LUIGINO CALVI

BASI DI DATI

2022

Una *grande* scoperta risolve un grande problema ma c'è un granello di scoperta nella soluzione di ogni problema.
Il tuo problema può essere modesto; ma se esso sfida la tua curiosità e mette in gioco le tue facoltà inventive, e lo risolvi con i tuoi mezzi, puoi sperimentare la tensione e godere del trionfo della scoperta.
Questa esperienza ad una età suscettibile può creare un gusto per il lavoro mentale e lasciare un'impronta nella mente e un carattere per tutta la vita.

G. Polya, *How to solve it*

Indice

1

TABELLARE

I programmi per computer usualmente operano su tabelle d'informazioni. Nella maggioranza dei casi queste tabelle non sono semplicemente masse amorfe di valori numerici; esse coinvolgono importanti relazioni strutturali fra i dati elementi.

D. Knuth,
The Art of Computer Programming

La *tabella* è una delle forme più utilizzate per strutturare i dati. In una tabella gli elementi vengono organizzati in una struttura bidimensionale in cui ogni riga contiene la registrazione delle varie componenti di un elemento. Il modello relazionale delle basi di dati si fonda sulla organizzazione dei dati in forma tabellare.

1.1 La nascita del modello relazionale

Il modello relazionale dei dati venne ideato, studiato e proposto nel 1970 da E. F. Codd, un ricercatore del Centro Ricerche dell'IBM. Il modello venne originariamente descritto nel famoso articolo di Codd, *A Relational Model of Data for Large Shared Data Banks*, CACM, Vol.13, N.6, giugno 1970. L'obiettivo di Codd era quello di formulare un modello semplice per superare le complessità degli allora dominanti modelli gerarchico e reticolare (che avevano il difetto di essere molto legati alla sottostante organizzazione fisica dei dati) e sufficientemente potente per gestire tutte le problematiche relative alle basi di dati. Nella metà degli anni '70 del secolo scorso, il lavoro pionieristico di Codd diede l'avvio a molto studi teorici (molti dei quali opera dello stesso Codd), sui quali si sviluppò una matura teoria. La teoria non trovò però immediata applicazione a causa delle impegnative risorse (memoria e tempo di elaborazione) richieste da questo modello e rimase sulla carta per quasi un decennio; soltanto agli inizi degli anni '80 si videro le prime significative implementazioni.

Il modello relazionale si basa su semplici e ben definiti concetti matematici; ha come fondamento teorico il concetto di relazione matematica, la teoria elementare degli insiemi e la logica dei predicati del primo ordine. Il modello relazionale rappresenta la base di dati come una collezione di relazioni. Ogni relazione può essere graficamente descritta mediante una tabella in cui ciascuna riga corrisponde ad un elemento della relazione. Proprio questa estrema semplicità e linearità della percezione dei dati che ha l'utente, ha costituito dapprima l'attrazione e successivamente il successo del modello relazionale.

Al giorno d'oggi esistono molte implementazioni efficienti di questo modello dei dati che risulta utilizzabile sul più modesto computer attuale e si dispone di software libero (MySQL e PostgreSQL i più noti) e di specifici linguaggi per l'elaborazione dei dati (algebra relazionale, linguaggio SQL).

1.2 Il modello relazionale

Il modello relazionale, adottando una strategia minimalista, permette di rappresentare *insiemi di elementi omogenei aventi una struttura piatta.* In altri termini, gli elementi devono appartenere ad una stessa categoria ed avere una struttura le cui componenti sono di un tipo atomico (numero, valore logico, stringa) e non aventi struttura nidificata (struttura di strutture). Nonostante questa semplice strutturazione, il modello permette di rappresentare le situazioni più complesse ed articolate. In taluni casi è necessario una fase di traduzione in modo da ricondursi ad uno schema relazionale.

Aderentemente all'impostazione sopra descritta, gli elementi vengono rappresentati in una tabella bidimensionale in cui ogni riga rappresenta un elemento e le colonne corrispondono alle singole componenti (attributi) degli elementi. È da tenere presente che la sequenzialità delle righe di una tabella è ininfluente ai fini dell'apporto informativo.

Esempio 1.2.1 - La tabella riportata in figura 1.1 descrive le informazioni relative ad un insieme di libri di geografia, riportando il loro titolo, numero di pagine e prezzo di copertina.

LIBRO

Titolo	*Pagine*	*Prezzo*
`Mari del Nord`	176	18.50
`Africa selvaggia`	212	24.90
`Navigazione fluviale`	148	16.90
`Castelli di Francia`	289	28.50
`Gli 8000`	208	32.50
`Oceania`	271	33.90
`Isole del Pacifico`	234	29.50

Figura 1.1: Rappresentazione tabellare di un insieme di libri di geografia.

Come si può notare in questo esempio, una tabella risulta caratterizzata dal nome (LIBRO), dall'intestazione delle colonne (*Titolo*, *Pagine*, *Prezzo*), dal dominio dei valori inseribili in corrispondenza di ciascuna colonna e dai dati registrati nelle righe della tabella.

1.3 La definizione dei dati

In una tabella, come quella descritta nella figura 1.1, i dati vengono descritti a due livelli:

- livello *intensionale*: viene descritto il nome della tabella, i nomi di ciascuna colonna della tabella, il dominio di ciascuna colonna, ossia l'insieme dei valori in corrispondenza della data colonna; tutte queste informazioni costituiscono lo *schema* della tabella
- livello *estensionale*: vengono descritti gli specifici valori delle varie righe della tabella; questi valori devono essere coerenti con la definizione dello schema della tabella

Esempio 1.3.1 - I linguaggi di programmazione, sia quelli di tipo generale (ad esempio i linguaggio C, Java, Python) che quelli specificamente orientati alla gestione della basi di dati (ad esempio il linguaggio SQL) predispongono delle istruzioni per la definizione della struttura delle tabelle (nome della tabella, nomi dei campi e loro domini). Questa componente del linguaggio viene detta *Data Definition Language* (o *DDL*).

Libro	*Titolo*	*Pagine*	*Prezzo*

```
/* Linguaggio C */
typedef struct {
    char  titolo[20];
    short pagine;
    float prezzo;
} Libro;

# LINGUAGGIO SQL:
CREATE TABLE Libro
(
    titolo CHAR(20),
    pagine SMALLINT,
    prezzo DECIMAL(5,2)
);
```

Figura 1.2: Definizione dello schema della tabella descritta nella figura 1.1: mediante una notazione grafica, mediante il linguaggio C e mediante il linguaggio SQL.

Nel caso in cui gli elementi da rappresentare abbiano una struttura in cui alcune componenti siano a loro volta strutturate, è necessario *appiattire* la struttura, come descritto nell'esempio 1.3.2.

Esempio 1.3.2 - Un colore può essere descritto mediante un nome identificativo e la lista delle sue 3 componenti rgb che esprimono le gradazioni d'intensità, ad esempio nell'intervallo $[0, 255]$, delle componenti dei colori fondamentali (*rosso*, *verde*, *blu*). Le informazioni di ciascun colore possono essere rappresentate mediante una sequenza strutturata ed eterogenea di valori. Ad esempio, il colore *rosso* può essere descritto mediante la struttura $['rosso', [255, 0, 0]]$. Il modello relazionale richiede che le varie righe delle tabelle siano composte da elementi atomici; è pertanto necessario appiattire le strutture che rappresentano ciascun elemento. Pertanto un *insieme di colori* può essere rappresentato mediante una tabella come descritto nella figura 1.3.

Colore

Nome	R	G	B
`rosso`	255	0	0
`verde`	0	255	0
`blu`	0	0	255
`viola`	255	0	255

Figura 1.3: Rappresentazione tabellare di un insieme di 4 colori.

Nel caso in cui la struttura degli elementi da rappresentare sia più forte che non la semplice organizzazione insiemistica, è necessario integrare nel modello relazionale *anche* la descrizione della struttura organizzativa.

Esempio 1.3.3 - Una tabella descrive un *insieme* di elementi, uno per riga. Nel caso sia importante l'ordine delle righe è necessario inserire l'informazione che permetta di ricostruire l'ordine sequenziale degli elementi. Ad esempio, la classifica finale di una gara di podistica può essere rappresentata mediante una tabella come la seguente:

ATLETA

Ordine	*Cognome*	*Nome*	*Società*
2	De Marchi	Diego	U.S. Runners
4	Bianchi	Marino	Feltrese
1	Rinaldi	Michele	Podisti
5	Bernardi	Alessandro	U.S. Runners
3	Chiarini	Stefano	Podisti

Figura 1.4: Rappresentazione tabellare di una classifica finale di una gara podistica.

In una situazione come questa, nel caso in cui venisse registrato anche il tempo impiegato da ciascun atleta, il valore corrispondente all'ordine di classifica potrebbe non essere memorizzato e calcolato, all'occorrenza, in base al tempo impiegato.

Problema 1.3.1 Rappresentare mediante il modello relazionale un torneo a 4 squadre ad eliminazione diretta, similmente a quanto descritto nella figura 1.5 (per rendere più significativo l'esempio si pensi ad un torneo ad eliminazione diretta con 2^n concorrenti, con n, ad esempio, uguale a 4).

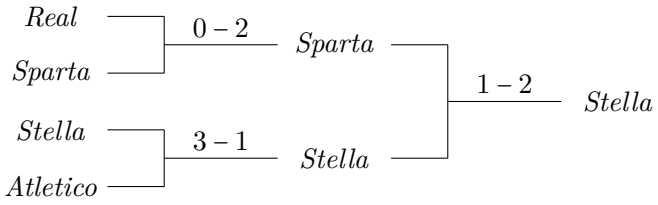


Figura 1.5: Tabellone del torneo.

Soluzione. Per la traduzione in forma tabellare dello schema riportato in figura 1.5 è necessario tenere presente che una tabella, nel modello relazionale, deve essere considerata composta da un *insieme* di righe e pertanto la scrittura sequenziale delle righe è ininfluente. Una possibile traduzione tabellare è riportata nella figura 1.6.

TORNEO

*Squadra*1	*Squadra*2	*Reti*1	*Reti*2	*Turno*
Real	Sparta	0	2	1
Stella	Atletico	3	1	1
Sparta	Stella	1	2	2

Figura 1.6: Rappresentazione tabellare dell'albero del torneo di calcio descritto in figura 1.5.

Osservazione. Nella soluzione del problema 1.3.1 si è partiti da uno schema originario dei dati (albero del tabellone del torneo) e si è cercato successivamente di ricondurlo ad uno schema relazionale. Questo approccio risulta produttivo nelle situazioni in cui si tratta con una sola tabella (o poche tabelle). In situazioni più articolate o complesse serve adottare delle tecniche di progettazione concettuale: la soluzione sarebbe stata più naturale, immediata ed efficiente se si fosse definito dapprima un modello concettuale (che sarà presentato più avanti) e poi si fosse tradotto questo in uno schema relazionale.

Problema 1.3.2 Rappresentare mediante il modello relazionale i dati che descrivono i confinamenti fra regioni come descritto nella figura 1.7.

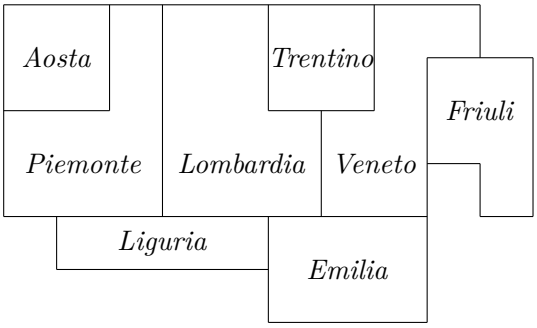


Figura 1.7: Confinamenti fra regioni.

Soluzione. La situazione descritta nella figura 1.7 può essere rappresentata mediante il grafo duale riportato in figura 1.8. Si tratta di un grafo non orientato, non pesato e labellato con i nomi delle regioni.

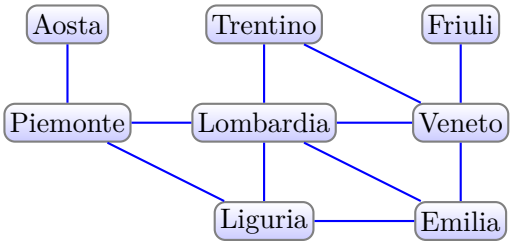


Figura 1.8: Grafo dei confinamenti fra regioni.

A sua volta il grafo dei confinamenti riportato nella figura 1.8 può essere rappresentato mediante la tabella riportata nella figura 1.9.

CONFINAMENTO

*Regione*1	*Regione*2
`Aosta`	`Piemonte`
`Piemonte`	`Lombardia`
`Piemonte`	`Liguria`
`Lombardia`	`Trentino`
`Lombardia`	`Veneto`
`Lombardia`	`Emilia`
`Lombardia`	`Liguria`
`Trentino`	`Veneto`
`Veneto`	`Friuli`
`Veneto`	`Emilia`
`Emilia`	`Liguria`

Figura 1.9: Rappresentazione tabellare del grafo riportato in figura 1.8.

Problema 1.3.3 Rappresentare mediante il modello relazionale un albero binario.

Soluzione. Come esempio di riferimento consideriamo l'albero riportato nella figura 1.10. Si noti che ci sono dei dati uguali memorizzati in nodi diversi. Questo comporta che un nodo è univocamente individuato in base alla sua posizione all'interno dell'albero ed univocamente definito se ne viene specificato anche il dato memorizzato nel nodo stesso.

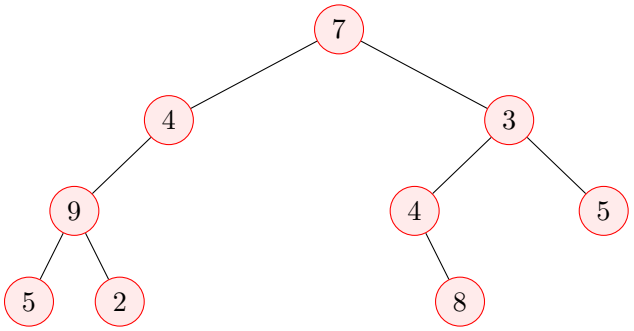


Figura 1.10: Albero binario.

Per rappresentare un albero binario mediante il modello relazionale si può passare attraverso una rappresentazione intermedia a livelli realizzata mediante un array come descritto nella figura 1.11.

0	1	2	3	4	5	6	7	8	9	10	11	12
7	4	3	9	•	4	5	5	2	•	•	•	8

Figura 1.11: Rappresentazione sequenziale dell'albero binario riportato nella figura 1.10.

Richiamando la modalità di rappresentazione di una sequenza mediante il modello relazionale, componendo le due rappresentazioni, si perviene alla rappresentazione dell'albero di figura 1.10

ALBERO

Posizione	*Valore*
0	7
1	4
2	3
3	9
5	4
6	5
7	5
8	2
12	8

Figura 1.12: Rappresentazione tabellare di un albero.

Il problema che segue propone un significativo esempio di come può essere progettata e realizzata una singola tabella che rappresenta un insieme di dati elementari fra loro organizzati in categorie logiche.

Problema 1.3.4 Rappresentare mediante il modello relazionale un *disegno* composto da *spezzate*. Sono note le etichette e le coordinate dei vertici delle spezzate.

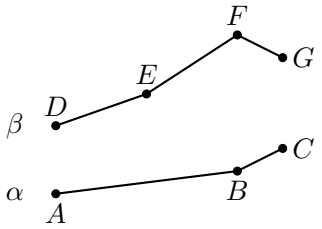


Figura 1.13: Disegno composto da spezzate composte da punti.

Soluzione. La soluzione può essere basata sulla seguente idea, impostata secondo la metodologia top-down:

$$Disegno = insieme(Spezzata)$$

$$Spezzata = sequenza(Punto)$$

quindi

$$Disegno = insieme(sequenza(Punto))$$

Un punto può essere rappresentato mediante una tupla costituita dalla etichetta (label) e dalle coordinate del punto; ad esempio:

$rap(Punto\ A)$ =

Label	*X*	*Y*
A	100	200

Una spezzata può essere identificata mediante un numero progressivo e rappresentata mediante una relazione in cui ogni tupla è costituita dalla rappresentazione di un punto e dal numero d'ordine del punto nella sequenza dei punti che costituiscono la spezzata:

$rap(Spezzata\ \alpha)$ =

Punto	*N*
rap(punto A)	1
rap(punto B)	2
rap(punto C)	3

Un disegno può essere rappresentato mediante una tabella in cui ogni tupla costituisce la rappresentazione di una spezzata:

$rap(Disegno)$ =

Spezzata
rap(spezzata α)
rap(spezzata β)

Dovendo poi far confluire la rappresentazione nel modello relazionale, ogni tupla che rappresenta una spezzata dovrà essere esplosa in tuple ciascuna delle quali rappresenta un punto.
Unendo tutte le considerazioni sopra esposte si perviene alla seguente rappresentazione relazionale del disegno riportato in figura 1.13.

$rap(Disegno)$ =

S	N	L	X	Y
α	1	A	100	100
α	2	B	150	100
α	3	C	160	110
β	1	D	100	110
β	2	E	130	110
β	3	F	150	130
β	4	G	160	120

Osservazione. Gli esempi ed i problemi presentati in questo paragrafo hanno evidenziato la potenza del modello relazionale che si presta a modellare strutture di dati anche molto lontane da una rapresentazione tabellare bidimensionale. Ci sono comunque delle situazioni in cui la realtà non può essere facilmente costretta entro i quattro lati una tabella, in quanto essa non si presta ad essere ricondotta ad uno schema piatto bidimensionale; ad esempio quando gli oggetti hanno delle strutture nidificate (disegni composti da elementi complessi a loro volta composti da primitive grafiche; pezzo meccanico composto da sottosistemi composti da elementi di base, ...). In casi come questi si fa ricorso ad altri modelli (ad esempio, a modelli di dati orientati agli oggetti oppure a modelli ibridi).

1.4 Relazioni

Il modello relazionale dei dati descritto informalmente negli esempi riportati nei precedenti paragrafi, può essere definito in modo rigoroso e formale in quanto tale modello si fonda sul concetto matematico di relazione descritto nella definizione che segue.

Dati n insiemi $D_1, D_2, \ldots, D_n$ $(n \geq 1)$, non necessariamente distinti, dicesi *relazione* un sottoinsieme $\mathcal{R}$ del prodotto cartesiano $D_1 \times D_2 \times \cdots \times D_n$. Nei casi particolari $n = 1$ ed $n = 2$ la relazione $\mathcal{R}$ dicesi, rispettivamente, *unaria* e *binaria*. Gli elementi di $\mathcal{R}$ sono costituiti dalle ennuple $(a_1, a_2, \ldots, a_n)$, dove $a_i \in D_i$, per ogni $i \in \{1, 2, \ldots, n\}$. Nella terminologia delle basi di dati gli insiemi D_i vengono detti *domini*, il numero n viene detto *grado* della relazione $\mathcal{R}$ e viene indicato con $deg(\mathcal{R})$, le n-ple vengono dette *tuple*, il numero delle tuple viene detto *cardinalità* della relazione $\mathcal{R}$ ed è indicato con $card(\mathcal{R})$ o con $|\mathcal{R}|$. I domini corrispondono al concetto di *tipo di dato* dei linguaggi di programmazione; nel contesto delle basi di dati relazionali si impone che i domini siano costituiti da un tipo di dato elementare atomico (carattere, intero, decimale, stringa, data, ...). In una relazione ogni dominio è associato ad un nome identificativo, detto *attributo*, che lo identifica univocamente all'interno della relazione. Il dominio di un attributo A viene indicato con $dom(A)$. Più attributi possono avere lo stesso dominio ma devono avere nomi distinti. Ogni relazione viene individuata mediante un identificatore detto *nome* (*della relazione*). Se t è una tupla con attributi $(A_1, \ldots, A_n)$ e $X \subseteq \{A_1, \ldots, A_n\}$, con $t[X]$ si denota la (sotto)tupla corrispondente ai soli attributi di X.

A differenza delle relazioni matematiche, le relazioni del modello relazionale sono variabili nel tempo (*time-varying relations*) in quanto le ennuple di una relazione possono essere dinamicamente inserite, aggiornate e cancellate. Generalmente, dal punto di vista pratico, si considerano solo relazioni di cardinalità finita, coerentemente con il fatto che le basi di dati sono sempre finite; tuttavia, in alcune trattazioni teoriche si considerano anche relazioni aventi cardinalità infinita.

DEFINIZIONE 1. Uno *schema di relazione* è costituito da un identificatore del nome della relazione e da un insieme finito di attributi e domini ad essi associati. Uno schema di relazione viene indicato con la notazione

$$\mathcal{R}(A_1 : D_1, \ldots, A_n : D_n)$$

dove gli A_i sono gli attributi della relazione $\mathcal{R}$ e $D_i = dom(A_i)$ sono i corrispondenti domini. Qualora non vi sia ambiguità, uno schema di relazione viene denotato semplicemente con la notazione

$$\mathcal{R}(A_1, \ldots, A_n)$$

oppure con $\mathcal{R}(X)$, dove X denota la lista degli attributi di $\mathcal{R}$. Una (*istanza di*) *relazione* sullo schema di relazione $\mathcal{R}(A_1, \ldots, A_n)$ è un insieme finito di tuple della forma $(a_1, \ldots, a_n)$, dove $a_i \in dom(A_i)$.

Esempio 1.4.1 - Nella figura che segue è descritto, con tre diverse notazioni, uno schema di relazione.

$$\textsc{Aula}(Nome : string, Piano : int, Posti : int, LIM : boolean)$$

$$\textsc{Aula}(Nome, Piano, Posti, LIM)$$

Aula	*Nome*	*Piano*	*Posti*	*LIM*

Anche se, per definizione, una relazione è un insieme di tuple, per comodità, una relazione viene solitamente rappresentata mediante uno schema a tabella, dove ogni riga rappresenta una tupla ed ogni colonna rappresenta la sequenza dei valori assunti dalle tuple in corrispondenza di un attributo, disponendo le ennuple in una sequenza, come descritto nell'esempio 1.4.2.

Esempio 1.4.2 - Nella figura che segue è riportata una rappresentazione tabellare di una relazione di rango 4, cardinalità 3 ed avente lo schema descritto nel precedente esempio 1.4.1.

Aula

Nome	*Piano*	*Posti*	*LIM*
1	1	28	✓
4	1	22	
3	1	25	✓
LabInf1	0	26	✓
LabSis	1	24	

Nella rappresentazione tabellare uno schema di relazione corrisponde all'intestazione della tabella mentre l'istanza di relazione corrisponde ai dati delle tuple della tabella. Schema ed istanza di relazione vengono detti anche *intensione* ed *estensione* della relazione. Con *estendere* o *popolare* una relazione si intende l'operazione che consiste nell'aggiungere i dati alla relazione.

Le definizioni date precedentemente relative alle relazioni trovano corrispondenza nella rappresentazione tabellare di una relazione come descritto nella seguente tabella che descrive la corrispondenza dei termini del modello relazionale e della descrizione tabellare.

relazione	↔	tabella
tupla	↔	riga
attributo	↔	nome della colonna
dominio	↔	insieme di valori di una colonna
cardinalità	↔	numero di righe
grado	↔	numero di colonne

1.5 Basi di dati relazionali

In prima approssimazione, una base di dati relazionale è costituita da un insieme di relazioni. Una caratterizzazione più precisa è riportata nella definizione che segue.

DEFINIZIONE 2. Uno *schema di una base di dati relazionale* o semplicemente *schema relazionale* è un insieme finito di schemi di relazione aventi nomi distinti. Una *istanza di una base di dati relazionale* o semplicemente *base di dati relazionale* è un insieme di relazioni che estendono uno schema relazionale.

Esempio 1.5.1 - Nella figura che segue è descritto uno schema relazionale composto da 2 schemi di relazione ed una corrispondente base di dati relazionale composta da 2 istanze di relazione. La figura intende rappresentare i dati relativi ai dipendenti ed ai reparti di un'azienda.

DIPENDENTE	*Matricola*	*Cognome*	*Nome*

REPARTO	*Sigla*	*Descrizione*	*Telefono*

DIPENDENTE

Matricola	*Cognome*	*Nome*
AN01	Antico	Nicola
BA01	Barbieri	Amedeo
GF03	Gaio	Felice
FG01	Ferretti	Gaetano
AF01	Armellini	Francesca
DC02	Donati	Claudia
RG02	Rovetta	Gianni
BD05	Bonato	Diego

REPARTO

Sigla	*Descrizione*	*Telefono*
A	amministrazione	210
P	produzione	204
M	magazzino	254
S	spedizione	212

1.6 Chiavi

Il modello relazionale dei dati si fonda sul concetto di *chiave* (di una relazione), costituita da un sottoinsieme non vuoto degli attributi della relazione stessa. A seconda delle proprietà godute, una chiave viene qualificata con specifici attributi (*candidata*, *primaria*, *super*, *secondaria*, *esterna*), come descritto nelle definizioni che seguono.

DEFINIZIONE 3. In una relazione si definisce *chiave candidata* un insieme minimale di attributi che permette di identificare univocamente una tupla all'interno della relazione, ossia se soddisfa alle seguenti due proprietà:

1. *univocità* : i valori dell'insieme di attributi sono diversi per ciascuna tupla e quindi consentono di identificare ciascuna tupla
2. *minimalità* : togliendo uno qualsiasi degli attributi, i valori corrispondenti agli altri attributi possono non risultare diversi tra loro in tutte le tuple

In una relazione possono esistere più chiavi candidate; fra queste ne viene scelta una che viene detta *chiave primaria* e viene indicata sottolineando gli attributi che la compongono. In ogni (schema di) relazione deve essere definita la chiave primaria.
Un insieme di attributi che soddisfa alla proprietà di univocità si dice *superchiave*. Un insieme di attributi che non soddisfi alla proprietà di univocità viene detto *chiave secondaria*.

Osservazione. In una relazione esiste sempre almeno una chiave candidata: basta prendere la chiave composta da tutti gli attributi in quanto una relazione è sempre composta da tuple distinte. Per motivi di efficienza ed ottimizzazione delle operazioni di elaborazione viene scelta come chiave primaria una chiave candidata costituita dal minor numero di attributi ed in genere ne viene predisposta una composta da un solo attributo; si tratta spesso di una stringa o un numero identificativo, ad esempio: un codice, una sigla, un numero progressivo.

Esempio 1.6.1 - Nella figura che segue sono descritte delle idonee chiavi primarie relative agli schemi di relazione riportati nell'esempio 1.5.1; si assume che ogni dipendente sia univocamente individuato dalla sua matricola ed ogni reparto dalla sua sigla.

DIPENDENTE	*Matricola*	*Cognome*	*Nome*

REPARTO	*Sigla*	*Descrizione*	*Telefono*

Problema 1.6.1 Lo schema relazionale che segue descrive gli studenti attualmente iscritti presso un dato istituto di istruzione superiore; l'attributo *Matr* è un numero progressivo che individua univocamente uno studente che si è iscritto all'istituto nel corso degli anni, *CF* è il codice fiscale, *Classe* è la sigla della classe attualmente frequentata, *NReg* è il numero di registro, *Nascita* è la data di nascita, *EMail* è l'indirizzo di posta elettronica dello studente e *Tel* è il numero di telefono della famiglia dello studente. Individuare delle chiavi candidate e fra queste una idonea chiave primaria.

STUDENTE

Matr	*CF*	*Classe*	*NReg*	*Cognome*	*Nome*	*Nascita*	*EMail*	*Tel*

Soluzione. Possibili chiavi candidate sono:

$K_1 = \{Matr\}$
$K_2 = \{CF\}$
$K_3 = \{Classe, NReg\}$
$K_4 = \{EMail\}$

Fra queste chiavi, K_1 è una adeguata chiave primaria.

1.7 Associazioni fra tabelle

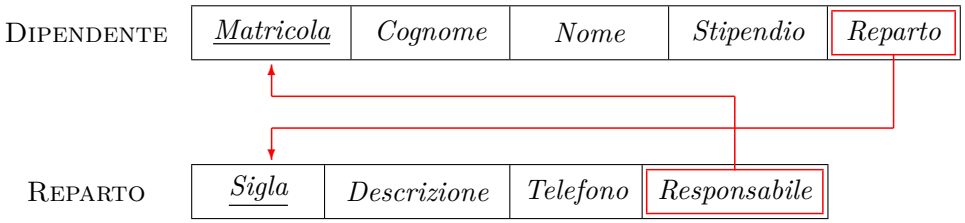
Finora abbiamo visto esempi di dati organizzati in un'unica relazione. Nella maggior parte delle situazioni i dati sono organizzati in più relazioni fra loro collegate mediante dei valori registrati nelle relazioni stesse; questi valori hanno il significato di collegamenti logici fra le varie tuple delle relazioni, come descritto nella seguente

DEFINIZIONE 4. Uno o più attributi di una relazione costituiscono una *chiave esterna* (*foreign key*) se fanno riferimento agli attributi di una chiave primaria di un'altra relazione (in taluni casi della relazione stessa); la chiave esterna deve essere composta dallo stesso numero di attributi che costituiscono la chiave primaria della relazione alla quale fa riferimento; inoltre il dominio di ciascun attributo della chiave esterna deve essere uguale al dominio del corrispondente attributo della chiave primaria.

La definizione di una chiave esterna avviene a livello intensionale, sugli schemi di relazione, al momento della definizione degli schemi, prima di popolare le relazioni. Questo comporta l'imposizione di vincoli sui dati che popoleranno successivamente le relazioni.

In uno schema relazionale le chiavi esterne vengono evidenziate mediante un riquadro dal quale parte una freccia che unisce la chiave esterna con la corrispondente chiave primaria, come descritto nell'esempio che segue.

Esempio 1.7.1 - Ampliamo lo scarno scenario descritto nell'esempio 1.5.1 aggiungendo i seguenti fatti: ciascun dipendente lavora in un reparto; ogni reparto ha un responsabile. Questa situazione è descritta nella figura che segue dove è descritto uno schema relazionale corrispondente alla base di dati riportata nell'esempio 1.5.1.



Già da quanto descritto nello schema precedente si capisce che le relazioni che costituiscono una base di dati sono fra loro associate logicamente e permettono di risolvere interrogazioni coinvolgenti più relazioni, come ad esempio: *Determinare il cognome ed il nome del responsabile di un dato dipendente.*
Il precedente schema relazionale può essere popolato come descritto nella seguente basi di dati relazionale.

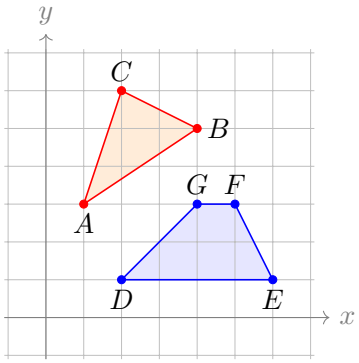
DIPENDENTE

Matricola	*Cognome*	*Nome*	*Stipendio*	*Reparto*
AN01	Antico	Nicola	1500	P
BA01	Barbieri	Amedeo	1870	P
GF03	Gaio	Felice	1500	P
FG01	Ferretti	Gaetano	1450	P
AF01	Armellini	Francesca	1750	A
DC02	Donati	Claudia	1500	A
RG02	Rovetta	Gianni	1780	M
MD01	Marini	Daniele		
AF01	Abete	Fiorello	1550	
BD05	Bonato	Diego	1600	S

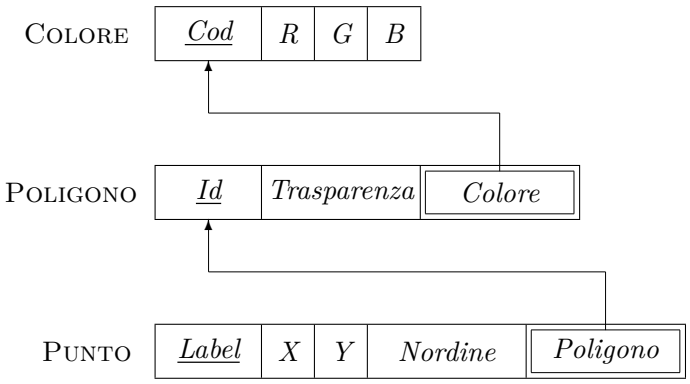
REPARTO

Sigla	*Descrizione*	*Telefono*	*Responsabile*
A	amministrazione	210	AF01
P	produzione	204	BA01
M	magazzino	254	RG02
S	spedizione	212	RG02
K	marketing	212	

Problema 1.7.1 Descrivere uno schema relazionale adatto a rappresentare un disegno nel piano cartesiano composto da generici poligoni colorati; ogni poligono ha tutti i lati dello stesso colore ed un colore di riempimento uguale a quello del suo bordo, con un fattore di trasparenza compreso fra 0 (vuoto) ed 1 (pieno, con il colore del bordo). Ogni vertice è univocamente individuato da una label. Popolare lo schema in modo da rappresentare la figura che segue.

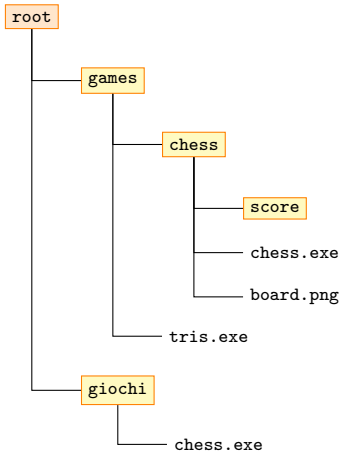


Soluzione. L'idea di fondo della rappresentazione consiste nel identificare un poligono mediante una *sequenza circolare* (anello, ring) dei suoi punti. Lo schema che segue costituisce una soluzione del problema proposto: *Cod* è un codice (numerico o stringa) che identifica univocamente un colore, R, G, B sono le sue tre componenti rgb, *Id* è un numero intero che identifica univocamente un poligono, *Trasparenza* è un valore numerico compreso nell'intervallo $[0, 1]$ che specifica il grado di trasparenza del colore di riempimento del poligono, *Colore* è la chiave esterna che individua il colore (dei lati e del riempimento) del poligono, *Label* è la stringa (A, B, C, ...) del nome di ciascun punto, X, Y le coordinate numeriche del punto, *Nordine* è un numero naturale che denota l'indice di posizione del punto nella sequenza dei punti di ciascun poligono, *Poligono* è la chiave esterna che individua a quale poligono appartiene il punto.

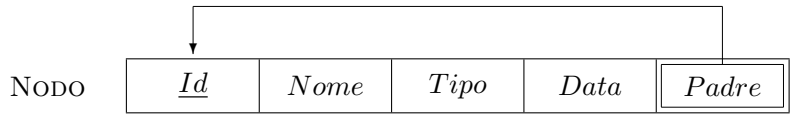


Si lascia per esercizio l'estensione dello schema con i dati che rappresentano la figura. □

Problema 1.7.2 Rappresentare mediante il modello relazionale una porzione porzione di file system, organizzato ad albero, composto da cartelle e file. Lo schema deve imporre il vincolo che una cartella non possa contenere cartelle aventi lo stesso nome. Sullo schema definito, individuare le chiavi candidate, le chiavi primarie e le chiavi esterne. Estendere lo schema in modo da rappresentare la situazione descritta nella figura che segue.



Soluzione. Una modalità di rappresentazione consiste nell'individuare ciascun nodo dell'albero mediante un identificatore univoco, ad esempio mediante un numero naturale. A questo identificatore vanno aggiunti i dati relativi al valore del nodo. La posizione del nodo all'interno della struttura viene specificata mediante il riferimento al nodro padre e mediante il posizionamento a sinistra (0) o destra (1) rispetto al padre. Mediante il modello logico relazionale questa situazione può essere rappresentata mediante una sola relazione le cui tuple registrano i dati di ciascun nodo dell'albero (cartella o file). Ogni nodo può essere identificata mediante un numero naturale. La posizione di ciascun nodo all'interno della struttura viene specificata mediante il riferimento al nodo padre. Altri dati satellite di ciascuna cartella possono essere, ad esempio: la data di creazione, la data dell'ultima modifica e simili. Dati quali il livello (nell'albero) o il numero di sottocartelle di ciascuna cartella richiedono una particolare attenzione nelle operazioni di modifica della base di dati, in quanto potrebbero generare delle inconsistenze; è preferibile, pertanto, che tali dati vengano calcolati al momento in cui servono, derivandoli dagli altri memorizzati. Tutte queste considerazioni possono essere sintetizzate mediante lo schema relazionale riportato nella figura che segue. Imponendo che $K = \{Nome, Padre\}$ sia chiave candidata, si garantisce che le cartelle figlie di uno stesso padre abbiano nomi distinti.



A seguire è riportata l'estensione della base di dati che descrive la struttura ad albero riportata nel testo dell'esercizio.

Cartella

Id	*Nome*	*Tipo*	*Data*	*Padre*
0	root	C	2018/7/20	NULL
1	games	C	2018/8/21	0
2	chess	C	2018/3/12	1
3	score	C	2018/10/2	2
4	chess.exe	F	2018/7/20	2
5	board.png	F	2018/7/20	2
6	tris.exe	F	2018/9/20	1
7	giochi	C	2018/9/20	0
9	scacchi.exe	F	2019/4/18	7

□

1.8 Vincoli

I *vincoli* sono proprietà che devono essere soddisfatte dai valori che popolano le relazioni. I vincoli vengono definiti sullo schema, a livello intensionale, ed agiscono su tutte le istanze. A seconda degli elementi che ne sono coinvolti, i vincoli si possono suddividere in:

- *vincoli intrarelazionali* : sono dei vincoli definiti su una singola relazione. Fra questi ci sono i seguenti:
  - *vincoli di dominio* : sono delle restrizioni sui valori assumibili in corrispondenza di un particolare attributo; ad esempio, il voto conseguito in un esame universitario deve essere un numero intero compreso fra 18 e 30
  - *vincoli di tupla* : sono definiti su ciascuna tupla, indipendentemente dalle altre; ad esempio, nel caso il voto in un esame universitario sia 30, il campo lode può essere assegnato a `TRUE`
  - *vincoli di chiave* : sulle chiavi candidate non sono ammessi valori duplicati; sulle chiavi primarie non sono ammessi valori nulli (e neanche duplicati)
- *vincoli interrelazionali* : sono vincoli che coinvolgono più relazioni; fra questi il più importante è il *vincolo di integrità referenziale* descritto nella seguente definizione.

DEFINIZIONE 5. Data una relazione $\mathcal{R}$ (relazione *primaria*) avente chiave primaria K_p ed una relazione $\mathcal{S}$ (relazione *secondaria*) avente chiave esterna K_f, il *vincolo di integrità referenziale* (in inglese *referential integrity constraint*) è soddisfatto se per ogni valore non nullo della chiave esterna K_f nella relazione $\mathcal{S}$ esiste un corrispondente valore della chiave primaria K_p nella relazione primaria $\mathcal{R}$.

Osservazione. In una base di dati relazionale la definizione delle chiavi (candidate, primarie, esterne) costituisce un meccanismo per imporre, a livello intensionale, dei vincoli sui dati che popolano la base di dati, in modo tale da riflettere dei vincoli logici del minimondo reale modellizzato e rappresentato mediante lo schema della base di dati.

In uno schema relazionale la definizione delle chiavi candidate e delle chiavi esterne induce dei vincoli sui valori che potranno popolare lo schema stesso.

Esempio 1.8.1 - Con riferimento allo schema relazionale dell'esempio 1.7.1 imporre che l'insieme di attributi $\{Cognome, Nome\}$ costituisca una chiave candidata significa imporre che non si possano avere dipendenti con lo stesso cognome e nome; il vincolo di integrità referenziale fra $\{Reparto\}$ e $\{Sigla\}$ comporta che esista nella relazione REPARTO il corrispondente valore della *Sigla*. Definire l'insieme di attributi $\{Responsabile\}$ come chiave candidata significa imporre il vincolo che un dipendente possa essere responsabile di al più un reparto e ciò, unitamente al vincolo di chiave primaria *Sigla*, comporta che ogni reparto abbia un solo responsabile.

Osservazione. Ci possono essere dei vincoli che non si riesce a definire mediante le chiavi, come, ad esempio, con riferimento all'esempio 1.7.1, il fatto che il responsabile di un reparto lavori nel reparto di cui è responsabile. Questi vincoli vengono gestiti a livello applicativo, con dei meccanismi esterni al modello relazionale.

1.9 I valori nulli

Nel modello relazionale l'assenza di un dato viene gestita assumendo che in ogni dominio ci sia un particolare valore, chiamato *valore nullo* e denotato con `NULL` o con ω. Il significato del valore nullo può essere di diverse tipologie; un rapporto ANSI del 1975 ha individuato le seguenti categorie di significati del valore nullo:

1. non applicabile
2. applicabile ma al momento inestistente
3. esistente ma riservato
4. esistente ma non disponibile
5. esistente ma al momento non memorizzato
6. memorizzato ma successivamente cancellato

7. memorizzato ma ancora non confermato
8. disponibile ma in corso di aggiornamento
9. disponibile ma non affidabile
10. disponibile ma non valido
11. non accessibile perchè la relazione in cui compare è bloccata
12. non accessibile perchè la tupla in cui compare è bloccata
13. momentaneamente bloccato
14. derivato da dati contenenti valori nulli

Dal punto di vista della teoria del modello relazionale è possibile ricondursi ai seguenti casi di valori nulli:

- valore inesistente
- valore sconosciuto;
- valore mancante (inesistente o sconosciuto)

Questi diversi casi sono descritti nell'esempio che segue.

Esempio 1.9.1 - Nella tabella riportata nella figura che segue in cui sono registrati i voti di esame di alcuni studenti, sono descritte le tre tipiche casistiche di valori nulli sopra elencate: *Verdi* è iscritto al corso di *Programmazione* ma non ha ancora svolto l'esame; *Bianchi* ha seguito il corso e superato l'esame di *Algoritmi* ma il voto dell'esame non è ancora stato registrato; *Rossi* è iscritto al corso di *Basi di Dati* ma non si sa se abbia superato l'esame e, se superato, quale voto abbia conseguito.

REGISTRO

Studente	*Corso*	*Voto*
Verdi	Programmazione	NULL
Bianchi	Algoritmi	NULL
Rossi	Basi di Dati	NULL

Nella teoria del modello relazionale le problematiche relative ai valori nulli si incontrano nei seguenti casi:

- interpretazione dei valori nulli
- operazioni di interrogazione in presenza di valori nulli
- vincoli di integrità su relazioni con valori nulli

Tutte queste problematiche sono state ampiamente analizzate e descritte in molti lavori di ricerca e sono state, per la maggior parte, recepite ed implementate nei vari DBMS. Rimangono tuttavia alcune nicchie di questioni a tuttoggi irrisolte e, per questo, ancora oggetto di ricerca.

1.10 Linguaggi per la definizione di schemi relazionali

Per poter utilizzare un DBMS è necessario tradurre lo schema ad un formato compatibile e comprensibile dal DBMS. A questo scopo i linguaggi orientati alla gestione delle basi di dati predispongono uno specifico insieme di istruzioni e funzionalità del DDL. Questi meccanismi vengono generalmente predisposti in modalità testuale (linguaggio SQL) oppure in modalità visuale che consente di operare in modo più agevole, anche se, poi, il tutto viene spesso tradotto in formato testuale, spesso in linguaggio SQL.

Per descrivere gli schemi relazionali abbiamo finora utilizzato una notazione grafico-testuale che ci ha consentito di definire la seguenti caratteristiche: nomi delle relazioni, nomi degli attributi, chiavi primarie e chiavi esterne. Altre informazioni quali il dominio di ciascun attributo e le chiavi candidate diverse dalle chiavi primarie devono essere indicate a parte. Oltre a questa modalità esistono specifici linguaggi (testuali e visuali) per definire uno schema relazionale, come descritto negli esempi che seguono.

Esempio 1.10.1 - Il linguaggio SQL è un linguaggio testuale che permette di definire in modo completo uno schema relazionale. SQL è un linguaggio case-insensitive ed a formato libero. Nonostante queste libertà di scrittura è bene, per questioni di leggibilità, adottare una modalità uniforme nell'uso dell'indentazione e delle convenzioni lessicali sull'uso dei caratteri minuscoli/-maiuscoli. A seguire, come esempio indicativo, è definito lo schema relazionale corrispondente all'esempio 1.7.1 già esaminato in precedenza.

```
CREATE DATABASE Azienda;

CREATE TABLE DIPENDENTE (
    Matricola CHAR(4),
    Cognome   CHAR(20),
    Nome      CHAR(20),
    Stipendio INTEGER,
    Reparto   CHAR(2),
    PRIMARY KEY (Matricola));

CREATE TABLE REPARTO (
    Sigla        CHAR(2),
    Descrizione  CHAR(15),
    Telefono     INTEGER,
    Responsabile CHAR(4) DEFAULT NULL,

    CHECK (Telefono>0),
    PRIMARY KEY (Sigla),
    UNIQUE (Descrizione),
    FOREIGN KEY (Responsabile)
        REFERENCES DIPENDENTE(Matricola)
        ON UPDATE CASCADE
        ON DELETE SET NULL);
```

```
ALTER TABLE DIPENDENTE
    ADD FOREIGN KEY (Reparto)
        REFERENCES REPARTO(Sigla);

CREATE UNIQUE INDEX Id1 ON DIPENDENTE(Matricola);
CREATE INDEX Id2 ON DIPENDENTE(Cognome,Nome);
```

Alcuni commenti:

- *CREATE TABLE* è l'istruzione per definire una relazione (detta *tabella* nella terminologia del linguaggio SQL)
- la specifica *DEFAULT* denota il valore di default di un attributo, nel caso in cui, in fase di inserimento, non venga specificato alcun valore
- *CHECK* denota un vincolo sul valore che viene inserito in una tupla in corrispondenza di un dato attributo
- *PRIMARY KEY* denota una chiave primaria (composta da uno o da più attributi); non sono ammessi valori nulli
- *UNIQUE* denota una chiave candidata (composta da uno o da più attributi); sono ammessi valori nulli
- *FOREIGN KEY* denota una chiave esterna (e la corrispondente chiave primaria a cui fa riferimento)
- *ON UPDATE* specifica come deve essere gestito dal DBMS il vincolo di integrità referenziale in caso di operazione di modifica (*UPDATE*) o inserimento (*INSERT*): *CASCADE* indica che, in caso di modifica del valore di una chiave primaria nella tabella principale, venga automaticamente modificato il corrispondente valore della chiave esterna (in modo da mantenere il legame) nella tabella secondaria; un'altra frequente specifica è *ON DELETE SET NULL* che specifica di porre uguale al valore nullo le chiavi esterne nel caso in cui venga eliminata la tupla alla quale le chiavi esterne fanno riferimento; è da usare con molta cautela la clausola *ON DELETE CASCADE* che darebbe indicazione al DBMS, nel caso di eliminazione di una tupla nella tabella principale, di eliminare, nella tabella secondaria, le tuple associata alla tupla eliminata.
- *ALTER TABLE* permette di modificare la precedente definizione di una relazione; nel caso specifico permette di aggiungere una chiave esterna nel caso in cui due relazioni siano mutualmente relazionate mediante due chiavi esterne; questa posticipazione della definizione della chiave esterna si rende necessaria nel caso in cui il processamento del programma SQL avvenga in una singola passata sequenziale in cui si farebbe riferimento (nella definizione della prima chiave esterna) ad una tabella non ancora definita.
- *CREATE INDEX* da l'indicazione al DBMS di creare, a livello fisico, un indice per velocizzare gli accessi alle tuple qualora vengano utilizzate come chiavi di accesso gli attributi indicati; con il qualificatore *UNIQUE* non viene permesso che vengano inserite tuple duplicate sugli attributi indicati.

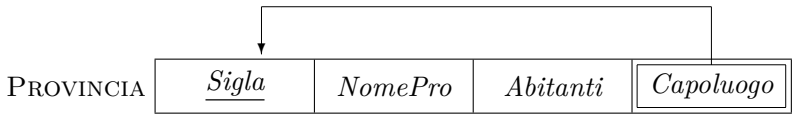
Il comando *CREATE TABLE* è caratterizzato da molte altre caratteristiche, per le quali si rimanda al manuale del linguaggio.

1.11 Estensione di uno schema relazionale

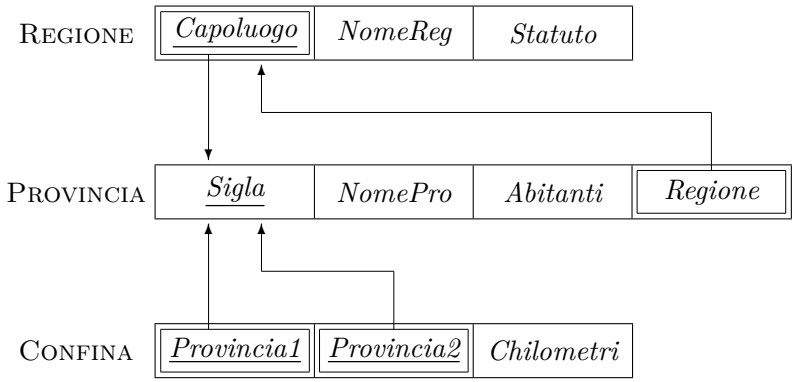
Estendere uno schema relazionale significa apportare delle modifiche integrative allo schema, mediante aggiunta o modifica di attributi, di chiavi e di relazioni. L'estensione di uno schema si rende necessaria qualora si voglia ampliare o modificare la porzione di realtà rappresentata mediante uno schema, oppure per allentare i vincoli indotti da uno schema relazionale.

Esempio 1.11.1 - Al fine di aggiungere un ulteriore dato relativo all'indirizzo di posta elettronica di ciascun dipendente, lo schema presentato nell'esempio 1.7.1 può essere esteso aggiungendo un attributo *Email* nella relazione DIPENDENTE.

Esempio 1.11.2 - Lo schema relazionale che segue descrive alcuni dati relativi alle province italiane; la chiave esterna *Capoluogo* denota la sigla della provincia capoluogo della regione in cui si trova la provincia considerata.



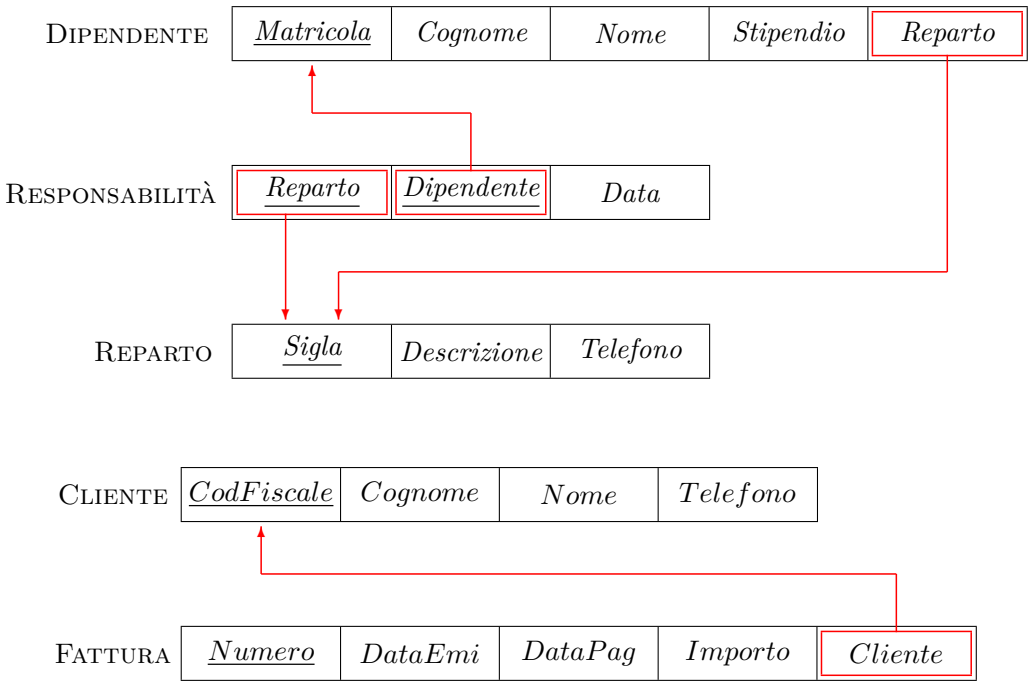
Questo schema può essere esteso includendo i dati geografici relativi alle regioni italiane nelle quali si trovano le varie province ed ai loro confinamenti, come descritto nello schema relazionale che segue:



Problema 1.11.1 Estendere lo schema presentato nell'esempio 1.7.1 in modo da gestire le seguenti variazioni:

1. un reparto può avere più responsabili ed un dipendente può essere responsabile di più reparti
2. serve registrare la data di quando un responsabile è diventato responsabile del reparto
3. serve gestire i dati amministrativi dei clienti dell'azienda e gli estremi dei dati contabili delle fatture loro emesse al fine di gestire i pagamenti delle fatture stesse

Soluzione. Una soluzione è fornita dal seguente schema relazionale.



□

Osservazione. La definizione dello schema di una base di dati viene realizzata mediante la definizione di uno schema relazionale ed eventuale sua successiva estensione solo nel caso di schemi molto elementari, composti da poche relazioni. Per situazioni più complesse ed articolate la definizione dello schema di una base di dati viene realizzata con sistemi a più alto livello, più facili e produttivi: è quanto si vedrà più avanti quando si progetteranno le basi di dati mediante dei "modelli concettuali" che verranno successivamente tradotti a più basso livello in uno schema relazionale.

1.12 Ridondanza ed anomalie

Una delle finalità delle basi di dati è rappresentata dall'eliminazione della *ridondanza* dei dati, ossia della presenza di dati duplicati. Al di là di un evidente obiettivo di efficienza nell'occupazione della memoria, la ridondanza costituisce la causa di anomalie che si possono verificare durante le fasi di elaborazione di una base di dati.

Le *anomalie* sono delle situazioni indesiderate che si verificano su una base di dati relazionali a causa della cattiva definizione dello schema relazionale. Queste situazioni sono descritte nell'esempio che segue.

Esempio 1.12.1 - La tabella che segue contiene le informazioni relative ai dipendenti ed ai loro reparti di assegnazione.

AZIENDA

Cognome	*Nome*	*Reparto*	*Telefono*
Antico	Nicola	Produzione	204
Barbieri	Amedeo	Produzione	204
Gaio	Felice	Produzione	204
Ferretti	Gaetano	Produzione	204
Armellini	Francesca	Amministrazione	210
Donati	Claudia	Amministrazione	210
Rovetta	Giorgio	Magazzino	254
Bonato	Diego	Spedizione	212

Lo schema di questa relazione offre il fianco al verificarsi delle seguenti situazioni di anomalie:

- *anomalie in inserimento*: non è possibile inserire i dati di un reparto se non viene contestualmente inserito un dipendente che lavora in quel reparto
- *anomalie in modifica*: se viene modificato il nome o in numero di telefono di un reparto, bisogna replicare queste modifiche a tutte le tuple che registrano i dati di quel reparto
- *anomalie in cancellazione*: se viene eliminata la tupla di un dipendente che lavora da solo in un reparto, vengono eliminate anche le informazioni relative a quel reparto

Per evitare di incorrere nei casi di anomalie precedentemente esaminati, si considerano delle particolari classi di schemi di relazione *ben fatti*, detti *forme normali*. La definizione delle forme normali si basa sul concetto di dipendenza funzionale che è l'oggetto del prossimo paragrafo.

1.13 Dipendenze funzionali

Le *dipendenze funzionali* (abbreviato in *df*) costituiscono dei particolari vincoli per il modello relazionale, definiti mediante dei legami di tipo funzionale tra gli

attributi della relazione. Una df tra gli attributi A di uno schema $\mathcal{R}(A)$ è un vincolo intensionale sulle relazioni di schema $\mathcal{R}(A)$ tale che ogni ennupla delle relazioni con schema $\mathcal{R}(A)$ deve soddisfare tale vincolo. In termini precisi una df può essere definita come segue.

DEFINIZIONE 6. Siano X, Y due sottoinsiemi non vuoti di attributi di una relazione $\mathcal{R}$. Si dice che X *determina funzionalmente* Y o, equivalentemente, che Y *dipende funzionalmente da* X e si scrive $X \to Y$ se per ogni coppia di tuple $t_1, t_2 \in \mathcal{R}$ per le quali $t_1[X] = t_2[X]$ si deve avere che $t_1[Y] = t_2[Y]$, ossia se le due tuple t_1 e t_2 hanno gli stessi valori sugli attributi di X allora hanno gli stessi valori sugli attributi di Y. In altri termini, $X \to Y$ se in una tupla i valori in corrispondenza degli attributi di X determinano univocamente i valori in corrispondenza degli attributi Y. L'insieme di attributi X è detto *parte sinistra* mentre Y è detto *parte destra*. Si dice anche che X è un *determinante* per Y. Una df $X \to Y$ può essere interpretata anche affermando che esiste una relazione funzionale f tale che $Y = f(X)$. Si parla di *dipendenza* (*funzionale*) *transitiva* fra tre insiemi X, Y, Z di attributi quando: se ($X \to Y$ *e* $Y \to Z$) allora $X \to Z$. In questo caso si dice che Z *dipende transitivamente* da X.

Come conseguenza della precedente definizione si ha: se X è una chiave condidata di una relazione, allora X determina funzionalmente ogni attributo della relazione.

Osservazione. Le df sono dei vincoli di natura semantica sui dati, che riflettono dei vincoli del mondo reale rappresentato mediante i dati.

Esempio 1.13.1 - La seguente relazione descrive le informazioni relative all'occupazione delle aule nell'orario scolastico settimanale:

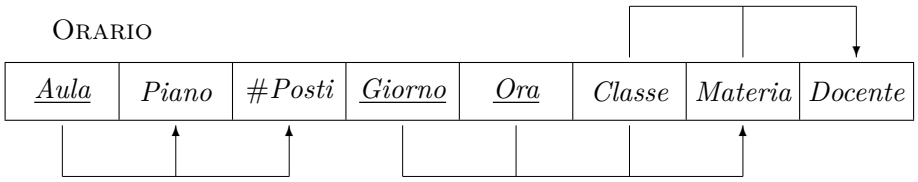
ORARIO

Aula	*Piano*	*#Posti*	*Giorno*	*Ora*	*Classe*	*Materia*	*Docente*
16	terra	25	Lun	I	5IT	Storia	Fabbri
16	terra	25	Lun	II	5IT	Sistemi	Roldo
20	terra	28	Mar	IV	2BS	Storia	Fabbri
5	primo	27	Lun	III	2MM	Diritto	Bini
8	primo	27	Lun	V	2MM	Scienze	Grando
16	terra	25	Lun	III	5IT	Lettere	Fabbri

Su questo schema consideriamo le seguenti df con il vincolo descritto in corrispondenza a ciascuna:

- $Aula \to Piano, \#Posti$: ogni aula si trova ad un dato piano ed ha un fissato numero di posti
- $Classe, Materia \to Docente$: in ogni classe, una materia è insegnata da un solo docente
- $Giorno, Ora, Classe \to Materia$: in una determinata ora del giorno una classe segue una sola materia

Le df possono essere descritte mediante uno schema grafico come descritto nella figura che segue.



1.14 Forme normali

Le *forme normali* costituiscono dei modelli di schemi di relazione *normali*, ossia *ben fatti*. Vengono definiti descrivendo le proprietà che devono essere soddisfatte dagli schemi di relazione ed assicurano la proprietà che ciascuna relazione rappresenta delle informazioni che sono concettualmente distinte da quelle delle altre relazioni. Le forme normali costituiscono una tecnica fondamentale nella progettazione delle basi di dati relazionali. Le df costituiscono lo strumento per definire le forme normali.

Prima forma normale (*1NF*)

La prima forma normale è un requisito fondamentale per il modello relazionale; può essere descritta mediante la seguente definizione.

DEFINIZIONE 7. Una relazione si dice in *prima forma normale* (*First Normal Form* o $1NF$) se soddisfa alle seguenti proprietà:

1. tutte le tuple hanno lo stesso numero di attributi
2. i valori in corrispondenza a ciascun attributo sono dello stesso tipo
3. i domini degli attributi sono semplici (non strutturati)
4. tutte le tuple sono fra loro distinte
5. ogni riga, in corrispondenza a ciascun attributo, contiene un solo valore (attributi univalore)
6. l'ordine delle tuple è ininfluente

Esempio 1.14.1 - Esempio di relazione in prima forma normale:

ISBN	*Titolo*	*Editore*	*Prezzo*
9783161484100	`Mari del Nord`	`Wiki Books`	26.80
9788832547712	`Terremoti`	`Libri & Libri`	25.50
9781745363837	`Esploratori`	`Explora`	29.60
9786458353637	`Africa selvaggia`	`Wiki Books`	24.90
9787385226262	`Il Tibet`	`Explora`	25.00
9788826411086	`Il Kilimangiaro`	`EdiViaggi`	19.50
9783122200340	`Galapagos`	`Wiki Books`	28.00

Esempio 1.14.2 - Consideriamo la seguente relazione in cui il significato di ogni tupla è: *All'ora* Ora *sulla linea* Linea *avente* Origine *come capolinea di partenza e* Destin *come capolinea di arrivo parte l'autobus con numero* #Bus *avente una capienza di* #Posti *posti.*

Ora	*Linea*	*Percorso*	*#Bus*	*#Posti*
7.30	rossa	Stazione → Centro	215	55
8.00	rossa	Stazione → Centro	174	57
	verde	Mercato → Ospedale	102	48
	gialla	Stazione → Ospedale	141	54
8.30	rossa	Stazione → Centro	102	48
	gialla	Stazione → Ospedale	174	57

Questa relazione non è in $1NF$; per portarla alla $1NF$ è sufficiente suddividere in più tuple le tuple con valori multipli:

Ora	*Linea*	*Origine*	*Destinazione*	*#Bus*	*#Posti*
7.30	rossa	Stazione	Centro	215	55
8.00	rossa	Stazione	Centro	174	57
8.00	verde	Mercato	Ospedale	102	48
8.00	gialla	Stazione	Ospedale	141	54
8.30	rossa	Stazione	Centro	102	48
8.30	gialla	Stazione	Ospedale	174	57

Su questo schema relazionale, oltre alle df banali, si possono assumere le seguenti df:

1. $\#Bus \rightarrow \#Posti$: ogni autobus ha una sua specifica capienza
2. $Linea \rightarrow Origine, Destinazione$: ogni linea ha una specifica origine e destinazione
3. $Origine, Destinazione \rightarrow Linea$: l'origine e la destinazione individuano univocamente la linea

Seconda forma normale (*2NF*)

Un *attributo primo* o *attributo chiave* è un attributo che fa parte di una chiave candidata.

DEFINIZIONE 8. Uno schema di relazione $\mathcal{R}$ è in *seconda forma normale* (*2NF*) se è *1NF* e ogni attributo non primo non dipende in modo parziale da alcuna delle chiavi candidate di $\mathcal{R}$.

Come immediata conseguenza della precedente definizione si ha: se le chiavi di una relazione sono costituite da un solo attributo, allora la relazione è *2NF*.

Esempio 1.14.3 - La relazione riportata nell'esempio precedente non è *2NF* in quanto la df

$$Linea \rightarrow Origine, Destinazione$$

evidenzia che esistono attributi non primi che dipendono solo parzialmente dalla chiave. La relazione può essere trasformata in *2NF*, scomponendola in due relazioni Corsa e Linea:

Corsa

Ora	*Linea*	*#Bus*	*#Posti*
7.30	rossa	215	55
8.00	rossa	174	57
8.00	verde	102	48
8.00	gialla	141	54
8.30	rossa	102	48
8.30	gialla	174	57

Linea

Linea	*Origine*	*Destinazione*
rossa	Stazione	Centro
verde	Mercato	Ospedale
gialla	Stazione	Ospedale

Terza forma normale (*3NF*)

DEFINIZIONE 9. Uno schema di relazione $\mathcal{R}$ è in *terza forma normale* (*3NF*) se e solo se è *2NF* e ogni attributo non primo non dipende transitivamente da alcuna delle chiavi candidate di $\mathcal{R}$. Equivalentemente, visto che ogni chiave determina funzionalmente ogni attributo non primo, ogni attributo non deve dipendere funzionalmente da un attributo non primo.

Esempio 1.14.4 - Le relazioni riportate nell'esempio precedente sono *2NF* ma presentano ancora evidenti ridondanze. Infatti, dalla catena di dipendenze funzionali

$$Ora, Linea \rightarrow \#Bus \rightarrow \#Posti$$

si vede che *#Posti* è un attributo che dipende solo transitivamente dalla chiave. La relazione CORSA riportata nell'esempio precedente può essere normalizzata alla *3NF* scomponendola nelle due relazioni CORSA e BUS come sotto descritto.

CORSA

Ora	*Linea*	*#Bus*
7.30	`rossa`	215
8.00	`rossa`	174
8.00	`verde`	102
8.00	`gialla`	141
8.30	`rossa`	102
8.30	`gialla`	174

BUS

#Bus	*#Posti*
215	55
174	57
102	48
141	54

LINEA

Linea	*Origine*	*Destin*
`rossa`	`Stazione`	`Centro`
`verde`	`Mercato`	`Ospedale`
`gialla`	`Stazione`	`Ospedale`

Forma normale di Boyce-Codd ($BCNF$)

DEFINIZIONE 10. Una relazione $\mathcal{R}$ è in *forma normale di Boyce-Codd* (*BCNF, Boyce-Codd Normal Form*) se è *1NF* ed ogni determinate è una chiave candidata, cioè ogni insieme di attributi dal quale dipendono altri attributi può svolgere la funzione di chiave candidata. Equivalentemente: per ogni df $X \to Y$, l'insieme di attributi X è una chiave candidata.

Dalla precedente definizione discende che vengono eliminate tutte le dipendenze funzionali sia parziali sia transitive (essendo che ogni determinante è una chiave candidata, e quindi non dipende da nessun altro attributo). Una relazione che rispetta la *BCNF* è quindi anche *3NF*; non vale però il viceversa. La *BCNF* è quindi una forma di normalizzazione più forte della *3NF*. Questo fatto può essere dedotto anche dall'osservazione che la *3NF* richiede che ogni attributo non-chiave dipenda da un insieme di attributi che possono formare una chiave, mentre la *BCNF* lo richiede per *ogni* attributo.

La *BCNF* non impone che la relazione soddisfi nessun'altra forma normale di grado inferiore, ma è da sola sufficiente ad eliminare la ridondanza. Dal punto di vista pratico, nel processo di normalizzazione bisogna comunque passare attraverso i livelli inferiori.

Osservazione. Nella teoria della basi di dati relazionali, oltre alla *BCNF* vengono considerate anche delle forme di normalizzazione più restrittive (*4NF* e *5NF*) che permettono di evitare situazioni di anomalie nelle operazioni sulle tabelle. Queste forme di normalizzazione permettono inoltre di minimizazre il numero di attributi che compongono le chiavi composte. Queste forme hanno però il difetto (come anche la *BCNF*) di perdere delle df. La *3NF* ha invece il vantaggio di essere sempre raggiunta senza perdita di df; pertanto, la *3NF* è quella che solitamente offre il compromesso ottimale.

1.15 Analisi e normalizzazione di schemi relazionali

Analizzare il grado di normalizzazione di uno schema relazionale significa, per ciascuna relazione, svolgere i seguenti passi:

1. definire delle chiavi candidate, motivandone le scelte
2. definire delle dipendenze funzionali, spiegandone il significato
3. analizzare il grado di normalizzazione di ciascuna relazione dello schema

Problema 1.15.1 Analizzare il grado di normalizzazione della seguente relazione che riporta alcuni dati relativi ai comuni italiani:

ITALIA	*Comune*	*Abitanti*	*AltezzaSLM*	*Provincia*	*Regione*

Soluzione. Osservando che in Italia ci sono comuni con lo stesso nome ma localizzati in province e regioni diverse, si evidenzia la seguente chiave candidata $K_1 = \{Comune, Provincia\}$. Esiste inoltre la df *Provincia* $\to$ *Regione* in cui un attributo dipende parzialmente dalla chiave; pertanto non è $2NF$.

Mediante il processo di *normalizzazione*, una relazione viene scomposta in più relazioni che complessivamente forniscono le stesse informazioni della relazione di partenza. Le informazioni presenti nella relazione originale possono essere recuperate mediante delle operazioni di join. Nello schema relazionale che si ottiene vengono generalmente mantenute le df che vigevano sulla relazione originale ed inoltre, come effetto collaterale benefico, vengono risolti i problemi di ridondanza, di inconsistenza ed evitate le situazioni di anomalia.

Il procedimento di normalizzazione di una relazione può essere ottenuto mediante delle operazioni di scomposizione di schemi di relazione, applicando il seguente algoritmo.

Algoritmo 1 - Normalizzazione di uno schema di relazione $\mathcal{R}$

Input: schema di relazione $\mathcal{R}$
Output: schema relazionale Σ equivalente allo schema di relazione $\mathcal{R}$

identifica tutte le chiavi candidate
identifica tutte le df
$\Sigma \leftarrow \{\mathcal{R}\}$
while in Σ ci sono schemi di relazione non normalizzati **do**
  considera uno schema di relazione non normalizzato $\mathcal{T}$
  individua l'insieme $\mathcal{D}$ delle df su $\mathcal{T}$
  for all $d \in \mathcal{D}$ che viola la regola di normalizzazione **do**
    costruisci un nuovo schema di relazione $\mathcal{T}_1$ con tutti gli attributi di d
    assumi il determinante di d come chiave primaria di $\mathcal{T}_1$
    costruisci un secondo schema di relazione $\mathcal{T}_2$ rimuovendo da $\mathcal{T}$ tutti gli attributi che sono determinati da d
    $\Sigma \leftarrow \Sigma \setminus \{\mathcal{T}\} \cup \{\mathcal{T}_1, \mathcal{T}_2\}$
  end for
end while
return schema relazionale Σ

Problema 1.15.2 Normalizzare alla *2NF* il seguente schema relazionale:

$$R(\underline{A}, \underline{B}, C, D, E)$$

in cui valgono le seguenti due df:

d_1: $A \rightarrow D$

d_2: $B \rightarrow E$

La relazione R non è *2NF* in quanto ci sono degli attributi non primi che dipendono parzialmente dalla chiave. Applichiamo l'algoritmo 1 di normalizzazione. Considerando la prima df d_1 si ottengono le seguenti due relazioni:

$R_1(\underline{A}, D)$
$R_2(\underline{A}, \underline{B}, C, E)$

La relazione R_2 non è $2NF$ a causa della df d_2. Applicando l'algoritmo 1 questa relazione può essere scomposta come segue:

$R_3(\underline{B}, E)$
$R_4(\underline{A}, \underline{B}, C)$

Complessivamente la relazione R originale viene scomposta nel seguente schema relazionale, dove tutte le relazioni sono *2NF*:

$R_1(\underline{A}, D)$
$R_3(\underline{B}, E)$
$R_4(\underline{A}, \underline{B}, C)$

In questo schema non ci sono attributi non primi che dipendono transitivamente dalla chiave; si è ottenuta pertanto una normalizzazione alla *3NF*.

Problema 1.15.3 Analizzare il grado di normalizzazione della seguente relazione:

$$R(\underline{A}, B, C, D, E)$$

in cui valgono le seguenti df:

$B \to C$
$B \to D$

Normalizzare lo schema alla terza forma normale.

Soluzione. La relazione è $2NF$ in quanto non ci sono delle df parziali dalla chiave. Considerando la df banale $A \to B$ si nota che C e D sono attributi non-primi che dipendono transitivamente dalla chiave; pertanto la relazione non è $3NF$. Applicando l'algoritmo di normalizzazione si ottiene la seguente scomposizione:

$R_1(\underline{B}, C)$
$R_2(\underline{A}, B, D, E)$

La relazione R_2, non essendo in *3NF*, viene ulteriormente scomposta nelle due relazioni

$R_3(\underline{B}, D)$
$R_4(\underline{A}, B, E)$

Lo schema $\{R_1, R_3, R_4\}$ costituisce il risultato della normalizzazione della relazione R.

ESERCIZI

1.1 Rappresentare mediante il modello relazionale un *multiinsieme* (detto anche *sacco* o *bag*). Un *multiinsieme* è una sorta di insieme in cui gli elementi sono presenti con una specifica frequenza; ad esempio $\{a, a, a, b, c, c\}$ è un multiinsieme in cui l'elemento a è presente con frequenza 3.

1.2 Rappresentare mediante il modello relazionale una *sequenza* di elementi di una stessa categoria, come descritto nella seguente figura. Rappresentare ciascun elemento mediante una tupla di una relazione ovvero mediante una riga di una tabella. In questo caso è essenziale osservare che una relazione (tabella) è composta da un *insieme* di tuple (righe) e che la sequenzialità della descrizione è ininfluente.

1	2	3	4	5
a	b	c	d	e

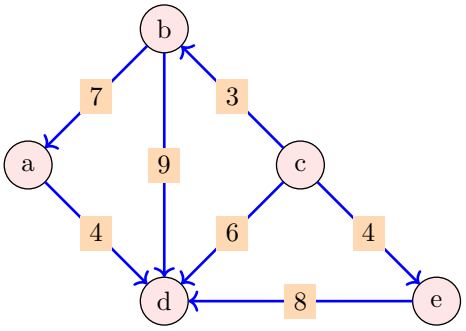
1.3 Rappresentare mediante il modello relazionale un *bucket* di elementi di una stessa categoria, come descritto nella seguente figura. Un *bucket* è una sorta di array-sequenza in cui ad una stessa posizione possono essere presenti più elementi.

1	2	3	4	5
$a\ b$		c	$d\ e\ f$	$g\ h$

1.4 Rappresentare mediante il modello relazionale il seguente *bucket* di *bag*:

1	2	3	4	5
$a\ a\ b$		c	$b\ d\ d$	$e\ e$

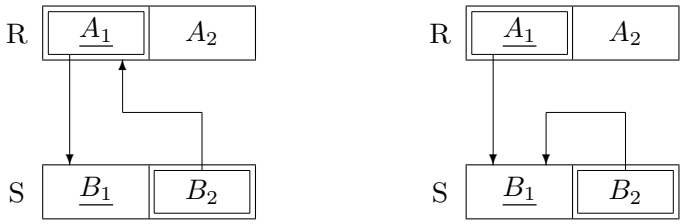
1.5 Rappresentare mediante il modello relazionale il grafo descritto nella seguente figura.



1.6 Popolare con alcuni dati significativi lo schema relazionale descritto nel problema 1.7.1.

1.7 Popolare con alcuni dati significativi lo schema relazionale descritto nel problema 1.11.1.

1.8 Discutere se i seguenti due schemi relazionali sono equivalenti.



1.9 Con riferimento alla base di dati descritta nell'esempio 1.3.1, individuare quali fra i seguenti vincoli sono esprimibili direttamente mediante il modello relazionale:

- due squadre non possono incontarsi più volte
- una squadra non può incontrare se stessa
- non è previsto il pareggio
- supera il turno la squadra che vince
- ogni squadra in ciascun turno gioca una sola partita
- ogni squadra gioca un turno se ha superato tutti i precedenti

1.10 Con riferimento allo schema della relazione REPARTO descritto nell'esempio 1.7.1, discutere le differenze di vincoli che si hanno imponendo come chiavi candidate le seguenti:

$K_1 = \{Sigla\}$
$K_2 = \{Descrizione\}$
$K_3 = \{Sigla, Descrizione\}$
$K_4 = \{Sigla, Descrizione, Telefono\}$

1.11 Con riferimento allo schema relazionale descritto nel problema 1.11.1, stabilire quali delle seguenti affermazioni sono vere:

1. ogni dipendente deve lavorare in almeno un reparto
2. un dipendente può lavorare in più reparti
3. ci possono esserere dipendenti con lo stesso cognome e nome
4. ci possono essere più responsabili di uno stesso reparto
5. ogni reparto deve avere almeno un responsabile
6. un dipendente può essere responsabile di al più un reparto
7. un dipendente può essere responsabile di più reparti
8. un responsabile deve lavorare nel reparto di cui è responsabile

9. in ogni reparto deve lavorare almeno un dipendente
10. ogni dipendente può lavorare in al più un reparto
11. ci deve esere almeno un dipendente in ciascun reparto
12. ci deve essere più di un dipendente in ciascun reparto
13. ogni dipendente deve lavorare in un qualche reparto
14. ogni dipendente deve lavorare in un solo reparto
15. ogni dipendente può lavorare in al più un reparto
16. ogni dipendente può lavorare in più di un reparto
17. ogni responsabile di reparto deve lavorare in uno dei reparti di cui è responsabile
18. ogni dipendente può essere responsabile di al più un reparto
19. ogni reparto deve avere almeno un responsabile
20. ogni reparto può avere più di un responsabile
21. ogni reparto può avere al massimo un responsabile
22. ogni reparto deve avere un solo responsabile
23. un dipendente, se è responsabile, è responsabile anche del reparto in cui lavora

1.12 Modificare lo schema relazionale dell'esempio 1.7.1 in modo da permettere che uno stesso dipendente possa essere responsabile di al più un reparto e che un reparto possa avere più responsabili.

1.13 Modificare lo schema relazionale dell'esempio 1.7.1 in modo da permettere che un dipendente possa lavorare in più di un reparto.

1.14 Con riferimento allo schema relazionale riportato nell'esempio 1.7.1, discutere il significato dei vincoli indotti singolarmente e complessivamente dalle seguenti definizioni di chiavi candidate:

$K_1 = \{Sigla\}$
$K_2 = \{Responsabile\}$
$K_3 = \{Sigla, Responsabile\}$

1.15 Con riferimento allo schema relazionale riportato nel problema **??**, discutere il significato dei vincoli indotti singolarmente e complessivamente dalle seguenti definizioni di chiavi candidate:

$K_1 = \{X, Y\}$
$K_2 = \{Nordine, Poligono\}$
$K_3 = \{X, Y, Nordine, Poligono\}$

1.16 Con riferimento al seguente schema relazionale adatto a rappresentare una porzione del registro elettronico:

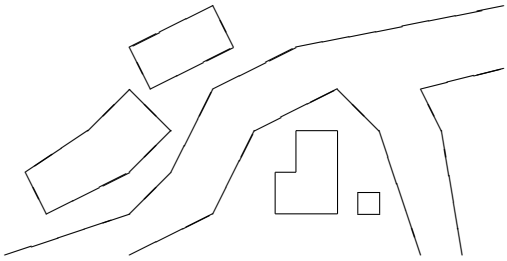
STUDENTE (*Matricola*, *Cognome*, *Nome*, *Classe*, *Nregistro*)
INTERROGAZIONE (*Studente*, *Materia*, *Voto*, *Data*)
CLASSE (*Sigla*, *Anno*, *Sezione*, *Indirizzo*, *Aula*)

1. individuare in ciascuna relazione delle possibili chiavi candidate
2. individuare in ciascuna relazione delle idonee chiavi primarie
3. associare fra loro le relazioni mediante delle chiavi esterne

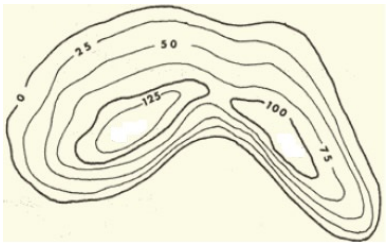
1.17 Estendere il modello utilizzato nell'esempio 1.3.4 in modo da utilizzarlo per rappresentare una base di dati secondo la seguente struttura:

- la base di dati è composta da *disegni*
- ciascun disegno è strutturato in *livelli*
- ciascun livello è composto da diverse *primitive grafiche* (segmenti, spezzate, simboli, ...)
- le primitive grafiche hanno degli *attributi* (colori, tipi di linea, ...)

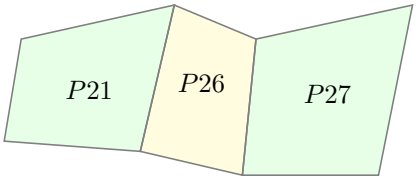
1.18 Rappresentare mediante il modello relazionale un disegno composto da *spezzate* e *poligonali chiuse*. Sono note le coordinate e le etichette dei vertici delle spezzate.



1.19 Descrivere uno schema di una base di dati relazionale per rappresentare i dati di una serie di curve di livello, composte da linee spezzate (aperte o chiuse), simili a quelle riportate in figura. Ad ogni curva è associato un codice che specifica il tipo di linea (sottile, spessa) ed un numero che esprime la quota s.l.m. della curva.



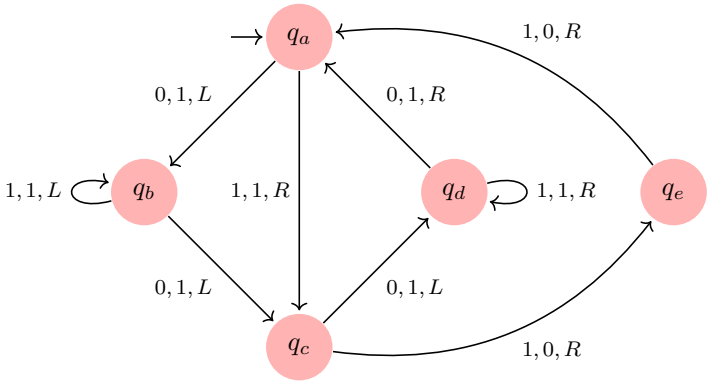
1.20 Rappresentare mediante il modello relazionale una porzione di mappa catastale che descrive delle particelle di terreni. Ciascuna particella è caratterizzata da un codice univoco e da una classificazione (agricola, edificabile, verde pubblico, ...). In fase di disegno le particelle vengono riprodotte con colori diversi in base alla loro classificazione catastale. I punti che delimitano le particelle sono individuati dalle loro coordinate. Si richiede di descrivere uno schema relazionale adatto a rappresentare la mappa e di individuare le chiavi primarie ed eventuali chiavi esterne. Successivamente estendere lo schema in modo da rappresentare la figura riportata a seguire.



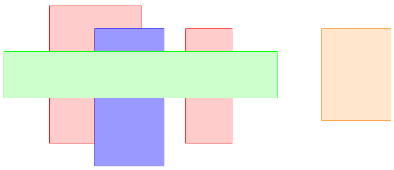
1.21 Il magazzino di una ditta gestisce tutti i componenti elementari per assemblare biciclette (forcelle, assi, cerchioni, raggi, pedali, corone, catene, selle, freni). Una bicicletta è composta dal telaio, dalle ruote, dal corpo di pedalata, dalla sella e dai freni. Il telaio è composto dalla forcella anteriore, dalla forcella posteriore e dall'asse. Ciascuna ruota è composta da un cerchione e da 24 raggi. Il corpo di pedalata è composto dai pedali, dalla corona e dalla catena. Descrivere mediante il modello relazionale una base di dati per registrare la composizione di una bicicletta, il numero delle componenti elementari presenti in magazzino ed il loro prezzo unitario. I dati registrati devono essere sufficienti a rispondere alle seguenti interrogazioni:

- costo complessivo dei pezzi necessari a comporre una bicicletta
- numero di biciclette che si riesce ad assemblare con i pezzi presenti
- quali e quanti pezzi serve ordinare per comporre n biciclette

1.22 Descrivere mediante il modello relazionale un grafo di transizione di stato di una macchina di Turing, simile a quello riportato nella figura che segue. Si richiede di descrivere dapprima uno schema relazionale e di individuare le chiavi primarie ed eventuali chiavi esterne. Popolare lo schema in modo da rappresentare la figura.



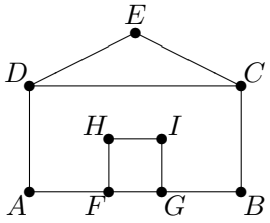
1.23 Si vuole rappresentare un insieme di rettangoli colorati disposti nel piano cartesiano, aventi i lati paralleli agli assi; si deve tener conto della reciproca posizione d'ordine sul piano, come descritto nella figura che segue.



1.24 Rappresentare mediante il modello relazionale un disegno composto da punti uniti mediante segmenti. Sono note le coordinate e le etichette dei vertici dei punti. I punti sono di due tipologie:

1. vertici di segmenti: possono essere condivisi fra più segmenti
2. punti cursore: sono ancorati ad un segmento e possono essere fatti scorrere sul segmento

Lo schema deve garantire la consistenza delle modifiche alle coordinate dei punti eseguite in fase di editing grafico.



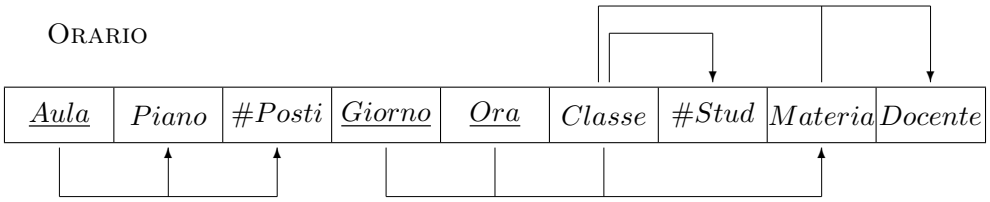
1.25 Con riferimento al seguente schema di relazione che descrive le informazioni relative ai dipendenti che lavorano nei vari reparti di un'azienda (l'attributo *Reparto* denota il codice del reparto nel quale lavora il dipendente e l'attributo *Resp* denota la matricola del dipendente che è responsabile del dato reparto) :

DIPENDENTE($\underline{Matricola}, Cognome, Nome, Mansione, Stipendio, Reparto, Resp$)

esprimere il significato delle seguenti df:

1. $Matricola \rightarrow Cognome, Nome$
2. $Cognome, Nome \rightarrow Matricola$
3. $Cognome, Nome \rightarrow Reparto$
4. $Mansione, Reparto \rightarrow Stipendio$
5. $Mansione \rightarrow Stipendio$
6. $Reparto \rightarrow Stipendio$
7. $Mansione \rightarrow Reparto$
8. $Reparto \rightarrow Resp$
9. $Resp \rightarrow Reparto$

1.26 Analizzare il grado di normalizzazione del seguente schema di relazione. Normalizzare lo schema alla terza forma normale.



1.27 Normalizzare la relazione descritta nell'esempio 1.13.1.

1.28 Analizzare il grado di normalizzazione della seguente relazione che riporta le informazioni della segreteria di un reparto ospedaliero per programmare gli interventi chirurgici da effettuare.

$$\text{INTERVENTO}(Paziente, Data, Ora, Chirurgo, Sala)$$

Normalizzare lo schema alla terza forma normale.

1.29 Analizzare il grado di normalizzazione del seguente schema di relazione che registra i dati relativi alle prenotazioni dei laboratori da parte degli insegnanti.

$$\text{PRENOTA}(Lab, \#Posti, Docente, Data, DalleOre, \#Ore, Classe, \#Studenti)$$

Per la definizione delle dipendenze funzionali si adottino i vincoli vigenti nell'istituto. Normalizzare lo schema alla terza forma normale.

1.30 Analizzare il grado di normalizzazione del seguente schema di relazione che registra i dati degli studenti che nel corrente anno scolastico vanno in visita di istruzione, accompagnati da un docente:

$$\text{VISITA}(Cognome, Nome, Classe, Meta, Dal, Al, Quota, Pagato, Docente)$$

Normalizzare lo schema alla terza forma normale.

1.31 Analizzare il grado di normalizzazione del seguente schema di relazione che registra le assenze degli studenti nel corrente anno scolastico:

$$\text{ASSENZA}(Cognome, Nome, Classe, Giorno, Ora, Materia, Docente)$$

Normalizzare lo schema alla terza forma normale.

2

INTERROGARE

La vecchietta, sull'autobus: "Ragazzino, sai dirmi quando devo scendere per andare a via Pasadena?"
Il ragazzino: "Guarda proprio me, e scendi due fermate prima che scenda io".

D. Knuth,
The Art of Computer Programming

Lo scopo fondamentale di una base di dati consiste nel memorizzare (molti) dati in modo che il DBMS che lo gestisce possa rispondere a delle interrogazioni ottenendo (velocemente), come risposta, dei dati da cui dedurre delle informazioni. Nel modello relazionale le risposte alle interrogazioni sono relazioni.

2.1 Interrogazioni

Le interrogazioni costituiscono il meccanismo mediante il quale gli utenti utilizzano una base di dati. Lo schema di questo meccanismo è descritto in figura 2.1.

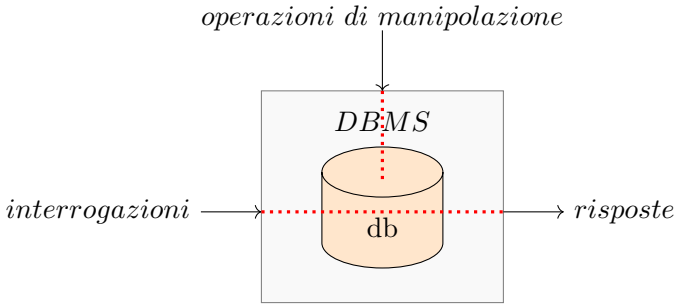


Figura 2.1: Operazioni di manipolazione ed interrogazioni su una base di dati.

In fase di progettazione di una base di dati, le interrogazioni possono fungere da controllo per verificare se i dati memorizzati siano completi.

Esempio 2.1.1 - Con riferimento all'esempio 1.3.2 si possono considerare le seguenti interrogazioni:

- Con quali regioni confina il *Veneto*?
- La *Liguria* confina con il *Veneto*?
- Quante regioni si devono attraversare (al minimo) per andare dal *Piemonte* al *Friuli*?
- Quali regioni confinano con una sola regione?

Rispetto a queste interrogazioni i dati memorizzati nella tabella CONFINAMENTO sono completi.

Una traduzione verso il modello relazionale è fatta correttamente se non fa perdere informazioni, almeno non quelle che interessano. Un test di verifica della qualità della traduzione può essere svolto considerando delle interrogazioni interessanti e verificando se le risposte a queste interrogazioni sono deducibile dallo schema ottenuto nella traduzione.

Esempio 2.1.2 - Con riferimento all'esempio 1.3.1 si possono considerare le seguenti interrogazioni:

- elenco delle squadre partecipanti al torneo
- classifica generale del torneo
- squadra che ha vinto il torneo
- squadre che hanno disputato la finale
- squadre incontrate dall'*Atletico*

- squadra che ha eliminato il *Real*
- numero di reti segnate dalla *Stella*
- numero di turni superati dallo *Sparta*
- elenco delle squadre eliminate al primo turno
- numero di reti subite dalla squadra vincitrice del torneo

Tutte queste interrogazioni possono ottenere risposta attingendo ai dati contenuti nella tabella TORNEO.

Nel modello relazionale il risultato di una interrogazione è sempre costituito da una relazione (rappresentata in forma tabellare).

Esempio 2.1.3 - La risposta all'interrogazione: *Regioni confinanti con il Veneto.* è costituita dalla relazione che segue:

CONFINANTEVENETO

Regione
Emilia
Friuli
Lombardia
Trentino

2.2 L'algebra relazionale

Per il modello relazionale delle basi di dati è definito un insieme di operazioni aventi come operandi delle relazioni. Questo insieme di operazioni costituisce l'*algebra relazionale* e soddisfa alla *proprietà di chiusura*: il risultato di un'operazione fra relazioni è ancora una relazione.

Mediante le operazioni dell'algebra relazionale si scrivono le espressioni che specificano le interrogazioni. Analogamente ad una espressione aritmetica, la valutazione di un'espressione dell'algebra relazionale fornisce una relazione come risultato; tale risultato costituisce la risposta ad una interrogazione sulla base di dati.

L'algebra relazionale costituisce un linguaggio formale di tipo procedurale per operare sui dati organizzati secondo il modello relazionale. L'algebra relazionale viene usata principalmente come strumento base per implementare ed ottimizzare le interrogazioni su una base di dati espresse mediante linguaggi a più alto livello dichiarativo, ad esempio mediante il linguaggio SQL.

2.3 Operatori insiemistici

Gli *operatori insiemistici* agiscono sulle relazioni considerandole come *insiemi* di tuple. Questi operatori richiedono che le relazioni coinvolte come operandi siano *compatibili* ossia che abbiano lo stesso grado e che gli attributi di stessa posizione siano uguali o perlomeno abbiano domini uguali ossia aventi lo stesso tipo. Nel caso in cui le relazioni siano compatibili ma abbiano schemi diversi (ossia i nomi degli attributi delle relazioni coinvolte siano diversi) si assume la convenzione che la relazione risultante abbia lo schema della prima relazione coinvolta nell'operazione. Gli operatori insiemistici sono descritti nella lista che segue.

- unione : Date due relazioni $\mathcal{R}$ ed $\mathcal{S}$, la loro unione, denotata con

  $$\mathcal{R} \cup \mathcal{S}$$

  è costituita dalle tuple che appartengono ad $\mathcal{R}$ oppure ad $\mathcal{S}$.

  Esempio 2.3.1 - Unione fra due relazioni:

A	B	C
a	3	x
b	1	y
a	2	y
a	1	x

$\cup$

A	B	C
b	1	y
a	1	y
a	3	x

$=$

A	B	C
a	3	x
b	1	y
a	2	y
a	1	x
a	1	y

- intersezione : Date due relazioni $\mathcal{R}$ ed $\mathcal{S}$, la loro intersezione, denotata con

  $$\mathcal{R} \cap \mathcal{S}$$

  è costituita dalle tuple che appartengono sia ad $\mathcal{R}$ che ad $\mathcal{S}$.

  Esempio 2.3.2 - Intersezione fra due relazioni:

A	B	C
a	3	x
b	1	y
a	2	y
a	1	x

$\cap$

A	B	C
a	1	y
a	3	x
b	1	y

$=$

A	B	C
a	3	x
b	1	y

- differenza : Date due relazioni $\mathcal{R}$ ed $\mathcal{S}$, la loro differenza, denotata con

$$\mathcal{R} \setminus \mathcal{S}$$

è costituita dalle tuple che appartengono ad $\mathcal{R}$ e che non appartengono ad $\mathcal{S}$.

Esempio 2.3.3 - Differenza fra due relazioni:

A	B	C
a	3	x
b	1	y
a	2	y
a	1	x

$\setminus$

A	B	C
b	1	y
a	1	y
a	3	x

=

A	B	C
a	2	y
a	1	x

- prodotto cartesiano : Date due relazioni $\mathcal{R}(A_1, \ldots, A_n)$ ed $\mathcal{S}(B_1, \ldots, B_m)$, il loro prodotto cartesiano, denotato con

$$\mathcal{R} \times \mathcal{S}$$

è una relazione avente schema $(A_1, \ldots, A_n, B_1, \ldots, B_m)$ e costituita dalle tuple ottenute concatenando, in tutte le combinazioni possibili, una tupla di $\mathcal{R}$ con una tupla di $\mathcal{S}$. Si deve supporre che gli identificatori A_i e B_i degli attributi di $\mathcal{R}$ e di $\mathcal{S}$ siano fra loro diversi, per evitare ambiguità nei nomi degli attributi del prodotto; in caso contrario di deve ricorrere ad una ridenominazione degli attributi.

Esempio 2.3.4 - Prodotto cartesiano fra due relazioni:

A	B
b	3
a	1
b	2

$\times$

C	D	E
x	1	r
y	3	p

=

A	B	C	D	E
b	3	x	1	r
b	3	y	3	p
a	1	x	1	r
a	1	y	3	p
b	2	x	1	r
b	2	y	3	p

Osservazione. Le operazioni di unione ed intersezione godono della proprietà commutativa ed associativa, ossia, indicando con $\circ$ una di queste due operazioni, valgono le seguenti identità:

$$\mathcal{R} \circ \mathcal{S} = \mathcal{S} \circ \mathcal{R}$$

$$(\mathcal{R} \circ \mathcal{S}) \circ \mathcal{T} = \mathcal{R} \circ (\mathcal{S} \circ \mathcal{T})$$

2.4 Operatori relazionali

Gli *operatori relazionali* sono degli specifici operatori dell'algebra relazionale che permettono di estrarre delle colonne o delle righe di una relazione.

- operatore di proiezione : Data una relazione $\mathcal{R}$ ed un sottoinsieme $\mathcal{A} = \{A_1, \ldots, A_k\}$ degli attributi di $\mathcal{R}$, si definisce proiezione di $\mathcal{R}$ su $\mathcal{A}$ e si scrive

$$\pi_{A_1,\ldots,A_k}(\mathcal{R})$$

la relazione di grado k che si ottiene eliminando gli attributi (colonne) non presenti in $\mathcal{A}$ ed eliminando eventuali tuple duplicate che si vengono a determinare.

Esempio 2.4.1 - Proiezione di una relazione su una lista di attributi:

$\pi_{A,C}$

A	B	C	D
a	1	x	2
a	1	y	3
a	2	x	1
b	1	x	4

=

A	C
a	x
a	y
b	x

- operatore di selezione : Data una relazione $\mathcal{R}$ ed una condizione C definita sugli attributi di $\mathcal{R}$ si definisce selezione di $\mathcal{R}$ sulla condizione C e si scrive

$$\sigma_C(\mathcal{R})$$

la relazione avente schema uguale a quello di $\mathcal{R}$ e composta dalle tuple di $\mathcal{R}$ che soddisfano alla condizione C.

Esempio 2.4.2 - Selezione di una relazione su una condizione:

$\sigma_{B<D}$

A	B	C	D
a	1	x	2
a	1	y	3
a	2	x	1
b	1	x	4

=

A	B	C	D
a	1	x	2
a	1	y	3
b	1	x	4

- operatore di ridenominazione : Data una relazione $\mathcal{R}(A_1, \ldots, A_n)$ ed una lista $(B_1, \ldots, B_n)$ di attributi, con

$$\rho_{B_1,\ldots,B_n}(\mathcal{R})$$

si denota la relazione ottenuta da $\mathcal{R}$ ridenominando gli attributi di $\mathcal{R}$ con $B_1, \ldots, B_n$; nel caso gli attributi che si vogliono ridenominare siano pochi, viene utilizzata la notazione alternativa equivalente

$$\rho_{A'_1 \to B_1,\ldots,A'_k \to B_k}(\mathcal{R})$$

nella quale vengono indicati solo gli attributi che devono essere ridenominati (lasciando inalterati gli altri).

Esempio 2.4.3 - Ridenominazione di una relazione:

$$\rho_{C\to P}\left(\begin{array}{|c|c|c|c|}\hline A & B & C & D\\\hline \texttt{a} & 1 & \texttt{x} & 2\\ \texttt{a} & 1 & \texttt{y} & 3\\ \texttt{a} & 2 & \texttt{x} & 1\\ \texttt{b} & 1 & \texttt{x} & 4\\\hline\end{array}\right) = \begin{array}{|c|c|c|c|}\hline A & B & P & D\\\hline \texttt{a} & 1 & \texttt{x} & 2\\ \texttt{a} & 1 & \texttt{y} & 3\\ \texttt{a} & 2 & \texttt{x} & 1\\ \texttt{b} & 1 & \texttt{x} & 4\\\hline\end{array}$$

- divisione : Date due relazioni $\mathcal{R}(X)$ ed $\mathcal{S}(Y)$ in cui $Y \subseteq X$, la loro divisione, denotata con

$$\mathcal{R} \div \mathcal{S}$$

è una relazione avente schema $Z = X \setminus Y$ e costituita dalle tuple t se in $\mathcal{R}$ sono presenti tuple con $t_{\mathcal{R}}[Z] = t$ e con $t_{R}[Y] = t_{\mathcal{S}}$ per ogni tupla $t_{\mathcal{S}}$ di $\mathcal{S}$. Ciò significa che una tupla t si trova in $\mathcal{R} \div \mathcal{S}$ se in $\mathcal{R}$ compaiono i valori di t in combinazione con ogni tupla di $\mathcal{S}$. L'operazione di divisione può essere espressa come combinazione delle operazioni $\pi, \times$ e $\setminus$, come segue:

$$\begin{aligned} T_1 &\leftarrow \pi_{X\setminus Y}(\mathcal{R})\\ T_2 &\leftarrow \pi_{X\setminus Y}((\mathcal{S}\times T_1)\setminus \mathcal{R})\\ T &\leftarrow T_1 \setminus T_2 \end{aligned}$$

L'operatore di divisione risulta utile per interrogazioni della forma: *Determinare le regioni che confinano con tutte le regioni con le quali confina il Trentino.*

Esempio 2.4.4 - Divisione fra due relazioni:

$$\begin{array}{|c|c|c|}\hline A & B & C\\\hline \texttt{a} & 1 & \texttt{x}\\ \texttt{a} & 1 & \texttt{y}\\ \texttt{b} & 1 & \texttt{y}\\ \texttt{a} & 2 & \texttt{x}\\ \texttt{a} & 2 & \texttt{y}\\ \texttt{b} & 2 & \texttt{z}\\ \texttt{a} & 3 & \texttt{x}\\ \texttt{a} & 3 & \texttt{y}\\ \texttt{b} & 3 & \texttt{y}\\ \texttt{b} & 3 & \texttt{z}\\\hline\end{array} \div \begin{array}{|c|c|}\hline A & C\\\hline \texttt{a} & \texttt{x}\\ \texttt{b} & \texttt{y}\\\hline\end{array} = \begin{array}{|c|}\hline B\\\hline 1\\ 3\\\hline\end{array}$$

2.5 Operatori di giunzione interna

Il prodotto fra due relazioni risulta particolarmente oneroso dal punto di vista computazionale. Inoltre, la relazione ottenuta dal prodotto cartesiano risulta composta da ennuple non tutte aventi significato; tale relazione viene successivamente sottoposta ad altre operazioni che ne riducono la cardinalità (in genere mediante una operazione di selezione) ed aumentano il significato dei dati. In base a tutte queste considerazioni, al posto dell'operatore di prodotto cartesiano viene spesso usato l'operatore *join* che viene presentato a seguire.

L'operatore di giunzione permette di unire i dati delle tuple di relazioni distinte. Nella sua forma più generale l'operatore di giunzione, detto *giunzione interna* (*inner join*), viene definito come segue: Date due relazioni $\mathcal{R}(A_1, \ldots, A_n), \mathcal{S}(B_1, \ldots, B_m)$ ed una condizione C definita sugli attributi di $\mathcal{R}$ ed $\mathcal{S}$ si definisce giunzione delle due relazioni $\mathcal{R}$ ed $\mathcal{S}$ sulla condizione C e si scrive

$$\mathcal{R} \bowtie_C \mathcal{S}$$

la relazione, avente schema $(A_1, \ldots, A_n, B_1, \ldots, B_m)$, formata dalle tuple di $\mathcal{R}$ concatenate con le tuple di $\mathcal{S}$ (prodotto cartesiano) soddisfacenti alla condizione C. Questa definizione evidenzia che l'operatore di giunzione è derivabile dalle operazioni primitive in base alla seguente identità:

$$\mathcal{R} \bowtie_C \mathcal{S} = \sigma_C(\mathcal{R} \times \mathcal{S})$$

Nonostante questa equivalenza, risulta nettamente più efficiente l'espressione coinvolgente l'operatore $\bowtie$ di giunzione, al posto della seconda che coinvolge l'operatore $\times$ di prodotto cartesiano.

Per motivi di efficienza la condizione C delle giunzione interna viene solitamente limitata a delle forme ridotte particolari; a seconda della forma della condizione C l'operatore di giunzione assume diverse denominazioni, come indicato nel seguito.

- theta-giunzione (θ-*join*) : la condizione C è una generica espressione logica coinvolgente sottoespressioni della forma $A_i \circ B_j$, essendo A_i e B_j due generici attributi, fra loro compatibili al confronto, rispettivamente delle due relazioni $\mathcal{R}$ ed $\mathcal{S}$ e $\circ$ un operatore di confronto $=, \neq, <, >, \leq, \geq$ composte fra loro con gli operatori logici $\vee, \wedge, \neg$.

  Esempio 2.5.1 - θ-giunzione fra due relazioni:

A	B	C
a	7	x
b	3	y
a	8	y

$\bowtie_{B<E}$

D	E
p	5
q	8

$=$

A	B	C	D	E
a	7	x	q	8
b	3	y	p	5
b	3	y	q	8

- equigiunzione (*equijoin*) : la condizione C è della forma $A_i = B_j$, essendo A_i e B_j due generici attributi rispettivamente delle due relazioni $\mathcal{R}$ ed $\mathcal{S}$; l'equigiunzione costituisce un caso particolare della θ-giunzione

Esempio 2.5.2 - Equigiunzione fra due relazioni:

A	B	C
a	1	p
b	4	q
a	3	p

$\bowtie_{B=D}$

D	E
1	x
1	y
2	y
3	x

=

A	B	C	D	E
a	1	p	1	x
a	1	p	1	y
a	3	p	3	x

- giunzione naturale (*natural join*) : la condizione C è costituita dall'uguaglianza fra tutti gli attributi omonimi delle due relazioni operande; in questo caso viene automaticamente eliminata una delle due colonne uguali aventi lo stesso attributo.

Esempio 2.5.3 - Giunzione naturale fra due relazioni:

A	B	C	D
a	4	x	p
b	4	y	q
a	3	x	p
b	3	z	p

$\bowtie$

B	D	E
4	p	t
5	q	t
3	p	w

=

A	B	C	D	E
a	4	x	p	t
a	3	x	p	w
b	3	z	p	w

In un'operazione di giunzione, se ciascuna tupla delle due relazioni coinvolte contribuisce ad almeno una tupla del risultato, si parla di *join completo*.

2.6 Operatori di giunzione esterna

Nelle operazioni di join precedentemente esaminate (θ-join, equi-join, join naturale) nella relazione risultante sono presenti solo le tuple che hanno valori correlati nelle due relazioni operande. Tutte le altre tuple vengono eliminate. Quando si desidera mantenere nel risultato anche le tuple non correlate si ricorre all'operatore di join esterno o outer join. Si hanno i seguenti casi di join esterno:

- join esterno sinistro (*left outer join*) : nel risultato vengono mantenute tutte le tuple della relazione $\mathcal{R}$ di sinistra, combinate con le tuple di $\mathcal{S}$ che hanno corrispondenza sugli attributi omonimi, completando con valori nulli le tuple di $\mathcal{R}$ che non hanno corrispondenza in $\mathcal{S}$. L'operatore di join esterno sinistro fra le due relazioni $\mathcal{R}$ ed $\mathcal{S}$ viene indicato con la notazione

$$\mathcal{R} \mathbin{⟕}_C \mathcal{S}$$

Esempio 2.6.1 - Join esterno sinistro:

A	B	C
a	4	x
b	8	y
c	7	x

$⟕_{B \leq E}$

D	E
p	2
p	5
q	1
q	4

=

A	B	C	D	E
a	4	x	p	5
a	4	x	q	4
b	8	y	ω	ω
c	7	x	ω	ω

- join esterno destro (*right outer join*) : nel risultato vengono mantenute tutte le tuple della relazione $\mathcal{S}$ di destra, combinate con le tuple di $\mathcal{S}$ che hanno corrispondenza sugli attributi omonimi, completando con valori nulli le tuple di $\mathcal{S}$ che non hanno corrispondenza in $\mathcal{R}$. L'operatore di join esterno sinistro fra le due relazioni $\mathcal{R}$ ed $\mathcal{S}$ viene indicato con la notazione

$$\mathcal{R} ⟖_{C} \mathcal{S}$$

Esempio 2.6.2 - Join esterno destro:

A	B	C
a	4	x
b	8	y
c	7	x

$⟖_{B \leq E}$

D	E
p	2
p	5
q	1
q	4

=

A	B	C	D	E
a	4	x	p	5
a	4	x	q	4
ω	ω	ω	p	2
ω	ω	ω	q	1

- join esterno completo (*full outer join*) : nel risultato vengono mantenute tutte le tuple delle relazioni $\mathcal{R}$ ed $\mathcal{S}$, completando con valori nulli le tuple che non trovano corrispondenza sugli attributi omonimi con tuple dell'altra relazione. L'operatore di join esterno completo fra le due relazioni $\mathcal{R}$ ed $\mathcal{S}$ viene indicato con la notazione

$$\mathcal{R} ⟗_{C} \mathcal{S}$$

Esempio 2.6.3 - Join esterno completo:

A	B	C
b	8	y
c	7	x

$⟗_{B \leq E}$

D	E
p	2
p	5
q	1
q	4

=

A	B	C	D	E
a	4	x	p	5
a	4	x	q	4
b	8	y	ω	ω
c	7	x	ω	ω
ω	ω	ω	p	2
ω	ω	ω	q	1

2.7 Espressioni relazionali

Gli operatori dell'algebra relazionale possono essere combinati per costruire complesse espressioni che rispondono a delle interrogazioni molto specifiche ed articolate. Gli esempi che seguono dimostrano la potenza dell'algebra relazionale in queste situazioni.

Esempio 2.7.1 - La combinazione degli operatori di proiezione e selezione permette di definire interessanti interrogazioni; ad esempio, con riferimento alla relazione STUDENTE riportata nell'esempio 1.4.2, la valutazione dell'espressione

$$\pi_{Nome,Piano}(\sigma_{Posti>24}(\text{AULA}))$$

produce come risultato la seguente relazione che rappresenta il nome ed il piano delle aule con più di 24 posti.

Nome	*Piano*
1	1
3	1
LabInf1	0

Esempio 2.7.2 - L'operatore di giunzione costituisce uno degli operatori più potenti dell'algebra relazionale e consente di esprimere in modo molto elegante e conciso delle interrogazioni complesse che coinvolgono più relazioni. Ad esempio, con riferimento alla base di dati descritta nell'esempio 1.7.1, l'interrogazione

$$\pi_{Cognome,Nome,Descrizione}(\text{DIPENDENTE} \bowtie_{Reparto=Sigla} \text{REPARTO})$$

fornisce come risultato la relazione

Cognome	*Nome*	*Reparto*
Antico	Nicola	produzione
Barbieri	Amedeo	produzione
Gaio	Felice	produzione
Armellini	Francesca	amministrazione
Donati	Claudia	amministrazione
Rovetta	Giorgio	magazzino
Bonato	Diego	spedizione

che esprime il cognome, il nome ed il reparto in cui lavora ciascun dipendente.

Esempio 2.7.3 - Con riferimento allo schema relazionale riportato nell'esempio 1.11.2, si può esprimere l'interrogazione *Nomi delle province del Veneto.* mediante la seguente espressione:

$$\pi_{NomePro}(\sigma_{NomeReg='\texttt{Veneto}'}(\textsc{Provincia} \bowtie_{Regione=Capoluogo} \textsc{Regione}))$$

Come si vede in questo caso, qualora l'espressione risulti complessa e di difficile decifrazione, si spezza l'espressione in più sottoespressioni, collegate mediante l'operatore $\leftarrow$ di *assegnazione*; l'espressione sopra riportata si esprime in modo equivalente come segue:

$$\begin{aligned} T &\leftarrow \textsc{Provincia} \bowtie_{Regione=Capoluogo} \textsc{Regione} \\ T &\leftarrow \sigma_{NomeReg='\texttt{Veneto}'}(T) \\ Q &\leftarrow \pi_{NomePro}(T) \end{aligned}$$

In questa sequenza di assegnazioni è stata indicata con Q la relazione che costituisce il risultato finale dell'interrogazione, mentre con T sono state indicate delle relazioni temporanee di supporto al processo di calcolo.

Sempre con riferimento allo schema relazionale riportato nell'esempio 1.11.2, si possono risolvere le seguenti interrogazioni come riportato in corrispondenza di ciascuna:

1. Nomi delle province con più di un milione di abitanti.

$$\begin{aligned} T &\leftarrow \sigma_{Abitanti>1000000}(\textsc{Provincia}) \\ Q &\leftarrow \pi_{NomePro,Abitanti}(T) \end{aligned}$$

2. Nomi delle regioni a statuto speciale.

$$\begin{aligned} T &\leftarrow \sigma_{Statuto='\texttt{speciale}'}(\textsc{Regione}) \\ Q &\leftarrow \pi_{NomeReg}(T) \end{aligned}$$

2.8 Proceduralità dell'AR

L'algebra relazionale costituisce un linguaggio di tipo procedurale in quanto descrive *come* l'esecutore deve agire per arrivare al risultato finale, specificando la sequenza delle operazioni che devono essere svolte. Questa caratteristica di proceduralità è alla base del fatto che, in generale, ci siano delle diverse sequenze equivalenti di operazioni dell'algebra relazionale che conducono allo stesso risultato, come descritto nell'esempio che segue.

Esempio 2.8.1 - Con riferimento allo schema relazionale riportato nell'esempio **??**, per determinare i nomi delle province capoluogo di regione si può usare la semplice espressione sulla sola relazione PROVINCIA:

$$\pi_{NomePro}(\sigma_{Sigla=Regione}(\textsc{Provincia}))$$

oppure

$$\begin{aligned} T &\leftarrow \textsc{Regione} \bowtie_{Capoluogo=Sigla} \textsc{Provincia} \\ Q &\leftarrow \pi_{NomePro}(T) \end{aligned}$$

oppure ancora

$$\begin{aligned} T &\leftarrow \textsc{Provincia} \bowtie_{Regione=Capoluogo} \textsc{Regione} \\ T &\leftarrow \sigma_{Sigla=Regione}(\textsc{Provincia}) \\ Q &\leftarrow \pi_{NomePro}(T) \end{aligned}$$

2.9 Funzioni di raggruppamento

Le operazioni di base dell'algebra relazionale non consentono di esprimere molte interessanti interrogazioni che coinvolgono funzioni su insiemi di valori della base di dati (conteggio, somma, media, minimo, massimo). Spesso queste funzioni aggregate vengono riferite a dei raggruppamenti di tuple stabiliti in base a dei valori di alcuni loro attributi. L'operazione *funzione aggregata* dell'algebra relazionale, nella forma più generale si denota come segue ($\mathcal{F}$ si legge *f corsivo*)

$$_X\mathcal{F}_F(\mathcal{R})$$

dove X è una lista di attributi della relazione $\mathcal{R}$, detti *attributi di raggruppamento* ed F è una lista di coppie della forma

$$fA$$

dove f è una funzione di raggruppamento (*COUNT*, *SUM*, *AVERAGE*, *MINIMUM*, *MAXIMUM*)) ed A è un attributo di $\mathcal{R}$. La funzione $\mathcal{F}$ opera come segue:

1. le tuple della relazione $\mathcal{R}$ vengono ripartite in gruppi omogenei aventi uguali valori sugli attributi X
2. ciascuna funzione della lista F viene applicata a ciascuno dei gruppi individuati al punto precedente

La relazione risultante ha gli attributi di raggruppamento più un attributo per ogni elemento della lista di coppie F. L'identificatore di ciascuno di questi attributi aggiuntivi ha come identificatore `f_A`. Se non vengono specificati attributi di raggruppamento viene creato un unico gruppo e le funzioni della lista F vengono applicate ai valori degli attributi di ciascuna tupla della relazione, cosicché la relazione risultante sarà composta da una sola tupla.

Gli esempi che seguono fanno riferimento allo schema relazionale riportato nell'esempio 1.11.2.

Esempio 2.9.1 - Numero di province:

$$\mathcal{F}_{COUNT\ NomePro}(\text{PROVINCIA})$$

$COUNT\_NomePro$
107

C'è da osservare che, benché il risultato sia costituito da un solo numero, si tratta comunque di una relazione composta da una sola riga e da una sola colonna.

Esempio 2.9.2 - Nome e numero di abitanti della provincia meno popolata:

$$_{NomePro}\mathcal{F}_{MIN\ Abitanti}(\text{PROVINCIA})$$

$NomePro$	$MIN\_Abitanti$
Aosta	114279

In questo caso l'attributo di raggruppamento $NomePro$, benché ininfluente ai fini del raggruppamento (in quanto i nomi delle province sono tutti distinti), risulta necessario al fine di ottenere, nel risultato finale, la colonna corrispondente al nome della provincia.

Esempio 2.9.3 - Numero di province di ciascuna regione:

$$\begin{aligned} T &\leftarrow \text{PROVINCIA} \bowtie_{Regione=Capoluogo} \text{REGIONE} \\ T &\leftarrow {}_{NomeReg}\mathcal{F}_{COUNT\ NomePro}(T) \\ Q &\leftarrow \rho_{Regione,NumeroProvince}(T) \end{aligned}$$

Esempio 2.9.4 - Popolazione di ciascuna regione:

$$\begin{aligned} T &\leftarrow \text{PROVINCIA} \bowtie_{Regione=Capoluogo} \text{REGIONE} \\ T &\leftarrow {}_{NomeReg}\mathcal{F}_{SUM\ Abitanti}(T) \\ Q &\leftarrow \rho_{Regione,Popolazione}(T) \end{aligned}$$

2.10 Interrogazioni mediante il linguaggio SQL

SQL è un linguaggio dichiarativo, definito e sviluppato a partire dal 1987, che è diventato uno standard per la gestione delle basi di dati relazionali. SQL è un linguaggio completo in quanto offre tutte le componenti DDL, DML e QL. Comprende inoltre molte altre funzionalità tipiche di un linguaggio di programmazione tradizionale, fra le quali: definizione di nuovi tipi di dati (oltre a quelli nativi predisposti dal linguaggio), possibilità d'uso di variabili ed assegnazioni, possibilità di definire funzioni e procedure (con argomenti), e molte altre.

La parte più caratteristica di SQL è costituita dalla sua componente QL che, in pratica, si riduce ad una singola istruzione avente, nella sua forma basica, la struttura descritta nel listato 2.1.

```
SELECT A1, A1, ..., An
FROM   TABELLA
WHERE  Condizione
```

Listato 2.1: Esempio di query

Questa interrogazione corrisponde ad una combinazione delle due istruzioni π e σ dell'algebra relazionale:

$$\pi_{A_1,A_2,\ldots,A_n}(\sigma_{Condizione}(\text{TABELLA}))$$

Nell'espressione del listato 2.1, la clausola `WHERE` può essere omessa e l'interrogazione assume la seguente forma:

```
SELECT Ah, ..., Ak
FROM   TABELLA;
```

Questa scrittura è equivalente a mettere la condizione `True` e corrisponde all'operazione di proiezione dell'algebra relazionale [1] :

$$\pi_{A_h,\ldots,A_k}(\text{TABELLA})$$

Per specificare la lista di tutti gli attributi si può scrivere semplicemente `*`; in questo caso l'interrogazione si scrive nella forma

```
SELECT *
FROM   TABELLA
WHERE  Condizione
```

e corrisponde all'espressione

$$\sigma_{Condizione}(\text{TABELLA})$$

[1] Si noti la terminologia poco felice: la clausola *SELECT* specifica la lista degli attributi che compaiono nell'operazione π dell'algebra relazionale; per questo motivo l'operazione σ viene spesso denotata con il termine *restrizione* anziché *selezione*.

La condizione che viene specificata nella clausola *WHERE* è spesso costituita da un confronto ($=, \neq, <, \leq, >, \geq$) fra un attributo ed un valore oppure un confronto fra due attributi; per controllare l'uguaglianza o la diversità fra un attributo *A* ed il valore nullo si scrive *A IS NULL* oppure *A IS NOT NULL*. Per costruire una condizione per controllare se il valore corrispondente ad un dato attributo *A* appartiene ad un dato insiemi di valori si scrive *A IN (V1, V2, ..., Vn)*; per controllare se un valore non appartiene ad un insieme i valori si usa il predicato *NOT IN*. Una condizione può essere costruita anche con i tradizionali operatori *AND*, *OR* e *NOT*.

A differenza di quanto avviene nell'algebra relazionale, in SQL la proiezione (per questioni di efficienza computazionale) non elimina eventuali righe duplicate; nel caso si desideri che le righe duplicate vengano eliminate bisogna aggiungere alla clausola *SELECT* l'attributo *DISTINCT* e l'interrogazione assume la seguente forma:

```
SELECT DISTINCT Ah, ..., Ak
FROM   TABELLA
WHERE  Condizione
```

Interrogazioni elementari

L'istruzione *SELECT*, nella sua forma basica descritta nel nel listato 2.1 permette di esprimere tutte le operazioni dell'algebra relazionale. Negli esempi che seguono si fa riferimento allo schema relazionale descritto nell'esempio **??**.

Esempio 2.10.1 - Nomi delle regioni a statuto speciale:

```
SELECT Nomereg
FROM REGIONE
WHERE Statuto = 'speciale';
```

Esempio 2.10.2 - Nomi ed abitanti delle province con più di un milione di abitanti, ordinate decrescentemente in base agli abitanti:

```
SELECT NomePro AS Provincia, Abitanti
FROM PROVINCIA
WHERE Abitanti>1000000
ORDER BY Abitanti DESC
```

La particella *AS* serve per ridenominare uno o più attributi e corrisponde all'operatore ρ di ridenominazione dell'algebra relazionale; la particella *DESC* indica che le righe del risultato verranno fornite in ordine decrescente in base all'attributo indicato; la particella *ASC* indicherebbe l'ordinamento crescente (default).

Interrogazioni con JOIN fra tabelle

Il linguaggio SQL predispone l'operazione di join in tutte le forme previste dall'algebra relazionale (completo, sinistro e destro). L'operazione di join

$$\mathcal{R} \bowtie_C \mathcal{S}$$

in SQL si scrive come segue:

```
SELECT *
FROM R JOIN S ON C
```

La sintassi d'uso è deducibile dai seguenti esempi.

Esempio 2.10.3 - Nomi delle province di regioni a statuto speciale e corrispondente nome della regione.

```
SELECT NomePro, NomeReg
FROM PROVINCIA JOIN REGIONE
                ON Regione=Capoluogo
WHERE Statuto='speciale'
ORDER BY NomeReg, NomePro;
```

Esempio 2.10.4 - Nomi delle provincie del *Veneto* e della *Lombardia*, con corrispondente regione di appartenenza, ordinate decrescentemente per nome della regione e crescentemente per nome della provincia.

```
SELECT NomePro, NomeReg
FROM PROVINCIA JOIN REGIONE
                ON Regione=Capoluogo
WHERE NomeReg IN ('Lombardia','Veneto')
ORDER BY Regione DESC, NomePro ASC;
```

Esempio 2.10.5 - Nomi delle provincie che si trovano nella stessa regione nella quale si trova *Belluno*.

```
SELECT P2.NomePro
FROM PROVINCIA AS P1 JOIN PROVINCIA AS P2
                      ON P1.Regione=P2.Regione
WHERE (P1.NomePro='Belluno')
              AND
      (P2.NomePro<>'Belluno');
```

Interrogazioni annidate

Le interrogazioni danno come risultato delle tabelle formate, in generale, da più righe e più colonne. Si possono avere i seguenti casi particolari:

- tabella di una sola colonna: i valori della colonna possono essere interpretati come un *insieme di valori*
- tabella di una sola riga ed una sola colonna: la tabella può essere interpretata come un *singolo valore*

Esempio 2.10.6 - Nomi delle provincie che si trovano nella stessa regione nella quale si trova *Belluno*.

```
SELECT NomePro
FROM PROVINCIA
WHERE Regione =
    (
    SELECT Regione
    FROM PROVINCIA
    WHERE NomePro='Belluno'
    )
 AND (NomePro<>'Belluno');
```

Esempio 2.10.7 - Nomi delle provincie capoluogo delle regioni a statuto *speciale*.

```
SELECT NomePro
FROM PROVINCIA
WHERE Sigla IN
    (
    SELECT Capoluogo
    FROM REGIONE
    WHERE Statuto='speciale'
    );
```

In molti casi una interrogazione annidata può esser espressa in modo alternativo mediante una interrogazione con `JOIN`; ad esempio l'interrogazione sopra può essere espressa come segue:

```
SELECT Nomepro
FROM PROVINCIA JOIN REGIONE
                ON Regione=Capoluogo
WHERE (Statuto='speciale') AND (Sigla=Regione);
```

Questa seconda interrogazione, a differenza della precedente, permette di accedere anche all'attributo `NomeReg`.

Utilizzando il predicato *EXISTS* è possibile esprimere delle condizioni della forma

EXISTS (*SELECT* ...)

che risultano vere se la tabella prodotta come risultato della query è composta da almeno una riga.

Nel caso in cui una interrogazione produca come risultato una singola colonna, i valori di questa colonna possono essere interpretati come un insieme di valori; su un insieme di valori si possono applicare i seguenti predicati formati da un confronto fra un singolo valore x, un generico predicato di confronto $=, \neq, <, \leq, >, \geq$ ed un *InsiemeDiValori*:

- *ANY*: la condizione $x \lessgtr$ *ANY* $InsiemeDiValori$ è vera se il confronto è vero per almeno un valore dell'insieme; risulta falsa anche se l'insieme di valori è vuoto
- *ALL*: la condizione $x \lessgtr$ *ALL* $InsiemeDiValori$ è vera se il confronto è vero per tutti i valori dell'insieme

Indicando con A un generico attributo, valgono le seguenti equivalenze:

A *IN* (*SELECT* ...) equivale a A = *ANY* (*SELECT* ...)

A *NOT IN* (*SELECT* ...) equivale a A <> *ALL* (*SELECT* ...)

Interrogazioni con funzioni di raggruppamento

In SQL si possono eseguire dei raggruppamenti di righe ed utilizzare le funzioni di raggruppamento previste nell'algebra relazionale: *COUNT*, *SUM*, *MIN*, *MAX*, *AVG*. Il raggruppamento viene specificato mediante la clausola *GROUP BY* che ha l'effetto di ragruppare in gruppi omogenei le tuple aventi gli stessi valori in corrispondenza degli attributi specificati nella clusola e le funzioni di raggruppamento vengono applicate a ciascun gruppo. Nel caso in cui non venga specificata la clausola *GROUP BY* viene creato un unico gruppo.

Esempio 2.10.8 - Numero delle regioni:

```
SELECT COUNT(*) AS NumeroRegioni
FROM REGIONE;
```

Se come argomento nella funzione *COUNT* venisse indicato, al posto di *, un attributo verrebero contate solo le righe non aventi valore nullo in corrispondenza dell'attributo indicato. Il numero delle regioni può essere calcolato anche mediante la tabella `PROVINCIA`, contando una sola volta le righe avente lo stesso valore in corrispondenza dell'attributo `Regione`, come segue:

```
SELECT COUNT(DISTINCT Regione) AS NumeroRegioni
FROM PROVINCIA;
```

Esempio 2.10.9 - Popolazione italiana:

```
SELECT SUM(Abitanti) AS 'Popolazione italiana'
FROM PROVINCIA;
```

Esempio 2.10.10 - Provincia più popolata:

```
SELECT NomePro, Abitanti
FROM PROVINCIA
WHERE Abitanti =
    (
    SELECT MAX(Abitanti)
    FROM PROVINCIA
    );
```

Esempio 2.10.11 - Numero di province di ciascuna regione:

```
SELECT NomeReg, COUNT(*) AS NumProvince
FROM PROVINCIA JOIN REGIONE ON Regione=Capoluogo
GROUP BY Regione
ORDER BY NumProvince DESC, NomeReg;
```

Esempio 2.10.12 - Regioni a statuto *ordinario* aventi più di 5 province:

```
SELECT NomeReg, COUNT(*) AS NumProvince
FROM PROVINCIA JOIN REGIONE ON Capoluogo=Regione
WHERE Statuto='ordinario'
GROUP BY Regione
HAVING COUNT(*) > 5
ORDER BY NumProvince DESC;
```

La condizione indicata nella clausola ***WHERE*** esegue un filtro sulle righe mentre la condizione indicata nella clausola ***HAVING*** esegue un filtro sui raggruppamenti.

Esempio 2.10.13 - Regioni aventi tutte le province con meno di 500000 abitanti:

```
SELECT NomeReg
FROM Provincia JOIN Regione ON Capoluogo=Sigla
GROUP BY Regione
HAVING MAX(Abitanti)<500000;
```

Viste logiche

Mediante l'istruzione ***CREATE VIEW*** viene definita una *vista* ossia una tabella virtuale generata mediante un'istruzione ***SELECT***; le tuple della vista vengono rigenerate dinamicamente ad ogni uso della vista, ricalcolando l'interrogazione sulla corrente istanza della base di dati.

Esempio 2.10.14 - La seguente istruzione crea una vista corrispondente alle province capoluogo di regione.

```
CREATE VIEW CAPOLUOGO
            (Provincia,Abitanti,Regione) AS
SELECT NomePro, Abitanti, NomeReg
FROM PROVINCIA JOIN REGIONE ON Capoluogo=Sigla;
```

Usando la vista `CAPOLUOGO` sopra definita si possono determinare le province capoluogo di regione aventi più di 1 milione di abitanti mediante la seguente interrogazione:

```
SELECT Provincia, Abitanti
FROM CAPOLUOGO
WHERE Abitanti>1000000;
```

Funzioni e procedure

…

I trigger

…

I diversi utenti ed i loro permessi

…

Il dizionario della base di dati

…

ESERCIZI

2.1 Indicando con $|\mathcal{R}|$ la cardinalità una relazione $\mathcal{R}$, dimostrare che valgono le seguenti relazioni, precisando in quali casi vale il segno di uguaglianza:

1. $|\sigma_C(\mathcal{R})| \leq |\mathcal{R}|$
2. $|\pi_L(\mathcal{R})| \leq |\mathcal{R}|$

2.2 Discutere in quali situazioni sono verificate le seguenti uguaglianze:

1. $|\mathcal{R} \cup \mathcal{S}| = |\mathcal{R}| + |\mathcal{S}|$
2. $|\mathcal{R} \cap \mathcal{S}| = |\mathcal{R}| + |\mathcal{S}|$
3. $|\mathcal{R} \bowtie \mathcal{S}| = |\mathcal{R}| + |\mathcal{S}|$

2.3 Siano $\mathcal{R}$, $\mathcal{S}$ due relazioni. Dimostrare che

$$\mathcal{R} \cap \mathcal{S} = \mathcal{R} \setminus (\mathcal{R} \setminus \mathcal{S})$$

Suggerimento: rappresentare le due relazioni $\mathcal{R}$, $\mathcal{S}$ mediante dei diagrammi di Venn.

2.4 Siano $\mathcal{R}$, $\mathcal{S}$ due relazioni. Dimostrare che

$$|\mathcal{R} \times \mathcal{S}| = |\mathcal{R}| \cdot |\mathcal{S}|$$

2.5 Dimostrare che l'insieme $\{\cup, \setminus, \times, \pi, \sigma\}$ di operatori dell'algebra relazionale costituisce un *insieme completo*, cioè ogni altro operatore dell'algebra relazionale può essere espresso come combinazione di operatori di questo insieme.

2.6 Sia $\mathcal{R}$ una relazione e P, Q due predicati sugli attributi di $\mathcal{R}$. Stabilire se le seguenti due uguaglianze sono vere:

1. $\sigma_{P \wedge Q}(\mathcal{R}) = \sigma_P(\mathcal{R}) \cap \sigma_Q(\mathcal{R})$
2. $\sigma_{P \vee Q}(\mathcal{R}) = \sigma_P(\mathcal{R}) \cup \sigma_Q(\mathcal{R})$

2.7 Analizzare e discutere se le due operazioni di proiezione e restrizione sono commutative, ossia se in generale

$$\pi_L(\sigma_C(\mathcal{R})) = \sigma_C(\pi_L(\mathcal{R}))$$

Si dimostri la relazione se vera, altrimenti si proponga un controesempio.

2.8 Date le seguenti due relazioni

$\mathcal{R}$

A	B	C
6	x	h
2	x	k
5	y	k
1	z	k
8	z	h

$\mathcal{S}$

D	E
p	1
x	5
x	3
t	4
z	1

calcolare il valore delle seguenti espressioni, evidenziando i passaggi ed i risultati intermedi che portano al risultato finale.

1. $\pi_{B,C,D}(\mathcal{R} \bowtie_{A<E} \mathcal{S})$
2. $\pi_{A,C}(\sigma_{A<E}(\mathcal{R} \bowtie_{B=D} \mathcal{S}))$
3. $\pi_{A,C}(\sigma_{A<E}(\mathcal{R} \bowtie (\rho_{B,D}(\mathcal{S}))))$
4. $\pi_{A,C,D}((\sigma_{A>3}(\mathcal{R})) \bowtie_{B=D} \mathcal{S})$

2.9 Eliminare da una relazione $\mathcal{R}$ le tuple che soddisfano ad una data condizione C.

2.10 Dimostrare che l'operazione di *join naturale* è commutativa ed associativa, ossia che, date tre generiche relazioni $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$ si ha

1. $\mathcal{R} \bowtie \mathcal{S} = \mathcal{S} \bowtie \mathcal{R}$
2. $(\mathcal{R} \bowtie \mathcal{S}) \bowtie \mathcal{T} = \mathcal{R} \bowtie (\mathcal{S} \bowtie \mathcal{T})$

2.11 Dimostrare che l'operazione di *join naturale* soddisfa alla seguente proprietà:

$$\mathcal{R} \bowtie \mathcal{R} = \mathcal{R}$$

2.12 Definire, in modo informale o in modo formale mediante una grammatica, un linguaggio con istruzioni testuali, simile ad un usuale linguaggio di programmazione, corrispondente agli operatori dell'algebra relazionale.

2.13 Scrivere un'espressione E dell'algebra relazionale (la più semplice possibile) che, operando sulla relazione $\mathcal{R}(A, B, C, D)$ sottoriportata e coinvolgendo gli operatori π, σ, ρ, produca il seguente risultato:

E

A	B	C	D
4	2	2	1
1	3	4	2
3	3	1	3
4	4	2	4
1	2	3	4

=

M	N
4	2
3	1

2.14 Scrivere un'espressione E dell'algebra relazionale (la più semplice possibile) che, operando sulle relazioni $\mathcal{R}(A, B, C)$ ed $\mathcal{S}(D, E)$ sottoriportate e coinvolgendo gli operatori π, σ, ρ, produca il seguente risultato:

E

A	B	C
a	5	x
a	3	y
b	1	x
b	4	x

,

D	E
4	p
7	q
6	p
2	q

=

X	Y	Z
a	x	p
a	x	q
a	y	q
b	x	q

2.15 Dato lo schema relazionale costituito dalle relazioni $\mathcal{R}(\underline{A}, B, C)$, $\mathcal{S}(\underline{D}, A, E)$ di cardinalità m ed n, legate dal vincolo di integrità referenziale $\mathcal{R}.C \to \mathcal{S}.D$, deter-

minare la cardinalità delle relazioni che si ottengono come risultato della valutazione delle seguenti espressioni:

1. $\mathcal{R} \bowtie \mathcal{S}$
2. $\mathcal{R} \bowtie_{C=D} \mathcal{S}$
3. $\mathcal{R} \bowtie_{B=E} \mathcal{S}$
4. $\pi_{A,B}(\mathcal{R} \bowtie_{A=E} \mathcal{S})$

2.16 Rappresentare mediante il modello relazionale un *sacco*. Adottando la rappresentazione scelta, scrivere delle espressioni dell'algebra relazionale che forniscano le risposte alle seguenti interrogazioni:

1. insieme degli elementi presenti nel sacco; ad esempio, con riferimento al sacco $s = \{a, a, a, b, c, c\}$ l'espressione deve fornire come risultato la relazione di grado unitario avente estensione `a b c`
2. numero degli elementi presenti nel sacco
3. numero di elementi aventi un dato valore
4. unione di due sacchi

2.17 Rappresentare mediante il modello relazionale un sacco in modo tale che, indicando con s un sacco e con $\mathfrak{R}(s)$ la relazione che rappresenta il sacco secondo il modello relazionale adottato, valga l'isomorfismo operazionale

$$\mathfrak{R}(s_1 \cup s_2) = \mathfrak{R}(s_1) \cup \mathfrak{R}(s_e)$$

2.18 Rappresentare mediante il modello relazionale un *insieme di punti colorati*. Ogni punto è descritto dalle coordinate, dal nome (label) e dal colore. Sullo schema descritto, individuare delle chiavi candidate compatibili con le seguenti condizioni:

- ogni punto è individuato univocamente dalla propria label
- ci possono essere punti coincidenti (con stesse coordinate) purché si differenzino in base al colore

Con riferimento allo schema relazionale descritto, esprimere mediante l'algebra relazionale le risposte alle seguenti interrogazioni:

1. punti situati nel rettangolo $[0, 100] \times [0, 200]$
2. punti di colore *rosso* posti su un asse
3. coordinate dei punti non distinti (che hanno coordinate uguali a quelle di altri punti)
4. label dei punti distinti (che non hanno coordinate uguali a quelle di altri punti)

2.19 Rappresentare mediante il modello relazionale un disegno composto da spezzate dotate di nome, definite mediante dei punti colorati ed etichettati. Con riferimento allo schema relazionale descritto, scrivere delle espressioni dell'algebra relazionale per rispondere alle seguenti interrogazioni:

1. punti isolati

2. nomi delle spezzate interamente contenute nel primo quadrante
3. punti situati sugli assi
4. spezzate che hanno un punto sull'origine degli assi

2.20 Descrivere uno schema relazionale adatto a rappresentare un disegno nel piano cartesiano composto da dischi pieni di diverse dimensioni e colori. Scrivere un'espressione dell'algebra relazionale per determinare i dischi *rossi* completamente contenuti nel primo quadrante.

2.21 Con riferimento allo schema relazionale descritto nell'esempio 1.7.1, esprimere mediante l'algebra relazionale le seguenti interrogazioni:

1. cognome e nome dei dipendenti
2. cognome e nome dei dipendenti che si chiamano *Francesco* o *Francesca*
3. cognome e nome dei dipendenti non afferenti ad alcun reparto
4. cognome e nome dei dipendenti colleghi di reparto di *Armellini Francesca*
5. cognome e nome dei dipendenti che lavorano nel reparto *produzione*
6. cognome e nome del responsabile del reparto in cui lavora *Antico Nicola*
7. descrizione dei reparti e corrispondente responsabile
8. dipendente responsabile del reparto *produzione*
9. dipendenti sotto la responsabilità di *Rovetta Giorgio*
10. numero di telefono del reparto in cui lavora *Donati Claudia*
11. dipendenti che lavorano nel reparto avente n. telefono 204
12. dipendenti che sono responsabili di qualche reparto
13. dipendenti che non sono responsabili di alcun reparto
14. dipendenti che sono responsabili di più di un reparto
15. dipendenti che non lavorano in alcun reparto
16. dipendenti che lavorano da soli in reparto
17. dipendenti che lavorano in reparto con altri colleghi
18. reparti con i corrispondenti responsabili
19. reparti che non hanno responsabile
20. reparti che hanno responsabile
21. reparti in cui lavora più di un dipendente
22. reparti in cui lavora un solo dipendente
23. reparti in cui lavora almeno un dipendente
24. reparti in cui non lavora alcun dipendente
25. dipendenti che guadagnano di più
26. reparti e corrispondente numero di dipendenti
27. reparti con più di 10 dipendenti
28. numero di reparti senza responsabile

29. numero di dipendenti che sono responsabili di qualche reparto
30. dipendenti che lavorano in reparto da soli
31. reparti e corrispondenti ammontare degli stipendi dei dipendenti
32. media degli stipendi dei dipendenti di ciascun reparto
33. dipendendenti che guadagnano più di qualche responsabile
34. dipendendenti che guadagnano più del loro responsabile
35. responsabili che guadagnano meno di qualche dipendente
36. responsabili che guadagnano meno di qualche qualche dipendente sotto la loro responsabilità

2.22 Con riferimento all'esempio riportato nell'esempio **??**, spiegare cosa si ottiene dalla valutazione delle seguenti espressioni di algebra relazionale:

1. $\pi_{Abitanti}(\sigma_{Abitanti>500000}(\text{PROVINCIA} \bowtie_{Regione=Capoluogo} \text{REGIONE}))$
2. $\pi_{Abitanti}(\sigma_{Capoluogo=Sigla}(\text{PROVINCIA} \bowtie_{Regione=Capoluogo} \text{REGIONE}))$

2.23 Con riferimento allo schema relazionale riportato nell'esempio **??** esprimere, mediante espressioni dell'algebra relazionale, le seguenti interrogazioni:

1. dati completi relativi a tutte le province
2. popolazione della provincia di *Belluno*.
3. sigle delle province capoluogo di regione
4. province del *Veneto* con più di 500000 abitanti
5. province del *Veneto* e della *Lombardia* con corrispondente nome di regione
6. province capoluogo di regione
7. province con più di 1 milione di abitanti
8. province della stessa regione di *Belluno*
9. provincia capoluogo della regione nella quale si trova *Belluno*
10. provincia più popolata
11. popolazione italiana
12. province con un numero di abitanti maggiori della media italiana
13. province che si trovano nella regione più popolata
14. regioni a statuto *speciale*
15. province del *Veneto*
16. province del *Veneto* e della *Lombardia* e corrispondente regione
17. province del *Veneto* con più di 500000 abitanti
18. province e corrispondente nome della regione
19. province capoluogo di regione e corrispondente regione
20. province delle regioni a statuto speciale
21. province capoluogo di regione ed aventi più di 1 milione di abitanti

22. province capoluogo delle regioni a statuto speciale
23. popolazione delle province appartenenti a regioni a statuto speciale
24. province che si trovano in regioni aventi una popolazione maggiore di 2 milioni di abitanti
25. regioni aventi la provincia capoluogo di regione con più di 1 milione di abitanti
26. provincia capoluogo del *Veneto*
27. regione nella quale si trova *Belluno*
28. regioni e corrispondenti nomi delle province capoluogo
29. province capoluogo delle regioni a statuto speciale
30. province delle regioni a statuto speciale e corrispondente nome della regione
31. regione nella quale si trova la provincia più popolata
32. regioni aventi almeno una provincia con più di 500000 di abitanti.
33. regioni aventi tutte le province con più di 400000 abitanti
34. province capoluogo di regioni con capoluogo con più di 1 milione di abitanti
35. regioni aventi tutte le province con più di 300000 abitanti
36. regioni con province con più di mezzo milione di abitanti
37. province e corrispondente nome della regione
38. province capoluogo delle regioni a statuto speciale
39. province capoluogo di regione e corrispondente regione
40. province delle regioni a statuto speciale
41. regioni aventi la provincia capoluogo di regione con più di un milione di abitanti
42. regioni aventi almeno una provincia con più di un milione di abitanti
43. regioni aventi tutte le province con più di 200000 abitanti
44. numero di province
45. numero di province con più di 1 milione di abitanti
46. numero di province di ciascuna regione
47. numero di province con abitanti maggiori della media italiana
48. numero delle regioni
49. numero di regioni aventi province con più di un milione di abitanti
50. regioni aventi più di 5 province e corrispondente numero di province
51. regioni e corrispondente numero di province aventi più di mezzo milione di abitanti
52. regione più popolata
53. regioni con più di 3 milioni di abitanti
54. provincia più popolata di ciascuna regione e corrispondente numero di abitanti
55. popolazione complessiva delle regioni a statuto speciale e di quelle a statuto ordinario (2 righe)
56. province confinanti con *Belluno*

57. regioni confinanti con *Belluno*
58. province confinanti con il *Veneto*
59. regioni confinanti con il *Veneto*
60. province capoluoghi delle regioni a statuto speciale
61. province capoluoghi di regione aventi più di un milione di abitanti
62. regioni confinanti con il *Veneto*
63. province della regione di *Belluno* e confinanti con *Belluno*
64. province che non confinano con regioni diverse dalla propria
65. relazione dei confinamenti fra regioni, costituita dalle coppie di regioni fra loro confinanti
66. perimetro di ciascuna provincia
67. perimetro di ciascuna regione
68. regioni che confinano con nessun'altra regione (isole)
69. numero di gruppi di regioni fra loro connesse (1 continente e 2 isole)
70. nome della provincia più popolata di ciasuna regione

2.24 Aggiungendo alla tabella PROVINCIA un ulteriore attributo *Estensione*, formulare in linguaggio SQL le interrogazioni elencate sotto.

1. Densità di popolazione di ciascuna provincia.
2. Nome della regione più estesa.
3. Densità di popolazione italiana.
4. Densità di popolazione di ciascuna regione.
5. Densità di popolazione delle province del *Veneto*.
6. Estensione e popolazione di ciascuna regione.

2.25 Con riferimento all'esempio presentato nel problema 1.7.2, scrivere delle espressioni dell'algebra relazionale che risolvano le seguenti interrogazioni:

1. nome della cartella radice
2. cartelle di primo livello (figlie della cartella radice)
3. cartelle contenute nella cartella di nome `games`
4. cartelle contenenti una cartella di nome `chess`
5. cartelle che non contengono altre cartelle (foglie)
6. cartelle che compaiono più di una volta

2.26 Con riferimento all'esempio presentato nel problema 1.11, scrivere delle espressioni dell'algebra relazionale che risolvano le seguenti interrogazioni.
noindent L'attributo *DataEmi* denota la data di emissione della fattura e *DataPag* la data di pagamento. L'attributo *DataPag* ha valore $NULL$ nel caso in cui fattura non sia stata ancora pagata.

1. clienti di cui si conosce il numero di telefono
2. clienti ai quali è stata emessa qualche fattura
3. clienti ai quali non è stata emessa alcuna fattura
4. clienti che hanno fatture da pagare
5. clienti che hanno pagato qualche fattura
6. clienti che hanno pagato tutte le fatture
7. clienti che non hanno pagato alcuna fattura
8. clienti aventi fatture non ancora pagate ed aventi un importo superiore a 1000 Euro
9. clienti aventi fatture da pagare e corrispondente importo totale da pagare
10. clienti che hanno fatture da pagare per un ammontare superiore a 1000 euro

2.27 Nelle interrogazioni che seguono si fa riferimento ad uno schema relazionale VINCOLI $(Corso, Propedeutico)$ che esprime i corsi che sono propedeutici ad un dato corso di studi universitario. Ad esempio la tupla (*Algoritmi*, *Basi di Dati*) denota che il corso *Algoritmi* è propedeutico al corso *Basi di Dati*. Mediante gli operatori dell'algebra relazionale risolvere le seguenti interrogazioni:

1. lista completa dei corsi
2. corsi che hanno qualche propedeuticità, con l'indicazione delle propedeuticità di ciascuno
3. corsi direttamente propedeutici al corso *Basi di Dati*
4. corsi che non hanno alcuna propedeuticità

2.28 Descrivere uno schema relazionale adatto a memorizzare le informazioni relative ai libri di testo adottati nelle varie classi dell'*I.I.S. Negrelli-Forcellini* nel corrente anno scolastico. Ogni testo è indicato come *da acquistare* oppure *già acquistato*. Si richiede che vengano evidenziate anche le chiavi primarie e le chiavi esterne. Con riferimento allo schema descritto, scrivere delle espressioni dell'algebra relazionale che calcolino:

1. titolo ed autore dei testi di informatica adottati nell'istituto
2. elenco completo di tutti i testi adottati nella classe *5IT*
3. libri di testo della casa editrice *Atlas*
4. classi che adottano libri di testo della casa editrice *Loescher*
5. numero di studenti che acquistano libri della casa editrice *CEDAM*
6. spesa per l'acquisto di testi di ciascuna classe

2.29 Descrivere uno schema relazionale adatto a rappresentare le informazioni relative alle auto immatricolate ed i corrispondenti intestatari. Con riferimento allo schema descritto scrivere delle espressioni dell'algebra relazionale per risolvere le seguenti interrogazioni:

1. cognome e nome dei possessori di auto con cilindrata maggiore di 2000 cc

2. cognome e nome dei possessori di auto immatricolate nel 2014
3. elenco completo delle persone che possiedono auto
4. marche delle auto intestate a giovani che hanno meno di 20 anni
5. marche delle auto con cilindrata maggiore di 2000 cc
6. cognome e nome dei possessori di più di un'auto
7. cognome e nome dei possessori di una sola auto

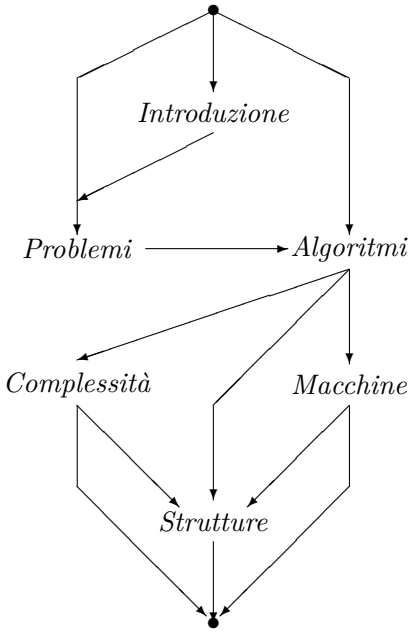
2.30 Un libro è organizzato in capitoli ognuno dei quali è suddiviso in paragrafi. Ogni capitolo ed ogni paragrafo è caratterizzato da un titolo ed è identificato da un numero progressivo. Serve inoltre registrare i numeri di pagina iniziale e finale di ciascun paragrafo. Descrivere graficamente, mediante una struttura ad albero, un esempio a piacere di una tale struttura di libro. Descrivere uno schema relazionale adatto a rappresentare la struttura ad albero del libro. Estendere lo schema con i dati che compaiono nell'esempio proposto. Con riferimento allo schema relazionale descritto, scrivere delle espressioni dell'algebra relazionale corrispondenti alle seguenti interrogazioni:

1. lista dei paragrafi con il corrispondente capitolo di appartenenza
2. lista dei paragrafi del secondo capitolo
3. numero e titolo dei capitoli contenenti paragrafi dal titolo *Esercizi*
4. lista dei capitoli con i corrispondenti numeri di pagina di inizio e fine capitolo
5. numero e titolo dei capitoli interamente contenuti nelle prime 100 pagine

2.31 In appendice ad un libro è riportato un grafo diretto aciclico che descrive le propedeuticità di lettura consigliata dei vari capitoli, secondo un percorso guidato dall'inizio alla fine; chiameremo *percorso di lettura* un qualsiasi percorso dal punto iniziale al punto finale. Individuare un possibile schema relazionale adatto a tradurre il grafo. Estendere lo schema con i dati che compaiono nel grafo riportato nella figura che segue. Con riferimento allo schema individuato, scrivere delle adeguate espressioni dell'algebra relazionale per determinare la risposta alle seguenti interrogazioni:

1. elenco di tutti i capitoli
2. elenco dei capitoli dai quali si può iniziare la lettura (ossia dei capitoli che non hanno alcuna propedeuticità)
3. elenco dei capitoli obbligatori (che devono essere obbligatoriamente letti in ogni percorso di lettura)
4. elenco dei capitoli facoltativi (che vengono omessi in un qualche percorso di lettura)

2.32 Descrivere uno schema di una base di dati relazionale per rappresentare i dati relativi alle *fatture* ed ai *clienti*. Le fatture devono registrare, fra gli altri dati, la data di emissione e la data di pagamento. Estendere lo schema con qualche dato significativo. Con riferimento allo schema proposto, scrivere un'espressione dell'algebra relazionale per determinare i nominativi dei clienti che hanno delle fatture da pagare.



2.33 Descrivere uno schema di una base di dati relazionale per rappresentare i dati relativi alle *fatture* ed ai *clienti*. Le fatture devono registrare, fra gli altri dati, la data di emissione e la data di pagamento. Descrivere in linguaggio SQL la struttura delle tabelle descritte al punto precedente. Con riferimento allo schema descritto al punto precedente, assumendo l'ipotesi che la data di pagamento abbia valore nullo se la fattura non è stata ancora pagata, scrivere un'interrogazione SQL per determinare nome, cognome e numero di telefono dei clienti che hanno fatture da pagare.

2.34 Descrivere uno schema relazionale adatto a rappresentare i modelli delle auto prodotte e le loro case costruttrici. Sullo schema evidenziare le chiavi primarie e le chiavi esterne. Lo schema descritto deve permettere le interrogazioni sotto descritte. Risolvere le seguenti interrogazioni:

1. Nomi delle case costruttrici tedesche che producono modelli *Coupè*.
2. Modelli di auto della *BMW* con motori *diesel* aventi più di 2000 cc di cilindrata.
3. Auto giapponesi con motori *diesel* aventi più di 2000 cc di cilindrata.
4. Case costruttrici che producono auto 4×4 con prezzi di listino inferiori a 12000 Euro.
5. Prezzo medio dei vari modelli di auto (*Station Wagon*, *SUV*, *Coupè*, ...).
6. Nome della casa costruttrice che fornisce un modello *Coupè* al prezzo più economico.

2.35 Descrivere uno schema relazionale adatto a rappresentare i modelli di computer (*tablet*, *notebook*, *desktop*, ...), le loro caratteristiche tecniche e le loro case

costruttrici. Sullo schema evidenziare le chiavi primarie e le chiavi esterne. Lo schema descritto deve permettere le interrogazioni descritte al seguente punto (c).

Risolvere le seguenti interrogazioni:

1. Nomi delle case costruttrici statunitensi che producono dei modelli *notebook*.
2. Computer aventi monitor da 10 pollici ed aventi un disco fisso di almeno 500 GB.
3. Prezzo medio dei vari modelli di computer (*tablet*, *notebook*, *desktop*, ...).

2.36 Descrivere uno schema relazionale adatto a rappresentare le varie attrezzature (computer, oscilloscopi, proiettori, ...) presenti nei vari laboratori dell'Istituto. Sullo schema evidenziare le chiavi primarie e le chiavi esterne. Lo schema descritto deve permettere le interrogazioni descritte al seguente punto (c).

Risolvere le seguenti interrogazioni:

1. Laboratori nei quali sono presenti dei videoproiettori ed aventi almeno 24 posti.
2. Numero di computer presenti in ciascun laboratorio.

3

MODELLARE

Modellare significa descrivere una versione semplificata di una porzione di interesse della realtà, individuandone gli elementi caratterizzanti ed i legami intercorrenti tra essi, al fine di poterla analizzare, capire e rappresentare in modo da rispettare più fedelmente possibile gli aspetti semantici.

Nel contesto delle basi di dati il processo di modellizzazione viene denominato *progettazione concettuale* e consiste nella definizione di uno schema concettuale ad alto livello che sintetizza i requisiti del progetto in modo rigoroso e formale in termini di entità, associazioni e vincoli sui dati.

3.1 Modelli concettuali

I modelli concettuali vengono utilizzati per la definizione ad alto livello dei dati e forniscono concetti, terminologie e notazioni utilizzati principalmente nelle fasi di analisi del progetto delle basi di dati. A livello concettuale la descrizione dei dati avviene mediante degli schemi e notazioni grafiche.

Il *modello Entità-Associazioni* (*Entity-Relationship* [1] *Model*), noto anche come *modello ER*, venne proposto nel 1976 da P.S.Chen e si è inizialmente imposto come standard nella definizione del modello concettuale dei dati; consente di rappresentare le entità, i loro attributi e le associazioni fra le entità.

Più recentemente, specialmente in connessione con le metodologie orientate agli oggetti e con lo sviluppo dell'ingegneria del software, si è affermata la metodologia e la notazione UML (*Unified Modelling Language*); in questo capitolo verrà utilizzato un sottoinsieme della notazione UML, limitandosi alla parte specifica relativa alla progettazione delle basi di dati.

3.2 Entità

Un'*entità* è una classe di oggetti (reali o astratti) che sono di interesse per l'applicazione, che hanno un'esistenza autonoma e delle proprietà comuni. Ogni entità ha un nome identificativo (generalmente declinato al singolare) che la individua univocamente nel contesto considerato.

Esempio 3.2.1 - In uno scenario relativo ad un'azienda che produce e vende prodotti, delle possibili entità potrebbero essere: `DIPENDENTE`, `REPARTO`, `PRODOTTO`, `CLIENTE`, `ORDINE`, `FATTURA`.

Esempio 3.2.2 - In uno scenario relativo ad un istituto scolastico delle possibili entità potrebbero essere: `STUDENTE`, `DOCENTE`, `CLASSE`, `AULA`.

Esempio 3.2.3 - In uno scenario relativo alla gestione di una biblioteca civica delle possibili entità potrebbero essere: `LIBRO`, `ARGOMENTO`, `AUTORE`, `LETTORE`.

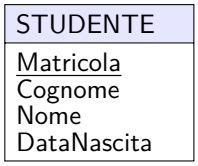
In notazione UML un'entità viene descritta graficamente mediante un rettangolo contenente il nome dell'entità stessa.

Esempio 3.2.4 - L'entità `STUDENTE` che rappresenta gli studenti che frequentano un istituto viene descritta mediante il seguente diagramma:

[1] In inglese il termine *relationship* indica un legame fra entità e si differenzia dal termine *relation* utilizzato per indicare una relazione con il significato di tabella secondo il modello relazionale; per questo motivo, anche in italiano si usa certe volte, benché impropriamente, il termine *relazione* con il significato di *associazione*.

Un'entità è caratterizzata da attributi; fra questi viene individuata obbligatoriamente la chiave primaria. Gli attributi vengono descritti in una apposita sezione rettangolare che segue il nome dell'entità [2] gli attributi che costituiscono la chiave primaria vengono sottolineati.

Esempio 3.2.5 - L'entità `STUDENTE` descritta nell'esempio precedente, arricchita degli attributi propri, viene descritta mediante il seguente diagramma:



Nella progettazione di una base di dati uno degli aspetti più delicati e decisivi consiste nell'individuare le entità a partire da un testo discorsivo e non formalizzato che descrive lo scenario di riferimento. I seguenti suggerimenti forniscono delle indicazioni per individuare le entità:

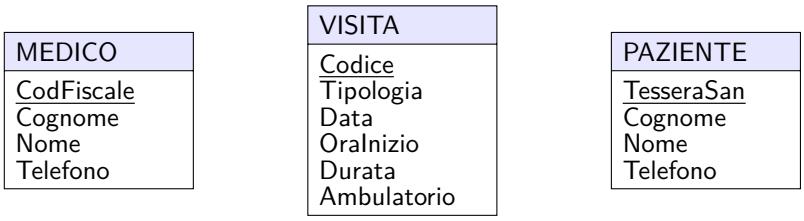
1. le entità vanno ricercate ed individuate tra i sostantivi del testo
2. ad ogni entità devono corrispondere più istanze di quell'entità
3. i nomi delle entità devono essere espressi al singolare
4. bisogna non confondere un'entità con un attributo (di un'entità)
5. non bisogna assumere come entità gli elementi del testo che costituiscono dei legami logici (associazioni) fra le entità (vedi prossimo paragrafo)
6. negli attributi delle entità non devono comparire chiavi esterne; queste sono rappresentate dalle associazioni (vedi prossimo paragrafo)

Osservazione. Un errore in cui cade spesso chi inizia a progettare una base di dati è quello di considerare come entità dei concetti che non lo sono: passando all'implementazione della base di dati, ad un'entità corrisponderà una tabella che verrà popolata da varie tuple; se si capisce che ad un'ipotetica entità corrisponderà una sola tupla, significa che non si tratta di un'effettiva entità ma di una singola istanza che eventualmente appartiene ad una classe in un modello più esteso di quello che si è considerato. Ad esempio sono da evitare i sostantivi che corrispondono al singolo ambiente di riferimento, quali: *Azienda*, *Ospedale*. Sono da evitare come entità anche i termini generici ed inconsistenti quali: *Controllo*, *Informazioni*, *Gestione*. Per accorgersi di essere sulla strada sbagliata è spesso sufficiente riscontrare delle difficoltà nell'individuare gli attributi della presunta entità.

[2] Nella notazione dei diagrammi ER gli attributi vengono indicati mediante un segmento con un pallino finale (—∘), attaccato al rettangolo dell'entità, a fianco del quale viene riportato il nome dell'attributo; con simile simbolo pieno (—•) si denota un attributo che costituisce la chiave primaria.

Esempio 3.2.6 - Si deve realizzare la gestione automatizzata delle prestazioni specialistiche fornite da un Ospedale, per consentire le prenotazioni agli sportelli, con controllo della disponibilità e registrazione della prenotazione. I dati da organizzare riguardano i medici che operano presso il dato Ospedale, i diversi tipi di visite specialistiche, le informazioni anagrafiche dei pazienti che richiedono le visite, le prenotazioni delle viste da parte dei pazienti.

Analizzando lo scenario proposto cerchiamo di individuare le entità tra i sostantivi corrispondenti agli attori ed agli oggetti fondamentali. Una soluzione minimale, che si limita a riportare le entità descritte dai pochi elementi forniti esplicitamente dal testo del problema, è la seguente:



3.3 Associazioni

Un'*associazione* esprime un legame logico fra due o più entità; nel caso in cui le entità associate siano due, si parla di *associazione binaria*. Ogni associazione è individuata univocamente da un identificatore che descrive il tipo di legame; questo nome corrisponde generalmente ad un predicato verbale e viene scritto nel modo infinito.

3.3.1 Associazioni binarie

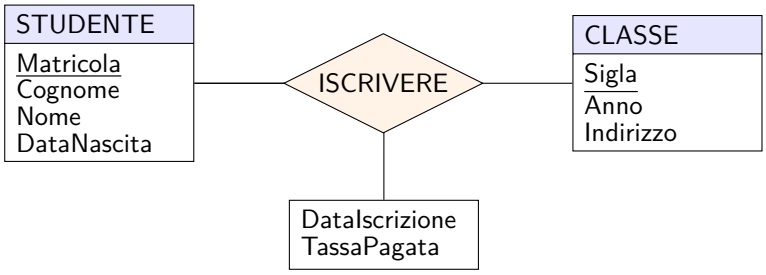
Un'associazione (binaria) viene descritta graficamente mediante due rettangoli uniti mediante un segmento avente nella parte centrale un rombo [3] in cui è scritto il nome dell'associazione, come descritto nell'esempio che segue.

Esempio 3.3.1 - Completiamo lo schema concettuale abbozzato nell'esempio 3.2.6 inserendo le associazioni che legano fra loro le varie entità; le associazioni sono caratterizzate da predicati verbali o da sostantivi aventi nella loro radice la derivazione da un predicato verbale (ad esempio, *prenotazioni*, da *prenotare*). La soluzione, nella quale vengono per semplicità omessi gli attributi delle entità, è la seguente:

[3] Nella notazione UML standard il nome dell'associazione viene scritto in corrispondenza del segmento che unisce le due entità, senza essere racchiuso in un rombo; la notazione a rombo, caratteristica dei diagrammi ER, viene comunque usata in queste pagine per dare maggiore evidenza alle associazioni.

Esempio 3.3.2 - Il diagramma che segue esprime il fatto che gli studenti sono iscritti in certe classi.

Questo diagramma può essere ulteriormente dettagliato riportando gli attributi delle entità coinvolte nell'associazione; può accadere che serva indicare degli attributi riferibili all'associazione stessa, come ad esempio: la data di quando è stata fatta l'iscrizione e la tassa pagata; questi attributi vengono riportati in un apposito riquadro agganciato all'associazione stessa, come si vede nel diagramma che segue.



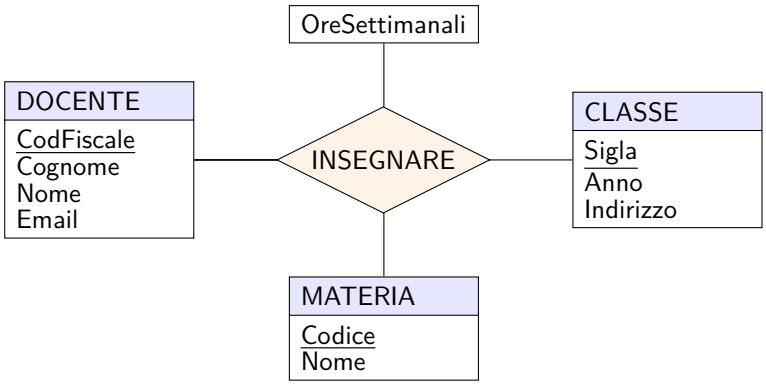
Osservazione. Un frammento di schema concettuale contenente (almeno) un'associazione può essere letto mediante delle proposizioni della lingua italiana che possono essere indagate a livello di analisi logica. Nel caso di un'associazione binaria fra due entità si ottengono proposizioni elementari in cui una delle due entità gioca il ruolo di soggetto, l'associazione corrisponde al predicato verbale e l'altra entità assume il ruolo di complemento. Questa lettura può essere esplicitata mediante una singola proposizione di tipo intensionale (ad esempio, "Uno studente è iscritto ad una classe") alla quale corrispondono, a livello estensionale, molte istanze di proposizioni (ad esempio: "Lo studente Bini Augusto è iscritto alla classe 2A", "Lo studente Mariani Aldo è iscritto alla classe 4C", ...). Queste proposizioni devono essere completate con i valori corrispondenti agli attributi *DataIscrizione* e *TassaPagata* che arrichiscono la proposizione di ulteriori due complementi. In riferimento ad una base di dati, ciascuna di queste proposizioni atomiche darà vita ad una tupla che verrà memorizzata in una (opportuna) tabella che verrà definita traducendo lo schema concettuale in uno schema logico relazionale. Nel caso in cui il predicato verbale corrispondente all'associazione sia transitivo e, quindi, le due entità assumano il ruolo di soggetto e di complemento oggetto, la proposizione corrispondente al dato frammento di schema può essere tradotta sia in forma attiva che passiva. Ad esempio, utilizzando un predicato verbale transitivo (*frequentare*) al posto del predicato verbale intransitivo (*iscrivere*), si possono derivare le seguenti due proposizioni semanticamente equivalenti: "Uno studente frequenta una classe" e "Una classe è frequentata da degli studenti". Nella

progettazione di una base di dati si richiede di svolgere il processo inverso: passare da una descrizione testuale, informale, con eventuali lacune ed incoerenze, ad una sua distillazione precisa, coerente e completa che avrà, come prodotto, uno schema concettuale che poi verrà tradotto nel modello logico relazionale che successivamente verrà implementato mediante un DBMS relazione.

3.3.2 Associazioni ternarie

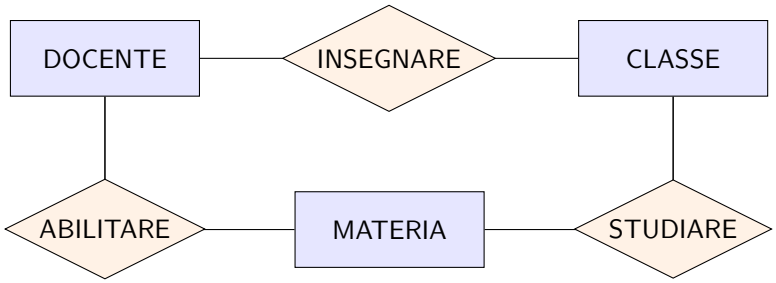
In talune situazioni si fa ricorso a delle associazioni che legano fra loro più di due entità. Il seguente esempio descrive il caso di un'*associazione ternaria*.

Esempio 3.3.3 - Lo schema che segue descrive i docenti insegnano delle materie in delle classi.



Un'istanza di affermazione derivabile da questo frammento di schema concettuale è: "Il prof. Rossi insegna Italiano nella classe 2A per 4 ore settimanali". Dal punto di vista della struttura secondo l'analisi del periodo, questa proposizione è una principale formata da un soggetto (entità `DOCENTE`) e da due complementi corrispondenti alle due entità `CLASSE` e `MATERIA` e da un complemento di tempo corrispondente all'attributo OreSettimanali.

Osservazione. Per confronto con lo schema sopra esaminato, risulta interessante prendere in esame il seguente schema concettuale che descrive i docenti che insegnano nelle varie classi, le materie nelle quali i docenti sono abilitati all'insegnamento e le materie che vengono studiate nelle classi.

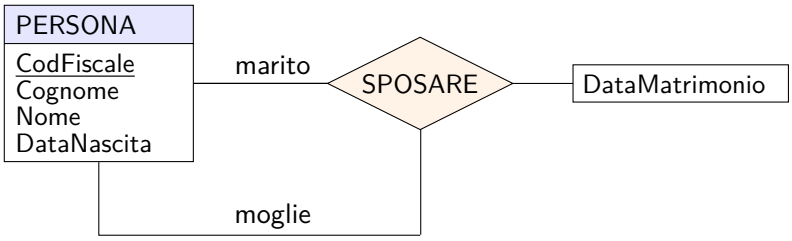


Questa soluzione alternativa è sostanzialmente diversa dalla precedente in quanto non indica quali materie insegna un docente in una classe. Il vincolo che un docente insegni in una classe una materia per la quale è abilitato non è garantito da nessuna delle due soluzioni; vincoli come questo non si possono definire mediante uno schema concettuale ma vanno descritti a parte e gestiti poi mediante apposite procedure del DBMS sul quale verrà implementato lo schema concettuale.

3.3.3 Associazioni ricorsive

Una associazione si dice *ricorsiva* se lega un'entità con essa stessa. Nel caso di relazioni ricorsive non simmetriche, occorre definire i *ruoli* di partecipazione dell'entità alla relazione; tali ruoli vengono indicati nello schema, in corrispondenza dei due rami dell'associazione.

Esempio 3.3.4 - Consideriamo la situazione in cui un Ufficio Anagrafe di un Comune deve registrare i vari matrimoni celebrati presso il dato Comune. In questo caso si evidenzia un'associazione fra due istanze della stessa entità `PERSONA`. In corrispondenza ai due rami dell'associazione vengono indicati i due ruoli *marito* e *moglie* ed in corrispondenza dell'associazione è riportato l'attributo DataMatrimonio.

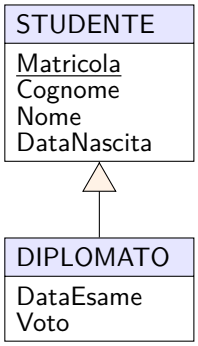


Questo schema si presta facilmente ad essere adattato per poter rappresentare le situazioni di separazione e di vedovanza: per il primo caso è sufficiente aggiungere un attributo DataSeparazione agli attributi dell'associazione `SPOSARE`, per il secondo caso è sufficiente aggiungere un attributo DataMorte agli attributi dell'entità `PERSONA`. Questi attributi, a livello estensivo nel modello logico relazionale, potranno avere dei valori nulli.

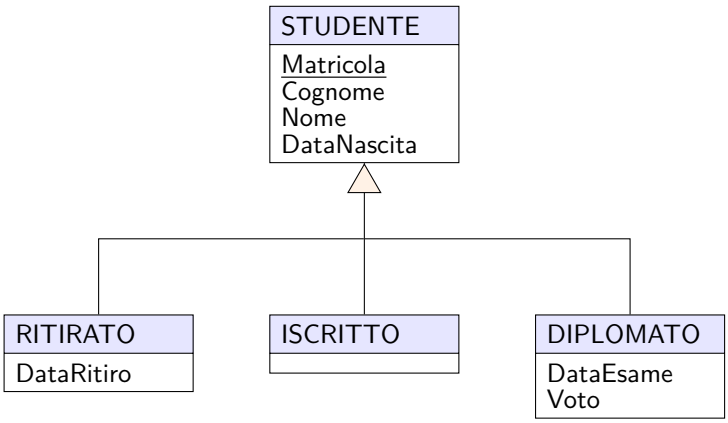
3.3.4 Associazioni di specializzazione/generalizzazione

Un'associazione di *specializzazione/generalizzazione* o *gerarchia is-a* esprime una particolare forma di associazione fra due entità in cui un'entità padre corrisponde ad un insieme base di oggetti e l'altra entità figlia corrisponde ad una specializzazione dell'entità padre; l'entità figlia eredita gli attributi dell'entità padre e può aggiungerne degli altri.

Esempio 3.3.5 - Consideriamo gli studenti iscrittisi negli anni in un dato istituto e gli studenti che si sono ivi diplomati. Questa situazione viene descritta mediante il seguente diagramma:



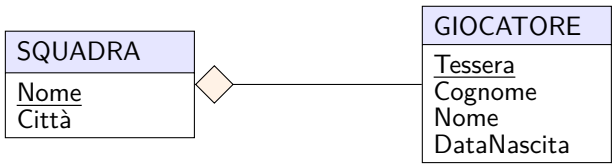
Un'entità padre può avere più entità figlie che la specializzano. Ad esempio, per descrivere gli studenti che si sono ritirati, quelli che attualmente frequentano e quelli che si sono diplomati negli anni si può ricorrere al seguente diagramma:



3.3.5 Associazioni per aggregazione

In molte situazioni si incontrano particolari forme di associazione che corrispondono al caso in cui un'entità è costituita da un'*aggregazione* di elementi di un'altra entità. Questa forma di associazione, pur non avendo delle proprietà strutturali particolari, viene denotata con una specifica notazione, come descritto nel diagramma che segue:

Esempio 3.3.6 - Per rappresentare una squadra composta da giocatori, si può usare il seguente diagramma:



3.4 Cardinalità delle associazioni

Le associazioni devono generalmente rispettare dei vincoli che limitano le possibilità di associazione delle istanze delle entità coinvolte nell'associazione.

I vincoli più importanti sono quelli che riguardano la cardinalità. La *cardinalità* (o *molteplicità*) di un'associazione binaria specifica il numero di possibili istanze di un'entità che viene messo in corrispondenza con un'istanza dell'altra entità. Una notazione per descrivere la cardinalità si basa sul rapporto di cardinalità per associazioni binarie: in corrispondenza a ciascun arco partecipante all'associazione viene indicato 1 oppure N che assumono il significato di "solo uno" e "un numero qualsiasi".

Le possibilità che si possono verificare sono le seguenti tre:

1. associazione *uno-a-uno* indicata con 1:1
2. associazione *uno-a-molti* indicata con 1:N
3. associazione *molti-a-molti* indicata con N:N

In aggiunta, sull'arco partecipante si può aggiungere un trattino che indica "opzionalità" della partecipazione, ossia la possibilità che alcune istanze di un'entità possano non essere associate ad alcuna istanza dell'entità associata; in queste situazioni l'associazione viene qualificata come *parziale*; altrimenti si dice *totale*. In combinazione con con l'1 o N l'opzionalità ha il significato di "0 o 1" oppure "0 o più". Tutte queste diverse situazioni sono illustrate negli esempi che seguono.

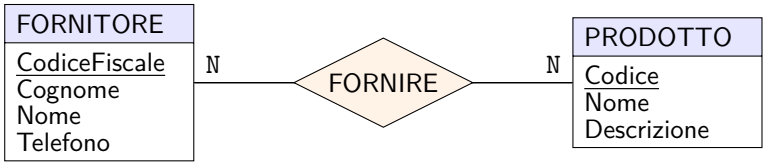
Esempio 3.4.1 - Un'azienda di vendite è organizzata con più filiali nel territorio, a capo delle quali viene nominato un direttore responsabile:

In questo caso le cardinalità dell'associazione `DIRIGERE` vanno lette in questo modo: un direttore dirige una sola filiale ed una filiale è diretta da un solo direttore. Il trattino di parzialità indica che ci possono essere delle filiali che non hanno un direttore.

Esempio 3.4.2 - In un'azienda di produzioni industriali lavorano vari dipendenti ciascuno dei quali presta servizio in uno dei vari reparti in cui è strutturata l'azienda:

Le cardinalità dell'associazione `LAVORARE` indicano che in un reparto possono lavorare 0 o più dipendenti ed un dipendente può lavorare al più in un solo reparto.

Esempio 3.4.3 - Un'azienda di trasformazione industriale si approvigiona dei prodotti primari attraverso una rete di fornitori non esclusivi. La situazione è descritta mediante il seguente schema concettuale:



In questo caso le cardinalità dell'associazione `FORNIRE` indicano che un fornitore può fornire diversi prodotti ed uno stesso prodotto può essere fornito da più fornitori.

Osservazione. Le cardinalità di un'associazione saranno un elemento importante quando più avanti si vedranno le modalità con le quali uno schema concettuale viene tradotto in uno schema logico relazionale.

3.5 Entità deboli

Un'entità si dice *entità debole* se non ha una propria chiave primaria. In contrapposione, le entità dotate di una loro chiave primaria (quelle viste finora) vengono dette *entità forti*. Le entità deboli vengono identificate tramite le loro associazioni ad altre (una o più) entità forti, dette *entità proprietarie*. Un entità debole ha sempre un vincolo di partecipazione totale relativo alla sua *associazione identificante*. Un'entità debole viene identificata univocamente mediante dei propri attributi (chiave primaria parziale) unitamente agli attributi chiave dell'entità proprietario alla quale è legata.

In uno schema concettuale un'entità debole e le associazioni proprietarie che la identificano vengono rappresentate mediante una linea tratteggiata [4]; gli attributi che costituiscono la chiave primaria parziale dell'entità debole vengono sottolineati con una linea tratteggiata.

[4] In taluni testi viene utilizzata una linea doppia al posto della linea tratteggiata.

Esempio 3.5.1 - Consideriamo gli studenti iscritti alle università italiane. Uno studente ottiene un numero di matricola presso l'università alla quale si è iscritto; un'università è identificata univocamente dal proprio nome e dalla città nella quale ha sede; pertanto uno studente iscritto ad un'università italiana risulta univocamente identificato dal proprio nome di matricola, dal nome dell'università presso la quale è escritto e dalla città dove ha sede l'università; ad esempio è univocamente determinato lo studente avente numero di matricola 123456 iscritto presso il *Politecnico* di *Milano*. La situzione è descritta nel seguente schema.

3.6 Sviluppo di uno schema concettuale

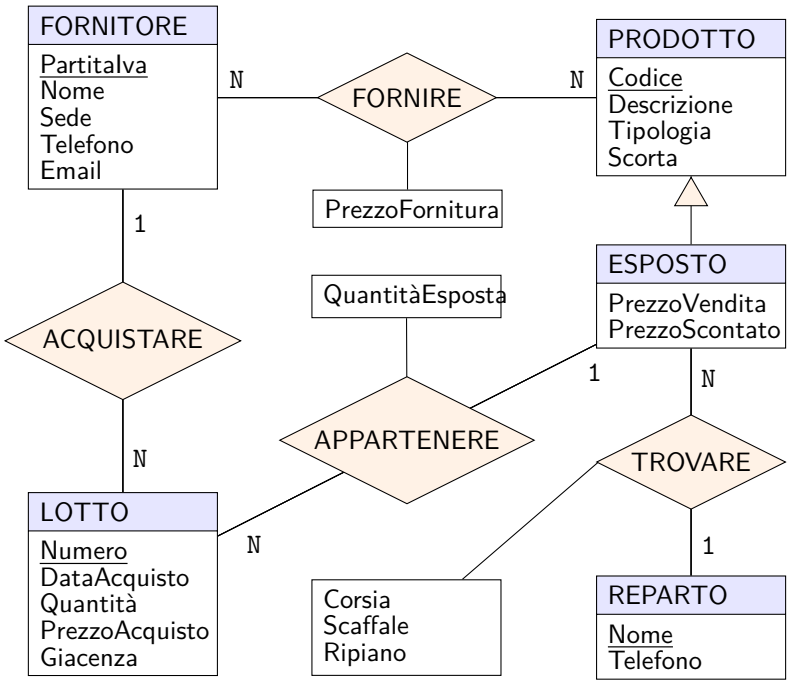
La definizione dei dati di una base di dati deve essere funzionale a supportare i programmi che elaborano i dati. Con questi obiettivi generali, lo sviluppo di uno schema concettuale è un processo dinamico che si evolve a partire da alcune entità fondamentali che vengono fra loro assocciate; si prosegue con l'aggiunta di altre entità che vengono agganciate allo schema complessivo mediante delle associazioni. È importante aver ben chiaro il significato delle varie entità che vengono inserite nello schema, definendo per ciascuna gli attributi caratterizzanti. Per maggiore chiarezza si usa spesso associare allo schema concettuale un *dizionario dei termini* dove vengono spiegati i significati e gli attributi delle entità coinvolte nello schema.

Esempio 3.6.1 - Si vuole gestire un magazzino di prodotti organizzato per reparti, tenendo sotto controllo anche la situazione dei fornitori; in particolare si vuole poi rispondere alle seguenti esigenze:

- produrre listini di prodotti con descrizione, prezzi e sconti
- produrre elenchi di prodotti con i relativi fornitori
- controllare i prodotti sotto scorta

Nelle considerazioni che seguono ci si può lasciare guidare dalla situazione di un supermercato di prodotti alimentari. Un primo abbozzo dello schema concettuale emerge direttamente dall'analisi testuale dello scenario di riferimento in base alla quale emergono immediatamente le seguenti tre entità, legate fra loro da due associazioni binarie:

Di queste, l'entità PRODOTTO va precisata meglio: con "prodotto" si intende, ad esempio, "pomodoro" o "una confezione da 500 g. di pomodoro ciliegino della ditta Orto Fresco"? In certe situazioni si hanno margini di discrezionalità, ma nel presente contesto di un magazzino che vende prodotti alimentari è evidente che si tratta della seconda alternativa. Rimane ancora da differenziare il concetto di "prodotto su catalogo" e "prodotto fisico presente in magazzino ed esposto in un reparto in una data quantità"; considerazioni commerciali portano ad introdurre un'entità LOTTO intesa come unità di acquisto dal fornitore di un certa quantità di un prodotto ed unità su cui applicare sconti o altre operazioni di gruppo. In base alle precedenti considerazioni si arriva al seguente schema concettuale:



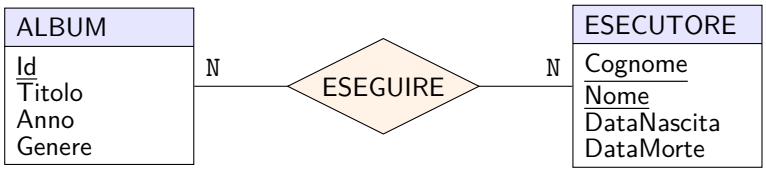
Per descrivere in modo più dettagliato lo schema concettuale riportato sopra si può fare ricorso al seguente *dizionario dei termini*:

- FORNITORE: ditta o persona che fornisce i prodotti ad un dato prezzo
- PRODOTTO: prodotto fornito da un fornitore, avente una data categoria di qualità, una grandezza ed un valore di scorta desiderabile da avere in magazzino
- ESPOSTO: prodotto esposto (in una data quantità)
- LOTTO: partita di acquisto di una certa quantità di prodotto; la giacenza indica la quantità del lotto non ancora venduta (presente in magazzino o in reparto)
- REPARTO: parte del negozio dove si trovano i prodotti esposti

Nel processo di sviluppo di uno schema concettuale può succedere che un attributo di un'entità venga elevato al rango di entità di un'istanza ed incapsuli dei suoi attributi; tale nuova entità verrà poi agganciata allo schema complessivo mediante un'associazione. In questo processo di distaccamento di un attributo è importante che nell'entità che perde l'attributo non vengano mantenuti dei rimasugli di attributo con significato di chiave esterna referente all'entità che si è creata: in un modello concettuale dei dati non esistono chiavi esterne.

Esempio 3.6.2 - Uno schema concettuale minimale per descrivere una base di dati per catalogare gli album musicali per la propria discografia potrebbe fondarsi sulla seguente singola entità:

Questa prima versione dello schema concettuale risulta inadeguata in quanto i dati di un album inciso da più esecutori non troverebbe rappresentazione. Si può risolvere il problema estrapolando l'attributo Esecutore elevandolo al rango di entità; in questo modo l'entità `ALBUM` si indebolisce e d'altra parte non può essere identificata esternamente dall'entità `ESECUTORE` visto che l'associazione fra `ALBUM` ed `ESECUTORE` avrà cardinalità N:N. Una possibile soluzione può essere ottenuta rendendo forte l'entità `ALBUM` con l'aggiunta di un attributo identificante, da assumere come chiave primaria. Si arriva così alla seguente situazione:



3.7 Regole di derivazione

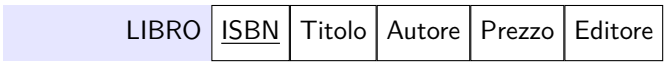
I modelli concettuali vengono usati come strumento per la descrizione del progetto di una base di dati; per passare alle fasi realizzative è necessario tradurlo in un corrispondente modello logico relazionale mediante un procedimento sostanzialmente univoco e meccanico, basato su dei principi guida, denominati *regole di derivazione*, che verranno descritte nei seguenti punti.

1. Ogni entità viene tradotta con una relazione avente il nome e gli stessi attributi dell'entità; la chiave primaria della relazione coincide con la chiave primaria dell'entità.

   Esempio 3.7.1 - L'entità rappresentata dallo schema concettuale

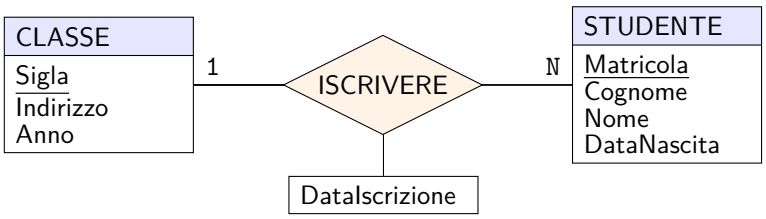
   

   viene tradotta in una relazione avente il seguente schema logico:

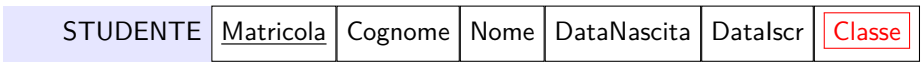
2. Un'*associazione binaria* 1:N viene tradotta mediante due relazioni che traducono le due entità, aggiungendo nella relazione che traduce l'entità che svolge il ruolo N un attributo aggiuntivo che funge da chiave esterna per relazionarsi all'entità che gioca il ruolo 1. Solitamente questo attributo ha il nome dell'entità verso la quale si relaziona. Eventuali attributi dell'associazione vengono inglobati nella relazione che svolge il ruolo N.

   Esempio 3.7.2 - Il frammento di schema concettuale che segue, rappresenta gli studenti iscritti alle classi.

   Può essere tradotto in uno schema logico relazionale composto da due relazioni come segue [5]:

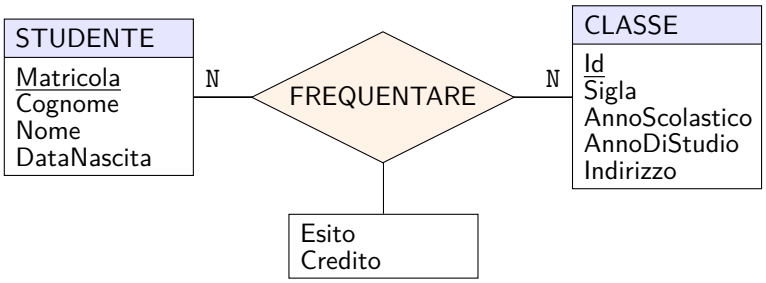
   

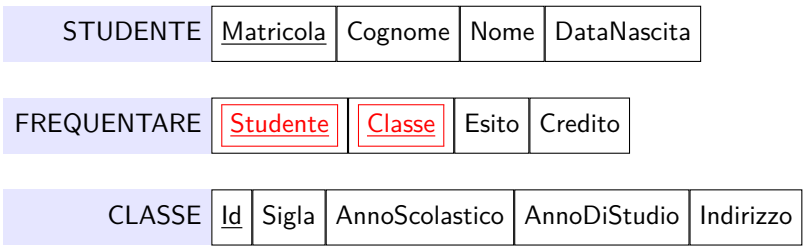
[5] Gli attributi racchiusi in un rettangolo ed in colore rosso denotano una chiave esterna referente alla relazione omonima.

3. Un'*associazione binaria* N:N viene tradotta mediante tre relazioni: due traducono le due entità coinvolte nell'associazione e la terza traduce l'associazione; questa relazione ha come attributi le due chiavi esterne per relazionare le due entità coinvolte nell'associazione, unitamente agli eventuali attributi dell'associazione. La chiave primaria risulta costituita dalla coppia delle chiavi esterne.

Esempio 3.7.3 - Per registrare il percorso scolastico di uno studente, ossia le varie classi frequentate, l'esito finale di ciascun anno (*promosso*, *non ammesso*, *ritirato*) ed il credito scolastico, si può utilizzare il seguente frammento di schema concettuale:

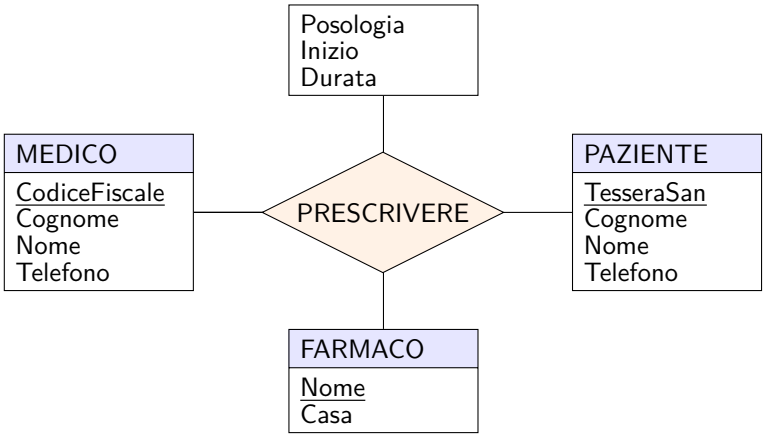


Questo schema può essere tradotto in uno schema logico relazionale composto dai seguenti tre schemi di relazione:



4. Un'*associazione ternaria* viene tradotta come segue: le tre entità vengono tradotte ciascuna mediante una relazione con i corrispondenti attributi e le rispettive chiavi primarie; l'associazione viene tradotta mediante una apposita relazione avente per attributi tre chiavi esterne che la legano alle tre entità ai quali si aggiungono gli eventuali attributi dell'associazione.

Esempio 3.7.4 - Lo schema concettuale che segue modellizza il seguente scenario: "In un dato ospedale i medici prescrivono delle cure con farmaci ai pazienti". Un'istanza di proposizione che deve trovare rappresentazione nello schema è la seguente: "Il dott. Nicola Pagani ha prescritto al paziente Giuseppe Verdini di prendere mattina e sera 2 compresse da 10 mg del farmaco Laremisol".



L'attributo *posologia* precisa la quantità e le modalità di assunzione di un farmaco. Questo schema concettuale viene tradotto come segue:

MEDICO	CodiceFiscale	Cognome	Nome	Telefono
PAZIENTE	TesseraSan	Cognome	Nome	Telefono
FARMACO	Nome	Casa		
PRESCRIVERE	Medico	Paziente	Farmaco	Posologia	Inizio	Durata

5. Un'*entità debole* viene tradotta come un'associazione binaria N:1, aggiungendo fra gli attributi dell'entità che gioca il ruolo di entità debole degli attributi corrispondenti all'entità forte associata attraverso l'associazione identificante; questi attributi aggiuntivi, che hanno la funzione di chiavi esterne, unitamente agli attributi propri dell'entità debole (che costituivamo la chiave primaria parziale) costituiscono una chiave primaria per la relazione corrispondente all'entità debole.

Esempio 3.7.5 - La seguente porzione di schema concettuale descrive i dati relativi a delle città del mondo e i dati della corrispondente nazione in cui queste città si trovano. Assumendo che nel mondo esistano (com'è vero) città con lo stesso nome in nazioni diverse, si riconosce che l'entità `CITTA` è debole. Lo schema che segue descrive questa situazione.

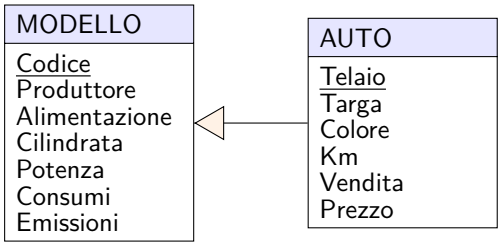
Questo schema può essere tradotto in uno schema logico relazionale composto dai seguenti tre schemi di relazione:

CITTA	Nome	Abitanti	Nazione

NAZIONE	Sigla	Nome	Capitale

6. Un'associazione di *specializzazione/generalizzazione* (*is-a*) viene tradotta nel modello logico relazionale come fosse un'associazione 1:N dove il ruolo di 1 è rappresentato dall'entità padre.

Esempio 3.7.6 - Consideriamo il parco macchine presente presso un concessionario, dove si trovano auto nuove da vendere, auto già vendute ed in attesa di consegna al cliente e auto usate. Alcuni dati generali sono specifici del modello di auto della casa costruttrice Si ed altri sono relativi alle singole auto immatricolate. Si può descrivere questa situazione mediante il seguente schema concettuale:

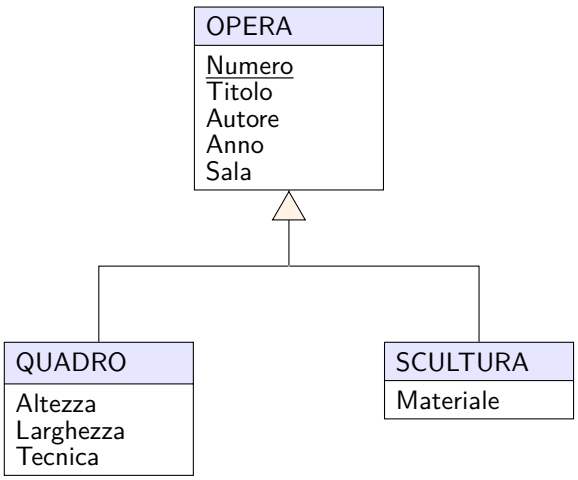


La traduzione nel modello logico relazionale è la seguente:

MODELLO	Codice	Produttore	Cilindrata	Consumi	Emissioni

AUTO	Telaio	Targa	Colore	Km	Vendita	Prezzo	Modello

Esempio 3.7.7 - Un museo d'arte moderna classifica le varie opere presenti nelle sue sale mediante il seguente schema concettuale:



La traduzione nel modello logico relazionale è la seguente:

OPERA	Numero	Titolo	Autore	Anno	Sala

QUADRO	Altezza	Larghezza	Tecnica	Opera

SCULTURA	Materiale	Opera

3.8 Realizzazione di una base di dati

La progettazione di una base di dati si inserisce in un processo molto articolato, denominato *ciclo di vita del software*, che può essere schematizzato nelle seguenti fasi:

1. *RACCOLTA ED ANALISI DEI REQUISITI*: la prima fase del processo di realizzazione di una base di dati consiste nella *raccolta e analisi dei requisiti*: queste informazioni vengono acquisite mediante interviste al committente e poi strutturate in modo preciso, non ambiguo e completo mediante vari passaggi di affinamento e confronto con il committente. Assieme al committente viene analizzato il problema e vengono individuati gli obiettivi ed i requisiti da soddisfare. Al termine di questa fase viene redatto un documento in cui il committente ed i tecnici formalizzano quanto dovrà essere realizzato.

2. *PROGETTAZIONE*: si svolge in vari passi successivi:

   (a) progettazione concettuale: stesura (su carta o mediante programma apposito) di un modello del database che si intende realizzare: viene prodotto lo schema concettuale dei dati, svolgendo le seguenti azioni, in modo interlacciato e soggette a continue integrazioni e revisioni, in un processo dinamico svolto con continui riferimenti al progetto emerso nella fase precedente; vengono individuati e definiti i seguenti elementi

      - entità
      - associazioni fra le varie entità
      - attributi delle entità e delle associazioni
      - cardinalità delle associazioni

      La progettazione avviene in fasi successive ad un sempre più spinto livello di dettaglio (top-down) ed integrazione di schemi già definiti (bottom-up).

   (b) progettazione logica: consiste nella traduzione dello schema concettaule in uno schema logico relazionale (in tabelle).

3. *REALIZZAZIONE*: Si possono individuare le seguenti sottofasi:

   (a) definizione: viene definita la struttura della base di dati: lo schema logico dei dati viene realizzato usando appositi linguaggi di programmazione (ad esempio SQL) ed un DBMS (ad esempio MySQL, Postgres, Oracle).

   (b) popolamento: vengono inseriti i dati iniziali nelle tabelle; in molti casi i dati vengono importati da vecchie versioni del database o da preesistenti archivi di dati (digitali o cartacei)

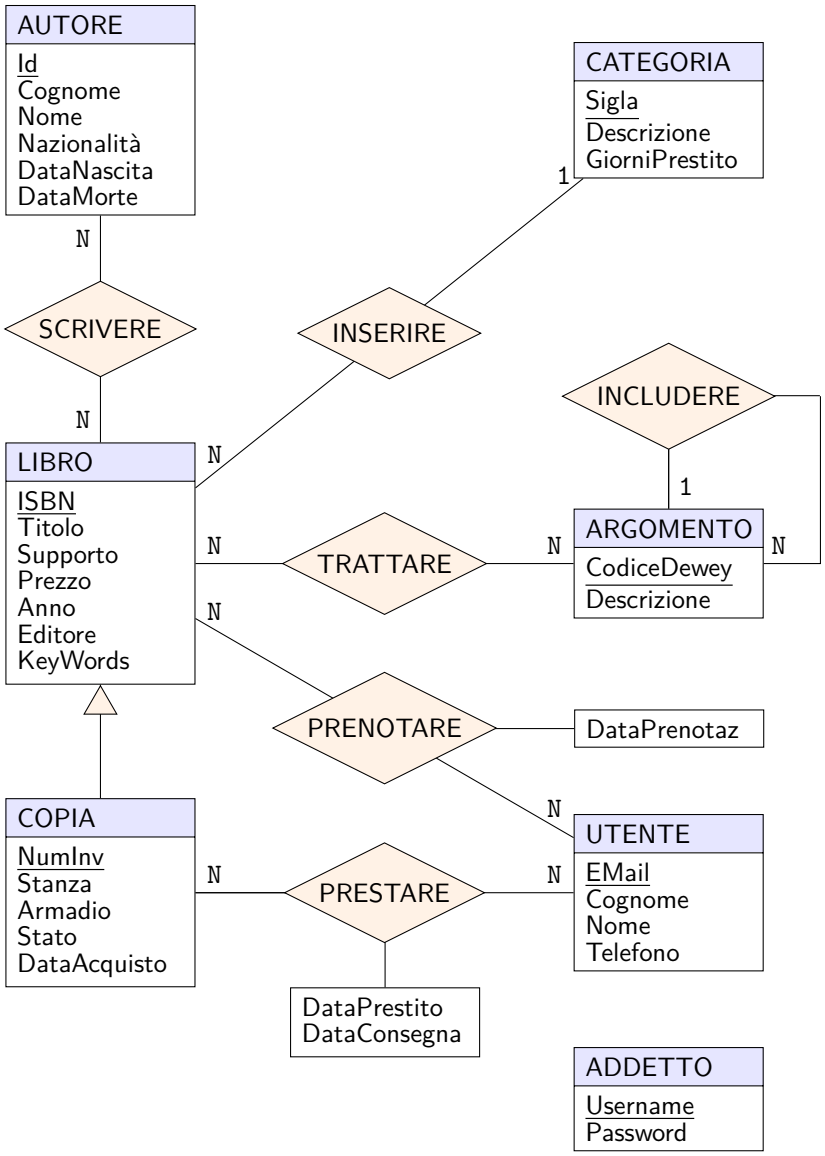
4. *UTILIZZO*: Si possono individuare le seguenti sottofasi:
   (a) elaborazione: sulle tabelle vengono eseguite operazioni di inserimento/modifica/cancellazione di tuple
   (b) interrogazione: sulle tabelle contenenti i dati vengono eseguite delle interrogazioni da parte degli utenti finali, attraverso specifiche interfacce grafiche (che innescano delle queries SQL già predisposte dai programmatori); il DBMS interpreta le interrogazioni e presenta, a video o su carta, appositi report.
   (c) manutenzione: in fase di esercizio, la base di dati ha bisogno di continue azioni di manutenzione.

Problema 3.8.1 Un istituto scolastico, dislocato su più sedi, vuole realizzare un sistema automatizzato per la gestione dei libri della propria biblioteca. I libri della biblioteca sono localizzati all'interno di armadi e mensole presso alcune stanze dislocate presso le varie sedi dell'istituto; in ogni caso, per avere il prestito gli utenti si rivolgono ai docenti responsabili della biblioteca presso la sede frequentata, nei giorni e negli orari stabiliti e consultabili sul sito dell'istituto. Ci possono essere più copie di uno stesso libro. Ogni pubblicazione riguarda uno o più argomenti. Sono ammessi ai prestiti tutti gli studenti iscritti all'istituto, tutti i docenti che insegnano nell'istituto ed il personale dell'istituto (segretarie, tecnici, ...). Gli utenti autorizzati possono, via web o collegandosi direttamente mediante i computer interni all'istituto, consultare per argomenti e sotto argomenti i libri presenti oppure fare delle interrogazioni sui libri presenti, precisando argomento o autori o delle parole chiave. Un utente può prenotare un libro al momento non disponibile. È possibile prendere in prestito fino ad un massimo di 3 libri contemporaneamente. Il periodo massimo del prestito dipende dalla tipologia del libro (romanzo, libro tecnico, vocabolario, ...). Alla scadenza del prestito viene inviata in automatico una email di avvertimento del prestito che sta per scadere e, nel caso di prestiti non rientrati, vengono inviati dei solleciti. Al termine dell'anno scolastico gli studenti devono consegnare tutti i libri avuti in prestito. Per questioni statistiche ed organizzative bisogna tenere traccia di uno storico di tutti i prestiti passati. Alcune persone vengono abilitate a svolgere particolari operazioni di gestione quali: inserire nuovi libri, concedere o recuperare dei prestiti, inviare mail per sollecitare la consegna di prestiti scaduti, autorizzare nuovi utenti e sospendere o cancellare utenti inadempienti, svolgere interrogazioni statistiche sui prestiti.

Soluzione. È importante distinguere i "dati" dalle "operazioni" (manipolazioni, interrogazioni) che devono essere eseguite sui dati: la progettazione della base di dati si limita alla definizione dei soli dati, tenendo comunque presente che i dati devono permettere la realizzazione di tutte le operazioni e funzionalità richieste. L'analisi dei requisiti del problema fa emergere le seguenti entità: *LIBRO*, *AUTORE*, *UTENTE*. Ragionando sul significato da attribuire all'entità *LIBRO* si capisce che bisogna distinguere il concetto di "libro" inteso

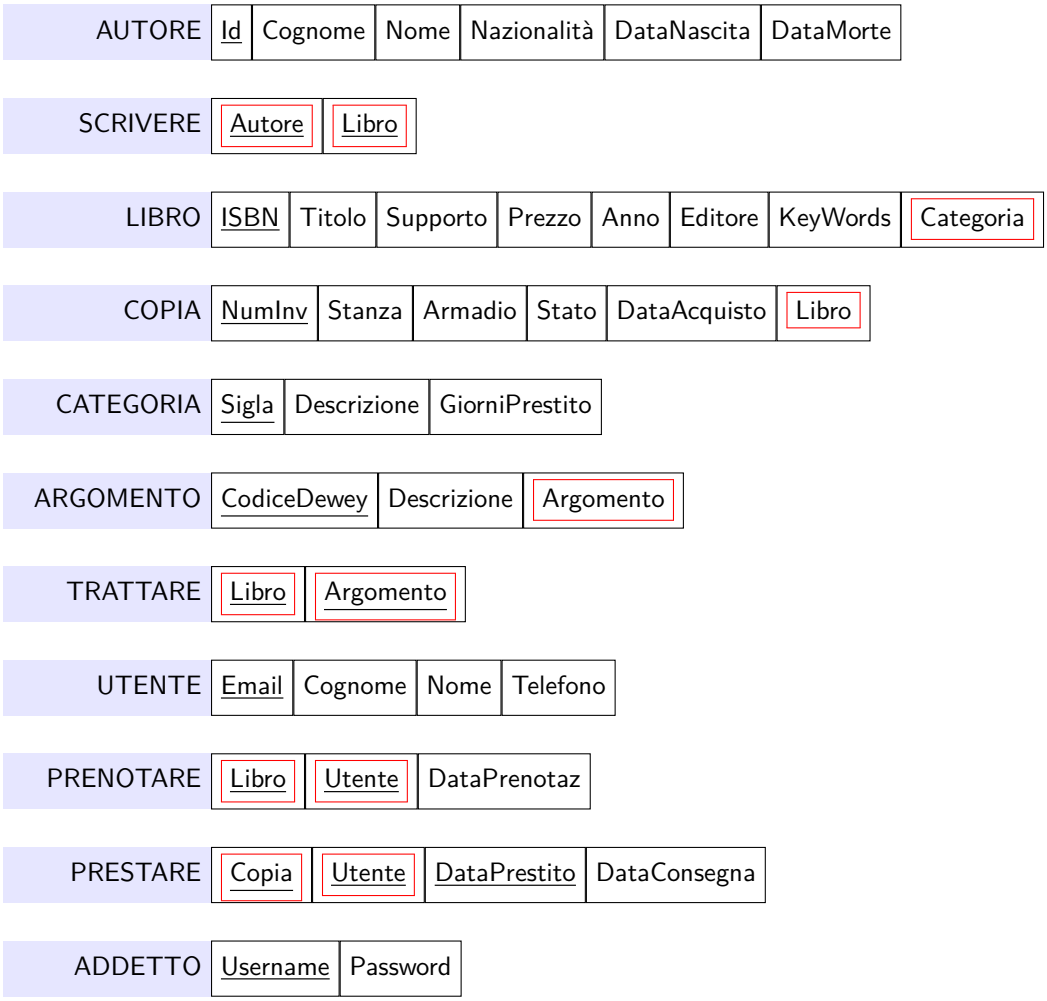
come insieme di dati che si possono trovare sul catalogo delle pubblicazioni di un dato editore dal concetto di "copia fisica di un libro presente in biblioteca". Ogni libro pubblicato è identificato univocamente dal suo codice ISBN. Gli utenti ammessi alla consultazione e prestiti dei libri sono identificati dalla loro email istituzionale; i responsabili della biblioteca, hanno una password specifica per accedere alle funzionalità di gestione della biblioteca.

A partire da queste osservazioni si arriva a definire il seguente schema concettuale:



Osservazione. Volendo mantenere uno "storico" dei prestiti, al fine di registrare chi ha preso in prestito nel passato una data copia, si possono intraprendere due strade: una (quella proposta nella soluzione sopra) in cui le entità $COPIA$ ed $UTENTE$ sono legate da un'associazione N:N; un'altra possibile soluzione consiste nel considerare questa associazione con cardinalità $N:1$ e, al recupero di un prestito di una copia, attivare un trigger che inserisce in un'apposita relazione $PRESTITO$ gli estremi del prestito appena rientrato, con riferimenti alla copia data in prestito ed all'utente che l'ha presa in prestito.

Utilizzando le regole di derivazione, il precedente schema concettuale viene tradotto nel seguente schema logico relazionale:



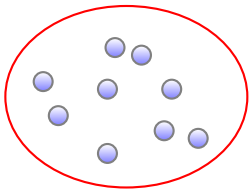
3.9 Associazioni per strutture organizzative

Gli oggetti di una classe vengono spesso organizzati in strutture organizzative. Questa strutturazione può essere descritta mediante un frammento di schema concettuale che si integra con il resto dello schema concettuale che descrive le associazioni degli elementi nello specifico contesto applicativo. Le strutture organizzative corrispondono ad alcune delle tradizionali strutture dei dati già viste (insiemi, sequenze, alberi, grafi, ...).

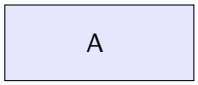
È bene mantenere separata la struttura logica degli elementi dalla loro struttura organizzativa; questa, negli esempi che seguono, verrà evidenziata in colore rosso.

3.9.1 Organizzazioni insiemistiche

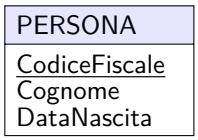
La forma più elementare di organizzazione degli elementi di un insieme A corrisponde al caso in cui non è definita alcuna organizzazione; la situazione è descritta, nella figura che segue, da un tradizionale diagramma di Venn in cui gli elementi dell'insieme sono descritti da pallini.



A livello concettuale questa situazione viene tradotta da una singola entità descritta con lo schema

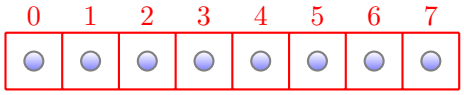


Esempio 3.9.1 - I dati di un insieme di persone può essere descritto mediante il seguente schema di entità con attributi:



3.9.2 Organizzazioni sequenziali

Una delle forme più elementari di organizzazione degli elementi di un insieme consiste nel metterli in sequenza, uno dopo l'altro; si tratta della tradizionale organizzazione ad *array*, descritta nella seguente figura:



Le posizioni sono individuate univocamente da un numero naturale (indice della posizione); lo schema concettuale che rappresenta la situazione è il seguente:

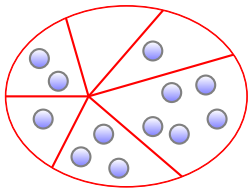
Si ammette anche il caso in cui l'associazione abbia cardinalità N:1 che indica che in una stessa posizione possono essere allocati più elementi (*bucket array*). L'opzionalità corrisponde alla situazione in cui alcune posizioni rimangano vuote.

Esempio 3.9.2 - Ad un torneo partecipano diverse squadre. La classifica finale può essere descritta mediante il seguente schema concettuale:

La cardinalità N dalla parte di $SQUADRA$ indica che si ammette che più squadre si classifichino a pari merito.

3.9.3 Partizione di un insieme

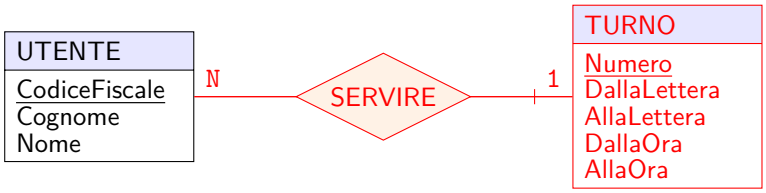
Consideriamo la situazione in cui un insieme A di elementi viene ripartito in classi; ciascuna classe viene individuata mediante un identificatore univoco; ciascun elemento appartiene ad una sola classe; le classi sono fra loro mutuamente disgiunte e l'unione delle classi costituisce l'intero insieme. La situazione è descritta nella seguente figura.



Questa situazione può essere rappresentata mediante la seguente porzione di schema concettuale; le parti in colore rosso denotano la struttura organizzativa corrispondente alla partizione degli elementi in sottoclassi disgiunte; la

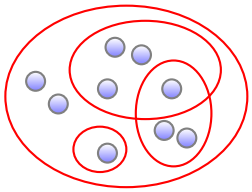
cardinalità totale N:1 indica che si tratta di una partizione, che una partizione può contenere più di un elemento e la parzialità denota che ci possono essere partizioni vuote. La situazione è descritta dal seguente schema concettuale:

Esempio 3.9.3 - Per accedere ad un servizio gli utenti, nel corso di una giornata, vengono suddivisi in turni, in base alla prima lettera del loro cognome. La situazione è descritta dal seguente schema concettuale:



3.9.4 Classificare in categorie indipendenti

Consideriamo la situazione in cui un insieme A di elementi vengono associati in delle categorie; una categoria può contenere più elementi ed un elemento può appartenere a più categorie.

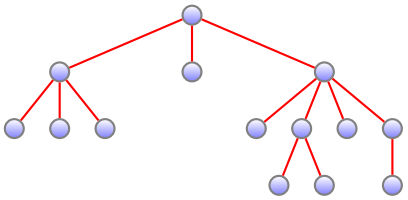


Questa situazione può essere descritta mediante la seguente porzione di schema concettuale; le parti in colore rosso denotano la struttura organizzativa corrispondente alla classificazione degli elementi in sottoclassi.

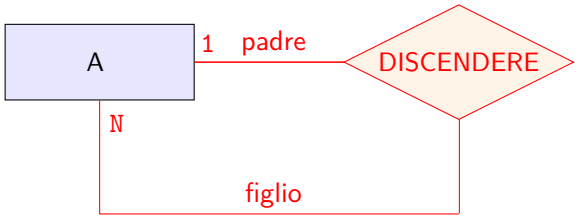
Esempio 3.9.4 - Un insieme di foto viene classificato in categorie, al fine di migliorare la ricerca.

3.9.5 Organizzazioni gerarchiche ad albero

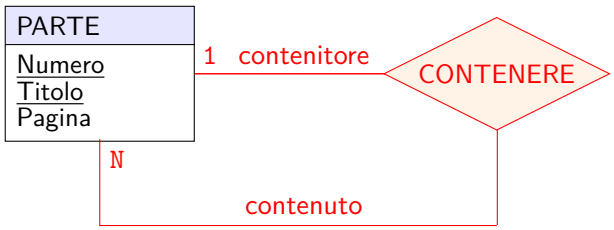
Consideriamo un insieme A di elementi organizzati in modo gerarchico ad albero, come descritto nella figura che segue.



Questa struttura gerarchica ad albero può essere descritta mediante il seguente frammento di schema concettuale; la parzialità indica che un elemento può non avere figli.

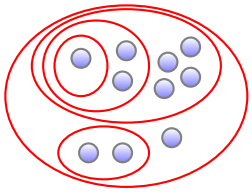


Esempio 3.9.5 - Un libro è strutturato in parti organizzate gerarchicamente: sezioni, capitoli e paragrafi; ciascuna di queste parti ha un titolo ed un numero progressivo che lo identifica univocamente all'interno dell'insieme dei nodi fratelli.

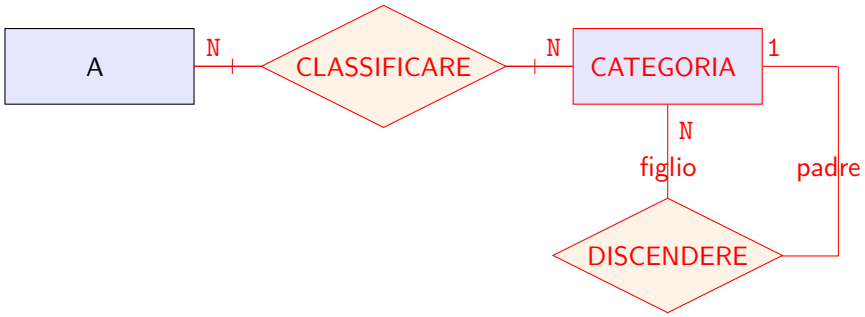


3.9.6 Organizzazioni in categorie gerachiche

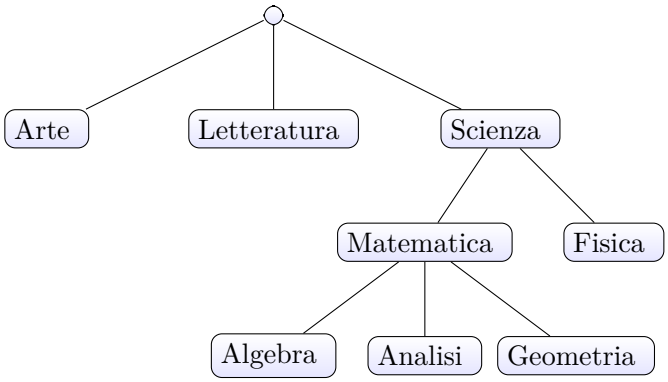
Consideriamo la situazione in cui un insieme A di elementi è organizzato in categorie e sottocategorie organizzate ad albero. La seguente figura descrive una possibile organizzazione.



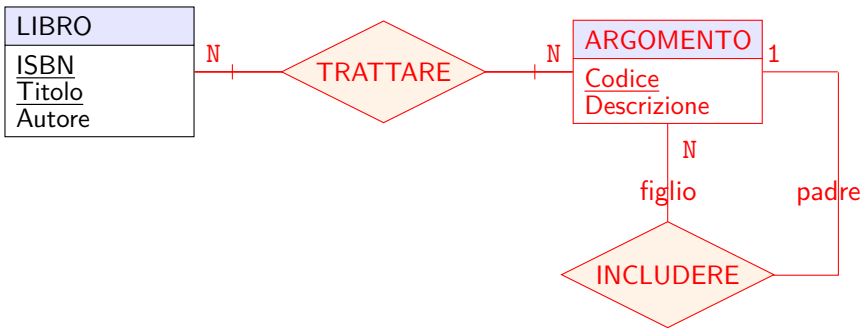
Un'organizzazione come questa può essere descritta mediante il seguente schema concettuale.



Esempio 3.9.6 - Consideriamo un insieme di libri che trattano di argomenti classificati in categorie e sottocategorie con una struttura gerarchica ad albero come la seguente:

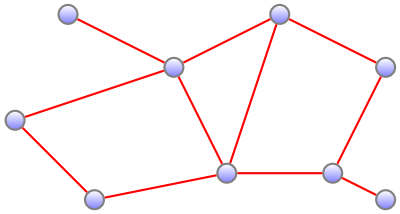


La classificazione dei libri può essere descritta mediante il seguente schema concettuale:

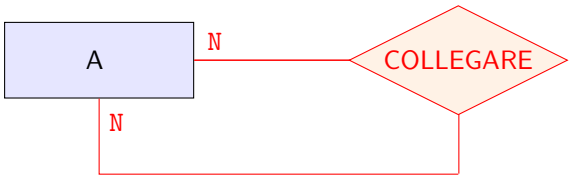


3.9.7 Organizzazioni a grafo

La figura che segue illustra una situazione in cui un insieme A di elementi è organizzato a grafo: ci sono dei *nodi* collegati fra loro mediante degli *archi*.

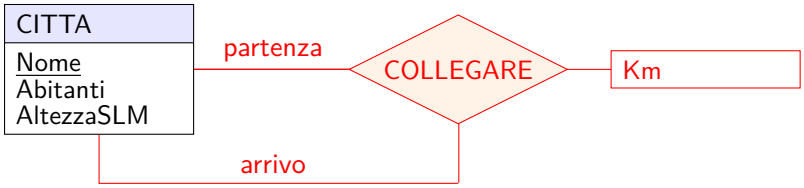


Questa forma di organizzazione può essere descritta mediante il seguente schema:



Nel caso in cui il grafo sia pesato, ossia ad ogni arco sia associata un'informazione è sufficiente aggiungere un attributo all'associazione *COLLEGARE*.

Esempio 3.9.7 - Consideriamo un insieme di città fra loro collegate da una rete di strade aventi ciascuna un dato kilometraggio. La situazione è descritta dal seguente schema:



3.10 Un esempio

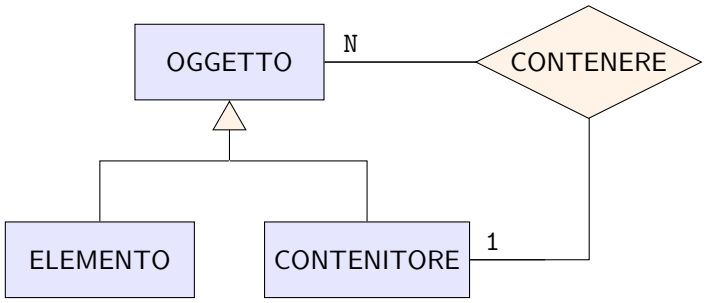
Applichiamo ora le precedenti situazioni di strutturazione degli elementi ad un caso specifico. Cercheremo comunque di costruire un 'modello' di situazione che possa interpolare più situazioni pratiche; d'altra parte utilizziamo una situazione guida che possa suggerirci delle linee guida per lo sviluppo del modello.

Un sistema di oggetti è così strutturato: ci sono due classi di oggetti: *contenitori* ed *elementi*. Un contenitore può contenere elementi ed altri contenitori; gli elementi sono entità elementari che non possono contenere altri oggetti. I contenitori sono organizzati secondo una struttura gerarchica ad albero, similmente alla struttura organizzativa ad albero dei file e cartelle di un file system di un sistema operativo; ed è questa la situazione particolare che ci farà da guida, ma vogliamo trovare una soluzione che possa adattarsi anche a casi di strutture analoghe; ad esempio per gestire gli utenti di una generica applicazione dove gli utenti vengono suddivisi in gruppi a loro volta eventualmente strutturati in sottogruppi.

Gli elementi informativi che bisogna rappresentare nello schema concettuale:

1. differenziazione dei ruoli fra contenitori ed elementi
2. struttura organizzativa gerarchica ad albero degli oggetti

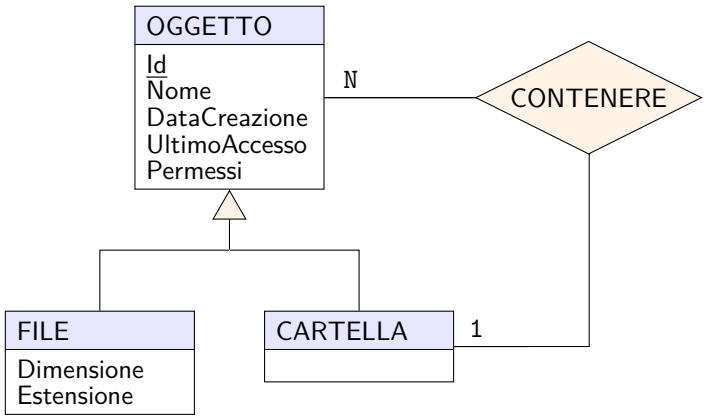
Il sistema di organizzazione degli oggetti è descritto mediante il seguente schema concettuale:



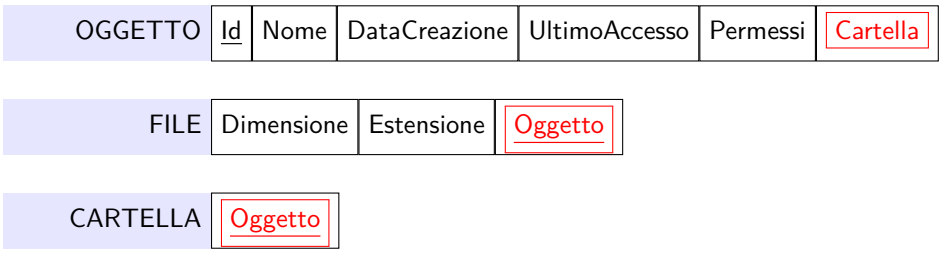
Il precedente schema concettuale può essere istanziato in diversi contesti:

1. - contenitori → cartelle
   - elemento → file
2. - contenitori → gruppo di utenti
   - elemento → singolo utente

Lo schema generale appena analizzato può essere declinato al caso di un file system di un Sistema Operativo dove le cartelle svolgono il ruolo di contenitori di altri oggetti (cartelle e file).



A seguire è riportata la traduzione del precedente schema concettuale in uno schema logico relazionale.

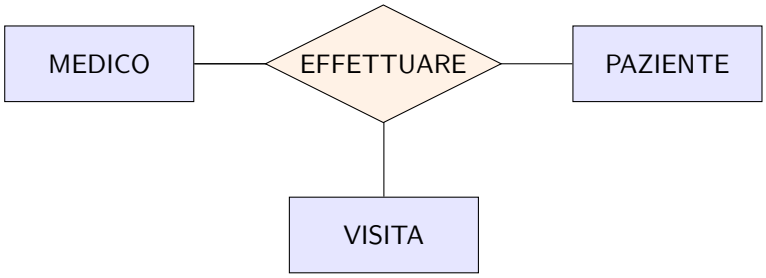


Poiché file e cartelle sono sotto entità mutuamente esclusive, nello schema logico relazionale sopra descritto si deve gestire il vincolo che i valori delle chiavi esterne `Oggetto` nelle due relazioni `FILE` e `CARTELLA` siano disgiunti. Una soluzione alternativa può essere ottenuta aggiungendo un attributo discriminante `Tipo` nell'entità padre `OGGETTO`, eliminando le due chiavi esterne `Oggetto` ed aggiungendo una chiave esterna nella relazione `OGGETTO` referente alla relazione `FILE`, al fine di associare gli attributi specifici del file.

ESERCIZI

3.1 Analizzare, nel contesto scolastico, l'entità *Verifica* corrispondente ad una *prova di verifica in una disciplina*. Definire gli attributi di questa entità.

3.2 Discutere comparativamente gli schemi concettuali che seguono: stabilire se sono equivalenti ossia se forniscono le stesse informazioni; in caso contrario stabilire quale dei due fornisce più informazioni. Suggerimento: scrivere alcune istanze di proposizioni compatibili con questi schemi e controllare se queste trovano corrispondenza negli altri schemi, senza perdere o aggiungere informazioni. Analizzare e discutere i diversi significati delle seguenti porzioni di schemi concettuali:

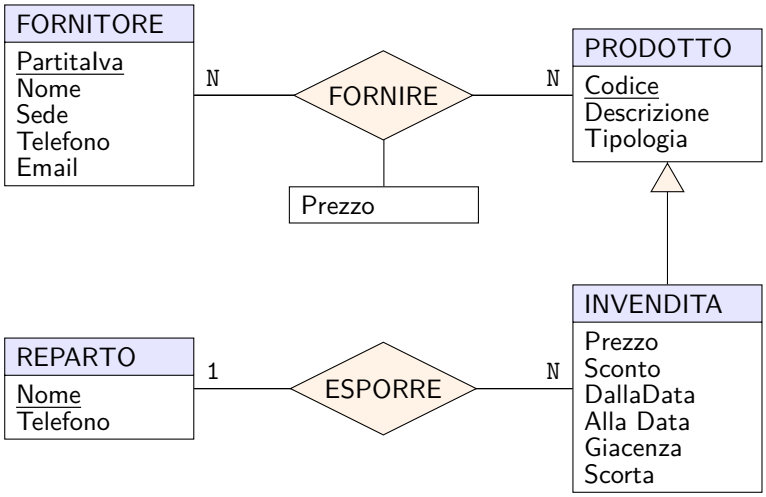


3.3 Proporre delle associazioni con cardinalità 1:1, 1:N e N:N nel contesto di un'attività di pratica di uno sport.

3.4 Modificare lo schema concettuale presentato nell'esempio 3.4.1 in modo da gestire i vari dipendenti di una banca che lavorano presso delle filiali a capo delle quali può esserci un dipendente avente il ruolo di direttore.

3.5 Con riferimento allo schema concettuale presentato nell'esempio 3.4.2, analizzare in che modo vengono tradotte nel modello logico relazionale le parzialità dell'associazione.

3.6 Analizzare il seguente schema concettuale, comparandolo allo schema presentato nell'esempio 3.6.1, evidenziandone gli aspetti critici.

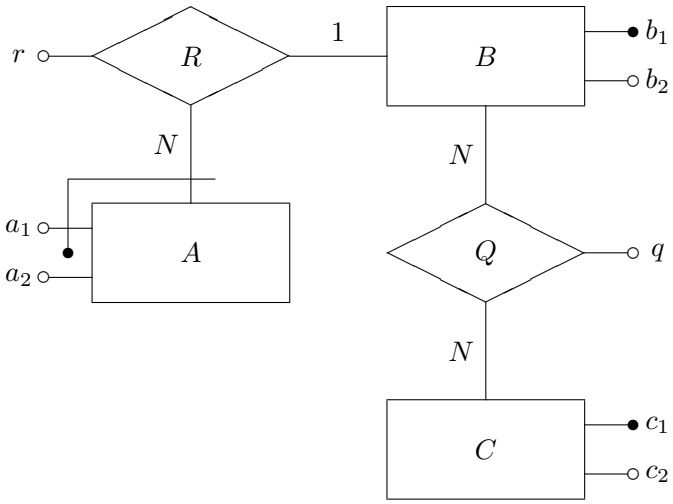


3.7 Modificare lo schema concettuale presentato nell'esempio 3.6.1 in modo da poter gestire delle campagne promozionali con sconti sui prezzi di vendita di alcuni prodotti in certi periodi oppure limitatamente ad alcuni lotti di acquisto.

3.8 Tradurre nel modello relazionale lo schema concettuale riportato nell'esempio 3.6.1.

3.9 Modificare lo schema concettuale presentato nell'esempio 3.6.2 in modo da poter gestire album singoli, doppi, tripli. Tradurre nel modello relazionale lo schema concettuale ottenuto.

3.10 Tradurre nel modello relazionale la seguente porzione di schema concettuale in notazione ER:



3.11 Un *social* funziona così: un utente chiede ad un altro utente di essere presentato e di poter entrare a fare parte al gruppo a cui egli appartiene. Se viene accettato, l'utente viene ammesso al gruppo. Serve conoscere la composizione attuale dei gruppi e chi ha presentato chi. Descrivere questa situazione mediante un diagramma concettuale.

3.12 Progettare una base di dati per gestire la segreteria dell'Istituto Superiore di Feltre, gli studenti iscritti, gli indirizzi di studio attivati e le classi afferenti a ciascun corso.

3.13 Con riferimento all'esempio 3.6.1 (magazzino):

1. adattare lo schema concettuale in modo da gestire le seguenti condizioni aggiuntive:
   (a) il prezzo dei prodotti acquistati dipende, oltre che dal fornitore e dal prodotto, dal numero di unità che vengono acquistate
   (b) i reparti vengono suddivisi in corsie che contengono scaffali organizzati in ripiani su cui vengono alloggiati i prodotti in vendita
   (c) per ogni reparto viene designato un responsabile
   (d) i prodotti vengono acquistati a lotti, ognuno dei quali ha una data di scadenza

   Soluzione:

   Ipotizziamo una organizzazione degli spazi di vendita in modo gerachico in reparti suddivisi in corsie che contengono scaffali disposti con ripiani in cui vengono alloggiati i prodotti in vendita. Gli attributi *Corsia*, *Scaffale* e *Ripiano* potrebbero confluire nell'entità *REPARTO*, che potrebbe allora essere ridenominato in *POSTAZIONE*, rinominando l'attributo *Nome* in *Reparto* e sostituendo la chiave primaria (*Reparto*, *Corsia*, *Scaffale*, *Ripiano*) con un codice univoco che individua un dato ripiano; si arriva quindi all'entità:

   Questo adattamento risulta debole nel caso in cui si volesse definire i "responsabili di reparto".
2. Tradurre lo schema concettuale in uno schema logico relazionale.
3. Applicare uno sconto del 10% a tutti i prodotti che scadono nei prossimi 7 giorni.
4. Produrre un depliant in automatico di tutti i prodotti che sono in sconto in un dato intervallo di data.

3.14 Con riferimento all'esempio 3.8.1 (biblioteca), risolvere le seguenti situazioni aggiuntive:

1. Valutare se la base di dati come sopra definita è in grado di supportare uno *storico* dei prestiti. In caso negativo, adattarla opportunamente, in modo da gestire uno storico dei prestiti (per sapere i lettori che hanno preso in prestito una data copia (in diversi periodi)).
2. adattare lo schema concettuale in modo da gestire le seguenti condizioni aggiuntive:
   (a) gestire le riviste; le riviste non hanno un proprio codice ISBN ma vengono identificate in base alla collana alla quale appartengono ed al numero progressivo nel corso dell'annata; si può costruire un'"impronta ISBN" dal nome della collana, numero ed anno
   (b) per gestire la possibilità che i lettori possano proporre dei testi da acquistare.
   (c) gestire le riviste; possono essere date in prestito, ma non l'ultimo numero uscito che deve essere consultato sul posto
   (d) i lettori possono suggerire l'acquisto di uno o più libri; viene proposto un form in cui l'utente indica gli estremi del libro da acquistare
3. Realizzare delle interrogazioni delle forme:
   - *interrogazioni individuali*:
     - di quali argomenti si è interessato un dato lettore?
     - quali/quanti lettori hanno letto un dato libro?
     - statistiche dei libri più richiesti - interrogazioni statistiche
   - *interrogazioni statistiche*:
     - quali sono i libri che trattano della Prima Guerra Mondiale?
     - quali sono i libri sul Manzoni disponibili?
     - nuovi acquisti del 2020
4. Tradurre lo schema concettuale nel modello relazionale.
5. Definire le tabelle in linguaggio SQL.
6. Realizzare dei trigger SQL per gestire l'invio automatico di email per il sollecito dei prestiti scaduti e per l'avviso di prenotazioni che si sono rese disponibili.

4

TEMI DI ESAME DI STATO

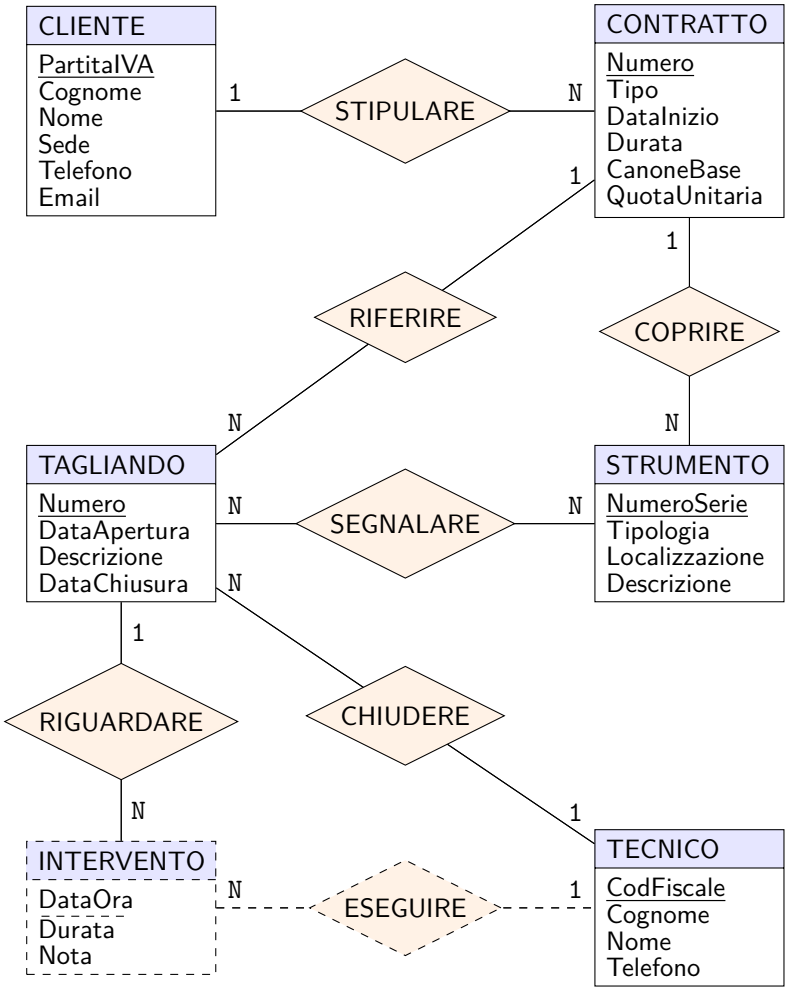
4.1 Ditta di assistenza strumentazioni informatiche

Una ditta informatica gestisce, mediante specifici contratti, un servizio di assistenza e manutenzione delle strumentazioni hardware (tablet, personal computer, stampanti, server, ...) per diversi clienti. Ciascun cliente può sottoscrivere diversi contratti e ciascun contratto può riguardare diverse strumentazioni. I contratti possono essere di due tipologie: *normale*, nel quale caso il cliente deve portare le strumentazioni presso la ditta di manutenzione, oppure *on-site* nel quale caso l'intervento viene eseguito direttamente presso la sede del cliente. I contratti possono avere una durata di 1, 2 o 3 anni. In caso di necessità i clienti, mediante una telefonata oppure accedendo ad un'area riservata del sito della ditta, "aprono" un tagliando di assistenza, indicando il numero del contratto di riferimento, una sintetica descrizione del problema e le strumentazioni, fra quelle coperte dal contratto, per le quali si chiede assistenza. Ciascun contratto comporta il pagamento di una quota base (dipendente dalla durata del contratto, dalla sua tipologia, dal numero di strumentazioni coperte da assistenza, dalla loro tipologia) e di una quota aggiuntiva ottenuta come prodotto di una quota unitaria (fissata nel contratto) moltiplicata per il numero di strumentazioni sulle quali sono stati aperti tagliandi di assistenza nel periodo del contratto. Ogni tecnico della ditta esegue interventi su strumentazioni coperte da assistenza e, al termine di ogni intervento, accedendo mediante browser o App ad un'apposita area riservata del sito della ditta, stila un rapportino di intervento, indicando la data, la durata oraria e il numero del tagliando per il quale ha operato. Per uno stesso tagliando di assistenza possono operare più tecnici ed ogni tecnico può eseguire più interventi. Un tagliando viene "chiuso" dal tecnico che esegue l'ultimo intervento.

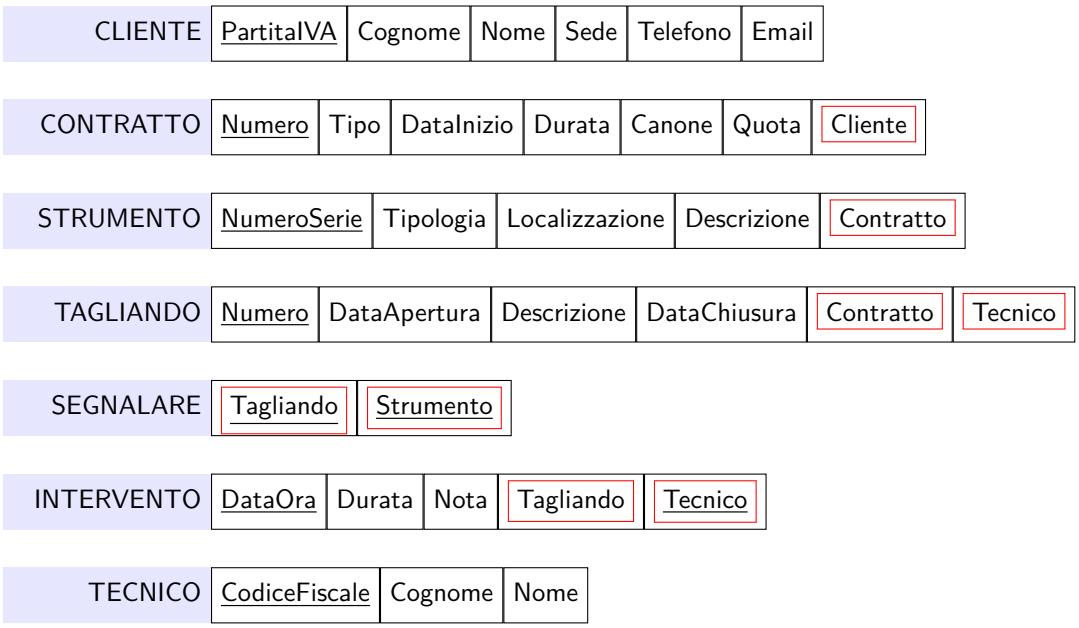
Con riferimento allo scenario sopra delineato, si definiscano eventuali ipotesi integrative che si ritengano opportune e si svolgano i seguenti punti:

1. Scrivere uno schema concettuale che rappresenti la situazione descritta.
2. Tradurre in uno schema logico relazionale lo schema concettuale descritto al punto 1. (o una sua porzione significativa).
3. Esprimere in linguaggio SQL le seguenti interrogazioni:
   (a) nominativi dei clienti che hanno stipulato contratti di assistenza
   (b) tagliandi aperti ed estremi del cliente che lo ha aperto
   (c) numero complessivo di ore impiegate negli interventi relativi a ciascun contratto
4. Proporre il progetto della homepage del sito e della pagina dell'interfaccia Web mediante la quale i tecnici compilano il report al termine di ciascun intervento.
5. Codificare in un linguaggio a scelta un segmento significativo dell'applicazione Web che permetta l'esecuzione di una delle interrogazioni indicate al punto 3. e visualizzi i corrispondenti risultati.

Soluzione. Schema concettuale:



Schema logico relazionale:



Query SQL - caso (a):

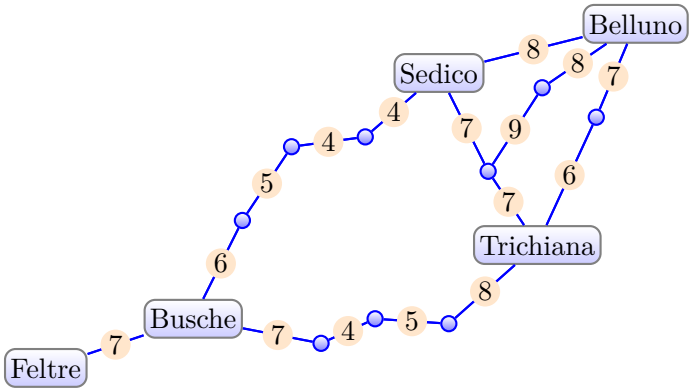
```
SELECT Cognome, Nome
FROM CLIENTE JOIN CONTRATTO ON PartitaIVA=Cliente;
```

Query SQL - caso (b):

```
SELECT Numero, Descrizione, Cognome, Nome, Telefono
FROM TAGLIANDO JOIN CONTRATTO ON Contratto=Numero
               JOIN CLIENTE ON Cliente=PartitaIVA
WHERE DataChiusura IS NULL;
```

4.2 Una base di dati per un navigatore stradale

Si vuole realizzare una base di dati per un navigatore stradale. Gli elementi geometrici da memorizzare sono richiamati nella seguente figura dove sono riportati i nomi delle località, i punti di passaggio delle strade ed i pesi lungo i tratti di percoso denotano la lunghezza in chilometri del tratto di strada.



La base di dati deve essere finalizzata alla realizzazione delle classiche funzionalità di un navigatore stradale: caricamento di mappe stradali, selezionare le località di partenza e di arrivo, avere indicazioni sul percorso da seguire.

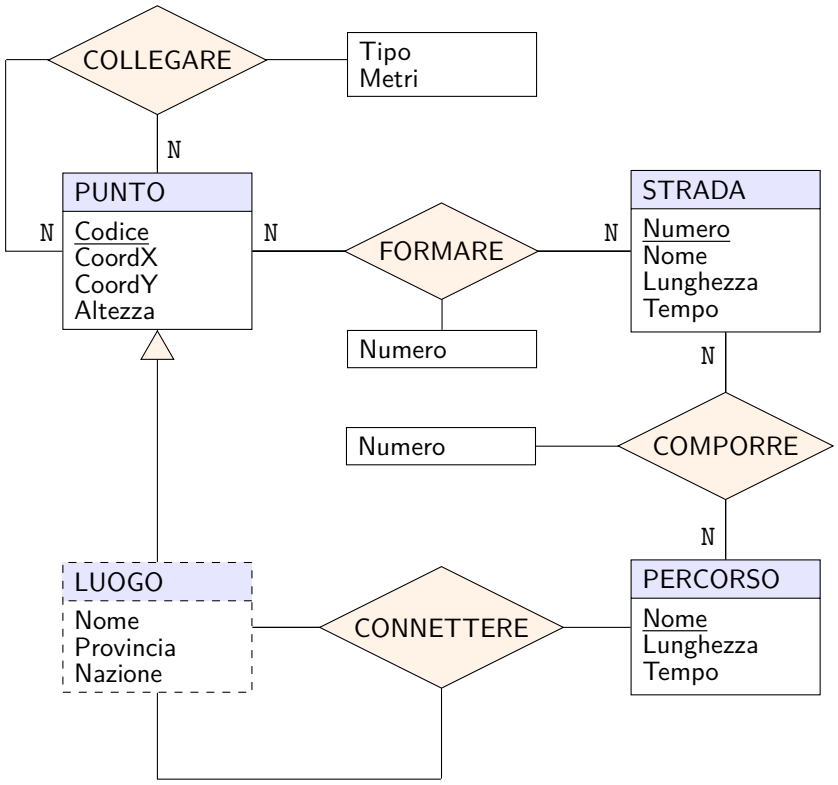
Soluzione. Dizionario dei termini:

- *Punto*: punto geometrico del piano, dotato di coordinate piane (rispetto ad un opportuno sistema di riferimento) ed inoltre di una terza coordinata che individua l'altezza sul livello del mare; definiscono la geometria dei percorsi
- *Luogo*: località geografica individuata da un nome proprio; potrebbe essere il nome di una città (ad esempio *Belluno*) oppure una località più dettagliata (ad esempio *Belluno, via Garibaldi 15*).
- *Strada*: sequenza di punti costituenti un arco del grafo avente per nodi le località o punti in cui si incrociano più strade. La lunghezza di una strada può essere calcolata e memorizzata nella base dati in modo da velocizzare la determinazione dei percorsi richiesti dall'utente
- *Percorso*: sequenza contigua di strade che congiungono due località. La lunghezza dei percorsi, data la loro enorme numerosità, viene calcolata (basandosi sulla lunghezza dei tratti di strada componenti il percorso) solo al momento in cui vengono richiesti dall'utente

Osservazione: si possono evidenziare i seguenti livelli di persistenza dei dati della base di dati:

- dati caricati dal'esterno, al momento del caricamento della mappa (punti e luoghi)
- dati calcolati in una fase di pre-processing (strade e loro lunghezze)
- dati calcolati solo al momento dell'interrogazione (percorsi)

Schema concettuale:



APPENDICE A

RELATIONAL ALGEBRA CALCULATOR

```
T ← σ(|Descrizione='produzione'|) REPARTO
T ← T ⋈(|Sigla=Reparto|) DIPENDENTE
T ← π(|Cognome,Nome|) T
```

Esempio di programma per RAC.

Il programma RAC (R*elational* A*lgebra* C*alculator*) costituisce un ambiente per la valutazione di espressioni di algebra relazionale. Permette solamente di interrogare la base di dati e non offre alcuna funzionalità di modifica della base di dati che deve essere creata, definita e popolata 'a mano' scrivendo mediante un editor di testo i file .csv presenti nelle cartelle di ciascun database. Si possono interrogare più database.

Il programma è in fase di sviluppo; pertanto alcune funzionalità sono incomplete ed alcune possono causare delle anomalie in fase di esecuzione. Quindi, salvare il programma prima di mandare in esecuzione!

A.1 Architettura del programma

Il programma `RAC` è organizzato secondo un'architettura client-server:

- il *server* costituisce il motore del sistema: riceve le istruzioni dal client, le esegue e comunica il risultato al client
- il *client* costituisce l'interfaccia grafica mediante la quale l'utente scrive il programma da eseguire, lo manda in esecuzione e visualizza i risultati ottenuti dal server

A.2 Installazione

Il programma viene installato copiando la cartella contenente il programma ed i file di supporto ad esso associati. Questa cartella è completa ed autosufficiente e non richiede alcuna installazione di altri componenti.

La cartella `RAC_versione` contenente il programma `RAC` contiene i seguenti file e sottocartelle:

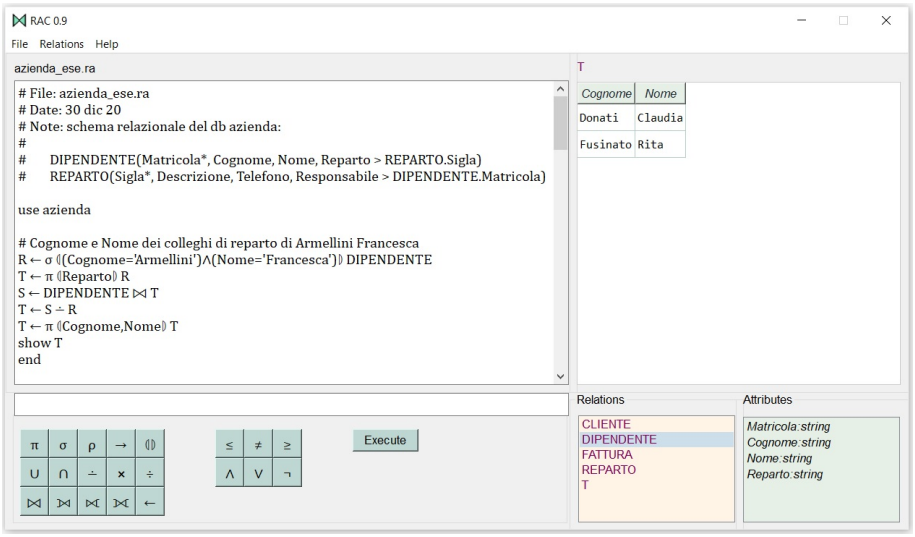
- `RAC_server.BAT`: file di comandi per attivare l'esecuzione del server
- `RAC_client.BAT`: file di comandi per attivare l'esecuzione del client
- `server`: cartella contenente il programma server
  - `raeng`: cartella contenente il programma server
  - `db`: cartella contenente il database (i file delle tabelle del db (in formato `.csv`))
- `client`: cartella contenente il programma client
  - `raui`: cartella contenente il programma client
  - `prog`: cartella contenente i programmi costituiti da sequenze di istruzioni dell'algebra relazionale (in formato `.ra`)
  - `qt5lib`: cartella contenente le librerie grafiche *Qt5*

A.3 Attivazione

L'attivazione del programma `RAC` avviene eseguendo con doppio clic il file di comandi `RAC_server.BAT` e successivamente il file di comandi `RAC_client.BAT` (oppure eseguendo prima il programma server `raeng` (presente nella cartella `server\raeng`) e poi il programma client `raui` (nella cartella `client\raui`)). All'attivazione il server legge il file testo `init_rac.cfg` dove è definito il percorso (relativo) della cartella contenenti le basi di dati e genera dinamicamente il catalogo costituito dalle relazioni che registrano i dati che definiscono le basi di dati presenti; queste relazioni costituenti il catalogo possono essere consultate per conoscere i nomi delle basi di dati esistenti ed i nomi delle loro tabelle e dei corrispondenti attributi.

A.4 Interfaccia

La figura che segue illustra l'interfaccia grafica dell'ambiente che viene attivato.



La finestra principale è costituita da quattro sotto-finestre e da una tastiera virtuale:

- finestra del programma: posta in alto a sinistra, consente di scrivere il programma costituito da una sequenza di comandi e di espressioni; sono utilizzabili le usuali funzionalità taglia-copia-incolla di un editor di testo (con le solite combinazioni di tasti *Ctrl-C*, *Ctrl-X*, *Ctrl-V*); attraverso la tradizionale voce di menu *File* si possono caricare e salvare da/su disco i programmi scritti; il linguaggio è case-sensitive; i programmi vengono salvati in file con estensione `.ra`
- finestra di visualizzazione: posta in alto a destra, visualizza il contenuto di una relazione; con un clic su un attributo della relazione visualizzata, le righe vengono ordinate in base ai valori della colonna selezionata
- finestra dei nomi delle relazioni: posta in basso a destra, mostra l'elenco delle relazioni disponibili presso il client
- finestra dei nomi degli attributi: posta in basso a destra, mostra i nomi ed i domini degli attributi delle relazioni disponibili presso il client; con un clic sul nome della relazione si evidenziano gli attributi della relazione (sulla corrispondente finestra a lato); con un doppio clic, si ottiene in più la visualizzazione (sulla finestra in alto a destra) i dati della relazione selezionata
- bottoni delle operazioni: 21 bottoni che costituiscono una tastiera virtuale per scrivere le operazioni dell'algebra relazionale
- bottone di esecuzione (bottone Execute): esegue in sequenza i comandi e le istruzioni che costituiscono il programma presente nella finestra di edit

A.5 Comandi generali

A seguire sono elencati i principali comandi generali del programma RAC:

- comando `use`: imposta il nome del database di lavoro (il nome della cartella contenente i file che memorizzano le relazioni):

  `use` *nomedatabase*

- comando `show`: visualizza il contenuto di una tabella:

  `show` *nometabella*

  Il comando `show` permette di indagare il catalogo (dizionario) delle basi di dati e di conoscere i nomi dei database attualmente presenti (`show DATABASES`), i nomi delle tabelle presenti (`show TABLES`) ed i nomi degli attributi di ciascuna tabella (`show COLUMNS`)

- il comando `end` termina l'esecuzione del programma ed eventuali istruzioni che seguono non vengono prese in considerazione

A.6 Espressioni dell'algebra relazionale

Nella finestra principale del programma vengono scritti, oltre ad eventuali comandi generali (elencati al paragrafo precedente), viene scritto il programma costituito da assegnazioni dell'algebra relazionale aventi la seguente struttura:

$$idRel \leftarrow expAR$$

avente il seguente effetto: viene valutata l'espressione $expAR$; il risultato, costituito da una relazione, viene assegnato all'identificatore $idRel$; tale identificatore potrà essere utilizzato in altre espressioni che seguono.

Esempio A.6.1 - Esempio di programma:

```
#-- dipendenti che lavorano nel reparto produzione --
use azienda
T ← σ(Descrizione='produzione') REPARTO
T ← T ⋈ (Sigla=Reparto) DIPENDENTE
T ← π(Cognome,Nome) T
show T
```

Esempio A.6.2 - E' possibile gestire i valori nulli secondo la teoria prevista dal modello relazionale. Nelle espressioni il valore nullo viene denotato mediante ls stringa `NULL`, mentre nella visualizzazione a video il valore nullo appare come una cella vuota. La seguente sequenza di assegnazioni genera la relazione `T` contenente la descrizione dei reparti senza responsabile.

```
#-- reparti senza responsabile --
T ← σ(Responsabile=NULL) REPARTO
T ← π(Descrizione) T
```

A.7 Gli operatori dell'algebra relazionale

A seguire è riportato l'elenco delle operazioni dell'algebra relazionale:

- σ : selezione
- π : proiezione
- ρ : ridenominazione
- $\times$: prodotto cartesiano
- $\div$: divisione
- $\cup$: unione
- $\cap$: intersezione
- $\doteq$: differenza
- $\bowtie$: congiunzione
- $⟕$: congiunzione esterna sinistra
- $⟖$: congiunzione esterna destra
- $⟗$: congiunzione esterna completa
- $\leftarrow$: assegnazione di una relazione
- $\rightarrow$: ridenominazione di un attributo

A.8 Il dizionario della base di dati

La figura che segue descrive lo schema logico relazionale della base di dati.