

Unità **C3**



LABORATORI "CASE STUDIES"  
Approfondisci con i casi pratici

# La progettazione logica: il modello relazionale

In questa unità imparerai...

- A che cosa serve la progettazione logica
- Come tradurre uno schema concettuale in uno schema logico
- Quali sono e come si usano gli operatori dell'algebra relazionale
- Come normalizzare le relazioni

# 1 La progettazione logica

L'obiettivo di questa fase è produrre uno schema logico in grado di descrivere, in modo corretto ed efficiente, tutte le informazioni rappresentate dallo schema ER prodotto nella fase di progettazione concettuale. Non si tratta però di una semplice trasformazione da uno schema a un altro, in quanto, prima di passare allo schema logico, il diagramma ER deve essere ristrutturato in modo da **semplificare** la traduzione e **ottimizzare** il progetto finale.

---

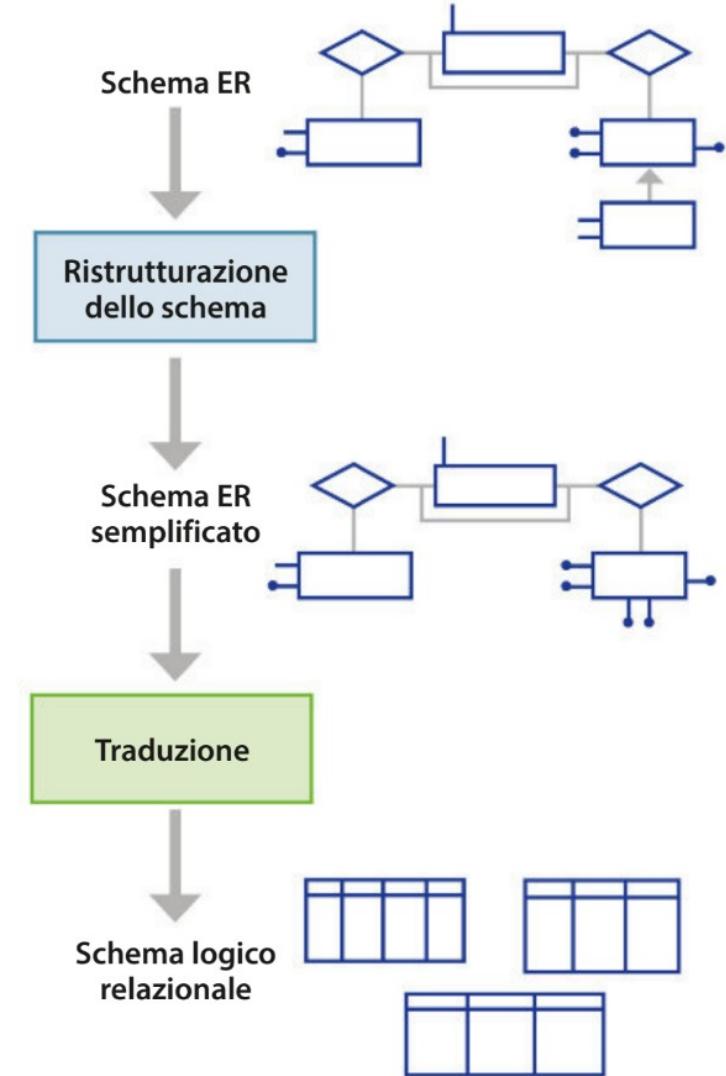
**ristrutturazione dello schema Entità-Associazione:** è una fase indipendente dal modello logico scelto e si concentra sui criteri di ottimizzazione dello schema e di semplificazione per la fase successiva;

---

---

**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.

---



## 2 Ristrutturazione dello schema ER

Il seguente elenco rappresenta l'insieme dei passi da svolgere, in sequenza, nella fase di ristrutturazione dello schema Entità-Associazioni.

- **Analisi delle ridondanze:** vengono individuate eventuali ridondanze per stabilire se è necessario eliminarle o mantenerle.
- **Eliminazione delle generalizzazioni:** tutte le generalizzazioni vengono sostituite con altri costrutti del modello ER.
- **Partizionamento/accorpamento di entità e associazioni:** si valuta se è opportuno separare (partizionare) o aggregare (accorpate) concetti presenti nello schema.
- **Scelta delle chiavi primarie:** la scelta delle chiavi primarie è essenziale nella traduzione verso lo schema relazionale, perché le chiavi che identificano le occorrenze consentono di stabilire i legami tra i dati.

## Analisi delle ridondanze

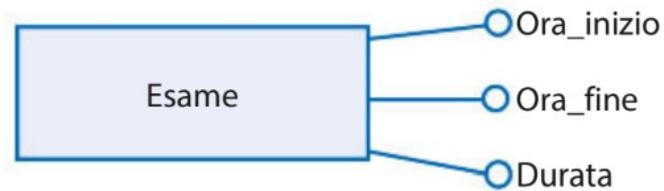
Una **ridondanza** in uno schema ER è una informazione significativa ma derivabile da altre. Di norma le ridondanze andrebbero eliminate, ma occorre tenere conto di vari fattori da analizzare.

Riassumiamo di seguito i differenti casi di casi di ridondanza che si possono presentare.

### Attributi derivabili all'interno della stessa entità

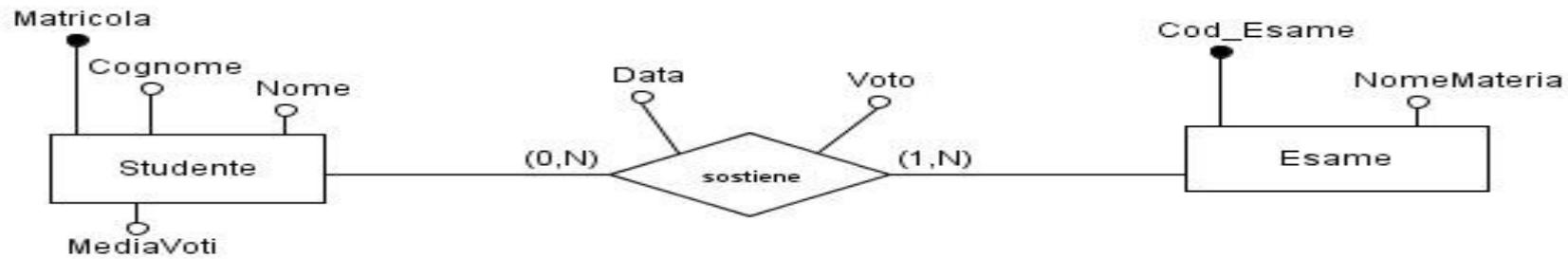
Questo tipo di ridondanza deve essere eliminata rimuovendo gli attributi che possono essere dedotti da altri attraverso dei calcoli.

La durata dell'esame verrà ricavata dalla differenza tra l'ora di inizio e di fine.



## Attributi derivabili da altre entità

L'associazione fra due entità può portare a rendere ridondante qualche attributo; anche in questo caso, la stessa informazione può essere dedotta a partire da altre.



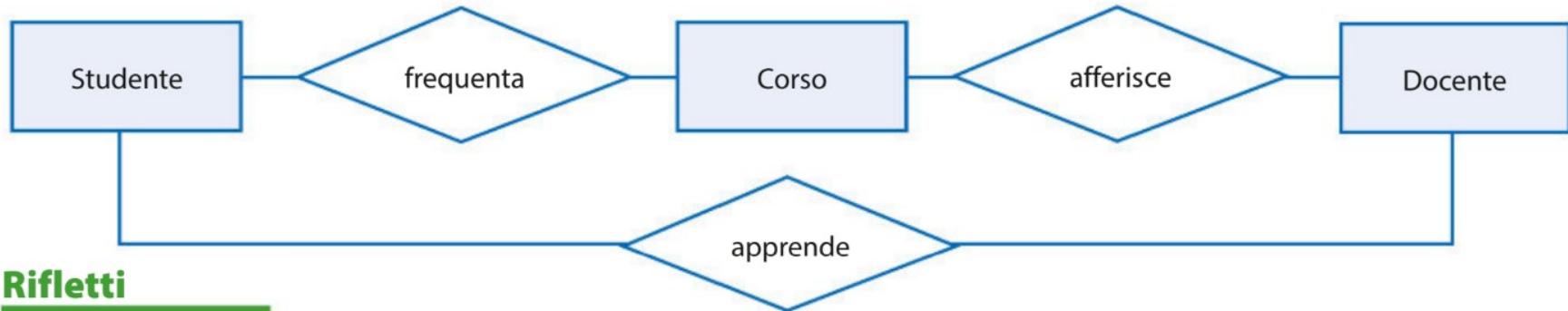
La media degli esami, per ogni studente, sarà calcolata in base a tutti i voti degli esami sostenuti dal singolo studente.

## Attributi derivabili da operazioni di conteggio di occorrenze

Ad esempio, un attributo *Numeri Abitanti* per un'entità *Città* potrebbe essere desunto dal conteggio degli abitanti residenti; si tratta in effetti di una variante del caso precedente, che viene però discussa separatamente in quanto si verifica molto di frequente.

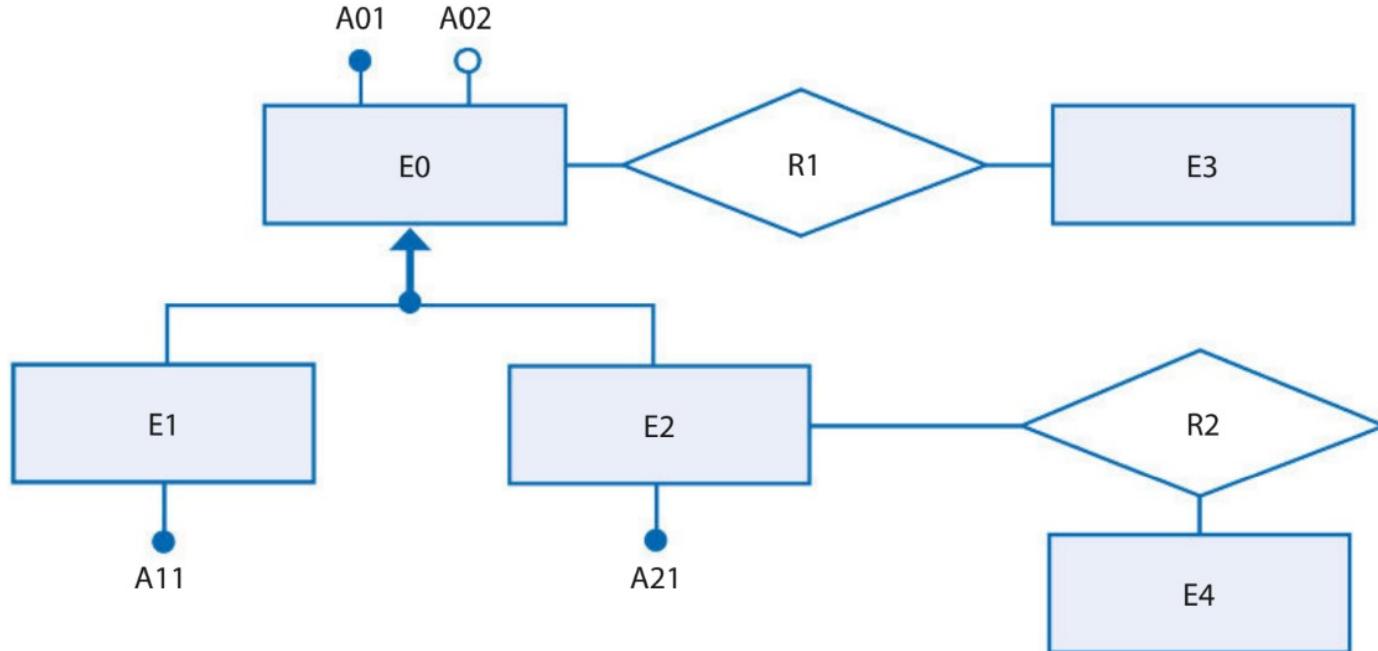
## Ridondanza di associazioni in presenza di cicli

Un uso eccessivo di associazioni può portare ad avere dei *cicli*, cioè dei percorsi alternativi per raggiungere la stessa informazione.



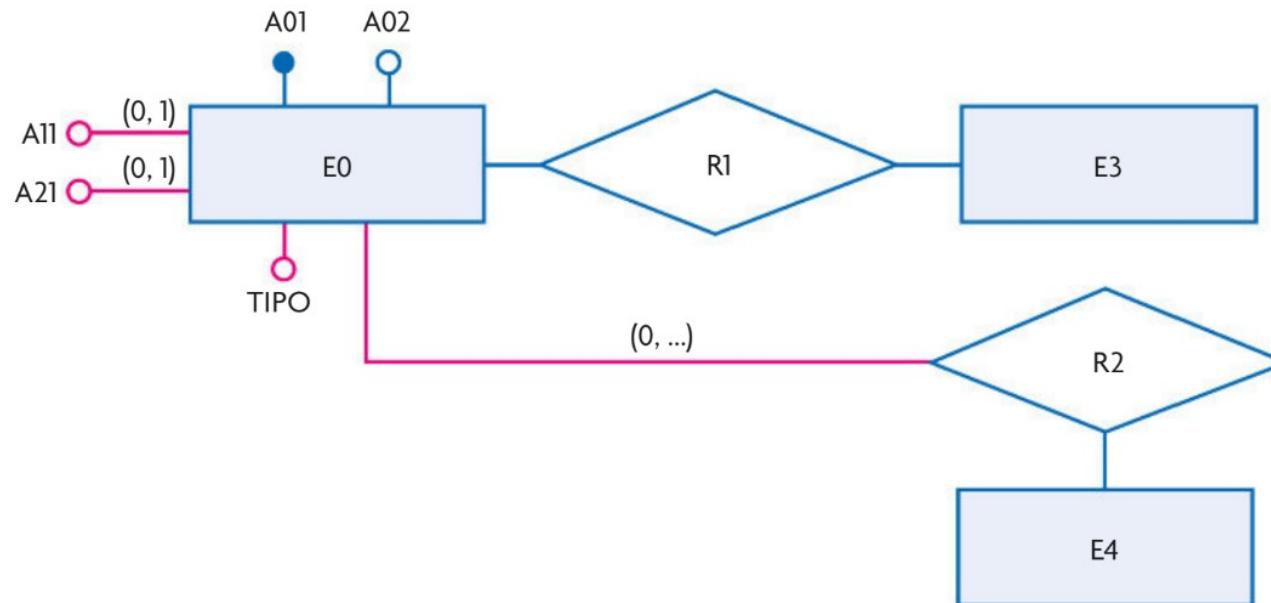
## Eliminazione delle generalizzazioni

I modelli logici tradizionali, tra cui il modello relazionale, non consentono di rappresentare direttamente una generalizzazione, pertanto è necessario trasformare questo costrutto in altri costrutti del modello ER che si basano esclusivamente su entità e associazioni. Per fare ciò abbiamo a disposizione tre alternative possibili. Facendo riferimento al seguente schema ER osserviamo come sia possibile risolvere la generalizzazione attraverso le differenti strategie di ristrutturazione.



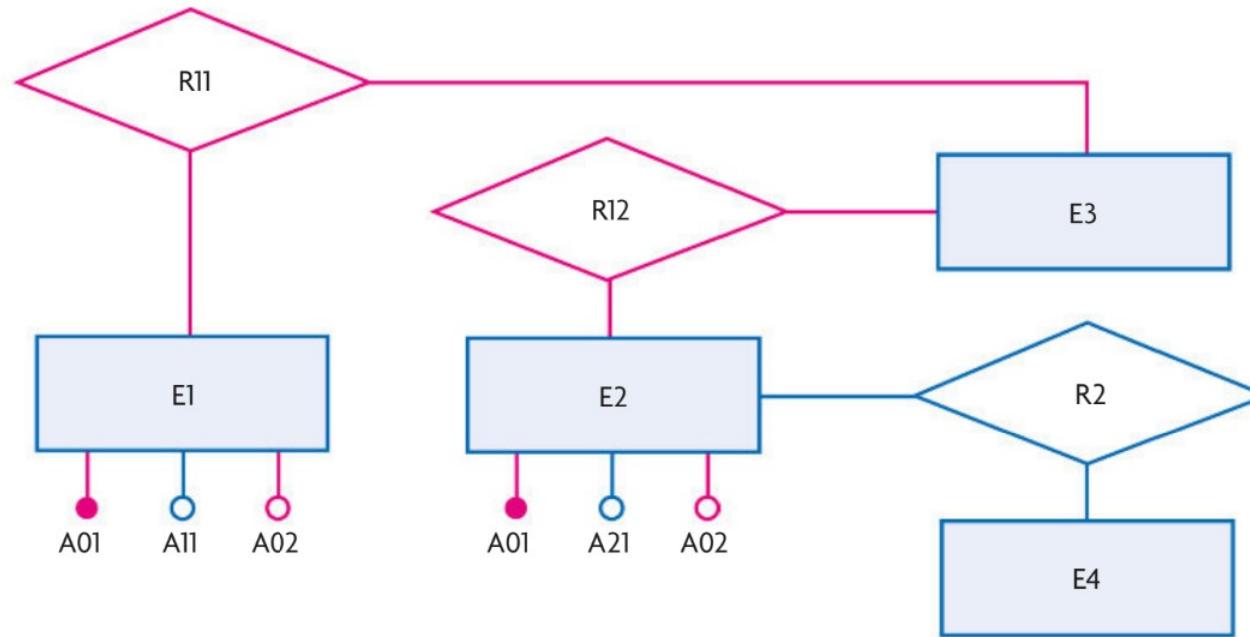
## 1. Accorpamento delle figlie della generalizzazione nel padre.

Le entità E1 e E2 vengono eliminate e i loro attributi, che divengono opzionali, vengono aggiunti a quelli dell'entità padre E0; a tale entità va inoltre aggiunto un attributo che serve a distinguere il tipo, cioè se l'occorrenza di E0 apparteneva a E1 o E2. Ad esempio, una generalizzazione tra l'entità *Persona* e le entità *Uomo* e *Donna* viene ristrutturata aggiungendo all'entità *Persona* l'attributo *Sesso*. Tale soluzione conviene quando gli accessi all'entità padre e alle entità figlie sono contestuali.



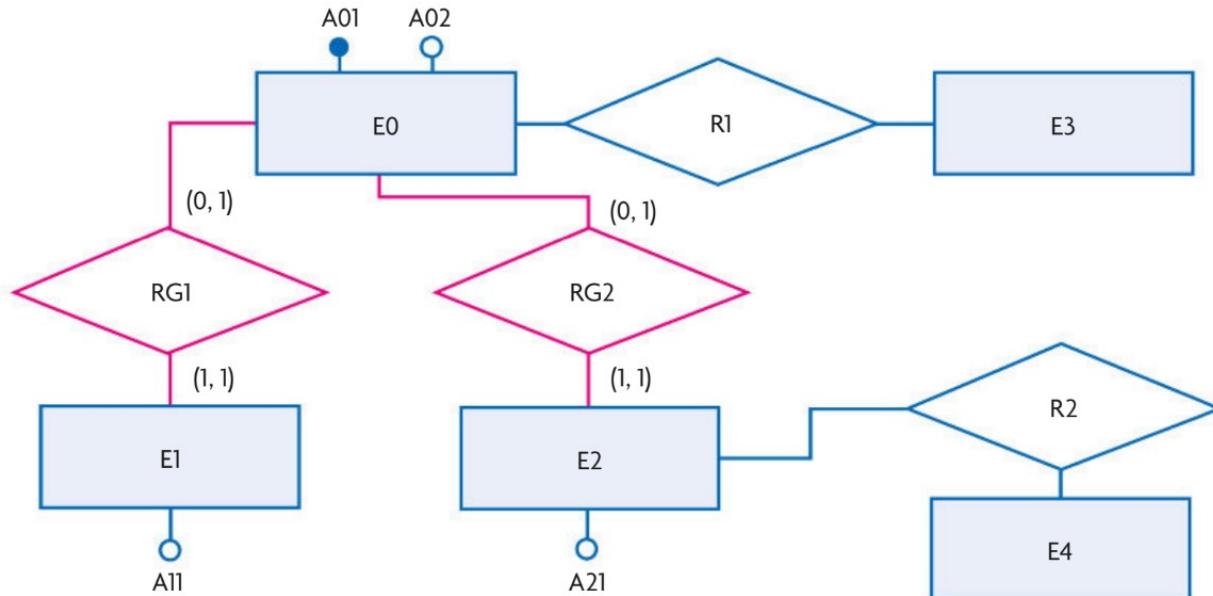
## 2. Accorpamento del padre della generalizzazione nelle figlie.

L'entità padre E0 viene eliminata e i suoi attributi trasferiti a entrambe le figlie; l'associazione R1 viene scissa in due associazioni distinte R11 e R12. Questo tipo di ristrutturazione può essere impiegato quando la generalizzazione è totale, altrimenti le occorrenze di E0 che non sono occorrenze né di E1 né di E2 non sarebbero rappresentate. È conveniente quando ci sono operazioni che si riferiscono solo a occorrenze di E1 oppure di E2, facendo una netta distinzione fra le entità. La scelta garantisce un risparmio di memoria rispetto alla soluzione n. 1 per la mancanza dei valori nulli, e una riduzione degli accessi rispetto alla soluzione successiva (n. 3), in quanto non occorre visitare E0 per raggiungere gli attributi delle figlie.



### 3. Sostituzione della generalizzazione con associazioni.

La generalizzazione si trasforma in due associazioni uno a uno che legano l'entità padre con le figlie, non sono previsti trasferimenti di attributi con le entità E1 e E2 e sono identificate esternamente dall'entità E0. Tale soluzione conviene quando la generalizzazione non è totale e quando gli accessi alle entità figlie sono separati dagli accessi al padre. In questo caso vi è un risparmio di memoria rispetto alla scelta n.1 per l'assenza dei valori nulli, ma un maggior numero di accessi per garantire la consistenza dei dati.

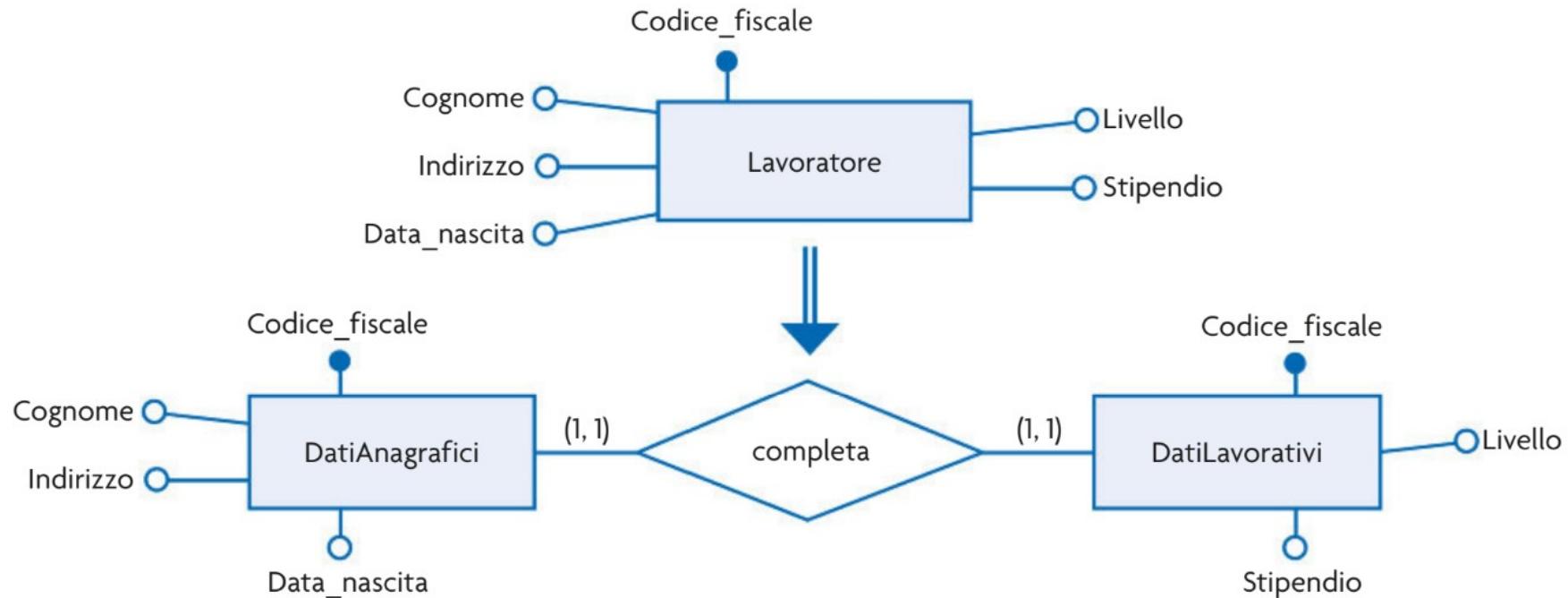


## **Partizionamento/accorpamento di entità e associazioni**

Le entità e le relazioni di uno schema ER possono essere partionate o accorpate con lo scopo di ottenere una maggiore efficienza delle operazioni, ossia di ridurre il numero di accessi mediante separazione di attributi dello stesso concetto o raggruppando attributi di concetti differenti.

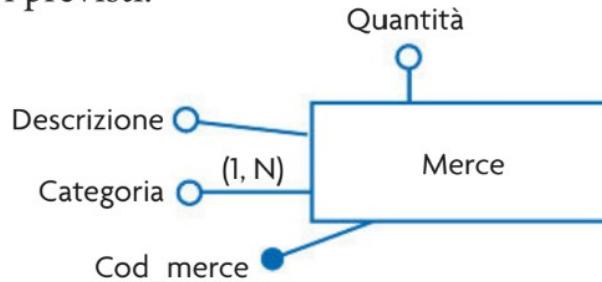
## Partizionamento di entità

Un esempio di partizionamento di entità è mostrato nella figura seguente; l'entità *Lavoratore* viene sostituita da due entità collegate da un'associazione uno a uno che descrivono, rispettivamente, i dati anagrafici dei lavoratori e i dati relativi alla loro retribuzione.

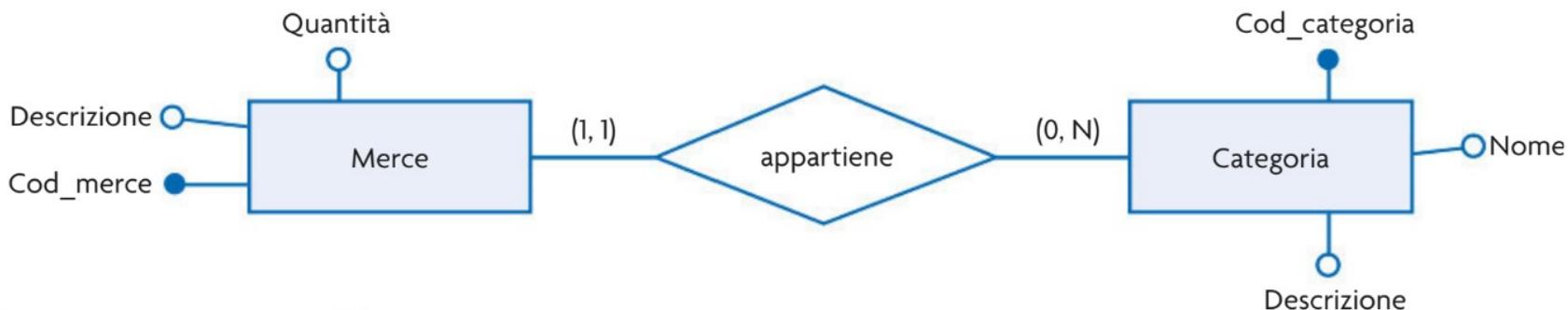


## Eliminazione di attributi multivalore

Come abbiamo già visto nell'unità didattica precedente, in questo esempio notiamo la presenza di un attributo multivalore (*Categoria*), cioè di un attributo che può contenere uno dei particolari valori previsti.

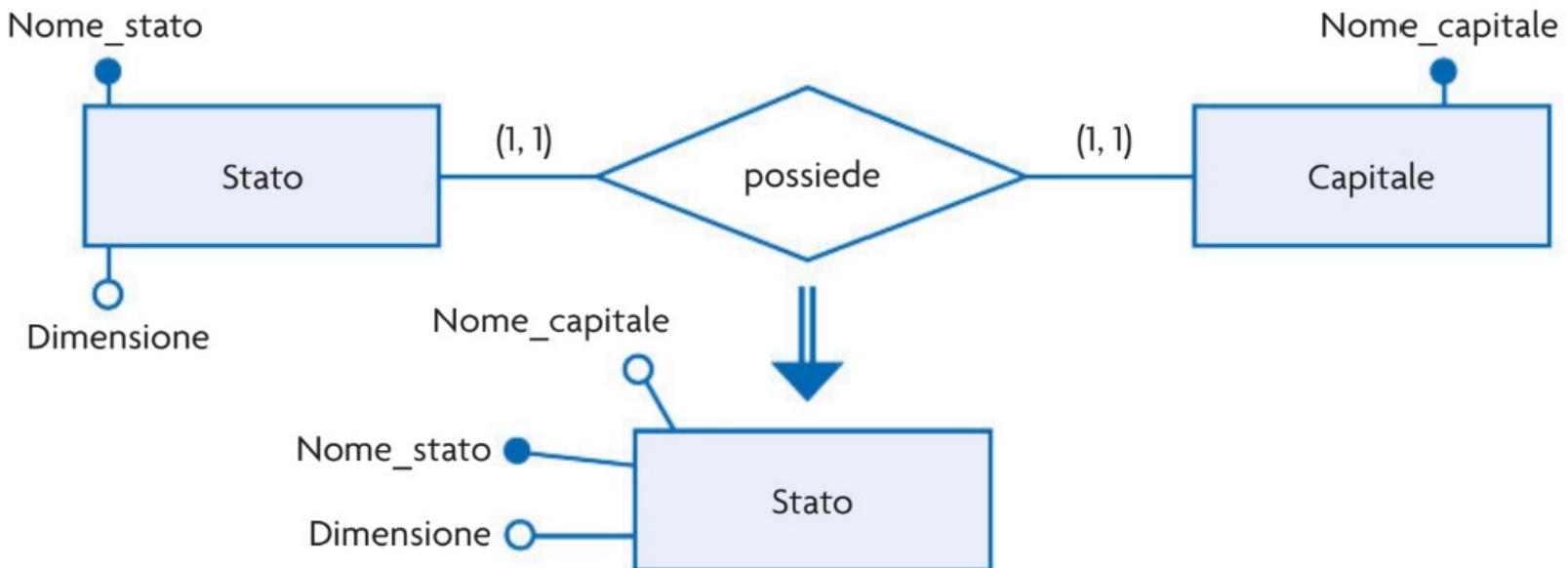


La ristrutturazione si rende necessaria perché il modello relazionale non permette di rappresentare questo tipo di attributi. La ristrutturazione è molto semplice e consiste nel realizzare un'associazione 1 a N tra l'entità originaria e una nuova entità che prende il nome dell'attributo multivalore; eventuali nuovi attributi possono essere inseriti nella nuova entità generata per rendere il tutto più chiaro.



## Accorpamento di entità

L'accorpamento è l'operazione inversa del partizionamento e si effettua in genere su associazioni di tipo uno a uno. Questo tipo di ristrutturazione può essere suggerito dal fatto che le operazioni più frequenti su una determinata entità richiedono sempre i dati relativi all'entità potenzialmente da accoppare.



## Scelta delle chiavi primarie

Ecco alcuni criteri per la decisione di quale attributi utilizzare come chiave primaria:

- non possono essere utilizzati attributi con valori nulli;
- è preferibile utilizzare identificatori composti da uno o pochi attributi; questo garantisce una minore complessità nella realizzazione dei legami fra le varie relazioni.

Se nessuno degli attributi soddisfa i requisiti di chiave, viene introdotto un ulteriore attributo che conterrà dei codici generati appositamente (*chiave artificiale*) per identificare le occorrenze delle entità. È consigliabile in questa fase prendere in esame quegli identificatori che, pur non essendo considerati primari, rivestono una particolare importanza nell'accesso ai dati.

## 3 Le relazioni

Il passo successivo nella progettazione di una base di dati è la progettazione logica relazionale, che consiste nel trasformare la rappresentazione ancora astratta e indipendente del diagramma ER in una rappresentazione più efficiente (ma comunque indipendente da un particolare DBMS) detta schema logico relazionale.

---

La **progettazione logica relazionale** consiste quindi 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.

---

Il modello relazionale dei dati, introdotto fin dal 1970 da E.F. Codd, prevede un unico e semplice meccanismo di strutturazione dei dati, basato sul concetto matematico di relazione fra insiemi.

---

Una **relazione** R su una sequenza di insiemi D<sub>1</sub>, D<sub>2</sub>, ..., D<sub>n</sub> (non necessariamente distinti) è un sottoinsieme finito del prodotto cartesiano D<sub>1</sub> × D<sub>2</sub> ... × D<sub>n</sub>, che possiamo esprimere con:

$$R \subseteq D_1 \times D_2 \dots \times D_n$$

---

dove:

- gli insiemi D sono detti **domini** della relazione e ciascuno di essi si riferisce a un tipo di dato elementare (carattere, intero, reale, booleano e così via). A ogni dominio è associato un nome, detto **attributo**, che lo distingue dagli altri all'interno della relazione;
- n ( $n \geq 1$ ) è detto **grado** della relazione ed è indicato con Grado(R).

---

Gli elementi di R sono detti **n-uple** o **tuple** e vengono indicati con:

$$(d_1, d_2, \dots, d_n)$$

dove  $d_1 \in D_1, d_2 \in D_2, \dots, d_n \in D_n$ .

---

## Rifletti

Coerentemente con la definizione di insieme, una relazione non può contenere tuple uguali.

---

Chiameremo **istanza** di una relazione R l'insieme delle sue tuple in un determinato istante di tempo. L'istanza rappresenta il significato estensionale della relazione.

---

Il numero m di tuple presenti in un dato istante in una relazione R viene detto **cardinalità** (corrente) della relazione e indicato con  $\text{Card}(R)$ .

---

---

Chiameremo **schema di una relazione** il suo nome e la lista dei suoi attributi.

---

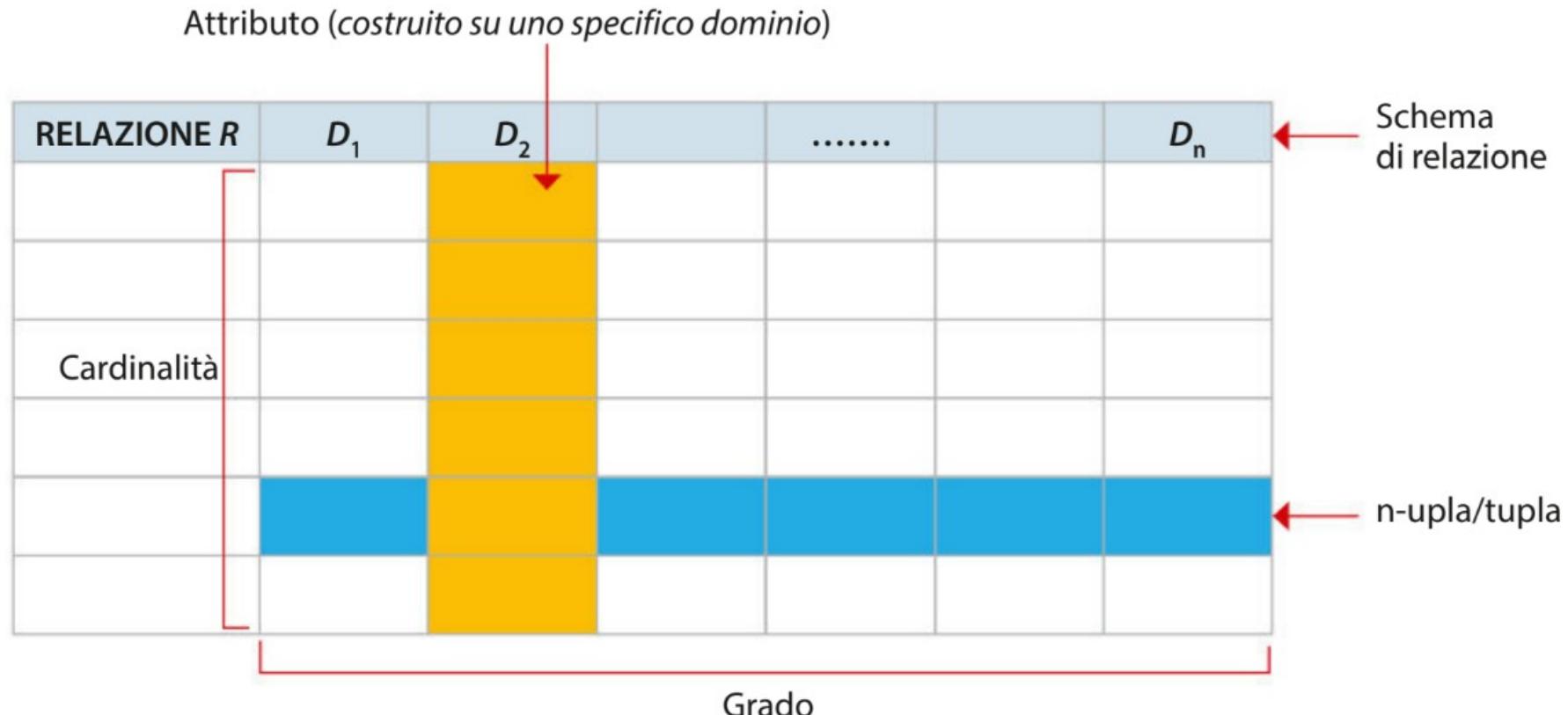
Lo schema di una relazione indica il significato **intensionale** di quest'ultima. Utilizzeremo la seguente sintassi per rappresentarlo:

NOMERELAZIONE(Attributo1, Attributo2, ..., AttributoN)

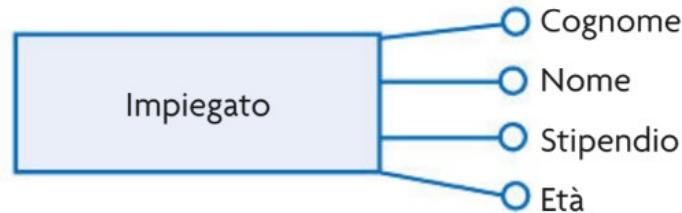
Per convenzione scriveremo i nomi delle relazioni in maiuscolo e gli attributi con la sola iniziale maiuscola.

PERSONA(Cognome, Nome, Età, Sesso)

Nella figura successiva sono schematizzati i concetti appena introdotti:



Prendiamo in esame la seguente entità:



e rappresentiamola graficamente secondo quanto appena visto.

IMPIEGATO(Cognome, Nome, Stipendio, Età) ← Schema di relazione

IMPIEGATO	Cognome	Nome	Stipendio	Età
	Rossi	Mario	1200	40
	Grimaldi	Gianni	1500	45
Cardinalità = 5	Brezza	Andrea	2000	58
	Bruni	Anna	1850	32
	Frasca	Aldo	1500	40

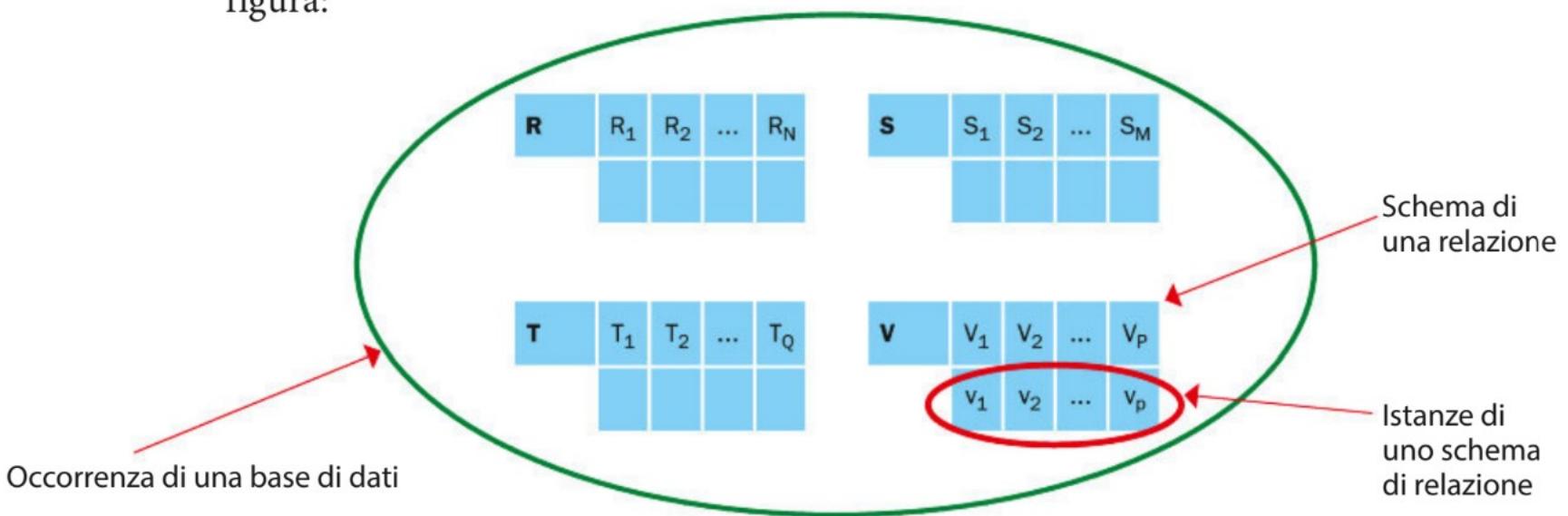
Cardinalità = 5      Grado = 4      Istanza di relazione formata da 5 tuple

---

Si definisce **schema di una base di dati** relazionale l'insieme di tutti gli schemi di relazione.  
Si definisce **occorrenza** di una base di dati relazionale l'insieme delle istanze degli schemi di relazione.

---

È quindi possibile rappresentare la base di dati con un insieme di schemi di relazione. Se consideriamo una base di dati in un determinato momento, avremo una particolare occorrenza di quella base di dati, così come rappresentato nello schema della seguente figura:



## 4 Il mapping delle entità e degli attributi

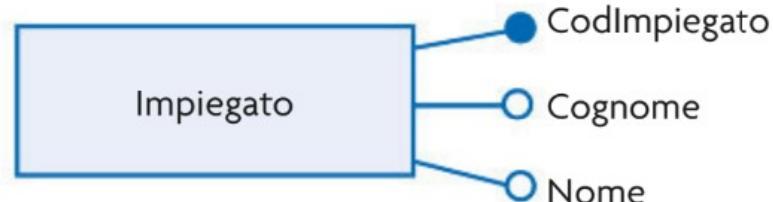
Per modellare i vari aspetti della realtà, il modello relazionale mette a disposizione del progettista solo le relazioni. Partendo dal diagramma ER, il progettista deve quindi effettuare un “mapping” delle entità e delle associazioni trasformandole in relazioni del modello relazionale, applicando alcune semplici regole di derivazione. Esaminiamo in dettaglio queste regole.

Un’entità E con attributi elementari  $A_1, A_2, \dots, A_n$  è rappresentata attraverso una relazione:

$$R(A_1, A_2, \dots, A_n)$$

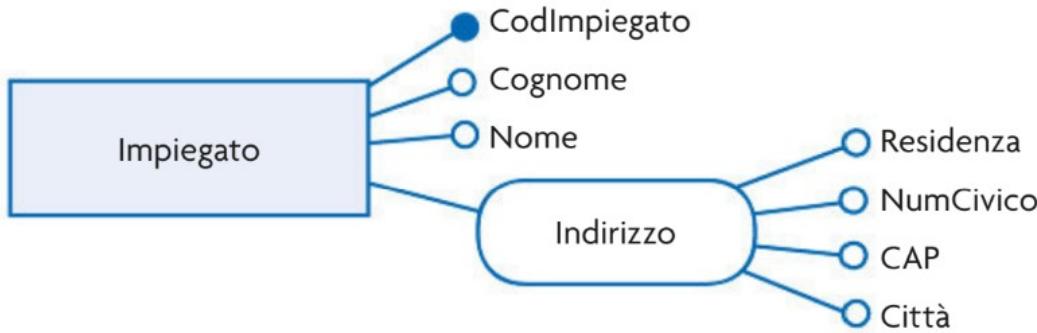
dove:

- 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.



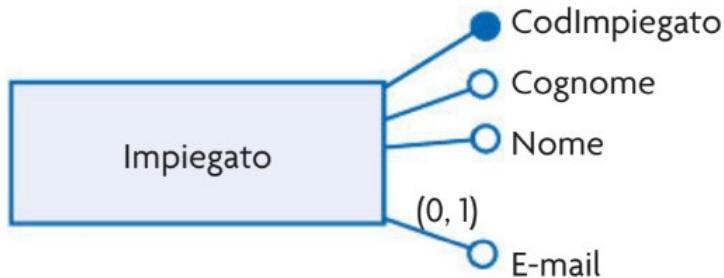
IMPIEGATO(CodImpiegato, Cognome, Nome)

- Eventuali attributi composti devono essere sostituiti con gli attributi componenti.



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

- Eventuali attributi opzionali vengono indicati facendoli seguire da un asterisco.



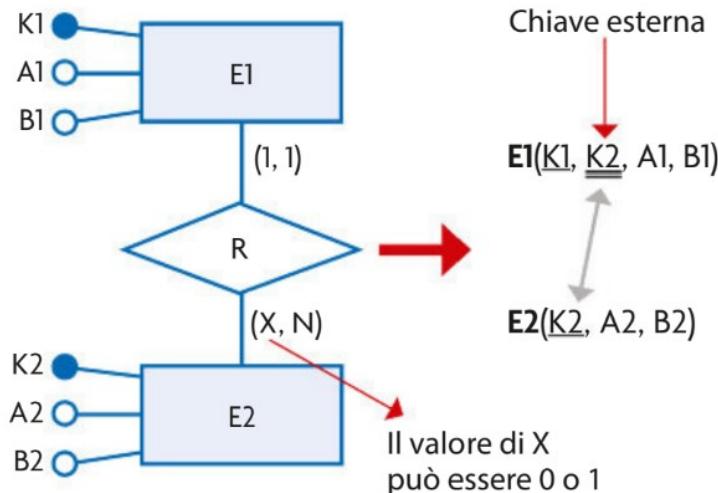
**IMPIEGATO(CodImpiegato, Cognome, Nome, **E-mail**\*)**

# 5 Rappresentazione delle associazioni

Vediamo ora come si rappresentano formalmente i vari tipi di associazioni.

## Mapping delle associazioni binarie 1:N

Un'associazione R di tipo 1:N tra due entità E<sub>1</sub> ed E<sub>2</sub>, a cui corrispondono le rispettive relazioni E<sub>1</sub> ed E<sub>2</sub>, è “mappata” aggiungendo alla relazione E<sub>1</sub> gli attributi chiave primaria E<sub>2</sub>. In altri termini, si aggiungono gli attributi chiave della relazione che partecipa con cardinalità massima N alla relazione che partecipa con cardinalità massima 1. Gli attributi chiave primaria di E<sub>2</sub> presenti in E<sub>1</sub> costituiscono una cosiddetta **chiave esterna** per la relazione E<sub>1</sub>, che viene rappresentata con una **doppia sottolineatura**.



### Rifletti

Il valore di una chiave esterna in una tupla funge da puntatore logico alla tupla dell'altra relazione con la quale è in associazione.

### Rifletti

Sebbene nella nuova relazione sia possibile utilizzare nomi qualsiasi per gli attributi chiavi esterne, utilizzeremo quando possibile lo stesso nome dell'attributo relativo alla chiave primaria.

# OSSERVA COME SI FA

- Il seguente schema ER descrive l'associazione *possiede* che esiste tra le persone e le auto possedute.



Tra l'entità *Persona* e l'entità *Auto* esiste un'associazione *possiede* di tipo uno a molti (**1:N**). Infatti, la cardinalità massima dell'entità *Persona* rispetto a *possiede* (ossia la diretta) è **N** mentre la cardinalità massima di *Auto* rispetto a *possiede* (ossia l'inversa) è **1**. Ciò significa che una persona può possedere più automobili, ma un'automobile ha un solo proprietario. L'associazione diretta è parziale poiché non tutte le persone posseggono un'automobile, mentre l'inversa è totale poiché ogni automobile deve avere un proprietario.

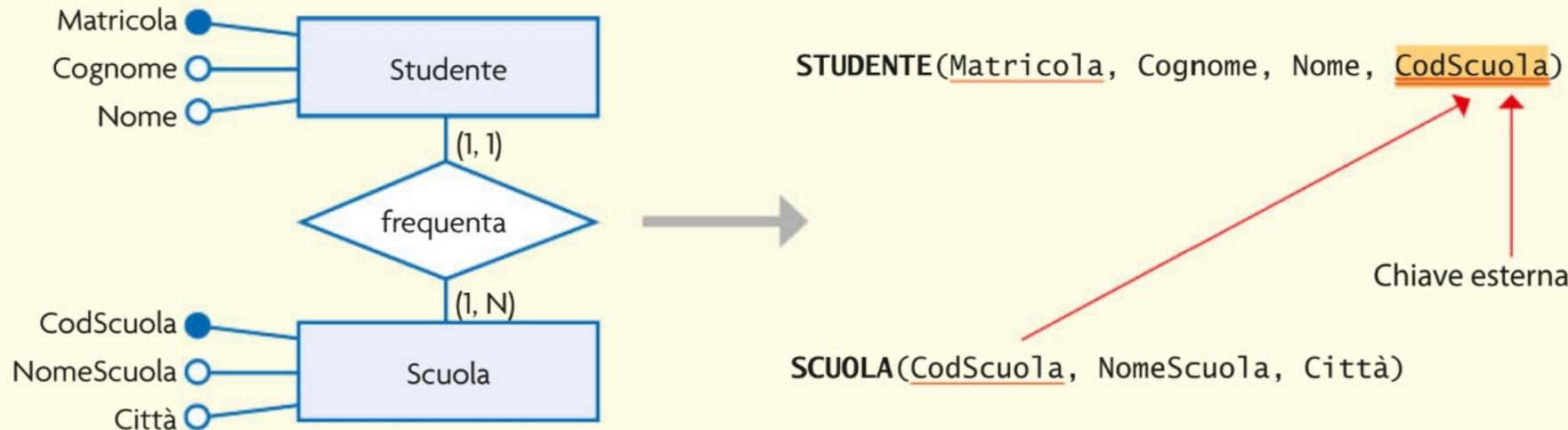
Le relazioni che si ottengono applicando le regole di derivazione sono le seguenti:

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

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

# OSSERVA COME SI FA

## 2. Consideriamo il seguente diagramma ER:



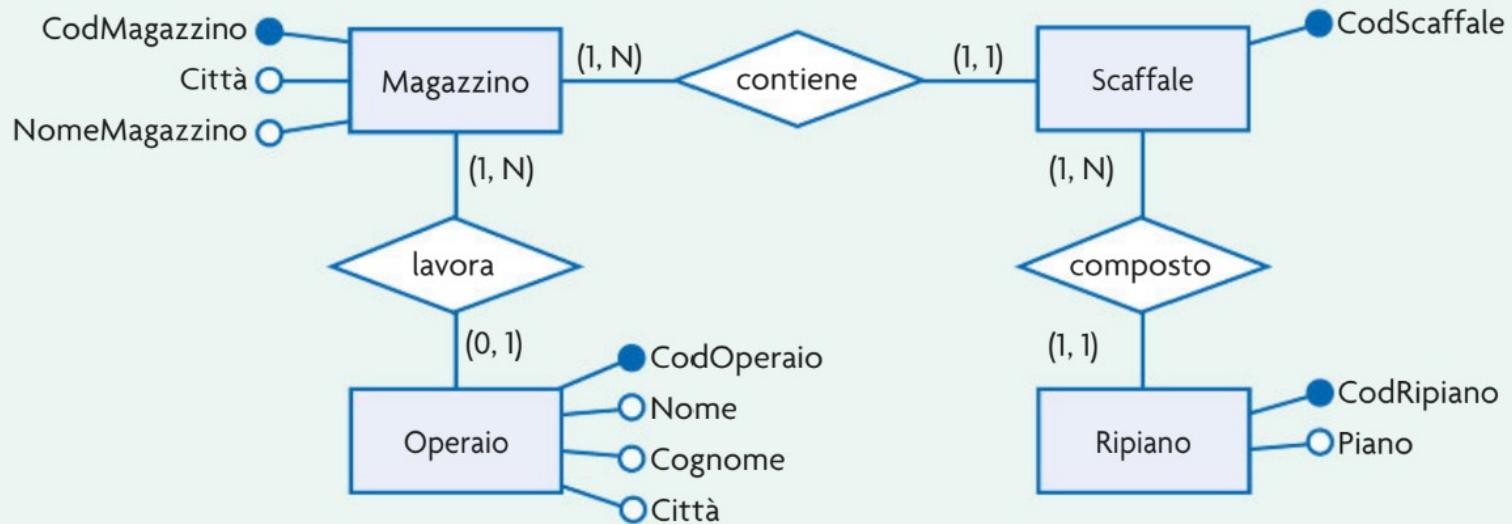
Nella relazione *Studente* l'attributo *CodScuola* è una chiave esterna che viene utilizzata per rappresentare l'associazione 1:N esistente tra *Scuola* e *Studente*.

# ORA TOCCA A TE

- Effettua il mapping del seguente schema ER generando le varie relazioni.



2. Effettua il mapping del seguente schema ER generando le varie relazioni. Sull'entità Scaffale sono presenti due specificatori di attributo semplice che potresti inserire in base alla tua analisi (nel caso in cui non ritieni opportuno corredare l'entità di questi ulteriori attributi puoi ignorarli).



## Mapping delle associazioni binarie 1:1

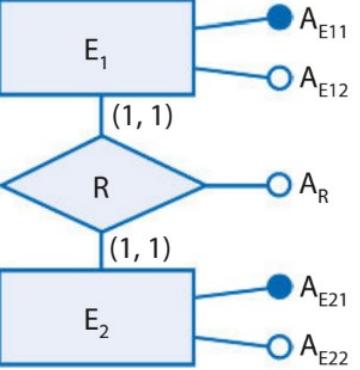
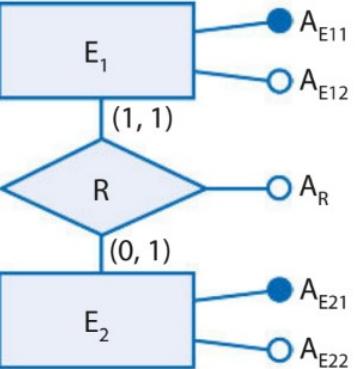
Le associazioni binarie 1:1 sono un caso particolare delle associazioni 1:N e, per esse, esistono varie possibilità di mapping.

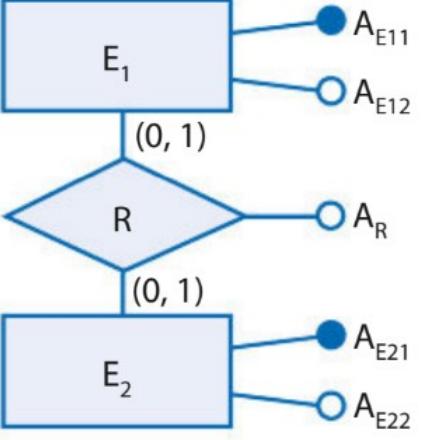
In particolare:

- se entrambe le associazioni sono totali, si mappano i due tipi di entità in associazione 1:1 in un'unica relazione avente tutti gli attributi dell'uno e dell'altro; è anche consentito conservare le due entità separate aggiungendo la chiave esterna a una qualunque delle due relazioni;
- se esistono associazioni con partecipazione facoltativa, si opta per la realizzazione di due relazioni distinte, aggiungendo la chiave esterna alla relazione rispetto a cui l'associazione è totale, perché potrebbero esserci valori nulli per l'entità con partecipazione opzionale;
- se entrambe le entità hanno partecipazione facoltativa, si prevedono solo relazioni separate e mai la relazione unica.

Se l'associazione ha degli attributi, questi vanno aggiunti alla relazione a cui si aggiunge la chiave esterna.

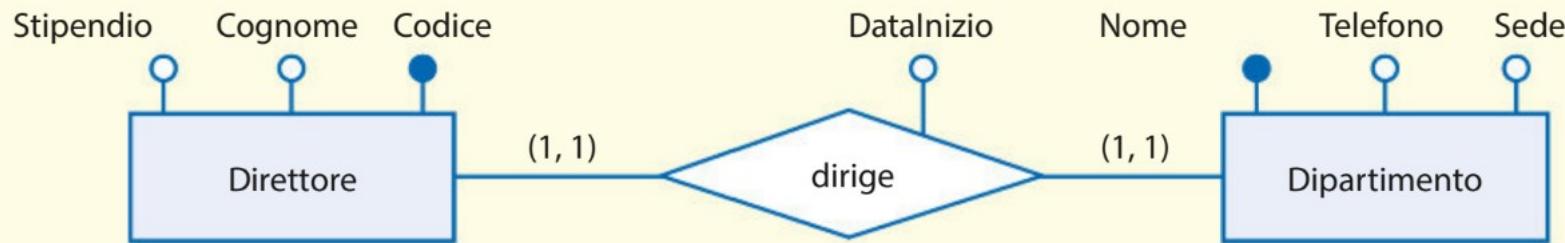
Graficamente la situazione è la seguente:

Tipo di associazione	Schema concettuale	Schema logico
Associazione uno a uno con partecipazione obbligatoria per entrambe le entità	 <pre> graph TD     E1[E1] --&gt; A_E11((A_E11))     E1 --&gt; A_E12((A_E12))     E2[E2] --&gt; A_E21((A_E21))     E2 --&gt; A_E22((A_E22))     R{R} -- "1, 1" --&gt; A_R((A_R))     E1 --&gt; R     E2 --&gt; R   </pre>	$E_1(A_{E11}, A_{E12}, \underline{\underline{A_{E21}}}, A_R)$ $E_2(\underline{\underline{A_{E21}}}, A_{E22})$ oppure $E_1(A_{E11}, A_{E12})$ $E_2(\underline{\underline{A_{E21}}}, A_{E22}, \underline{\underline{A_{E11}}}, A_R)$
Associazione uno a uno con partecipazione opzionale per una entità	 <pre> graph TD     E1[E1] --&gt; A_E11((A_E11))     E1 --&gt; A_E12((A_E12))     E2[E2] --&gt; A_E21((A_E21))     E2 --&gt; A_E22((A_E22))     R{R} -- "1, 1" --&gt; A_R((A_R))     E1 --&gt; R     E2 -- "(0, 1)" --&gt; R   </pre>	$E_1(A_{E11}, A_{E12}, \underline{\underline{A_{E21}}}, A_R)$ $E_2(\underline{\underline{A_{E21}}}, A_{E22})$

Tipo di associazione	Schema concettuale	Schema logico
Associazione uno a uno con partecipazione opzionale per entrambe le entità	 <pre> classDiagram     class E1 {         A_E11         A_E12     }     class R {         A_R     }     class E2 {         A_E21         A_E22     }     E1 "0..1" -- "0..1" R   </pre>	$E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22}, \underline{A_{E11}}, A_R)$ oppure $E_1(A_{E11}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ oppure $E_1(A_{E11}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, A_{E21}, A_R)$ <p>All'interno della relazione R è anche possibile scegliere <math>A_{E21}</math> come chiave primaria piuttosto che <math>A_{E11}</math>.</p>

# OSSERVA COME SI FA

- Per le associazioni uno a uno ci sono, in genere, diverse possibilità di traduzione. Cominciamo a vedere le associazioni uno a uno con partecipazioni obbligatorie per entrambe le entità, come quella rappresentata nel seguente schema:



Abbiamo due possibilità simmetriche e ugualmente valide per effettuare il mapping:

- DIRETTORE(Codice, Cognome, Stipendio, Nome, Datalnizio)**

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

oppure

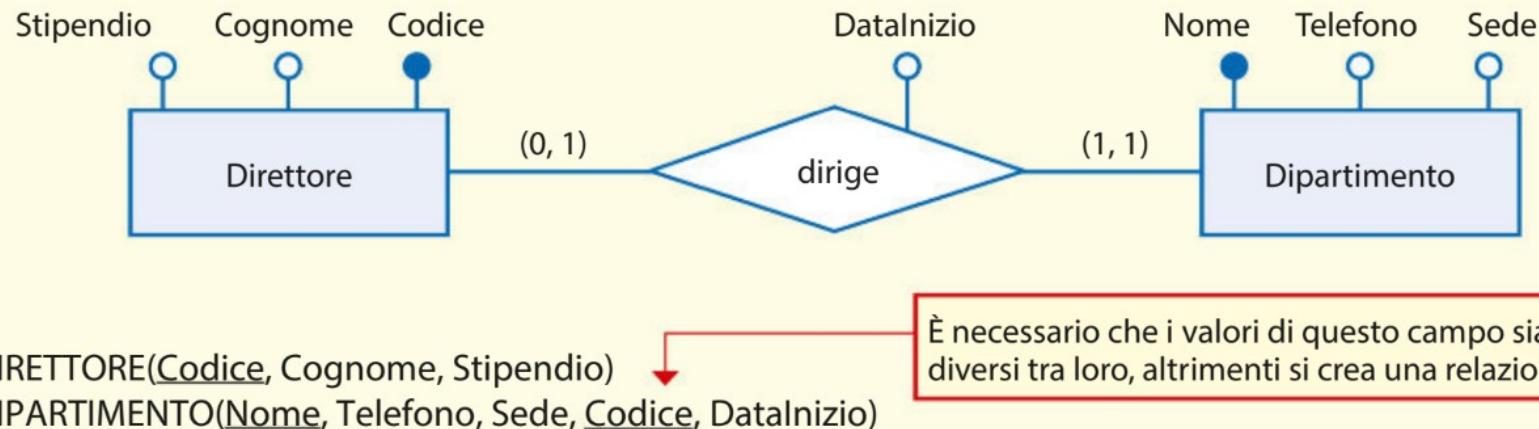
- DIRETTORE(Codice, Cognome, Stipendio)**

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

È necessario che i valori di questo campo siano tutti diversi tra loro, altrimenti si crea una relazione 1:N

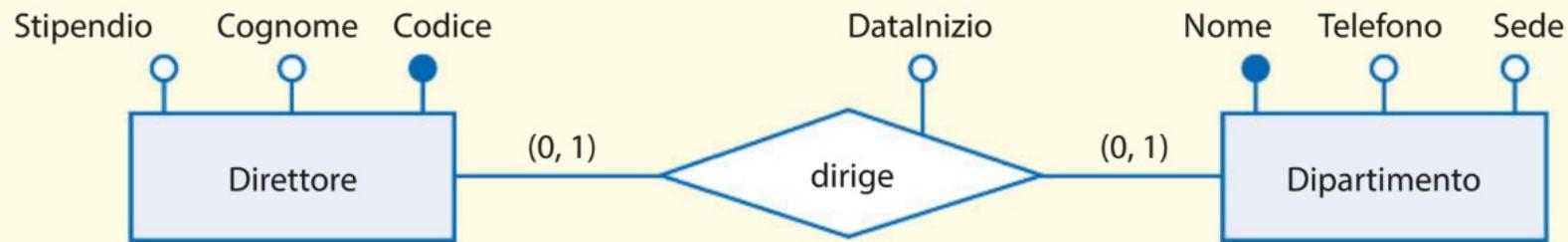
È necessario che i valori di questo campo siano tutti diversi tra loro, altrimenti si crea una relazione 1:N

Consideriamo ora il caso di associazione uno a uno con partecipazione opzionale per una sola entità, come quella nel seguente schema:



Questa alternativa è preferibile rispetto a quella in cui l'associazione viene rappresentata nella relazione DIRETTORE mediante il nome del dipartimento diretto perché avremmo, per questo attributo, possibili valori nulli.

Consideriamo infine il caso in cui entrambe le entità hanno partecipazione opzionale, come nel caso in cui possono esistere dipartimenti senza direttori (e quindi la cardinalità dell'entità DIPARTIMENTO diventa  $(0, 1)$ ).



In questo caso esiste una ulteriore possibilità che prevede tre relazioni separate:

DIRETTORE(Codice, Cognome, Stipendio)

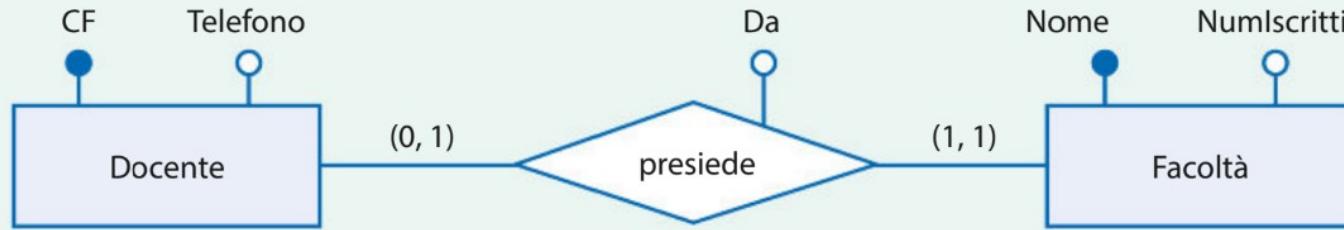
**DIPARTIMENTO**(Nome, Telefono, Sede)

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

All'interno della relazione DIRIGE è anche possibile scegliere *Codice* come chiave primaria piuttosto che *Nome*.

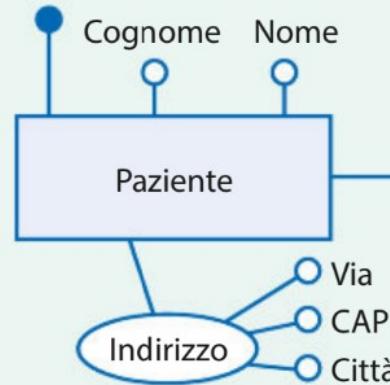
## ORA TOCCA A TE

- Effettua il mapping del seguente schema ER generando le varie relazioni.

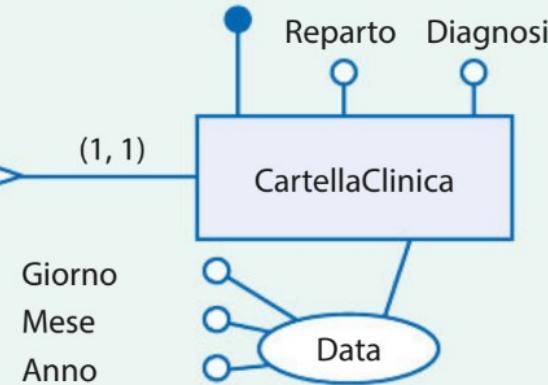


**2. Effettua il mapping del seguente schema ER generando le varie relazioni.**

CodPaziente

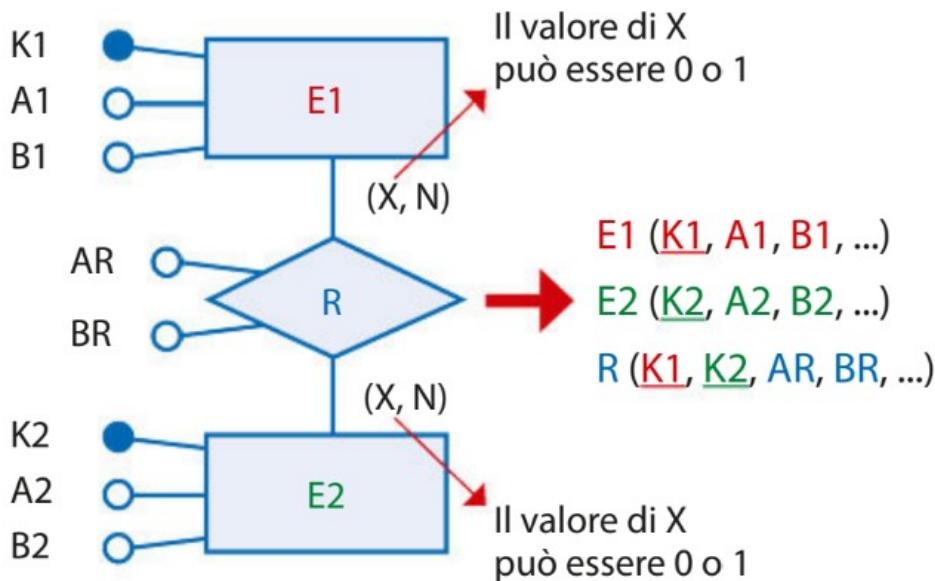


NumCartella



## Rappresentazione delle associazioni N:N

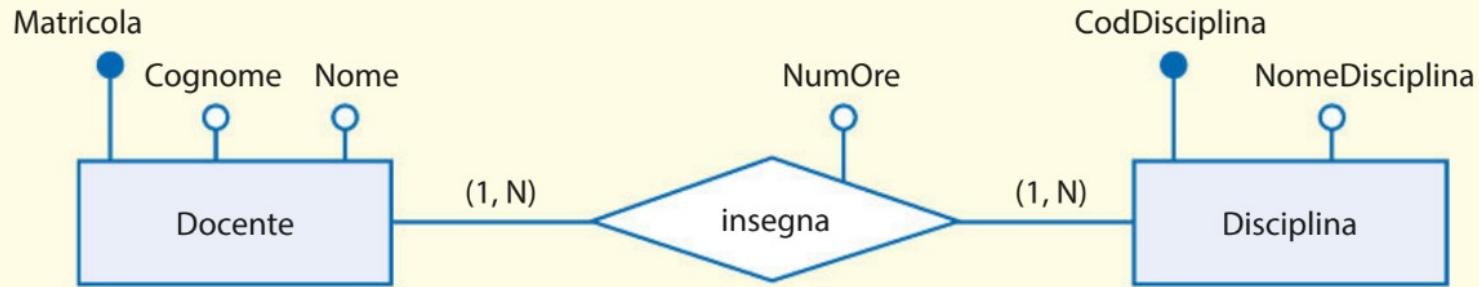
Un'associazione R di tipo N:N tra due entità E<sub>1</sub> ed E<sub>2</sub>, a cui corrispondono le relazioni E<sub>1</sub> ed E<sub>2</sub>, è mappata creando per ogni entità una relazione R avente almeno gli attributi chiave primaria di E<sub>1</sub> e gli attributi chiave primaria di E<sub>2</sub> (quindi R ha come minimo due attributi) che insieme formano la chiave primaria della relazione R.



Come si evince dalla precedente figura, il mapping è effettuato sempre nello stesso modo, prescindendo dal tipo di partecipazione delle entità.

## OSSERVA COME SI FA

1. Consideriamo il seguente schema concettuale caratterizzato da partecipazione obbligatoria di entrambe le entità:



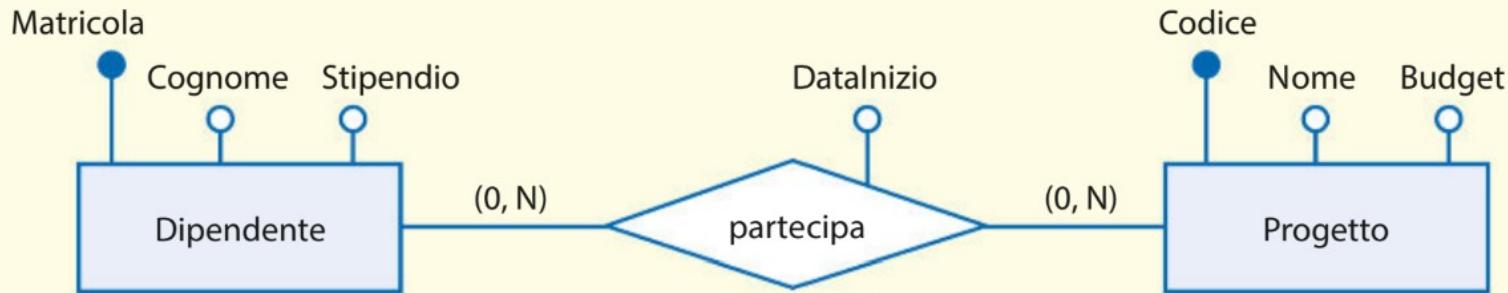
Applicando le regole di derivazione si ottiene il seguente schema relazionale:

DOCENTE(Matricola, Cognome, Nome)

DISCIPLINA(CodDisciplina, NomeDisciplina)

INSEGNA(CodDisciplina, Matricola, NumOre)

2. Consideriamo il seguente schema concettuale caratterizzato da partecipazione opzionale di entrambe le entità:



Applicando le regole di derivazione si ottiene il seguente schema relazionale:

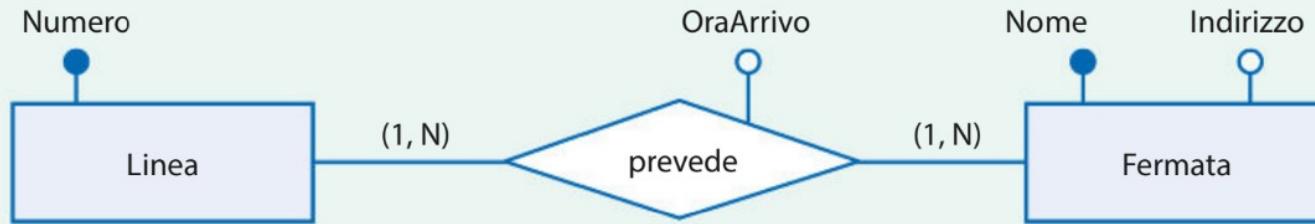
**DIPENDENTE**(Matricola, Cognome, Stipendio)

**PROGETTO**(Codice, Nome, Budget)

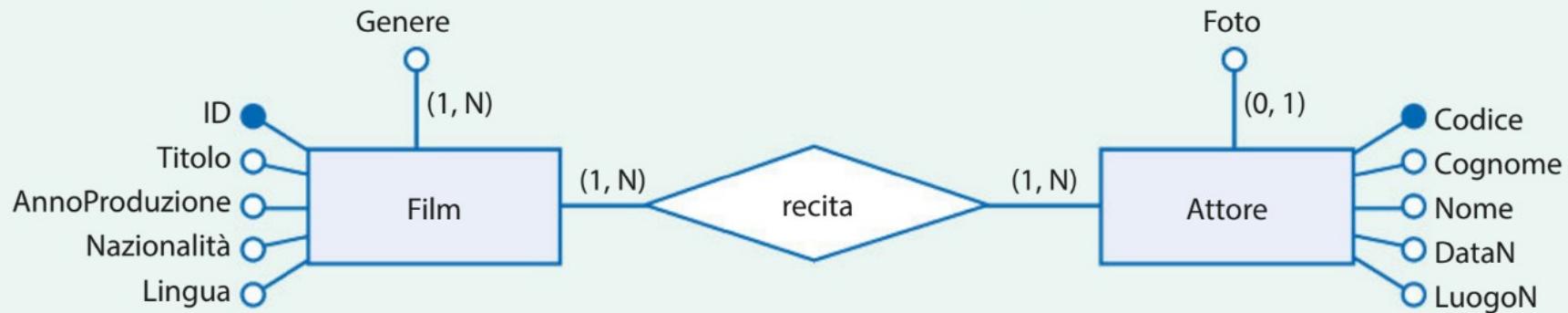
**PARTECIPA**(Matricola, Codice, Datalnizio)

## ORA TOCCA A TE

1. Effettua il mapping del seguente schema ER generando le varie relazioni.



2. Ristruttura il seguente schema ER e, di seguito, effettua il mapping generando le varie relazioni.



## Mapping delle associazioni ricorsive

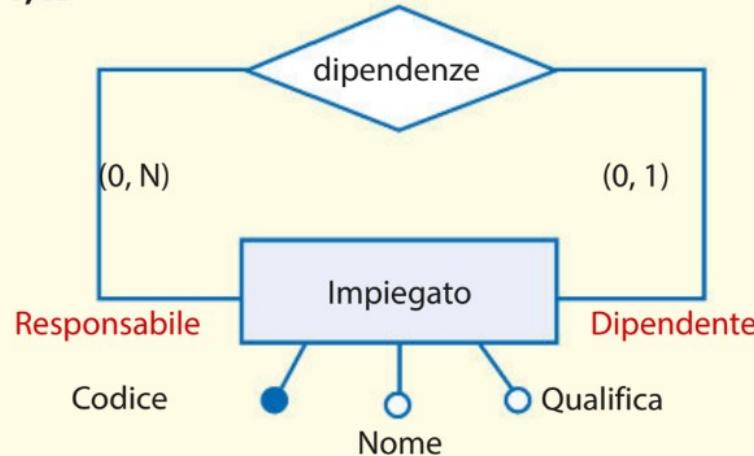
Nel caso di associazioni **ricorsive** (o **unarie**), ossia quelle in cui l'entità di partenza coincide con quella di arrivo, si seguono le seguenti regole di derivazione:

- in caso di associazione ricorsiva **1,N** (o **N,1**) verrà convertita in relazione la sola entità presente, con l'aggiunta nel suo schema di un attributo con funzione di chiave esterna corrispondente alla chiave primaria della relazione stessa;
- in caso di associazione unaria o ricorsiva **1,1** verrà convertita in relazione la sola entità presente, con l'aggiunta nel suo schema di un attributo con funzione di chiave esterna corrispondente alla chiave primaria della relazione stessa;
- in caso di associazione unaria **N,N** l'unica entità viene convertita in relazione aggiungendo una nuova tabella (o relazione) contenente sempre due attributi con funzione di chiave\_esterna corrispondenti questa volta alla stessa chiave primaria della relazione su cui agisce l'associazione; anche in questo caso la nuova relazione potrebbe contenere anche altri attributi propri.

# OSSERVA COME SI FA

1. Consideriamo i seguenti schemi ER che riportano ciascuno una singola tipologia di associazione:

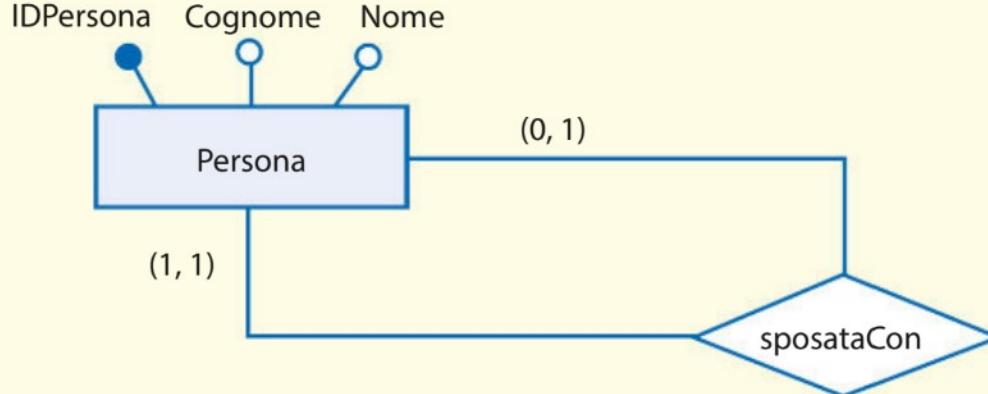
## Associazione ricorsiva di tipo 1, N



Applicando le regole di derivazione si ottiene il seguente schema relazionale:

IMPIEGATO(Codice, Nome, Qualifica, CodiceResponsabile)

### Associazione ricorsiva di tipo 1, 1

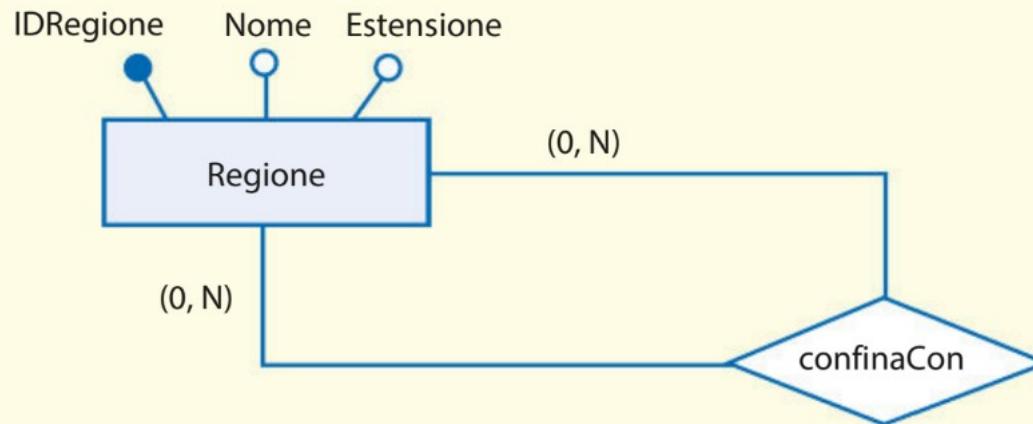


Applicando le regole di derivazione si ottiene il seguente schema relazionale:

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

Il campo **IDPersonaSposata** funge da chiave esterna e deve rispettare il vincolo dell'unicità.

## Associazione ricorsiva di tipo N,N



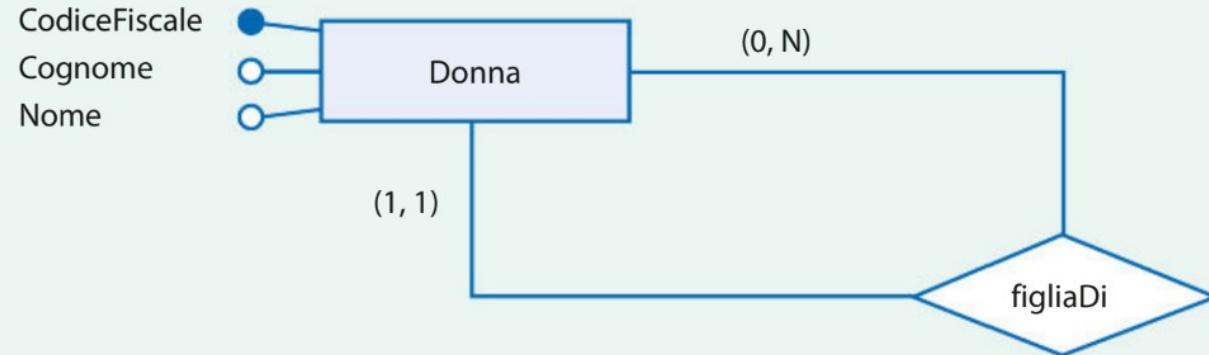
Applicando le regole di derivazione si ottiene il seguente schema relazionale:

REGIONE(IDRegione, Nome, Estensione)

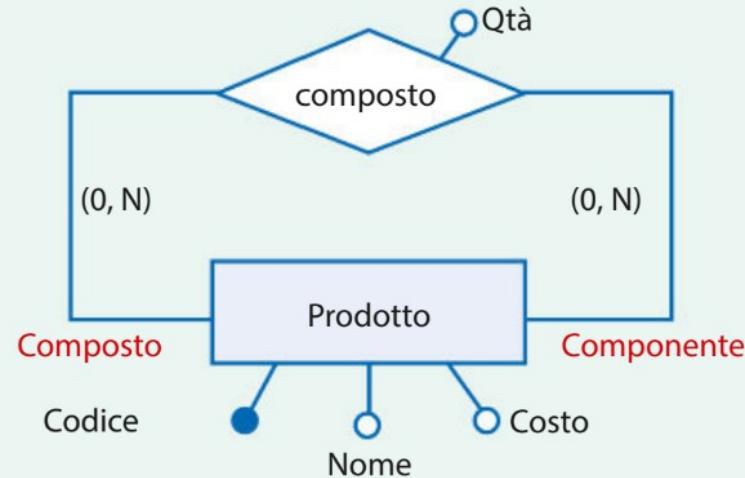
CONFINACON(IDRegione1, IDRegione2)

## ORA TOCCA A TE

- Effettua il mapping del seguente schema ER generando le varie relazioni.



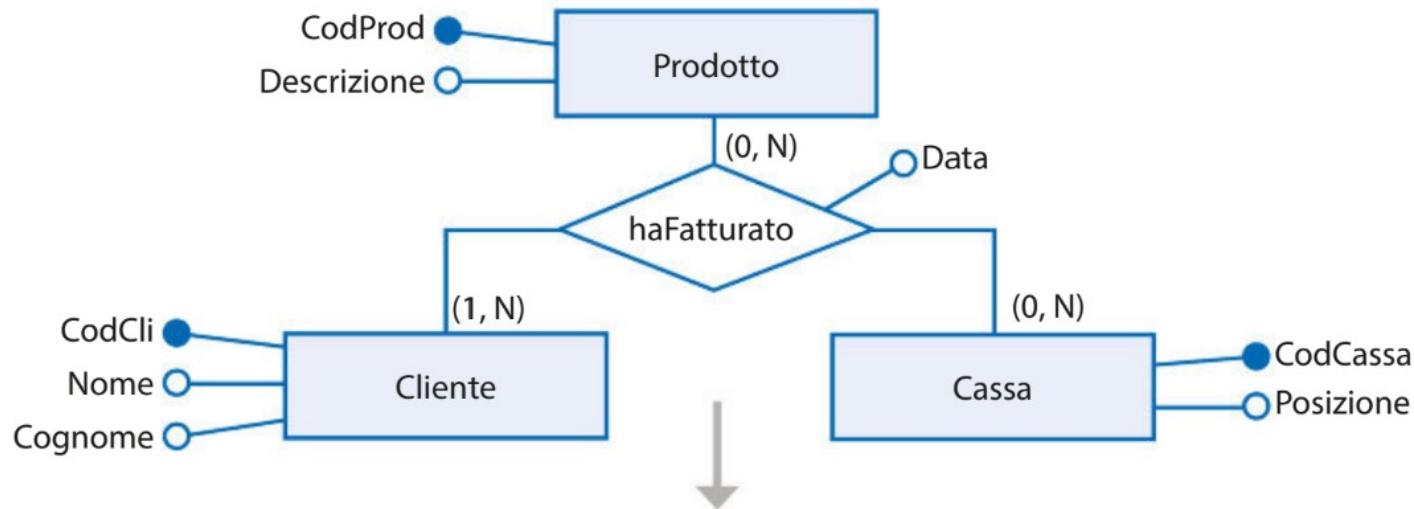
2. Effettua il mapping del seguente schema ER generando le varie relazioni.



## Mapping di associazioni n-arie

Alle associazioni n-arie (ossia quelle con più di due entità) si applicano le stesse regole di traduzione viste sinora per le associazioni binarie: dipende, quindi, dal tipo di associazione.

Consideriamo il seguente diagramma ER relativo a un supermercato in cui i clienti possono acquistare più prodotti e pagarli in casse diverse:



**PRODOTTO**(CodProd, Descrizione)

**CLIENTE**(CodCli, Nome, Cognome)

**CASSA**(CodCassa, Posizione)

**HAFATTURATO**(CodProd, CodCli, CodCassa, Data)

# 6 I vincoli di integrità

Una relazione non deve (e non può) essere vista come un contenitore di dati arbitrari. Se così fosse, non riusciremmo più a interpretare correttamente i dati e molte operazioni non si potrebbero eseguire in maniera affidabile. È quindi necessaria un'attività di analisi volta a evidenziare quali sono **i vincoli che le istanze devono soddisfare** affinché possano essere considerate  **valide** (o “corrette”, “legali”, “ammissibili”, ecc.).

Studente	Matricola	Nome	Nascita
	276545	Lezzi	25/12/1975
	276545	Serafini	12/12/1980
	784563	Trezza	28/03/1982

Esame	Studente	Voto	Lode	Corso
?	276545	28	SI	01
?	276545	32	NO	02
	784563	23	NO	03
?	200768	30	SI	03

Corso	Codice	Nome	Docente
	01	Analisi	Rossi
	02	NULL	NULL
	03	Fisica	Grandi

Possiamo classificare i vincoli di integrità del modello relazionale in:

- **vincoli intrarelazionali** o **interni**, che sono definiti all'interno di una singola relazione. Questi possono essere suddivisi in:
  - **vincoli su singola ennupla**, che esprimono una condizione:
    - **sul dominio degli attributi**: sono i vincoli che coinvolgono un solo attributo, il cui soddisfacimento può essere verificato facendo riferimento a un singolo valore alla volta;
    - **su più attributi**: sono i vincoli che coinvolgono più attributi, ma sempre di ciascuna ennupla, indipendentemente dalle altre;
  - **vincoli su più ennuple**, che coinvolgono i valori di più ennuple; rientrano in questa categoria i **vincoli di chiave primaria** (le tuple presenti in una relazione devono essere tutte diverse tra loro);
- **vincoli interrelazionali** o **esterni**, che sono definiti tra più relazioni; rientrano in questa categoria i **vincoli referenziali**.

Diamo uno sguardo alla seguente relazione rappresentata sotto forma di tabella:

Matricola	Cognome	Nome	DataNascita	Reddito	Email
	Verdi		12/05/1980	(Gennaio, 1500) (Febbraio, 2000) (Marzo, 1800) (Aprile, 2500)	vmar@lib.it
	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

Violazione del vincolo di obbligatorietà: l'attributo *Nome* è obbligatorio (altrimenti avremmo aggiunto un asterisco alla fine)

Violazione del vincolo di atomicità: l'attributo *Reddito* deve contenere un dato atomico e non composto

Violazione del vincolo di chiave: due istanze non possono avere chiave uguale. Inoltre, il campo chiave non può avere valori nulli

Violazione del vincolo di dominio: i dati presenti "non hanno senso" e/o non rispettano le regole implicite del singolo tipo di dato

Abbiamo considerato tutti i vincoli impliciti ed esplicativi che sono stati già visti per il modello concettuale.

Ora consideriamo la relazione *Dipendente*, il cui schema è:

DIPENDENTE(Matricola, StipendioLordo, Trattenute, DataAssunzione, DataNascita)

Possiamo individuare i seguenti vincoli intrarelazionali su singola ennupla:

1. *StipendioLordo* > 0
2. *DataAssunzione* > *DataNascita*
3. *Trattenute* > 0

Un'istanza della relazione soddisfa questo vincolo solo quando tutte le tuple lo soddisfano. In particolare, i vincoli 1 e 3 sono vincoli di dominio.

Il vincolo 2, invece, è un vincolo su più attributi.

Per rappresentare i vincoli intrarelazionali utilizziamo la seguente sintassi:

V<NumProgressivo>(<NomeRelazione>): <Espressione>

dove:

- <NumProgressivo> è il numero progressivo del vincolo relativo alla relazione <NomeRelazione>;
- <Espressione> è una qualsiasi espressione in pseudolinguaggio o linguaggio naturale che serve per specificare il vincolo.

Ad esempio, i vincoli sulla relazione *Dipendente*, vista in precedenza, possono essere i seguenti:

V1(DIPENDENTE): StipendioLordo > 0

V2(DIPENDENTE): DataAssunzione > DataNascita

V3(DIPENDENTE): Trattenute > 0

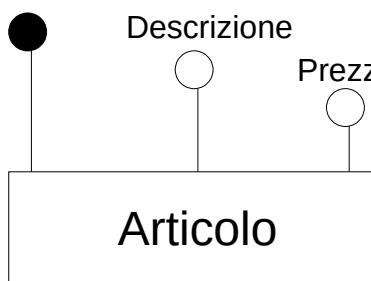
I **vincoli di integrità referenziale** riguardano i valori assunti dalle chiavi esterne nelle relazioni. Poiché una chiave esterna è utilizzata per stabilire un legame tra relazioni, il suo valore deve essere tenuto in stretto controllo per le operazioni di inserimento, modifica e cancellazione.

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

Consideriamo le relazioni *Articolo* e *Fornitore* per stabilire un legame tra esse, al fine di conoscere gli articoli forniti da un certo fornitore, sapendo che un articolo può essere fornito da più fornitori e un fornitore fornisce più articoli.

Per questo dobbiamo creare una nuova relazione che chiameremo *Fornisce*, utilizzando le chiavi primarie delle due relazioni *Articolo* e *Fornitore*, che diventano chiavi esterne (e primarie) della nuova relazione.

CodArt

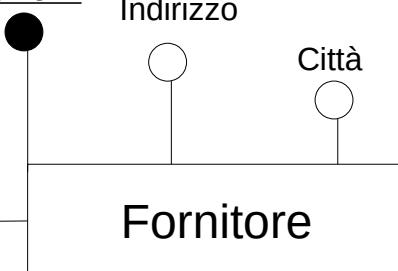


è fornito  
(1,N)

fornisce

associazione  
(N:N)

CodForn



fornisce  
(1,N)

**ARTICOLO**(CodArt, Descrizione, Prezzo)

ARTICOLO	<u>CodArt</u>	Descrizione	Prezzo
	A01	Batteria	100,00
	A04	Antenna	25,00
	A12	Radiatore	1200,00

Chiave primaria

**FORNITORE**(CodForn, Indirizzo, Città)

FORNITORE	<u>CodForn</u>	Indirizzo	Città
	F03	Via Po, 3	Roma
	F07	Via Tevere, 6	Torino
	F16	Via Bari, 5	Milano

Chiave primaria

**FORNISCE**(CodForn, CodArt)

FORNISCE	<u>CodForn</u>	<u>CodArt</u>
	F03	A01
	F03	A04
	F16	A04
	F07	A12

Chiavi esterne

Un esempio di inconsistenza dei dati si ha se dalla relazione *Fornitore* si cancella il fornitore con chiave F03. Nella relazione *Fornisce*, infatti, avremo la chiave esterna F03 alla quale non corrisponde alcun fornitore. Vale lo stesso discorso se si cancella un articolo.

Per assicurare l'integrità referenziale, prima di cancellare una tupla occorre verificare che non ci siano in altre relazioni tuple che facciano riferimento alla tupla da cancellare.

## Rifletti

L'integrità referenziale è assicurata direttamente dal DBMS, che prevede la possibilità di dichiarare le regole di validazione attraverso appositi linguaggi dichiarativi. Tali regole vengono mantenute su speciali archivi detti cataloghi delle regole.

## Regole pratiche

Più in generale, le regole pratiche per l'integrità referenziale sono le seguenti:

- 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 tupla dalla tabella primaria, se esistono righe a essa legate attraverso la chiave esterna nella tabella correlata;
- non si può modificare il valore alla chiave nella tabella primaria, se a essa corrispondono righe nella tabella correlata.

# 8 Le operazioni relazionali

Focalizziamo l'attenzione sulle operazioni che consentono di interrogare una base di dati relazionale. Interrogare una base di dati significa ottenere le informazioni desiderate estraendo da una tabella una sottotabella, oppure combinando tra loro due o più tabelle per generare nuove relazioni. 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 (o **query**) è una relazione che si ottiene formulando un'interrogazione con l'uso di alcuni operatori di algebra relazionale.

---

Gli operatori dell'algebra relazionale si classificano in **primitivi** e **derivati**.

**Gli operatori primitivi sono:**    **Gli operatori derivati sono:**

- |  |  |
|--|--|
| <ol style="list-style-type: none"><li>1. <b>ridenominazione</b></li><li>2. <b>unione</b></li><li>3. <b>differenza</b> di relazioni</li><li>4. <b>proiezione</b> di relazioni</li><li>5. <b>selezione</b> (o <b>restrizione</b>)</li><li>6. <b>prodotto</b></li></ol> | <ol style="list-style-type: none"><li>1. <b>intersezione</b></li><li>2. <b>giunzione</b></li></ol> |
|--|--|

Prima di procedere alla descrizione delle singole operazioni, formuliamo la seguente definizione, particolarmente utile nel prosieguo:

---

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.
- 

Ad esempio, le seguenti relazioni *Persona* e *Dipendente* sono compatibili:

PERSONA(Nome: STRINGA, Stipendio: INTERO, DataNascita: DATA)

DIPENDENTE(Nominativo: STRINGA, Stip: INTERO, DataNas: DATA)

# 9 Gli operatori primitivi

## Ridenominazione

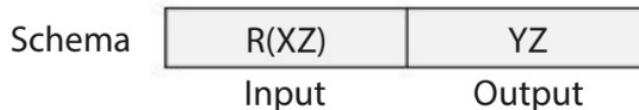
*rho*

L'operatore di **ridenominazione** **p** modifica lo schema di una relazione, cambiando i nomi di uno o più attributi.

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

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

Espressione:  $\rho_{Y \rightarrow X}(r)$



# OSSERVA COME SI FA

---

## Importo

CF	Imponibile
BNCGRG78F21A	250000

$\rho_{\text{CodiceFiscale} \leftarrow \text{CF}} (\text{Importo})$

CodiceFiscale	Imponibile
BNCGRG78F21A	250000

## Volo

Numero	Giorno
AZ400	28/07/2001
AZ158	30/07/2001

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

Codice	Data
AZ400	28/07/2001
AZ158	30/07/2001

## Unione

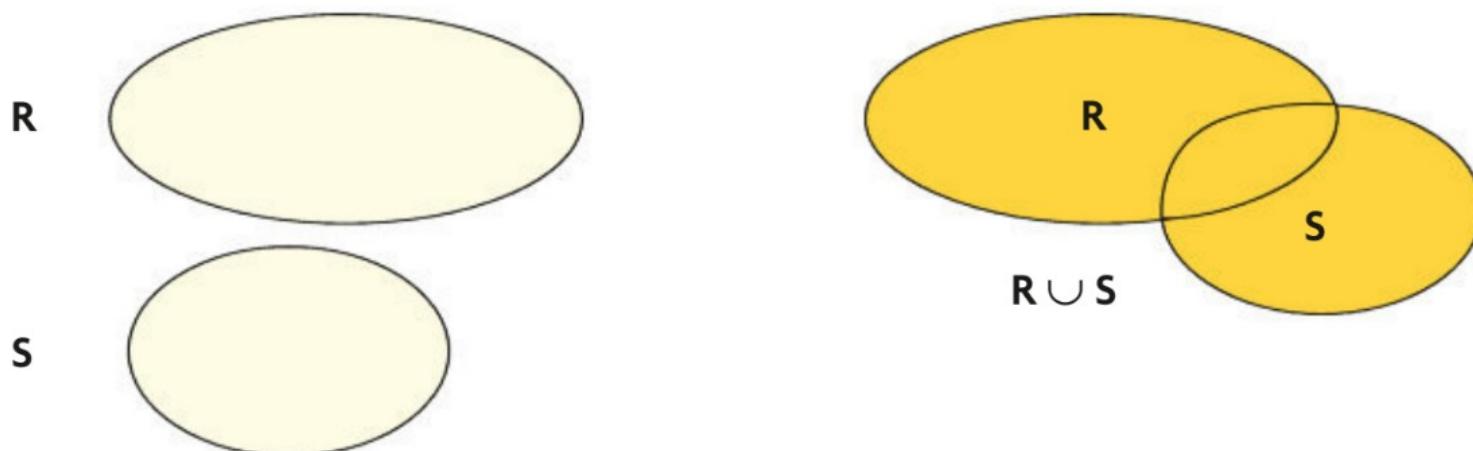
---

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

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

---

Utilizzando la rappresentazione insiemistica abbiamo:



Ad esempio, se R rappresenta i clienti del primo semestre di attività di un'azienda e S i clienti del secondo semestre,  $R \cup S$  rappresenta i clienti dell'anno.

**R = Clienti1Semestre**

R	Cognome			
	Rossi			
	Bianchi			
	Verdi			

**S = Clienti2Semestre**

S	Cognome			
	Gialli			
	Bianchi			
	Neri			

$$R \cup S = \text{Clienti} = \text{Clienti1Semestre} \cup \text{Clienti2Semestre}$$

R $\cup$ S	Cognome			
	Rossi			
	Bianchi			
	Neri			
	Verdi			
	Gialli			

Per come è stata definita l'operazione di unione abbiamo che:

$$\text{Grado}(R \cup S) = \text{Grado}(R) = \text{Grado}(S)$$

$$\text{Card}(R \cup S) = \text{Card}(R) + \text{Card}(S) - \text{numero di tuple ripetute}$$

Ovviamente abbiamo supposto che le tuple con nome “Bianchi” presenti in entrambe le relazioni fossero uguali (per semplicità abbiamo considerato solo *Cognome* come attributo chiave primaria).

# Differenza

---

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

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

---

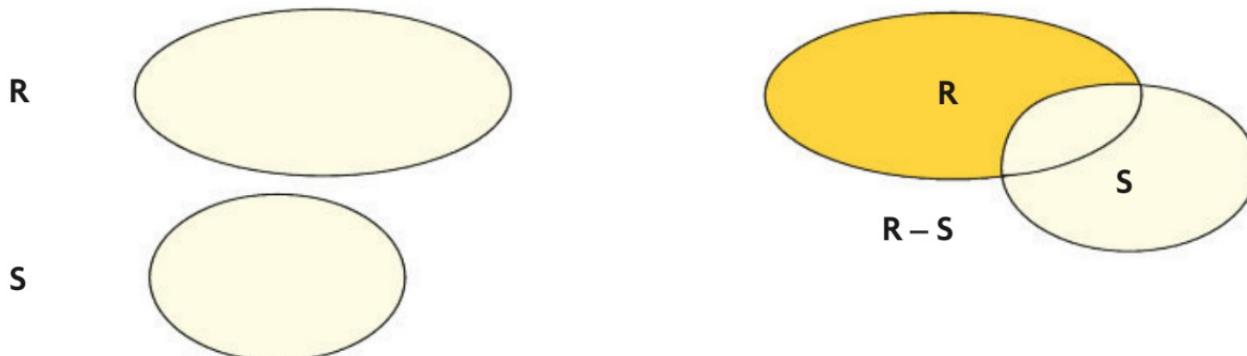
La differenza insiemistica non gode della proprietà commutativa, quindi:

$$R - S \neq S - R$$

Infatti:

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

Utilizzando la rappresentazione insiemistica abbiamo:



Ad esempio, se R rappresenta tutti i clienti di una certa azienda e S rappresenta i clienti dell'anno 2009,  $R - S$  rappresenta tutti i clienti esclusi quelli relativi al 2009.

<b>R = Clienti</b>				
<b>R</b>	<u>Cognome</u>			
	Rossi			
	Bianchi			
	Neri			

<b>S = Clienti09</b>				
<b>S</b>	<u>Cognome</u>			
	Gialli			
	Bianchi			
	Neri			

<b>R - S = Clienti - Clienti09</b>				
<b>R - S</b>	<u>Cognome</u>			
	Rossi			

Le tuple presenti in S non vengono inserite.

Per come è stata definita l'operazione di differenza abbiamo che:

$$\text{Grado}(R - S) = \text{Grado}(R) = \text{Grado}(S)$$

$$\text{Card}(R - S) = \text{Card}(R) - \text{numero di tuple presenti anche in } S$$

## Proiezione

---

Data una relazione R e un sottoinsieme  $A = \{A_1, A_2, \dots, A_k\}$  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 tuple duplicate.

$$\pi_{A_1, A_2, \dots, A_k}(R) = \{ t[A_1, A_2, \dots, A_k] \mid t \in R \}$$

---

L'effetto di una proiezione è quello di selezionare un certo numero di colonne dalla tabella della relazione. Quindi il grado della relazione ottenuta con la proiezione è minore o uguale al grado della relazione R di partenza, quindi  $\text{Grado}(S) \leq \text{Grado}(R)$ .

Consideriamo la seguente relazione CLIENTI relativa ai clienti di un'azienda e supponiamo di voler estrarre solo il cognome e il nome dei clienti.

<b>R = Clienti</b>					
<b>R</b>	<u>CodCli</u>	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

Pi greco

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

<b>S</b>	Cognome	Nome
	Bianchi	Andrea
	Neri	Paolo
	Verdi	Gianfranco

Card(S) ≤ Card(R)

Card(S) ≤ Card(R), infatti le tuple presenti nella proiezione possono anche essere di numero inferiore a quelle di R, poiché le tuple duplicate vengono scartate.

## Selezione

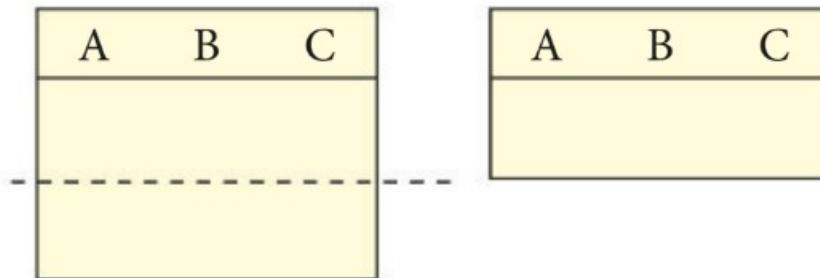
Data una relazione R e un predicato P (semplice o composto) sui suoi attributi, la **selezione** (o **restrizione**) di R a P è la relazione costituita dalle tuple di R che soddisfano P.

Scriveremo:

delta

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

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



Prendiamo sempre in esame la precedente relazione CLIENTI e selezioniamo le informazioni relative ai clienti di Torino. Il nostro predicato sarà:  $P(t) = \{\text{Citta} = \text{"Torino"}\}$ .

**R = Clienti**

<b>R</b>	<u>CodCli</u>	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 =  $\sigma_{Citta='Torino'}(R)$**

<b>S</b>	<u>CodCli</u>	Cognome	Nome	Indirizzo	Citta
	C001	Bianchi	Andrea	Via Po, 23	Torino
	C003	Verdi	Gianfranco	Via Europa, 150	Torino

$Grado(S) = Grado(R)$

$Card(S) \leq Card(R)$ , è uguale solo quando tutte le tuple soddisfano P.

## Prodotto cartesiano

Il prodotto cartesiano tra due insiemi A e B (indicato con  $A \times B$ ) 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.

Il **prodotto cartesiano** di due relazioni coincide con il prodotto cartesiano tra due insiemi, ed è dunque l'insieme di tutte le possibili tuple ottenute concatenando ogni tupla della prima relazione con ogni tupla della seconda.

A	B
1	b
2	a



C	D
3	d
4	g

A	B	C	D
1	b	3	d
1	b	4	g
2	a	3	d
2	a	4	g

La chiave primaria della relazione prodotto è data dall'unione delle chiavi primarie delle due relazioni di partenza.

Abbiamo una relazione R, che rappresenta gli alunni di una classe terza, e una relazione S che rappresenta i testi adottati per quella stessa classe (gli attributi sono solo a titolo di esempio). Vogliamo costruire una tabella con l'elenco di tutti i testi per ogni alunno.

R = Alunni			S = Testi			
R	Matricola	Nominativo	S	CodTesto	Titolo	Materia
	12346	Bianchi Stefania		T1	L'italiano oggi	Italiano
	12347	De Pascalis Alberto		T2	Conoscere la matematica	Matematica
	12348	Manca Roberta		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

$$\text{Grado}(R \times S) = g_1 + g_2$$

$$\text{Cardinalità}(R \times S) = g_1 \times g_2$$

# 10 Gli operatori derivati

## Intersezione

---

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

---

Scrivremo:

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

Supponiamo di avere le informazioni relative ai clienti del 2008 e a quelli del 2009 della nostra azienda. Vogliamo ottenere una tabella con le persone che sono state nostre clienti sia nel 2008 sia nel 2009.

**R = Clienti08**

<b>R</b>	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C006	Bianchi	MI	Via Po, 23
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

**S = Clienti09**

<b>S</b>	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C016	Verdi	CO	Via Moro, 13
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

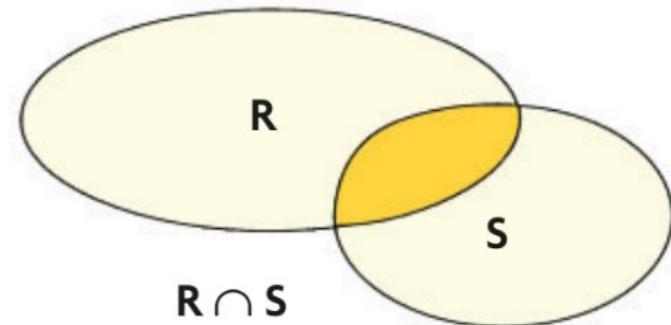
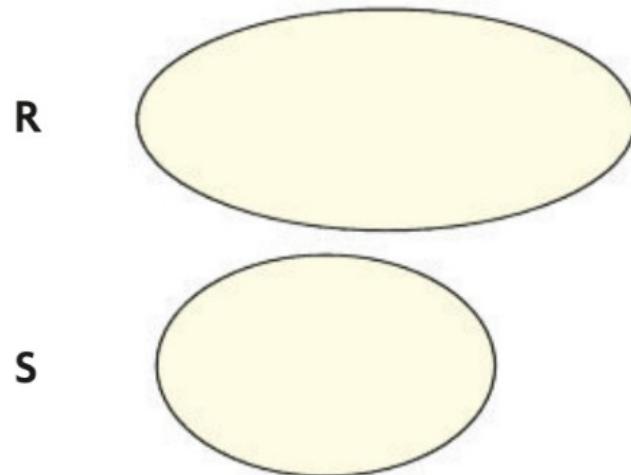
**R ∩ S**

<b>R ∩ S</b>	<u>CodCli</u>	Cognome	Provincia	Indirizzo
	C002	Neri	LE	Via Roma, 12
	C005	Rossi	MI	Via Moro, 2

$$\text{Grado}(R \cap S) = \text{Grado}(R) = \text{Grado}(S)$$

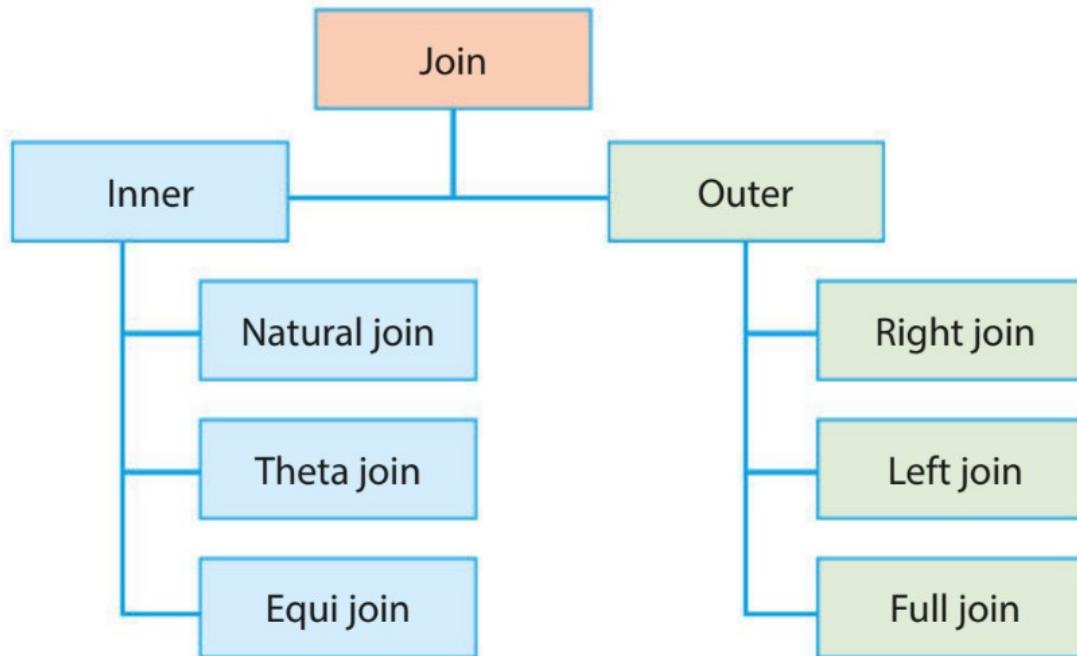
$\text{Card}(R \cap S) \leq$  alla cardinalità minore tra  $\text{Card}(R)$  e  $\text{Card}(S)$ . In particolare si ha:  $\text{Card}(R \cap S) = \text{Card}(R)$  se  $R \subseteq S$   
 $\text{Card}(R \cap S) = \text{Card}(S)$  se  $S \subseteq R$

Utilizzando la rappresentazione insiemistica abbiamo:



# Join

Tra le operazioni derivate, quelle che rivestono maggiore utilità sono quelle di giunzione (*join*) che consentono di costruire una relazione partendo da due relazioni e applicando uno specifico criterio di restrizione sul loro prodotto cartesiano. In base alla natura specifica del criterio di restrizione e al risultato che si intende ottenere, si distinguono diversi tipi di join:



## Inner join

L'inner join restituisce solo i record verificati esistenti in entrambe le tabelle, escludendo, di conseguenza, tutte le tuple che non hanno corrispondenza.

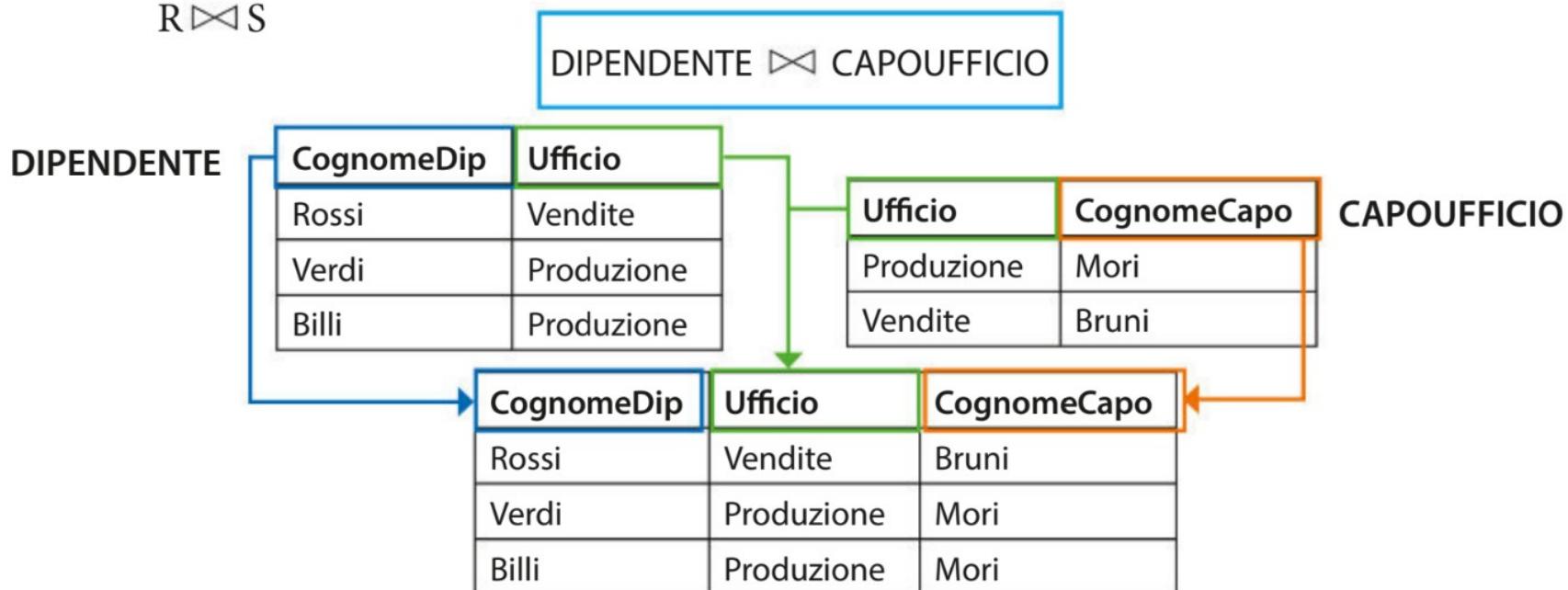
## Natural join (giunzione naturale)

È un operatore che correla i dati di due relazioni R e S sulla base di **valori uguali in attributi con lo stesso nome** in R e S definiti sugli stessi domini. La relazione risultante:

- ha per attributi l'unione degli attributi delle relazioni di partenza;
- le sue tuple sono ottenute selezionando le tuple delle relazioni con valori uguali negli attributi comuni.

In simboli:

$R \bowtie S$



# ORA TOCCA A TE

1. Completa la seguente tabella riportando le tuple che soddisfano la giunzione naturale indicata.

Volo

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

Codice	Partenza	Arrivo
AZ427	FCO	JFK
TW056	LAX	FCO

Tratta

Prenotazione

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

Volo  $\bowtie$  Tratta

Codice	Data	Comandante	Partenza	Arrivo

Volo  $\bowtie$  Prenotazione

Codice	Data	Comandante	Classe	Cliente

## Theta join

---

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

---

$$\begin{array}{c} R \bowtie S \\ A \theta B \end{array}$$

A e B sono gli attributi delle due relazioni corrispondenti al dominio in comune e  $\theta$  è un operatore di confronto ( $=, <, >, \geq, \leq, \neq$ ).

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

1. viene effettuato il prodotto cartesiano tra le due tabelle;
2. sulla tabella così risultante viene eseguita la selezione delle righe in cui gli attributi appartenenti al dominio comune soddisfano la condizione;
3. vengono ridenominati gli attributi comuni con lo stesso nome, in modo che compaiano una volta sola.

Nel caso in cui l'operatore  $\theta$  sia quello di uguaglianza ( $=$ ) si parla di **equi join**.

Il **theta join** è la combinazione di prodotto cartesiano e selezione; vale infatti la seguente uguaglianza:

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

## OSSERVA COME SI FA

IMPIEGATO

Cognome	CodProgetto
Rossi	A
Neri	A
Neri	B

PROGETTO

ID	NomeProgetto
A	Venere
B	Marte

IMPIEGATO  $\bowtie$  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  $\bowtie$  PROGETTO

CodProgetto = ID

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

# ORA TOCCA A TE

1. Completa la seguente tabella riportando le tuple che soddisfano il theta join indicato.

IMPIEGATO

Cognome	CodProgetto
Rossi	HK27
Verdi	HAL2000
Bianchi	HK27
Verdi	HK28
Neri	HAL2000

PROGETTO

ID	Direttore
HK27	Bianchi
HAL2000	Neri
HK28	Verdi

IMPIEGATO  $\bowtie$  PROGETTO

CodProgetto = ID

Cognome	CodProgetto	ID	Direttore

2. Esegui il seguente theta join compilando la tabella con gli attributi e i valori interessati.

**IMPIEGATO  $\bowtie$  PROGETTO**  
(CodProgetto = ID) AND (Cognome < $\neq$  Direttore)


## Outer join

Cerchiamo di comprendere, ora, in che cosa consiste il **join esterno** (*outer join*). Per poterne esplicare al meglio le funzionalità e il significato, serviamoci di un esempio basato sulle due relazioni *Persona* e *Automobile*.

PERSONA

	<u>Codice</u>	Nome	Cognome
	1	Mario	Rossi
	2	Luigi	Bianchi
	3	Giuseppe	Neri

AUTOMOBILE

	<u>Targa</u>	Modello	Codice
	AASSGG	Tip01	1
	UUJJHH	Tip02	1
	PPLLBB	Tip03	2
	WWYYXX	Tip04	

Effettuiamo un join naturale sul campo *Codice* comune alle due relazioni. Otteniamo la tabella:

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

Particolare attenzione deve essere rivolta al ruolo che i campi nulli ricoprono negli inner join. L'automobile di targa WWYYXX, così come *Giuseppe Neri*, non sono presenti nella tabella risultante dall'inner join. Se avessimo voluto informazioni sui proprietari di tutte le automobili, avremmo cercato il risultato visibile qui a lato.

L'operazione di giunzione naturale esclude automaticamente tutti i record aventi valore nullo lungo la colonna impiegata come elemento di giunzione.

Per includere anche tali valori, è stato messo a punto il join esterno (*outer join*).

Targa	Modello	Codice
AASSGG	Tip01	1
UUJJHH	Tip02	1
PPLLBB	Tip03	2
WWYYXX	Tip04	Null

## Right outer join

Date due relazioni A e B, il **right outer join** restituisce una relazione selezionando tutte le tuple di A che corrispondono con B, più i record di B (ossia la relazione di destra) che non corrispondono. Le tuple che non corrispondono vengono valorizzate a NULL.

Utilizzando un right outer join fra le tabelle *Automobile* e *Persona* sul comune campo *Codice* si ottiene il risultato visibile a lato.

Come è facile osservare, fa parte dell'insieme delle righe restituite anche *Giuseppe Neri*. Il suo record proviene dalla relazione di destra (*right*, appunto). Benché non abbia corrispondenze con la relazione di sinistra, è stato ugualmente incluso. Chiaramente, non è possibile determinare dei valori significativi per i due campi *Targa* e *Modello*, pertanto il DBMS ha completato la riga servendosi di due valori nulli, scelti appositamente per rappresentare l'assenza di informazione.

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 tuple di A che corrispondono con B, più i record di A (ossia la relazione di sinistra) che non corrispondono. Le tuple che non corrispondono vengono valorizzate a NULL.

Utilizzando un left outer join si ottiene il risultato visibile a lato. Anche in questo caso, i campi che non hanno corrispondenze sono stati completati con dei valori nulli.

Targa	Modello	Nome	Cognome
AASSGG	Tip01	Mario	Rossi
UUJJHH	Tip02	Mario	Rossi
PPLLBB	Tip03	Luigi	Bianchi
WWYYXX	Tip04	Null	Null

## Full outer join

Il full outer join applica contemporaneamente sia il left outer join sia il right outer join. Utilizzando un full outer join si ottiene il seguente risultato:

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

# 11 La normalizzazione

Se lo schema della base di dati non è costruito correttamente, può accadere che si abbiano delle **anomalie** nel database. Ciò può compromettere la congruenza dei dati contenuti nella base di dati durante le operazioni di inserimento, aggiornamento e modifica. Facciamo un esempio. Consideriamo la seguente relazione *Atipica*.

<u>CodCliente</u>	Città	<u>CodArticolo</u>	Descrizione	NumPezzi
Rossi	Roma	Art1	Matita colorata	10
Rossi	Roma	Art2	Penna biro	20
Rossi	Roma	Art7	Penna stilografica	54
Neri	Lecce	Art5	Gomma	62
Verdi	Milano	Art1	Matita colorata	1
Verdi	Milano	Art2	Penna biro	25

Valutiamo, ora, le anomalie.

- **Anomalia in inserimento:** per inserire un nuovo cliente è necessario inserire contestualmente un articolo ordinato. Allo stesso modo, non è possibile inserire un nuovo articolo senza specificare un acquirente (ciò perché la chiave primaria della relazione è *CodCliente*, *CodArticolo* e non può mai essere nulla);

<u>CodCliente</u>	Città	<u>CodArticolo</u>	Descrizione	NumPezzi
Rossi	Roma	Art1	Matita colorata	10
Rossi	Roma	Art2	Penna biro	20
Rossi	Roma	Art7	Penna stilografica	54
Neri	Lecce	Art5	Gomma	62
Verdi	Milano	Art1	Matita colorata	1
Verdi	Milano	Art2	Penna biro	25

- **Anomalia in cancellazione:** se si cancella una tupla relativa a un acquisto, si corre il rischio di cancellare anche dati relativi al cliente. Ad esempio, cancellando la tupla *(Rossi, Art1)* non si cancella l'indirizzo del cliente che continua a rimanere grazie alle tuple *(Rossi, Art2)* e *(Rossi, Art7)*. Se, invece, si cancella la tupla *(Neri, Art5)* si perdono irrimediabilmente anche i dati del cliente. Analoghe considerazioni possono essere fatte per l'articolo.

<u>CodCliente</u>	Città	<u>CodArticolo</u>	Descrizione	NumPezzi
Rossi	Roma	Art1	Matita colorata	10
Rossi	Roma	Art2	Penna biro	20
Rossi	Roma	Art7	Penna stilografica	54
Neri	Lecce	Art5	Gomma	62
Verdi	Milano	Art1	Matita colorata	1
Verdi	Milano	Art2	Penna biro	25

- **Anomalia in aggiornamento:** se occorre variare l'indirizzo di un cliente, occorrerà aggiornare ogni tupla in cui compare quel cliente.

Queste anomalie, in particolare quelle in cancellazione e in aggiornamento, sono una diretta conseguenza della **ridondanza**, ossia della presenza di dati ripetuti inutilmente, cioè senza l'apporto di nuova informazione.

Lo scopo della **teoria della normalizzazione** è quello di fornire un metodo per progettare basi di dati senza anomalie, ossia per creare tabelle corrette.

---

La **normalizzazione** è un procedimento di tipo graduale, che realizza un'ottimizzazione progressiva a partire da relazioni non normalizzate fino a raggiungere un certo livello di normalizzazione. In particolare, consente di verificare se la definizione dello schema corrisponde ai "canoni standard" di correttezza della base di dati e, in caso, avvalendosi di un preciso insieme di regole, di riportare le tabelle in quelle che sono definite le **forme normali** delle tabelle relazionali.

---

Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la "qualità", cioè l'assenza di determinati difetti.

---

---

Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la “qualità”, cioè l'assenza di determinati difetti.

---

Nella teoria delle basi di dati relazionali esistono diverse forme normali (prima, seconda, terza, forma normale di Boyce-Codd, quarta e quinta). Nella nostra trattazione giungeremo sino alla forma normale di Boyce-Codd, lasciando la trattazione della quarta e della quinta a studi superiori. È possibile riassumere la teoria della normalizzazione nelle seguenti regole:

- ogni tabella deve avere una chiave primaria;
- ogni campo deve contenere un solo valore;
- i campi di una tabella non devono dipendere da altri campi che non siano la chiave primaria;
- bisogna evitare le ripetizioni e la ridondanza dei dati.

In fase di progettazione occorre sempre verificare che ogni tabella abbia una chiave primaria (composta da un campo o da un insieme di campi). Facciamo un esempio. Consideriamo la seguente tabella *OrdiniAcquisto*:

Data	Cliente	Quantità	Articolo	Prezzo	PrezzoTotale	TipoPagamento
15/12/14	Rossi	25	A023	€ 50,25	€ 1.256,25	Contanti
15/12/14	Neri	120	A789	€ 25,80	€ 3.096,00	Contanti
15/12/14	Rossi	100	A976	€ 22,14	€ 2.214,00	Bonifico
15/12/14	Rossi	45	G324	€ 50,25	€ 2.261,25	Assegno
29/12/14	Verdi	120	A023	€ 25,80	€ 3.096,00	Contanti

La tabella contiene chiavi candidate realizzate su molti attributi. Decidiamo pertanto di aggiungere un nuovo campo, ad esempio il codice dell'ordine, che sia una chiave minimale.

CodOrdine	Data	Cliente	Quantità	Articolo	Prezzo	PrezzoTotale	TipoPagamento
1	15/12/14	Rossi	25	A023	€ 50,25	€ 1.256,25	Contanti
2	15/12/14	Neri	120	A789	€ 25,80	€ 3.096,00	Contanti
3	15/12/14	Rossi	100	A976	€ 22,14	€ 2.214,00	Bonifico
4	15/12/14	Rossi	45	G324	€ 50,25	€ 2.261,25	Assegno
5	29/12/14	Verdi	120	A023	€ 25,80	€ 3.096,00	Contanti

# 12 La prima forma normale (1FN)

Partiamo dalla definizione:

---

Una relazione si dice in prima forma normale (1FN), chiamata anche forma atomica, se:

- esiste una chiave primaria (un insieme di attributi che identifica in modo univoco ogni tupla della relazione);
  - ogni attributo è definito su un dominio di valori atomici (deve essere cioè un campo semplice, quindi non composto e non multiplo).
- 

La 1FN è di norma implicita: ogni informazione deve essere “atomica”, cioè un campo deve contenere una e una sola informazione.

Qualsiasi intervento di scomposizione va sempre fatto nei limiti del buon senso. Per esempio, un indirizzo “Via Santa Maria 80 Bari”, dovrebbe essere scomposto in parti come *Topotassia*, *Toponomia*, *Civico*, *Città* vale a dire (“Via”, “Santa Maria”, “80”, “Bari”), ma di solito, non si arriva così in dettaglio, a meno che non vi sia un’esigenza particolare. Un esempio potrebbe essere quello legato a una realtà che si occupa delle spedizioni postali divise per città; in questo caso l’indirizzo andrebbe disaggregato in maniera profonda. In tutti gli altri casi è sufficiente avere *Indirizzo* e *Città*.

In generale, quindi, quando si incontrano campi che contengono più valori, questi campi devono essere suddivisi in modo che contengano un unico valore su ogni tupla. In alcuni casi la suddivisione è semplice e naturale, in altri è molto più complessa. Come al solito, un esempio vale molto più di tante parole e, a tal proposito, riprendiamo proprio il caso di un indirizzo. Analizziamo la seguente relazione *Persona*.

<u>CodPersona</u>	<u>Cognome</u>	<u>Nome</u>	<u>Indirizzo</u>
1	Neri	Antonio	Viale Unità d'Italia, 34 Lecce
2	Verde	Marco	Via Piave, 6/B Ancona
3	Bianco	Piero	Piazza Dante, 9 Milano

La tabella ha una chiave primaria, quindi la prima regola è confermata. Il campo *Indirizzo*, però, contiene più valori: la via con il numero civico e la città. Per normalizzare la relazione si suddivide il campo *Indirizzo* in modo che ogni informazione sia rappresentata con un apposito attributo.

<u>CodPersona</u>	Cognome	Nome	Indirizzo	Città
1	Neri	Antonio	Viale Unità d'Italia, 34	Lecce
2	Verde	Marco	Via Piave, 6/B	Ancona
3	Bianco	Piero	Piazza Dante, 9	Milano

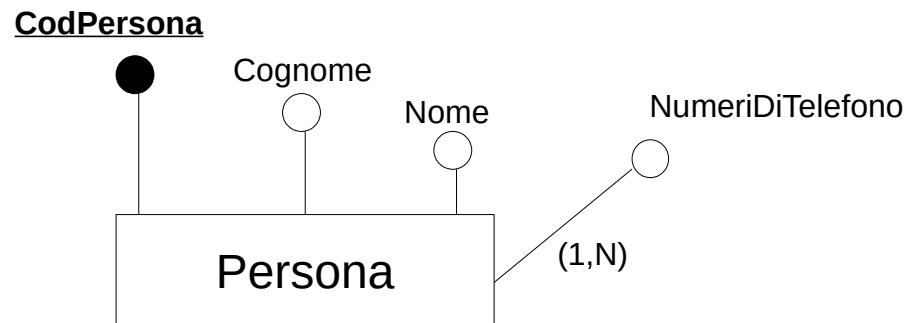
Osserviamo che *Indirizzo*, ora, contiene più informazioni, la via e il numero civico, ma risulta comunque di tipo stringa, che è un tipo semplice.

<u>CodPersona</u>	Cognome	Nome	Indirizzo	NomeVia	Numero	Città
1	Neri	Antonio	Viale	Unità d'Italia	34	Lecce
2	Verde	Marco	Via	Piave	6/B	Ancona
3	Bianco	Piero	Piazza	Dante	9	Milano

Un altro caso di relazione non normalizzata è quello in cui sono presenti attributi multipli. Ricordiamo che un attributo è multiplo se può essere una sequenza di valori tutti dello stesso tipo. Ad esempio, per una persona l'attributo *NumeriDiTelefono*, può essere multiplo poiché una persona può avere più recapiti telefonici.

<u>CodPersona</u>	<u>Cognome</u>	<u>Nome</u>	<u>NumeriDiTelefono</u>		
1	Neri	Antonio	02/324328525	02/89346874	348/4324324
2	Bianco	Marco	06/90596590	335/94054946	
3	Testo	Piero	06/90596590	380/39676654	

In un certo senso è come se a 1 persona corrispondessero N numeri di telefono.

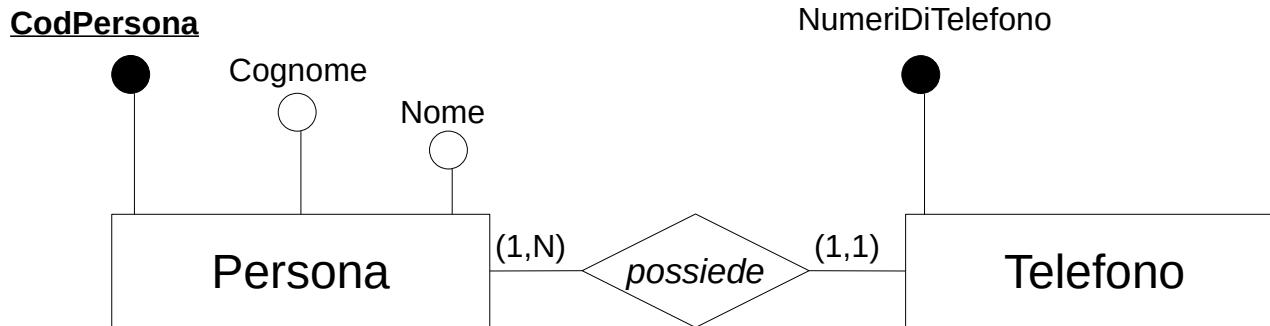


Per avere uno schema in 1FN si sostituisce la relazione non normalizzata con due relazioni: una simile a quella di origine, ma senza l'attributo multiplo, l'altra contenente la chiave primaria della prima relazione e un attributo semplice che contiene ogni singolo valore della sequenza originale. La chiave primaria di quest'ultima relazione è data dall'unione della chiave primaria della relazione iniziale e dell'attributo.

Nel nostro esempio avremmo:

<u>CodPersona</u>	Cognome	Nome
1	Neri	Antonio
2	Verde	Marco
3	Bianco	Piero

<u>CodPersona</u>	<u>NumeroDiTelefono</u>
1	02/32432825
1	02/89346874
1	348/4324324
2	06/90596590
2	335/94054946
3	06/90596590
3	380/39676654



## 13 La seconda forma normale (2FN)

Posto che i dati devono essere tutti semplici, in questo secondo livello normativo si va ad affrontare il vero e proprio aspetto logico della struttura. Partiamo da qualche concetto.

---

Sia  $R$  una relazione che contiene almeno due attributi  $X$  e  $Y$ . Se il valore di  $Y$  varia al variare del valore di  $X$ , si dice che  $Y$  ha una **dipendenza funzionale** da  $X$  e si indica con  $X \rightarrow Y$  (si dice anche che  $X$  determina  $Y$ , o che  $X$  è un determinante per  $Y$ ). Lo stesso vale se, invece di singoli attributi, si considerano insiemi di attributi.

---

Ciò significa che, per ogni insieme di ennuple che possono esistere in  $R$ , non possono esistere due ennuple che hanno lo stesso valore di  $X$  e valori diversi di  $Y$ .

Questo concetto è molto importante. Se il valore di un campo può variare al variare di qualsiasi parte della chiave primaria, possiamo dire che quel campo *dipende dall'intera chiave*, mentre se può variare alla variazione di *una sola parte della chiave*, allora quel campo non dipende dalla chiave intera, ma solo da una parte di essa.

---

Una relazione R è in **seconda forma normale (2FN)** se è in prima forma normale e ogni attributo non chiave dipende funzionalmente e completamente (cioè non parzialmente) dalla chiave primaria.

---

Tutti i campi diversi dalla chiave primaria, pertanto, devono dipendere dall'intera chiave primaria e non da una sua parte.

Prendiamo in considerazione il seguente schema di relazione:

ORDINE(CodOrdine, CodCliente, CodProdotto, DataOrdine, Quantità, PrezzoUnit, Descrizione)

Questa relazione non è in seconda forma normale perché al suo interno sono memorizzate informazioni riguardanti più di un oggetto, cioè più elementi distinti: gli ordini, i prodotti, i prodotti ordinati in ciascun ordine. Supponiamo che le informazioni sugli ordini e sui prodotti non siano presenti in altri schemi.

<u>CodOrdine</u>	<u>CodCliente</u>	<u>CodProdotto</u>	DataOrdine	Quantità	PrezzoUnit	Descrizione
1	C1	A023	15/12/04	25	€ 50,25	Cintura
1	C1	A789	15/12/04	120	€ 25,80	PortachiaviA
1	C1	A976	15/12/04	100	€ 22,14	PortachiaviB
2	C24	G324	15/12/04	45	€ 30,25	Portapatente
3	C56	A023	29/12/04	120	€ 50,25	Cintura

<u>CodOrdine</u>	<u>CodCliente</u>	<u>CodProdotto</u>	<u>DataOrdine</u>	<u>Quantità</u>	<u>PrezzoUnit</u>	<u>Descrizione</u>
1	C1	A023	15/12/04	25	€ 50,25	Cintura
1	C1	A789	15/12/04	120	€ 25,80	PortachiaviA
1	C1	A976	15/12/04	100	€ 22,14	PortachiaviB
2	C24	G324	15/12/04	45	€ 30,25	Portapatente
3	C56	A023	29/12/04	120	€ 50,25	Cintura

Per questi motivi lo schema presenta delle anomalie:

- **anomalia in inserimento:** non è possibile inserire un nuovo articolo in magazzino sino a quando non è ordinato (ciò perché la chiave primaria della relazione è *CodOrdine* e *CodProdotto* e non può mai essere nulla);
- **anomalia in cancellazione:** se si cancellano la seconda o la terza o la quarta tupla, si perdono informazioni sugli articoli che compaiono in esse;
- **anomalia in aggiornamento:** se varia il prezzo unitario di un articolo, occorre aggiornare tutte le tuple in cui compare, con i relativi totali.

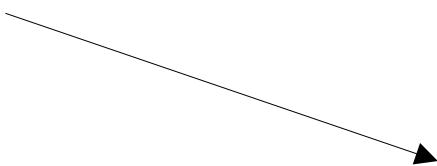
<u>CodOrdine</u>	<u>CodCliente</u>	<u>CodProdotto</u>	<u>DataOrdine</u>	<u>Quantità</u>	<u>PrezzoUnit</u>	<u>Descrizione</u>
1	C1	A023	15/12/04	25	€ 50,25	Cintura
1	C1	A789	15/12/04	120	€ 25,80	PortachiaviA
1	C1	A976	15/12/04	100	€ 22,14	PortachiaviB
2	C24	G324	15/12/04	45	€ 30,25	Portapatente
3	C56	A023	29/12/04	120	€ 50,25	Cintura

Verifichiamo le dipendenze funzionali dalla chiave primaria, composta dagli attributi *CodOrdine* e *CodProdotto*:

*CodOrdine* → *CodCliente*, *DataOrdine*

*CodOrdine*, *CodProdotto* → *Quantità*

*CodProdotto* → *PrezzoUnit*, *Descrizione*



PRODOTTIORDINATI(CodOrdine, CodProdotto, Quantità)  
 PRODOTTO(CodProdotto, PrezzoUnit, Descrizione)  
 ORDINE(CodOrdine, DataOrdine, CodCliente)

Per risolvere questi problemi, si deve scomporre la relazione in relazioni più semplici, ciascuna relativa a una data categoria: gli ordini, i prodotti e i prodotti ordinati. Le relazioni devono essere collegate tramite le chiavi primarie.

Il nuovo schema relazionale che soddisfa la seconda forma normale è, pertanto, il seguente:

PRODOTTIORDINATI(CodOrdine, CodProdotto, Quantità)  
PRODOTTO(CodProdotto, PrezzoUnit, Descrizione)  
ORDINE(CodOrdine, DataOrdine, CodCliente)

Per normalizzare, quindi, si procede in questo modo:

- nello schema originario rimangono la chiave primaria e tutti gli attributi non chiave, se ci sono, che dipendono completamente da essa;
- si crea un nuovo schema di relazione per ogni parte di chiave primaria da cui dipendono completamente altri attributi non chiave.

# OSSERVA COME SI FA

1. Per consolidare le conoscenze e applicare i precedenti punti, facciamo un esempio generico.

Consideriamo lo schema di relazione:

$R(\underline{A}, \underline{B}, C, D, E)$  con le seguenti dipendenze funzionali:

$$A, B \rightarrow C \quad A \rightarrow D \quad B \rightarrow E$$

Gli attributi D ed E non dipendono dall'intera chiave, pertanto la relazione non è in 2FN. Decomponiamo cominciando dalla dipendenza  $A \rightarrow D$ . Dobbiamo costruire due nuove relazioni:

- una prima relazione R1 che comprende tutti gli attributi presenti nella dipendenza funzionale che viola la 2FN. In questa relazione il determinante della dipendenza funzionale diventa, così, la chiave della nuova relazione;
- una seconda relazione R2 composta dagli attributi di R1 privati dell'attributo che dipende parzialmente dalla chiave.

Otteniamo, quindi:

$$R1(\underline{A}, D) \quad R2(\underline{A}, \underline{B}, C, E)$$

Il processo non può terminare poiché è presente l'altra dipendenza funzionale  $B \rightarrow E$ . Applicando ancora il procedimento di decomposizione otteniamo:

$$R3(\underline{B}, E) \quad R4(\underline{A}, \underline{B}, C)$$

Le relazioni R1, R3 e R4 rappresentano la decomposizione di  $R(\underline{A}, \underline{B}, C, D, E)$  in 2FN.

## 14 La terza forma normale (3FN)

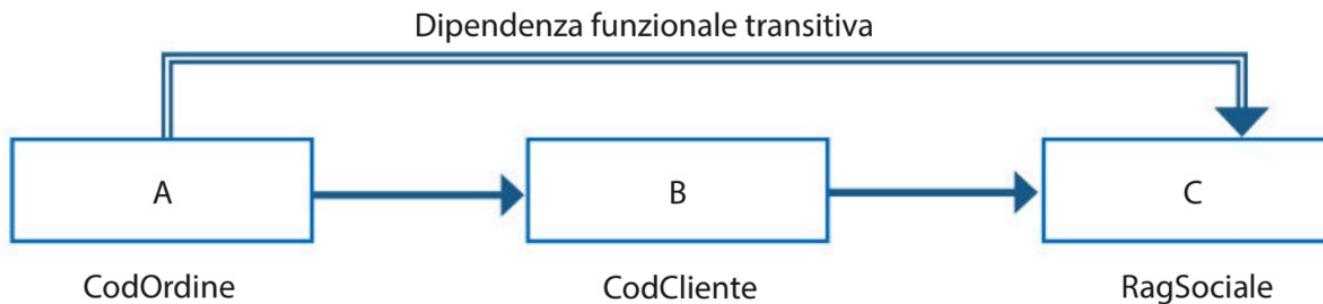
Una relazione R è in **terza forma normale (3FN)** se è in 2FN e ogni attributo non chiave dipende direttamente dalla chiave. In altri termini, la relazione R non deve possedere attributi non chiave che dipendano funzionalmente da altri attributi non chiave.

La 3FN elimina la *dipendenza transitiva* degli attributi dalla chiave. Diamo una definizione di dipendenza transitiva in modo da ben comprendere questa nuova forma normale:

Data una relazione con attributi A, B e C e con chiave primaria A:

- se C dipende funzionalmente dal determinante B, a sua volta funzionalmente dipendente da A;
  - se il determinante B non è una chiave candidata della relazione, alternativa ad A;

allora C ha una **dipendenza funzionale transitiva** da A.



Riprendiamo lo schema di relazione visto in precedenza e aggiungiamo alla relazione *ORDINE* la ragione sociale del cliente:

**ORDINE(CodOrdine, CodCliente, RagSociale)**

<u>CodOrdine</u>	CodCliente	RagSociale
1	C1	Rossi Accessori
2	C2	Verdi e figli
3	C1	Rossi Accessori
4	C3	Accessori per tutti
5	C2	Verdi e figli

Possiamo osservare che anche in questo caso ci sono delle anomalie:

- **anomalia in inserimento:** non è possibile inserire la ragione sociale relativa a un cliente sino a quando quest'ultimo non compare in un ordine (ciò perché la chiave primaria della relazione è *CodOrdine* e non può mai essere nulla);
- **anomalia in cancellazione:** se si cancella, ad esempio, la quarta tupla si perde l'informazione che il cliente di codice *C3* ha ragione sociale *Accessori per tutti*;
- **anomalia in aggiornamento:** se varia la ragione sociale di un certo cliente, occorre aggiornare tutte le tuple interessate.

Questi problemi sono dovuti al fatto che la ragione sociale, in effetti, è indipendente dal codice dell'ordine e dipende solo dal codice del cliente:

CodCliente → RagSociale

ORDINE( <u>CodOrdine</u> , CodCliente, RagSociale)		
<u>CodOrdine</u>	CodCliente	RagSociale
1	C1	Rossi Accessori
2	C2	Verdi e figli
3	C1	Rossi Accessori
4	C3	Accessori per tutti
5	C2	Verdi e figli

Siamo quindi di fronte al caso in cui un attributo non chiave (*RagSociale*) dipende da un altro attributo non chiave (*CodCliente*). Lo schema deve essere, pertanto, trasformato nel seguente:

ORDINE(CodOrdine, CodCliente)  
CLIENTE(CodCliente, RagSociale)

Per normalizzare si procede in questo modo:

- nello schema originario rimangono la chiave primaria e tutti gli attributi non chiave che dipendono direttamente da essa;
- si crea un nuovo schema di relazione per ogni attributo da cui dipendono altri attributi non chiave.

## **ORDINE(CodOrdine, CodCliente, RagSociale)**

<u>CodOrdine</u>	CodCliente	RagSociale
1	C1	Rossi Accessori
2	C2	Verdi e figli
3	C1	Rossi Accessori
4	C3	Accessori per tutti
5	C2	Verdi e figli

<u>CodOrdine</u>	CodCliente
1	C1
2	C2
3	C1
4	C3
5	C2

<u>CodCliente</u>	RagSociale
C1	Rossi Accessori
C2	Verdi e figli
C3	Accessori per tutti



## OSSERVA COME SI FA

1. Per consolidare le conoscenze e applicare i precedenti punti, facciamo un esempio generico. Consideriamo il seguente schema di relazione:

**R(A, B, C, D, E)**

con le seguenti dipendenze funzionali:

$$A \rightarrow B$$

$$A \rightarrow E$$

$$B \rightarrow C$$

$$B \rightarrow D$$

Si evince facilmente che  $A \rightarrow C$  e  $A \rightarrow D$  transitivamente. Questa relazione, quindi, non è in 3FN: procediamo alla decomposizione. Consideriamo la dipendenza funzionale  $B \rightarrow C$ ; otteniamo le due seguenti relazioni:

**R1(B, C) R2(A, B, D, E)**

Il processo non può terminare poiché è presente l'altra dipendenza funzionale  $B \rightarrow D$ . Applicando ancora il procedimento di decomposizione otteniamo:

**R3(B, D) R4(A, B, E)**

Le relazioni R1, R3 e R4 rappresentano la decomposizione di **R(A, B, C, D, E)** in 3FN.

# 15 La forma normale di Boyce-Codd

Generalmente uno schema relazionale è normalizzato, quindi si arresta la decomposizione, una volta che le varie relazioni sono in 3FN. Si può tuttavia continuare a decomporre per portare lo schema nella forma normale di Boyce-Codd.

---

Una relazione R è in **forma normale di Boyce-Codd** (BCNF, Boyce-Codd Normal Form) quando rispetta le caratteristiche fondamentali del modello relazionale (1FN) e se, per ogni dipendenza funzionale  $X \rightarrow Y$  definita su di essa, X (cioè il determinante) è una chiave candidata.

---

Dalla definizione emerge che, se in una relazione è definita la dipendenza funzionale  $X \rightarrow Y$ , allora l'insieme di attributi X deve contenere una chiave candidata (quindi può svolgere la funzione di chiave).

La conseguenza è che una relazione che soddisfa la BCNF è anche in 2FN e in 3FN, poiché la BCNF non ammette che un determinante possa essere formato solo da una parte della chiave, come avviene per le violazioni alla 2FN, o che possa essere esterno alla chiave, come avviene per le violazioni alla 3FN.