# Il modello relazionale

**Prerequisiti**

- Terminologia e concetti introduttivi sulle basi di dati.
- Elementi di metodologia per la progettazione di basi di dati non a oggetti.
- I modelli concettuali.
- Il diagramma ER.

**Obiettivi**

- Modellare una realtà utilizzando uno schema relazionale.

**Conoscenze da apprendere**

- Conoscere le relazioni.
- Conoscere il grado e la cardinalità di una relazione.
- Conoscere le differenti rappresentazioni delle relazioni.
- Conoscere la differenza tra schema e istanze.
- Conoscere le operazioni dell'algebra relazionale.
- Conoscere la normalizzazione delle relazioni.
- Conoscere i vincoli d'integrità.

**Competenze da acquisire**

- Saper trasformare un diagramma ER in uno schema relazionale.

## 1 Introduzione

Nell'unità precedente abbiamo analizzato il diagramma ER, apprezzandone la semplicità e la completezza di rappresentazione. Il diagramma ER è un *modello concettuale,* poiché è adatto a descrivere una realtà di interesse, indipendentemente da come le aggregazioni di dati e le loro associazioni verranno poi implementate.

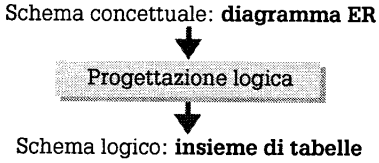
Abbiamo detto che, dopo la *progettazione concettuale*, il passo successivo nella progettazione di una base di dati è la *progettazione logica relazionale.*

Tale passo consiste nel trasformare la rappresentazione, ancora astratta e indipendente, del diagramma ER in una rappresentazione più efficiente (anche se 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.**

Riassumiamo gli ingressi e le uscite della fase di progettazione logica.

Input e Output della progettazione logica



## 2 Le relazioni

Modello relazionale dei dati

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

Una **relazione** R su una sequenza di insiemi $D_1$ ... $D_N$ (non necessariamente distinti) è un sottoinsieme finito del prodotto cartesiano $D_1X$ ... X $D_N$, che possiamo esprimere con:

$$R \subseteq D_1X \ldots X D_N$$

dove:

Grado e dominio della relazione

- N ($N \geq 1$) è detto **grado** della relazione ed è indicato con **Grado(R)**;
- gli insiemi $D_i$ sono detti **domini** della relazione e ciascuno di essi può essere un insieme a supporto di un tipo di dato elementare (ad esempio carattere, intero, reale, booleano ecc.). A ogni dominio è associato un nome, detto **attributo**, che lo identifica univocamente all'interno della relazione.

Schema della relazione

Due domini possono quindi avere lo stesso tipo ma nomi diversi (ad esempio, gli attributi *Peso* e *Altezza* possono essere di tipo reale). Chiameremo **schema di una relazione** il nome della relazione e la lista dei suoi attributi racchiusi tra parentesi tonde e separate da virgole. Lo schema di una relazione indica il significato intensionale di quest'ultima. Utilizzeremo la seguente sintassi per rappresentarlo:

**<NomeRelazione>(<Attributo1>:<Tipo1>, ..., <AttributoN>:<TipoN>)**

dove <Tipo1>, ..., <TipoN> sono i tipi degli attributi che spesso ometteremo per maggiore concisione. Il tipo può essere:

- *Intero*;
- *Reale*;
- *Stringa*;
- *Booleano*;
- *Data*.

Per convenzione scriveremo i nomi delle **relazioni** con la sola **iniziale in maiuscolo**. La stessa cosa accadrà per i nomi degli **attributi** delle relazioni.

### Esempio

Consideriamo il seguente schema della relazione *Persona* per rappresentare le caratteristiche peculiari di un essere umano:

**Persona**(Cognome: Stringa(30), Nome: Stringa(20), Età: Intero(3), Sesso: (Booleano))

Per comodità di esposizione, possiamo rappresentare nel seguente modo lo schema:

| Persona | Attributi | Tipo degli attributi | Dimensione |
|---|---|---|---|
| | Cognome | Stringa | 30 |
| | Nome | Stringa | 20 |
| | Età | Intero | 3 |
| | Sesso | Booleano | |

Per chiarezza espositiva negli esempi che seguiranno per l'attributo Sesso identificheremo Femmina con Falso e Maschio con Vero.

Spesso per brevità di trattazione scriveremo:

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

Gli elementi di R sono detti **ennuple** o **tuple** e indicate con:

$$(d_1, d_2, ...., d_N)$$

dove $d_1 \in D_1$, $d_2 \in D_2$ , ..., $d_N \in D_N$.

Si osservi che, 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 di una relazione rappresenta il suo significato estensionale.

A differenza delle relazioni matematiche, le relazioni del modello relazionale sono **variabili nel tempo**: le tuple possono essere inserite, cancellate, aggiornate.

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

Cardinalità della relazione

## 2.1 Differenti modi di rappresentare una relazione

Essendo definita come sottoinsieme del prodotto cartesiano dei suoi domini, una relazione (come accade per le relazioni matematiche) può essere rappresentata anche nei seguenti modi:

- per **elencazione**;
- in **forma tabellare**;
- in **forma insiemistica**.

### 2.1.1 Rappresentazione per elencazione

Consideriamo la relazione *Persona* precedentemente definita. Possiamo rappresentare un'istanza di *Persona* **elencando** tutte le sue *tuple* come faremmo per gli elementi di un insieme.

**Persona**(Cognome, Nome, Età, Sesso) = {
(Rossi, Paolo, 30, Maschio),
(Bianchi, Antonio, 23, Maschio),
(Neri, Ada, 35, Femmina)
}

Tupla — Istanza composta da 3 tuple

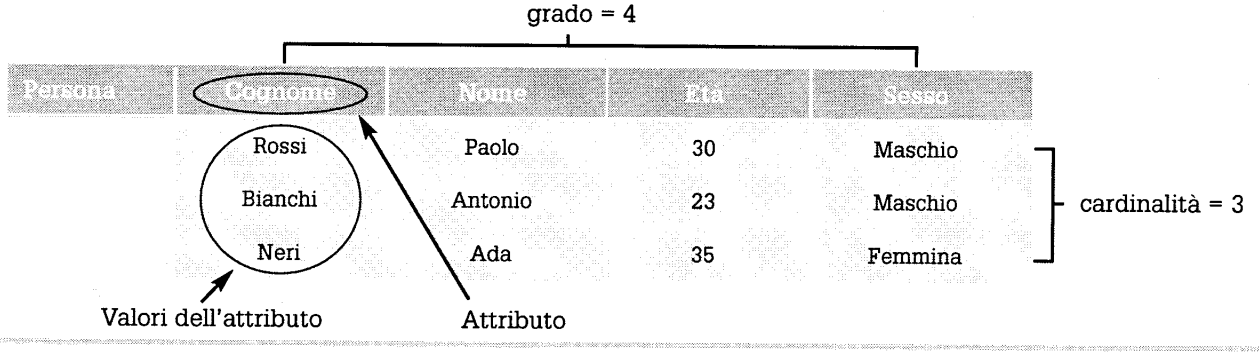
### 2.1.2 Rappresentazione tabellare

Nella **rappresentazione tabellare** invece utilizziamo una tabella di *n righe* × *m colonne*, dove *n* rappresenta il grado e *m* la cardinalità, cioè:

- ogni *riga* rappresenta una *tupla*;
- ogni *colonna* rappresenta la *sequenza* dei valori assunti dal corrispondente attributo.

**Esempio**

Rappresentiamo la stessa istanza della relazione *Persona* con la seguente tabella:

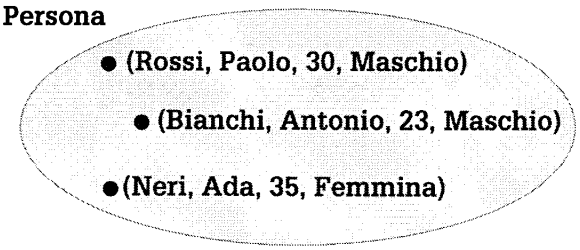


Campo

D'ora in poi considereremo *tupla* sinonimo di *riga* e *attributo* sinonimo di *colonna*. Nelle rappresentazioni tabellari, spesso si utilizza *campo* come sinonimo di *attributo*.

### 2.1.3 Rappresentazione insiemistica

In tale rappresentazione grafica un'ellisse rappresenta l'insieme delle tuple.




✦ Figura A3.1
La rappresentazione insiemistica di una relazione

## 2.2 Chiavi di una relazione

Lo schema di una base di dati deve esprimere la struttura dei dati e gli eventuali vincoli espliciti di integrità dei dati.

Abbiamo appena visto che la *struttura* dei dati è quella delle relazioni in senso matematico.

Per quanto riguarda invece i *vincoli*, tutte le definizioni del modello relazionale prevedono di specificare come vincolo quello relativo alla presenza di una *chiave* per ciascuna relazione.

> Si dice **chiave candidata**, o **superchiave**, di una relazione $R$ un insieme non vuoto $K$ di attributi di $R$, attraverso i quali è possibile individuare univocamente una tupla per ogni possibile istanza della relazione $R$.

Utilizzando i valori degli attributi presenti in $K$, è possibile identificare **un'unica tupla**.

Si osservi che una chiave candidata esiste sempre, al limite essa è costituita da tutti gli attributi di $R$ poiché, come abbiamo già osservato, non possono esistere tuple uguali.

In generale, una relazione può ammettere diverse chiavi candidate. Fra queste ne viene scelta una con il minor numero di attributi (cioè una *superchiave minimale*) che viene designata come **chiave primaria**. D'ora in poi la chiameremo semplicemente **chiave**.

Chiave primaria

Nello schema di una relazione si sottolineano gli attributi che compongono la chiave.

**Esempio**

Consideriamo i dati personali dei clienti di un albergo trascritti su un registro, che può essere assimilato a una relazione avente il seguente schema:

**Ospite**(NumProgressivo, Nome, DataNascita, LuogoNascita, TipoDocumento, NumDocumento)

Sono *chiavi candidate*:

1. (NumProgressivo);
2. (Nome, DataNascita, LuogoNascita);
3. (TipoDocumento, NumDocumento).

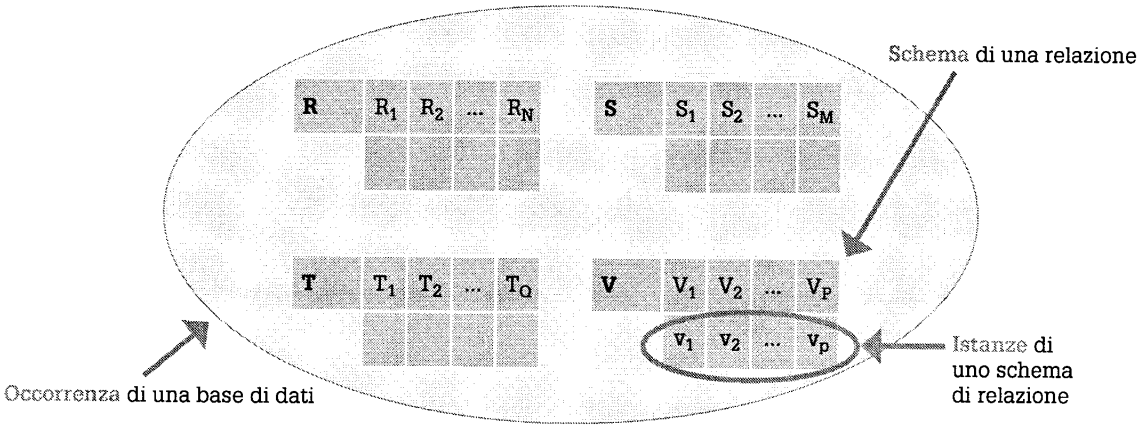
Fra le *chiavi candidate* è stata scelta (NumProgressivo) come *chiave primaria* perché contiene il minor numero di attributi. Nella progettazione di basi di dati, quando non sia possibile utilizzare come chiave un numero limitato di attributi, si ricorre all'aggiunta di un nuovo attributo (come, ad esempio, un numero progressivo che identifica univocamente le tuple).

Le chiavi in una relazione vengono rappresentate **sottolineando** i relativi attributi anche nella rappresentazione tabellare.

## 2.3 Schema e occorrenza di una base di dati

Definiamo **schema** di una base di dati relazionale l'insieme di tutti gli schemi di relazione. Possiamo allora definire **occorrenza** di una base di dati relazionale l'insieme delle istanze degli schemi di relazione.

Possiamo quindi rappresentare la nostra base di dati con un insieme di schemi di relazione. All'interno di ognuno di questi schemi troviamo le istanze. Se consideriamo una base di dati in un determinato momento, avremo una particolare occorrenza di quella base di dati, così come rappresentato dal seguente schema:




✦ Figura A3.2
L'occorrenza di una base di dati

I **legami** tra tali relazioni si realizzano utilizzando le loro **chiavi**.

Legami tra relazioni

Vedremo in seguito come rappresentare nel modello relazionale tali legami, che corrispondono alle *associazioni* del modello ER. Prima però occorre introdurre il concetto di vincolo anche nel modello relazionale, così come abbiamo già fatto nel modello concettuale.

# 3 I vincoli di integrità

Abbiamo detto che una base di dati è un *insieme di relazioni* che varia nel tempo e che è soggetto a continue modifiche, cancellazioni e inserimenti di nuovi dati.

Nella nostra definizione iniziale di base di dati abbiamo detto che queste operazioni devono essere eseguite rispettando un **insieme di regole** che servono per non compromettere l'integrità dei dati (che altrimenti, a volte, potrebbero non avere senso).

Regole per garantire l'integrità dei dati

Consideriamo la relazione *Studente*, che contiene informazioni anagrafiche relative a uno studente. Supponiamo che l'attributo *Matricola* sia un numero progressivo che identifica esattamente quello studente.

| Studente | Nominativo | Matricola | Età |
|---|---|---|---|
| | Rossi | 2345 | 17 |
| | Neri | 7628 | 250 |
| | Bianchi | 7628 | 16 |

Come possiamo notare l'attributo *Età* del nominativo Neri ha un valore pari a 250 (cosa impossibile).

Dal punto di vista delle definizioni questa relazione è *corretta*, infatti 250 è un valore del dominio *intero*. Non tutto ciò che è rappresentato in questa relazione ha però senso, cioè corrisponde a dati verosimili o possibili. Infatti il valore che ci indica un'età di 250 anni non è realistico.

Nella relazione *Studente* ci sono inoltre due tuple che hanno lo stesso valore per il numero di matricola: anche qui dal punto di vista strutturale è tutto legale, ma stiamo considerando il numero di matricola per identificare univocamente due studenti all'interno di una Università. Quindi, questa relazione non rappresenta effettivamente i dati di interesse di una realtà universitaria.

In effetti, proprio per specificare il fatto che alcune relazioni sono corrette dal punto di vista di chi sviluppa l'applicazione e altre non lo sono, viene introdotto il concetto di vincolo di integrità.

> Possiamo definire **vincolo di integrità** una proprietà che deve essere soddisfatta dalle istanze che rappresentano informazioni corrette per l'applicazione che utilizza la base di dati.

Nell'esempio dovremmo quindi specificare un vincolo di integrità che ci dica che l'età deve essere inferiore a 120.

Analogamente, un altro vincolo ci dirà che non possono esserci due studenti con lo stesso numero di matricola: riferendoci al modello relazionale diremo che non devono esserci due tuple della relazione *Studente* con lo stesso valore per l'attributo *Matricola*. Quindi il vincolo di integrità indica proprietà che devono essere soddisfatte dalla nostra base di dati.

Possiamo considerare ogni vincolo come un'**asserzione** che può essere, rispetto a un'istanza, vera o falsa.

Se l'asserzione è vera diciamo che il vincolo è **soddisfatto** dalla nostra istanza.

Questo significa che abbiamo varie istanze possibili sulla base dei domini, degli schemi di relazione e degli schemi di base di dati. Su queste istanze i vari vincoli di integrità possono essere veri o falsi: noi consideriamo accettabili tutte quelle istanze di base di dati per le quali tutti i vincoli di integrità sono veri e diciamo che sono solo quelle le istanze che possono verificarsi nella realtà; il vincolo di integrità è utile come strumento di verifica della qualità della nostra base di dati. Se arriva un dato che non rispetta il vincolo diremo che il dato **viola** un vincolo qualsiasi di integrità. Possiamo quindi scartarlo, perché i dati contenuti in esso non sono realistici. Possiamo classificare i vincoli di integrità del modello relazionale in:

Violazione di un vincolo

Vincoli intrarelazionali

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

Vincoli interrelazionali

- **vincoli interrelazionali o esterni**. Sono quei vincoli definiti tra più relazioni. Rientrano in questo tipo di vincoli i **vincoli referenziali.**

Come possiamo vedere, abbiamo considerato tutti i vincoli impliciti ed espliciti già visti per il modello concettuale.

**Esempio**

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*

Una relazione soddisfa questo vincolo quando tutte le tuple lo soddisfano.
In particolare i vincoli 1 e 3 sono *vincoli di dominio* (come lo era Età > 0 visto per la relazione *Studente*).
Il vincolo 2 invece è un vincolo su *più attributi*.

Se un *vincolo di ennupla* non fosse soddisfatto, allora avremmo una **violazione** di un vincolo di integrità riferito a una singola ennupla.

**Esempio**

Un esempio di vincolo *intrarelazionale su più ennuple* è stato visto in precedenza: uno studente non può avere lo stesso numero di matricola di un altro studente. Questo è un vincolo di **chiave primaria**.

Vincoli di integrità referenziale

Chiavi esterne

Un esempio di vincoli interrelazionali sono i **vincoli di integrità referenziale** o semplicemente **vincoli referenziali**. Vediamo in cosa consistono.

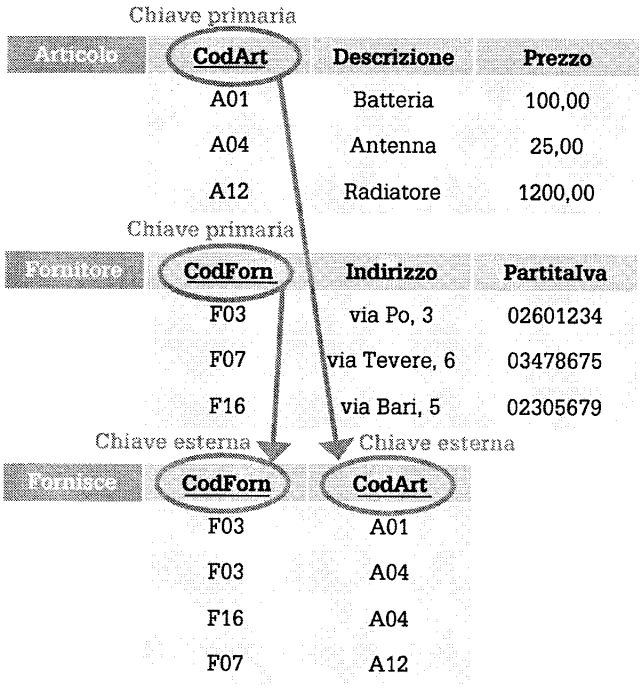
Abbiamo detto che, per stabilire un legame tra due (o più) relazioni, utilizziamo le chiavi primarie delle due relazioni (creando altre relazioni che contengono i valori di queste chiavi), le quali prendono il nome di **chiavi esterne**. I vincoli di integrità referenziale riguardano i valori assunti dalle chiavi esterne nelle relazioni. Poiché una chiave esterna è utilizzata per stabilire un legame tra relazioni, il valore di una chiave esterna deve essere tenuto in stretto controllo per le operazioni di modifica, cancellazione e inserimento.

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. Dobbiamo creare una nuova relazione che chiameremo *Fornisce*, utilizzando le chiavi primarie delle due relazioni *Articolo* e *Fornitore*, che diventano chiavi esterne della nuova relazione. Scriveremo quindi:

**Articolo**(CodArt, Descrizione, Prezzo)

**Fornitore**(CodForn, Indirizzo, PartitaIva)

**Fornisce**(CodForn, CodArt)




✦ Figura A3.3
Un esempio di vincolo referenziale

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

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

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

Regole di validazione e cataloghi delle regole

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**.

## 3.1 Rappresentazione dei vincoli di integrità

Per rappresentare *vincoli di chiave primaria* sottolineiamo i relativi attributi (così come abbiamo fatto nel diagramma ER).

Per rappresentare gli *altri vincoli* ricorriamo anche qui, come già fatto per il diagramma ER, a un nostro pseudolinguaggio naturale.

Vediamo le diverse rappresentazioni dei vincoli intrarelazionali, referenziali e interrelazionali.

### 3.1.1 Rappresentazione dei vincoli intrarelazionali

Per rappresentare i **vincoli intrarelazionali** utilizziamo la seguente sintassi:

**V**<NumProg>(<NomeRelazione>): "<Espressione>"

dove:

- <NumProg> è il numero progressivo del vincolo relativo alla relazione <Nome Relazione>;
- <Espressione> è una qualsiasi espressione in pseudolinguaggio naturale che serva per specificare il vincolo.

Ad esempio, i vincoli sulla relazione *Dipendente,* precedentemente visti, possono essere espressi come:

**V1(Dipendente)**: "*StipendioLordo* > 0"
**V2(Dipendente)**: "*DataAssunzione* > *DataNascita*"
**V3(Dipendente)**: "*Trattenute* > 0"

### 3.1.2 Rappresentazione dei vincoli referenziali

Per rappresentare un **vincolo di integrità referenziale** utilizzeremo la seguente sintassi:

$$\mathbf{VR}_{\text{<Attributo1>}}(\text{<Relazione1>}) \subseteq \mathbf{VR}_{\text{<Attributo2>}}(\text{<Relazione2>})$$

Utilizziamo il simbolo di contenuto insiemistico $\subseteq$ con il seguente significato: tutti i valori dell'attributo <Attributo1> presenti nelle tuple della relazione <Relazione1> devono essere anche presenti nelle tuple della relazione <Relazione2>.

Ad esempio, per esprimere il vincolo referenziale tra *Fornitore* e *Fornisce* visto precedentemente, scriveremo:

$$\mathbf{VR}_{\text{CodForn}}(\mathbf{Fornisce}) \subseteq \mathbf{VR}_{\text{CodForn}}(\mathbf{Fornitore})$$

che significa: tutti i valori di *CodForn* presenti nelle tuple di *Fornisce* devono anche essere presenti nelle tuple di *Fornitore.* In altre parole: in *Fornisce* non possono essere presenti codici di fornitori non più presenti in *Fornitore.*

### 3.1.3 Rappresentazione dei vincoli interrelazionali

Infine, per rappresentare i **vincoli interrelazionali** che non siano vincoli referenziali ricorriamo alla seguente sintassi:

**V**(<Relazione1>, ..., <RelazioneN>): "<Espressione>"

dove:

- <Relazione1>, ..., <RelazioneN> sono i nomi delle relazioni che sono legate tramite chiavi esterne;
- <Espressione> è una qualsiasi espressione in pseudolinguaggio naturale che serve per specificare il vincolo.

Ad esempio, consideriamo le seguenti relazioni:

**Azienda**(CodAzienda, RagioneSociale, CodAttività, MatricolaDip, StipendioLordoMedio)
**Dipendente**(Matricola, Cognome, Nome, DataAssunzione, Livello, StipendioLordo, Trattenute)

Per esprimere il vincolo che impone che lo stipendio lordo di un dipendente di ottavo livello sia maggiore dello stipendio medio dell'azienda cui appartiene, scriveremo:

**V1(Dipendente, Azienda)**: "**SE Dipendente**.Livello ≥ 8 **ALLORA Dipendente**.StipendioLordo > **Azienda**.StipendioLordoMedio"

Come possiamo notare, per riferirci all'attributo di una particolare relazione, facciamo riferimento alla seguente *dot-notation*:

<NomeRelazione>.<NomeAttributo>

Nell'unità dedicata all'**SQL** vedremo come sia possibile esprimere tali vincoli con un vero e proprio linguaggio dichiarativo.

# 4 Dal diagramma ER allo schema relazionale

Nella progettazione di basi di dati abbiamo detto che il passo successivo alla progettazione concettuale è la progettazione logica tramite il modello relazionale.
Il modello relazionale mette a disposizione del progettista *solo le relazioni* per modellare i vari aspetti della realtà.
Partendo dal diagramma ER il progettista deve quindi effettuare un "mapping" delle entità e delle associazioni trasformandole in relazioni del modello relazionale.

Schema relazionale

Lo **schema relazionale** si ricava dal **diagramma ER** applicando alcune semplici **regole di derivazione** per rappresentare:

- entità e attributi;
- associazioni di tipo 1:1;
- associazione di tipo 1:N;
- associazioni di tipo N:N;
- generalizzazioni;
- aggregazioni.

Vediamo in dettaglio ognuna di queste regole.

## 4.1 Rappresentazione delle entità e degli attributi

Una entità $E$ con attributi elementari $A_1, A_2, \ldots, A_n$ è immediatamente rappresentata attraverso una relazione:

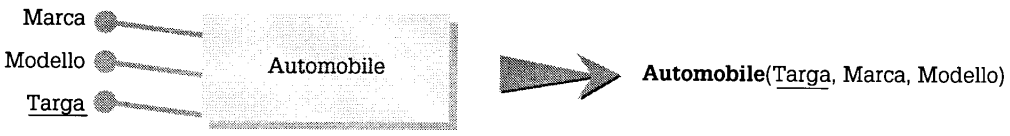
$$\mathbf{R}(A_1, A_2, \ldots, A_n)$$

dove:

- ogni *entità* diventa una relazione, rappresentata mediante una tabella;
- ogni *attributo* dell'entità diventa un attributo della relazione, rappresentato mediante una colonna della tabella;
- *l'attributo chiave* dell'entità diventa attributo chiave della relazione, *campi chiave nella tabella.*

Ad esempio:

✦ Figura A3.4
La trasformazione di un'entità in una relazione.




Attributi composti

Eventuali **attributi composti** vengono sostituiti con gli attributi componenti. Ad esempio, l'attributo composto *Indirizzo* si sostituisce con l'insieme degli attributi elementari: *Via, Città, Cap.*

Attributi multipli

Nel caso di **attributi multipli** si procederà alla normalizzazione della relazione. La normalizzazione verrà discussa in seguito.

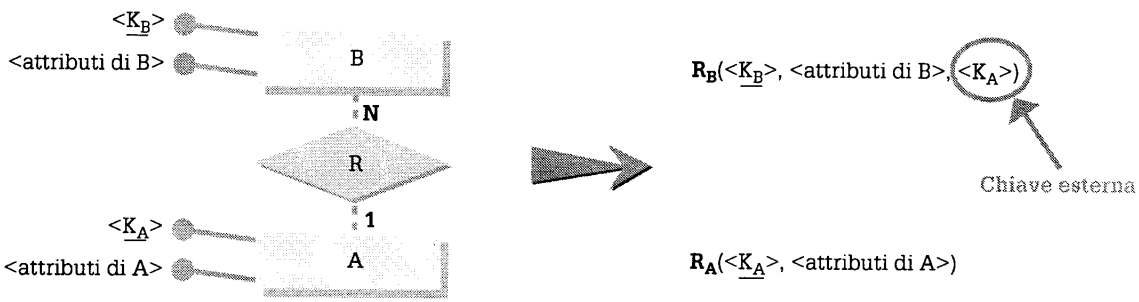
## 4.2 Rappresentazione delle associazioni binarie 1:N

Date due entità $A$ e $B$ e un'associazione $R$ di tipo *1:N* da $A$ a $B$, una soluzione classica per "mappare" tale associazione nel modello relazionale prevede di introdurre due relazioni costituite nel seguente modo:

- un relazione $R_A$ avente gli attributi di $A$;
- una relazione $R_B$ avente gli attributi di $B$ e gli attributi chiave $K_A$ di $R_A$.

Gli attributi chiave di $R_A$ presenti in $R_B$ costituiscono una cosiddetta *chiave esterna* per la relazione $R_B$. Il valore di una chiave esterna rappresenta un puntatore logico alla tupla della relazione esterna.

✦ Figura A3.5
Il mapping di un'associazione binaria 1:N




Le chiavi esterne *non devono essere sottolineate* poiché non fanno parte della chiave della relazione.

Associazione diretta e inversa parziali

Nel diagramma ER della *Fig. A3.6* abbiamo rappresentato l'associazione con *linee tratteggiate*: ciò significa che sia l'associazione diretta sia quella inversa sono **parziali**, dove per diretta intendiamo l'associazione da $B$ verso $A$.

Per trasformare un diagramma ER con associazione diretta totale (linea continua da $B$ al rombo dell'associazione), occorre specificare un vincolo di integrità referenziale che forzi l'esistenza nella relazione $R_A$ di una chiave primaria uguale alla chiave esterna $K_A$. Solo in questo modo, infatti, a ogni chiave esterna di $B$ deve corrispondere necessariamente una chiave primaria di $A$.

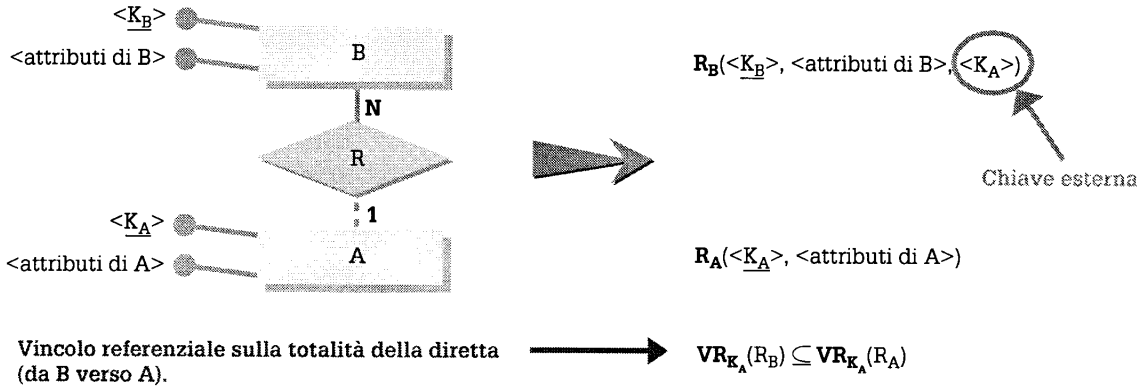
Ricorriamo allora al seguente vincolo referenziale:

$$\mathbf{VR}_{K_A}(R_B) \subseteq \mathbf{VR}_{K_A}(R_A)$$

Esso significa che tutti i valori della chiave esterna $K_A$ di $R_B$ devono anche essere presenti nell'attributo $K_A$ di $R_A$.

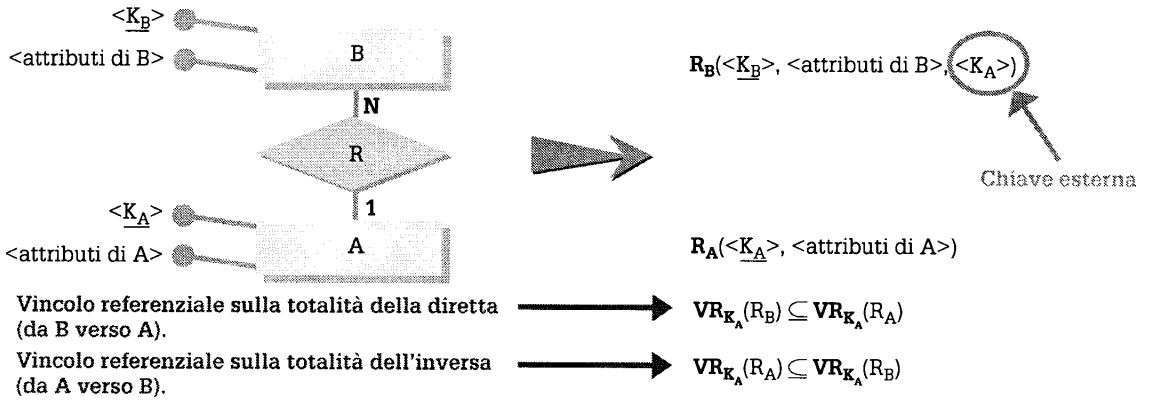
Possiamo anche dire che non si può inserire una tupla in una relazione con il valore di una chiave esterna non presente nella corrispondente relazione".

In questo modo si assicura la totalità dell'**associazione diretta** da $B$ verso $A$. Potremmo riscrivere la precedente trasformazione nel seguente modo:




✦ Figura A3.6
La trasformazione di un'associazione 1:N con diretta totale e inversa parziale

A questo punto, per assicurare anche la totalità dell'inversa, scriveremo:




✦ Figura A3.7
La trasformazione di un'associazione 1:N con diretta e inversa totali

Come possiamo notare, abbiamo aggiunto un altro vincolo referenziale:

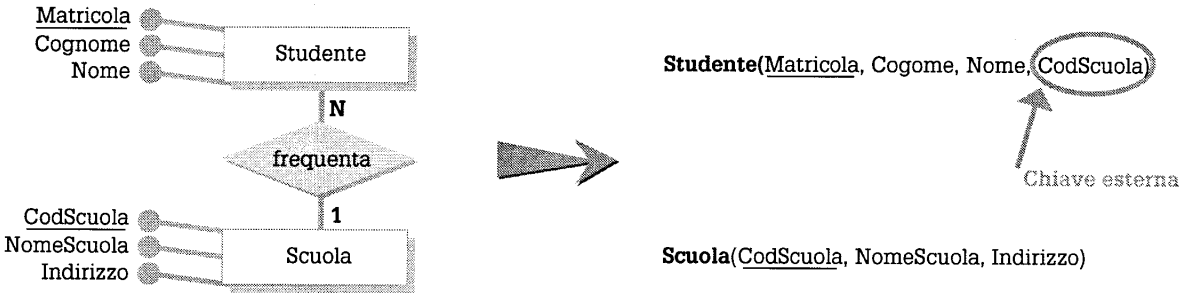
$$\mathbf{VR}_{K_A}(R_A) \subseteq \mathbf{VR}_{K_A}(R_B)$$

che assicura la totalità dell'inversa. Infatti significa che tutti i valori dell'attributo $K_A$ di $R_A$ (chiave primaria) devono anche essere presenti nell'attributo $K_A$ di $R_B$ (chiave esterna).

Possiamo anche dire che "non si può cancellare una tupla la cui chiave primaria compaia in almeno una tupla di altre relazioni come chiave esterna".

## Esempio

Consideriamo il seguente diagramma ER:




✦ Figura A3.8
Un esempio di mapping di associazione con diretta e inversa totali

Nella relazione *Studente* l'attributo *CodScuola* è una chiave esterna. Essa viene utilizzata per rappresentare l'associazione 1:N esistente tra *Studente* e *Scuola*.
Il vincolo significa che tutti i valori di *Scuola* presenti nelle tuple di *Studente* sono presenti nelle tuple di *Scuola*.

In questo esempio sia la diretta che l'inversa sono totali: infatti non possiamo inserire un alunno senza specificare la scuola cui appartiene, così come non possiamo cancellare una scuola senza cancellare tutti gli alunni che ne fanno parte.

Il meccanismo delle chiavi esterne comporta inevitabilmente una duplicazione di informazioni, in quanto la chiave esterna deve essere presente nella relazione. Se lo spazio richiesto in memoria per la rappresentazione degli attributi che compongono tale chiave è elevato, lo spreco di memoria può essere considerevole. In questi casi conviene introdurre una **chiave artificiale** formata da un unico attributo che viene aggiunto alla relazione esterna (della quale diventa la nuova chiave primaria). Un esempio di chiave artificiale è un **codice progressivo**. Per rappresentare associazioni *1:N* parziali basta specificare come indefinito il valore della chiave esterna nelle tuple di $R_B$ non associate ad alcuna tupla di $R_A$ (si può ricorrere al valore speciale **null**).

Chiave artificiale

## 4.3 Rappresentazione delle associazioni binarie 1:1

Le **associazioni binarie 1:1** rappresentano un caso particolare delle associazioni *1:N* e seguono quindi le stesse regole appena viste. Un vincolo di integrità nelle associazioni 1:1 significa che solo una chiave esterna deve corrispondere alla chiave primaria, e si esprime con:

$$\mathbf{VR}_{K_A}(R_B) = \mathbf{VR}_{K_A}(R_A)$$

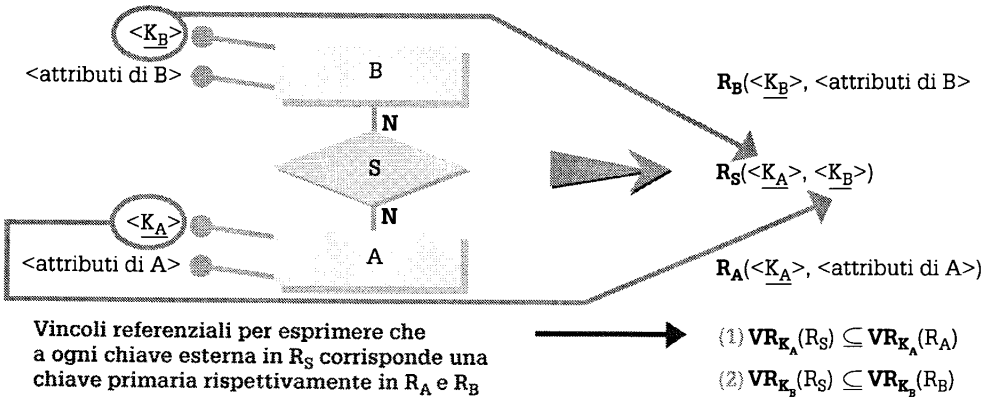
Si tende a trasformare le due entità con associazione 1:1 in un'unica relazione ottenuta dalla fusione delle due, che possiede gli attributi dell'una e dell'altra. Altre volte si conservano le due entità separate per motivi di efficienza. Se si accede a un'unità più frequentemente dell'altra, conviene avere entità "snelle", cioè col minor numero possibile di attributi.

## 4.4 Rappresentazione delle associazioni binarie N:N

Date due entità *A* e *B* e un'associazione *S* di tipo *N:N* da *A* a *B*, per rappresentare tale associazione nel modello relazionale occorre introdurre:

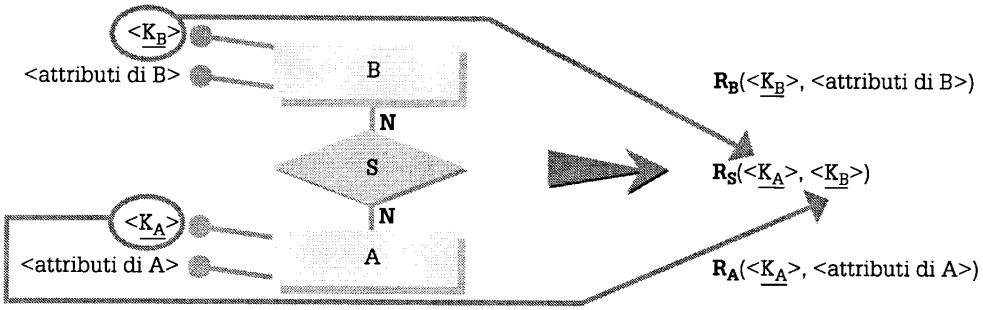
- una relazione $R_A$ avente gli attributi di *A*;
- un relazione $R_B$ avente gli attributi di *B*;
- una relazione $R_S$ avente gli attributi chiave $K_A$ di $R_A$ e gli attributi chiave $K_B$ di $R_B$ (quindi come minimo $R_S$ ha due attributi);
- un vincolo di integrità che assicuri che a ogni chiave $K_A$ ($K_B$) presente in $R_S$ corrisponda una chiave $K_A$ ($K_B$) primaria della relazione $R_A$ ($R_B$).

Ogni tupla di $R_S$ rappresenta una coppia dell'associazione binaria *S*.




✦ Figura A3.9
Il mapping di un'associazione N:N

Da notare che le chiavi esterne $K_A$ e $K_B$ della relazione $R_S$ sono sottolineate in quanto sono anche chiavi interne della relazione $R_S$. Poiché le associazioni diretta e inversa possono essere totali, allora dobbiamo introdurre i seguenti vincoli referenziali sulla totalità della diretta (da *B* verso *A*) e dell'inversa (da *A* verso *B*).

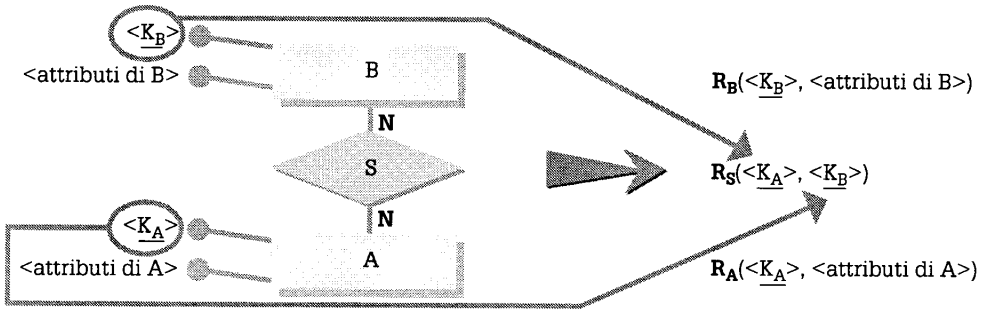



Vincoli referenziali per esprimere che a ogni chiave esterna in $R_S$ corrisponde una chiave primaria rispettivamente in $R_A$ e $R_B$ → (1) $VR_{K_A}(R_S) \subseteq VR_{K_A}(R_A)$; (2) $VR_{K_B}(R_S) \subseteq VR_{K_B}(R_B)$

Vincolo referenziale sulla totalità della diretta (da B verso A). → (3) $VR_{K_B}(R_B) \subseteq VR_{K_B}(R_S)$

✦ Figura A3.10
La rappresentazione di un'associazione N:N con diretta totale

L'ultimo vincolo dice che a ogni tupla presente in $R_B$ deve corrispondere una tupla in $R_S$.



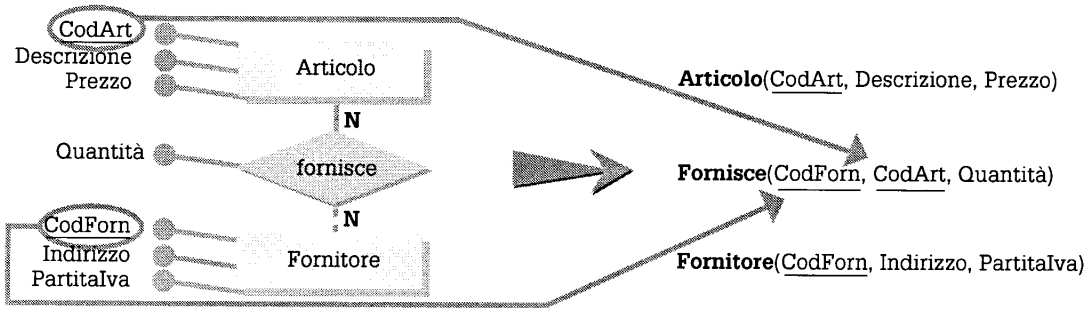

Vincoli referenziali per esprimere che a ogni chiave esterna in $R_S$ corrisponde una chiave primaria rispettivamente in $R_A$ e $R_B$ → (1) $VR_{K_A}(R_S) \subseteq VR_{K_A}(R_A)$; (2) $VR_{K_B}(R_S) \subseteq VR_{K_B}(R_B)$

Vincolo referenziale sulla totalità della diretta (da B verso A). → (3) $VR_{K_B}(R_B) \subseteq VR_{K_B}(R_S)$

Vincolo referenziale sulla totalità dell'inversa (da A verso B). → (4) $VR_{K_A}(R_A) \subseteq VR_{K_A}(R_S)$

✦ Figura A3.11
La rappresentazione di un'associazione N:N con diretta e inversa totali

L'ultimo vincolo dice che a ogni tupla presente in $R_A$ deve corrispondere una tupla in $R_S$. Vediamo un esempio di mapping di un'associazione *N:N* con la sola diretta totale:




Non può esistere il codice di un articolo non presente in Articolo → (1) $VR_{CodArt}(\text{Fornisce}) \subseteq VR_{CodArt}(\text{Articolo})$

Non può esistere il codice di un fornitore non presente in Fornitore → (2) $VR_{CodForn}(\text{Fornisce}) \subseteq VR_{CodForn}(\text{Fornitore})$

Ogni CodArt presente in Articolo deve avere un corrispondente CodArt in Fornisce → (3) $VR_{CodArt}(\text{Articolo}) \subseteq VR_{CodArt}(\text{Fornisce})$

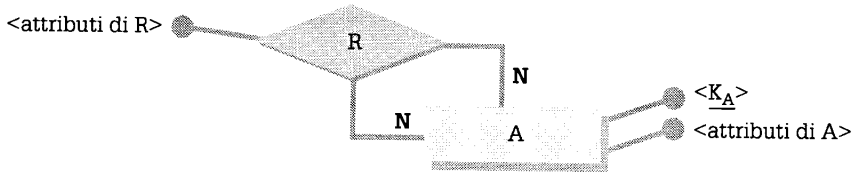
✦ Figura A3.12
Un esempio di mapping di un'associazione N:N

Come possiamo notare, l'inversa è parziale, ovvero può esistere un fornitore che non fornisce alcun articolo. Il metodo appena descritto può essere facilmente generalizzato, sia per rappresentare attributi dell'associazione (basta aggiungerli alla relazione $R_S$) sia per rappresentare associazioni non binarie (ad esempio: *ternarie di tipo N:M:P*) è infatti sufficiente introdurre le chiavi esterne in numero pari al grado dell'associazione. Inoltre anche le **associazioni 1:N** possono essere rappresentate come un caso particolare delle associazioni *N:N*, seguendo quindi le regole appena viste.

## 4.5 Rappresentazione di associazioni su una stessa entità

Un caso particolare di associazioni *1:N* o *N:N* è quello in cui l'entità di partenza è uguale all'entità di arrivo. Consideriamo il seguente diagramma ER relativo a un'associazione *N:N* su una stessa entità. Anche qui valgono le medesime considerazioni fatte per la totalità della diretta e dell'inversa. Devono quindi essere specificati i vincoli di integrità. Supponiamo che la nostra associazione sia totale. Scriveremo:

✦ Figura A3.13
La rappresentazione di un'associazione su una stessa entità




Per rappresentare tale associazione nel modello relazionale si opera allo stesso modo visto per le associazioni su più entità. Dipende quindi dal tipo di associazione. In questo caso, essendo *N:N* avremo:

✦ Figura A3.14
I vincoli per un'associazione su una stessa entità

**$R_A$**(<$\underline{K_A}$>, <attributi di A>)

**$R_R$**(<$\underline{K1_A}$>, <$\underline{K2_A}$>, <attributi di R>)

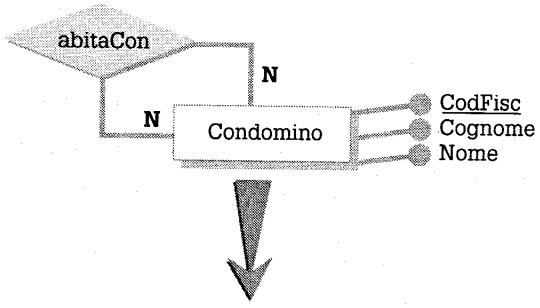
$$\mathbf{VR}_{\mathbf{K1_A}}(R_R) \subseteq \mathbf{VR}_{\mathbf{K_A}}(R_A)$$
$$\mathbf{VR}_{\mathbf{K2_A}}(R_R) \subseteq \mathbf{VR}_{\mathbf{K_A}}(R_A)$$
$$\mathbf{VR}_{\mathbf{K_A}}(R_A) \subseteq \mathbf{VR}_{\mathbf{K1_A}}(R_R)$$
$$\mathbf{VR}_{\mathbf{K_A}}(R_A) \subseteq \mathbf{VR}_{\mathbf{K2_A}}(R_R)$$

La chiave <$\underline{K_A}$> comparirà in due colonne <$\underline{K1_A}$>, <$\underline{K2_A}$> della tabella che rappresenta l'associazione *R*.

### Esempio

Consideriamo il seguente diagramma ER relativo ai condomini di un condominio:

✦ Figura A3.15
Un esempio di mapping di un'associazione sulla stessa entità



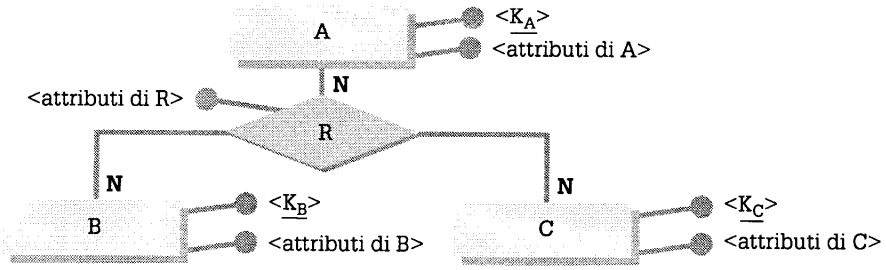

**Condomino**(CodFisc, Cognome, Nome)

**AbitaCon**(CodFisc1, CodFisc2)

$$\mathbf{VR}_{\mathbf{CodFisc1}}(\text{AbitaCon}) \subseteq \mathbf{VR}_{\mathbf{CodFisc}}(\text{Condomino})$$
$$\mathbf{VR}_{\mathbf{CodFisc2}}(\text{AbitaCon}) \subseteq \mathbf{VR}_{\mathbf{CodFisc}}(\text{Condomino})$$
$$\mathbf{VR}_{\mathbf{CodFisc}}(\text{Condomino}) \subseteq \mathbf{VR}_{\mathbf{CodFisc1}}(\text{AbitaCon})$$
$$\mathbf{VR}_{\mathbf{CodFisc}}(\text{Condomino}) \subseteq \mathbf{VR}_{\mathbf{CodFisc2}}(\text{AbitaCon})$$

## 4.6 Rappresentazione di associazioni non binarie

Supponiamo di avere il seguente diagramma ER relativo a un'associazione totale su tre entità:




✦ Figura A3.16
Il mapping di associazioni non binarie

Per rappresentare tale associazione nel modello relazionale, si opera come abbiamo visto per le associazioni binarie; dipende quindi dal tipo di associazione. In questo caso essendo *N:N* avremo:

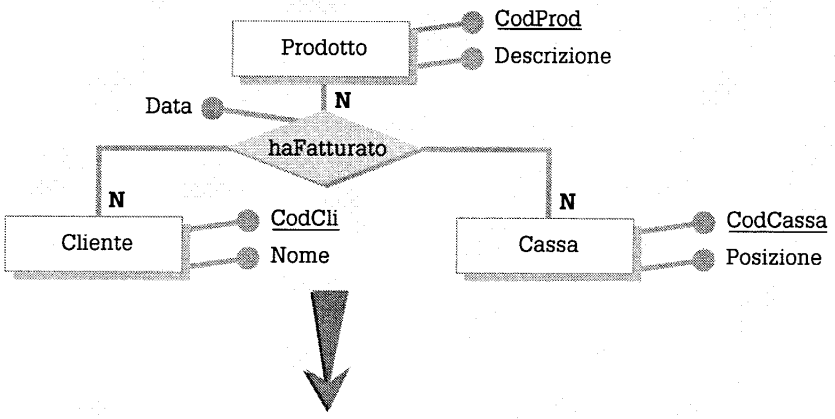
$\mathbf{R_A}(\langle \underline{K_A} \rangle, \langle \text{attributi di A} \rangle)$
$\mathbf{R_B}(\langle \underline{K_B} \rangle, \langle \text{attributi di B} \rangle)$
$\mathbf{R_C}(\langle \underline{K_C} \rangle, \langle \text{attributi di C} \rangle)$
$\mathbf{R_R}(\langle \underline{K_A} \rangle, \langle \underline{K_B} \rangle, \langle \underline{K_C} \rangle, \langle \text{attributi di R} \rangle)$

$\mathbf{VR_{K_A}}(R_R) \subseteq \mathbf{VR_{K_A}}(R_A)$
$\mathbf{VR_{K_B}}(R_R) \subseteq \mathbf{VR_{K_B}}(R_B)$
$\mathbf{VR_{K_C}}(R_R) \subseteq \mathbf{VR_{K_C}}(R_C)$
$\mathbf{VR_{K_A}}(R_A) \subseteq \mathbf{VR_{K_A}}(R_R)$
$\mathbf{VR_{K_B}}(R_B) \subseteq \mathbf{VR_{K_B}}(R_R)$
$\mathbf{VR_{K_C}}(R_C) \subseteq \mathbf{VR_{K_C}}(R_R)$

✦ Figura A3.17
I vincoli referenziali nel mapping di associazioni non binarie

**Esempio**

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



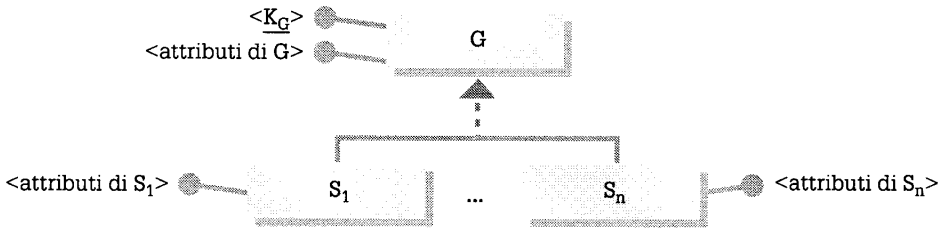

✦ Figura A3.18
Un esempio di mapping di un'associazione non binaria

**Prodotto**(CodProd, Descrizione)
**Cliente**(CodCli, Nome)
**Cassa**(CodCassa, Posizione)
**HaFatturato**(CodProd, CodCli, CodCassa, Data)

$\mathbf{VR_{CodProd}}(\text{HaFatturato}) \subseteq \mathbf{VR_{CodProd}}(\text{Prodotto})$
$\mathbf{VR_{CodCli}}(\text{HaFatturato}) \subseteq \mathbf{VR_{CodCli}}(\text{Cliente})$
$\mathbf{VR_{CodCassa}}(\text{HaFatturato}) \subseteq \mathbf{VR_{CodCassa}}(\text{Cassa})$
$\mathbf{VR_{CodProd}}(\text{Prodotto}) \subseteq \mathbf{VR_{CodProd}}(\text{HaFatturato})$
$\mathbf{VR_{CodCli}}(\text{Cliente}) \subseteq \mathbf{VR_{CodCli}}(\text{HaFatturato})$
$\mathbf{VR_{CodCassa}}(\text{Cassa}) \subseteq \mathbf{VR_{CodCassa}}(\text{HaFatturato})$

## 4.7 Rappresentazione delle associazioni di generalizzazione

Consideriamo un'entità generalizzazione padre *G* e le entità specializzazioni o figlie $S_1, \ldots, S_n$, così come mostrato nella seguente figura:




✦ Figura A3.19
Il mapping di associazioni di generalizzazione

per rappresentare tale associazione nel modello relazionale possiamo seguire tre strade diverse:

- **accorpamento** delle figlie nel padre;
- **accorpamento** del padre nelle figlie;
- **sostituzione** della generalizzazione con associazioni *1:1*.

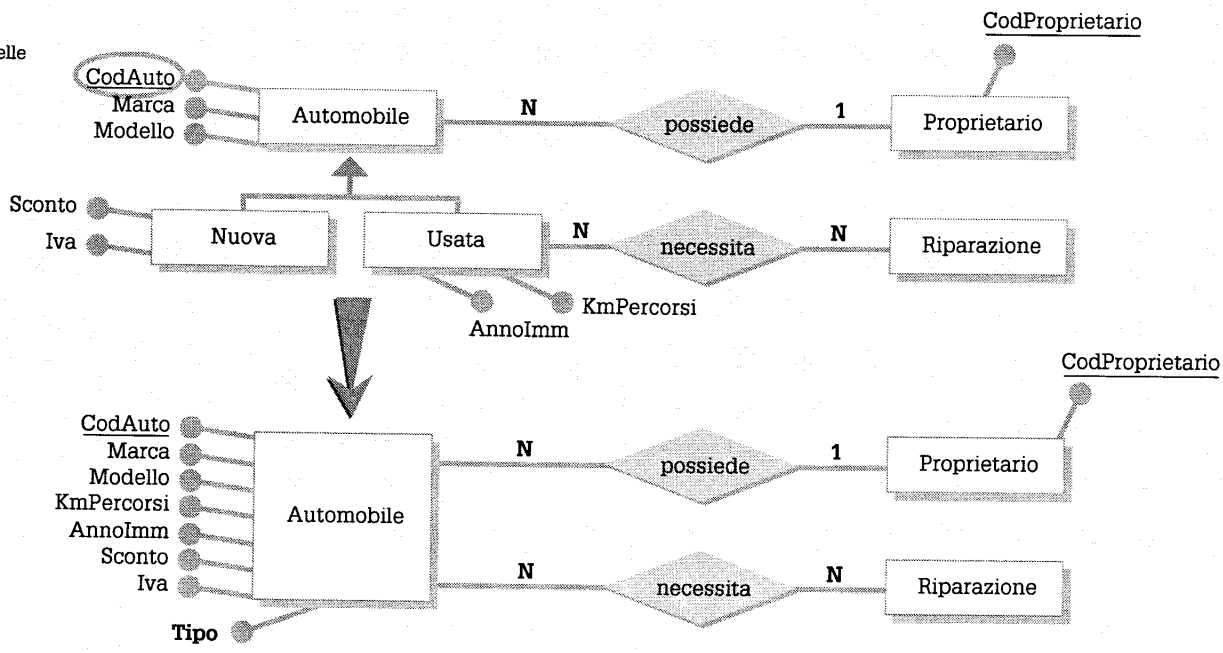
Accorpamento delle figlie nel padre

Nell'**accorpamento delle figlie nel padre**, le entità $S_1$ e $S_n$ vengono eliminate e i loro attributi e le partecipazioni ad associazioni vengono aggiunti all'entità padre. Al padre viene inoltre aggiunto un altro attributo, che serve per distinguere il **tipo di ogni tupla** del padre, ovvero per distinguere se ciascuna tupla appartiene a $S_1, \ldots, S_n$.

**Esempio**

Consideriamo la generalizzazione *Automobile* composta dalle entità figlie *Usate* e *Nuove*, che rappresentano rispettivamente auto usate o auto nuove. Possiamo accorpare le figlie nel padre nel seguente modo:

✦ Figura A3.20
Un esempio di accorpamento delle figlie nel padre




**Automobile**(CodAuto, Marca, Modello, KmPercorsi, AnnoImm, Sconto, Iva, **Tipo**, CodProprietario)

dove notiamo:

- che è stato aggiunto l'attributo *Tipo*;
- che è stata aggiunta la chiave esterna *CodProprietario* relativa alla chiave primaria delle entità *Proprietario*, poiché l'associazione *possiede* è di tipo *1:N*.
- che alcuni attributi possono assumere valori nulli perché non significativi per alcune tuple: ad esempio, un'auto nuova non avrà valori per gli attributi *KmPercorsi* e *AnnoImm*.

Deduciamo quindi che tale traduzione è conveniente quando le operazioni sulla base di dati non fanno molta distinzione tra tuple di una figlia o di un'altra e tra gli attributi di una figlia o di un'altra. In questo caso avremmo minimizzato gli accessi alla memoria, anche se con un maggiore spreco di memoria, dovuto alla presenza di valori nulli per alcuni attributi.

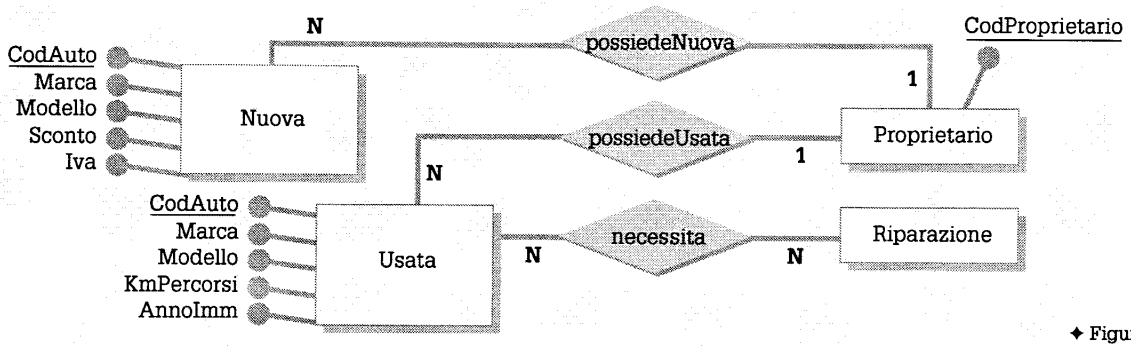
Accorpamento del padre nelle figlie

Nell'**accorpamento del padre nelle figlie**, l'entità padre $G$ viene eliminata e i suoi attributi e le associazioni alle quali il padre partecipa vengono aggiunti alle figlie $S_1$ e $S_n$.

Questa traduzione è possibile solo se la generalizzazione è totale, cioè se ogni tupla di $G$ è una tupla o di $S_1$, o di $S_2$, ..., o di $S_n$.

**Esempio**

Considerando l'esempio precedente avremmo la seguente traduzione:




✦ Figura A3.21
Un esempio di accorpamento del padre nelle figlie

**Nuova**(CodAuto, Marca, Modello, Sconto, Iva, CodProprietario)
**Usata**(CodAuto, Marca, Modello, KmPercorsi, AnnoImm, CodProprietario)

dove notiamo che:

- sono state aggiunte chiavi esterne *CodProprietario* per le associazioni *possiedeNuova* e *possiedeUsata*;
- la *traduzione* è possibile in quanto un'automobile o è nuova o è usata (ovvero l'aggregazione è totale).

Deduciamo che tale traduzione è conveniente se ci sono operazioni sulla base di dati che si riferiscono solo a tuple di una figlia o solo a tuple dell'altra figlia, facendo distinzione tra le entità figlie. In questo caso abbiamo un risparmio di memoria rispetto alla prima soluzione, in quanto non compaiono attributi dai valori nulli e, come vedremo tra poco, abbiamo un vantaggio nel numero degli accessi rispetto alla successiva soluzione. L'ultima possibile traduzione è quella della **sostituzione della generalizzazione con associazioni 1:1**. In questa traduzione si trasforma la generalizzazione in tante associazioni *1:1* quante sono le entità figlie.

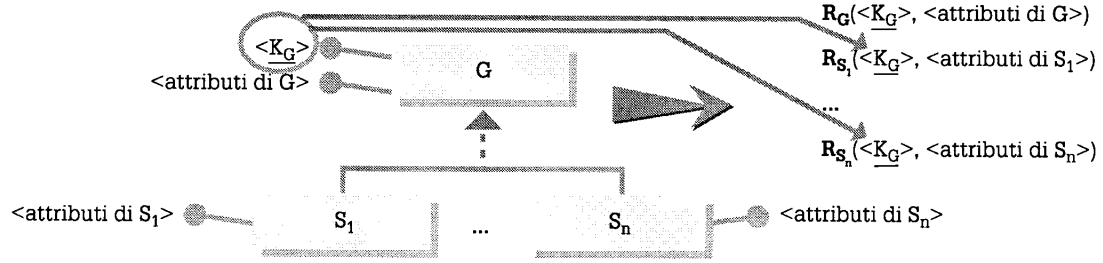
Sostituzione della generalizzazione con associazioni 1:1

In questo modo non ci sono trasferimenti di attributi tra entità (cioè non si accorpano le figlie con i padri o viceversa). Le entità figlie sono identificate esternamente, o meglio la chiave dell'entità padre viene utilizzata nelle figlie sia come chiave primaria che come chiave esterna. Occorre pertanto introdurre:

- una relazione $R_G$, avente gli attributi di $G$ e gli attributi chiave primaria $K_G$;
- una relazione $R_{S_1}$, avente gli attributi chiave $K_G$ di $R_G$ come attributi chiave primaria, poi gli altri attributi di $S_1$ e infine gli eventuali attributi per le chiavi esterne;
- ...
- una relazione $R_{S_N}$, avente gli attributi chiave $K_G$ di $R_G$ come attributi chiave primaria, poi gli altri attributi di $S_N$ e infine gli eventuali attributi per le chiavi esterne;
- un vincolo che dica: "a ogni tupla di $G$ non possono corrispondere contemporaneamente più tuple in entità figlie distinte" (*proprietà di esclusività della generalizzazione*).

Proprietà di esclusività della generalizzazione

Osserviamo il seguente schema:



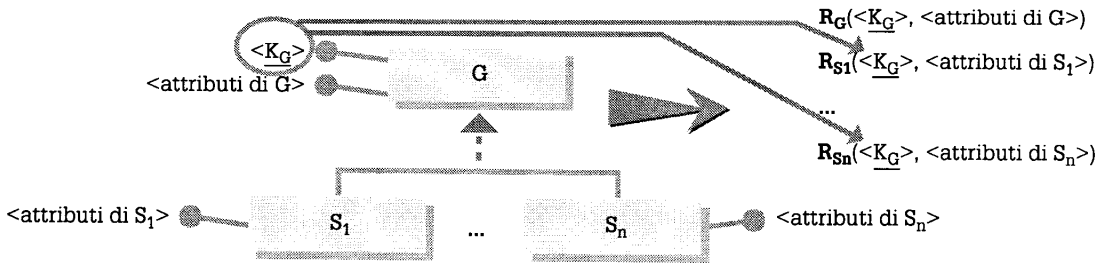

✦ Figura A3.22
Una sostituzione della generalizzazione con associazioni 1:1

Vincolo referenziale per esprimere che a ogni tupla di $G$ non possono corrispondere contemporaneamente più tuple in entità figlie distinte.

$$VR_{K_G}(S_1) \subseteq VR_{K_G}(G)$$
**XOR**
....
**XOR**
$$VR_{K_G}(S_n) \subseteq VR_{K_G}(G)$$

Se la generalizzazione è **totale**, va introdotto un altro vincolo che dica che "a ogni tupla di G deve corrispondere almeno una tupla delle entità figlie", cioè riassumendo:

✦ Figura A3.23
Una sostituzione di una generalizzazione totale con associazioni 1:1




**Vincolo referenziale per esprimere che a ogni tupla di *G* non possono corrispondere contemporaneamente più tuple in entità figlie distinte.**

$$VR_{K_G}(S_1) \subseteq VR_{K_G}(G)$$
**XOR**
.....
**XOR**
$$VR_{K_G}(S_n) \subseteq VR_{K_G}(G)$$

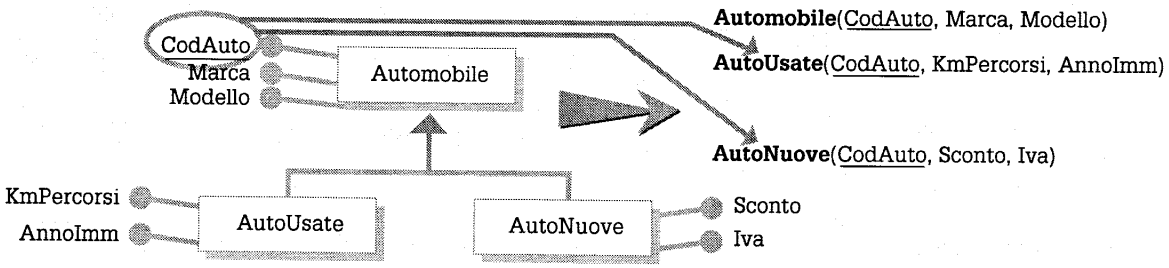
**Vincolo referenziale per esprimere la totalità della generalizzazione, cioè a ogni tupla di *G* deve corrispondere almeno una tupla in una entità figlia**

$$VR_{K_G}(G) \subseteq VR_{K_G}(S_1)$$
**OR**
.....
**OR**
$$VR_{K_G}(G) \subseteq VR_{K_G}(S_n)$$

**Esempio**

Considerando sempre il nostro esempio delle automobili, avremo:

✦ Figura A3.24
Un esempio di sostituzione con associazioni 1:1




**A una tupla di Automobile non possono corrispondere contemporaneamente tuple di AutoUsate e tuple di AutoNuove**

$$VR_{CodAuto}(\text{AutoUsate}) \subseteq VR_{CodAuto}(\text{Automobile})$$
**XOR**
$$VR_{CodAuto}(\text{AutoNuove}) \subseteq VR_{CodAuto}(\text{Automobile})$$

**(Totalità della generalizzazione)**
**A una tupla di automobile deve corrispondere almeno una tupla in AutoUsate o in AutoNuove**

$$VR_{CodAuto}(\text{Automobile}) \subseteq VR_{CodAuto}(\text{AutoUsate})$$
**OR**
$$VR_{CodAuto}(\text{Automobile}) \subseteq VR_{CodAuto}(\text{AutoNuove})$$

## 4.8 Rappresentazione delle associazioni di aggregazione

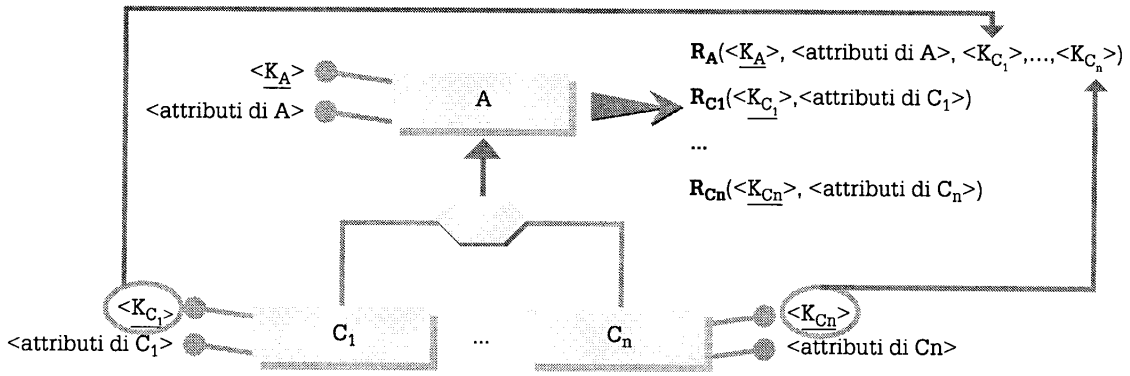
Data un'associazione di aggregazione di $A$ e $C_1$, $C_2$,..., $C_n$ entità componenti, per rappresentare tale associazione nel modello relazionale occorre introdurre:

- una relazione $R_{C_1}$ avente gli attributi di $C_1$;
- ...
- una relazione $R_{C_n}$ avente gli attributi di $C_n$;

- una relazione $R_A$ avente gli attributi chiave $K_A$ di $A$, gli altri attributi di $A$ e poi:
  - gli attributi chiave $K_{C_1}$ di $C_1$;
  - ...
  - gli attributi chiave $K_{C_n}$ di $C_n$;
- un vincolo referenziale che ci dice che "le chiavi esterne presenti in $R_A$ devono avere le corrispondenti chiavi primarie presenti in $R_{C_1}$, $R_{C_n}$".

Altri vincoli di integrità a seconda che l'aggregazione sia *lasca* o *stretta*.

Vediamo i due casi, cominciando dall'*aggregazione lasca*.




✦ Figura A3.25
Il mapping di un'aggregazione lasca

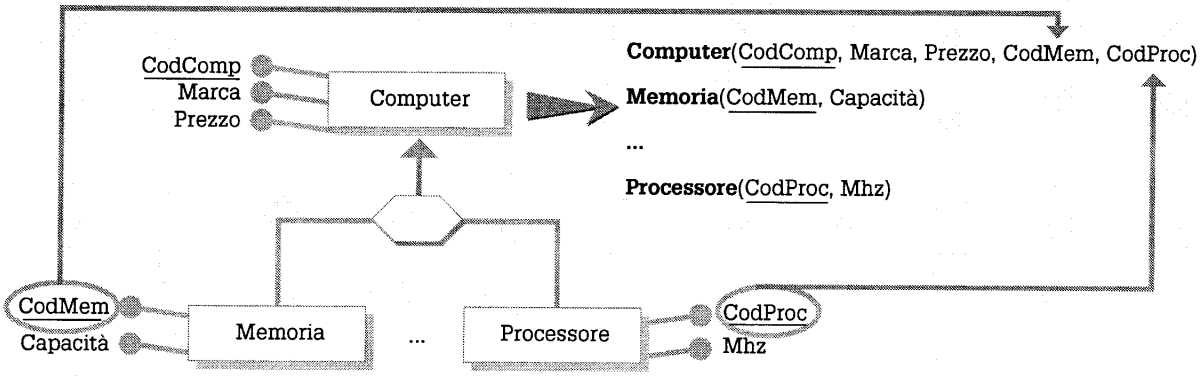
**Vincolo referenziale per esprimere che ogni tupla di $R_A$ deve avere le corrispondenti chiavi primarie presenti in $R_{C_1}$, $R_{C_n}$ (totalità dell'aggregazione)**

$$VR_{K_{C_1}}(A) \subseteq VR_{K_{C_1}}(C_1)$$
**AND**
...
**AND**
$$VR_{K_{C_n}}(A) \subseteq VR_{K_{C_n}}(C_n)$$

In questo caso i vincoli di integrità esprimono il fatto che alle chiavi esterne di $A$ devono corrispondere chiavi primarie di $C_1$, ..., $C_n$.

**Esempio**

Consideriamo il seguente diagramma ER:



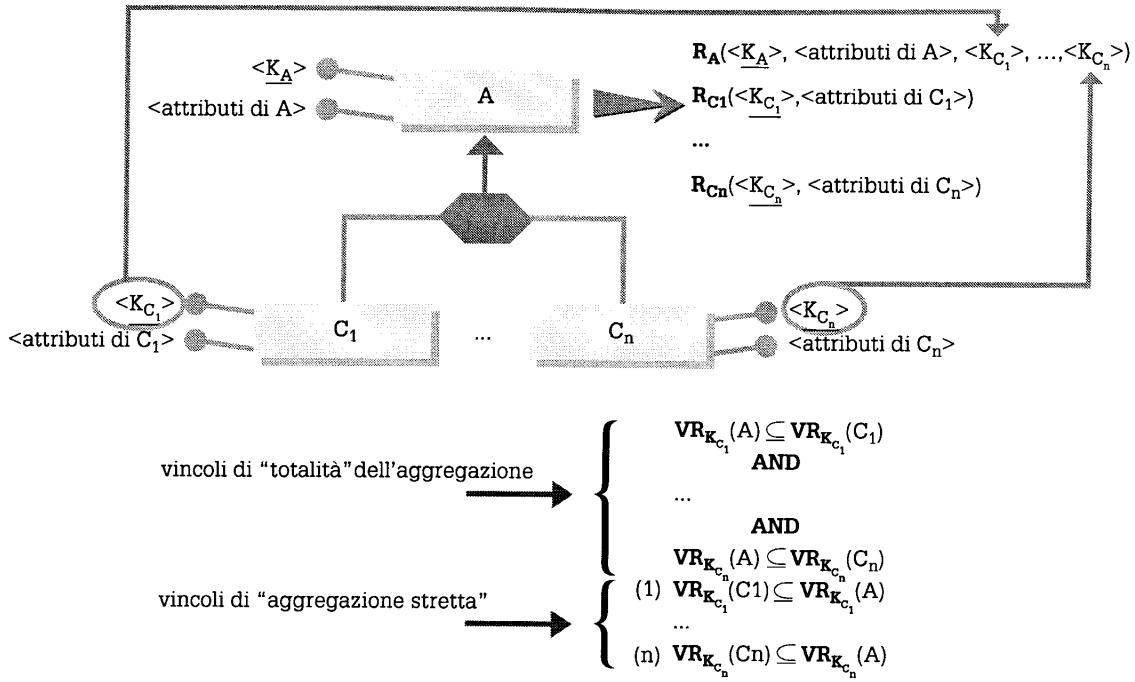

**Ogni chiave esterna presente in *Computer* deve essere chiave primaria delle rispettive relazioni figlie.**

$$VR_{CodMem}(Computer) \subseteq VR_{CodMem}(Memoria)$$
**AND**
...
**AND**
$$VR_{CodProc}(Computer) \subseteq VR_{CodProc}(Processore)$$

✦ Figura A3.26
Un esempio di mapping di un'aggregazione lasca

I vincoli si traducono in: "se *cancelliamo/inseriamo* un computer, dobbiamo anche *cancellare/inserire* una memoria e un processore".
Ma possiamo *inserire/cancellare* una memoria o un processore (come pezzi di ricambio) senza dover necessariamente *inserire/cancellare* un computer.

Proseguiamo con l'*aggregazione stretta*:



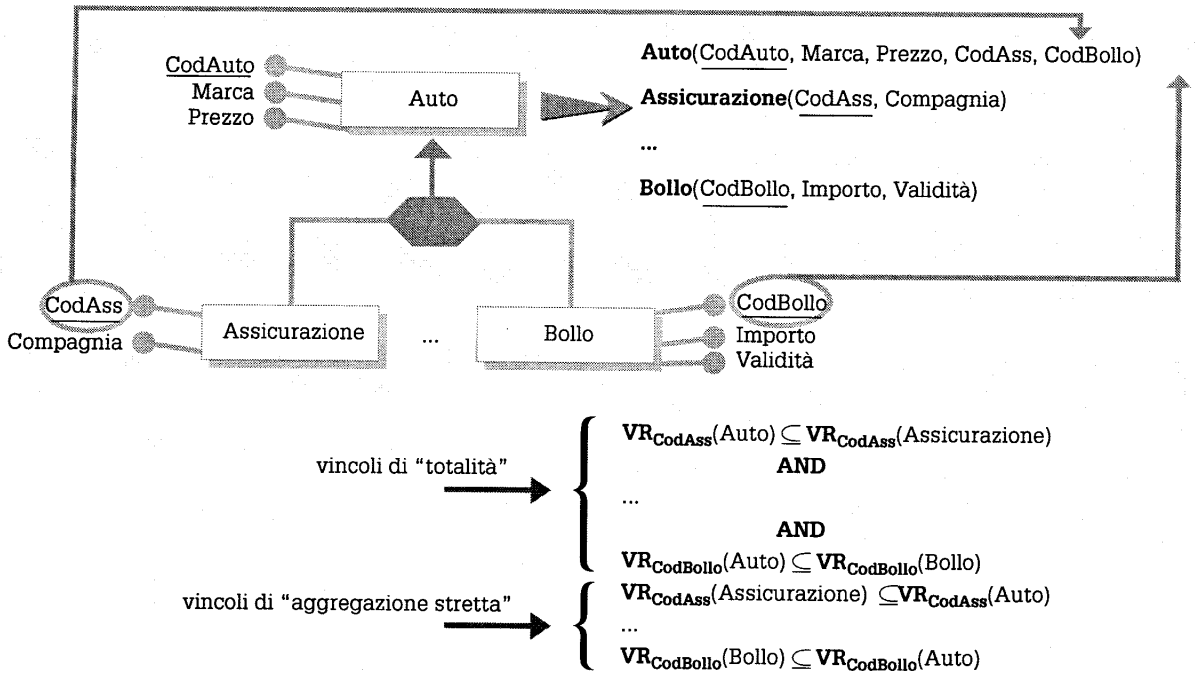

✦ Figura A3.27
Il mapping di un'aggregazione stretta

In questo caso, i vincoli di integrità (1) ... (n) esprimono i seguenti fatti:

- alle chiavi esterne di *A* devono corrispondere chiavi primarie di $C_1, \ldots, C_n$ (totalità);
- alle chiavi primarie di $C_1, \ldots, C_n$ devono corrispondere chiavi esterne di A (aggregazione stretta).

## Esempio

Consideriamo il seguente diagramma ER:




✦ Figura A3.28
Un esempio di mapping di un'aggregazione stretta

I vincoli si traducono in:

- se *cancelliamo/inseriamo* un'automobile dobbiamo anche *cancellare/inserire* la sua assicurazione e il suo bollo;
- se *cancelliamo/inseriamo* un nuovo bollo e una nuova assicurazione dobbiamo necessariamente *inserire* un'auto cui siano collegati.

Da notare che il connettivo AND è superfluo ma lo abbiamo inserito per chiarezza espositiva.

## 4.9 Una semplice formula

Possiamo avere subito un riscontro della "bontà" del nostro diagramma ER tramite la seguente formula che, dato un diagramma ER, serve per calcolare il numero di relazioni che ne derivano.

**Numero di relazioni = Numero di entità – Numero di associazioni 1:1 + Numero di associazioni N:N**

Come possiamo notare, non compaiono le associazioni *1:N* in quanto sono inglobate nelle entità che sono legate. Inoltre viene sottratto il numero di associazioni *1:1*, poiché, generalmente, le due entità legate vengono fuse in una sola relazione. Per le generalizzazioni non occorre inserire altro, visto che le abbiamo implementate con associazioni *1:1* interne alle entità figlie. Anche per le aggregazioni non occorre inserire altro, poiché sono implementate con associazioni *1:N* interne alle entità componenti.

## 4.10 L'esempio della concessionaria di automobili: derivazione dal diagramma ER

Vediamo un altro esempio di utilizzo di derivazione del modello relazionale dal diagramma ER e di utilizzo del calcolo relazionale, riprendendo il diagramma ER relativo alla concessionaria di automobili (introdotto nell'unità precedente).

Applicando al diagramma ER le regole di derivazione, otteniamo il seguente schema relazionale:

**SCHEMA RELAZIONALE** Concessionaria di automobili

*Traduzione delle entità*

**Automobile**(CodAuto, Marca, Modello, Targa, Prezzo, CodFiscale, CodMotore, CodCarr, CodRuota)
**Proprietario**(CodFiscale, Cognome, Nome)
**Motore**(CodMotore, Cilindrata, TipoCarburante)
**Carrozzeria**(CodCarr, NumTelaio)
**Ruota**(CodRuota, Diametro, Larghezza)
**AutoUsata**(CodAuto, AnnoImm, KmPercorsi)
**AutoNuova**(CodAuto, AnniGaranzia)
**Riparazione**(CodRip, Tipo, Gravità, Spesa)
**Optional**(CodOpt, Descrizione, Prezzo)

*Traduzione delle associazioni*

**Necessita**(CodAuto, CodRip)
**ÈDotata**(CodAuto, CodOpt)

*Alcuni vincoli di integrità*

- Vincoli di **dominio:**

  V1(**Automobile**): "Prezzo > 0"
  V2(**AutoUsata**): "AnnoImm > 1990"
  V3(**AutoUsata**): "KmPercorsi < 300 000"
  V4(**Riparazione**): "Spesa > 0"

- Vincolo **referenziale**. Per esprimere la totalità dell'associazione *acquista* scriveremo:

  $VR_{CodFiscale}$(**Automobile**) ⊆ $VR_{CodFiscale}$(**Proprietario**)
  $VR_{CodFiscale}$(**Proprietario**) ⊆ $VR_{CodFiscale}$(**Automobile**)

- Vincolo **referenziale**. Per esprimere la totalità dell'inversa dell'associazione *ÈDotata* scriveremo:

$VR_{CodOpt}$(**Optional**) ⊆ $VR_{CodOpt}$(**ÈDotata**)
$VR_{CodAuto}$(**ÈDotata**) ⊆ $VR_{CodAuto}$(**AutoNuova**)

oltre al vincolo sulle chiavi esterne:

$VR_{CodOpt}$(**ÈDotata**) ⊆ $VR_{CodOpt}$(**Optional**)

- Vincolo **referenziale**. Per esprimere la totalità dell'inversa dell'associazione *necessita* scriveremo:

VRCodRip(**Riparazione**) ⊆ VRCodRip(**Necessita**)

che esprime l'impossibilità di inserire una riparazione senza collegarla alla relativa auto usata.

- Vincoli **referenziali**. Per esprimere il vincolo legato all'associazione di generalizzazione scriveremo:

$VR_{CodAuto}$(AutoUsate) ⊆ $VR_{CodAuto}$(Automobile)
**XOR**
$VR_{CodAuto}$(AutoNuove) ⊆ $VR_{CodAuto}$(Automobile)

$VR_{CodAuto}$(Automobile) ⊆ $VR_{CodAuto}$(AutoUsate)
**OR**
$VR_{CodAuto}$(Automobile) ⊆ $VR_{CodAuto}$(AutoNuove)

- Vincoli **referenziali**. Per esprimere il vincolo legato all'unica associazione per aggregazione, che è esclusiva e totale, scriveremo:

$VR_{CodMotore}$(**Automobile**) ⊆ $VR_{CodMotore}$(**Motore**)
**AND**
$VR_{CodCarr}$(**Automobile**) ⊆ $VR_{CodCarr}$(**Carrozzeria**)
**AND**
$VR_{CodRuota}$(**Automobile**) ⊆ $VR_{CodRuota}$(**Ruota**)

e

$VR_{CodMotore}$(**Motore**) ⊆ $VR_{CodMotore}$(**Automobile**)
$VR_{CodCar}$(**Carrozzeria**) ⊆ $VR_{CodCar}$(**Automobile**)
$VR_{CodRuota}$(**Ruota**) ⊆ $VR_{CodRuota}$(**Automobile**)

- Vincolo intrarelazionale di **singola ennupla**:

V4(**Riparazione**): "**SE** (**Riparazione**.Gravità ≥ 7)
**ALLORA** **Riparazione**.Spesa < 1000"

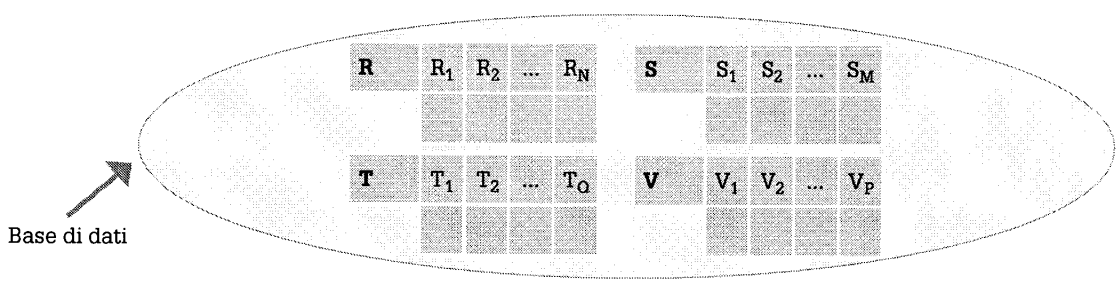
Notiamo che:

- l'associazione *acquista* è stata inserita nella relazione *Automobile*;
- l'associazione per aggregazione è stata creata inserendo le chiavi delle entità *Motore*, *Carrozzeria*, *Ruota* nella relazione *Automobile*. Nel seguito del volume, per semplicità di trattazione, potremmo non considerare esplicitamente i vincoli referenziali.

## 5 Le operazioni relazionali

Abbiamo detto che possiamo rappresentare la nostra base di dati con un insieme di relazioni.

✦ Figura A3.29
Una base di dati relazionale



Focalizziamo ora l'attenzione sulle operazioni che consentono di **interrogare** una base di dati relazionale appena creata. Nel tempo sono stati proposti diversi linguaggi per l'interrogazione delle basi di dati relazionali, quasi tutti di tipo **non procedurale**. Tali linguaggi utilizzano uno dei due seguenti approcci:

Interrogazione di una base di dati relazionale

1. approccio basato sull'**algebra relazionale**: in questo approccio il risultato di un'**interrogazione** (o **query**) è una relazione. Per ottenere tale relazione si formula un'interrogazione, utilizzando alcuni operatori di algebra relazionale. Tali operatori vengono composti fra loro e applicati alle relazioni della base di dati.
   Alcuni operatori sono, ad esempio, quelli di *unione*, *intersezione* e *differenza fra relazioni* in senso insiemistico.

Algebra relazionale

2. approccio basato sul **calcolo relazionale**: anche in questo approccio il risultato di un'**interrogazione** (o **query**) è una relazione. Per ottenere tale relazione si formula un'interrogazione, utilizzando il calcolo dei predicati del primo ordine sulle relazioni della base di dati.

Calcolo relazionale

Vediamo un esempio per capire meglio la differenza tra i due approcci.

**Esempio**

Abbiamo il seguente schema relazionale:

**Proprietario**(CodFisc, Cognome, Nome, Età)
**Automobile**(Targa, Marca, Modello, CodFiscProp)

L'interrogazione: "quali sono e che età hanno i proprietari di automobili Ferrari o Maserati" può essere formulata nei seguenti modi:

1. utilizzando l'algebra relazionale, la relazione *Risultato* può essere espressa:

```
project(restrict(Proprietario.CodFisc join Automobile.CodFiscProp)
where Marca ="Ferrari" or Marca = "Maserati")
   on Cognome, Nome, Età
```

   Il significato degli operatori project, restrict, join sarà chiaro in seguito.

2. utilizzando il *calcolo relazionale*, la relazione *Risultato* può essere espressa:

```
{t| (∃p) (∃a) (Proprietario(p) and Automobile(a)
   and p.CodFisc = a.CodFiscProp
   and  (a.Marca = "Ferrari" or a.Marca = "Maserati")
   and t.Cognome = p.Cognome
   and t.Nome = p.Nome
   and t.Età = p.Età)}
```

   Il significato può essere espresso come segue: "il risultato dell'interrogazione è formato da tutte le tuple *t* per le quali esiste una tupla *p* di *Proprietario* e una tupla *a* di *Automobili*, aventi lo stesso valore per l'attributo che rappresenta il codice fiscale e l'attributo *Marca* di *a* è "Ferrari" oppure "Maserati"; gli attributi di *t* sono gli attributi *Cognome*, *Nome*, *Età* di *p*".

I due approcci possono considerarsi equivalenti dal **punto di vista espressivo**, ovvero la relazione *Risultato* che deriva dall'espressione algebrica può essere espressa con un equivalente predicato del primo ordine.

I due approcci sono equivalenti anche dal **punto di vista implementativo**.

Scegliamo di utilizzare l'approccio dell'algebra relazionale, data la maggiore familiarità con gli operatori algebrici rispetto al calcolo dei predicati.

## 5.1 Algebra relazionale

Occorre scegliere l'insieme degli operatori che vogliamo utilizzare nelle interrogazioni. Tale insieme deve essere **funzionalmente completo**, cioè deve consentire di ottenere gli stessi risultati ottenibili con altri linguaggi relazionali come, ad esempio, quelli legati all'approccio basato sul calcolo dei predicati.

Operatori relazionali

Un insieme funzionalmente completo è quello formato dai seguenti cinque operatori relazionali:

1. **unione** di relazioni;
2. **differenza** di relazioni;
3. **prodotto** di relazioni;
4. **proiezione** di relazioni;
5. **restrizione** di relazioni.

Componendo opportunamente tali operatori, è possibile formulare qualsiasi interrogazione sulla nostra base di dati.

Oltre a questi cinque operatori di base, è opportuno introdurne altri due:

6. **intersezione** di due relazioni;
7. **giunzione naturale** di due relazioni.

Questi ultimi possono essere derivati da quelli base, cioè ottenuti per loro composizione, ma vengono introdotti perché il loro utilizzo permette di scrivere formule più semplici e sintetiche.

Per descrivere i precedenti sette operatori utilizzeremo una nostra sintassi, rifacendoci a quella oramai largamente utilizzata dalla maggior parte della letteratura sull'argomento.

### 5.1.1 Unione di relazioni

Relazioni compatibili

Due relazioni *R* e *S* vengono chiamate **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:

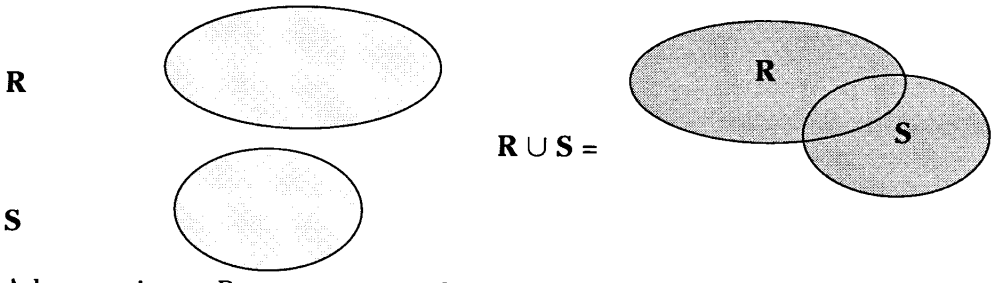
**Persona**(Nome: Stringa, Stipendio: Intero, DataNascita: Data)
**Dipendente**(Nominativo: Stringa, Stip: Intero, DNascita: Data)

Unione di relazioni

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

$$\textbf{union}(R, S) = \mathbf{R \cup S} = \{ t \mid t \in R \ \textbf{or} \ t \in S \}$$

Utilizzando la rappresentazione insiemistica abbiamo:




✦ Figura A3.30
La rappresentazione insiemistica dell'unione di relazioni

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 = Clienti1Semestre99**

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

**S = Clienti2Semestre99**

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

✦ Figura A3.31
Un esempio di due relazioni compatibili

**R∪S = Clienti99 = Clienti1Semestre99 ∪ Clienti2Semestre99**

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

✦ Figura A3.32
Un esempio di unione di relazioni compatibili

Per come è stata definita l'operazione di unione abbiamo che:
Grado(R ∪ S) = Grado(R) = Grado(S)
Cardinalità(R ∪ S ) = Cardinalità(R) + Cardinalità(S) – numero di tuple ripetute

Ovviamente abbiamo supposto che le tuple con nome "Bianchi" presenti in entrambe le relazioni siano uguali (per semplicità abbiamo consideriamo *Nome* come attributo chiave).

### 5.1.2 Differenza di relazioni

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

$$\textbf{difference}(R, S) = \mathbf{R - S} = \{ t \mid t \in R \textbf{ and } t \notin S \}$$

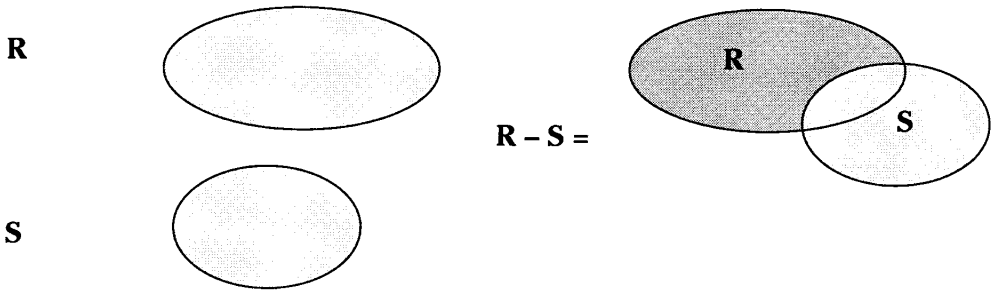
notare che la differenza insiemistica non gode della proprietà commutativa quindi:

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

dove

$$S - R = \textbf{difference}(S, R) = \{ t \mid t \in S \textbf{ and } t \notin R \}$$

Utilizzando la rappresentazione insiemistica abbiamo:




✦ Figura A3.33
La rappresentazione insiemistica della differenza di relazioni

Ad esempio, se *R* rappresenta tutti i clienti relativi all'attività di un'azienda e *S* i clienti relativi all'anno 1998, *R*–*S* rappresenta tutti i clienti esclusi quelli relativi al 1998.

**R = Clienti**

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

**S = Clienti98**

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

✦ Figura A3.34
Un esempio di differenza di relazioni

**R – S = Clienti – Clienti98**

| R – S |
|---|
| Rossi |
| Verdi |

Le tuple presenti in *S* non vengono inserite.

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

Grado(R – S) = Grado(R) = Grado(S)
Cardinalità(R – S) = Cardinalità(R) – numero di tuple presenti anche in *S*

### 5.1.3 Prodotto cartesiano di due relazioni

Date due relazioni qualsiasi *R* e *S*, rispettivamente di grado *g1* e *g2* e cardinalità *c1* e *c2*, il **prodotto** di *R* e *S* è la relazione di grado *g1* + *g2* e cardinalità *c1* x *c2*, le cui tuple si ottengono concatenando ogni tupla di *R* con ogni tupla di *S*.

Cioè se $r = (a_1, a_2, \ldots, a_{n1})$ e $s = (b_1, b_2, \ldots, b_{n2})$ sono due tuple, la concatenazione di r e s è data da:

$$\mathbf{r\ conc\ s} = (a_1, a_2, \ldots, a_{n1}, b_1, b_2, \ldots, b_{n2})$$

Il prodotto di *R* per *S* è quindi definito come:

$$\mathbf{R \times S} = \{\, t \mid t = \mathbf{r\ conc\ s},\ r \in R,\ s \in S \,\}$$

Per evitare ogni ambiguità nei nomi degli attributi di *R* x *S*, occorre che i nomi degli attributi di *R* e *S* siano diversi tra loro. Se così non fosse, si può pensare di ricorrere a una preventiva operazione di *ridenominazione* degli attributi.

**Esempio**

Consideriamo il caso in cui abbiamo una relazione *R* che rappresenti gli *Ordini* e una relazione *S* che rappresenti gli *Articoli*; vogliamo riunire in un'unica tabella tutti gli ordini e per ogni ordine le informazioni di ogni articolo.

✦ Figura A3.35

**R = Ordini**

| R | CodOrd | Data | Cliente |
|---|---|---|---|
| | 003 | 12/10/99 | Rossi |
| | 004 | 23/12/99 | Bianchi |

**S = Articoli**

| S | CodArt | Prezzo | Q.tà |
|---|---|---|---|
| | A001 | 1000 | 2 |
| | A002 | 2000 | 3 |
| | A003 | 3000 | 1 |

**R x S = Ordini x Articoli**

| R x S | CodOrd | Data | Cliente | CodArt | Prezzo | Q.tà |
|---|---|---|---|---|---|---|
| | 003 | 12/10/99 | Rossi | A001 | 1000 | 2 |
| | 003 | 12/10/99 | Rossi | A002 | 2000 | 3 |
| | 003 | 12/10/99 | Rossi | A003 | 3000 | 1 |
| | 004 | 23/12/99 | Bianchi | A001 | 1000 | 2 |
| | 004 | 23/12/99 | Bianchi | A002 | 2000 | 3 |
| | 004 | 23/12/99 | Bianchi | A003 | 3000 | 1 |

Grado(R x S) = g1 + g2 = 3 + 3 = 6
Card(R x S) = c1 x c2 = 2 x 3 = 6

Occorre precisare che *R* x *S* va visto come un semplice prodotto cartesiano che potrebbe non avere un significato ben chiaro anche in un contesto molto semplice come il nostro esempio, generalmente è visto nell'algebra relazionale come un risultato intermedio di una elaborazione più complessa.

### 5.1.4 Proiezione di una relazione

Data una relazione *R* e un sottoinsieme $A = \{A_1, A_2, \ldots, A_k\}$ dei suoi attributi, si definisce **proiezione** di *R* su *A* la relazione di grado *K* che si ottiene da *R* ignorando le colonne relative agli attributi non contenuti in *A* ed eliminando le eventuali tuple duplicate.

Pertanto scriveremo:

**project** R **on** $A_1, A_2, \ldots, A_k$

**Esempio**

Consideriamo la seguente relazione per i clienti di un'azienda. Dalla relazione *Clienti* si vuole estrapolare solo l'elenco degli agenti (relativi a quei clienti) e l'indirizzo dei clienti.

**R = Clienti**

| R | CodCli | Nome | Agente | IndirizzoCliente |
|---|---|---|---|---|
| | C001 | Bianchi | Polis | via Po, 23 |
| | C002 | Neri | Conte | via Roma, 12 |

**S = project R on** Agente, IndirizzoCliente

| S | Agente | IndirizzoCliente |
|---|---|---|
| | Polis | via Po, 23 |
| | Conte | via Roma, 12 |

✦ Figura A3.36 Un esempio di proiezione

Grado(S) è ≤ Grado(R)
Card(S) ≤ Card(R), infatti le tuple presenti nella proiezione possono essere anche di cardinalità inferiore a *R*, poiché le tuple uguali vengono scartate.

Come possiamo notare, l'effetto di una proiezione è quello di *selezionare un certo numero di colonne* dalla tabella della relazione.

Effetto della proiezione

### 5.1.5 Restrizione di una relazione

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

Pertanto scriveremo:

**restrict** R **where** P = {t | t ∈ R **and** P(t)}

**Esempio**

Supponiamo di avere la seguente tabella con informazioni relative ai clienti di un'azienda. Vogliamo selezionare le informazioni relative ai clienti della provincia di Milano. Il nostro predicato sarà: P(t) = {Provincia = "MI"}

**R = Clienti**

| R | CodCli | Nome | Provincia | IndirizzoCli |
|---|---|---|---|---|
| | C002 | Neri | LE | via Roma, 12 |
| | C006 | Bianchi | MI | via Po, 23 |
| | C005 | Rossi | MI | via Moro, 2 |

**S = restrict R where P**

| S | CodCli | Nome | Provincia | IndirizzoCli |
|---|---|---|---|---|
| | C006 | Bianchi | MI | via Po, 23 |
| | C005 | Rossi | MI | via Moro, 2 |

✦ Figura A3.37 Un esempio di restrizione

Grado(S) = Grado(R):
Card(S) ≤ Card(R), è uguale quando tutte le tuple soddisfano *P*.

Effetto di una restrizione

Come possiamo notare, l'effetto di una restrizione è quello di *selezionare un certo numero di righe* dalla tabella della relazione.

### 5.1.6 Intersezione di due relazioni

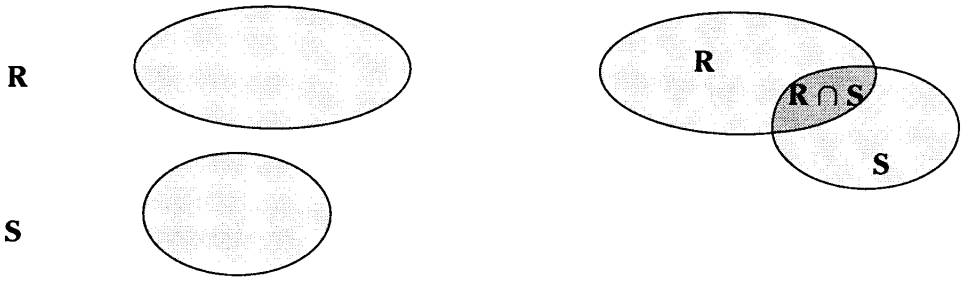
Date due relazioni compatibili $R$ e $S$, l'**intersezione** di $R$ e $S$ restituisce la relazione composta da tutte le tuple presenti sia in $R$ che in $S$.

Pertanto scriveremo:

$$\textbf{intersect}(R, S) = R \cap S = \{ t \mid t \in R \textbf{ and } t \in S \}$$

Utilizzando la rappresentazione insiemistica abbiamo:

✦ Figura A3.38
La rappresentazione insiemistica dell'intersezione di due relazioni




Possiamo verificare che:

$$\textbf{intersect}(R, S) = \textbf{difference}(R, \textbf{difference}(R, S))$$

**Esempio**

Supponiamo di avere le informazioni relative al clienti del 1998 e a quelli del 1999 della nostra azienda. Vogliamo ottenere una tabella con le persone che erano clienti sia nel 1998 che nel 1999.

**R = Clienti98**

| R | CodCli | Nome | Provincia | IndirizzoCli |
|---|---|---|---|---|
| | C006 | Bianchi | MI | via Po, 23 |
| | C002 | Neri | LE | via Roma, 12 |
| | C005 | Rossi | MI | via Moro, 2 |

**S = Clienti99**

| S | CodCli | Nome | Provincia | IndirizzoCli |
|---|---|---|---|---|
| | C016 | Verdi | CO | via Moro, 13 |
| | C002 | Neri | LE | via Roma, 12 |
| | C005 | Rossi | MI | via Moro, 2 |

✦ Figura A3.39
Un esempio di intersezione di relazioni

**R∩S**

| R∩S | CodCli | Nome | Provincia | IndirizzoCli |
|---|---|---|---|---|
| | C002 | Neri | LE | via Roma, 12 |
| | C005 | Rossi | MI | via Moro, 2 |

Grado(R ∩ S) = Grado(R) = Grado(S):
Card(R ∩ S ) ≤ alla cardinalità minore tra $R$ e $S$ in particolare è pari a Card(R) se $R \subseteq S$, invece è pari a Card(S) se $S \subseteq R$.

### 5.1.7 Giunzione di due relazioni

Date due relazioni $R$ e $S$ di grado $g1$ e $g2$, l'operazione di **giunzione naturale** 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 tuple 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 volta a selezionare le tuple aventi lo stesso valore degli attributi $A$ e $B$, ottenendo così una relazione con le colonne $A$ e $B$ uguali (si parla anche di **equi-join**);
3. si elimina una di queste due colonne.

Indicheremo l'operazione con:

**R.A join S.B**

Lo scopo della *giunzione naturale* è quello di combinare due relazioni aventi uno o più attributi in comune, generando una nuova relazione che contiene:

Scopo della giunzione naturale

- le *colonne* della prima e della seconda, meno gli attributi in comune;
- le *righe* della prima e della seconda, combinate secondo i valori uguali dell'attributo comune.

## Esempio

Supponiamo di avere due tabelle:

- una relativa ai clienti della nostra azienda;
- l'altra relativa agli agenti.

Vogliamo ottenere in una stessa tabella, per ogni cliente, anche le informazioni degli agenti da cui vengono serviti.

**R = Clienti**

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

**S = Agenti**

| S | CodAg | NomeAg | TelAgente |
|---|---|---|---|
| | A0016 | Polis | 0346/5647523 |
| | A0052 | Rinaldi | 0328/7665541 |

✦ Figura A3.40 Le due relazioni sulle quali eseguire la giunzione

Vediamo passo per passo il procedimento appena descritto nella definizione. Avremo:

1) R x S

| R x S | CodCli | NomeCli | A CodAgente | IndirizzoCli | B CodAg | NomeAg | TelAgente |
|---|---|---|---|---|---|---|---|
| | C006 | Bianchi | A0052 | via Po, 23 | A0016 | Polis | 0346/5647523 |
| | C006 | Bianchi | A0052 | via Po, 23 | A0052 | Rinaldi | 0328/7665541 |
| | C002 | Neri | A0016 | via Roma, 12 | A0016 | Polis | 0346/5647523 |
| | C002 | Neri | A0016 | via Roma, 12 | A0052 | Rinaldi | 0328/7665541 |
| | C005 | Rossi | A0052 | via Moro, 2 | A0016 | Polis | 0346/5647523 |
| | C005 | Rossi | A0052 | via Moro, 2 | A0052 | Rinaldi | 0328/7665541 |

✦ Figura A3.41 Il prodotto cartesiano di R e S

2) **restrict** R x S **where** CodAgente = CodAg

| CodCli | NomeCli | A CodAgente | B IndirizzoCli | CodAg | NomeAg | TelAgente |
|---|---|---|---|---|---|---|
| C006 | Bianchi | A0052 | via Po, 23 | A0052 | Rinaldi | 0328/7665541 |
| C002 | Neri | A0016 | via Roma, 12 | A0016 | Polis | 0346/5647523 |
| C005 | Rossi | A0052 | via Moro, 2 | A0052 | Rinaldi | 0328/7665541 |

✦ Figura A3.42 La restrizione del prodotto cartesiano di R e S

3) eliminando l'attributo *CodAg* di *Agenti* otteniamo la relazione finale del *join*:

| RelazioneFinale | CodCli | NomeCli | IndirizzoCli | CodAgente | NomeAg | TelAgente |
|---|---|---|---|---|---|---|
| | C006 | Bianchi | via Po, 23 | A0052 | Rinaldi | 0328/7665541 |
| | C002 | Neri | via Roma, 12 | A0016 | Polis | 0346/5647523 |
| | C005 | Rossi | via Moro, 2 | A0052 | Rinaldi | 0328/7665541 |

✦ Figura A3.43 La relazione finale della congiunzione tra R e S

**project(restrict** R x S **where** CodAgente = CodAg) **on** CodCli, NomeCli, CodAgente, IndirizzoCli, NomeAg, TelAgente

Grado(R.A join S.B) = (g1 + g2 – 1), poiché l'attributo comune compare una volta sola.
Card(R.A join S.B) non è prevedibile a priori, in quanto si ottengono solo le righe che possono essere combinate attraverso i valori presenti in entrambe le tabelle per l'attributo comune.

L'operatore join si applica in modo indipendente dal contenuto delle due tabelle ovvero nel nostro esempio la tabella *Clienti* poteva avere un cliente al quale non corrispondeva nessun agente; viceversa la tabella *Agenti* poteva contenere un agente al quale non corrispondeva alcun cliente.

Se quindi una tabella non ha una tupla corrispondente nell'altra tabella, nel risultato del join non viene visualizzata alcuna delle due tuple. Se si desidera invece selezionare tutte le tuple da una tabella a prescindere se abbia o meno delle tuple corrispondenti nell'altra tabella, è possibile ricorrere al **join esterno** (da distinguere da quello trattato finora che è considerato **interno**).

Il risultato di un join esterno visualizzerà celle vuote là dove non sono presenti tuple corrispondenti nell'altra tabella. Il join esterno può essere:

- **sinistro** o **left join**: visualizza tutte le tuple della prima tabella e solo quelle della seconda che hanno un valore corrispondente per l'attributo comune;
- **destro** o **right join**: visualizza tutte le tuple della seconda tabella e solo quelle della prima che hanno un valore corrispondente per l'attributo comune.

Si parla infine di **self join** quando le tuple di una tabella vengono combinate con le tuple della stessa tabella nel caso in cui siano presenti valori corrispondenti per gli attributi.

Il join esterno viene utilizzato quando si desidera rispondere a specifiche interrogazioni che coinvolgono associazioni ISA.

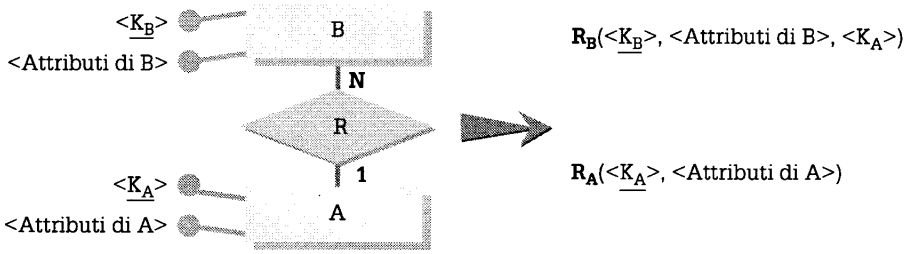
# 6 Interrogazioni sullo schema relazionale

Possiamo ora applicare l'algebra relazionale per effettuare semplici interrogazioni sulla nostra base di dati. Vediamo in particolare come si possono interrogare relazioni che provengono dalla traduzione dei diversi tipi di associazione del diagramma ER.

## 6.1 Interrogazione di associazioni 1:N

Consideriamo la seguente associazione *1:N*, nella quale per semplicità non abbiamo considerato i vincoli di integrità:

✦ Figura A3.44
Una generica associazione 1:N

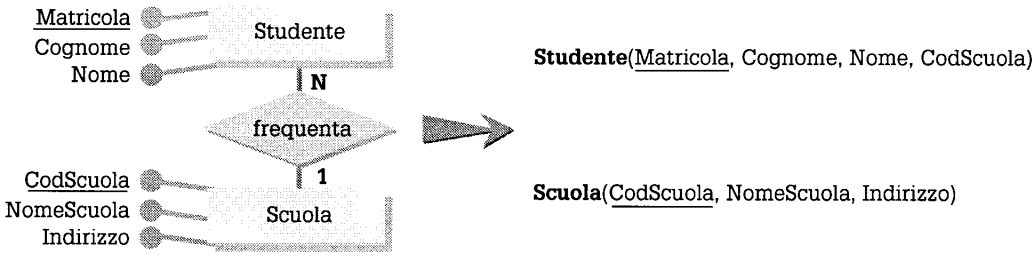



Per conoscere quali sono gli elementi di $R_B$ associati a un particolare elemento di $R_A$, con chiave *k1*, occorre formulare la seguente interrogazione nell'algebra relazionale:

**project**(**restrict** $R_B$ **where** $K_A$= k1) **on** <attributi di B>

Riconsideriamo l'esempio dello studente e della scuola:

✦ Figura A3.45
Un esempio di interrogazione su un'associazione 1:N



Vincoli di integrità referenziale per rappresentare la totalità della diretta e dell'inversa. → $VR_{CodScuola}(Studente) \subseteq VR_{CodScuola}(Scuola)$

$VR_{CodScuola}(Scuola) \subseteq VR_{CodScuola}(Studente)$

Per conoscere quali studenti frequentano l'ITIS "Fermi" che ha *CodScuola* pari a "S001", scriveremo:

**project(restrict Studente where** CodScuola = "S001") **on** Cognome, Nome

Possiamo vedere questa interrogazione come un'interrogazione **composta** formata da due interrogazioni:

Interrogazione composta

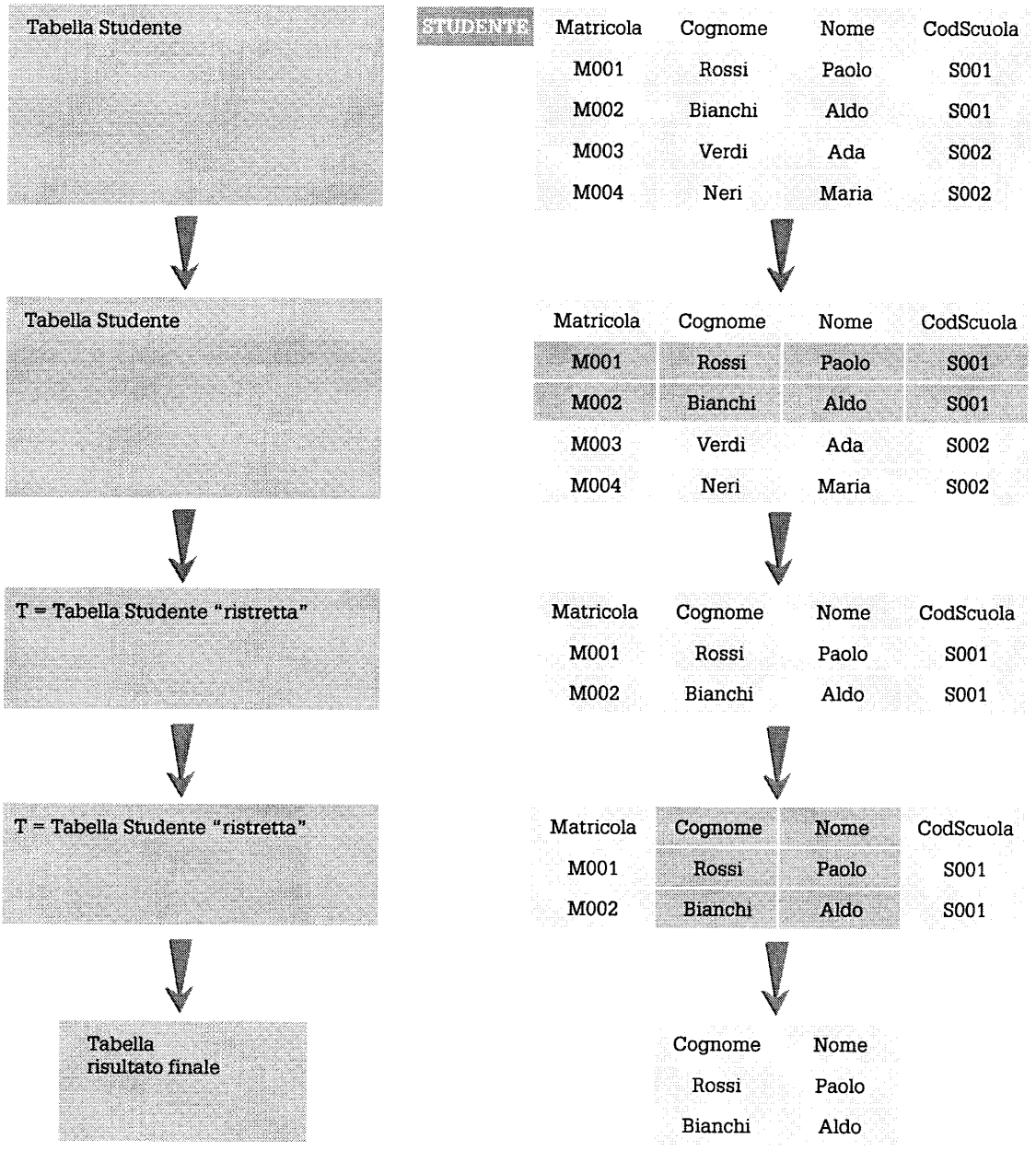
T = **restrict Studente where** CodScuola ="S001"

Possiamo chiamare questa prima interrogazione **sottointerrogazione**. Essa restituisce come risultato una tabella temporanea (che possiamo chiamare *T*); su tale tabella si esegue un'altra sottointerrogazione:

Sottointerrogazione

**project** T **on** Cognome, Nome

Otteniamo così la tabella **risultato finale**. Possiamo graficamente riassumere in:



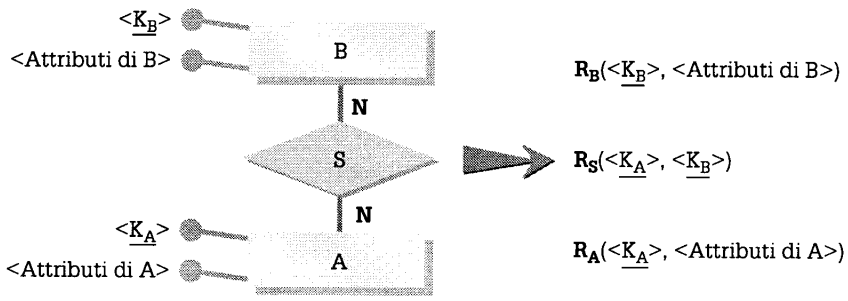

✦ Figura A3.46
I passi elementari in cui scomporre un'interrogazione composta

Tutte le volte che eseguiremo interrogazioni utilizzando le espressioni dell'algebra relazionale, faremo sempre ricorso alla *composizione* di interrogazioni e sottointerrogazioni.

## 6.2 Interrogazione di associazioni N:N

Consideriamo la seguente associazione *N:N* in cui, per semplificare, non abbiamo considerato i vincoli di integrità:

✦ Figura A3.47
Una generica associazione N:N




Per conoscere quali sono gli elementi di $R_B$ associati a un particolare elemento di $R_A$, con chiave k1, occorre formulare la seguente interrogazione nell'algebra relazionale:

$$R_B.K_B \textbf{ join(project(restrict } R_s \textbf{ where } K_A = k1) \textbf{ on } K_B).K_B$$

Si effettua la giunzione naturale tra la relazione $R_B$ e la colonna $K_B$ di $R_S$ ristretta ai valori associati con la chiave k1.

Naturalmente è possibile effettuare l'interrogazione simmetrica "quali sono gli elementi di $R_A$ associati all'elemento di $R_B$ avente come chiave k2":

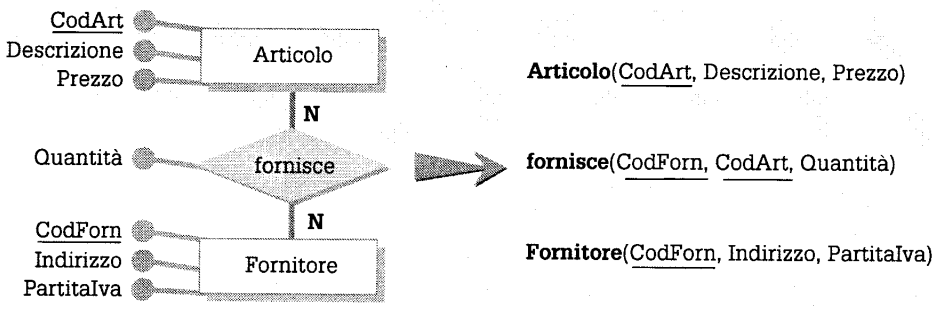
$$R_A.K_A \textbf{ join(project(restrict } R_s \textbf{ where } K_B = k2) \textbf{ on } K_A).K_A$$

Osserviamo che la relazione $R_S$ presenta due chiavi esterne, per le quali valgono i vincoli di integrità già rilevati in precedenza.

### Esempio

Consideriamo il seguente diagramma ER:

✦ Figura A3.48
Un esempio di interrogazione di associazione N:N




Per conoscere quali sono i fornitori dell'articolo "AP004", scriveremo:

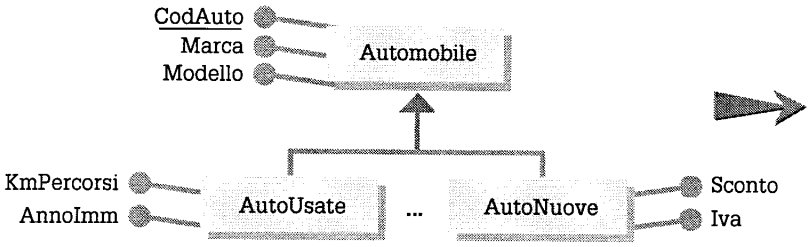
```
Fornitore.CodForn join
(project(restrict Fornisce where CodArt = "AP004") on CodForn).CodForn
```

Per conoscere quali articoli fornisce il fornitore "F11", scriveremo:

```
Articolo.CodArt join
(project(restrict Fornisce where CodForn = "F11") on CodArt).CodArt
```

## 6.3 Interrogazione per generalizzazioni

Consideriamo il seguente esempio di generalizzazione:




**Automobile**(CodAuto, Marca, Modello)

**AutoUsate**(CodAuto, KmPercorsi, AnnoImm)

...

**AutoNuove**(CodAuto, Sconto, Iva)

✦ Figura A3.49
Un esempio di generalizzazione

Per rispondere all'interrogazione: "Puoi elencare tutte le auto, riportando, per quelle usate, anche i loro attributi particolari?", scriveremo:

**Automobili**.CodAuto **left-join Usate**.CodAuto.

## 6.4 Interrogazioni nell'esempio della concessionaria di automobili

Consideriamo nuovamente l'esempio della concessionaria di automobili.
Utilizziamo le operazioni dell'algebra relazionale per rispondere alle seguenti query:

Q1. *Selezionare* il nome del proprietario del veicolo con targa "AB 450 TS".
Q2. *Elencare* le targhe delle automobili che sono state acquistate a "settembre".
Q3. *Elencare* le automobili usate con prezzo inferiore a un determinato valore.
Q4. *Elencare* le riparazioni da effettuare sulle auto acquistate da un determinato cliente.

Vediamo adesso come rispondere, esaminando le query una alla volta.

**Q1.** *Selezionare* il nome del proprietario del veicolo con targa "AB 450 TS".

```
project
   (restrict(Proprietario.CodFiscale
            join
         Automobile.CodFiscale
               )
   where Automobile.Targa = "AB450TS"
   )
on Cognome.Nome
```

Si esegue un'operazione relazionale di join tra *Proprietario* e *Automobili*, seguita da una restrizione per selezionare le tuple con *Targa* richiesta. Infine si applica una proiezione per scegliere solo le colonne relative al nome e cognome.

**Q2.** *Elencare* le targhe delle automobili che sono state acquistate a "settembre".

```
project
   (restrict Automobile
   where DataAcq > "31/08/2002" and DataAcq < "01/10/2002"
   )
on Targa
```

In questo caso basta una restrizione sulla tabella *Automobile*, seguita da una proiezione, per selezionare solo le colonne relative alla targa.

**Q3.** *Elencare* le automobili usate con prezzo inferiore a 10.000 euro.

```
project
   ( restrict( Automobile.CodAuto
                  join
               Autousata.CodAuto
               )
   where Automobile.Prezzo < 10000
   )
on Marca, Modello, Prezzo, AnnoImm, KmPercorsi
```

Si esegue una join tra *Automobile* e *AutoUsata* per ottenere un'unica relazione che abbia gli attributi di un'automobile e quelli di un'auto usata. Su questa unica relazione si effettua una restrizione sul prezzo e subito dopo una proiezione degli attributi che interessano.

# 7 La normalizzazione delle relazioni

## 7.1 Le anomalie

Forma normale

Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la qualità misurata in assenza di determinati difetti.

Quando uno schema relazionale non è normalizzato, può presentare **comportamenti indesiderati**, che possono compromettere la congruenza dei dati in esso contenuti durante le operazioni di:

- *inserimento* dei dati;
- *aggiornamento* dei dati;
- *cancellazione* dei dati.

Normalizzazione

La **normalizzazione** è un procedimento che serve per trasformare uno schema che presenta delle **anomalie** in uno equivalente in cui tali anomalie sono state eliminate. Il contenuto informativo dello schema non normalizzato e normalizzato deve essere equivalente.

Vediamo quali possono essere tali anomalie considerando la relazione *Magazzino*, che rappresenta lo schema relativo a un *magazzino di accessori per auto*.

In tale relazione un cliente (al quale è associato un indirizzo, la città e il codice di avviamento postale utili per la fatturazione) può ordinare più accessori e ogni accessorio può essere ordinato da più clienti. A ogni accessorio è inoltre associata una descrizione e il prezzo. La relazione è rappresentata dalla seguente tabella:

✦ Figura A3.50
Una relazione con anomalie

Magazzino

| CodCli | Indirizzo | Città | Cap | CodAcc | Descrizione | Prezzo | Quantità |
|---|---|---|---|---|---|---|---|
| C01 | via Po, 23 | Pisa | 56100 | M03 | Batteria | 100,00 | 3 |
| C01 | via Po, 23 | Pisa | 56100 | M12 | Radiatore | 1200,00 | 1 |
| C01 | via Po, 23 | Pisa | 56100 | M04 | Antenna | 25,00 | 3 |
| C02 | via Moro, 2 | Pisa | 56100 | M03 | Batteria | 100,00 | 2 |
| C02 | via Moro, 2 | Pisa | 56100 | M12 | Radiatore | 1200,00 | 1 |
| C03 | via Roma, 1 | Lucca | 55100 | M03 | Batteria | 100,00 | 2 |

La chiave di tale relazione è (CodCli, CodAcc). Possiamo definire le seguenti anomalie:

Anomalia in inserimento

- **anomalia in inserimento**. Non è possibile inserire un nuovo cliente del concessionario senza inserire i dati relativi all'auto e agli accessori ordinati. In modo analo-

go, non è possibile inserire solo un nuovo accessorio con la sua descrizione senza inserire il cliente che lo ha ordinato. La chiave infatti è: (CodCli, CodAcc);

- **anomalia in aggiornamento**. L'indirizzo di un cliente è ripetuto in diverse tuple. Se dovesse cambiare, occorre modificarlo in tutte le tuple in cui compare. Se tali modifiche dovessero essere parziali (se cioè si dovessero effettuare solo per alcune tuple e non per tutte), si lascerebbe la base di dati in uno stato detto **inconsistente**. Il discorso è analogo per la descrizione di un articolo;

Anomalia in aggiornamento

- **anomalia di cancellazione**. Cancellando la tupla di chiave (C01, M03) si perdono le informazioni relative all'articolo *M03*. Analogo discorso per la tupla di chiave (C03, M03). In quest'ultimo caso si perdono le informazioni relative al cliente *C03*.

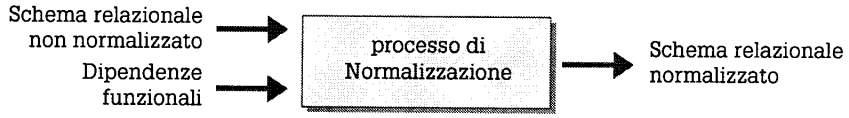
Anomalia di cancellazione

Questi fenomeni indesiderati si verificano poiché abbiamo rappresentato informazioni eterogenee con un'unica relazione:

- gli *articoli* presenti in magazzino;
- i *dati* anagrafici dei clienti;
- gli *ordini* dei clienti relativi a determinati articoli.

Il processo di *normalizzazione* elimina tali anomalie effettuando una serie di trasformazioni successive delle relazioni di partenza di uno schema relazionale, ottenendo relazioni che, in base al tipo di trasformazione applicata, possono trovarsi a diversi livelli di "bontà" chiamati **forme normali**.

Esistono molte forme normali per uno schema relazionale: 1FN, 2FN, 3FN, BCFN (*Forma Normale di Boyce – Codd*), 4FN, 5FN. Per i nostri scopi è sufficiente applicare il processo di normalizzazione per ottenere uno schema relazionale in 3FN. Nel prossimo paragrafo vedremo come ottenerlo attraverso lo studio delle dipendenze funzionali:




✦ Figura A3.51 Input e output del processo di normalizzazione

## 7.2 Prima forma normale

La **prima forma normale** di una relazione è quella più semplice e spesso viene trascurata, poiché è quella in cui si trovano la maggior parte delle relazioni appena create dall'utente.

Diremo che una relazione *R* è in **prima forma normale**, e scriveremo **1FN**, quando rispetta i requisiti fondamentali del modello relazionale.

I requisiti fondamentali del modello relazionale sono:

- i *valori di un attributo* (di una colonna) sono dello stesso tipo, ovvero appartengono allo stesso dominio;
- i *valori di una tupla* (di una riga) sono diversi da quelli delle altre tuple, ovvero non possono esistere due tuple uguali;
- l'*ordine delle tuple* è irrilevante;
- gli *attributi* sono di tipo elementare ovvero:
  - non possono essere ulteriormente *scomposti* in attributi più semplici,
  - non possono essere *composti* da gruppi di attributi ripetuti.

Esempio

Consideriamo la seguente relazione:

**Azienda**(CodAzienda, RagioneSociale, Indirizzo, TipoAttività, Dipendenti)

che contiene i dati relativi ad alcune aziende di un determinato settore.
Una possibile istanza di questa relazione è:

✦ Figura A3.52
Una relazione non in 1FN

| Azienda | CodAzienda | RagioneSociale | Indirizzo | TipoAttività | Dipendenti |
|---|---|---|---|---|---|
| | A001 | ETElettronica | via Po, 34 | Produzione | Bianchi Antonio<br>Rossi Paolo<br>Verdi Giovanni |
| | A002 | ABInformatica | via Rossini, 3 | Servizi | Neri Fabio<br>Rossi Alberto |

La relazione non è in *1FN*: infatti l'attributo *Dipendenti* non è un attributo elementare, in quanto è composto da un gruppo di attributi ripetuti (una lista di nomi di persone).
Per trasformare una relazione in *1FN* occorre rimpiazzarla con altre che soddisfino la definizione data precedentemente.
Si procede *decomponendola* sulla base degli attributi non elementari.
Dalla relazione precedente otterremo:

✦ Figura A3.53
La trasformazione della relazione in 1FN

| Azienda | CodAzienda | RagioneSociale | Indirizzo | TipoAttività |
|---|---|---|---|---|
| | A001 | ETElettronica | via Po, 34 | Produzione |
| | A002 | ABInformatica | via Rossini, 3 | Servizi |

| Dipendente | CodDip | Nome | CodAzienda |
|---|---|---|---|
| | D02 | Bianchi Antonio | A001 |
| | D04 | Rossi Paolo | A001 |
| | D13 | Verdi Giovanni | A001 |
| | D34 | Neri Fabio | A002 |
| | D45 | Rossi Alberto | A002 |

Con questa rappresentazione della realtà è ora più facile:

- *aggiungere* nuovi attributi che caratterizzino un dipendente;
- *inserire* un nuovo dipendente per una data azienda.

Quest'ultima operazione avrebbe comportato un'eventuale ridefinizione dello schema della relazione *Azienda* se le dimensioni del campo *Dipendenti* non fossero state opportunamente dimensionate e, comunque, qualsiasi dimensione sarebbe sempre stata scorretta poiché o troppo grande (per aziende con pochi dipendenti) o troppo piccola (per aziende con molti dipendenti).

## 7.3 Dipendenze funzionali

Data una relazione $R$ e un insieme $X = \{X_1, X_2, ..., X_n\}$ di $R$, si dice che un attributo $Y$ di $R$ **dipende funzionalmente** da $X$ e si scrive:

$$X_1, X_2, \ldots, X_n \rightarrow Y$$

se e solo se i valori degli attributi di $X$ determinano univocamente il valore dell'attributo $Y$ per ogni istanza della relazione $R$.
Si dice anche che X **determina** Y.

Dalla definizione precedente si evince che la dipendenza funzionale di un attributo da uno o più attributi è una proprietà che non dipende da una singola istanza della relazione. È quindi indipendente dal tempo: vale *in ogni istante* in cui consideriamo la relazione. Tale proprietà viene definita **proprietà semantica**.

Proprietà semantica

Nell'esempio della relazione *Magazzino*, vista nel *paragrafo 7.1*, possiamo individuare le seguenti dipendenze funzionali:

1. CodCli → Indirizzo

   L'indirizzo dipende funzionalmente da quel determinato cliente.

CodCli → Città

La città dipende funzionalmente da quel determinato cliente.

CodCli → Cap

Il codice di avviamento postale dipende funzionalmente da quel determinato cliente. Possiamo utilizzare la seguente forma abbreviata:

CodCli → Indirizzo, Città, Cap

2. CodAcc → Descrizione

   La descrizione di un articolo dipende da quel determinato articolo.

   CodAcc → Prezzo

   Il prezzo di un articolo dipende funzionalmente da quel particolare articolo. Possiamo utilizzare la seguente forma abbreviata:

   CodAcc → Descrizione, Prezzo

3. CodCli, CodAcc → Quantità

   La quantità ordinata di un determinato articolo da parte di un cliente dipende funzionalmente dal cliente e dall'articolo.

Possiamo notare che le dipendenze funzionali generalizzano il concetto di *chiave*, cioè tutti gli attributi non chiave dipendono funzionalmente dagli attributi chiave.

Possiamo infatti scrivere in forma abbreviata:

4. CodCli → Indirizzo, Descrizione, Prezzo, Quantità

   Possiamo notare che le *anomalie* sono legate alle dipendenze funzionali, infatti:

   a. Un cliente ha un unico indirizzo:
      CodCli → Indirizzo
      La coppia *CodCli*, *Indirizzo* appare in più tuple.
   b. Un accessorio ha un'unica descrizione (e un unico prezzo):
      CodAcc → Descrizione
      La coppia *CodAcc*, *Descrizione* compare in più tuple.

Le prime due dipendenze funzionali causano quindi anomalie, mentre non causa anomalie la nostra terza dipendenza funzionale:

   c. Una quantità ha un unico cliente e un unico articolo:
      CodCli, CodAcc → Quantità
      (Non esistono tuple con gli stessi valori di CodCli, CodAcc e Quantità).

Questo si verifica perché l'attributo *Quantità* dipende da tutti gli attributi che formano la chiave, mentre ciò non accade per gli attributi *Descrizione*, *Prezzo*, *Indirizzo* che dipendono solo da una parte della chiave.

## 7.4 Seconda forma normale

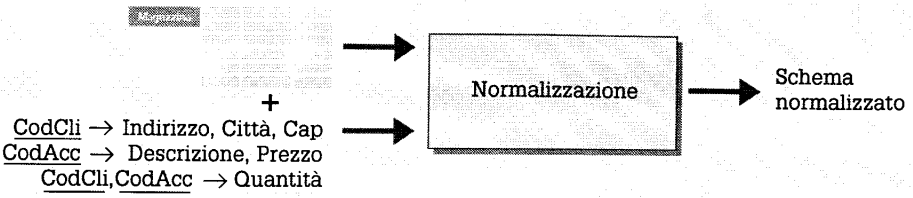
Diremo che una relazione *R* è in **seconda forma normale**, e scriveremo **2FN**, se non esistono attributi dipendenti solo da una parte della chiave, cioè *dipendenti parzialmente* dalla chiave.

Questa forma normale richiede cioè che gli attributi di una relazione siano omogenei, nel senso che devono essere solo proprietà associate direttamente alla chiave.

Per trasformare una relazione in *2FN* occorre rimpiazzarla con altre che soddisfino la definizione data precedentemente. Si procede *decomponendola sulla base delle dipendenze funzionali*, al fine di separare i concetti eterogenei.

**Esempio**

La relazione *Magazzino* (*paragrafo 7.1*) non è in *2FN*, poiché gli attributi *Indirizzo*, *Descrizione* e *Prezzo* dipendono solo da una parte della chiave. Abbiamo quindi la seguente situazione:




✦ Figura A3.54
Gli input e l'output del processo di normalizzazione del nostro esempio

Dove lo schema normalizzato è composto dalle seguenti relazioni:

✦ Figura A3.55
Lo schema normalizzato in 2FN

**Cliente**

| CodCli | Indirizzo | Città | Cap |
|---|---|---|---|
| C01 | via Po, 23 | Pisa | 56100 |
| C02 | via Moro, 2 | Pisa | 56100 |
| C03 | via Roma, 1 | Lucca | 55100 |

**Ordine**

| CodCli | CodAcc | Quantità |
|---|---|---|
| C01 | M03 | 3 |
| C01 | M04 | 3 |
| C01 | M12 | 1 |
| C02 | M03 | 2 |
| C02 | M12 | 1 |
| C03 | M03 | 2 |

**Accessorio**

| CodAcc | Descrizione | Prezzo |
|---|---|---|
| M03 | Batteria | 100,00 |
| M04 | Antenna | 25,00 |
| M12 | Radiatore | 1200,00 |

La *normalizzazione* risolve molte anomalie sulle operazioni che si possono effettuare sui dati, ma rende più complesse le operazioni di interrogazione sui dati.
Ovviamente viene data priorità all'integrità e alla consistenza dei dati piuttosto che alla velocità di risposta alle interrogazioni.

La *decomposizione* deve essere eseguita in modo tale da poter ricomporre l'informazione della relazione di partenza. Tutto ciò a partire dalle relazioni decomposte tramite un'operazione di *natural join* su queste ultime (per rendere possibile la ricostruzione, le tabelle decomposte devono avere attributi in comune).
Le relazioni in *2FN* possono ancora essere esposte ad anomalie, in quanto possono presentare delle ridondanze.

## 7.5 Terza forma normale

Osservando la relazione *Cliente*, possiamo notare che esiste la seguente dipendenza funzionale:

Città → Cap

in cui il Codice di Avviamento Postale dipende dalla città di residenza del cliente, e poiché vale anche:

CodCli → Città

(cioè l'attributo *Città* dipende funzionalmente dalla chiave), abbiamo la seguente catena di dipendenze:

CodCli → Città → Cap

dalla quale si evince che l'attributo *Cap* dipende **transitivamente**, e non direttamente, dalla chiave.

Dipendenza transitiva

Diremo che una relazione *R* è in **terza forma normale**, e scriveremo *3FN*, se per ogni possibile chiave di *R* accade che:

- *R* è in *2FN*, cioè non esistono *attributi non chiave* che dipendono solo da una parte della chiave;
- non esistono *attributi non chiave* che dipendono transitivamente dalla chiave, cioè non esistono *attributi non chiave* che dipendono da altri attributi non chiave.

Per trasformare una relazione in *3FN*, si crea una nuova relazione per ogni gruppo di *attributi non chiave* coinvolti nella dipendenza funzionale con *attributi non chiave*.

Ad esempio, la relazione *Cliente* con la dipendenza funzionale *Città → Cap*, deve decomporsi nelle seguenti relazioni:

**Cliente**

| CodCli | Indirizzo | Città |
|---|---|---|
| C01 | via Po, 23 | Pisa |
| C02 | via Moro, 2 | Pisa |
| C03 | via Roma, 1 | Lucca |

**Città**

| Città | Cap |
|---|---|
| Pisa | 56100 |
| Lucca | 55100 |

✦ Figura A3.56
Uno schema relazionale in 3FN

Una differente definizione di terza forma normale è quella di **Boyce** e **Codd**. In base alla loro definizione:

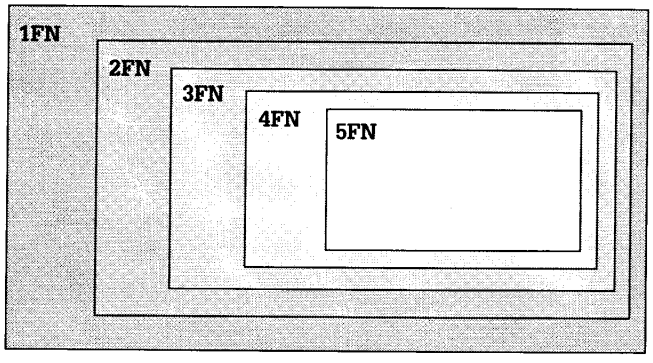
Boyce e Codd

si dice che una relazione è in **terza forma normale** se ogni determinante è una chiave candidata.
Si definisce **determinante** un attributo o un insieme di attributi da cui un altro attributo dipende funzionalmente in modo pieno.

Sebbene esistano, la **4FN** e la **5FN** non vengono considerate in questo contesto. La forma normale obbligatoria è solo la prima, mentre le altre servono per separare meglio concetti eterogenei, racchiudendoli in tabelle che al loro interno siano il più possibile omogenee.

Come abbiamo visto, passando dalla prima forma normale a quelle di livello più alto eliminiamo le anomalie, ma introduciamo una ridondanza di dati.

Una relazione in forma normale di livello più alto (ad esempio in **5FN**) è sicuramente anche una relazione in forma normale di qualunque livello più basso (ad esempio in **3FN** o in **2FN** ecc.), come possiamo notare dal seguente schema:




✦ Figura A3.57
L'annidamento di forme normali

# Competenze

■ QUESITI

1. Che cosa indica il grado di una relazione e cosa indica la sua cardinalità?
2. In quanti modi è possibile rappresentare una relazione?
3. A che cosa serve la chiave di una relazione?
4. Quale differenza c'è tra chiave candidata e chiave primaria?
5. In che cosa consistono i seguenti vincoli di integrità:
   - di dominio;
   - di ennupla;
   - intrarelazionali su più ennuple;
   - interrelazionali su valori qualsiasi degli attributi.
6. In che cosa consiste un vincolo referenziale?
7. Perché occorre specificare un vincolo referenziale?
8. Quali sono i possibili tipi di vincolo referenziale?
9. Come si rappresentano nel modello relazionale i vincoli di integrità referenziale sulle chiavi esterne?
10. Nella trasformazione dal diagramma ER di un'associazione N:N parziale, quali vincoli di integrità referenziale occorre specificare?
11. Che cosa indicano i seguenti vincoli referenziali?

$VR_{K_A}(R_B) \subseteq VR_{K_A}(R_A)$
$VR_{K_A}(R_A) \subseteq VR_{K_A}(R_B)$

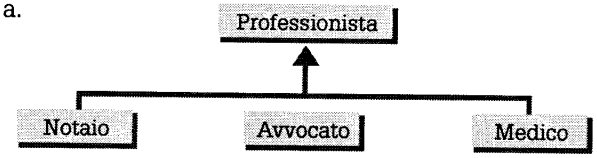
12. In quanti modi è possibile trasformare una generalizzazione in uno schema relazionale?
13. Come variano i vincoli quando si trasforma in schema relazionale un'aggregazione lasca o una stretta?
14. Quale formula permette una prima verifica della "bontà" del diagramma ER?
15. A quale operazione relazionale si riferiscono le seguenti equazioni relative alle relazioni R e S?
    Grado(R – S) = Grado(R) = Grado(S)
    Card(R – S) = Card(R) – numero di tuple presenti anche in S.
16. Quando una relazione si dice in terza forma normale?

# Competenze

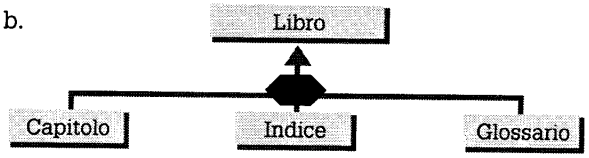
■ ESERCIZI

1. Traduci il diagramma ER, relativo ai film, introdotto negli esercizi dell'unità precedente, in uno schema relazionale.
2. Completa i seguenti diagrammi ER e traducili nei rispettivi schemi relazionali, compresi i vincoli di integrità:

a.




b.




3. Dato il seguente schema relazionale:

**Articolo** (CodArt, Descrizione, Prezzo)
**Fornitore** (CodForn, Indirizzo, PartitaIva, Provincia, Telefono)
**Fornisce** (CodForn, CodArt)

utilizza l'algebra relazionale per calcolare:
a. gli articoli con prezzo di vendita superiore a 100 euro;
b. gli articoli relativi ai fornitori della provincia di Milano;
c. il numero di telefono dei fornitori di un determinato articolo.

4. Modifica lo schema relazionale dell'esercizio precedente per poter calcolare:
   a. la data in cui è stato fornito un determinato articolo da un certo fornitore;
   b. il numero di pezzi forniti in una determinata data;
   c. la quantità totale disponibile per un determinato articolo.
5. Modifica lo schema relazionale dell'esercizio 3, eventualmente aggiungendo nuove relazioni, per poter calcolare:
   a. i clienti che hanno acquistato un determinato articolo;
   b. il prezzo medio di vendita di un determinato articolo;
   c. il prezzo medio di acquisto di un determinato articolo;
   d. il numero di vendite effettuate il 26 Marzo.
6. Stabilisci in che forma normale si trova l'esempio del *paragrafo 4.10* relativo alla concessionaria di automobili, descritto in questa unità.
7. Sia dato il seguente schema relazionale:

**BiciclettaAntica**(CodBiciA, CodTipoA, MarcaA, ModelloA, PrezzoA)
**Bicicletta**(CodBici, CodTipo, Marca, Modello, Prezzo)
**TipoBici**(CodTipo, Descrizione)

Esegui le seguenti operazioni relazionali:
- **union(BiciclettaAntica, Bicicletta)**
- **difference(BiciclettaAntica, Bicicletta)**
- **project Bicicletta on** Marca, Modello, Prezzo
- **restrict Bicicletta where** Marca = "Velox"
- **intersect(BiciclettaAntica, Bicicletta)**
- **Bicicletta**.CodTipo **join TipoBici**.CodTipo

8. Dato lo schema relazionale dell'esercizio precedente è possibile scrivere le seguenti operazioni relazionali?

   a. **project (restrict Bicicletta where** Marca = "Velox") **on** Marca Modello =

   b. **restrict (project Bicicletta on** Marca, Modello) **where** Marca = "Velox"

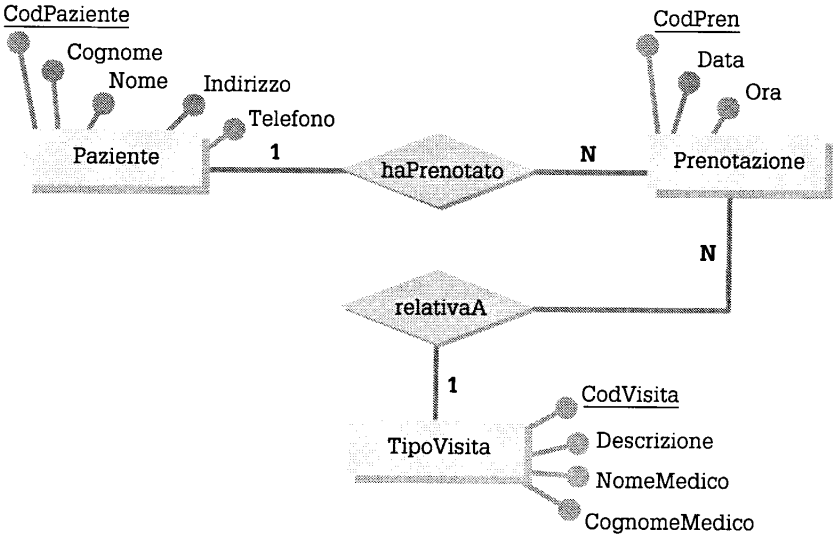
9. Della seguente relazione Film individua:
   - un'anomalia in cancellazione;
   - un'anomalia in aggiornamento;

| Titolo | Anno | Durata | Produttore | Attore |
|---|---|---|---|---|
| Un amico come te | 1999 | 120 | Studi Neri | John Bayles |
| Un amico come te | 1999 | 120 | Studi Neri | Giorgio Rocca |
| Il sole e la luna | 1997 | 110 | Red Studios | Giorgio Rocca |
| I sette amigos | 2000 | 95 | Red Studios | John Bayles |

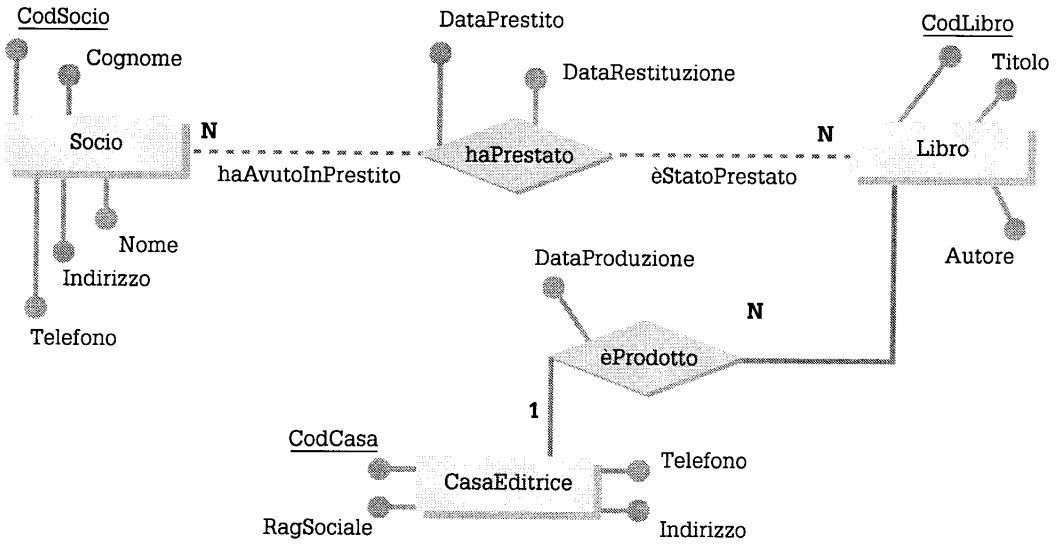
10. Individua le dipendenze funzionali della relazione dell'esercizio precedente.
11. Trasforma in terza forma normale la relazione dell'esercizio 9.
12. Determina in che forma normale è la seguente relazione:

| Titolo | Anno | Durata | Produttore |
|---|---|---|---|
| Un amico come te | 1999 | 120 | Studi Neri |
| Il sole e la luna | 1997 | 110 | Red Studios |
| I sette amigos | 2000 | 95 | Red Studios |

13. **Prenotazioni di visite specialistiche**
    Traduci il seguente diagramma ER relativo alla gestione di visite specialistiche presso gli Uffici del Servizio Sanitario.




14. **Biblioteca**
    Traduci il seguente diagramma ER relativo ai prestiti di libri ai soci di una Biblioteca.



15. Consideriamo le seguenti relazioni:

| R | X | Y | Z |
|---|---|---|---|
| | x1 | y1 | z1 |
| | x2 | y2 | z2 |
| | x3 | y3 | z3 |
| | x4 | y4 | z4 |

| S | Z | W |
|---|---|---|
| | z0 | w1 |
| | z2 | w2 |
| | z4 | w3 |

Che tipo di join è stato applicato per ottenere le seguenti relazioni?

| ? | X | Y | Z | W |
|---|---|---|---|---|
| | x2 | y2 | z2 | w2 |
| | x4 | y4 | z4 | w3 |

| ? | X | Y | Z |
|---|---|---|---|
| | x2 | y2 | z2 |
| | x4 | y4 | z4 |

| ? | X | Y | Z | W |
|---|---|---|---|---|
| | x1 | y1 | z1 | null |
| | x2 | y2 | z2 | w2 |
| | x3 | y3 | z3 | null |
| | x4 | y4 | z4 | w3 |

| ? | X | Y | Z | W |
|---|---|---|---|---|
| | null | null | z0 | w1 |
| | x2 | y2 | z2 | w2 |
| | x4 | y4 | z4 | w3 |

| ? | X | Y | Z | W |
|---|---|---|---|---|
| | x1 | y1 | z1 | null |
| | x2 | y2 | z2 | w2 |
| | x3 | y3 | z3 | null |
| | x4 | y4 | z4 | w3 |

## ESEMPI SVOLTI

1. Consideriamo l'esempio relativo alla mostra canina, il cui diagramma ER è stato presentato negli esempi svolti dell'unità A2. Fornisci il relativo schema relazionale con alcuni esempi di vincoli.
Schema relazionale:

**Razza**(CodRazza, NomeRazza, AltezzaStandard, PesoStandard)
**Cane**(CodCane, NomeCane, Altezza, Peso, DataNascita, Punteggio, CodProp, CodRazza)
**Proprietario**(CodProp, Nome, Cognome)
Alcuni Vincoli di integrità

- Vincolo di dominio:

V1(**Cane**): "Altezza < 110"
V2(**Cane**): "Peso < 150"

- Vincolo referenziale. Per esprimere la totalità della diretta dell'associazione appartiene scriveremo:

$VR_{CodRazza}$(**Cane**) ⊆ $VR_{CodRazza}$(**Razza**)

- Vincolo interrelazionale, che esprime possibili variazioni del peso e dell'altezza di un cane rispetto agli standard di razza:

V4(**Cane**, **Razza**): "**Razza**.Peso – 10 ≤ **Cane**.Peso ≤ **Razza**.Peso + 10"
V5(**Cane**, **Razza**): "**Razza**.Altezza – 5 ≤ **Cane**.Altezza ≤ **Razza**.Altezza + 5"

2. Consideriamo l'esempio relativo al condominio, il cui diagramma ER è stato presentato negli esempi svolti dell'unità A2.
Fornisci il relativo schema relazionale.
Rispondi, utilizzando l'algebra relazionale, alle seguenti query:
Q1. Elencare le quote da versare relative agli appartamenti di un determinato proprietario.
Q2. Elencare i proprietari che hanno versato la quota relativa a un determinato mese (e anno).
Schema relazionale:

**Appartamento**(CodApp, NumeroVani, Superficie, Indirizzo, Interno, Quota, CodPersona)
**Persona**(CodPersona, Cognome, Nome, Indirizzo, Telefono)
**èInquilino**(CodPersona, CodApp)
**haVersato**(Importo, DataVersamento, Anno, Mese, CodPersona, CodApp)

Query:
Q1. Elencare le quote da versare relative agli appartamenti di un determinato proprietario.

```
project (
    restrict Appartamento where CodPersona =
                                        "P003"
    )
on CodApp, Indirizzo, Quota
```

Q2. Elencare i proprietari che hanno versato la quota relativa a un determinato mese (e anno).

```
project (
    restrict HaVersato where Mese = "Settembre"
                        and Anno = 2003
    )
on DataVersamento, CodPersona, CodApp
```

In questo modo si ottengono i codici di Persona e di Appartamento.
Se volessimo il nome e cognome relativo ai proprietari scriveremmo:

```
project (
    (restrict HaVersato where Mese =
                        "Settembre").CodPersona
    join
    Persona.CodPersona
    )
on Cognome, Nome, DataVersamento, CodApp
```