

Indice

1	Progettazione Concettuale	1
1.1	I modelli concettuali dei dati	3
1.2	Il Modello Entity Relationship (E/R)	10
1.3	Metodologia per il progetto concettuale	20
1.3.1	Primitive per il progetto concettuale	21
1.3.2	Strategie per il progetto concettuale	30
1.4	Progettazione da requisiti in linguaggio naturale	34
1.5	Progettazione da file esistenti	40
2	Elementi di Teoria Relazionale	49
2.1	Modello Relazionale	51
2.2	Algebra Relazionale	57
3	Progettazione Logica	67
3.1	Progetto logico di alto livello con il modello E/R	69
3.2	Progettazione logica relazionale	81
3.2.1	Trasformazione di attributi e identificatori	81
3.2.2	Traduzione di entità e associazioni	85
4	Elementi ed esempi del linguaggio SQL	93
4.1	Definizione dei dati	96
4.2	Interrogazioni	105
4.3	Manipolazione dei dati	127
4.4	Viste, privilegi e cataloghi	129
5	Teoria della Normalizzazione	137
5.1	Dipendenze Funzionali	139
5.2	Ragionamento sulle dipendenze funzionali	141
5.3	Forme Normali	146
5.4	Decomposizione di schemi	150
5.5	Normalizzazione di uno schema relazionale	155

6	Esercizi	161
6.1	E/R e Progetto Logico	163
6.1.1	Soluzioni	173
6.1.2	Dati Derivati	196
6.1.3	Soluzioni	199
6.2	SQL e Algebra Relazionale	205
6.2.1	Soluzioni	213
6.3	Normalizzazione	234
6.3.1	Soluzioni	239
6.4	SQL e linguaggi di programmazione	247

Prefazione

L'obiettivo del volume è fornire al lettore le nozioni fondamentali di progettazione e di realizzazione di applicazioni di basi di dati relazionali.

Relativamente alla progettazione, vengono trattate le fasi di progettazione concettuale e logica e vengono presentati i modelli dei dati Entity-Relationship e Relazionale che costituiscono gli strumenti di base, rispettivamente, per la progettazione concettuale e la progettazione logica.

Viene inoltre introdotto lo studente alla teoria della normalizzazione di basi di dati relazionali.

Relativamente alla realizzazione, vengono presentati elementi ed esempi del linguaggio standard per RDBMS (Relational Database Management Systems) SQL. Ampio spazio è dedicato ad esercizi svolti sui temi trattati.

Il volume nasce dalla pluriennale esperienza didattica condotta dagli autori nel corso di Basi di Dati per studenti del corso di laurea e del corso di diploma universitario in Ingegneria Informatica dell'Università di Modena. Il materiale originale era costituito da una collezione di lucidi per lavagna luminosa, pensati come ausilio all'esposizione dei concetti. Tali lucidi sono stati via via arricchiti fino ad assumere la forma di dispense, su richiesta pressante degli studenti. Inoltre, sono stati integrati da esercitazioni svolte in aula e dalla collezione dei temi d'esame, assieme alla loro soluzione.

Il volume attuale costituisce più una *collezione di appunti* che un vero *libro* nel senso che tratta in modo *rigoroso* ma *essenziale* i concetti forniti. Inoltre, non esaurisce tutte le tematiche di un corso di Basi di Dati, la cui altra componente fondamentale è costituita dalla *tecnologia delle basi di dati*. Questa componente è, a parere degli autori, trattata in maniera eccellente da un altro testo di Basi di Dati (Lezioni di Basi di Dati [?]), scritto dai nostri colleghi e amici Paolo Ciaccia e Dario Maio dell'Università di Bologna.

Il volume, pure nella sua essenzialità, è ricco di esercizi svolti e quindi può costituire un ottimo strumento per gruppi di lavoro che, nell'ambito di software house, si occupino di progettazione di applicazioni di basi di dati relazionali.

Il libro si articola in 6 capitoli:

- *Progettazione Concettuale*. Vengono presentati i modelli concettuali dei dati, i loro meccanismi di astrazione fondamentali ed il modello Entity-Relationship. Vengono inoltre illustrate le metodologie per il progetto concettuale e l'attività di progettazione concettuale da requisiti in linguaggio naturale e da file esistenti. Questo capitolo riassume concetti fondamentali tratti da un testo in lingua inglese molto approfondito "Conceptual Database Design: an Entity-Relationship approach" di Batini, Ceri, Navathe [?].
- *Elementi di Teoria Relazionale*. Viene presentato il modello Relazionale dei

dati e l'Algebra Relazionale.

- *Progettazione Logica.* Vengono presentate le due fasi fondamentali di progettazione logica: di alto livello e relazionale. Anche questo capitolo riassume concetti fondamentali tratti da [?].
- *Teoria della normalizzazione.* Vengono presentati gli elementi fondamentali della teoria della normalizzazione, il cui obiettivo è quello di produrre uno schema di basi di dati con proprietà di qualità. Per una trattazione più approfondita si rimanda alla sezione 7 del testo [?].
- *Elementi ed Esempi del Linguaggio SQL.* Vengono presentati gli elementi fondamentali del linguaggio standard per basi di dati relazionali *SQL*: definizione dei dati, interrogazioni, manipolazione dei dati, Viste, Privilegi e Cataloghi.
- *Esercizi.* Vengono presentati numerosi esercizi svolti di progettazione concettuale e logica, di algebra relazionale e *SQL*, di normalizzazione di schemi relazionali.

Un ringraziamento speciale va rivolto ai nostri studenti che hanno sopportato per anni dispense parziali ed incoerenti contribuendo spesso al loro miglioramento.

Capitolo 1

Progettazione Concettuale

L'obiettivo della *progettazione concettuale* è quello di rappresentare la realtà di interesse ad un alto livello di astrazione. In questo capitolo, dopo una breve introduzione dei principali meccanismi di astrazione dei modelli concettuali dei dati, verrà presentato il modello Entity-Relationship (E/R), concepito da Chen nel 1976 e successivamente esteso con altre primitive di rappresentazione, che costituisce lo standard *de facto* per la progettazione concettuale delle applicazioni per basi di dati.

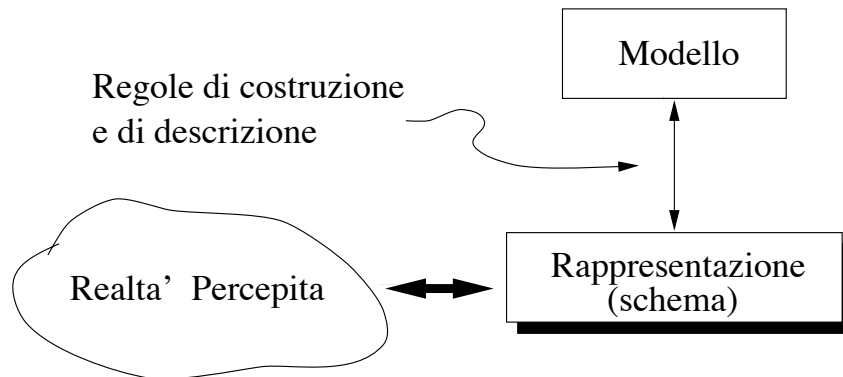
Nella prima parte del capitolo verranno illustrati i concetti fondamentali del modello E/R con la relativa rappresentazione grafica.

La seconda parte del capitolo introduce una metodologia per il progetto concettuale, che definisce la costruzione di uno schema E/R come un processo incrementale: la nostra percezione della realtà è progressivamente raffinata e arricchita e lo schema concettuale è formalmente sviluppato.

Nell'ultima parte del capitolo vengono trattati due casi particolarmente rilevanti di progettazione concettuale: progettazione da requisiti in linguaggio naturale e progettazione da file esistenti.

1.1 I modelli concettuali dei dati

- ◇ **modello**: l'insieme di regole e strutture che permettono la rappresentazione della realtà di interesse
- ◇ **schema**: la rappresentazione di una specifica realtà secondo un determinato modello



- ◇ il modello fornisce le regole per la costruzione della rappresentazione
- ◇ la rappresentazione è data da un insieme di simboli posti in corrispondenza con la realtà di interesse
- ◇ la realtà di interesse è una porzione di mondo reale così come è percepita da chi costruisce la rappresentazione

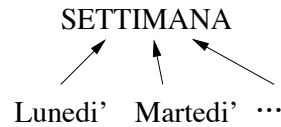
Astrazione

- ◇ è un procedimento mentale che si adotta quando si evidenziano alcune proprietà e caratteristiche di un insieme di oggetti
- ◇ altre proprietà, giudicate non rilevanti, vengono trascurate
- ◇ primitive di astrazione comuni a tutti i modelli:
 - **classificazione**
 - **aggregazione**
 - **generalizzazione**

Primitive di astrazione

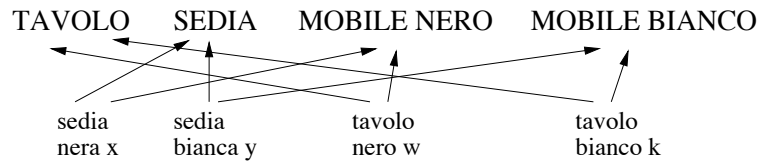
Classificazione: conduce alla definizione di una *classe* a partire da un insieme di oggetti caratterizzati da proprietà comuni.

◇ ESEMPIO: i membri della Classe SETTIMANA sono Lunedì, ..., Giovedì, ... Domenica.



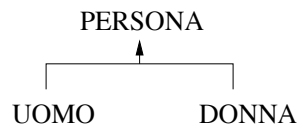
◇ relazione MEMBER_OF

- esiste tra un elemento della classe e la classe stessa
Giovedì è MEMBER_OF SETTIMANA
- gli stessi oggetti possono essere classificati in modi diversi



Generalizzazione: definisce una relazione di sovrainsieme tra una classe padre e altre classi figlie (sottoclassi).

◇ ESEMPIO: la classe PERSONA è una generalizzazione delle classi UOMO e DONNA



◇ la generalizzazione stabilisce una corrispondenza tra gli elementi di una sottoclasse e gli elementi della superclasse

GENERALIZZAZIONE

◇ Stabilisce un mapping tra gli elementi di una classe e gli elementi delle classi generalizzate. Per tale mapping si definiscono le proprietà di copertura:

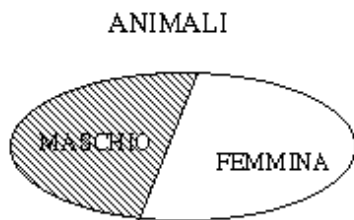
◇ **Copertura totale o parziale**

- **totale (t):** ogni elemento della classe generica è in relazione con almeno un elemento delle classi generalizzate;
- **parziale (p):** esistono alcuni elementi della classe generica che non sono in relazione con alcun elemento delle classi generalizzate;

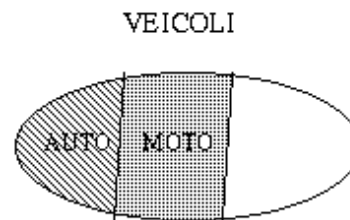
◇ **Copertura esclusiva o sovrapposta**

- **esclusiva (e):** ogni elemento della classe generica è in relazione con al massimo un elemento delle classi generalizzate;
- **sovrapposta (o):** esistono alcuni elementi della classe generica che sono in relazione con elementi di due o più classi generalizzate.

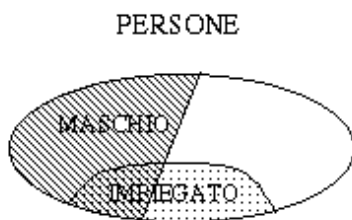
◇ **Esempi**



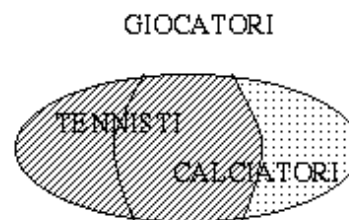
totale, esclusiva



parziale, esclusiva



parziale, sovrapposta



totale, sovrapposta

AGGREGAZIONE

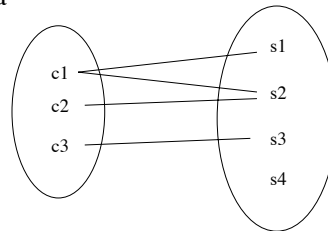
Aggregazione: permette di giungere alla definizione di un concetto a partire da altri concetti che ne rappresentano le parti componenti.

- ◇ ESEMPIO: il concetto BICICLETTA è la classe i cui componenti sono RUOTA, PEDALE e MANUBRIO
- ◇ relazione **PART-OF** : esiste tra un concetto componente ed il concetto composto; ad esempio, PEDALE è PART-OF BICICLETTA
- ◇ l'aggregazione stabilisce una corrispondenza (mapping) tra gli elementi di una classe componente e gli elementi della classe composta

Aggregazione binaria: : è una corrispondenza stabilita tra due classi.

- ◇ ESEMPIO: ESAME è una aggregazione binaria delle classi CORSO e STUDENTE, e stabilisce una corrispondenza tra le due classi

Rappresentazione grafica



CARDINALITÀ della partecipazione alla corrispondenza

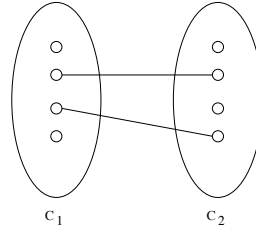
- ◇ sia A aggregazione di C_1 e C_2
 - *cardinalità minima di C_1 in A* : $\text{min-card}(C_1, A)$
è il minimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C_1 deve partecipare
 - *cardinalità massima di C_1 in A* : $\text{max-card}(C_1, A)$
è il massimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C_1 può partecipare
 - *partecipazione opzionale* : $\text{min-card}(C_1, A) = 0$
alcuni elementi di C_1 possono non essere aggregati tramite A a elementi di C_2
 - *partecipazione obbligatoria* : $\text{min-card}(C_1, A) > 0$
ad ogni elemento di C_1 deve essere aggregato, tramite A, almeno un elemento di C_2

Cardinalità della corrispondenza

◇ *Uno a uno (One-to-one)*

$$\max\text{-card}(C_1, A) = 1$$

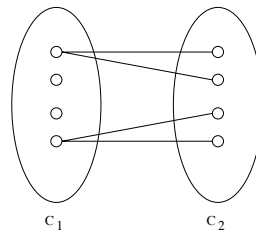
$$\max\text{-card}(C_2, A) = 1$$



◇ *Uno a molti (One-to-many)*

$$\max\text{-card}(C_1, A) = n$$

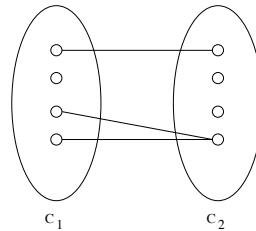
$$\max\text{-card}(C_2, A) = 1$$



◇ *Molti a uno (simmetrica) (Many-to-one)*

$$\max\text{-card}(C_1, A) = 1$$

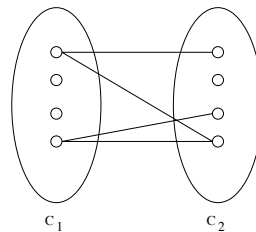
$$\max\text{-card}(C_2, A) = n$$



◇ *Molti a molti (Many-to-many)*

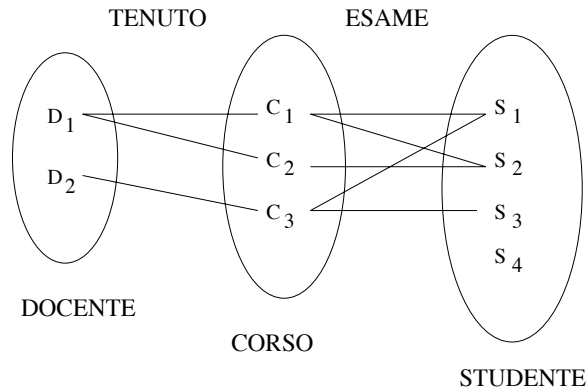
$$\max\text{-card}(C_1, A) = n$$

$$\max\text{-card}(C_2, A) = n$$



NB: $\max\text{-card}(C, A) = n$ significa che il massimo numero di partecipazioni di un elemento di C_1 in A è limitato soltanto dal numero di elementi in C_2 .

Esempi di aggregazione binaria



- per ogni corso c'è stato almeno un esame:
 $\min\text{-card}(\text{CORSO}, \text{ESAME}) = 1$
 - per ogni corso c'è un numero generico di esame:
 $\max\text{-card}(\text{CORSO}, \text{ESAME}) = n$
 - alcuni studenti non hanno sostenuto esami:
 $\min\text{-card}(\text{STUDENTE}, \text{ESAME}) = 0$
 - uno studente può sostenere più esami:
 $\max\text{-card}(\text{STUDENTE}, \text{ESAME}) = n$
-
- un corso è tenuto da uno ed un solo docente:
 $\min\text{-card}(\text{CORSO}, \text{TENUTO}) = 1$
 $\max\text{-card}(\text{CORSO}, \text{TENUTO}) = 1$
 - un docente tiene almeno un corso:
 $\min\text{-card}(\text{DOCENTE}, \text{TENUTO}) = 1$
 - un docente può tenere un generico numero di corsi:
 $\max\text{-card}(\text{DOCENTE}, \text{TENUTO}) = n$
-
- $\text{card}(\text{CORSO}, \text{ESAME}) = (1, n)$
 - $\text{card}(\text{STUDENTE}, \text{ESAME}) = (0, n)$
 - $\text{card}(\text{CORSO}, \text{TENUTO}) = (1, 1)$
 - $\text{card}(\text{DOCENTE}, \text{TENUTO}) = (1, n)$
- ◇ Si possono avere due o più aggregazioni tra la stessa coppia di concetti
 Ad esempio, tra i concetti **CORSO** e **STUDENTE** si può stabilire l'ulteriore aggregazione **SEGUE**

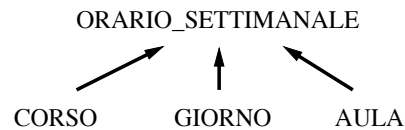
Aggregazione n-aria

◇ È stabilita tra tre o più classi C_1, C_2, \dots, C_n

- *cardinalità minima di C_i in A* : $\text{min-card}(C_i, A)$
 è il minimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C_i deve partecipare
- *cardinalità massima di C_i in A* : $\text{max-card}(C_i, A)$
 è il massimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C_i può partecipare

◇ **Esempio:**

ORARIO_SETTIMANALE è una aggregazione ternaria delle classi CORSO, GIORNO e AULA



- ogni corso può tenersi da 1 a 5 volte la settimana:
 $\text{min-card}(\text{CORSO}, \text{ORARIO_SETTIMANALE}) = 1$
 $\text{max-card}(\text{CORSO}, \text{ORARIO_SETTIMANALE}) = 5$
- in ogni giorno della settimana si può tenere un qualsiasi numero di lezioni:
 $\text{min-card}(\text{GIORNO}, \text{ORARIO_SETTIMANALE}) = 0$
 $\text{max-card}(\text{GIORNO}, \text{ORARIO_SETTIMANALE}) = n$
- ogni aula ospita al massimo 40 lezioni per settimana:
 $\text{min-card}(\text{AULA}, \text{ORARIO_SETTIMANALE}) = 0$
 $\text{max-card}(\text{AULA}, \text{ORARIO_SETTIMANALE}) = 40$
- $\text{card}(\text{CORSO}, \text{ORARIO_SETTIMANALE}) = (1, 5)$
- $\text{card}(\text{GIORNO}, \text{ORARIO_SETTIMANALE}) = (0, n)$
- $\text{card}(\text{AULA}, \text{ORARIO_SETTIMANALE}) = (0, 40)$

1.2 Il Modello Entity Relationship (E/R)

È stato sviluppato da Chen nel 1976 (P.P. Chen, “The Entity-Relationship Model: Toward a Unified View of Data”, ACM Transactions on Database Systems 1, no. 1, 9-37, March 1976).

È stato poi esteso nell’ambito della metodologia di progettazione di basi di dati DATAID sviluppata nel progetto italiano omonimo (S. Ceri, “Methodology and Tools for Data Base Design”, North-Holland, 1983)

Elementi di base

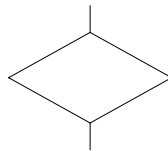
ENTITÀ: Una entità rappresenta un insieme di oggetti della realtà di cui si individuano proprietà comuni¹.

Rappresentazione grafica:



ASSOCIAZIONE: Una associazione rappresenta un legame logico tra due o più entità.

Rappresentazione grafica:

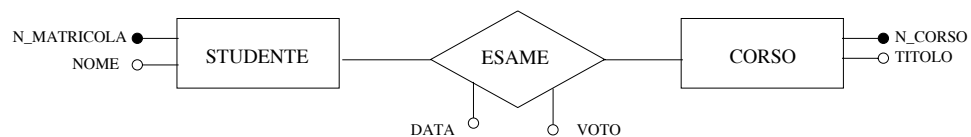


ATTRIBUTO: L’attributo rappresenta proprietà elementari di entità o associazioni.

Rappresentazione grafica:



ESEMPIO:

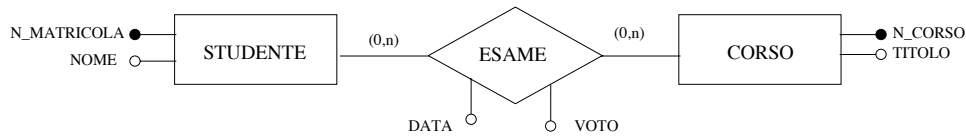


- ◇ STUDENTE e CORSO sono entità.
- ◇ ESAME è una associazione tra STUDENTE e CORSO.
- ◇ N_MATRICOLA e NOME sono attributi di STUDENTE.
- ◇ DATA e VOTO sono attributi di ESAME.
- ◇ N_CORSO e TITOLO sono attributi di CORSO.

¹entità, associazione corrispondono in realtà ad insiemi di entità, insiemi di associazioni che hanno proprietà comuni; per semplicità vengono indicate come entità e associazioni.

ASSOCIAZIONI

◇ Le associazioni sono caratterizzate in termini di cardinalità.



◇ uno studente può aver sostenuto zero, uno o più esami; con cardinalità minima 0; si afferma che uno studente esiste anche se non ha sostenuto esami:

$$\begin{aligned}
 \text{min-card(STUDENTE,ESAME)} &= 0 \\
 \text{max-card(STUDENTE,ESAME)} &= n \\
 \Downarrow \\
 \text{card(STUDENTE,ESAME)} &= (0,n)
 \end{aligned}$$

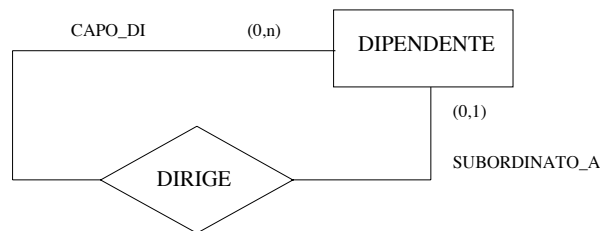
◇ un corso può esistere indipendentemente dal fatto che vi siano o meno studenti che abbiano già sostenuto esami relativi a quel corso:

$$\begin{aligned}
 \text{min-card(CORSO,ESAME)} &= 0 \\
 \text{max-card(CORSO,ESAME)} &= n \\
 \Downarrow \\
 \text{card(CORSO,ESAME)} &= (0,n)
 \end{aligned}$$

ANELLI: Un'anello è una associazione binaria tra un'entità e se stessa.
Il ruolo di un'entità è indicato tramite una *label*.

ESEMPIO:

DIRIGE connette i capi ai subordinati, entrambi istanze dell'entità DIPENDENTE (ruoli CAPO_DI e SUBORDINATO_A).



◇ Ogni capo può dirigere (CAPO-DI) più dipendenti.

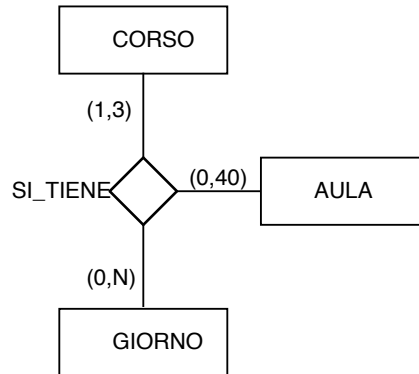
◇ Ogni dipendente è direttamente SUBORDINATO_A un solo capo.

ASSOCIAZIONI n-arie

Una associazione n-aria connette più di due entità.

Esempio:

SI_TIENE è una associazione ternaria che connette le entità CORSO, GIORNO e AULA, con le seguenti cardinalità:



$\text{card}(\text{CORSO}, \text{SI_TIENE}) = (1,3)$

$\text{card}(\text{GIORNO}, \text{SI_TIENE}) = (0,n)$

$\text{card}(\text{AULA}, \text{SI_TIENE}) = (0,40)$

ATTRIBUTI

Dominio dell'attributo: insieme di valori legali per l'attributo.

Un attributo è detto semplice se è definito su un solo dominio.

Sia E un'entità o un'associazione e A un attributo di E:

Cardinalità minima: $\text{min-card}(A,E)$
 è il minimo numero di valori dell'attributo associati a ogni istanza dell'entità o associazione E.

Cardinalità massima: $\text{max-card}(A,E)$
 è il massimo numero di valori dell'attributo associato a ogni istanza dell'entità o associazione E.

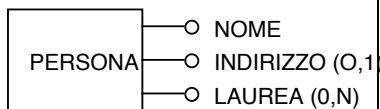
opzionale: $\text{min-card}(A,E) = 0$
 può essere non specificato il valore dell'attributo

obbligatorio: $\text{min-card}(A,E) = 1$
 almeno un valore dell'attributo deve essere specificato

valore-singolo: $\text{max-card}(A,E) = 1$

valore-multiplo: $\text{max-card}(A,E) > 1$

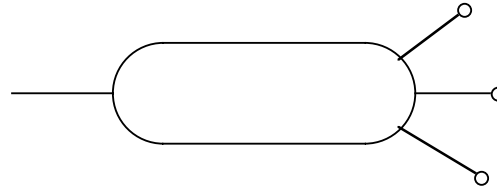
- una persona ha esattamente un nome
- una persona può avere al massimo un indirizzo
- una persona può avere più lauree



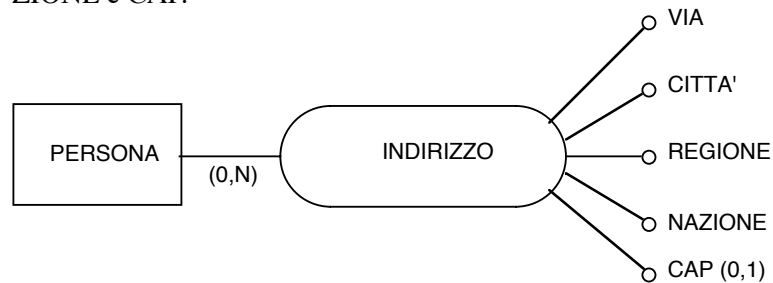
ATTRIBUTI COMPOSTI

- ◇ Un attributo composto è costituito da un gruppo di attributi che hanno affinità nel significato o nell'uso.

Rappresentazione grafica:



Esempio: INDIRIZZO denota il gruppo di attributi VIA, CITTA', REGIONE, NAZIONE e CAP.



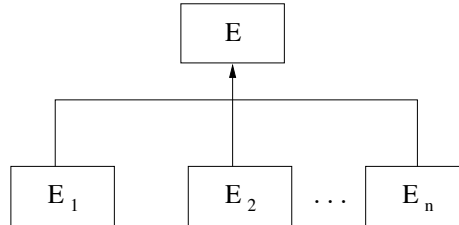
Si noti la maggiore capacità espressiva:

- ◇ Con la cardinalità sull'attributo composto **INDIRIZZO** si dice che una persona può avere più indirizzi, ciascuno dei quali composto da via, città, regione, nazione e c.a.p. (opzionale).
- ◇ Se invece si fossero usati cinque attributi semplici si poteva solo stabilire la cardinalità di ciascuno di essi indipendentemente da quella degli altri.

GERARCHIE DI GENERALIZZAZIONE

Un'entità E è una **generalizzazione** di un gruppo di entità E_1, E_2, \dots, E_n se ogni oggetto delle classi E_1, E_2, \dots, E_n è anche un oggetto della classe E .

Rappresentazione grafica:

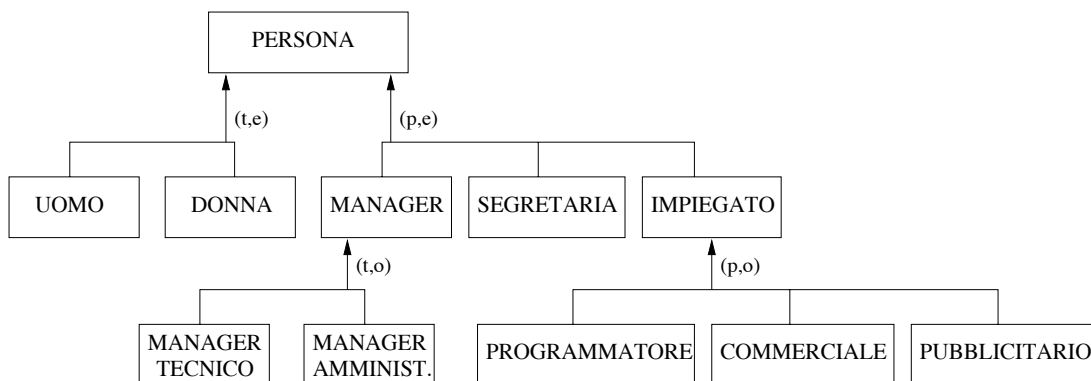


Proprietà di copertura della generalizzazione:

- ◇ totale ed esclusiva: **(t,e)**
- ◇ parziale ed esclusiva: **(p,e)**
- ◇ parziale e sovrapposta: **(p,o)**
- ◇ totale e sovrapposta: **(t,o)**

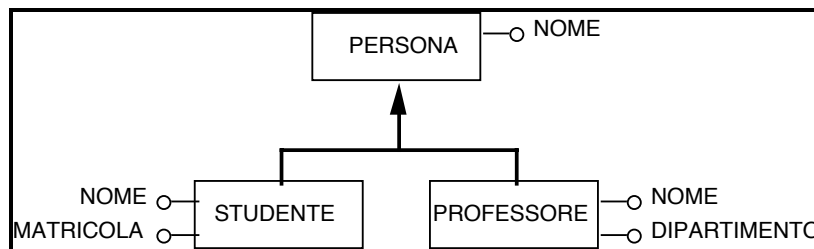
Esempio:

- ◇ la generalizzazione delle persone basata sul sesso è **(t,e)**;
- ◇ vi sono persone che non sono nè impiegati, nè segretari e nè manager: la generalizzazione basata sull'impiego è **(p,e)**;
- ◇ gli impiegati possono avere più di un lavoro, anche diverso da quelli rappresentati in figura: tale generalizzazione è **(p,o)**;
- ◇ tutti i manager ricoprono il ruolo tecnico e/o amministrativo: la generalizzazione basata sul ruolo manageriale è **(t,o)**;

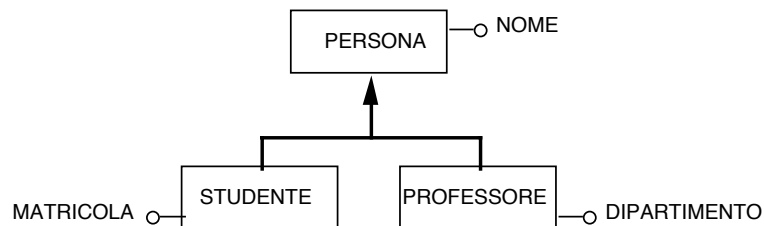


Ereditarietà delle proprietà

- ◇ Nell'astrazione di generalizzazione tutte le proprietà dell'entità generica sono **ereditate** dalle entità generalizzate. Nel modello E/R ogni attributo, associazione e generalizzazione definita per l'entità generica E è **ereditata automaticamente** da tutte le entità generalizzate E_1, E_2, \dots, E_n .



L'attributo NOME della classe PERSONA è anche attributo delle classi STUDENTE e PROFESSORE, pertanto essi possono essere eliminati da tali classi ottenendo il seguente schema semplificato:

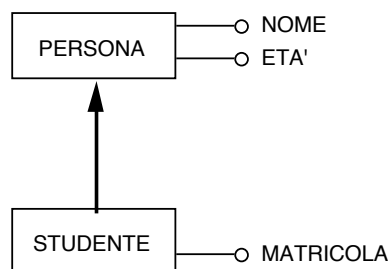


SUBSET: Un subset è una gerarchia di generalizzazione con una sola entità generalizzata.

La copertura di un subset è **parziale**.

Esempio:

STUDENTE ha, oltre agli attributi ereditati da PERSONA, l'attributo addizionale MATRICOLA



IDENTIFICATORI

Un identificatore di un'entità E è una collezione di attributi o di entità in associazione con E che individua in modo univoco tutte le istanze di E .

Definizione formale di identificatore Sia E un'entità e siano

- A_1, \dots, A_n : attributi a **valore singolo** ed **obbligatori** per E
- E_1, \dots, E_m : entità diverse da E e connesse ad E tramite associazioni binarie R_1, \dots, R_m **obbligatorie** ($\min\text{-card}(E, R_i)=1$) e **one-to-one** o **many-to-one** ($\max\text{-card}(E, R_i)=1$)

Possibili identificatori $I = \{A_1, \dots, A_n, E_1, \dots, E_m\}$, $n \geq 0, m \geq 0, n + m \geq 1$.

Valore dell'identificatore di un'istanza di E : l'insieme di tutti i valori degli attributi A_i , $i=1, \dots, n$ e di tutte le istanze E_j , $j=1, \dots, m$.

I è un identificatore di E se:

1. non ci sono due istanze di E con lo stesso valore dell'identificatore;
2. eliminando un attributo A_i oppure un'entità E_j da I , la proprietà 1. non è più valida.

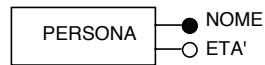
◇ Per le ipotesi fatte sulla cardinalità degli attributi e delle entità nelle associazioni che costituiscono un identificatore, il valore di un identificatore è sempre ben definito. Cioè per ogni istanza E , esiste **uno ed un solo** valore dell'attributo A_i oppure **una ed una sola** istanza dell'entità E_j .

Classificazione degli identificatori:

- ◇ **semplice** se $n+m=1$; **composto** se $n+m>1$.
- ◇ **interno** se $m=0$; **esterno** se $n=0$.
- ◇ **mixed** se $n>0$ e $m>0$.
- ◇ Ogni entità deve avere almeno un identificatore.
 Gli identificatori interni sono preferibili rispetto a quelli esterni.
 Gli identificatori semplici sono preferibili rispetto a quelli composti.
- ◇ È importante evitare circolarità nella definizione degli identificatori: se l'entità E_j è usata nell'identificazione dell'entità E_i , allora E_i non può essere usata nell'identificazione di E_j .
- ◇ **entità forte**: ha solo identificatori interni.
- ◇ **entità debole**: ha solo identificatori esterni.

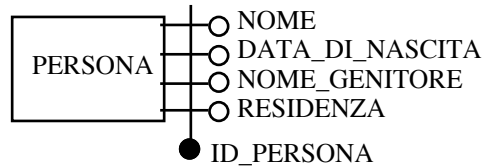
Esempi di identificatori

a) identificatore semplice ed interno $I = \{NOME\}$



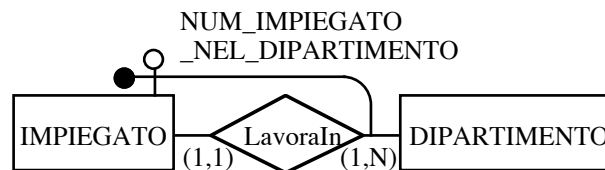
b) identificatore composto ed interno

$I = \{NOME, DATA_DI_NASCITA, NOME_GENITORE, RESIDENZA\}$
 (il nome dell'identificatore, ID_PERSONA è opzionale)

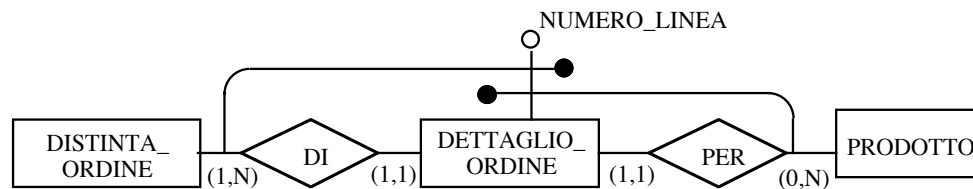


c) identificatore composto e mixed

$I = \{NUM_IMPIEGATO_NEL_DIPARTIMENTO, DIPARTIMENTO\}$



◇ identificatori per l'entità DISTINTA_ORDINE



Si assume che due istanze dell'entità DETTAGLIO_ORDINE non possano riferirsi alla stessa istanza dell'entità PRODOTTO e alla stessa istanza dell'entità DISTINTA_ORDINE. In altre parole, si assume che in una distinta d'ordine, non può comparire due volte lo stesso prodotto.

$$\Downarrow$$


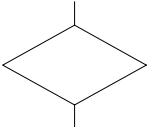

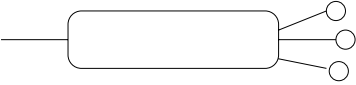
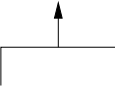

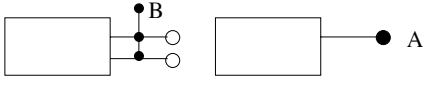
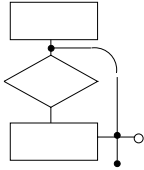
$$\text{card}(\text{DETTAGLIO_ORDINE}, \text{DI}) = (1,1)$$

$$\text{card}(\text{DETTAGLIO_ORDINE}, \text{PER}) = (1,1)$$

A) composto e esterno : $I = \{DISTINTA_ORDINE, PRODOTTO\}$

B) composto e mixed : $I = \{DISTINTA_ORDINE, NUM_LINEA\}$

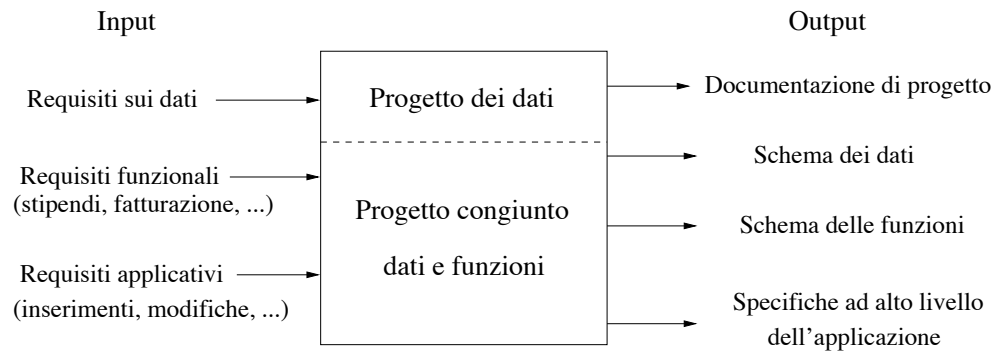
SIMBOLI GRAFICI DEL MODELLO E/R

Concetto	Rappresentazione Grafica
Entita'	
Associazione	
Attributo	
Attributo composto	
Gerarchia di generalizzazione	
Subset	
Identificatore	
Identificatore esterno/mixed	

1.3 Metodologia per il progetto concettuale

Questa sezione è tratta dal libro C. Batini, S. Ceri, S. Navathe, “Conceptual Database Design - An Entity-Relationship approach”, Benjamin, Cummings 1992.

La metodologia di progetto generale è presentata in figura.



I requisiti sui dati possono essere forniti secondo diverse modalità:

1. requisiti in linguaggio naturale
2. moduli
3. tracciati record o schermate
4. schemi di dati

Nel seguito verranno trattati i requisiti sui dati relativamente ai soli requisiti in linguaggio naturale e tracciati record.

I requisiti funzionali e applicativi verranno espressi nella forma di schemi navigazionali E/R.

Metodologia per il progetto concettuale

La costruzione di uno schema E/R è un processo incrementale: la nostra percezione della realtà è progressivamente raffinata e arricchita e lo schema concettuale è formalmente sviluppato.

Primitive di raffinamento: trasformazioni che applicate allo schema iniziale producono lo schema finale.

Strategie di progetto	<ul style="list-style-type: none"> ● top-down ● bottom-up ● mixed
------------------------------	---

Primitive di raffinamento	+		=	Metodologia di progetto
Strategie di progetto				

Una metodologia di progetto dovrebbe idealmente essere un compromesso tra due aspetti contrastanti:

1. **rigorosità:** metodologia come approccio formale nel quale ogni processo di decisione dovrebbe idealmente corrispondere ad un algoritmo;
2. **flessibilità:** metodologia flessibile in modo che ogni progettista possa adattarla ad un problema specifico e seguire il proprio stile di progettazione.

1.3.1 Primitive per il progetto concettuale

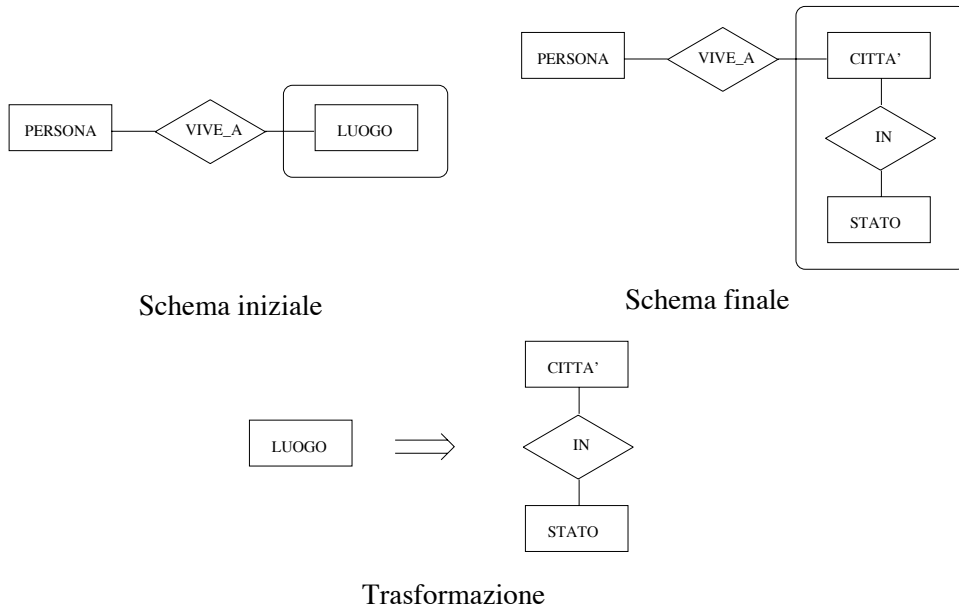
Il progetto di uno schema concettuale è un processo iterativo durante il quale, partendo da uno schema iniziale, si effettuano delle **trasformazioni di schema** per produrre lo schema finale.

Caratteristiche delle trasformazioni dello schema:

1. Ogni trasformazione si applica ad uno **schema iniziale** e produce uno **schema finale**.
Nell'esempio successivo, nello schema iniziale è presente un'entità LUOGO e nello schema finale LUOGO è rappresentato da una coppia di entità CITTÀ, STATO connesse da un'associazione IN.
2. Ogni trasformazione mappa **nomi** di concetti dello schema iniziale in nomi di concetti dello schema finale.
Nell'esempio al nome iniziale LUOGO corrisponde l'insieme di nomi CITTÀ, IN, STATO.

3. I concetti dello schema finale devono ereditare tutte le **connessioni logiche** definite per i concetti dello schema iniziale.

Nell'esempio l'associazione VIVE_A tra PERSONA e LUOGO è ereditata dall'entità CITTÀ.



Trasformazioni primitive

Le trasformazioni primitive sono trasformazioni che non possono essere decomposte in altre più semplici.

PRIMITIVE TOP-DOWN

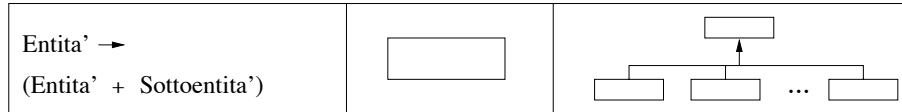
- Hanno una struttura semplice: lo schema iniziale è un concetto semplice e lo schema finale consiste di un piccolo insieme di concetti;
- Tutti i nomi sono raffinati in nuovi nomi che descrivono il concetto originale ad un livello di astrazione più basso;
- Le connessioni logiche dello schema iniziale devono essere ereditate dai concetti dello schema finale.

PRIMITIVE BOTTOM-UP

- Introducono nuovi concetti e proprietà che non comparivano nelle precedenti versioni dello schema oppure modificano alcuni concetti esistenti;
- Sono usate quando alcune caratteristiche del dominio di applicazione non sono rappresentate nelle versioni precedenti dello schema;
- Sono applicate per integrare schemi differenti in uno schema globale più comprensivo (**integrazione di schemi**).

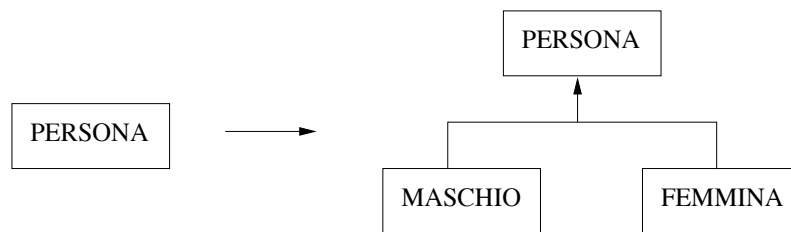
Esempi di primitive top-down

- ◇ Un'entità è trasformata in una gerarchia di generalizzazione o in un subset.

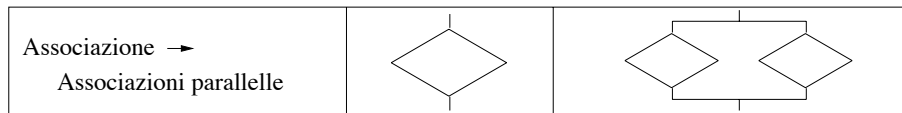


Esempio

L'entità PERSONA è trasformata in una generalizzazione che include MASCHIO e FEMMINA.

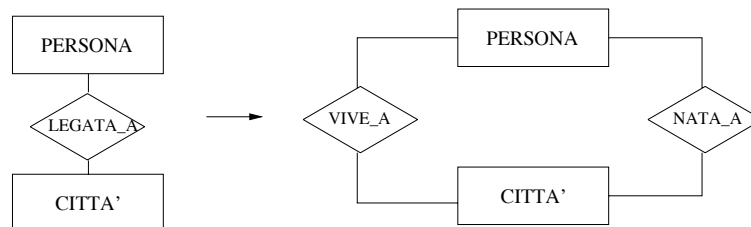


- ◇ Un'associazione è trasformata in una o più associazioni tra le stesse entità.

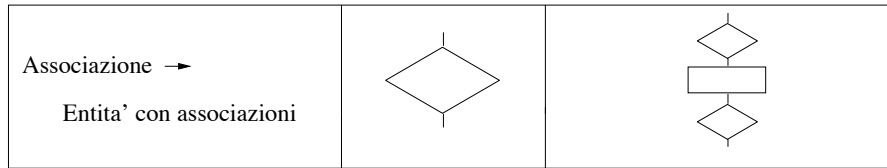


Esempio

L'associazione LEGATA_A tra PERSONA e CITTÀ è raffinata in due associazioni, VIVE_A e NATA_A, tra le stesse entità.

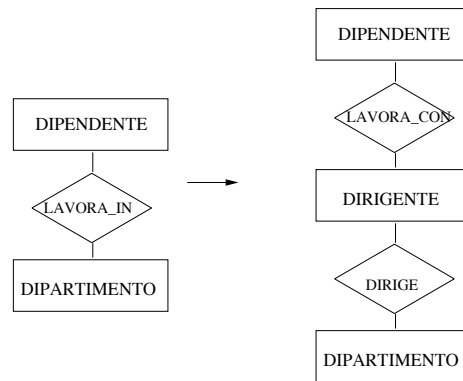


- ◇ Un'associazione è trasformata in un *cammino* di entità e associazioni. Questo corrisponde a riconoscere che un'associazione tra due entità deve essere espressa tramite una terza entità e due un'associazioni. Questa trasformazione è detta anche *reifificazione*.

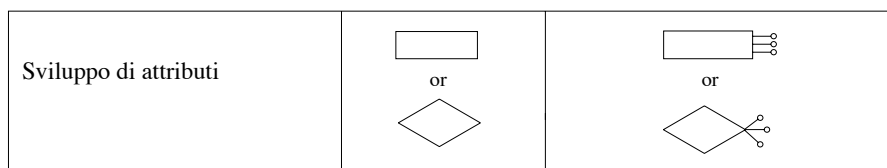


Esempio

L'associazione LAVORA_IN tra DIPENDENTE e DIPARTIMENTO è raffinata in una aggregazione più complessa che include l'entità DIRIGENTE e due nuove associazioni.

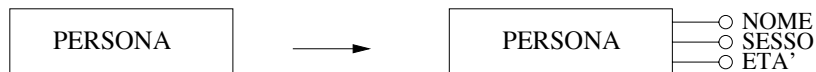


- ◇ Un'entità o un'associazione è trasformata introducendo i suoi attributi.



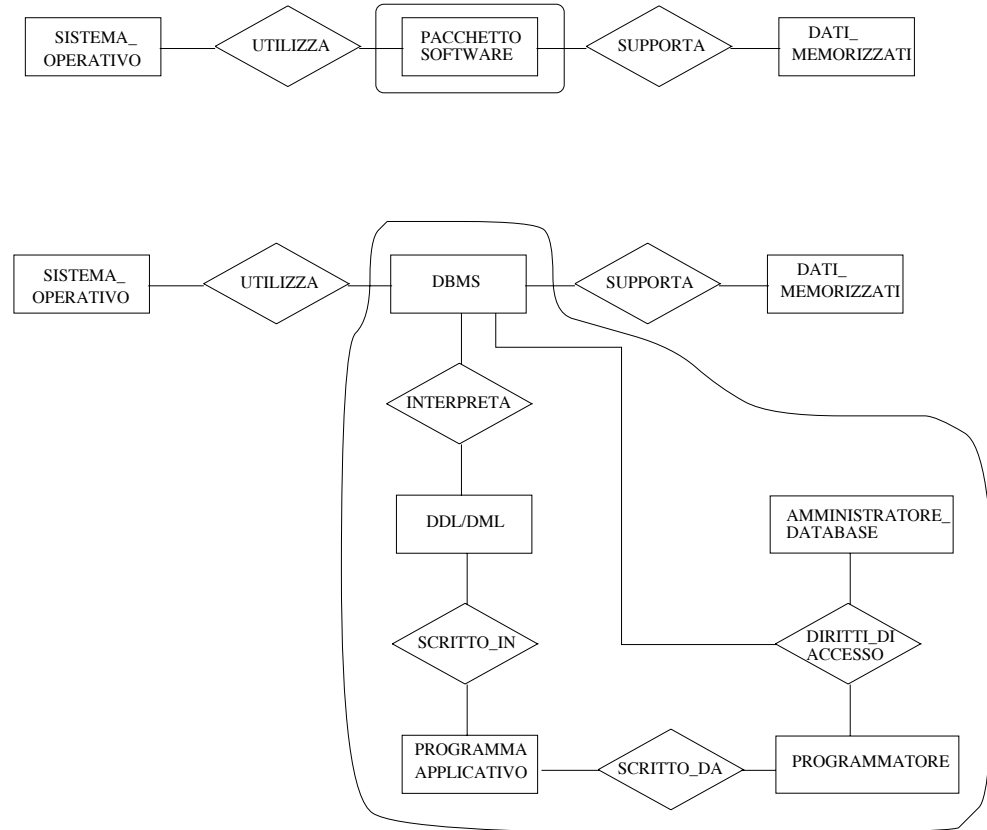
Esempio

Per l'entità PERSONA sono introdotti gli attributi NOME, SESSO ed ETÀ.



Esempio di applicazione delle primitive top-down

- ◇ Le primitive presentate sono alcune tra le più significative di tipo top-down;
- ◇ Altre e più complesse trasformazioni possono essere classificate come top-down, come mostrato nel seguente esempio, dove un'entità è trasformata in un insieme di entità e associazioni.

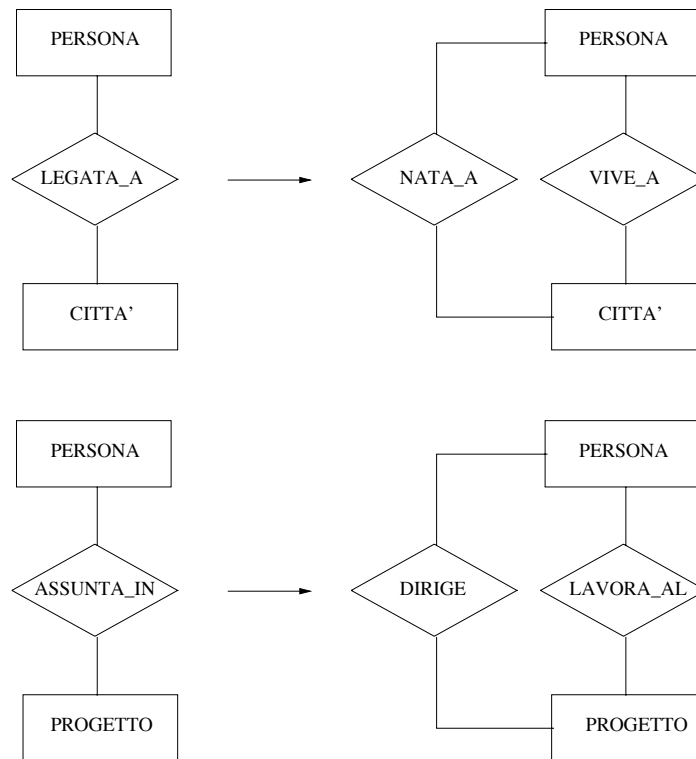


- ◇ Nell'applicazione delle primitive di raffinamento si devono rispettare alcuni vincoli.

Consideriamo l'applicazione della trasformazione di un'associazione in *cammino* di entità e associazioni: esiste un legame tra le cardinalità, minima e massima, dell'associazione nello schema di partenza e quelle delle due nuove associazioni introdotte. Ad esempio, se nello schema di partenza si ha un'associazione one-to-one, lo schema risultante non può includere un'associazione many-to-many.

◇ I vincoli dipendono anche dal significato dei concetti.

Consideriamo due diverse applicazioni della suddetta primitiva:



Nel primo caso le associazioni prodotte dalla trasformazione sono indipendenti da quella originale.

Nel secondo caso invece le istanze dell'associazione ASSUNTA.IN sono partizionate nelle istanze delle associazioni DIRIGE e LAVORA.AL.

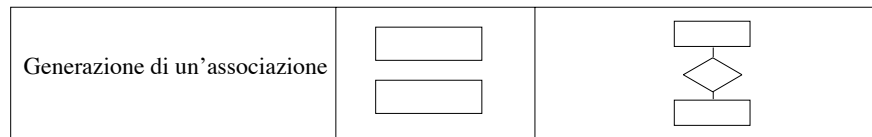
Esempi di primitive bottom-up

- ◇ Generazione di una nuova entità.

È usata per introdurre un nuovo concetto nello schema.

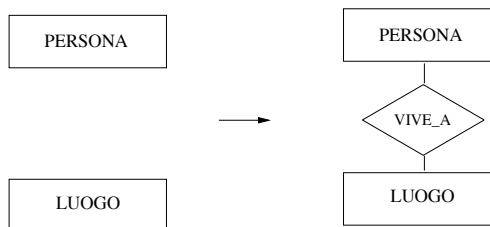


- ◇ Generazione di una nuova associazione tra entità precedentemente definite.

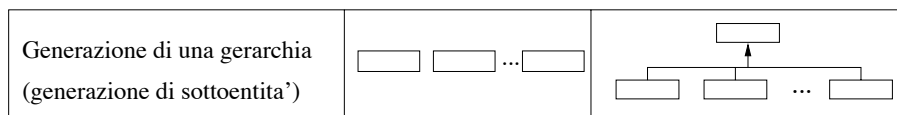


Esempio

La nuova associazione VIVE_A è stabilita tra le entità PERSONA e LUOGO.

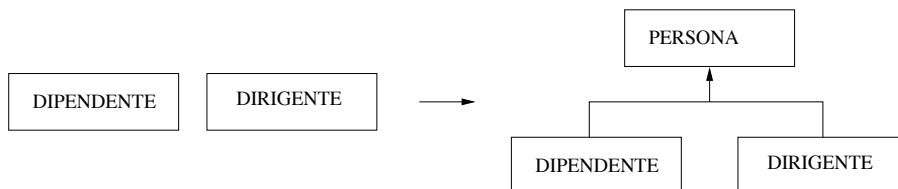


- ◇ Generazione di una nuova entità come generalizzazione di entità esistenti.



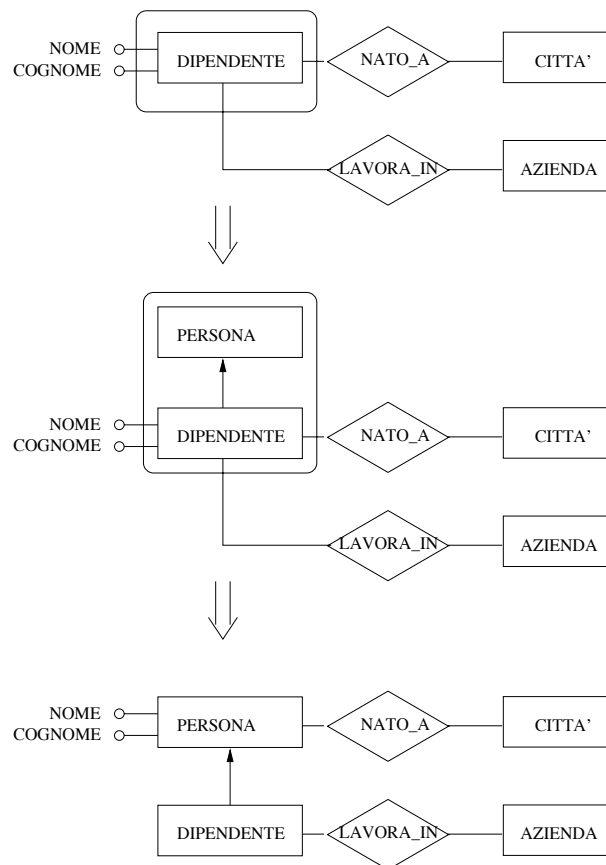
Esempio

Si forma una gerachia di generalizzazione, introducendo la classe PERSONA.



L'applicazione della primitiva di generazione di una gerarchia rende necessario controllare se le proprietà possono migrare tra le entità per effetto della nuova generalizzazione.

Nell'esempio seguente, usando la primitiva di generazione di una gerarchia, si introduce una nuova entità PERSONA e una relazione di generalizzazione tra PERSONA e DIPENDENTE. Di conseguenza gli attributi NOME e COGNOME e l'associazione NATO_A devono essere spostate e assegnate all'entità generalizzazione. Si noti invece come l'associazione LAVORA_IN rimane assegnata all'entità DIPENDENTE.



1.3.2 Strategie per il progetto concettuale

1. **top-down**

Lo schema è ottenuto applicando solo le primitive top-down. Il processo finisce quando tutti i requisiti sono stati rappresentati.

In una strategia top-down *pura* tutti i concetti da rappresentare nello schema finale devono essere presenti in ogni schema di raffinamento intermedio (naturalmente ad un diverso livello di astrazione).

Il processo termina quando tutti i requisiti sono stati rappresentati.

2. **bottom-up**

Lo schema è ottenuto applicando solo le primitive bottom-up, iniziando da concetti elementari e costruendo concetti più complessi.

I concetti vengono progressivamente integrati nello schema finale.

3. **mixed**

Utilizzo sia delle strategie top-down che di quelle bottom-up.

I requisiti, se il dominio è molto complesso, sono partizionati in sottinsiemi e considerati separatamente; nello stesso tempo si produce uno **schema scheletro** che serve come collegamento tra le varie partizioni.

Applicazione delle strategie ad un esempio

Base di dati demografica che rappresenta persone e luoghi.

Requisiti:

- Per le persone sono considerate le seguenti proprietà: nome, cognome, sesso, età, luogo di nascita e di residenza, durata della residenza, posizione militare per gli uomini e il nome da nubile per le donne.
- Un luogo può essere uno stato estero o una città nazionale; ognuno di essi ha un nome e una popolazione. Per uno stato estero deve essere indicato il continente e per una città nazionale il nome della regione.

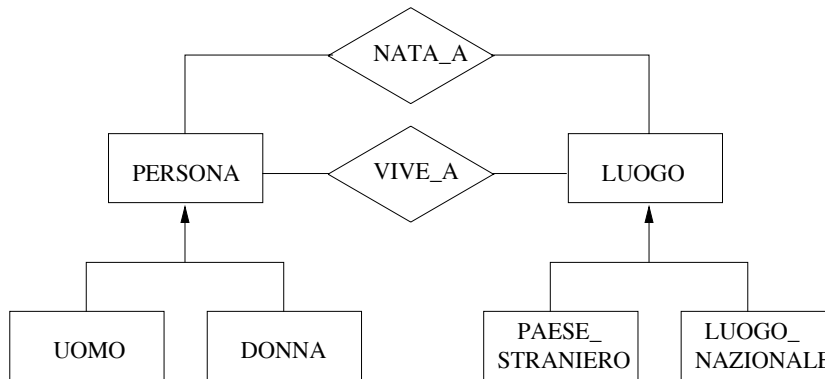
Applicazione della strategia top-down



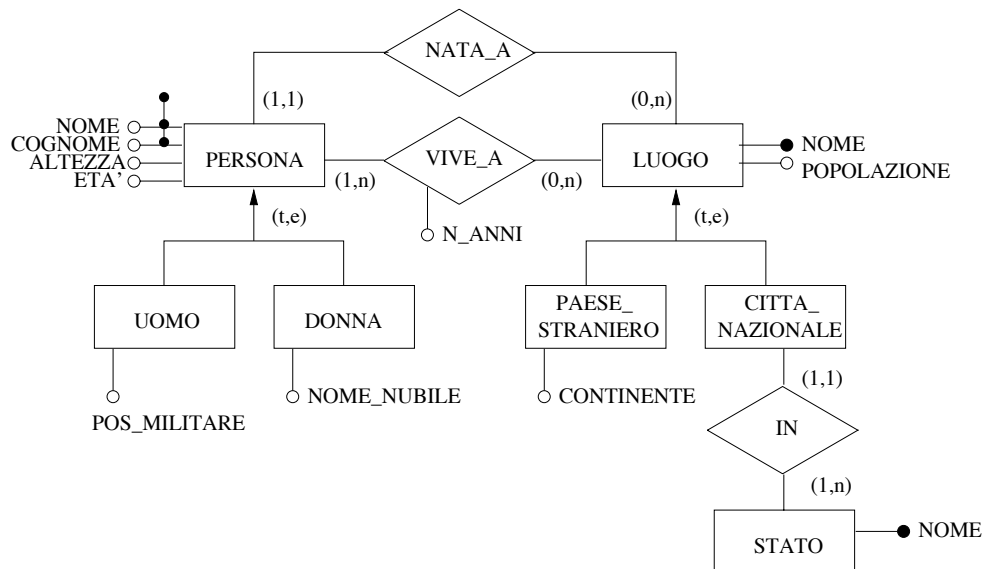
L'entità è raffinata in due entità e un'associazione tra di esse.



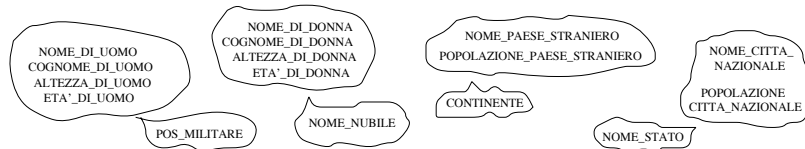
L'associazione LEGATI_A è raffinata in due associazioni differenti e le entità PERSONA e LUOGO sono trasformate in due gerarchie di generalizzazione.



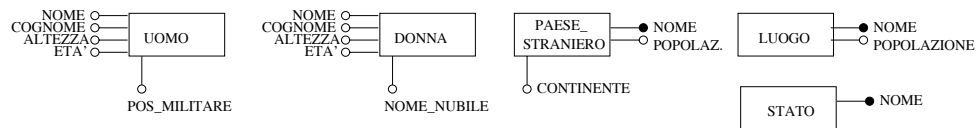
L'entità LUOGO_NAZIONALE è raffinata in due entità e un'associazione tra di esse. Per ottenere lo schema finale sono specificati gli attributi, gli identificatori, la cardinalità di partecipazione alle associazioni.



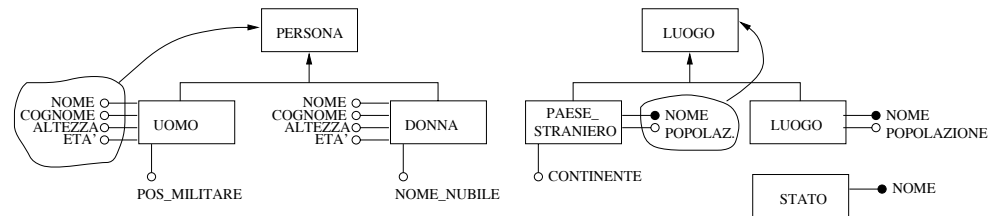
Applicazione della strategia bottom-up



Si introducono le entità che rappresentano l'aggregazione degli attributi dati.

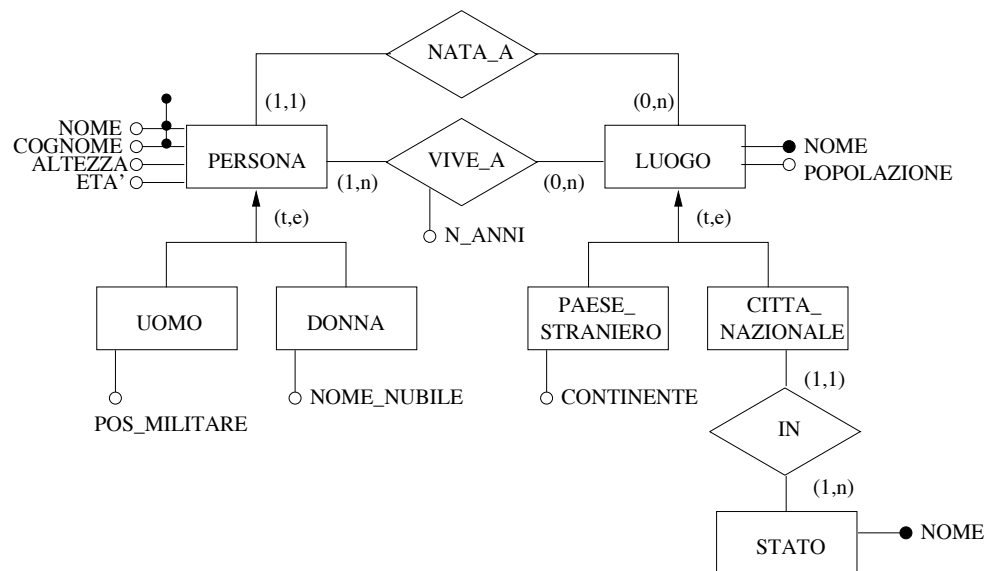


Si formano due gerarchie di generalizzazione sulle entità; le proprietà dovranno migrare per effetto delle generalizzazioni introdotte.



Vengono definite associazioni tra le entità.

Si introducono cardinalità e identificatori.



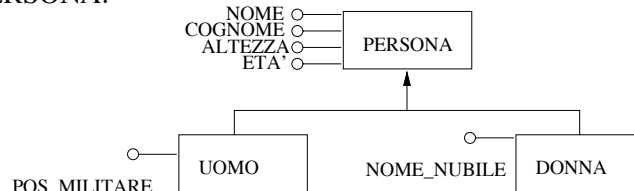
Applicazione della strategia mixed

Schema scheletro:

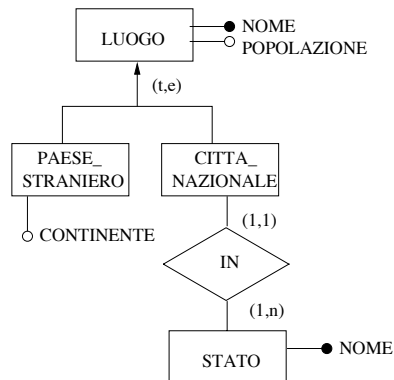
sono individuate le entità PERSONA e LUOGO e tra di esse viene stabilita un'associazione. Gli schemi di PERSONA e LUOGO sono ottenuti separatamente.



Schema di PERSONA:

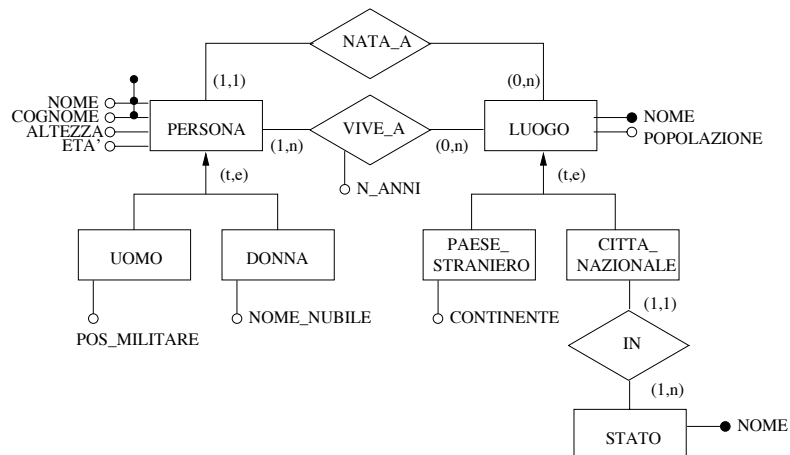


Schema di LUOGO:



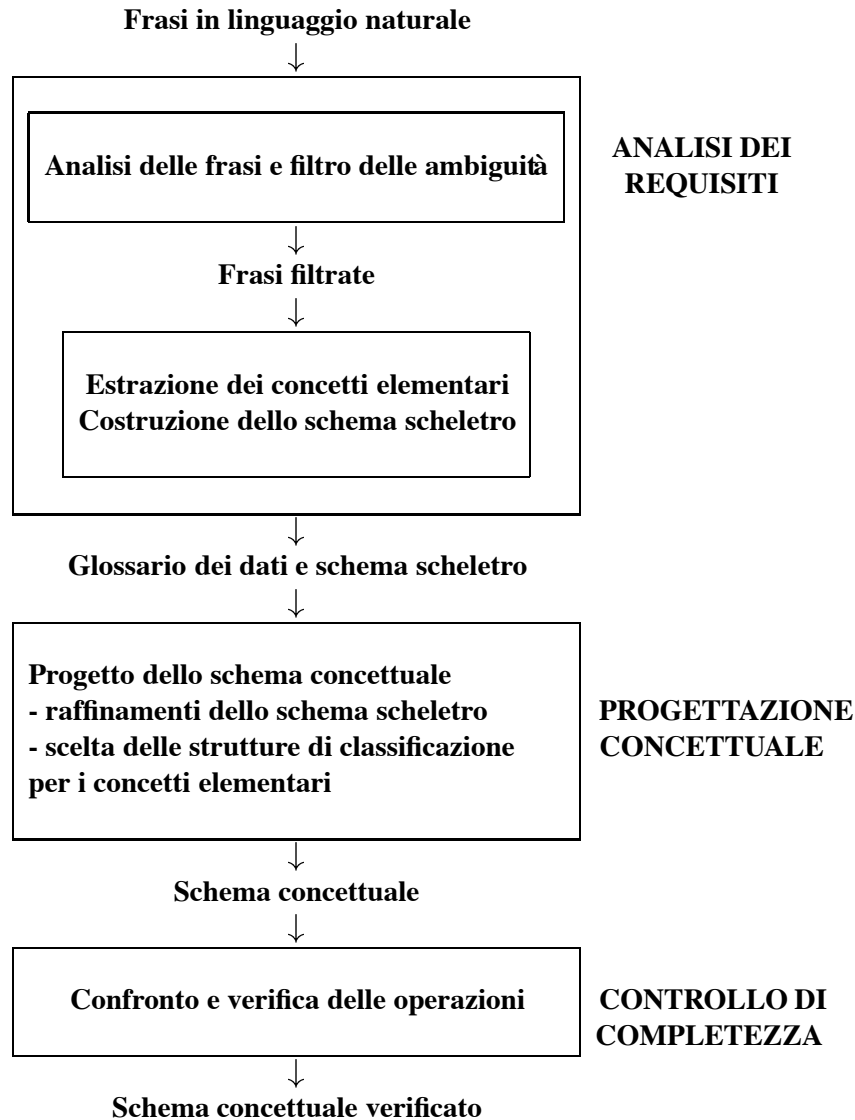
Schema integrato:

gli schemi di PERSONA e di LUOGO sono connessi allo schema scheletro e l'associazione LEGATLA è raffinata in due associazioni:



1.4 Progettazione da requisiti in linguaggio naturale

Fasi delle attività di progettazione concettuale a partire da requisiti espressi in linguaggio naturale



Requisiti per un Database di Società Sportiva:

1.	Si vuole costruire un database per gestire le informazioni relative agli
2.	atleti e alle squadre di una società sportiva. Per gli atleti della società
3.	è necessario memorizzare il numero della tessera di iscrizione, il codice
4.	fiscale, il nome, il cognome, il sesso, l'indirizzo, la data ed il luogo di
5.	nascita e la squadra di appartenenza. Gli atleti possono frequentare
6.	corsi annuali organizzati dalla società. Per gli atleti frequentatori
7.	interessano: la data dell'ultima visita medica, i corsi annuali che
8.	hanno seguito prima, con l'esito ottenuto e i corsi che stanno
9.	seguendo attualmente. Per gli atleti professionisti si indica la
10.	disciplina sportiva e il preparatore atletico. In generale, per i corsi
11.	si rappresenta il codice corso e la sua descrizione; per i corsi correnti
12.	si rappresentano, oltre al loro costo e al numero dei partecipanti,
13.	i giorni, le relative ore di inizio e di fine e il luogo in cui si svolgono,
14.	con relativa descrizione, indirizzo e telefono. Un corso si può svolgere
15.	una o più volte nello stesso giorno, in più impianti o nello stesso
16.	impianto. Ogni corso è tenuto da un allenatore per il quale si
17.	rappresenta il codice fiscale, il nome, il cognome, il sesso, l'indirizzo,
18.	la squadra di appartenenza e la specializzazione sportiva.

Termini ambigui nei requisiti e correzioni possibili:

Linea	Termine	Nuovo termine	Motivo correzione
8	Prima	In anni precedenti	Parola generica
9	Attualmente	Nell'anno corrente	Parola ambigua
10	Preparatore atletico	Allenatore	Termine più specifico
15	Giorno	Giorno della sett.	Termine più specifico
18	Specializzazione	Disciplina	Termine più specifico

Requisiti dopo il filtraggio delle ambiguità:

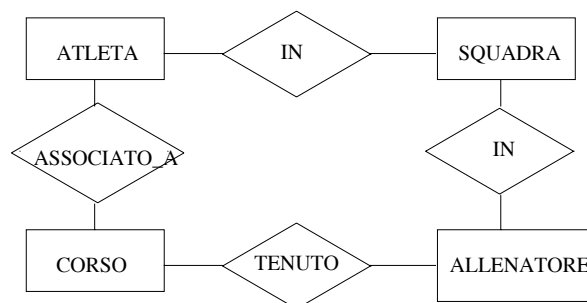
1.	Si vuole costruire un database per gestire le informazioni relative agli
2.	atleti e alle squadre di una società sportiva. Per gli atleti della società
3.	è necessario memorizzare il numero della tessera di iscrizione, il codice
4.	fiscale, il nome, il cognome, il sesso, l'indirizzo, la data ed il luogo di
5.	nascita e la squadra di appartenenza. Gli atleti possono frequentare
6.	corsi annuali organizzati dalla società. Per gli atleti frequentatori
7.	interessano: la data dell'ultima visita medica, i corsi annuali che
8.	hanno seguito in anni precedenti , con l'esito ottenuto e i corsi che
9.	stanno seguendo nell'anno corrente. Per gli atleti professionisti si
10.	indica la disciplina sportiva e l'allenatore. In generale, per i corsi
11.	si rappresenta il codice corso e la sua descrizione; per i corsi correnti
12.	si rappresentano, oltre al loro costo e al numero dei partecipanti,
13.	i giorni, le relative ore di inizio e di fine e il luogo in cui si svolgono,
14.	con relativa descrizione, indirizzo e telefono. Un corso si può svolgere
15.	una o più volte nello stesso giorno della settimana, in più impianti
16.	o nello stesso impianto. Ogni corso è tenuto da un allenatore per il
17.	quale si rappresenta il codice fiscale, il nome, il cognome, il sesso,
18.	l'indirizzo, la squadra di appartenenza e la disciplina sportiva.

Progetto Iniziale:

Si deve costruire uno schema scheletro con i concetti più evidenti espressi nei requisiti. I concetti referenziati con maggiore frequenza nei requisiti sono buoni candidati per tale scelta:

ATLETA, CORSO, ALLENATORE, SQUADRA.

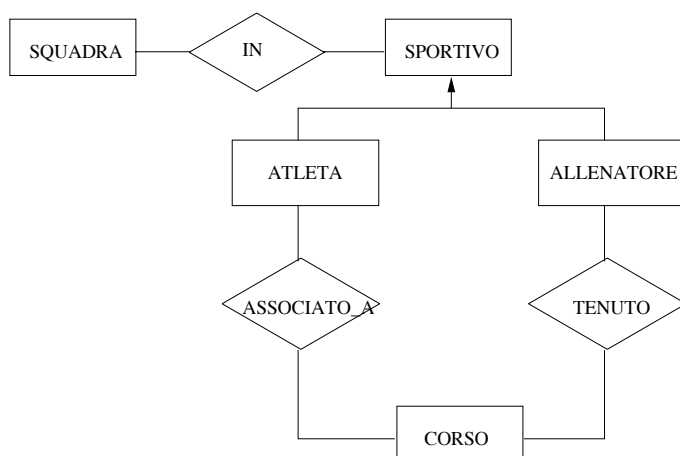
A tali entità si impongono le associazioni più evidenti; l'associazione ASSOCIATO_A è deliberatamente vaga e sarà specificata dopo.



Schema scheletro iniziale:

Prima di continuare il progetto, si controlla se è possibile ristrutturare lo schema scheletro precedente.

Osservando l'associazione IN, si scoprono similarità tra le entità ATLETA e ALLENATORE: si introduce SPORTIVO come generalizzazione di tali entità e si riferisce ad essa l'associazione SPORTIVO.

Raffinamento dello schema scheletro:**Progetto dello schema****Raffinamenti top-down :**

1. l'entità ATLETA è ridefinita in termini di due sottoentità: ATLETA_FREQUENTATORE e ATLETA_PROFESSIONISTA
2. l'associazione ASSOCIATO_A pe ridefinita con due associazioni: HANNO_SEGUITO e SEGUONO

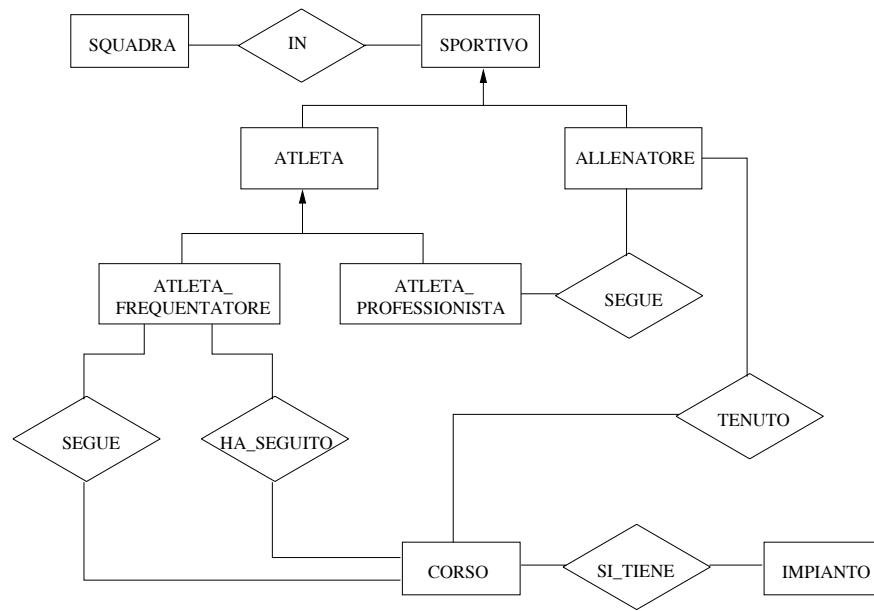
Raffinamenti bottom-up :

introduzione dell'associazione SEGUE tra ALLENATORE e ATLETA_PROFESSIONISTA

Raffinamenti inside-out :

poichè una proprietà di CORSO è l'impianto in cui viene svolto che ha a sua volta diverse proprietà (indirizzo,telefono) si rappresenta l'impianto come entità e si esprime il legame logico tra CORSO e IMPIANTO con l'associazione SL TIENE.

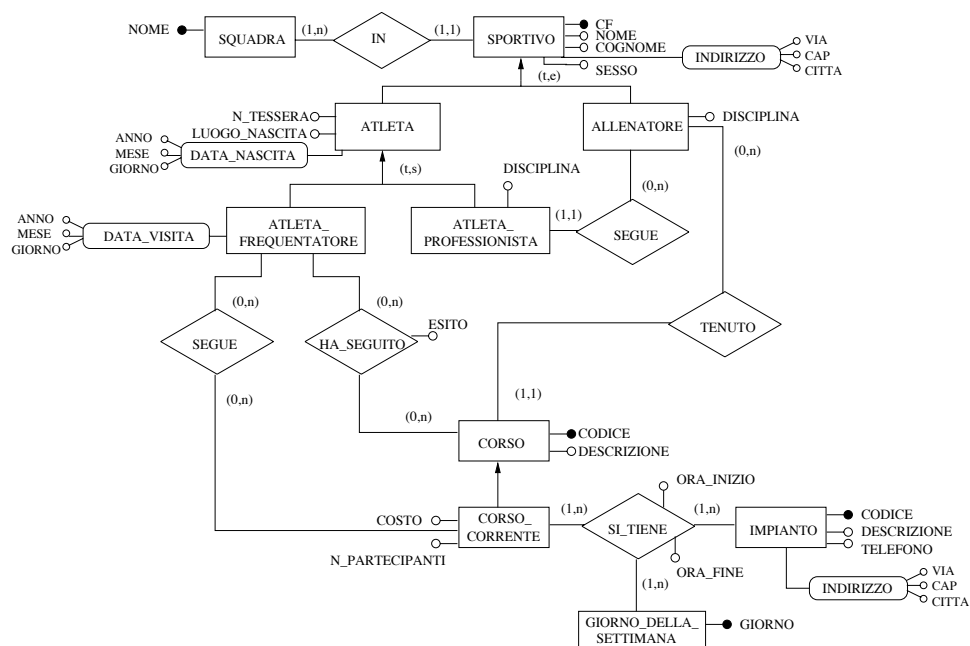
Ne risulta il seguente schema:



Ulteriori raffinamenti :

1. l'associazione SLTIENE viene raffinata per rappresentare l'orario dei vari corsi. Poichè un CORSO si può tenere una o più volte nello stesso giorno, si introduce l'entità GIORNO.DELLA.SETTIMANA e si trasforma l'associazione SLTIENE in ternaria
2. si specializza l'entità CORSO con un subset CORSO_CORRENTE: solo per i corsi correnti viene specificato l'orario tramite l'associazione SL TIENE
3. attributi, identificatori, cardinalità delle associazioni e coperture delle gerarchie

Si ottiene il seguente schema finale:



1.5 Progettazione da file esistenti

Le applicazioni commerciali usano file, cioè insiemi di record. Spesso le applicazioni di basi di dati devono tenere conto di applicazioni pre-esistenti basate su file e file-system². Per sviluppare un progetto a partire dalla progettazione concettuale è quindi necessario compiere un reverse-engineering che ha come input un insieme di file e che deve produrre come output una vista concettuale E/R da integrare. I linguaggi più usati sono COBOL, PL/1, FORTRAN e C.

Esempio

Consideriamo la dichiarazione in linguaggio COBOL di un file FATTURA. Alcuni campi sono elementari (QUANTITA, CODICE_ARTICOLO, CLIENTE) mentre altri campi sono composti (AMMONTARE, DATA_DLEMISSIONE, ...).

```

01  FATTURA.
    02 NUMERO_FATTURA                PIC X(5).
    02 DATA_DLEMISSIONE.
        03 ANNO_DLEMISSIONE          PIC 9(4).
        03 MESE_DLEMISSIONE          PIC 9(2).
        03 GIORNO_DLEMISSIONE        PIC 9(2).
    02 DATA_DLPAGAMENTO.
        03 ANNO_DLPAGAMENTO          PIC 9(4).
        03 MESE_DLPAGAMENTO          PIC 9(2).
        03 GIORNO_DLPAGAMENTO        PIC 9(2).
    02 AMMONTARE.
        03 PREZZO_NETTO              PIC 9(9)V99.
        03 IVA                      PIC 9(2).
        03 COSTO_IVA                 PIC 9(9) COMPUTATIONAL.
        03 PREZZO_TOTALE             PIC 9(9) COMPUTATIONAL.
    02 LINEA_FATTURA OCCURS 15 TIMES.
        03 NUMLINEA                  PIC 9(2).
        03 CODICE_ARTICOLO           PIC X(5).
        03 QUANTITA                  PIC 9(3).
        03 PREZZO_UNITARIO           PIC 9(9)V99.
    02 CLIENTE                        PIC X(9).
    02 AZIENDA                        PIC X(9).
```

In COBOL varie clausole nella definizione del file specificano il ruolo di un campo, la sua allocazione fisica, il tipo di accesso previsto per il file e altre caratteristiche. Queste informazioni sono molto importanti per determinare il significato dei campi, le loro associazioni logiche interne e le astrazioni definite tra di essi in modo da poter rappresentare il file in termini di uno schema E/R.

Il progetto degli schemi E/R a partire da file esistenti inizia con l'introduzione di una singola entità per rappresentare il file (infatti il file è una collezione di dati con la stessa struttura) con lo stesso nome del file.

²Questa situazione è nota nel mondo database, in lingua inglese, come legacy system.

Quindi si considerano le clausole definite sul file allo scopo di individuare ulteriori proprietà strutturali del file.

In questo modo la rappresentazione iniziale del file è progressivamente arricchita tramite l'introduzione di nuove entità, associazioni, generalizzazioni, attributi e altri concetti.

Nel seguito si analizzano alcune clausole che possono comparire nella definizione di un file e vengono dati metodi generali per la loro traduzione in concetti del modello E/R.

Campi semplici e composti

- ◇ Un campo è semplice quando ha un singolo valore in ogni record istanza;
- ◇ I campi semplici sono trasformati in attributi semplici.
- ◇ I campi composti sono trasformati in attributi composti.

Esempio

Queste linee sono tradotte in attributi semplici dell'entità FATTURA	02 CLIENTE 02 AZIENDA	PIC X(9). PIC X(9).
---	--------------------------	------------------------

Queste linee sono tradotte in attributi composti dell'entità FATTURA	02 DATA_DLEMISSIONE. 03 ANNO_DLEMISS. 03 MESE_DLEMISS. 03 GIORNO_DLEMISS.	PIC 9(4). PIC 9(2). PIC 9(2).
---	--	-------------------------------------

Campi ripetitivi

- ◇ Un campo ripetitivo ha valori multipli. Esso è definito tramite la clausola OCCURS che specifica il numero di volte che il campo compare in un record istanza;
- ◇ Un campo ripetitivo con una **singola** clausola OCCURS viene trasformato in un attributo che ha entrambe min-card e max-card uguali al valore specificato in OCCURS;
- ◇ Un campo ripetitivo con **più** clausole OCCURS (tabella o array) viene trasformato introducendo una nuova entità.

Esempio La tabella mostra l'andamento delle vendite di un prodotto, classificata per mese e anno.

QUANTITÀ VENDUTA DI PRODOTTO

Mese	1994	1996	1997	1998
Gen	81	34	87	48
Feb	12	60	26	27
Mar	67	93	57	36
Apr	54	73	54	09
Mag	57	38	34	05
Giu	55	18	56	09
Lug	23	29	63	18
Ago	65	58	25	39
Set	67	36	95	55
Ott	48	49	48	49
Nov	38	39	75	40
Dic	48	75	39	80

La tabella è descritta dal record VENDITE.PRODOTTO in linguaggio COBOL, tramite l'uso dei campi ripetitivi.

```

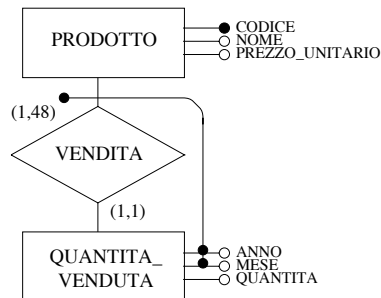
01  VENDITE.PRODOTTO.
    02  NOME                      PIC X(10).
    02  CODICE                    PIC X(4).
    02  PREZZO_UNITARIO          PIC 9(9)V99.
    02  QUANTITA_DISPONIBILE.
        03  QUANTITA_VENDUTA_PER_ANNO OCCURS 4 TIMES.
        04  QUANTITA_VENDUTA_PER_MESE PIC X(3) OCCURS 12 TIMES.

```

Per trasformare il record VENDITE.PRODOTTO si introduce una nuova entità, QUANTITÀ VENDUTA, che è connessa all'entità VENDITE.PRODOTTO tramite l'associazione VENDITA.

Si noti che

$\text{card}(\text{VENDITE.PRODOTTO}, \text{VENDITA}) = (1,48)$



Campi ridefiniti

Si può definire la stessa porzione di un record usando differenti clausole.

La ridefinizione di campo può essere usata per due motivi:

1. vedere gli stessi dati da differenti punti di vista;

Esempio

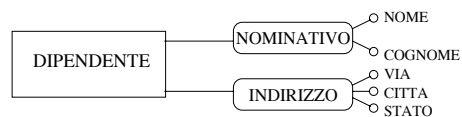
Consideriamo il record DIPENDENTE per la memorizzazione dei dipendenti di una azienda, dove memorizziamo il nome e il cognome e l'indirizzo di ciascuna persona. Il nominativo e la città sono aggregati in due modi diversi: il campo NOMINATIVO è utile per inviare comunicazioni al dipendente; il campo RICERCA può essere utile per operazioni di ricerca.

```

01  DIPENDENTE.
    02 DATI-PERSONALI.
        03 NOMINATIVO.
            04 COGNOME      PIC X(20).
            04 NOME         PIC X(20).
        03 INDIRIZZO.
            04 VIA          PIC X(20).
            04 CITTA        PIC X(10).
            04 STATO        X(10).
    02 DATI-PERSONALI-BIS REDEFINES DATI-PERSONALI.
        03 RICERCA.
            04 NOMINATIVO-R PIC X(40).
            04 CITTA-R      PIC X(10).
  
```

◇ La rappresentazione concettuale può essere una delle due o una loro combinazione.

È preferibile avere tutti gli attributi; si ottiene quindi lo schema concettuale riportato in figura:



2. ottimizzare lo spazio di memorizzazione fisica.

Questo usualmente è indice della presenza di una gerarchia di generalizzazione tra i concetti descritti nel file.

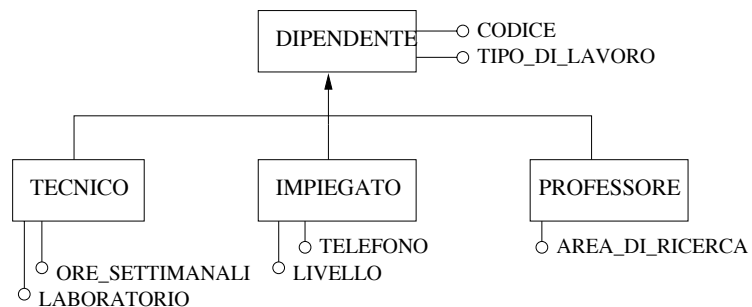
Esempio

Per il record DIPENDENTE il campo INFO_TECNICO è ridefinito due volte nei campi INFO_IMPIEGATO e INFO_PROFESSORE.

```

01  DIPENDENTE.
    02 CODICE                PIC X(7).
    02 TIPO_DI_LAVORO        PIC X.
    02 INFO_TECNICO.
        03 ORE_SETTIMANALI    PIC 99.
        03 LABORATORIO        PIC X(6).
    02 INFO_IMPIEGATO        REDEFINES INFO_TECNICO.
        03 LIVELLO            PIC 9(2).
        03 TELEFONO           PIC 9(7).
    02 INFO_PROFESSORE        REDEFINES INFO_IMPIEGATO.
        03 AREA_DI_RICERCA    PIC X(20)
  
```

Il record viene tradotto in uno schema con l'entità DIPENDENTE e con una gerarchia di generalizzazione con sottoclassi TECNICO, IMPIEGATO e PROFESSORE.



Puntatore simbolico: è un campo di un record che denota l'identificatore di un altro record. I puntatori sono tradotti in associazioni:

queste associazioni connettono le entità con il campo puntatore all'entità corrispondente al file che contiene il record. Le cardinalità dell'associazione dipendono dal significato specifico dei campi.

Esempio

Consideriamo tre record relativi a PROFESSORE, DIPARTIMENTO e PROGETTO. Le associazioni tra questi concetti sono espressi tramite tre differenti campi usati come puntatori:

CODICE-DIP lega i professori ai loro dipartimenti di appartenenza

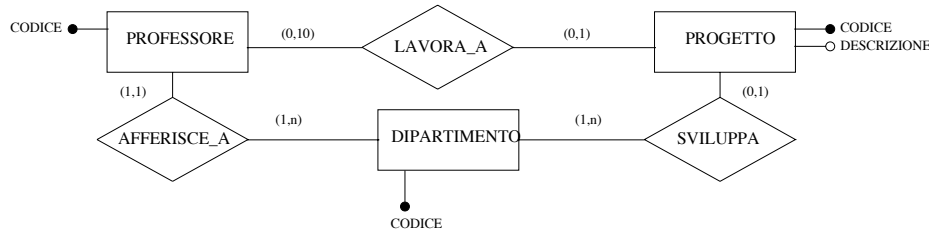
CODICE-PROGETTO connette i professori ai progetti ai quali lavorano

CODICE-DIP lega i progetti ai dipartimenti che li controllano.

```

02 PROFESSORE.
03 CODICE          PIC X(0).
03 CODICE-DIP      PIC X(5).
03 CODICE-PROGETTO PIC X(7) OCCURS 10 TIMES.
02 DIPARTIMENTO.
03 CODICE          PIC X(5).
02 PROGETTO.
03 CODICE          PIC X(7).
03 CODICE-DIP      PIC X(5).
03 DESCRIZIONE     PIC X(30).
  
```

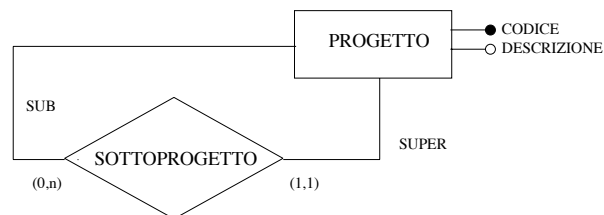
Lo schema E/R risultante è quindi il seguente:



Un puntatore che si riferisce al record stesso nel quale è definito, è tradotto in un'associazione binaria "ad anello" sull'entità stessa:

```

01 PROGETTO.
02 CODICE
02 DESCRIZIONE
02 CODICE-SOTTOPROGETTO
02 BUDGET
  
```



Flag

Flag: flag è riferito ad un campo (o ad un gruppo di campi) e indica se il campo (o il gruppo di campi) del record istanza ha un valore o è lasciato vuoto.

I flag possono essere tradotti in due modi differenti:

1. al file corrisponde un' unica entità con il campo indicato dal flag come attributo opzionale (min-card = 0);
2. al file corrispondono due entità legate da una relazione di generalizzazione.

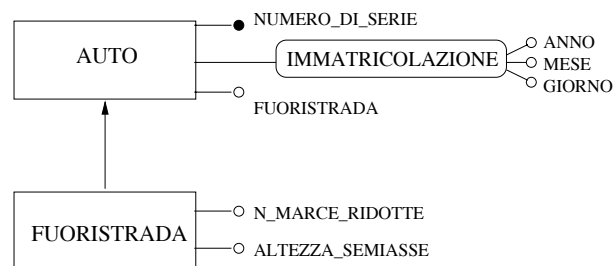
Esempio

Consideriamo il file relativo alle auto di un concessionario. Poichè solo alcune auto sono considerate fuoristrada, si introduce il flag FLAG-FUORISTRADA: il valore è 0 per le auto fuoristrada e 1 per le altre. Se il valore è 0 allora si devono specificare i campi N_MARCE_RIDOTTE e ALTEZZA_SEMIASSE.

```

01  AUTO.
    02 NUMERO_DL SERIE      PIC X(10).
    02 DATA-DI-IMMATRICOLAZIONE.
        03 ANNO      PIC 9(2).
        03 MESE      PIC 9(2).
        03 GIORNO     PIC 9(2).
    02 FLAG-FUORISTRADA    PIC 9.
        88 SI-FUORISTRADA  VALUE 0.
        88 NO-FUORISTRADA  VALUE 1.
    02 N_MARCE_RIDOTTE     PIC 9(2).
    02 ALTEZZA_SEMIASSE    PIC 9(5).
  
```

Poichè l'esempio ricade nel caso 2, lo schema E/R risultante introduce due entità legate da una relazione di generalizzazione:



Regole per i valori di un campo

Regole: l'uso specifico dei campi può essere espresso in termini di regole che definiscono il valore dei record istanza.

Esempio

Consideriamo un file relativo alla strutturazione del bilancio di una compagnia. Sono previsti tre livelli diversi: STATO_CONTABILE, INDICE_DLSETTORE e VOCE_DI_BILANCIO. Ciascuno STATO_CONTABILE raggruppa un insieme di INDICE_DLSETTORE e un INDICE_DLSETTORE raggruppa un insieme di VOCE_DI_BILANCIO. Pertanto, nel file di BILANCIO, il campo INDICE_DLSETTORE (codice di un indice di settore) è significativo solo per le voci di bilancio e il campo STATO_CONTABILE (codice di uno stato contabile) è significativo solo per gli indici di settore.

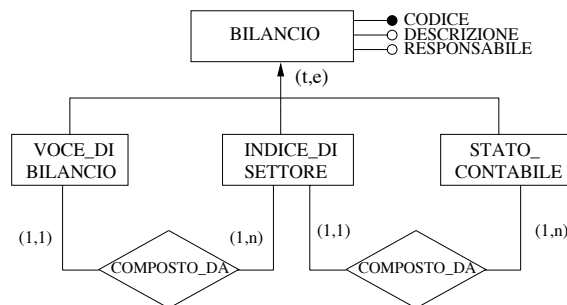
```

01 BILANCIO.
02 CODICE          PIC 9(4).
02 DESCRIZIONE     PIC 9(4).
02 INDICE_DLSETTORE PIC 9(4).
02 STATO_CONTABILE PIC X(5).
02 RESPONSABILE    PIC X(30).
  
```

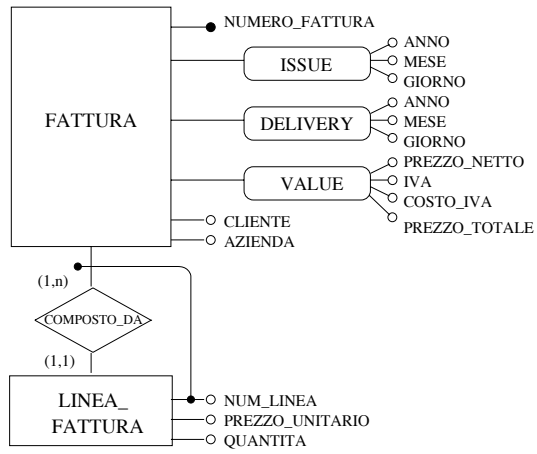
Le seguenti regole stabiliscono l'appartenenza di un record istanza ad uno dei tre livelli:

SE IL CODICE È TRA	ALLORA IL CODICE INDIVIDUA
1 e 999	uno stato contabile
1000 e 2999	un indice di settore
3000 e 9999	una voce di bilancio

Si può concludere che il file è la fusione di tre tipi di record logicamente differenti, connessi gerarchicamente. Un possibile schema E/R è il seguente, dove la struttura gerarchica è espressa da due associazioni:



Il file FATTURA precedentemente descritto è tradotto in uno schema con due entità, FATTURA e LINEA_FATTURA.



Capitolo 2

Elementi di Teoria Relazionale

In questo capitolo vengono presentati i concetti fondamentali del Modello Relazionale dei dati, introdotto da Codd nel 1970, e dell'Algebra Relazionale.

Nella prima parte del capitolo, dopo aver introdotto il concetto matematico di relazione sul quale è basato il modello relazionale, vengono presentati i vincoli di integrità più importanti che possono essere espressi sui valori di una relazione.

La seconda parte introduce gli operatori principali dell'algebra relazionale ed alcuni esempi di interrogazioni di una base di dati relazionale tramite tali operatori.

2.1 Modello Relazionale

Il modello relazionale dei dati è stato introdotto da Codd nel 1970 (E.F. Codd, “A relational model of data for large shared data banks”, *Comm. of the ACM*, 1970) ed è basato sul concetto matematico di relazione.

Dominio : insieme di valori $D = \{v_1, v_2, \dots, v_k\}$

Tupla :

Dati n domini D_1, D_2, \dots, D_n , non necessariamente distinti una *ennupla* (*tupla*) ordinata è definita come

$$t = (v_1, v_2, \dots, v_n), v_i \in D_i, \forall 1 \leq i \leq n$$

Prodotto Cartesiano :

Il *prodotto cartesiano* di n domini D_1, D_2, \dots, D_n , non necessariamente distinti, indicato con $D_1 \times D_2 \times \dots \times D_n$, è l'insieme di tutte le tuple t su D_1, D_2, \dots, D_n .

Relazione :

Una *relazione* R su n domini D_1, D_2, \dots, D_n , non necessariamente distinti, è un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \dots \times D_n$:

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

- Il valore di n viene chiamato **Grado** della relazione
- Il numero di tuple viene chiamata **Cardinalità** della relazione

Esempi :

$$\begin{aligned} D_1 &= \{v_{11}, v_{12}\} & D_2 &= \{v_{21}, v_{22}, v_{23}\} \\ D_1 \times D_2 &= \{(v_{11}, v_{21}), (v_{11}, v_{22}), (v_{11}, v_{23}), (v_{12}, v_{21}), (v_{12}, v_{22}), (v_{12}, v_{23})\} \\ R_1 &= \{(v_{11}, v_{21}), (v_{11}, v_{22})\} & R_2 &= \{\} \\ R_3 &= \{(v_{11}, v_{21}), (v_{11}, v_{22}), (v_{12}, v_{21}), (v_{12}, v_{22})\} \end{aligned}$$

Rappresentazione di relazioni :

Una relazione viene rappresentata generalmente tramite una tabella con un numero di righe pari alla cardinalità e un numero di colonne pari al grado

R_3

v_{11}	v_{21}
v_{11}	v_{22}
v_{12}	v_{21}
v_{12}	v_{22}

Schema ed Istanza

Attributo : Nome dato ad un dominio in una relazione

- si ottiene l'**indipendenza** dall'ordinamento dei domini
- si attribuisce **significato** ai valori del dominio

Schema di relazione :

Uno *schema di relazione* è una coppia costituita dal nome della relazione R e da un insieme di nomi degli attributi $X = (A_1, A_2, \dots, A_n)$, indicato con $R(X)$.

Dato uno schema $R(X)$ si dice anche che lo schema della relazione R è X .

◇ I nomi degli attributi A_i devono essere tutti diversi.

Istanza di relazione :

Una *istanza di relazione* o *relazione* su uno schema $R(A_1, A_2, \dots, A_n)$ è un insieme r di tuple su (A_1, A_2, \dots, A_n)

Schema di base di dati :

È un insieme di schemi di relazioni $\mathbf{R} = \{R_1(X_1), R_2(X_2), \dots, R_n(X_n)\}$

Tutti i nomi di relazione R_i devono essere differenti.

Istanza di base di dati :

Dato uno schema di base di dati $\mathbf{R} = \{R_1(X_1), R_2(X_2), \dots, R_n(X_n)\}$, una *istanza* su \mathbf{R} è un insieme di relazioni

$$\mathbf{r} = \{r_1, r_2, \dots, r_n\}, \text{ dove } r_i \text{ è una relazione su } R_i, \forall 1 \leq i \leq n$$

◇ Differenza tra *tabella* (usata nei DMBS) e *relazione*

- Righe duplicate vs tuple distinte

Notazione :

◇ **Insieme di attributi** : Un insieme di attributi Y dello schema $R(X)$, cioè $Y \subseteq X$, può essere denotato anche con $R.Y$

A denota $\{A\}$, XY denota $\{X\} \cup \{Y\}$,

◇ **Tuple** : Data una tupla t su $R(X)$, un attributo $A \in X$ e un sottoinsieme $Y \subseteq X$

$t[A]$, oppure $t.A$, denota il valore di t su A

$t[Y]$, denota la sottotupla di t ottenuta considerando i valori degli attributi in Y

Esempio

Schema di base di dati :

DB_UNIVERSITÀ = { STUDENTE(MATR, NOME, CITTÀ, ACORSO)
 CORSO(CODCOR, NOME, CODDOC)
 DOCENTE(CODDOC, CF, CITTÀ)
 FREQUENZA(MATR, CODCOR) }

Istanza di base di dati :

STUDENTE

MATR	NOME	CITTÀ	ACORSO
M1	Marco Quaranta	SA	1
M2	Giacomo Tedesco	PA	2
M3	Maria Mei	BO	1
M4	Ugo Rossi	MO	2
M5	Sara Neri	MO	2
M6	Agata Verdi	MI	1

CORSO

CODICE	NOME	CODDOC
C1	Fisica 1	D1
C2	Analisi Matematica 1	D2
C3	Fisica 2	D1
C4	Analisi Matematica 2	D2
C5	Meccanica Razionale	D4

DOCENTE

CODICE	CF	CITTÀ
D1	CF1	MO
D2	CF2	BO
D3	CF3	MO
D4	CF4	FI

FREQUENZA

MATR	CODCOR
M1	C1
M1	C3
M2	C1
M2	C2
M3	C1
M3	C2
M3	C3
M3	C4

Chiavi

- ◇ Informalmente, per *chiave* di una relazione si intende un sottoinsieme dei suoi attributi che identifica univocamente ogni tupla della relazione stessa.
- ◇ Dato uno schema di relazione $R(X)$, un sottoinsieme di attributi $K \subseteq X$ è detto *chiave* dello schema di relazione $R(X)$ se e solo se per ogni relazione r su $R(X)$ valgono le seguenti proprietà:
 1. **Univocità** : $\forall t_1, t_2 \in r, K[t_1] = K[t_2] \implies t_1 \equiv t_2$
cioè, non esistono due tuple distinte di r con lo stesso valore della chiave
 2. **Minimalità** : $\forall A_i \in K, K - A_i$ non verifica la proprietà 1.
cioè, non esiste un sottoinsieme proprio di K con la proprietà di univocità.
- ◇ Un insieme di attributi $Y \subseteq X$ che contiene in modo stretto una chiave K , $Y \supset K$, è detto **superchiave** dello schema di relazione $R(X)$.
In altri termini, una superchiave soddisfa **solo** la proprietà 1. di Univocità.
- ◇ Per ogni schema di relazione $R(X)$, l'insieme X è una superchiave in quanto una relazione, essendo un insieme, contiene tuple distinte.
Quindi, ogni schema di relazione $R(X)$ ha almeno una chiave.
- ◇ Uno schema $R(X)$ può avere più chiavi dette **chiavi candidate**
Tra le chiavi candidate ne viene scelta una detta **chiave primaria**
Le rimanenti chiavi vengono dette **chiavi alternative**

Notazione :

- ◇ La **chiave primaria** di uno schema $R(X)$ si indica sottolineando gli attributi che la compongono: $R(\underline{K_1}, \underline{K_2}, \dots, \underline{K_m}, A_2, \dots, A_n)$
- ◇ Una **chiave alternativa** di uno schema $R(X)$ è riportata di seguito allo schema, contraddistinta dalla parola chiave **AK**:
 $R(X)$
AK: K_1, \dots, K_m

Esempio :

DOCENTE (CODDOC, CF, CITTÀ)

AK: CF

Valori nulli e Vincolo di Entity Integrity

Valori nulli :

Ogni dominio di relazione viene esteso con un particolare valore, detto *valore nullo* e denotato con null, che rappresenta assenza di informazione. In questo modo è possibile introdurre nelle relazioni anche tuple in cui il valore di uno o più attributi non è disponibile.

◇ Ad esempio, con riferimento allo schema di relazione

DOCENTE (CODICE, CF, NOME, CITTA_DI_NASCITA)

si possono elencare vari casi in cui si deve inserire nella relazione DOCENTE una tupla il cui valore di un attributo non è disponibile:

- CITTA_DI_NASCITA del docente è sconosciuto
- il CF non è previsto per docenti di determinati paesi
- ...

◇ Il gruppo di standardizzazione ANSI (*American National Standard Institute*) ha specificato 14 diverse interpretazione per il valore null.

◇ Nei sistemi relazionali è possibile specificare se un attributo può o meno assumere il valore null.

Vincoli di Integrità :

I *vincoli* (o *regole*) di *integrità* stabiliscono condizioni di correttezza delle informazioni nella base di dati.

◇ La stessa dichiarazione di chiave K di uno schema di relazione R è una dichiarazione di vincolo di integrità in quanto stabilisce l'univocità dei valori assunti dagli attributi di K .

◇ In presenza di valori nulli, non sarebbe quindi possibile controllare l'univocità dei valori assunti dagli attributi di una chiave. Per questo motivo viene imposto il seguente vincolo.

Vincolo di Entity Integrity :

Tale vincolo stabilisce che gli attributi che costituiscono la chiave primaria di una relazione non possono assumere valore nullo.

◇ Formalmente, un'istanza r di uno schema di relazione R con chiave primaria K_1, K_2, \dots, K_m , $R(\underline{K_1}, \underline{K_2}, \dots, \underline{K_m}, A_2, \dots, A_n)$, soddisfa il vincolo di Entity integrity se e solo se

$$\forall t \in r, t[K_i] \neq \text{null}, \forall 1 \leq i \leq m$$

Vincolo di Integrità Referenziale

- ◇ Nel Modello Relazionale, i riferimenti tra le tuple delle relazioni vengono stabiliti tramite i valori assunti dagli attributi nelle tuple.

Ad esempio, nella relazione CORSO, il docente di “Fisica1” viene stabilito assegnando all’attributo CODDOC il valore D1, che corrisponde al valore della chiave CODICE, nella relazione DOCENTE, della tupla (D1,CF1,MO).

Il vincolo di integrità referenziale assicura che quando in una tupla si utilizza il valore di un attributo per riferirsi ad un’altra tupla, quest’ultima sia una tupla esistente.

- ◇ In uno schema di base di dati \mathbf{R} un *vincolo di integrità referenziale* viene dichiarato specificando:

Foreign Key o Chiave Esterna : insieme di attributi $FK = \{FK_1, FK_2, \dots, FK_n\}$ di uno schema di relazione $R_1 \in \mathbf{R}$

Chiave della Relazione riferita : schema di relazione $R_2 \in \mathbf{R}$, non necessariamente distinto da R_1 , con una chiave $K = \{K_1, K_2, \dots, K_m\}$, con $m = n$.

- ◇ Informalmente, un’istanza $r = \{r_1, r_2, \dots\}$ su \mathbf{R} soddisfa il vincolo di integrità referenziale se i valori sulla foreign key FK di ciascuna tupla di r_1 sono valori della chiave K di r_2 , o sono valori nulli.
- ◇ Formalmente, un’istanza $r = \{r_1, r_2, \dots\}$ su \mathbf{R} , soddisfa il vincolo di integrità referenziale se e solo se

$$\forall t_1 \in r_1, (t_1[FK_i] = \text{null} \vee \exists t_2 \in r_2 : t_1[FK_i] = t_2[K_i]), \forall 1 \leq i \leq n$$

Notazione : $R_1(X)$

FK: FK_1, \dots, FK_n **REFERENCES** $R_2(K_1, \dots, K_n)$

- Se K_1, \dots, K_n è la chiave primaria di R_2 può essere omessa:

FK: FK_1, \dots, FK_n **REFERENCES** R_2

Esempio : CORSO (CODICE, NOME, CODDOC)

FK: CODDOC **REFERENCES** DOCENTE

Istanza Legale di Base di Dati :

- ◇ Dato uno schema di basi di dati $\mathbf{R} = \{R_1(X_1), R_2(X_2), \dots, R_n(X_n)\}$, un’istanza $\mathbf{r} = \{r_1, r_2, \dots, r_n\}$ su \mathbf{R} tale che

- ciascuna relazione r_i soddisfa il vincolo di entity integrity
- \mathbf{r} soddisfa tutti i vincoli di integrità referenziale imposti su \mathbf{R}

verrà detta istanza **legale** della base di dati \mathbf{R} .

2.2 Algebra Relazionale

- ◇ L'*Algebra Relazionale* è un insieme di operatori che si applicano alle relazioni e restituiscono relazioni.
- ◇ Tramite le *espressioni* dell'*Algebra Relazionale* è possibile formulare *interrogazioni* anche complesse sulla base di dati.
- ◇ L'*Algebra Relazionale* è composta da cinque operatori di base a partire dai quali si possono definire altri operatori, detti *derivati*.
- ◇ **Operatori Base :**
 - **Unari :**
 - Selezione
 - Proiezione
 - **Binari :**
 - Unione
 - Differenza
 - Prodotto Cartesiano
- ◇ **Principali Operatori Derivati :**
 - Intersezione
 - Join
 - Divisione

Operatori Unari

Selezione : Operatore σ

- ◇ Data una relazione r con schema X e un *predicato di selezione* F
- F è una espressione booleana di predicati semplici p
 - un predicato semplice p è il confronto tra due espressioni che utilizzano nomi di attributi, costanti e operazioni aritmetiche
- l'operazione di selezione $\sigma_F(r)$ ha come risultato una relazione con schema X definita dal sottoinsieme di tuple di r che soddisfano il predicato F

$$\sigma_F(r) = \{t \mid t \in r, F(t) = \text{true}\}$$

Es. “Studenti del secondo anno di corso”

$\sigma_{\text{ACORSO}=2}(\text{STUDENTE})$

MATR	NOME	CITTÀ	ACORSO
M2	Giacomo Tedesco	PA	2
M4	Ugo Rossi	MO	2
M5	Sara Neri	MO	2

Proiezione : Operatore π

- ◇ Data una relazione r con schema X e un insieme di attributi $Y \subseteq X$, il risultato dell'operazione di proiezione $\pi_Y(r)$ è una relazione con schema Y definita dall'insieme di sottotuple di r ottenute considerando solo i valori su Y :

$$\pi_Y(r) = \{t[Y] \mid t \in r\}$$

Es. “Città e anno di corso degli studenti”

$\pi_{\text{CITTÀ}, \text{ACORSO}}(\text{STUDENTE})$

CITTÀ	ACORSO
SA	1
PA	2
BO	1
MO	2

Es. “Città degli studenti del secondo anno di corso”

$\pi_{\text{CITTÀ}}(\sigma_{\text{ACORSO}=2}(\text{STUDENTE}))$

CITTÀ
PA
MO

Operatori Binari

Unione : Operatore \cup

- ◇ Date due relazioni r e s con lo stesso schema X , il risultato dell'operazione di unione $r \cup s$ è una relazione che ha come schema X ed è definita dall'unione delle tuple di r con le tuple di s

$$r \cup s = \{t \mid t \in r \vee t \in s\}$$

Es. “Città di studenti o docenti”

$$\pi_{\text{CITTA}}(\text{STUDENTE}) \cup \pi_{\text{CITTA}}(\text{DOCENTE})$$

Differenza : Operatore $-$

- ◇ Date due relazioni r e s con lo stesso schema X , l'operazione di unione $r \cup s$ è una relazione che ha come schema X ed è definita dalla differenza tra le tuple di r e quelle di s

$$r - s = \{t \mid t \in r \wedge t \notin s\}$$

Es. “Città di studenti ma non di docenti”

$$\pi_{\text{CITTA}}(\text{STUDENTE}) - \pi_{\text{CITTA}}(\text{DOCENTE})$$

Intersezione : Operatore \cap

- ◇ L'operatore di intersezione \cap è un **operatore derivato** definito come

$$r \cap s = r - (r - s) = \{t \mid t \in r \wedge t \notin (r - s)\} = \{t \mid t \in r \wedge t \in s\}$$

Es. “Città di studenti e di docenti”

$$\pi_{\text{CITTA}}(\text{STUDENTE}) \cap \pi_{\text{CITTA}}(\text{DOCENTE})$$

Prodotto Cartesiano e Join

Prodotto Cartesiano : Operatore \times

- ◇ Date due relazioni r e s con schema $R(X)$ e $S(Y)$, il risultato dell'operazione di prodotto cartesiano $r \times s$ è una relazione che ha come schema $R.X \cup S.Y$ ed è definita dall'insieme di tuple

$$r \times s = \{t \mid t = t_R t_S : t_R \in r \wedge t_S \in s\}$$

dove $t_R t_S$ indica la concatenazione della tupla t_R e della tupla t_S .

- Nello schema del prodotto cartesiano $R.X \cup S.Y$, un attributo A che è solo in $R.X$ ($S.Y$) può essere indicato semplicemente con A .

Theta-Join : Operatore \bowtie_F

- ◇ L'operazione di *join* serve per combinare tuple prese da due o più relazioni sulla base di condizioni espresse sugli attributi delle tuple.
- ◇ Il Theta-join tra r e s definisce un sottoinsieme del prodotto cartesiano $r \times s$ ottenuto tramite una operazione di selezione.
- ◇ Date due relazioni r e s con schema $R(X)$ e $S(Y)$, e un *predicato di join* F costituito da una espressione booleana di predicati di confronto $A \Theta B$, dove $A \in X$, $B \in Y$ e Θ è un operatore di confronto, il Theta-join tra r e s è definito come

$$r \bowtie_F s = \sigma_F(r \times s)$$

◇ Equijoin

I predicati di confronto sono esclusivamente predicati di uguaglianza

Naturaljoin : Operatore \bowtie

- ◇ Date due relazioni r e s con schemi $R(X)$ e $S(Y)$, il *naturaljoin* (o *join naturale*) tra r e s , indicato con $r \bowtie s$, è il risultato dell'equijoin tra r e s su tutti gli attributi comuni a X e Y proiettato sull'unione degli attributi dei due schemi (XY)

$$r \bowtie s = \pi_{X \cup Y}(r \bowtie_{\bigwedge_{A_i \in X \cap Y} (R.A_i = S.A_i)} s)$$

SemiJoin e OuterJoin

Semijoin : Operatore \bowtie

- ◇ Date due relazioni r e s con schemi $R(X)$ e $S(Y)$ il semijoin da s a r , indicato con $r \bowtie s$ è la proiezione su X del join naturale di r e s

$$r \bowtie s = \pi_X(r \bowtie s)$$

- ◇ Si può verificare che $r \bowtie s = r \bowtie (\pi_{X \cap Y} s)$

OuterJoin :

- ◇ Il risultato di un join $r \bowtie s$ comprende solo le tuple t_R di r (t_S di s) che possono essere messe in corrispondenza, in base al predicato di join, con una tupla t_S di s (t_R di r).

Una tupla t_R di r (t_S di s) che non partecipa a tale corrispondenza, e che quindi non contribuisce al join, è detta *dangling*.

Più precisamente, una tupla t_R di r è detta *dangling* se non esiste $t \in r \bowtie s$ tale che $t[X] = t_R$ (stessa cosa per t_S di s).

- ◇ Gli operatori di OuterJoin servono per includere nel risultato del join anche le tuple dangling: tale tuple sono concatenate con tuple composte da valori nulli

left-outerjoin : $r \bowtie\!=\! s$

comprende le tuple dangling t_R di r

right-outerjoin : $r \bowtie\!=\! s$

comprende le tuple dangling t_S di s

full-outerjoin : $r \bowtie\!=\! s$

comprende sia le tuple dangling t_R di r che le tuple dangling t_S di s

Esempi di Join

Es. “Studenti e docenti della stessa città”

$\text{STUDENTE} \bowtie_{\text{STUDENTE.CITTÀ}=\text{DOCENTE.CITTÀ}} \text{DOCENTE}$

MATR	NOME	STUDENTE. CITTÀ	ACORSO	CODICE	CF	DOCENTE. CITTÀ
M3	Maria Mei	BO	1	D2	CF2	BO
M4	Ugo Rossi	MO	2	D1	CF1	MO
M4	Ugo Rossi	MO	2	D3	CF3	MO
M5	Sara Neri	MO	2	D1	CF1	MO
M5	Sara Neri	MO	2	D3	CF3	MO

Oppure, usando il join naturale

$\text{STUDENTE} \bowtie \text{DOCENTE}$

MATR	NOME	CITTÀ	ACORSO	CODICE	CF
M3	Maria Mei	BO	1	D2	CF2
M4	Ugo Rossi	MO	2	D1	CF1
M4	Ugo Rossi	MO	2	D3	CF3
M5	Sara Neri	MO	2	D1	CF1
M5	Sara Neri	MO	2	D3	CF3

Es. “Studenti che frequentano i corsi (almeno uno) del docente D2”

$\text{FREQUENZA} \bowtie \sigma_{\text{CODDOC}=\text{D2}}(\text{CORSO})$

MATR	CODICE	NOME	CODDOC
M2	C2	Analisi Matematica 1	D2
M3	C2	Analisi Matematica 1	D2
M3	C4	Analisi Matematica 2	D2

La relazione risultante contiene solo la matricola degli studenti; per ottenere tutto lo schema della relazione STUDENTE:

$\text{STUDENTE} \bowtie \pi_{\text{MATR}}(\text{FREQUENZA} \bowtie (\sigma_{\text{CODDOC}=\text{D2}}(\text{CORSO})))$

oppure

$\text{STUDENTE} \bowtie (\text{FREQUENZA} \bowtie \sigma_{\text{CODICE}=\text{'D1'}}(\text{CORSO}))$

MATR	NOME	CITTÀ	ACORSO
M2	Giacomo Tedesco	PA	2
M3	Maria Mei	BO	1

Divisione

- ◇ Informalmente, date due relazioni r e s , l'operazione di *divisione* tra r (*dividendo*) e s (*divisore*), indicata con $r \div s$, serve per individuare le tuple di r associate a *tutte* le tuple di s
- ◇ Date due relazioni r e s con schemi $R(X)$ e $S(Y)$ tali che $Y \subset X$, l'operazione di *divisione* tra r e s , $r \div s$, ha come risultato una relazione che ha schema $(X - Y)$ ed è definita da

$$r \div s = \{t_D \mid \forall t_S \in S, t_D t_S \in r\}$$

Es. “Studenti che frequentano *tutti* i corsi del docente D1”

FREQUENZA

MATR	CODCOR
M1	C1
M1	C3
M2	C1
M2	C2
M3	C1
M3	C2
M3	C3
M3	C4

$$\div \pi_{\text{CODCOR}} (\sigma_{\text{CODDOC}='D1'} (\text{CORSO}))$$

CODCOR
C1
C3

- ◇ L'operatore divisione \div può essere derivato dagli operatori di base:

$$r \div s = \pi_{X-Y}(r) - \pi_{X-Y}((\pi_{X-Y}(r) \times s) - r)$$

Verifichiamo tale equivalenza sull'esempio precedente:

$\pi_{X-Y}(r) \times s$

MATR	CODCOR
M1	C1
M1	C3
M2	C1
M2	C3
M3	C1
M3	C3

$(\pi_{X-Y}(r) \times s) - r$

MATR	CODCOR
M2	C3

$\pi_{X-Y}((\pi_{X-Y}(r) \times s) - r)$

MATR
M2

$\pi_{X-Y}(r) - \pi_{X-Y}((\pi_{X-Y}(r) \times s) - r)$

MATR
M1
M3

Esercizio

Consideriamo il seguente schema relazionale

AUTO(CODAUTO,COSTRUTTORE)

ACCESSORIO(CODACC,DESCRIZIONE)

INSTALLABILE(CODAUTO,ANNOPROD,CODACC)

FK: CODAUTO REFERENCES AUTO

FK: CODACC REFERENCES ACCESSORIO

L'accessorio CODACC è installabile sull'auto CODAUTO prodotta nell'anno ANNOPROD.

e una sua istanza:

AUTO		ACCESSORIO	
CODAUTO	COSTRUTTORE	CODAUTO	DESCRIZIONE
Tipo	Fiat	PSci9	PortaSci
Uno	Fiat	FNeb11	FariAntiNebbia
Fiesta	Ford	VSport32	VolanteSportivo
Escort	Ford	RLega6	RuoteInLega

INSTALLABILE		
CODAUTO	ANNOPROD	CODACC
Tipo	1989	PSci9
Tipo	1990	FNeb11
Fiesta	1990	PSci9
Uno	1989	VSport32
Uno	1989	PSci9
Uno	1990	PSci9
Uno	1991	FNeb11
Escort	1991	PSci9
Escort	1992	RLega6

a) selezionare i dati relativi agli accessori installabili su *almeno* un'auto 'Fiat';

Per determinare il codice degli accessori installabili su *almeno* un'auto 'Fiat' occorre fare il join naturale tra la relazione INSTALLABILE e la relazione AUTO ristretta con la condizione COSTRUTTORE='Fiat'.

$$(INSTALLABILE \bowtie \sigma_{COSTRUTTORE='Fiat'}(AUTO))$$

CODAUTO	ANNOPROD	CODACC	COSTRUTTORE
Tipo	1989	PSci9	Fiat
Tipo	1990	FNeb11	Fiat
Uno	1989	VSport32	Fiat
Uno	1989	PSci9	Fiat
Uno	1990	PSci9	Fiat
Uno	1991	FNeb11	Fiat

In tale join c'è solo il codice degli accessori: per ottenere i dati (cioè tutti gli attributi) relativi a tali accessori, possiamo scrivere

$$\text{ACCESSORIO} \bowtie (\pi_{\text{CODACC}} (\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE}='Fiat'} (\text{AUTO})))$$

oppure, in modo più sintetico, utilizzando l'operatore di semijoin

$$\text{ACCESSORIO} \ltimes (\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE}='Fiat'} (\text{AUTO}))$$

- b)** selezionare i dati relativi agli accessori che non sono installabili su nessuna auto 'Fiat';

Gli accessori che non sono installabili su nessuna auto 'Fiat' si ottengono sottraendo dall'insieme di tutti gli accessori quelli installabili su almeno un'auto 'Fiat', ricavati in precedenza:

$$\text{ACCESSORIO} - \text{ACCESSORIO} \ltimes (\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE}='Fiat'} (\text{AUTO}))$$

La differenza tra i due insiemi di tuple della relazione ACCESSORI può essere fatta considerando solo la chiave di tale relazione, CODACC:

$$\pi_{\text{CODACC}} (\text{INSTALLABILE})$$

CODACC
PSci9
FNeb11
VSport32
RLega16

$$\pi_{\text{CODACC}} ((\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE}='Fiat'} (\text{AUTO})))$$

CODACC
PSci9
FNeb11
VSport32

però a questo punto occorre fare ancora una operazione di join per ricavare tutti gli attributi della relazione ACCESSORI:

$$\text{ACCESSORIO} \bowtie (\pi_{\text{CODACC}} (\text{ACCESSORIO}) - \pi_{\text{CODACC}} (\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE}='Fiat'} (\text{AUTO})))$$

◇ Si noti che l'interrogazione **b)** non può essere risolta con un semplice join.

In particolare, con il seguente join:

$$(\text{INSTALLABILE} \bowtie \sigma_{\text{COSTRUTTORE} <> 'Fiat'} (\text{AUTO}))$$

CODAUTO	ANNOPROD	CODACC	COSTRUTTORE
Fiesta	1990	PSci9	Ford
Escort	1991	PSci9	Ford
Escort	1992	RLega16	Ford

si ottengono i codici degli accessori installabili su *almeno* un auto **non** 'Fiat';

c) selezionare i dati relativi agli accessori installabili su *tutte* le auto 'Fiat';

Questa interrogazione si risolve con un'operazione di divisione, in cui il divisore è costituito dal codice delle auto 'Fiat':

$$\pi_{\text{CODAUTO}}(\sigma_{\text{COSTRUTTORE}='Fiat'}(\text{AUTO}))$$

Per quanto riguarda il dividendo, occorre far riferimento all'installabilità di un accessorio su un'auto, indipendentemente dall'anno di produzione

$$\pi_{\text{CODACC}, \text{CODAUTO}}(\text{INSTALLABILE})$$

Quindi, in definitiva

$$\pi_{\text{CODACC}, \text{CODAUTO}}(\text{INSTALLABILE}) \div \pi_{\text{CODAUTO}}(\sigma_{\text{COSTRUTTORE}='Fiat'}(\text{AUTO}))$$

CODACC	CODAUTO
PSci9	Tipo
FNeb11	Tipo
PSci9	Fiesta
VSport32	Uno
PSci9	Uno
FNeb11	Uno
PSci9	Escort
RLegal6	Escort

CODAUTO
Tipo
Uno

Lo schema risultante di tale divisione è CODACC: per ottenere i dati dello schema di ACCESSORI occorre fare il join con tale relazione:

$$\text{ACCESSORIO} \bowtie (\pi_{\text{CODACC}, \text{CODAUTO}}(\text{INSTALLABILE}) \div \pi_{\text{CODAUTO}}(\sigma_{\text{COSTRUTTORE}='Fiat'}(\text{AUTO})))$$

◇ Si noti che se si considerasse come dividendo tutta la relazione INSTALLABILE

$$\text{INSTALLABILE} \div \pi_{\text{CODAUTO}}(\sigma_{\text{COSTRUTTORE}='Fiat'}(\text{AUTO}))$$

CODAUTO	ANNOPROD	CODACC
Tipo	1989	PSci9
Tipo	1990	FNeb11
Fiesta	1990	PSci9
Uno	1989	VSport32
Uno	1989	PSci9
Uno	1990	PSci9
Uno	1991	FNeb11
Escort	1991	PSci9
Escort	1992	RLegal6

CODAUTO
Tipo
Uno

si otterrebbero i dati relativi agli accessori che nello *stesso anno di produzione* sono installabili su tutte le auto 'Fiat' ;

Capitolo 3

Progettazione Logica

Con il termine *progettazione logica* si intende la traduzione di uno schema disegnato tramite un modello concettuale in uno schema disegnato tramite un modello logico. Il presente capitolo contiene le fasi principali della progettazione logica a partire da schemi concettuali realizzati tramite il modello E/R. Questa fase è indispensabile in quanto non esistono DBMS in grado di operare direttamente sugli oggetti di uno schema E/R.

La prima fase consiste in una semplificazione dello schema E/R (eliminazione di gerarchie e identificazioni esterne, normalizzazione di attributi composti o multipli, scelta di chiavi primarie) basata su criteri di ottimizzazione dello schema. Il risultato è ancora uno schema E/R, quindi, questa fase risulta indipendente dal modello logico scelto per l'implementazione della base di dati.

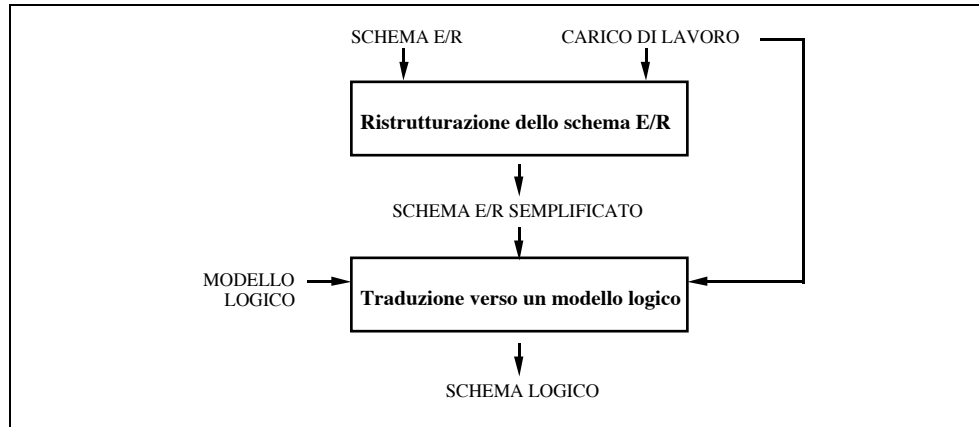
La seconda fase è riferita ad un particolare modello logico, il modello relazionale, e consiste nella vera e propria trasformazione dello schema E/R semplificato in uno schema relazionale. Verranno illustrate in dettaglio le regole di traduzione di entità e associazioni in schemi di relazioni e le ulteriori ottimizzazioni basate sulle caratteristiche del modello relazionale.

3.1 Progetto logico di alto livello con il modello E/R

- ◇ Non esistono DBMS in grado di operare direttamente sugli oggetti di uno schema E/R: è quindi necessario tradurli in modelli di dati supportati da DBMS, quali il gerarchico, reticolare, relazionale.

Fasi della progettazione logica

- 1) ristrutturazione dello schema E/R
 - è indipendente dal modello logico
 - si basa su criteri di ottimizzazione dello schema
 - consiste nella eliminazione di gerarchie e identificazioni esterne, normalizzazione di attributi composti o multipli, scelta di chiavi primarie.
- 2) traduzione verso il modello logico
 - è riferita ad un particolare modello logico: modello relazionale
 - ulteriori ottimizzazioni basate sulle caratteristiche del modello logico vengono effettuate
 - consiste nella traduzione di entità e associazioni in schemi di relazioni



- ◇ Dopo la prima fase, lo schema E/R è costituito soltanto da entità associazioni e attributi semplici.
- ◇ Ogni trasformazione impoverisce semanticamente lo schema; la semantica persa resterà sotto forma di vincoli di integrità che governeranno l'uso delle relazioni.
- ◇ In entrambe le fasi si deve considerare il carico di lavoro previsto, in termini di dimensione dei dati e caratteristiche delle operazioni.
- ◇ Si possono individuare alcune regole intuitive da seguire in questa fase:
- le proprietà logiche sono comunque primarie rispetto ai motivi di efficienza
 - sullo stesso concetto vanno mantenute le informazioni che verranno di frequente consultate insieme
 - informazioni che verranno consultate separatamente vanno suddivise su concetti distinti
 - l'incidenza di valori nulli per attributi opzionali va limitata

Carico di Lavoro

Definizione: Il carico di lavoro sul DB è rappresentato sia dalla dimensione dei dati che dalle operazioni più significative che si stima saranno eseguite sul DB.

Regola 20-80: il 20% delle operazioni produce l'80% del carico.

- ◇ Si fa uso di informazioni non direttamente legate alla struttura fisica dei dati, in quanto questa non è ancora data.

Volume dei dati:

numero medio di istanze di ogni entità e associazione
cardinalità e dimensioni di ciascun attributo
percentuali di copertura di gerarchie

Tabella dei volumi:

Concetto	Tipo	Volume dei dati
----------	------	-----------------

Dove nel campo Concetto compare il nome, nel campo tipo il tipo che può essere E (entità), R (associazione) o A (attributo).

Descrizione delle operazioni:

tipo dell'operazione: Interattiva o Batch

frequenza: numero medio di esecuzioni in un certo periodo di tempo

schema di operazione: frammento dello schema E-R interessato dall'operazione sul quale viene disegnato il "cammino logico" da percorrere per accedere alle informazioni di interesse.

Tabella delle operazioni:

Operazione	Tipo (I o B)	Frequenza
------------	--------------	-----------

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

Tabella degli accessi:

Concetto	Accessi	Tipo
----------	---------	------

Dove nel campo accessi compare il numero degli accessi.

- ◇ Le operazioni di scrittura (S) sono generalmente più onerose di quelle in lettura (L): il peso degli accessi in scrittura è doppio di quello delle letture.

Dati derivati

Definizione: Un dato derivato è un dato che può essere ottenuto attraverso una serie di operazioni da altri dati.

Sulla base delle operazioni e delle loro frequenze è possibile valutare se è conveniente o meno mantenere nello schema attributi derivati

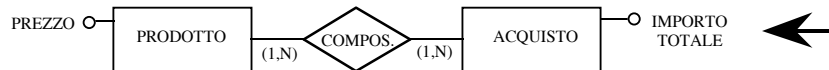
Vantaggio: a tempo di accesso non è richiesta alcuna operazione per ricavare il valore dell'attributo

Svantaggi: occorre eseguire operazioni di aggiornamento per mantenere la consistenza dei dati, si spreca memoria; tuttavia, allo stato attuale dei costi di CPU e memorizzazione si può ritenere in generale trascurabile l'onere dovuto ai maggiori costi di memorizzazione.

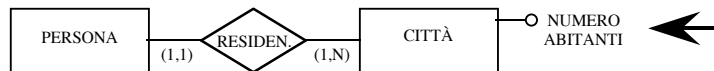
◇ Attributi derivabili da altri attributi della stessa entità o associazione



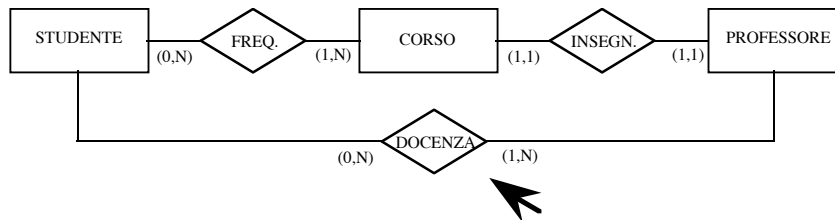
◇ Attributi derivabili da attributi di altre entità o associazioni



◇ Attributi derivabili da operazioni di conteggio di istanze

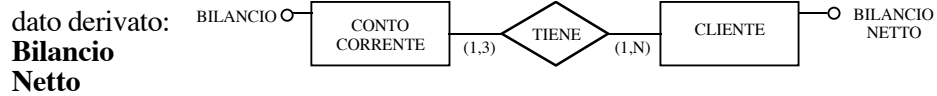


◇ Associazioni derivabili dalla composizione di altre associazioni



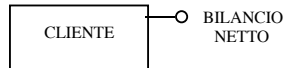
Esempio di Dato derivato

Esempio:

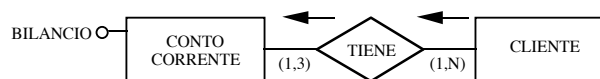


operazione 1: lettura del bilancio netto di un cliente

Con il dato derivato



Senza il dato derivato



operazione 2: deposito su un conto corrente

Con il dato derivato



Senza il dato derivato

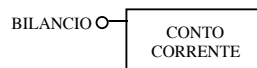


Tabella dei volumi

CONCETTO	TIPO	VOL.
Cliente	E	15000
ContoCorrente	E	20000
Tiene	R	30000

Tabella delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	3000/Giorno
Oper . 2	I	1000/Giorno

Esempio di Dato derivato (2)

Con il dato derivato:

Occupazione di memoria: se ogni valore di “Bilancio Netto” richiede 6 bytes di memoria, la memoria richiesta è di 90 Kbytes.

Operazione 1	CONCETTO	ACC.	TIPO
1 accesso in lettura $1 \cdot 3000 = 3000$ /giorno	Cliente	1	L
Operazione 2	ContoCorrente	1	L
4 accessi in lettura	ContoCorrente	1	S
2.5 accessi in scrittura	Tiene	1.5	L
	Cliente	1.5	L
$9 \cdot 1000 = 9000$ /giorno	Cliente	1.5	S

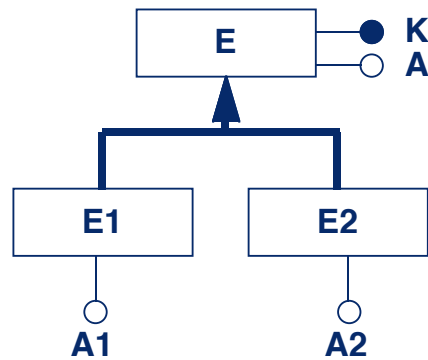
Senza il dato derivato:

Operazione 1	CONCETTO	ACC.	TIPO
5 accesso in lettura	Cliente	1	L
$5 \cdot 3000 = 15000$ /giorno	Tiene	2	L
	ContoCorrente	2	L
Operazione 2	ContoCorrente	1	L
1 accesso in lettura	ContoCorrente	1	S
1 accesso in scrittura			
$3 \cdot 1000 = 3000$ /giorno			

Conclusione: Conviene tenere il dato derivato.

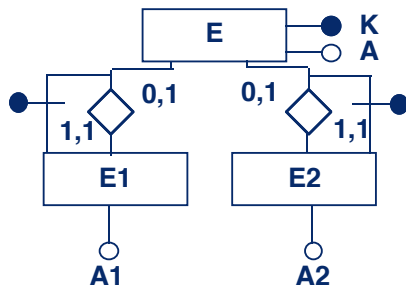
Eliminazione delle gerarchie

- ◇ In questa attività si ottiene uno schema E/R semplificato in cui ogni gerarchia è sostituita da appropriate entità ed associazioni.



- ◇ Le alternative che si presentano sono tre:
- 1) mantenimento delle entità con associazioni,
 - 2) collasso verso l'alto
 - 3) collasso verso il basso
- ◇ Le trasformazioni delle gerarchie possono modificare, in generale, le cardinalità di attributi ed associazioni.
- ◇ l'applicabilità e la convenienza delle soluzioni dipendono dalle proprietà di copertura della gerarchia e dalle operazioni previste.
- ◇ le operazioni influiscono sulla convenienza delle varie soluzioni in base al modo in cui accedono a istanze e attributi della porzione di schema considerata:
- una operazione del tipo “fornire i valori di A e A1 per tutti gli E1” usa “insieme” gli attributi di E ed E1
 - una operazione del tipo “fornire i valori di A per tutti gli E” usa “insieme” le istanze della gerarchia, con i soli attributi della classe generalizzazione

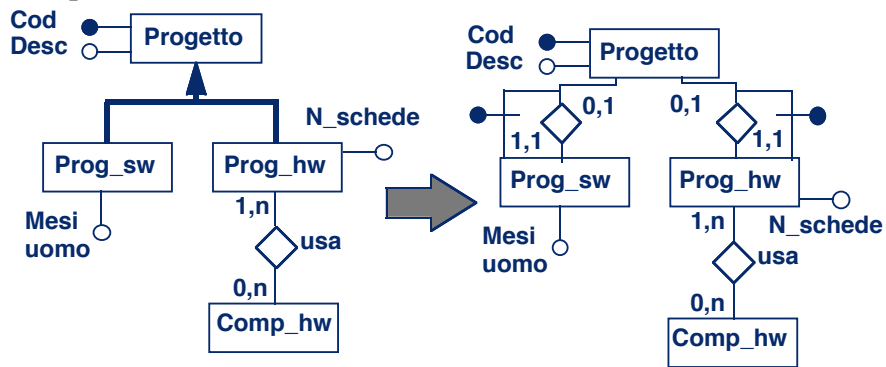
Mantenimento delle entità con associazioni



◇ Tutte le entità vengono mantenute: Le entità *figlie* sono in associazione binaria con l'entità *padre* e sono identificate esternamente.

◇ Questa soluzione è sempre possibile, indipendentemente dalla copertura.

Esempio:

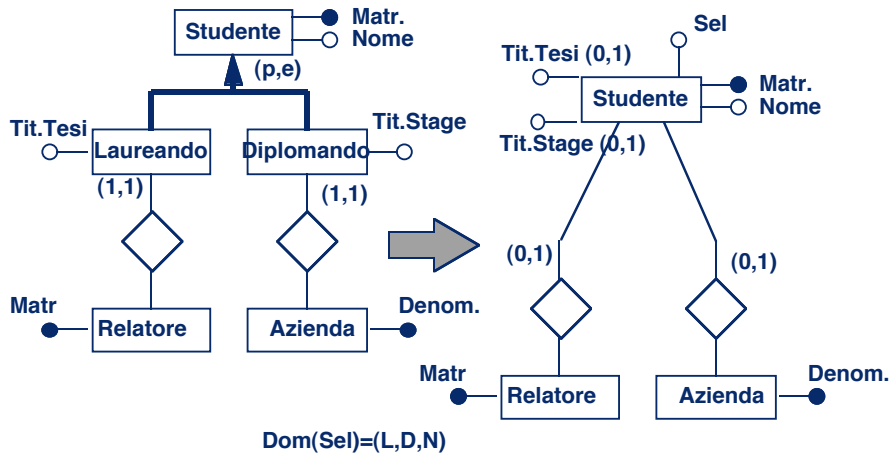


Collasso verso l'alto

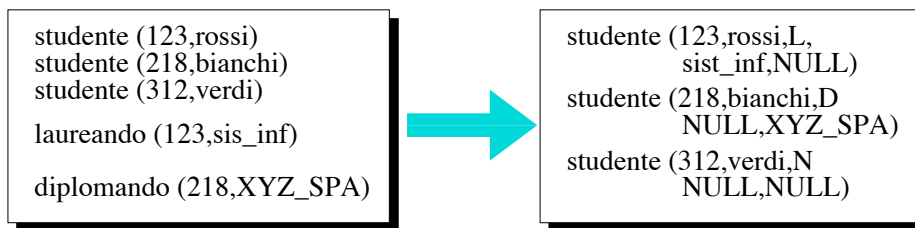


- ◇ Il collasso verso l'alto riunisce tutte le entità figlie nell'entità padre
- ◇ Sel è un attributo selettore che specifica se una singola istanza di E appartiene a una delle N sottoentità. Sel dipende dalla copertura:
 - copertura totale esclusiva (t,e): Sel ha N valori, quante sono le sottoentità
 - copertura parziale esclusiva (p,e): Sel ha N+1 valori; il valore in più serve per le istanze che non appartengono a nessuna sottoentità
 - copertura sovrapposta (o): occorrono tanti selettori Sel_i quante sono le sottoentità, ciascuno a valore booleano, che è vero per ogni istanza di E che appartiene a E_i ; se la copertura è parziale i selettori possono essere tutti falsi, oppure si può aggiungere un selettore
- ◇ Gli attributi obbligatori per le entità figlie divengono opzionali per l'entità padre oppure possono parzialmente sovrapporsi si avrà una certa percentuale di valori nulli
- ◇ le eventuali associazioni connesse alle sottoentità si trasportano su E, le eventuali cardinalità minime diventano 0
- ◇ Questa soluzione favorisce operazioni che consultano insieme gli attributi dell'entità padre e quelli di una entità figlia:
 - in questo caso si accede a una sola entità, anziché a due attraverso una associazione

Collasso verso l'alto esempio

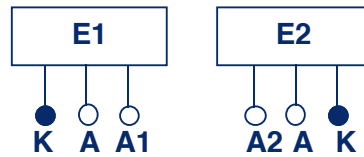


- ◇ esiste una precisa relazione tra il valore del selettore e i campi che possono avere valore diverso da NULL



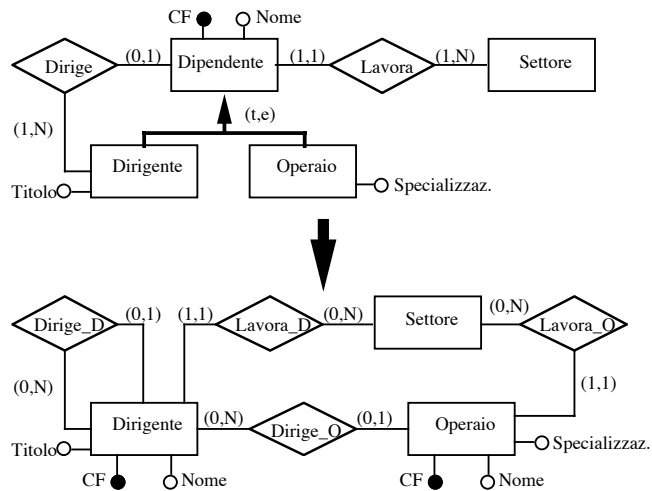
- ◇ per una stima delle percentuali di NULL occorre conoscere le percentuali di laureandi e diplomandi

Collasso verso il basso



- ◇ Il collasso verso il basso elimina l'entità padre trasferendone attributi e associazioni su tutte le entità figlie. Una associazione dell'entità padre si moltiplica, tante volte quante sono le entità figlie
- ◇ Se la copertura non è completa *non si può fare*
 - non si saprebbe dove mettere le istanze di E che non sono istanze né di E1, né di E2
- ◇ Se la copertura non è esclusiva *introduce ridondanza*
 - una certa istanza può essere sia in E1 che in E2, rappresentando due volte gli attributi che provengono da E
- ◇ Soluzione interessante in presenza di molti attributi di specializzazione. Favorisce le operazioni in cui si accede separatamente a ciascuna delle entità figlie
 - esempio: fornire i valori di A per tutti gli E1

Esempio:



Ulteriori Trasformazioni (1)

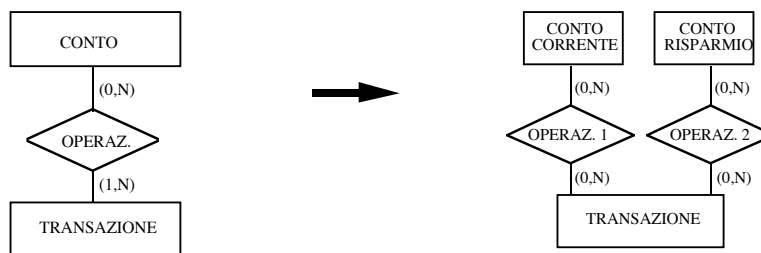
◇ Partizionamento di entità

Un'entità può essere partizionata orizzontalmente (istanze) o verticalmente (attributi) al fine di meglio rispondere alle operazioni previste.

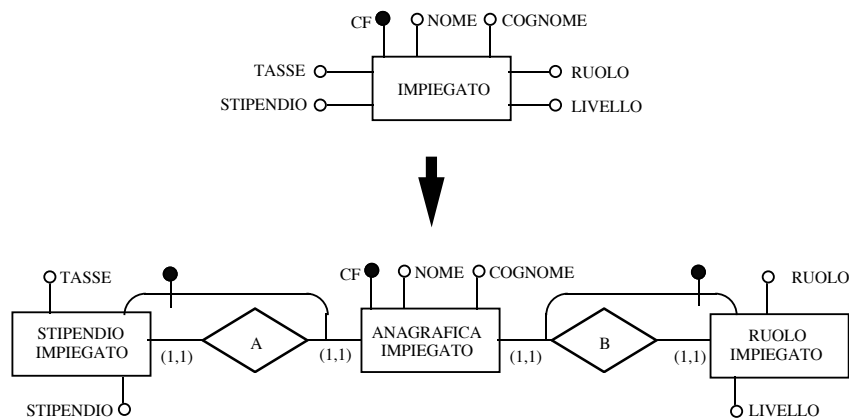
Il partizionamento (orizzontale e/o verticale) può anche essere indotto dalla presenza di diversi privilegi di accesso.

Il partizionamento (orizzontale e/o verticale) è utile soprattutto nel progetto di DBMS distribuiti.

Esempio di partizionamento orizzontale:



Esempio di partizionamento verticale:



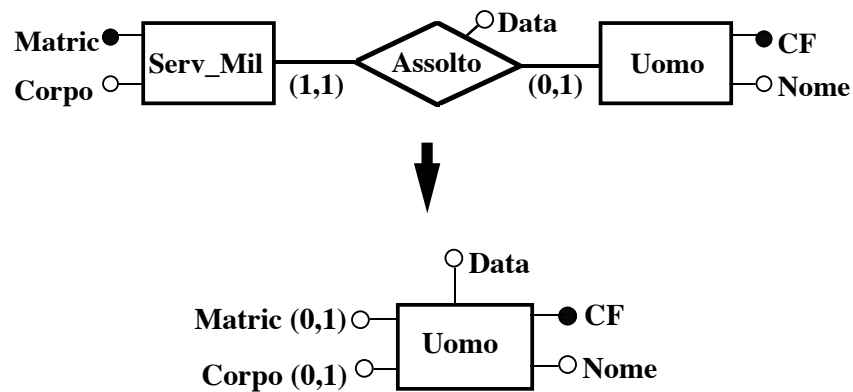
Ulteriori Trasformazioni (2)

◇ Accorpamento di entità:

Due entità partecipanti ad un'associazione uno a uno possono essere accorpate in un'unica entità contenente gli attributi di entrambi.

Per le associazioni uno a molti e molti a molti gli accorpamento di entità generano ridondanze.

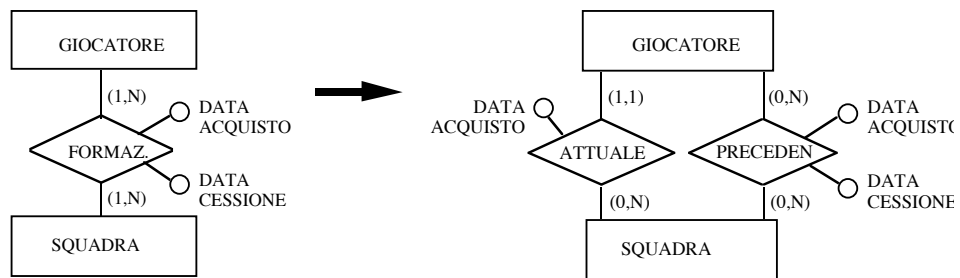
Esempio:



◇ Partizionamento e accorpamento di associazioni:

In questo caso il partizionamento è tipicamente verticale.

Esempio:



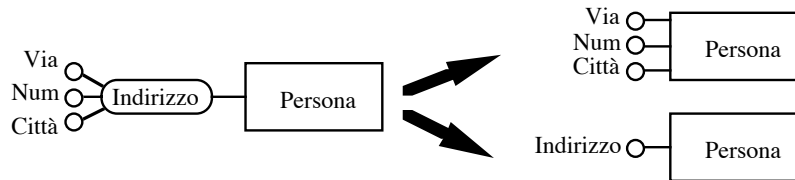
3.2 Progettazione logica relazionale

3.2.1 Trasformazione di attributi e identificatori

Attributi composti : Si hanno due alternative

1. Eliminare l'attributo composto e considerare i suoi componenti come attributi semplici: in questo modo si perde la visione unitaria ma si mantiene l'articolazione dei componenti;
2. Eliminare i componenti e considerare l'attributo come semplice: in questo modo lo schema risulta semplificato, perdendo parte dei dettagli.

◇ **Esempio:**



Attributi ripetuti per le entità :

- ◇ La prima forma normale impone che, se una entità E ha un attributo ripetuto A, si crei una nuova entità EA che ha A ed è collegata ad E

Caso a) un valore può comparire una volta sola nella ripetizione:

l'entità EA ha l'identificatore composto dall'identificatore di E più l'attributo A

Caso b) un valore può comparire più volte nella ripetizione:

l'entità EA ha l'identificatore composto dall'entità E più un attributo identificante sintetico (ad esempio, un numero d'ordine)

Soluzione Particolare: l'entità EA ha l'identificatore composto dall'entità E più l'attributo A; inoltre, l'entità EA ha un attributo che indica il numero di ripetizioni.

Caso c) cardinalità massima nota: $\max\text{-card}(A,E)=K$

in questo caso si può evitare l'introduzione di EA, sostituendo direttamente l'attributo A con i K attributi il cui valore sarà non nullo per i primi H quando la ripetizione è $H < K$

per evitare eccessivi valori nulli, questa soluzione conviene per K piccoli

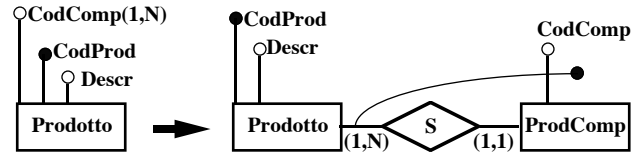
Attributi ripetuti per le associazioni :

Se l'attributo ripetuto fa parte di un'associazione R tra le entità E1, E2, ..., En, la componente di identificazione esterna della nuova entità EA è:

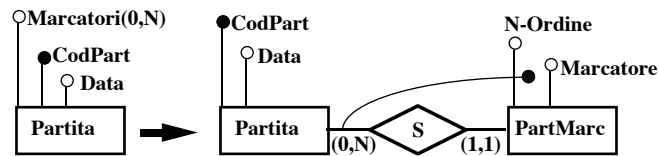
- l'entità E_i tale che $\max\text{-card}(E_i, R) = 1$
- l'insieme di tutte le entità E_1, E_2, \dots, E_n , nel caso in cui E_1, E_2, \dots, E_n è superchiave delle associazioni n-arie

Attributi ripetuti - Esempi

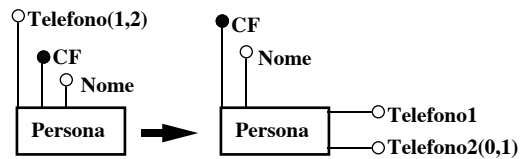
Caso a) un valore può comparire una volta sola nella ripetizione:



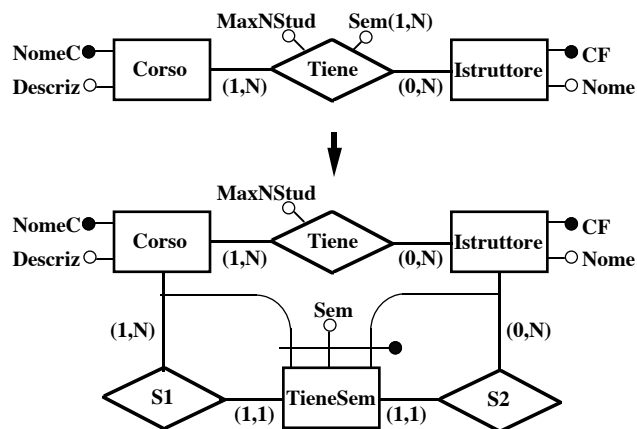
Caso b) un valore può comparire più volte nella ripetizione:



Caso c) cardinalità massima nota:



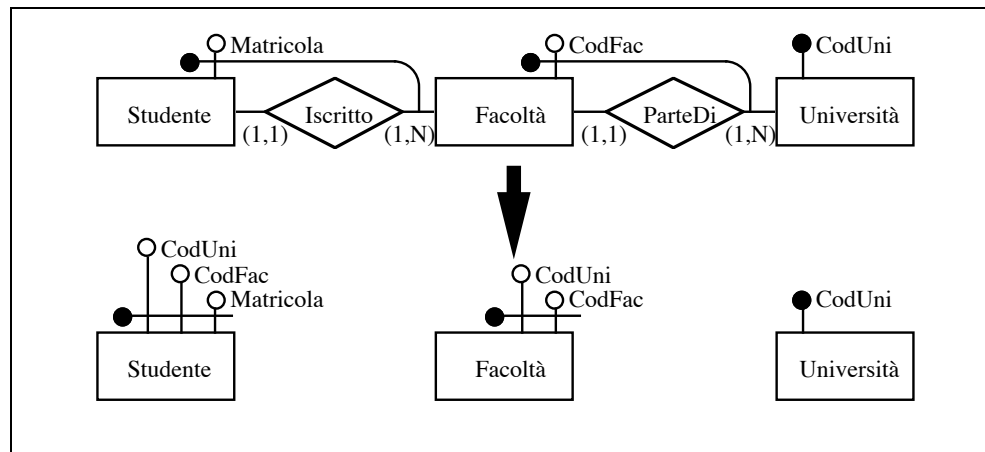
Attributi ripetuti per le associazioni



Eliminazione degli identificatori esterni

◇ Una componente di identificazione esterna di una entità E2 da una entità E1 tramite una associazione R comporta il trasporto dell'identificatore di E1 su E2.

- L'associazione R tra E1 e E2 è automaticamente tradotta e può essere eliminata.
- In presenza di più identificazioni in cascata, è necessario iniziare la propagazione dall'entità che non ha identificazioni esterne



Nell'esempio quindi si parte dall'entità Università, si prosegue su Facoltà e poi su Studente.

Scelta della chiave primaria : È necessario che tra i diversi identificatori di una entità sia designata la chiave primaria.

◇ Criteri euristici:

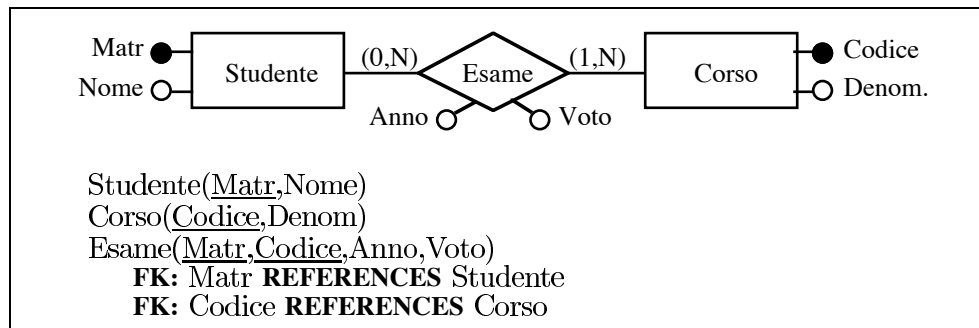
- scegliere la chiave che è usata più frequentemente per accedere direttamente alle istanze dell'entità
- preferire chiavi semplici a chiavi composte
- eventualmente introdurre una *chiave surrogata*, ovvero un attributo che non ha significato e di cui è garantita l'unicità all'interno dell'entità (es. numero progressivo, appositi codici interni di identificazione).

◇ Tutti gli identificatori diversi dalla chiave primaria sono denominati *chiavi alternative*. Per le chiavi alternative occorrerà, in fase di implementazione sul DBMS, prevedere strumenti per garantire l'unicità dei valori.

3.2.2 Traduzione di entità e associazioni

Traduzione standard

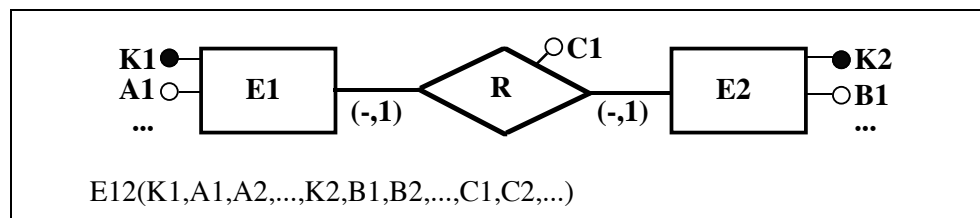
- ◇ Ogni entità è tradotta con una relazione con gli stessi attributi
 - la chiave primaria della relazione è quella dell'entità stessa
 - ogni altro identificatore dell'entità è chiave alternativa della relazione
- ◇ Ogni associazione R tra le entità E1, E2, ..., En, è tradotta con una relazione con gli stessi attributi, cui si aggiungono le chiavi primarie di tutte le entità che essa collega
 - ogni chiave primaria di un'entità Ei tale che $\max\text{-card}(E_i, R)=1$, è una chiave (candidata) della relazione R; altrimenti la chiave della relazione è composta dall'insieme di tutte le chiavi primarie delle entità collegate (o da un sottoinsieme, nel caso che tale insieme denoti una superchiave);
 - le chiavi primarie delle entità collegate sono chiavi straniere (FK) riferite alle corrispondenti entità



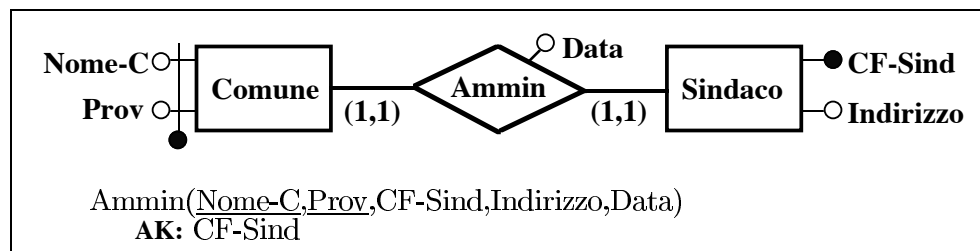
Altre traduzioni

- La traduzione standard è sempre possibile
- La traduzione standard è praticamente l'unica possibilità per le associazioni in cui tutte le entità partecipano con molteplicità maggiore di 1 ($\max\text{-card}(E_i, R) > 1$, per ogni i)
- Altre forme di traduzione dell'associazione sono possibili per casi particolari di cardinalità
- Le altre forme di traduzione tendono a fondere in una stessa relazione entità e associazioni dando origine a un minore numero di relazioni e generano quindi uno schema più semplice
- richiedono un minore numero di operazioni di join per la navigazione attraverso una associazione, ovvero per conoscere le istanze di E1 connesse a E2 tramite R, ma penalizzano le operazioni che consultano soltanto gli attributi di una entità che è stata fusa

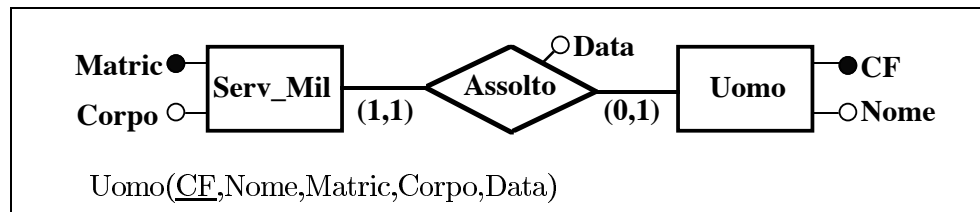
Associazione binaria uno a uno tradotta con una relazione



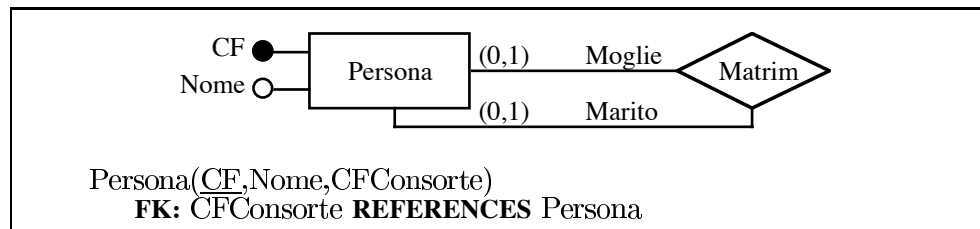
- se l'associazione è obbligatoria per entrambe le entità la chiave primaria può essere indifferentemente K1 o K2 (l'altra sarà chiave alternativa);
ogni altro identificatore delle entità è chiave (alternativa) della relazione.



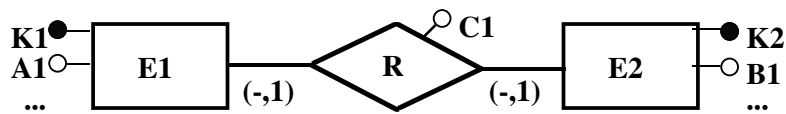
- se l'associazione è parziale per una entità (E1) e obbligatoria per l'altra (E2), la chiave deve essere K1;
saranno possibili valori nulli per gli attributi di E2 e di R.



- se l'associazione è parziale per entrambe, occorre che entrambe le entità abbiano lo stesso identificatore; la chiave può essere indifferentemente K1 o K2 (l'altra sarà foreign key);
saranno possibili valori nulli per gli attributi di E1, di E2 e di R.



Associazione binaria uno a uno tradotta con due relazioni

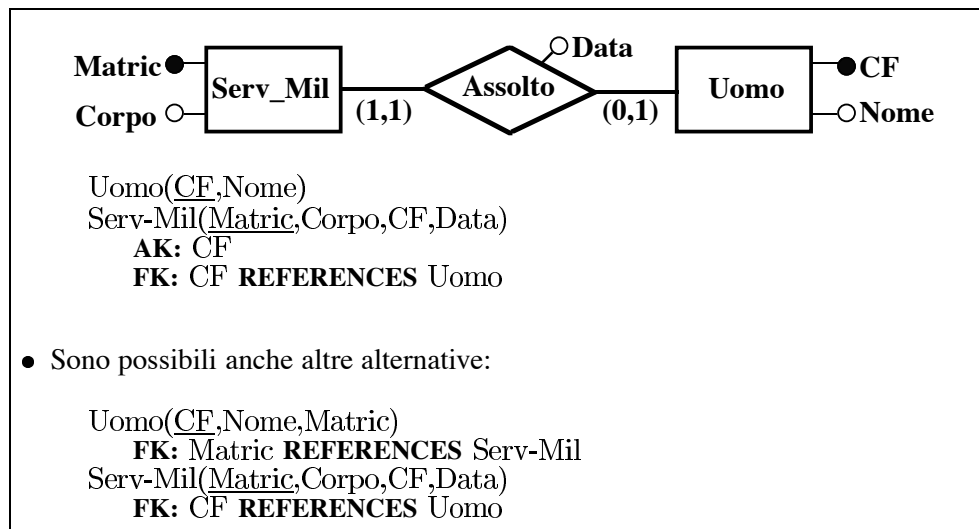


- L'associazione si può compattattare in una delle entità, diciamo E1, includendo in E1 gli attributi di R e la chiave primaria di E2 come foreign key.

E2(K2,B1,B2,...)

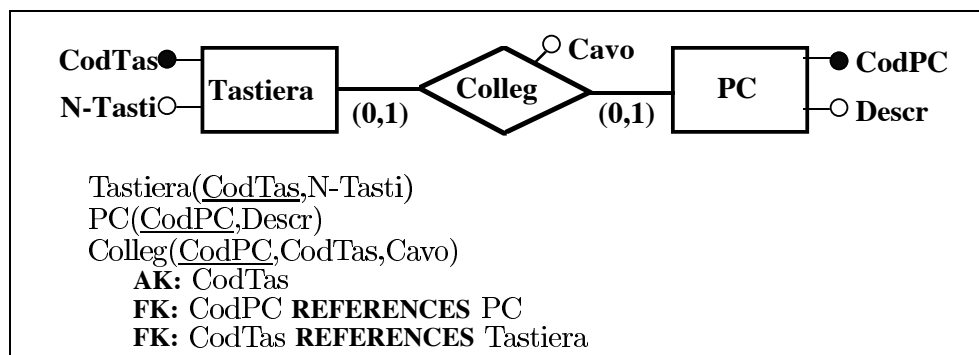
E1(K1,A1,A2,...,K2,C1,C2,...) **FK: K2 REFERENCES E2**

- Per evitare i valori nulli, è preferibile compattare l'associazione in un'entità che partecipa obbligatoriamente.



◇ Associazione binaria uno a uno tradotta con tre relazioni

- È preferibile se l'associazione è parziale per entrambe le entità. Praticamente forzata se, oltre alla parzialità, le due chiavi primarie delle entità hanno domini distinti



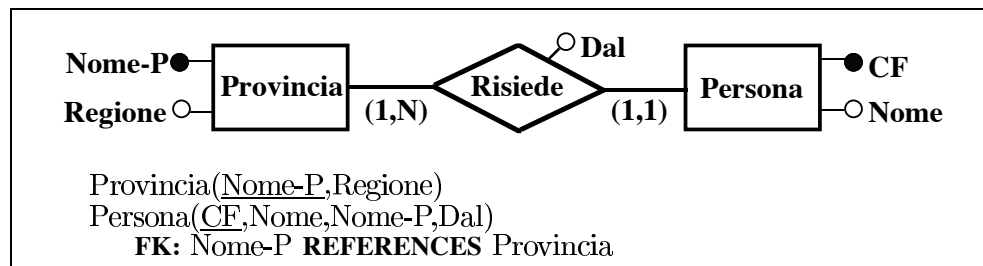
Associazione binaria uno a molti

◇ Traduzione con due relazioni

L'associazione può essere compattata nell'entità che partecipa con molteplicità unitaria, diciamo E1, includendo in E1 gli attributi di R e la chiave primaria di E2 come foreign key:

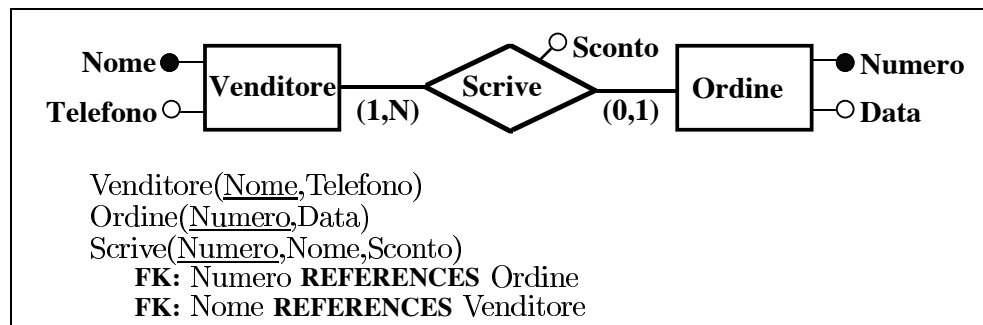
$E2(\underline{K2}, B1, B2, \dots)$

$E1(\underline{K1}, A1, A2, \dots, K2, C1, C2, \dots)$ **FK: K2 REFERENCES E2**



◇ Traduzione con tre relazioni

Se la partecipazione di E1 è parziale, per evitare i valori nulli, si può optare per la traduzione standard con tre relazioni:

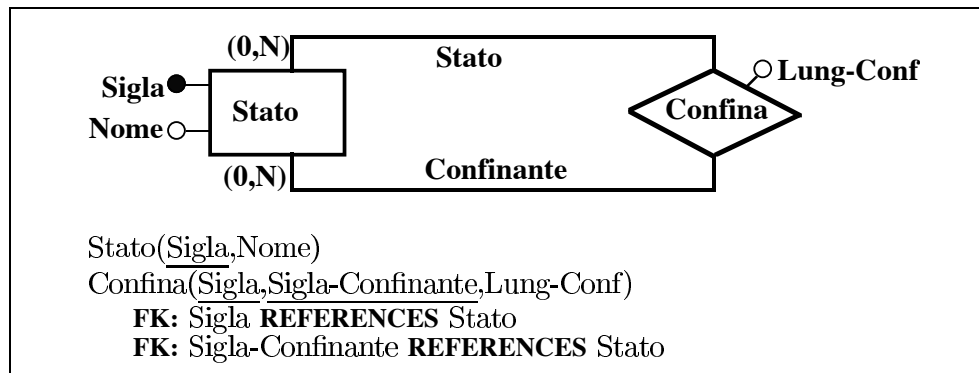


Associazioni unarie

Una associazione unaria può dar luogo ad una o due relazioni, dipendentemente dalle molteplicità in gioco.

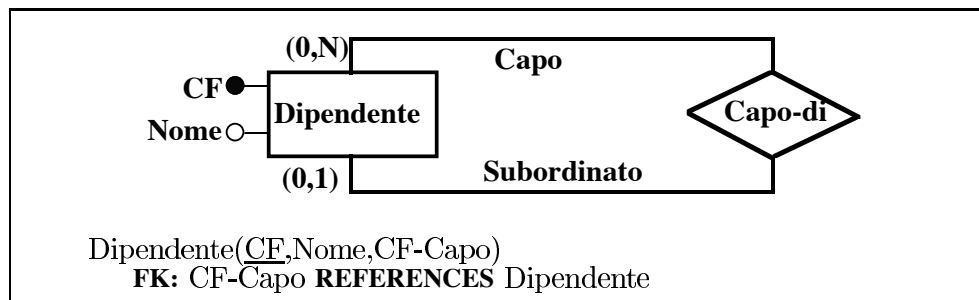
◇ Anello molti a molti

È tradotto con due relazioni, una per l'entità e una per l'associazione; la chiave della relazione che modella l'associazione è composta da due attributi, i cui nomi riflettono il diverso ruolo dell'entità ; ognuno di questi due attributi è anche foreign key.



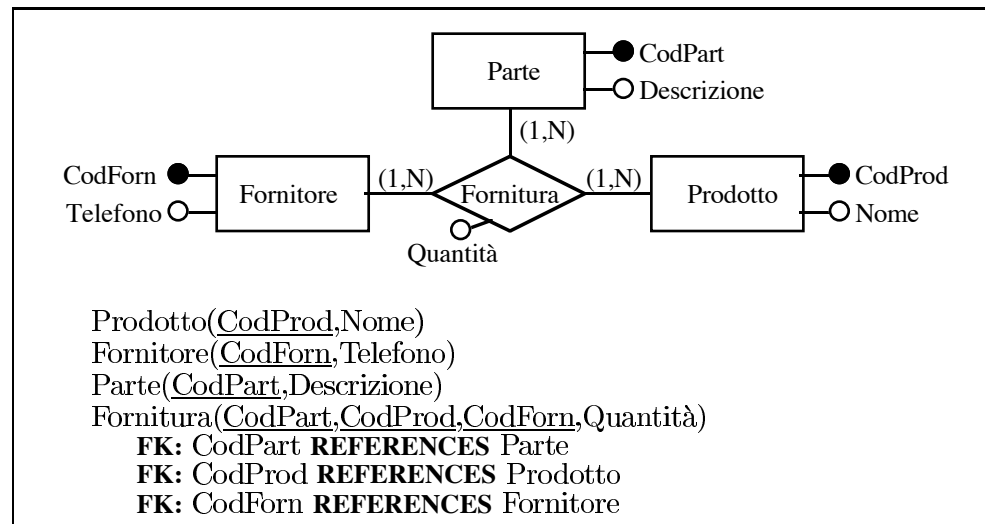
◇ Anello uno a molti

Oltre che con due relazioni, è traducibile con una sola relazione che contiene due volte l'attributo identificatore: una volta come chiave primaria e una volta come chiave esterna con un nome che riflette il ruolo dell'entità.



Associazione n-aria

◇ Segue la traduzione standard



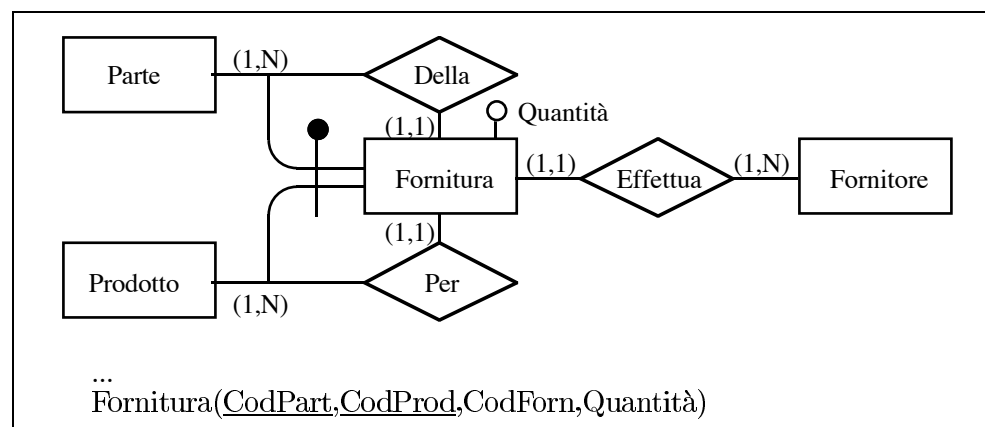
- Supponiamo di voler aggiungere al precedente schema E/R il seguente vincolo: ogni parte di un dato prodotto è fornita da un *unico* fornitore.

Tale vincolo introduce la seguente dipendenza funzionale: il fornitore dipende dalla parte e dal prodotto

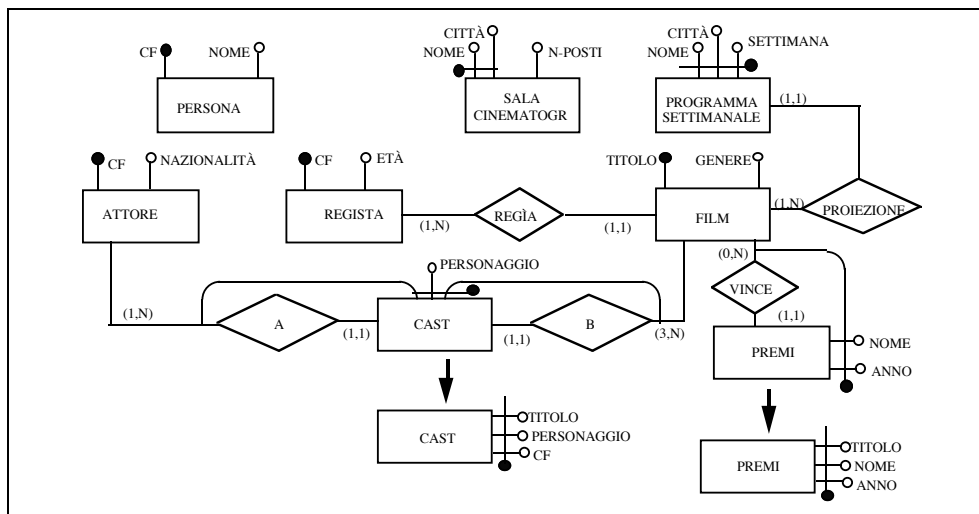
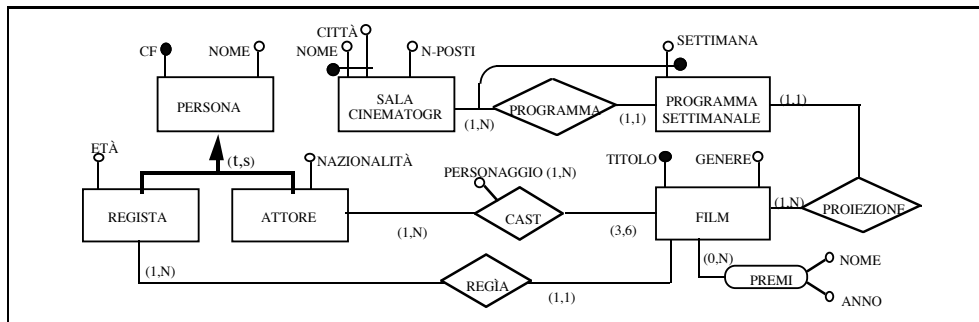
- 1 il vincolo *non viene espresso* nello schema E/R, ma viene aggiunto nello schema relazionale, cioè si usa ancora l'associazione ternaria Fornitura ma come chiave della relazione Fornitura si prende solo CodPart, CodProd:

Fornitura(CodPart, CodProd, CodForn, Quantità)

- 2 il vincolo *viene espresso* nello schema E/R, *reificando* l'associazione tramite una entità Fornitura identificata da Prodotto e Parte



Esempio



Persona(CF,Nome)
 Regista(CF,Età) **FK: CF REFERENCES** Persona
 Attore(CF,Nazionalità) **FK: CF REFERENCES** Persona
 Sala(Nome,Città,N-Posti)
 Film(TITOLO,Genere,CFRegista)
 FK: CFRegista REFERENCES Regista
 Premi(TITOLO,Nome,Anno)
 FK: TITOLO REFERENCES Film
 Cast(TITOLO,CF,Personaggio)
 FK: TITOLO REFERENCES Film
 FK: CF REFERENCES Attore
 Programma(Nome,Città,Settimana,TITOLO)
 FK: Nome,Città REFERENCES Sala
 FK: TITOLO REFERENCES Film

Capitolo 4

Elementi ed esempi del linguaggio SQL

Il linguaggio SQL (*Structured Query Language*) è il linguaggio standard per la definizione, manipolazione e interrogazione delle basi di dati relazionali. In questo capitolo ne verranno presentati gli elementi fondamentali.

Nella prima parte illustreremo l'uso di SQL come *DDL* (*Data Definition Language*), cioè come linguaggio per la definizione di schemi relazionali, la dichiarazione di vincoli di integrità e l'introduzione di indici.

Nella parte centrale descriveremo l'uso di SQL come *DML* (*Data Manipulation Language*), cioè come linguaggio per l'interrogazione e l'aggiornamento (inserimento, modifica e cancellazione) delle istanze della base di dati.

Nell'ultima parte mostreremo, brevemente, le caratteristiche di SQL come *DCL* (*Data Control Language*), cioè come linguaggio per la gestione degli utenti e l'assegnazione dei privilegi di accesso.

Gli aspetti principali del linguaggio SQL non trattati in questo testo sono i seguenti: l'uso di SQL all'interno di linguaggi di programmazione (*embedded SQL*); l'uso di SQL per scrivere *Stored Procedure* (procedure eseguite dal RDBMS su esplicita richiesta delle applicazioni o degli utenti) e *Trigger* (procedure attivate automaticamente dal RDBMS al verificarsi di determinate condizioni). D'altra parte tale uso di SQL, pur essendo di grande rilievo, non ha ancora raggiunto un livello accettabile di standardizzazione nei vari RDBMS commerciali.

Il linguaggio SQL

- ◇ Il linguaggio SQL (*Structured Query Language*) è il linguaggio standard per la definizione, manipolazione e interrogazione delle basi di dati relazionali.
- ◇ Il linguaggio SQL è stato originariamente sviluppato nei laboratori di ricerca IBM per il sistema relazionale System R.
- ◇ Il linguaggio SQL è stato successivamente sottoposto a varie fasi di standardizzazione che hanno portato allo standard attuale, noto come SQL92 (o SQL2) concluso nel 1992.

In seguito faremo riferimento allo standard SQL92.

Caratteristiche principali del linguaggio :

- ◇ è dichiarativo
- ◇ opera su *multiset* di tuple, cioè su insiemi di tuple che possono contenere duplicati
- ◇ permette di esprimere tutte le interrogazioni formulabili in Algebra Relazionale
- ◇ Notazione utilizzata per la sintassi degli elementi del linguaggio:
 - elementi **TERMINALI**
 - elementi *non-terminali*
 - [termine-opzionale]
 - $\text{termine}_1 \text{ — } \text{termine}_2 \text{ — } \dots \text{ — } \text{termine}_n$
 elenco di termini in alternativa di cui deve esserne scelto uno;
 per delimitare elenchi consecutivi si useranno le parentesi $i \dots j$

4.1 Definizione dei dati

◇ L'SQL mette a disposizione i seguenti sei **tipi di dati**

Caratteri :

CHARACTER [**VARYING**] [(*lunghezza_i*)]

[**CHARACTER SET** *set-di-caratteri_i*]

- Lunghezza fissa o variabile, con indicazione della lunghezza massima
- Possibilità di scegliere tra insiemi di caratteri
- Abbreviazioni: lung. fissa: **CHAR**(X), lung. variabile: **VARCHAR**(X)

Bit :

BIT [**VARYING**] [(*lunghezza_i*)]

Numerico Esatto :

- Interi : **INTEGER** (**INT**) e **SMALLINT**
- Decimali : **DECIMAL** [(*precisione_i* [, *scala_i*])]

Numerico Approssimato :

- **FLOAT** [(*precisione_i*)]
- **REAL**
- **DOUBLE PRECISION**

Datetime :

- **DATE** : Anno-Mese-Giorno, 'AAAA-MM-GG'
- **TIME** : Ore:Minuti:Secondi, 'OO-MM-SS'
- **TIMESTAMP** : Anno-Mese-Giorno Ore:Minuti:Secondi

Intervallo :

INTERVAL *campo-iniziale_i* **TO** *campo-finale_i*

- utilizzati per rappresentare periodi continui di tempo
- Esempio: **INTERVAL YEAR(2) TO MONTH**
Intervallo con valori compresi tra 0 anni - 0 mesi e 99 anni - 11 mesi

Valori nulli

- ◇ Per considerare i valori nulli l'SQL utilizza una logica a tre valori basata sulle seguenti tabelle di verità

AND	true	null	false
true	true	null	false
null	null	null	false
false	false	false	false

OR	true	null	false
true	true	true	true
null	true	null	null
false	true	null	false

NOT	
true	false
null	null
false	true

- ◇ Un qualsiasi confronto con il valore null restituisce ancora null (e quindi non risulta essere true)

$$(X \text{ } \text{op} \text{ } \text{null}) = (\text{null} \text{ } \text{op} \text{ } X) = \text{null}$$

dove op è un operatore algebrico o relazionale e X è una costante o variabile

- due valori null sono considerati diversi
- due valori null sono considerati uguali ai fini dell'indicizzazione e ai fini del raggruppamento
- Operatore di test per i valori nulli : $\text{attr}_i \text{ IS [NOT] NULL}$

Valori nulli (2)

- ◇ Come è stato detto a pagina 55, introducendo il concetto di valore nullo nel modello relazionale, è possibile dare diverse interpretazioni per il valore null.

Siccome l'SQL non distingue tra queste diverse interpretazioni di null, occorre diversificare la condizione di ricerca a secondo del test da effettuare, come evidenziato nel seguente esempio:

Consideriamo il seguente schema:

Studente(Matricola,Media)

e la sua istanza:

Matricola	Media
M1	28
M2	null
M3	23
M4	26

- ◇ Supponiamo di voler selezionare gli studenti sulla base della condizione “media superiore a 24”. Siccome l'attributo Media può assumere il valore null occorre fare attenzione sul significato di tale condizione quando si interroga la base di dati. Nella seguente tabella sono riportati alcuni possibili test da effettuare con la condizione “media superiore a 24”, con la relativa condizione di ricerca in SQL ed il risultato ottenuto sulla precedente istanza:

Test effettuato	Condizione di ricerca	Risultato
è noto che (Media \geq 24)	(Media \geq 24)	M1, M4
è possibile che (Media \geq 24)	(Media \geq 24) OR (Media IS NULL)	M1, M2, M4
è noto che non è (Media \geq 24)	NOT (Media \geq 24)	M3
non è noto che (Media \geq 24)	NOT (Media \geq 24) OR (Media IS NULL)	M2, M3
Media è non nota	(Media IS NULL)	M2
Media è nota	(Media IS NOT NULL)	M1, M3, M4

Schema

◇ Uno schema è una collezione di oggetti del database (tabelle, domini, indici, viste, privilegi, ...)

Tutti gli oggetti dello schema hanno lo stesso proprietario

```
CREATE SCHEMA [nome-schemai]
[AUTHORIZATION nome-proprietarioi]
(nome-oggetti-dello-schemai )
```

- Se **AUTHORIZATION** nome-proprietario_i viene omissso, si assume come proprietario l'utente che ha eseguito il comando
- Se nome-schema_i viene omissso, si assume come nome dello schema il nome del proprietario
- La dichiarazione degli oggetti dello schema può avvenire al di fuori del comando **CREATE SCHEMA**.

Tabella :

Una tabella è costituita da una collezione ordinata (lista) di uno o più attributi (colonne) e da un insieme di zero o più vincoli (*constraint*)

```
CREATE TABLE nome-tabellai
(nome-colonnai dominioi [vincoli-di-colonnai],
...
nome-colonnai dominioi [vincoli-di-colonnai],
[vincoli-di-tabellai]
)
```

```
CREATE TABLE ESAME
( CC CHAR(5) REFERENCES CORSO,
  MATR CHAR(9) REFERENCES STUDENTE(MATR),
  VOTO INTEGER CHECK ((VOTOi=18) AND (VOTOi=30)) OR (VOTO=33),
  PRIMARY KEY(CC,MATR),
  ...);
```

Schema (2)

Domini : Un dominio è un insieme di valori consentiti

Un dominio può essere un tipo di dato base dell'SQL oppure può essere definito dall'utente tramite la seguente sintassi:

```
CREATE DOMAIN ¡nome-dominio¿ AS ¡tipo-di-dati¿  
[¡valore-default¿ ] [¡vincoli-di-dominio¿]
```

```
CREATE DOMAIN VOTOESAME AS INTEGER  
CHECK ((VOTOi=18) AND (VOTOi=30)) OR (VOTO=33));
```

```
CREATE TABLE ESAME  
...  
VOTO VOTOESAME
```

Indici : I comandi di definizione degli indici non fanno parte dello standard del linguaggio.

Comunque, una sintassi accettata da vari sistemi è la seguente

```
CREATE [UNIQUE] INDEX ¡nome-indice¿  
ON ¡tabella¿ (¡lista-colonne¿)
```

- **UNIQUE** : non sono ammessi valori duplicati degli attributi specificati in *¡lista-colonne¿*
equivale quindi a dichiarare *¡lista-colonne¿* come chiave di *¡tabella¿*.

◇ Altri oggetti di uno schema, quali viste e privilegi, verranno introdotti in sezione 4.4.

Rimozione di uno schema e dei suoi oggetti :

È possibile rimuovere interamente uno schema con tutti i suoi oggetti

```
DROP SCHEMA ¡nome-schema¿
```

oppure singolarmente i vari oggetti, come ad esempio:

```
DROP TABLE ¡nome-tabella¿
```


Vincoli di tabella

◇ Permettono di controllare i valori che possono essere registrati in una tabella.

Primary Key :

PRIMARY KEY (*lista-colonne_i*)

- esprime la chiave primaria di una tabella
- può essere definito solo una volta in una tabella

Foreign Key :

FOREIGN KEY (*lista-colonne_i*)

REFERENCES *tabella_j* [(*lista-colonne_i*)]

- *column list_i* deve essere definita in *tabella_j* come chiave
- in assenza di (*column list_i*) è riferita alla primary key di *tabella_j*

Check :

CHECK (*condizione_i*)

- esprime un generico vincolo sulla tabella tramite una espressione che deve essere vera per tutte le tuple della tabella

```
CREATE TABLE CORSO
```

```
(  CC CHAR(5)
```

```
    ...,
```

```
    PRIMARY KEY (CC));
```

```
CREATE TABLE STUDENTE
```

```
(  MATR CHAR(9)
```

```
    CF CHAR(9)
```

```
    ...,
```

```
    PRIMARY KEY(MATR),
```

```
    CHECK (CF UNIQUE) );
```

```
CREATE TABLE ESAME
```

```
(  CC CHAR(5),
```

```
    MATR CHAR(9),
```

```
    VOTO INTEGER,
```

```
    PRIMARY KEY(CC,MATR),
```

```
    FOREIGN KEY(CC) REFERENCES CORSO(CC),
```

```
    FOREIGN KEY(MATR) REFERENCES STUDENTE,
```

```
    CHECK (((VOTOi=18) AND (VOTOi=30)) OR (VOTO=33)));
```

◇ Ad un vincolo può essere dato un nome con **CONSTRAINT** *nome_i*:

```
CONSTRAINT FK-CC-TO-CORSO FOREIGN KEY(CC) REFERENCES
CORSO(CC),
```

Vincoli di colonna

- ◇ Permettono di controllare i valori che possono essere registrati in una singola colonna.

Tutti i vincoli di colonna possono essere espressi come vincoli di tabella.

Not Null :

stabilisce che l'attributo non può assumere il valore null;

Unique :

esprime l'unicità dell'attributo;

- tramite Not Null Unique si dichiara che l'attributo è una chiave della tabella;

Primary Key :

stabilisce che l'attributo è la chiave primaria della tabella;

References :

esprime il vincolo della Foreign Key: la colonna nella tabella riferita può essere la primary key oppure una colonna esplicitamente indicata;

Check :

esprime un generico vincolo sulla colonna tramite una espressione logico-relazionale;

```
CREATE TABLE CORSO
  ( CC CHAR(5) PRIMARY KEY
    ...);
CREATE TABLE STUDENTE
  ( MATR CHAR(9) PRIMARY KEY
    CF CHAR(9) NOT NULL UNIQUE
    ...);
CREATE TABLE ESAME
  ( CC CHAR(5) REFERENCES CORSO,
    MATR CHAR(9) REFERENCES STUDENTE(MATR),
    VOTO INTEGER CHECK ((VOTOi=18) AND (VOTOi=30)) OR (VOTO=33)
    PRIMARY KEY(CC,MATR),
    ...);
```

Violazione di vincoli di integrità

- ◇ Generalmente, quando il sistema rileva una violazione di un vincolo di integrità, il comando di aggiornamento viene rifiutato e viene segnalato l'errore all'utente.
- ◇ Per i vincoli di integrità referenziale SQL92 permette di scegliere quale reazione adottare quando viene rilevata una violazione.

FOREIGN KEY (*lista-colonne_i*)

REFERENCES *tabella_i* [(*lista-colonne_i*)]

ON **DELETE—UPDATE**_{*i*}

;CASCADE—SET NULL—SET DEFAULT—NO ACTION****_{*i*}

- ◇ Una operazione sulla tabella che effettua il riferimento, detta *dipendente*, che viola il vincolo (inserimento o modifica degli attributi referenti) viene rifiutata.
- ◇ Ad una operazione sulla tabella *riferita* che viola il vincolo (modifica di attributi riferiti o cancellazione) si può rispondere con:

CASCADE

In caso di modifica, il nuovo valore dell'attributo della tabella riferita viene riportato su tutte le corrispondenti righe della tabella dipendente. In caso di cancellazione, tutte le righe della tabella dipendente corrispondenti alla riga cancellata vengono cancellate.

SET NULL

all'attributo referente viene assegnato il valore null al posto del valore modificato/cancellato nella tabella riferita

SET DEFAULT

all'attributo referente viene assegnato il valore di default al posto del valore modificato/cancellato nella tabella riferita

NO ACTION

non viene eseguita alcuna reazione

Modifica della struttura di una tabella

- ◇ Il comando **ALTER TABLE** *nome-tabella*, *tipo-modifica*, permette di modificare la struttura di una tabella. In *tipo-modifica*, si possono specificare le seguenti azioni

aggiunta di una colonna :

ADD [COLUMN] *nome-colonna*, *dominio*, [*vincoli-di-colonna*]

rimozione di una colonna :

DROP [COLUMN] *nome-colonna*, **RESTRICT—CASCADE**

Con **RESTRICT** la colonna viene eliminata solo se non ci sono dipendenze da tale colonna in altre definizioni, mentre con **CASCADE** si forza l'eliminazione della colonna e di tutte le dipendenze nelle altre definizioni.

Ad esempio, il comando **ALTER TABLE CORSO DROP CC** fallisce con **RESTRICT** in quanto nella tabella **ESAME** c'è un vincolo di foreign key che usa **CC**, mentre con **CASCADE** viene cancellata sia la colonna **CC** che il vincolo di foreign key in **ESAME**.

aggiunta di un vincolo di tabella :

ADD *vincolo-di-tabella*

rimozione di un vincolo di tabella :

DROP *nome-vincolo*, **RESTRICT—CASCADE**

Con **RESTRICT** un vincolo di unicità non viene eliminato se è usato in un vincolo di foreign key, mentre con **CASCADE** si forza l'eliminazione anche del vincolo di foreign key.

imposta il valore di default per una colonna :

ALTER [COLUMN] *nome-colonna*, **SET DEFAULT** *valore*

annulla il valore di default per una colonna :

ALTER [COLUMN] *nome-colonna*, **DROP DEFAULT**

- ◇ Il comando **ALTER DOMAIN** *nome-dominio*, *tipo-modifica*, permette di modificare alcune proprietà di un dominio di valori. Le proprietà modificabili sono solo quelle che non influenzano i dati reali, infatti in *tipo-modifica*, si possono specificare le seguenti azioni

imposta/annulla il valore di default per il dominio

aggiunta/rimozione di un vincolo di dominio

4.2 Interrogazioni

◇ L'istruzione base dell'SQL per costruire interrogazioni di complessità arbitraria è lo statement **SELECT**

◇ Sintassi di base:

```
SELECT [DISTINCT—ALL] ¡lista-select¿  
FROM ¡lista-from¿  
[WHERE ¡condizione¿]  
[ORDER BY ¡lista-order¿]
```

ALL (Default): non c'è l'eliminazione dei duplicati (semantica del multiset)

DISTINCT: eliminazione dei duplicati

◇ *¡lista-select¿*:

- Uno o più attributi delle relazioni della lista-from
- Funzioni aggregate: COUNT,AVG,MIN,MAX,SUM
- Costanti
- Una generica espressione matematica che coinvolge uno o più degli oggetti precedenti
- Il simbolo *: tutti gli attributi

◇ *¡lista-from¿*:

- Una o più tabelle o viste
- Operazioni di join tra una o più tabelle o viste

◇ *¡condizione¿*: espressione booleana di:

- Predicati semplici
- Predicati di join
- Predicati con subquery

◇ *¡lista-order¿*:

- Uno o più attributi della *¡lista-select¿*
- Ordine ascendente (default) o discendente (**DESC**)

Esempio di riferimento per le interrogazioni

S(Matr, SNome, Città, ACorso)

C(CC, CNome, CD, anno)

FK: CD REFERENCES D

D(CD, CNome, Città)

E(Matr, CC, Data, Voto)

FK: Matr REFERENCES S

FK: CC REFERENCES C

S

Matr	SNome	Città	ACorso
M1	Lucia Quaranta	SA	1
M2	Giacomo Tedesco	PA	2
M3	Carla Longo	MO	1
M4	Ugo Rossi	MO	1
M5	Valeria Neri	MO	2
M6	Giuseppe Verdi	BO	1
M7	Maria Rossi	null	1

C

CC	CNome	CD
C1	Fisica 1	D1
C2	Analisi Matematica 1	D2
C3	Fisica 2	D1
C4	Analisi Matematica 2	D3

D

CD	CNome	Città
D1	Paolo Rossi	MO
D2	Maria Pastore	BO
D3	Paola Caboni	FI

E

Matr	CC	Data	Voto
M1	C1	06-29-1995	24
M1	C2	08-09-1996	33
M1	C3	03-12-1996	30
M2	C1	06-29-1995	28
M2	C2	07-07-1996	24
M3	C2	07-07-1996	27
M3	C3	11-11-1996	25
M4	C3	11-11-1996	33
M6	C2	01-02-1996	28
M7	C1	06-29-1995	24
M7	C2	04-11-1996	26
M7	C3	06-23-1996	27

Predicati semplici

- **Operatori relazionali** : attr_i **op-rel** cost_i

dove $\text{op-rel} \in \{=, <, >, \geq, \leq, \neq\}$

Es. “Studenti del secondo anno di corso”

```
SELECT *
FROM S
WHERE ACorso=2
```

Es. “Esami con voto compreso tra 24 e 28”

```
SELECT *
FROM E
WHERE Voto  $\geq$  24
AND Voto  $\leq$  28
```

- **Operatore di range** : attr_i **BETWEEN** cost1_i **AND** cost2_i

Es. “Esami con voto compreso tra 24 e 28”

```
SELECT *
FROM E
WHERE Voto BETWEEN 24 AND 28
```

- **Operatore di set** : attr_i **IN** (cost1_i , ..., costN_i)

Es. “Esami con voto pari a 29, 30 oppure con lode (voto pari a 33)”

```
SELECT *
FROM E
WHERE Voto IN (29,30,33)
```

- **Operatore di confronto stringhe** : attr_i **LIKE** stringa_i

dove stringa_i può contenere i caratteri speciali `_` (carattere arbitrario) e `%` (stringa arbitraria)

Es. “Studenti il cui nome inizia con A e termina con O”

```
SELECT *
FROM S
WHERE SNome LIKE 'A%O'
```

Predicati semplici (2)

• Operatori quantificati :

$iattr_i \text{ op-rel}_i [ANY-ALL] (icost1_i, \dots, icostN_i)$

Es. “Esami con voto pari a 29, 30 oppure con lode (voto pari a 33)”

```
SELECT *
FROM E
WHERE Voto = ANY (29,30,33)
```

Es. “Esami con voto diverso da 29, 30 e 33”

```
SELECT *
FROM E
WHERE Voto  $\neq$  ALL (29,30,33)
```

◇ Confronto con l’insieme vuoto ():

- $iattr_i \text{ op-rel}_i ANY ()$ ha valore false
- $iattr_i \text{ op-rel}_i ALL ()$ ha valore true

• Operatore di confronto con valori NULL : $iattr_i \text{ IS [NOT] NULL}$

Es. “Studenti con l’attributo città non specificato”

```
SELECT *
FROM S
WHERE Città IS NULL
```

◇ Ordinamento del risultato:

Es. “Studenti di Modena ordinati in senso ascendente rispetto all’anno di corso”

```
SELECT Matr, ACorso
FROM S
WHERE Città='MO'
ORDER BY ACorso
```

- L’ordinamento deve essere fatto rispetto a uno o più elementi della $i\text{lista-select}_i$: un tale elemento può essere indicato anche riportando la sua posizione nella $i\text{lista-select}_i$.

Es. “Esami del corso C1 ordinati in senso discendente rispetto al voto espresso in sessantesimi, e a parità di voto rispetto alla matricola”

```
SELECT Matr, CC, (60*Voto)/30
FROM E
WHERE CC='C1'
ORDER BY 3 DESC, Matr
```


Prodotto Cartesiano e Join

- ◇ Il **prodotto cartesiano** di due o più relazioni si ottiene riportando le relazioni nella *lista-from* della clausola FROM, senza clausola WHERE.
- ◇ Il **join** viene espresso generalmente riportando nella clausola FROM le relazioni interessate e nella clausola WHERE le *condizioni di join*.

Es. “Combinazioni di studenti e di docenti residenti nella stessa città”

```
SELECT S.Matr,S.Città,D.CD
FROM   S,D
WHERE  S.Città=D.Città
```

S.Matr	S.Città	D.CD
M6	BO	D2
M3	MO	D1
M4	MO	D1
M5	MO	D1

- ◇ Con l'SQL92 è possibile esprimere le operazioni di join nella clausola FROM:

```
tabella1, [INNER—LEFT—RIGHT—FULL] JOIN tabella2
ON condizione
```

Il default è il join interno è quindi la parola INNER può essere omessa.

Es. “Combinazioni di studenti e di docenti residenti nella stessa città”

```
SELECT S.Matr,S.Città,D.CD
FROM   S JOIN D ON (S.Città=D.Città)
```

- ◇ Join con predicati locali

Es. “Studenti residenti nella stessa città di residenza del docente D1”

```
SELECT S.*
FROM   S,D
WHERE  S.Città = D.Città
AND    D.CD='D1'
```

oppure

```
SELECT S.*
FROM   S JOIN D ON (S.Città = D.Città)
WHERE  D.CD='D1'
```

Matr	SNome	Città	ACorso
M3	Carla Longo	MO	1
M4	Ugo Rossi	MO	1
M5	Valeria Neri	MO	2

Outer-Join

- ◇ Oltre al join interno, con lo standard SQL92 sono stati introdotti gli operatori di outer join :

`¡tabella1¿ LEFT JOIN ¡tabella2¿ ON ¡condizione¿ :`

mantiene le tuple di `¡tabella1¿` per cui non esiste corrispondenza in `¡tabella2¿`

`¡tabella1¿ RIGHT JOIN ¡tabella2¿ ON ¡condizione¿ :`

mantiene le tuple di `¡tabella2¿` per cui non esiste corrispondenza in `¡tabella1¿`

`¡tabella1¿ FULL JOIN ¡tabella2¿ ON ¡condizione¿ :`

mantiene le tuple di `¡tabella1¿` e di `¡tabella2¿` per cui non esiste corrispondenza in `¡tabella2¿` e `¡tabella1¿` rispettivamente

- ◇ Le tuple senza corrispondenza vengono concatenate con tuple di valori null, di lunghezza opportuna.

- Es.** “Tutti gli studenti con i relativi esami sostenuti inclusi gli studenti che non hanno sostenuto alcun esame”

```
SELECT S.Matr,E.CC
FROM   S LEFT JOIN E ON (S.Matr=E.Matr)
```

- Es.** “Combinazioni di studenti e di docenti residenti nella stessa città inclusi gli studenti (docenti) che risiedono in una città che non ha corrispondenza nella relazione dei docenti (studenti)”

```
SELECT S.Matr,S.Città,D.CD, D.Città
FROM   S FULL JOIN D ON (S.Città=D.Città)
```

S.Matr	S.Città	D.CD	D.Città
M6	BO	D2	BO
M3	MO	D1	MO
M4	MO	D1	MO
M5	MO	D1	MO
M1	SA	null	null
M2	PA	null	null
M7	null	null	null
null	null	D3	FI

Outer-Join (2)

◇ Nella clausola FROM è possibile esprimere più di un'operazione di join.

Es. “Per ogni esame con voto superiore a 24 riportare il nome dello studente e il codice del docente del corso”

```
SELECT S.SNome,C.CD
FROM   (S JOIN E ON (S.Matr=E.Matr))
        JOIN C ON (E.CC=C.CC)
WHERE  Voto > 24
```

In modo equivalente

```
SELECT S.SNome,C.CD
FROM   S,E,C
WHERE  S.Matr=E.Matr
AND    E.CC=C.CC
AND    Voto > 24
```

Es. “Matricole degli studenti con i codici dei corsi dei relativi esami sostenuti, inclusi gli studenti che non hanno sostenuto alcun esame e i corsi per i quali non ci sono esami sostenuti”

```
SELECT DISTINCT S.Matr,C.CC
FROM   (S LEFT JOIN E ON (S.Matr=E.Matr))
        FULL JOIN C ON (E.CC=C.CC)
```

S.Matr	C.CC
M1	C1
M1	C2
M1	C3
M2	C1
M2	C2
M3	C2
M3	C3
M4	C3
M5	null
M6	C2
M7	C1
M7	C2
M7	C3
null	C4

Self Join

- ◇ Nel join tra una tabella e se stessa occorre necessariamente utilizzare dei sinonimi (*alias*) per distinguere le diverse occorrenze della tabella.

Es. “Coppie di studenti residenti nella stessa città”

```
SELECT S1.Matr,S2.Matr
FROM   S S1, S S2
WHERE  S1.Città = S2.Città
AND    S1.Matr < S2.Matr
```

oppure

```
SELECT S1.Matr,S2.Matr
FROM   S S1 Join S S2
      ON (S1.Città = S2.Città)
WHERE  S1.Matr < S2.Matr
```

S1.Matr	S2.Matr
M3	M5
M3	M4
M4	M5

Es. “Matricole degli studenti che hanno sostenuto almeno uno degli esami sostenuti dallo studente di nome ‘Giuseppe Verdi’ ”

```
SELECT E1.Matr
FROM   S, E E1, E E2
WHERE  E2.Matr = S.Matr
AND    E1.CC = E2.CC
AND    S.SNome='Giuseppe Verdi'
```

Interrogazioni innestate

- ◇ Una interrogazione viene detta *innestata* o *nidificata* se la sua condizione è formulata usando il risultato di un'altra interrogazione, chiamata *subquery*.

In generale, un'interrogazione innestata viene formulata con:

Operatori quantificati : `¡attr¿ ¡op-rel¿[ANY—ALL] ¡subquery¿`

Operatore di set : `¡attr¿ [NOT] IN ¡subquery¿`

Quantificatore esistenziale : `[NOT] EXISTS ¡subquery¿`

◇ Interrogazione innestata con operatori quantificati

Il confronto tra un attributo e il risultato di una interrogazione,

`¡attr¿ ¡op-rel¿ ¡subquery¿`

non è in generale corretto in quanto si confronta il singolo valore assunto da `¡attr¿` con l'insieme di valori restituiti da `¡subquery¿`. Tuttavia, il confronto è possibile quando `¡subquery¿` produce (run-time) un valore atomico.

Nel confronto tra un attributo e il risultato di una interrogazione occorre specificare, dopo l'operatore relazionale `¡op-rel¿`, ANY oppure ALL.

Es. “Nome degli studenti che hanno sostenuto l'esame del corso C1”

```
SELECT SNome
FROM   S
WHERE  Matr =ANY ( SELECT Matr
                   FROM   E
                   WHERE  CC='C1')
```

Es. “Studenti con anno di corso più basso”

```
SELECT *
FROM   S
WHERE  ACorso ¡= ALL ( SELECT ACorso
                     FROM   S)
```

Interrogazioni innestate con l'operatore di set

Es. “Nome degli studenti che hanno sostenuto l'esame del corso C1”

```
SELECT SNome
FROM   E,S
WHERE  E.Matr=S.Matr
AND    E.CC = 'C1'
```

oppure

```
SELECT SNome
FROM   S
WHERE  Matr IN ( SELECT Matr
                  FROM   E
                  WHERE  CC='C1')
```

La subquery restituisce l'insieme (M1,M2,M7) e pertanto la condizione dell'interrogazione innestata equivale a `Matr IN (M1,M2,M7)`.

Es. “Nome degli studenti che hanno sostenuto l'esame di un corso del docente D1”

```
SELECT SNome
FROM   E,S,C
WHERE  E.Matr=S.Matr
AND    E.CC=C.CC
AND    CD='D1'
```

oppure

```
SELECT SNome
FROM   S
WHERE  Matr IN ( SELECT Matr
                  FROM   E
                  WHERE  CC IN ( SELECT CC
                                FROM   C
                                WHERE  CD='D1'))
```

La subquery più interna restituisce l'insieme (C1,C3) e pertanto la condizione della subquery intermedia equivale a `CC IN (C1,C3)`;

La subquery intermedia restituisce quindi l'insieme (M1,M2,M3,M4,M7) e pertanto la condizione dell'interrogazione innestata equivale a `Matr IN (M1,M2,M3,M4,M7)`.

Subquery correlate

- ◇ Una subquery viene detta correlata se la sua condizione è formulata usando relazioni e/o sinonimi definite nella query esterna.

Es. “Nome degli studenti che hanno sostenuto l’esame del corso C1”

```
SELECT SNome
FROM   S
WHERE  'C1' IN ( SELECT CC
                  FROM   E
                  WHERE  E.Matr=S.Matr)
```

Per ogni tupla della relazione S della query esterna, detta *tupla corrente*, si valuta la subquery che ha come risultato il codice degli esami sostenuti dallo studente corrente: se C1 è tra questi esami, il nome dello studente corrente viene riportato in uscita.

- ◇ Per una maggiore leggibilità è conveniente far uso di sinonimi

```
SELECT S1.SNome
FROM   S S1
WHERE  'C1' IN ( SELECT E1.CC
                  FROM   E E1
                  WHERE  E1.Matr=S1.Matr)
```

- ◇ I sinonimi sono indispensabili quando una stessa relazione compare sia nella query esterna che nella subquery (analogamente al self join)

Es. “Per ogni città, il nome degli studenti con anno di corso più alto”

```
SELECT S1.Città,S1.SNome
FROM   S S1
WHERE  S1.ACorso >= ALL ( SELECT S2.ACorso
                          FROM   S S2
                          WHERE  S1.Città=S2.Città)
```

- ◇ Per gli attributi non qualificati avviene la qualificazione automatica con il nome della relazione *più vicina*.

Ad esempio, la precedente interrogazione si può scrivere come:

```
SELECT Città,SNome
FROM   S S1
WHERE  ACorso >= ALL ( SELECT ACorso
                      FROM   S
                      WHERE  S1.Città=Città)
```

Il quantificatore esistenziale

◇ Il predicato

EXISTS (*j*subquery_{*i*})

ha valore true se e solo se l'insieme di valori restituiti da *j*subquery_{*i*} è non vuoto.

Es. “Nome degli studenti che hanno sostenuto l'esame del corso C1”

```
SELECT SNome
FROM   S
WHERE  EXISTS ( SELECT *
                  FROM   E
                  WHERE  E.Matr=S.Matr
                  AND     E.CC='C1' )
```

◇ Il predicato

NOT EXISTS (*j*subquery_{*i*})

ha valore true se e solo se l'insieme di valori restituiti da *j*subquery_{*i*} è vuoto.

Es. “Nome degli studenti che non hanno sostenuto l'esame del corso C1”

```
SELECT SNome
FROM   S
WHERE  NOT EXISTS ( SELECT *
                     FROM   E
                     WHERE  E.Matr=S.Matr
                     AND     E.CC='C1' )
```

◇ Generalmente, al fine di formulare query *significant* con EXISTS è indispensabile utilizzare subquery correlate:

```
SELECT SNome
FROM   S
WHERE  EXISTS ( SELECT *
                  FROM   E
                  WHERE  E.CC='C1' )
```

Restituisce tutti gli studenti (nessun studente) se c'è (se non c'è) un esame del corso C1.

Riduzioni di query innestate (1)

◇ Le query innestate formulate con i seguenti operatori si possono ridurre a query semplici equivalenti (stessa risposta per ogni possibile istanza della base di dati):

- IN
- ANY (con qualsiasi operatore di confronto)
- EXISTS con subquery correlata

Es. “Nome degli studenti che hanno sostenuto l’esame del corso C1”

1.

```
SELECT SNome
FROM   S
WHERE  Matr IN ( SELECT Matr
                  FROM   E
                  WHERE  CC='C1')
```
2.

```
SELECT SNome
FROM   S
WHERE  Matr =ANY ( SELECT Matr
                   FROM   E
                   WHERE  CC='C1')
```
3.

```
SELECT SNome
FROM   S
WHERE  EXISTS ( SELECT *
                FROM   E
                WHERE  E.Matr=S.Matr
                AND     E.CC='C1' )
```

Queste tre query sono equivalenti alla seguente query semplice:

```
SELECT SNome
FROM   E,S
WHERE  E.Matr=S.Matr
AND     E.CC='C1'
```

Riduzioni di query innestate (2)

◇ Le query innestate formulate con i seguenti operatori non si possono ridurre:

- NOT IN
- ALL (con qualsiasi operatore di confronto)
- NOT EXISTS con subquery correlata

Es. “Nome degli studenti che non hanno sostenuto l’esame del corso C1”

1.

```
SELECT SNome
FROM   S
WHERE  Matr NOT IN ( SELECT Matr
                     FROM   E
                     WHERE  CC='C1')
```
2.

```
SELECT SNome
FROM   S
WHERE  Matr <> ALL ( SELECT Matr
                    FROM   E
                    WHERE  CC='C1')
```
3.

```
SELECT SNome
FROM   S
WHERE  NOT EXISTS ( SELECT *
                   FROM   E
                   WHERE  E.Matr=S.Matr
                   AND     E.CC='C1' )
```

Queste tre query sono equivalenti tra di loro ma non si possono ridurre ad una query semplice.

Si noti infatti che la query semplice:

```
SELECT SNome
FROM   E,S
WHERE  E.Matr=S.Matr
AND    E.CC <> 'C1'
```

restituisce il nome degli studenti che hanno sostenuto almeno un esame di un corso diverso da C1.

Funzioni aggregate (column functions) (1)

- ◇ SQL mette a disposizione una serie di funzioni per elaborare i valori di un attributo (MAX, MIN, AVG, SUM) e per contare le tuple che soddisfano una condizione (COUNT).

Opzioni: * : conteggio del numero di righe (COUNT)

ALL : trascura i null (COUNT,AVG,SUM); default

DISTINCT : trascura i null ed elimina i duplicati (COUNT,AVG,SUM)

Es. “Numero di studenti presenti”

```
SELECT COUNT(*)
FROM S
```

Es. “Numero di studenti che hanno sostenuto almeno un esame”

```
SELECT COUNT(DISTINCT Matr)
FROM E
```

Es. “Numero di studenti con anno di corso non nullo”

```
SELECT COUNT(ACorso)
FROM S
```

Es. “Numero di anni di corso di studenti presenti”

```
SELECT COUNT(DISTINCT ACorso)
FROM S
```

Es. “Numero di coppie distinte matricola-voto”

```
SELECT COUNT(DISTINCT Matr,Voto)
FROM E
```

Es. “Voto medio degli esami sostenuti dalla matricola M1”

```
SELECT AVG(Voto)
FROM E WHERE Matr='M1'
```

che è equivalente a

```
SELECT SUM(Voto)/COUNT(Voto)
FROM E WHERE Matr='M1'
```

Es. “Studenti il cui anno di corso è minore di quello massimo presente”

```
SELECT *
FROM S
WHERE ACorso < ( SELECT MAX(ACorso)
                 FROM S)
```

- ◇ Non è lecita la presenza contemporanea nella *lista-select* di nomi di campi e funzioni aggregate, pertanto la seguente interrogazione non è corretta:

```
SELECT Matr,MAX(Voto)
FROM E
```

Raggruppamento: la clausola GROUP BY

- ◇ In una istruzione SELECT è possibile formare dei gruppi di tuple che hanno lo stesso valore di specificati attributi, tramite la clausola GROUP BY:

```
SELECT [DISTINCT—ALL] ¡lista-select¡  
FROM ¡lista-from¡  
[WHERE ¡condizione¡]  
[GROUP BY ¡lista-group¡ ]
```

- ◇ Il risultato della SELECT è un **unico record per ciascun gruppo**:
pertanto nella ¡lista-select¡ possono comparire solo:

- Uno o più attributi di raggruppamento, cioè specificati in ¡lista-group¡
- Funzioni aggregate: tali funzioni vengono valutate, e quindi forniscono un valore unico, per ciascun gruppo

Es. “Voto massimo ottenuto per ogni studente”

```
SELECT    Matr,MAX(Voto)  
FROM      E  
GROUP BY  Matr
```

Es. “Voto massimo e minimo ottenuto per ogni studente, escludendo il corso C1”

```
SELECT    Matr,MAX(Voto),MIN(Voto)  
FROM      E  
WHERE     CC != 'C1'  
GROUP BY  Matr
```

Es. “Codice e nome di un corso, e relativo numero di esami sostenuti”

```
SELECT    C.CC,C.CNome,COUNT(*)  
FROM      E,C  
WHERE     E.CC=C.CC  
GROUP BY  C.CC,C.CNome
```

- ◇ La seguente interrogazione non è corretta:

```
SELECT    C.CC,C.CNome,COUNT(*)  
FROM      E,C  
WHERE     E.CC=C.CC  
GROUP BY  C.CC
```

Raggruppamento: La clausola HAVING

- ◇ La clausola HAVING è l'equivalente della clausola WHERE applicata a gruppi di tuple: ogni gruppo costruito tramite GROUP BY fa parte del risultato dell'interrogazione solo se il predicato specificato nella clausola HAVING risulta soddisfatto.

[**GROUP BY** *lista-group*_i]

[**HAVING** *condizione*_i]

Il predicato espresso nella clausola HAVING è formulato utilizzando

- Uno o più attributi specificati in *lista-group*_i
- Funzioni aggregate

Es. “Numero di esami per ogni voto compreso tra 22 e 26”

```
SELECT  Voto,COUNT(*)
FROM    E
GROUP BY Voto
HAVING  Voto BETWEEN 22 AND 26
```

che è equivalente a

```
SELECT  Voto,COUNT(*)
FROM    E
WHERE   Voto BETWEEN 22 AND 26
GROUP BY Voto
```

Es. “Media dei voti per ogni esame sostenuto da più di due studenti”

```
SELECT  CC,AVG(Voto)
FROM    E
GROUP BY CC
HAVING  COUNT(*) > 2
```

Es. “Matricola degli studenti che hanno sostenuto almeno due esami con lo stesso voto”

```
SELECT  Matr
FROM    E
GROUP BY Matr,Voto
HAVING  COUNT(CC) >= 2
```

Unione, Differenza, Intersezione

◇ Sono definiti i seguenti operatori insiemistici relazionali:

Unione : `¡subquery¿ UNION [ALL] ¡subquery¿`

Differenza : `¡subquery¿ EXCEPT [ALL] ¡subquery¿`

Intersezione : `¡subquery¿ INTERSECTION [ALL] ¡subquery¿`

◇ Con ALL si considera la semantica del *multiset*

- $\{a, b\} \text{ UNION ALL } \{a\} = \{a, a, b\}$
- $\{a, a\} \text{ EXCEPT ALL } \{a\} = \{a\}$
- $\{a, a\} \text{ INTERSECTION ALL } \{a, b\} = \{a, a\}$

Es. “Città di studenti o docenti presenti”

```
SELECT Città FROM S
UNION
SELECT Città FROM D
```

Es. “Città di studenti ma non di docenti”

```
SELECT Città FROM S
EXCEPT
SELECT Città FROM D
```

Es. “Città di studenti e di docenti”

```
SELECT Città FROM S
INTERSECTION
SELECT Città FROM D
```

◇ Le interrogazioni formulate tramite gli operatori EXCEPT e INTERSECTION possono essere riscritte, in maniera equivalente, utilizzando operatori introdotti in precedenza, (ad esempio NOT IN e IN rispettivamente):

Es. “Città di studenti ma non di docenti”

```
SELECT Città
FROM S
WHERE Città NOT IN ( SELECT Città
                     FROM D)
```

Es. “Città in cui ci sono studenti e docenti”

```
SELECT Città
FROM S
WHERE Città IN ( SELECT Città
                 FROM D)
```

Questo non è valido per l'operatore UNION. Anche per questo motivo generalmente nelle implementazioni di SQL (compresa quella di INGRES) l'unico degli operatori presenti è UNION.

Divisione

◇ L'operazione di divisione non è definita in SQL. Pertanto, le interrogazioni che richiedono tale operatore, come ad esempio la seguente:

“Studenti che hanno sostenuto **tutti** gli esami relativi a corsi del docente D1”

vengono in genere riformulate con una doppia negazione nel seguente modo”

“Studenti per i quali **non** esiste alcun corso del docente D1 di cui **non** hanno sostenuto l'esame”

```
SELECT *
FROM S
WHERE NOT EXISTS
      ( SELECT *
        FROM C
        WHERE CD='D1'
        AND   NOT EXISTS
              ( SELECT *
                FROM E
                WHERE E.Matr=S.Matr
                AND   E.CC=C.CC))
```

Es. “Studenti che hanno sostenuto **tutti** gli esami sostenuti dallo studente M7”, riformulata come

“Studenti per i quali **non** esiste alcun esame sostenuto da M7 che essi **non** hanno sostenuto”

```
SELECT *
FROM S
WHERE Matr != 'M7'
AND   NOT EXISTS
      ( SELECT *
        FROM E E1
        WHERE Matr='M7'
        AND   NOT EXISTS
              ( SELECT *
                FROM E E2
                WHERE E2.Matr=S.Matr
                AND   E2.CC=E1.CC))
```

Alcuni esempi di interrogazioni (1)

- ◇ Questo primo esempio riassume tutte le clausole di uno statement SELECT, mettendone in evidenza l'ordine con il quale esse vengono considerate per determinare il risultato dell'interrogazione.

Es. “Riportare, per ogni studente, il voto massimo ottenuto, escludendo gli esami con voto pari a 33 e considerando solo gli studenti con più di 10 esami; ordinare il risultato per voti massimi crescenti”

Per risolvere tale interrogazione occorre eseguire i seguenti passi:

1. considerare la relazione E:
FROM E
2. selezionare gli esami con voto diverso da 33:
WHERE Voto \neq 33
3. raggruppare tali esami sulla base del valore di Matr:
GROUP BY Matr
4. per ciascuno di tali gruppi contare il numero di esami (tuple) e controllare che sia maggiore di 10:
HAVING COUNT(*) $>$ 10
5. per ciascuno dei gruppi selezionati determinare il massimo voto e riportarlo in uscita assieme alla Matr:
SELECT Matr, MAX(Voto)
6. ordinare il risultato sulla base del massimo voto calcolato:
ORDER BY 2

L'interrogazione SQL risulta pertanto essere la seguente:

```
(5) SELECT Matr, MAX(Voto)
(1) FROM E
(2) WHERE Voto  $\neq$  33
(3) GROUP BY Matr
(4) HAVING COUNT(*)  $>$  10
(6) ORDER BY 2
```


Alcuni esempi di interrogazioni (2)

Es. “Riportare, per ogni docente, il corso per il quale sono stati sostenuti il maggior numero di esami”

Per risolvere tale interrogazione occorre eseguire i seguenti passi:

1. effettuare un join tra C e E, in quanto il docente del corso (CD) è specificato in C e gli esami sono memorizzati in E:

```
SELECT ...
FROM   C C1,E E1
WHERE  C1.CC=E1.CC
```

2. formare dei gruppi di tuple con lo stesso valore della coppia CD,CC e contare per ciascun gruppo, tramite COUNT(*) oppure COUNT(Mat), il numero di tuple che lo compongono: il risultato del conteggio fornisce il numero di esami corrispondenti al docente CD per il suo corso CC

```
SELECT      ...
FROM        C C1,E E1
WHERE       C1.CC=E1.CC
GROUP BY    C1.CD,C1.CC
```

3. una coppia CD,CC viene riportata in uscita dal SELECT se relativo conteggio delle tuple restituisce il valore più grande tra tutte le coppie corrispondenti allo stesso docente, cioè allo stesso valore di CD.

Questa condizione deve essere espressa nella clausola HAVING:

```
GROUP BY C1.CD,C1.CC
HAVING COUNT(*) >= ALL (...)
```

dove (...) sono i conteggi delle tuple relativi allo stesso docente, che si ottengono ripetendo il raggruppamento del punto 2, considerando però solo gli esami di c1.CD. Quindi in definitiva l'interrogazione SQL risulta essere la seguente:

```
SELECT      C1.CD,C1.CC
FROM        C C1,E E1
WHERE       C1.CC=E1.CC
GROUP BY    C1.CD,C1.CC
HAVING      COUNT(*) >= ALL ( SELECT      COUNT(*)
                                FROM        C C2,E E2
                                WHERE       C2.CC=E2.CC
                                AND         C2.CD=C1.CD
                                GROUP BY    C2.CD,C2.CC)
```

Si noti che nella subquery il raggruppamento può essere fatto solo su C2.CC, in quanto tutte le tuple di tale subquery hanno lo stesso valore di C2.CD, che è uguale a C1.CD.

Alcuni esempi di interrogazioni (3)

Es. “Docenti che hanno registrato esami per **tutti** i corsi da essi sostenuti”

può essere risolta riformulandola con una doppia negazione:

“Docenti per i quali **non** esiste un loro corso con **nessun** esame registrato”

```
SELECT *
FROM D
WHERE NOT EXISTS
  ( SELECT *
    FROM C
    WHERE C.CD=D.CD
    AND NOT EXISTS
      ( SELECT *
        FROM E E2
        WHERE E2.Matr=S.Matr
        AND E2.CC=E1.CC))
```

◇ Come ultima osservazione sulle interrogazioni formulabili in SQL, notiamo che in questo testo non abbiamo considerato la possibilità, permessa in SQL92, di esprimere condizioni su n -ple di attributi e non solo su un unico attributo. Ad esempio, in SQL92 si può scrivere la condizione $(ACorso, Città)=(2, 'MO')$, equivalente a $(ACorso=2) \text{ AND } (Città='MO')$. Un ulteriore esempio è il seguente:

Es. “Matricole degli studenti che hanno sostenuto un esame insieme (stesso corso, stessa data) alla matricola M2”

```
SELECT Matr
FROM E
WHERE (Data,CC) IN ( SELECT (Data,CC)
                    FROM E
                    WHERE Matr = 'M2')
```

equivalente a:

```
SELECT E1.Matr
FROM E E1
WHERE EXISTS ( SELECT *
               FROM E E2
               WHERE E2.Data = E1.Data
               AND E2.CC = E1.CC
               AND E2.Matr = 'M2' )
```

4.3 Manipolazione dei dati

• Inserimento di record :

◇ Inserimento esplicito di un singolo record:

```
INSERT INTO ;tabella; [;lista-attr;] VALUES ;lista-valori;
```

◇ Inserimento del risultato di una interrogazione:

```
INSERT INTO ;tabella; [;lista-attr;] ;subquery;
```

- I valori riportati in ;lista-valori; devono corrispondere in numero e tipo agli attributi specificati in ;lista-attr;.
- ;lista-attr; deve essere contenuta nella lista di attributi di ;tabella;: i valori corrispondenti agli attributi mancanti vengono posti uguali al valore di default oppure a null

◇ Esempi :

```
INSERT INTO S(Mat, SNome, Città, ACorso)
VALUES ('M1', 'Lucia Quaranta', 'SA', 1)
```

```
INSERT INTO S(Mat, SNome, ACorso)
VALUES ('M1', 'Maria Rossi', 1)
```

```
CREATE TABLE MEDIA
(MATR CHAR(9) NOT NULL PRIMARY KEY,
MEDIA INTEGER );
```

```
INSERT INTO MEDIA
SELECT Matr, AVG(Voto)
FROM E
GROUP BY Matr
```

- Nel caso in cui la ;lista-attr; non sia specificata, essa viene assunta uguale alla lista di attributi nella creazione di ;tabella;:

```
INSERT INTO S
VALUES ('M2', 'Giacomo Tedesco', 'PA', 2)
```

● **Modifica di record :**

```
UPDATE itabellai SET iattri = ipressionei
[WHERE icondizionei]
```

Es. “ Sostituire il codice corso C8 con C10 nella tabella degli esami”

```
UPDATE E
SET      CC='C10'
WHERE   CC='C8'
```

Es. “ Incrementare del 10% i voti inferiori a 24”

```
UPDATE E
SET      Voto=1.1*Voto
WHERE   Voto < 24
```

Es. “ Incrementare di 1 l’anno di corso degli studenti che hanno sostenuto più di 3 esami”

```
UPDATE S
SET      ACorso=ACorso + 1
WHERE   3 < ( SELECT COUNT(*)
              FROM   E
              WHERE  E.Matr=S.Matr)
```

Es. “ Incrementare di 1 l’anno di corso di tutti gli studenti”

```
UPDATE S
SET      ACorso=ACorso+1
```

● **Cancellazione di record :**

```
DELETE FROM itabellai [WHERE icondizionei]
```

Es. “ Cancellare i corsi del docente D1”

```
DELETE FROM E
WHERE   CC in ( SELECT CC
                FROM   C
                WHERE  CD='D1')
```

Es. “ Cancellare tutti i corsi ”

```
DELETE FROM C
```

4.4 Viste, privilegi e cataloghi

Viste

- ◇ Una vista è una tabella “virtuale” definita, tramite un’interrogazione, da altre tabelle base o da altre viste (non sono ammesse però dipendenze ricorsive).

Es. “La vista STUDENTIBIENNIO riporta la matricola, il nome e la città degli studenti con anno di corso minore o uguale a 2”

```
CREATE VIEW STUDENTIBIENNIO(Matricola, Città, ACorso)
AS SELECT Matricola, Città, ACorso
FROM S
WHERE ACorso ≤ 2
```

- ◇ L’interrogazione non viene eseguita all’atto di creazione della vista ma quando la vista viene utilizzata.

◇ **Viste aggiornabili:**

sono possibili operazioni di modifica (INSERT, UPDATE, DELETE), tradotte in operazioni di modifica sulle tabelle base

- ◇ In SQL una vista è aggiornabile solo quando una sola riga di ciascuna tabella di base corrisponde ad una riga della vista.

In particolare, quindi **non** sono aggiornabili viste ottenute:

- tramite GROUP BY e funzioni aggregate
- tramite DISTINCT senza inclusione di una chiave
- tramite riferimenti a viste non aggiornabili

- ◇ In pratica, le viste aggiornabili variano da sistema a sistema, e tipicamente includono quelle definite come proiezioni e/o selezioni di una singola tabella di base; in alcuni sistemi viene anche richiesto che l’insieme di attributi della lista contenga una chiave. La restrizione più comune riguarda la non aggiornabilità di viste definite come join (2 o più variabili nella clausola FROM).

Viste - WITH CHECK OPTION

- ◇ Tale opzione, utilizzabile solo per viste aggiornabili, specifica che, dopo una operazione di inserimento o di modifica, le righe aggiornate devono continuare ad appartenere alla vista.

```
CREATE VIEW STUDENTIBIENNIO(Matricola,Città,ACorso)
AS   SELECT Matricola,Città,ACorso
      FROM   S
      WHERE  ACorso = 2
      WITH CHECK OPTION
```

Lo statement:

```
UPDATE STUDENTIBIENNIO
SET ACORSO = 2
WHERE MATR = M1
```

viene accettato e la tabella base S viene modificata, mentre lo statement:

```
UPDATE STUDENTIBIENNIO
SET ACORSO = 3
WHERE MATR = M1
```

non viene accettato (viene generato un errore e la tabella base S non viene modificata) in quanto la tupla modificata non soddisferebbe più il predicato che definisce la vista.

◇ **WITH [LOCAL — CASCADED] CHECK OPTION**

LOCAL : si controllano solo i predicati locali

CASCADED : si controllano tutti i predicati (default)

```
CREATE VIEW STUDENTIBIENNIOMODENA
AS   SELECT *
      FROM   STUDENTIBIENNIO
      WHERE  Città = 'MO'
      WITH [LOCAL — CASCADED] CHECK OPTION
```

Lo statement:

```
INSERT INTO STUDENTIBIENNIO
VALUES ('M13','MO',3)
```

viene accettato con LOCAL ma non con CASCADED.

Uso delle viste per la formulazione di interrogazioni

◇ Le viste sono utili per:

1. Formulare interrogazioni non esprimibili tramite una singola SELECT.

Ad esempio, per determinare “Il numero medio di esami sostenuti dagli studenti” si dovrebbe:

- (a) raggruppare la relazione E su Matr: GROUP BY Matr
- (b) contare per ogni gruppo il numero esami: COUNT(CC)
- (c) calcolare la media dell’insieme di valori ottenuti in b: AVG(COUNT(CC))

◇ In SQL le funzioni aggregate **non si possono comporre**, quindi la seguente espressione non è corretta:

```
SELECT    AVG(COUNT(CC))
FROM      E
GROUP BY Matr
```

◇ La composizione può essere effettuata usando le viste:

```
CREATE VIEW STOT(Mat,NEsami)
AS SELECT  Mat,COUNT(CC)
FROM      E
GROUP BY Mat
```

```
SELECT AVG(NEsami)
FROM   STOT
```

2. Semplificare interrogazioni complesse.

Es. “Matricola dello studente che ha sostenuto il maggior numero di esami”

```
SELECT  Mat
FROM    E
GROUP BY Mat
HAVING  COUNT(*) >= ALL ( SELECT  COUNT(*)
                          FROM    E
                          GROUP BY Mat)
```

Usando la vista STOT:

```
SELECT Mat
FROM   STOT
WHERE NESAMI = ( SELECT MAX(NESAMI)
                FROM   STOT)
```

Privilegi

- ◇ SQL include il concetto di *identificatore d'autorizzazione* (IA), o *user*, cioè il nome con cui un utente è conosciuto al sistema.
- ◇ L'utente che crea una risorsa ne è il proprietario ed è autorizzato a compiere su di essa qualsiasi operazione. L'utente può specificare quali sono le risorse a cui possono accedere gli altri utenti.
- ◇ Il sistema basa il controllo di accesso sul concetto di **privilegio**: per eseguire un'operazione, l'utente deve avere il privilegio necessario.
- ◇ Ogni privilegio è caratterizzato da:
 1. **risorsa**: tabelle, attributi, viste, domini, indici
 2. **utente che concede il privilegio** (grantor): UC
 3. **utenti che ricevono il privilegio** (grantee): UR_i
 4. **azione permessa sulla risorsa**:
 - **insert** (tabelle e attributi)
 - **update** (tabelle e attributi)
 - **delete** (tabelle)
 - **select** (tabelle; per gli attributi si deve definire una vista)
 - **references** (tabelle e attributi): un utente UR_i può definire, in una sua tabella, una foreign key riferita a tabelle e/o attributi di UC: all'utente UC possono quindi essere vietate alcune operazioni su tali tabelle e/o attributi.
 - **usage** (domini)
 5. **possibilità di trasmettere o meno il privilegio ad altri utenti**
- ◇ In fase di installazione del DBMS, uno user particolare, detto **System Administrator (SA)** riceve il privilegio di eseguire qualsiasi operazione legale nel sistema.

Grant

- ◇ L'utente UC concede su `risorsai` i privilegi specificati in `azionii` agli utenti elencati in `utentii` tramite il comando:

```
GRANT ALL—azionii [ON risorsai]  
TO PUBLIC—utentii [WITH GRANT OPTION];
```

ALL : tutti i privilegi posseduti da UC sulla risorsa

PUBLIC: tutti gli utenti, presenti e futuri, del sistema

WITH GRANT OPTION: L'utente che ha ricevuto il privilegio può trasmetterlo ad altri

- ◇ **Esempi:**

```
SARAi CREATE TABLE STUDENTE  
      (MATRICOLA CHAR(9) NOT NULL PRIMARY KEY  
      CF CHAR(9) NOT NULL UNIQUE  
      ... );  
  
SARAi GRANT UPDATE,SELECT,REFERENCES,REFERENCES(CF)  
      ON STUDENTE TO ROCCO WITH GRANT OPTION;  
  
ROCCOi GRANT UPDATE(CORSO),REFERENCES(CF)  
      ON STUDENTE TO MARIA, CARLO WITH GRANT OPTION;  
  
MARIAi GRANT ALL ON STUDENTE TO PUBLIC  
  
MARIAi CREATE TABLE ESAME  
      (MATRICOLA CHAR(9) NOT NULL REFERENCES STUDENTE  
      ... );  
  
CARLOi CREATE TABLE ESAME  
      (SCF CHAR(9) NOT NULL REFERENCES STUDENTE(CF)  
      ... );
```

- ◇ Si noti che i nomi degli oggetti vengono implicitamente qualificati con i nomi dei proprietari: `MARIA.ESAME` e `CARLO.ESAME`.
- ◇ Se MARIA e/o CARLO fanno degli inserimenti nelle proprie tabelle ESAME, il proprietario della tabella STUDENTE, SARA, non potrà cancellare e/o modificare i record di STUDENTE riferiti in STUDENTE da MARIA e/o CARLO.

Revoke

- ◇ L'utente UC può sottrarre alcuni o tutti i privilegi che aveva precedentemente accordato ad un altro utente tramite il comando:

```
REVOKE ALL—GRANT OPTION—privilegi [ON risorsa]  
FROM PUBLIC —utenti [RESTRIC—CASCADE];
```

GRANT OPTION: si revoca la possibilità di trasmettere ad altri il privilegio

RESTRIC (default): la revoca del privilegio non è possibile se vi sono oggetti o privilegi da esso dipendenti

CASCADE: si forza l'esecuzione del comando e la revoca viene propagata in cascata

- ◇ **Esempio:**

```
SARAi CREATE TABLE STUDENTE  
    ... );
```

```
SARAi GRANT SELECT ON STUDENTE TO ROCCO, CIRO  
    WITH GRANT OPTION;
```

```
ROCCOi GRANT ALL ON STUDENTE TO MATTEO  
    WITH GRANT OPTION;
```

```
CIROi CREATE VIEW V1 AS SELECT ACORSO FROM STUDENTE
```

```
MATTEOi CREATE VIEW V2 AS SELECT SNAME FROM STUDENTE
```

```
SARAi REVOKE GRANT OPTION ON STUDENTE  
    FROM ROCCO RESTRICT;
```

```
SARAi REVOKE SELECT ON STUDENTE  
    FROM CIRO RESTRICT;
```

- ◇ I due comandi di revoca non vengono eseguiti. Invece con:

```
SARAi REVOKE GRANT OPTION ON STUDENTE  
    FROM ROCCO CASCADE;
```

il comando viene eseguito e la revoca si propaga a “cascata”: la vista V2 viene distrutta.

Cataloghi SQL

- ◇ La struttura di un database è memorizzata in un insieme di cataloghi. I cataloghi sono, a loro volta, tabelle e pertanto sono interrogabili in SQL.

Anche se la struttura dei cataloghi è definita solo in parte dallo standard SQL92, tutti le implementazioni gestiscono un proprio insieme di cataloghi.

- ◇ I principali cataloghi sono i seguenti (tra parentesi è riportato il nome utilizzato da INGRES):

Users (iusers): utenti - identificatori di autorizzazione;

Tables (iitables): Tabelle, incluse le viste e gli indici;

Views (iiviews): In aggiunta alle informazioni di Tables c'è lo statement che definisce la vista e l'indicazione di check option;

Indexes (iiindexes): In aggiunta alle informazioni di Tables c'è il tipo di indice e il nome della tabella di base sul quale è definito;

Columns (iicolumns): Colonne di qualsiasi tabella;

Table_Constraint (iiconstraint): Vincoli di tabella;

Key_Column_Usage (iikeys): una o più righe per ogni riga di Table_Constraint che rappresenta un vincolo unique, primary key o foreign key;

Referential_Constraint (iiref_constraint): una riga per ogni riga di Table_Constraint che rappresenta un vincolo Foreign key;

Table_Privileges (iipermits): privilegi di tabella.

- ◇ Principali informazioni sulla struttura dei cataloghi (Ingres):

```
SELECT TABLE_NAME
FROM   IITABLES
WHERE  TABLE_OWNER = '$INGRES'
```

```
SELECT COLUMN_NAME
FROM   IICOLUMNS
WHERE  TABLE_NAME = 'IICOLUMNS'
```

- ◇ Principali informazioni sullo schema del database (Ingres):

```
SELECT TABLE_NAME, TABLE_OWNER, TABLE_TYPE
FROM   IITABLES
WHERE  TABLE_OWNER = '$INGRES'
```

```
SELECT TABLE_NAME, TABLE_OWNER, TABLE_TYPE
FROM   IICOLUMNS
WHERE  TABLE_OWNER = '$INGRES'
```

Cataloghi SQL (Ingres)

- ◇ INGRES definisce automaticamente un indice B⁺tree per le primary key e le foreign key dichiarate nella definizione di una tabella; pertanto in presenza delle seguenti definizioni:

```
CREATE TABLE ESAME
( ...
CONSTRAINT PK_ESAME PRIMARY KEY(MATRICOLA,CCORSO),
CONSTRAINT FK_ES_ST FOREIGN KEY(MATRICOLA) REFERENCES STUDENTE)
```

con la seguente interrogazione

```
SELECT INDEX_NAME, BASE_NAME,
       STORAGE_STRUCTURE, UNIQUE_RULE
FROM   IIINDEXES
```

si ottengono, tra l'altro, le seguenti informazioni:

INDEX_NAME	BASE_NAME	STORAGE_STRUCTURE	UNIQUE_RULE
\$PK_ESAME	ESAME	BTREE	U
\$FK_ES_ST	ESAME	BTREE	D

- ◇ Alcuni cataloghi mantengono **informazioni statistiche** usate dal modulo ottimizzatore per valutare i costi di diverse strategie di esecuzione di una istruzione SQL.
- ◇ In INGRES, il catalogo iistats contiene una riga per ogni colonna per la quale sono note le statistiche. Le informazioni statistiche vengono create ed aggiornate tramite il programma di utilità OPTIMIZEDB jnomedb_i.
- ◇ I principali campi di iistats sono i seguenti:

table_name, table_owner: per identificare la tabella

column_name: il nome della colonna

num_unique: valori distinti nella colonna

rept_factor: il fattore di ripetizione

has_unique: indica se la colonna è una chiave

pct_nulls: percentuale di valori nulli

Capitolo 5

Teoria della Normalizzazione

Questo capitolo riassume la teoria della normalizzazione e le tecniche di ragionamento sulle dipendenze funzionali sviluppate negli anni '70 dalla comunità di ricerca delle basi di dati. Una trattazione più approfondita di tali temi si può trovare in [?, ?].

Il capitolo inizia introducendo il concetto di *dipendenza funzionale* tramite il quale è possibile modellare in maniera astratta le dipendenze tra dati. Vengono quindi presentate le principali tecniche per ragionare sulle dipendenze funzionali, ovvero per dedurre dalle dipendenze funzionali definite esplicitamente dal progettista della base di dati quelle implicate logicamente.

Vengono poi definite le proprietà di qualità di uno schema di relazione introducendo il concetto di *forma normale*. Uno schema di relazione che violi la forma normale desiderata può essere decomposto in sottoschemi allo scopo di raggiungere tale forma normale; si discuteranno le proprietà di tali decomposizioni, quali la preservazione dei dati e la preservazione delle dipendenze funzionali.

5.1 Dipendenze Funzionali

- ◇ Uno degli elementi di conoscenza a disposizione del progettista che deve modellare un'applicazione database è che esistono delle “dipendenze tra dati”. In maniera astratta, questo viene modellato dal concetto di “dipendenza funzionale”.

Convenzioni di Notazione :

1. Le prime lettere dell'alfabeto maiuscole (A, B, C, D, E) denotano un attributo;
2. Le ultime lettere dell'alfabeto maiuscole (U, V, X, Y, Z) denotano insieme di attributi;
3. R denota lo schema di una relazione, r l'istanza corrente dello schema R .
4. Dato un insieme di attributi $A_1 A_2 \dots A_n$, uno schema di relazione R avente tale insieme di attributi verrà denotato indifferentemente con $R(A_1 A_2 \dots A_n)$ oppure con $R(A_1, A_2, \dots, A_n)$.

Definizione 1 (Dipendenza Funzionale) Dato uno schema di relazione $R(X)$, una dipendenza funzionale (FD) su R è un vincolo di integrità espresso nella forma $Y \rightarrow Z$, dove Y e Z sono sottoinsiemi di X ; in tal caso si dice che Y determina funzionalmente Z .

- ◇ Un'istanza r di R soddisfa la dipendenza funzionale $Y \rightarrow Z$ se in ogni tupla di r il valore di Y determina univocamente il valore di Z . Formalmente, dato uno schema di relazione $R(T)$, un'istanza r di $R(T)$ soddisfa il vincolo di dipendenza funzionale $Y \rightarrow Z$ su $R(T)$ se e solo se

$$\forall t_1, t_2 \in r, t_1[Y] = t_2[Y] \rightarrow t_1[Z] = t_2[Z]$$

- ◇ È facile verificare che se $Y \subseteq Z$ allora $Y \rightarrow Z$. Queste dipendenze funzionali vengono dette **banali**.
- ◇ Uno schema $R(T)$ su cui è definito un insieme F di FD verrà denotato con $\langle R(T), F \rangle$.

Definizione 2 (Istanza Legale) Dato uno schema $\langle R(T), F \rangle$, un'istanza r di $R(T)$ è detta istanza legale di $\langle R(T), F \rangle$ se soddisfa tutte le dipendenze funzionali in F .

Dipendenze Funzionali implicate logicamente

◇ Uno schema $\langle R(T), F \rangle$ dà luogo a istanze legali che oltre alle *FD* in F soddisfano necessariamente altre *FD*.

Esempio : In ogni istanza legale dello schema $\langle R(ABC), F = A \rightarrow B, B \rightarrow C \rangle$, vale anche la dipendenza funzionale $A \rightarrow C$ (proprietà *transitiva*).

Infatti, non possono esistere due tuple $t1, t2 \in r$ tali che $t1[A] = t2[A]$ e $t1[C] \neq t2[C]$, in quanto se $t1[A] = t2[A]$, per la $A \rightarrow B$, si ha che $t1[B] = t2[B]$, e quindi per la $B \rightarrow C$ si deve avere necessariamente $t1[C] = t2[C]$.

Definizione 3 (Implicazione Logica) Dato uno schema $\langle R(T), F \rangle$ e una dipendenza funzionale $X \rightarrow Y$, si dice che F implica logicamente $X \rightarrow Y$, denotata con $F \models X \rightarrow Y$, se ogni istanza legale r di $\langle R(T), F \rangle$ soddisfa anche $X \rightarrow Y$.

Definizione 4 (Chiusura di un insieme di FD) Dato uno schema $\langle R(T), F \rangle$, la chiusura di F , denotato con F^+ , è l'insieme delle dipendenze funzionali implicate logicamente da F :

$$F^+ = \{X \rightarrow Y \mid F \models X \rightarrow Y\}$$

◇ Usando la nozione di dipendenza funzionale diamo una nuova formulazione del concetto di *chiave* e *superchiave*. Si vedrà nel seguito come sarà possibile dotarsi di strumenti effettivi di verifica che un insieme di attributi sia una chiave o una superchiave di uno schema di relazione.

Definizione 5 (Chiave) Dato $\langle R(T), F \rangle$, un insieme $K \subseteq T$ è detto *chiave* di $\langle R(T), F \rangle$ se

1. $K \rightarrow T$ è in F^+ ;
2. non esiste nessun sottinsieme proprio Y di K tale che $Y \rightarrow T$ è in F^+ .

◇ Con il termine **superchiave** denotiamo un qualsiasi soprainsieme di una chiave.

5.2 Ragionamento sulle dipendenze funzionali

◇ Per determinare le FD che sono logicamente implicate da quelle esplicite si usano regole di inferenza, note come assiomi di Armstrong [?].

Definizione 6 (Assiomi di Armstrong) Dato uno schema $\langle R(T), F \rangle$, siano X, Y e Z insiemi di attributi contenuti in T .

Riflessività $\{Y \subseteq X\} \vdash X \rightarrow Y$

Estensione $\{X \rightarrow Y\} \vdash XZ \rightarrow YZ$

Transitività $\{X \rightarrow Y, Y \rightarrow Z\} \vdash X \rightarrow Z$

◇ La derivazione sintattica di f da un insieme F di FD attraverso l'applicazione degli assiomi di Armstrong si indica con $F \vdash f$.

Esempio : Sia dato lo schema $\langle R(ABCD), F = \{A \rightarrow C, B \rightarrow D\} \rangle$; la FD $AB \rightarrow ABCD$ è derivabile da F attraverso gli assiomi:

- 1 $A \rightarrow C$ data
- 2 $AB \rightarrow ABC$ applicazione della estensione alla 1 con AB
- 3 $B \rightarrow D$ data
- 4 $ABC \rightarrow ABCD$ applicazione della estensione alla 3 con ABC
- 5 $AB \rightarrow ABCD$ applicazione della transitività alla 2 e 4

◇ Nella derivazione di una FD da un insieme di dipendenze si possono usare, oltre agli assiomi di Armstrong, alcune regole di inferenza aggiuntive:

Unione : $\{X \rightarrow Y, X \rightarrow Z\} \vdash X \rightarrow YZ$

- 1 $X \rightarrow Y$ data
- 2 $X \rightarrow XY$ applicazione della estensione alla 1 con X
- 3 $X \rightarrow Z$ data
- 4 $XY \rightarrow YZ$ applicazione della estensione alla 3 con Y
- 5 $X \rightarrow YZ$ applicazione della transitività alla 2 e 4

Pseudotransitività : $\{X \rightarrow Y, WY \rightarrow Z\} \vdash XW \rightarrow Z$

- 1 $X \rightarrow Y$ data
- 2 $XW \rightarrow WY$ applicazione della estensione alla 1 con W
- 3 $WY \rightarrow Z$ data
- 4 $XW \rightarrow Z$ applicazione della transitività alla 2 e 3

Decomposizione : $\{X \rightarrow Y, Z \subseteq Y\} \vdash X \rightarrow Z$

- 1 $X \rightarrow Y$ data
- 2 $Y \rightarrow Z$ applicazione della riflessività alla $Z \subseteq Y$
- 3 $X \rightarrow Z$ applicazione della transitività alla 1 e 2

◇ Dato un insieme di attributi $Y = A_1A_2 \dots A_n$, una conseguenza notevole è che dalla regola di unione: $\{X \rightarrow Y\} \vdash \{X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n\}$, e da quella di decomposizione: $\{X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n\} \vdash \{X \rightarrow Y\}$.

Ragionamento sulle dipendenze funzionali

- ◇ Informalmente, dato uno schema $\langle R(T), F \rangle$, la chiusura di un insieme di attributi $X \subseteq T$ è l'insieme di attributi derivabili da F mediante gli assiomi di Armstrong.

Definizione 7 (Chiusura di un insieme di attributi) Dato $\langle R(T), F \rangle$, sia $X \subseteq T$ un insieme di attributi; la chiusura di X rispetto a F , denotata con X^+ , è l'insieme di attributi $A \in T$, tali che $X \rightarrow A$ deriva da F tramite gli assiomi di Armstrong:

$$X^+ = \{A \in T \mid F \vdash X \rightarrow A\}$$

Esempio :

Dato $\langle R(ABCDE), F = \{A \rightarrow B, B \rightarrow C, CD \rightarrow E\} \rangle$, la chiusura di AD è $(AD)^+ = ABCDE$:

- 1** $AD \rightarrow A$ applicazione della riflessività
- 2** $AD \rightarrow B$ applicazione della transitività alla **1** e $A \rightarrow B$
- 3** $AD \rightarrow C$ applicazione della transitività alla **2** e $B \rightarrow C$
- 4** $AD \rightarrow D$ applicazione della riflessività
- 5** $AD \rightarrow E$
 - 5a** $AD \rightarrow CD$ applicazione dell'unione alla **3** e **4**
 - 5b** $AD \rightarrow E$ applicazione della transitività alla **5a** e $CD \rightarrow E$

- ◇ Il prossimo teorema stabilisce che il problema di determinare se $X \rightarrow A$ è derivabile, tramite gli assiomi di Armstrong, da un insieme F di FD è risolvibile calcolando la chiusura dell'insieme X .

Teorema 1 Dato $\langle R(T), F \rangle$, siano X e Y insiemi di attributi contenuti in T ,

$$F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$$

Dimostrazione : Supponiamo che sia $Y = A_1 A_2 \dots A_n$,

$F \vdash X \rightarrow Y \Rightarrow Y \subseteq X^+$: Se $F \vdash X \rightarrow Y$, allora per la regola di decomposizione si ha che, per ogni i , $F \vdash X \rightarrow A_i$, quindi $A_i \in X^+$ e pertanto $Y \subseteq X^+$.

$F \vdash X \rightarrow Y \Leftarrow Y \subseteq X^+$: Se $Y \subseteq X^+$, per definizione si ha che, per ogni i , $F \vdash X \rightarrow A_i$, quindi per la regola dell'unione $F \vdash X \rightarrow Y$.

Ragionamento sulle dipendenze funzionali

◇ Un semplice algoritmo (XPIU) per il calcolo di X^+ può essere definito:

Algoritmo XPIU

- **Input :** $\langle R(T), F \rangle$, con $F = \{V_1 \rightarrow W_1, \dots, V_n \rightarrow W_n\}$ e $X \subseteq T$
- **Output :** XPIU(X, F), chiusura di X rispetto a F

begin

XPIU := X ;

repeat

for $k := 1$ **to** n **do**

if $V_k \rightarrow W_k \in F$ **and** $V_k \subseteq \text{XPIU}$ **and** $W_k \not\subseteq \text{XPIU}$

then XPIU := XPIU $\cup W_k$;

until (XPIU **non è cambiato**);

end.

◇ Si può dimostrare che l'algoritmo XPIU termina e calcola correttamente X^+ . Inoltre, l'algoritmo ha una buona complessità proporzionale alla lunghezza di tutte le dipendenze funzionali in F .

Esempio :

Dato $\langle R(ABCDE), F = \{A \rightarrow B, B \rightarrow C, CD \rightarrow E\} \rangle$, sia $X = AD$:

XPIU⁰ = AD

XPIU¹ = XPIU⁰ $\cup B = ADB$, in quanto $A \subseteq \text{XPIU}^0 \wedge A \rightarrow B \in F$

XPIU² = XPIU¹ $\cup C = ADBC$, in quanto $B \subseteq \text{XPIU}^1 \wedge B \rightarrow C \in F$

XPIU³ = XPIU² $\cup E = ADBCE$, in quanto $CD \subseteq \text{XPIU}^2 \wedge CD \rightarrow E \in F$

XPIU⁴ = XPIU³ = X^+

quindi XPIU(AD, F) = $ABCDE$.

Ragionamento sulle dipendenze funzionali

◇ Il prossimo teorema stabilisce che la derivazione sintattica \vdash è equivalente all'implicazione logica \models .

Teorema 2 (Correttezza e completezza degli assiomi di Armstrong).

$F \vdash f \Rightarrow F \models f$: gli assiomi di Armstrong sono **corretti**, cioè ogni FD derivata attraverso l'applicazione degli assiomi è implicata logicamente

$F \vdash f \Leftarrow F \models f$: gli assiomi di Armstrong sono **completi**, cioè ogni FD implicata logicamente è derivabile attraverso l'applicazione degli assiomi

Dimostrazione

$F \vdash f \Rightarrow F \models f$: la dimostrazione di correttezza è immediata. Verifichiamo, ad esempio, la correttezza della regola di Estensione, ovvero che, in ogni istanza r , se è valida $X \rightarrow Y$, vale anche $XZ \rightarrow YZ$. Infatti, non possono esistere due tuple $t1, t2 \in r$ tali che $t1[XZ] = t2[XZ]$ e $t1[YZ] \neq t2[YZ]$, in quanto se $t1[XZ] = t2[XZ]$, per la $X \rightarrow Y$, si ha che $t1[Y] = t2[Y]$, e quindi si deve avere necessariamente $t1[YZ] = t2[YZ]$.

$F \vdash f \Leftarrow F \models f$: questo equivale a dimostrare che $F \not\vdash f \Rightarrow F \not\models f$, ovvero, che se f non deriva da F tramite gli assiomi di Armstrong allora f non è implicata logicamente da F , cioè esiste un'istanza legale r di $\langle R(T), F \rangle$ in cui f non è soddisfatta.

Dato $\langle R(T), F \rangle$, supponiamo che $F \not\vdash X \rightarrow Y$ cioè che dagli assiomi non si possa derivare $X \rightarrow Y$. L'istanza legale r in cui $X \rightarrow Y$ non è soddisfatta è costituita da due sole tuple che hanno gli stessi valori per tutti gli attributi in X^+ e valori differenti per gli altri attributi ($T - X^+$):

Attributi di X^+	Attributi di $T - X^+$
1 1 ... 1	1 1 ... 1
1 1 ... 1	0 0 ... 0

Ora occorre dimostrare r è una istanza legale e che $X \rightarrow Y$ non è soddisfatta in r .

Per dimostrare che r è una istanza legale di $\langle R(T), F \rangle$, si procede per assurdo, supponendo che esista $V \rightarrow W \in F$ non soddisfatta da r . Allora deve essere $V \subseteq X^+$ e $W \not\subseteq X^+$. Poichè $V \subseteq X^+$, si ha che $F \vdash X \rightarrow V$, e per transitività si ottiene che $F \vdash X \rightarrow W$. Di conseguenza $W \subseteq X^+$, e questo contraddice l'ipotesi. Pertanto r è legale.

Per dimostrare che $X \rightarrow Y$ non è soddisfatta in r , basta osservare che per l'ipotesi che $F \not\vdash X \rightarrow Y$, si ha che $Y \not\subseteq X^+$ e quindi $X \rightarrow Y$ non è soddisfatta da r .

Ragionamento sulle dipendenze funzionali

- ◇ Una importante conseguenza del teorema 2 è quella di poter calcolare F^+ in base alla seguente definizione: $F^+ = \{X \rightarrow Y \mid F \vdash X \rightarrow Y\}$
- ◇ Non è proponibile generare la chiusura F^+ , dato l'elevato numero di FD in essa contenute (esponenziale nel numero di attributi).
Si preferisce quindi trovare una soluzione efficiente al problema di determinare se una $FD X \rightarrow A$ appartiene a F^+ .
Tale problema è riconducibile, in base al Teorema 1, al calcolo della chiusura di un insieme di attributi: $X \rightarrow A \in F^+$ **se e solo se** $A \in \text{XPIU}(X, F)$
- ◇ Pertanto, tramite XPIU si possono risolvere i problemi legati alle implicazioni di dipendenze funzionali che stanno alla base delle proprietà di normalizzazione di schemi di relazione: *test di superchiave, sintesi di una chiave, attributi primi, FD ridondanti e equivalenza di insiemi di FD*.
- **Test di superchiave** : K è superchiave di $\langle R(T), F \rangle$ sse $\text{XPIU}(K, F) = T$;
- **Sintesi di una chiave** : Sia K una superchiave di $\langle R(T), F \rangle$; il seguente algoritmo sintetizza una chiave $X \subseteq K$ di $\langle R(T), F \rangle$.

Algoritmo KEY

```

● Input:  $\langle R(T), F \rangle$ , e  $X \subseteq T$ , con  $X = A_1 \dots A_n$  superchiave di  $R(T)$ 
● Output: KEY, chiave di  $\langle R(T), F \rangle$  contenuta in  $X$ 
begin
  KEY := X ;
  for  $i := 1$  to  $n$  do
    if  $A_i \in \text{XPIU}(\text{KEY} - A_i, F)$ 
      then KEY = KEY -  $A_i$ ;
end.
```

- **Attributo primo** : Verificare se l'attributo appartiene ad una delle chiavi calcolate con l'algoritmo KEY.
- **FD ridondanti** : Dato $\langle R(T), F \rangle$, una $FD X \rightarrow Y \in F$ è *ridondante* se è implicata logicamente dalle altre. In termini di XPIU: $Y \in \text{XPIU}(X, F - \{X \rightarrow Y\})$.
- **Equivalenza di insiemi di FD** : Dato $R(T)$, due insiemi di dipendenze funzionali F, G definiti su $R(T)$ sono *equivalenti* se e solo se $F^+ = G^+$.

Teorema 3 Dato $R(T)$, due insiemi di dipendenze funzionali F, G definiti su $R(T)$ sono equivalenti se e solo se $F \subseteq G^+$ e $G \subseteq F^+$. In termini di XPIU: per ogni $X \rightarrow Y \in F$ si ha $Y \in \text{XPIU}(X, G)$ e per ogni $X \rightarrow Y \in G$ si ha $Y \in \text{XPIU}(X, F)$.

Dimostrazione : Si veda [?] oppure [?].

5.3 Forme Normali

- ◇ Le Forme Normali sono alla base della *teoria della normalizzazione* di schemi relazionali, il cui obiettivo principale è quello di definire formalmente la *qualità degli schemi*.
- ◇ La qualità di uno schema viene essenzialmente definita come l'**assenza** di:
 - **ridondanza nei dati**
 - **anomalie di aggiornamento dei dati.**

Prima Forma Normale - 1NF : La 1NF nasce in realtà dall'esigenza di introdurre un modello dei dati particolarmente semplice, *piatto*, che sta alla base di semplici linguaggi di interrogazione e manipolazione.

Definizione 8 (1NF) Uno schema $R(X)$ è in **1NF** se e solo se i valori di tutti i domini degli attributi $A \in X$ sono atomici.

- ◇ Normalmente, nel modello relazionale, si assume come implicito il vincolo di avere domini atomici per gli attributi.
- ◇ In altri modelli di dati, quali ad esempio i *modelli ad oggetti*, *modelli relazionali estesi* e *modelli relazionali ad oggetti*, il vincolo di avere domini atomici è rimosso e un attributo può avere come valore, oltre ad un valore elementare, anche un valore complesso, quale ad esempio una tupla, un insieme oppure una loro combinazione.
- ◇ **Esempio:** Consideriamo lo schema di relazione
 $ESAMI(\underline{MATR}, \underline{CODCOR}, VOTO, GIORNO, MESE, ANNO)$ e la sua istanza

<u>MATR</u>	<u>CODCOR</u>	VOTO	GIORNO	MESE	ANNO
Matr1	CodCor1	29	10	Luglio	1994
Matr1	CodCor3	24	15	Giugno	1995
Matr2	CodCor1	27	21	Febbraio	1997
Matr2	CodCor2	30	12	Luglio	1998

allora, in un modello relazionale esteso, la stessa informazione potrebbe essere espressa come:

<u>MATR</u>	ESAMI
	set of (CODCOR,VOTO,(GIORNO,MESE,ANNO))
Matr1	{ (CodCor1,29,(10,Luglio,1994)), (CodCor3,24,(15,Giugno,1995)) }
Matr2	{ (CodCor1,27,(21,Febbraio,1997)), (CodCor2,30,(12,Luglio,1998)), }

Seconda Forma Normale (2NF)

- ◇ **Esempio:** Consideriamo lo schema di relazione FREQUENZA
 FREQUENZA(MATR,CODCOR,NUMEROORE,CODDOC)
 e la dipendenza funzionale $CODCOR \rightarrow CODDOC$

<u>MATR</u>	<u>CODCOR</u>	NUMEROORE	CODDOC
Matr1	CodCor1	102	CodDoc1
Matr1	CodCor3	98	CodDoc1
Matr2	CodCor1	111	CodDoc1
Matr2	CodCor2	101	CodDoc2

ridondanza: in tutte le frequenze di un corso si ripete lo stesso docente

anomalia di modifica: se un corso cambia docente si devono modificare tutte le tuple relative alla frequenza di quel corso

anomalia di inserimento: non si può inserire un corso senza inserire almeno uno studente che frequenta

anomalia di cancellazione: se vengono cancellate tutte le frequenze relative ad un corso si perde anche l'informazione sul docente del corso

- I problemi derivano da $CODCOR \rightarrow CODDOC$ che stabilisce la dipendenza di un attributo (CODDOC) **solo da una parte** (CODCOR) della chiave. Questo esempio è una violazione della Seconda Forma Normale.
- ◇ Informalmente, la 2NF serve per definire schemi esenti dai problemi derivanti dalla dipendenza *parziale* di un attributo dalla chiave.

Nelle definizioni successive di forme normali si assume sempre, senza perdita di generalità, uno schema $\langle R(T), F \rangle$, dove F contiene solo dipendenze funzionali del tipo $X \rightarrow A$.

- ◇ Dato uno schema $\langle R(T), F \rangle$, un attributo $A \in T$ e un insieme di attributi $Y \subseteq T$, si dice che A *depende completamente* da Y se $Y \rightarrow A \in F$ e non esiste nessun sottoinsieme proprio Z di Y , $Z \subset Y$, tale che $Z \rightarrow A$.

Definizione 9 (2NF) Uno schema $\langle R(T), F \rangle$ è in **2NF** se e solo se ogni attributo non primo $A \in T$ dipende completamente da ognuna delle chiavi di R .

In principio, le definizioni di forme normali dovrebbero essere date con riferimento a F^+ , cioè anche alle dipendenze implicate logicamente, ma è possibile dimostrare che è sufficiente controllare la non violazione in F per garantirsi la non violazione in F^+ .

Terza Forma Normale (3NF)

- ◇ **Esempio :** Consideriamo lo schema di relazione CORSO
 CORSO(CODCOR,NOME,CODDOC,CODDIP)
 e la dipendenza funzionale CODDOC \rightarrow CODDIP

<u>CODCOR</u>	NOME	CODDOC	CODDIP
CodCor1	AnalisiI	CodDoc1	CodDipA
CodCor3	Biologia	CodDoc1	CodDipA
CodCor2	Chimica	CodDoc2	CodDipA
CodCor4	Fisica	CodDoc3	CodDipB

ridondanza: in tutti i corsi di un docente si ripete lo stesso dipartimento

anomalia di modifica: se un docente cambia dipartimento si devono modificare tutte le tuple relative ai corsi di quel docente

anomalia di inserimento: non si può inserire un dipartimento senza inserire almeno un corso di un suo docente

anomalia di cancellazione: se vengono cancellati tutti i corsi di un docente si perde anche l'informazione sul dipartimento del docente

- I problemi derivano da CODDOC \rightarrow CODDIP che stabilisce che un attributo *non* superchiave (CODDOC) determina funzionalmente un altro attributo (CODDIP). Questo esempio è una violazione della 3NF.
- ◇ Informalmente, la 3NF serve per evitare problemi derivanti da attributi non superchiave che determinano funzionalmente altri attributi non primi.

Definizione 10 (3NF) Uno schema $\langle R(T), F \rangle$ è in **3NF** se e solo se per ogni dipendenza funzionale non banale $X \rightarrow A \in F$, o X è una superchiave oppure A è primo.

Teorema 4 (3NF \Rightarrow 2NF) Uno schema $\langle R(T), F \rangle$ che è in **3NF** è anche in **2NF**.

Dimostrazione : Equivale a dimostrare che **not 2NF** \rightarrow **not 3NF**, ovvero che uno schema che non è in **2NF** allora non è in **3NF**. Se non è in **2NF** esiste una chiave K e un sottoinsieme proprio K' di K , $K' \subset K$, che determina un attributo non primo A , $K' \rightarrow A$. Allora lo schema non è in **3NF** in quanto si ha $K' \rightarrow A$, con K' non superchiave e A non primo.

- ◇ Uno schema in **3NF** non è esente da problemi di ridondanza e anomalie di aggiornamento dei dati.

La **3NF** definisce comunque schemi con un buon livello di qualità e costituisce quindi lo standard da noi adottato per il progetto logico.

Forma Normale di Boyce–Codd (BCNF)

- ◇ **Esempio :** Consideriamo lo schema di relazione CAMPIONATO
 CAMPIONATO(SQUADRA,PARTITA,GIOCATORE,RUOLO)
 con la dipendenza funzionale GIOCATORE \rightarrow SQUADRA

SQUADRA	PARTITA	GIOCATORE	RUOLO
Inter	SerieA2161998	Bergomi	TerzinoDestro
Inter	SerieA1241998	Bergomi	Libero
Inter	SerieA1241998	Moriero	AlaDestra
Bologna	SerieA12121998	Signori	Centravanti
Salernitana	SerieA12121998	Fresi	Libero

ridondanza: l'informazione sulla squadra di un giocatore viene ripetuta

anomalia di modifica: se un giocatore cambia squadra si devono modificare tutte le tuple relative al giocatore stesso

anomalia di inserimento: non si può inserire un giocatore in una squadra senza assegnargli almeno un ruolo in una certa partita;

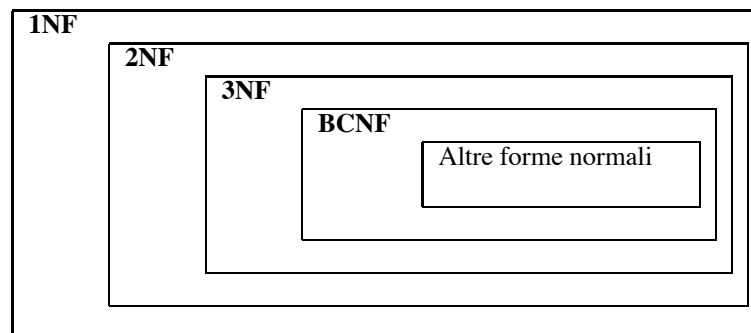
anomalia di cancellazione: se vengono cancellate tutti i ruoli relativi ad un giocatore si perde anche l'informazione sulla squadra del giocatore

- I problemi derivano da GIOCATORE \rightarrow SQUADRA che stabilisce che un attributo *non* superchiave (GIOCATORE) determina funzionalmente un attributo primo (SQUADRA). Questo esempio è una violazione della BCNF.
- ◇ Informalmente, la BCNF afferma che tutti gli attributi (anche quelli primi) dipendono funzionalmente solo dalle superchiavi.

Definizione 11 (BCNF) Uno schema $\langle R(T), F \rangle$ è in **BCNF** se e solo se per ogni dipendenza funzionale non banale $X \rightarrow A \in F$, X è una superchiave.

Teorema 5 (BCNF \Rightarrow 3NF) Uno schema $\langle R(T), F \rangle$ che è in **BCNF** è anche in **3NF**.

Dimostrazione : Segue immediatamente dalle definizioni.



5.4 Decomposizione di schemi

Uno schema di relazione che violi la forma normale desiderata può essere decomposto in sottoschemi allo scopo di raggiungere tale forma normale.

Definizione 12 (Decomposizione di uno schema) Una decomposizione di uno schema $R(T)$ è un insieme di schemi $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$, tale che $T = T_1 \cup T_2 \cup \dots \cup T_n$.

◇ A livello di istanze, decomporre uno schema significa memorizzare, al posto della istanza r dello schema iniziale $R(T)$, le sue proiezioni sui sottoschemi $\pi_{T_i}(r)$. Le proprietà auspicabili per una decomposizione sono sia quella della *preservazione dell'informazione* contenuta nell'istanza r che la *preservazione delle dipendenze funzionali*.

Esempio : Dato lo schema $\langle \text{RUOLI}(S, G, R), F = \{SR \rightarrow G, G \rightarrow S\} \rangle$, dove S , G e R rappresentano rispettivamente gli attributi SQUADRA, GIOCATORE e RUOLO, consideriamo la decomposizione: $\rho = \{R_1(S, R), R_2(G, S)\}$. Intuitivamente, tale decomposizione non preserva i dati, in quanto se consideriamo una istanza legale r di RUOLI e le sue proiezioni sugli schemi decomposti:

r	SQUADRA GIOCATORE RUOLO		
	Italia	Bergomi	TerzinoDestro
	Italia	Bergomi	Libero
	Italia	Moriero	AlaDestra
	Brasile	Ronaldo	Centravanti
	Argentina	Simeone	Centrocampista

$\pi_{SR}(r)$	
SQUADRA	RUOLO
Italia	TerzinoDestro
Italia	Libero
Italia	AlaDestra
Brasile	Centravanti
Argentina	Centrocampista

$\pi_{GS}(r)$	
GIOCATORE	SQUADRA
Bergomi	Italia
Moriero	Italia
Ronaldo	Brasile
Simeone	Argentina

il join naturale $\pi_{SR}(r) \bowtie \pi_{GS}(r)$, effettuato per ricostruire l'istanza r , contiene *pù* tuple di r , come ad esempio la tupla (Italia, Bergomi, AlaDestra) $\notin r$.

Inoltre, ρ_1 non preserva le dipendenze, infatti nell'istanza r di RUOLI, non è possibile inserire la tupla (Argentina, Zanetti, CentroCampista) in quanto verrebbe violata la dipendenza $SR \rightarrow G$ (infatti l'Argentina ha già un CentroCampista). Invece nello schema decomposto l'inserimento andrebbe a buon fine: infatti è possibile inserire le due tuple (Argentina, CentroCampista) e (Zanetti, Argentina) in $\pi_{SR}(r)$ e $\pi_{GS}(r)$ rispettivamente, in quanto non vengono violate le dipendenze dei singoli schemi.

Preservazione dei dati

- ◇ Una decomposizione preserva i dati se una qualunque istanza dello schema iniziale è ricostruibile a partire dalle istanze dei sottoschemi attraverso il join naturale. Formalmente:

Definizione 13 (Decomposizione lossless) Dato uno schema $R(T)$, una sua decomposizione $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$ viene detta senza perdita di informazioni o lossless se e solo se per ogni istanza r di R si ha che

$$r = \pi_{T_1}(r) \bowtie \pi_{T_2}(r) \bowtie \dots \bowtie \pi_{T_n}(r).$$

- ◇ Dato uno schema $R(T)$ e una sua qualsiasi decomposizione $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$, si ha che, per ogni istanza legale r di R

$$r \subseteq \pi_{T_1}(r) \bowtie \pi_{T_2}(r) \bowtie \dots \bowtie \pi_{T_n}(r)$$

quindi verificare che ρ sia *senza perdita di informazioni* equivale a verificare che in ρ non siano state aggiunte tuple.

- ◇ Verificare che una generica decomposizione n -arie sia lossless è un problema complesso. D'altra parte, per le decomposizioni binarie, che sono in generale quelle utilizzate in pratica, esiste invece un teorema molto semplice:

Teorema 6 (Decomposizioni Binarie Lossless) Dato uno schema $R(T)$, una sua decomposizione binaria $\rho = \{R_1(T_1), R_2(T_2)\}$ è lossless se e solo se

$$T_1 \cap T_2 \rightarrow T_1 \in F^+ \quad \text{oppure} \quad T_1 \cap T_2 \rightarrow T_2 \in F^+$$

ovvero, se e solo se il join naturale è eseguito su una superchiave di uno dei due sottoschemi.

Esempio : Riprendiamo lo schema RUOLI(S,G,R) dell'esempio precedente; la sua decomposizione $\rho_1 = \{R_1(S, R), R_2(G, S)\}$ non è lossless in quanto il join naturale viene eseguito su S che non è superchiave in nessuno dei due sottoschemi. Invece, la decomposizione $\rho_2 = \{R_1(G, R), R_2(G, S)\}$ è lossless in quanto il join naturale viene eseguito sull'attributo G che è superchiave in R_2 .

Preservazione delle dipendenze

- ◇ Intuitivamente, una decomposizione preserva le dipendenze se dall'insieme delle dipendenze relative ai sottoschemi è possibile ottenere l'insieme delle dipendenze iniziali.
- ◇ Per definire formalmente la proprietà di preservazione delle dipendenze funzionali, si introduce il concetto di *proiezione di un insieme di dipendenze*:

Definizione 14 (Proiezione di F) Dato uno schema $\langle R(T), F \rangle$ e un sottoschema $T_i \subseteq T$, si definisce proiezione di F su T_i , denotato con $\pi_{T_i}(F)$, l'insieme di dipendenze funzionali:

$$\pi_{T_i}(F) = \{X \rightarrow Y \mid X \rightarrow Y \in F^+, XY \subseteq T_i\}$$

- ◇ Si noti che $\pi_{T_i}(F)$ è definita considerando le dipendenze funzionali di F^+ , non solo quelle di F , come è evidente dall'esempio che segue:

Esempio : Dato lo schema $\langle R(ABCD), F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\} \rangle$ e la sua decomposizione $\rho = \{R_1(AB), R_2(BC), R_3(CD)\}$, si ha che:

$$\pi_{AB}(F) = \{A \rightarrow B, B \rightarrow A\}$$

$$\pi_{BC}(F) = \{B \rightarrow C, C \rightarrow B\}$$

$$\pi_{CD}(F) = \{C \rightarrow D, D \rightarrow C\}$$

Definizione 15 (Decomposizione che preserva le dipendenze) Dato uno schema $\langle R(T), F \rangle$, una sua decomposizione $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$, preserva le dipendenze se e solo se:

$$\left(\pi_{T_1}(F) \cup \pi_{T_2}(F) \cup \dots \cup \pi_{T_n}(F) \right)^+ = F^+$$

- ◇ In base a tale definizione, per verificare che una decomposizione preservi le dipendenze, si dovrebbe calcolare F^+ e poi proiettarlo su ciascun T_i per ottenere $G = \pi_{T_1}(F) \cup \pi_{T_2}(F) \cup \dots \cup \pi_{T_n}(F)$. Inoltre si dovrebbe calcolare la chiusura G^+ per verificare l'equivalenza con F^+ .
- ◇ Per verificare che $G^+ = F^+$, siccome $G^+ \subseteq F^+$, è sufficiente verificare che $F^+ \subseteq G^+$, e a tale scopo è sufficiente controllare che ogni dipendenza in F sia anche in G^+ . Questo equivale a verificare che, per ogni $X \rightarrow Y \in F$, la chiusura di X calcolata rispetto a G contenga Y .

Preservazione delle dipendenze: algoritmo XPIUG

- ◇ Per determinare la chiusura di un insieme di attributi X rispetto a G , senza calcolare G in modo esplicito, ma utilizzando F , è stato proposto il seguente algoritmo [?] che ha complessità polinomiale nella dimensione di F :

Algoritmo XPIUG

- **Input** : decomposizione di $\langle R(T), F \rangle$ con T_1, T_2, \dots, T_n e $X \subseteq T$
- **Output** : XPIUG

begin

XPIUG := X ;

repeat

for $i := 1$ **to** n **do**

XPIUG = XPIUG \cup (XPIU(XPIUG $\cap T_i, F$) $\cap T_i$);

until (XPIUG **non è cambiato**);

end.

- ◇ Si noti che XPIU(XPIUG $\cap T_i, F$) $\cap T_i$ individua gli attributi semplici A tali che XPIUG $\cap T_i \rightarrow A \in \pi_{T_i}(F)$, cioè nell'algoritmo viene calcolata *ripetutamente* la chiusura di X rispetto alle proiezioni di F .
- ◇ Si può dimostrare che l'algoritmo XPIUG termina e calcola correttamente X^+ rispetto a G , cioè a $\cup \pi_{T_i}(F)$.
- ◇ Pertanto, se per ogni dipendenza funzionale $X \rightarrow Y \in F$, Y è un sottoinsieme di X^+ rispetto a G (valutato con l'algoritmo XPIUG) ρ preserva le dipendenze, altrimenti non le preserva.

Esempio : Riprendiamo la decomposizione $\rho_1 = \{R_1(S, R), R_2(G, S)\}$ dello schema $\langle \text{RUOLI}(S, G, R), F = \{SR \rightarrow G, G \rightarrow S\} \rangle$ e verifichiamo che non viene preservata la dipendenza funzionale $SR \rightarrow G$:

$$\text{XPIUG}^0 = SR$$

$$\begin{aligned} \text{XPIUG}^1 &= SR \cup (\text{XPIU}(SR \cap SR, F) \cap SR) \cup (\text{XPIU}(SR \cap GS, F) \cap GS) \\ &= SR \cup (SRG \cap SR) \cup (\emptyset \cap GS) = SR \end{aligned}$$

Poichè $G \notin \text{XPIUG}(F, \rho_1, SR)$, $SR \rightarrow G$ non appartiene alla chiusura di SR rispetto a $\cup \pi_{T_i}(F)$ e quindi le dipendenze non sono preservate in ρ_1 .

- ◇ Dato uno schema $\langle R(T), F \rangle$, è possibile verificare che se una decomposizione $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$ *proietta* una dipendenza funzionale $X \rightarrow Y \in F$ su almeno un sottoschema, cioè se esiste i tale che $XY \subseteq T_i$, allora ρ preserva tale dipendenza funzionale. Quindi se ρ proietta ogni $f \in F$ su almeno un sottoschema, ρ preserva le dipendenze funzionali.

Osservazioni sulle decomposizioni

- ◇ Le proprietà di preservazione dei dati e di preservazione delle dipendenze sono *ortogonali*: vi sono decomposizioni che preservano i dati ma non le dipendenze, e viceversa.

Consideriamo ad esempio lo schema:

$$\langle R(ABCD), F = \{AB \rightarrow C, A \rightarrow D, DB \rightarrow C\} \rangle$$

e le seguenti decomposizioni:

- $\rho_1 = \{R_1(AD), R_2(BCD)\}$
 1. **non preserva i dati**, in quanto il join naturale viene eseguito su D che non è superchiave in nessuno dei due sottoschemi.
 2. **preserva le dipendenze**, in quanto $A \rightarrow D$ è preservata perchè proiettata su R_1 , $DB \rightarrow C$ è preservata perchè proiettata su R_2 , e queste due dipendenze implicano logicamente la dipendenza $AB \rightarrow C$ (tramite l'algoritmo XPIUG: $C \in \text{XPIUG}(F, \rho_1, AB)$).
- $\rho_2 = \{R_1(AD), R_2(ABC)\}$
 1. **preserva i dati**, in quanto il join naturale viene eseguito su A che è una superchiave dello schema R_1 .
 2. **non preserva le dipendenze**, in quanto $A \rightarrow D$ è preservata perchè proiettata su R_1 , $AB \rightarrow C$ è preservata perchè proiettata su R_2 , ma $DB \rightarrow C$ non è preservata in quanto $C \notin \text{XPIUG}(F, \rho_2, DB)$.
- $\rho_3 = \{R_1(ADB), R_2(BCD)\}$
 1. **preserva i dati**, in quanto il join naturale viene eseguito su DB che è superchiave in R_2 .
 2. **preserva le dipendenze**, in quanto $A \rightarrow D$ è preservata perchè proiettata su R_1 , $DB \rightarrow C$ è preservata perchè proiettata su R_2 , e queste due dipendenze implicano logicamente $AB \rightarrow C$.

- ◇ In [?] viene presentato il seguente teorema che stabilisce una condizione sufficiente, ma non necessaria, affinché una decomposizione che preserva le dipendenze preservi anche i dati:

Teorema 7 Dato uno schema $\langle R(T), F \rangle$, sia $\rho = \{R_1(T_1), R_2(T_2), \dots, R_n(T_n)\}$ una sua decomposizione che preserva le dipendenze e tale che T_j , per qualche j , è una superchiave per $\langle R(T), F \rangle$. Allora ρ preserva i dati.

5.5 Normalizzazione di uno schema relazionale

◇ Intuitivamente, la normalizzazione di uno schema avviene applicando ricorsivamente il seguente criterio di *decomposizione per proiezione*:

- Uno schema $\langle R(X, Y, A), F \rangle$ che non è nella forma normale desiderata a causa di $X \rightarrow A$ viene decomposto in $R_1(X, Y)$ e $R_2(X, A)$.

◇ La decomposizione complessiva è lossless ma la preservazione delle dipendenze non è garantita e dipende dall'ordine di applicazione delle singole decomposizioni.

Esempio : Riconsideriamo lo schema dell'esempio di pagina 154:

$$\langle R(ABCD), F = \{AB \rightarrow C, A \rightarrow D, DB \rightarrow C\} \rangle$$

L'unica chiave di R è $K_1 = AB$; lo schema quindi non è in BCNF a causa di $A \rightarrow D$. Pertanto consideriamo la decomposizione $R_1(AD)$ e $R_2(ABC)$.

Abbiamo visto che tale decomposizione è lossless, ma non preserva la dipendenza funzionale $DB \rightarrow C$. Siccome entrambi gli schemi di relazione ottenuti sono in BCNF, il processo di decomposizione termina.

D'altra parte, se avessimo considerato la dipendenza funzionale $DB \rightarrow C$, che rende lo schema iniziale non in BCNF, si sarebbe ottenuto $R_1(ADB)$ e $R_2(BCD)$. Abbiamo visto che tale decomposizione è lossless e preserva tutte le dipendenze. Lo schema $R_1(ADB)$ non è in BCNF a causa di $A \rightarrow D$; pertanto si decompone ulteriormente in $R_{11}(AD)$ e $R_{12}(AB)$; anche tale decomposizione è lossless e preserva le dipendenze. In definitiva si ottiene la decomposizione $R_{11}(AD), R_{12}(AB), R_2(BCD)$ che risulta essere lossless e preserva le dipendenze. I sottoschemi ottenuti sono in BCNF.

◇ Usando il criterio di decomposizione per proiezione sono stati realizzati algoritmi di decomposizione per la normalizzazione di schemi in BCNF (si veda [?, ?, ?]), tramite i quali:

qualunque schema può essere decomposto preservando i dati in un insieme di schemi in BCNF.

Gli algoritmi hanno complessità esponenziale in quanto nella decomposizione di R in R_1 e R_2 occorre calcolare le proiezioni di F su R_1 e R_2 .

Normalizzazione di schemi

- ◇ Dato uno schema giustifichiamo con un esempio perchè *non è sempre possibile trovarne una decomposizione in BCNF che preserva le dipendenze*. Riprendiamo lo schema $\langle \text{RUOLI}(\text{S}, \text{G}, \text{R}), F = \{\text{SR} \rightarrow \text{G}, \text{G} \rightarrow \text{S}\} \rangle$
 Una decomposizione di RUOLI, per essere in BCNF, non deve presentare un sottoschema contenente tutti e tre gli attributi; pertanto la dipendenza funzionale $\text{SR} \rightarrow \text{G}$ non può essere in nessuna proiezione $\cup \pi_{T_i}(F)$. Inoltre $\text{SR} \rightarrow \text{G} \notin (\cup \pi_{T_i}(F))^+$, in quanto né S né R, presi separatamente, determinano G.

- ◇ Per la normalizzazione di schemi in 3NF, che stabilisce condizioni meno restrittive rispetto alla BCNF, sono stati realizzati algoritmi tramite i quali:
qualunque schema di relazione può essere decomposto preservando dati e dipendenze in un insieme di schemi di relazione in 3NF.

- ◇ Questi algoritmi (si veda [?, ?, ?]) sono detti di *sintesi*, in quanto partizionano l'insieme delle dipendenze dello schema secondo certi criteri e fanno corrispondere ad ogni elemento della partizione uno schema di relazione.

- ◇ Noi non presentiamo né gli algoritmi di decomposizione né gli algoritmi di sintesi ma nel seguito (soprattutto nella sezione 6.3 relativa agli esercizi) procederemo alla normalizzazione di uno schema $\langle R, F \rangle$ solo in maniera “euristica”, applicando ricorsivamente il criterio di decomposizione per proiezione negli schemi R_1 e R_2 . Inoltre verificheremo generalmente la violazione della forma normale desiderata sugli schemi R_1 e R_2 solo rispetto alle dipendenze di F proiettate sugli schemi di R_1 e R_2 .

Normalizzazione e progettazione E/R

- ◇ La progettazione di basi di dati usando il modello E/R è molto più facile di quella relazionale a partire da un insieme dato di dipendenze funzionali.
- ◇ La traduzione di uno schema E/R “ben progettato” in schema logico relazionale ottenuta applicando le regole di progetto logico fornite nel capitolo 3 produce generalmente uno schema relazionale in 3NF.
- ◇ Per definire la nozione di schema E/R “ben progettato” la teoria della normalizzazione, sviluppata nel modello relazionale, si potrebbe estendere al modello E/R, definendo le forme normali sia per le entità che per le associazioni. In tali definizioni si dovrebbe fare uso del concetto di dipendenza funzionale tra gli attributi di un’entità oppure di una associazione.

La principale difficoltà da affrontare nell’estendere la teoria della normalizzazione dal modello relazionale al modello E/R è il differente modo con il quale si esprime nei due modelli il concetto di identificatore. Tale teoria è presentata in [?].

- ◇ Il modo usuale di procedere è quindi il seguente:
 1. il progettista disegna uno schema E/R;
 2. genera lo schema relazionale applicando le regole viste nel capitolo 3;
 3. “controlla” in maniera euristica eventuali violazioni della 2NF, 3NF e BCNF e le risolve tramite decomposizione per proiezione;
 4. modifica lo schema E/R iniziale in modo tale che vi sia corrispondenza con lo schema relazionale ottenuto al passo 3.
- ◇ Gli esempi seguenti mostrano come quando gli schemi E/R non sono “ben progettati” sorgono problemi di normalizzazione nello schema relazionale generato.

Esempi di schemi E/R che violano la 2NF

Per entità – Il progetto logico del seguente schema E/R:



produce lo schema di relazione:

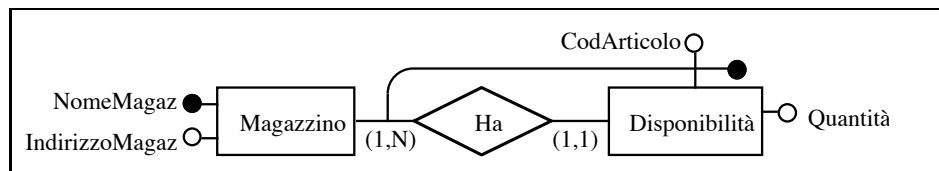
Magazzino(NomeMagaz, CodArticolo, Quantità, IndirizzoMagaz)

che non è in 2NF se $\text{NomeMagaz} \rightarrow \text{IndirizzoMagaz}$; decomponendo:

Magazzino1(NomeMagaz, CodArticolo, Quantità)

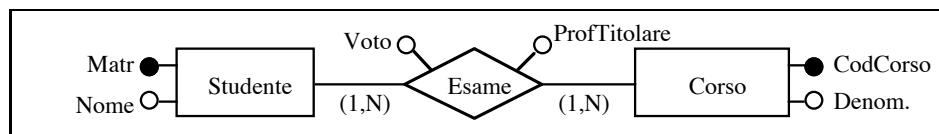
Magazzino2(NomeMagaz, IndirizzoMagaz)

Uno schema E/R ben progettato, se $\text{NomeMagaz} \rightarrow \text{IndirizzoMagaz}$, è:



◇ Il progetto logico di questo schema porta a schemi di relazione che sono in 3NF e non c'è quindi la necessità di effettuare la decomposizione.

Per associazioni – Il progetto logico del seguente schema E/R:



produce lo schema relazionale:

Studente(Matr, Nome)

Corso(CodCor, Denom)

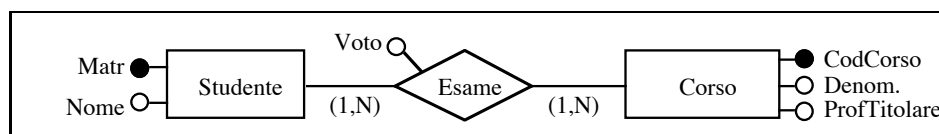
Esame(Matr, CodCor, Voto, ProfTitolare)

Esame non è in 2NF se $\text{CodCor} \rightarrow \text{ProfTitolare}$; decomponendo:

Esame1(Matr, CodCor, Voto)

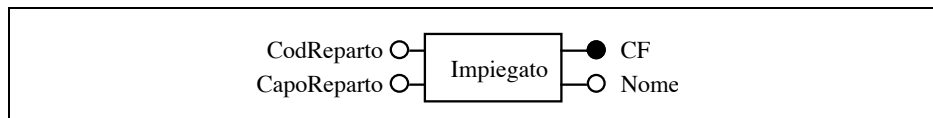
Esame2(CodCor, ProfTitolare)

Uno schema E/R ben progettato, se $\text{CodCor} \rightarrow \text{ProfTitolare}$, è:



Esempi di schemi E/R che violano la 3NF

Per entità – Il progetto logico del seguente schema E/R:



produce lo schema di relazione

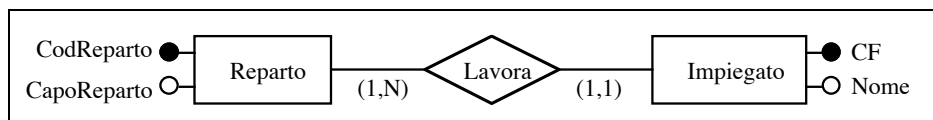
Impiegato(CF, Nome, CodReparto, CapoReparto)

che non è in 3NF se $\text{CodReparto} \rightarrow \text{CapoReparto}$; decomponendo:

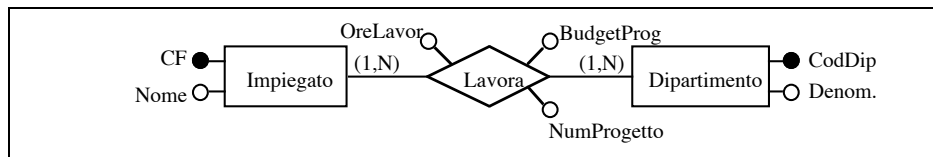
Impiegato1(CF, Nome, CodReparto)

Impiegato2(CodReparto, CapoReparto)

Uno schema E/R ben progettato, se $\text{CodReparto} \rightarrow \text{CapoReparto}$, è:



Per associazioni – Il progetto logico del seguente schema E/R:



produce lo schema relazionale:

Impiegato(CF, Nome)

Dipartimento(CodDip, Denom)

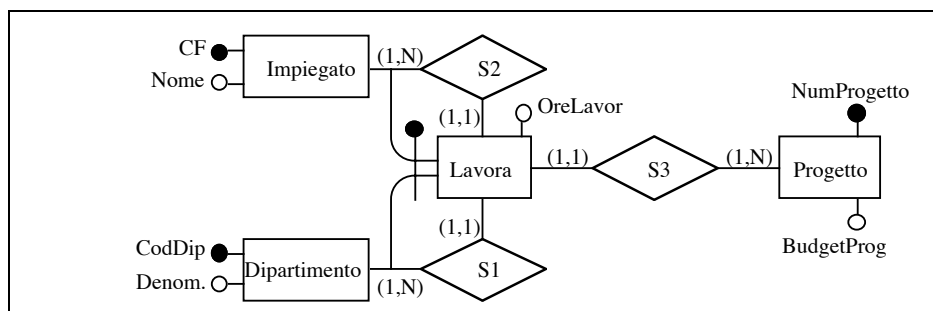
Lavora(CF, CodDip, OreLavor, NumProgetto, BudgetProg)

Lavora non è in 3NF se $\text{NumProgetto} \rightarrow \text{BudgetProg}$; decomponendo:

Lavora1(CF, CodDip, OreLavor, NumProgetto)

Lavora2(NumProgetto, BudgetProg)

Uno schema E/R ben progettato, se $\text{NumProgetto} \rightarrow \text{BudgetProg}$, è:



Capitolo 6

Esercizi

In questo capitolo vengono presentati numerosi esercizi, integralmente risolti, su tutte le tematiche introdotte.

La prima parte contiene esercizi di progettazione concettuale a partire da requisiti in linguaggio naturale e conseguente progettazione logica. Nella soluzione di questi esercizi viene presentato uno tra i possibili schemi E/R corretti corrispondenti alle specifiche date e la relativa traduzione in schema relazionale. In alcune soluzioni, soprattutto quelle relative ai primi esercizi, viene riportato un breve commento allo schema E/R per giustificare le scelte effettuate. Questa prima parte termina proponendo alcuni esercizi relativi ai dati derivati.

La seconda parte contiene esercizi in cui è richiesto di esprimere interrogazioni sia in algebra relazionale che in SQL. Anche nella soluzione di questi esercizi viene presentata una possibile espressione, sia in SQL sia in algebra relazionale, che risolve l'interrogazione richiesta. I commenti sono limitati ad alcune interrogazioni particolarmente complicate.

La terza parte contiene esercizi sulla normalizzazione in cui viene generalmente richiesto, dato uno schema relazionale e un insieme di dipendenze funzionale (esplicite oppure descritte a parole), di determinare se lo schema è in 2NF, 3NF e BCNF e di discutere eventuali decomposizioni.

6.1 E/R e Progetto Logico

Per ciascun esercizio, viene richiesto di:

1. Progettare lo schema E/R
2. Tradurre lo schema E/R in schema relazionale in terza forma normale.

◇ Esercizio 1

Il Consiglio Nazionale delle Ricerche vuole memorizzare dati sui progetti di ricerca secondo le seguenti specifiche. Ogni progetto ha un codice, un nome, l'anno al quale è relativo e una durata. Ogni progetto di ricerca fa riferimento ad una o più discipline: per ognuno di questi riferimenti occorre riportare da 2 a quattro parole chiave. Le discipline sono raggruppate in settori: ogni disciplina è identificata da un codice univoco all'interno del settore di appartenenza ed ha una descrizione; un settore ha un nome (univoco) e un comitato. Ogni progetto di ricerca ha un coordinatore scientifico che è un ricercatore; d'altra parte un ricercatore può essere coordinatore in uno ed un solo progetto. Ad ogni progetto di ricerca partecipano da 1 a 8 ricercatori e per ciascuno di essi viene specificato il numero di "mesi uomo" e il costo orario; in un certo anno, un ricercatore può partecipare al massimo a 2 progetti di ricerca.

◇ Esercizio 2

Un sistema informativo memorizza dati sulle sale cinematografiche e sui film, secondo le seguenti specifiche. Per ogni sala cinematografica viene riportato il nome, la città, l'indirizzo e il numero dei posti; il nome di una sala è univoco solo all'interno della rispettiva città. La programmazione dei film nelle sale cinematografiche è organizzata settimanalmente: per ogni settimana e per ogni sala occorre indicare il singolo film in programma. Per ogni film occorre riportare da 3 a 6 attori protagonisti, con i rispettivi personaggi interpretati: in un dato film, un attore può interpretare più di un personaggio. Per i film vengono riportati il titolo (univoco), il genere, l'anno di produzione, il regista e gli eventuali premi vinti. Attori e registi sono descritti dal codice fiscale, dal nome, dal cognome e dalla nazionalità.

◇ Esercizio 3

Una azienda di assistenza tecnica per Personal Computer vuole memorizzare informazioni sulle riparazioni secondo le seguenti specifiche. Le riparazioni riguardano i componenti hardware dei PC; per ogni tipo di componente viene memorizzato il nome (univoco), il costo e una breve descrizione. Ogni PC è identificato da un codice ed ha un proprietario, descritto tramite il codice fiscale, l'indirizzo e il recapito telefonico. L'assistenza tecnica per la riparazione dei PC avviene memorizzando un resoconto orario organizzato nel seguente modo: in una certa ora di un certo giorno, un tecnico dell'azienda opera su un unico PC riparando uno o più guasti e sostituendo zero o più componenti; si assuma che in una certa ora di un certo giorno, su un PC

operi un solo tecnico. Un tecnico è descritto tramite il codice fiscale, l'indirizzo, la qualifica e il costo orario. Ogni tecnico stila mensilmente una relazione nella quale riporta il numero totale di ore che in quel mese ha dedicato alla riparazione di PC e i tipi di componenti (fino ad un massimo di 5) più sostituiti, con il relativo numero di sostituzioni.

◇ **Esercizio 4**

Un sistema informativo deve gestire le gare di Coppa del Mondo di Sci, secondo le seguenti specifiche.

Gli atleti sono individuati da un numero di tessera FIS e hanno cognome, nome, luogo e data di nascita, nazionalità, sesso. Le gare hanno un luogo, una data, un nome della pista, un tipo (SlalomFemminile, SlalomMaschile, GiganteFemminile, ... , LiberaMaschile). In una certa data, non si possono svolgere due gare dello stesso tipo; in un certo luogo, non si possono svolgere due gare dello stesso tipo.

Gli atleti sono raggruppati in squadre nazionali, rispettivamente maschili e femminili, e ogni squadra ha un allenatore, che è individuato con numero di tessera FIS e ha cognome, nome, luogo e data di nascita, nazionalità. Un allenatore allena un'unica squadra.

Ogni gara ha un tracciatore, che è un allenatore; per le gare in due manche il tracciatore è diverso per ogni manche.

Per ogni partecipazione di un atleta a una gara si registra la posizione di arrivo ed il tempo finale; per le gare in due manche si registra anche il tempo di prima manche.

◇ **Esercizio 5**

Il sistema informativo di una facoltà memorizza dati sulle attività di studio all'estero degli studenti, secondo le seguenti specifiche. Esistono bandi di concorso, ciascuno dei quali è identificato da un numero intero, dall'anno accademico e dal paese europeo in cui l'attività di studio deve essere svolta. Ogni bando ha un unico professore che ne è responsabile ed è associato ad almeno un requisito sufficiente per la partecipazione al concorso; ogni requisito è espresso con l'anno di corso e il corso di laurea o di dottorato a cui lo studente deve essere iscritto; ogni corso di laurea ha un consiglio composto da professori e ogni corso di dottorato ha un coordinatore che è un professore. Ogni bando è relativo ad esattamente un'area disciplinare. Le aree disciplinari sono raccolte in insiemi disgiunti detti settori disciplinari. Ogni area disciplinare ha un codice, univoco all'interno del settore disciplinare, e una denominazione. Ogni settore disciplinare ha un codice univoco e una denominazione. Per ogni bando di concorso vengono inoltre specificati gli eventuali esami che possono essere sostenuti dagli studenti durante la loro attività di studio all'estero; ogni esame sostenuto all'estero ha una denominazione univoca all'interno di un paese europeo ed una descrizione; per un dato anno accademico, un esame sostenuto all'estero corrisponde esattamente ad un esame della facoltà (caratterizzato da un codice ed una descrizione).

◇ Esercizio 6

Una società distributrice di Compact Disk vuole memorizzare dati sulla vendita dei CD, sui negozi e sui clienti secondo le seguenti specifiche. Per i CD viene riportato un codice univoco, il titolo e la casa discografica; i CD sono partizionati in singoli, per i quali viene rappresentato solo l'autore, e in compilation, dei quali vengono descritti, per ognuno dei dieci brani in essi contenuti, il titolo e l'autore; il titolo del brano in una compilation è identificativo. Ogni negozio, descritto da un codice (univoco) e da un nome, stila mensilmente una classifica delle vendite dei CD dove riporta per i CD più venduti (fino ad un massimo di venti CD) il relativo numero di vendite. Infine vengono memorizzate informazioni dettagliate sull'acquisto di CD da parte dei clienti: in una certa data, un cliente acquista presso un negozio una certa quantità di un CD. Del cliente si memorizza il suo numero di tessera (univoco) e l'indirizzo.

◇ Esercizio 7

Una galleria d'arte vuole memorizzare dati sulla esposizione di quadri e sul personale secondo le seguenti specifiche.

La galleria d'arte è composta di diverse sale di esposizione, ognuna delle quali è rappresentata tramite un nome (univoco), una descrizione ed un orario di apertura e di chiusura. Ogni sala di esposizione è custodita giornalmente da un custode, in base ad un orario settimanale organizzato nel seguente modo: in un dato giorno della settimana un custode è assegnato ad una unica sala e, viceversa, in un dato giorno della settimana una sala è custodita da un unico custode. Un custode è descritto tramite il codice fiscale e l'indirizzo.

Per un custode vengono riportate le eventuali assenze: per una certa data in cui esso è assente, viene indicata la causa dell'assenza e una persona esterna che lo ha sostituito in quella data. Del personale esterno occorre conoscere, oltre al codice fiscale e all'indirizzo, anche il recapito telefonico; una persona esterna può sostituire uno o più custodi.

La galleria d'arte organizza delle mostre di quadri; una mostra è descritta dal nome, dall'anno, dall'organizzatore e dalla data di inizio e di fine; una mostra con un certo nome non può essere svolta più di una volta nello stesso anno. Per ogni mostra devono essere riportati i quadri esposti con le relative sale di esposizione: un quadro può essere esposto in una o più mostre e la sua sala di esposizione può variare da mostra a mostra. Di un quadro viene riportato un codice identificativo, il nome e l'autore.

◇ Esercizio 8

Si vogliono rappresentare informazioni relative alla gestione di manifestazioni artistiche durante l'estate. Una manifestazione, descritta da un codice e da un nome, consiste di 2 o più spettacoli; ogni spettacolo è descritto da un numero univoco all'interno della manifestazione nella quale è inserito e dall'ora di inizio. Durante uno spettacolo si esibiscono uno o più artisti (un'artista si può esibire al massimo una volta durante lo stesso spettacolo) ricevendo un certo compenso. Un'artista è descritto dal codice SIAE e dal nome d'arte. Per ogni artista si deve indicare necessariamente un altro artista che lo sostituisca in caso di indisponibilità; un'artista può essere indicato come sostituto di più artisti. Per ospitare gli spettacoli vengono adibiti opportuni luoghi; un luogo è caratterizzato da un nome (univoco) e da un indirizzo. Uno spettacolo è ospitato in un unico luogo; inoltre, in una certa data, un luogo può ospitare al massimo 3 spettacoli, sia della stessa manifestazione che di manifestazioni differenti.

- Modificare lo schema relazionale ottenuto per esprimere i seguenti vincoli:
 1. un'artista si può esibire al massimo una volta durante la stessa manifestazione (cioè si può esibire al massimo in un solo spettacolo di una certa manifestazione), ricevendo un certo compenso.
 2. non si possono svolgere due spettacoli della stessa manifestazione nello stesso luogo.

◇ Esercizio 9

Si vogliono memorizzare dati sulle lezioni organizzate da un ente di formazione secondo le seguenti specifiche. Ogni lezione è tenuta da un docente e riguarda un argomento. In una certa data, un docente può tenere una ed una sola lezione ed un argomento può essere trattato in una ed una sola lezione. Per ogni lezione può essere riportato il numero totale degli studenti presenti e le interrogazioni fatte agli studenti con il relativo voto; durante una lezione uno studente può essere interrogato al massimo una volta. Gli studenti sono organizzati in gruppi: un gruppo è descritto da un codice e da un nome ed è costituito da esattamente tre studenti; uno studente appartiene ad uno ed un solo gruppo. Per un dato argomento, un gruppo può effettuare una ed una sola tesina ed è seguito in questa attività da un unico docente. Per ogni tesina deve essere riportata la data di consegna e zero o più parole chiave.

◇ Esercizio 10

Si vogliono rappresentare informazioni relative alle offerte ed agli sconti praticati da una cooperativa di vendita ai propri soci. La cooperativa è costituita da un certo numero di centri di vendita, descritti da un codice e dal rispettivo indirizzo; i centri di vendita sono suddivisi in ipermarket, caratterizzati dal numero delle casse, e minimarket, caratterizzati dal numero dei reparti. In una certa data, un socio (descritto dal codice fiscale, dal nome e dal telefono) sottoscrive l'iscrizione alla cooperativa, in un preciso centro di vendita. I prodotti venduti nei centri di vendita sono caratterizzati da un codice, dal prezzo e da una descrizione. I minimarket offrono alcuni prodotti scontati di una certa percentuale; per un dato prodotto, la percentuale di sconto varia da minimarket a minimarket e da mese a mese. La cooperativa effettua delle offerte che possono essere ritirate dai soci nei vari supermarket; un'offerta è descritta da un codice, dal prezzo e dal periodo di validità ed è costituita da uno o più prodotti, in una determinata quantità. Occorre memorizzare l'ipermarket in cui un socio ritira eventualmente una certa offerta (tale supermarket non è necessariamente il punto vendita in cui il socio ha sottoscritto l'iscrizione); un socio non può usufruire due volte della stessa offerta. Infine, occorre memorizzare, per ogni ipermarket, le dieci offerte più vendute con il relativo numero di vendite.

◇ Esercizio 11

Il sistema informativo di una impresa che produce motocicli memorizza dati su: produzione di motocicli da competizione, disputa di gare internazionali e il personale impiegato in tali attività in base alle seguenti specifiche:

Ogni modello di motociclo ha un codice, un nome e una cilindrata. Ogni membro del personale ha: un nome, cognome, una età e un codice univoco; il personale è diviso in piloti, meccanici e progettisti: nessuno può fare parte di due categorie. Ogni meccanico è assegnato a un pilota; un pilota può avere assegnati al massimo tre meccanici; i piloti hanno un ingaggio; i progettisti hanno un titolo di studio. Un progettista può avere progettato uno o più modelli di motocicli; un modello di motociclo è stato progettato da uno o più progettisti.

Ogni gara ha un nome (esempio: G.P. d'Italia, di Germania, ecc.), un anno e un autodromo in cui è disputata. Una gara non viene disputata sempre nello stesso autodromo, e può essere disputata in più anni, ma viene disputata al massimo una volta sola nello stesso anno in un unico autodromo.

Un pilota partecipa ad una gara guidando un preciso modello di motociclo; per ognuna di queste partecipazioni si deve memorizzare la posizione iniziale e finale del pilota e zero o più problemi riscontrati durante la gara. Un modello di motociclo può essere stato impiegato in una o più gare.

◇ Esercizio 12

Un'azienda di produzione e di distribuzione di energia elettrica vuole memorizzare informazioni sulla rete di distribuzione e sugli utenti secondo le seguenti specifiche. Fanno parte della rete di distribuzione dell'energia un insieme di impianti, di nodi intermedi e di nodi finali. Per ognuno di questi elementi viene riportato un codice, una descrizione e l'indirizzo. Gli impianti sono collegati a uno o più nodi intermedi; a loro volta, i nodi intermedi possono essere collegati, oltre che con gli impianti, ad altri nodi intermedi oppure a nodi finali. I nodi finali sono collegati ad uno o più nodi intermedi. Per ognuno di questi collegamenti viene riportata la distanza e la tensione utilizzata. Completano la rete di distribuzione i contatori degli utenti; un contatore è allacciato ad un solo nodo finale.

Un contatore, rappresentato da un codice univoco e da una tipologia, è assegnato ad un preciso utente; un utente può essere assegnatario di più contatori e in tal caso riceverà una bolletta trimestrale unica per tutti i consumi, riportante l'importo complessivo da pagare e la data di scadenza. L'azienda mantiene comunque, per ciascuna bolletta, un dettaglio, nel quale viene riportato il consumo mensile (cioè per ciascuno dei tre mesi del trimestre) di ciascuno dei contatori conteggiati nella bolletta.

Vengono infine gestite le richieste di nuovi contratti, cioè le richieste di allacciamento di nuovi contatori, memorizzando il nodo finale al quale sarà allacciato il nuovo contatore, il cliente che ha richiesto l'allacciamento ed il mese di inizio servizio; per un dato nodo finale ed un dato mese di inizio servizio vi possono essere al massimo 10 allacciamenti. Un cliente è rappresentato tramite gli usuali dati anagrafici.

◇ Esercizio 13

Si vogliono rappresentare informazioni relative agli accessori ed ai ricambi per autoveicoli. Un costruttore di autoveicoli, descritto dal nome (univoco), indirizzo e telefono, realizza modelli di autoveicolo e produce ricambi originali. Ogni modello di autoveicolo è identificato da una sigla alfanumerica e dal costruttore di autoveicoli che lo realizza. Ogni ricambio originale ha un codice univoco solo nell'ambito del produttore di autoveicoli che lo produce. Un accessorio per autoveicoli ha un codice univoco e una descrizione. Ogni accessorio è sostitutivo oppure non sostitutivo. Nel primo caso può sostituire uno o più ricambi originali, nel secondo nessuno. Ogni ricambio originale può essere sostituito da un preciso accessorio sostitutivo. Un accessorio è prodotto da un produttore di accessori ed è fornito da uno o più fornitori ad un prezzo che varia da fornitore a fornitore. I fornitori sono caratterizzati da un codice e da un indirizzo; i produttori di accessori sono dei fornitori. Ogni accessorio è installabile su un numero arbitrario di modelli di autoveicolo (almeno uno). Per ogni modello di autoveicolo e accessorio installabile su tale autoveicolo, devono essere specificati gli anni di produzione del modello che sono compatibili con l'accessorio.

◇ Esercizio 14

Si vogliono rappresentare informazioni relative ad una catena di montaggio per autoveicoli. Ogni modello di autoveicolo ha un nome univoco e viene assemblato con una

o più fasi di montaggio. Le fasi di montaggio sono distinte in automatiche, semiautomatiche e manuali; ogni fase di montaggio ha un nome e fa parte dell'assemblaggio di un solo autoveicolo. I dipendenti che lavorano alla catena di montaggio sono descritti tramite il codice fiscale e il nome; i dipendenti sono partizionati in operai e tecnici. Una fase di montaggio manuale è eseguita da una e una sola squadra; una squadra può eseguire fino a 2 fasi di montaggio manuale. Ogni squadra è composta da almeno 2 e al più 4 operai ed è capeggiata da esattamente un tecnico. Ogni operaio fa parte di una e una sola squadra; un tecnico può capeggiare una sola squadra. Una fase di montaggio semiautomatica è eseguita da uno o più dipendenti tramite l'uso di macchinari: durante una fase di montaggio semiautomatica un dipendente utilizza uno o più macchinari per un certo numero di ore. Un macchinario ha un codice, un nome ed una descrizione.

◇ **Esercizio 15**

Progettare uno schema E-R per la gestione delle ore di lezione di corsi di specializzazione post-laurea:

Per i docenti, identificati con un codice, si richiede di registrare il nome, cognome, luogo e data di nascita, la residenza, l'area disciplinare, il codice fiscale, il titolo di studio, il telefono ed eventuali note. Ogni corso è rappresentato da un identificatore costituito da un codice assegnato da organi esterni e da un codice interno, inoltre possiede un titolo e una data di inizio e di fine (quest'ultima facoltativa). Un altro soggetto coinvolto è lo studente e un docente potrebbe anche essere stato studente. Per ognuno di essi è richiesta una serie di informazioni anagrafiche: nome, cognome, luogo e data di nascita, la residenza, il codice fiscale, il titolo di studio, il telefono, l'Università di provenienza ed un codice identificativo. Ogni studente può frequentare più corsi (anche in anni diversi) e le relative presenze, raccolte mensilmente, vengono registrate sia in numero di giorni e sia in numero di ore effettive. Anche le ore svolte dai docenti sono raccolte mensilmente. Un docente può svolgere queste ore di lezione per diversi corsi, per più mesi e queste ore possono essere di differenti tipologie. Il compenso per le ore di lezione può cambiare rispetto alla tipologia, al mese, al corso e al docente. Le tipologie di ore sono identificate con un codice e hanno una descrizione ed un coefficiente che indica il grado di importanza. Un docente infine può cambiare mensilmente la sua posizione fiscale (dipendente, libero professionista, etc.) ma non può avere nello stesso mese due posizioni fiscali diverse.

◇ **Esercizio 16**

Un sistema informativo memorizza dati sulle tappe del Giro d'Italia, secondo le seguenti specifiche.

Per ogni tappa, identificata da un numero intero progressivo, viene riportata la data, la lunghezza e la città di partenza. Le tappe si suddividono in tappe in linea e a cronometro; per quelle in linea occorre riportare anche la città di arrivo, mentre per quelle a cronometro si deve riportare il tipo di pista.

Un ciclista partecipa ad una o più tappe del Giro; per ciascuna di queste partecipazioni si devono riportare i modelli di bicicletta utilizzati con i relativi problemi riscontrati. Un modello di bicicletta è rappresentato tramite un codice, un nome, la ditta produttrice e una descrizione.

Per ogni tappa occorre memorizzare una serie di classifiche, ciascuna delle quali ha una denominazione: ad esempio, la classifica generale è denominata “MagliaRosa”, quella a punti è denominata “MagliaCiclamino”. In una certa classifica, un posto è occupato da un preciso ciclista e, viceversa, in una certa classifica un ciclista occupa uno ed un sol posto; ad esempio, nella terza tappa, il ciclista “MarioCipollini” occupa la quarta posizione nella classifica della “MagliaRosa” e la prima posizione in quella della “MagliaCiclamino”. I ciclisti sono rappresentati tramite un numero univoco, un nome e la squadra di appartenenza.

◇ **Esercizio 17**

Una società petrolifera vuole memorizzare dati sulle stazioni di servizio e la distribuzione di carburanti secondo le seguenti specifiche.

Ogni rifornimento giornaliero fa riferimento ad una autobotte e ad un autista che la guida: in una certa data un autista non può effettuare più di un rifornimento e una autobotte non può essere impegnata in più di un rifornimento. In ogni rifornimento vengono trasportati vari tipi di carburante, per ciascuno dei quali si deve memorizzare la relativa quantità trasportata (ad esempio, un certo rifornimento trasporta 10 tonnellate di “BenzinaVerde”, 30 tonnellate di “BenzinaSuper”, e così via).

Ciascun rifornimento rifornisce una o più stazione di servizio, scaricando in ciascuna di esse uno o più dei tipi di carburante trasportati in una certa quantità; (ad esempio, il precedente rifornimento scarica nella stazione “Pioppa” 3 tonnellate di “BenzinaVerde” e 5 di “BenzinaSuper”, nella stazione “ModenaOvest” 4 tonnellate di “BenzinaVerde” e 5 di “BenzinaSuper”, e così via).

Ciascuna stazione di servizio memorizza giornalmente, per ciascun tipo di carburante, la quantità disponibile. In una stazione di servizio lavorano uno o più gestori, un gestore lavora in una sola stazione.

Il personale della società petrolifera, che comprende i gestori e gli autisti, è rappresentato con gli usuali dati anagrafici. Le autobotti atte alla distribuzione sono identificate da una targa ed hanno una descrizione.

◇ **Esercizio 18**

Un sistema informativo per la gestione di una compagnia assicurativa memorizza informazioni relative a polizze, assicurati, rischi assicurati e premi pagati, secondo le seguenti specifiche.

Ogni polizza ha un numero identificativo, un tipo di rischio coperto, un importo massimo di copertura e un premio annuale (che può variare di anno in anno). Una polizza è riferita ad un solo assicurato e può ricoprire uno o più individui. Le polizze di tipo “assicurazione sulla vita” hanno anche uno o più beneficiari.

Le polizze automobilistiche hanno un tipo e ricoprono un veicolo. Un veicolo può essere ricoperto da più di una polizza, anche nello stesso anno, però con il seguente vincolo: in un certo anno, un veicolo può avere una sola polizza di un certo tipo; ad esempio, il veicolo XYZ non può avere nell'anno 1998 due polizze di tipo "Casco". Gli assicurati, gli individui ricoperti e i beneficiari di una polizza sono individuati dal codice fiscale e descritti dagli usuali dati anagrafici.

◇ **Esercizio 19**

Un sistema informativo per la gestione di una banca memorizza informazioni relative ai conti correnti e alle carte bancomat, secondo le seguenti specifiche.

Un conto corrente ha da uno a cinque clienti intestatari; per un dato contocorrente e per un suo cliente intestatario può essere rilasciata una sola carta bancomat, descritta da un codice univoco e da una tipologia. Una data carta bancomat è assegnata ad un unico cliente ed ad un unico conto corrente.

Tramite una carta bancomat è possibile effettuare fino ad un massimo di cinquanta operazioni al mese; per un'operazione, identificata da un numero progressivo, viene memorizzato il codice dello sportello in cui è stata effettuata e il tipo dell'operazione; tra le operazioni vi sono i prelievi, per i quali si memorizza anche l'importo prelevato. Un conto corrente ha un codice univoco e una descrizione. I clienti sono identificati dal codice fiscale e descritti dagli usuali dati anagrafici.

◇ **Esercizio 20**

Una società di vendite per corrispondenza si vuole dotare di un sistema informativo con le seguenti specifiche.

La società pubblica periodicamente un catalogo dei prodotti caratterizzato da un codice, da un nome e da un prezzo di copertina; ogni prodotto trattato ha un codice, un nome e una descrizione; in un certo catalogo, un prodotto può comparire un'unica volta; il prezzo di vendita di uno stesso prodotto può variare da un catalogo all'altro. Tra i prodotti vi sono i piccoli elettrodomestici (per i quali occorre riportare il numero di anni di garanzia e il voltaggio di funzionamento) e i capi di abbigliamento (per i quali si deve riportare la stoffa, i modelli e le taglie). In particolare, le taglie disponibili dipendono dai modelli: ad esempio, per la camicia CAM123, sono previsti i modelli "ManicaLunga" nelle taglie 38,40,42,48 e "ManicaCorta" nelle taglie 50 e 52.

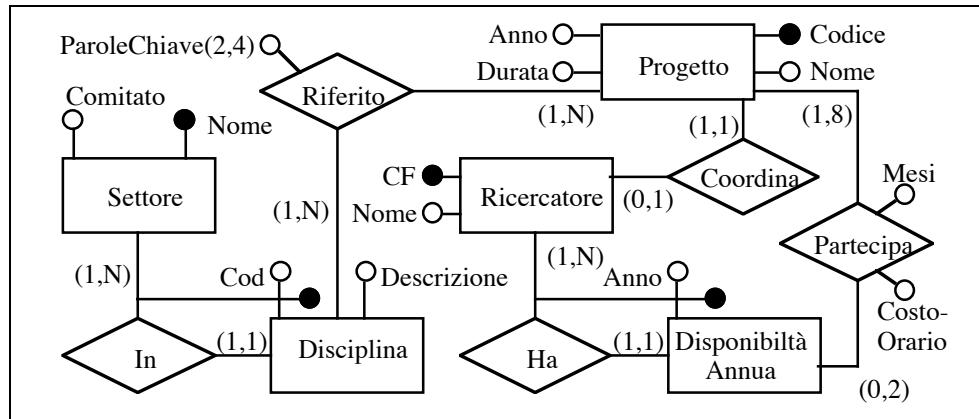
Un cliente effettua una richiesta di acquisto facendo riferimento ai prodotti contenuti in un determinato catalogo, cioè tutti i prodotti compresi in una richiesta devono comparire in uno stesso catalogo; per ciascuno degli articoli compresi in una richiesta viene specificata la quantità ed eventuali note (come, ad esempio, il colore desiderato per un vestito). Una richiesta di acquisto è completata con altre informazioni, quali la data, l'importo complessivo e la modalità di pagamento. Per ogni richiesta evasa la società memorizza la data di invio della merce e la modalità di trasporto.

A scopo promozionale, la società estrae a sorte, per ogni catalogo, un cliente e gli assegna un buono spesa di un determinato importo; un cliente non può essere estratto

più di una volta. Per un cliente si memorizza il CF, il nome, il cognome, l'indirizzo e il numero telefonico.

6.1.1 Soluzioni

Esercizio 1



Commento sullo schema E/R

Se si esprimesse PARTECIPA come associazione binaria tra PROGETTO e RICERCATORE sarebbe un errore riportare

$\text{card}(\text{RICERCATORE}, \text{PARTECIPA}) = (1, 2)$

infatti in questo modo si vincolerebbe il RICERCATORE a partecipare, in tutto, a due progetti.

Il vincolo che un ricercatore partecipa in un anno al massimo a due progetti non si può esprimere nell'associazione binaria PARTECIPA tra PROGETTO e RICERCATORE: per esprimere tale vincolo occorre necessariamente introdurre, come nella soluzione proposta, l'entità DISPONIBILITÀ ANNUA.

Schema Relazionale

RICERCATORE(CF, NOME)

PROGETTO(CODICE, ANNO, DURATA, NOME, CFCOORD) **AK:** CFCOORD

FK: CFCOORD **REFERENCES** RICERCATORE

DISPONIBILITÀ-ANNUA(ANNO, CFRIC)

FK: CFRIC **REFERENCES** RICERCATORE

PARTECIPA(ANNO, CFRIC, CODICEPR, MESI, COSTOORARIO)

FK: ANNO, CFRIC **REFERENCES** DISPONIBILITÀ-ANNUA

FK: CODICEPR **REFERENCES** PROGETTO

SETTORE(NOME, COMITATO)

DISCIPLINA(NOMESET, CODICE, DESCRIZIONE)

FK: NOMESET **REFERENCES** SETTORE

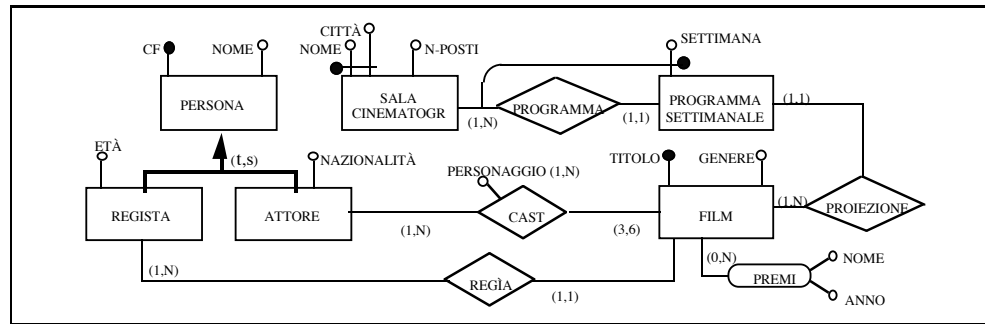
PAROLECHIAVE(NOMESET, CODICE, PAROLACHIAVE, CODICEPR)

FK: NOMESET, CODICE **REFERENCES** DISCIPLINA

FK: CODICEPR **REFERENCES** PROGETTO

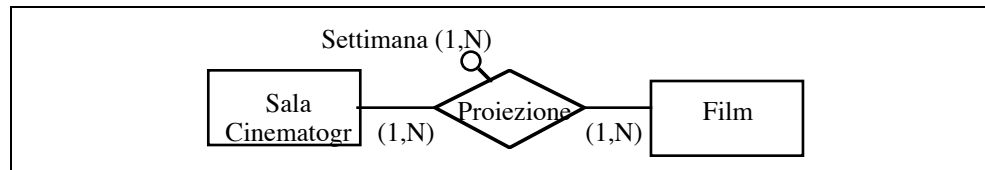
Commento sul progetto logico

L'associazione RIFERITO non è stata tradotta in quanto tutti i riferimenti tra PROGETTO e DISCIPLINA sono rappresentati nella relazione PAROLECHIAVE: per ognuno di questi riferimenti c'è almeno una parola chiave.

Esercizio 2

Il progetto logico di questo schema è a pagina 91.

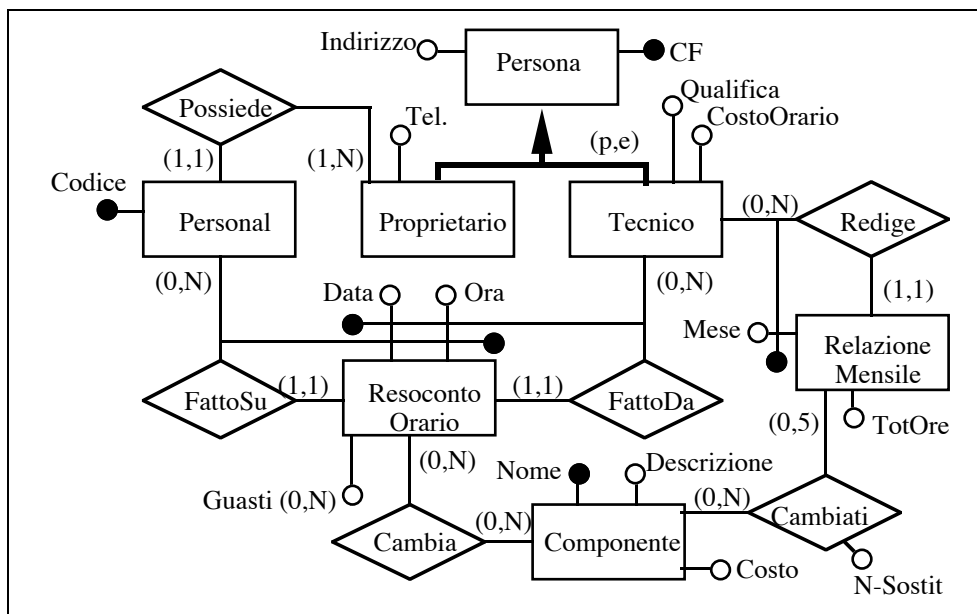
Discutiamo una soluzione alternativa con l'associazione PROIEZIONE tra SALA e FILM:



In questo schema sarebbe un errore mettere $\text{card}(\text{SALA}, \text{PROIEZIONE}) = (1,1)$ in quanto ciò implicherebbe che una sala proietti sempre e solo lo stesso film.

Inoltre l'attributo SETTIMANA deve essere multiplo in quanto una stessa SALA potrebbe proiettare lo stesso FILM in settimane differenti: mettendo SETTIMANA come attributo semplice questo non potrebbe essere espresso (non si può ripetere la stessa coppia in una associazione binaria).

Con questa soluzione non si esprime il fatto che, in una certa settimana, una sala proietta un unico film; d'altra parte questo vincolo può essere aggiunto allo schema relazionale escludendo dalla chiave della relazione SETTIMANA, che traduce l'omonimo attributo multiplo, l'attributo TITOLO.

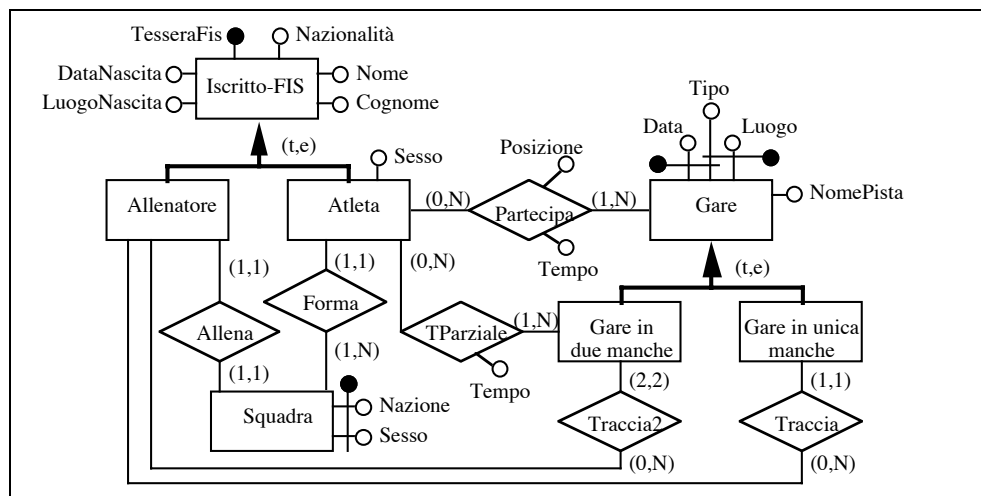
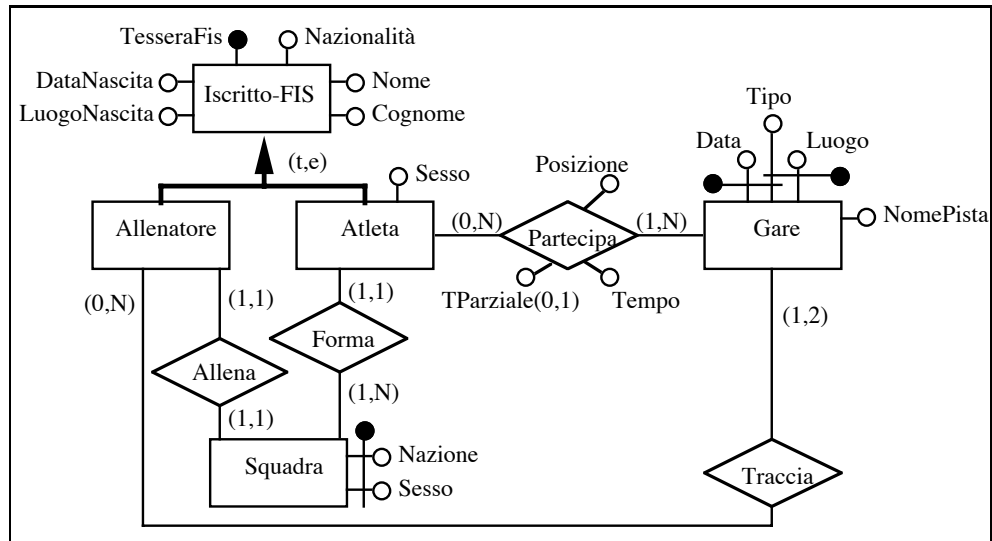
Esercizio 3**Commento sullo schema E/R**

In RESOCONTO-ORARIO l'identificatore (TECNICO, DATA, ORA) assicura che in una certa ora di un certo giorno un tecnico effettui un'unica riparazione riparando, essendo $\text{card}(\text{PC}, \text{RESOCONTO-ORARIO}) = (1,1)$, un unico PC; d'altra parte, l'identificatore (PC, DATA, ORA) assicura che in una certa ora di un certo giorno su un PC operi un unico tecnico.

Schema RelazionalePERSONA(CF,INDIRIZZO)PROPRIETARIO(CF,TELEFONO) **FK: CF REFERENCES PERSONA**TECNICO(CF,QUALIFICA,COSTOORARIO) **FK: CF REFERENCES PERSONA**PERSONAL(CODICE,CF) **FK: CF REFERENCES PROPRIETARIO**RESOCONTO(DATA,ORA,CFTECNICO,CODICEPC) **AK:**

DATA,ORA,CODICEPC

FK: CFTECNICO REFERENCES TECNICO **FK: CODICEPC REFERENCES PERSONAL**GUASTI(DATA,ORA,CFTECNICO,GUASTO) **FK: DATA,ORA,CFTECNICO REFERENCES RESOCONTO**CAMBIA(DATA,ORA,CFTECNICO,NOMECOMP) **FK: DATA,ORA,CFTECNICO REFERENCES RESOCONTO** **FK: NOMECOM REFERENCES COMPONENTE**RELAZIONEMENSILE(CFTECNICO,MESE,TOTORE) **FK: CFTECNICO REFERENCES TECNICO**COMPONENTE(NOME,DESCRIZIONE,COSTO)CAMBIATI(CFTECNICO,MESE,NOMECOMP,NSOSTIT) **FK: CFTECNICO,MESE REFERENCES RELAZIONEMENSILE** **FK: NOMECOM REFERENCES COMPONENTE**

Esercizio 4**Commento sullo schema E/R**

Il vincolo che per le gare in due manche il tracciatore sia diverso per ogni manche è ottenuto dal fatto che nell'associazione TRACCIA non si possono avere due coppie uguali (GARA, ALLENATORE). D'altra parte, con un'unica associazione binaria tra GARA e ALLENATORE non si può discriminare tra le gare in una manche, che hanno un unico tracciatore, e quelle in due manche che hanno due tracciatori. Per esprimere anche questo aspetto si può mettere una gerarchia sull'entità GARA come riportato nel secondo schema E/R.

Effettuiamo il progetto logico del primo schema:

Schema Relazionale

ALLENATORE(TESSERAFIS,NOME,COGNOME,NAZIONALITÀ,DATAN,LUOGON)
 ATLETA(TESSERAFIS,NOME,COGNOME,NAZIONALITÀ,DATAN,LUOGON,NAZIONE,SESSO)
FK: NAZIONE,SESSO **REFERENCES** SQUADRA
 SQUADRA(NAZIONE,SESSO,TESSERAFIS) **AK:** TESSERAFIS
FK: TESSERAFIS **REFERENCES** ALLENATORE
 GARA(DATA,TIPO,LUOGO,NOMEPISTA) **AK:** TIPO,LUOGO
 TRACCIA(DATA,TIPO,TESSERAFIS)
FK: DATA,TIPO **REFERENCES** GARA
FK: TESSERAFIS **REFERENCES** ALLENATORE
 PARTECIPA(DATA,TIPO,TESSERAFIS,POSIZIONE,TEMPO,PARZIALE)
FK: DATA,TIPO **REFERENCES** GARA
FK: TESSERAFIS **REFERENCES** ATLETA

Commento sul progetto logico

Si noti che nell'incorporare l'associazione FORMA nella relazione ATLETA, è stato riportato un'unica volta l'attributo SESSO: in questo modo, un'atleta di sesso femminile (maschile) può essere solo in una squadra femminile (maschile); tale vincolo non era stato espresso a livello di schema E/R.

Effettuiamo il progetto logico del secondo schema, riportando solo le differenze rispetto al precedente schema relazionale:

GARA(DATA,TIPO,LUOGO,NOMEPISTA)
AK: TIPO,LUOGO
 GARAUNICAMANCHE(DATA,TIPO,TESSERAFIS)
FK: DATA,TIPO **REFERENCES** GARA
FK: TESSERAFIS **REFERENCES** ALLENATORE
 GARADUEMANCHE(DATA,TIPO)
FK: DATA,TIPO **REFERENCES** GARA
 TRACCIA2(DATA,TIPO,TESSERAFIS)
FK: DATA,TIPO **REFERENCES** GARADUEMANCHE
FK: TESSERAFIS **REFERENCES** ALLENATORE

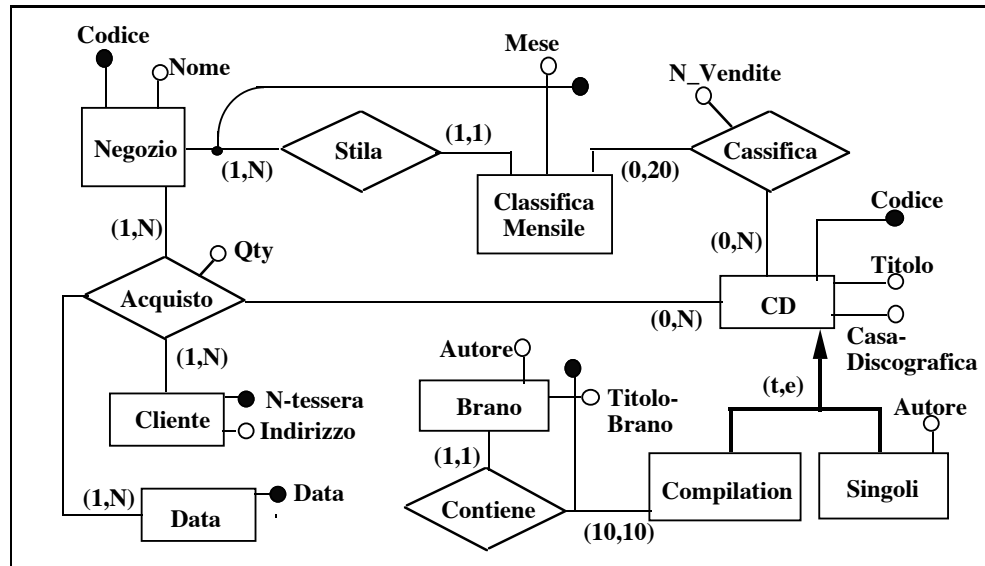
In GARADUEMANCHE si potevano riportare direttamente i due tracciatori: GARADUEMANCHE(DATA,TIPO,TESSERAFIS1,TESSERAFIS2)
FK: TESSERAFIS1 **REFERENCES** ATLETA
FK: TESSERAFIS2 **REFERENCES** ATLETA

In questo modo si perde il vincolo che i due tracciatori siano diversi da manche a manche (vincolo facilmente aggiungibile con TESSERAFIS1 ≠ TESSERAFIS2); d'altra parte si esprime la cardinalità massima di due tracciatori. Si noti che il tempo parziale viene riportato ancora in PARTECIPA. In alternativa, si può introdurre la relazione

TPARZIALE(DATA,TIPO,TESSERAFIS,PARZIALE)
FK: DATA,TIPO,TESSERAFIS **REFERENCES** PARTECIPA

dove la FK assicura che un tempo parziale può essere inserito solo in presenza della relativa partecipazione.

Esercizio 6



Commento sullo schema E/R

Se si esprimesse CLASSIFICA-MENSILE come associazione binaria molti-a-molti tra NEGOZIO e CD non si potrebbe più rappresentare il vincolo che la classifica mensile stilata da un negozio contiene fino ad un massimo di 20 CD; inoltre occorrerebbe riportare un attributo multiplo composto con componenti MESE e N-VENDITE. L'attributo deve essere multiplo perchè uno stesso CD può comparire più volte nella CLASSIFICA-MENSILE dello stesso NEGOZIO: per ognuna di queste occorrenze si riporta il MESE e il relativo N-VENDITE.

Schema Relazionale

NEGOZIO(CODICE, NOME)

CD(CODICECD, TITOLO, CASADISCOGRAFICA)

CLIENTE(NTESSERA, INDIRIZZO)

CLASSIFICAMENSILE(CODICE, MESE)

FK: CODICE REFERENCES NEGOZIO

CLASSIFICA(CODICE, MESE, CODICECD, NVENDITE)

FK: CODICE, MESE REFERENCES CLASSIFICAMENSILE

FK: CODICECD REFERENCES CD

SINGOLI(CODICECD, AUTORE)

FK: CODICECD REFERENCES CD

COMPILATION(CODICECD)

FK: CODICECD REFERENCES CD

BRANI(CODICECD, TITOLOBRANO, AUTORE)

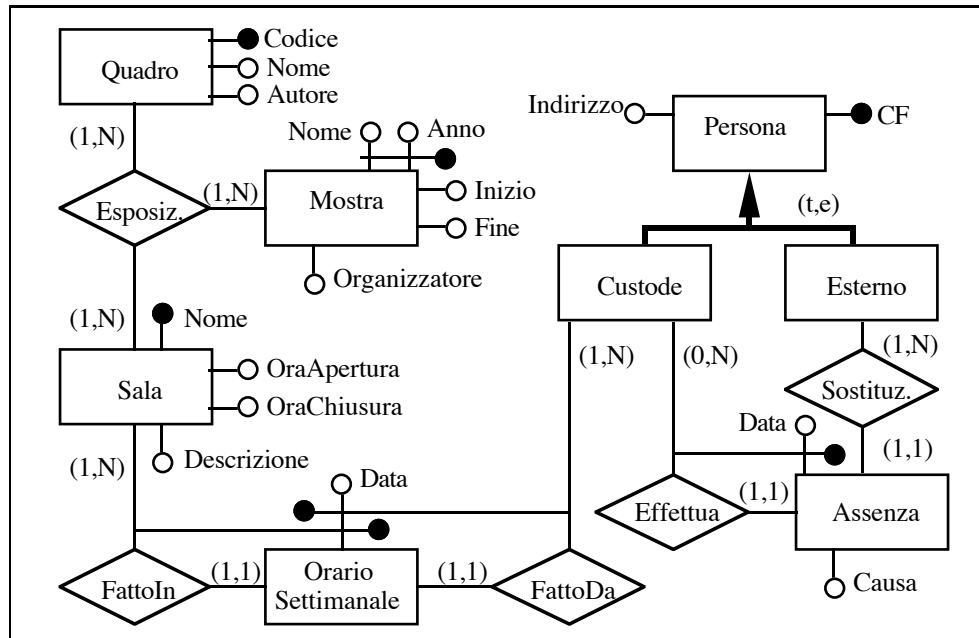
FK: CODICECD REFERENCES COMPILATION

ACQUISTO(CODICE, CODICECD, NTESSERA, DATA, QTY)

FK: CODICE REFERENCES NEGOZIO

FK: CODICECD REFERENCES CD

FK: NTESSERA REFERENCES CLIENTE

Esercizio 7**Commento sullo schema E/R**

Nell'entità ORARIO-SETTIMANALE l'identificatore (SALA, GIORNO) assicura che in un dato giorno della settimana una sala sia custodita da un unico custode; l'identificatore (CUSTODE, GIORNO) assicura che in un dato giorno della settimana un custode custodisca un'unica sala

Un'altra soluzione possibile è quella di esprimere ORARIO-SETTIMANALE come associazione ternaria tra una (nuova) entità GIORNO e le entità SALA e CUSTODE con le seguenti cardinalità:

$\text{card}(\text{SALA}, \text{ORARIO-SETTIMANALE}) = (1, 7)$

$\text{card}(\text{CUSTODE}, \text{ORARIO-SETTIMANALE}) = (1, 7)$

$\text{card}(\text{GIORNO}, \text{ORARIO-SETTIMANALE}) = (1, N)$

In questo caso i vincoli richiesti verrebbero espressi a livello relazionale con:

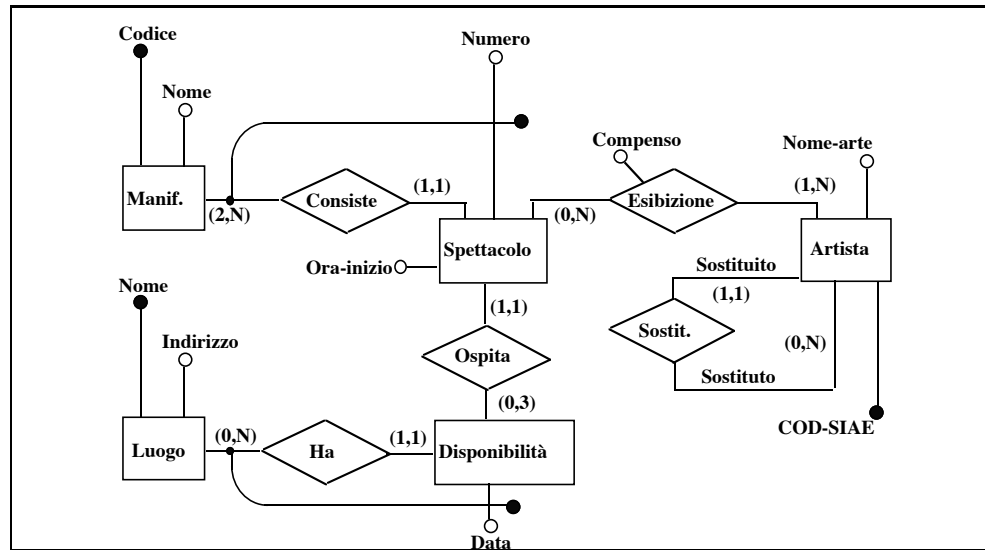
$\text{ORARIO-SETTIMANALE}(\underline{\text{GIORNO}}, \underline{\text{NOME SALA}}, \text{CFCUSTODE})$

AK: GIORNO, CFCUSTODE

Una ulteriore soluzione è quella esprimere ORARIO-SETTIMANALE come associazione binaria tra le entità SALA e CUSTODE riportando GIORNI come attributo multiplo di tale relazione con cardinalità (1,7). Anche in questo caso i vincoli richiesti verrebbero riportati solo a livello relazionale nella relazione che traduce l'attributo multiplo GIORNI.

Il legame tra QUADRO, MOSTRA e SALA è stato rappresentato tramite l'associazione ternaria ESPOSIZIONE in quanto le specifiche non riportano nessun vincolo particolare.

Il progetto logico di questo schema è a pagina 183.

Esercizio 8**Commento sullo schema E/R**

Siccome lo spettacolo ha un numero univoco all'interno della manifestazione nel quale è inserito, il suo identificatore è (MANIFESTAZIONE,NUMERO)

Il vincolo che un'artista si possa esibire solo una volta in uno spettacolo è assicurato dal fatto che nell'associazione binaria ESIBIZIONE non si possono inserire due coppie uguali spettacolo-artista.

Se si esprimesse OSPITA come associazione binaria tra LUOGO e SPETTACOLO sarebbe un errore riportare $\text{card}(\text{LUOGO}, \text{OSPITA}) = (0,3)$: infatti in questo modo si vincolerebbe un luogo ad ospitare, in tutto, tre spettacoli, mentre questo vincolo é valido solo per una certa data. In questo caso si dovrebbe mettere $\text{card}(\text{LUOGO}, \text{OSPITA}) = (0,N)$ e $\text{card}(\text{SPETTACOLO}, \text{OSPITA}) = (1,1)$, e riportare DATA come attributo semplice di OSPITA. Con la semplice associazione binaria OSPITA il vincolo in questione non si può esprimere: per esprimerlo occorre necessariamente introdurre, come nella soluzione proposta, l'entità DISPONIBILITÀ.

Schema Relazionale

MANIFESTAZIONE(CODMAN,NOME)
 LUOGO(NOME-LUOGO,INDIRIZZO)
 DISPONIBILITÀ(NOME-LUOGO,DATA)
 FK: NOME-LUOGO **REFERENCES** LUOGO
 SPETTACOLO(CODMAN,NUMERO,ORA-INIZIO,NOME-LUOGO,DATA)
 FK: CODMAN **REFERENCES** MANIFESTAZIONE
 FK: NOME-LUOGO,DATA **REFERENCES** DISPONIBILITÀ
 ARTISTA(CODSIAE,NOME-ARTE,CODSIAE-SOSTITUTO)
 FK: CODSIAE-SOSTITUTO **REFERENCES** ARTISTA
 ESIBIZIONE(CODSIAE,CODMAN,NUMERO,COMPENSO)
 FK: CODSIAE **REFERENCES** ARTISTA
 FK: CODMAN,NUMERO **REFERENCES** SPETTACOLO

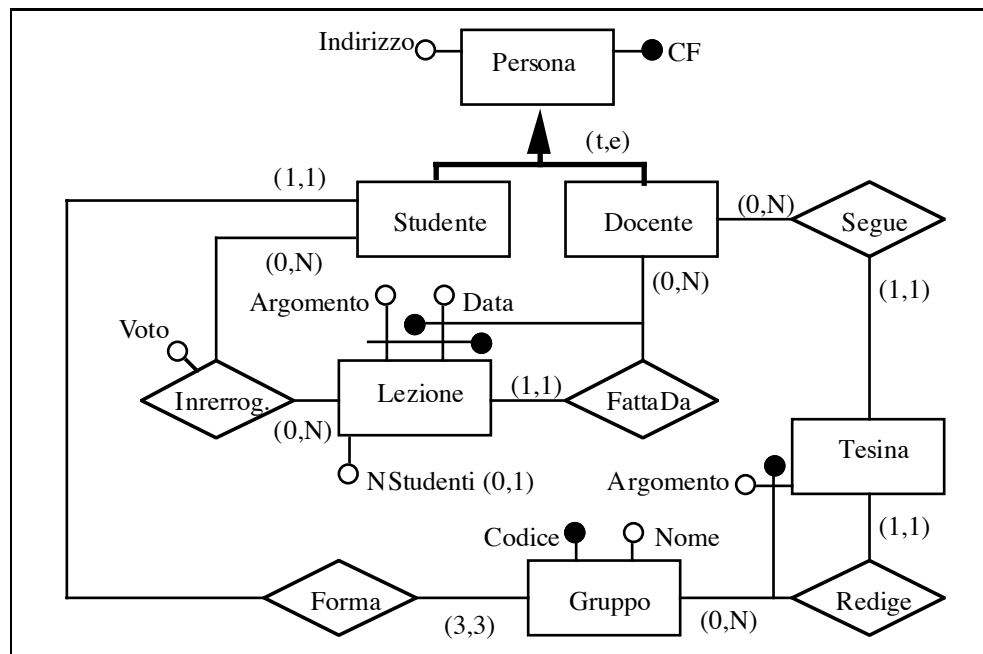
Schema relazionale modificato

Schema Relazionale

ESIBIZIONE(CODSIAE,CODMAN,NUMERO,COMPENSO)
 SPETTACOLO(CODMAN,NUMERO,ORA-INIZIO,NOME-LUOGO,DATA)
 AK: CODMAN,NOME-LUOGO

Progetto Logico relativo all'Esercizio 7**Schema Relazionale**

CUSTODE(CF,INDIRIZZO)
 ESTERNO(CF,INDIRIZZO,TELEFONO)
 ORARIO-SETTIMANALE(GIORNO,NOMESALA,CFCUSTODE)
 ORARIO-SETTIMANALE(GIORNO,NOMESALA,CFCUSTODE)
 AK: GIORNO,CFCUSTODE
 FK: CFCUSTODE **REFERENCES** CUSTODE
 FK: NOMESALA **REFERENCES** SALA
 ASSENZA(CFCUSTODE,DATA,CAUSA,CFSOSTITUTO)
 FK: CFCUSTODE **REFERENCES** CUSTODE
 FK: CFSOSTITUTO **REFERENCES** ESTERNO
 MOSTRA(NOME,ANNO,INIZIO,FINE,ORGANIZZATORE)
 QUADRO(NOME,AUTORE)
 ESPOSIZIONE(NOMEMOSTRA,ANNOMOSTRA,NOMEQUADRO,NOMESALA)
 FK: NOMEMOSTRA,ANNOMOSTRA **REFERENCES** MOSTRA
 FK: NOMESALA **REFERENCES** SALA
 FK: NOMEQUADRO **REFERENCES** QUADRO

Esercizio 9**Commento sullo schema E/R**

Nell'entità LEZIONE l'identificatore (ARGOMENTO, DATA) assicura che per un certo argomento in una certa data venga svolta un'unica lezione; l'identificatore (DOCENTE, DATA) assicura che un certo docente svolga, in una certa data, un'unica lezione.

Schema Relazionale

GRUPPO(CODICE, NOME)

STUDENTE(CF, INDIRIZZO, CODICEGRUPPO)

FK: CODICEGRUPPO **REFERENCES** GRUPPO

DOCENTE(CF, INDIRIZZO)

LEZIONE(DATA, ARGOMENTO, NSTUDENTI, CFDOCENTE)

AK: DATA, CFDOCENTE

FK: CFDOCENTE **REFERENCES** DOCENTE

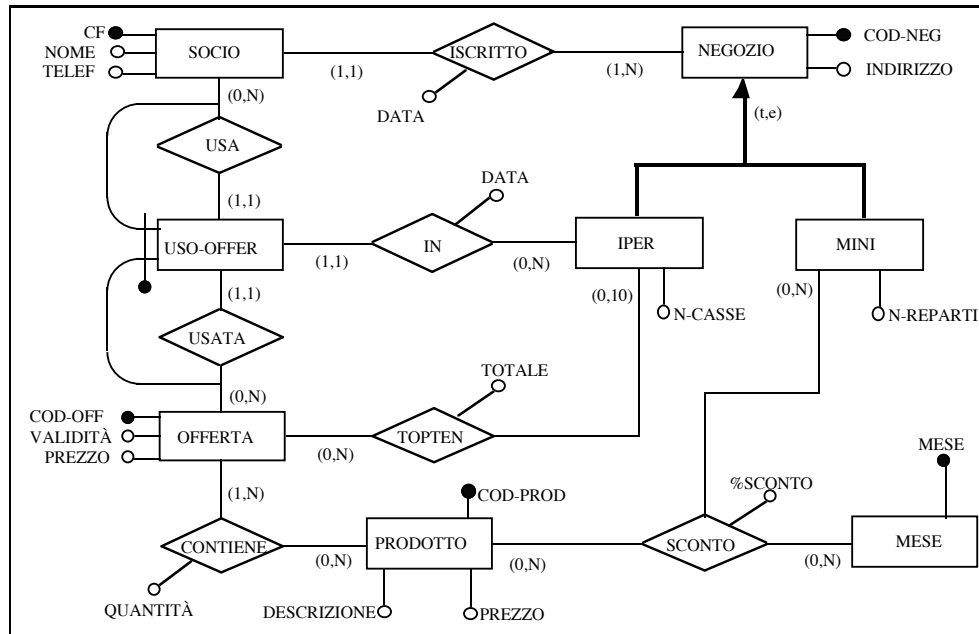
INTERROGAZIONE(DATA, ARGOMENTO, CFSTUDENTE, VOTO)

FK: DATA, ARGOMENTO **REFERENCES** LEZIONE

FK: CFSTUDENTE **REFERENCES** STUDENTE

TESINA(CODICEGRUPPO, ARGOMENTO, CFDOCENTE)

FK: CFDOCENTE **REFERENCES** DOCENTE

Esercizio 10

Commento sullo schema E/R Il fatto che un socio non possa usufruire due volte della stessa offerta, si potrebbe esprimere tramite una associazione binaria (UsoOffer) multi-a-molti tra Socio e Offerta. Però siccome l'ipermarket dove viene effettuato il ritiro dell'offerta non è necessariamente il punto vendita (Negozio) in cui il socio ha sottoscritto l'iscrizione, occorre indicare tale ipermarket in UsoOffer, in quanto esso potrebbe variare da ritiro a ritiro. Per questo motivo l'associazione UsoOffer viene reificata in una entità identificata da Socio e Offerta e tale entità viene collegata all'ipermarket dove avviene il ritiro dell'offerta.

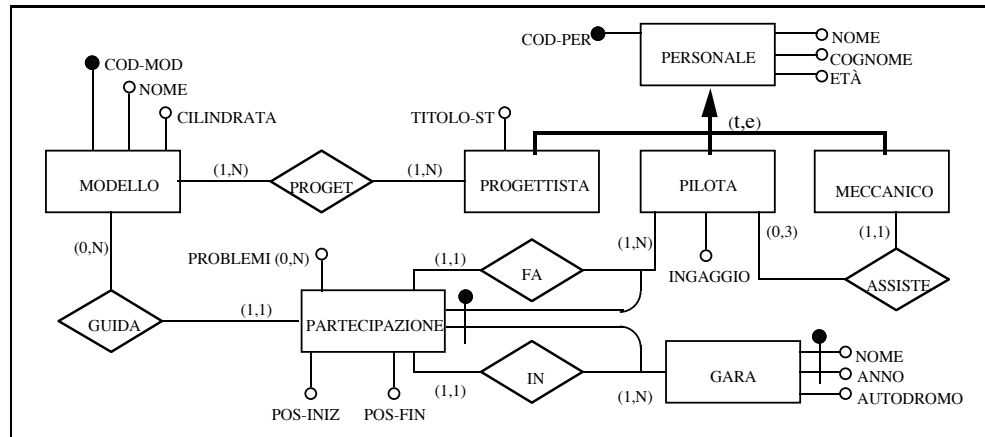
La percentuale di sconto dipende dal prodotto, dal minimarket e dal mese e viene quindi espresso tramite un'attributo sull'associazione ternaria (Sconto).

È possibile aggiungere allo schema una associazione multi-a-molti tra Negozio e Prodotto che rappresenta i prodotti venduti in un certo negozio.

Siccome occorre memorizzare, per ogni ipermarket, le dieci offerte più vendute è possibile mettere $\text{Card}(\text{Iper}, \text{TopTen}) = (10, 10)$. D'altra parte con $\text{CardMin}(\text{Iper}, \text{TopTen}) = 0$ e $\text{CardMin}(\text{Iper}, \text{TopTen}) = N$, si considerano anche i casi, rispettivamente, di offerta non ritirata in nessun ipermarket e di più di dieci offerte a pari merito.

Il progetto logico di questo schema non viene riportato.

Esercizio 11



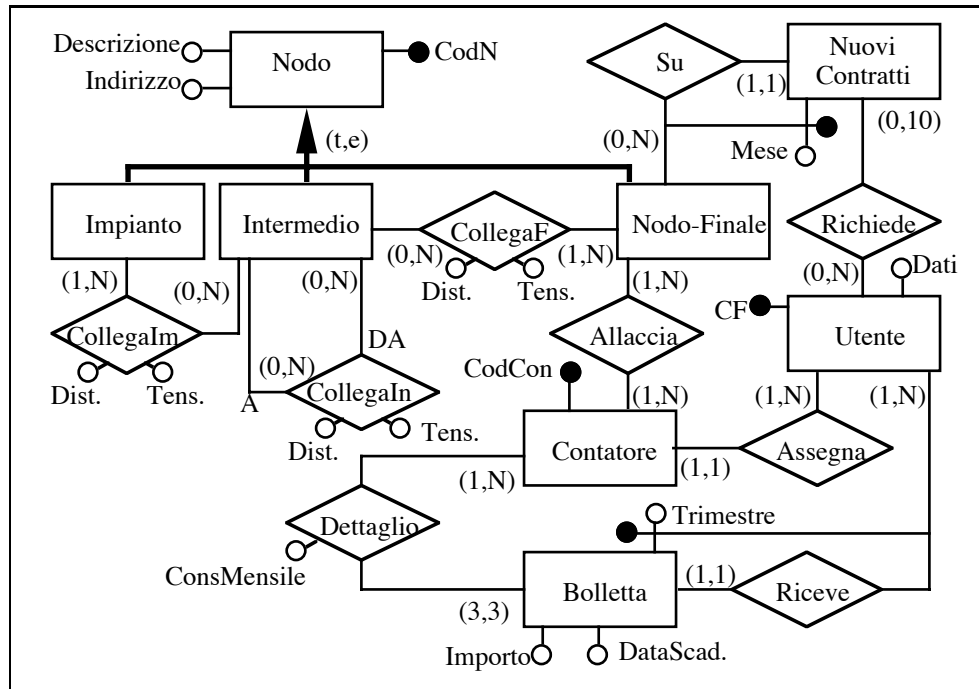
Schema Relazionale

Progettista(CodPer, Nome, Cognome, Età, TitoloSt)Pilota(CodPer, Nome, Cognome, Età, Ingaggio)Meccanico(CodPer, Nome, Cognome, Età, CodPerPil)**FK:** CodPerPil **REFERENCES** PilotaModello(CodMod, Nome, Cilindrata)Progettazione(CodMod, CodPerProget)**FK:** CodMod **REFERENCES** Modello**FK:** CodPerProget **REFERENCES** ProgettistaGara(NomeG, AnnoG, Autodromo)Partecipazione(CodPerPil, NomeG, AnnoG, CodMod, PosIn, PosFin)**FK:** CodPerPil **REFERENCES** Pilota**FK:** NomeG, AnnoG **REFERENCES** Gara**FK:** CodMod **REFERENCES** ModelloProblemi(CodPerPil, NomeG, AnnoG, Problema)**FK:** CodPerPil, NomeG, AnnoG **REFERENCES** Partecipazione

Commento sul progetto logico

Dato che la gerarchia su PERSONALE è totale ed esclusiva e l'entità padre non è coinvolta in nessuna associazione, si risolve tale gerarchia eliminando l'entità padre. Nell'attributo multiplo PROBLEMI si assume che un valore possa comparire una volta sola nella ripetizione: ciò equivale ad assumere che durante una data partecipazione di un pilota ad una gara, un certo problema possa essere riscontrato una volta sola.

Esercizio 12



Schema Relazionale

IMPIANTO(CODN,DESCRIZIONE,INDIRIZZO)

NODOINTERMEDIO(CODN,DESCRIZIONE,INDIRIZZO)

NODOFINALE(CODN,DESCRIZIONE,INDIRIZZO)

COLLEGAIM(DA-CODN,A-CODN,DISTANZA,TENSIONE)

FK: DA-CODN REFERENCES IMPIANTO

FK: A-CODN REFERENCES NODOINTERMEDIO

COLLEGAIN(DA-CODN,A-CODN,DISTANZA,TENSIONE)

FK: DA-CODN REFERENCES NODOINTERMEDIO

FK: A-CODN REFERENCES NODOINTERMEDIO

COLLEGAF(A-CODN,A-CODN,DISTANZA,TENSIONE)

FK: DA-CODN REFERENCES NODOINTERMEDIO

FK: A-CODN REFERENCES NODOFINALE

UTENTE(CF,DATI)

CONTATORE(CODCON)

ALLACCIA(CODCON,CODN) **FK:** CODCON REFERENCES CONTATORE

FK: CODN REFERENCES NODOFINALE

BOLLETTA(CF,TRIMESTRE,DATASCAD,IMPORTO)

FK: CF REFERENCES UTENTE

DETTAGLIO(CF,TRIMESTRE,CODCON,CONSMENSILE)

FK: CF,TRIMESTRE REFERENCES BOLLETTA

FK: CODCON REFERENCES CONTATORE

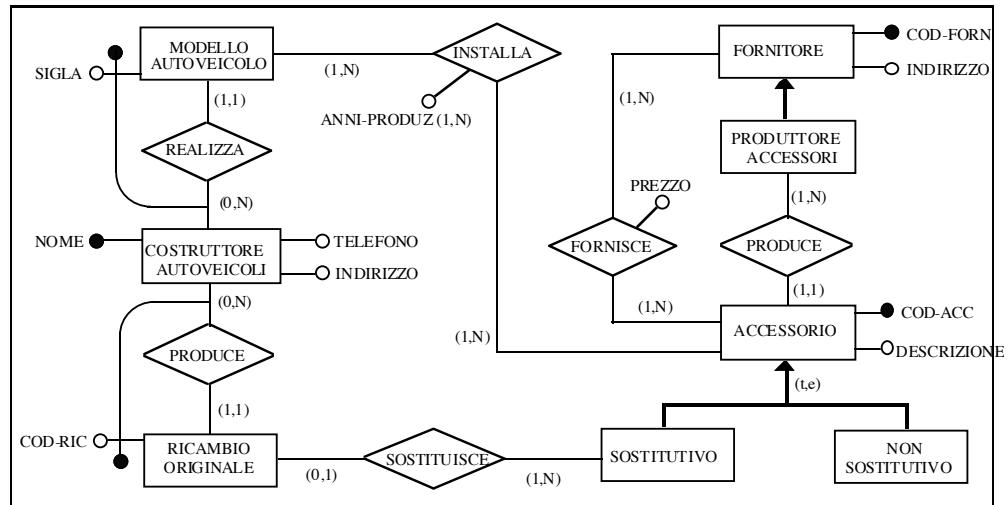
NUOVICONTRATTI(CODN,MESE)

FK: CODCON REFERENCES CONTATORE

RICHIEDE(CF,CODN,MESE) **FK:** CF REFERENCES UTENTE

FK: CODN,MESE REFERENCES NUOVICONTRATTI

Esercizio 13



Schema Relazionale

COSTRUTTOREAUTO(NOME,TELEFONO,INDIRIZZO)

MODELLOAUTO(NOMECOSTRUTTORE,SIGLA)

RICAMBIOORIGINALE(NOMECOSTRUTTORE,CODRIC,CODACCSOSTITUTIVO)

FK: NOME COSTRUTTORE REFERENCES COSTRUTTOREAUTO

FK: CODACC SOSTITUTIVO REFERENCES ACCESSORIO

ACCESSORIO(CODACC,DESCRIZIONE,SOSTITUTIVO,CODPRODUTTORE)

FK: CODPRODUTTORE REFERENCES PRODUTTORE

INSTALLA(CODACC,NOMECOSTRUTTORE,SIGLA,ANNIPRODUZ)

FK: CODACC REFERENCES ACCESSORIO

FK: NOME COSTRUTTORE, SIGLA REFERENCES MODELLOAUTO

FORNISCE(CODACC,CODPRODUTTORE,PREZZO)

FK: CODACC REFERENCES ACCESSORIO

FK: CODPRODUTTORE REFERENCES PRODUTTORE

FORNITORE(CODFORN,INDIRIZZO)

PRODUTTOREACCESSORI(CODPRODUTTORE)

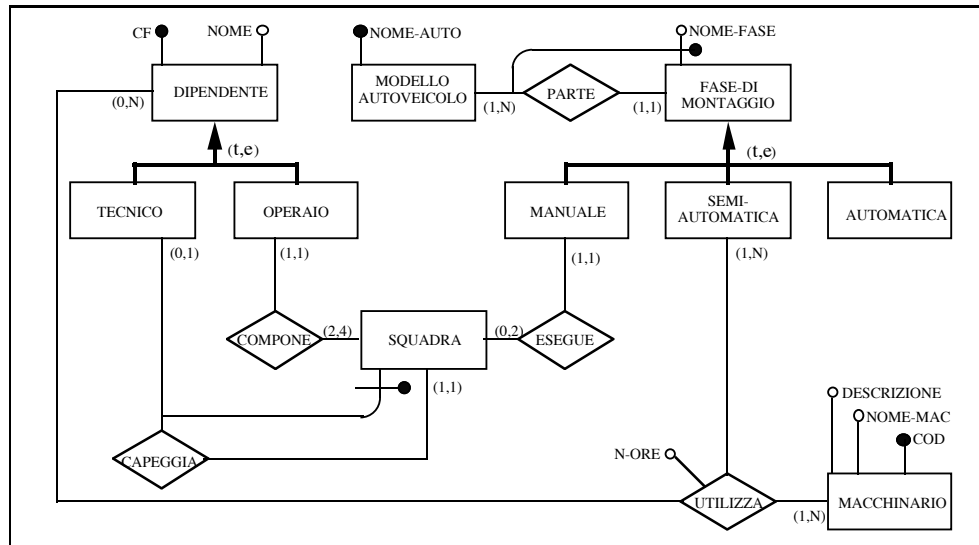
FK: CODPRODUTTORE REFERENCES FORNITORE

PRODUCE(CODPRODUTTORE,CODACC)

FK: CODPRODUTTORE REFERENCES PRODUTTOREACCESSORI

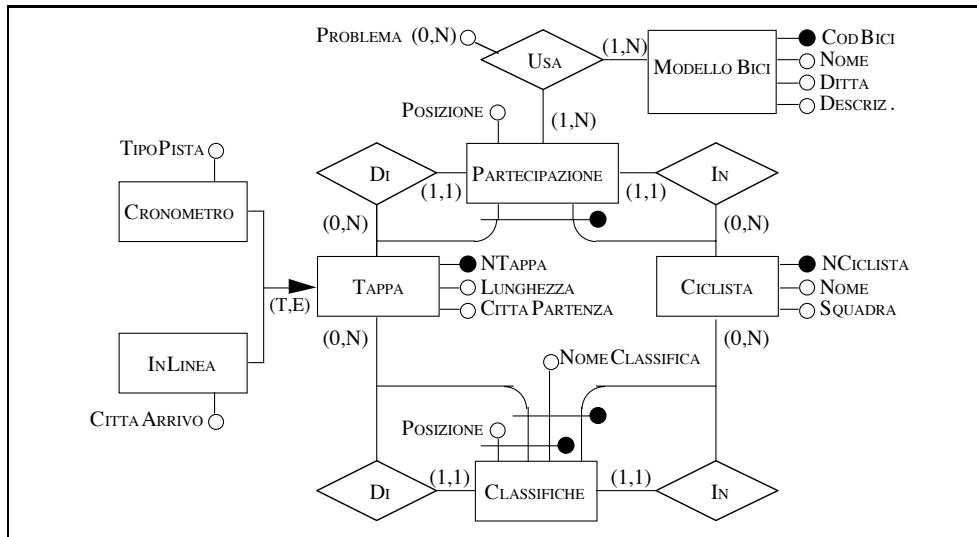
FK: CODACC REFERENCES ACCESSORIO

Esercizio 14



Schema Relazionale

ModelloAuto(NomeAuto)FaseMontaggio(NomeAuto, NomeFase, Tipo, CFCapoSquadra)**FK:** NomeAuto **REFERENCES** ModelloAuto**FK:** CFCapoSquadra **REFERENCES** SquadraDipendente(CF, Nome)Tecnico(CF) **FK:** CF **REFERENCES** DipendenteSquadra(CF) **FK:** CF **REFERENCES** TecnicoOperaio(CF, CFCapoSquadra)**FK:** CF **REFERENCES** Dipendente**FK:** CFCapoSquadra **REFERENCES** SquadraMacchinario(Cod, NomeMacchinario, Descrizione)Utilizza(CF, Cod, NomeFase, NomeAuto, NumOre)**FK:** CF **REFERENCES** Dipendente**FK:** Cod **REFERENCES** Macchinario**FK:** NomeFase, NomeAuto **REFERENCES** FaseMontaggio

Esercizio 16**Schema Relazionale**

TAPPA(NTAPPA,LUNGHEZZA,CITTAPARTENZA,CITTAARRIVO,TIPOPISTA)

CICLISTA(NCICLISTA,NOME,SQUADRA)

MODELLOBICI(CODBIC,NOME,DITTA,DESCRIZIONE)

PARTECIPAZIONE(NTAPPA,NCICLISTA,POSIZIONE)

FK: NTAPPA REFERENCES TAPPA

FK: NCICLISTA REFERENCES CICLISTA

USA(NTAPPA,NCICLISTA,CODBIC)

FK: NTAPPA,NCICLISTA REFERENCES PARTECIPAZIONE

FK: CODBIC REFERENCES MODELLOBICI

PROBLEMI(NTAPPA,NCICLISTA,CODBIC,PROBLEMA)

FK: NTAPPA,NCICLISTA,CODBIC REFERENCES USA

CLASSIFICA(NTAPPA,NOMECLASS,POSIZIONE,NCICLISTA)

AK: NTAPPA,NOMECLASS,NCICLISTA

FK: NTAPPA REFERENCES TAPPA

FK: NCICLISTA REFERENCES CICLISTA

- ◇ Si noti che nello schema relazionale è possibile aggiungere il vincolo “un ciclista può essere riportato nelle classifiche di una certa tappa solo se ha partecipato a quella tappa” mettendo, nello schema di relazione CLASSIFICA, al posto delle due foreign key, un’unica foreign key che si riferisce a PARTECIPAZIONE:

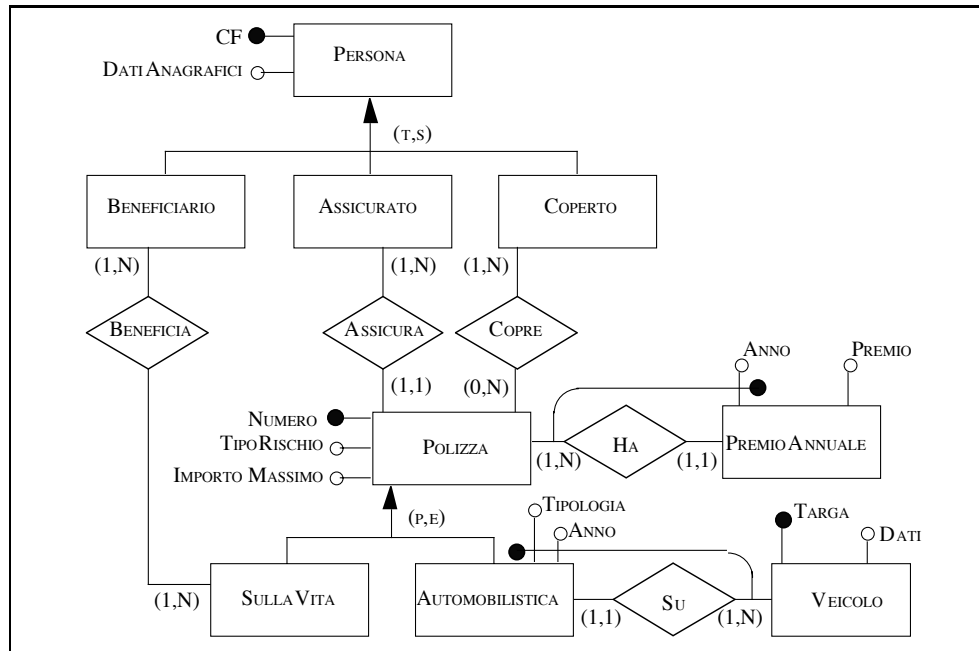
CLASSIFICA(NTAPPA,NOMECLASS,POSIZIONE,NCICLISTA)

AK: NTAPPA,NOMECLASS,NCICLISTA

FK: NTAPPA,NCICLISTA REFERENCES PARTECIPAZIONE

FK: CODSTA REFERENCES STAZIONE

Esercizio 18



Schema Relazionale

PERSONA(CF,DATI ANAGRAFICI, TIPO PERSONA)

POLIZZA(NUMPOL, TIPO RISCHIO, MAX IMPORTO, CF ASSICURATO)

FK: CF ASSICURATO **REFERENCES** PERSONA

PREMIO ANNUALE(NUMPOL, ANNO, PREMIO)

FK: NUMPOL **REFERENCES** POLIZZA

COPRE(NUMPOL, CF COPERTO)

FK: NUMPOL **REFERENCES** POLIZZA**FK:** CF COPERTO **REFERENCES** PERSONA

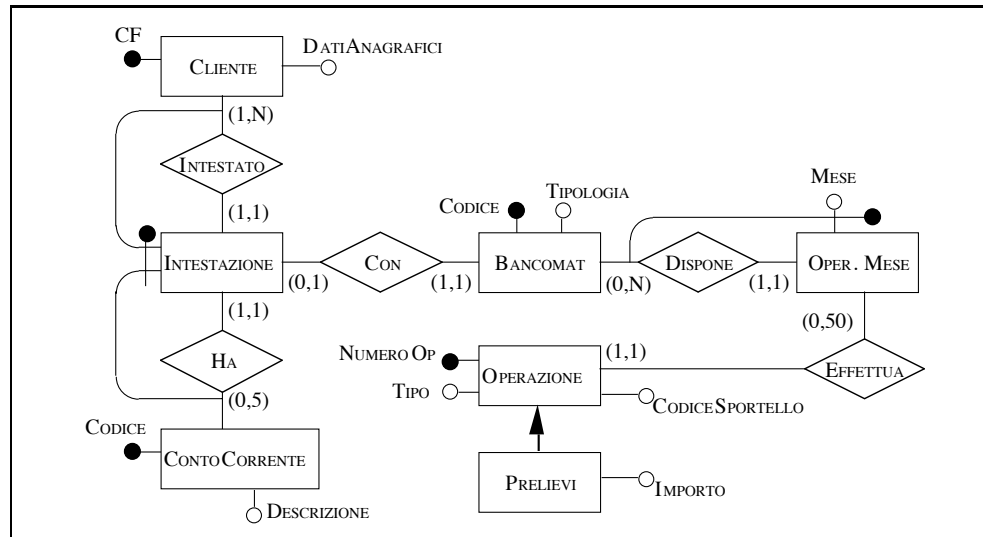
POLIZZA SULLA VITA(NUMPOL)

FK: NUMPOL **REFERENCES** POLIZZA

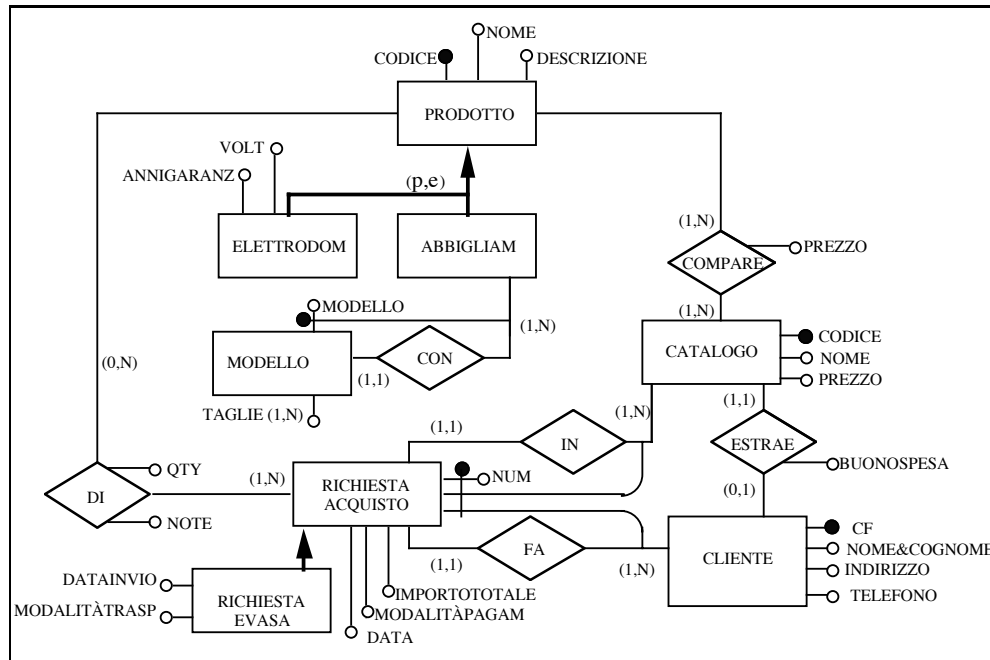
POLIZZA AUTOMOBILISTICA(NUMPOL, TIPOLOGIA, ANNO, TARGA)

AK: TIPOLOGIA, ANNO, TARGA**FK:** NUMPOL **REFERENCES** POLIZZA**FK:** TARGA **REFERENCES** VEICOLO

VEICOLO(TARGA, DATI)

Esercizio 19**Schema Relazionale**CLIENTE(CF,DATIANAGRAFICI)CONTOCORRENTE(CODCONT,DESCRIZIONE)**FK:** CFASSICURA **REFERENCES** PERSONAINTESAZIONE(CF,CODCONT)**FK:** CF **REFERENCES** CLIENTE**FK:** CODCONT **REFERENCES** CONTOCORRENTEBANCOMAT(CODBANCOMAT,CF,CODCONT,TIPOLOGIA)**AK:** CF,CODCONT**FK:** CF,CODCONT **REFERENCES** INTESAZIONEOPERAZIONIMENSILI(CODBANCOMAT,MESE)**FK:** CODBANCOMAT **REFERENCES** BANCOMATOPERAZIONE(NUMOPER,TIPO,CODICESPORTELLO,CODBANCOMAT,MESE)**FK:** CODBANCOMAT,MESE **REFERENCES** OPERAZIONIMENSILIPRELIEVO(NUMOPER,IMPORTO)**FK:** NUMOPER **REFERENCES** OPERAZIONE

Esercizio 20



Schema Relazionale

PRODOTTO(CODPROD,NOME,DESCRIZIONE)

ELETTRODOMESTICO(CODPROD,ANNIGARANZIA,VOL)

FK: CODPROD REFERENCES PRODOTTO

ABBIGLIAMENTO(CODPROD)

FK: CODPROD REFERENCES PRODOTTO

MODELLOTAGLIA(CODPROD,MODELLO,TAGLIA)

FK: CODPROD REFERENCES ABBIGLIAMENTO

CLIENTE(CF,NOME & COGNOME,INDIRIZZO,TELEFONO)

CATALOGO(CODCAT,NOME,PREZZO,CFESTRATTO,BUONOSPESA)

AK: CFESTRATTO

FK: CFESTRATTO REFERENCES CLIENTE

COMPARE(CODCAT,CODPROD,PREZZO)

FK: CODCAT REFERENCES CATALOGO

FK: CODPROD REFERENCES PRODOTTO

RICHIESTA(CF,CODCAT,NUM,IMPORTOTOTALE,MODALITÀPAG,DATA)

FK: CF REFERENCES CLIENTE

FK: CODCAT REFERENCES CATALOGO

RICHIESTAEVASA(CF,CODCAT,NUM,IMPORTOTOTALE,MODALITÀTRAS,DATAINVIO)

FK: CF, CODCAT, NUM REFERENCES RICHIESTA

DI(CF,CODCAT,NUM,CODPROD,QTY,NOTE)

FK: CF, CODCAT, NUM REFERENCES RICHIESTA

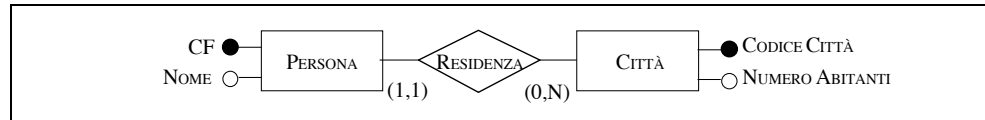
- ◇ Il vincolo “tutti i prodotti compresi in una richiesta devono comparire in uno stesso catalogo” viene espresso solo nello schema relazionale aggiungendo, nello schema di relazione DI, la foreign key:

FK: CODCAT,CODPROD REFERENCES COMPARE

6.1.2 Dati Derivati

◇ Esercizio 1

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo derivato NUMEROABITANTI calcolato contando il numero delle persone che risiedono in una certa città. Si trascuri l'occupazione di memoria del dato derivato.



Operazione 1) Dato il codice di una città, visualizzarne tutti i suoi dati.

Operazione 2) Dato il codice di una città e di una persona, memorizza la relativa residenza (incrementando anche l'eventuale dato derivato).

Tavola dei volumi

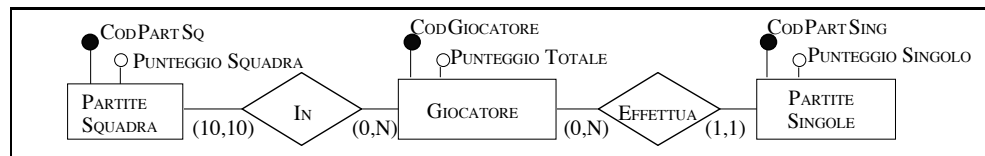
CONCETTO	TIPO	VOL.
Città	E	200
Persona	E	1000000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	2/Giorno
Oper . 2	I	500/Giorno

◇ Esercizio 2

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo derivato PUNTEGGIOTOTALE, che per una certa giocatore è calcolato sommando sia i punteggi delle partite singole che quelli delle partite in squadra. Si noti che il punteggio di una partita in squadra deve essere sommato a tutti i giocatori della squadra. Si trascuri l'occupazione di memoria del dato derivato.



Operazione 1) Inserimento di una nuova partita singola (si suppone noto e valido il codice del giocatore che effettua la partita);

Operazione 2) Inserimento di una nuova partita di squadra (si suppongono noti e validi i codici dei giocatori che effettuano la partita);

Operazione 3) Visualizzare tutti i dati di un giocatore, compreso il dato derivato.

Tavola dei volumi

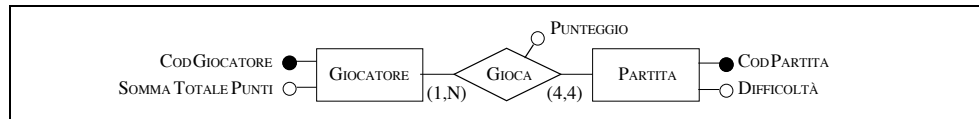
CONCETTO	TIPO	VOL.
Giocatore	E	100
PartiteSquadra	E	500
PartiteSingole	E	1000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	100/Giorno
Oper . 2	I	50/Giorno
Oper . 3	I	15/Giorno

◇ Esercizio 3

Dato il seguente schema E/R, con il seguente volume dei dati e le seguenti operazioni, decidere se è conveniente conservare nello schema l'attributo derivato **SOMMA-TOTALEPUNTI**, che per un certo **GIOCATORE** è calcolato come la somma di **PUNTEGGIO*DIFFICOLTÀ** di tutte le partite giocate. Si trascuri l'occupazione di memoria del dato derivato.



Operazione 1) Introduzione di una nuova partita;

Operazione 2) Visualizzare i dati di un giocatore.

Tavola dei volumi

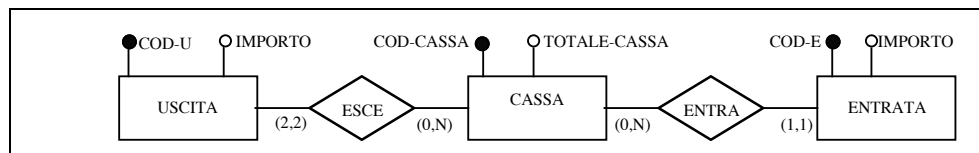
CONCETTO	TIPO	VOL.
Giocatore	E	100
Partita	E	1000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	40/Giorno
Oper . 2	I	10/Giorno

◇ Esercizio 4

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo derivato **TOTALECASSA**, che per una certa cassa è calcolato come la differenza tra la somma degli importi delle entrate e la somma degli importi delle uscite. Si supponga che l'importo di una entrata venga aggiunto al **TOTALECASSA** di un'unica cassa mentre l'importo di una uscita venga sottratto al **TOTALECASSA** di due casse definite. Si trascuri l'occupazione di memoria del dato derivato.



Operazione 1) Inserimento di una nuova entrata (si suppone noto e valido il codice della cassa sulla quale l'entrata deve essere registrata);

Operazione 2) Visualizzare tutti i dati di un cassa. Si noti che nel caso in cui si deve determinare il **TOTALECASSA** occorre leggere tutte le entrate e tutte le uscite di quella cassa;

Operazione 3) Inserimento di una nuova uscita (si suppongono noti e validi i codici delle due casse sulle quali l'uscita deve essere registrata).

Tavola dei volumi

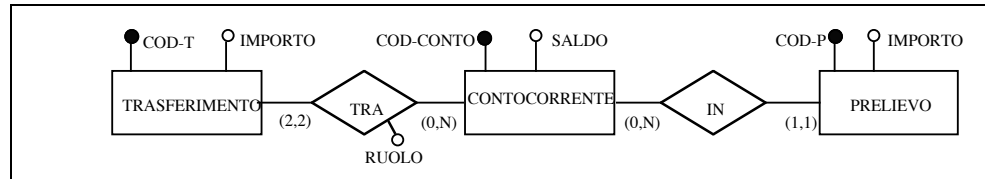
CONCETTO	TIPO	VOL.
Cassa	E	100
Entrata	E	3000
Uscita	E	1000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	100/Giorno
Oper . 2	I	10/Giorno
Oper . 3	I	1/Giorno

◇ Esercizio 5

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo derivato SALDO, che per una certo contocorrente è calcolato tenendo in considerazione sia gli importi dei prelievi sia gli importi dei trasferimenti. Si supponga che l'importo di un prelievo venga sottratto al SALDO di un unico contocorrente mentre l'importo di un trasferimento venga sottratto dal SALDO di un contocorrente (RUOLO="Origine") ed aggiunto al SALDO di un altro contocorrente (RUOLO="Destinazione"). Si trascuri l'occupazione di memoria del dato derivato.



Operazione 1) Inserimento di un nuovo prelievo (si suppone noto e valido il codice del contocorrente sul quale viene effettuato il prelievo);

Operazione 2) Inserimento di un nuovo trasferimento (si suppongono noti e validi i codici dei due contocorrenti che interessano il trasferimento);

Operazione 3) Visualizzare tutti i dati di un contocorrente, anche il SALDO.

Tavola dei volumi

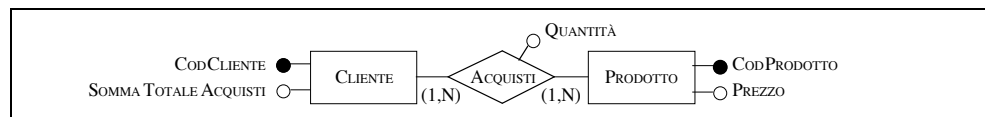
CONCETTO	TIPO	VOL.
ContoCorrente	E	100
Prelievo	E	3000
Trasferimento	E	1000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	100/Giorno
Oper . 2	I	10/Giorno
Oper . 3	I	4/Giorno

◇ Esercizio 6

Dato il seguente schema E/R, con il seguente volume dei dati e le seguenti operazioni, decidere se è conveniente conservare nello schema l'attributo SOMMATOTALEACQUISTI, trascurando l'occupazione di memoria di tale dato.



Operazione 1) Dati i codici di un cliente e di un prodotto già esistenti, inserire l'acquisto del prodotto da parte del cliente;

Operazione 2) Visualizzare i dati di un cliente;

Operazione 3) Modificare il prezzo di un prodotto.

Tavola dei volumi

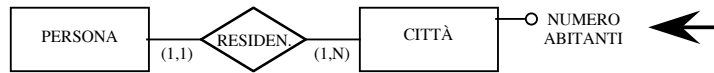
CONCETTO	TIPO	VOL.
Prodotto	E	600
Cliente	E	300
Acquisti	R	12000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper . 1	I	400/Giorno
Oper . 2	I	10/Giorno
Oper . 3	I	1/Giorno

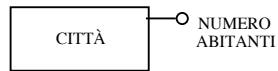
6.1.3 Soluzioni

Esercizio 1

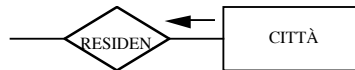


operazione 1:

Con il dato
derivato

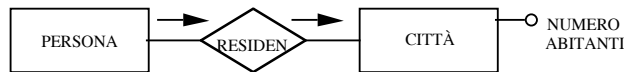


Senza il dato
derivato

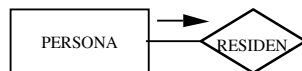


operazione 2:

Con il dato
derivato



Senza il dato
derivato



Con il dato derivato:

Operazione 1

1 accesso in lettura
 $1 \times 2 = 2/\text{giorno}$

CONCETTO	ACC.	TIPO
Città	1	L
Persona	1	S
Residenza	1	S
Città	1	L
Città	1	S

Operazione 2

1 accesso in lettura
3 accessi in scrittura
 $7 \times 500 = 3500/\text{giorno}$

Senza il dato derivato:

Operazione 1

5001 accessi in lettura
 $5001 \times 2 = 10002/\text{giorno}$

CONCETTO	ACC.	TIPO
Città	1	L
Residenza	5000	L
Persona	1	S
Residenza	1	S

Operazione 2

2 accessi in scrittura
 $4 \times 500 = 2000/\text{giorno}$

Conclusione: Conviene tenere il dato derivato.

Esercizio 2**Con il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	PartitaSingola	1	S
1 accesso in lettura	In	1	S
3 accessi in scrittura	Giocatore	1	L
$(1+3*2)*100 = 700/\text{giorno}$	Giocatore	1	S
Operazione 2	PartitaSquadra	1	S
10 accessi in lettura	Effettua	10	S
21 accessi in scrittura	Giocatore	10	L
$(10+21*2)*50 = 2600/\text{giorno}$	Giocatore	10	S
Operazione 3	Giocatore	1	L
1 accesso in lettura			
$1*15 = 15/\text{giorno}$			

Totale : 3315/giorno**Senza il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	PartitaSingola	1	S
2 accessi in scrittura	In	1	S
$(2*2)*100 = 400/\text{giorno}$			
Operazione 2	PartitaSquadra	1	S
3 accessi in scrittura	Effettua	10	S
$(3*2)*50 = 300/\text{giorno}$			
Operazione 3	Giocatore	1	L
121 accessi in lettura	In	50	L
$121*15 = 1815/\text{giorno}$	PartitaSquadra	50	L
	Effettua	10	L
	PartitaSingola	10	L

Totale : 3315/giorno

Conclusione : Dal punto di vista del numero di accessi, è indifferente tenere o meno il dato derivato.

Esercizio 3**Con il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Partita	1	S
4 accessi in lettura	Gioca	4	S
9 accessi in scrittura	Giocatore	4	L
$22*40 = 880$ /giorno	Giocatore	4	S
Operazione 2	Giocatore	1	L
1 accesso in lettura			
$1*100 = 10$ /giorno			
Totale : 890/giorno			

Senza il dato derivato :

	CONCETTO	ACC.	TIPO
Operazione 1	Partita	1	S
5 accessi in scrittura	Gioca	4	S
$10*40 = 400$ /giorno			
Operazione 2	Giocatore	1	L
81 accessi in lettura	Gioca	40	L
$81*100 = 810$ /giorno	Partita	40	L
Totale : 1210/giorno			

Conclusione : Conviene tenere il dato derivato.

Esercizio 4**Con il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Entrata	1	S
1 accesso in lettura	Entra	1	S
3 accessi in scrittura	Cassa	1	L
$7 \cdot 100 = 700/\text{giorno}$	Cassa	1	S
Operazione 2	Cassa	1	L
1 accesso in lettura			
$1 \cdot 10 = 10/\text{giorno}$			
Operazione 3	Uscita	1	S
2 accesso in lettura	Esce	2	S
5 accessi in scrittura	Cassa	2	L
$12 \cdot 1 = 12/\text{giorno}$	Cassa	2	S

Totale : 722/giorno**Senza il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Entrata	1	S
2 accessi in scrittura	Entra	1	S
$4 \cdot 100 = 400/\text{giorno}$			
Operazione 2	Cassa	1	L
121 accesso in lettura	Entra	40	L
$121 \cdot 10 = 1210/\text{giorno}$	Entrata	40	L
	Esce	20	L
	Uscita	20	L
Operazione 3	Uscita	1	S
3 accessi in scrittura	Esce	2	S
$3 \cdot 1 = 3/\text{giorno}$			

Totale : 1613/giorno**Conclusione :** Conviene tenere il dato derivato.

Esercizio 5**Con il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Cliente	1	L
2 accessi in lettura	Acquisti	1	S
2 accessi in scrittura	Prodotto	1	L
6*100 = 600 /giorno	Cliente	1	S
Operazione 2	Cliente	1	L
1 accesso in lettura			
1*10 = 10 /giorno			
Operazione 3	Prodotto	1	L
41 accessi in lettura	Prodotto	1	S
21 accessi in scrittura	Acquisti	20	L
83*1 = 83 /giorno	Cliente	20	L
	Cliente	20	S

Totale : 693/giorno**Senza il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Acquisti	1	S
1 accessi in scrittura			
2*100 = 200 /giorno			
Operazione 2	Cliente	1	L
81 accessi in lettura	Acquisti	40	L
81*10 = 1210 /giorno	Prodotto	40	L
Operazione 3	Prodotto	1	L
1 accesso in lettura	Prodotto	1	S
1 accesso in scrittura			
3*1 = 3 /giorno			

Totale : 1413/giorno**Conclusioni :** Conviene tenere il dato derivato.

Esercizio 6**Con il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Cliente	1	L
2 accessi in lettura	Acquisti	1	S
2 accessi in scrittura	Prodotto	1	L
$6 \cdot 400 = 2400$ /giorno	Cliente	1	S
Operazione 2	Cliente	1	L
1 accesso in lettura			
$1 \cdot 10 = 10$ /giorno			
Operazione 3	Prodotto	1	L
41 accessi in lettura	Prodotto	1	S
21 accessi in scrittura	Acquisti	20	L
$83 \cdot 1 = 83$ /giorno	Cliente	20	L
	Cliente	20	S

Totale : 2493/giorno**Senza il dato derivato :**

	CONCETTO	ACC.	TIPO
Operazione 1	Acquisti	1	S
1 accessi in scrittura			
$2 \cdot 400 = 800$ /giorno			
Operazione 2	Cliente	1	L
81 accessi in lettura	Acquisti	40	L
$81 \cdot 10 = 1210$ /giorno	Prodotto	40	L
Operazione 3	Prodotto	1	L
1 accesso in lettura	Prodotto	1	S
1 accesso in scrittura			
$3 \cdot 1 = 3$ /giorno			

Totale : 2093/giorno**Conclusione :** Conviene eliminare il dato derivato.

6.2 SQL e Algebra Relazionale

Per ciascun esercizio, viene richiesto di scrivere in SQL tutte le interrogazioni riportate; le interrogazioni da scrivere anche in Algebra Relazionale sono esplicitamente indicate.

◇ Esercizio 1

Schema Relazionale

DOCENTE(CFDOC,NOME,COGNOME)

STUDENTE(CFSTUD,NOME,COGNOME)

ARGOMENTO(CODARG,DESCRIZIONE)

LEZIONE(CODARG,DATA,CFDOC,NUMSTUDENTI)

AK: DATA,CFDOC

FK: CODARG **REFERENCES** ARGOMENTO

FK: CFDOC **REFERENCES** DOCENTE

dove NUMSTUDENTI è il numero di studenti presenti alla lezione

INTERROGAZIONE(CODARG,DATA,CFSTUD,VOTO)

FK: CODARG,DATA **REFERENCES** LEZIONE

FK: CFSTUD **REFERENCES** STUDENTE

Interrogazione e voto relativo dello studente CFSTUD durante una lezione.

Interrogazioni (Algebra Relazionale **a**), **b**) e **c**).

- a)** selezionare il codice fiscale, il nome ed il cognome degli studenti che non sono mai stati interrogati su un argomento con descrizione 'Fisica';
- b)** selezionare il codice fiscale del docente che ha svolto lezioni su tutti gli argomenti con descrizione 'Fisica';
- c)** selezionare il codice fiscale del docente che ha sempre interrogato, cioè che durante ogni sua lezione ha fatto almeno una interrogazione;
- d)** selezionare, per ogni argomento, la media dei voti riportati dagli studenti interrogati sull'argomento, considerando solo gli studenti che sono stati interrogati almeno tre volte sull'argomento in questione;
- e)** selezionare, per ogni studente, il codice fiscale del docente con il quale ha effettuato il maggior numero di interrogazioni.

◇ **Esercizio 2****Schema Relazionale**CICLISTA(NOMECICLISTA, NAZIONALITÀ, ETÀ)GARA(NOMECORSA, ANNO, PARTENZA, ARRIVO)

edizione di un certo ANNO della corsa ciclistica NOME CORSA; PARTENZA e ARRIVO sono la città di partenza e di arrivo.

PARTECIPA(NOMECORSA, ANNO, NOMECICLISTA, POSIZIONE)**FK:** NOME CORSA, ANNO **REFERENCES** GARA**FK:** NOME CICLISTA **REFERENCES** CICLISTA

NOME CICLISTA ha partecipato alla gara (NOME CORSA, ANNO) classificandosi in una certa POSIZIONE o ritirandosi (POSIZIONE='R').

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a)** selezionare i ciclisti che si sono classificati in prima posizione in una gara ciclistica partita da Milano;
- b)** selezionare il nome dei ciclisti che non si sono mai ritirati al Giro (corsa con nome Giro);
- c)** selezionare le corse per le quali in ogni edizione c'è stato almeno un ritirato;
- d)** selezionare, per ogni corsa ciclistica, l'anno in cui c'è stato il maggior numero di ciclisti ritirati.

◇ **Esercizio 3****Schema Relazionale**QUADRO(CQ, AUTORE, PERIODO)MOSTRA(CM, NOME, ANNO, ORGANIZZATORE)ESPONE(CM, CQ, SALA) **FK:** CM **REFERENCES** MOSTRA**FK:** CQ **REFERENCES** QUADRO

Nella mostra CM, il quadro CQ è stato esposto in una certa SALA.

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a)** selezionare le sale nelle quali è stato esposto, nell'anno 1997, un quadro di Picasso;
- b)** selezionare tutti i dati dei quadri di Picasso che non sono mai stati esposti nell'anno 1997;
- c)** selezionare tutti i dati dei quadri che non sono mai stati esposti insieme ad un quadro di Picasso, cioè nella stessa mostra in cui compariva anche un quadro di Picasso;
- d)** selezionare tutti i dati delle mostre in cui sono stati esposti quadri di almeno 5 autori distinti;
- e)** selezionare, per ogni mostra, l'autore di cui si esponevano il maggior numero di quadri.

◇ **Esercizio 4****Schema Relazionale**PRODOTTO(CP, DESCRIZIONE)SOCIO(CS, NOME, COGNOME)OFFERTA(CO, VALIDITÀ)COMPRENDE(CO, CP, QUANTITÀ)**FK: CO REFERENCES OFFERTA****FK: CP REFERENCES PRODOTTO**

QUANTITÀ è il numero di unità del prodotto CP comprese nell'offerta CO

RITIRA(CO, CS, DATA)**FK: CO REFERENCES OFFERTA****FK: CS REFERENCES SOCIO**

DATA è il giorno in cui il socio CS ritira l'offerta CO

Interrogazioni (Algebra Relazionale **a**), **b**) e **c**)).

- a) selezionare il codice e la descrizione dei prodotti che non sono mai stati offerti insieme ad un prodotto con descrizione = 'Uva';
- b) selezionare il codice, nome e cognome dei soci che non hanno ritirato alcuna offerta che comprende un prodotto con descrizione 'Uva'.
- c) selezionare il codice, nome e cognome dei soci che hanno ritirato tutte le offerte che comprendono un prodotto con descrizione = 'Uva'.
- d) selezionare, per ogni socio, il numero delle offerte ritirate che comprendono un prodotto con descrizione = 'Uva'.

◇ **Esercizio 5****Schema Relazionale**VIA(CV, NOME, QUARTIERE, LUNGHEZZA)INC(CVA, CVB, N-VOLTE)**FK: CVA REFERENCES VIA****FK: CVB REFERENCES VIA**

La via CVA incrocia N-VOLTE la via CVB

si assume che se nella relazione INC è presente la tupla (codviax, codviay, 5) non sia presente la tupla simmetrica (codviay, codviax, 5);

Interrogazioni (Algebra Relazionale **a**) e **b**)).

- a) selezionare le vie che incrociano almeno una via del quartiere 'Pastena';
- b) selezionare le vie che non incrociano via 'Marco Polo';
- c) selezionare le coppie (CODICE1, CODICE2) tali che le vie con codice CODICE1 e CODICE2 abbiano la stessa lunghezza;
- d) selezionare il quartiere che ha il maggior numero di vie;
- e) selezionare, per ogni quartiere, la via di lunghezza maggiore;
- f) selezionare le vie che incrociano tutte le vie del quartiere 'Pastena'.

◇ **Esercizio 6****Schema Relazionale**CAMPO(NCAMPO, TIPO, INDIRIZZO)TENNISTA(CE, NOME, NAZIONE)INCONTRO(CI, NCAMPO, GIOC1, GIOC2, SET1, SET2)**FK:** NCAMPO REFERENCES CAMPO**FK:** GIOC1 REFERENCES TENNISTA**FK:** GIOC2 REFERENCES TENNISTA

Incontro, svolto nel campo NCAMPO, tra i tennisti GIOC1 e GIOC2: in SET1 e SET2 sono riportati il numero di set vinti da GIOC1 e GIOC2 rispettivamente.

Interrogazioni (Algebra Relazionale **a**), **b**), **c**) e **d**)).

- a)** selezionare gli incontri disputati sull'erba (campo con tipo 'erba');
- b)** selezionare i campi in erba sui quali non c'è stato nessun incontro;
- c)** selezionare i dati dei tennisti vincitori di almeno una partita sull'erba;
- d)** selezionare i dati delle nazioni in cui tutti i giocatori hanno sempre vinto le partite disputate;
- e)** selezionare il campo in erba che ha ospitato il maggior numero di incontri.

◇ **Esercizio 7****Schema Relazionale**OPERATORE(CODOP, INDIRIZZO, QUALIFICA, COSTO-ORARIO)ARTICOLO(CODART, DESCRIZIONE)LOTTO(CODART, COPOP, TOTESEM)**FK:** CODART REFERENCES ARTICOLO**FK:** CODOP REFERENCES OPERATORE

dove TOTESEM è il numero di pezzi dell'articolo CODART.

RECLAMO(CODART, COPOP, NESEMPLARE, NOMECL)**FK:** CODART, CODOP REFERENCES LOTTO

Reclamo effettuato dal cliente NOMECL sull'esemplare NESEMPLARE del lotto confezionato dall'operatore CODOP relativo all'articolo CODART.

Interrogazioni (Algebra Relazionale **a**), **b**) e **c**)).

- a)** selezionare il codice e il nome degli operatori per i quali non esiste alcun reclamo, cioè per i quali nessun esemplare di nessun lotto da essi confezionato ha ricevuto un reclamo;
- b)** selezionare il codice degli operatori per i quali ogni lotto da essi confezionato contiene almeno un esemplare al quale si riferisce un reclamo;
- c)** selezionare il nome del cliente che ha fatto reclami per tutti gli operatori;
- d)** selezionare, per ogni articolo, il codice dell'operatore che ha confezionato il lotto con il maggior numero di esemplari, senza considerare i lotti con un numero di esemplari TOTESEM non specificato.
- e)** selezionare il lotto che ha ricevuto più reclami.

◇ **Esercizio 8****Schema Relazionale**MANIFESTAZIONE(CM,NOME)LUOGO(NOME-LUOGO,INDIRIZZO,CITTÀ)SPETTACOLO(CM,NUM,ORA-INIZIO,NOME-LUOGO,DATA)**FK:** CM REFERENCES MANIFESTAZIONE**FK:** NOME-LUOGO REFERENCES LUOGO

NUM è il numero dello spettacolo all'interno della manifestazione CM

Interrogazioni (Algebra Relazionale **a)** e **b)**).

- a) selezionare il codice e il nome delle manifestazioni che non hanno interessato luoghi della città di Modena;
- b) selezionare i nomi dei luoghi che hanno ospitato tutte le manifestazioni (hanno ospitato almeno uno spettacolo di ciascuna manifestazione);
- c) selezionare il nome dei luoghi che, in una certa data, ospitano più di tre spettacoli dopo le ore 15;
- d) selezionare, per ogni luogo, il numero totale delle manifestazioni e il numero totale degli spettacoli ospitati;
- e) Descrivere sinteticamente a parole e riportare in SQL l'interrogazione descritta dalla seguente espressione dell'algebra relazionale:

$$\pi_{CM}(SPETTACOLO) - \pi_{CM}(\pi_{CM,NUM}(SPETTACOLO) - \pi_{CM,NUM}(\sigma_{ORA-INIZIO > 15}(SPETTACOLO)))$$

◇ **Esercizio 9****Schema Relazionale**FORNITORE(CODE,NOMEFORNITORE,CITTÀ)PRODOTTO(CODP,DESCRIZIONE,PREZZO)FORNISCE(ANNO,CODP,CODF,QTY)**FK:** CODP REFERENCES PRODOTTO**FK:** CODF REFERENCES FORNITORE

Nell'ANNO specificato, il prodotto CODP è stato fornito dal fornitore CODF in quantità QTY

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a) selezionare i prodotti che nell'anno 1995 sono stati forniti da almeno un fornitore di Modena;
- b) selezionare i prodotti non forniti da nessun fornitore di Modena;
- c) selezionare i prodotti che nell'anno 1994 sono stati forniti esclusivamente da fornitori di Modena;
- d) selezionare, per ogni anno, la quantità totale dei prodotti forniti dai fornitori di Modena;
- e) selezionare, per ogni anno, il codice del fornitore che ha fornito in totale la maggiore quantità di prodotti.

◇ **Esercizio 10****Schema Relazionale**GARA(CG,NOME,CAMPO,LIVELLO)GIOCATOREGOLF(CF,NOME,NAZIONE)PARTECIPA(CG,CF,PUNTEGGIO)**FK:** CG REFERENCES GARA**FK:** CF REFERENCES GIOCATOREGOLF

Il giocatore CF partecipa alla gara CG totalizzando un certo PUNTEGGIO. La gara è vinta dal giocatore che totalizza il punteggio più basso.

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a)** selezionare i dati dei giocatori di golf che hanno partecipato ad almeno una gara disputata a livello 'nazionale';
- b)** selezionare le nazioni in cui tutti i giocatori hanno ottenuto un punteggio minore o uguale a 0 nelle gare disputate;
- c)** selezionare i dati dei giocatori di golf che hanno partecipato a tutte le gare disputate a livello 'nazionale';
- d)** selezionare i dati dei giocatori di golf che hanno vinto almeno una gara disputata a livello 'internazionale';
- e)** selezionare, per ogni nazione che nelle gare di livello 'internazionale' ha schierato più di 5 giocatori distinti, il punteggio medio ottenuto dai giocatori in tali gare; si ordini il risultato in modo decrescente rispetto al punteggio medio.

◇ **Esercizio 11****Schema Relazionale**

CD(CODCD,AUTORE,CASADISCO)

CLIENTE(NTESS,NOME,INDIRIZZO)ACQUISTO(CODCD,NTESS,DATA,QTY)**FK:** CODCD REFERENCES CD**FK:** NTESS REFERENCES CLIENTE

Il cliente identificato da NTESS ha acquistato, in una certa DATA, un certo numero QTY di copie del compact disk CODCD

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a)** selezionare tutti i dati dei clienti che dopo il 1/1/1997 non hanno acquistato nessun CD prodotto dalla casa discografica 'DiscoJolli';
- b)** selezionare il numero tessera dei clienti che hanno acquistato tutti i CD dell'autore 'Francesco Guccini';
- c)** selezionare, per ogni CD, il numero totale delle copie vendute;
- d)** selezionare, per ogni casa discografica, il numero tessera del cliente che ha acquistato il maggior numero di copie di CD di quella casa.

◇ Esercizio 12

Schema Relazionale

DIPENDENTE(CF,NOME,CITTÀ)

PROGETTO(CP,NOME,ANNO,DURATA)

LAVORA(CP,CF,MESI,RUOLO) **FK: CP REFERENCES PROGETTO**
FK: CF REFERENCES DIPENDENTE

Nel progetto CP, il dipendente CF lavora per un certo numero di MESI, svolgendo un certo RUOLO.

Interrogazioni (Algebra Relazionale a) e b)).

- a) selezionare i dati dei dipendenti di Modena che non hanno lavorato in alcun progetto dell'anno 1995;
- b) selezionare i dati dei dipendenti che non hanno mai lavorato insieme ad un dipendente di Modena, cioè nello stesso progetto in cui lavorava anche un dipendente di Modena;
- c) selezionare le coppie (CF1, CF2) tali che i dipendenti con codice fiscale CF1 e CF2 hanno lavorato nello stesso progetto;
- d) selezionare i dipendenti che hanno lavorato in almeno 3 ruoli distinti;
- e) selezionare, per ogni dipendente, il progetto in cui esso ha lavorato il maggior numero di mesi.

◇ Esercizio 13

Schema Relazionale

TECNICO(CF,INDIRIZZO,QUALIFICA,COSTO-ORARIO)

PC(CP,NOME,TIPO,NOMEPROPRIETARIO)

[illegible]

Nella DATA specificata, il tecnico CF ha riparato il personal computer CP impiegando un certo numero di ORE

Interrogazioni (Algebra Relazionale **a**) e **b**).

- a) selezionare i personal di tipo 'Mac' che non sono stati riparati tra il 1/7/97 e il 1/11/97;
- b) selezionare i tecnici che hanno riparato tutti i personal di tipo 'Mac';
- c) selezionare le coppie (CP1, CP2) tali che i personal con codice CP1 e CP2 sono stati riparati nella stessa data dallo stesso tecnico;
- d) selezionare i dati dei personal che hanno richiesto almeno 10 ore di riparazione;
- e) selezionare, per ogni data, il tecnico che ha riparato il maggior numero di personal.
- f) selezionare il personal che ha totalizzato il maggior costo di riparazione, considerando le ore di riparazione e il relativo costo orario.

◇ **Esercizio 14****Schema Relazionale**NAZIONE(CN, PRESIDENTE, CONTINENTE)CONFERENZA(CC, DESCRIZIONE, CNSEDE)**FK:** CNSEDE REFERENCES NAZIONE

CNSEDE rappresenta la nazione in cui si è tenuta la conferenza CC

PARTECIPA(CC, CN, NUMEROP) **FK:** CC REFERENCES CONFERENZA**FK:** CN REFERENCES NAZIONE

NUMEROP è il numero di rappresentanti della nazione CN partecipanti alla conferenza CC

Interrogazioni (Algebra Relazionale **a)**, **b)** e **c)**).

- a)** selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi in una nazione del continente Europa;
- b)** selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi nella nazione stessa;
- c)** selezionare i dati relativi alle nazioni che non hanno mai partecipato ad una conferenza assieme ad una nazione del continente Europa;
- d)** selezionare i dati relativi alla nazione in cui si è tenuta la conferenza con il maggior numero di rappresentanti complessivo;
- e)** selezionare, per ogni nazione, la conferenza in cui vi ha partecipato con il maggior numero di rappresentanti.

◇ **Esercizio 15****Schema Relazionale**QUADRO(CQ, AUTORE, PERIODO)MOSTRA(CM, NOME, ANNO, ORGANIZZATORE)ESPONE(CM, CQ, SALA) **FK:** CM REFERENCES MOSTRA**FK:** CQ REFERENCES QUADRO

Nella mostra CM, il quadro CQ è stato esposto in una certa SALA

Interrogazioni (Algebra Relazionale **a)** e **b)**).

- a)** selezionare tutti i dati sulle mostre nelle quali è stato esposto un quadro di Picasso nel 97 oppure nel 96
- b)** selezionare il nome della mostra nella quale sono stati esposti tutti i quadri di Picasso
- c)** selezionare le quaterne (ANNO, NOMEMOSTRA1, NOMEMOSTRA2, ORGANIZZATORE) tali che nello stesso ANNO le mostre con nome NOMEMOSTRA1 e NOMEMOSTRA2 hanno avuto lo stesso ORGANIZZATORE;
- d)** selezionare tutti i dati dei quadri che sono stati esposti più di tre volte insieme ad un quadro di Picasso, cioè che sono stati esposti in più di tre mostre nelle quali compariva anche un quadro di Picasso;

6.2.1 Soluzioni

Di alcuni esercizi viene omessa la soluzione in SQL dell'interrogazione 1), in quanto si tratta in genere di un semplice join.

Inoltre, per semplicità, dati due schemi di relazioni

$$R_1(\underline{A}, \dots)$$

$$R_2(\dots, B, \dots)$$

FK: B REFERENCES R_1

il theta-join che collega R_1 e R_2 in base alla FK, $R_1 \bowtie_{R_1.A=R_2.B} R_2$, verrà indicato con il simbolo del join naturale: $R_1 \bowtie R_2$

Esercizio 1

- 1a)**

```
SELECT *
FROM STUDENTE
WHERE NOT EXISTS
    ( SELECT *
      FROM INTERROGAZIONE, ARGOMENTO
      WHERE STUDENTE.CFSTUD = INTERROGAZIONE.CFSTUD
        AND ARGOMENTO.CODARG = INTERROGAZIONE.CODARG
        AND ARGOMENTO.DESCRIZIONE = 'Fisica')
```
- 2a)** $\text{STUDENTE} \bowtie (\pi_{\text{CFSTUD}}(\text{STUDENTE}) - \pi_{\text{CFSTUD}}(\text{INTERROGAZIONE} \bowtie \sigma_{\text{DESCRIZIONE}='Fisica'}(\text{ARGOMENTO})))$
- 1b)**

```
SELECT distinct CFDOC
FROM LEZIONE X
WHERE NOT EXISTS
    ( SELECT *
      FROM ARGOMENTO
      WHERE DESCRIZIONE = 'Fisica'
        AND NOT EXISTS
            ( SELECT *
              FROM LEZIONE Y
              WHERE X.CFDOC = Y.CFDOC
                AND ARGOMENTO.CODARG = Y.CODARG))
```
- 2b)** $\pi_{\text{CFDOC}, \text{CODARG}}(\text{LEZIONE}) \div \pi_{\text{CODARG}}(\sigma_{\text{DESCRIZIONE}='Fisica'}(\text{ARGOMENTO}))$
- 1c)** L'interrogazione può essere riformulata come “i docenti per i quali *non esiste* una loro lezione *senza alcuna* interrogazione”;

Esercizio 2

1a) $\text{CICLISTA} \bowtie (\sigma_{\text{PARTENZA}='Milano'}(\text{GARA}) \bowtie \sigma_{\text{POSIZIONE}='Primo'}(\text{PARTECIPA}))$

1b)

```
SELECT *
FROM   CICLISTA
WHERE  NOMEICICLISTA NOT IN ( SELECT NOMEICICLISTA
                              FROM   PARTECIPA
                              WHERE  NOMEICORSA='Giro'
                              AND    POSIZIONE='R')
```

2b) $\text{CICLISTA} - \text{CICLISTA} \bowtie (\sigma_{\text{NOMEICORSA}='Giro' \text{ and } \text{POSIZIONE}='R'}(\text{PARTECIPA}))$

1c)

```
SELECT NOMEICORSA
FROM   GARA GX
WHERE  NOT EXISTS
      ( SELECT *
        FROM   GARA GY
        WHERE  GX.NOMEICORSA = GY.NOMEICORSA
        AND    NOT EXISTS
              ( SELECT *
                FROM   PARTECIPA P
                WHERE  GY.NOMEICORSA=P.NOMEICORSA
                AND    GY.ANNO=P.ANNO
                AND    P.POSIZIONE='R'))
```

2c) Indichiamo con R_1 le gare ciclistiche (limitandoci alla chiave) per le quali non c'è stato nessun ciclista ritirato

$$R_1 = \pi_{\text{NOMEICORSA}, \text{ANNO}}(\text{GARA}) - \pi_{\text{NOMEICORSA}, \text{ANNO}}(\sigma_{\text{POSIZIONE}='R'}(\text{PARTECIPA}))$$

Proiettando R_1 su NOMEICORSA si ottengono i nomi delle corse per le quali c'è stata almeno una edizione *senza* ritirati. Sottraendo tale insieme da tutti i nomi delle corse, specificati in GARA, si ottengono i nomi delle corse per le quali, in ogni edizione, c'è stato almeno un ritirato:

$$\pi_{\text{NOMEICORSA}}(\text{GARA}) - \pi_{\text{NOMEICORSA}}(R_1)$$

1d)

```
SELECT  NOMEICORSA, ANNO
FROM    PARTECIPA PX
WHERE   POSIZIONE='R'
GROUP BY NOMEICORSA, ANNO
HAVING  COUNT(*) >= ALL
      ( SELECT COUNT(*)
        FROM    PARTECIPA PY
        WHERE   POSIZIONE='R'
        AND     PY.NOMEICORSA=PX.NOMEICORSA
        GROUP BY ANNO)
```

Esercizio 3

2a) $\pi_{\text{SALA}}(\sigma_{\text{ANNO}='1997'}(\text{MOSTRA}) \bowtie \text{ESPONE} \bowtie \sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO}))$

1b) SELECT *
 FROM QUADRO
 WHERE AUTORE='Picasso'
 AND CQ NOT IN (SELECT CQ
 FROM MOSTRA, ESPONE
 WHERE MOSTRA.CM=ESPONE.CM
 AND ANNO='1997'

2b) $\sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO}) - \text{QUADRO} \bowtie (\text{ESPONE} \bowtie \sigma_{\text{ANNO}='1997'}(\text{MOSTRA}))$

1c) SELECT *
 FROM QUADRO
 WHERE CQ NOT IN (SELECT E2.CQ
 FROM QUADRO, ESPONE E1, ESPONE E2
 WHERE QUADRO.CQ=E1.CQ
 AND E1.CM=E2.CM
 AND AUTORE='Picasso')

2c) Se indichiamo con R_1 i codici delle mostre nelle quali è stato esposto almeno un quadro di Picasso

$$R_1 = \pi_{\text{CM}}(\text{ESPONE} \bowtie \sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO}))$$

i codici dei quadri esposti in tali mostre si ottengono dal join di ESPONE con R_1 . Sottraendo tali quadri dalla relazione QUADRO si ottengono quelli richiesti dall'interrogazione:

$$\text{QUADRO} - \text{QUADRO} \bowtie (\text{ESPONE} \bowtie \pi_{\text{CM}}(\text{ESPONE} \bowtie \sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO})))$$

d) SELECT *
 FROM MOSTRA
 WHERE 5 >= (SELECT COUNT(DISTINCT AUTORE)
 FROM QUADRO, ESPONE
 WHERE QUADRO.CQ=ESPONE.CQ
 AND MOSTRA.CM=ESPONE.CM)

```
e) SELECT    CM,AUTORE
FROM        QUADRO Q1,ESPONE E1
WHERE       Q1.CQ=E1.CQ
GROUP BY    CM,AUTORE
HAVING      COUNT(*) >= ALL ( SELECT    COUNT(*)
                                FROM      QUADRO Q2,ESPONE E2
                                WHERE      Q2.CQ=E2.CQ
                                AND        E2.CM=E1.CM
                                GROUP BY  AUTORE)
```

Esercizio 4

- 1a)** SELECT *
 FROM PRODOTTO
 WHERE PRODOTTO.CP NOT IN
 (SELECT C2.CP
 FROM COMPRENDE C1, PRODOTTO, COMPRENDE C2
 WHERE C1.CP=PRODOTTO.CP
 AND C2.CO=C1.CO
 AND PRODOTTO.DESCRIZIONE='Uva')
- 2a)** $\text{PRODOTTO} - \text{PRODOTTO} \bowtie$
 $(\text{COMPRENDE} \bowtie \pi_{CO}(\text{COMPRENDE} \bowtie \sigma_{\text{DESCRIZIONE}='Uva'}(\text{PRODOTTO})))$
- 1b)** SELECT *
 FROM SOCIO
 WHERE SOCIO.CS NOT IN
 (SELECT RITIRA.CS
 FROM COMPRENDE, PRODOTTO, RITIRA
 WHERE COMPRENDE.CP=PRODOTTO.CP
 AND COMPRENDE.CO= RITIRA.CO
 AND PRODOTTO.DESCRIZIONE='Uva')
- 2b)** $\text{SOCIO} - \text{SOCIO} \bowtie$
 $(\text{RITIRA} \bowtie \pi_{CO}(\text{COMPRENDE} \bowtie \sigma_{\text{DESCRIZIONE}='Uva'}(\text{PRODOTTO})))$
- 1c)** SELECT *
 FROM SOCIO
 WHERE NOT EXISTS
 (SELECT *
 FROM COMPRENDE, PRODOTTO
 WHERE COMPRENDE.CP=PRODOTTO.CP
 AND PRODOTTO.DESCRIZIONE='Uva'
 AND NOT EXISTS
 (SELECT *
 FROM RITIRA
 WHERE SOCIO.CS= RITIRA.CS
 AND COMPRENDE.CO= RITIRA.CO))
- 2c)** $\text{SOCIO} \bowtie (\pi_{CS, CO}(\text{RITIRA}) \div \pi_{CO}(\text{COMPRENDE} \bowtie \sigma_{\text{DESCRIZIONE}='Uva'}(\text{PRODOTTO})))$
- d)** SELECT RITIRA.CS, COUNT(*)
 FROM COMPRENDE, PRODOTTO, RITIRA
 WHERE COMPRENDE.CP = PRODOTTO.CP
 AND COMPRENDE.CO = RITIRA.CO
 AND PRODOTTO.DESCRIZIONE='Uva'
 GROUP BY RITIRA.CS

Esercizio 5

1a) SELECT *
 FROM VIA
 WHERE CV IN (SELECT CVA
 FROM INC,VIA
 WHERE INC.CVB=VIA.CV
 AND QUARTIERE='Pastena'
 OR CV IN (SELECT CVB
 FROM INC,VIA
 WHERE INC.CVA=VIA.CV
 AND QUARTIERE='Pastena'

2a) $R_1 = \pi_{CV}(\sigma_{\text{QUARTIERE}='Pastena'}(\text{VIA}))$
 $\text{VIA} \bowtie_{CV=CVB} \pi_{CVB}(\text{INC} \bowtie_{CVA=CV} (R_1)) \cup \text{VIA} \bowtie_{CV=CVA} \pi_{CVA}(\text{INC} \bowtie_{CVB=CV} (R_1))$

1b) SELECT *
 FROM VIA
 WHERE CV NOT IN (SELECT CVA
 FROM INC,VIA
 WHERE INC.CVB=VIA.CV
 AND NOME='MarcoPolo')
 AND CV NOT IN (SELECT CVB
 FROM INC,VIA
 WHERE INC.CVA=VIA.CV
 AND NOME='MarcoPolo')

2b) $R_2 = \pi_{CV}(\sigma_{\text{QUARTIERE}='MarcoPolo'}(\text{VIA}))$
 $\text{VIA} - (\text{VIA} \bowtie_{CV=CVB} \pi_{CVB}(\text{INC} \bowtie_{CVA=CV} (R_2)) \cup \text{VIA} \bowtie_{CV=CVA} \pi_{CVA}(\text{INC} \bowtie_{CVB=CV} (R_2)))$

1c) SELECT V1.CV AS CODICE1, V2.CV AS CODICE2
 FROM VIA V1, VIA V2
 WHERE V1.LUNGHEZZA=V2.LUNGHEZZA
 AND V1.CV > V2.CV

d) SELECT QUARTIERE
 FROM VIA
 GROUP BY QUARTIERE
 HAVING COUNT(*) >= ALL (SELECT COUNT(*)
 FROM VIA
 GROUP BY QUARTIERE)

e) SELECT QUARTIERE, CV
 FROM VIA V1
 WHERE LUNGHEZZA = (SELECT MAX(LUNGHEZZA)
 FROM VIA V2
 WHERE V1.QUARTIERE=V2.QUARTIERE)

```

f) SELECT *
      FROM VIA X
      WHERE NOT EXISTS
            ( SELECT *
              FROM VIA Y
              WHERE QUARTIERE='Pastena'
              AND    NOT EXISTS ( SELECT *
                                FROM INC
                                WHERE (X.CV=CVA AND Y.CV=CVB)
                                OR     (X.CV=CVB AND Y.CV=CVA))
            )

```

La soluzione in SQL di **a)**, **b)** e **f)** può essere semplificata usando la view INCSIM che contiene le tuple di INC e le rispettive tuple simmetriche:

```

CREATE VIEW INCSIM(CV,CVINC)
AS      SELECT CVA,CVB
        FROM INC
        UNION
        SELECT CVB,CVA
        FROM INC

```

Con tale view, possiamo risolvere le interrogazioni a), b) e f) come segue:

```

1a) SELECT V1.*
      FROM VIA V1, INCSIM, VIA V2
      WHERE V1.CV=INCSIM.CV
      AND   INCSIM.CVINC=V2.CV
      AND   V2.QUARTIERE='Pastena'

1b) SELECT *
      FROM VIA
      WHERE CV NOT IN ( SELECT INCSIM.CV
                        FROM VIA,INCSIM
                        WHERE INCSIM.CVINC=VIA.CV
                        AND     NOME='MarcoPolo')

f) SELECT *
      FROM VIA X
      WHERE NOT EXISTS
            ( SELECT *
              FROM VIA Y
              WHERE QUARTIERE='Pastena'
              AND    NOT EXISTS
                    ( SELECT *
                      FROM INCSIM
                      WHERE X.CV=INCSIM.CV
                      AND    Y.CV=INCSIM.CVINC))

```


Esercizio 6

1a) SELECT INCONTRO.*
 FROM INCONTRO,CAMPO
 WHERE INCONTRO.NCAMPO= CAMPO.NCAMPO
 AND TIPO='erba'

2a) $Ra = INCONTRO \bowtie \pi_{NCAMPO}(\sigma_{TIPO='erba'}(CAMPO))$

1b) SELECT *
 FROM CAMPO
 WHERE TIPO='erba'
 AND NCAMPO NOT IN
 (SELECT NCAMPO
 FROM INCONTRO)

2b) $Rb = \sigma_{TIPO='erba'}(CAMPO) - CAMPO \bowtie \pi_{NCAMPO}(INCONTRO)$

1c) SELECT *
 FROM TENNISTA
 WHERE CF IN (SELECT GIOC1
 FROM INCONTRO I,CAMPO C
 WHERE I.NCAMPO= C.NCAMPO
 AND TIPO='erba'
 AND SET1 < SET2)
 OR CF IN (SELECT GIOC2
 FROM INCONTRO I, CAMPO C
 WHERE I.NCAMPO= C.NCAMPO
 AND TIPO='erba'
 AND SET1 > SET2)

2c) $Rc = TENNISTA \bowtie_{CF=GIOC1} \pi_{GIOC1}(\sigma_{SET1 > SET2}(Ra)) \cup$
 $TENNISTA \bowtie_{CF=GIOC2} \pi_{GIOC2}(\sigma_{SET1 < SET2}(Ra))$

1d) SELECT DISTINCT NAZIONE
 FROM TENNISTA
 WHERE NAZIONE NOT IN (SELECT NAZIONE
 FROM INCONTRO,TENNISTA
 WHERE GIOC1= CF
 AND SET1 < SET2)
 AND NAZIONE NOT IN (SELECT NAZIONE
 FROM INCONTRO,TENNISTA
 WHERE GIOC2= CF
 AND SET1 > SET2)

2d) Se indichiamo con $R1$ i giocatori che hanno perso almeno una partita:

$R1 = TENNISTA \bowtie_{CF=GIOC1} \pi_{GIOC1}(\sigma_{SET1 < SET2}(INCONTRO)) \cup$
 $TENNISTA \bowtie_{CF=GIOC2} \pi_{GIOC2}(\sigma_{SET1 > SET2}(INCONTRO))$

allora le nazioni dei giocatori che hanno sempre vinto sono:

$\pi_{NAZIONE}(TENNISTA) - \pi_{NAZIONE}(R1)$

```
e) SELECT CAMPO.NCAMPO
FROM INCONTRO,CAMPO
WHERE INCONTRO.NCAMPO= CAMPO.NCAMPO
AND TIPO='erba'
GROUP BY CAMPO.NCAMPO
HAVING COUNT(*) <= ALL ( SELECT COUNT(*)
FROM INCONTRO I, CAMPO C
WHERE I.NCAMPO=C.NCAMPO
AND TIPO='erba'
GROUP BY CAMPO.NCAMPO)
```

La soluzione in SQL delle interrogazioni c) e d) può essere formulata in modo più semplice introducendo la seguente view:

```
CREATE VIEW INCTEN(CI,GIOC,AVV,SETG,SETA,NCAMPO)
AS SELECT CI,GIOC1,GIOC2,SET1,SET2,NCAMPO
FROM INCONTRO
UNION
SELECT CI,GIOC2,GIOC1,SET2,SET1,NCAMPO
FROM INCONTRO
```

che riporta, per ogni coppia incontro - giocatore (CI,GIOC), l'avversario AVV, il numero di set SETG,SETA vinti da GIOC e AVV rispettivamente, e il nome del campo. Con tale view, possiamo risolvere c) e d) come segue:

```
1c) SELECT TENNISTA.*
FROM TENNISTA,INCTEN,CAMPO
WHERE TENNISTA.CF=INCTEN.GIOC
AND INCTEN.NCAMPO=CAMPO.NCAMPO
AND SETG > SETA
AND TIPO='erba'
```

```
1d) SELECT DISTINCT NAZIONE
FROM TENNISTA
WHERE NAZIONE NOT IN ( SELECT NAZIONE
FROM INCTEN,TENNISTA
WHERE GIOC= CF
AND SETG < SETA)
```

Nota) Se nell'interrogazione d) vogliamo escludere gli incontri tra due giocatori della stessa nazione:

```
SELECT DISTINCT NAZIONE
FROM TENNISTA
WHERE NAZIONE NOT IN ( SELECT NAZIONE
FROM INCTEN,TENNISTA T1,TENNISTA T2,
WHERE GIOC=T1.CF
AND AVV=T2.CF
AND T1.NAZIONE <>T2.NAZIONE
AND SETG < SETA)
```

Esercizio 7

- 1a)** SELECT CODOP,NOME FROM OPERATORE
WHERE CODOP NOT IN (SELECT CODOP FROM RECLAMO)
- 2a)** $\pi_{\text{CODOP},\text{NOME}}(\text{OPERATORE}) - \pi_{\text{CODOP},\text{NOME}}(\text{OPERATORE} \bowtie \pi_{\text{CODOP}}(\text{RECLAMO}))$
- 1b)** SELECT LX.CODOP
FROM LOTTO LX
WHERE NOT EXISTS
 (SELECT *
 FROM LOTTO LY
 WHERE LX.CODOP = LY.CODOP
 AND NOT EXISTS
 (SELECT *
 FROM RECLAMO
 WHERE LY.CODOP = RECLAMO.CODOP
 AND LY.CODART = RECLAMO.CODART))
- 2b)** $\pi_{\text{CODOP}}(\text{LOTTO}) - \pi_{\text{CODOP}}(\pi_{\text{CODART},\text{CODOP}}(\text{LOTTO}) - \pi_{\text{CODART},\text{CODOP}}(\text{RECLAMO}))$
- 1c)** SELECT NOMECL
FROM RECLAMO RX
WHERE NOT EXISTS
 (SELECT *
 FROM OPERATORE
 WHERE NOT EXISTS
 (SELECT *
 FROM RECLAMO RY
 WHERE RX.NOMECL = RY.NOMECL
 AND OPERATORE.CODOP = RY.CODOP))
- 2c)** $\pi_{\text{NOMECL},\text{CODOP}}(\text{RECLAMO}) \div \pi_{\text{CODOP}}(\text{OPERATORE})$
- d)** SELECT LX.CODART, LX.CODOP
FROM LOTTO LX
WHERE LX.TOTESEM IS NOT NULL
AND LX.TOTESEM >= ALL (SELECT LY.TOTESEM
 FROM LOTTO LY
 WHERE LY.TOTESEM is not null
 AND LX.CODART = LY.CODART)

e)

```
SELECT  LOTTO.CODART, LOTTO.CODOP, LOTTO.TOTEM  
FROM    LOTTO,RECLAMO  
WHERE   LOTTO.CODART = RECLAMO.CODART  
AND     LOTTO.CODOP = RECLAMO.OP  
GROUP BY LOTTO.CODART, LOTTO.CODOP,LOTTO.TOTEM  
HAVING  COUNT(*) >= ALL ( SELECT  COUNT(*)  
                           FROM    RECLAMO  
                           GROUP BY CODART,CODOP)
```

Esercizio 8

- 1a)** SELECT *
 FROM MANIFESTAZIONE M
 WHERE CM NOT EXISTS (SELECT *
 FROM LUOGO L, SPETTACOLO S
 WHERE M.CM = S.CM
 AND L.NOME-LUOGO = S.NOME-LUOGO
 AND L.CITTÀ = 'Modena')
- 2a)** $MANIFESTAZIONE \bowtie (\pi_{CM}(MANIFESTAZIONE) - \pi_{CM}((SPETTACOLO \bowtie \sigma_{CITTÀ='Modena'}(LUOGO))))$
- 1b)** SELECT DISTINCT NOME-LUOGO
 FROM SPETTACOLO X
 WHERE NOT EXISTS
 (SELECT *
 FROM MANIFESTAZIONE M
 WHERE NOT EXISTS
 (SELECT *
 FROM SPETTACOLO Y
 WHERE X.NOME-LUOGO = Y.NOME-LUOGO
 AND M.CM=Y.CM))
- 2b)** $\pi_{NOME-LUOGO, COD-MAN}(SPETTACOLO) \div \pi_{COD-MAN}(MANIFESTAZIONE)$
- 1c)** SELECT DISTINCT NOME-LUOGO
 FROM SPETTACOLO SX
 WHERE ORA-INIZIO > 15
 AND 3 < (SELECT COUNT(*)
 FROM SPETTACOLO SY
 WHERE ORA-INIZIO > 15
 AND SX.NOME-LUOGO = SY.NOME-LUOGO
 AND SX.DATA = SY.DATA)
- d)** SELECT NOME-LUOGO, COUNT(distinct CM) as N-MAN,
 COUNT(*) as N-SPET
 FROM SPETTACOLO
 GROUP BY NOME-LUOGO
- e)** Seleziona i codici delle manifestazioni i cui spettacoli sono iniziati sempre dopo le ore 15. In SQL:
 SELECT DISTINCT X.CM
 FROM SPETTACOLO X
 WHERE NOT EXISTS (SELECT *
 FROM SPETTACOLO Y
 WHERE X.CM = Y.CM
 AND (Y.ORA-INIZIO < 15))

Esercizio 9

- 1a)** SELECT PRODOTTO.*
 FROM PRODOTTO,FORNISCE,FORNITORE
 WHERE PRODOTTO.CODP=FORNISCE.CODP
 AND FORNISCE.CODF=FORNITORE.CODF
 AND ANNO=1994
 AND CITTÀ='MO'
- 2a)** $R_a = \text{PRODOTTO} \bowtie (\sigma_{\text{ANNO}=1994}(\text{FORNISCE}) \bowtie \sigma_{\text{CITTÀ}='MO'}(\text{FORNITORE}))$
- 1b)** SELECT *
 FROM PRODOTTO
 WHERE CODP NOT IN (SELECT CODP
 FROM FORNISCE,FORNITORE
 WHERE FORNISCE.CODF=FORNITORE.CODF
 AND CITTÀ='MO')
- 2b)** $\text{PRODOTTO} - \text{PRODOTTO} \bowtie (\text{FORNISCE} \bowtie \sigma_{\text{CITTÀ}='MO'}(\text{FORNITORE}))$
- 1c)** SELECT PRODOTTO.*
 FROM PRODOTTO,FORNISCE,FORNITORE
 WHERE PRODOTTO.CODP=FORNISCE.CODP
 AND FORNISCE.CODF=FORNITORE.CODF
 AND ANNO=1994
 AND CITTÀ='MO'
 AND PRODOTTO.CODP NOT IN
 (SELECT PRODOTTO.CODP
 FROM PRODOTTO,FORNISCE,FORNITORE
 WHERE PRODOTTO.CODP=FORNISCE.CODP
 AND FORNISCE.CODF=FORNITORE.CODF
 AND ANNO=1994
 AND CITTÀ <> 'MO')
- 2c)** $R_a - \text{PRODOTTO} \bowtie (\sigma_{\text{ANNO}=1994}(\text{FORNISCE}) \bowtie \sigma_{\text{CITTÀ} <> 'MO'}(\text{FORNITORE}))$
- d)** SELECT ANNO,SUM(QTY)
 FROM FORNISCE,FORNITORE
 WHERE FORNISCE.CODF=FORNITORE.CODF
 AND CITTÀ = 'MO'
 GROUP BY ANNO

e)

```
SELECT  ANNO,CODF
FROM    FORNISCHE FX
GROUP BY ANNO,CODF
HAVING  SUM(qty) >=ALL ( SELECT  SUM(qty)
                          FROM    FORNISCHE FY
                          WHERE    FY.ANNO=FX.ANNO
                          GROUP BY CODF)
```

Esercizio 10

- 2a) $\text{GIOCATOREGOLF} \bowtie (\text{PARTECIPA} \bowtie \sigma_{\text{LIVELLO}='nazionale'}(\text{GARA}))$
- 1b)

```
SELECT NAZIONE
FROM   GIOCATOREGOLF
WHERE  NAZIONE NOT IN
      ( SELECT NAZIONE
        FROM   GIOCATOREGOLF G, PARTECIPA P
        WHERE  G.CF= P.CF AND PUNTEGGIO > 0 )
```
- 2b) $\pi_{\text{NAZIONE}}(\text{GIOCATOREGOLF}) - \pi_{\text{NAZIONE}}(\text{GIOCATOREGOLF} \bowtie \sigma_{\text{LIVELLO}='nazionale'}(\text{PARTECIPA}))$
- 1c)

```
SELECT *
FROM   GIOCATOREGOLF G
WHERE  NOT EXISTS
      ( SELECT *
        FROM   GARA
        WHERE  LIVELLO='nazionale'
        AND    NOT EXISTS
              ( SELECT *
                FROM   PARTECIPA P
                WHERE  GARA.CG=P.CG
                AND    G.CF = P.CF))
```
- 2c) $\text{GIOCATOREGOLF} \bowtie (\pi_{\text{CF,CG}}(\text{PARTECIPA}) \div \pi_{\text{CG}}(\sigma_{\text{LIVELLO}='nazionale'}(\text{GARA})))$
- d)

```
SELECT *
FROM   GIOCATOREGOLF G
WHERE  CF IN ( SELECT P.CF-GIOCATOREGOLF
              FROM   GARA, PARTECIPA P
              WHERE  GARA.CG=P.CG
              AND    LIVELLO='internazionale'
              AND    PUNTEGGIO =
                    ( SELECT MIN(PUNTEGGIO)
                      FROM   PARTECIPA P1
                      WHERE  P1.CG=P.CG ) )
```
- e)

```
SELECT      G.NAZIONE, AVG(P.PUNTEGGIO)
FROM        GIOCATOREGOLF G,GARA, PARTECIPA P
WHERE       G.CF= P.CF
AND         GARA.CG=P.CG
AND         LIVELLO='internazionale'
GROUP BY    G.NAZIONE
HAVING      COUNT(distinct P.CF) > 5
ORDER BY    2 DESC
```


Esercizio 11

- 1a)

```
SELECT *
FROM CLIENTE
WHERE NTESS NOT IN
      ( SELECT NTESS
        FROM ACQUISTO, CD
        WHERE ACQUISTO.CODCD=CD.CODCD
        AND   CASADISCO='DiscoJolli'
        AND   DATA > '1/1/1997')
```
- 2a) $\text{CLIENTE} - \text{CLIENTE} \bowtie \left(\sigma_{\text{CASADISCO}='DiscoJolli'}(\text{CD}) \bowtie \sigma_{\text{DATA}>'1/1/1997'}(\text{ACQUISTO}) \right)$
- 1b)

```
SELECT NTESS
FROM CLIENTE
WHERE NOT EXISTS
      ( SELECT *
        FROM CD
        WHERE AUTORE='Francesco Guccini'
        AND   NOT EXISTS
              ( SELECT *
                FROM ACQUISTO
                WHERE ACQUISTO.NTESS=CLIENTE.NTESS
                AND   ACQUISTO.CODCD= CD.CODCD))
```
- 2b) $\pi_{\text{NTESS}, \text{CODCD}}(\text{ACQUISTO}) \div \pi_{\text{CODCD}}(\sigma_{\text{AUTORE}='FrancescoGuccini'}(\text{CD}))$
- c)

```
SELECT  CODCD, SUM(QTY)
FROM    ACQUISTO
GROUP BY CODCD
```
- d)

```
SELECT  CASADISCO, NTESS
FROM    ACQUISTO AX, CD
WHERE   AX.CODCD=CD.CODCD
GROUP BY CASADISCO, NTESS
HAVING  SUM(QTY) >= ALL
      ( SELECT  SUM(QTY)
        FROM    ACQUISTO AY, CD
        WHERE   AY.CODCD=CD.CODCD
        AND     AY.CASADISCO=AX.CASADISCO
        GROUP BY NTESS)
```

Esercizio 12

- 1a)** SELECT *
 FROM DIPENDENTE
 WHERE CITTÀ='MO'
 AND CF NOT IN (SELECT CF
 FROM LAVORA,PROGETTO
 WHERE LAVORA.CP=PROGETTO.CP
 AND ANNO=1995)
- 2a)** $\sigma_{CITTÀ='MO'}(DIPENDENTE) -$
 $DIPENDENTE \bowtie (LAVORA \bowtie \sigma_{ANNO=1995}(PROGETTO))$
- 1b)** SELECT *
 FROM DIPENDENTE
 WHERE CF NOT IN (SELECT L1.CF
 FROM LAVORA L1, DIPENDENTE, LAVORA L2
 WHERE L1.CP=L2.CP
 AND L2.CF=DIPENDENTE.CF
 AND DIPENDENTE.CITTÀ='MO')
- 2b)** DIPENDENTE —
 $DIPENDENTE \bowtie (LAVORA \bowtie \pi_{CP}(LAVORA \bowtie \sigma_{CITTÀ='MO'}(DIPENDENTE)))$
- c)** SELECT L1.CF as CF1,L2.CF as CF2
 FROM LAVORA L1, LAVORA L2
 WHERE L1.CP=L2.CP
 AND L1.CF < L2.CF
- d)** SELECT *
 FROM DIPENDENTE
 WHERE 3 <= (SELECT COUNT(DISTINCT RUOLO)
 FROM LAVORA
 WHERE LAVORA.CF=DIPENDENTE.CF)
- e)** SELECT CF,CP
 FROM LAVORA LX
 GROUP BY CF,CP
 HAVING SUM(LX.MESI) >= ALL (SELECT SUM(LY.MESI)
 FROM LAVORA LY
 WHERE LY.CF=LX.CF
 GROUP BY CP)

Esercizio 13

2a) $\sigma_{\text{TIPO}='Mac'}(\text{PC}) - \text{PC} \bowtie (\sigma_{\text{DATA} >= 1/7/97 \text{ and } \text{DATA} <= 1/11/97}(\text{RIPARAZIONE}))$

1b) SELECT *
 FROM TECNICO
 WHERE NOT EXISTS
 (SELECT *
 FROM PC
 WHERE TIPO='Mac'
 AND NOT EXISTS
 (SELECT *
 FROM RIPARAZIONE
 WHERE RIPARAZIONE.PC=PC.PC
 AND RIPARAZIONE.CF=TECNICO.CF))

2b) $\text{TECNICO} \bowtie (\pi_{\text{PC}, \text{CF}}(\text{RIPARAZIONE}) \div \pi_{\text{PC}}(\sigma_{\text{MARCA}='IBM'}(\text{PC})))$

1c) SELECT RG1.PC, RG2.PC
 FROM RIPARAZIONE RG1, RIPARAZIONE RG2
 WHERE RG1.CF=RG2.CF
 AND RG1.DATA=RG2.DATA
 AND RG1.PC < RG2.PC

1d) SELECT *
 FROM PC
 WHERE 10 <= (SELECT SUM(ORE)
 FROM RIPARAZIONE
 WHERE RIPARAZIONE.PC=PC.PC)

1e) SELECT DISTINCT DATA, CF
 FROM RIPARAZIONE RG1
 GROUP BY DATA, CF
 HAVING COUNT(PC) >= ALL (SELECT COUNT(PC)
 FROM RIPARAZIONE RG2
 WHERE RG2.DATA=RG1.DATA
 GROUP BY CF)

1f) SELECT *
 FROM PC
 WHERE PC IN
 (SELECT PC
 FROM RIPARAZIONE , TECNICO
 WHERE RIPARAZIONE.CF=TECNICO.CF
 GROUP BY PC
 HAVING SUM(ORE*COSTO) >= ALL
 (SELECT SUM(ORE*COSTO)
 FROM RIPARAZIONE, TECNICO
 WHERE RIPARAZIONE.CF=TECNICO.CF
 GROUP BY PC))

Esercizio 14

- 1a)** SELECT N1.*
 FROM NAZIONE N1,CONFERENZA C,PARTECIPA P, NAZIONE N2
 WHERE N1.CN=P.CN
 AND P.CC=C.CC
 AND C.CNSEDE=N2.CN
 AND N2.CONTINENTE='Europa'
- 2a)** $NAZIONE \bowtie (PARTECIPA \bowtie CONFERENZA \bowtie \sigma_{CONTINENTE \neq 'Europa'}(NAZIONE))$
- 1b)** SELECT N.*
 FROM NAZIONE N,CONFERENZA C,PARTECIPA P
 WHERE N.CN=P.CN
 AND P.CC=C.CC
 AND C.CNSEDE=N.CN
- 2b)** $NAZIONE \bowtie (\pi_{CC}(NAZIONE \bowtie \sigma_{PARTECIPA.CC = CONFERENZA.CC \wedge (CONFERENZA \bowtie \sigma_{PARTECIPA.CC = CONFERENZA.CC})}))$
- 1c)** SELECT *
 FROM NAZIONE
 WHERE CN NOT IN (SELECT P1.CN
 FROM PARTECIPA P1,NAZIONE,PARTECIPA P2
 WHERE P1.CC=P2.CC
 AND P2.CN=NAZIONE.CN
 AND CONTINENTE='Europa')
- 2c)** $Ra = \pi_{CC}(PARTECIPA \bowtie \sigma_{CONTINENTE \neq 'Europa'}(NAZIONE))$
 $NAZIONE \bowtie (\pi_{CN}(NAZIONE) - \pi_{CN}(PARTECIPA \bowtie Ra))$
- d)** SELECT NAZIONE.*
 FROM NAZIONE,CONFERENZA
 WHERE NAZIONE.CN=CONFERENZA.CNSEDE
 AND CC in (SELECT CC
 FROM PARTECIPA
 GROUP BY CC
 HAVING SUM(NUMEROP) >= ALL
 (SELECT SUM(NUMEROP)
 FROM PARTECIPA
 GROUP BY CC))
- e)** SELECT CN,CC
 FROM PARTECIPA P1
 WHERE NUMEROP = (SELECT MAX(NUMEROP)
 FROM PARTECIPA P2
 WHERE P2.CN=P1.CN)

Esercizio 15

- 1a)** SELECT MOSTRA.*
 FROM QUADRO, MOSTRA, ESPONE
 WHERE QUADRO.CQ=ESPONE.CQ
 AND MOSTRA.CM=ESPONE.CM
 AND (ANNO='96' OR ANNO='97')
 AND AUTORE='Picasso'
- 2a)** $\sigma_{\text{ANNO}='96' \text{ OR } \text{ANNO}='97'}(\text{MOSTRA}) \bowtie (\text{ESPONE} \bowtie \sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO}))$
- 1b)** SELECT NOME
 FROM MOSTRA
 WHERE NOT EXISTS
 (SELECT *
 FROM QUADRO
 WHERE AUTORE='Picasso'
 AND NOT EXISTS
 (SELECT *
 FROM ESPONE
 WHERE MOSTRA.CM=ESPONE.CM
 AND QUADRO.CQ=ESPONE.CQ))
- 2b)** $\pi_{\text{NOME}, \text{CQ}}(\text{ESPONE}) \div \pi_{\text{CQ}}(\sigma_{\text{AUTORE}='Picasso'}(\text{QUADRO}))$
- c)** SELECT M1.ANNO, M1.NOME, M2.NOME, M1.ORGANIZZATORE
 FROM MOSTRA M1, MOSTRA M2
 WHERE M1.ANNO=M2.ANNO
 AND M1.ORGANIZZATORE=M2.ORGANIZZATORE
 AND M1.NOME > M2.NOME
- d)** SELECT *
 FROM QUADRO Q1
 WHERE 3 > (SELECT COUNT(*)
 FROM ESPONE E1, QUADRO Q2, ESPONE E2
 WHERE E1.CQ=Q2.CQ
 AND Q2.AUTORE='Picasso'
 AND E2.CM=E1.CM
 AND E2.CQ=Q1.CQ)

6.3 Normalizzazione

◇ Esercizio 1

Dato il seguente schema di relazione:

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

e considerando le seguenti dipendenze funzionali:

- (FD1) $A B \rightarrow C E$
- (FD2) $C \rightarrow D$
- (FD3) $D \rightarrow B$

viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione;
2. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
3. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali;
4. Produrre uno schema E/R che descriva lo schema di relazione e soddisfi le dipendenze funzionali date.

◇ Esercizio 2

Dato il seguente schema di relazione:

$\text{Magazzino}(\text{Locale}, \text{Prodotto}, \text{Stanza}, \text{Scaffale})$

e considerando i seguenti vincoli:

- Un prodotto è immagazzinato in uno ed un solo locale;
- un prodotto può essere immagazzinato in uno o più stanze e in uno o più scaffali;
- in una stanza di un locale, uno scaffale immagazzina un preciso prodotto.

viene richiesto di:

1. Determinare le dipendenze funzionali (non banali) insite nello schema di relazione.
2. Determinare la chiave o le chiavi dello schema di relazione.
3. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF.
4. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

Esempio di istanza r dello schema di relazione Magazzino che soddisfa i vincoli:

Locale	Prodotto	Stanza	Scaffale
Pelletteria	Scarpa	A	15
Pelletteria	Scarpa	B	6
Pelletteria	Cintura	A	16
Abbigliamento	Pantalone	A	14
Abbigliamento	Pantalone	A	15

◇ **Esercizio 3**

Dato il seguente schema di relazione:

Tornei(Torneo,Squadra,Categoria,Capitano)

e considerando i seguenti vincoli:

- Un capitano gioca in una precisa squadra di una precisa categoria;
- Per ogni squadra e categoria c'è un solo capitano;
- Per ogni torneo, una squadra partecipa con una sola categoria;

viene richiesto di:

1. Determinare le dipendenze funzionali (non banali) insite nello schema di relazione.
2. Determinare la chiave o le chiavi dello schema di relazione.
3. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF.
4. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

Esempio di istanza r dello schema di relazione Tornei che soddisfa i vincoli:

Torneo	Squadra	Categoria	Capitano
TorneoEstivo1997	Modena	Ragazzi	Neri
TorneoTopolino1998	Modena	Pulcini	Bianchi
TorneoDiNatale1997	Modena	Juniores	Pergola
TorneoTopolino1997	Modena	Pulcini	Bianchi
TorneoEstivo1997	Bologna	Pulcini	Verdi
TorneoDiNatale1997	Bologna	Ragazzi	Russo
TorneoTopolino1998	Bologna	Pulcini	Verdi
TorneoEstivo1997	Firenze	Pulcini	Rossi

◇ **Esercizio 4**

Dato il seguente schema di relazione:

$R(A,B,C,D)$

e considerando le seguenti dipendenze funzionali:

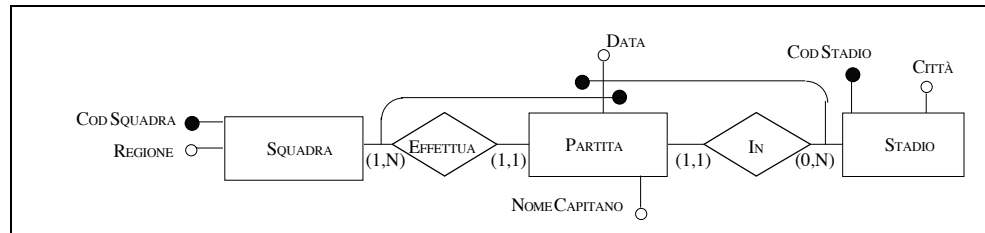
- (FD1) $A \rightarrow B$
- (FD2) $BC \rightarrow D$
- (FD3) $A \rightarrow C$

viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione;
2. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
3. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

◇ Esercizio 5

Dato il seguente schema E/R:



si consideri la seguente traduzione in schema relazionale:

Squadra(CodSquadra, Regione)

Stadio(CodStadio, Città)

Partita(Data, CodSquadra, NomeCapitano, CodStadio)

e si consideri sullo schema di relazione Partita la seguente dipendenza funzionale aggiuntiva (che esprime il vincolo che un capitano gioca in una sola squadra): (FD) NomeCapitano \rightarrow CodSquadra.

Viene richiesto di

1. Determinare la chiave o le chiavi dello schema di relazione Partita, prendendo in considerazione sia i vincoli dello schema E/R sia la dipendenza funzionale aggiunta;
2. Determinare se lo schema di relazione Partita è in 2NF, 3NF e BCNF;
3. Produrre eventuali decomposizioni dello schema di relazione Partita che preservano i dati ma non le dipendenze e scrivere una query SQL che ne controlli la violazione.

◇ Esercizio 6

Dato il seguente schema di relazione:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

- (FD1) $AB \rightarrow C$
- (FD2) $AB \rightarrow D$
- (FD3) $C \rightarrow A$
- (FD4) $D \rightarrow B$

viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione.
2. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF.
3. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

◇ **Esercizio 7**

Dato il seguente schema di relazione:

Libretto(Matricola, NomeStudente, Corso, Professore, Voto)

e considerando i seguenti vincoli:

- Ad ogni studente viene attribuito un numero di matricola unico;
- Ogni studente può registrare un unico voto per ogni corso seguito;
- Ogni professore tiene un unico corso;
- Ogni corso può essere tenuto da più professori (es. Rossi e Neri tengono entrambi un corso di Informatica I); gli studenti vengono assegnati ai corsi sulla base del loro nome (es. Bianchi ha seguito Informatica I con Rossi mentre Verdi lo ha seguito con Neri);
- Possono essere attivati corsi anche senza studenti iscritti.

viene richiesto di:

1. Determinare le dipendenze funzionali (non banali) insite nello schema di relazione;
2. Determinare la chiave o le chiavi dello schema di relazione;
3. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
4. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali;
5. Discutere brevemente se le relazioni in BCNF ottenute presentano comunque problemi.

◇ **Esercizio 8**

Dato il seguente schema di relazione:

Progetti(Anno, Progetto, CapoProgetto, Reparto, Responsabile)

e considerando le seguenti dipendenze funzionali:

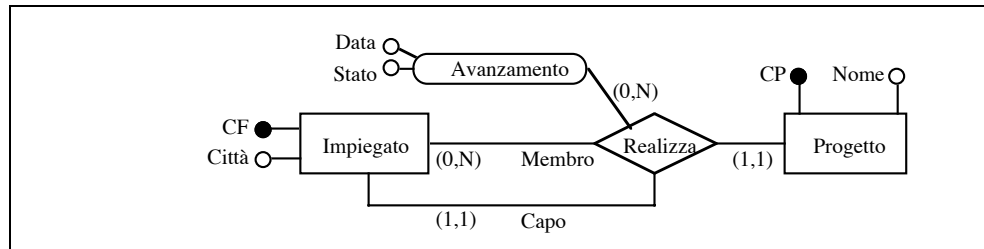
- (FD1) Anno Progetto \rightarrow CapoProgetto
- (FD2) CapoProgetto \rightarrow Reparto
- (FD3) Reparto \rightarrow Responsabile
- (FD4) Anno Progetto \rightarrow Responsabile

viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione;
2. Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
3. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali;
4. Produrre uno schema E/R che descriva lo schema di relazione e soddisfi le dipendenze funzionali date.

◇ **Esercizio 9**

Dato il seguente schema E/R:



viene richiesto di

1. Tradurre lo schema E/R in schema relazionale in terza forma normale;
2. Aggiungere allo schema relazionale ottenuto al punto 1) il vincolo che in una certa Data lo Stato di Avanzamento di un certo Progetto è unico;
3. Tradurre lo schema E/R in un'unico schema di relazione e
 - (a) determinare le dipendenze funzionali (non banali) insite nello schema di relazione;
 - (b) determinare la chiave o le chiavi dello schema di relazione;
 - (c) Determinare se lo schema di relazione è in 2NF, 3NF e BCNF;
 - (d) Determinare una eventuale decomposizione almeno in 3NF che sia lossless.

6.3.1 Soluzioni

Nella soluzione di questi esercizi la normalizzazione di uno schema $\langle R, F \rangle$ viene effettuata in maniera “euristica”, applicando ricorsivamente il criterio di decomposizione:

Uno schema $\langle R(X, Y, A), F \rangle$ che non è nella forma normale desiderata a causa di $X \rightarrow A$ viene decomposto in $R1(X, Y)$ e $R2(X, A)$.

Inoltre la violazione della forma normale desiderata sugli schemi $R1$ e $R2$ verrà generalmente verificata solo rispetto alle dipendenze di F proiettate sugli schemi di $R1$ e $R2$.

Esercizio 1

1. Le chiavi dello schema di relazione sono $K_1=AB$, $K_2=AC$ e $K_3=AD$.

Ad esempio, dimostriamo che $AC \rightarrow ADBCE$:

- 1 $C \rightarrow B$ applicazione della transitività alla FD2 e FD3
- 2 $C \rightarrow DB$ applicazione dell'unione alla 1 e FD2
- 3 $AC \rightarrow ADB$ applicazione della estensione alla 2 con A
- 4 $AC \rightarrow ADBCE$ applicazione dell'unione alla 3 e FD1

Quindi AC è superchiave; è facile verificare che è anche chiave.

2. Lo schema di relazione non è in BCNF a causa sia della FD2 che della FD3. Lo schema di relazione è in 3NF nonostante la presenza di tali dipendenze funzionali, in quanto gli attributi B e D sono attributi primi.
3. Consideriamo le seguenti decomposizioni:

- Siccome lo schema non è in BCNF a causa della FD2, consideriamo la decomposizione binaria:

$R1(C,D)$, con dipendenza funzionale FD2

$R2(A,B,C,E)$, con dipendenza funzionale FD1

Tale decomposizione è lossless in quanto il join naturale riguarda l'attributo C che è chiave in $R1$, ma non preserva la dipendenza FD3. Lo schema di relazione $R2$ ha come chiavi $K_1=AB$ e $K_2=AC$.

Entrambi i sottoschemi relazionali ottenuti risultano essere in BCNF.

- Se si decompone per proiezione sulla base della dipendenza funzionale FD3 che viola la BCNF si ottiene:

$R1(D,B)$, con dipendenza funzionale FD3

$R2(A,C,D,E)$, con dipendenza funzionale FD2

Tale decomposizione è lossless in quanto il join naturale riguarda l'attributo D che è chiave in $R1$, ma non preserva la dipendenza FD3. Lo schema di relazione $R2$ ha come chiavi $K_1=AB$ e $K_2=AC$.

Entrambi i sottoschemi relazionali ottenuti risultano essere in BCNF.

Esercizio 2

1. Dato che un prodotto è immagazzinato in uno ed un solo locale, è valida la (FD1) $\text{Prodotto} \rightarrow \text{Locale}$;
inoltre, dato che in una stanza di un locale, uno scaffale immagazzina un preciso prodotto, si ha che (FD2) $\text{Locale Stanza Scaffale} \rightarrow \text{Prodotto}$.
2. Le chiavi dello schema di relazione sono $K_1 = \text{Locale Stanza Scaffale}$ e $K_2 = \text{Prodotto Stanza Scaffale}$.
3. Lo schema di relazione non è in BCNF a causa della FD1. Lo schema di relazione è in 3NF nonostante la presenza di tale dipendenza funzionale, in quanto l'attributo Locale è un attributo primo.
4. Siccome lo schema non è in BCNF a causa della FD1, consideriamo la decomposizione binaria:

Magazzino1(Prodotto,Locale), con dipendenza funzionale FD1

Magazzino2(Prodotto,Stanza,Scaffale)

Tale decomposizione è lossless ma non preserva la dipendenza FD2.

Entrambi i sottoschemi risultano essere in BCNF.

Esercizio 3

1. Un capitano gioca in una precisa squadra di una precisa categoria:
(FD1) $\text{Capitano} \rightarrow \text{Squadra}$ e (FD2) $\text{Capitano} \rightarrow \text{Categoria}$;
per ogni squadra e categoria c'è un solo capitano:
(FD3) $\text{Squadra Categoria} \rightarrow \text{Capitano}$;
per ogni torneo, una squadra partecipa con una sola categoria:
(FD4) $\text{Squadra Torneo} \rightarrow \text{Categoria}$.
2. Le chiavi sono $K_1 = \text{Torneo Squadra}$ e $K_2 = \text{Torneo Capitano}$.
3. Lo schema di relazione non è in 2NF sia a causa della FD1 che della FD2;
Inoltre, lo schema di relazione non è in BCNF a causa della FD3.
4. Consideriamo le seguenti decomposizioni:

- (a) per proiezione sulla base di FD1 e FD2 che violano la 2NF:

Tornei1(Capitano,Squadra,Categoria), con FD1 e FD2

Tornei2(Torneo,Capitano)

- (b) proiezione sulla base di FD3 che viola la BCNF:

Tornei1(Capitano,Squadra,Categoria), con FD1 e FD2

Tornei3(Torneo,Squadra,Capitano), con FD4

Entrambe queste decomposizioni sono lossless, preservano le dipendenze funzionali e generano sottoschemi che risultano essere in BCNF.

Esercizio 4

1. Lo schema di relazione ha come unica chiave $K_1 = A$.
2. Lo schema di relazione è in 2NF ma non è in 3NF a causa della FD2.
3. La decomposizione $R1(\underline{A}, B, C)$, $R2(\underline{B}, \underline{C}, D)$ è lossless, preserva le dipendenze funzionali e genera sottoschemi che risultano essere in BCNF.

Esercizio 5

1. Dallo schema E/R si ha che Partita ha come chiavi $K_1 = \text{Data CodSquadra}$ e $K_2 = \text{Data CodStadio}$. Dalla (FD) aggiuntiva, risulta che un'ulteriore chiave di Partita è $K_3 = \text{Data NomeCapitano}$.
2. Partita non è in BCNF a causa della FD aggiuntiva (è in 3NF in quanto l'attributo CodSquadra è primo).
3. Siccome lo schema non è in BCNF a causa di FD, si considera:

$\text{Capitano}(\underline{\text{NomeCapitano}}, \text{CodSquadra})$

$\text{Partita1}(\text{Data}, \text{NomeCapitano}, \text{CodStadio})$, con chiavi K_2 e K_3

Entrambi i sottoschemi risultano essere ora in BCNF. Tale decomposizione è lossless ma non preserva la dipendenza (dovuta alla chiave K_1)

(FD1) $\text{Data CodSquadra} \rightarrow \text{CodStadio NomeCapitano}$.

Tale dipendenza vieta l'introduzione di una tupla (X, Y, Z, W) in una istanza di Partita, se l'istanza contiene già una tupla (partita) con data X e squadra Y . Per controllare che nello schema decomposto tale dipendenza non sia violata, prima di inserire (Z, Y) in un'istanza r_1 di Capitano e (X, Z, W) in un'istanza r_2 di Partita1, occorre verificare che nella data X per la squadra Y di Z non ci sia già in r_2 una partita, ovvero che il risultato della seguente query sia zero:

```
SELECT count(*)
FROM   Partita1, Capitano
WHERE  Partita1.NomeCapitano=Capitano.NomeCapitano
AND    Capitano.CodSquadra=Y
AND    Partita1.Data=X
```

Se invece si inserisce la sola tupla (X, Z, W) in r_2 occorre determinare quale sia la squadra di Z (self-join su Capitano) e quindi verificare che nella data X non ci sia già in r_2 una partita per tale squadra:

```
SELECT count(*)
FROM   Partita1, Capitano C1, Capitano C2
WHERE  C1.NomeCapitano=Z
AND    C1.CodSquadra=C2.CodSquadra
AND    Partita1.NomeCapitano=C2.NomeCapitano
AND    Capitano.CodSquadra=Y
AND    Partita1.Data=X
```

Esercizio 6

1. Le chiavi dello schema di relazione sono:
 $K_1 = AB$, $K_2 = BC$, $K_3 = CD$ e $K_4 = DA$.
2. Lo schema di relazione non è in BCNF a causa sia della FD3 che della FD4 (è in 3NF, in quanto gli attributi A e B sono attributi primi).
3. Essendo lo schema non in BCNF a causa della FD3, consideriamo la decomposizione binaria:

$R1(AC)$ con FD3, $R2(BCD)$ con FD4

Tale decomposizione è lossless ma non preserva le dipendenze FD1 e FD2. Lo schema R2 non è in BCNF a causa di FD4, quindi:

$R21(CD)$, $R22(BD)$ con FD4

che risulta essere lossless e preserva le dipendenze funzionali.

In definitiva, si ottiene la decomposizione $R1(AC)$, $R21(CD)$ e $R22(BD)$, che risulta essere lossless ma non preserva le dipendenze FD1 e FD2. Lo schema relazionale risultante è in BCNF.

Esercizio 7

1. Le dipendenze funzionali sono:
 - (FD1) $\text{Matricola} \rightarrow \text{NomeStudente}$
 - (FD2) $\text{Professore} \rightarrow \text{Corso}$
 - (FD3) $\text{NomeStudente} \text{ Corso} \rightarrow \text{Professore}$
 - (FD4) $\text{Matricola} \text{ Corso} \rightarrow \text{Voto}$
2. Le chiavi dello schema di relazione sono $K_1 = \text{Matricola} \text{ Professore}$ e $K_2 = \text{Matricola} \text{ Corso}$.
3. Lo schema di relazione non è in 2NF a causa della FD1. Lo schema di relazione non è in BCNF sia a causa della FD2 che della FD3.
4. Siccome lo schema non è in 2NF a causa della FD1:

$\text{Libretto1}(\text{Matricola}, \text{NomeStudente})$, con FD1

$\text{Libretto2}(\text{Matricola}, \text{Corso}, \text{Professore}, \text{Voto})$, con FD2 e FD4

Tale decomposizione è lossless ma non preserva la dipendenza FD3. Lo schema di relazione Libretto2 non è in BCNF a causa FD2, quindi:

$\text{Libretto21}(\text{Professore}, \text{Corso})$, con FD2

$\text{Libretto22}(\text{Matricola}, \text{Professore}, \text{Voto})$

Tale decomposizione è lossless ma non preserva la dipendenza FD4. Entrambi i sottoschemi risultano essere ora in BCNF.

Esercizio 8

È facile verificare la dipendenza FD4 è ridondante rispetto alle altre dipendenze e quindi non viene presa in considerazione.

1. Lo schema di relazione ha un'unica chiave $K_1 = \text{Anno Progetto}$.
2. Lo schema di relazione non è in 3NF a causa sia della FD2 che della FD3.
3. Decomponiamo lo schema sulla base della FD2 che viola la 3NF:

R1(CapoProgetto,Reparto), con FD2

R2(Anno,Progetto,CapoProgetto,Responsabile), con FD1

Tale decomposizione è lossless ma non preserva la dipendenza FD3.

Invece se decomponiamo lo schema sulla base della FD3 che viola la 3NF:

R1(Reparto,Responsabile), con FD3

R2(Anno,Progetto,CapoProgetto,Reparto), con FD1 e FD2

Tale decomposizione è lossless e preserva le dipendenze. Lo schema R2 non è in 3nf a causa della FD2, quindi viene decomposto in:

R21(CapoProgetto,Reparto), con FD2

R22(Anno,Progetto,CapoProgetto), con FD1

In definitiva si ottiene la decomposizione che è lossless e preserva le dipendenze:

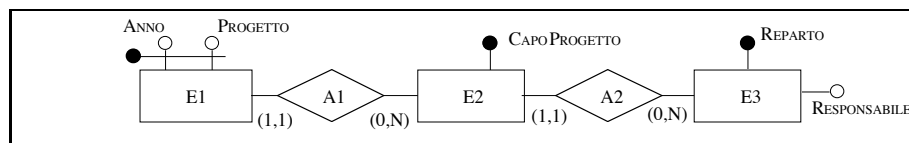
R1(Reparto,Responsabile), con FD3

R21(CapoProgetto,Reparto), con FD2

R22(Anno,Progetto,CapoProgetto), con FD1

Tutti gli schemi di relazione sono ora in BCNF.

4. Uno schema E/R equivalente allo schema di relazione e che soddisfa le dipendenze funzionali date è il seguente:



Esercizio 9

1. Impiegato(CF, Città)
 Progetto(CP, Nome)
 Realizza(CP, CFCapo, CFMembro)
 AK: CFCapo
 FK: CFCapo **REFERENCES** Impiegato
 FK: CFMembro **REFERENCES** Impiegato
 Avanzamento(CP, Data, Stato)
 FK: CP **REFERENCES** Realizza
2. È sufficiente modificare la chiave della relazione Avanzamento:
 Avanzamento(CP, Data, Stato)
3. La soluzione è riferita allo schema E/R di figura, senza prendere in considerazione il vincolo introdotto nel punto 2):
 R(CP, Nome, CFCapo, CittàCapo, CFMembro, CittàMembro, Data, Stato)

(a) Le dipendenze funzionali sono le seguenti:

- (FD1) CFCapo \rightarrow CP
- (FD2) CP \rightarrow CFCapo
- (FD3) CFCapo \rightarrow CittàCapo
- (FD4) CFMembro \rightarrow CittàMembro
- (FD5) CP \rightarrow Nome
- (FD6) CFCapo \rightarrow CFMembro
- (FD7) CP \rightarrow CFMembro

(b) Le chiavi dello schema di relazione R sono $K_1 = \text{CFCapo Data Stato}$ e $K_2 = \text{CP Data Stato}$;

(c) Chiaramente lo schema non è in 2NF dato che gli attributi non primi Nome, CittàCapo, CFMembro e CittàMembro dipendono parzialmente dalle chiavi. Si considera quindi:

R1(CP, Data, Stato, CFCapo) **AK:** CFCapo, Data, Stato
 R2(CP, Nome, CFMembro)
 R3(CFCapo, CittàCapo)
 R4(CFMembro, CittàMembro)

Tutte questi schemi di relazione sono in 3NF, però R1 non è in BCNF a causa delle dipendenze (FD1) e (FD2).

R11(CP, Data, Stato)
 R12(CP, CFCapo) **AK:** CFCapo

Questa decomposizione, oltre ad essere lossless, preserva anche le dipendenze. Lo schema relazionale risultante è in BCNF.

Appendice

In questo capitolo vengono presentati numerosi esercizi, integralmente risolti, su tutte le tematiche introdotte.

La prima parte contiene esercizi di progettazione concettuale a partire da requisiti in linguaggio naturale e conseguente progettazione logica. Nella soluzione di questi esercizi viene presentato uno tra i possibili schemi E/R corretti corrispondenti alle specifiche date e la relativa traduzione in schema relazionale. In alcune soluzioni, soprattutto quelle relative ai primi esercizi, viene riportato un breve commento allo schema E/R per giustificare le scelte effettuate. Questa prima parte termina proponendo alcuni esercizi relativi ai dati derivati.

La seconda parte contiene esercizi in cui è richiesto di esprimere interrogazioni sia in algebra relazionale che in SQL. Anche nella soluzione di questi esercizi viene presentata una possibile espressione, sia in SQL sia in algebra relazionale, che risolve l'interrogazione richiesta. I commenti sono limitati ad alcune interrogazioni particolarmente complicate.

La terza parte contiene esercizi sulla normalizzazione in cui viene generalmente richiesto, dato uno schema relazionale e un insieme di dipendenze funzionale (esplicite oppure descritte a parole), di determinare se lo schema è in 2NF, 3NF e BCNF e di discutere eventuali decomposizioni.

6.4 SQL e linguaggi di programmazione

- ◇ È possibile eseguire enunciati SQL da un programma scritto in un linguaggio di programmazione quale COBOL, PL/1, RPG, C, ... ed avere una interazione tra variabili di programma e oggetti SQL.
- ◇ Per inserire statement SQL all'interno di un programma scritto in linguaggio ospite, la sintassi generica è: **exec sql** SQLstatement ;

Esempi:

```
exec sql  select  *
          from    STUDENTE
          where   MATRICOLA = '1234';

exec sql  update  STUDENTE
          set     NOME = 'Mario'
          where   MATRICOLA = '1234';
```

PROBLEMATICHE:

1. SQL opera in modo *orientato agli insiemi*, mentre i linguaggi di programmazione imperativi operano in modo *orientato al record*
 - ◇ Soluzione: si introduce la nozione di *cursore*:
 - si associa un cursore ad una query di selezione
 - si apre il cursore come se si trattasse di un file sequenziale
 - si acquisisce un record alla volta dal cursore, con la possibilità di consultare i valori dei campi, aggiornare, cancellare
 - si chiude il cursore
2. È necessario poter usare variabili di programma per la composizione di statement SQL e poter copiare valori di campi in variabili di programma
 - ◇ Soluzione: ogni implementazione di embedded SQL è dotata di adeguati strumenti sintattici.
3. È necessario disporre di strumenti per comunicare al programma lo stato delle esecuzioni SQL.
 - ◇ Soluzione: c'è un parametro SQLCODE che restituisce al programma ospite un valore che indica l'esito dell'istruzione SQL eseguita:
 - SQLCODE = 0 → ok
 - SQLCODE < 0 → situazione di errore
 - SQLCODE = 100 → non trovato

Operazioni su singolo record

- ◇ **Selezione:** selezione del livello LIV e della CITTA del fornitore F con codice COD pari a :COD

```
exec sql  select  LIV, CITTA
          into    :LIVELLO, :CITTA
          from    F
          where   COD = :COD ;
```

- ◇ **Aggiornamento:** aumento del livello LIV di una quantità pari a VALORE di tutti i fornitori F di 'Modena'

```
exec sql  update  F
          set      LIV = LIV + :VALORE
          where   CITTA = 'Modena' ;
```

- ◇ **Cancellazione:** rimozione degli ordini O eseguiti dai fornitori F nella propria CITTA

```
exec sql delete from O
          where CITTA = (select CITTA
                        from F
                        where F.COD = O.COD) ;
```

- ◇ **Inserimento:** inserimento di un articolo A con codice :A.COD, nome :NOME e peso :PESO

```
exec sql  insert
          into    A(COD, NOME, PESO)
          values  (: A.COD, : NOME, : PESO) ;
```

Operazioni con Cursore

- ◇ Un cursore è una variabile associata ad uno statement SELECT che assume come possibili valori le tuple risultanti dall'esecuzione dello statement stesso.
- ◇ Ogni cursore è dichiarato ed associato ad uno specifico statement SELECT:
DECLARE <cursor_name>
CURSOR FOR <select_clause>;
- ◇ La query *select_clause* non è eseguita all'atto della dichiarazione, ma quando il cursore viene aperto.
OPEN <cursor_name>;
- ◇ Quando è aperto, il cursore identifica una sequenza ordinata di righe, e una specifica posizione all'interno dell'ordinamento:
sopra una specifica riga
prima di una specifica riga
dopo l'ultima riga .
L'esecuzione dello statement OPEN posiziona il cursore prima della prima tupla.
- ◇ L'istruzione
FETCH <cursor_name> INTO <target_list>;
genera l'avanzamento del cursore alla riga successiva e la copia dei valori della tupla corrente nelle variabili della <target_list> corrispondenti alle colonne indicate nella <select_clause>.
- ◇ Dopo l'esecuzione di n FETCH, pari alla cardinalità del risultato, il cursore è posizionato dopo l'ultima tupla e SQLCODE assume il valore 100.
- ◇ Quando tutte le righe sono state esaminate possiamo disattivare il cursore con lo statement
CLOSE <cursor_name>;

Esempio di utilizzo del Cursore

Calcolare la media voto di partenza per la tesi di laurea dello studente avente matricola '1234' (media calcolata su tutti esami scartando i due voti peggiori).

Disponendo della tabella ESAME, selezioniamo gli esami (con i relativi voti) sostenuti dallo studente in questione.

```
select  COD_CORSO, VOTO
from    ESAME
where   MATRICOLA = '1234'
```

Volendo trasferire il risultato, una tupla alla volta, alle variabili di programma CORSO e VOTO si introduce un cursore, Cur_Esame, che permette di scandire le tuple secondo l'ordine con cui vengono prodotte in fase di esecuzione. Operando algebricamente sui VOTI si può calcolare la media.

Cur_Esame →	<table border="1"> <thead> <tr> <th>ESAME.CORSO</th><th>ESAME.VOTO</th></tr> </thead> <tbody> <tr> <td>Analisi</td><td>28</td></tr> <tr> <td>Fisica I</td><td>26</td></tr> <tr> <td>Chimica</td><td>30</td></tr> <tr> <td>...</td><td></td></tr> </tbody> </table>	ESAME.CORSO	ESAME.VOTO	Analisi	28	Fisica I	26	Chimica	30	...	
ESAME.CORSO	ESAME.VOTO										
Analisi	28										
Fisica I	26										
Chimica	30										
...											

```
exec  sql declare Cur_Esame cursor for
      select COD_CORSO, VOTO
      from ESAME
      where MATRICOLA = '1234'
      order by VOTO DESC;
...
for   (i=1;i<28;i++) {
      exec sql fetch Cur_Esame into :CORSO,:VOTO;
      somma_voto += VOTO;
    }
      media_voto = somma_voto / 27 * 11 / 3;
...

```

Dichiarazioni di variabili

VARIABILI: Gli statement Embedded SQL utilizzano variabili C per trasferire dati dal database verso il programma (e viceversa).

exec sql begin declare section;

Dichiarazione di variabili e tipi in linguaggio C

exec sql end declare section;

- ◇ La definizione della variabili Embedded SQL può essere sia globale (esterna alle funzioni e procedure), sia locale, al pari delle variabili C. Le variabili utilizzate in statement embedded vengono definite in una sezione. I tipi di dati accettati dal preprocessore SQL sono i medesimi di quelli presenti nel linguaggio C (per ciascun DBMS esiste una tabella di compatibilità di tipi data tra Embedded SQL e C).

ETICHETTE: È possibile definire *etichette* referenziabili, aventi le seguenti caratteristiche: una label inizia con una lettera (o underscore '_'), è la prima parola di una riga, termina con i due punti ':'.
Esempio: close_cursor: **exec sql close cursor1;**

- ◇ Una label non può precedere una dichiarazione ed in generale è bene utilizzare le label solo in statement di comandi di esecuzione.

DCLGEN: L'utilità DCLGEN (Declaration Generator) permette di generare automaticamente una struttura C a partire da una tabella di database.

dclgen language dbname tablename filename structname;

dove language è l'host language (ad esempio, il C), dbname è il nome del database contenente la tabella, tablename è il nome della tabella da tradurre, structname è il nome della struttura generata. Il tag della struttura è dato da structname seguito da underscore.

Esempio: Sia studente una tabella del database Università; il comando dclgen c Università studente st.dcl st_rec; genera il file 'st.dcl' con la dichiarazione della tabella e della struttura C corrispondente:

```
EXEC SQL DECLARE studente TABLE (MATRICOLA    integer not null,
                                     ... );
struct st_rec - long    MATRICOLA;
                                     ... " st_rec;
```

Nella dichiarazione delle variabili si userà quindi include per includere il file:

exec sql begin declare section;

exec sql include 'st.dcl';

exec sql end declare section;

Ora è possibile utilizzare la struttura st_rec come variabile nei comandi embedded SQL.

SQL Communications Area

SQLCA è la struttura dati predefinita per la gestione degli errori generati in ambiente SQL. L'istruzione **exec sql include sqlca;** specifica al preprocessore di includere la variabile **sqlca** che contiene le informazioni relative all'ultima istruzione SQL eseguita. **sqlca** contiene (tra gli altri) i campi:

sqlcode : specifica lo stato di ritorno di una istruzione SQL:

0 per esecuzione corretta, < 0 per situazione di errore e 100 risultato vuoto.

sqlwarn : struttura di 8 campi (sqlwarn0 ... sqlwarn7) con le warning relative alle ultime istruzioni eseguite;

sqlerrd[6] : il terzo campo [2] indica il numero di tuple coinvolte durante l'ultima esecuzione.

Error Handling con sqlca : **exec sql whenever** condition action;

Condition può assumere uno dei seguenti tre valori:

sqlwarning: indica che l'ultima istruzione SQL Embedded eseguita ha prodotto una warning. La variabile sqlwarn0 di SQLCA vale W. È quindi possibile specificare azioni condizionate a warning del DBMS.

sqlerror: indica che l'ultima istruzione SQL Embedded eseguita ha prodotto un errore. La variabile sqlcode di SQLCA ha valore negativo.

not found: indica che l'istruzione di select, update, insert, delete, ... non ha avuto effetto su alcuna riga. La variabile sqlcode di SQLCA è posta a 100.

Action può assumere uno dei seguenti quattro valori:

continue: continua l'esecuzione a partire dalla prossima istruzione. Nel caso di fatal error viene visualizzato un messaggio di errore e il programma termina.

stop: viene visualizzato un messaggio di errore e il programma termina. Nel caso in cui il database sia connesso quando la condizione è raggiunta la terminazione non esegue il commit delle operazioni compiute. L'azione di stop non può essere eseguita per la condizione di not found.

goto label: trasferisce il controllo del programma alla label specificata.

call procedure: chiama la procedura specificata. Nessun argomento può essere passato nella chiamata. Ad esempio, call sqlprint chiama una procedura che visualizza un messaggio riguard

◇ Ciascuna action ha effetto sulle istruzioni Embedded SQL fino alla successiva **whenever**

Compilare un programma Embedded C/SQL unix

I passi necessari in INGRES per eseguire la compilazione sono i seguenti.

- ◇ Eseguire la precompilazione da embedded SQL a C:

Il preprocessore è chiamato **esqlc**

esqlc <filename>

Preprocessa <filename>.sc e genera <filename>.c

- ◇ Compilare il codice C ottenuto

cc -c <filename>.c

- ◇ Linkare i codici oggetto con le librerie del DBMS

cc -o <filename> <filename>.o

\$ILSYSTEM/ingres/lib/libingres.a

-lm -lc

Esempio Completo

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <stdio.h>
```

```
/* Inclusionione dell'area di comunicazione */
```

```
exec sql include sqlca;
```

```
main()—
```

```
/* Calcolo della media di laurea degli studenti di Ingegneria */
```

```
int code'esami;
```

```
int somma'voto, cont;
```

```
/* La section e' l'area di definizione delle variabili utilizzate dal DB */
```

```
exec sql begin declare section;
```

```
int n'esami;
```

```
float media'laurea;
```

```
long matricola'attuale;
```

```
exec sql include 'esame.dcl';
```

```
exec sql include 'studente.dcl';
```

```
exec sql end declare section;
```

```
/* in caso di errore termina */
```

```
exec sql whenever sqlerror stop;
```

Compilare un programma Embedded C/SQL unix (1)

```

exec sql connect universita; /* Connessione al Database */

exec sql declare c'esami cursor for /* Dichiarazione del cursore */
select COD'CORSO, MATRICOLA, VOTO
from esame
order by MATRICOLA, VOTO DESC;

exec sql open c'esami; /* Apertura del cursore */

/* Caricamento del cursore nelle variabili */
exec sql fetch c'esami
into :es'rec.COD'CORSO,
    :es'rec.MATRICOLA,
    :es'rec.VOTO;
code'c'esami = sqlca.sqlcode;
/* scansione sequenzialmente il cursore */
while (code'c'esami == 0) -
    somma'voto = 0;
/* conteggio esami superati da uno studente */
exec sql select count(*) into :n'esami
    from esame
    where MATRICOLA = :es'rec.MATRICOLA;
matricola'attuale = es'rec.MATRICOLA;
cont = 1;
/* totale esami scartando i due peggiori */
while (es'rec.MATRICOLA == matricola'attuale
    && code'c'esami == 0 ) -

/* sommatoria degli N-2 esami sostenuti */
if (cont != (n'esami-2)) -
    cont++;
    somma'voto += es'rec.VOTO;
"

/* caricamento esame successivo */
exec sql fetch c'esami
into :es'rec.COD'CORSO,
    :es'rec.MATRICOLA,
    :es'rec.VOTO;
code'c'esami = sqlca.sqlcode; "
```


SQL SERVER e la programmazione C

È possibile eseguire istruzioni di SQL Server da un programma C utilizzando il linguaggio Embedded SQL.

Configurazione:

1. Connessione a SQL Server;
2. Memorizzazione delle istruzioni SQL in un buffer ed invio a SQL Server;
3. Utilizzo del risultato: è possibile processare una riga per volta (utilizzo di cursori), memorizzare i dati in variabili di programma;
4. Gestione gli errori utilizzando le opportune librerie;
5. Sconnessione dal Server.

Esempio Completo con uso di cursori

◇ Il programma manipola i dati di un database relativo ad una libreria: viene applicato uno sconto del 20% ai negozi con vendite maggiori di 50 e del 10% a quelli comprese tra 10 e 50.

```
DECLARE @tid    char(6),    -- Title ID.
        @dtype  varchar(40), -- Discount Type.
        @sid    char(4),    -- Store ID.
        @totsold int        -- Total Sold.
```

```
SELECT @tid = 'PS2091'
SELECT @dtype = (@tid + ' Book Discount')
```

```
USE pubs
```

```
DECLARE SalesDiscnt CURSOR
FOR
  SELECT st.stor`id,
         SUM(qty) AS 'Total Units Sold'
  FROM stores st INNER JOIN sales s
    ON st.stor`id = s.stor`id
    INNER JOIN titles t
    ON t.title`id = s.title`id
```

```
WHERE ord`date BETWEEN '1/1/1998'
      AND '12/31/1998'
      AND t.title`id = @tid
GROUP BY st.stor`id
```

```
OPEN SalesDiscnt
```

```
FETCH NEXT FROM SalesDiscnt INTO @sid, @totsold
WHILE (@@FETCH`STATUS < -1)
BEGIN
  IF @totsold < 50
  BEGIN
    INSERT discounts (discounttype, stor`id,
                    discount)
    VALUES (@dtype, @sid, 20)
  END
  ELSE IF @totsold <= 20 AND @totsold <= 50
  BEGIN
    INSERT discounts (discounttype, stor`id,
                    discount)
    VALUES (@dtype, @sid, 10)
  END
  FETCH NEXT FROM SalesDiscnt
  INTO @sid, @totsold
END
```

```
DEALLOCATE SalesDiscnt
```