, Telefono, Sede)

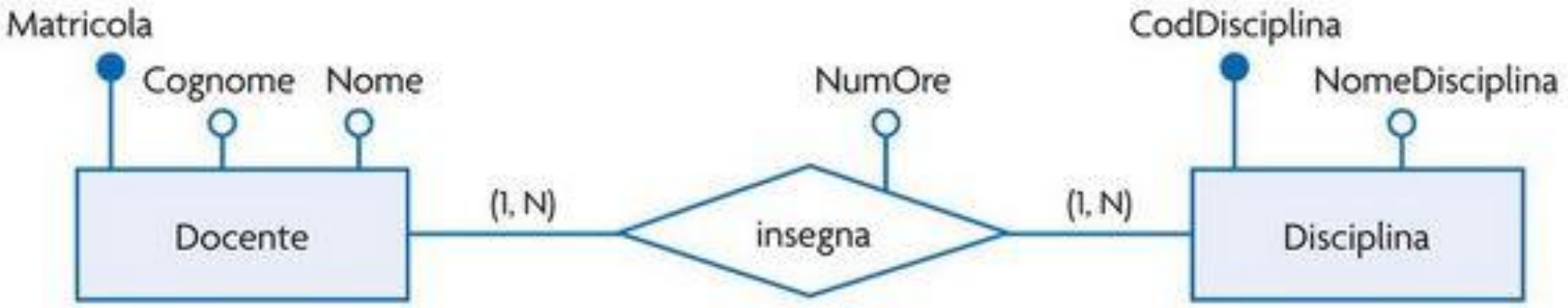
**DIRIGE**(Nome, Codice, DataInizio)

# Mapping delle associazioni binarie N:N

Un'associazione R di tipo **N:N** tra due entità $E_1$ ed $E_2$ è mappata creando per ogni entità una relazione R avente almeno gli attributi chiave primaria di $E_1$ e gli attributi chiave primaria di $E_2$ che insieme formano la chiave primaria della relazione R.



# Mapping delle associazioni binarie N:N




Relazione derivata:

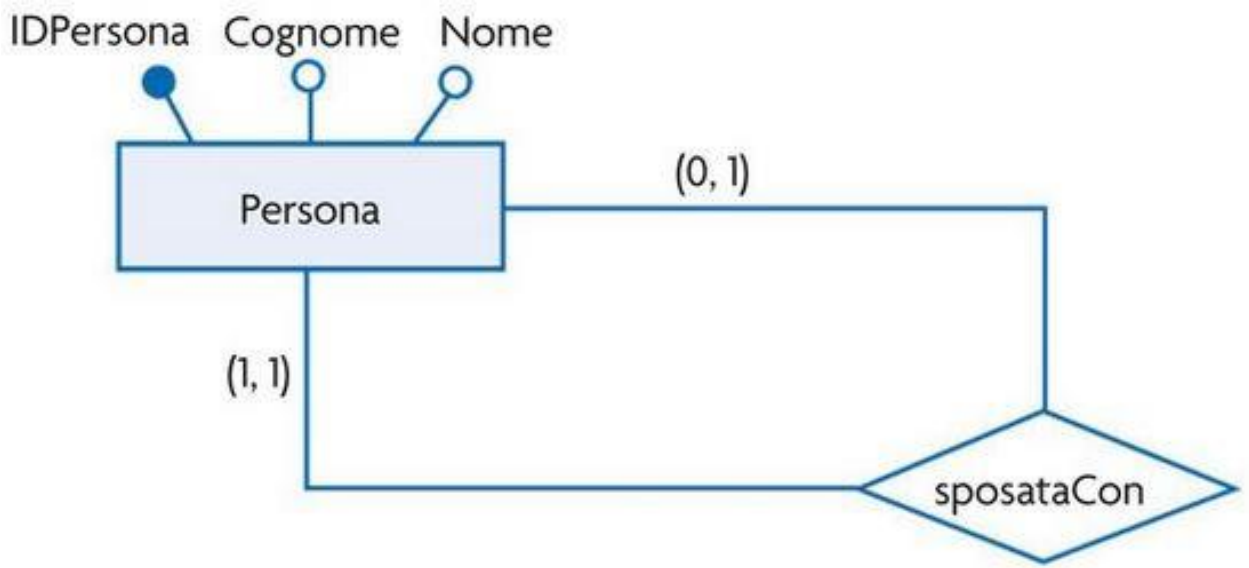
**DOCENTE**(Matricola, Cognome, Nome)

**DISCIPLINA**(CodDisciplina, NomeDisciplina)

**INSEGNA**(Matricola, CodDisciplina, NumOre)

# Mapping delle associazioni ricorsive

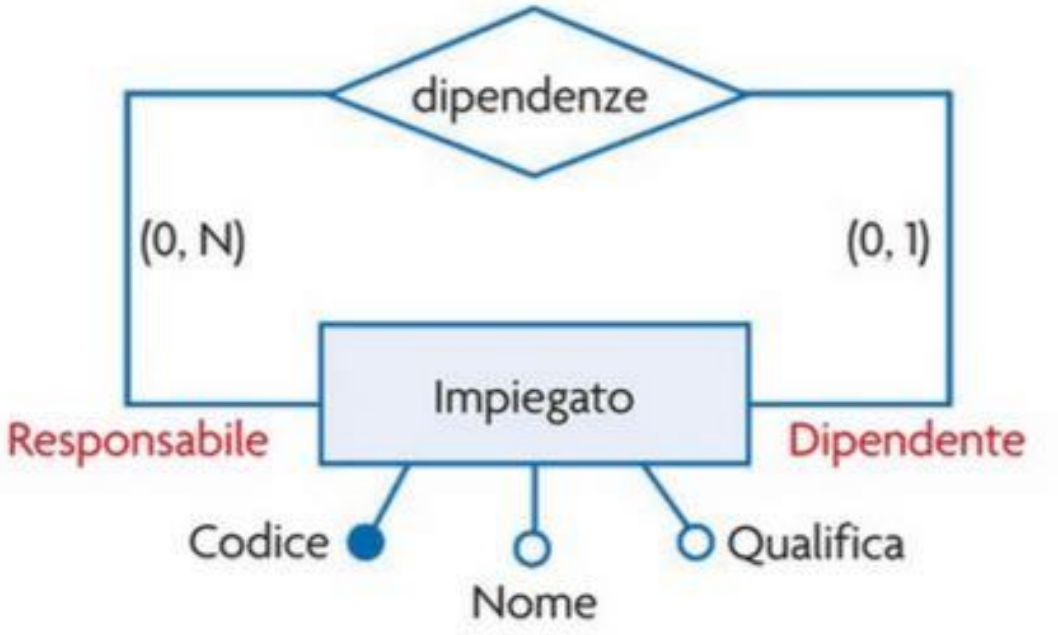
- **Mapping delle associazioni ricorsive di tipo 1:1:**




**PERSONA**(IDPersona, Cognome, Nome, IDPersonaSposata)

# Mapping delle associazioni ricorsive

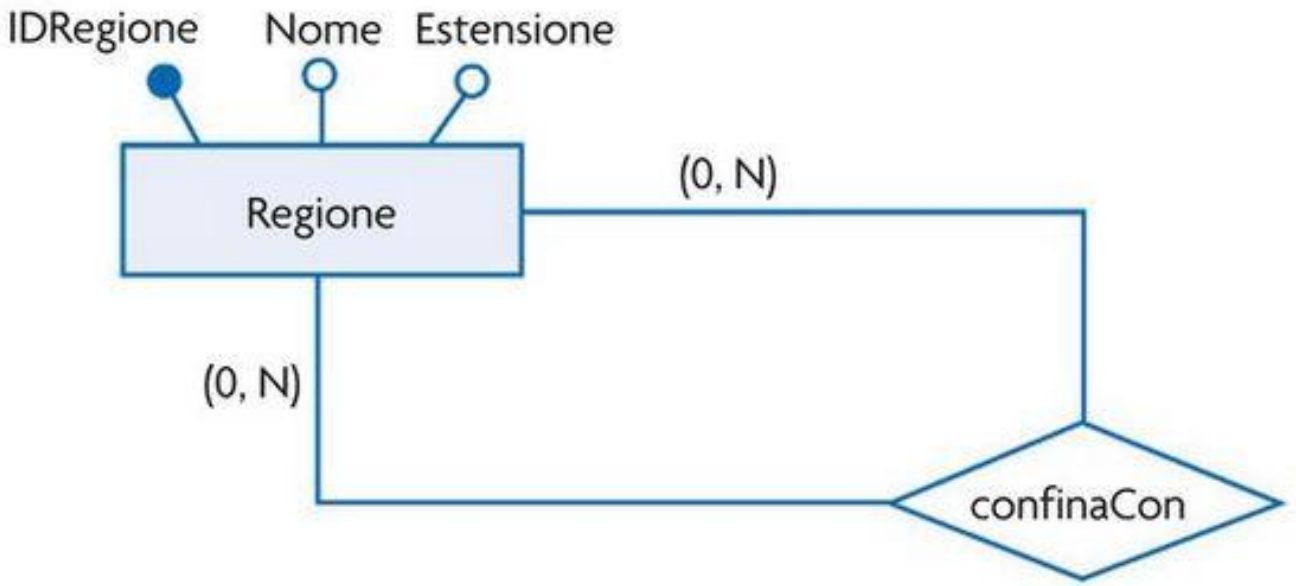
- **Mapping delle associazioni ricorsive di tipo 1:N:**




**IMPIEGATO**(Codice, Nome, Qualifica, CodiceResponsabile)

# Mapping delle associazioni ricorsive

- **Mapping delle associazioni ricorsive di tipo N:N:**



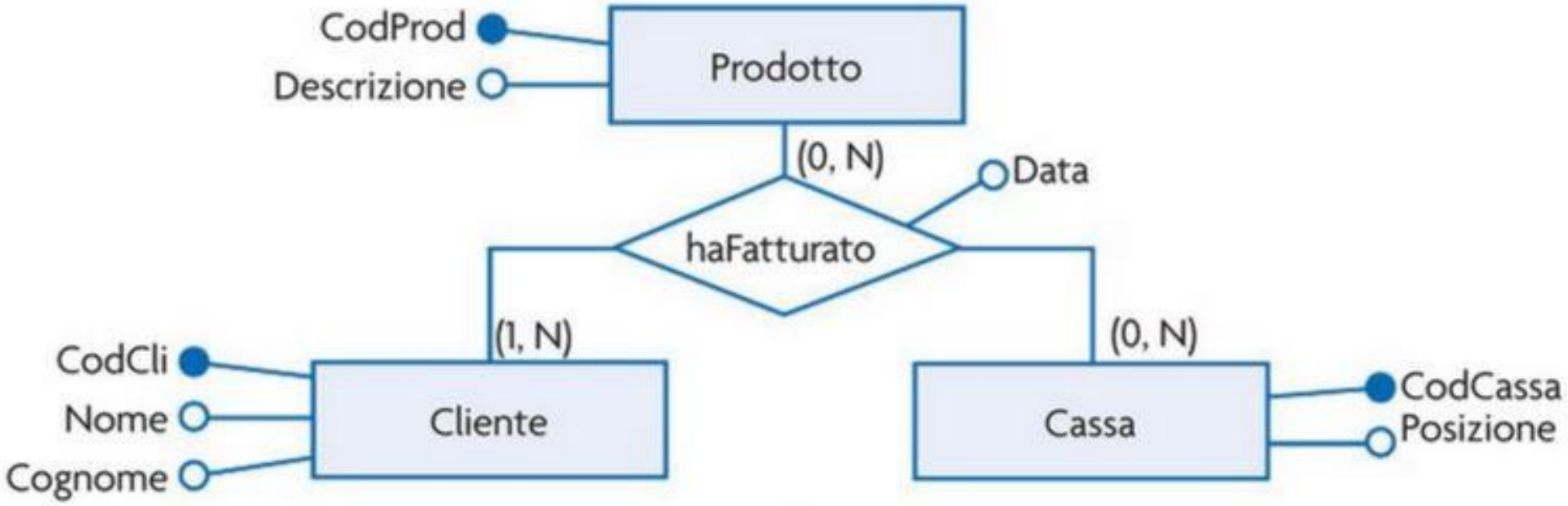

**REGIONE**(IDRegione, Nome, Estensione)

**CONFINACON**(IDRegione1, IDRegione2)

# Mapping delle associazioni n-arie

Si applicano le stesse regole viste per le associazioni binarie.

Esempio:




**PRODOTTO**(<u>CodProd</u>, Descrizione)

**CLIENTE**(<u>CodCli</u>, Nome, Cognome)

**CASSA**(<u>CodCassa</u>, Posizione)

**HAFATTURATO**(<u>CodProd</u>, <u>CodCli</u>, <u>CodCassa</u>, Data)

# Vincoli d'integrità

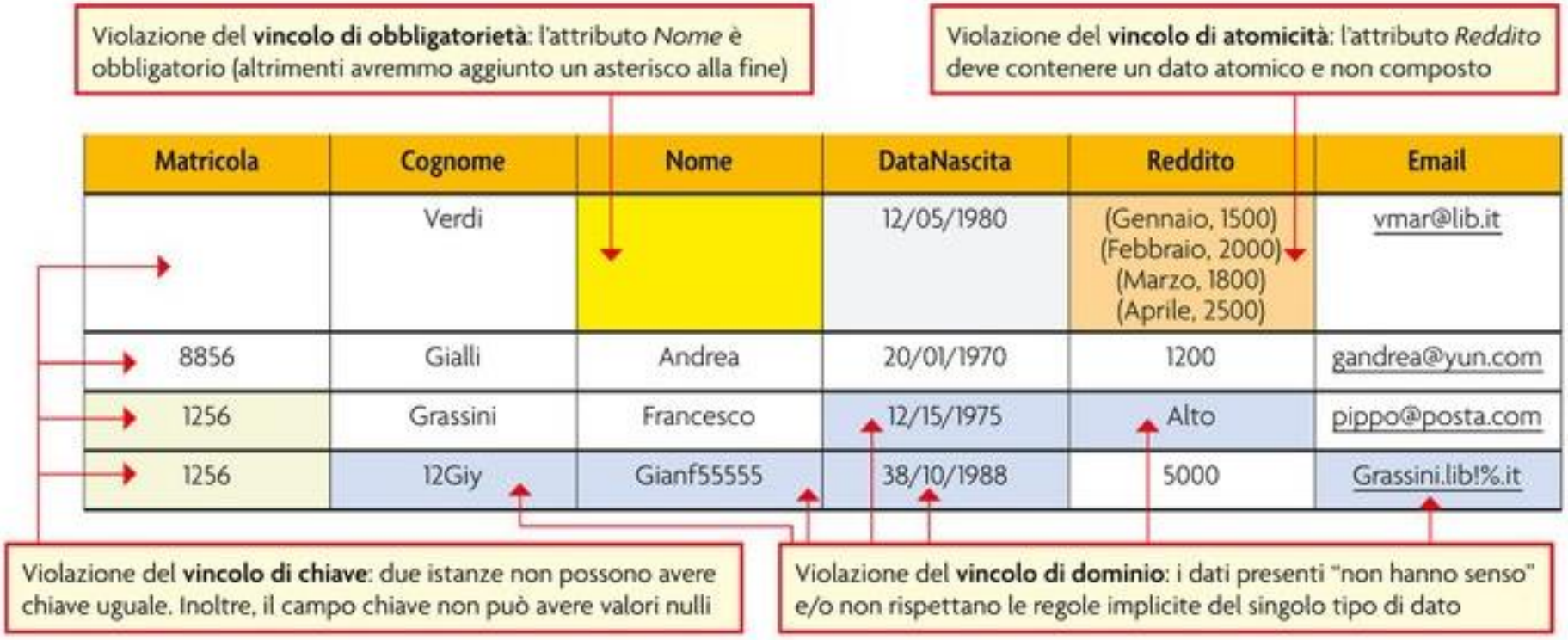
Una relazione non deve (e non può) essere vista come un contenitore di dati arbitrari. È necessaria un'attività di analisi volta a evidenziare quali sono i vincoli che le istanze devono soddisfare affinché possano essere considerate valide.

Classificazione vincoli d'integrità:

1) **Vincoli intrarelazionali** o **interni**, definiti all'interno di una singola relazione:
   a) vincoli su singola ennupla, che esprimono una condizione:
      - sul dominio degli attributi;
      - su più attributi;
   b) vincoli su più ennuple (in questa categoria i *vincoli di chiave primaria*);
2) **Vincoli interrelazionali** o **esterni**, definiti tra più relazioni (rientrano in questa categoria i *vincoli referenziali*).

# Vincoli d'integrità

Esempio di violazione di vincoli:




| Matricola | Cognome | Nome | DataNascita | Reddito | Email |
|---|---|---|---|---|---|
| | Verdi | | 12/05/1980 | (Gennaio, 1500)<br>(Febbraio, 2000)<br>(Marzo, 1800)<br>(Aprile, 2500) | vmar@lib.it |
| 8856 | Gialli | Andrea | 20/01/1970 | 1200 | gandrea@yun.com |
| 1256 | Grassini | Francesco | 12/15/1975 | Alto | pippo@posta.com |
| 1256 | 12Giy | Gianf55555 | 38/10/1988 | 5000 | Grassini.lib!%.it |

Esempio di vincoli intrarelazionali:

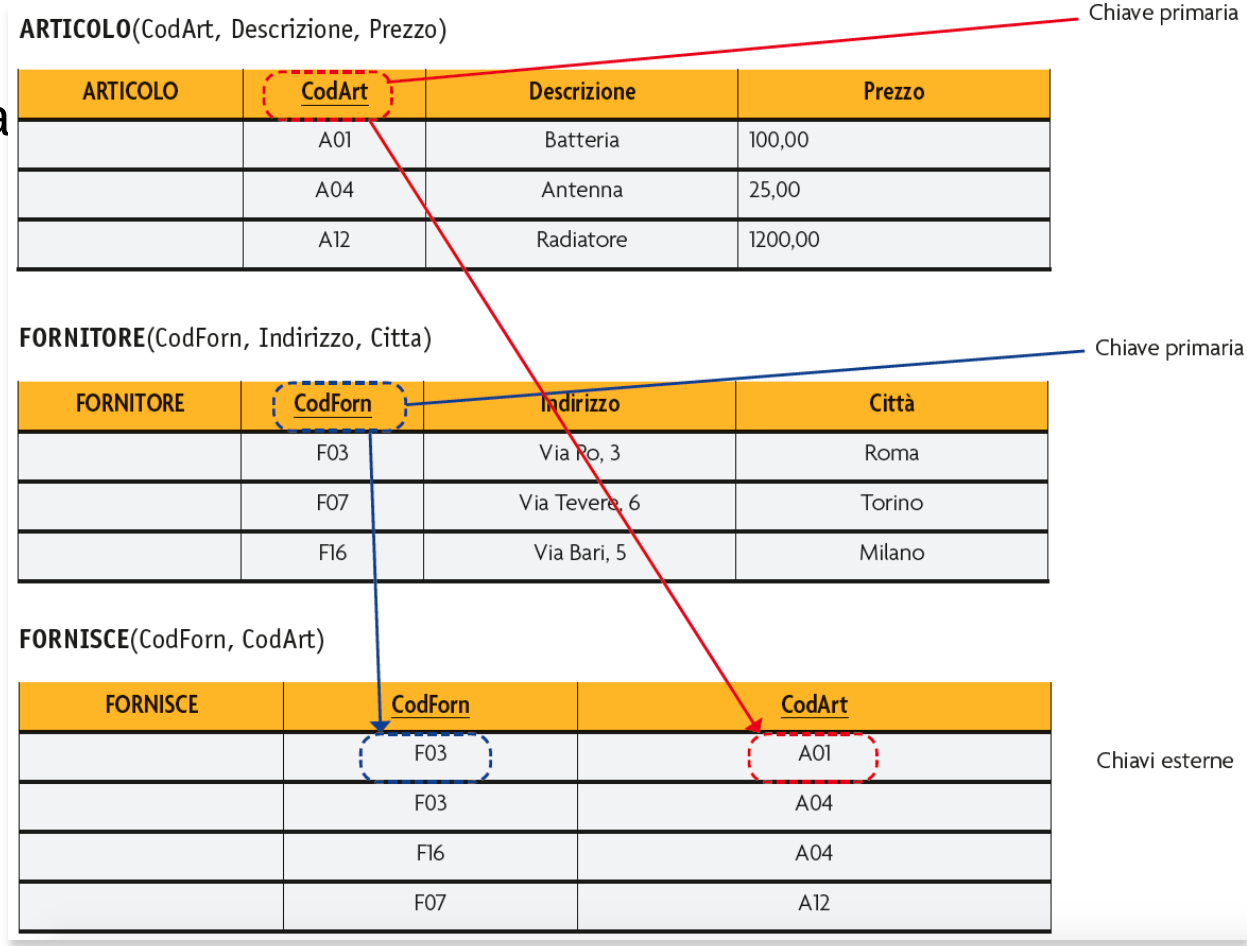
**DIPENDENTE**(Matricola, StipendioLordo, Trattenute, DataAssunzione, DataNasciata)

1. V1(Dipendente): StipendioLordo>0
2. V2(Dipendente): DataAssunzione>DataNascita
3. V3(Dipendente): Trattenute>0

# L’integrità referenziale

I **vincoli di integrità referenziale** riguardano i valori assunti dalle chiavi esterne nelle relazioni.

Poiché una chiave esterna è utilizzata per stabilire un legame tra relazioni, il suo valore deve essere tenuto in stretto controllo per le operazioni di inserimento, modifica e cancellazione.



# L’integrità referenziale

**OSS**.

*Mantenere l’integrità referenziale significa impedire agli utenti del database di interrompere accidentalmente le associazioni tra le tabelle correlate.*

Regole pratiche per l’integrità referenziale:

- Non è possibile immettere un valore nella chiave esterna della tabella associata se tale valore non esiste tra le chiavi della tabella primaria;
- Non è possibile eliminare una n-upla dalla tabella primaria se esistono righe a essa legate attraverso la chiave esterna nella tabella primaria;
- Non si può modificare il valore della chiave nella tabella primaria se a essa corrispondono righe nella tabella correlata.

# Gli operatori relazionali

I linguaggi di programmazione utilizzati per l'interrogazione sono di tipo **non procedurale** e si basano sull'algebra relazionale.

Secondo l'approccio basato sull'**algebra relazionale**, il risultato di un'interrogazione (**query**) è una relazione che si ottiene formulando un'interrogazione con l'uso di alcuni operatori dell'algebra relazionale.

Due relazioni R e S si dicono **compatibili** se:

- hanno lo stesso numero di attributi;
- ogni attributo nella stessa posizione all'interno delle due relazioni è dello stesso tipo.

Esempio di relazioni compatibili:

**MANAGER**(Nome: STRINGA, Stipendio: INTERO, DataNascita: DATA)

**DIPENDENTE**(Nominativo: STRINGA, RetribuzioneNetta: INTERO, DataAssunzione: DATA)

# Gli operatori relazionali: primitivi e derivati

Gli **operatori primitivi** sono:

1. Ridenominazione;
2. Unione;
3. Differenza;
4. Proiezione;
5. Restrizione (selezione);
6. Prodotto cartesiano.

Gli **operatori derivati** sono:

1. Intersezione;
2. Giunzione (Join).

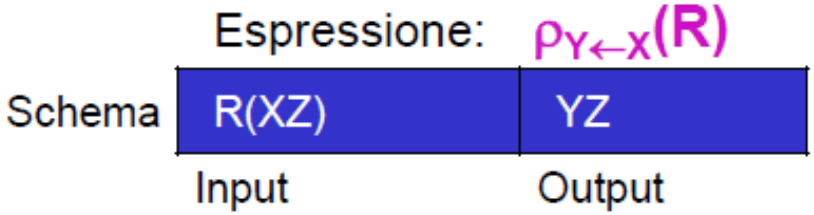
# Gli operatori primitivi: la ridenominazione

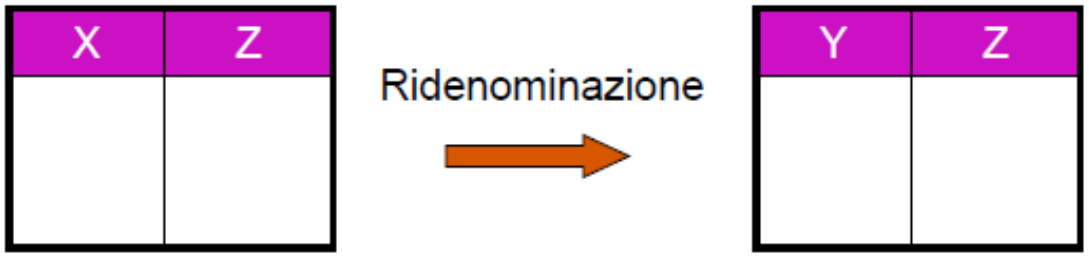
L'operazione di **ridenominazione ρ** modifica lo schema di una relazione, cambiando il nome di uno o più attributi.

In simboli:

$$\rho_{Y \leftarrow X}(r)$$

con r su R(XZ), cambia lo schema in YZ, lasciando invariati i valori delle n-uple. Nel caso in cui si cambi più di un attributo, l'ordine in cui si elencano gli attributi è significativo.



# Ridenominazione: esempi

Redditi

| CF | Imponibile |
|---|---|
| BNCGRG78F21A | 10000 |

$\rho_{CodiceFiscale \leftarrow CF}(Redditi)$

| CodiceFiscale | Imponibile |
|---|---|
| BNCGRG78F21A | 10000 |

VoliNoSmoking

| Numero | Giorno |
|---|---|
| SC278 | 28/07/2001 |
| SC315 | 30/07/2001 |

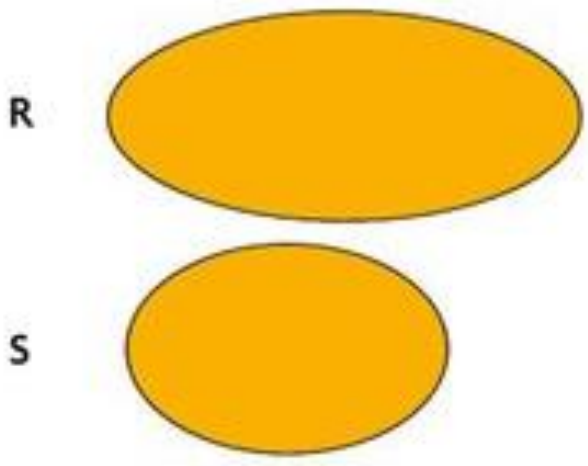
$\rho_{Codice,Data \leftarrow Numero,Giorno}(VoliNoSmoking)$

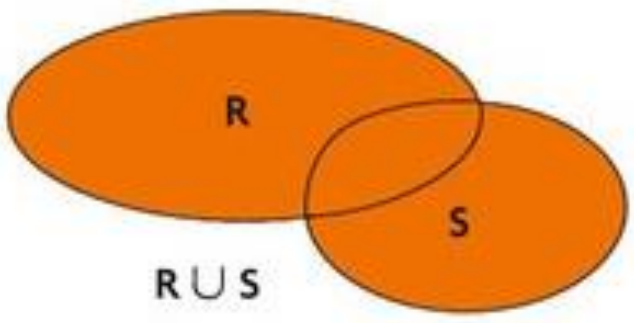
| Codice | Data |
|---|---|
| SC278 | 28/07/2001 |
| SC315 | 30/07/2001 |

# Gli operatori primitivi: l'unione

Date due relazioni compatibili R e S, l'**unione** di R con S è la relazione ottenuta dall'unione insiemistica delle due relazioni:

$$R \cup S = \{t \mid t \in R \text{ OR } t \in S\}$$



# Gli operatori primitivi: l'unione

**R = Clienti1Semestre**

| R | Cognome | | | |
|---|---|---|---|---|
| | Rossi | | | |
| | Bianchi | | | |
| | Verdi | | | |

**S = Clienti2Semestre**

| S | Cognome | | | |
|---|---|---|---|---|
| | Gialli | | | |
| | Bianchi | | | |
| | Neri | | | |

**R ∪ S = Clienti = Clienti1Semestre ∪ Clienti2Semestre**

| R ∪ S | Cognome | | | |
|---|---|---|---|---|
| | Rossi | | | |
| | Bianchi | | | |
| | Neri | | | |
| | Verdi | | | |
| | Gialli | | | |

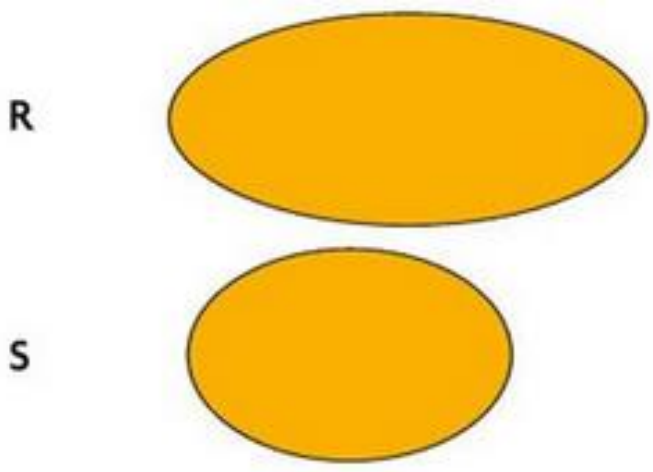
Grado(R ∪ S) = Grado(R) = Grado(S)

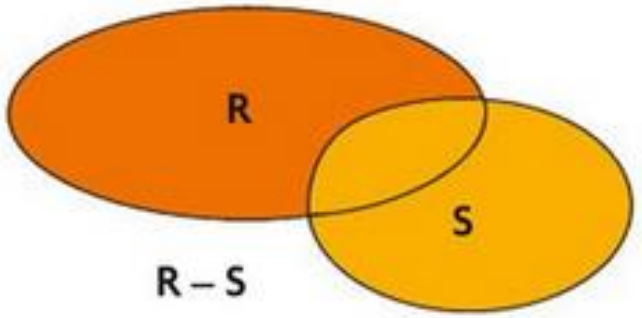
Card(R ∪ S) = Card(R) + Card(S) – numero n-uple ripetute

# Gli operatori primitivi: la differenza

Date due relazioni compatibili R e S, la **differenza** di R con S è la relazione data dalla differenza insiemistica delle due relazioni:

$$R - S = \{ t \mid t \in R \text{ AND } t \notin S \}$$



# Gli operatori primitivi: la differenza

**R = Clienti**

| R | Cognome | | | |
|---|---|---|---|---|
| | Rossi | | | |
| | Bianchi | | | |
| | Neri | | | |

**S = Clienti09**

| S | Cognome | | | |
|---|---|---|---|---|
| | Gialli | | | |
| | Bianchi | | | |
| | Neri | | | |

**R – S = Clienti – Clienti09**

| R – S | Cognome | | | |
|---|---|---|---|---|
| | Rossi | | | |

Grado(R – S) = Grado(R) = Grado(S)

Card(R – S) = Card(R) – numero n-uple presenti anche in S

# Unione e differenza: esempi

VoliCharter

| Codice | Data |
|---|---|
| XY123 | 21/07/2001 |
| SC278 | 28/07/2001 |
| XX338 | 18/08/2001 |

VoliNoSmoking

| Codice | Data |
|---|---|
| SC278 | 28/07/2001 |
| SC315 | 30/07/2001 |

VoliCharter ∪ VoliNoSmoking

| Codice | Data |
|---|---|
| XY123 | 21/07/2001 |
| SC278 | 28/07/2001 |
| XX338 | 18/08/2001 |
| SC315 | 30/07/2001 |

VoliCharter - VoliNoSmoking

| Codice | Data |
|---|---|
| XY123 | 21/07/2001 |
| XX338 | 18/08/2001 |

VoliNoSmoking - VoliCharter

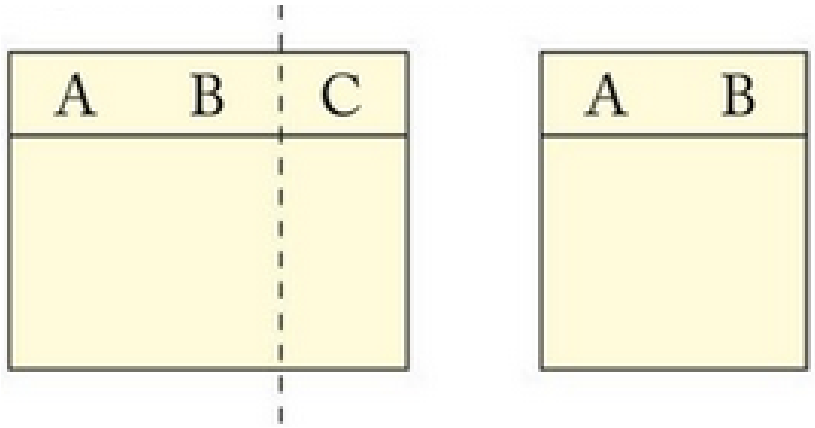
| Codice | Data |
|---|---|
| SC315 | 30/07/2001 |

# Gli operatori primitivi: la proiezione

Data una relazione R e un sottoinsieme $A = \{A_1, A_2, ..., A_k\}$ dei suoi attributi, si definisce **proiezione** di R su A, la relazione di grado K che si ottiene considerando solo le colonne di R relative agli attributi contenuti in A, ed eliminando le eventuali n-uple duplicate:

$$\pi_{A1, A2, ..., Ak}(R) = \{ t[A_1, A_2, ..., A_k] \; t \in R \}$$

L'effetto di una proiezione è di selezionare un certo numero di colonne dalla tabella della relazione.



# Gli operatori primitivi: la proiezione

**R = Clienti**

| R | CodCli | Cognome | Nome | Indirizzo | Citta |
|---|---|---|---|---|---|
| | C001 | Bianchi | Andrea | Via Po, 23 | Torino |
| | C002 | Neri | Paolo | Via Roma, 12 | Milano |
| | C003 | Verdi | Gianfranco | Via Europa, 150 | Torino |

$S = \pi_{Cognome,\ Nome}(R)$

| S | Cognome | Nome |
|---|---|---|
| | Bianchi | Andrea |
| | Neri | Paolo |
| | Verdi | Gianfranco |

Grado(S) ≤ Grado(R)

Card(S) ≤ Card(R)

# Proiezione: esempi

Corsi

| CodCorso | Titolo | Docente | Anno |
|---|---|---|---|
| 483 | Analisi | Biondi | 1 |
| 729 | Analisi | Neri | 1 |
| 913 | Sistemi Informativi | Castani | 2 |

$\pi_{CodCorso,Docente}(Corsi)$

| CodCorso | Docente |
|---|---|
| 483 | Biondi |
| 729 | Neri |
| 913 | Castani |

$\pi_{CodCorso,Anno}(Corsi)$

| CodCorso | Anno |
|---|---|
| 483 | 1 |
| 729 | 1 |
| 913 | 2 |

# Proiezione: esempi

Corsi

| CodCorso | Titolo | Docente | Anno |
|---|---|---|---|
| 483 | Analisi | Biondi | 1 |
| 729 | Analisi | Neri | 1 |
| 913 | Sistemi Informativi | Castani | 2 |

$\pi_{Titolo}(Corsi)$

| Titolo |
|---|
| Analisi |
| Sistemi Informativi |

$\pi_{Docente}(Corsi)$

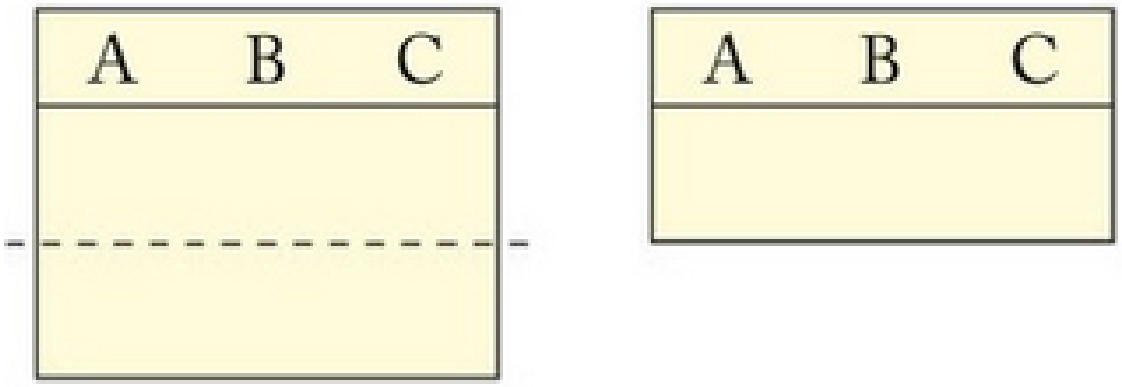
| Docente |
|---|
| Biondi |
| Neri |
| Castani |

# Gli operatori primitivi: la restrizione

Data una relazione R e un predicato P (semplice o composto) sui suoi attributi, la **restrizione** (**selezione**) di R a P è la relazione costituita dalle n-uple di R che soddisfano P:

$$\sigma_P(R) = \{ t \mid t \in R \text{ AND } P(t) = \text{VERO}\}$$

L'effetto di una restrizione è di selezionare un certo numero di righe dalla tabella della relazione.



# Gli operatori primitivi: la restrizione

**R = Clienti**

| R | CodCli | Cognome | Nome | Indirizzo | Citta |
|---|---|---|---|---|---|
| | C001 | Bianchi | Andrea | Via Po, 23 | Torino |
| | C002 | Neri | Paolo | Via Roma, 12 | Milano |
| | C003 | Verdi | Gianfranco | Via Europa, 150 | Torino |

$S = \sigma_{Citta="Torino"}(R)$

| S | CodCli | Cognome | Nome | Indirizzo | Citta |
|---|---|---|---|---|---|
| | C001 | Bianchi | Andrea | Via Po, 23 | Torino |
| | C003 | Verdi | Gianfranco | Via Europa, 150 | Torino |

Grado(S) = Grado(R)

Card(S) ≤ Card(R)

# Restrizione: esempi

Esami

| Matricola | CodCorso | Voto | Lode |
|---|---|---|---|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | SÌ |
| 29323 | 913 | 26 | NO |
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

$\sigma_{(Voto = 30)\ AND\ (Lode = NO)}(Esami)$

| Matricola | CodCorso | Voto | Lode |
|---|---|---|---|
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

$\sigma_{(CodCorso = 729)\ OR\ (Voto = 30)}(Esami)$

| Matricola | CodCorso | Voto | Lode |
|---|---|---|---|
| 39654 | 729 | 30 | SÌ |
| 35467 | 913 | 30 | NO |
| 31283 | 729 | 30 | NO |

# Restrizione: esempi

Partite

| Giornata | Casa | Ospite | GolCasa | GolOspite |
|---|---|---|---|---|
| 4 | Venezia | Bologna | 0 | 1 |
| 5 | Brescia | Atalanta | 3 | 3 |
| 5 | Inter | Bologna | 1 | 0 |
| 5 | Lazio | Parma | 0 | 0 |

$\sigma_{(Giornata = 5)\ AND\ (GolCasa = GolOspite)}(Partite)$

| Giornata | Casa | Ospite | GolCasa | GolOspite |
|---|---|---|---|---|
| 5 | Brescia | Atalanta | 3 | 3 |
| 5 | Lazio | Parma | 0 | 0 |

$\sigma_{(Ospite = Bologna)\ AND\ (GolCasa < GolOspite)}(Partite)$

| Giornata | Casa | Ospite | GolCasa | GolOspite |
|---|---|---|---|---|
| 4 | Venezia | Bologna | 0 | 1 |

# Proiezione e restrizione: esempio

Consideriamo le seguenti relazioni:

**STUDENTE**(Matricola, Cognome, Nome, CodScuola)

**SCUOLA**(CodScuola, NomeScuola, Città)

Per conoscere quali studenti frequentano la scuola "ITIS Fermi", che ha CodScuola=S001, scriveremo:

$$\pi_{\text{Cognome, Nome}}(\sigma_{\text{CodScuola} = \text{“S001”}}(\text{Studente}))$$

Si ha un’interrogazione composta da due sottointerrogazioni.

# Proiezione e restrizione: esempio

$$T = \sigma_{\text{CodScuola} = \text{"S001"}}(\text{Studente})$$

| Matricola | Cognome | Nome | CodScuola |
|---|---|---|---|
| M001 | Rossi | Paolo | S001 |
| M002 | Bianchi | Aldo | S001 |
| M003 | Verdi | Ada | S002 |
| M004 | Neri | Maria | S002 |

| Matricola | Cognome | Nome | CodScuola |
|---|---|---|---|
| M001 | Rossi | Paolo | S001 |
| M002 | Bianchi | Aldo | S001 |

# Proiezione e restrizione: esempio

$$\pi_{\text{Cognome, Nome}}(T)$$

| Matricola | Cognome | Nome | CodScuola |
|---|---|---|---|
| M001 | Rossi | Paolo | S001 |
| M002 | Bianchi | Aldo | S001 |

| Cognome | Nome |
|---|---|
| Rossi | Paolo |
| Bianchi | Aldo |

# Gli operatori primitivi: il prodotto cartesiano

Il prodotto cartesiano tra due insiemi A e B si definisce come l'insieme di tutte le possibili coppie che hanno come primo elemento un elemento di A e come secondo elemento un elemento di B.

$$A \times B = \{ (a,b) \mid a \in A, b \in B\}$$

Date due relazioni R e S, il **prodotto cartesiano** di R e S è l'insieme di tutte le possibili n-uple ottenute concatenando ogni n-upla di R con ogni n-upla di S. La chiave primaria del prodotto cartesiano è data dall'unione delle chiavi primarie di R e S.

Se Grado(R) = $g_1$, Grado(S) = $g_2$ e Card(R) = $c_1$, Card(S) = $c_2$, si ha

Grado(R × S) = $g_1 + g_2$

Card(R × S) = $c_1 \times c_2$

# Prodotto cartesiano: esempio

**R = Alunni**

| R | Matricola | Nominativo | Data di nascita |
|---|---|---|---|
| | 12346 | Bianchi Stefania | 6/4/1988 |
| | 12347 | De Pascalis Alberto | 12/9/1988 |
| | 12348 | Manca Roberta | 10/5/1988 |

**S = Testi**

| S | CodTesto | Titolo | Materia |
|---|---|---|---|
| | T1 | L'italiano oggi | Italiano |
| | T2 | Conoscere la matematica | Matematica |
| | T3 | Informatica e Azienda | Informatica |

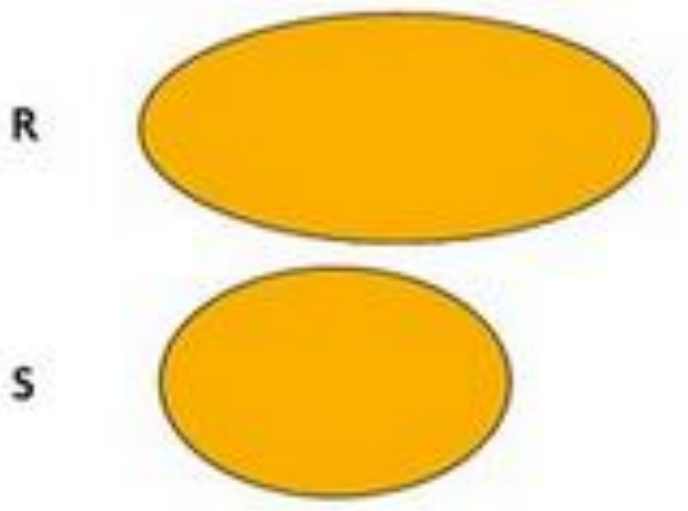
**R x S = Alunni x Testi**

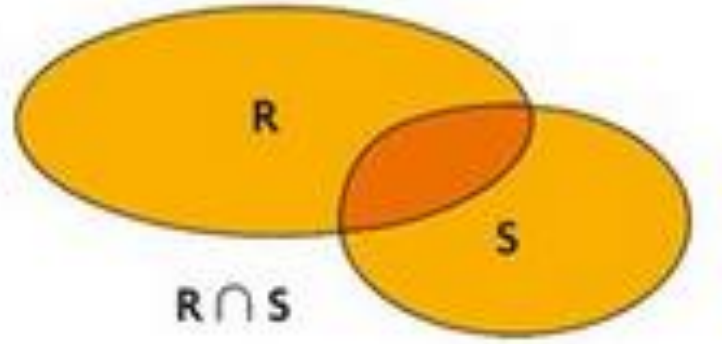
| R x S | Matricola | Nominativo | Data di nascita | CodTesto | Titolo | Materia |
|---|---|---|---|---|---|---|
| | 12346 | Bianchi Stefania | 6/4/1988 | T1 | L'italiano oggi | Italiano |
| | 12346 | Bianchi Stefania | 6/4/1988 | T2 | Conoscere la matematica | Matematica |
| | 12346 | Bianchi Stefania | 6/4/1988 | T3 | Informatica e Azienda | Informatica |
| | 12347 | De Pascalis Alberto | 12/9/1988 | T1 | L'italiano oggi | Italiano |
| | 12347 | De Pascalis Alberto | 12/9/1988 | T2 | Conoscere la matematica | Matematica |
| | 12347 | De Pascalis Alberto | 12/9/1988 | T3 | Informatica e Azienda | Informatica |
| | 12348 | Manca Roberta | 10/5/1988 | T1 | L'italiano oggi | Italiano |
| | 12348 | Manca Roberta | 10/5/1988 | T2 | Conoscere la matematica | Matematica |
| | 12348 | Manca Roberta | 10/5/1988 | T3 | Informatica e Azienda | Informatica |

# Gli operatori derivati: l'intersezione

Date due relazioni compatibili R e S, l'**intersezione** di R e S restituisce la relazione composta da tutte le n-uple presenti sia in R sia in S.

$$R \cap S = \{ t \mid t \in R \text{ AND } t \in S \}$$



,

# Gli operatori derivati: l'intersezione

**R = Clienti08**

| R | CodCli | Cognome | Provincia | Indirizzo |
|---|---|---|---|---|
| | C006 | Bianchi | MI | Via Po, 23 |
| | C002 | Neri | LE | Via Roma, 12 |
| | C005 | Rossi | MI | Via Moro, 2 |

**S = Clienti09**

| S | CodCli | Cognome | Provincia | Indirizzo |
|---|---|---|---|---|
| | C016 | Verdi | CO | Via Moro, 13 |
| | C002 | Neri | LE | Via Roma, 12 |
| | C005 | Rossi | MI | Via Moro, 2 |

**R ∩ S**

| R ∩ S | CodCli | Cognome | Provincia | Indirizzo |
|---|---|---|---|---|
| | C002 | Neri | LE | Via Roma, 12 |
| | C005 | Rossi | MI | Via Moro, 2 |

Grado(R ∩ S ) = Grado(S) = Grado(R)

Card(R ∩ S ) ≤ cardinalità minore tra Card(R) e Card(S)

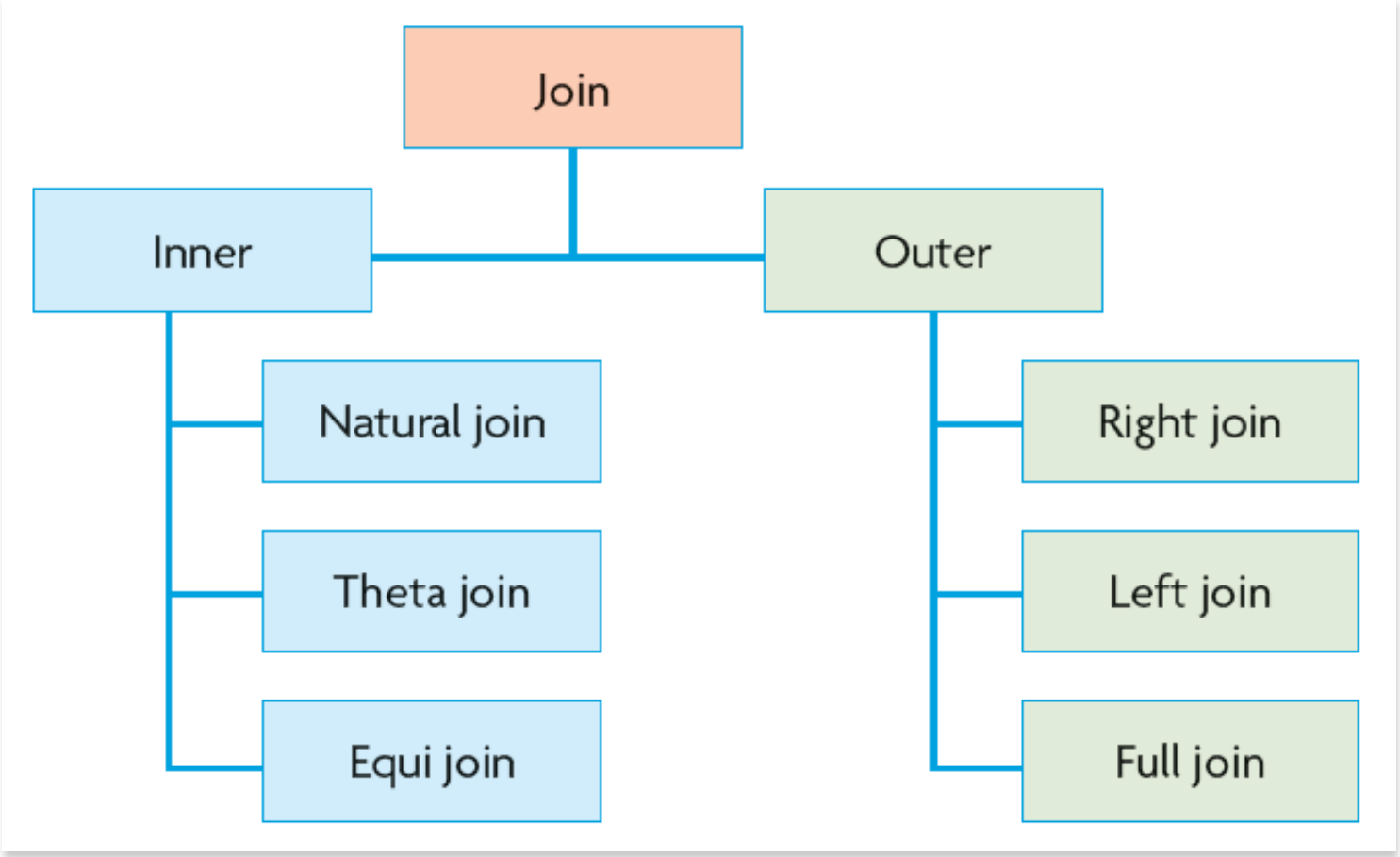
In particolare, si ha:

Card(R ∩ S ) = Card(R) se R ⊆ S

Card(R ∩ S ) = Card(S) se S ⊆ R

# Gli operatori derivati: il join (giunzione)

L'operazione di **join (giunzione)** consente di costruire una relazione applicando uno specifico criterio di restrizione sul prodotto cartesiano di due relazioni.



# Gli operatori derivati: natural join

Il **natural join** (**giunzione naturale**) è un operatore che correla i dati di due relazioni sulla base di uno o più attributi in comune (valori uguali in attributi con lo stesso nome) definiti sugli stessi domini. La relazione risultante contiene:

- l'unione degli attributi delle relazioni di partenza;
- le n-uple della prima relazione concatenate alle n-uple della seconda relazione secondo i valori uguali negli attributi comuni.

Date due relazioni R e S rispettivamente di grado $g_1$ e $g_2$, l'operazione di natural join di R e S su un attributo A di R e un attributo B di S, aventi lo stesso tipo, restituisce una relazione di grado (g1 + g2 – 1) le cui n-uple si ottengono con il seguente procedimento:

1. Si effettua il prodotto cartesiano di R e S;
2. Sulla relazione così ottenuta si effettua una restrizione sulle n-uple aventi gli attributi A e B dello stesso valore;
3. La relazione così ottenuta ha le colonne A e B uguali, per cui si elimina una di tali colonne.

In simboli: R ⋈ S

# Natural join: esempio

Siano R e S due tabelle che rappresentano i clienti e gli agenti di un'azienda. Determinare un'unica tabella in modo tale da avere le informazioni di ogni cliente e degli agenti da cui esso viene servito.

**R = Cliente**

| R | CodCli | NomeCli | IndirizzoCli | CodAg |
|---|---|---|---|---|
| | C006 | Bianchi | Via Po, 23 | A0052 |
| | C002 | Neri | Via Roma, 12 | A0016 |
| | C005 | Rossi | Via Moro, 2 | A0052 |

**S = Agente**

| S | CodAg | NomeAg | TelAgente |
|---|---|---|---|
| | A0016 | Polis | 346/5647523 |
| | A0052 | Rinaldi | 322/7665541 |

1. Effettuiamo il prodotto cartesiano:

| CodCli | NomeCli | IndirizzoCli | CodAg | CodAg | NomeAg | TelAgente |
|---|---|---|---|---|---|---|
| C006 | Bianchi | Via Po, 23 | A0052 | A0016 | Polis | 346/5647523 |
| C006 | Bianchi | Via Po, 23 | A0052 | A0052 | Rinaldi | 322/7665541 |
| C002 | Neri | Via Roma, 12 | A0016 | A0016 | Polis | 346/5647523 |
| C002 | Neri | Via Roma, 12 | A0016 | A0052 | Rinaldi | 322/7665541 |
| C005 | Rossi | Via Moro, 2 | A0052 | A0016 | Polis | 346/5647523 |
| C005 | Rossi | Via Moro, 2 | A0052 | A0052 | Rinaldi | 322/7665541 |

# Natural join: esempio

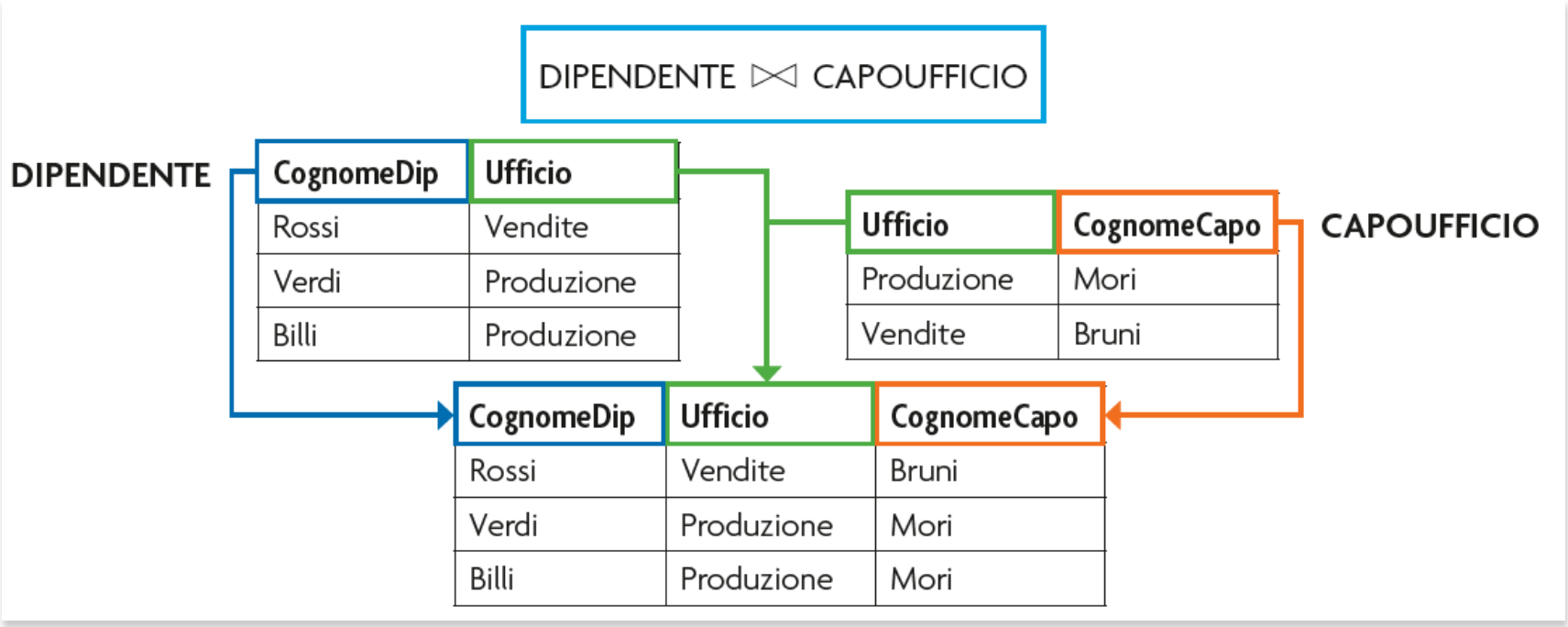
2. Effettuiamo la restrizione del prodotto cartesiano sull'attributo CodAg:

| CodCli | NomeCli | IndirizzoCli | CodAg | CodAg | NomeAg | TelAgente |
|---|---|---|---|---|---|---|
| C006 | Bianchi | Via Po, 23 | A0052 | A0052 | Rinaldi | 322/7665541 |
| C002 | Neri | Via Roma, 12 | A0016 | A0016 | Polis | 346/5647523 |
| C005 | Rossi | Via Moro, 2 | A0052 | A0052 | Rinaldi | 322/7665541 |

3. Eliminando una delle due colonne uguali, otteniamo:

| CodCli | NomeCli | IndirizzoCli | CodAg | NomeAg | TelAgente |
|---|---|---|---|---|---|
| C006 | Bianchi | Via Po, 23 | A0052 | Rinaldi | 322/7665541 |
| C002 | Neri | Via Roma, 12 | A0016 | Polis | 346/5647523 |
| C005 | Rossi | Via Moro, 2 | A0052 | Rinaldi | 322/7665541 |

# Natural join: esempio



# Natural join: esempio

Esami

| Matricola | CodCorso | Voto | Lode |
|---|---|---|---|
| 29323 | 483 | 28 | NO |
| 39654 | 729 | 30 | SÌ |
| 29323 | 913 | 26 | NO |
| 35467 | 913 | 30 | NO |

Corsi

| CodCorso | Titolo | Docente | Anno |
|---|---|---|---|
| 483 | Analisi | Biondi | 1 |
| 729 | Analisi | Neri | 1 |
| 913 | Sistemi Informativi | Castani | 2 |

Esami $\triangleright\triangleleft$ Corsi

| Matricola | CodCorso | Voto | Lode | Titolo | Docente | Anno |
|---|---|---|---|---|---|---|
| 29323 | 483 | 28 | NO | Analisi | Biondi | 1 |
| 39654 | 729 | 30 | SÌ | Analisi | Neri | 1 |
| 29323 | 913 | 26 | NO | Sistemi Informativi | Castani | 2 |
| 35467 | 913 | 30 | NO | Sistemi Informativi | Castani | 2 |

# Natural join: esempi

Voli

| Codice | Data | Comandante |
|---|---|---|
| AZ427 | 21/07/2001 | Bianchi |
| AZ427 | 23/07/2001 | Rossi |
| TW056 | 21/07/2001 | Smith |

Linee

| Codice | Partenza | Arrivo |
|---|---|---|
| AZ427 | FCO | JFK |
| TW056 | LAX | FCO |

Prenotazioni

| Codice | Data | Classe | Cliente |
|---|---|---|---|
| AZ427 | 21/07/2001 | Economy | Anna Bini |
| AZ427 | 21/07/2001 | Business | Franco Dini |
| AZ427 | 23/07/2001 | Economy | Ada Cini |

Voli $\triangleright\!\triangleleft$ Linee

| Codice | Data | Comandante | Partenza | Arrivo |
|---|---|---|---|---|
| AZ427 | 21/07/2001 | Bianchi | FCO | JFK |
| AZ427 | 23/07/2001 | Rossi | FCO | JFK |
| TW056 | 21/07/2001 | Smith | LAX | FCO |

# Natural join: esempi

Voli ⋈ Prenotazioni

| Codice | Data | Comandante | Classe | Cliente |
|---|---|---|---|---|
| AZ427 | 21/07/2001 | Bianchi | Economy | Anna Bini |
| AZ427 | 21/07/2001 | Bianchi | Business | Franco Dini |
| AZ427 | 23/07/2001 | Rossi | Economy | Ada Cini |

LInee ⋈ Prenotazioni

| Codice | Partenza | Arrivo | Data | Classe | Cliente |
|---|---|---|---|---|---|
| AZ427 | FCO | JFK | 21/07/2001 | Economy | Anna Bini |
| AZ427 | FCO | JFK | 21/07/2001 | Business | Franco Dini |
| AZ427 | FCO | JFK | 23/07/2001 | Economy | Ada Cini |

# Natural join e intersezione

Se due relazioni hanno lo stesso schema allora il natural join coincide con l'intersezione delle due relazioni.

VoliCharter

| Codice | Data |
|---|---|
| XY123 | 21/07/2001 |
| SC278 | 28/07/2001 |
| XX338 | 18/08/2001 |

VoliNoSmoking

| Codice | Data |
|---|---|
| SC278 | 28/07/2001 |
| SC315 | 30/07/2001 |

VoliCharter ⋈ VoliNoSmoking

| Codice | Data |
|---|---|
| SC278 | 28/07/2001 |

# Natural join e prodotto cartesiano

Se due relazioni non hanno attributi in comune allora il natural join coincide con il prodotto cartesiano delle due relazioni.

VoliCharter

| Codice | Data |
|---|---|
| XY123 | 21/07/2001 |
| SC278 | 28/07/2001 |
| XX338 | 18/08/2001 |

VoliNoSmoking

| Numero | Giorno |
|---|---|
| SC278 | 28/07/2001 |
| SC315 | 30/07/2001 |

VoliCharter ⋈ VoliNoSmoking

| Codice | Data | Numero | Giorno |
|---|---|---|---|
| XY123 | 21/07/2001 | SC278 | 28/07/2001 |
| SC278 | 28/07/2001 | SC278 | 28/07/2001 |
| XX338 | 18/08/2001 | SC278 | 28/07/2001 |
| XY123 | 21/07/2001 | SC315 | 30/07/2001 |
| SC278 | 28/07/2001 | SC315 | 30/07/2001 |
| XX338 | 18/08/2001 | SC315 | 30/07/2001 |

# Natural join e ridenominazione: self-join

La ridenominazione permette di eseguire il natural join di una relazione su se stessa (**self-join**) in modo significativo.

Genitori

| Genitore | Figlio |
|---|---|
| Luca | Anna |
| Maria | Anna |
| Giorgio | Luca |
| Silvia | Maria |
| Enzo | Maria |

**Per trovare nonni e nipoti:**

$\rho_{Nonno,Genitore \leftarrow Genitore,Figlio}(Genitori)$

| Nonno | Genitore |
|---|---|
| Luca | Anna |
| Maria | Anna |
| Giorgio | Luca |
| Silvia | Maria |
| Enzo | Maria |

$\rho_{Nonno,Genitore \leftarrow Genitore,Figlio}(Genitori) \triangleright\triangleleft Genitori$

| Nonno | Genitore | Figlio |
|---|---|---|
| Giorgio | Luca | Anna |
| Silvia | Maria | Anna |
| Enzo | Maria | Anna |

… poi si può ridenominare Figlio in Nipote e proiettare su {Nonno,Nipote}

# Gli operatori derivati: theta join

Il **theta join** di due relazioni R e S, rispetto a un attributo A di R e un attributo B di S, è una relazione che si ottiene dal prodotto R × S selezionando le n-uple che soddisfano il criterio di selezione presente nella condizione di join A $\theta$ B. In simboli:

$$R \underset{A\,\theta\,B}{\bowtie} S \equiv \delta_{A\,\theta\,B}(R \times S)$$

dove A e B sono gli attributi delle relazioni corrispondenti al dominio in comune, $\theta$ è l'operatore di confronto (=, <, >, ≤, ≥, ≠).

L'operazione di theta join restituisce una relazione ottenuta con il seguente procedimento:

1. Si effettua il prodotto cartesiano di R e S;
2. Sulla relazione così ottenuta si effettua una selezione delle n-uple in cui gli attributi appartenenti al dominio in comune soddisfano la condizione;
3. Si rinominano gli attributi comuni con lo stesso nome, in modo che compaiano una volta sola.

Se l'operatore $\theta$ è quello di uguaglianza (=) si parla di **equi join**.

# Gli operatori derivati: theta join

Il **theta join** di due relazioni R e S, rispetto a un attributo A di R e un attributo B di S, è una relazione che si ottiene dal prodotto R × S selezionando le n-uple che soddisfano il criterio di selezione presente nella condizione di join A $\theta$ B. In simboli:

$$R \underset{A\,\theta\,B}{\bowtie} S \equiv \delta_{A\,\theta\,B}(R \times S)$$

dove A e B sono gli attributi delle relazioni corrispondenti al dominio in comune, $\theta$ è l'operatore di confronto (=, <, >, ≤, ≥, ≠).

L'operazione di theta join restituisce una relazione ottenuta con il seguente procedimento:

1. Si effettua il prodotto cartesiano di R e S;
2. Sulla relazione così ottenuta si effettua una selezione delle n-uple in cui gli attributi appartenenti al dominio in comune soddisfano la condizione;
3. Si rinominano gli attributi comuni con lo stesso nome, in modo che compaiano una volta sola.

Se l'operatore $\theta$ è quello di uguaglianza (=) si parla di **equi join**.

# Theta join: esempi

IMPIEGATO

| Cognome | CodProgetto |
|---|---|
| Rossi | A |
| Neri | A |
| Neri | B |

| ID | NomeProgetto |
|---|---|
| A | Venere |
| B | Marte |

PROGETTO

IMPIEGATO ⋈ PROGETTO

| Cognome | CodProgetto | ID | NomeProgetto |
|---|---|---|---|
| Rossi | A | A | Venere |
| Neri | A | A | Venere |
| ~~Neri~~ | ~~B~~ | ~~A~~ | ~~Venere~~ |
| ~~Rossi~~ | ~~A~~ | ~~B~~ | ~~Marte~~ |
| ~~Neri~~ | ~~A~~ | ~~B~~ | ~~Marte~~ |
| Neri | B | B | Marte |

IMPIEGATO ⋈ PROGETTO
CodProgetto = ID

| Cognome | CodProgetto | ID | NomeProgetto |
|---|---|---|---|
| Rossi | A | A | Venere |
| Neri | A | A | Venere |
| Neri | B | B | Marte |

# Theta join: esempi

Ricercatori

| Nome | CodProgetto |
|---|---|
| Rossi | HK27 |
| Verdi | HAL2000 |
| Bianchi | HK27 |
| Verdi | HK28 |
| Neri | HAL2000 |

Progetti

| Sigla | Responsabile |
|---|---|
| HK27 | Bianchi |
| HAL2000 | Neri |
| HK28 | Verdi |

Ricercatori $\triangleright\triangleleft_{\text{CodProgetto=Sigla}}$ Progetti

| Nome | CodProgetto | Sigla | Responsabile |
|---|---|---|---|
| Rossi | HK27 | HK27 | Bianchi |
| Verdi | HAL2000 | HAL2000 | Neri |
| Bianchi | HK27 | HK27 | Bianchi |
| Verdi | HK28 | HK28 | Verdi |
| Neri | HAL2000 | HAL2000 | Neri |

Ricercatori $\triangleright\triangleleft_{\text{(CodProgetto=Sigla) AND (Nome} \neq \text{Responsabile)}}$ Progetti

| Nome | CodProgetto | Sigla | Responsabile |
|---|---|---|---|
| Rossi | HK27 | HK27 | Bianchi |
| Verdi | HAL2000 | HAL2000 | Neri |

# Gli operatori derivati: esercizio

Consideriamo le seguenti relazioni:

**FILM1**(CodFilm1, Titolo, Lunghezza, NomeStudio)

**FILM2**(CodFilm2, AttorePrincipale, CodiceFilm1)

Interrogazione: "Trovare l'attore principale dei film che durano almeno 120 minuti".

Query:

$$\pi_{\text{AttorePrincipale}}(\sigma_{\text{Lunghezza} > 120}(\text{Film1} \underset{\text{CodFilm1} = \text{CodiceFilm1}}{\triangleright\!\!\!\bowtie} \text{Film2}))$$

# Algebra con valori NULL:

- La presenza di valori nulli nelle istanze richiede un'estensione della semantica degli operatori;
- Inoltre, è utile considerare una estensione del join naturale che non scarta le n-uple dangling, ma genera valori nulli;
- Va premesso che esistono diversi approcci al trattamento dei valori nulli, nessuno dei quali è completamente soddisfacente (per ragioni formali e/o pragmatiche) ;
- L'approccio che qui si presenta è quello "tradizionale", che ha il pregio di essere molto simile a quello adottato in SQL (e quindi dai DBMS relazionali) .

# Proiezione, unione e differenza con valori NULL

Proiezione, unione e differenza continuano a comportarsi usualmente, quindi due n-uple sono uguali anche se ci sono dei valori NULL.

Impiegati

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 435 | Verdi | NULL |

$\pi_{Nome,Ufficio}(Impiegati)$

| Nome | Ufficio |
|---|---|
| Rossi | A12 |
| Verdi | NULL |
| Verdi | A27 |

Responsabili

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| NULL | NULL | A27 |
| 435 | Verdi | NULL |

Impiegati $\cup$ Responsabili

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 435 | Verdi | NULL |
| NULL | NULL | A27 |

# Selezione con valori NULL

Per la selezione il problema è stabilire se, in presenza di NULL, un predicato è vero o meno per una data n-upla.

Impiegati

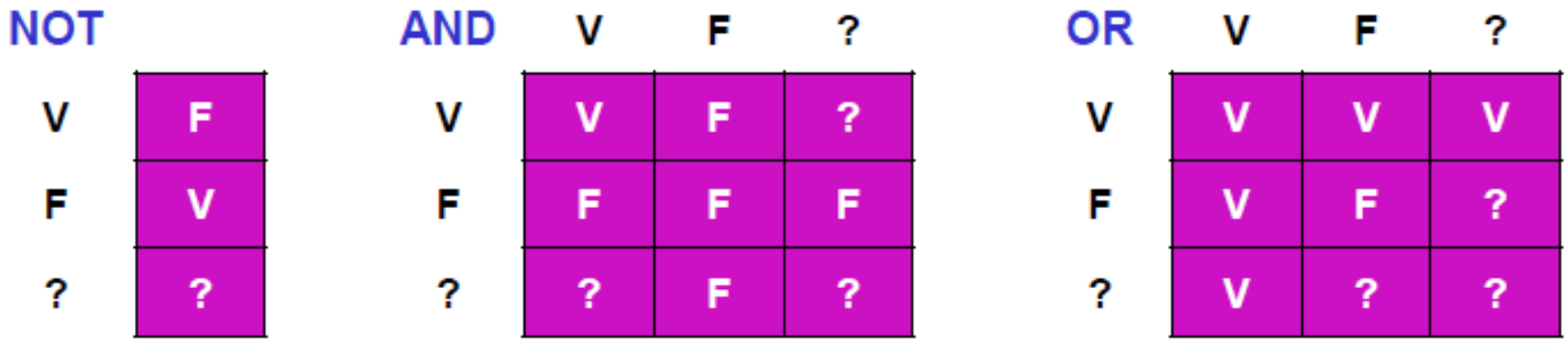
| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |

$\sigma_{Ufficio = A12}(Impiegati)$

- Sicuramente la prima n-upla fa parte del risultato e la terza no;
- Ma la seconda? Nulla si può dire;
- E la stessa cosa nel caso in cui si avesse $\sigma_{Ufficio \neq A12}(Impiegati)$!

# Selezione: logica a tre valori

Oltre ai valori di verità V e F, si introduce il valore Sconosciuto (?):



| NOT | |
|---|---|
| V | F |
| F | V |
| ? | ? |

| AND | V | F | ? |
|---|---|---|---|
| V | V | F | ? |
| F | F | F | F |
| ? | ? | F | ? |

| OR | V | F | ? |
|---|---|---|---|
| V | V | V | V |
| F | V | F | ? |
| ? | V | ? | ? |

- Una selezione produce le sole n-uple per cui l'espressione dei predicati risulti vera;
- Per lavorare esplicitamente con i NULL si introduce l'operatore di confronto IS, ad esempio A IS NULL;
- NOT ( A IS NULL) si scrive anche A IS NOT NULL.

# Selezione con valori NULL: esempi

Impiegati

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{\text{Ufficio} = \text{A12}}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |

$\sigma_{(\text{Ufficio} = \text{A12}) \text{ OR } (\text{Ufficio} \neq \text{A12})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{(\text{Ufficio} = \text{A27}) \text{ AND } (\text{Nome} = \text{Verdi})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|
| 373 | Verdi | A27 |

$\sigma_{(\text{Ufficio} = \text{A27}) \text{ OR } (\text{Nome} = \text{Verdi})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 385 | NULL | A27 |

$\sigma_{\text{Ufficio IS NULL}}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|
| 231 | Verdi | NULL |

$\sigma_{(\text{Ufficio IS NULL}) \text{ AND } (\text{Nome IS NULL})}(\text{Impiegati})$

| Cod | Nome | Ufficio |
|---|---|---|

# Natural join con valori NULL

Il natural join non combina due n-uple se queste hanno entrambe valore null su un attributo in comune (e valori uguali sugli eventuali altri attributi comuni)

Impiegati

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |
| 231 | Verdi | NULL |
| 373 | Verdi | A27 |
| 435 | Verdi | NULL |

Responsabili

| Ufficio | Cod |
|---|---|
| A12 | 123 |
| A27 | NULL |
| NULL | 231 |

Impiegati ⋈ Responsabili

| Cod | Nome | Ufficio |
|---|---|---|
| 123 | Rossi | A12 |

# Inner join e outer join

L’**inner join (giunzione interna)** restituisce solo i record verificati esistenti in entrambe le tabelle, escludendo, di conseguenza, tutte le n-uple che non hanno corrispondenza.

L’**outer join (giunzione esterna)** restituisce tutti i record delle due tabelle, senza escludere le n-uple che non hanno corrispondenza. In particolare, l’outer join completa con valori NULL le n-uple che non hanno corrispondenza.

# Outer join: il problema

**PERSONA**

| Codice | Nome | Cognome |
|---|---|---|
| 1 | Mario | Rossi |
| 2 | Luigi | Bianchi |
| 3 | Giuseppe | Neri |

**AUTOMOBILE**

| Targa | Modello | Codice |
|---|---|---|
| AASSGG | Tipo1 | 1 |
| UUJJHH | Tipo2 | 1 |
| PPLLBB | Tipo3 | 2 |
| WWYYXX | Tipo4 | |

Applichiamo un natural join alle due tabelle sull'attributo Codice, e proiettiamo su tutte le colonne tranne Codice:

| Targa | Modello | Nome | Cognome |
|---|---|---|---|
| AASSGG | Tipo1 | Mario | Rossi |
| UUJJHH | Tipo2 | Mario | Rossi |
| PPLLBB | Tipo3 | Luigi | Bianchi |

# Right outer join

Date due relazioni A e B, il **right outer join** restituisce una relazione selezionando tutte le n-uple di A che corrispondono a B, più i record di B (relazione di destra) che non corrispondono. Le n-uple che non corrispondono vengono riempite col valore NULL.

| Targa | Modello | Nome | Cognome |
|---|---|---|---|
| AASSGG | Tipo1 | Mario | Rossi |
| UUJJHH | Tipo2 | Mario | Rossi |
| PPLLBB | Tipo3 | Luigi | Bianchi |
| Null | Null | Giuseppe | Neri |

# Left outer join

Date due relazioni A e B, il **left outer join** restituisce una relazione selezionando tutte le n-uple di A che corrispondono a B, più i record di A (relazione di sinistra) che non corrispondono. Le n-uple che non corrispondono vengono riempite col valore NULL.

| Targa | Modello | Nome | Cognome |
|---|---|---|---|
| AASSGG | Tipo1 | Mario | Rossi |
| UUJJHH | Tipo2 | Mario | Rossi |
| PPLLBB | Tipo3 | Luigi | Bianchi |
| WWYYXX | Tipo4 | Null | Null |

# Full outer join

Il **full outer join** applica contemporaneamente sia il left outer join sia il right outer join.

| Targa | Modello | Nome | Cognome |
|---|---|---|---|
| AASSGG | Tipo1 | Mario | Rossi |
| UUJJHH | Tipo2 | Mario | Rossi |
| PPLLBB | Tipo3 | Luigi | Bianchi |
| Null | Null | Giuseppe | Neri |
| WWYYXX | Tipo4 | Null | Null |

# Outer join: esempi

Ricercatori

| Nome | CodProgetto |
|---|---|
| Rossi | HK27 |
| Bianchi | HK27 |
| Verdi | HK28 |

Progetti

| CodProgetto | Responsabile |
|---|---|
| HK27 | Bianchi |
| HAL2000 | Neri |

Ricercatori =▷◁ Progetti

| Nome | CodProgetto | Responsabile |
|---|---|---|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| Verdi | HK28 | NULL |

Ricercatori ▷◁= Progetti

| Nome | CodProgetto | Responsabile |
|---|---|---|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| NULL | HAL2000 | Neri |

Ricercatori =▷◁= Progetti

| Nome | CodProgetto | Responsabile |
|---|---|---|
| Rossi | HK27 | Bianchi |
| Bianchi | HK27 | Bianchi |
| Verdi | HK28 | NULL |
| NULL | HAL2000 | Neri |

# Database per esercizi

## Imp

| CodImp | Nome | Sede | Ruolo | Stipendio |
|---|---|---|---|---|
| E001 | Rossi | S01 | Analista | 2000 |
| E002 | Verdi | S02 | Sistemista | 1500 |
| E003 | Bianchi | S01 | Programmatore | 1000 |
| E004 | Gialli | S03 | Programmatore | 1000 |
| E005 | Neri | S02 | Analista | 2500 |
| E006 | Grigi | S01 | Sistemista | 1100 |
| E007 | Violetti | S01 | Programmatore | 1000 |
| E008 | Aranci | S02 | Programmatore | 1200 |

## Sedi

| Sede | Responsabile | Citta |
|---|---|---|
| S01 | Biondi | Milano |
| S02 | Mori | Bologna |
| S03 | Fulvi | Milano |

## Prog

| CodProg | Citta |
|---|---|
| P01 | Milano |
| P01 | Bologna |
| P02 | Bologna |

# Esercizi proposti

Scrivere delle espressioni per rispondere alle seguenti interrogazioni.

1. Nome, sede e stipendio degli impiegati che guadagnano più di 1300 Euro;

$$\pi_{Nome,Sede,Stipendio}(\sigma_{Stipendio > 1300}(Imp))$$

| Nome | Sede | Stipendio |
|---|---|---|
| Rossi | S01 | 2000 |
| Verdi | S02 | 1500 |
| Neri | S02 | 2500 |

2. Sedi, responsabili e città degli impiegati che guadagnano più di 1300 Euro;

$$\pi_{Sede,Responsabile,Citta}(Sedi \triangleright\!\triangleleft (\sigma_{Stipendio > 1300}(Imp)))$$

| Sede | Responsabile | Citta |
|---|---|---|
| S01 | Biondi | Milano |
| S02 | Mori | Bologna |

# Esercizi proposti

3. Progetti nelle città delle sedi degli impiegati che guadagnano più di 1300 Euro;

| CodProg |
| --- |
| P01 |
| P02 |

4. Responsabili delle sedi senza sistemisti.

| Responsabile |
| --- |
| Fulvi |

# Equivalenza di espressioni

Un'interrogazione su un database con schema **R** può a tutti gli effetti essere vista come una funzione che a ogni istanza **r** di **R** associa una relazione con un dato schema.

Un'espressione è quindi un modo specifico per esprimere (rappresentare) tale funzione, e due espressioni sono tra loro equivalenti se rappresentano la stessa funzione:

*Due espressioni E1 ed E2 espresse su un DB **R** si dicono equivalenti rispetto a **R** se e solo se per ogni istanza **r** di **R** producono lo stesso risultato.*

# Equivalenze: considerazioni

- Due espressioni equivalenti E1 ed E2 garantiscono lo stesso risultato, ma ciò non significa che la scelta sia indifferente in termini di “risorse” necessarie;
- Considerazioni di questo tipo sono essenziali in fase di ottimizzazione, in cui la conoscenza delle regole di equivalenza può consentire di eseguire delle trasformazioni che possono portare a un’espressione valutabile in modo più efficiente rispetto a quella iniziale;
- In particolare, le regole più interessanti sono quelle che permettono di ridurre la cardinalità degli operandi e quelle che portano a una semplificazione dell’espressione.

# Regole di equivalenza

Tra le regole base di equivalenza, si hanno le seguenti:

- Il natural join è commutativo e associativo:

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1 \qquad (E_1 \bowtie E_2) \bowtie E_3 \equiv E_1 \bowtie (E_2 \bowtie E_3) \equiv E_1 \bowtie E_2 \bowtie E_3$$

- Selezione e proiezione si possono raggruppare:

$$\sigma_{F1}(\sigma_{F2}(E)) \equiv \sigma_{F1 \text{ AND } F2}(E) \qquad \pi_Y(\pi_{YZ}(E)) \equiv \pi_Y(E)$$

- Selezione e proiezione commutano:

$$\pi_Y(\sigma_F(E)) \equiv \sigma_F(\pi_Y(E))$$

- “Push-Down” della selezione rispetto al join:

$$\sigma_F(E_1 \bowtie E_2) \equiv \sigma_F(E_1) \bowtie E_2$$