



UNIVERSITÀ DEGLI STUDI DI PALERMO

Corso di Basi di Dati e Sistemi Informativi

Prof. R. Sorbello

Anno Accademico 2016/2017

Corso di laurea in Ingegneria informatica e delle telecomunicazioni

Quanderno degli esercizi

Barbera Antonella

Lucchese Rachele

Pittari Maria

Argomenti: Algebra Relazionale

MySQL

Modelli E-R

Calcolo computazionale

Normalizzazione

Palermo, 19/06/2017

Indice

ALGEBRA RELAZIONALE e SQL	
Esercitazione 1 – Cinema e Fiumiciattoli	pag. 1
Esercitazione 2 – Biblioteca	pag. 7
Esercitazione 3 – Listino	pag. 9
 MYSQL	
Esercitazione 4 – Rifornimenti	pag. 13
Esercitazione 5 – Fabbrica	pag. 19
Esercitazione 6 – Azienda	pag. 25
Esercitazione 7 – Segreteria	pag. 31
Esercitazione 8 – Corso	pag. 35
Esercitazione 9 – Fornitura	pag. 37
Esercitazione 10 – Scuola	pag. 38
Esercitazione 11 – Ciclismo	pag. 44
Esercitazione 12 – Museo	pag. 50
Esercitazione 13 – Trading	pag. 56
Esercitazione 14 – Stradario	pag. 62
Esercitazione 15 – Tennis	pag. 68
Esercitazione 16 – Eventi	pag. 74
Esercitazione 17 – Golf	pag. 79
Esercitazione 18 – Musica	pag. 84
Esercitazione 19 – Azienda	pag. 88
Esercitazione 20 – Assistenza	pag. 94
Esercitazione 21 – Meeting	pag. 100
 MODELLI ENTITÀ RELAZIONE	
Modello E-R 1	pag. 105
Modello E-R 2	pag. 106
Modello E-R 3	pag. 107
Modello E-R 4	pag. 109
Modello E-R 5	pag. 111
Modello E-R 6	pag. 113
Modello E-R 7	pag. 114
Modello E-R 8	pag. 116
Modello E-R 9	pag. 117
Modello E-R 10	pag. 118
Modello E-R 11	pag. 119
Modello E-R 12	pag. 120
Modello E-R 13	pag. 121
Modello E-R 14	pag. 122
Modello E-R 15	pag. 123
Modello E-R 16	pag. 124
Modello E-R 17	pag. 125
Modello E-R 18	pag. 126
Modello E-R 19	pag. 127
Modello E-R 20	pag. 128
 CALCOLO COMPUTAZIONALE	
Calcolo computazionale 1	pag. 129
Calcolo computazionale 2	pag. 130

Calcolo computazionale 3	pag. 131
Calcolo computazionale 4	pag. 132
Calcolo computazionale 5	pag. 133
Calcolo computazionale 6	pag. 134
NORMALIZZAZIONE	
Normalizzazione 1	pag. 135
Normalizzazione 2	pag. 136
Normalizzazione 3	pag. 137
Normalizzazione 4	pag. 138
Normalizzazione 5	pag. 139
Normalizzazione 6	pag. 140
Normalizzazione 7	pag. 141
Normalizzazione 8	pag. 142
Normalizzazione 9	pag. 143

ESERCITAZIONE 1

(31/03/2017)

CINEMA

Considerare lo schema di base di dati contenente le relazioni:

FILM (Cod_F, Titolo, Regista, Anno, Costo_Noleggio)
ATTORE (Cod_A, Cognome, Nome, Sesso, Data_Nascita, Nazionalità)
INTERPRETAZIONE (Ref_Film, Ref_Attore, Personaggio)

FK: Ref_Film REFERENCES Film (Cod_F)
FK: Ref_Attore REFERENCES Attore (Cod_A)

Un Attore (Cod_A, Cognome) ha interpretato un determinato Personaggio in un certo Film (Cod_F, Titolo).

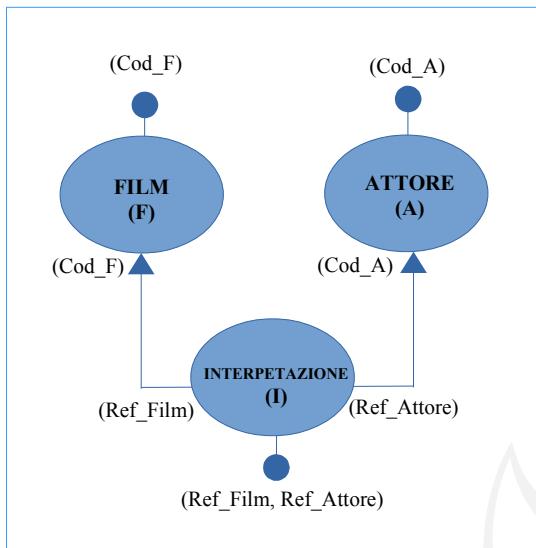


Fig. 1.1 – Modello dello schema di base di dati

1) Effettuare le seguenti interrogazioni, mostrando le istanze ottenute:

- 1.1** Mostrare una base di dati su questo schema per la quale i join fra le varie relazioni siano tutti completi.
- 1.2** Supponendo che esistano due vincoli di integrità referenziale fra la relazione INTERPRETAZIONI e le altre due, discutere i possibili casi di join non completo.
- 1.3** Mostrare un prodotto cartesiano che coinvolga relazioni in questa base di dati.
- 1.4** Mostrare una base di dati per la quale uno (o più) dei join sia vuoto.

2) Con riferimento alla figura 1.1, formulare in algebra relazionale le interrogazioni che trovano:

- 2.1** I titoli dei film prodotti nel 2003
- 2.2** Nome e cognome degli attori di sesso maschile nati dopo il 1/1/1970
- 2.3** Nome e cognome degli attori che hanno interpretato il personaggio Tarzan
- 2.4** I titoli dei film nei quali Henry Fonda sia stato interprete.
- 2.5** Nome e cognome di tutti gli attori che hanno partecipato al film “Il Signore degli anelli”.
- 2.6** Titolo e anno di produzione di tutti i film a cui ha partecipato almeno un attore di nazionalità italiana
- 2.7** Titolo e anno di produzione di tutti i film a cui hanno partecipato almeno due attori di nazionalità italiana

1.1 Mostrare una base di dati su questo schema per la quale i join fra le varie relazioni siano tutti completi.

F

Cod_F	Titolo	Regista	Anno	Costo_Noleggio
001	Titanic	James Cameron	1997	€4,50
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50
003	Guerra e Pace	King Vidor	1956	€4,00

A

Cod_A	Cognome	Nome	Sesso	Data_Nascita	Nazionalità
A1	Fonda	Henry	M	16/05/1905	Statunitense
A2	Bloom	Orlando	M	13/01/1977	Britannica
A3	Di Caprio	Leonardo	M	11/11/1974	Statunitense
A4	Winslet	Kate	F	05/10/1975	Britannica

I

Ref_Film	Ref_Attore	Personaggio
001	A3	Jack Dawson
001	A4	Rose DeWitt Bukater
002	A2	Legolas
003	A1	Pierre Bezukhov

In questa base di dati tutti i JOIN fra le relazioni danno luogo a una tabella che contiene tutte le tuple delle relazioni coinvolte. Quindi tutti i JOIN possibili sono completi.

1.2 Supponendo che esistano due vincoli di integrità referenziale fra la relazione INTERPRETAZIONI e le altre due, discutere i possibili casi di join non completo.

F

Cod_F	Titolo	Regista	Anno	Costo_Noleggio
001	Titanic	James Cameron	1997	€4,50
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50

A

Cod_A	Cognome	Nome	Sesso	Data_Nascita	Nazionalità
A1	Fonda	Henry	M	16/05/1905	Statunitense
A2	Bloom	Orlando	M	13/01/1977	Britannica
A3	Di Caprio	Leonardo	M	11/11/1974	Statunitense
A4	Winslet	Kate	F	05/10/1975	Britannica

I

Ref_Film	Ref_Attore	Personaggio
001	A3	Jack Dawson
001	A4	Rose DeWitt Bukater
002	A2	Legolas
003	A1	Pierre Bezukhov

In questa base di dati i JOIN non sono più completi, perché nelle tabelle risultanti non sono presenti tutte le tuple delle relazioni coinvolte. Ad esempio, effettuando il seguente JOIN , poiché nella relazione F non è più presente il Cod_F 003 , non viene più coinvolta la quarta tupla della relazione I.

PROJ_{Cod_F, Titolo, Ref_Attore (F JOIN_{F.Cod_F=I.Ref_Film} I)}

Cod_F	Titolo	Ref_Attore
001	Titanic	A3
001	Titanic	A4
002	Il Signore degli Anelli	A2

1.3 Mostrare un prodotto cartesiano che coinvolga relazioni in questa base di dati.

F

<u>Cod_F</u>	<u>Titolo</u>	<u>Regista</u>	<u>Anno</u>	<u>Costo_Noleggio</u>
001	Titanic	James Cameron	1997	€4,50
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50

A

Cod_A	Cognome	Nome	Sesso	Data_Nascita	Nazionalità
A1	Bloom	Orlando	M	13/01/1977	Britannica
A2	Di Caprio	Leonardo	M	11/11/1974	Statunitense

I

<u>Ref_Film</u>	<u>Ref_Attore</u>	Personaggio
001	A2	Jack Dawson
002	A1	Legolas

Effettuando il prodotto cartesiano fra le relazioni F ed A ottengo una tabella le cui righe sono tutte le possibili combinazioni ordinate delle tuple iniziali.

FXA

Cod_F	Titolo	Regista	Anno	Costo_Noleggio	Cod_A	Cognome	Nome	Sesso	Data_Nascita	Nazionalità
001	Titanic	James Cameron	1997	€4,50	A1	Bloom	Orlando	M	13/01/1977	Britannica
001	Titanic	James Cameron	1997	€4,50	A2	Di Caprio	Leonardo	M	11/11/1974	Statunitense
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50	A1	Bloom	Orlando	M	13/01/1977	Britannica
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50	A2	Di Caprio	Leonardo	M	11/11/1974	Statunitense

1.4 Mostrare una base di dati per la quale uno (o più) dei join sia vuoto.

F

Codice	Titolo	Regista	Anno	Costo_Noleggio
001	Titanic	James Cameron	1997	€4,50
002	Il Signore degli Anelli	Peter Jackson	2001	€4,50

A

Codice	Cognome	Nome	Sesso	Data_Nascita	Nazionalità
A1	Bloom	Orlando	M	13/01/1977	Britannica
A2	Di Caprio	Leonardo	M	11/11/1974	Statunitense

Effettuando il JOIN fra le relazioni F ed A otteniamo un JOIN vuoto perché gli attributi **Codice** delle due relazioni non hanno dati in comune.

F JOIN A

2.1 Trovare i titoli dei film prodotti nel 2003

INFORMAZIONI: anno e titolo si trovano nella relazione FILM(F)

OPERAZIONI: una selezione e una proiezione

LOGICA: seleziono tutti i film del 2003 e ne proietto i titoli

PROJ Titolo (**SEL** Anno=2003' F)

2.2 Trovare nome e cognome degli attori di sesso maschile nati dopo il 1/1/1970

INFORMAZIONI: Nome ,Cognome , sesso e data di nascita sono attributi della relazione Attore (A)

OPERAZIONI: una selezione e una proiezione

LOGICA: seleziono tutti gli attori di sesso maschile, nati dopo 01/01/1970 e ne proietto Nome e Cognome

PROJ Cognome, Nome (**SEL** Sesso='M', Data_Nascita>'01/01/1970' A)

2.3 Nome e cognome degli attori che hanno interpretato il personaggio Tarzan

INFORMAZIONI: Nome e Cognome sono attributi della relazione Attore (A) , mentre il Personaggio è in Interpretazione

OPERAZIONI: un join, una selezione e una proiezione

LOGICA: con l'equijoin fra Interpretazione e Attore ottengo tutte le informazioni sugli attori e le loro interpretazioni, seleziono quelli che hanno interpretato come personaggio Tarzan e ne proietto Nome e Cognome

PROJ Cognome, Nome (**SEL** I.Personaggio ='Tarzan' (**I JOIN** I.Ref_Attore=A.Cod_A A))

2.4 I titoli dei film nei quali Henry Fonda sia stato interprete.

INFORMAZIONI: Nome e Cognome sono attributi della relazione Attore (A) , mentre Titolo è in Film(F), per collegare le due relazioni master mi serve anche lo slave Interpretazione(I)

OPERAZIONI: 2 join, una selezione e una proiezione

LOGICA: con l'equijoin fra Interpretazione e Attore ottengo tutte le informazioni sugli attori e le loro interpretazioni, con l'altro join con Film ottengo anche le informazioni sui film interpretati. Seleziono fra le tuple quelle che hanno Cognome="Fonda" e Nome="Henry" e proietto i Titoli.

PROJ Titolo (**SEL** Cognome='Fonda', Nome='Henry' (**F JOIN** F.Cod_F=Ref_Film
(**I JOIN** I.Ref_Attore=A.Cod_A A)))

2.5 Nome e cognome di tutti gli attori che hanno partecipato al film "Il Signore degli anelli".

INFORMAZIONI: Nome e Cognome sono attributi della relazione Attore (A) , mentre Titolo è in Film(F), per collegare le due relazioni master mi serve anche lo slave Interpretazione(I)

OPERAZIONI: 2 join, una selezione e una proiezione

LOGICA: con l'equijoin fra Interpretazione e Attore ottengo tutte le informazioni sugli attori e le loro interpretazioni, con l'altro join con Film ottengo anche le informazioni sui film interpretati. Seleziono fra le tuple quelle che hanno il Titolo= "Il Signore degli Anelli" e proietto Nome e Cognome

PROJ Cognome,Nome (**SEL** Titolo='Il Signore degli Anelli' (**F JOIN** F.Cod_F=Ref_Film (**I JOIN** I.Ref_Attore=A.Cod_A A)))

2.6 Titolo e anno di produzione di tutti i film a cui ha partecipato almeno un attore di nazionalità italiana

INFORMAZIONI: Nazionalità è attributo della relazione Attore (A) , mentre Titolo e Anno si trovano in Film(F), per collegare le due relazioni master mi serve anche lo slave Interpretazione(I)

OPERAZIONI: 2 join, una selezione e una proiezione

LOGICA: con l'equijoin fra Interpretazione e Attore ottengo tutte le informazioni sugli attori e le loro interpretazioni, con l'altro join con Film ottengo anche le informazioni sui film interpretati. Seleziono fra le tuple quelle che hanno Nazionalità= "Italiana" e proietto Titolo e Anno.

PROJ Titolo,Anno (**SEL** Nazionalità='Italiana' (**F JOIN** F.Cod_F=Ref_Film (**I JOIN** I.Ref_Attore=A.Cod_A A)))

2.7 Titolo e anno di produzione di tutti i film a cui hanno partecipato almeno due attori di nazionalità italiana

INFORMAZIONI: Nazionalità è attributo della relazione Attore (A) , mentre Titolo e Anno si trovano in Film(F), per collegare le due relazioni master mi serve anche lo slave Interpretazione(I)

OPERAZIONI: 1 ridenominazione, 4 join, una selezione e una proiezione

LOGICA: con i join fra Attore, Interpretazione e Film ottengo tutte le informazioni e, ridenominando le relazioni, effettuo di nuovo i join sulle stesse relazioni. Col join sugli attributi CodF e Nazionalità fra le due relazioni risultanti ottengo tutte le coppie di attori per lo stesso film, comprese le tuple identità e le duali. Seleziono le tuple degli attori di nazionalità italiana ed elimino le tuple senza informazione imponendo che gli attori siano diversi, e infine proietto titolo e anno.

PROJ Titolo,Anno (**SEL** Nazionalità='Italiana' AND CodA < Cod_A_R (A **JOIN** A.Cod_A=Ref_Attore (F **JOIN** F.Cod_F=I.Ref_Film I)) **JOIN** Cod_F=Ref_Film AND Nazionalità=Nazionalità_R
(I **JOIN** I.Ref_Attore=Cod_A_R (**REN** Cod_A_R, Cognome_R, Nome_R, Sesso_R, Data_Nascita_R,Nazionalità_R-- A.Cod_A, A.Cognome, A.Nome, A.Sesso, A.Data_Nascita, A.Nazionalità_A)))

FIUMICIASTOLI

Con riferimento al seguente schema di base di dati:

CITTÀ (Nome_C, Regione, Abitanti)

FIUME (Nome_F, Lunghezza)

ATTRAVERSAMENTO (Ref_Città, Ref_Fiume)

FK: Ref_Città REFERENCES Città(Nome_C)

FK: Ref_Fiume REFERENCES Fiume(Nome_F)

Una Città (Nome_C) può essere attraversata da un Fiume (Nome_F), di una certa Lunghezza.

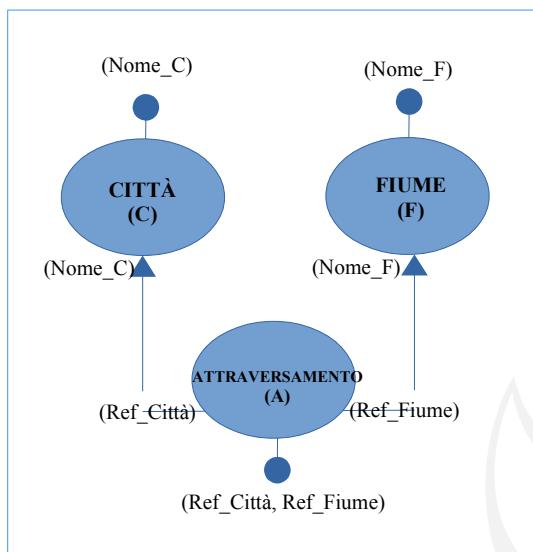


Fig. 1.2 – Modello dello schema di base di dati

Formulare in algebra relazionale le seguenti interrogazioni:

1. Visualizzare nome, regione e abitanti per le città che hanno più di 50000 abitanti e sono attraversate dal Po o dall'Adige.
2. Trovare le città che sono attraversate da almeno due fiumi, visualizzando il nome della città e quello del più lungo di tali fiumi.

SOLUZIONI

1. Visualizzare nome, regione e abitanti per le città che hanno più di 50000 abitanti e sono attraversate dal Po o dall'Adige.

INFORMAZIONI: Nome, regione e abitanti sono attributo della relazione Città (C) , mentre il nome del fiume si trova in Attraversamento (A)

OPERAZIONI: un join, una selezione e una proiezione

LOGICA: con il join ottengo una tabella con gli attraversamenti e le informazioni sulle città, seleziono quelle con più di 50.000 abitanti e che sono attraversate o dal Po o dall'Adige, ne proietto poi il Nome, la Regione e il numero di abitanti.

PROJ Nome_C,Regione,Abitanti (**SEL** Abitanti>50.000 AND (Nome_F='Po' OR Nome_F='Adige')) (**C JOIN** C.Nome_C= A.Ref_Città A))

2. Trovare le città che sono attraversate da almeno due fiumi, visualizzando il nome della città e quello del più lungo di tali fiumi.

INFORMAZIONI: Lunghezza è attributo della relazione Fiume (F), mentre le informazioni sulle città attraversate le ottengo dalla tabella Attraversamenti (A)

OPERAZIONI: 3 join, 4 ridenominazioni, una selezione e una proiezione

LOGICA: con il join fra Attraversamento e Fiume ottengo una tabella con gli attraversamenti e le informazioni sui fiumi e, ridenominando le relazioni, effettuo di nuovo il join sulle stesse relazioni. Col join sull'attributo Ref_Città fra le due relazioni risultanti ottengo tutte le coppie di fiumi per la stessa città, comprese le tuple identità e le duali. Seleziono le tuple che abbiano il primo fume più lungo ed elimino così le tuple senza informazione, ed infine proietto la città e il fiume.

```
PROJ Ref_Città1, Nome_F1 (
  SEL Lunghezza1>Lunghezza2 (
    (
      (REN Ref_Città1← Ref_Città, Ref_Fiume1← Ref_Fiume A) JOIN Ref_Fiume1=Nome_F1
      (REN Nome_F1← Nome_F, Lunghezza1← Lunghezza F)
    )
    JOIN Ref_Città=Ref_Città2
    (
      (REN Ref_Città2← Ref_Città, Ref_Fiume2← Ref_Fiume A) JOIN Ref_Fiume2=Nome_F2
      (REN Nome_F2← Nome_F, Lunghezza2← Lunghezza F)
    )
  )
))
```

ESERCITAZIONE 2 - BIBLIOTECA

07/04/2017

Dato il seguente schema per la base di dati BIBLIOTECA:

LIBRERIA (Partita_IVA, Nome_L, Indirizzo, Città)
LIBRO (Codice_L, Titolo, Editore, Pagine)
AUTORE (Codice_A, Nome_A, Cognome, Nazionalità, Sesso)
VENDITA (Ref_Libreria, Ref_Libro, Copie)
 FK: Ref_Libreria REFERENCES Libreria (Partita_IVA)
 FK: Ref_Libro REFERENCES Libro (Codice_L)
SCRITTO (Ref_L, Ref_Autore)
 FK: Ref_L REFERENCES Libro (Codice_L)
 FK: Ref_Autore REFERENCES Autore (Codice_A)

Una Libreria (Partita_IVA) vende un Libro (Codice_L) in un determinato numero di Copie.
 Un Libro è scritto da un Autore (Codice_A) (Fig. 2.1).

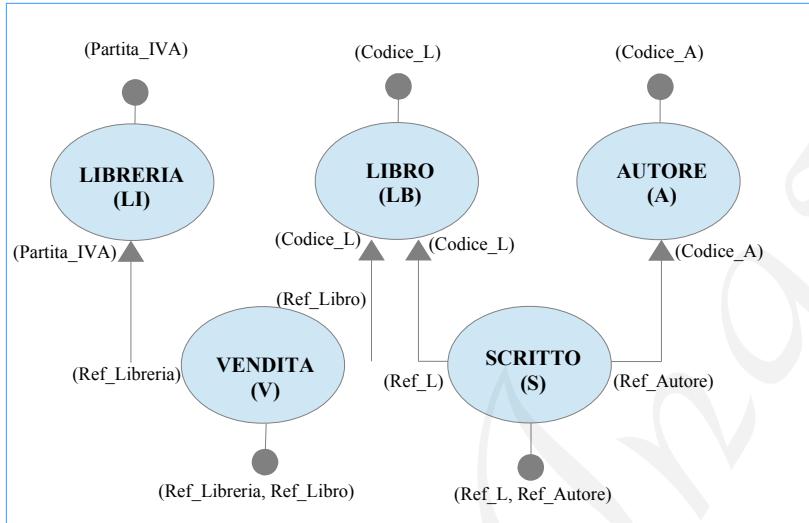


Fig. 2.1 – Modello dello schema di base di dati

Considerando lo schema in fig. 2.1, formulare in algebra relazionale le seguenti interrogazioni:

1. Trovare le librerie (nome e indirizzo) che vendono libri scritti da autori italiani
2. Trovare le librerie di Roma che vendono libri di Umberto Eco
3. Trovare le librerie che hanno venduto più di 10 copie di un libro di autori tedeschi
4. Trovare i libri (titolo ed editore) che non sono in vendita nella libreria Feltrinelli
5. Trovare le librerie che vendono tutti i libri di Piero Angela

1. Trovare le librerie (nome e indirizzo) che vendono libri scritti da autori italiani

PROJ Nome_L, Indirizzo (**SEL** Nazionalità = 'Italiana' $\left(\left(\left(\text{LI} \text{ JOIN } \text{LI.Partita_IVA} = \text{V.Ref_Libreria} \text{ V} \right) \text{ JOIN } \text{Ref_Libro} = \text{LB.Codice_L} \text{ LB} \right) \text{ JOIN } \text{Codice_L} = \text{S.Ref_L} \text{ S} \right) \text{ JOIN } \text{Ref_Autore} = \text{A.Codice_A} \text{ A} \right)$)

2. Trovare le librerie di Roma che vendono libri di Umberto Eco

PROJ Partita_IVA, Nome_L, Indirizzo (**SEL** (Città = 'Roma') AND (Cognome = 'Eco') $\left(\left(\left(\text{LI} \text{ JOIN } \text{LI.Partita_IVA} = \text{V.Ref_Libreria} \text{ V} \right) \text{ JOIN } \text{Ref_Libro} = \text{LB.Codice_L} \text{ LB} \right) \text{ JOIN } \text{Codice_L} = \text{S.Ref_L} \text{ S} \right) \text{ JOIN } \text{Ref_Autore} = \text{A.Codice_A} \text{ A} \right)$)

3. Trovare le librerie che hanno venduto più di 10 copie di un libro di autori tedeschi

PROJ Partita_IVA, Nome_L, Indirizzo (**SEL** (Copie > 10) AND (Nazionalità = 'Tedesca') ((((**LI JOIN** LI.Partita_IVA = V.Ref_Libreria V) **JOIN** Ref_Libro = LB.Codice_L LB) **JOIN** Codice_L = S.Ref_L S) **JOIN** Ref_Autore = A.Codice_A A))

4. Trovare i libri (titolo ed editore) che non sono in vendita nella libreria Feltrinelli

PROJ Titolo, Editore ((**LI JOIN** LI.Partita_IVA = V.Ref_Libreria V) **JOIN** Ref_Libro = LB.Codice_L LB) - (**SEL** Nome_L = 'Feltrinelli' (**LI JOIN** LI.Partita_IVA = V.Ref_Libreria V) **JOIN** Ref_Libro = LB.Codice_L LB)))

5. Trovare le librerie che vendono tutti i libri di Piero Angela

PROJ Partita_IVA, Nome_L, Indirizzo (**LI JOIN** LI.Partita_IVA = Ref_Libreria (V \ominus)

(**PROJ** Ref_Libro (**REN** Ref_Libro \leftarrow Codice_L (**SEL** (Nome = 'Piero') AND (Cognome = 'Angela') ((**LB JOIN** LB.Codice_L = S.Ref_L S) **JOIN** Ref_Autore = A.Codice_A A))))))

Esplosione della divisione nell'esercizio 5:

PROJ Partita_IVA, Nome_L, Indirizzo (**LI JOIN** LI.Partita_IVA = Ref_Libreria (**PROJ** Ref_Libreria (V) - **PROJ** Ref_Libreria ((**PROJ** Ref_Libreria (V) X (**PROJ** Ref_Libro (**REN** Ref_Libro \leftarrow Codice_L (**SEL** (Nome = 'Piero') AND (Cognome = 'Angela') ((**LB JOIN** LB.Codice_L = S.Ref_L S) **JOIN** Ref_Autore = A.Codice_A A))))))) - V))))

ESERCITAZIONE 3 - LISTINO

05/05/2017

Dato il seguente schema della base di dati Listino :

FORNITORE (Fid, Fnome, Indirizzo)

PEZZO (Pid, Pnome, Colore)

CATALOGO (CodF, CodP, Costo)

FK: CodF REFERENCES Fornitore (Fid)

FK: CodP REFERENCES Pezzo (Pid)

Fornitore (Fid, Fnome) fornisce un determinato Pezzo (Pid, Pnome) ad un certo Costo (Fig 3.1).

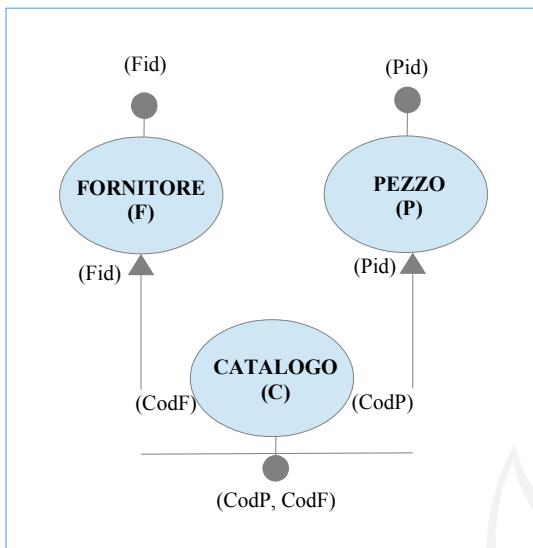


Fig. 3.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1, 2, 3) :

1. Trovare il Pnome dei pezzi per cui esiste un qualche fornitore;
2. Trovare gli Fnome dei fornitori che forniscono ogni pezzo;
3. Trovare gli Fnome dei fornitori che forniscono tutti i pezzi rossi;
4. Trovare i pezzi forniti dall' Acme e da nessun altro;
5. Trovare i Fid dei fornitori che ricaricano su alcuni pezzi più del costo medio di quel pezzo;
6. Per ciascun pezzo trovare gli Fnome dei fornitori che ricaricano di più su quel pezzo;
7. Trovare i Fid dei fornitori che forniscono solo pezzi rossi;
8. Trovare i Fid dei fornitori che forniscono un pezzo rosso e un pezzo verde;
9. Trovare i Fid dei fornitori che forniscono un pezzo rosso, oppure uno verde, oppure entrambi;
10. Trovare i Pid dei pezzi forniti da almeno due fornitori;

1. Trovare il P.nome dei pezzi per cui esiste un qualche fornitore.

SQL

INFORMAZIONI: Pezzo e Catalogo

OPERAZIONI: un Join , una Proiezione

LOGICA: Supponiamo che tutti i pezzi che si trovano nella relazione Catalogo abbiano almeno un fornitore, li ottengo con un'unica interrogazione in cui vi è un join fra Pezzo e Catalogo.

```
SELECT DISTINCT P.Pnome AS Nome_Pezzo  
FROM Pezzo P, Catalogo C  
WHERE P.Pid=C.CodP
```

ALGEBRA

INFORMAZIONI: Pezzo e Catalogo

OPERAZIONI: un Join , una Proiezione

LOGICA: Diversamente da SQL, non posso ridenominare l'attributo proiettato e non ha senso mettere il distinct, in quanto le tuple uguali vengono automaticamente tagliate dalla proiezione.

```
PROJ P.nome ( P JOIN P.Pid = C.CodP C )
```

2. Trovare gli F.nome dei fornitori che forniscono ogni pezzo.

SQL

INFORMAZIONI: Fornitore, Pezzo e Catalogo

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri.

LOGICA: Effettuo la divisione utilizzando il doppio Not exists. Nell'interrogazione C ottengo le tuple per ogni coppia di fornitore e prodotto passati come parametri , con la B seleziono i pezzi per i quali l'interrogazione C da luogo ad una relazione vuota e con la A selezioni i fornitori per i quali l'interrogazione B da luogo ad una relazione vuota.

```
SELECT DISTINCT F.Fnome AS Nome_Fornitore  
A  FROM Fornitore F  
WHERE NOT EXISTS ( SELECT *  
B      FROM Pezzo P  
WHERE NOT EXISTS ( SELECT *  
C          FROM Catalogo C  
WHERE C.CodF=F.Fid AND C.CodP=P.Pid ) )
```

ALGEBRA

INFORMAZIONI: Pezzo e Catalogo

OPERAZIONI: una ridenominazione, due proiezioni, una divisione e un join

LOGICA: A differenza di SQL non posso utilizzare interrogazioni innestate quindi effettuo una divisione fra Catalogo e Pezzo.

```
PROJ F.nome ( F JOIN  
( ( PROJ C.CodF, C.CodP C )  
     $\div$   
( PROJ CodP ( REN CodP  $\leftarrow$  P.Pid P ) ) )  
)
```

3. Trovare gli F.nome dei fornitori che forniscono tutti i pezzi rossi.

SQL

INFORMAZIONI: Fornitore, Pezzo e Catalogo

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri e una selezione

LOGICA: Effettuo la divisione utilizzando il doppio Not exists. Nell'interrogazione C ottengo le tuple per ogni coppia di fornitore e prodotto. Nella B seleziono solo i pezzi di colore rosso. Infine dalla A prendo solo i fornitori per i quali non esista un pezzo di colore rosso che non sia venduto da essi.

```

SELECT DISTINCT F.FNome AS Nome_Fornitore
A   FROM Fornitore F
WHERE NOT EXISTS ( SELECT *
B     FROM Pezzo P
      WHERE P.Colore='Rosso' AND NOT EXISTS ( SELECT *
C       FROM Catalogo C
          WHERE C.CodF=F.Fid AND C.CodP=P.Pid )
)

```

ALGEBRA

INFORMAZIONI: Pezzo e Catalogo

OPERAZIONI: Ridenominazione, proiezioni, selezione , divisione e join

LOGICA: A differenza di SQL non posso utilizzare interrogazioni innestate quindi effettuo una divisione fra Catalogo e Pezzo, selezionando solo le tuple il cui valore sull'attributo Colore = "Rosso".

PROJ_{FNome} (F **JOIN**

((**PROJ** C.CodF, C.CodP C)

⋮

(**PROJ** CodP (**SEL** P.Colore = 'Rosso' (**REN** CodP ← P.Pid P))))

)

4. Trovare i pezzi forniti dall' Acme e da nessun altro

INFORMAZIONI: Fornitore, Pezzo e Catalogo

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri, 3 JOIN e 2 selezioni.

LOGICA: Con l'interrogazione B ottengo tutte le tuple per ogni pezzo passato come parametro che abbia almeno un fornitore diverso da ACME tramite la negazione seleziono solo quei prodotti per i quali l'interrogazione B produca una relazione vuota, ovvero i pezzi forniti solo da ACME.

```

SELECT P.Pid AS Codice_Pezzo, P.PNome AS Nome_Pezzo, P.Colore AS Colore_Pezzo
A   FROM Fornitore F, Catalogo C, Pezzo P
      WHERE P.Pid=C.CodP AND C.CodF=F.Fid AND F.FNome='ACME'
            AND NOT EXISTS ( SELECT *
B               FROM Catalogo C1, Fornitore F1
                  WHERE C1.CodP=P.Pid AND
                      C1.CodF=F1.Fid AND F1.FNome <> 'ACME' )

```

5. Trovare i Fid dei fornitori che ricaricano su alcuni pezzi più del costo medio di quel pezzo.

INFORMAZIONI: Catalogo

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri, operatore aggregato AVG

LOGICA: Con l'interrogazione B ottengo la media dei costi relativi al pezzo passato come parametro, dunque con l'interrogazione A prendo il fornitore che, per quello stesso pezzo , ha nel catalogo costo maggiore del risultato dell'interrogazione B.

```

SELECT DISTINCT C.CodF AS Codice_Fornitore
A   FROM Catalogo C
      WHERE C.Costo > ( SELECT AVG (C1.Costo)
B           FROM Catalogo C1
              WHERE C1.CodP = C.CodP )

```

6. Per ciascun pezzo trovare gli Fname dei fornitori che ricaricano di più su quel pezzo.

INFORMAZIONI: Catalogo, Fornitore

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri, 1 join e l' operatore aggregato MAX

LOGICA: Con l'interrogazione B ottengo il massimo dei costi relativi al pezzo passato come parametro, dunque con l'interrogazione A prendo il fornitore che, per quello stesso pezzo , ha nel catalogo costo maggiore del risultato dell'interrogazione B.

```

SELECT C.CodP AS Codice_Pezzo, F.FNome AS Codice_Fornitore
A   FROM Catalogo C, Fornitore F
      WHERE C.CodF=F.Fid AND C.Costo = ( SELECT MAX (C1.Costo)
B           FROM Catalogo C1
              WHERE C1.CodP = C.CodP )

```

7. Trovare i Fid dei fornitori che forniscono solo pezzi rossi.

INFORMAZIONI: Catalogo, Pezzo

OPERAZIONI: 1 interrogazione innestata , 1 join e 1 selezione

LOGICA: Con l'interrogazione B ottengo i codici dei fornitori che vendono pezzi di colore diverso dal rosso, dunque con l'interrogazione A prendo il codice dei fornitori che non compare fra i risultati dell'interrogazione B.

```
SELECT C.CodF AS Codice_Fornitore  
A  FROM Catalogo C  
WHERE C.CofF NOT IN ( SELECT C1.CodF  
B      FROM Catalogo C1 , Pezzo P  
      WHERE C1.CodP = P.Pid AND P.Colore <> Rosso )
```

8. Trovare i Fid dei fornitori che forniscono un pezzo rosso e un pezzo verde.

INFORMAZIONI: Catalogo, Prodotto

OPERAZIONI: 2 interrogazioni innestate parallele, join , selezioni e operatore insiemistico IN

LOGICA: Con le interrogazione B e C ottengo i codici dei fornitori che forniscono pezzi rossi e pezzi verdi rispettivamente. Infine con l'interrogazione A prendo i fornitori che si trovano sia fra i risultati della B e della C.

```
SELECT F.Fid AS Codice_Fornitore  
A  FROM Fornitore F  
WHERE F.Fid IN ( ( SELECT C.CodF  
B      FROM Catalogo C, Pezzo P  
      WHERE C.CodP = P.Pid AND P.Colore = 'Rosso' )  
      AND ( SELECT C1.CodF  
            FROM Catalogo C1, Pezzo P1  
            WHERE C1.CodP = P1.CodP AND P1.Colore = 'Verde' ) )
```

9. Trovare i Fid dei fornitori che forniscono un pezzo rosso, oppure uno verde, oppure entrambi.

INFORMAZIONI: Catalogo, Pezzo

OPERAZIONI: 1 interrogazione innestata , 1 join , selezioni e operatore insiemistico IN

LOGICA: Con l' interrogazione B ottengo i codici dei fornitori che forniscono pezzi rossi o pezzi verdi . Infine con l'interrogazione A prendo i fornitori i cui codici si trovano sia fra i risultati della B.

```
SELECT F.Fid AS Codice_Fornitore  
A  FROM Catalogo C  
WHERE C.CodF IN ( SELECT C1.CodF  
B      FROM Catalogo C1, Pezzo P  
      WHERE C1.CodP = P.Pid AND ( P.Colore = 'Rosso' OR P.Colore = 'Verde' )  
      )
```

10. Trovare i Pid dei pezzi forniti da almeno due fornitori.

INFORMAZIONI: Catalogo

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri, join e selezione

LOGICA: Con l'interrogazione B ottengo le informazioni dei fornitori che forniscono il prodotto passato come parametro e il cui codice sia diverso dal codice fornitore passato anch'esso come parametro.Dunque con l'interrogazione A prendo i codici dei prodotti tali per cui l'interrogazione B da luogo ad una relazione non vuota.

```
SELECT DISTINCT P.Pid AS Codice_Pezzo  
A  FROM Catalogo C, Pezzo P  
WHERE C.CodP = P.Pid AND EXISTS  
      ( SELECT *  
        B      FROM Catalogo C1, Pezzo P1  
        WHERE C1.CodP = P1.Pid AND P1.Pid = P.Pid AND C1.CodF <> C.CodF  
      )
```

ESERCITAZIONE 4 - RIFORNIMENTI

11/05/2017

Dato il seguente schema della base di dati Rifornimenti:

FORNITORE (CodF, Nome_Fornitore, Città)

PRODOTTO (CodP, Descrizione, Prezzo)

FORNISCE (Anno, Ref_Prod, Ref_Forn, QTY)

FK: Ref_Prod REFERENCES Prodotto (CodP)

FK: Ref_Forno REFERENCES Fornitore (CodF)

Fornitore (CodF, Nome_Fornitore) fornisce un determinato Prodotto (CodP, Descrizione) ad un certo Prezzo (Fig 4.1).

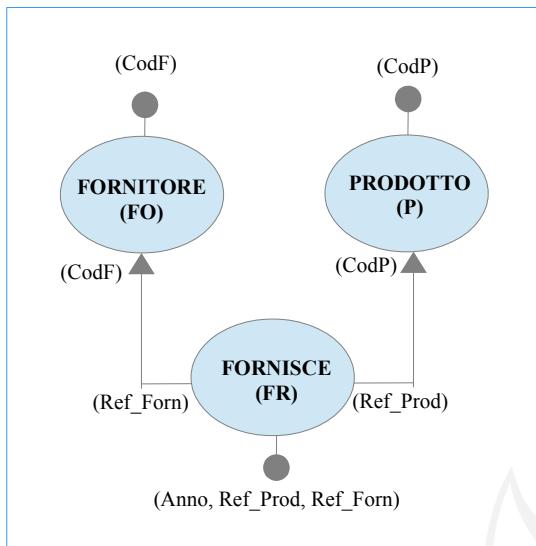


Fig. 4.1 – Modello dello schema di base di dati

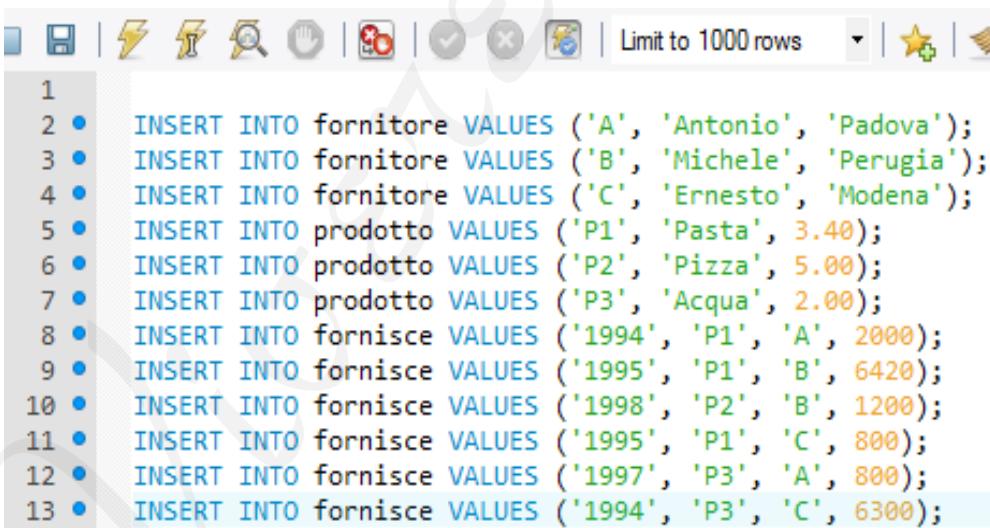
Interrogazioni (Algebra Relazionale 1, 2, 3) :

1. Selezionare i prodotti che nel 1995 sono stati forniti da almeno un fornitore di Modena;
2. Selezionare i prodotti non forniti da nessun fornitore di Modena;
3. Selezionare i prodotti che nel 1994 sono forniti in esclusiva da fornitori di Modena;
4. Selezionare per ogni anno la quantità totale dei prodotti forniti dai fornitori di Modena;
5. Selezionare per ogni anno il codice del fornitore che ha fornito in totale la maggiore quantità di prodotti;

Fig. 4.2 – Codice MySQL per la creazione del database Rifornimenti

```
1  --| Creazione del database rifornimenti
2  --
3 • CREATE DATABASE IF NOT EXISTS rifornimenti;
4 • USE rifornimenti;
5
6  --
7  -- Definizione della tabella fornitore
8  --
9
10 • └─CREATE TABLE fornitore (
11    cf char(16) NOT NULL,
12    nome_fornitore varchar(75) NOT NULL,
13    citta varchar(30) NOT NULL,
14    PRIMARY KEY (cf)
15);
16
17  --
18  -- Definizione della tabella prodotto
19  --
20
21 • └─CREATE TABLE prodotto (
22    cod_p char(5) NOT NULL,
23    descrizione varchar(45) NOT NULL,
24    prezzo numeric(6,2),
25    PRIMARY KEY (cod_p)
26);
27
28  --
29  -- Definizione della tabella fornisce
30  --
31
32 • └─CREATE TABLE fornisce (
33    anno int(4) NOT NULL,
34    cod_p_f char(5) NOT NULL,
35    cf_f char(16) NOT NULL,
36    qty int(2) NOT NULL,
37    PRIMARY KEY (anno, cod_p_f, cf_f),
38    FOREIGN KEY (cod_p_f) REFERENCES prodotto(cod_p),
39    FOREIGN KEY (cf_f) REFERENCES fornitore(cf)
40);
41
```

Fig. 4.3 – Codice MySQL per la popolazione del database



```
1
2 • INSERT INTO fornitore VALUES ('A', 'Antonio', 'Padova');
3 • INSERT INTO fornitore VALUES ('B', 'Michele', 'Perugia');
4 • INSERT INTO fornitore VALUES ('C', 'Ernesto', 'Modena');
5 • INSERT INTO prodotto VALUES ('P1', 'Pasta', 3.40);
6 • INSERT INTO prodotto VALUES ('P2', 'Pizza', 5.00);
7 • INSERT INTO prodotto VALUES ('P3', 'Acqua', 2.00);
8 • INSERT INTO fornisce VALUES ('1994', 'P1', 'A', 2000);
9 • INSERT INTO fornisce VALUES ('1995', 'P1', 'B', 6420);
10 • INSERT INTO fornisce VALUES ('1998', 'P2', 'B', 1200);
11 • INSERT INTO fornisce VALUES ('1995', 'P1', 'C', 800);
12 • INSERT INTO fornisce VALUES ('1997', 'P3', 'A', 800);
13 • INSERT INTO fornisce VALUES ('1994', 'P3', 'C', 6300);
```

Fig. 4.4 – Istanze della base di dati

cf	nome_fornitore	citta		cod_p	descrizione	prezzo		anno	cod_p_f	cf_f	qty
A	Antonio	Padova		P1	Pasta	3.40		1994	P1	A	2000
B	Michele	Perugia		P2	Pizza	5.00		1994	P3	C	6300
C	Ernesto	Modena		P3	Acqua	2.00		1995	P1	B	6420
NULL	NULL	NULL		NULL	NULL	NULL		1995	P1	C	800
								1997	P3	A	800
								1998	P2	B	1200
								NULL	NULL	NULL	NULL

fornitore 1 × prodotto1 × fornisce 1 ×

1. Selezionare i prodotti che nel 1995 sono stati forniti da almeno un fornitore di Modena.

SQL

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: join, selezione

LOGICA: Seleziono tutti i prodotti tali che la città del fornitore sia Modena e che l'anno di riferimento sia il 1995

```
SELECT P. CodP AS CodiceProdotto, P. Descrizione, P.Prezzo
FROM Fornitore FO, Fornisce FR, Prodotto P
WHERE FO. CodF= FR. Ref_Forn AND FR.Ref_Prod= P.CodP AND FO.Città='Modena' AND FR.Anno=1995
```

ALGEBRA

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: Join, selezione, proiezione

LOGICA: Applico la stessa logica di SQL

```
PROJCodP, Descrizione, Prezzo(SELAnno=1995 AND Città='Modena'(( FO JOINFO.CodF=FR.Ref_Forn FR ) JOINRef_Prod=P.CodP P ))
```

MYSQL

```

1 • SELECT
2     p.*
3     FROM
4         prodotto p,
5         fornisce fs,
6         fornitore fr
7     WHERE
8         p.cod_p = fs.cod_p_f AND fs.cf_f = fr.cf
9             AND fs.anno = 1995
10            AND citta = 'Modena';

```

The screenshot shows the MySQL Workbench interface with the query editor containing the above SQL code. Below the editor is the result grid showing one row of data:

cod_p	descrizione	prezzo
P1	Pasta	3.40

Fig. 4.5 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

2. Selezionare i prodotti non forniti da nessun fornitore di Modena.

SQL

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, operatore esistenziale Not Exists, join e selezione

LOGICA: Con l'interrogazione B seleziono per ogni prodotto, passato come parametro, le tuple tali che il fornitore sia della città di Modena. Con l'interrogazione A seleziono i prodotti per i quali l'interrogazione B da luogo ad una relazione vuota.

```
A   SELECT P.CodP AS CodiceProdotto, P. Descrizione, P. Prezzo  
     FROM Prodotto P  
 WHERE NOT EXISTS (SELECT *  
                      FROM Fornitore FO, Fornisce FR  
                     WHERE FO.CodF=FR.Ref_Forn AND FR.Ref_Prod=P.CodP AND FO.Città='Modena')
```

ALGEBRA

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: Join, selezione, proiezione, operatore insiemistico Differenza

LOGICA: Ottengo tutte le informazioni sui prodotti forniti dai fornitori di Modena, e li sottraggo all'insieme di tutti i prodotti. Infine proietto solo i dati relativi ai prodotti.

PROJ_{CodP, Descrizione, Prezzo}(((FO JOIN_{FO.CodF=FR.Ref_Forn} FR) JOIN_{Ref_Prod=P.CodP} P) - (SEL_{Città='Modena'}((FO JOIN_{FO.CodF=FR.Ref_Forn} FR)
JOIN_{Ref_Prod=P.CodP} P)))

MYSQL

The screenshot shows the MySQL Workbench interface. In the top pane, a code editor displays the following SQL query:

```
1 •  SELECT *  
2   FROM prodotto p  
3 WHERE p.cod_p NOT IN (SELECT  
4                           fs.cod_p_f  
5                         FROM fornisce fs,  
6                               fornitore fr  
7                         WHERE  
8                           fs.cf_f = fr.cf AND citta = 'Modena');
```

In the bottom pane, the result grid shows one row of data:

cod_p	descrizione	prezzo
P2	Pizza	5.00

Fig. 4.6 – Codice MySQL per l'interrogazione 2 e le istanze risultanti

3. Sia in algebra che in SQL selezionare i prodotti che nel 1994 sono forniti in esclusiva da fornitori di Modena.

SQL

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: Un'interrogazione innestata con operatore insiemistico Not In, join, selezione

LOGICA: Con l'interrogazione B ottengo tutti i prodotti che sono forniti da fornitori di una città diversa da Modena nel 1994. Con l'interrogazione A seleziono i fornitori di Modena che nel 1994 non hanno fornito nessun prodotto che appartiene all'insieme risultante dall'interrogazione B.

```
A   SELECT P.CodP AS CodiceProdotto, P. Descrizione, P. Prezzo  
     FROM Fornitore FO1, Fornisce FR1, Prodotto P  
    WHERE FO1.CodF= FR1.Ref_Forn AND FR1.Ref_Prod= P.CodP AND FO1.Città='Modena' AND FR1.Anno='1994' AND  
          CodP NOT IN ( SELECT Ref_Prod  
                        B   FROM Fornitore FO2, Fornisce FR2  
                       WHERE FO2.CodF=FR2.Ref_Forn AND Anno='1994' AND Città <> 'Modena')
```

ALGEBRA

INFORMAZIONI: Fornitore, Fornisce, Prodotto

OPERAZIONI: Join, selezione, proiezione, operatore insiemistico Differenza

LOGICA: Con due join e una selezione ottengo tutti i prodotti forniti nel 1994 da fornitori di una città diversa da Modena, e li sottraggo a tutti i prodotti forniti da Modena nel 1994. Infine proietto tutti i dati di tali prodotti.

```
PROJ CodP,Descrizione, Prezzo (( SEL Anno=1994 AND Città='Modena' ( ( FO JOINFO.CodF=FR.Ref_Forn FR ) JOINRef_Prod=P.CodP P )) -  
          ( SEL Anno='1994' AND Città <> 'Modena' ( ( FO JOINFO.CodF=FR.Ref_Forn FR ) JOINRef_Prod=P.CodP P )))
```

MYSQL

```

1 • SELECT
2     p.*
3     FROM
4         prodotto p,
5         fornisce fs,
6         fornitore fr
7     WHERE
8         p.cod_p = fs.cod_p_f AND fs.cf_f = fr.cf
9             AND fs.anno = 1994
10            AND citta = 'Modena'
11            AND p.cod_p NOT IN (SELECT
12                p.cod_p
13                FROM
14                    prodotto p,
15                    fornisce fs,
16                    fornitore fr
17                WHERE
18                    p.cod_p = fs.cod_p_f AND fs.cf_f = fr.cf
19                        AND fs.anno = 1994
20                            AND citta <> 'Modena');

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cod_p	descrizione	prezzo
P3	Acqua	2.00

Fig. 4.7 – Codice MySQL per l'interrogazione 3 e le istanze risultanti

4. Selezionare per ogni anno la quantità totale dei prodotti forniti dai fornitori di Modena.

INFORMAZIONI: Fornitore, Fornisce

OPERAZIONI: Join, selezione, Group By, operatore aggregato Sum

LOGICA: Seleziono i fornitori di Modena e per ogni anno proietto la quantità totale dei prodotti forniti da tali fornitori.

```
SELECT Anno, SUM (QTY) AS Tot_Prod_Annuale
FROM Fornitore FO, Fornisce FR
WHERE FO.CodF= FR.Ref_Forn AND FO.Città='Modena'
GROUP BY Anno
```

MYSQL

```

1 • SELECT
2     fs.anno, fs.qty
3     FROM
4         fornisce fs,
5         fornitore fr
6     WHERE
7         fs.cf_f = fr.cf AND citta = 'Modena'
8     GROUP BY fs.anno;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

anno	qty
1994	6300
1995	800

Fig. 4.8 – Codice MySQL per l'interrogazione 4 e le istanze risultanti

5. Selezionare per ogni anno il codice del fornitore che ha fornito in totale la maggiore quantità di prodotti.

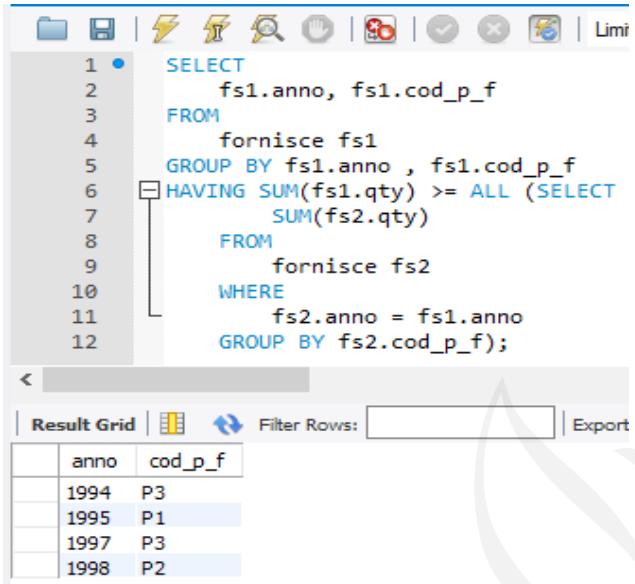
INFORMAZIONI: Fornisce

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, Group By, operatore \geq All, operatore aggregato Sum

LOGICA: Con l'interrogazione B ottengo per ogni anno la quantità totale di prodotti forniti da ciascun fornitore. Con l'interrogazione A seleziono per ogni anno il fornitore la cui quantità totale di prodotti venduti è maggiore o uguale al risultato dell'interrogazione B (è pari al massimo)

```
A   SELECT FR1.Anno, FR1.Ref_Forn
      FROM Fornisce FR1
     GROUP BY Anno, Ref_Forn
    HAVING SUM(QTY)>= ALL ( SELECT SUM(QTY)
                                FROM Fornisce FR2
                               B   WHERE FR2.Anno=FR1.Anno
                                 GROUP BY Ref_Forn)
```

MYSQL



The screenshot shows the MySQL Workbench interface. In the top bar, there are various icons for file operations, search, and navigation. Below the toolbar, the SQL editor window displays a query numbered from 1 to 12. The query selects the year and supplier code for suppliers whose total quantity is greater than or equal to the maximum total quantity for each year. The result grid below shows four rows of data: 1994 P3, 1995 P1, 1997 P3, and 1998 P2.

```
1  SELECT
2      fs1.anno, fs1.cod_p_f
3  FROM
4      fornisce fs1
5  GROUP BY fs1.anno , fs1.cod_p_f
6  HAVING SUM(fs1.qty) >= ALL (SELECT
7          SUM(fs2.qty)
8      FROM
9          fornisce fs2
10     WHERE
11         fs2.anno = fs1.anno
12     GROUP BY fs2.cod_p_f);
```

	anno	cod_p_f
1	1994	P3
2	1995	P1
3	1997	P3
4	1998	P2

Fig. 4.9 – Codice MySQL per l'interrogazione 5 e le istanze risultanti

ESERCITAZIONE 5 - FABBRICA

12/05/2017

Dato il seguente schema della base di dati Fabbrica:

OPERATORE (Cod_Op, Nome, Indirizzo, Qualifica, CostoOrario)

ARTICOLO (Cod_Art, Descrizione)

LOTTO (Ref_Art, Ref_Op, TotEsem)

FK: Ref_Art REFERENCES Articolo (Cod_Art)

FK: Ref_Op REFERENCES Operatore (Cod_Op)

RECLAMO (R_Art, R_Op, Num_Es_Lotto, Nome_Cliente)

FK: (R_Art, R_Op) REFERENCES Lotto (Ref_Art, Ref_Op)

- 1) Di ogni Lotto (L) posso ottenere le informazioni sull'Operatore (O) che lo ha confezionato, sull'Articolo (A) realizzato e in che quantità (TotEsem)
- 2) Di ogni Reclamo (R) posso ottenere le informazioni sul Lotto in questione, su che esemplare del relativo Lotto è stato fatto tale Reclamo e da chi (Nome_Cliente)

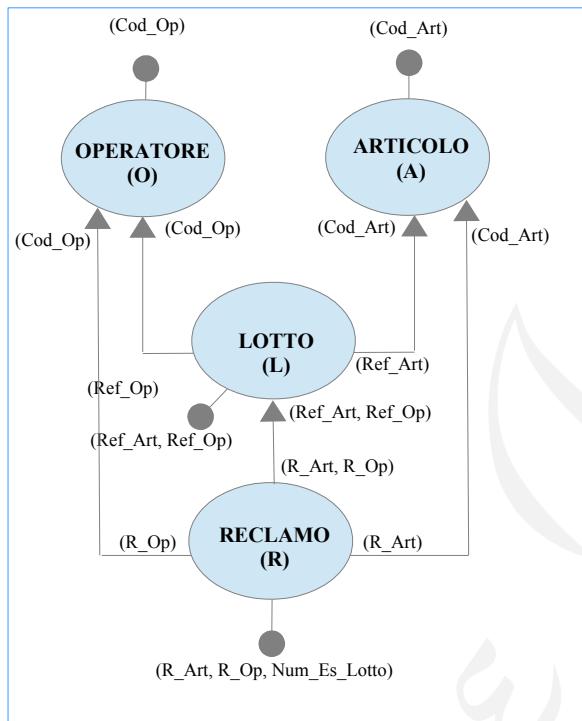


Fig. 5.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1, 2, 3):

1. Selezionare il nome del cliente che ha fatto reclami per tutti gli operatori;
2. Selezionare gli operatori per i quali non esiste alcun reclamo;
3. Selezionare i codici degli operatori per i quali ogni lotto da essi confezionato contiene almeno un esemplare al quale si riferisce un reclamo;
4. Selezionare il lotto con più reclami;
5. Selezionare per ogni articolo il codice dell'operatore che ha confezionato il lotto con il maggiore numero di esemplari, senza considerare i lotti per il quali l'attributo "TotEsem" è non specificato;

Fig. 5.2 – Creazione del database fabbrica

```
1  -- Creazione del database
2  --
3  ● CREATE DATABASE IF NOT EXISTS fabbrica;
4  ● USE fabbrica;
5  --
6  --
7  -- Definizione della tabella operatore
8  --
9
10 ● └─ CREATE TABLE operatore (
11     cod_op char(16) NOT NULL,
12     nome varchar(75) NOT NULL,
13     indirizzo varchar(100) NOT NULL,
14     qualifica varchar(75) NOT NULL,
15     costo_orario char(16) NOT NULL,
16     PRIMARY KEY (cod_op)
17 );
18
19
20 -- Definizione della tabella articolo
21 --
22 ● └─ CREATE TABLE articolo (
23     cod_art char(16) NOT NULL,
24     Descrizione varchar(75) NOT NULL,
25     PRIMARY KEY (cod_art)
26 );
27
28 -- Definizione della tabella lotto
29 --
30
31
32
33 ● └─ CREATE TABLE lotto (
34     ref_art char(16) NOT NULL,
35     ref_op char(16) NOT NULL,
36     tot_esem char(16) NOT NULL,
37     PRIMARY KEY (ref_art, ref_op),
38     FOREIGN KEY (ref_art) REFERENCES articolo(cod_art),
39     FOREIGN KEY (ref_op) REFERENCES operatore(cod_op)
40 );
41
42
43
44 -- Definizione della tabella reclamo
45 --
46
47 ● └─ CREATE TABLE reclamo (
48     r_art char(16) NOT NULL,
49     r_op char(16) NOT NULL,
50     num_es_lotto char(16) NOT NULL,
51     nome_cliente varchar(75) NOT NULL,
52     PRIMARY KEY (r_art, r_op, num_es_lotto),
53     FOREIGN KEY (r_art) REFERENCES articolo(cod_art),
54     FOREIGN KEY (r_op) REFERENCES operatore(cod_op)
55 );
56
57
```

Fig. 5.3 – Popolazione del database

```
4
5  ● INSERT INTO articolo values ('A', 'Iphone');
6  ● insert into articolo values('B', 'Laptop');
7  ● insert into articolo values('C', 'PC');
8  ● insert into articolo values ('D', 'Stampante');
9  ● insert into operatore values('01', 'Giulio', 'Via Sciuti 33', 'Tecnico', 50);
10 ● insert into operatore values('02', 'Marco', 'Via Basile 12', 'Ingegnere', 60);
11 ● insert into operatore values('03', 'Luigi', 'Via Maqueda 6', 'Macchinista', 40);
12 ● insert into lotto values('A', '01', 100);
13 ● insert into lotto values('B', '01', 70);
14 ● insert into lotto values('C', '02', 120);
15 ● insert into lotto values('D', '03', 95);
16 ● insert into reclamo values('A', '01', 32, 'Pietro Rossi');
17 ● insert into reclamo values('C', '02', 40, 'Pietro Rossi');
18 ● insert into reclamo values('D', '03', 20, 'Pietro Rossi');
19 ● insert into reclamo values('A', '01', 56, 'Riccardo Belli');
```

Fig. 5.3 – Istanze del database

The diagram illustrates four database instances:

- operatori**: A table with columns cod_op, nome, indirizzo, qualifica, and costo_orario. Data: (01, Giulio, Via Sciuti 33, Tecnico, 50), (02, Marco, Via Basile 12, Ingegnere, 60), (03, Luidi, Via Maqueda 6, Macchinista, 40).
- articoli**: A table with columns cod_art and Descrizione. Data: (A, Iphone), (B, Laptop), (C, PC), (D, Stampante).
- reclamo**: A table with columns r_art, r_op, num_es_lotto, and nome_cliente. Data: (A, 01, 32, Pietro Rossi), (a, 01, 56, Riccardo Belli), (C, 02, 40, Pietro Rossi), (D, 03, 20, Pietro Rossi).
- lotto**: A table with columns art, op, and prezzo. Data: (A, 01, 100), (B, 01, 70), (C, 02, 120), (D, 03, 95).

1. Selezionare il nome del cliente che ha fatto reclami per tutti gli operatori

SQL

INFORMAZIONI: Reclamo, Operatore

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, doppia negazione.

LOGICA: L'interrogazione C restituisce le tuple di reclamo relative al cliente passato come parametro dalla A e al codice operatore passato come parametro dalla B. Tramite la doppia negazione selezioniamo solo quei clienti per i quali non esiste un operatore per cui essi non abbiano fatto reclamo.

```

A   SELECT DISTINCT R.Nome_Cliente
      FROM Reclamo R
      WHERE NOT EXISTS (SELECT *
                         B   FROM OPERATORE O
                         WHERE NOT EXISTS( SELECT *
                                           FROM RECLAMO R1
                                           C   WHERE R1.Nome_Cliente=R.Nome_Cliente AND R1.R_Op= O.Cod_Op ) )

```

ALGEBRA

INFORMAZIONI: Reclamo, Operatore

OPERAZIONI: Ridenominazione, proiezioni, join e differenza

LOGICA: Con la prima sottrazione (la più interna) sottraggo a tutti le coppie Cod_Op, Nome_Cliente ottenute come risultato di un join naturale fra Operatore e Reclamo quelli che sono effettivamente presenti nella tabella Reclamo. Ottengo così le coppie false Cod_Op, Nome_Cliente, pertanto proiettando solo il Nome_Cliente sto selezionando quei clienti per cui esiste almeno un operatore per il quale essi non hanno fatto reclamo. Con la sottrazione più esterna ottengo infine i Clienti che hanno effettuato reclami per tutti gli operatori.

$$(\text{PROJ}_{\text{Nome_Cliente}} \text{R}) - (\text{PROJ}_{\text{Nome_Cliente}}((\text{PROJ}_{\text{Cod_Op}, \text{Nome_Cliente}}(\text{O JOIN R})) - (\text{PROJ}_{\text{Cod_Op}, \text{Nome_Cliente}}(\text{REN}_{\text{Cod_Op} \leftarrow \text{R}_\text{Op}} \text{R}))))$$

MYSQL

```

5
6 • select distinct r.nome_cliente
7   from reclamo r
8   where not exists (
9     select *
10    from operatore o
11   where not exists (
12     select *
13       from reclamo r1
14      where r.nome_cliente = r1.nome_cliente and r1.r_op = o.cod_op)
15   )
16
17
18
19
20
21
  
```

Result Grid | Filter Rows: Search | Export:

nome_cliente
Pietro Rossi

Fig. 5.4 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare gli operatori per i quali non esiste alcun reclamo.

SQL

INFORMAZIONI: Reclamo, Operatore

OPERAZIONI: 1 interrogazione innestata, operatore insiemistico NOT IN

LOGICA: L'interrogazione B restituisce tutti i codici degli operatori per i quali è stato effettuato almeno un reclamo, con l'interrogazione A selezioniamo i codici degli operatori che non si trovano fra i risultati dell'interrogazione B.

A **SELECT DISTINCT ***
FROM OPERATORE O
WHERE O.Cod_Op NOT IN (SELECT DISTINCT R.R_Op
B FROM RECLAMO R)

ALGEBRA

INFORMAZIONI: Reclamo, Operatore

OPERAZIONI: join, proiezioni e differenza

LOGICA: Con il join ottengo le informazioni su tutti gli operatori il cui codice è presente almeno una volta fra i valori dell'attributo R_OP di Reclamo, ne proietto solo codice e nome e li sottraggo alle coppie cod_Op e Nome della relazione Operatore.

(PROJ_{Cod_Op, Nome} O) - (PROJ_{Cod_Op, Nome} (O JOIN_{O.Cod_Op=R.R_Op} R))

MYSQL

```

1 • select o.cod_op, o.nome as nome_operatore
2   from operatore o
3   where o.cod_op not in (
4     select distinct r_op
5       from reclamo r
6     )
  
```

Result Grid | Filter Rows: Search | Edit:

cod_op	nome_operatore
NULL	NULL

Fig. 5.5 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare i codici degli operatori per i quali ogni lotto da essi confezionato contiene almeno un esemplare al quale si riferisce un reclamo.

SQL

INFORMAZIONI: Reclamo, Lotto

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, doppia negazione.

LOGICA: L'interrogazione C restituisce le tuple di reclamo relative all'operatore e all'articolo passati come parametri dalla B. Tramite la doppia negazione selezioniamo solo quegli operatori per i quali non esiste un lotto da essi confezionato per il quale non esista un reclamo.

```
A  SELECT DISTINCT L1.Ref_Op
  FROM LOTTO L1
 WHERE NOT EXISTS (SELECT *
    B   FROM LOTTO L2
      WHERE L2.Ref_Op = L1.Ref_Op AND NOT EXISTS( SELECT *
          FROM RECLAMO R1
          C   WHERE R1.R_Op = L2.R_O AND R1.R_Art =
              L2.Ref_Art )
        )
```

ALGEBRA

INFORMAZIONI: Reclamo, Lotto

OPERAZIONI: proiezioni , differenza e ridefinizione.

LOGICA: Con la differenza più interna ottengo i codici degli operatori per i quali esiste almeno un lotto che non è stato reclamato, sottraendoli infine ai codici operatore di lotto ottengo tutti i codici degli operatori per i quali non esiste un lotto da essi confezionato che non sia stato reclamato almeno una volta.

$(\text{PROJ}_{\text{Ref_Op}} \text{L}) - (\text{PROJ}_{\text{Ref_Op}} ((\text{PROJ}_{\text{Ref_Art}, \text{Ref_Op}} \text{L}) - (\text{PROJ}_{\text{Ref_Art}, \text{Ref_Op}} (\text{REN}_{\text{Ref_Art}, \text{Ref_Op} \leftarrow \text{R_Art}, \text{R_Op}} \text{R}))))$

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following SQL code:

```
1 • select l1.ref_op
  2   from lotto l1
  3   where not exists (
  4     select *
  5       from lotto l2
  6       where l2.ref_op = l1.ref_op and not exists (
  7         select *
  8           from reclamo r
  9           where r.r_op = l2.ref_op and r.r_art = l2.ref_art)
 10      )
```

Below the query editor is a result grid labeled "Result Grid". It has a single column named "ref_op" with two rows: "02" and "03".

Fig. 5.6 – Codice MySQL per l'interrogazione 3 e istanze risultanti

4. Selezionare il lotto con più reclami.

INFORMAZIONI: Reclamo, Lotto

OPERAZIONI: 1 interrogazione innestata, 1 group by, join.

LOGICA: Tramite l'interrogazione A suddivido la relazione ottenuta come join fra lotto e reclamo in tante sottotabelle quanti sono i valori diversi delle terne(Ref_Art, Ref_Op, Tot_Esem). Con l'interrogazione B suddivido la relazione R1 in tante sottotabelle quanti sono i valori diversi delle coppie (R_Art, R_Op) ed ottengo tramite l'operatore aggregato count il numero di tuple presenti in ogni sottotabella. Attraverso l'operatore di confronto \geq ALL seleziono nell'interrogazione A solo i valori di quelle terne le cui sottotabelle hanno un numero di tuple maggiore di tutti i valori ottenuti come risultato dell'interrogazione B.

```

A   SELECT DISTINCT L.Ref_Op, L.Ref_Art, L.Tot_Esem
    FROM LOTTO L, RECLAMO R
    WHERE L.Ref_Art=R.R_Art AND L.REF_OP=R.R_Op
    GROUPBY L.Ref_Art, L.Ref_Op, L.Tot_Esem
    HAVING COUNT(*) >= ALL (SELECT COUNT(*)
B       FROM RECLAMO R1
        GROUPBY R1.R_Art, R1.R_Op)

```

MYSQL

```

1
2 • select l.ref_op, l.ref_art, l.tot_esem
3   from lotto l, reclamo r
4   where l.ref_art = r.r_art and l.ref_op = r.r_op
5   group by l.ref_art, l.ref_op, l.tot_esem
6   having count(*) >= all (
7     select count(*)
8       from reclamo r1
9       group by r1.r_art, r1.r_op)

```

ref_op	ref_art	tot_esem
01	A	100

Fig. 5.7 – Codice MySQL per l'interrogazione 4 e istanze risultanti

5. Selezionare per ogni articolo il codice dell'operatore che ha confezionato il lotto con il maggiore numero di esemplari, senza considerare i lotti per il quali l'attributo "TotEsem" è non specificato.

INFORMAZIONI: Lotto

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri.

LOGICA: Tramite l'interrogazione B ottengo i totali degli esemplari per ogni lotto relativo all' articolo passato come parametro. Così tramite l'interrogazione A seleziono solo quegli operatori che hanno confezionato il lotto con numero di esemplari maggiore o uguale a tutti i valori risultanti dall'operazione B. Ripeto l'interrogazione innestata per ogni valore di Ref_Art

```

A   SELECT L1.Ref_Art, L1. Ref_Op
    FROM LOTTO L1
    WHERE L1.Tot_Esem IS NOT NULL AND
          L1.Tot_Esem>= ALL (SELECT L2.Tot_Esem
B       FROM LOTTO L2
          WHERE L2.Tot_Esem IS NOT NULL AND L2.Ref_Art=L1.Ref_Art )

```

MYSQL

```

3 • select l1.ref_art, l1.ref_op
4   from lotto l1
5   where l1.tot_esem is not null and l1.tot_esem >= all(
6     select l2.tot_esem
7       from lotto l2
8       where l2.tot_esem is not null and l2.ref_art = l1.ref_art)

```

ref_art	ref_op
A	01
B	01
C	02
D	03

Fig. 5.8 – Codice MySQL per l'interrogazione 5 e istanze risultanti

ESERCITAZIONE 6 - AZIENDA

19/05/2017

Dato il seguente schema della base di dati AZIENDA:

SEDE (Cod_Sede, Responsabile, Città)

IMPIEGATO (Cod_Imp, I_Nome, Cognome, Ref_Sede, Ruolo, Stipendio)

FK: Ref_Sede REFERENCES Sede (Cod_Sede)

PROGETTO (Cod_Prog, P_Nome, Id_Sede)

FK: Id_Sede REFERENCES Sede (Cod_Sede)

Impiegato (Cod_Imp, I_Nome) lavora in una determinata Sede(Cod_Sede) che può essere luogo di un Progetto(Cod_Imp) (Fig 6.1)

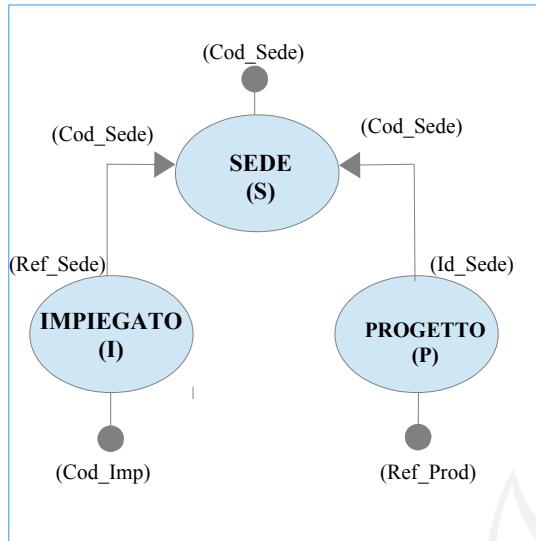


Fig. 6.1 – Modello dello schema di base di dati

Interrogazioni SQL:

1. Creare la vista progetti_sedi che mi dia il codice del progetto e il codice della sede;
2. Con query annidate, visualizzare il codice e la città della sede che ha il massimo numero di impiegati;
3. Riscrivere la precedente creando prima la vista e poi utilizzandola;
4. Scrivere una vista non aggiornabile che mi fornisca le informazioni degli impiegati di Bologna;
5. Scrivere una vista aggiornabile che mi fornisca le informazioni degli impiegati di Bologna

Fig 6.2- Codice MYSQL per la creazione del database azienda

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • CREATE DATABASE IF NOT EXISTS azienda;
2 • USE azienda;
3
4 --
5 -- Definition of table Sede
6 --
7
8 • └─ CREATE TABLE sede (
9     cod_sede char(5) NOT NULL,
10    responsabile varchar(20) NOT NULL,
11    città varchar(30) NOT NULL,
12    PRIMARY KEY (cod_sede)
13 );
14
15 --
16 -- Definition of table impiegato
17 --
18
19 • └─ CREATE TABLE impiegato (
20     cod_imp char(5) NOT NULL,
21     nome varchar(20) NOT NULL,
22     cognome varchar(20) NOT NULL,
23     ref_sede char(5) NOT NULL,
24     ruolo varchar(15) NOT NULL,
25     stipendio numeric(7),
26     PRIMARY KEY (cod_imp),
27     FOREIGN KEY (ref_sede) REFERENCES sede(cod_sede)
28 );
29
30 --
31 -- Definition of table progetto
32 --
33
34 • └─ CREATE TABLE progetto (
35     cod_prog char(5) NOT NULL,
36     p_nome varchar(20) NOT NULL,
37     id_sede char(5) NOT NULL,
38     PRIMARY KEY (cod_prog),
39     FOREIGN KEY (id_sede) REFERENCES sede(cod_sede)
40 );
41
```

Figura 6.3- Codice MYSQL per la popolazione del database

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
1 • INSERT INTO sede VALUES ('00001', 'Rossi', 'Padova');
2 • INSERT INTO sede VALUES ('00002', 'Neri', 'Perugia');
3 • INSERT INTO sede VALUES ('00003', 'Bianchi', 'Modena');
4 • INSERT INTO impiegato VALUES ('00A01', 'Michele', 'Rossi', '00001', 'Dirigente', 2000);
5 • INSERT INTO impiegato VALUES ('00B02', 'Giovanni', 'Piazza', '00001', 'Operaio', 800);
6 • INSERT INTO impiegato VALUES ('00C03', 'Francesca', 'Bevilacqua', '00002', 'Ingegnere', 1800);
7 • INSERT INTO impiegato VALUES ('00D04', 'Luca', 'Alberti', '00003', 'Architetto', 1000);
8 • INSERT INTO progetto VALUES ('P0001', 'Spartacus', '00001');
9 • INSERT INTO progetto VALUES ('P0002', 'Athena', '00002');
10 • INSERT INTO progetto VALUES ('P0003', 'Argo', '00003');
```

Figura 6.4- Istanze della base di dati

cod_imp	nome	cognome	ref_sede	ruolo	stipendio	cod_sede	responsabile	città	cod_prog	p_nome	id_sede
00A01	Michele	Rossi	00001	Dirigente	2000	00001	Rossi	Padova	P0001	Spartacus	00001
00B02	Giovanni	Piazza	00001	Onerario	800	00002	Neri	Perudia	P0002	Athena	00002
00C03	Francesca	Bevilacqua	00002	Ingegnere	1800	00003	Bianchi	Modena	P0003	Arao	00003
00D04	Luca	Alberti	00003	Architetto	1000	NULL	NULL	NULL	NULL	NULL	NULL

1. Creare la vista progetti_sedi che mi dia il codice del progetto e il codice della sede.

SQL

INFORMAZIONI: Sede e Progetto

OPERAZIONI: Join

LOGICA: Con il join all'interno dell'interrogazione ottengo le informazioni sui progetti e le relative sedi, dunque proietto solo la città, il codice e il nome del progetto. Infine creo una vista sugli attributi ottenuti.

```
CREATE VIEW Progetti_Sedi (Cod_Prog, Cod_Sede,Città_Sede) AS
SELECT S.Città; P.Cod_Prog, P.PNome
FROM SEDE S, PROGETTO P
WHERE S.Cod_Sede=P.Id_Sede
```

MYSQL

```
1 • create view progetti_sedi (cod_prog, cod_sede, città_sede) as
2 select s.città, p.cod_prog, p.p_nome
3 from sede s, progetto p
4 where s.cod_sede=p.id_sede;
```

cod_prog	cod_sede	città_sede
Padova	P0001	Spartacus
Perudia	P0002	Athena
Modena	P0003	Arao

Fig. 6.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Con query annidate, visualizzare il codice e la città della sede che ha il massimo numero di impiegati.

SQL

INFORMAZIONI: Sede e Impiegato

OPERAZIONI: Interrogazione innestata, doppio groupby , join

LOGICA: Con l'interrogazione B ottengo il numero di impiegati per ogni sede e con l'interrogazione A, dopo aver suddiviso la relazione ,ottenuta dal join fra impiegato e sede, in tante sottotabelle quante sono le sedi diverse. Considero i valori di quelle sedi tali per cui il raggruppamento da luogo a relazioni che hanno un numero di tuple maggiori di tutti i valori ottenuti come risultato dell'interrogazione B.

```
A SELECT S.Cod_Sede, S.Città
  FROM SEDE S, IMPIEGATO I
 WHERE S.Cod_Sede=I.Ref_Sede
 GROUPBY S.Cod_Sede, S.Città
 HAVING COUNT(*)>= ALL ( SELECT COUNT(*)
                           FROM IMPIEGATO I2
                           GROUPBY S.Cod_Sede )
```

MYSQL

```

1 • select S.cod_sede, S.città
2   from impiegato I1, sede S
3   where I1.ref_sede=S.cod_sede
4   group by S.cod_sede, S.città
5   having count(*) >=all
6   (select count(*)
7    from impiegato I2
8    group by I2.ref_sede
9   );
10

```

The screenshot shows the MySQL Workbench interface. The top part is a query editor with the code above. Below it is a result grid titled "Result Grid" with two columns: "cod_sede" and "città". A single row is displayed with values "00001" and "Padova". There are also buttons for "Filter Rows" and a search bar.

Fig. 6.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Riscrivere la precedente creando prima la vista e poi utilizzandola.

SQL

INFORMAZIONI: Impiegato

OPERAZIONI: group by, operatore aggregato count

LOGICA: Suddivo la tabella Impiegato in tante sottotabelle quanti sono i valori diversi dell'attributo Ref_sede, ottengo una relazione finale i cui attributi sono i valori diversi di ref_sede e il numero di tuple relativi ad ogni sottotabella corrispondente. Infine creo una lista sul risultato ottenuto.

```

CREATE VIEW Impiegati_Sedi (Ref_Sede,NImp) AS
SELECT I.Ref_Sede, Count(*) AS Numero_Imp
FROM IMPIEGATO I
GROUPBY I.Ref_Sede

```

```

1 • create view impiegati_sede(ref_sede, NImp) AS
2   select I.ref_sede, count(*) as numero_imp
3   from impiegato I
4   group by I. ref_sede;

```

The screenshot shows the MySQL Workbench interface. The top part is a query editor with the code above. Below it is a result grid titled "Result Grid" with two columns: "ref_sede" and "numero_imp". A single row is displayed with values "00001" and "10". There are also buttons for "Filter Rows" and a search bar.

Fig. 6.7 – Codice MySQL per la creazione della vista dell'interrogazione 3

INFORMAZIONI: Sede ,Impiegato, Impiegati_Sede

OPERAZIONI: Interrogazione innestata , operatore aggregato max e operatore di confronto =

LOGICA: Nell'interrogazione B ottengo il valore massimo dell'attributo Nimp della vista IMPIEGATI_SEDE, dunque con l'interrogazione A seleziono solo le tuple della relazione, ottenuta dal join fra SEDE E IMPIEGATI_SEDE , che hanno valore in corrispondenza all'attributo Nimp pari al risultato dell'interrogazione B.

```

A   SELECT S.Cod_Sede, S.Citt...
     FROM SEDE S, IMPIEGATI_SEDE ImS
     WHERE S.Cod_Sede=ImS.Ref_Sede AND ImS.Nimp= ( SELECT MAX(ImS1.Nimp)
                                                 FROM IMPIEGATI_SEDE ImS1
                                                 )

```

MYSQL

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL query:

```
1 • select S.cod_sede, S.città
2   from impiegati_sede ImS, sede S
3   where ImS.ref_sede=S.cod_sede and
4     ImS.NImp =
5     (select max(ImS1.NImp)
6      from impiegati_sede ImS1
7    );
8
9
```

In the bottom-right pane, there is a results grid titled "Result Grid" with the following data:

	cod_sede	città
	00001	Padova

Fig. 6.8 – Codice MySQL per l'interrogazione 3 e istanze risultanti (vedi fig. 6.7)

4. Scrivere una vista non aggiornabile che mi fornisca le informazioni degli impiegati di Bologna.

SQL

INFORMAZIONI:Impiegato, Sede

OPERAZIONI: join, selezione

LOGICA: con l'interrogazione ottengo le informazioni degli impiegati la cui sede si trova a Bologna. Sui risultati ottenuti creo una vista.

```
CREATE VIEW Impiegati_Bologna (Cod_Imp, Nome,Cognome, Ref_Sede, Ruolo, Stipendio) AS
SELECT I.*  
FROM IMPIEGATO I, SEDE S  
WHERE I.Ref_Sede =S.Cod_Sede AND S.Città= 'Bologna'
```

MYSQL

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a code editor with the following SQL query:

```
1 • create view impiegati_Bologna(cod_imp, nome, cognome, ref_sede, ruolo, stipendio)
2 select I.*
3   from impiegato I, sede S
4  where I.ref_sede=s.cod_sede and
5    S.città='Bologna'
6
```

In the bottom-right pane, there is a results grid titled "Result Grid" with the following data:

	cod_imp	nome	cognome	ref_sede	ruolo	stipendio
--	---------	------	---------	----------	-------	-----------

Fig. 6.9 – Codice MySQL per l'interrogazione 4 e istanze risultanti

5. Scrivere una vista aggiornabile che mi fornisca le informazioni degli impiegati di Bologna.

SQL

INFORMAZIONI:Impiegato, Sede

OPERAZIONI: interrogazione innestata, operatore insiemistico in

LOGICA: Con l'interrogazione B ottengo i codici delle sedi che si trovano a Bolona). Con l'interrogazione A ottengo le informazioni degli impiegati la cui sede si trova fra i risultati della B.

```

A CREATE VIEW Impiegati_Bo (Cod_Imp, Nome,Cognome, Ref_Sede, Ruolo, Stipendio) AS
  SELECT I.*  

  FROM IMPIEGATO I  

  WHERE I.Ref_Sede IN ( SELECT S.Cod_Sede  

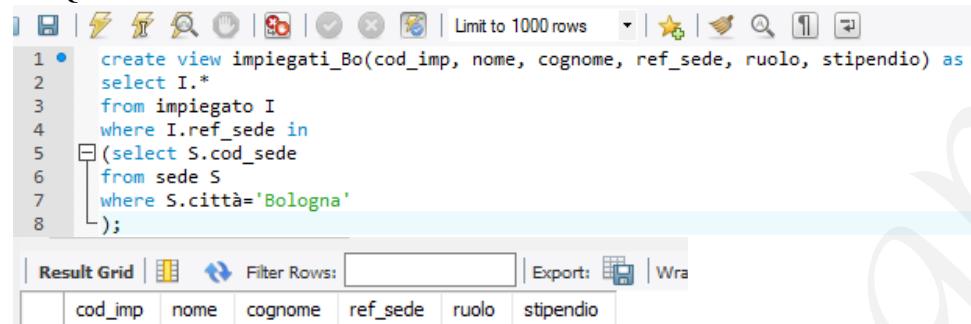
                           FROM SEDE S  

                           B WHERE S.Città= 'Bologna'  

                           )

```

MYSQL



The screenshot shows the MySQL Workbench interface with a query editor window. The code entered is:

```

1 • create view impiegati_Bo(cod_imp, nome, cognome, ref_sede, ruolo, stipendio) as
2   select I.*
3     from impiegato I
4   where I.ref_sede in
5     (select S.cod_sede
6       from sede S
7      where S.città='Bologna'
8    );

```

Below the code, there is a 'Result Grid' tab, a 'Filter Rows' input field, and an 'Export' button.

Fig. 6.10 – Codice MySQL per l'interrogazione 5 e istanze risultanti

ESERCITAZIONE 7- SEGRETERIA

19/05/2017

Dato il seguente schema della base di dati SEGRETERIA:

STUDENTE (Sid, Nome, Cognome, Login, Età, Media)

CORSO (Cid, Cnome, Crediti, Professore)

ESAME (Stud_Id, Ref_Cid, Voto)

FK: Stud_Id REFERENCES Studente (Sid)

FK: Ref_Cid REFERENCES Corso (Cid)

CIRCOLO (Ref_Sid, Nome_Circ, Anno)

FK: Ref_Sid REFERENCES Studente (Sid)

Uno Studente (Sid, Nome) sostiene un Esame relativamente ad un Corso (Cid, Cnome) e può essere iscritto a un Circolo (Nome_Circ) (Fig. 7.1).

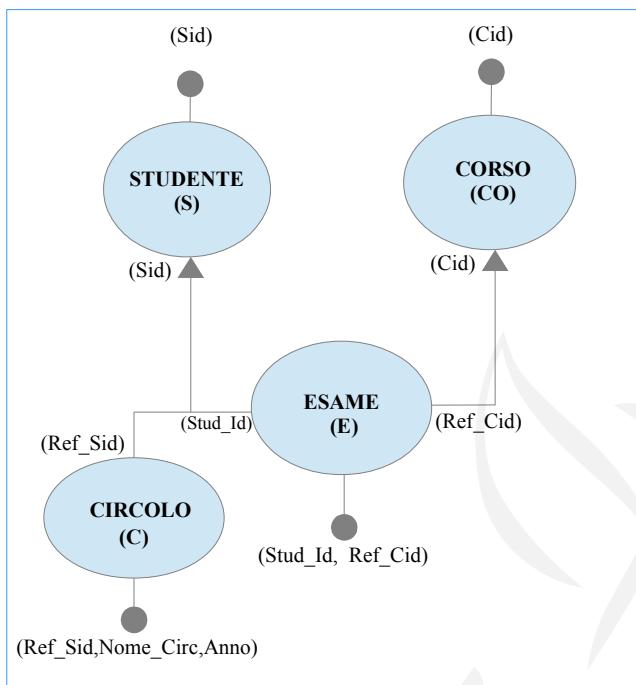


Fig. 7.1 – Modello dello schema di base di dati

Con riferimento alla fig. 7.1, scrivere le seguenti interrogazioni:

1. Creare una vista '28_studenti' per ottenere matricola, nome, cognome e nome del corso degli studenti che hanno preso 28;
2. Creare la vista 'bravi_studenti' che fornisca sid e media degli studenti che abbiano la media maggiore o uguale a 28;
3. Creare una vista 'studenti_attivi' che fornisca nome, cognome, e-mail, nome del circolo e anno di iscrizione degli studenti con media superiore a 26 che appartengono ad un circolo;

Fig. 7.2 – Codice MySQL per la creazione del database Segreteria

The screenshot shows the MySQL Workbench interface with a query editor containing the SQL code for creating the Segreteria database and its tables. The code is color-coded for syntax highlighting, and the 'Limit to 1000 rows' option is selected at the top.

```
1 -- creazione del database
2 --
3 • CREATE DATABASE IF NOT EXISTS segreteria;
4 • USE segreteria;
5 --
6 -- Definizione della tabella studente
7 --
8 • └─ CREATE TABLE studente (
9     sid char(16) NOT NULL,
10    nome varchar(70),
11    cognome varchar(70),
12    login varchar(90),
13    età int (16),
14    media int (16),
15    PRIMARY KEY (sid)
16 );
17 --
18 -- Definizione della tabella corso
19 --
20 • └─ CREATE TABLE corso (
21     cid char (16) NOT NULL,
22     cnome varchar(70),
23     crediti int (16),
24     professore varchar(70),
25     PRIMARY KEY (cid)
26 );
27 --
28 -- Definizione della tabella esame
29 --
30 • └─ CREATE TABLE esame (
31     stud_id char (16) NOT NULL,
32     ref_cid char (16) NOT NULL,
33     voto int (16) NOT NULL,
34     PRIMARY KEY (stud_id, ref_cid),
35     FOREIGN KEY (stud_id) REFERENCES studente(sid),
36     FOREIGN KEY (ref_cid) REFERENCES corso(cid),
37     CHECK (voto >= 18 AND voto <= 30)
38 );
39 --
40 -- Definizione della tabella circolo
41 --
42 • └─ CREATE TABLE circolo (
43     ref_sid char (16) NOT NULL,
44     nome_circ varchar (60) NOT NULL,
45     anno int (32) NOT NULL,
46     PRIMARY KEY (ref_sid, nome_circ, anno),
47     FOREIGN KEY (ref_sid) REFERENCES studente(sid)
48 );
```

Fig. 7.3 – Codice MySQL per la popolazione del database

The screenshot shows the MySQL Workbench interface with a query editor containing the SQL code for populating the Segreteria database with sample data. The code is color-coded for syntax highlighting, and the 'Limit to 1000 rows' option is selected at the top.

```
1 --
2 -- Popolazione del database
3 --
4 
5 • INSERT INTO studente VALUES ('001', 'Marco', 'Rossi', 'marco.rossi@unipa.it', 20, 28);
6 • INSERT INTO studente VALUES ('002', 'Giovanni', 'Polizzi', 'giovanni.polizzi@unipa.it', 23, 25);
7 • INSERT INTO studente VALUES ('003', 'Lucia', 'Fiore', 'lucia.fiore@unipa.it', 23, 30);
8 • INSERT INTO corso VALUES ('A', 'Analisi 1', 12, 'Cusimano');
9 • INSERT INTO corso VALUES ('B', 'Fisica 1', 12, 'Fazio');
10 • INSERT INTO corso VALUES ('C', 'Geometria', 6, 'Valenti');
11 • INSERT INTO esame VALUES ('001', 'A', 29);
12 • INSERT INTO esame VALUES ('001', 'B', 27);
13 • INSERT INTO esame VALUES ('002', 'B', 24);
14 • INSERT INTO esame VALUES ('002', 'C', 26);
15 • INSERT INTO esame VALUES ('003', 'C', 30);
16 • INSERT INTO circolo VALUES ('001', 'Giornalisti', 2015);
17 • INSERT INTO circolo values ('002', 'Giornalisti', 2014);
```

Fig. 7.4 – Istanze della base di dati

Result Grid Filter Rows: Search Edit:						
sid	nome	cognome	login	età	media	
001	Marco	Rossi	marco.rossi@unipa.it	20	28	
002	Giovanni	Polizzi	giovanni.polizzi@unipa.it	23	25	
003	Lucia	Fiore	lucia.fiore@unipa.it	23	30	
NULL	NULL	NULL	NULL	NULL	NULL	

Result Grid Filter Rows: Search Edit:						
stud_id	ref_cid	voto				
001	A	29				
001	B	27				
002	B	24				
002	C	26				
003	C	30				
NULL	NULL	NULL				

Result Grid Filter Rows: Search Edit:						
cid	cnome	crediti	professore			
A	Analisi 1	12	Cusimano			
B	Fisica 1	12	Fazio			
C	Geometria	6	Valenti			
NULL	NULL	NULL	NULL			

Result Grid Filter Rows: Search Edit:						
ref_sid	nome_circ	anno				
001	Giornalisti	2015				
002	Giornalisti	2014				
NULL	NULL	NULL				

1. Creare una vista '28_studenti' per ottenere matricola, nome, cognome e nome del corso degli studenti che hanno preso 28

INFORMAZIONI: Studente, Esame, Corso

OPERAZIONI: join e selezione

LOGICA: Effettuo join per ottenere una relazione con tutte le informazioni su studenti corsi ed esami e seleziono solo le tuple per le quali l'attributo voto corrisponde a 28, di queste prendo gli attributi che mi danno informazioni sullo studente e il nome del corso. Infine creo la vista sul risultato ottenuto.

MYSQL

```

1 -- 
2 • CREATE VIEW 28_studenti (Matricola, Nome, Cognome, NomeCorso) as
3     SELECT s.sid, s.nome, s.cognome, c.cnome
4         FROM studente s, esame e, corso c
5             WHERE s.sid = e.stud_id AND e.ref_cid = c.cid AND e.voto = 28
6

```



```

1 • |SELECT * FROM segreteria.28_studenti;

```

Matricola	Nome	Cognome	NomeCorso

Fig. 7.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Creare la vista 'bravi_studenti' che fornisca sid e media degli studenti che abbiano la media maggiore o uguale a 28

INFORMAZIONI: Studente

OPERAZIONI: Selezione

LOGICA: seleziono le tuple di studente per le quali l'attributo media ha valore maggiore o uguale a 28 e proietto solo l'id dello studente e la sua media. Infine creo la vista sul risultato ottenuto.

MYSQL

The screenshot shows the MySQL Workbench interface. In the top pane, a SQL editor window displays the following code:

```
1 --  
2 • CREATE VIEW bravi_studenti (Matricola_Stud, Media) as  
3     SELECT s.sid, s.media  
4     FROM studente s  
5     WHERE s.media IS NOT NULL AND s.media >= 28
```

In the bottom pane, a result grid shows the data returned by the query:

Matricola_Stud	Media
001	28
003	30

Fig. 7.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Creare una vista 'studenti_attivi' che fornisca nome, cognome, e-mail, nome del circolo e anno di iscrizione degli studenti con media superiore a 26 che appartengono ad un circolo

INFORMAZIONI: Studente,Circolo

OPERAZIONI: Join e Selezione

LOGICA: Effettuo il join fra Studente e Circolo ottenendo le informazioni sugli studenti che sono iscritti a qualche circolo. Dunque seleziono le tuple per le quali l'attributo media ha valore maggiore o uguale a 26 e proietto i dati dello studente e quelli del circolo. Infine creo la vista sul risultato ottenuto.

MYSQL

The screenshot shows the MySQL Workbench interface. In the top pane, a SQL editor window displays the following code:

```
1 --  
2 • CREATE VIEW studenti_attivi (Nome, Cognome, Email_Portale, Nome_Circolo, Anno_Iscrizione) as  
3     SELECT s.nome, s.cognome, s.login, c.nome_circ, c.anno  
4     FROM studente s, circolo c  
5     WHERE s.sid = c.ref_sid AND s.media > 26
```

In the bottom pane, a result grid shows the data returned by the query:

Nome	Cognome	Email_Portale	Nome_Circolo	Anno_Iscrizione
Marco	Rossi	marco.rossi@unipa.it	Giornalisti	2015

Fig. 7.7 – Codice MySQL per l'interrogazione 3 e istanze risultanti

ESERCITAZIONE 8 - CORSO

19/05/2017

Dato il seguente schema della base di dati CORSO:

CORSO (Cod_Corso, C_Nome, Docente, Ref_Aula)

AULA (Cod_Aula, A_Nome, Edificio)

FK: Ref_Aula REFERENCES Corso (Cod_Aula)

Corso (Cod_Corso, Nome, Docente) si svolge in una determinata Aula (Cod_Aula, A_Nome, Edificio) (Fig 8.1).

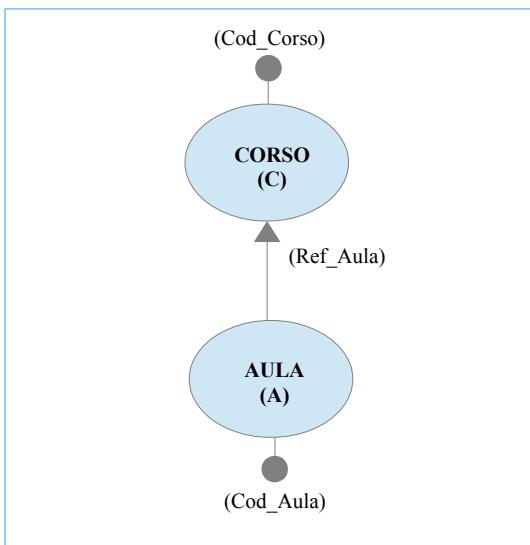


Fig. 8.1 – Modello dello schema di base di dati

Fig. 8.2 – Codice MySQL per la creazione del database Corso.

```
CREATE DATABASE IF NOT EXISTS Corso;
-- 
-- Definition of table Aula
--

CREATE TABLE aula (
    cod_aula char(3) NOT NULL,
    a_nome varchar(20) NOT NULL,
    edificio varchar(3) NOT NULL,
    PRIMARY KEY (cod_aula)
);

-- 
-- Definition of table Corso
--

CREATE TABLE corso (
    cod_corso char(3) NOT NULL,
    c_nome varchar(20) NOT NULL,
    docente varchar(20) NOT NULL,
    ref_aula char(3) NOT NULL,
    PRIMARY KEY (cod_corso),
    FOREIGN KEY (ref_aula) REFERENCES aula (cod_aula)
);
```

Fig. 8.3 – Codice MySQL per la popolazione del database

```
-- 
-- populating the DB
--

INSERT INTO aula VALUES ('001', 'Sellerio', '9');
INSERT INTO aula VALUES ('002', 'Croce', '19');
INSERT INTO aula VALUES ('003', 'Gentile', '6');
INSERT INTO aula VALUES ('004', 'Marconi', '15');
INSERT INTO corso VALUES ('C01', 'BasiDiDati', 'Sorbello', '001');
INSERT INTO corso VALUES ('P34', 'Elettrotecnica', 'Romano', '002');
INSERT INTO corso VALUES ('D27', 'Dispositivi', 'Busacca', '003');
INSERT INTO corso VALUES ('E87', 'Fisica', 'Fazio', '004');
```

Fig. 8.4 – Istanze della base di dati

	cod_aula	a_nome	edificio
001	Sellerio	9	
002	Croce	19	
003	Gentile	6	
004	Marconi	15	
HULL	HULL	HULL	

	cod_corso	c_nome	docente	ref_aula
C01	BasiDiDati	Sorbello	001	
D27	Dispositivi	Busacca	003	
E87	Fisica	Fazio	004	
P34	Elettrotecnica	Romano	002	
HULL	HULL	HULL	HULL	

Vista in MySQL:

1. Creare una vista corsisedi che contiene i nomi dei corsi, i codici e gli edifici delle aule dove si svolgono.

INFORMAZIONI: Corso, Aula

OPERAZIONI: Join

LOGICA: Con il Join tra Corso ed Aula ottengo una tabella che mi da informazioni sui corsi e le aule in cui si svolgono. Proietto solo gli attributi C_Nome, Cod_Aula, Edificio e con questi creo la vista.

CREATE VIEW corsisedi (corso, aula, edificio) AS

SELECT C.C_Nome, A.Cod_Aula, A.Edificio

FROM Corso C, Aula A

WHERE C.Ref_Aula=A.Cod_Aula

MYSQL

	corso	aula	edificio
BasiDiDati	001	9	
Elettrotecnica	002	19	
Dispositivi	003	6	
Fisica	004	15	

Fig. 8.5 – Codice MySQL per la creazione della vista e le istanze risultanti

ESERCITAZIONE 9 - FORNITURA

05/05/2017

Dato il seguente schema della base di dati FORNITURA:

NEGOZIO (Id, Nome N, Città)

PRODOTTO (Codice, Nome P, Marca)

LISTINO (Ref Negozio, Ref Prodotto, Prezzo)

EK-Ref-Negozi REFERENCES Negozio (Id)

PR: Ref_Negozio REFERENCES Negozio (Id)
EK: Ref_Prodotto REFERENCES Prodotto (Codice)

Negozio (Id, Nome, N) vende un determinato Prodotto (Codice, Nome, P) ad un certo Prezzo (Fig 9.1).

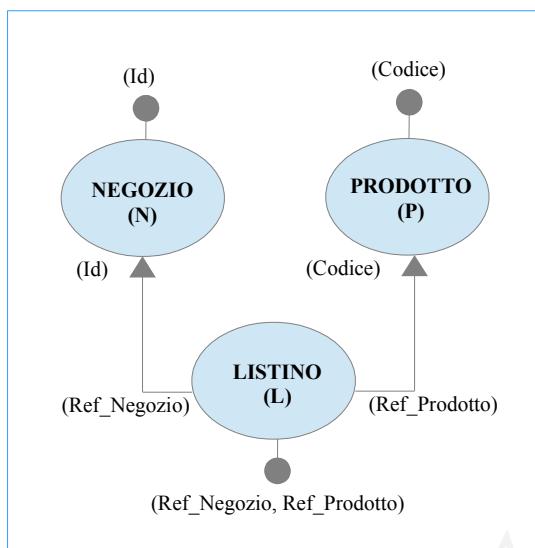


Fig. 9.1 – Modello dello schema di base di dati

Interrogazione in SQL:

1. Trovare, per ciascun prodotto, la città in cui viene venduto al prezzo più basso

SQL

INFORMAZIONI: Negozio, Listino Prodotto

OPERAZIONI: 1 interrogazione innestata con passaggio di parametri, 4 JOIN, operatore di confronto.

LOGICA: Prendiamo ogni valore all' interno della relazione P e lo passiamo come parametro all' interrogazione B, che restituirà una relazione ad una colonna in cui vi sono tutti i prezzi per ogni singolo prodotto passato dall' esterno.

SELECT DISTINCT P.Nome P, N.Citt...

A FROM Negozio N, Listino L, Prodotto P

WHERE N.Id=L.Ref Negozio AND P.Codice=L.Ref Prodotto A AND Prezzo <= ALL (SELECT Prezzo

FROM Prodotto P1. Listino L1

```
B FROM Prodotto P1, Listino L1  
WHERE P1.Nome=P.Nome AND  
L1.Prodotto=P1.Codice )
```

ESERCITAZIONE 10 - SCUOLA

06/06/2017

Dato il seguente schema della base di dati Scuola:

DOCENTE (Cf_Doc, Nome_D, Cognome_D)
STUDENTE (Cf_Stud, Nome_S, Cognome_S)
ARGOMENTO (Cod_Arg, Descrizione)
LEZIONE (Ref_Arg_Data, Ref_Doc, Num_Studenti)

AK: Data, Cf_Doc
FK: Ref_Arg REFERENCES Argomento (Cod_Arg)
FK: Ref_Doc REFERENCES Docente (Cf_Doc)

INTERROGAZIONE (R_Arg, Ref_Data, Ref_Stud, Voto)
FK: R_Arg REFERENCES Lezione (Ref_Arg)
FK: Ref_Data REFERENCES Lezione (Data)
FK: Ref_Stud REFERENCES Studente (Cf_Stud)

Il Docente (Cf_Doc) tiene una Lezione (Ref_Arg) su un Argomento (Descrizione).
Uno Studente (Cf_Stud) sostiene un' Interrogazione su un determinato Argomento (Fig 10.1).

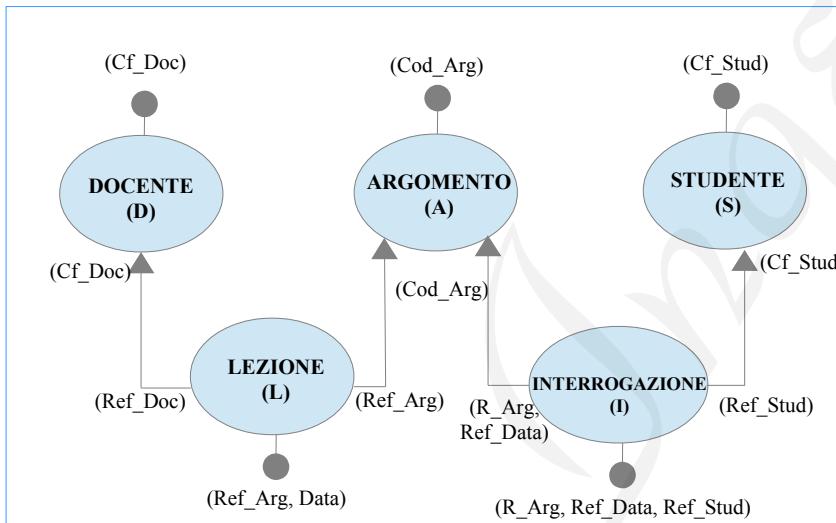


Fig. 10.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1, 2, 3) :

1. Selezionare il codice Fiscale, il nome ed il cognome degli studenti che non sono mai stati interrogati su un argomento con descrizione 'Fisica';
2. Selezionare il codice fiscale del docente che ha svolto lezione su tutti gli argomenti con descrizione 'Fisica';
3. Selezionare il codice fiscale del docente che ha sempre interrogato, cioè che durante ogni sua lezione ha fatto almeno un' interrogazione;
4. Selezionare, per ogni argomento, la media dei voti riportati dagli studenti interrogati sull'ultimo argomento in questione;
5. Selezionare, per ogni studente il codice fiscale del docente con il quale ha effettuato il maggior numero di interrogazioni;

Fig. 10.2 – Codice MySQL per la creazione del database Scuola

```

5 • CREATE DATABASE IF NOT EXISTS scuola;
6 • USE scuola;
7
8
9     -- Definition of table docente
10    --
11
12 • ┌─ CREATE TABLE docente (
13     cf_doc char(16) NOT NULL,
14     nome_d varchar(45) NOT NULL,
15     cognome_d varchar(45) NOT NULL,
16     PRIMARY KEY (cf_doc)
17 );
18
19     --
20     -- Definition of table studente
21     --
22
23 • ┌─ CREATE TABLE studente (
24     cf_stud char(16) NOT NULL,
25     nome_s varchar(45) NOT NULL,
26     cognome_s varchar(45) NOT NULL,
27     PRIMARY KEY (cf_stud)
28 );
29
30
31     -- Definition of table argomento
32     --
33
34 • ┌─ CREATE TABLE argomento (
35     cod_arg char(5) NOT NULL,
36     descrizione varchar(45) NOT NULL,
37     PRIMARY KEY (cod_arg)
38 );
39
40     --
41     -- Definition of table lezione
42     --
43
44 • ┌─ CREATE TABLE lezione (
45     cod_arg_l char(5) NOT NULL,
46     data_lezione date NOT NULL,
47     cf_doc_l char(16) NOT NULL,
48     numstudenti int(30) NOT NULL,
49     PRIMARY KEY (cod_arg_l, data_lezione),
50     FOREIGN KEY (cod_arg_l) REFERENCES argomento (cod_arg),
51     FOREIGN KEY (cf_doc_l) REFERENCES docente (cf_doc)
52 );
53
54
55     -- Definition of table interrogazione
56     --
57
58 • ┌─ CREATE TABLE interrogazione (
59     cod_arg_i char(5) NOT NULL,
60     data_lezione_i date NOT NULL,
61     cf_stud_i char(16) NOT NULL,
62     voto numeric(3,1),
63     PRIMARY KEY (cod_arg_i, data_lezione_i, cf_stud_i),
64     FOREIGN KEY (cod_arg_i, data_lezione_i) REFERENCES lezione (cod_arg_l, data_lezione),
65     FOREIGN KEY (cf_stud_i) REFERENCES studente (cf_stud)
66 );

```

Fig. 10.3 – Codice MySQL per la popolazione del database

```

2     -- populating the DB
3     --
4
5 • INSERT INTO docente VALUES
6     ('001', 'Albert', 'Einstein'),
7     ('002', 'Galileo', 'Galilei'),
8     ('003', 'Karl', 'Marx');
9
10 • INSERT INTO studente VALUES
11     ('999', 'Mario', 'Rossi'),
12     ('998', 'Maria', 'Neri'),
13     ('997', 'Piero', 'Gialli'),
14     ('996', 'Giorgio', 'Verdi');
15
16 • INSERT INTO argomento VALUES
17     ('555', 'Fisica'),
18     ('666', 'Religione'),
19     ('777', 'Fisica'),
20     ('888', 'Matematica'),
21     ('444', 'Filosofia');
22
23 • INSERT INTO lezione VALUES -- cod arg, data, cod doc, numstud
24     ('555', '1-1-1', '001', 15),
25     ('555', '2-1-1', '002', 15),
26     ('555', '3-1-1', '002', 15),
27     ('555', '4-1-1', '002', 15),
28     ('555', '5-1-1', '002', 15),
29     ('555', '6-1-1', '002', 15),
30     ('666', '7-1-1', '003', 15),
31     ('666', '8-1-1', '002', 15),
32     ('555', '9-1-1', '003', 15),
33     ('777', '1-1-1', '002', 15),
34     ('888', '2-1-1', '002', 15),
35     ('444', '2-1-1', '003', 15);
36
37 • INSERT INTO interrogazione VALUES -- cod arg, data, cod stud, voto
38     ('555', '1-1-1', '999', 10),
39     ('555', '2-1-1', '999', 6),
40     ('555', '3-1-1', '998', 6),
41     ('555', '4-1-1', '999', 6),
42     ('555', '5-1-1', '999', 6),
43     ('555', '6-1-1', '998', 8),
44     ('666', '7-1-1', '998', 6),
45     ('666', '8-1-1', '998', 5),
46     ('555', '9-1-1', '997', 6),
47     ('777', '1-1-1', '999', 4),
48     ('888', '2-1-1', '996', 4),
49     ('444', '2-1-1', '998', 7);

```

Fig. 10.4 – Istanze della base di dati

The screenshot shows a database interface with five tables displayed in separate windows:

- docente:** Columns: cf_doc, nome_d, cognome_d. Data: (001, Albert, Einstein), (002, Galileo, Galilei), (003, Karl, Marx).
- studente:** Columns: cf_stud, nome_s, cognome_s. Data: (996, Giordio, Verdi), (997, Piero, Gialli), (998, Maria, Neri), (999, Mario, Rossi).
- argomento:** Columns: cod_arg, descrizione. Data: (444, Filosofia), (555, Fisica), (666, Religione), (777, Fisica), (888, Matematica).
- lezione:** Columns: cod_arg_l, data_lezione, cf_doc_l, numstudenti. Data: (444, 0002-01-01, 003, 15), (555, 0001-01-01, 001, 15), (555, 0002-01-01, 002, 15), (555, 0003-01-01, 002, 15), (555, 0004-01-01, 002, 15), (555, 0005-01-01, 002, 15), (555, 0006-01-01, 002, 15), (555, 0009-01-01, 003, 15), (666, 0007-01-01, 003, 15), (666, 0008-01-01, 002, 15), (777, 0001-01-01, 002, 15), (888, 0002-01-01, 002, 15).
- interrogazione:** Columns: cod_arg_j, data_lezione_j, cf_stud_j, voto. Data: (444, 0002-01-01, 998, 7.0), (555, 0001-01-01, 999, 10.0), (555, 0002-01-01, 999, 6.0), (555, 0003-01-01, 998, 6.0), (555, 0004-01-01, 999, 6.0), (555, 0005-01-01, 999, 6.0), (555, 0006-01-01, 998, 8.0), (555, 0009-01-01, 997, 6.0), (666, 0007-01-01, 998, 6.0), (666, 0008-01-01, 998, 5.0), (777, 0001-01-01, 999, 4.0), (888, 0002-01-01, 996, 4.0).

1. Selezionare il codice fiscale, il nome ed il cognome degli studenti che non sono mai stati interrogati su un argomento con descrizione 'Fisica'.

SQL

INFORMAZIONI: Studente, Argomento

OPERAZIONI: 1 interrogazione innestata, operatore insiemistico NOT IN, Join, Selezione

LOGICA: L'interrogazione B mi restituisce il riferimento degli studenti che hanno sostenuto almeno un'interrogazione con descrizione 'Fisica'. L'interrogazione A seleziona il codice fiscale degli studenti che non si trovano tra i risultati dell'interrogazione B.

```
A   SELECT *
  FROM Studente S
 WHERE S.Cf_Stud NOT IN (SELECT I.Ref_Stud
                           FROM Argomento A
                           WHERE A.Cod_Arg=I.R_Arg AND A.Descrizione='Fisica')
      B
```

ALGEBRA

INFORMAZIONI: Studente, Argomento

OPERAZIONI: Differenza, Proiezione, Selezione, Join

LOGICA: Con i join ottengo le informazioni su tutti gli studenti il cui codice è presente almeno una volta fra i valori dell'attributo R_Arg di interrogazione. Tramite la selezione prendo solo quelli che hanno come descrizione fisica e ne proietto solo codice fiscale, nome e cognome degli studenti. Alla relazione Studente sottraggo questo risultato e ottengo gli studenti che non sono mai stati interrogati su un argomento con descrizione 'Fisica'.

```
S = (PROJCf_Stud, Nome_S, Cognome_S(SELDescrizione='Fisica'((S JOINS.Cf_Stud=I.Ref_StudI) JOINR_Arg=L.Ref_Arg AND Ref_Data=L.DataL)
      JOINRef_Arg=A.Cod_Arg_A)) )
```

MYSQL

```

1 •   SELECT
2      *
3  FROM
4      studente s
5  WHERE
6      NOT EXISTS( SELECT
7          *
8      FROM
9          interrogazione i,
10         argomento a
11     WHERE
12         s.cf_stud = i.cf_stud_i
13         AND a.cod_arg = i.cod_arg_i
14         AND a.descrizione = 'Fisica');
15

```

Result Grid | Filter Rows: | Edit: | |

cf_stud	nome_s	cognome_s
996	Gioraio	Verdi
HULL	HULL	HULL

Fig. 10.5 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

2. Selezionare il codice fiscale del docente che ha svolto lezione su tutti gli argomenti con descrizione 'Fisica'.

SQL

INFORMAZIONI: Lezione, Argomento

OPERAZIONI: 2 operazioni innestate con passaggio di parametri, operatore insiemistico NOT EXISTS, doppia negazione, selezione, join

LOGICA: Attraverso l'interrogazione ottengo il codice dei docenti che hanno tenuto almeno una lezione, di questi seleziono, tramite l'interrogazione B quelli che hanno svolto lezioni sull'argomento con descrizione 'Fisica', tramite l'interrogazione A trovo il codice fiscale dei docenti che hanno che ha svolto lezione su tutti gli argomenti con descrizione 'Fisica'.

A **SELECT DISTINCT L. Ref_Doc**
FROM Lezione L
WHERE NOT EXISTS (SELECT *

B **FROM Argomento A**
WHERE A.Descrizione='Fisica' NOT EXISTS (SELECT *

C **FROM Lezione L1**
WHERE A.Cod_Arg=L1.Ref_Arg AND
L1.Ref_Doc=L.Ref_Doc)

ALGEBRA

INFORMAZIONI: Lezione, Argomento

OPERAZIONI: Divisione, Proiezione, Ridenominazione e Selezione

LOGICA: Con il secondo membro della divisione seleziono gli argomenti con descrizione Fisica, ridenomino il codice dell'argomento e proietto il riferimento dell'argomento in modo da poter effettuare la divisione con la relazione Lezione.

(PROJ_{Ref_Arg, Ref_Doc} L)

÷

(PROJ_{Ref_Arg} (REN_{Ref_Arg ← Cod_Arg} (SEL_{Descrizione='Fisica'} A)))

MYSQL

```

1 •   SELECT DISTINCT
2      l1.cf_doc_1
3  FROM
4      lezione l1
5  WHERE
6      NOT EXISTS( SELECT
7          *
8      FROM
9          argomento a
10     WHERE
11         a.descrizione = 'Fisica'
12         AND NOT EXISTS( SELECT
13             *
14             FROM
15                 lezione l2
16             WHERE
17                 l1.cf_doc_1 = l2.cf_doc_1
18                 AND a.cod_arg = l2.cod_arg_1));
19

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

cf_doc_1
002

Fig. 10.6 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

3. Selezionare il codice fiscale del docente che ha sempre interrogato, cioè che durante ogni sua lezione ha fatto almeno un'interrogazione.

SQL

INFORMAZIONI: Lezione, Interrogazione

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, join, operatore insiemistico NOT EXISTS, doppia negazione
LOGICA: Con l'interrogazione C ottengo i docenti che hanno svolto lezioni con i relativi argomenti e date, con l'interrogazione B prendo soltanto quelli che hanno interrogato e con l'interrogazione A prendo i docenti che hanno fatto almeno un'interrogazione durante ogni sua lezione

```

A   SELECT DISTINCT L. Ref_Doc
    FROM Lezione L
    WHERE NOT EXISTS (SELECT *
                      B   FROM Interrogazione I
                      WHERE NOT EXISTS (SELECT *
                                         C   FROM Lezione L1
                                         WHERE L1.Ref_Doc = L.Ref_Doc AND L1.Ref_Arg = I.R_Arg AND
                                              L1.Ref_Data = I.Data )
                               )
  
```

ALGEBRA

INFORMAZIONI: Lezione, Interrogazione

OPERAZIONI: Divisione, Ridenominazione, Proiezione

LOGICA: Con il secondo membro della divisione ridenomino gli attributi R_Arg e Ref_Data della relazione Interrogazione e li proietto potendo così effettuare a divisione con la relazione Lezione, ottengo le informazioni di tutti i docenti che hanno sempre interrogato ad ogni loro lezione.

(PROJ_{Ref_Arg, Data, Ref_Doc} L)

⋮

(PROJ_{Ref_Arg, Data} (REN_{Ref_Arg, Data ↪ R_Arg, Ref_Data} I))

```

1 •   SELECT DISTINCT
2      11.cf_doc_1
3      FROM
4        lezione 11
5      WHERE
6        NOT EXISTS( SELECT
7                      *
8                      FROM
9                        lezione 12
10                     WHERE
11                       11.cf_doc_1 = 12.cf_doc_1
12                       AND NOT EXISTS( SELECT
13                           *
14                           FROM
15                             interrogazione i
16                           WHERE
17                             12.data_lezione = i.data_lezione_i
18                             AND 12.cod_arg_1 = i.cod_arg_i));
  
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
cf_doc_1			
001			
002			
003			

Fig. 10.7 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

4. Selezionare, per ogni argomento, la media dei voti riportati dagli studenti interrogati sull'ultimo argomento, considerando gli studenti che sono stati interrogati almeno tre volte sull'argomento in questione.

INFORMAZIONI: Interrogazione

OPERAZIONI: 1 interrogazione innestata, operatori COUNT, DISTINCT, AVG, <=, having, group by, join

LOGICA: Tramite l'interrogazione A suddivido la relazione Interrogazione in tante sottotabelle quanti sono i valori diversi di R_Arg. Con l'interrogazione B ottengo tramite l'operatore aggregato count il numero di tuple presenti ottenute dal join con passaggio di parametri tra I e II. Attraverso l'operatore di confronto <= seleziono nell'interrogazione A solo i valori le cui sottotabelle hanno un numero di tuple maggiore di 3.

```

A   SELECT DISTINCT I.R_Arg, AVG(Voto)
    FROM Interrogazione I
    GROUP BY R_Arg
    HAVING 3 <= (SELECT COUNT (*)
      B   FROM Interrogazione I1
          WHERE L1.Ref_Arg=L.Ref_Arg AND L1.Ref_Stud=L.Ref_Stud)

```

MySQL

```

1 •  SELECT
2      i1.cod_arg_i, AVG(i1.voto)
3  FROM
4      interrogazione i1
5  WHERE
6      3 <= (SELECT
7          COUNT(*)
8          FROM
9              interrogazione i2
10             WHERE
11                 i2.cod_arg_i = i1.cod_arg_i
12                     AND i2.cf_stud_i = i1.cf_stud_i)
13  GROUP BY i1.cod_arg_i;

```

Result Grid	
cod_arg_i	AVG(i1.voto)
555	7.00000

Fig. 10.8 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

5. Selezionare, per ogni studente il codice fiscale del docente con il quale ha effettuato il maggior numero di interrogazioni.

INFORMAZIONI: Lezione, Interrogazione

OPERAZIONI: 1 interrogazione innestata, group by, operatori count >=ALL, join

LOGICA: Tramite l'interrogazione A suddivido le relazioni Lezione e Interrogazione in tante sottotabelle quante sono le diverse tuple della coppia (Ref_Doc, Ref_Stud). Con l'interrogazione B faccio la stessa suddivisione in modo da ottenere un confronto tramite l'operatore >=ALL. Così ottengo il codice fiscale dei docenti che hanno effettuato il maggior numero di interrogazioni.

```

A   SELECT L.Ref_Doc
    FROM Lezione L, Interrogazione I
    WHERE L.Ref_Arg = I.R_Arg AND L.Data = I.Ref_Data
    GROUP BY L.Ref_Doc, I.Ref_Stud
    HAVING COUNT (*) >= ALL (SELECT COUNT (*)
      B   FROM Lezione L1, Interrogazione I1
          WHERE L1.Ref_Arg = I1.R_Arg AND L1.Data = I1.Ref_Data
          GROUP BY L1.Ref_Doc, I1.Ref_Stud )

```

MySQL

```

1 •  SELECT
2      i1.cf_stud_i, l1.cf_doc_1
3  FROM
4      interrogazione i1,
5      lezione l1
6  WHERE
7      i1.cod_arg_i = l1.cod_arg_1
8          AND i1.data_lezione_i = l1.data_lezione
9  GROUP BY i1.cf_stud_i , l1.cf_doc_1
10 HAVING COUNT (*) >= ALL (SELECT
11     COUNT(*)
12     FROM
13         lezione l2,
14         interrogazione i2
15     WHERE
16         l2.cod_arg_1 = i2.cod_arg_i
17             AND l2.data_lezione = i2.data_lezione_i
18                 AND i1.cf_stud_i = i2.cf_stud_i
19  GROUP BY i2.cf_stud_i , l2.cf_doc_1);

```

Result Grid	
cf_stud_i	cf_doc_1
996	002
997	003
998	002
999	002

Fig. 10.9 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

ESERCITAZIONE 11 - CICLISMO

03/06/2017

Dato il seguente schema della base di dati CICLISMO:

CICLISTA (nome_ciclista, nazionalita, eta)

GARA (nome_corsa, anno, partenza, arrivo)

PARTECIPA (nome_ciclista_p, nome_corsa_p, anno_p, posizione)

FK: nome_ciclista_p REFERENCES Gara (nome_ciclista)

FK: nome_corsa_p,anno_p REFERENCES Gara (nome_corsa, anno)

Ciclista (nome_ciclista) partecipa ad una determinata Gara (nome_corsa,anno) classificandosi in una certa posizione (Fig. 11.1).

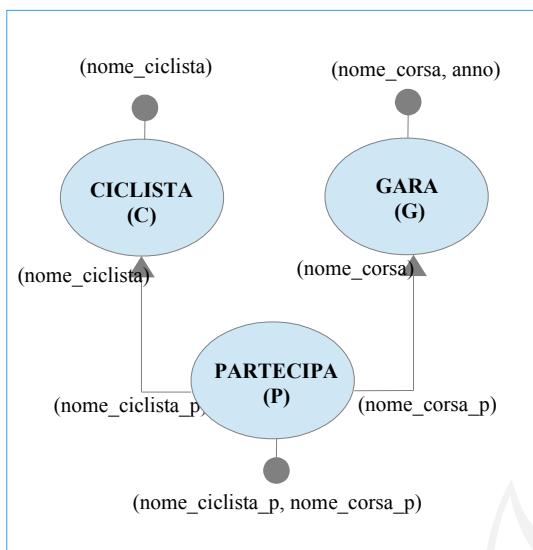


Fig. 11.1 – Modello dello schema di base di dati

Interrogazioni (Algebra relazionale 1, 2, 3):

1. Selezionare i ciclisti che si sono classificati in prima posizione in una gara ciclistica partita da Milano;
2. Selezionare i ciclisti che non si sono mai ritirati al Giro (corsa con nome Giro);
3. Selezionare le corse per le quali in ogni edizione c'è stato almeno un ritirato;
4. Selezionare ,per ogni corsa ciclistica, l'anno in cui c'è stato il maggior numero di ritirati;

Figura 11.2- Codice MYSQL per la creazione del database CICLISMO

```

1
2 -- Create schema ciclismo
3 --
4
5 • CREATE DATABASE IF NOT EXISTS ciclismo;
6 • USE ciclismo;
7
8 --
9 -- Definition of table ciclista
10 --
11
12 • CREATE TABLE ciclista (
13     nome_ciclista varchar(75) NOT NULL,
14     nazionalita char(75) NOT NULL,
15     eta char(2) NOT NULL,
16     PRIMARY KEY (nome_ciclista)
17 );
18

22
23 • CREATE TABLE gara (
24     nome_corsa varchar(45) NOT NULL,
25     anno int(4) NOT NULL,
26     partenza varchar(45) NOT NULL,
27     arrivo varchar(45) NOT NULL,
28     PRIMARY KEY (nome_corsa, anno)
29 );
30
31 --
32 -- Definition of table partecipa
33 --
34
35 • CREATE TABLE partecipa (
36     nome_corsa_p varchar(45) NOT NULL,
37     anno_p int(4) NOT NULL,
38     nome_ciclista_p varchar(75) NOT NULL,
39     posizione char,
40     PRIMARY KEY (nome_corsa_p, anno_p, nome_ciclista_p),
41     FOREIGN KEY (nome_corsa_p, anno_p) REFERENCES gara(nome_corsa, anno),
42     FOREIGN KEY (nome_ciclista_p) REFERENCES ciclista(nome_ciclista)
43 );

```

Figura 11.3 – Codice MYSQL per la popolazione del database

```

15
14 • INSERT INTO gara VALUES
15     ('Giro d_Italia', 2002, 'Milano', 'Palermo'),
16     ('Giro d_Italia', 2003, 'Milano', 'Torino'),
17     ('Giro d_Italia', 2004, 'Milano', 'Palermo'),
18     ('Giro d_Italia', 2005, 'Napoli', 'Trento'),
19     ('Giro d_Italia', 2006, 'Aosta', 'Scilla'),
20     ('Giro d_Italia', 2007, 'Milano', 'Salerno'),
21     ('Tour de France', 2004, 'Parigi', 'Strasburgo'),
22     ('Tour de France', 2006, 'Marsiglia', 'Versaille'),
23     ('Tour de France', 2008, 'Parigi', 'Lione'),
24     ('Tour de France', 2010, 'Parigi', 'Marsiglia'),
25     ('Tour de France', 2012, 'Nyon', 'Versaille');

-- 26
27 • INSERT INTO partecipa VALUES
28     ('Giro d_Italia', 2002, 'Rossi', '1'),
29     ('Giro d_Italia', 2002, 'Gialli', 'R'),
30     ('Giro d_Italia', 2002, 'Hamilton', 'R'),
31     ('Giro d_Italia', 2003, 'Williams', '1'),
32     ('Giro d_Italia', 2004, 'Rossi', '1'),
33     ('Giro d_Italia', 2004, 'Bernabeu', '2'),
34     ('Giro d_Italia', 2005, 'Neri', '1'),
35     ('Giro d_Italia', 2006, 'Federer', '1'),
36     ('Giro d_Italia', 2006, 'Gialli', '2'),
37     ('Giro d_Italia', 2006, 'Hamilton', '3'),
38     ('Giro d_Italia', 2006, 'Williams', '4'),
39     ('Giro d_Italia', 2007, 'Williams', '1'),
40     ('Giro d_Italia', 2007, 'Federer', 'R'),
41     ('Tour de France', 2004, 'Rossi', '1'),
42     ('Tour de France', 2006, 'Neri', '1'),
43     ('Tour de France', 2008, 'Rossi', '1'),
44     ('Tour de France', 2010, 'Rossi', '1'),
45     ('Tour de France', 2012, 'Rossi', '1');

1
2 -- populating the DB
3 --
4
5 • INSERT INTO ciclista VALUES
6     ('Rossi', 'Italia', '20'),
7     ('Gialli', 'Italia', '30'),
8     ('Neri', 'Italia', '50'),
9     ('Federer', 'Svizzera', '20'),
10    ('Williams', 'America', '19'),
11    ('Hamilton', 'Inghilterra', '24'),
12    ('Bernabeu', 'Spagna', '63');

```

Fig. 11.4 – Istanze del database

nome_corsa	anno	partenza	arrivo
Giro d Italia	2002	Milano	Palermo
Giro d Italia	2003	Milano	Torino
Giro d Italia	2004	Milano	Palermo
Giro d Italia	2005	Napoli	Trento
Giro d Italia	2006	Aosta	Scilla
Giro d Italia	2007	Milano	Salerno
Tour de France	2004	Parigi	Strasburgo
Tour de France	2006	Marsiglia	Versailles
Tour de France	2008	Parigi	Lione
Tour de France	2010	Parigi	Marsiglia
Tour de France	2012	Nyon	Versailles
NULL	NULL	NULL	NULL

nome_ciclista	nazionalita	eta
Bernabeu	Spaagna	63
Federer	Svizzera	20
Gialli	Italia	30
Hamilton	Inghilterra	24
Neri	Italia	50
Rossi	Italia	20
Williams	America	19
NULL	NULL	NULL

nome_corsa_p	anno_p	nome_ciclista_p	posizione
Giro d Italia	2002	Gialli	R
Giro d Italia	2002	Hamilton	R
Giro d Italia	2002	Rossi	1
Giro d Italia	2003	Williams	1
Giro d Italia	2004	Bernabeu	2
Giro d Italia	2004	Rossi	1
Giro d Italia	2005	Neri	1
Giro d Italia	2006	Federer	1
Giro d Italia	2006	Gialli	2
Giro d Italia	2006	Hamilton	3
Giro d Italia	2006	Williams	4
Giro d Italia	2007	Federer	R
Giro d Italia	2007	Williams	1
Tour de France	2004	Rossi	1
Tour de France	2006	Neri	1
Tour de France	2008	Rossi	1
Tour de France	2010	Rossi	1
Tour de France	2012	Rossi	1
NULL	NULL	NULL	NULL

1. Selezionare i ciclisti che si sono classificati in prima posizione in una gara ciclistica partita da Milano;

SQL

INFORMAZIONI: Ciclista, Partecipa, Gara

OPERAZIONI: join e selezione

LOGICA: con il join fra le tre relazioni ottengo una relazione con tutte le informazioni sui ciclisti e sulle gare a cui essi hanno partecipato, infine seleziono solo le tuple per le quali l'attributo partenza='Milano' e l'attributo posizione=1

SELECT DISTINCT C.*

FROM Ciclista C, Partecipa P, Gara G

WHERE C.nome_ciclista = P.nome_ciclista_p AND P.nome_corsa_p = G.nome_corsa AND P.anno_p = G.anno AND G.partenza='Milano' AND P.posizione='1'

ALGEBRA

INFORMAZIONI: Ciclista, Partecipa, Gara

OPERAZIONI: join, selezione e proiezione

LOGICA:(Vedi SQL)

PROJ_{nome_ciclista, nazionalita, eta} (SEL_{partenza='Milano' AND posizione='1'} ((C JOIN_{C.nome_ciclista=P.nome_ciclista_p} P) JOIN_{nome_corsa_p=G.nome_corsa AND anno_p=G.anno} G))

MYSQL

```

1 -- a) selezionare i ciclisti che si sono classificati in prima
2 -- posizione in una gara ciclistica partita da Milano
3
4 • SELECT DISTINCT c.*
5   FROM ciclista c, partecipa p, gara g
6   WHERE c.nome_ciclista=p.nome_ciclista_p AND p.nome_corso_p=g.nome_corso AND p.anno_p=g.anno
7   AND g.partenza='Milano' AND p.posizione='1';

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	nome_ciclista	nazionalita	eta
	Rossi	Italia	20
	Williams	America	19

Fig. 11.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare i ciclisti che non si sono mai ritirati al Giro (corsa con nome Giro);

SQL

INFORMAZIONI: Ciclista, Partecipa

OPERAZIONI: interrogazione innestata, selezione ed operatore insiemistico not in

LOGICA: Con l'interrogazione B ottengo i nomi dei ciclisti che hanno partecipato al Giro e che si sono ritirati, con l'interrogazione A seleziono i nomi dei ciclisti che non compaiono fra i risultati della B e ne proietto i dati.

```

A  SELECT *
  FROM Ciclista C
 WHERE C.nome_ciclista NOT IN (SELECT P.nome_ciclista_p
                                B  FROM Partecipa P
                                 WHERE P.nome_corso_p='Giro' AND P.posizione='R')

```

ALGEBRA

INFORMAZIONI: Ciclista, Partecipa

OPERAZIONI: join, selezione ,proiezione, differenza

LOGICA: A differenza dell'SQL utilizzo l'operatore insiemistico differenza e sottraggo a tutti i ciclisti quelli che hanno partecipato al Giro e la cui posizione è risultata R

$(\text{PROJ}_{\text{nome_ciclista_p}} \text{P}) - (\text{PROJ}_{\text{nome_ciclista_p}} (\text{SEL}_{\text{nome_corso_p}=\text{'Giro'} \text{AND} \text{posizione}=\text{'R'}} \text{P}))$

MYSQL

```

1 -- b) selezionare il nome dei ciclisti che non si sono mai ritirati al Giro
2 -- (corsa con nome Giro)
3
4 • SELECT
5   *
6   FROM
7     ciclista c
8   WHERE
9     c.nome_ciclista NOT IN (SELECT
10       p.nome_ciclista_p
11       FROM
12         partecipa p
13       WHERE
14         p.nome_corso_p = 'Giro d'Italia'
15         AND p.posizione = 'R');

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	nome_ciclista	nazionalita	eta
	Bernabeu	Saona	63
	Neri	Italia	50
	Rossi	Italia	20
	Williams	America	19
	NULL	NULL	NULL

Fig. 11.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare tutte le corse per le quali in ogni edizione c'è stato almeno un ritirato.

SQL

INFORMAZIONI: Gara, Partecipa

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, selezione e not exists

LOGICA: Con l'interrogazione B ottengo i nomi dei ciclisti che hanno partecipato al Giro e che si sono ritirati, con l'interrogazione A seleziono i nomi dei ciclisti che non compaiono fra i risultati della B e ne proietto i dati.

A **SELECT ***
 FROM GARA G1

WHERE NOT EXISTS (SELECT *

FROM GARA G2

WHERE G2.nome_corsa=G1.nome_corsa AND NOT EXISTS(SELECT *

FROM Partecipa P

WHERE P.anno_p = G2.anno AND
 P.Posizione = 'R'))

ALGEBRA

INFORMAZIONI: Gara, Partecipa

OPERAZIONI: join, selezione ,proiezione, differenza

LOGICA: A differenza dell'SQL utilizzo l'operatore insiemistico differenza e sottraggo a tutte le gare quelle che hanno avuto almeno un'edizione senza ritirati.

(**PROJ** nome_corsa(G)) - (**PROJ** nome_corsa((**PROJ** nome_corsa, anno G)-(**PROJ** nome_corsa, anno (**SEL** posizione=R(**REN** nome_corsa, anno<- nome_corsa_p, anno_p P)))))

MYSQL

-- c) selezionare le corse per le quali in ogni edizione c'è stato almeno un ritirato

```
1
2
3 • SELECT
4     g1.nome_corsa
5   FROM
6     gara g1
7   WHERE
8     NOT EXISTS( SELECT
9         *
10        FROM
11        gara g2
12      WHERE
13        g1.nome_corsa = g2.nome_corsa
14        AND NOT EXISTS( SELECT
15            *
16            FROM
17            partecipa p
18          WHERE
19            g2.nome_corsa = p.nome_corsa_p
20            AND g2.anno = p.anno_p
21            AND p.posizione = 'R'));
```

Fig. 11.7 – Codice MySQL per l'interrogazione 3 (nessuna istanza risultante)

4. Selezionare per ogni corsa ciclista, l'anno in cui c'è stato il maggior numero di ritirati

SQL

INFORMAZIONI: Partecipa

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, group by ,selezione e operatore di confronto.

LOGICA: Con l'interrogazione B ottengo il conteggio delle tuple di ogni sottotabella , ottenuta dalla tabella P2 per ogni valore diverso dell'attributo anno , per ogni nome della corsa passato come parametro. Dunque con l'interrogazione A seleziono quei raggruppamenti, ottenuti per tutti i valori diversi delle coppie (nome_corsa_p, anno_p) , il cui conteggio delle tuple risulta maggiore di tutti i valori ottenuti dall'interrogazione B. Ripeto l'interrogazione B per ogni valore diverso dell'attributo nome_corsa_p.

A **SELECT P1.nome_corsa_p, P1.anno_p**

FROM PARTECIPA P1

WHERE P1.Posizione= 'R'

GROUPBY P1.nome_corsa_p, P1.anno_p

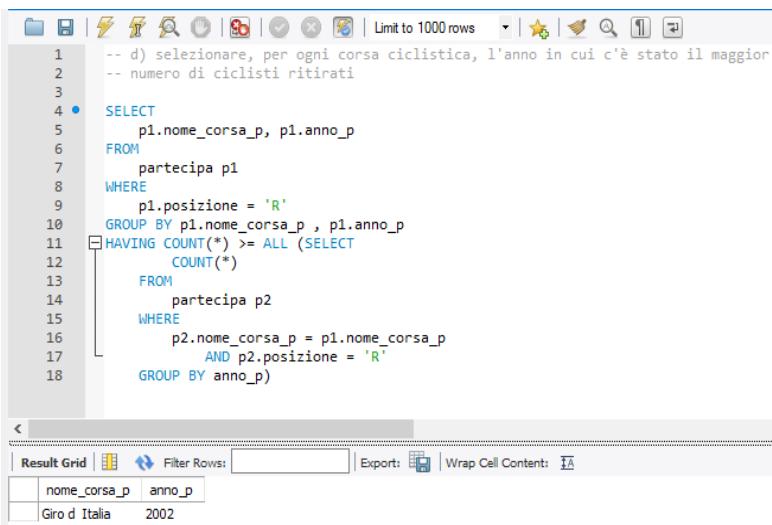
HAVING COUNT(*) >= ALL (SELECT COUNT(*)

FROM PARTECIPA P2

WHERE P2.nome_corsa_p = P1.nome_corsa_p

GROUPBY P2.anno)

MYSQL



The screenshot shows the MySQL Workbench interface. In the top panel, there is a code editor containing the following SQL query:

```
1 -- d) selezionare, per ogni corsa ciclistica, l'anno in cui c'è stato il maggior
2 -- numero di ciclisti ritirati
3
4 • SELECT
5     p1.nome_corsa_p, p1.anno_p
6     FROM
7         partecipa p1
8     WHERE
9         p1.posizione = 'R'
10    GROUP BY p1.nome_corsa_p , p1.anno_p
11    HAVING COUNT(*) >= ALL (SELECT
12        COUNT(*)
13        FROM
14        partecipa p2
15        WHERE
16            p2.nome_corsa_p = p1.nome_corsa_p
17            AND p2.posizione = 'R'
18        GROUP BY anno_p)
```

In the bottom panel, there is a results grid with the following data:

nome_corsa_p	anno_p
Giro d'Italia	2002

Fig. 11.8 – Codice MySQL per l'interrogazione 4 e istanze risultanti

ESERCITAZIONE 12- MUSEO

06/06/2017

Dato il seguente schema della base di dati MUSEO:

QUADRO (CQ, Autore, Periodo)

MOSTRA (CM, Nome, Anno , Organizzatore)

ESPONE (Ref_CQ,Ref_CM, Sala)

FK: Ref_CQ REFERENCES Quadro (CQ)

FK: Ref_CM REFERENCES Mostra (CM)

Quadro (CQ,Autore) è esposto in una determinata Mostra(CM, Nome) in una sala ben precisa (Fig 12.1).

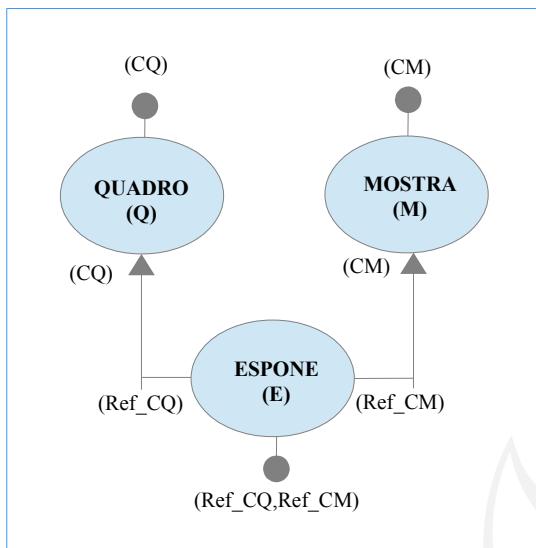


Fig. 12.1 – Modello dello schema di base di dati

Interrogazioni (Algebra relazionale 1,2,3)

1. Selezionare le sale nelle quali è stato esposto nell'anno 1997 un quadro di Picasso;
2. Selezionare tutti i dati dei quadri di Picasso che non sono mai stati esposti nel 1997;
3. 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;
4. Selezionare tutti i dati delle mostre in cui sono stati esposti quadri di almeno 5 autori distinti ;
5. Selezionare , per ogni mostra, l'autore di cui si esponevano il maggior numero di quadri;

Figura 12.2 – Codice MYSQL per la creazione del database MUSEO

```
1  --
2  -- Create schema museo
3  --
4
5 • CREATE DATABASE IF NOT EXISTS museo;
6 • USE museo;
7
8  --
9  -- Definition of table quadro
10 --
11
12 • □ CREATE TABLE quadro (
13   cq char(5) NOT NULL,
14   autore varchar(75) NOT NULL,
15   periodo varchar(16) NOT NULL,
16   PRIMARY KEY (cq)
17 );
18
19
20  --
21  -- Definition of table mostra
22  --
23
24 • □ CREATE TABLE mostra (
25   cm char(5) NOT NULL,
26   nome varchar(75) NOT NULL,
27   anno int(4) NOT NULL,
28   organizzatore varchar(75) NOT NULL,
29   PRIMARY KEY (cm)
30 );
31
32  --
33  -- Definition of table espone
34
35
36 • □ CREATE TABLE espone (
37   ref_cm char(5) NOT NULL,
38   ref_cq char(5) NOT NULL,
39   sala int(3) NOT NULL,
40   PRIMARY KEY (ref_cm, ref_cq),
41   FOREIGN KEY (ref_cm) REFERENCES mostra(cm),
42   FOREIGN KEY (ref_cq) REFERENCES quadro(cq)
43 );
44
```

Figura 12.3 – Codice MYSQL per la popolazione del database

```
1
2
3  --
4  -- populating the DB
5  --
6
7 • INSERT INTO mostra (cm,nome,anno,organizzatore)
8   VALUES ('01','AAA',1997,'P'),
9   ('02','BBB',2001,'S'),
10  ('03','CCC',2222,'AS');
11
12 • INSERT INTO quadro (cq,autore,periodo)
13   VALUES ('01','Picasso','1'),
14   ('02','Picasso','2'),
15   ('03','Neruda','321'),
16   ('04','Picasso','0'),
17   ('05','Giuseppe','12'),
18   ('06','Neruda','1');
19
20 • INSERT INTO espone (ref_cm, ref_cq, sala)
21   VALUES ('01','01',1),
22   ('01','02',1),
23   ('01','03',1),
24   ('01','05',2),
25   ('02','01',1),
26   ('02','03',1),
27   ('02','02',1),
28   ('03','03',1),
29   ('03','04',1);
```

Figura 12.4- Istanze della base di dati

Result Grid			Result Grid				Result Grid		
cq	autore	periodo	cm	nome	anno	organizzatore	ref_cm	ref_cq	sala
01	Picasso	1	01	AAA	1997	P	01	01	1
02	Picasso	2	02	BBB	2001	S	01	02	1
03	Neruda	321	03	CCC	2222	AS	01	03	1
04	Picasso	0	NULL	NULL	NULL	NULL	01	05	2
05	Giuseppe	12					02	01	1
06	Neruda	1					02	02	1
NULL	NULL	NULL					02	03	1
							03	03	1
							03	04	1
							NULL	NULL	NULL

1. Selezionare le sale nelle quali è stato esposto nell'anno 1997 un quadro di Picasso.

SQL

INFORMAZIONI: Quadro,Mostra, Espone.

OPERAZIONI: join e selezione

LOGICA: con il join fra le tre relazioni ottengo una relazione con tutte le informazioni sulle mostre e sui quadri che in esse sono stati esposti, infine seleziono solo le tuple per le quali l'attributo Autore='Picasso' e l'attributo anno='1997'

SELECT DISTINCT E.Sala

FROM Quadro Q, Mostra M, Espone E

WHERE Q.CQ=E.Ref_CQ AND M.CM=E.Ref_CM AND Q.Autore='Picasso' AND M.Anno=1997

ALGEBRA

INFORMAZIONI: Quadro, Mostra, Espone

OPERAZIONI: join, selezione e proiezione

LOGICA:(Vedi SQL)

PROJ_{Sala}(SEL_{Anno=1997 AND Autore='Picasso'}.((Q JOIN_{Q.CQ=E.Ref_CQ} E) JOIN_{Ref_CM=M.CM} M))

MYSQL

Result Grid			Filter Rows:		Search	Export:
sala						
1						

Figura 12.5- Codice MYSQL per l'interrogazione 1 e relative istanze

2. Selezionare tutti i dati dei quadri di Picasso che non sono mai stati esposti nell'anno 1997

SQL

INFORMAZIONI: Quadro,Mostra, Espone.

OPERAZIONI: interrogazione innestate, join, selezione, operatore insiemistico NOT IN

LOGICA: Con l'interrogazione B ottengo i codici dei quadri che sono stati esposti nell'anno 1997, con l'interrogazione A seleziono i quadri di Picasso i cui codici non si trovano fra i risultati dell'interrogazione B, dunque ne proietto tutte le informazioni.

A **SELECT ***
 FROM Quadro Q
 WHERE Q.Autore='Picasso' AND Q.CQ NOT IN (SELECT E.Ref_CQ

B **FROM Mostra M, ESPONE E**
 WHERE M.CM= E.Ref_CM AND M.Anno=1997)

ALGEBRA

INFORMAZIONI: Quadro, Mostra, Espone

OPERAZIONI: join, selezione ,proiezione, differenza

LOGICA: A differenza dell'SQL utilizzo l'operatore insiemistico differenza e sottraggo a tutti i quadri quelli che sono stati esposti nel 1997.

SEL Autore='Picasso' ((Q)- (PROJ CQ_Autore, Periodo (SEL Anno=1997 ((Q JOIN_{Q.CQ=E.Ref_CQ} E) JOIN_{Ref_CM=M.CM} M))))

MYSQL

```

1 • SELECT *
2   FROM
3     quadro q
4   WHERE
5     q.autore = 'Picasso'
6     AND q.cq NOT IN (SELECT e.ref_cq
7                         FROM mostra m, espone e
8                         WHERE m.cm=e.ref_cm)
9
10
    75%  13:4
  Result Grid | Filter Rows: Search | Edit: 
  cq  autore  periodo
  NULL  NULL  NULL
  ▶ NULL  NULL  NULL

```

Figura 12.6- Codice MYSQL per l'interrogazione 2 (nessuna istanza risultante)

3. Selezionare tutti i dati dei quadri che non sono mai stati esposti con un quadro di Picasso, cioè nella stessa mostra in cui compariva anche un quadro di Picasso.

SQL

INFORMAZIONI: Quadro, Espone.

OPERAZIONI: interrogazione innestate, join , selezione , operatore insiemistico NOT IN

LOGICA: Con l'interrogazione B ottengo i codici delle mostre nelle quali sono stati esposti quadri di Picasso , con l'interrogazione A seleziono le mostre i cui codici non si trovano fra i risultati dell'interrogazione A, dunque proietto i codici dei quadri ivi esposti.

A **SELECT Q.***
FROM Quadro Q,Espone E
WHERE Q.CQ= E.Ref_CQ AND E.Ref_CM NOT IN (SELECT E1.Ref_CM

B **FROM QUADRO Q1, ESPONE E1**
WHERE Q1.CQ=E1.Ref_CQ AND Q.Autore='Picasso')

ALGEBRA

INFORMAZIONI: Quadro,Espone

OPERAZIONI: join, selezione ,proiezione

LOGICA: Utilizzo l'operatore differenza, non potendo utilizzare interrogazioni innestate.

PROJ CQ_Autore, Periodo((Q JOIN_{Q.CQ=E.Ref_CQ} E) - (SEL Autore='Picasso'(Q JOIN_{Q.CQ=E.Ref_CQ} E)))

MYSQL

```

1 • select q.*
2   from quadro q, espone e
3   where q.cq=e.ref_cq
4     AND e.ref_cm NOT IN ( select e1.ref_cm
5                           from espone e1, quadro q1
6                           where q1.cq=e1.ref_cq and q1.autore='Picasso'
7                         )
8
    100%  7:7
  Result Grid | Filter Rows: Search | Export: 
  cq  autore  periodo
  03  Neruda  321
  03  Neruda  321
  03  Neruda  321
  05  Giuseppe  12
  ▶ 03  Neruda  321
  ▶ 03  Neruda  321
  ▶ 03  Neruda  321
  ▶ 05  Giuseppe  12

```

Figura 12.7- Codice MYSQL per l'interrogazione 3 e relative istanze

4. Selezionare tutti i dati delle mostre in cui sono stati esposti quadri di almeno 5 autori distinti.

SQL

INFORMAZIONI: Quadro, Espone, Mostra

OPERAZIONI: interrogazione innestata con passaggio di parametri, join , operatore di confronto <=

LOGICA: Con l'interrogazione B ottengo il conteggio degli autori diversi che sono stati esposti in ogni mostra passata come parametro. Dunque con l'interrogazione A seleziono quelle mostre per cui 5 è minore o uguale del risultato dell'interrogazione B.

```
A   SELECT M.*  
    FROM MOSTRA M  
 WHERE 5 <=(SELECT Count(distinct Q.Autore)  
      B   FROM QUADRO Q, ESPONE E  
        WHERE Q.CQ=E.Ref_CQ AND E.RefCM=M.CM)
```

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the code from above. The results pane shows a single row with four columns: cm, nome, anno, and organizzatore, all containing NULL values. The status bar at the bottom indicates the result set has 0 rows.

cm	nome	anno	organizzatore
NULL	NULL	NULL	NULL

Figura 12.8- Codice MYSQL per l'interrogazione 4 (nessuna istanza risultante)

5. Selezionare per ogni mostra, l'autore di cui si esponevano il maggior numero di quadri

SQL

INFORMAZIONI: Quadro, Espone

OPERAZIONI: interrogazione innestata con passaggio di parametri, join , groupby , operatore di confronto >=ALL

LOGICA: Con l'interrogazione B ottengo il conteggio del numero di tuple per ogni sottotabella ottenuta dividendo la tabella di partenza in tante sottotabelle quanti sono i valori diversi dell'attributo autore, relativamente alla mostra passata come parametro. Dunque l'interrogazione A seleziona quelle sottotabelle , ottenute suddividendo quella di partenza in tante sottotabelle quanti sono i valori diversi della coppia REF-CM e AUTORE, che hanno un numero di tuple maggiore o uguale di tutti i valori ottenuti nell'interrogazione B , e di queste proietto il codice della mostra e l'autore.

```
A   SELECT Ref_CM,Autore  
    FROM ESPONE E, QUADRO Q  
 WHERE E. Ref_CQ=Q.CQ  
 GROUPBY E.Ref_CM, Q.Autore  
 HAVING COUNT(*)>= ALL (SELECT COUNT(*)  
      B   FROM QUADRO Q1, ESPONE E1  
        WHERE Q1.CQ=E1.Ref_CQ AND E1.Ref_CM=E.Ref_CM  
 GROUPBY Q1.Autore)
```

MYSQL

```
1
2 •  SELECT
3   e1.ref_cm, q1.autore
4   FROM
5     espone e1,
6     quadro q1
7   WHERE
8     e1.ref_cq = q1.cq
9   GROUP BY e1.ref_cm , q1.autore
10  HAVING COUNT(*) >= ALL (SELECT COUNT(*)
11
12    FROM espone e2, quadro q2
13    WHERE e2.ref_cq = q2.cq
14    AND e2.ref_cm = e1.ref_cm
15    GROUP BY q2.autore)
```

75% 14:3

Result Grid Filter Rows: Search Export:

ref_cm	autore
01	Picasso
02	Picasso
03	Neruda
03	Picasso

Figura 12.9- Codice MYSQL per l'interrogazione 5 e relative istanze

ESERCITAZIONE 13 - TRADING

Dato il seguente schema della base di dati TRADING:

PRODOTTO (CP, Descrizione)

SOCIO (CS, Nome, Cognome)

OFFERTA (CO, Validità)

COMPRENDE (Ref_CO, Ref_CP, quantità)

FK: Ref_CO REFERENCES Offerta (CO)

FK: Ref_CP REFERENCES Prodotto (CP)

RITIRA (R_CO, R_CS, Data)

FK: R_CO REFERENCES Offerta (CO)

FK: R_CS REFERENCES Socio(CS)

Prodotto (CP) è compreso, in una determinata quantità, in un'offerta (CO), che può essere ritirata in una specifica data da un Socio(CS, Nome) (Fig 13.1).

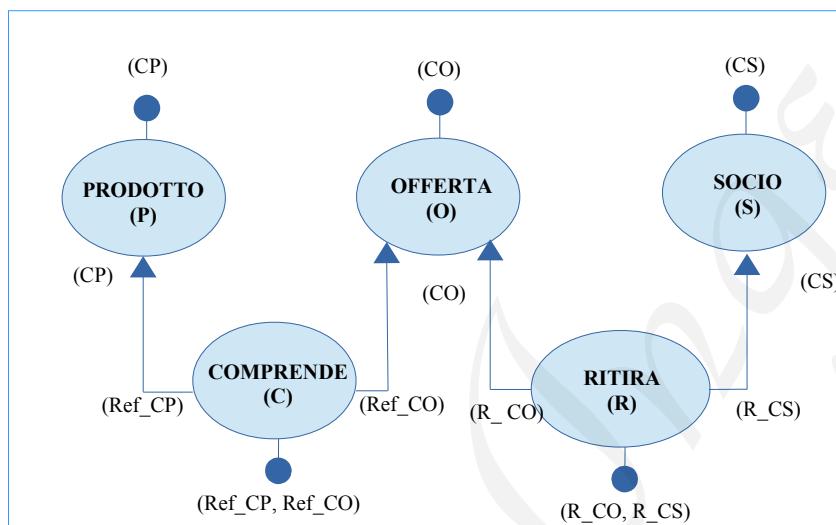


Fig. 13.1 – Modello dello schema di base di dati

Interrogazioni (Algebra relazionale 1,2,3):

1. Selezionare il codice e la descrizione dei prodotti che non sono mai stati offerti insieme ad un prodotto con descrizione “UVA”;
2. Selezionare nome, cognome e codice dei soci che non hanno mai ritirato un'offerta che comprende un prodotto con descrizione “UVA”;
3. Selezionare nome, cognome e codice dei soci che hanno ritirato tutte le offerte che comprendono un prodotto con descrizione “UVA”;
4. Selezionare, per ogni socio, il numero delle offerte ritirate che comprendono un prodotto con descrizione “UVA”.

Figura 13.2- Codice MYSQL per la creazione del database Trading

```

1   --
2   -- Create schema trading
3   --
4
5   ● CREATE DATABASE IF NOT EXISTS trading;
6   ● USE trading;
7
8   --
9   -- Definition of table prodotto
10  --
11
12  ● ┌ CREATE TABLE prodotto (
13    cp char(5) NOT NULL,
14    descrizione varchar(45) NOT NULL,
15    PRIMARY KEY (cp)
16  );
17
18  --
19  -- Definition of table socio
20  --
21
22  ● ┌ CREATE TABLE socio (
23    cs char(16) NOT NULL,
24    nome varchar(45) NOT NULL,
25    cognome varchar(45) NOT NULL,
26    PRIMARY KEY (cs)
27  );
28
29  --
30  -- Definition of table offerta
31  --
32
33  ● ┌ CREATE TABLE offerta (
34    co char(5) NOT NULL,
35    validita date NOT NULL,
36    PRIMARY KEY (co)
37  );
38
39  --
40  -- Definition of table comprende
41  --
42
43  ● ┌ CREATE TABLE comprende (
44    ref_co char(5) NOT NULL,
45    ref_cp char(5) NOT NULL,
46    quantita int(4) NOT NULL, -- E' IL NUMERO DI UNITA' DEL PRODOTTO cod_p_c COMPRESE
47    PRIMARY KEY (ref_co, ref_cp),
48    FOREIGN KEY (ref_co) REFERENCES offerta (co),
49    FOREIGN KEY (ref_cp) REFERENCES prodotto (cp)
50  );
51
52  --
53  -- Definition of table ritira
54  --
55
56  ● ┌ CREATE TABLE ritira (
57    r_co char(5) NOT NULL,
58    r_cs char(16) NOT NULL,
59    giorno date NOT NULL, -- E' IL GIORNO IN CUI IL SOCIO cod_s_r RITIRA L'OFFERTA co
60    PRIMARY KEY (r_co, r_cs),
61    FOREIGN KEY (r_co) REFERENCES offerta (co),
62    FOREIGN KEY (r_cs) REFERENCES socio (cs)
63  );

```

Figura 13.3- Codice MYSQL per la popolazione del database

```

1   --
2   -- populating the DB
3   --
4
5
6   ● INSERT INTO prodotto VALUES
7   ('01', 'Uva'),
8   ('02', 'Pera'),
9   ('03', 'Mela'),
10  ('04', 'Uva'),
11  ('05', 'Miele'),
12  ('06', 'Banana');
13
14  ● INSERT INTO socio VALUES
15  ('99', 'Mario', 'Rossi'),
16  ('98', 'Mario', 'Bonois'),
17  ('97', 'Mario', 'Brutti'),
18  ('96', 'Mario', 'Belli'),
19  ('95', 'Mario', 'Bianchi'),
20  ('94', 'Mario', 'Verdi'),
21  ('93', 'Mario', 'Gialli'),
22  ('92', 'Mario', 'Neri');
23
24  ● INSERT INTO offerta VALUES
25  ('55', '1-1-1'),
26  ('66', '1-3-1'),
27  ('44', '1-2-1');
28
29  ● INSERT INTO comprende VALUES
30  ('55', '01', '12'),
31  ('55', '02', '14'),
32  ('55', '06', '15'),
33  ('44', '01', '12'),
34  ('44', '04', '11'),
35  ('44', '05', '12),
36  ('66', '06', '25);
37
38  ● INSERT INTO ritira VALUES
39  ('55', '99', '1-1-1'),
40  ('44', '99', '1-1-1'),
41  ('55', '98', '1-1-1'),
42  ('66', '97', '1-1-1'),
43  ('66', '93', '1-1-1'),
44  ('66', '96', '1-1-1'),
45  ('44', '92', '1-1-1');
46
47

```

Figura 13.4- Istanze della base di dati

1 • select * from prodotto

Result Grid | Filter Rows: Search Edit: |

cp	descrizione
01	Uva
02	Pera
03	Mela
04	Uva
05	Miele
06	Banana
NULL	NULL

100% 23:1

1 • select * from offerta

Result Grid | Filter Rows: Search Edit: |

co	validita
44	0001-02-01
55	0001-01-01
66	0001-03-01
NULL	NULL

100% 22:1

1 • select * from socio

Result Grid | Filter Rows: Search Edit: |

cs	nome	cognome
92	Mario	Neri
93	Mario	Gialli
94	Mario	Verdi
95	Mario	Bianchi
96	Mario	Belli
97	Mario	Brutti
98	Mario	Bonolis
99	Mario	Rossi
NULL	NULL	NULL

100% 20:1

1 • select * from ritira

Result Grid | Filter Rows: Search Edit: |

r_co	r_cs	giorno
44	92	0001-01-01
44	99	0001-01-01
55	98	0001-01-01
55	99	0001-01-01
66	93	0001-01-01
66	96	0001-01-01
66	97	0001-01-01
NULL	NULL	NULL

100% 21:1

1 • select * from comprende

Result Grid | Filter Rows: Search Edit: |

ref_co	ref_cp	quantita
44	01	12
44	04	11
44	05	12
55	01	12
55	02	14
55	06	15
66	06	25
NULL	NULL	NULL

100% 24:1

1. Selezionare il codice e la descrizione dei prodotti che non sono mai stati offerti insieme ad un prodotto con descrizione “UVA”

SQL

INFORMAZIONI: Prodotto, Comprende

OPERAZIONI: interrogazione innestata , operatore insiemistico NOT IN, join

LOGICA: Con l'interrogazione B ottengo i codici dei prodotti che sono venduti in una stessa offerta con un prodotto con descrizione Uva, tramite l'interrogazione A seleziono solo quei prodotti i cui codici non si trovano fra i risultati dell'interrogazione B. Ne proietto dunque codice e descrizione.

```
A  SELECT P.CP, P.Descrizione
  FROM Prodotto P
 WHERE P.CP NOT IN (SELECT C1.Ref_Cp
B   FROM COMPRENDE C1, COMPRENDE C2, PRODOTTO P
      WHERE C1.Ref_CO= C2.Ref_CO AND
            C2.Ref_Cp=P.CP AND P.descrizione='Uva'
      )
```

ALGEBRA

INFORMAZIONI: Prodotto, Comprende

OPERAZIONI: join, selezione, differenza, proiezione e ridenominazione.

LOGICA: Con l'interrogazione R3 ottengo i codici dei prodotti che sono stati offerti insieme ad un prodotto con descrizione Uva, dunque sottraggo a tutti i codici prodotto quelli risultanti dalla R3 ottenendo quindi quanto richiesto dall'interrogazione

```

PROJ CP, Descrizione (P JOIN P.CP=CP (
    (PROJ CP P) -
    R3 (PROJ CP (REN CP← Ref_CO_R ( SEL Descrizione='UVA' ( (C JOIN C.Ref_Cp=P.CP P) JOIN Ref_CO=C.Ref_CO_R (REN Ref_CO_R, Ref_Cp_R, Quantità_R ←
        Ref_CO, Ref_Cp, Quantità C )))))
)

```

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```

1 • SELECT *
2   FROM prodotto p1
3     WHERE p1.cp NOT IN (SELECT c2.ref_cp
4                           FROM comprende c1,
5                                comprende c2,
6                                prodotto p2
7                               WHERE c1.ref_cp = p2.cp
8                                 AND c1.ref_co = c2.ref_co
9                                 AND p2.descrizione = 'Uva')
10
11
12
13
14
15

```

The results grid shows one row:

cp	descrizione
03	Mela

Fig 13.5- Codice MYSQL per l'interrogazione 1 e istanze risultanti

2. Selezionare nome, cognome e codice dei soci che non hanno mai ritirato un'offerta che comprende un prodotto con descrizione 'UVA'.

SQL

INFORMAZIONI: Prodotto, Comprende, Offerta, Ritira, Socio

OPERAZIONI: interrogazione innestata , operatore insiemistico NOT IN, join, selezione.

LOGICA: Con l'interrogazione B ottengo i codici dei soci che hanno ritirato un'offerta che comprende un prodotto con descrizione UVA, tramite l'interrogazione A seleziono i codici dei soci che non compaiono fra i risultati della B.

```

A SELECT S.*
  FROM SOCIO S
 WHERE S.cs NOT IN (SELECT R.R_CS
    B  FROM PRODOTTO P, COMPRENDE C, OFFERTA O, RITIRA R
      WHERE P.CP=C.Ref_Cp AND C.Ref_CO=O.CO AND O.CO=R.R_CO
            P.descrizione='Uva'
)

```

ALGEBRA

INFORMAZIONI: Prodotto, Comprende, Offerta, Ritira, Socio

OPERAZIONI: interrogazione innestata , operatore insiemistico NOT IN, join, selezione.

LOGICA: Con l'interrogazione R2 ottengo i codici dei soci che hanno ritirato un'offerta che comprende un prodotto con descrizione Uva, sottraendoli poi a tutti i codici dei soci ottenuto quanto richiesto dall'interrogazione .

```

S JOIN S.cs=cs(
    (PROJ CS S) -
    R1 (PROJ CS (REN CS←R_CS (SEL Descrizione='Uva' ( (( P JOIN P.CP=C.Ref_Cp C) JOIN Ref_CO=O.CO O) CO=R.R_CO R ) )))
)

```

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```

1 • SELECT *
2   FROM socio s, ritira r, offerta o
3     WHERE s.cs=r.r_cs AND r.r_co=o.co AND NOT EXISTS (SELECT *
4                                               FROM comprende c, prodotto p
5                                               WHERE c.ref_cp = p.cp
6                                               AND c.ref_co = o.co AND p.descrizione='Uva')
7
8
9
10
11

```

The results grid shows three rows:

cs	nome	cognome	r_co	r_cs	giorno	co	validita
93	Mario	Gialli	66	93	0001-01-01	66	0001-03-01
96	Mario	Belli	66	96	0001-01-01	66	0001-03-01
97	Mario	Brutti	66	97	0001-01-01	66	0001-03-01

Fig 13.6- Codice MYSQL per l'interrogazione 2 e istanze risultanti

3. Selezionare nome, cognome e codice dei soci che hanno ritirato tutte le offerta che comprendono un prodotto con descrizione "UVA".

SQL

INFORMAZIONI: Prodotto, Comprende, Offerta, Ritira, Socio

OPERAZIONI: interrogazione innestate con passaggio di parametri , join, selezione e doppio not exists

LOGICA: Con l'interrogazione C ottengo tutte le tuple di ritira per cui il codice dell'offerta è uguale a quello passato come parametro dall'interrogazione B e il codice del socio è quello passato come parametro dalla A. Nell'interrogazione B seleziono quelle tuple della relazione ottenuta dal join fra O , C e P che hanno descrizione='Uva'. Infine con l'interrogazione A, tramite la doppia negazione, ottengo le informazioni su quei soci tali per cui non esista un'offerta da essi ritirata che non comprenda un prodotto con descrizione UVA.

```
A  SELECT S.*  
  FROM SOCIO S  
 WHERE NOT EXISTS (SELECT *  
    B   FROM OFFERTA O, COMPRENDE C, PRODOTTO P  
    WHERE O.CO=C.Ref_CO AND C.Ref_CP=P.CP  
          AND P.descrizione='Uva' AND NOT EXISTS ( SELECT *  
                                              FROM RITIRA R  
                                              WHERE R.R_CO=O.CO AND R.R_CS= S.CS ))
```

ALGEBRA

INFORMAZIONI: Prodotto, Comprende, Ritira, Socio

OPERAZIONI: divisione, join, selezione, ridenominazione, proiezione.

LOGICA: Con l'interrogazione R1 ottengo i codici delle offerte in cui comparivano prodotti con descrizione 'Uva', effettuando la divisione fra Ritira ed R1 ottengo i codici dei soci che soddisfano quanto richiesto dall'interrogazione.

```
S JOIN S.CS=R.CS  
(PROJ R.CS,R.CO R) ÷  
R1 (PROJ R.CO (REN R.CO- Ref_CO( SEL descrizione='Uva' (C JOIN C.Ref_CP=P.CP P))))
```

MYSQL

```
1 •  SELECT *  
2   FROM socio s  
3   WHERE NOT EXISTS( SELECT *  
4     FROM  
5       comprende c,  
6       prodotto p,  
7       offerta o  
8     WHERE  
9       c.ref_cp = p.cp AND  
10      c.ref_co=o.co AND  
11      p.descrizione = 'Uva'  
12      AND NOT EXISTS( SELECT *  
13        FROM ritira r  
14        WHERE r.r_cs=s.cs  
15        AND r.r_co=o.co));
```

The screenshot shows the MySQL Workbench interface with the SQL query in the editor and the results grid below. The results grid shows one row with columns CS, nome, cognome, containing values 99, Mario, Rossi respectively.

CS	nome	cognome
99	Mario	Rossi

Fig 13.7- Codice MYSQL per l'interrogazione 3 e istanze risultanti

4. Selezionare, per ogni socio, il numero delle offerte ritirate che comprendono un prodotto con descrizione "UVA".

SQL

INFORMAZIONI: Offerta, Ritira, Comprende, Prodotto.

OPERAZIONI: join, groupby, selezione

LOGICA: seleziono tra le tuple della relazione ,ottenuta effettuando join fra le 4 relazioni, quelle il cui attributo descrizione coincide con il valore 'UVA'. Le suddivido in tante sottotabelle quanti sono i valori diversi dell'attributo R_CS e li proietto insieme al conteggio delle tuple di ogni sottotabella.

```
SELECT R.R_CS, COUNT(*)  
  FROM OFFERTA O, RITIRA R, COMPRENDE C, PRODOTTO P  
 WHERE O.CO=R.R_CO AND C.Ref_CO=O.CO AND C.Ref_CP=P.CP  
       P.descrizione='Uva'  
 GROUPBY R.R_CS
```

MYSQL

```
1 •  SELECT
2      r.r_cs, COUNT(*)
3  FROM
4      comprende c,
5      prodotto p,
6      ritira r,
7      offerta o
8 WHERE
9      c.ref_cp = p.cp AND
10     c.ref_co=o.co AND
11     r.r_co=o.co AND
12     p.descrizione = 'Uva'
13 GROUP BY r.r_cs;
```

100% 17:13

Result Grid Filter Rows: Search Export

r_cs	COUNT(*)
92	2
98	1
99	3

Fig 13.8 - Codice MYSQL per l'interrogazione 4 e istanze risultanti

ESERCITAZIONE 14 - STRADARIO

06/06/2017

Dato il seguente schema della base di dati STRADARIO:

VIA (CV, Nome, Quartiere, Lunghezza)

INCROCIO (CVA, CVB, n_volte)

FK: CVA REFERENCES VIA(CV)

FK: CVB REFERENCES VIA(CV)

Coppie di Vie(CV, Nome) si incrociano un certo numero di volte.

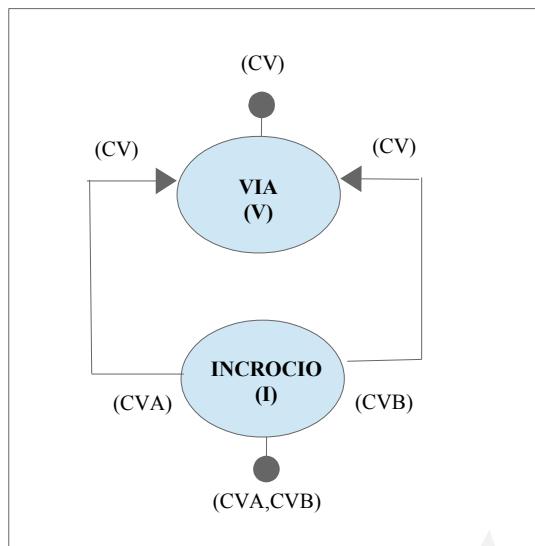


Fig. 14.1 – Modello dello schema di base di dati

Interrogazioni (Algebra relazionale 1,2)

1. Selezionare le vie che incrociano almeno una via del quartiere 'Pastena';
2. Selezionare le vie che non incrociano via 'Marco Polo';
3. Selezionare le coppie (CODICE1, CODICE2) tali che le vie CODICE1 e CODICE2 abbiano la stessa lunghezza;
4. Selezionare il quartiere che ha il maggior numero di vie;
5. Selezionare, per ogni quartiere, la via di lunghezza maggiore;
6. Selezionare le vie che incrociano tutte le vie del quartiere 'Pastena'.

Fig. 14.2- Codice MySQL per la creazione del database Stradario

```
1  -- Create schema stradario
2
3
4
5  • CREATE DATABASE IF NOT EXISTS stradario;
6  • USE stradario;
7
8  --
9  -- Definition of table via
10 --
11
12 • □ CREATE TABLE via (
13   cv char(5) NOT NULL,
14   nome varchar(75) NOT NULL,
15   quartiere varchar(45) NOT NULL,
16   lunghezza numeric(6,3),
17   PRIMARY KEY (cv)
18 );
19
20  --
21  -- Definition of table incrocio
22  --
23
24 • □ CREATE TABLE incrocio (
25   cva char(5) NOT NULL,
26   cvb char(5) NOT NULL,
27   n_volte int,
28   PRIMARY KEY (cva,cvb),
29   FOREIGN KEY (cva) REFERENCES via(cv),
30   FOREIGN KEY (cvb) REFERENCES via(cv),
31 );
32
```

Fig. 14.3 - Codice MySQL per la popolazione del database

```

1
2
3 -- populating the DB
4
5
6 • INSERT INTO via VALUES
7 ('01','Main St','Pastena',100),
8 ('02','2nd Av','Pastena',101),
9 ('03','3rd St','02',102),
10 ('04','4th St','03',103),
11 ('05','5th St','03',104),
12 ('06','6th St','03',105),
13 ('07','Wall St','02',106),
14 ('08','New St','04',107),
15 ('09','Old St','03',108),
16 ('10','Marco Polo','Pastena',109);
17
18 • INSERT INTO incrocio VALUES
19 ('01','02',3),
20 ('03','02',5),
21 ('04','03',1),
22 ('10','07',2),
23 ('05','03',2),
24 ('01','10',31),
25 ('09','06',3),
26 ('07','08',4),
27 ('02','09',5),
28 ('01','05',12),
29 ('05','06',10)

```

Fig. 14.4 - Istanze della base di dati

cv	nome	quartiere	lunghezza	cva	cvb	n_volte
01	Main St	Pastena	100.000	01	05	12
02	2nd Av	Pastena	101.000	01	10	31
03	3rd St	02	102.000	02	09	5
04	4th St	03	103.000	03	02	5
05	5th St	03	104.000	04	03	1
06	6th St	03	105.000	05	03	2
07	Wall St	02	106.000	05	06	10
08	New St	04	107.000	07	08	4
09	Old St	03	108.000	09	06	3
10	Marco Polo	Pastena	109.000	10	07	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL

1. Selezionare le vie che incrociano almeno una via del quartiere 'Pastena'

SQL

INFORMAZIONI: Via, Incrocio.

OPERAZIONI: Interrogazioni innestate parallele, join , selezione, operatore insiemistico IN

LOGICA: Con le interrogazioni B e C ottengo i codici delle vie che incrociano almeno una via del quartiere Pastena, con l'interrogazione A seleziono solo le tuple di via il cui codice è contenuto fra i risultati della B o fra i risultati della C.

A **SELECT ***
FROM Via V
WHERE V.CV **IN** (**SELECT** I1.cvb
 B **FROM** Incrocio I1, Via V1
 WHERE I1.cva=V1.cv AND
 V1.quartiere='Pastena')
OR V.CV **IN** (**SELECT** I2.cva
 FROM Incrocio I2, Via V2
 C **WHERE** I2.cvb=V2.cv AND
 V2.quartiere='Pastena')

ALGEBRA

INFORMAZIONI: Via, Incrocio

OPERAZIONI: join, selezione, proiezione, operatore insiemistico unione

LOGICA: Con le due interrogazioni ottengo i codici delle vie che incrociano almeno una via del quartiere 'Pastena', facendo rispettivamente l'equi join considerando in un caso la cva e nell'altro la cvb. Poi effettuo l'unione.

(PROJ CVB (SEL quartiere='Pastena' (I JOIN I.CVA=V.CV V)))

U

(PROJ CVA(SEL quartiere='Pastena' (I JOIN I.CVB=V.CV V)))

MYSQL

The screenshot shows a MySQL query editor with the following code:

```
1 • SELECT *
2   FROM
3     via v
4   WHERE
5     v.cv IN (SELECT
6       i1.cva
7         FROM
8           incrocio i1,
9             via v1
10            WHERE
11              i1.cvb = v1.cv AND v1.quartiere = 'Pastena'
12              OR v.cv IN (SELECT
13                i2.cvb
14                  FROM
15                    incrocio i2,
16                      via v2
17                        WHERE
18                          i2.cva = v2.cv AND v2.quartiere = 'Pastena'));
```

Below the code is a result grid showing the following data:

cv	nome	quartiere	lunghezza
01	Main St	Pastena	100.000
02	2nd Av	Pastena	101.000
03	3rd St	02	102.000
05	5th St	03	104.000
07	Wall St	02	106.000
09	Old St	03	108.000
10	Marco Polo	Pastena	109.000
NULL	HULL	NULL	NULL

Fig.14.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare le vie che non incrociano via 'Marco Polo'.

SQL

INFORMAZIONI: Via, Incrocio.

OPERAZIONI: Interrogazioni innestate parallele, join , selezione, operatore insiemistico NOT IN.

LOGICA: Con le interrogazioni B e C ottengo i codici delle vie che incrociano almeno una volta via "Marco Polo", con l'interrogazione A seleziono solo le tuple di via il cui codice non è contenuto fra i risultati della B o fra i risultati della C.

A **SELECT ***
A **FROM** Via V
A **WHERE** V.CV **NOT IN** (**SELECT** I1.cvb
 FROM Incrocio I1, Via V1
 WHERE I1.cva=V1.cv AND
 V1.nome='Marco Polo')
OR V.CV **NOT IN** (**SELECT** I2.cva
 FROM Incrocio I2, Via V2
 WHERE I2.cvb=V2.cv AND
 V2.nome ='Marco Polo')

ALGEBRA

INFORMAZIONI: Via, Incrocio

OPERAZIONI: join, selezione, proiezione, operatore insiemistico unione , sottrazione

LOGICA: Con le due interrogazioni R1 e R2 ottengo i codici delle vie che incrociano almeno una volta la via 'Marco Polo', ne faccio l'unione e la sottraggo a tutti le vie ottenendo quindi quelle che non la incrociano.

(PROJ CV V) -

R1: ((PROJ CVB (SEL nome='Marco Polo' (I JOIN I.CVA=V.CV V)))

U

R2: (PROJ CVA(SEL nome='Marco Polo' (I JOIN I.CVB=V.CV V))))

MYSQL

The screenshot shows a MySQL query editor with a complex multi-table join query. Below the query results is a table with four columns: cv, nome, quartiere, and lunghezza. The data is as follows:

cv	nome	quartiere	lunghezza
06	6th St	03	105.000
07	Wall St	02	106.000
08	New St	04	107.000
NULL	NULL	NULL	NULL

Fig.14.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare le coppie (CODICE1, CODICE2) tali che le vie CODICE1 e CODICE2 abbiano la stessa lunghezza;

SQL

INFORMAZIONI: Via

OPERAZIONI: join, selezione

LOGICA: Effettuo un join la tabella via e se stessa ottenendo una relazione in cui le tuple sono informazioni riguardanti coppie di vie con la stessa lunghezza, infine seleziono solo quelle tuple che effettivamente riguardano due vie diverse e ne proietto i codici, ridenominandoli CODICE 1 e CODICE 2.

```
SELECT V1.CV as CODICE1, V2.CV as CODICE2  
FROM Via V1, Via V2  
WHERE V1.lunghezza=V2.lunghezza AND V1.CV>V2.CV
```

MYSQL

The screenshot shows a MySQL query editor with a query that joins the 'via' table with itself. The result table has two columns: codice1 and codice2. There are no visible rows in the result grid.

Fig 14.7– Codice MySQL per l'interrogazione 3 e istanze risultanti

4. Selezionare il quartiere che ha il maggior numero di vie;

SQL

INFORMAZIONI: Via

OPERAZIONI: groupby, operatore aggregato count, operatore di confronto \geq

LOGICA: Con la relazione B suddivido la relazione Via in tante sottotabelle quanti sono i valori diversi di quartiere, e proietto i conteggi delle tuple relativi a tutte le sottotabelle. Dunque con la relazione A effettuo lo stesso groupby e proietto il valore di quartiere per il quale la rispettiva sottotabella ha un numero di tuple maggiore o uguale di tutti i valori ottenuti dalla relazione B.

```

SELECT V.quartiere
A  FROM Via V
GROUPBY V.quartiere
HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
B      FROM VIA V1
        GROUPBY V1.quartiere
)

```

MySQL

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor with the following SQL query:

```

1 •  SELECT v.quartiere
2   FROM via v
3   GROUP BY v.quartiere
4   HAVING COUNT(*) >= ALL (SELECT COUNT(*)
5           FROM via v1
6           group by v1.quartiere);
7

```

Below the code editor is a toolbar with buttons for Result Grid, Filter Rows, Search, and Export. The Result Grid tab is selected. The results table has two columns: quartiere and count. The data shows one row with quartiere 03 and count 03.

quartiere	count
03	03

Fig.14.8 – Codice MySQL per l'interrogazione 4 e istanze risultanti

5. Selezionare, per ogni quartiere, la via di lunghezza maggiore

SQL

INFORMAZIONI: Via

OPERAZIONI: operazioni innestate con passaggio di parametri, operatore di confronto \geq

LOGICA: Con l'interrogazione B ottengo i valori delle lunghezze delle vie relative al quartiere passato come parametro, dunque con l'interrogazione A seleziono quelle tuple di Via la cui lunghezza risulta maggiore o uguale di tutti i valori ottenuti come risultato dell'interrogazione B. Quindi ne proietto il quartiere e il codice.

```

SELECT V.quartiere, V.CV
A  FROM Via V
WHERE V.lunghezza >= ALL ( SELECT V1.lunghezza
B      FROM VIA V1
        WHERE V1.quartiere=V.quartiere
)

```

MySQL

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor with the following SQL query:

```

1 •  SELECT
2       v1.quartiere, v1.cv
3     FROM
4       via v1
5     WHERE
6       v1.lunghezza = (SELECT MAX(v2.lunghezza)
7                         FROM via v2
8                         WHERE v1.quartiere = v2.quartiere);

```

Below the code editor is a toolbar with buttons for Result Grid, Filter Rows, Search, Edit, and Export. The Result Grid tab is selected. The results table has two columns: quartiere and cv. The data shows three rows: 02, 07; 04, 08; and 03, 09. There is also a row for Pastena with values NULL, NULL.

quartiere	cv
02	07
04	08
03	09
Pastena	10
NULL	NULL

Fig.14.9 – Codice MySQL per l'interrogazione 5 e istanze risultanti

6. Selezionare le vie che incrociano tutte le vie del quartiere Pastena.

SQL

INFORMAZIONI: Via, Incrocio

OPERAZIONI: operazioni innestate con passaggio di parametri, doppio not exists

LOGICA: Con l'interrogazione C ottengo tutti gli incroci possibili fra la via passata come parametro della relazione A e la via passata come parametro dalla relazione B (sulla quale è stata applicata la selezione che il quartiere che dev'essere necessariamente= 'Pastena'). Con la doppia negazione ottengo dall'interrogazione A come risultato finale le informazioni sulle vie per le quali non esiste una via del quartiere Pastena che esse non incrociano.

```

A   SELECT *
    FROM Via V
    WHERE NOT EXISTS ( SELECT *
B       FROM VIA V1
        WHERE V1.quartiere='Pastena'
        AND NOT EXISTS ( SELECT *
C           FROM INCROCIO I
            WHERE (I.CVA=V.CV AND I.CVB=V1.CV) OR (I.CVA=V1.CV AND
                I.CVB=V.CV)
        ) )

```

MYSQL

```

1 •  SELECT *
2   *
3   FROM
4     via v1
5   WHERE
6     NOT EXISTS( SELECT *
7       FROM via v2
8       WHERE v2.quartiere = 'Pastena'
9       AND NOT EXISTS( SELECT *
10      FROM incrocio i
11      WHERE (i.cva = v1.cv AND i.cvb=v2.cv)
12      OR (i.cvb=v1.cv AND i.cva=v2.cv)));

```

Result Grid | Filter Rows: Search | Edit: | Export/Import:

cv	nome	quartiere	lunghezza
NULL	NULL	NULL	NULL

Fig.14.10 – Codice MySQL per l'interrogazione 6 (nessuna istanza risultante)

ESERCITAZIONE 15 – TENNIS

Dato il seguente schema per la base di dati TENNIS:

CAMPO (NCampo, Tipo, Indirizzo)

TENNISTA (CF, Nome, Nazione)

INCONTRO (CodI, Ref_Campo, Gioc1, Gioc2, Set1, Set2)

FK: Ref_Campo REFERENCES Campo (NCampo)

FK: Gioc1 REFERENCES Tennista (CF)

FK: Gioc2 REFERENCES Tennista (CF)

Due Giocatori (CF) partecipano ad un Incontro (CodI), che si svolge in un determinato Campo (NCampo) (Fig. 15.1).

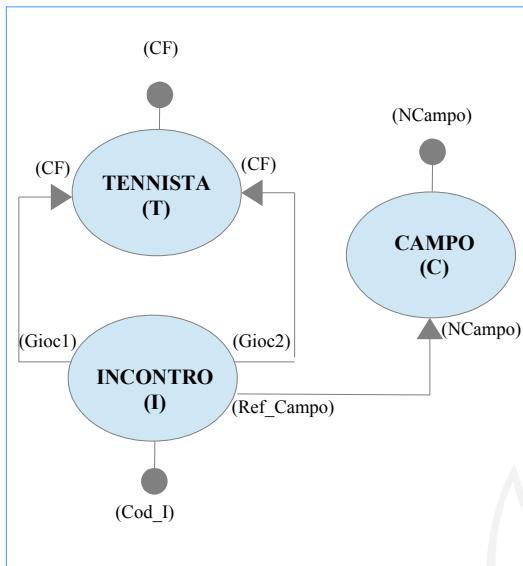


Fig. 15.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1, 2, 3, 4):

1. Selezionare gli incontri disputati sull'erba (campo con tipo 'erba');
2. Selezionare i campo in erba sui quali non c'è nessun incontro;
3. Selezionare i dati dei tennisti vincitori di almeno una partita sull'erba;
4. Selezionare i dati delle nazioni in cui tutti i giocatori hanno sempre vinto le partite disputate;
5. Selezionare il campo in erba che ha ospitato il maggior numero di incontri;

Fig. 15.2 – Creazione del database Tennis

```
1 -- Create schema tennis
2 • CREATE DATABASE IF NOT EXISTS tennis;
3 • USE tennis;
4 --
5 -- Definition of table campo
6 --
7 • └─ CREATE TABLE campo (
8     nome_campo varchar(75) NOT NULL,
9     tipo varchar(10) NOT NULL,
10    indirizzo varchar(45) NOT NULL,
11    PRIMARY KEY (nome_campo)
12 );
13 --
14 -- Definition of table tennista
15 --
16 • └─ CREATE TABLE tennista (
17     cf char(16) NOT NULL,
18     nome varchar(75) NOT NULL,
19     nazione char(3) NOT NULL,
20     PRIMARY KEY (cf)
21 );
22 --
23 -- Definition of table incontro
24 --
25 • └─ CREATE TABLE incontro (
26     cod_i char(6) NOT NULL,
27     nome_campo_i varchar(75) NOT NULL,
28     giocatore1 char(16) NOT NULL,
29     giocatore2 char(16) NOT NULL,
30     set1 int NOT NULL,
31     set2 int NOT NULL,
32     PRIMARY KEY (cod_i),
33     FOREIGN KEY (nome_campo_i) REFERENCES campo(nome_campo),
34     FOREIGN KEY (giocatore1) REFERENCES tennista(cf),
35     FOREIGN KEY (giocatore2) REFERENCES tennista(cf)
36 );
```

Fig. 15.3 – Popolazione del database

```
1 • INSERT INTO campo VALUES ('Arthur Ashe Stadium', 'Cemento', 'Queens'),
2   ('Rochus Club', 'Terra', 'Via Berlino'),
3   ('Comet Club', 'Erba', 'Via Roma'),
4   ('Tennis Club', 'Erba', 'Via Milano'),
5   ('Crazy Tennis', 'Erba', 'Via Palermo');
6 • INSERT INTO tennista VALUES ('N', 'Nadal', 'SPA'),
7   ('A', 'Agassi', 'USA'),
8   ('F', 'Federer', 'SVI'),
9   ('L', 'Laver', 'AUS');
10 • INSERT INTO incontro VALUES (1, 'Arthur Ashe Stadium', 'N', 'F', 6, 4),
11   (2, 'Arthur Ashe Stadium', 'A', 'N', 6, 0),
12   (3, 'Arthur Ashe Stadium', 'L', 'A', 2, 6),
13   (4, 'Arthur Ashe Stadium', 'F', 'L', 4, 6),
14   (5, 'Comet Club', 'A', 'F', 6, 2),
15   (6, 'Comet Club', 'L', 'N', 6, 1),
16   (7, 'Tennis Club', 'F', 'N', 6, 3);
```

Fig. 15.4 – Istanze della base di dati

cf	nome	nazione		nome_campo	tipo	indirizzo
A	Agassi	USA		Arthur Ashe Stadium	Cemento	Queens
F	Federer	SVI		Comet Club	Erba	Via Roma
L	Laver	AUS		Crazy Tennis	Erba	Via Palermo
N	Nadal	SPA		Rochus Club	Terra	Via Berlino
				Tennis Club	Erba	Via Milano

cod_i	nome_campo_i	giocatore1	giocatore2	set1	set2
1	Arthur Ashe Stadium	N	F	6	4
2	Arthur Ashe Stadium	A	N	6	0
3	Arthur Ashe Stadium	L	A	2	6
4	Arthur Ashe Stadium	F	L	4	6
5	Comet Club	A	F	6	2
6	Comet Club	L	N	6	1
7	Tennis Club	F	N	6	3

1. Selezionare gli incontri disputati sull'erba (campo con tipo 'erba') :

SQL

INFORMAZIONI: Incontro, Campo

OPERAZIONI: join, selezione e proiezione

LOGICA: Con il join fra incontro e campo ottengo tutti i campi in cui si è disputato almeno un incontro. Fra questi seleziono quelli disputati sull'erba e proietto tutte le informazioni dell'incontro

```
SELECT I.*  
FROM Incontro I, Campo C  
WHERE I.Ref_Campo = C.NCampi AND C.Tipo = 'erba'
```

ALGEBRA

INFORMAZIONI: Incontro, Campo

OPERAZIONI: join, selezione, proiezione

LOGICA: Seleziono i campi in erba in cui si è disputato almeno un incontro e proietto tutti i dati di tali incontri

PROJ CodI, Ref_Campo, Gioc1, Gioc2, Set1, Set2 (I JOIN I.Ref_Campo = NCampi (**PROJ** NCampi (**SEL** Tipo = 'erba' C)))

MYSQL

cod_i	nome_campo_i	giocatore1	giocatore2	set1	set2
5	Comet Club	A	F	6	2
6	Comet Club	L	N	6	1
7	Tennis Club	F	N	6	3

Fig. 15.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare i campo in erba sui quali non c'è nessun incontro

SQL

INFORMAZIONI: Incontro, Campo

OPERAZIONI: Interrogazione innestata con operatore insiemistico Not In

LOGICA: Con l'interrogazione B ottengo l'insieme dei campi in cui c'è stato almeno un incontro. Con l'interrogazione A seleziono i campi che non sono presenti nell'insieme risultante dall'interrogazione B

A **SELECT** C.*
 FROM Campo C
 WHERE C.Tipo = 'erba' AND C.NCampus NOT IN (**SELECT** I.Ref_Campo
 B **FROM** Incontro I)

ALGEBRA

INFORMAZIONI: Incontro, Campo

OPERAZIONI: Join, selezione, proiezione, operatore insiemistico differenza

LOGICA: Utilizzo la differenza al posto dell'operatore Not In di SQL. Inoltre in questo caso devo ottenere, con un join, due membri della differenza con attributi omogenei

(**SEL** C.Tipo = 'erba' C) – (**PROJ** NCampus, Tipo, Indirizzo (**I JOIN** I.Ref_Campo = C.NCampus C))

MYSQL

```
1 •  SELECT *
2   FROM campo c
3   WHERE c.tipo = 'erba' AND c.nome_campo NOT IN
4     (SELECT i.nome_campo_i
5      FROM incontro i);
6
7 100% 24:4
8 Result Grid | Filter Rows: Search | Edit: |
```

nome_campo	tipo	indirizzo
Crazy Tennis	Erba	Via Palermo

Fig. 15.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare i dati dei tennisti vincitori di almeno una partita sull'erba

SQL

INFORMAZIONI: Tennista, Incontro, Campo

OPERAZIONI: Join, selezione, due interrogazioni innestate allo stesso livello con operatore insiemistico In

LOGICA: Con l'interrogazione B ottengo tutti i giocatori vincitori sull'erba in casa, mentre con l'interrogazione C ottengo tutti i giocatori vincitori sull'erba fuori casa. Infine con l'interrogazione A seleziono tutti i dati dei tennisti che sono parte dell'insieme derivante dall'interrogazione B, oppure dell'insieme derivante dall'interrogazione C.

A **SELECT** *
 FROM Tennista T
 WHERE CF IN (**SELECT** Gioc1
 B **FROM** Incontro I, Campo C
 WHERE I.Ref_Campo = C.NCampus AND C.Tipo = 'erba' AND I.Set1 > I.Set2)
 OR CF IN (**SELECT** Gioc2
 C **FROM** Incontro I1, Campo C1
 WHERE I1.Ref_Campo = C1.NCampus AND C1.Tipo = 'erba' AND I1.Set1 < I1.Set2)

ALGEBRA

INFORMAZIONI: Tennista, Incontro, Campo

OPERAZIONI: Join, selezione, proiezione, ridenominazione, operatore insiemistico Unione

LOGICA: Seleziono i tennisti che hanno giocato sull'erba e hanno vinto, o in casa o fuori casa e ne proietto tutti i dati

IA, CA sono le ridenominazioni di tutti gli attributi delle tabelle I, C aggiungendo una 'A' al nome dell'attributo

IB, CB sono le ridenominazioni di tutti gli attributi delle tabelle I, C aggiungendo una 'B' al nome dell'attributo

PROJ CF, Nome, Nazione (**SEL** Set1A > Set2A ((**T JOIN** T.CF = IA.Gioc1A IA) **JOIN** Ref_CampoA = NCampiA (**SEL** TipoA = 'erba' CA))

U

PROJ CF, Nome, Nazione (**SEL** Set1B < Set2B ((**T JOIN** T.CF = IB.Gioc2B IB) **JOIN** Ref_CampoB = NCampiB (**SEL** TipoB = 'erba' CB))

```

1 •  SELECT *
2   FROM tennista t
3   WHERE t.cf IN (
4     SELECT i.giocatore1
5       FROM incontro i, campo c
6      WHERE i.nome_campo_i = c.nome_campo
7        AND c.tipo = 'erba'
8        AND i.set1 > i.set2)
9   OR t.cf IN (SELECT i.giocatore2
10    FROM incontro i, campo c
11   WHERE i.nome_campo_i = c.nome_campo
12     AND c.tipo = 'erba'
13     AND i.set1 < i.set2)

```

cf	nome	nazione
A	Agassi	USA
F	Federer	SVI
L	Laver	AUS

Fig. 15.7 – Codice MySQL per l'interrogazione 3 e istanze risultanti

4. Selezionare i dati delle nazioni in cui tutti i giocatori hanno sempre vinto le partite disputate

SQL

INFORMAZIONI: Tennista, Incontro

OPERAZIONI: Due interrogazioni innestate allo stesso livello con operatore insiemistico Not In, join, selezione

LOGICA: Con l'interrogazione C ottengo tutte le nazioni in cui un giocatore ha perso una partita fuori casa, mentre con l'interrogazione B ottengo tutte le nazioni in cui un giocatore ha perso una partita in casa. Infine con l'interrogazione A seleziono le nazioni che non si trovano in nessuno dei due insiemi risultanti dalle interrogazioni B e C

```

A  SELECT DISTINCT T.Nazione
  FROM Tennista T
 WHERE T.Nazione NOT IN ( SELECT T1.Nazione
                           FROM Incontro I1, Tennista T1
                         WHERE I1.Gioc1 = T1.CF AND I1.Set1 < I1.Set2)
 AND T.Nazione NOT IN ( SELECT T2.Nazione
                           FROM Incontro I2, Tennista T2
                         WHERE I2.Gioc2 = T2.CF AND T1.Set2 > I2.Set2)

```

ALGEBRA

INFORMAZIONI: Tennista, Incontro, Campo

OPERAZIONI: Join, selezione, proiezione, ridenominazione, operatori insiemistici Unione e Differenza

LOGICA: Seleziono le nazioni dei giocatori che hanno perso una partita o in casa o fuori casa, e sottraggo il risultato di quest'espressione a tutte le nazioni dei tennisti

IA, CA sono le ridenominazioni di tutti gli attributi delle tabelle I, C aggiungendo una 'A' al nome dell'attributo

IB, CB sono le ridenominazioni di tutti gli attributi delle tabelle I, C aggiungendo una 'B' al nome dell'attributo

(PROJ Nazione T) -

(PROJ Nazione (SEL Set1A < Set2A ((T JOIN T.CF = IA.Gioc1A IA) JOIN Ref_CampoA = CA.NCampA CA))

U

PROJ Nazione (SEL Set1B > Set2B ((T JOIN T.CF = IB.Gioc1B IB) JOIN Ref_CampoB = CB.NCampB CB)))

MYSQL

```

1 •  SELECT DISTINCT t.nazione
2   FROM tennista t
3   WHERE t.nazione NOT IN (
4     SELECT t.nazione
5       FROM incontro i1, tennista t1
6      WHERE i1.giocatore1 = t1.cf AND i1.set1 < i1.set2)
7   AND t.nazione NOT IN (
8     SELECT t2.nazione
9       FROM incontro i2, tennista t2
10      WHERE i2.giocatore2 = t2.cf AND i2.set1 > i2.set2)

```

Fig. 15.8 – Codice MySQL per l'interrogazione 4 (nessuna istanza risultante)

5. Selezionare il campo in erba che ha ospitato il maggior numero di incontri

SQL

INFORMAZIONI: Incontro, Campo

OPERAZIONI: Join, selezione, Grou By, operatore aggregato Count, interrogazione innestata nella selezione del raggruppamento

LOGICA: Con l'interrogazione B seleziono tutti gli incontri disputati sull'erba e per ogni campo conto il numero di incontri. Con l'interrogazione A seleziono solo i campi di tipo erba per i quali il conteggio degli incontri disputati sia il massimo (ovvero sia maggiore uguale al risultato dell'interrogazione B)

```
SELECT NCampo
  FROM Incontro I, Campo C
 WHERE I.Ref_Campo = C.NCampo AND C.Tipo = 'erba'
 GROUP BY NCampo
 HAVING count(*) >= ( SELECT count(*)
    FROM Incontro I1, Campo C1
   WHERE I1.Ref_Campo = C1.NCampo AND C1.Tipo = 'erba'
 GROUP BY C1.NCampo )
```

MYSQL

```
1 •  SELECT c1.nome_campo
2   FROM incontro i1, campo c1
3   WHERE i1.nome_campo_i = c1.nome_campo AND c1.tipo = 'erba'
4   GROUP BY c1.nome_campo
5   HAVING COUNT(*) >= ALL (
6       SELECT COUNT(*)
7       FROM incontro i2, campo c2
8       WHERE i2.nome_campo_i = c2.nome_campo AND c2.tipo = 'erba'
9       GROUP BY c2.nome_campo);
```

00% 1:10

Result Grid | Filter Rows: Search Export:

nome_campo
Comet Club

Fig. 15.9 – Codice MySQL per l'interrogazione 5 e istanza risultante

ESERCITAZIONE 16 - EVENTI

11/06/2017

Dato il seguente schema della base di dati EVENTI:

MANIFESTAZIONE (Cod_M, Nome_M)

LUOGO (Nome_L, Indirizzo, Città)

SPETTACOLO (Cod_M_S, Num, ore_inizio, Nome_L_S, Giorno)

FK: Cod_M_S REFERENCES Manifestazione (Cod_M)

FK: Nome_L_S REFERENCES Luogo (Nome_L)

Una Manifestazione (Cod_M) si svolge durante uno Spettacolo che ha sede in un determinato Luogo (Nome_L) (Fig. 16.1).

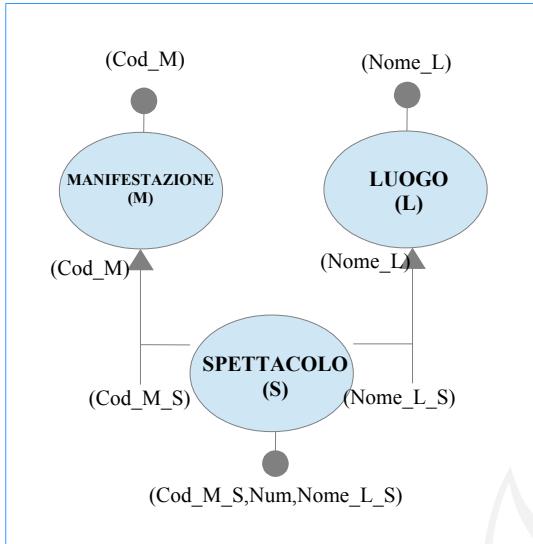
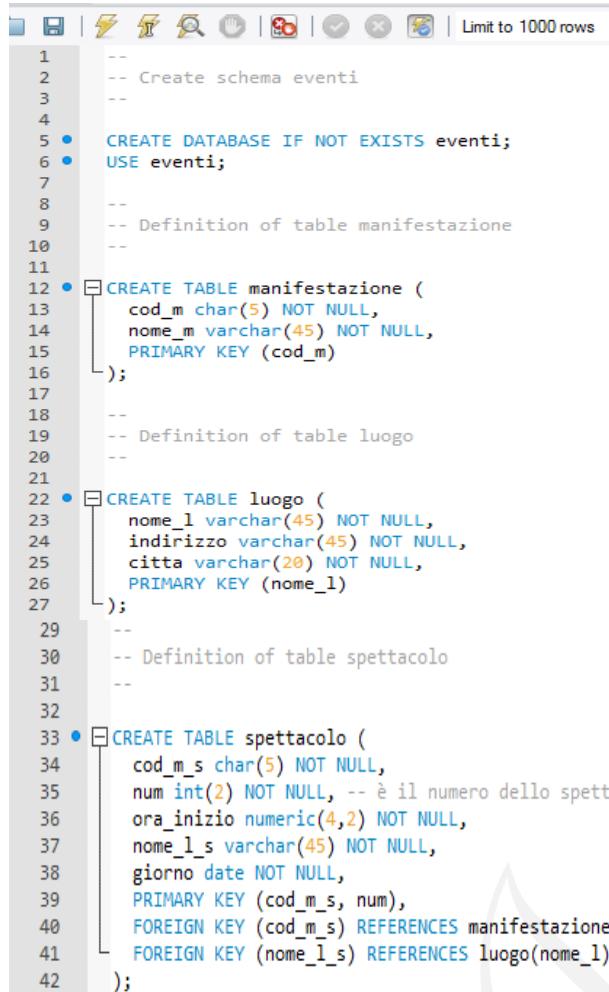


Fig. 16.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1, 2) :

1. Selezionare il codice e il nome delle manifestazioni che non hanno interessato luoghi della città di Modena;
2. Selezionare i nomi dei luoghi che hanno ospitato almeno uno spettacolo di ciascuna manifestazione;
3. Selezionare il nome dei luoghi che, in una certa data, ospitano più di 3 spettacoli dopo le ore 15;
4. Selezionare, per ogni luogo, il numero total delle manifestazioni e il numero totale degli spettacoli ospitati;
5. Selezionare i codici delle manifestazioni i cui spettacoli sono iniziati sempre dopo le ore 15.

Fig. 16.2 – Codice MySQL per la creazione del database Rifornimenti

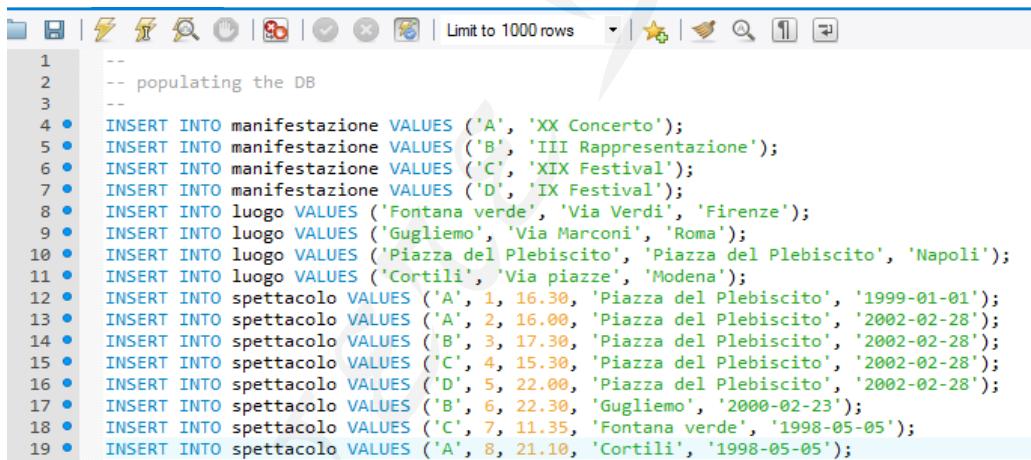


```

1   --
2   -- Create schema eventi
3   --
4
5 • CREATE DATABASE IF NOT EXISTS eventi;
6 • USE eventi;
7
8
9   -- Definition of table manifestazione
10
11
12 • CREATE TABLE manifestazione (
13     cod_m char(5) NOT NULL,
14     nome_m varchar(45) NOT NULL,
15     PRIMARY KEY (cod_m)
16 );
17
18
19   -- Definition of table luogo
20
21
22 • CREATE TABLE luogo (
23     nome_l varchar(45) NOT NULL,
24     indirizzo varchar(45) NOT NULL,
25     citta varchar(20) NOT NULL,
26     PRIMARY KEY (nome_l)
27 );
28
29   -- Definition of table spettacolo
30
31
32
33 • CREATE TABLE spettacolo (
34     cod_m_s char(5) NOT NULL,
35     num int(2) NOT NULL, -- è il numero dello spettacolo all'interno della manifestazione cod_m
36     ora_inizio numeric(4,2) NOT NULL,
37     nome_l_s varchar(45) NOT NULL,
38     giorno date NOT NULL,
39     PRIMARY KEY (cod_m_s, num),
40     FOREIGN KEY (cod_m_s) REFERENCES manifestazione(cod_m),
41     FOREIGN KEY (nome_l_s) REFERENCES luogo(nome_l)
42 );

```

Fig. 16.3 – Codice MySQL per la popolazione del database

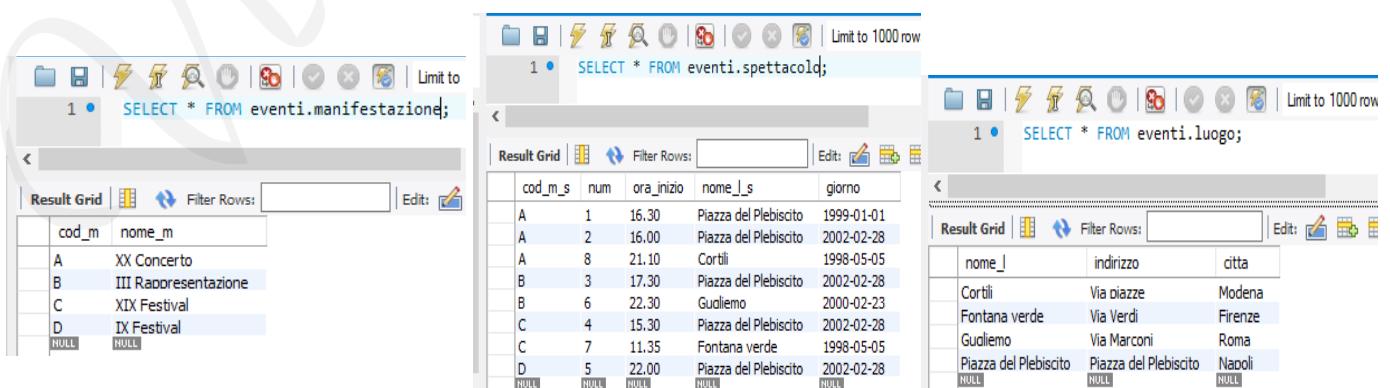


```

1   --
2   -- populating the DB
3
4 • INSERT INTO manifestazione VALUES ('A', 'XX Concerto');
5 • INSERT INTO manifestazione VALUES ('B', 'III Rappresentazione');
6 • INSERT INTO manifestazione VALUES ('C', 'XIX Festival');
7 • INSERT INTO manifestazione VALUES ('D', 'IX Festival');
8 • INSERT INTO luogo VALUES ('Fontana verde', 'Via Verdi', 'Firenze');
9 • INSERT INTO luogo VALUES ('Gugliemo', 'Via Marconi', 'Roma');
10 • INSERT INTO luogo VALUES ('Piazza del Plebiscito', 'Piazza del Plebiscito', 'Napoli');
11 • INSERT INTO luogo VALUES ('Cortili', 'Via piazze', 'Modena');
12 • INSERT INTO spettacolo VALUES ('A', 1, 16.30, 'Piazza del Plebiscito', '1999-01-01');
13 • INSERT INTO spettacolo VALUES ('A', 2, 16.00, 'Piazza del Plebiscito', '2002-02-28');
14 • INSERT INTO spettacolo VALUES ('B', 3, 17.30, 'Piazza del Plebiscito', '2002-02-28');
15 • INSERT INTO spettacolo VALUES ('C', 4, 15.30, 'Piazza del Plebiscito', '2002-02-28');
16 • INSERT INTO spettacolo VALUES ('D', 5, 22.00, 'Piazza del Plebiscito', '2002-02-28');
17 • INSERT INTO spettacolo VALUES ('B', 6, 22.30, 'Gugliemo', '2000-02-23');
18 • INSERT INTO spettacolo VALUES ('C', 7, 11.35, 'Fontana verde', '1998-05-05');
19 • INSERT INTO spettacolo VALUES ('A', 8, 21.10, 'Cortili', '1998-05-05');

```

Fig. 16.4 – Istanze della base di dati



cod_m	nome_m
A	XX Concerto
B	III Rappresentazione
C	XIX Festival
D	IX Festival
NULL	NULL

cod_m_s	num	ora_inizio	nome_l_s	giorno
A	1	16.30	Piazza del Plebiscito	1999-01-01
A	2	16.00	Piazza del Plebiscito	2002-02-28
A	8	21.10	Cortili	1998-05-05
B	3	17.30	Piazza del Plebiscito	2002-02-28
B	6	22.30	Gugliemo	2000-02-23
C	4	15.30	Piazza del Plebiscito	2002-02-28
C	7	11.35	Fontana verde	1998-05-05
D	5	22.00	Piazza del Plebiscito	2002-02-28
NULL	NULL	NULL	NULL	NULL

nome_l	indirizzo	citta
Cortili	Via piazze	Modena
Fontana verde	Via Verdi	Firenze
Gugliemo	Via Marconi	Roma
Piazza del Plebiscito	Piazza del Plebiscito	Napoli
NULL	NULL	NULL

1. Selezionare il codice e il nome delle manifestazioni che non hanno interessato luoghi della città di Modena.

SQL

INFORMAZIONI: Manifestazion, Luogo, Spettacolo

OPERAZIONI: 1 interrogazione annidata con passaggio di parametri, join, selezione, Operatore NOT EXISTS

LOGICA: Con l'interrogazione B ottengo le informazioni sugli spettacoli che si sono svolti a Modena per ogni manifestazione passata come parametro dall'interrogazione A, tramite il NOT EXISTS seleziono solo quelle manifestazioni per le quali l'interrogazione B restituisce una relazione vuota.

```
SELECT *
A  FROM Manifestazione M
    WHERE NOT EXIST(SELECT *
B      FROM Luogo L Spettacolo S
          WHERE M.Cod_M=S.Cod_M_S AND L.Nome=S.Nome_L_S
              AND L.Città='Modena' )
```

ALGEBRA

INFORMAZIONI: Manifestazione, Spettacolo, Luogo

OPERAZIONI: Sottrazione, join, selezione, proiezione

LOGICA: A tutti i codici delle manifestazioni sottraggo quelli presenti in spettacoli che hanno avuto luogo a Modena.

PROJ $\text{Cod}_M, \text{Nome}_M | (M \text{ JOIN}_{\text{M.Cod}_M=\text{Cod}_M_S} (\text{PROJ}_{\text{COD}_M_S}(S) - (\text{PROJ}_{\text{Cod}_M_S}(\text{SEL}_{\text{Città}=\text{Modena}}(S \text{ JOIN}_{\text{S.Nome}_L_S=\text{L.Nome}_L} L))))))$

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
-- a) selezionare il codice e il nome delle manifestazioni che non hanno interessato
-- luoghi della città di Modena
SELECT *
FROM manifestazione m
WHERE NOT EXISTS( SELECT *
                  FROM luogo l,
                      spettacolo s
                 WHERE m.cod_m = s.cod_m_s
                   AND l.nome_l = s.nome_l_s
                   AND l.città = 'Modena');
```

The results grid shows the following data:

cod_m	nome_m
B	III Rapresentazione
C	XIX Festival
D	IX Festival
HULL	HULL

Fig. 16.5 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

2. Selezionare i nomi dei luoghi che hanno ospitato almeno uno spettacolo di ciascuna manifestazione.

SQL

INFORMAZIONI: Manifestazion, Spettacolo

OPERAZIONI: 2 interrogazione annidata con passaggio di parametri, join, Operatore NOT EXISTS

LOGICA: Con l'interrogazione C otteniamo le informazioni degli spettacoli che si sono svolti nel luogo passato come parametro dall'interrogazione A e riguardanti la manifestazione passata come parametro dall'interrogazione B. Dunque con la doppia negazione selezioniamo i nomi dei luoghi per i quali non esiste una manifestazione che non sia apparsa in uno spettacolo tenutosi presso essi.

```
A  SELECT DISTINCT S1.Nome_L_S
    FROM Spettacolo S1
    WHERE NOT EXIST(SELECT *
B      FROM Manifestazion M
          WHERE NOT EXISTS (SELECT *
C              FROM Spettacolo S2
                  WHERE S1.Nome_L_S=S2.Nome_L_S AND
                      M.Cod_M=S2.Cod_M_S ))
```

ALGEBRA

INFORMAZIONI: Spettacolo, Manifestazione

OPERAZIONI: Join, Sottrazione, Proiezione

LOGICA: Sottraggo a tutti i luoghi dove si sono effettuati gli spettacoli quelli in cui non si è effettuata almeno una manifestazione, ottenuti sottraendo al risultato del join fra S ed M (che contiene anche tuple false) le tuple reali.

(PROJ_{Nome_L_S}S) - (PROJ_{Nome_L_S}((S JOIN M) - (S JOIN_{S.Cod_M_S=M.Cod_M} M)))

MYSQL

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 -- b)selezionare i nomi dei luoghi che hanno ospitato tutte le manifestazioni
2 -- (hanno ospitato almeno uno spettacolo di ciascuna manifestazione)
3
4 • SELECT DISTINCT
5     s1.nome_l_s
6     FROM
7         spettacolo s1
8     WHERE
9         NOT EXISTS( SELECT
10             *
11             FROM
12                 manifestazione m
13             WHERE
14                 NOT EXISTS( SELECT
15                     *
16                     FROM
17                         spettacolo s2
18                     WHERE
19                         s1.nome_l_s = s2.nome_l_s
20                         AND m.cod_m = s2.cod_m_s));
21
```

The result grid shows one row:

nome_l_s
Piazza del Plebiscito

Fig. 16.6 – Codice MySQL per l'interrogazione 2 e le istanze risultanti

3. Selezionare il nome dei luoghi che, in una certa data, ospitano più di 3 spettacoli dopo le ore 15.

SQL

INFORMAZIONI: Manifestazion, Spettacolo

OPERAZIONI: 2 interrogazione annidata con passaggio di parametri, join, Operatore NOT EXISTS

LOGICA: Con l'interrogazione B ottengo il conteggio degli spettacoli avvenuti nei luoghi e nei giorni passati come parametri dall'interrogazione A e che sono iniziati dopo le ore 15. Con l'interrogazione A seleziono quei luoghi per cui 3 sia minore del risultato dell'interrogazione B.

A **SELECT DISTINCT S1.Nome_L_S**
 FROM Spettacolo S1
 WHERE S1.Ora_Inizio >15 AND 3< (SELECT COUNT(*)
 FROM Spettacolo S2
 WHERE S1.Nome_L_S=S2.Nome_L_S AND S1.Giorno=S2.Giorno AND
 S2.Ora_Inizio>15)

MYSQL

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 -- c) selezionare il nome dei luoghi che, in una certa data, ospitano più di tre
2 -- spettacoli dopo le ore 15
3
4 • SELECT DISTINCT
5     s1.nome_l_s
6     FROM
7         spettacolo s1
8     WHERE
9         s1.ora_inizio > 15
10        AND 3 < (SELECT
11            COUNT(*)
12            FROM
13                spettacolo s2
14                WHERE
15                    s1.nome_l_s = s2.nome_l_s
16                    AND s1.giorno = s2.giorno
17                    AND s2.ora_inizio > 15);
18
```

The result grid shows one row:

nome_l_s
Piazza del Plebiscito

Fig. 16.7 – Codice MySQL per l'interrogazione 3 e le istanze risultanti

4. Selezionare, per ogni luogo, il numero total delle manifestazioni e il numero totale degli spettacoli ospitati.

SQL

INFORMAZIONI: Manifestazion, Spettacolo

OPERAZIONI: 2 interrogazione annidata con passaggio di parametri, join, Operatore NOT EXISTS

LOGICA: Per ogni valore diverso dell'attributo Nome_L_S ottengo una sottotabella di cui conto le diverse manifestazioni e il numero di tuple.

```
SELECT DISTINCT S.Nome_L_S, COUNT(DISTINCT S.Cod_M_S) AS N_MAN, COUNT(*) AS N_Spet
FROM Spettacolo S
GROUP BY S.Nome_L_S
```

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
1 -- d) selezionare, per ogni luogo, il numero total delle manifestazioni e il numero
2 -- totale degli spettacoli ospitati
3
4 • SELECT
5     s.nome_l_s,
6     COUNT(DISTINCT s.cod_m_s) AS n_man,
7     COUNT(*) AS n_spet
8 FROM
9     spettacolo s
10 GROUP BY s.nome_l_s;
```

The results grid displays the following data:

nome_l_s	n_man	n_spet
Cortili	1	1
Fontana verde	1	1
Gudliemo	1	1
Piazza del Plebiscito	4	5

Fig.16.8 – Codice MySQL per l'interrogazione 4 e le istanze risultanti

5. Selezionare i codici delle manifestazioni i cui spettacoli sono iniziati sempre dopo le ore 15.

SQL

INFORMAZIONI: Manifestazion, Spettacolo

OPERAZIONI: 2 interrogazione annidata con passaggio di parametri, join, Operatore NOT EXISTS

LOGICA: Con l'interrogazione B ottengo le informazioni sugli spettacoli relativi alle manifestazioni passati come parametri dall'interrogazione A e che hanno avuto inizio prima delle ore 15. Con l'interrogazione A proietto i codici delle mostre tali per cui l'interrogazione B restituisce una relazione vuota.

```
A   SELECT DISTINCT S1.Cod_M_S
      FROM Spettacolo S1
      WHERE NOT EXIST(SELECT *
                      B   FROM Spettacolo S2
                      WHERE S1.Cod_M_S=S2.Cod_M_S AND S2.Ora_Inizio<=15)
```

MYSQL

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
7
8 • SELECT DISTINCT
9     s1.cod_m_s
10    FROM
11        spettacolo s1
12    WHERE
13        NOT EXISTS( SELECT
14            *
15            FROM
16                spettacolo s2
17            WHERE
18                s1.cod_m_s = s2.cod_m_s
19                AND s2.ora_inizio <= 15);
```

The results grid displays the following data:

cod_m_s
A
B
D

Fig. 16.9 – Codice MySQL per l'interrogazione 5 e le istanze risultanti

ESERCITAZIONE 17 – GOLF

Dato il seguente schema per la base di dati GOLF:

GARA (CG, NomeCampo, Livello)
GIOCATORE (CF, NomeG, Nazione)
PARTECIPA (Ref_CG, Ref_CF, Punteggio)
 FK: Ref_CG REFERENCES Gara (CG)
 FK: Ref_CF REFERENCES Giocatore (CF)

Un Giocatore (CF) di una certa Nazione partecipa ad una Gara (CG) di un certo Livello, ottenendo un determinato Punteggio (fig. 17.1). La Gara è vinta dal Giocatore che ottiene il Punteggio più basso.

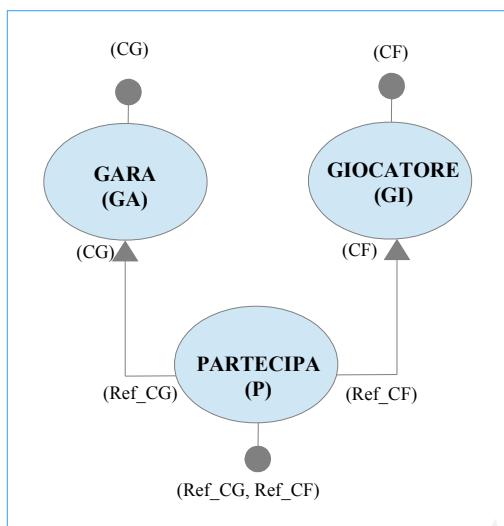


Fig. 17.1 – Modello dello schema di base di dati

Con riferimento allo schema in fig. 19.1, effettuare le seguenti interrogazioni (Algebra Relazionale 1, 2, 3):

1. Selezionare i dati dei giocatori di golf che hanno partecipato ad almeno una gara disputata a livello 'nazionale';
2. Selezionare le nazioni in cui tutti i giocatori hanno ottenuto un punteggio minore o uguale a 0 nelle gare disputate;
3. Selezionare i dati dei giocatori di golf che hanno partecipato a tutte le gare disputate a livello 'nazionale';
4. Selezionare i dati dei giocatori di golf che hanno vinto almeno una gara disputata a livello 'internazionale';
5. 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;

```

1 • CREATE DATABASE IF NOT EXISTS golf;
2 • USE golf;
3
4 -- Definition of table gara
5
6 • ┌─ CREATE TABLE gara (
7   cg char(5) NOT NULL,
8   nome_campo varchar(45) NOT NULL,
9   livello varchar(20),
10  PRIMARY KEY (cg)
11 );
12 -- Definition of table giocatore
13 • ┌─ CREATE TABLE giocatore (
14   cf char(16) NOT NULL,
15   nome varchar(75) NOT NULL,
16   nazione char(3) NOT NULL,
17   PRIMARY KEY (cf)
18 );
19 -- Definition of table partecipa
20 • ┌─ CREATE TABLE partecipa (
21   ref_cg char(5) NOT NULL,
22   ref_cf char(16) NOT NULL,
23   punteggio int(3) NOT NULL,
24   PRIMARY KEY (ref_cg, ref_cf),
25   FOREIGN KEY (ref_cg) REFERENCES gara(cg),
26   FOREIGN KEY (ref_cf) REFERENCES giocatore(cf)
27 );

```

Fig.17.2 – Codice MySQL per la creazione del database Golf

Fig.17.3 – Codice MySQL per la popolazione del database

```

1 • INSERT INTO gara VALUES('G1', 'Golf Club', 'Nazionale');
2 • INSERT INTO gara VALUES('G2', 'Star Golf Club', 'Internazionale');
3 • INSERT INTO gara VALUES('G3', 'ABC Golf', 'Nazionale');
4 • INSERT INTO giocatore VALUES('GCC', 'Orazio Genova', 'ITA');
5 • INSERT INTO giocatore VALUES('MRR', 'Mario Rossi', 'ITA');
6 • INSERT INTO giocatore VALUES('OSS', 'Osvaldo Marconi', 'ITA');
7 • INSERT INTO giocatore VALUES('COO', 'Carlo Orazio', 'ITA');
8 • INSERT INTO giocatore VALUES('MAA', 'Michele Antonelli', 'ITA');
9 • INSERT INTO giocatore VALUES('LMM', 'Lorenzo Meucci', 'ITA');
10 • INSERT INTO giocatore VALUES('SSH', 'John Smith', 'USA');
11 • INSERT INTO partecipa VALUES('G2', 'GCC', -13);
12 • INSERT INTO partecipa VALUES('G2', 'MRR', -19);
13 • INSERT INTO partecipa VALUES('G1', 'MRR', -1);
14 • INSERT INTO partecipa VALUES('G3', 'MRR', -2);
15 • INSERT INTO partecipa VALUES('G2', 'SSH', 1);
16 • INSERT INTO partecipa VALUES('G2', 'OSS', -15);
17 • INSERT INTO partecipa VALUES('G2', 'COO', -24);
18 • INSERT INTO partecipa VALUES('G2', 'MAA', -20);
19 • INSERT INTO partecipa VALUES('G2', 'LMM', -19);
20

```

Fig.17.4 – Istanze del database

cg	nome_campo	livello	cf	nome	nazione	ref_cg	ref_cf	punteggio
G1	Golf Club	Nazionale	COO	Carlo Orazio	ITA	G1	MRR	-1
G2	Star Golf Club	Internazionale	GCC	Orazio Genova	ITA	G2	COO	-24
G3	ABC Golf	Nazionale	LMM	Lorenzo Meucci	ITA	G2	GCC	-13
gara 1			MAA	Michele Antonelli	ITA	G2	LMM	-19
			MRR	Mario Rossi	ITA	G2	MAA	-20
			OSS	Osvaldo Marconi	ITA	G2	MRR	-19
			SSH	John Smith	USA	G2	OSS	-15
			giocatore 1			G2	SSH	1
						G3	MRR	-2
			partecipa 1					

1. Selezionare i dati dei giocatori di golf che hanno partecipato ad almeno una gara disputata a livello 'nazionale'

SQL

INFORMAZIONI: In Gara e Giocatore, utilizzo Partecipa come tabella di collegamento

OPERAZIONI: join, selezione e proiezione

LOGICA: Seleziono tutti i giocatori che hanno disputato almeno una gara facendo il join fra le tre relazioni, con la selezione prendo solo le gare di livello nazionale e infine proietto tutti gli attributi del giocatore

SELECT DISTINCT Giocatore.*

FROM Gara GA, Partecipa P, Giocatore GI

WHERE GA.CG = P.Ref_CG AND P.Ref_CF = GI.CF AND GA.Livello = 'nazionale'

ALGEBRA

INFORMAZIONI: Gara, Partecipa, Giocatore

OPERAZIONI: join, selezione e proiezione

LOGICA: In questo caso la logica è uguale a quella utilizzata per SQL

PROJ CF, NomeG, Nazione (SEL Livello = 'nazionale' ((GA JOIN GA.CG = P.Ref_CG P) JOIN Ref_CF = CF GI))

MYSQL

```

1 • SELECT DISTINCT GI.*
  FROM Gara GA, Partecipa P, Giocatore GI
 WHERE GA.cg = P.ref_cg AND P.ref_cf = GI.cf AND GA.Livello = 'Nazionale'

```

cf	nome	nazione
MRR	Mario Rossi	ITA

Fig.17.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare le nazioni in cui tutti i giocatori hanno ottenuto un punteggio minore o uguale a 0 nelle gare disputate

SQL

INFORMAZIONI: Giocatore, Partecipa

OPERAZIONI: Un'interrogazione innestata, operatore insiemistico Not In, join e selezione

LOGICA: L'interrogazione B restituisce tutte le nazioni dei giocatori che hanno ottenuto un punteggio positivo, con l'interrogazione A selezioniamo le nazioni dei giocatori che non si trovano fra i risultati dell'interrogazione B, escludendo i duplicati.

```
A   SELECT DISTINCT GI.Nazione
  FROM Giocatore GI
 WHERE GI.Nazione NOT IN ( SELECT GI1.Nazione
                           B      FROM Partecipa P, Giocatore GI1
                           WHERE P.Ref_CF = GI1.CF AND P.Punteggio > 0 )
```

ALGEBRA

INFORMAZIONI: Giocatore, Partecipa

OPERAZIONI: Un join, una selezione, una proiezione, una differenza

LOGICA: Con il join ottengo tutte le informazioni sulle partite a cui ha partecipato un giocatore, con la selezione ottengo quelli il cui punteggio è positivo. Proietto solo gli attributi di giocatore in modo da rendere possibile la differenza e ottengo le tuple in cui i giocatori non hanno punteggio positivo.

PROJ GI.Nazione (GI – (**PROJ** CF, NomeG, Nazione (**SEL** P.Punteggio > 0 (**P JOIN** P.Ref_CF = GI.CF GI)))))

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the following SQL query:

```
1 •  SELECT DISTINCT GI.nazione
  2   FROM giocatore GI
  3   WHERE GI.nazione NOT IN (SELECT GI.nazione
  4                             FROM partecipa P, giocatore GI
  5                             WHERE GI.cf = P.ref_cf AND P.punteggio > 0)
  6
```

Below the code editor is a results grid titled "Result Grid". It has a single column labeled "nazione". The result is a single row with the value "ITA".

Fig.17.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare i dati dei giocatori di golf che hanno partecipato a tutte le gare disputate a livello 'nazionale'

SQL

INFORMAZIONI: Giocatore, Gara e Partecipa

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametro, doppia negazione con Not Exists

LOGICA: Effettuo la divisione utilizzando il doppio Not exists. Nell'interrogazione C ottengo le tuple per ogni coppia di giocatore e gara passati come parametri , con la B seleziono la gara per la quale l'interrogazione C da luogo ad una relazione vuota e con la A selezioni i giocatori per i quali l'interrogazione B da luogo ad una relazione vuota.

```
A   SELECT GI.*
  FROM Giocatore GI
 WHERE NOT EXISTS ( SELECT *
                       B      FROM Gara GA
                       WHERE GA.Livello = 'Nazionale' AND NOT EXISTS ( SELECT *
                                                               C      FROM Partecipa P
                                                               WHERE P.Ref_CG = GA.CG AND P.Ref_CF =
                                                               GI.CF ) )
```

ALGEBRA

INFORMAZIONI: Giocatore, Gara e Partecipa

OPERAZIONI: una divisione, un join, una ridenominazione, una selezione e tre proiezioni

LOGICA: A differenza di SQL non posso utilizzare interrogazioni innestate quindi effettuo una divisione fra Partecipa e Gara, avendo selezionato solo le gare di livello nazionale. Infine ottengo tutte le informazioni sul Giocatore con un join.

PROJ CF, NomeG, Nazione (GI **JOIN** GI.CF = Ref_CF (

(**PROJ** P.Ref_CG, P.Ref_CF P)

⋮

(**PROJ** Ref_CG (**REN** Ref_CG ← CG (**SEL** GA.Livello = 'Nazionale' GA))

))

MYSQL

```

1 •  SELECT GI.*
2   FROM giocatore GI
3   WHERE NOT EXISTS (
4     SELECT *
5       FROM gara GA
6      WHERE GA.livello = 'Nazionale' AND NOT EXISTS (
7        SELECT *
8          FROM partecipa P
9         WHERE P.ref_cg = GA.cg AND P.ref_cf = GI.cf)
10    )

```

Result Grid | Filter Rows: Search | Edit: Export

cf	nome	nazione
MRR	Mario Rossi	ITA
NULL	NULL	NULL

Fig.17.7 – Codice MySQL per l'interrogazione 3 e istanze risultanti

4. Selezionare i dati dei giocatori di golf che hanno vinto almeno una gara disputata a livello 'internazionale'

INFORMAZIONI: Gara, Partecipa, Giocatore

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, join, selezione, operatore matematico minore uguale e operatore All

LOGICA: Con l'interrogazione B ottengo le tuple per ogni gara passata come parametro, con la A seleziono il giocatore che ha partecipato ad una gara di livello internazionale il cui il punteggio del giocatore è minore di tutti gli altri, ovvero il giocatore che ha vinto tale gara.

```

SELECT DISTINCT GI.*
A  FROM Gara GA, Partecipa P, Giocatore GI
WHERE GA.CG = P.Ref_CG AND P.Ref_CF = GI.CF AND GA.Livello = 'Internazionale'
      AND P.Punteggio <= ALL ( A SELECT P1.Punteggio
                                FROM Partecipa P1
                                WHERE P1.Ref_CG = P.Ref_CG)

```

```

1 •  SELECT GI.*
2   FROM gara GA, partecipa P, giocatore GI
3   WHERE GA.cg = P.ref_cg AND P.ref_cf = GI.cf AND GA.livello = 'Internazionale' AND P.Punteggio <= ALL (
4     SELECT P1.punteggio
5       FROM partecipa P1
6      WHERE P1.ref_cg = P.ref_cg)

```

Result Grid | Filter Rows: Search | Export:

cf	nome	nazione
COO	Carlo Orazio	ITA

Fig.17.8 – Codice MySQL per l'interrogazione 4 e istanze risultanti

5. 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.

INFORMAZIONI: Gara, Partecipa, Giocatore

OPERAZIONI: Un Group By, Order By, selezione, due join, operatore aggregato Count, operatore matematico maggiore

LOGICA: Suddivido la relazione, ottenuta col join fra le tre tabelle, in tante sottotabelle quante sono le nazioni diverse dei giocatori che hanno partecipato a gare di livello internazionale. Di ogni nazione verifico che il numero di partecipanti è maggiore di 5 e ne calcolo il punteggio medio. Infine ordino il risultato.

```

SELECT GI.Nazione, AVG(P.Punteggio) as PunteggioMedio
FROM Gara GA, Partecipa P, Giocatore GI
WHERE GA.CG = P.Ref_CG AND P.Ref_CF = GI.CF AND GA.Livello = 'Internazionale'
GROUP BY GI.Nazione
HAVING COUNT(DISTINCT P.Ref_CF) > 5
ORDER BY 2 DESC

```

MYSQL

```
1 •  SELECT GI.nazione, AVG(P.punteggio)as punteggio_medio
2   FROM gara GA, partecipa P, giocatore GI
3  WHERE GA.cg = P.ref_cg AND P.ref_cf = GI.cf AND GA.livello = 'internazionale'
4  GROUP BY GI.nazione
5  HAVING COUNT(DISTINCT p.ref_cf) > 5
6  ORDER BY 2 DESC;
```

100%

54:1

Result Grid



Filter Rows:

Search

Export:



nazione	punteggio_medio
ITA	-18.3333

Fig.17.9 – Codice MySQL per l'interrogazione 5 e istanze risultanti

ESERCITAZIONE 18 - MUSICA

06/06/2017

Dato il seguente schema della base di dati Musica:

CD (Cod_Cd, Autore, Casa_Disco)
CLIENTE (Ntess, Nome, Indirizzo)
ACQUISTO (Cod_Cd_A, Ntess_A, Data_Acquisto, Qty)
 FK: Cod_Cd_A REFERENCES CD (Cod_Cd)
 FK: Ntess_A REFERENCES Cliente (Ntess)

Il Cliente (Ntess, Nome, Indirizzo) Acquista dei CD (Cod_Cd, Autore, Casa_Disco) (Fig 18.1).

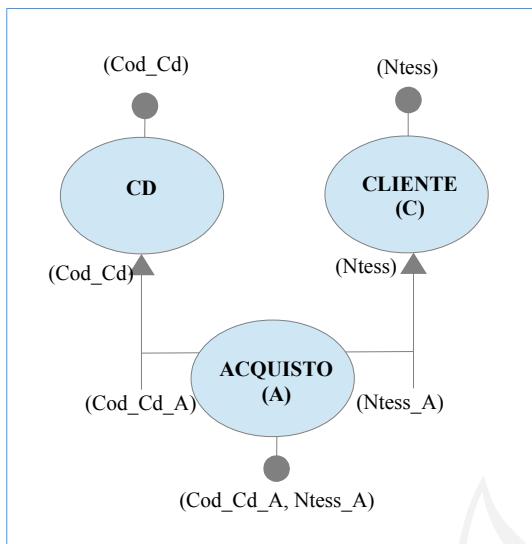


Fig. 18.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1,2) :

1. Selezionare tutti i dati dei clienti che dopo il 01/01/1997 non hanno acquistato nessun cd dalla casa discografica 'DiscoJolli';
2. Selezionare il numero di tessera dei clienti che hanno acquistato tutti i cd dell'autore 'Francesco Guccini';
3. Selezionare, per ogni cd, il numero totale delle copie vendute;
4. Selezionare, per ogni casa discografica, il numero tessera del cliente che ha acquistato il maggior numero di copie di quel cd di quella casa.

Fig. 18.2 – Codice MySQL per la creazione del database Rifornimenti

```

2   -- Create schema musica
3   --
4
5 • CREATE DATABASE IF NOT EXISTS musica;
6 • USE musica;
7
8   --
9   -- Definition of table fornitore
10  --
11
12 • CREATE TABLE cd (
13     cod_cd int(6) NOT NULL,
14     autore varchar(75) NOT NULL,
15     casa_disco varchar(30) NOT NULL,
16     PRIMARY KEY (cod_cd)
17 );
  
```

```

21   -- Definition of table cliente
22   --
23
24 • CREATE TABLE cliente (
25     ntess char(5) NOT NULL,
26     nome varchar(75) NOT NULL,
27     indirizzo varchar(45) NOT NULL,
28     PRIMARY KEY (ntess)
29 );
30
31   --
32   -- Definition of table acquisto
33   --
34
35 • CREATE TABLE acquisto (
36     cod_cd_a int(6) NOT NULL,
37     ntess_a char(5) NOT NULL,
38     data_acquisto date NOT NULL,
39     qty int(2) NOT NULL,
40     PRIMARY KEY (cod_cd_a, ntess_a),
41     FOREIGN KEY (cod_cd_a) REFERENCES cd(cod_cd),
42     FOREIGN KEY (ntess_a) REFERENCES cliente(ntess)
43 );
  
```

Fig.18 .3 – Codice MySQL per la popolazione del database

```

1 -- il cliente identificato da ntess ha acquistato, in una certa data_acquisto,
2 -- un certo numero qty di copie del compact disk cod_cd_a
3
4 --
5 -- populating the DB
6 --
7 • INSERT INTO cd VALUES(1, 'Orazio', 'DiscoJolli');
8 • INSERT INTO cd VALUES(2, 'Ernesto', 'DiscoLuna');
9 • INSERT INTO cd VALUES(3, 'Francesco Guccini', 'DiscoLuna');
10 • INSERT INTO cd VALUES(4, 'Francesco Guccini', 'DiscoLuna');
11 • INSERT INTO cliente VALUES('T1', 'Giovanna', 'Viale Michelino');
12 • INSERT INTO cliente VALUES('T2', 'Ludovica', 'Via Roma');
13 • INSERT INTO cliente VALUES('T3', 'Luca', 'Via Napoli');
14 • INSERT INTO acquisto VALUES(1, 'T1', '1998-04-06', 2);
15 • INSERT INTO acquisto VALUES(2, 'T2', '2001-01-28', 1);
16 • INSERT INTO acquisto VALUES(1, 'T2', '1996-01-28', 1);
17 • INSERT INTO acquisto VALUES(3, 'T3', '1991-01-04', 1);
18 • INSERT INTO acquisto VALUES(3, 'T1', '1993-05-23', 1);
19 • INSERT INTO acquisto VALUES(4, 'T1', '1996-01-21', 1);
```

Fig. 18.4 – Istanze della base di dati

Result Grid Filter Rows:			
cod_cd_a	ntess_a	data_acquisto	qty
1	T1	1998-04-06	2
1	T2	1996-01-28	1
2	T2	2001-01-28	1
3	T1	1993-05-23	1
3	T3	1991-01-04	1
4	T1	1996-01-21	1
NULL	NULL	NULL	NULL

Result Grid Filter Rows:		
cod_cd	autore	casa_disco
1	Orazio	DiscoJolli
2	Ernesto	DiscoLuna
3	Francesco Guccini	DiscoLuna
4	Francesco Guccini	DiscoLuna
NULL	NULL	NULL

Result Grid Filter Rows:		
ntess	nome	indirizzo
T1	Giovanna	Viale Michelino
T2	Ludovica	Via Roma
T3	Luca	Via Napoli
NULL	NULL	NULL

1. Selezionare tutti i dati dei clienti che dopo il 01/01/1997 non hanno acquistato nessun cd dalla casa discografica 'DiscoJolli'.

SQL

INFORMAZIONI: Cliente, Acquista, Cd

OPERAZIONI: 1 interrogazione annidata, Operatore NOT IN, join, selezione

LOGICA: Dal join dell'interrogazione B ottengo tutti i cd, che sono stati acquistati dai clienti, della casa discografica DiscoJolli, con l'interrogazione A prendo tutti gli attributi di clienti, tramite il numero tessera, riguardanti quelli che non hanno acquistato cd della casa discografica DiscoJolli dopo il 10-01-1997.

```

SELECT *
A FROM Cliente C
WHERE C.Ntess NOT IN (SELECT A.Ntess_A
B   FROM Acquista A, Cd
      WHERE A.Cod_Cd_A=Cd.Cod_Cd AND Cd.Casa_Disco='DiscoJolli' AND A.Data_Acquisto >
            '1997-01-01' )
```

ALGEBRA

INFORMAZIONI: Cliente, Acquisto

OPERAZIONI: Sottrazione, proiezione, selezione, join

LOGICA: Sottraiamo dalle tuple di cliente quelle ottenute effettuando l'equijoin tra cliente ed acquisto con una selezione sulla data.

C - (PROJ_{Ntess, Nome, Indirizzo}(SEL_{Data_Acquisto>01-01-1997}(C JOIN_{C.Ntess=A.Ntess_A} A)))

MYSQL

```

1 -- a) selezionare tutti i dati dei clienti che dopo il 1/1/1997 non hanno acquistato
2 -- nessun cd prodotto dalla casa discografica 'DiscoJolli'
3
4 • SELECT
5   *
6   FROM
7     cliente c
8   WHERE
9     c.n tess NOT IN (SELECT
10       a.n tess_a
11       FROM
12         acquisto a,
13         cd
14       WHERE
15         a.cod_cd_a = cd.cod_cd
16         AND cd.casa_disco = 'DiscoJolli'
17         AND a.data_acquisto > '1997-01-01');

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

n tess	nome	indirizzo
T2	Ludovica	Via Roma
T3	Luca	Via Napoli
NULL	HULL	NULL

Fig. 18.5 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

2. Selezionare il numero di tessera dei clienti che hanno acquistato tutti i cd dell'autore 'Francesco Guccini'.

SQL

INFORMAZIONI: Cliente, Cd

OPERAZIONI: 2 interrogazioni annidate con passaggio di parametri, Operatore NOT EXISTS, join, selezione

LOGICA: dall'interrogazione C ottengo i clienti che hanno acquistato dei cd, con l'interrogazione B prendo i cd di Guccini e con l'interrogazione A ottengo i clienti che hanno acquistato solo i cd di Guccini.

```

SELECT C.Ntess
A  FROM Cliente C
WHERE NOT IN (SELECT *
B  FROM Cd
WHERE Cd.Autore='Francesco Guccini' AND NOT EXISTS (SELECT *
C  FROM Acquisto A
WHERE A.Ntess_A=C.Ntess AND
A.Cod_Cd_A=C.Cod_Cd)

```

ALGEBRA

INFORMAZIONI: Acquisto, Cd

OPERAZIONI: Proiezione, sottrazione, join, Selezione

LOGICA: Sottrago da tutti i numeri di tessera di acquisto quelli ottenuti da un' ulteriore sottrazione tra il join naturale tra A e C e l'equi join con selezione sull'autore tra gli stessi attributi.

$$(\text{PROJ}_{\text{Ntess}} \text{ A}) - (\text{PROJ}_{\text{Ntess}} ((\text{A} \text{ JOIN } \text{C}) - (\text{SEL}_{\text{Autore}=\text{Francesco Guccini}} (\text{A} \text{ JOIN }_{\text{A.Cod_Cd_A}=\text{C.Cod_Cd}} \text{C}))))$$

MYSQL

```

1 -- b) selezionare il numero tessera dei clienti che hanno acquistato tutti cd
2 -- dell'autore 'Francesco Guccini'
3
4 • SELECT
5   c.n tess
6   FROM
7     cliente c
8   WHERE
9     NOT EXISTS( SELECT
10       *
11       FROM
12         cd
13       WHERE
14         cd.autore = 'Francesco Guccini'
15         AND NOT EXISTS( SELECT
16           *
17           FROM
18             acquisto a
19             WHERE
20               a.n tess_a = c.n tess
21               AND a.cod_cd_a = cd.cod_cd));

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

n tess
T1
NULL

Fig. 18.6 – Codice MySQL per l'interrogazione 2 e le istanze risultanti

3. Selezionare, per ogni cd, il numero totale delle copie vendute.

SQL

INFORMAZIONI: Acquisto

OPERAZIONI: Group by, operatore sum

LOGICA: Suddivido la relazione Acquisto in sottotabelle quanti sono i valori diversi di Cod_Cd_A. Con l'operatore sum ottengo la somma delle quantità di copie vendute.

```
SELECT A.Cod_Cd_A, SUM(A.QTY)
FROM Acquisto A
GROUP BY A.Cod_Cd_A
```

MYSQL

```
1 -- c) selezionare, per ogni cd, il numero totale delle copie vendute
2
3 • SELECT
4     a.cod_cd_a, SUM(a.qty)
5 FROM
6     acquisto a
7 GROUP BY a.cod_cd_a;
```

result Grid | Filter Rows: | Export: | Wrap Cell Content: |

cod_cd_a	SUM(a.qty)
1	3
2	1
3	2
4	1

Fig. 18.7 – Codice MySQL per l'interrogazione 3 e le istanze risultanti

4. Selezionare, per ogni casa discografica, il numero tessera del cliente che ha acquistato il maggior numero di copie di quel cd di quella casa.

SQL

INFORMAZIONI: Acquisto, Cd

OPERAZIONI: 1 interrogazione annidata con passaggio di parametri, Group by, operatore sum, Having, join, >=ALL

LOGICA: Con l'interrogazione B suddivido la relazione Acquisto in tante sottotabelle quanti sono i valori diversi di Ntess_A e per ognuna ne ottengo la somma adelle quantità. Tramite l'iterrogazione A seleziono quelle sottotabelle, ottenute suddividendo quella di partenza in tante quanti sono i valori diversi delle coppie(Ntess_A, Casa_Disco), la cui somma dei valori dell'attributo quantità è maggiore o uguale di tutti i valori ottenuti come risultato dell'interrogazione B.

```
SELECT A1.Ntess_A, Cd1.Casa_Disco
A FROM Acquisto A1, Cd Cd1
WHERE A1.Cod_Cd_A=Cd1.Cod_Cd
GROUP BY A1.Ntess_A, Cd1.Casa_Disco
HAVING SUM(A1.qty) >=ALL (SELECT SUM(A2.Qty)
B FROM Acquisto A2, Cd, Cd2
WHERE A2.Cod_Cd_A=Cd2.Cod_Cd AND Cd1.Casa_Disco=Cd2.Casa_Disco
GROUP BY A2.Ntess_A)
```

MYSQL

```
1 -- d) selezionare, per ogni casa discografica, il numero tessera del cliente che ha
2 -- acquistato il maggior numero di copie di cd di quella casa
3
4 • SELECT
5     a1.ntess_a, cd1.casa_disco
6 FROM
7     acquisto a1,
8     cd cd1
9 WHERE
10    a1.cod_cd_a = cd1.cod_cd
11   GROUP BY a1.ntess_a , cd1.casa_disco
12   HAVING SUM(a1.qty) >= ALL (SELECT
13       SUM(a2.qty)
14   FROM
15       acquisto a2,
16       cd cd2
17   WHERE
18       a2.cod_cd_a = cd2.cod_cd
19       AND cd1.casa_disco = cd2.casa_disco
20   GROUP BY a2.ntess_a);
```

result Grid | Filter Rows: | Export: | Wrap Cell Content: |

ntess_a	casa_disco
T1	DiscoJolly
T1	DiscoLuna

Fig. 18.8 – Codice MySQL per l'interrogazione 4 e le istanze risultanti

ESERCITAZIONE 19 - Azienda

06/06/2017

Dato il seguente schema della base di dati AZIENDA:

DIPENDENTE (Cf Nome, Città)

PROGETTO (Cod_P, Nome_P, Anno, Durata)

LAVORA (Cod_P_L, Cf_L, Mesi, Ruolo)

FK: Cod_P_L REFERENCES Progetto (Cod_P)

FK: Cf_L REFERENCES Dipendente (Cf)

Il Dipendente (Cf, Nome, Città) Lavora a dei Progetti(Cod_P, Nome_P,Ano, Durata) (Fig 19.1).

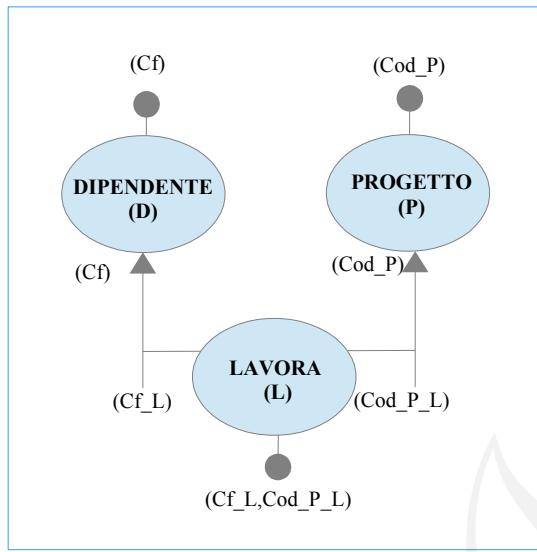


Fig. 19.1 – Modello dello schema di base di dati

Interrogazioni (Algebra Relazionale 1,2) :

1. Selezionare i dati dei dipendenti di Modena che non hanno lavorato in alcun progetto dell'anno 1995;
2. Selezionare i dati dei dipendenti che non hanno mai lavorato insieme ad un dipendente di Modena, cioè nello stesso progetto in cui lavora un dipendente di Modena;
3. Selezionare le coppie (cf1, cf2) tali che i dipendenti con codice fiscale cf1, cf2 hanno lavorato nello stesso progetto;
4. Selezionare i dipendenti che hanno lavorato in almeno tre ruoli distinti;
5. Selezionare per ogni dipendente il progetto il cui esso ha lavorato il maggior numero di mesi;

Fig. 19.2 – Codice MySQL per la creazione del database Azienda

```
1  --
2  -- Create schema azienda
3  --
4
5 • CREATE DATABASE IF NOT EXISTS azienda;
6 • USE azienda;
7
8  --
9  -- Definition of table dipendente
10 --
11
12 • └─CREATE TABLE dipendente (
13     cf char(16) NOT NULL,
14     nome varchar(75) NOT NULL,
15     citta varchar(30) NOT NULL,
16     PRIMARY KEY (cf)
17 );
18
19
20  --
21  -- Definition of table progetto
22  --
23
24 • └─CREATE TABLE progetto1 (
25     cod_p char(5) NOT NULL,
26     nome_progetto varchar(45) NOT NULL,
27     anno int(4),
28     durata int(2),
29     PRIMARY KEY (cod_p)
30 );
31
32  --
33  -- Definition of table lavora
34  --
35
36 • └─CREATE TABLE lavora (
37     cod_p_1 char(5) NOT NULL,
38     cf_1 char(16) NOT NULL,
39     mesi int(2) NOT NULL,
40     ruolo varchar(30) NOT NULL,
41     PRIMARY KEY (cod_p_1, cf_1),
42     FOREIGN KEY (cod_p_1) REFERENCES progetto1(cod_p),
43     FOREIGN KEY (cf_1) REFERENCES dipendente(cf)
44 );
```

Fig. 19.3 – Codice MySQL per la popolazione del database

```
1  --
2  -- populating the DB
3  --
4 • INSERT INTO dipendente VALUES('CFF', 'Carlo Fontana', 'Milano');
5 • INSERT INTO dipendente VALUES('OSA', 'Osvaldo Acqua', 'Modena');
6 • INSERT INTO dipendente VALUES('MRR', 'Mario Rossi', 'Modena');
7 • INSERT INTO dipendente VALUES('GMR', 'Guglielmo Marconi', 'Modena');
8 • INSERT INTO progetto1 VALUES('P1', 'Progetto Ludovico', 1994, 4);
9 • INSERT INTO progetto1 VALUES('P2', 'Progetto Stella', 1991, 6);
10 • INSERT INTO progetto1 VALUES('P3', 'Progetto Sole', 1995, 2);
11 • INSERT INTO lavora VALUES('P1', 'CFF', 2, 'Analista');
12 • INSERT INTO lavora VALUES('P3', 'CFF', 2, 'Responsabile');
13 • INSERT INTO lavora VALUES('P2', 'CFF', 3, 'Progettista');
14 • INSERT INTO lavora VALUES('P3', 'MRR', 2, 'Progettista');
15 • INSERT INTO lavora VALUES('P3', 'OSA', 2, 'Progettista');
```

Fig. 19.4 – Istanze della base di dati

The screenshot shows three separate result grids from MySQL Workbench. The first grid displays the results of the query `SELECT * FROM azienda.dipendente;`, showing four rows with columns cf, nome, and citta. The second grid displays the results of the query `SELECT * FROM azienda.lavora;`, showing five rows with columns cod_p_l, cf_l, mesi, and ruolo. The third grid displays the results of the query `SELECT * FROM azienda.progetto1;`, showing four rows with columns cod_p, nome_progetto, anno, and durata.

cf	nome	citta
CFF	Carlo Fontana	Milano
GMR	Gudliemo Marconi	Modena
MRR	Mario Rossi	Modena
OSA	Osvaldo Acqua	Modena
HULL	HULL	HULL

cod_p_l	cf_l	mesi	ruolo
P1	CFF	2	Analista
P2	CFF	3	Progettista
P3	CFF	2	Responsabile
P3	MRR	2	Progettista
P3	OSA	2	Progettista
HULL	HULL	HULL	HULL

cod_p	nome_progetto	anno	durata
P1	Progetto Ludovico	1994	4
P2	Progetto Stella	1991	6
P3	Progetto Sole	1995	2
HULL	HULL	HULL	HULL

1. Selezionare i dati dei dipendenti di Modena che non hanno lavorato in alcun progetto dell'anno 1995.

SQL

INFORMAZIONI: Dipendente

OPERAZIONI: 1 interrogazione annidata, Operatore NOT IN, join, selezione

LOGICA: Dall'interrogazione B ottengo tutti i dipendenti che lavorano ai progetti del 1995, da questi, attraverso l'interrogazione A, estraggo i dipendenti di Modena che non vi hanno lavorato.

```

SELECT *
A  FROM Dipendente D
      WHERE D.Città='Modena' AND D.Cf NOT IN (SELECT L.Cf_L
                                                B  FROM Lavora L, Progetto1 P
                                                WHERE L.Cod_P_L=P.Cod_P AND P.Anno='1995')
  
```

ALGEBRA

INFORMAZIONI: Dipendente, Lavora

OPERAZIONI: Selezione, proiezione, join

LOGICA: Con i join interni ottengo i codici dei dipendenti e i progetti a cui lavorano, con la selezione ottengo i dipendenti che lavorano ai progetti del 1995 e di questi risultati ottenuti proietto il cf, nome e città in modo da poter fare la differenza con i dipendenti di Modena e ottenere il risultato richiesto.

$(\text{SEL}_{\text{Città}=\text{Modena}} \text{D}) - (\text{PROJ}_{\text{cf}, \text{Nome}, \text{Città}} (\text{SEL}_{\text{Anno}=1995} (\text{D JOIN}_{\text{D.Cf}=\text{L.Cf}_L} \text{L}) \text{JOIN}_{\text{Cod_P_L}=\text{P.Cod_P}} \text{P}))$

MYSQL

The screenshot shows the MySQL code for the query and its resulting table. The code is as follows:

```

1 -- a) selezionare i dati dei dipendenti di Modena che non hanno lavorato
2 -- in alcun progetto dell'anno 1995
3
4 • SELECT
5   *
6   FROM
7     DIPENDENTE d
8   WHERE
9     d.città = 'Modena'
10    AND d.cf NOT IN (SELECT
11      l.cf_l
12      FROM
13        lavora l,
14        progetto1 p
15      WHERE
16        l.cod_p_l = p.cod_p AND p.anno = 1995);
  
```

The resulting table shows one row with columns cf, nome, and citta.

cf	nome	citta
GMR	Gudliemo Marconi	Modena
HULL	HULL	HULL

Fig. 19.5 – Codice MySQL per l'interrogazione 1 e le istanze risultanti

2. Selezionare i dati dei dipendenti che non hanno mai lavorato insieme ad un dipendente di Modena, cioè nello stesso progetto in cui lavora un dipendente di Modena.

SQL

INFORMAZIONI: Dipendente, Lavora

OPERAZIONI: 1 Interrogazione innestata, operatore NOT IN, join e selezione

LOGICA: Con l'interrogazione B ottengo i dipendenti che lavorano ad un progetto insieme ad un dipendente di 'Modena', con l'interrogazione A Ottengo attraverso l'attributo Cf i dipendenti che non hanno mai lavorato insieme a un dipendente di Modena.

```

SELECT *
A FROM Dipendente D
WHERE D.Cf NOT IN (SELECT L1.Cf_L
B      FROM Lavora L1, Lavora L2, Dipendente D
          WHERE L1.Cod_P_L=L2.Cod_P_L AND L2.Cf_L=D.Cf AND D.Città='Modena')
    
```

ALGEBRA

INFORMAZIONI: Dipendente, Lavora

OPERAZIONI: Join, sottrazione, ridenominazione, selezione

LOGICA: Con l'interrogazione R3 ottengo i codici dei dipendenti che hanno lavorato ad almeno un progetto insieme ad un dipendente di Modena. Sottraendoli ai codici di tutti i dipendenti ottengo quanto richiesto dall'interrogazione.

```

(D JOIN D.Cf=Cf)
(PROJ Cf D )
R3 (PROJ Cf-(REN Cf-Cf_L_R(SEL Città='Modena'((D JOIN D.Cf=L.Cf_L L) JOIN Cod_P_L=Cod_P_L_R
(REN Cod_P_L_R,Cf_L_R,Mesi_R,Ruolo_R<-Cod_P_L,Cf_L,Mesi,Ruolo L))))))
    ) )
    
```

MYSQL

The screenshot shows the MySQL Workbench interface. In the SQL editor pane, there is a code block with numbered lines. Lines 1-4 are comments. Line 5 starts with a SELECT statement. Lines 6-19 complete the query, which retrieves data from the Dipendente (d), Lavora (l1, l2), and Dipendente (d) tables, filtering for employees who have worked on projects with employees from Modena, but not on their own project.

```

1 -- b) selezionare i dti dei dipendenti che non hanno mai lavorato insieme
2 -- ad un dipendente di Modena, cioè nello stesso progetto in cui lavorava
3 -- anche un dipendente di Modena
4
5 • SELECT
6     *
7     FROM
8         dipendente d
9     WHERE
10        d.cf NOT IN (SELECT
11            l1.cf_l
12            FROM
13                lavora l1,
14                lavora l2,
15                dipendente d
16            WHERE
17                l1.cod_p_l = l2.cod_p_l
18                AND l2.cf_l = d.cf
19                AND d.città = 'Modena');
    
```

The results grid below shows one row of data:

cf	nome	città
GMR NULL	Guilermo Marconi NULL	Modena NULL

Fig. 19.6 – Codice MySQL per l'interrogazione 2 e le istanze risultanti

3. Selezionare le coppie (cf1, cf2) tali che i dipendenti con codice fiscale cf1, cf2 hanno lavorato nello stesso progetto.

SQL

INFORMAZIONI: Lavora

OPERAZIONI: join

LOGICA: Attraverso il Join tra due tabelle uguali otteniamo i dipendenti che lavorano allo stesso progetto.

```

SELECT L1.Cf_L AS CF1, L2.Cf_L AS CF2
A FROM Lavora L1, Lavora L2
WHERE L1.Cod_P_L=L2.Cod_P_L AND L1.Cf_L>L2.Cf_L
    
```

MYSQL

The screenshot shows the MySQL Workbench interface. In the top pane, there is a SQL editor window containing the following code:

```

1 -- c) selezionare le coppie (cf1, cf2) tali che i dipendenti con codice fiscale
2 -- cf1 e cf2 hanno lavorato nello stesso progetto
3
4 • SELECT
5     11.cf_1 AS CF1, 12.cf_1 AS CF2
6     FROM
7         lavora 11,
8         lavora 12
9     WHERE
10        11.cod_p_1 = 12.cod_p_1
11        AND 11.cf_1 <> 12.cf_1;

```

In the bottom pane, there is a "Result Grid" showing the following data:

CF1	CF2
CFF	MRR
CFF	OSA
MRR	CFF
MRR	OSA
OSA	CFF
OSA	MRR

Fig. 19.7 – Codice MySQL per l'interrogazione 3 e le istanze risultanti

4. Selezionare i dipendenti che hanno lavorato in almeno tre ruoli distinti.

SQL

INFORMAZIONI: Dipendente, Lavora

OPERAZIONI: 1 interrogazione annidata con passaggio di parametro, join, Operatore <=, COUNT, DISTINCT

LOGICA: per ogni dipendente passato dall'interrogazione A ottengo il conteggio dei ruoli distinti da esso ricoperto.

```

SELECT *
A  FROM Dipendente D
 WHERE 3<= (SELECT COUNT(DISTINCT L.Ruolo)
              B   FROM Lavora L
                  WHERE L.Cf_L=D.Cf)

```

MYSQL

The screenshot shows the MySQL Workbench interface. In the top pane, there is a SQL editor window containing the following code:

```

1 -- d) selezionare i dipendenti che hanno lavorato in almeno 3 ruoli distinti
2
3 • SELECT
4     *
5     FROM
6         dipendente d
7     WHERE
8         3 <= (SELECT
9             COUNT(DISTINCT 1.ruolo)
10            FROM
11                lavora 1
12                WHERE
13                    1.cf_1 = d.cf);

```

In the bottom pane, there is a "Result Grid" showing the following data:

cf	nome	citta
CFF	Carlo Fontana	Milano
HULL	HULL	HULL

Fig. 19.8 – Codice MySQL per l'interrogazione 4 e le istanze risultanti

5. Selezionare per ogni dipendente il progetto cui esso ha lavorato il maggior numero di mesi.

SQL

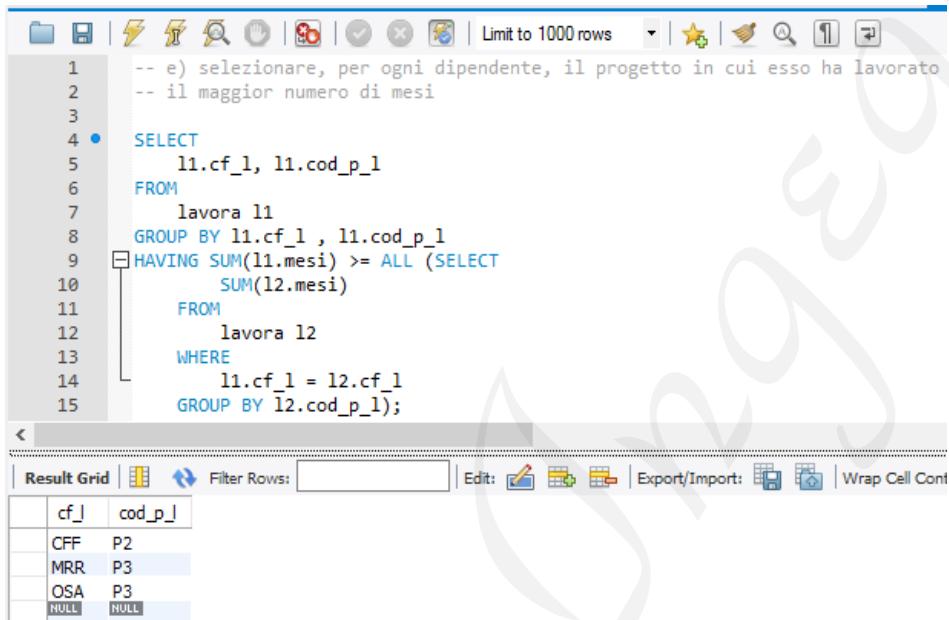
INFORMAZIONI: Lavora

OPERAZIONI: 1 interrogazione annidata con passaggio di parametri, selezione, join, operatore havig, >=ALL, Group by, SUM

LOGICA: Con l'interrogazione A suddivido la relazione Lavora in sottotabelle quanti sono i valori diversi della coppia (Cf_L, Cod_P_L). Con l'interrogazione B suddivido la relazione L1 in tante sottotabelle quanti sono i valori diversi (Cod_P_L) ed ottengo tramite l'operatore aggregato SUM dei mesi che hanno lavorato i dipendenti. Attraverso l'operatore di confronto >= ALL seleziono nell'interrogazione A solo i valori di quelle terne le cui sottotabelle hanno un numero di tuple maggiore di tutti i valori ottenuti come risultato dell'interrogazione B.

```
SELECT L1.Cf_L, L1.Cod_P_L
A  FROM Lavora L1
    GROUP BY L1.Cf_L, L1.Cod_P_L
    HAVING SUM (L1.Mesi) >= ALL ( SELECT SUM(L2.mesi)
                                    B   FROM Lavora L2
                                        WHERE L1.Cf_L=L2.Cf_L
                                            GROUP BY L2.Cod_P_L)
```

MYSQL



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following code:

```
-- e) selezionare, per ogni dipendente, il progetto in cui esso ha lavorato
-- il maggior numero di mesi

SELECT
    l1.cf_1, l1.cod_p_1
FROM
    lavora l1
GROUP BY l1.cf_1 , l1.cod_p_1
HAVING SUM(l1.mesi) >= ALL (SELECT
        SUM(l2.mesi)
    FROM
        lavora l2
    WHERE
        l1.cf_1 = l2.cf_1
    GROUP BY l2.cod_p_1);
```

The results grid shows the following data:

cf_1	cod_p_1
CFF	P2
MRR	P3
OSA	P3
NULL	NULL

Fig. 19.9 – Codice MySQL per l'interrogazione 5 e le istanze risultanti

ESERCITAZIONE 20 – ASSISTENZA

Dato il seguente schema per la base di dati ASSISTENZA:

TECNICO (CF, Indirizzo, Qualifica, CostoOrario)
PC (CP, Nome, Tipo, NomeProp)
RIPARAZIONE (Data_Rip, Ref_CF, Ref_CP, Ore)
FK: Ref_CF REFERENCES Tecnico (CF)
FK: Ref_CP REFERENCES PC (CP)

Un Tecnico (CF) ripara un PC (CP) nella Data specificata, impiegando un determinato numero di Ore (Fig. 20.1).

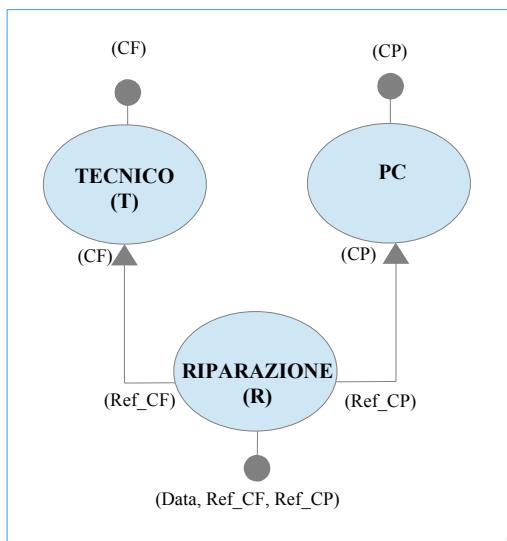


Fig. 20.1 – Modello dello schema di base di dati

Con riferimento alla fig. 20.1, effettuare le seguenti interrogazioni (Algebra Relazionale 1, 2) :

1. Selezionare i personal di tipo 'Mac' che non sono stati riparati tra il 1/7/97 e il 1/11/97;
2. Selezionare i tecnici che hanno riparato tutti i personal di tipo 'Mac';
3. Selezionare le coppie (CP1, CP2) tali che i personal con codice CP1 e CP2 sono stati riparati nella stessa data dallo stesso tecnico;
4. Selezionare i dati dei personal che hanno richiesto almeno 10 ore di riparazione;
5. Selezionare, per ogni data, il tecnico che ha riparato il maggior numero di personal;
6. Selezionare il personal che ha totalizzato il maggior costo di riparazione, considerando le ore di riparazione e il relativo costo orario.

Fig. 20.2 – Codice MySQL per la creazione del database Assistenza

```

1 • CREATE DATABASE IF NOT EXISTS assistenza;
2 • USE assistenza;
3 -- Definition of table tecnico
4 • └─ CREATE TABLE tecnico (
5   cf char(16) NOT NULL,
6   indirizzo varchar(45) NOT NULL,
7   qualifica varchar(30) NOT NULL,
8   costo_orario int(2) NOT NULL,
9   PRIMARY KEY (cf)
10 );
11 --
12 -- Definition of table pc
13 --
14 • └─ CREATE TABLE pc (
15   cp char(5) NOT NULL,
16   nome varchar(45) NOT NULL,
17   tipo varchar(20),
18   nome_proprietario varchar(75) NOT NULL,
19   PRIMARY KEY (cp)
20 );
21 --
22 -- Definition of table riparazione
23 --
24 • └─ CREATE TABLE riparazione (
25   data_rip date NOT NULL,
26   ref_cf char(16) NOT NULL,
27   ref_cp char(5) NOT NULL,
28   ore int(2) NOT NULL,
29   PRIMARY KEY (data_rip, ref_cf, ref_cp),
30   FOREIGN KEY (ref_cf) REFERENCES tecnico(cf),
31   FOREIGN KEY (ref_cp) REFERENCES pc(cp)
32 );

```

Fig. 20.3 – Codice MySQL per la popolazione del database

```

1
2 • insert into tecnico values ('A', 'Via delle Magnolie', 'Assemblatore', 40),
3   ('B', 'Via Roma', 'Assemblatore', 40),
4   ('C', 'Via Palermo', 'Analista', 50);
5
6 • insert into pc values (1, 'MacBook Pro', 'Mac', 'Rossi'),
7   (2, 'Notebook hp', 'HP', 'Verdi'),
8   (3, 'MacBook Air', 'Mac', 'Lucca'),
9   (4, 'MacBook Air 13', 'Mac', 'Belli');
10 -- uno vhr li ha riparati tutti e uno non nella data bls blsb ls
11 • insert into riparazione values ('2010-10-10', 'A', 1, 11),
12   ('1997-10-08', 'A', 3, 2),
13   ('2000-11-03', 'A', 4, 10),
14   ('2000-11-03', 'B', 2, 3),
15   ('2000-11-03', 'B', 1, 4);
16

```

Fig. 20.4 – Istanze del database

The screenshot shows three separate result grids from MySQL Workbench:

- Result Grid 1 (tecnico):** Shows data for the 'tecnico' table with columns: cf, indirizzo, qualifica, costo_orario. The data is:

cf	indirizzo	qualifica	costo_orario
A	Via delle Magnolie	Assemblatore	40
B	Via Roma	Assemblatore	40
C	Via Palermo	Analista	50
- Result Grid 2 (pc):** Shows data for the 'pc' table with columns: cp, nome, tipo, nome_proprietario. The data is:

cp	nome	tipo	nome_proprietario
1	MacBook Pro	Mac	Rossi
2	Notebook hp	HP	Verdi
3	MacBook Air	Mac	Lucca
4	MacBook Air 13	Mac	Belli
- Result Grid 3 (riparazione):** Shows data for the 'riparazione' table with columns: data_rip, ref_cf, ref_cp, ore. The data is:

data_rip	ref_cf	ref_cp	ore
1997-10-08	A	3	2
2000-11-03	A	4	10
2000-11-03	B	1	4
2000-11-03	B	2	3
2010-10-10	A	1	11

1. Selezionare i personal di tipo 'Mac' che non sono stati riparati tra il 1/7/97 e il 1/11/97

SQL

INFORMAZIONI: PC, Riparazione

OPERAZIONI: Interrogazione innestata, operatore insiemistico Not In, selezione

LOGICA: Con l'interrogazione B seleziono tutti i Personal che sono stati riparati almeno una volta nella data specificata dal testo. Con l'interrogazione A proietto tutti i dati dei personal di tipo Mac che non si trovano nella relazione risultante dall'interrogazione B

```
A   SELECT PC.*  
     FROM PC  
   WHERE PC.Tipo = 'Mac' AND PC.CP NOT IN ( SELECT Ref_Cp  
                                              B   FROM Riparazione R  
                                                WHERE R.Data_Rip > '1997-01-07' AND R.Data_Rip < '1997-01-11' )
```

ALGEBRA

INFORMAZIONI: PC, Riparazione

OPERAZIONI: Join, selezione, proiezione, differenza

LOGICA: Con join e selezione ottengo tutti i PC di tipo Mac che sono stati riparati almeno una volta nella data specificata dal testo, ne proietto codice, tipo e nome del proprietario e sottraggo la relazione risultante alla relazione PC

(SEL Tipo = 'Mac' PC)

—

(PROJ CP, Nome, Tipo, NomeProp (PC JOIN PC.CP = Ref_Cp (SEL Data_Rip > '1997-01-07' AND Data_Rip < '1997-01-11' R)))

MYSQL

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
1 •  SELECT *  
2   FROM pc  
3  WHERE pc.tipo = 'Mac' AND pc.cp NOT IN (  
4    SELECT r.ref_cp  
5    FROM riparazione r  
6   WHERE r.data_rip > '1997-01-07' AND r.data_rip < '1997-01-11')|
```

The result grid displays the following data:

cp	nome	tipo	nome_proprieta...
1	MacBook Pro	Mac	Rossi
3	MacBook Air	Mac	Lucca
4	MacBook Air 13	Mac	Belli

Fig. 20.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare i tecnici che hanno riparato tutti i personal di tipo 'Mac'

SQL

INFORMAZIONI: Tecnico, PC, Riparazione

OPERAZIONI: 2 interrogazioni innestate con passaggio di parametri, doppia negazione con Not Exists

LOGICA: Effettuo una divisione utilizzando una doppia negazione. L'interrogazione C restituisce le tuple di riparazione relative ad un personal, passato come parametro dall'interrogazione B, e ad un tecnico, passato come parametro dall'interrogazione A. Tramite la doppia negazione selezioniamo solo quei tecnici per i quali non esiste un personal di tipo 'Mac' che essi non abbiano riparato.

```
A   SELECT T.*  
     FROM Tecnico T  
   WHERE NOT EXISTS ( SELECT *  
                           B   FROM PC  
                         WHERE PC.Tipo = 'Mac' AND NOT EXISTS ( SELECT *  
                                                       FROM Riparazione R  
                                                     WHERE R.Ref_Cp = PC.CP AND R.Ref_Cf = T.CF )  
                           )
```

ALGEBRA

INFORMAZIONI: Tecnico, PC, Riparazione

OPERAZIONI: Divisione, join, proiezione, ridenominazione, selezione

LOGICA: Effettuo una divisione tra Riparazione e PC, avendo selezionato solo i personal di tipo Mac. Ottengo così i codici dei tecnici che hanno riparato tutti i personal di tipo Mac e recupero tutte le informazioni sui tecnici con un join.

```

PROJ CF, Indirizzo, Qualifica CostoOrario (
    T JOIN T.CF = Ref_CF (
        (PROJ Ref_CF, Ref_CP R)
        .
        .
        (PROJ Ref_CP (REN Ref_CP ← CP ( SEL Tipo = 'Mac' PC ) ) )
    ) )

```

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the following SQL query:

```

1 • SELECT *
  FROM tecnico t
  WHERE NOT EXISTS(
    SELECT *
      FROM pc
     WHERE pc.tipo = 'Mac' AND NOT EXISTS(
       SELECT *
         FROM riparazione r
        WHERE r.ref_cp = pc.cp AND r.ref_cf = t.cf))

```

Below the code editor is a "Result Grid" pane showing the following data:

cf	indirizzo	qualifica	costo_orario
A	Via delle Magnolie	Assemblatore	40

Fig. 20.6 – Codice MySQL per l'interrogazione 2 e istanza risultante

3. Selezionare le coppie (CP1, CP2) tali che i personal con codice CP1 e CP2 sono stati riparati nella stessa data dallo stesso tecnico

SQL

INFORMAZIONI: Riparazione

OPERAZIONI: join, selezione, proiezione, ridenominazione

LOGICA: Utilizzo due volte la tabella riparazione imponendo che siano uguali tecnico e data, mentre siano diversi i personal riparati. Infine proietto le coppie di codici risultanti.

```

SELECT R1.Ref_CP AS CP1, R1.Ref_CP AS CP2
FROM Riparazione R1, Riparazione R2
WHERE R1.Ref_CF = R2.Ref_CF AND R1.Data_Rip = R2.Data_Rip AND R1.Ref_CP > R2.Ref_CP

```

ALGEBRA

INFORMAZIONI: Riparazione

OPERAZIONI: Join, ridenominazione, proiezione, selezione

LOGICA: Ridenomino la tabella riparazione per poter fare un join con la stessa tabella sugli attributi tecnico e data. Infine proietto, dalla relazione ottenuta col join, le coppie diverse di personal ottenute.

```

PROJ CP1, CP2 ( REN CP1 ← Ref_Cp, CP2 ← Ref_Cp2 ( SEL Ref_Cp > Ref_Cp2
        (R JOIN Ref_Cp = Ref_Cp2 AND Data_Rip = Data_Rip2 ( REN Data_Rip2 ← Data, Ref_CF2 ← Ref_CF, Ref_Cp2 ← Ref_Cp, Ore2 ← Ore R ) )
    ) )

```

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the following SQL query:

```

1 • SELECT r1.ref_cp as CP1, r2.ref_cp as CP2
  FROM riparazione r1, riparazione r2
  WHERE r1.ref_cf = r2.ref_cf AND r1.data_rip = r2.data_rip AND r1.ref_cp > r2.ref_cp

```

Below the code editor is a "Result Grid" pane showing the following data:

CP1	CP2
2	1

Fig. 20.7 – Codice MySQL per l'interrogazione 3 e istanza risultante

4. Selezionare i dati dei personal che hanno richiesto almeno 10 ore di riparazione

INFORMAZIONI: PC, Riparazione

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, operatore matematico \leq , operatore aggregato Sum

LOGICA: Con l'interrogazione B ottengo, per ogni personal passato come parametro, il totale delle ore di riparazione. Con l'interrogazione A seleziono solo quei PC per cui l'interrogazione B da come risultato un numero almeno pari a 10.

```

A   SELECT PC.*
    FROM PC
    WHERE 10 <= ( SELECT SUM(Ore)
                    B      FROM Riparazione R
                            WHERE R.Ref_Cp = PC.CP)

```

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, the SQL editor contains the code from above. Below it, the 'Result Grid' tab is selected, showing the following data:

cp	nome	tipo	nome_proprieta...
1	MacBook Pro	Mac	Rossi
4	MacBook Air 13	Mac	Belli

Fig. 20.8 – Codice MySQL per l'interrogazione 4 e istanze risultanti

5. Selezionare, per ogni data, il tecnico che ha riparato il maggior numero di personal

INFORMAZIONI: Riparazione

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, due group by, operatore matematico \geq , operatore aggregato Count.

LOGICA: Con l'interrogazione B ottengo il numero di personal riparati da ogni tecnico nella data passata come parametro. Con l'interrogazione A seleziono per ogni data quel tecnico il cui numero di personal riparati risulta essere maggiore o uguale del risultato dell'interrogazione B, ottenendo così il massimo.

```

SELECT R.Ref_CF, R.Data_Rip
  FROM Riparazione R
A GROUP BY R.Ref_CF, R.Data_Rip
     HAVING COUNT(*) >= ALL ( SELECT COUNT(*)
                                FROM Riparazione R1
B      WHERE R1.Data_Rip = R.Data_Rip
          GROUP BY R1.Ref_CF )

```

MYSQL

The screenshot shows the MySQL Workbench interface. At the top, the SQL editor contains the code from above. Below it, the 'Result Grid' tab is selected, showing the following data:

ref_cf	data_rip
A	1997-10-08
A	2010-10-10
B	2000-11-03

Fig. 20.9 – Codice MySQL per l'interrogazione 5 e istanze risultanti

6. Selezionare il personal che ha totalizzato il maggior costo di riparazione, considerando le ore di riparazione e il relativo costo orario

INFORMAZIONI: Tecnico, Riparazione PC

OPERAZIONI: Un'interrogazione innestata, group by, join, operatore matematico \geq , operatore aggregato Sum

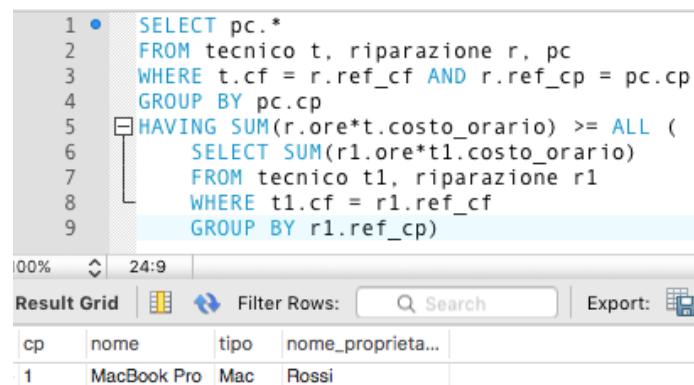
LOGICA: Con l'interrogazione B ottengo il totale del costo di riparazione per ciascun personal, calcolato sommando per ogni tecnico il costo di tutte le ore impiegate. Con l'interrogazione A seleziono quei personal per cui il costo totale è maggiore o uguale al risultato dell'interrogazione B.

```

SELECT PC.*
FROM Tecnico T, Riparazione R, PC
A WHERE T.CF = R.Ref_CF AND R.Ref_Cp = PC.CP
GROUP BY PC.CP
HAVING SUM (R.Ore * T.CostoOrario) >= ALL ( SELECT SUM ( R1.Ore * T1.CostoOrario)
                                              FROM Tecnico T1, Riparazione R1
B      WHERE T1.CF = R1.Ref_CF
      GROUP BY R1.Ref_Cp)

```

MYSQL



The screenshot shows the MySQL Workbench interface. At the top, there is a code editor window containing the SQL query. Below it is a result grid window showing the output of the query. The result grid has columns for cp, nome, tipo, and nome_proprieta... with one row of data: 1, MacBook Pro, Mac, Rossi.

```

1 •  SELECT pc.*
2   FROM tecnico t, riparazione r, pc
3   WHERE t.cf = r.ref_cf AND r.ref_cp = pc.cp
4   GROUP BY pc.cp
5   HAVING SUM(r.ore*t.costo_orario) >= ALL (
6       SELECT SUM(r1.ore*t1.costo_orario)
7       FROM tecnico t1, riparazione r1
8       WHERE t1.cf = r1.ref_cf
9       GROUP BY r1.ref_cp)

```

Result Grid			
cp	nome	tipo	nome_proprieta...
1	MacBook Pro	Mac	Rossi

Fig. 20.10 – Codice MySQL per l'interrogazione 6 e istanza risultante

ESERCITAZIONE 21 – MEETING

Dato il seguente schema per la base di dati MEETING:

NAZIONE (CN, Presidente, Continente)

CONFERENZA (CC, Descrizione, Sede)

FK: Sede REFERENCES Nazione (CN)

PARTECIPA (Ref_Conf, Ref_Naz, NumeroP)

FK: Ref_Conf REFERENCES Conferenza (CC)

FK: Ref_Naz REFERENCES Nazione (CN)

Una Nazione (CN) partecipa ad una Conferenza (CC) con un certo numero di partecipanti (NumeroP). La conferenza si svolge (Sede) in una determinata Nazione (fig. 21.1).

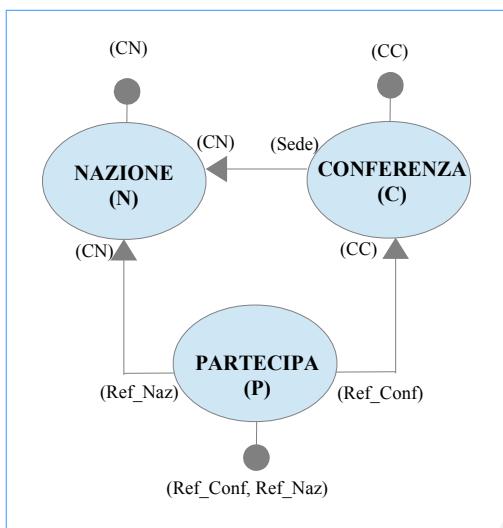


Fig. 21.1 – Modello dello schema di base di dati

Con riferimento alla fig. 21.1, effettuare le seguenti interrogazioni (Algebra Relazionale 1, 2, 3):

1. Selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi in una nazione del continente Europa;
2. Selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi nella nazione stessa;
3. Selezionare i dati relativi alle nazioni che non hanno mai partecipato ad una conferenza assieme ad una nazione del continente Europa;
4. Selezionare i dati relativi alla nazione in cui si è tenuta la conferenza con il maggior numero di rappresentanti complessivo;
5. Selezionare, per ogni nazione, la conferenza in cui vi ha partecipato con il maggior numero di rappresentanti;

Fig. 21.2 – Codice MySQL per la creazione del database Meeting

```
1 • CREATE DATABASE IF NOT EXISTS meeting;
2 • USE meeting;
3 -- Definition of table nazione
4 • ┌─ CREATE TABLE nazione (
5   cn char(3) NOT NULL,
6   presidente varchar(75) NOT NULL,
7   continente varchar(16) NOT NULL,
8   PRIMARY KEY (cn)
9 );
10 -- Definition of table conferenza
11 • ┌─ CREATE TABLE conferenza (
12   cc char(5) NOT NULL,
13   descrizione varchar(45) NOT NULL,
14   sede char(3) NOT NULL,
15   PRIMARY KEY (cc),
16   FOREIGN KEY (sede) REFERENCES nazione(cn)
17 );
18 -- Definition of table partecipa
19 • ┌─ CREATE TABLE partecipa (
20   ref_conf char(5) NOT NULL,
21   ref_naz char(3) NOT NULL,
22   numerop int(3) NOT NULL,
23   PRIMARY KEY (ref_conf, ref_naz),
24   FOREIGN KEY (ref_conf) REFERENCES conferenza(cc),
25   FOREIGN KEY (ref_naz) REFERENCES nazione(cn)
26 );
```

Fig. 21.3 – Codice MySQL per la popolazione del database

```

1 • insert into nazione values ('I', 'Mattarella', 'Europa'),
2 ('F', 'Macron', 'Europa'),
3 ('US', 'Trump', 'America'),
4 ('G', 'Merkel', 'Europa');
5 • insert into conferenza values ('A', 'Ambiente', 'I'),
6 ('B', 'Economia', 'US'),
7 ('C', 'Clima', 'F');
8 • insert into partecipa values ('A', 'I', 10),
9 ('A', 'F', 6),
10 ('B', 'I', 3),
11 ('B', 'G', 10),
12 ('B', 'US', 4),
13 ('C', 'US', 10);

```

Fig. 21.4 – Istanze del database

The screenshot shows three result grids from MySQL Workbench:

- Result Grid for meeting.nazione:**

cn	presidente	continent
F	Macron	Europa
G	Merkel	Europa
I	Mattarella	Europa
US	Trump	America
- Result Grid for meeting.conferenza:**

cc	descrizione	sede
A	Ambiente	I
B	Economia	US
C	Clima	F
- Result Grid for meeting.partecipa:**

ref_conf	ref_naz	numerop
A	F	6
A	I	10
B	G	10
B	I	3
B	US	4
C	US	10

1. Selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi in una nazione del continente Europa

SQL

INFORMAZIONI: Nazione, Partecipa, Conferenza

OPERAZIONI: 3 join, una selezione e proiezione

LOGICA: Utilizzo la relazione Nazione due volte: la prima per indicare le nazioni che partecipano alla conferenza e la seconda per indicare il luogo dove si svolge, e seleziono le nazioni ospitanti che si trovano in Europa

SELECT N1.*

FROM Nazione N1, Partecipa P, Conferenza C, Nazione N2

WHERE N1.CN = P.Ref_Naz AND P.Ref_Conf = C.CC AND C.Sede = N2.CN AND N2.Continent = 'Europa'

ALGEBRA

INFORMAZIONI: Nazione, Partecipa, Conferenza

OPERAZIONI: 3 join, una selezione, una ridenominazione e una proiezione

LOGICA: In questo caso svolgo le stesse operazioni di SQL, ma devo ridenominare tutti gli attributi della tabella Nazione per poterla utilizzare due volte

PROJ CN, Presidente, Continente (

```

    ( ( ( ( REN CN1 ← CN, Presidente1 ← Presidente, Continente1 ← Continente ( SEL Continente = 'Europa' N ) ) JOIN CN1 = C.Sede C )
        JOIN CC = P.Ref_Conf P ) JOIN Ref_Naz = N.CN N ) )

```

MYSQL

```
1 • SELECT n1.*  
2   FROM nazione n1, partecipa p, conferenza c, nazione n2  
3   WHERE n1.cn = p.ref_naz AND p.ref_conf = c.cc AND c.sede = n2.cn  
4       AND n2.continente = 'Europa'  
5  
100%  9:5  
Result Grid | Filter Rows: | Search | Export: |
```

cn	presidente	continente
US	Trump	America
F	Macron	Europa
I	Mattarella	Europa

Fig. 21.5 – Codice MySQL per l'interrogazione 1 e istanze risultanti

2. Selezionare i dati relativi alle nazioni che hanno partecipato ad una conferenza tenutasi nella nazione stessa

SQL

INFORMAZIONI: Nazione, Partecipa, Conferenza

OPERAZIONI: 3 join, una proiezione

LOGICA: Utilizzo due volte la stessa tabella nazione facendo tre join fra tre tabelle, utilizzando la stessa tabella per indicare sia la nazione partecipante che la nazione ospitante

```
SELECT N.*  
FROM Nazione N, Partecipa P, Conferenza C  
WHERE N.CN = P.Ref_Naz AND P.Ref_Conf = C.CC AND C.Sede = N.CN
```

ALGEBRA

INFORMAZIONI: Nazione, Partecipa, Conferenza

OPERAZIONI: 3 join, una proiezione

LOGICA: Come in SQL, utilizzo due volte la stessa tabella Nazione per due join con due tabelle diverse, una volta Partecipa e una volta Conferenza

PROJ CN, Presidente, Continente (((N JOIN N.CN = P.Ref_Naz P) JOIN Ref_Conf = C.CC C) JOIN Sede = N.CN N)

MYSQL

```
1 • SELECT n.*  
2   FROM nazione n, partecipa p, conferenza c  
3   WHERE n.cn = p.ref_naz AND p.ref_conf = c.cc AND c.sede = n.cn  
4  
100%  1:4  
Result Grid | Filter Rows: | Search | Export: |
```

cn	presidente	continente
I	Mattarella	Europa
US	Trump	America

Fig. 21.6 – Codice MySQL per l'interrogazione 2 e istanze risultanti

3. Selezionare i dati relativi alle nazioni che non hanno mai partecipato ad una conferenza assieme ad una nazione del continente Europa

SQL

INFORMAZIONI: Nazione, Partecipa

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, operatore insiemistico Not In, join e selezione

LOGICA: Con l'interrogazione B ottengo il codice delle nazioni, passate per parametro, che hanno partecipato ad una conferenza insieme ad una nazione del continente Europa. Con l'interrogazione A seleziono le nazioni il cui codice non è presente fra le tuple risultanti della relazione B

```
A  SELECT N.*  
  FROM Nazione N  
 WHERE N.CN NOT IN ( SELECT P2.Ref_Naz  
                      FROM Nazione N1, Partecipa P1, Partecipa P2  
                     WHERE N1.CN = P1.Ref_Naz AND P1.Ref_Conf = P2.Ref_Conf AND N1.Continent = 'Europa')  
B  
```

ALGEBRA

INFORMAZIONI: Nazione, Partecipa

OPERAZIONI: Differenza, join, proiezione, ridenominazione

LOGICA: Sottraggo dai codici della tabella Nazione la tabella che contiene i codici delle nazioni che hanno partecipato ad una conferenza insieme ad una nazione europea, ottenuta utilizzando due volte la tabella Partecipa. Infine ottengo tutti i dati sulle nazioni con un join.

```
PROJ CN, Presidente, Continente ( N JOIN N.CN = Ref_Naz1 (
    ( PROJ Ref_Naz1 (REN Ref_Naz1 ← CN N) ) -
    ( PROJ Ref_Naz1 (
        ( ( SEL Continente = 'Europa' N ) JOIN N.CN = P.Ref_Naz P ) JOIN Ref_Conf = Ref_Conf1 ( REN
            Ref_Conf1 ← Ref_Conf, Ref_Naz1 ← Ref_Naz, NumeroP1 ← NumeroP P )
    ) ) ))
```

MYSQL

The screenshot shows a MySQL query editor with the following code:

```
1 • SELECT n.*  
2   FROM nazione n  
3 WHERE n.cn NOT IN (  
4     SELECT p2.ref_naz  
5       FROM nazione n1, partecipa p1, partecipa p2  
6      WHERE n1.cn = p1.ref_naz AND p1.ref_conf = p2.ref_conf  
7          AND n1.continente = 'Europa'  
8 )
```

Below the code, the results of the query are displayed in a grid:

cn	presidente	continente
HULL	HULL	HULL

Fig. 21.7 – Codice MySQL per l'interrogazione 3 (nessuna istanza risultante)

4. Selezionare i dati relativi alla nazione in cui si è tenuta la conferenza con il maggior numero di rappresentanti complessivo

INFORMAZIONI: Nazione, Conferenza, Partecipa

OPERAZIONI: Un'interrogazione innestata con operatore aggregato Sum e operatore \geq ALL, join, group by

LOGICA: Con l'interrogazione B ottengo il numero totale di partecipanti per ciascuna conferenza. Con l'interrogazione A seleziono le nazioni che hanno ospitato la conferenza il cui numero totale di partecipanti è maggiore o uguale al risultato dell'interrogazione B (è il massimo).

```
A  SELECT N.*  
    FROM Nazione N, Conferenza C, Partecipa P  
 WHERE N.CN = C.Sede AND C.CC = P.Ref_Conf  
 GROUP BY P.Ref_Conf  
 HAVING SUM(P.NumeroP)  $\geq$  ALL ( B  
           SELECT SUM(P1.NumeroP)  
             FROM Partecipa P1  
           GROUP BY P1.Ref_Conf )
```

MYSQL

The screenshot shows a MySQL query editor with the following code:

```
1 • SELECT n.*  
2   FROM nazione n, conferenza c, partecipa p  
3 WHERE n.cn = c.sede AND c.cc = p.ref_conf  
4 GROUP BY p.ref_conf  
5 HAVING SUM(p.numeroP)  $\geq$  ALL ( 6  
          SELECT SUM(p1.numeroP)  
            FROM partecipa p1  
          GROUP BY p1.ref_conf )
```

Below the code, the results of the query are displayed in a grid:

cn	presidente	continente
US	Trump	America

Fig. 21.8 – Codice MySQL per l'interrogazione 4 e istanza risultante

5. Selezionare, per ogni nazione, la conferenza in cui vi ha partecipato con il maggior numero di rappresentanti

INFORMAZIONI: Partecipa

OPERAZIONI: Un'interrogazione innestata con passaggio di parametri, operatore aggregato Max

LOGICA: Con l'interrogazione B seleziono per ogni nazione il numero massimo di partecipanti. Con l'interrogazione A seleziono la conferenza per cui ciascuna nazione ha il numero dei partecipanti pari al risultato dell'interrogazione B.

```
A   SELECT P.Ref_Naz, P.Ref_Conf  
      FROM Partecipa P  
     WHERE P.NumeroP = ( SELECT MAX(NumeroP)  
                           FROM Partecipa P1  
                          WHERE P1.Ref_Naz = P.Ref_Naz)
```

MYSQL

```
1 •  SELECT p.ref_naz, p.ref_conf  
2    FROM partecipa p  
3  WHERE p.numeroP = (  
4      SELECT MAX(numeroP)  
5        FROM partecipa p1  
6       WHERE p1.ref_naz = p.ref_naz)|
```

00% 34:6

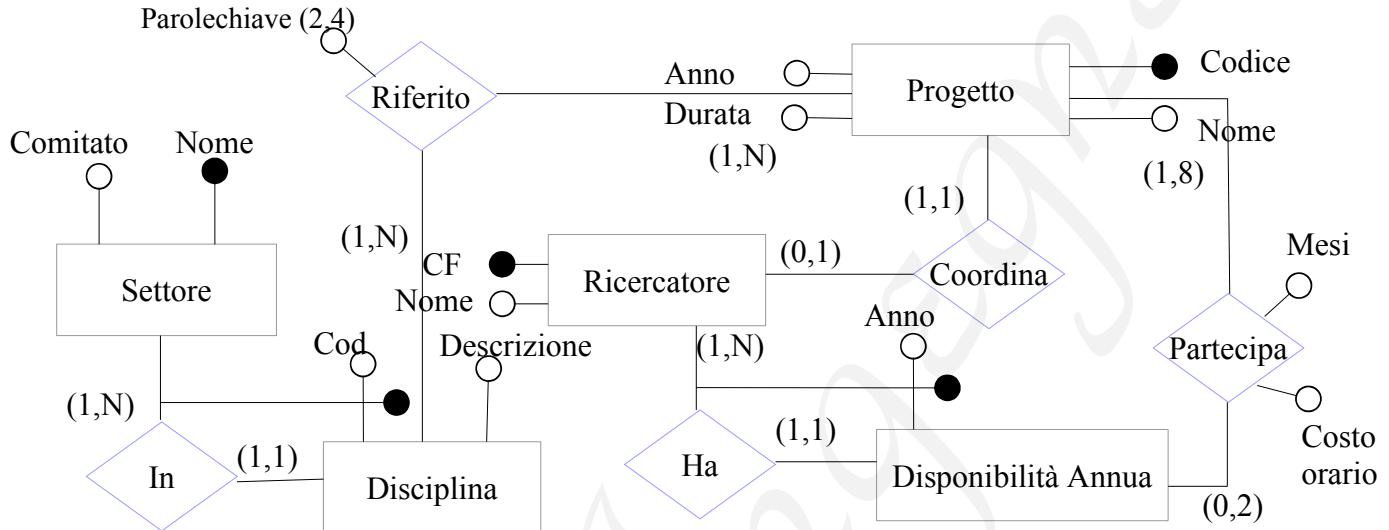
Result Grid | Filter Rows: Search | E

ref_naz	ref_conf
F	A
I	A
G	B
US	C

Fig. 21.9 – Codice MySQL per l'interrogazione 5 e istanze risultanti

Modello E-R 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 a più discipline: per ognuno di questi riferimenti occorre riportare da 2 a 4 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 e in 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.



Commento allo 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 l'entità DISPONIBILITÀ ANNUA.

Schema Relazionale

RICERCATORE(CF, NomeR)

PROGETTO(CodiceP, Anno, Durata, NomeP, CFCOORD)

FK: CFCOORD REFERENCES RICERCATORE (CF)

DISPONIBILITÀ ANNUA(Anno, CfRic)

FK: CfRic REFERENCES RICERCATORE (CF)

PARTECIPA (Anno, Ref_Ric, Ref_Progetto, Mesi, CostoOrario)

FK: Ref_Ric, Anno REFERENCES DISPONIBILITÀ ANNUA (CfRic)

FK: Ref_Progetto REFERENCES PROGETTO (CodiceP)

SETTORE(NomeS, Comitato)

DISCIPLINA (NomeSet, CodiceD, Descrizione)

FK: NomeSet REFERENCES SETTORE (NomeS)

PAROLECHIAVE(Ref_Set, CodicePC, ParolaChiave, CodicePR)

FK: Ref_Set, CodicePC REFERENCES DISCIPLINA (NomeSet, CodiceD)

FK: CodicePR REFERENCES PROGETTO (CodiceP)

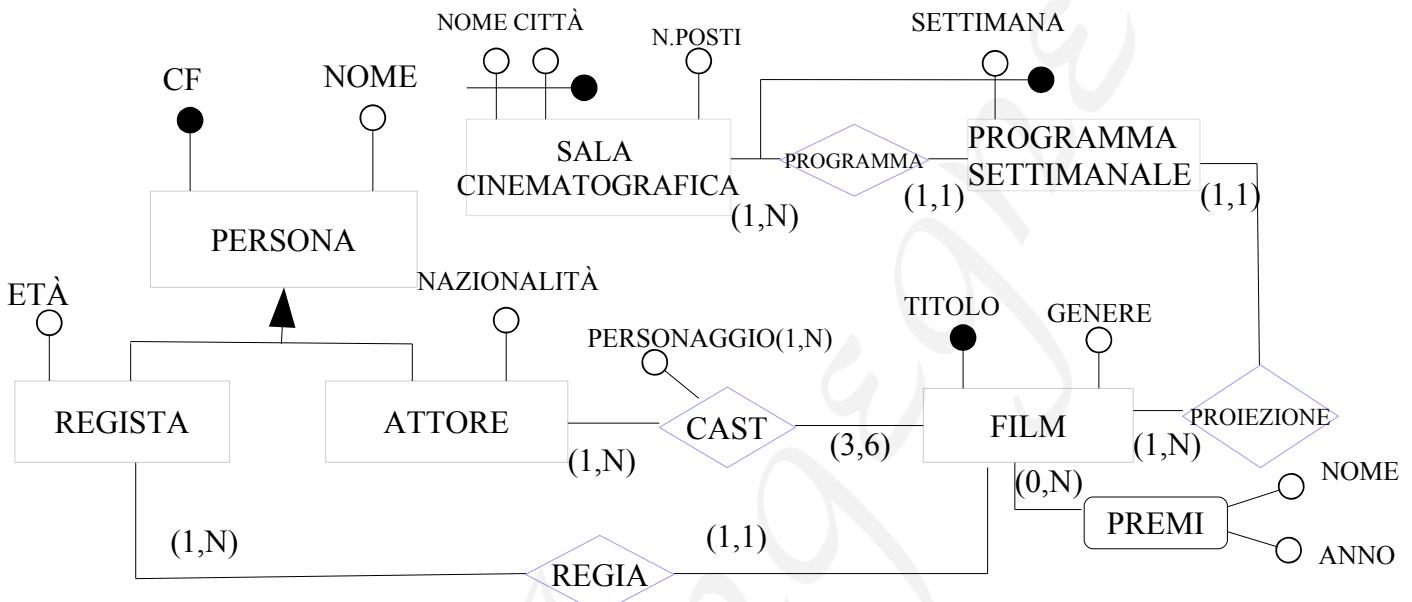
Commento sul progetto logico

L'associazione RIFERITO non è stata tradotta in quanto tutti i riferimenti tra PROGETTO e DISCIPLINA sono rappresentati dalla relazione PAROLECHIAVE : Per ognuno di questi riferimenti c'è almeno una parola chiave.

Modello E-R 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.



Schema Logico

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

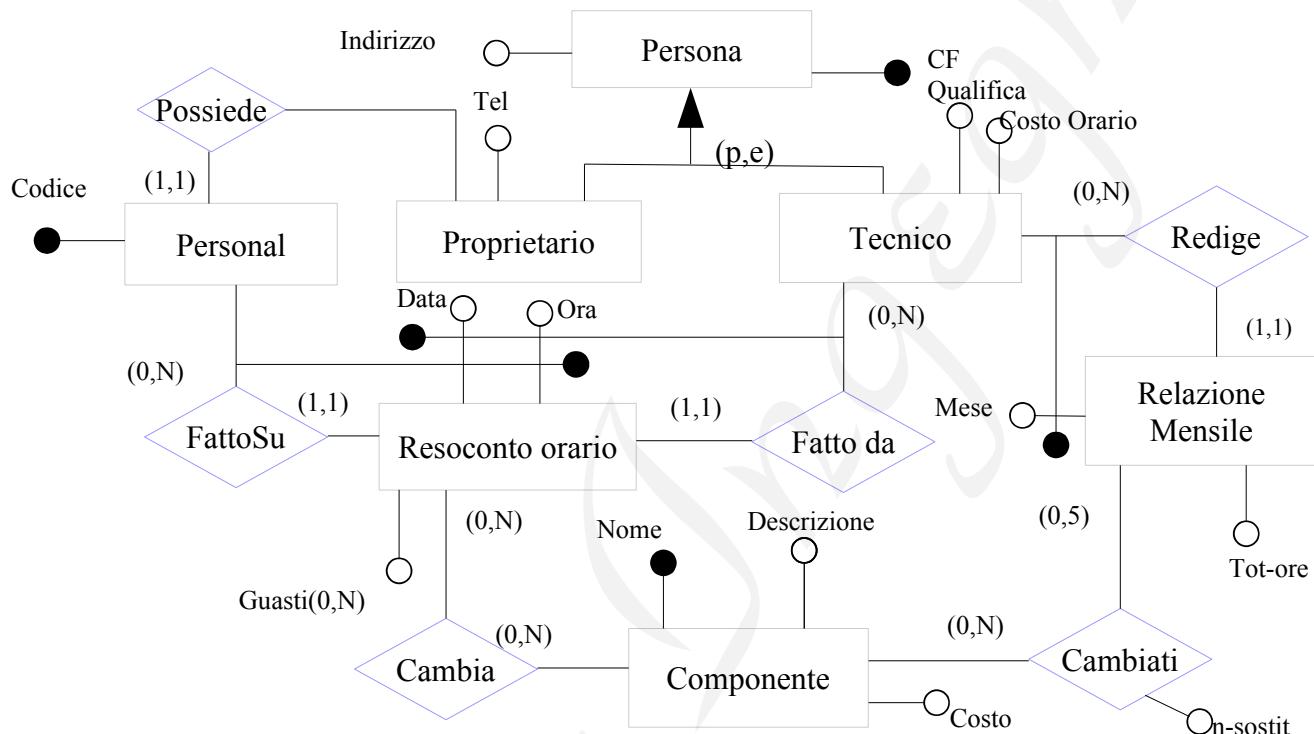
Commento allo schema E-R

In questo schema sarebbe un errore mettere card(SALA, 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.

Modello E-R 3

Una azienda di assistenza tecnica per Personal Computer vuole memorizzare informazioni sulle riparazioni secondo le seguenti specifiche. Le 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 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 certa ora 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 assume 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.



Commento sullo schema E/R

In RESOCONTORARIO l'identificatore (TECNICO,DATA,ORA) assicura che in una certa ora di un certo giorno un tecnico effettui un'unica riparazione riparando, essendo $\text{card}(\underline{\text{PC,RESOCONTORARIO}}) = (1,1)$, 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 Relazionale

PERSONA (CF, INDIRIZZO)

PROPRIETARIO (CF, TELEFONO)

FK: CF REFERENCES PERSONA

TECNICO(CF, QUALIFICA, COSTOORARIO)

FK: CF REFERENCES PERSONA

PERSONAL(CODICE, CF)

FK: CF REFERENCES PROPRIETARIO

RESOCONTORARIO (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 RESOCONTORARIO

CAMBIA(DATA, ORA, CFTECNICO, NOMECOMP)

FK: DATA, ORA, CFTECNICO REFERENCES RESOCONTORARIO

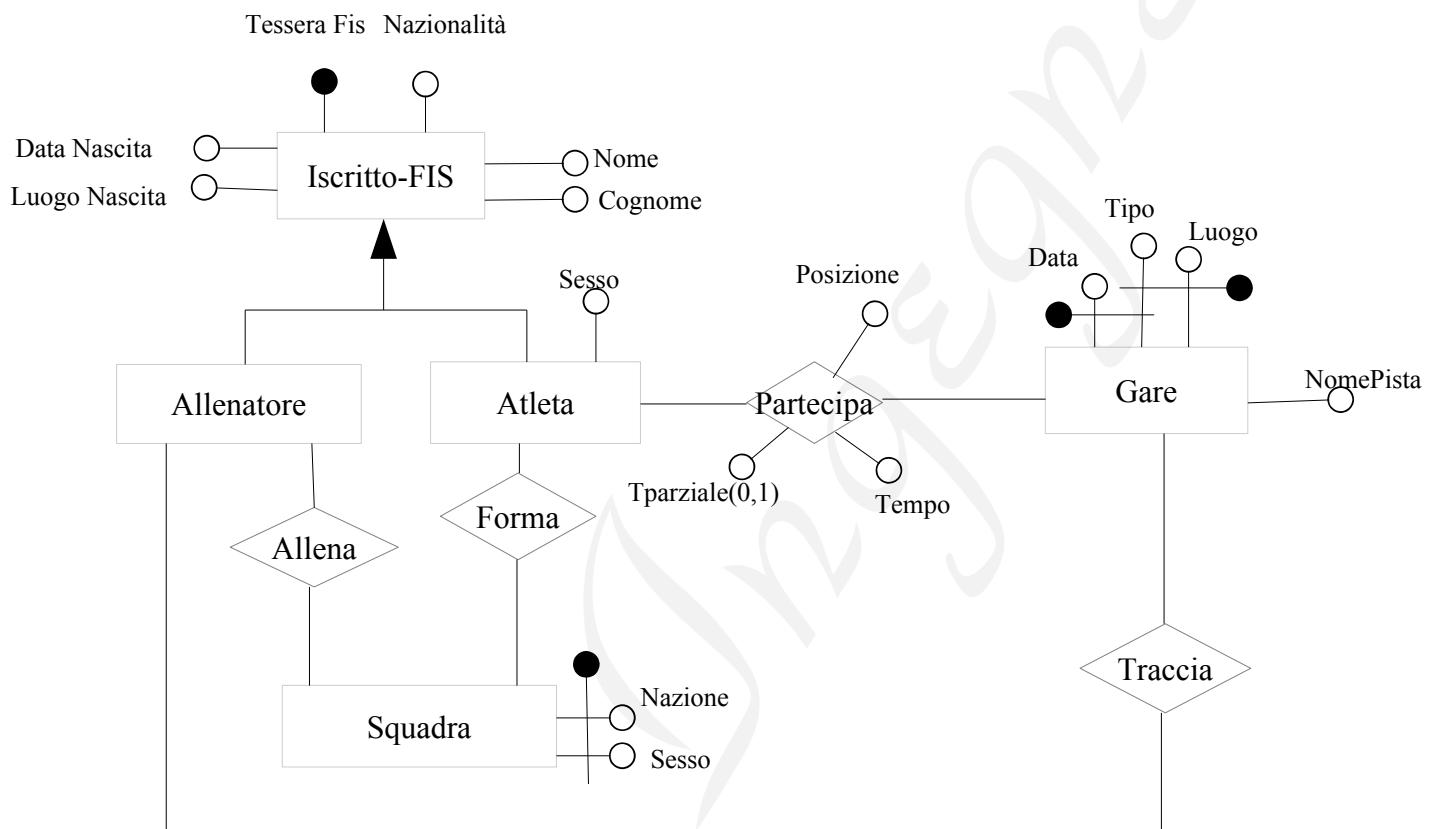
FK:NOMECOM REFERENCES COMPONENTE
RELAZIONEMENSILE (CFTECNICO, MESE, TOTORE)

FK: CFTECNICO REFERENCES TECNICO
COMPONENTE(NOME, DESCRIZIONE, COSTO)
CAMBIATI(CFTECNICO, MESE, NOMECOMP, NSOSTIT)

FK: CFTECNICO, MESE REFERENCES RELAZIONEMENSILE
FK:NOMECOM REFERENCES COMPONENTE

Modello E-R 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, Gigante Femminile, LiberaMaschile). In una certa data, non si possono svolgere due gare dello stesso tipo; in una 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.



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.

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

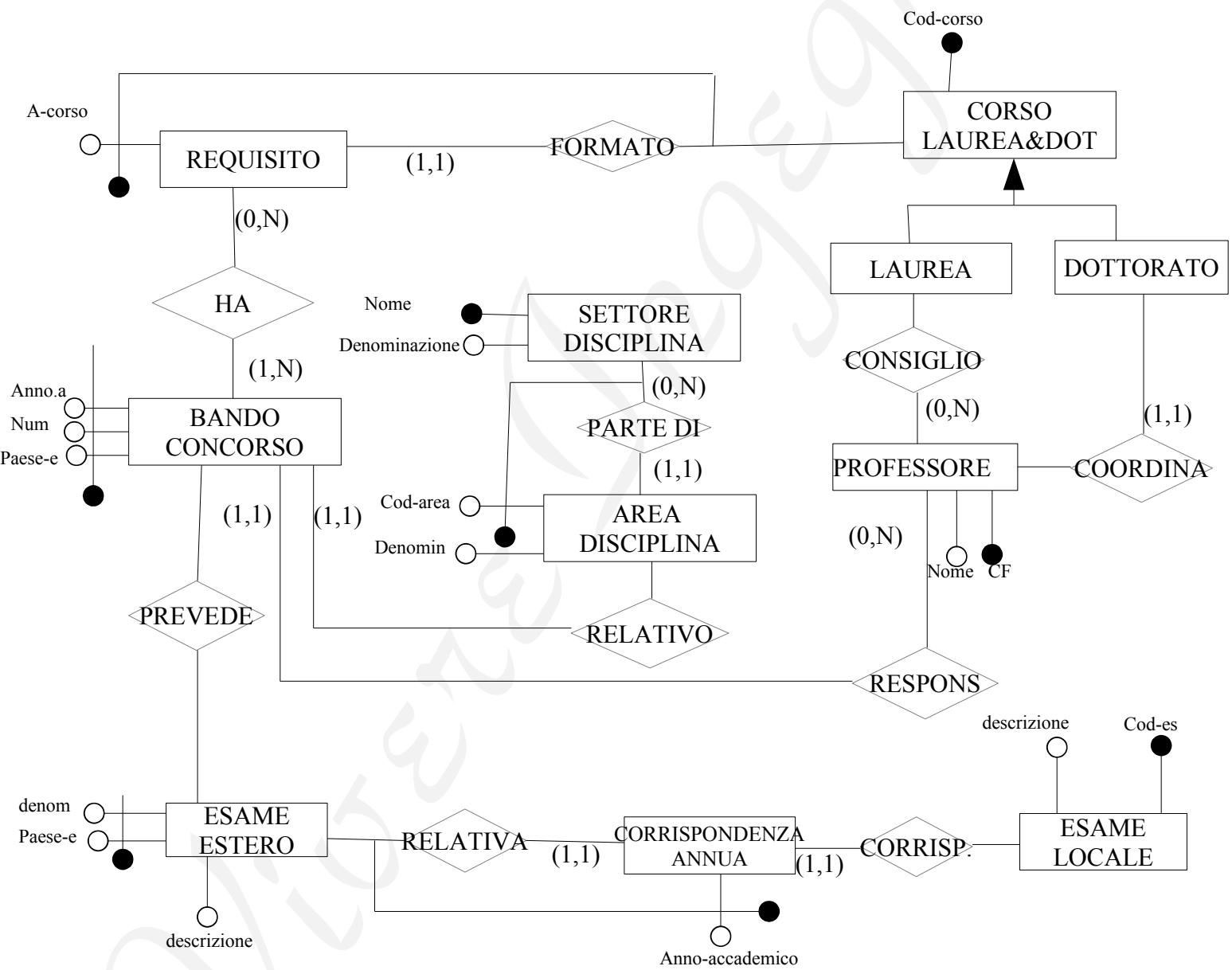
FK: TESSERAFIS REFERENCES ATLETA

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.

Modello E-R 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 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 e da una descrizione)



Schema Relazionale

Settore (Nome,Denominaz)

Area (Nome,CodArea,Denominaz)

FK: Nome REFERENCES Settore

Professore (CF,Nome)

Corso (Cod Corso,Laurea/Dottorato)

Coordina (CodCorso,CFProfessore)

FK: Cod Corso REFERENCES Corso

FK: CFProfessore REFERENCES Professore

Consiglio (CodCorso, CFProfessore)

FK: Cod Corso REFERENCES corso

FK: CFProfessore REFERENCES Professore

Requisito (ACorso,CodCorso)

FK: Cod Corso REFERENCES corso

Bandoconcorso (AA, Numero, PaeseEst, Area, Settore, CFResponsabile)

FK: Area Settore REFERENCES Area

FK: Responsabile REFERENCES Professore

HA (AA Numero, Paese Est, ACorso,CodCorso)

FK: PaeseEst REFERENCES Bandoconcorso

FK: Acorso, codcorso REFERENCES Requisito

Esame Estero (Denominaz,PaeseEst, Descrizione)

Prevede (AA,Numero, PaeseEst,Denominaz)

FK: AA Numero PaeseEst REFERENCES Bandoconcorso

FK: PaeseEst REFERENCES EsameEstero

Esame Locale (CodEs, Descrizione)

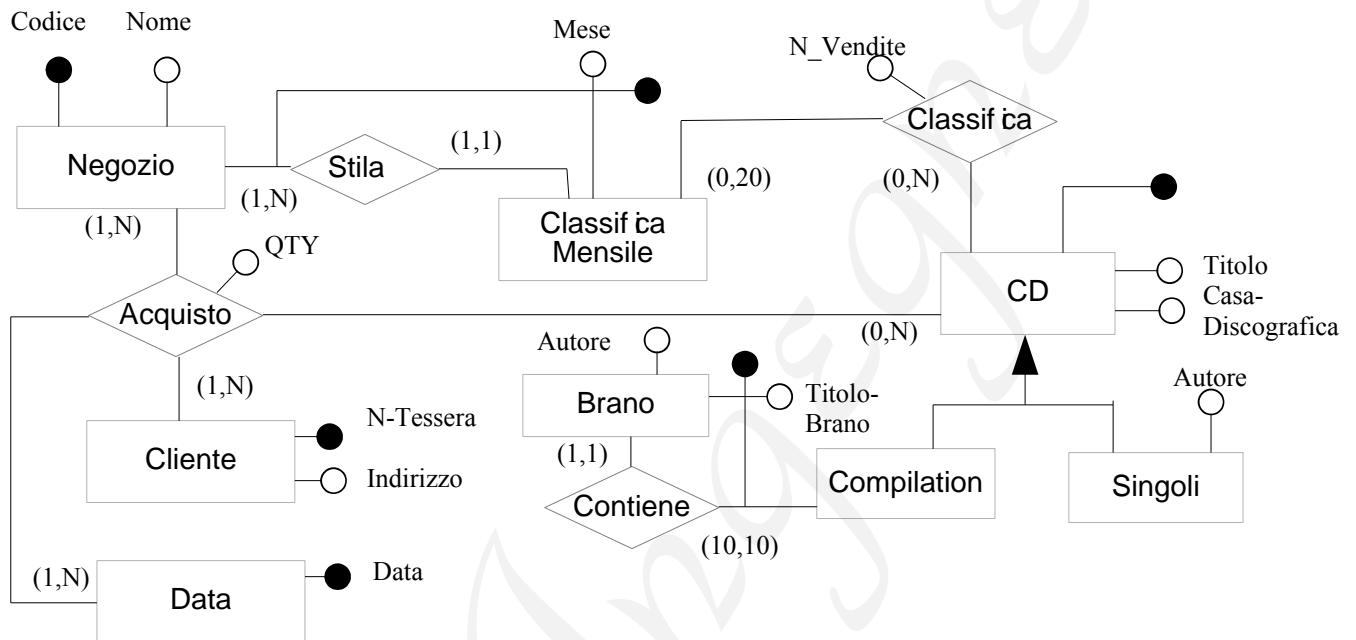
Corrispondenza Annuia (AA,Denominaz, PaeseEst, codEsLocale)

FK: Denominaz, PaeseEst REFERENCES Esame Estero

FK: Locale REFERENCES Esame Locale

Modello E-R

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 del brano in una compilation è identificativo. Ogni negozio, descritto da un codice (univoco) e da un nome, stila mensilmente una 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.



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 perché 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 (CODICE CD, AUTORE)

FK: CODICECD REFERENCES CD

COMPILATION (CODICECD)

FK: CODICECD REFERENCES CD

BRANI (CODICECD TITOLOBRANO,AUTORE)

FK: CODICE CD REFERENCES COMPILATION

ACQUISTO (CODICE,CODICECD NTESSERA,DATA,QTY)

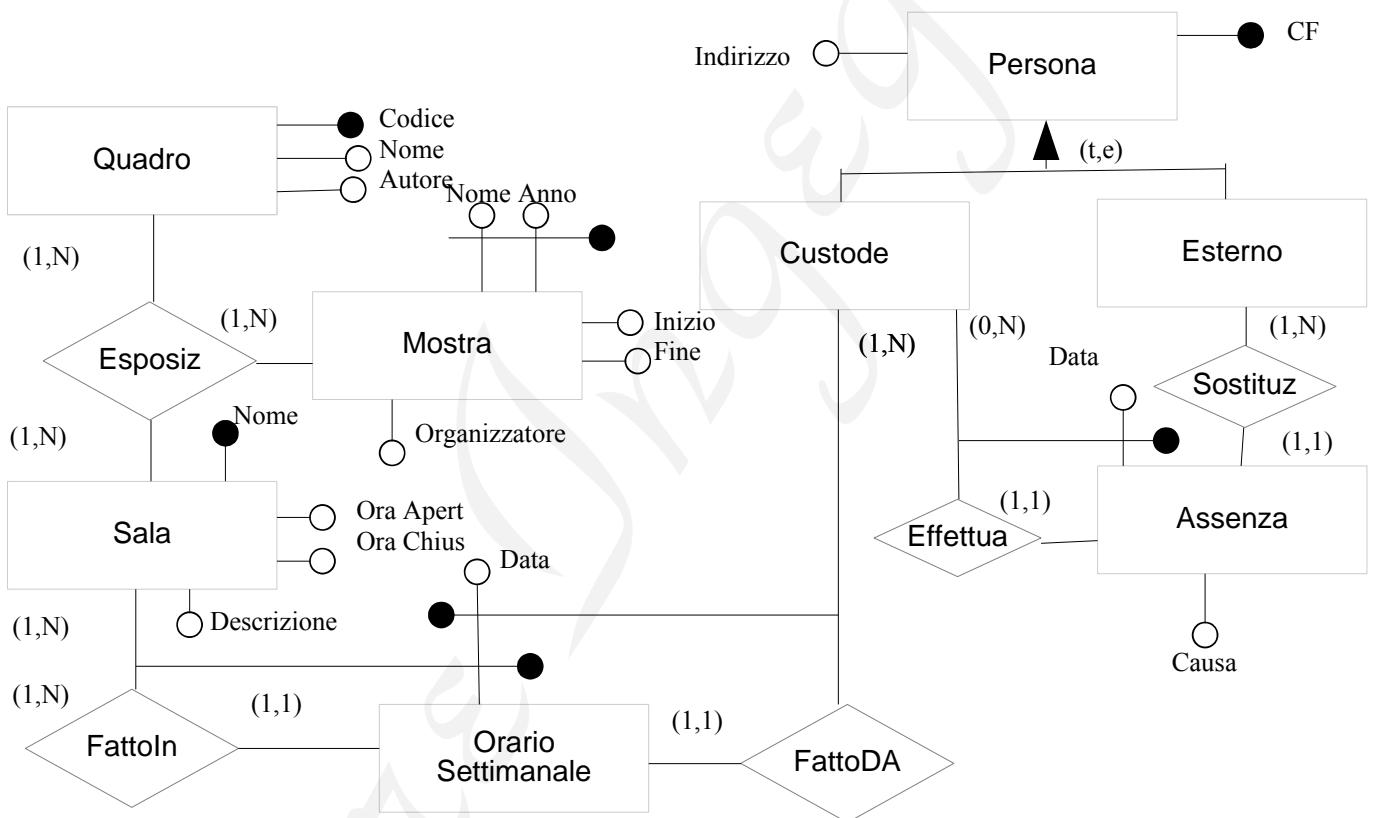
FK: CODICE REFERENCES NEGOZIO

FK: CODICECD REFERENCES CD

FK: NTESSERA REFERENCES CLIENTE

Modello E-R 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.



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.

Schema relazionale

CUSTODE(CF, INDIRIZZO)
 ESTERNO(CF, INDIRIZZO, TELEFONO)
 ORARIOSETTIMANALE(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

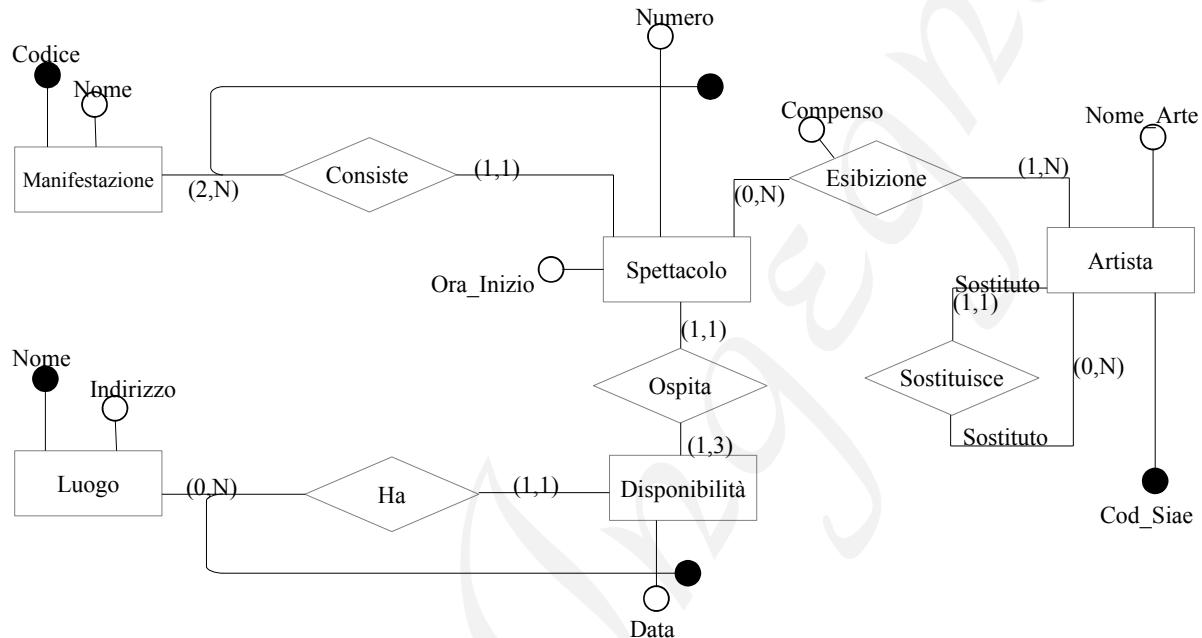
FK: NOMEQUADRO REFERENCES QUADRO

FK:NOMESALA REFERENCES SALA

Modello E-R 8

Si vogliono rappresentare informazioni relative alla gestione dei 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 dta, 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.



Schema Relazionale:

Manifestazione(Cod_Ma, Nome_M)

Luogo(Nome_L, Indirizzo)

Disponibilità(Ref_Nome_L, Data)

FK: Ref_Nome_L REFERENCES Luogo(Nome_L)

Spettacolo(Ref_Cod_Ma, Numero, Ora_Inizio, R_Nome_L, R_Data)

FK: Ref_Cod_Ma REFERENCES Manifestazione(Cod_Ma)

FK: (R_Nome_L, R_Data) REFERENCES Disponibilità(Ref_Nome_L, Data)

Artista:(Cod_Siae, Nome_Arte, CodSiae_Sost)

FK: CodSiae_Sost REFERENCES Artista(Codice_Siae)

Esbizione(R_Cod_Siae, R_Cod_Ma, R_Numero, Compenso)

FK: R_Cod_Siae REFERENCES Artista(Cod_Siae)

FK: (R_Cod_Ma, R_Numero) REFERENCES Spettacolo (Ref_Cod_Ma, Numero).

Schema Relazionale modificato:

Esbizione(R_Cod_Siae, R_Cod_Ma, R_Numero, Compenso)

Spettacolo(Ref_Cod_Ma, Numero, Ora_Inizio, R_Nome_L, R_Data)

AK: Ref_Cod_Ma, Luogo

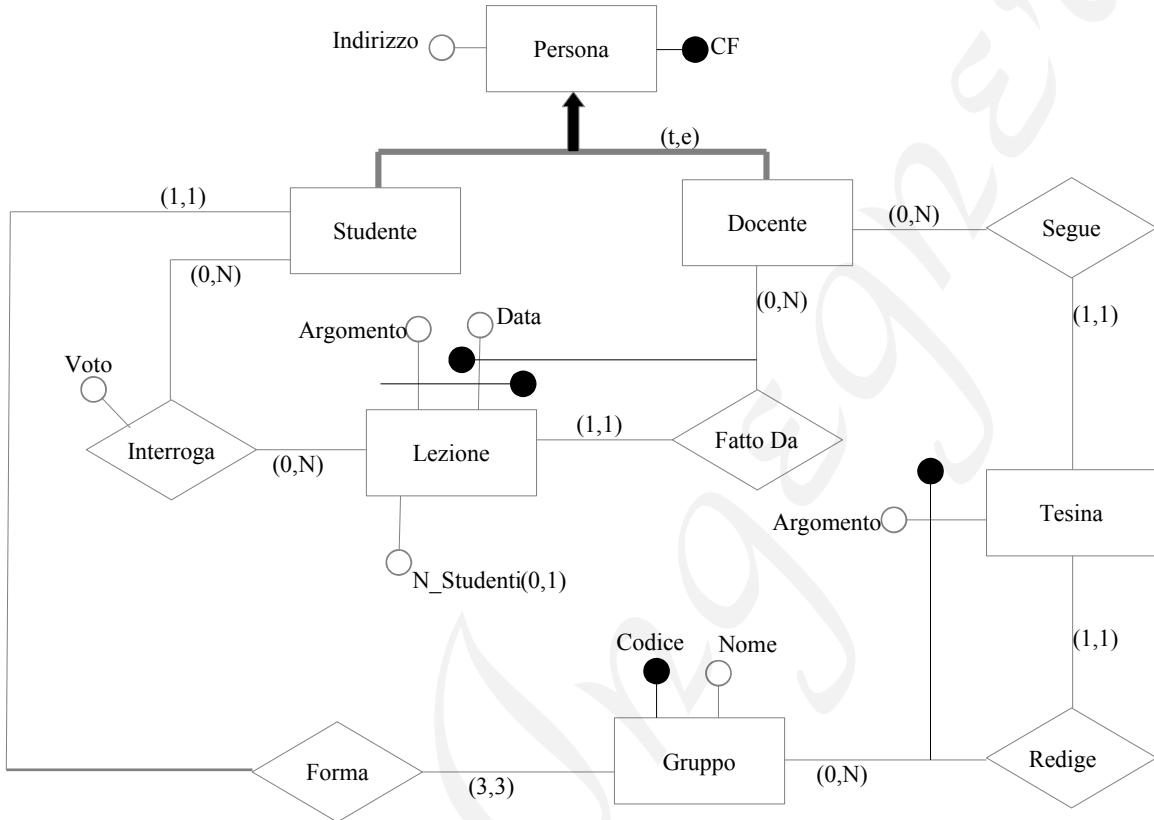
Commento sullo schema E/R:

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 di spettacolo-artista. Se si esprimesse OSPITA come associazione binaria tra LUOGO rSPETTACOLO sarebbe un errore riportare Card(LUOGO,OSPITA)=(0,3): infatti in questo modo si vincolerebbe un luogo ad ospitare, in tutto, tre spettacoli, mentre questo vincolo è valido solo per una certa data. In questo caso si dovrebbe mettere card(LUOGO, OSPITA)=(0,N) e card(SPETTACOLO,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 esprimere occorre necessariamente introdurre l'entità DISPONIBILITÀ'.

Modello E-R 9

Si vogliono memorizzare i dati delle 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 a 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.



Schema Relazionale:

Gruppo(Codice, Nome)

Studente(Cf, Indirizzo, Cod_G)

FK: Cod_G REFERENCES Gruppo(Codice)

Docente(Cf_D, Indirizzo_D)

Lezione(Data, Argomento, N_Stud, Ref_Cf_D)

AK: Data, Ref_Cf_D

FK: Ref_Cf_D REFERENCES Docente(Cf_D)

Interrogazione(Data_I, Argomento_I, Ref_Cf, Voto)

FK: Data_I, Argomento_I REFERENCES Lezione(Data, Argomento)

FK: Ref_Cf REFERENCES Studente(Cf)

Tesina(R_Cod_G, Ref_Argomento, R_Cf_D)

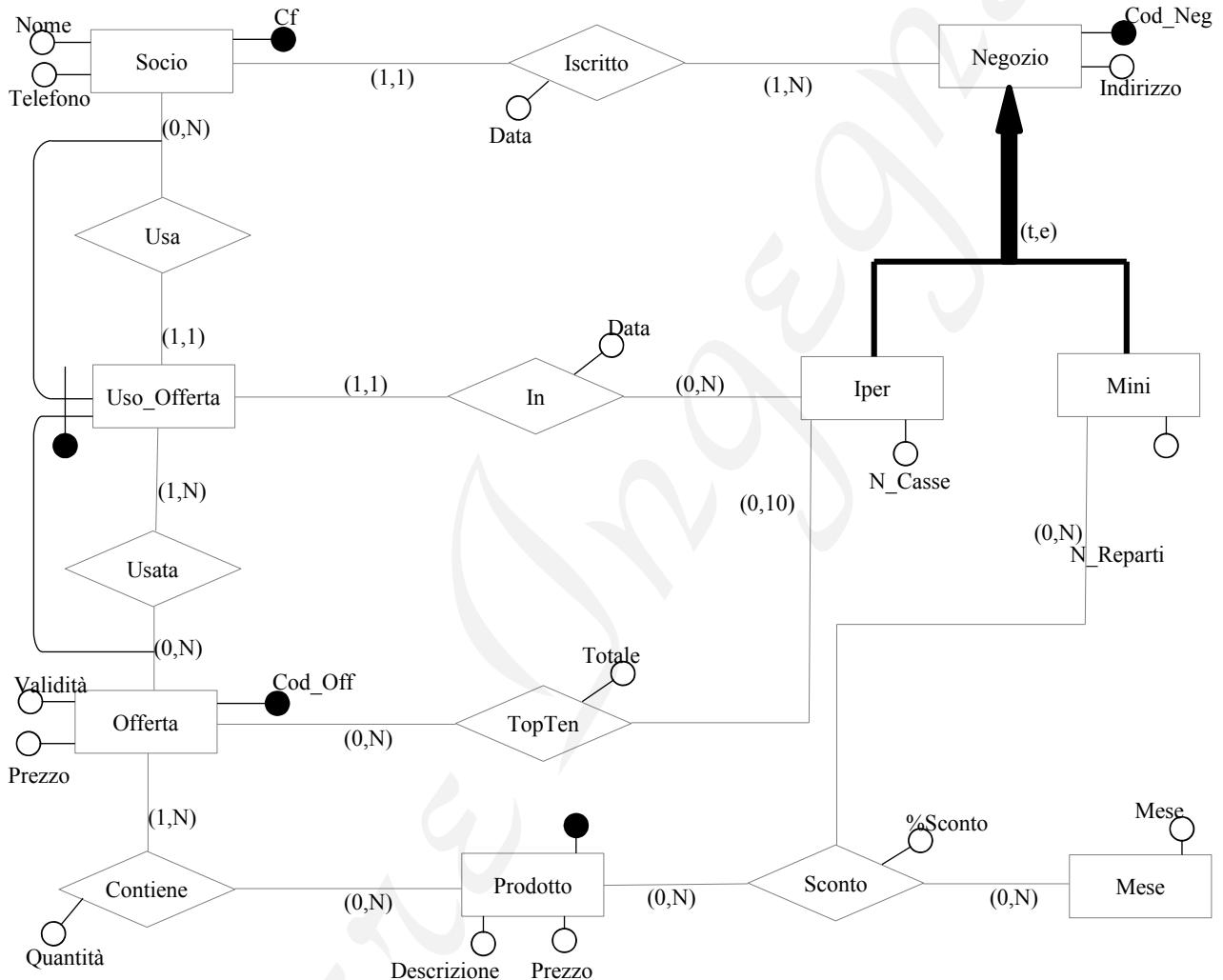
FK: R_Cf_D REFERENCES Docente(Cf_D)

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.

Modello E-R 10

Si vogliono rappresentare informazioni relative alle offerte ed agli conti 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 di 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.



Commento sullo schema E/R:

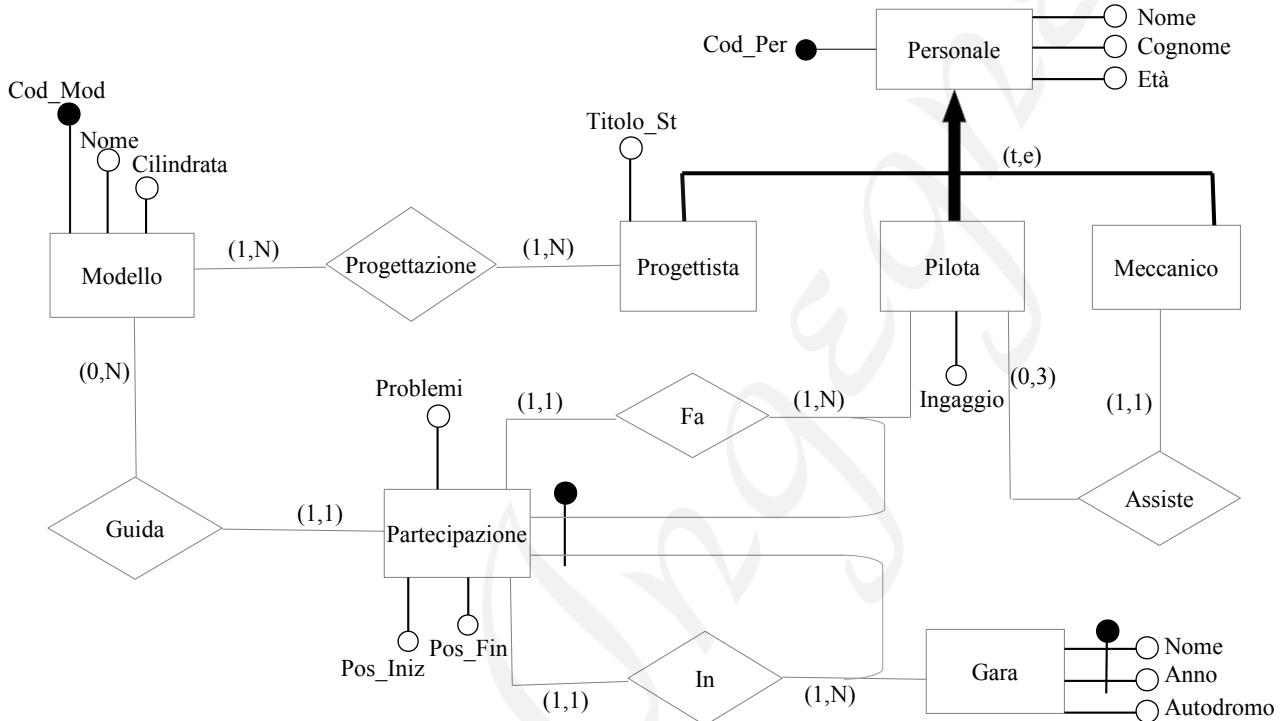
Il fatto che un socio non possa usufruire due volte della stessa offerta, si potrebbe esprimere tramite una associazione binaria (UsoOffer) molti a molti tra Socio e Offerta. Però siccome l'ipermarket dove viene effettuato il ritiro dell'offerta non è necessariamente il punto vendita (Negozio) in cui 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 un'associazione molti 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 Card(Iper, TopTen)=(10,10). D'altra parte con CardMin(Iper, TopTen)=N, si considerano anche i casi, rispettivamente, di offerta non ritirata in nessun ipermarket e di più di dieci offerte a pari merito.

Modello E-R 11

Il sistema informativo di un'azienda che produce motocicli memorizza dati su: produzione di motocicli da competizione, disputa di gare internazionali e il personale impiegato su tali attività in base alle seguenti specifiche: ogni modello di motociclo ha un codice, un nome e una cilindrata. Ogni membro del personale ha: nome, cognome, 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 ad un pilota; ogni pilota può avere assegnati al massimo tre meccanici, i piloti hanno un ingaggio; i progettisti hanno un titolo di studio. Ogni gara ha un nome (es. G.P. D'Italia, di Germania, ecc.), un anno e un autodromo in cui è disputata. Una gara non è 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 questa partecipazione 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.



Schema Relazionale:

Progettista(Cod_Per, Nome,Cognome, Età, Titolo_Stud)

Pilota(Cod_Per_P, Nome_P, Cognome_P, Età_P, Ingaggio)

Meccanico(Cod_Per_M, Nome_M, Cognome_M, Età_M, Ref_Cod_P)

FK: Ref_Cod_P REFERENCES Pilota(Cod_Per_P);

Modello(Cod_Mod, Nome_Mod, Cilindrata)

Progettazione(Ref_Cod_Mod, Ref_Cod_Per)

FK: Ref_Cod_Mod REFERENCES Modello(Cod_Mod)

FK: Ref_Cod_Per REFERENCES Progettista(Cod_Per);

Gara(Nome_G, Anno, Autodromo)

Partecipazione(R_Cod_Per_P, R_Nome_G, R_Anno, R_Cod_Mod, Pos_Iniz, Pos_Fin)

FK: R_Cod_Per_P REFERENCES Pilota(Cod_Per_P)

FK: R_Nome_G REFERENCES Gara(Nome_G)

FK: R_Anno REFERENCES Gara(Anno);

Problemi(Re_Cod_Per_P, Re_Nome_G, Re_Anno_G, Problema)

FK: (Re_Cod_Per_P, Re_Nome_G, Re_Anno_G) REFERENCES Partecipazione (R_Cod_Per_P, R_Nome_G, R_Anno_G);

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.

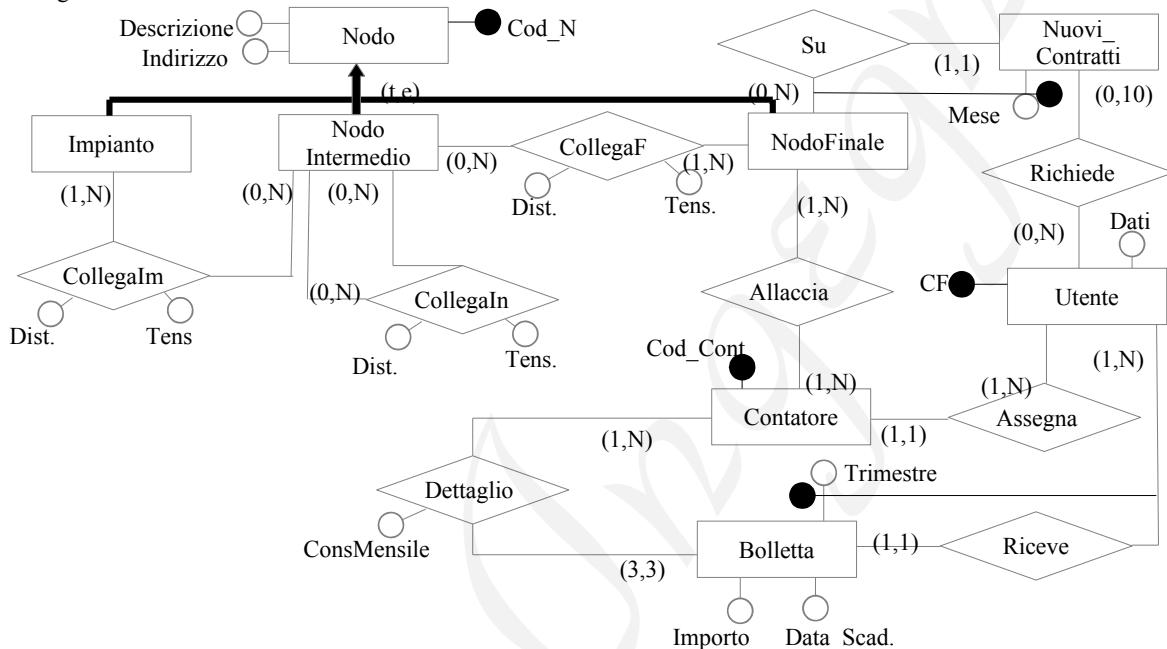
Modello E-R 12

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 connessi 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 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.



Schema Relazionale:

Impianto(Cod_I, Descrizione, Indirizzo)

NodoIntermedio(Cod_NI, Descrizione_NI, Indirizzo_NI)

NodoFinale(Cod_NF, Descrizione_NF, Indirizzo_NF)

CollegaIm(Ref_CodI, Ref_CodNI, Distanza, Tensione)

FK: Ref_CodI REFERENCES Impianto(Cod_I)

FK: Ref_CodNI REFERENCES NodoIntermedio(Cod_NI)

CollegaIn(Ri_CodNI, R_CodNI, Distanza1, Tensione1)

FK: Ri_CodNI REFERENCES NodoIntermedio(Cod_NI)

FK: R_CodNI REFERENCES NodoIntermedio(Cod_NI)

CollegaF(Re_Cod_NI, R_CodF, Distanza, Tensione)

FK: Re_CodNI REFERENCES NodoIntermedio(Cod_NI)

FK: R_CodF REFERENCES NodoFinale(Cod_NF)

Utente(CF, Dati)

Contatore(Cod_Con)

Allaccia(Ref_CodCn, Ref_CodFn)

FK: Ref_CodCn REFERENCES Contatore(Cod_Con)

FK: Ref_CodFn REFERENCES NodoFinale(Cod_NF)

Bolletta(R_CF, Trimestre, DataScad, Importo)

FK: R_CF REFERENCES Utente(CF)

Dettaglio(Ref_Cf, R_Trimestre, R_Cod_Con, ConsMensile)

FK: Ref_Cf REFERENCES Utente(CF)

FK: R_Trimestre REFERENCES Bolletta(Trimestre)

FK: R_CodC_Con REFERENCES Contatore(Cod_Con)

NuoviContratti(Ref_CodCo, Mese)

FK: Ref_CodCo REFERENCES Contatore(Cod_Con)

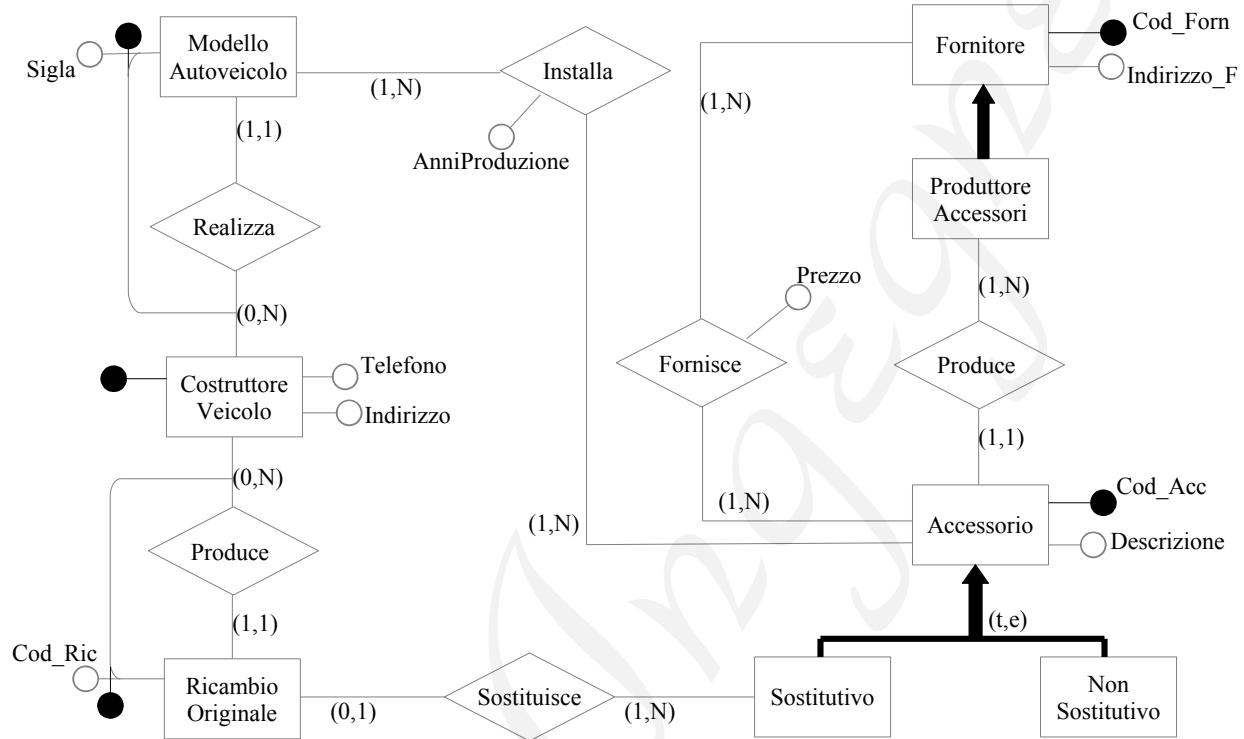
Richiede(Re_CF, RCod_Co, R_Mese)

FK: Re_CF REFERENCES Utente(CF)

FK: RCod_Co, R_Mese REFERENCES NuoviContratti(Ref_CodCo, Mese).

Modello E-R 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.



Schema Relazionale:

CostruttoreAuto(Nome, Telefono, Indirizzo)
ModelloAuto(NomeCostruttore, Sigla)

FK: NomeCostruttore **REFERENCES** CostruttoreAuto(Nome)

RicambioOriginale(Ref_Nome, Cod_Ric, Ref_Cod_AS)

FK: Ref_Nome **REFERENCE** CostruttoreAuto(Nome)

FK: Ref_Cod_AS **REFERENCES** Accessorio(CodAcc)

Accessorio(CodAcc, Descrizione, sostitutivo, R_Cod_P)

FK: R_Cod_P **REFERENCES** Produttore(Cod_P)

Installa((R_CodA, Ref_NCost, R_Sigla, AnnoProduzione))

FK: R_CodA **REFERENCES** Accessorio(CodAcc)

FK: Ref_NCost, R_Sigla **REFERENCES** ModelloAuto(NomeCostruttore, Sigla)

Fornisce(R_CodAcc, R_NomeC, Prezzo)

FK: R_CodAcc **REFERENCES** Accessorio(CodAcc)

FK: R_NomeC **REFERENCES** ModelloAuto(NomeCostruttore)

Fornitore(CodForn, IndirizzoF)

ProduttoreAccessori(CodProduttore)

FK: CodProduttore **REFERENCES** Fornitore(CodForm)

Produce(R_CodProduttore, Re_CodA)

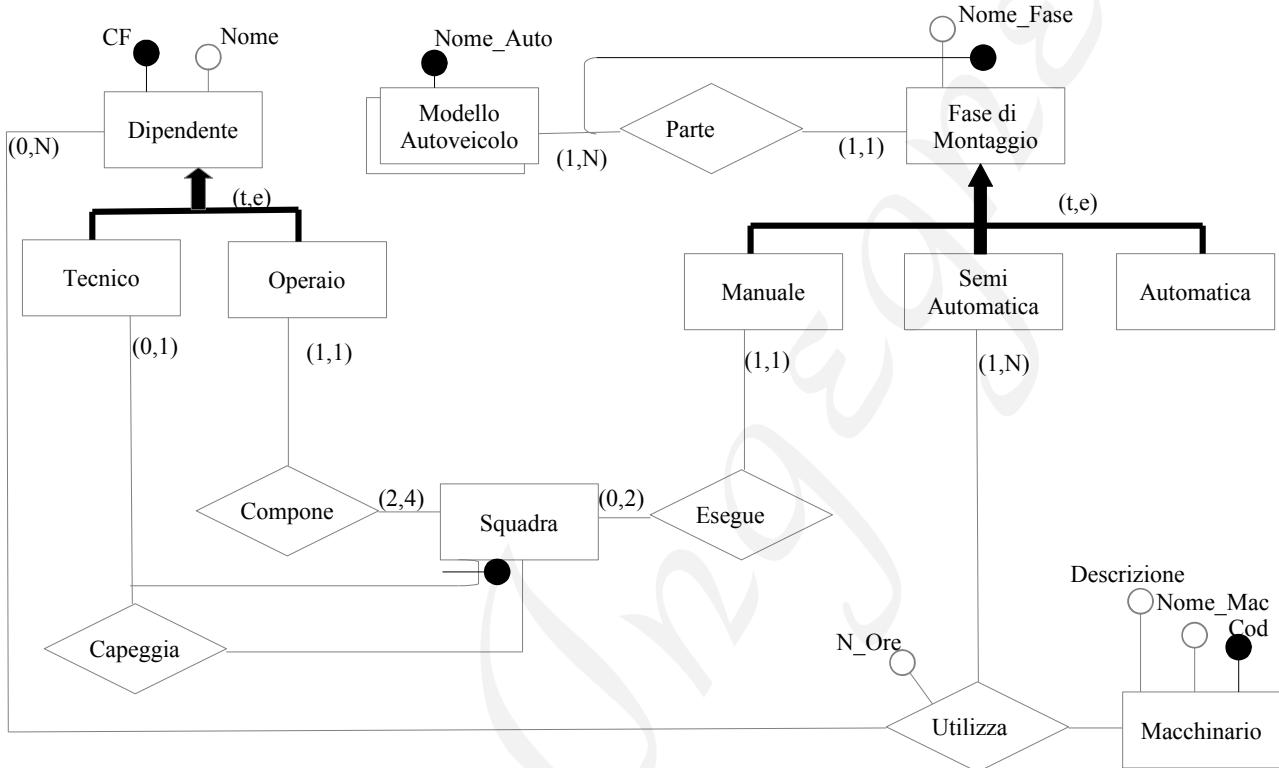
FK: R_CodProduttore **REFERENCES** ProduttoreAccessori(CodProduttore)

FK: Re_CodA **REFERENCES** Accessorio(CodAcc).

Modello E-R 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, semiautomatriche e manuali; ogni fase di montaggio ha un nome e fa parte dell'assemblaggio di un solo veicolo. 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 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.



Schema Relazionale:

ModelloAuto(NomeAuto)

FaseMontaggio(Ref_NomeA, NomeFase, Tipo, CfCapoSquadra)

FK: Ref_NomeA REFERENCES ModelloAuto(NomeAuto)

FK: CfCapoSquadra REFERENCES Squadra(CF_S)

Dipendente(Cf, Nome)

Tecnico(Cf_T)

FK: Cf_T REFERENCES Dipendente(Cf)

Squadra(Cf_S)

FK: Cf_S REFERENCES Dipendente(Cf)

Operaio(Cf_O, R_CfCapoSquadra)

FK: Cf_O REFERENCES Dipendente(Cf)

FK: R_CfCapoSquadra REFERENCES Squadra(Cf_S)

Macchinario(CodM, NomeMacchinario, Descrizione)

Utilizza(R_Cf, R_CodM, R_NomeFase, R_NomeAuto, NumOre)

FK: R_Cf REFERENCES Dipendente(Cf)

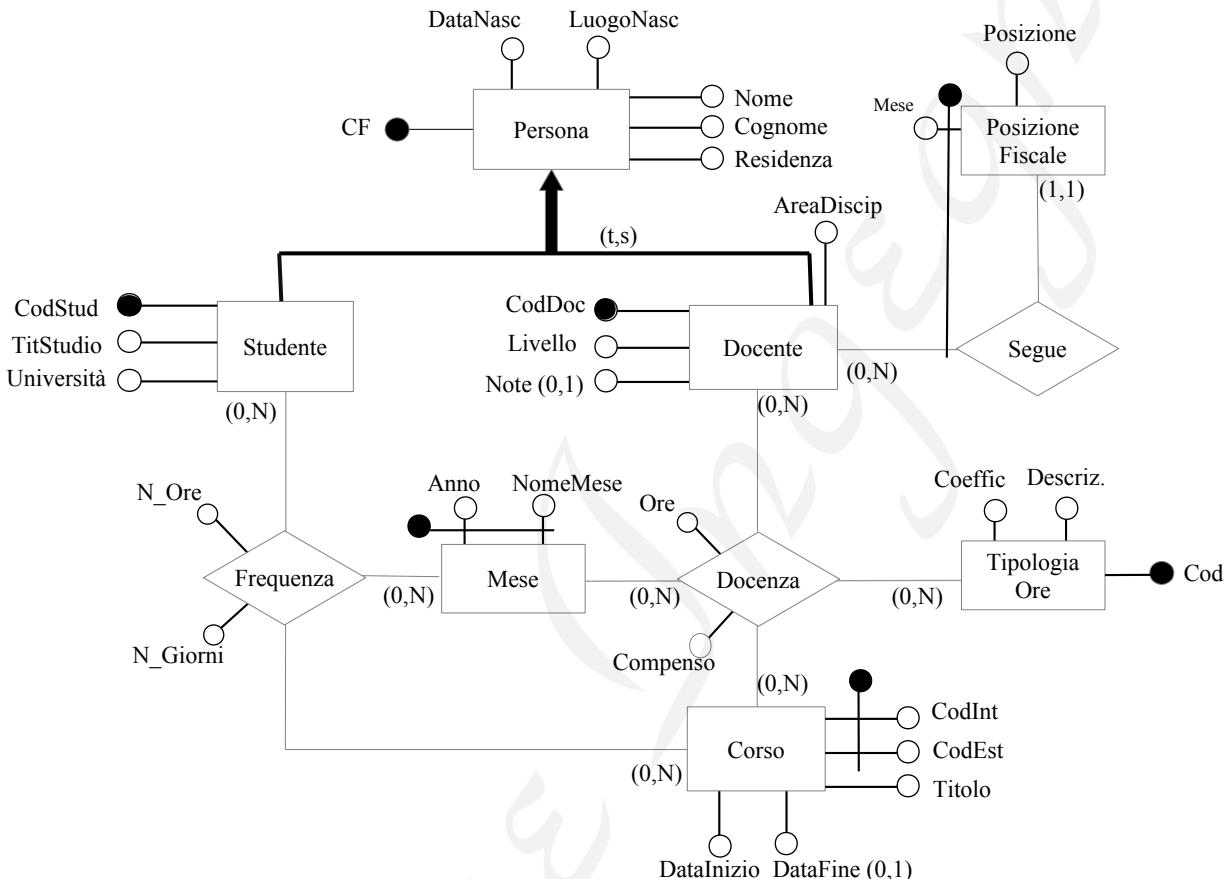
FK: R_CodM REFERENCES Macchinario(CodM)

FK: R_NomeFase, R_NomeAuto REFERENCES FaseMontaggio(Ref_NomeA, NomeFase).

Modello E-R 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 fine (quest'ultima facoltativa). Un altro soggetto coinvolto è lo studente e un docente potrebbe anche essere 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 delle 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.



Schema Relazionale:

PERSONA (CF, Nome, Cognome, Residenza, LuogoNasc, DataNasc)

STUDENTE (CFStud, CodStud, TitStudio, Università)

AK: CodStud

FK: CFStud REFERENCES Persona (CF)

DOCENTE (CFDoc, CodDoc, Livello, Note)

AK: CodDoc

FK: CFDoc REFERENCES Persona (CF)

POSIZIONEFISCALE (Ref_Doc, Mese, Posizione)

FK: Ref_Doc REFERENCES Docente (CFDoc)

CORSO (CodInt, CodEst, Titolo, DataInizio, DataFine)

TIPOLOGIAORE (Cod, Coeffic, Descrizione)

FREQUENZA (Ref_Stud, AnnoF, MeseF, Ref_CorsoInt, Ref_CorsoEst, N_Ore, N_Giorni)

FK: Ref_Stud REFERENCES Studente (CFStud)

FK: (Ref_CorsoInt, Ref_CorsoEst) REFERENCES Corso (CodInt, CodEst)

DOCENZA (R_Doc, AnnoD, MeseD, R_CorsoInt, R_CorsoEst, CodOre, Compenso, NumOre)

FK: R_Doc REFERENCES Docente (CFDoc)

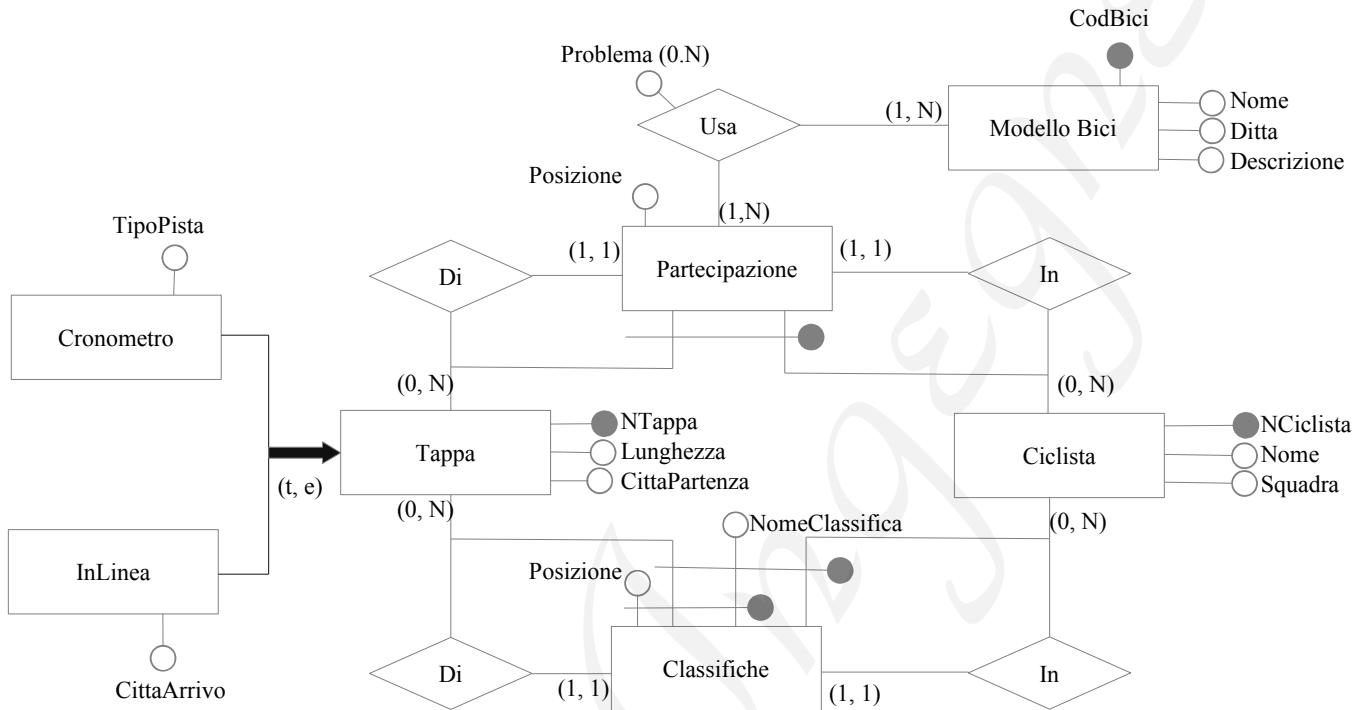
FK: (R_CorsoInt, R_CorsoEst) REFERENCES Corso (CodInt, CodEst)

FK: CodOre REFERENCES TipologiaOre (Cod)

Modello E-R 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 "MagliaCiclismo". 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 "MarcoCipollini" occupa la quarta posizione nella classifica della "MagliaRosa" e la prima posizione in quella della "MagliaCiclismo". I ciclisti sono rappresentati tramite un numero univoco, un nome e la squadra di appartenenza.



Schema Relazionale:

TAPPA (NTappa, Lunghezza, CittaPartenza, CittaArrivo, TipoPista)

CICLISTA (NCiclista, Nome, Squadra)

MODELLOBICI (CodBici, NomeBici, Ditta, Descrizione)

PARTECIPAZIONE (Ref_Tappa, Ref_Ciclista, PosizioneC)

FK: Ref_Tappa REFERENCES Tappa (NTappa)

FK: Ref_Ciclista REFERENCES Ciclista (NCiclista)

USA (R_Tappa, R_Ciclista, Ref_Bici)

FK: (R_Tappa, R_Ciclista) REFERENCES Partecipazione (Ref_Tappa, Ref_Ciclista)

FK: Ref_Bici REFERENCES Modellobici (CodBici)

PROBLEMI (Ref_NTappa, Ref_NCiclista, R_Bici, Problema)

FK: (Ref_NTappa, Ref_NCiclista, R_Bici) REFERENCES USA (R_Tappa, R_Ciclista, Ref_Bici)

CLASSIFICA (R_NTappa, NomeClassifica, Posizione, R_NCiclista)

AK: R_NTappa, NomeClassifica, R_NCiclista

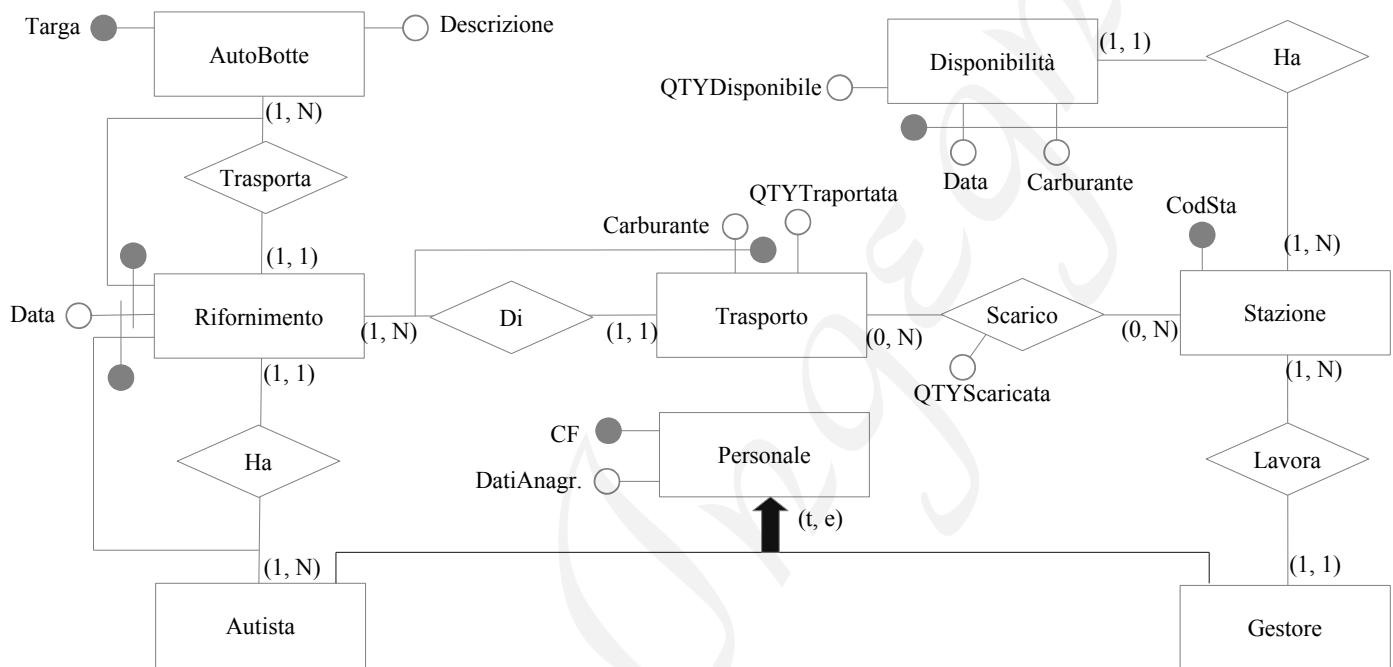
FK: R_NTappa REFERENCES Tappa (Ntappa)

FK: R_NCiclista REFERENCES Ciclista (NCiclista)

Modello E-R 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. 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 stazioni di servizio, scaricando in ciascuna di esse uno o più tipo di carburante trasportati in una certa quantità; (ad esempio, il precedente rifornimento scarica nella stazione "Pioppa" 2 tonnellate di "BenzinaVerde" e 5 di "BenzinaSuper", nella stazione "ModenaOvers" 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.



Schema Relazionale:

AUTOBOTTE (Targa, Descrizione)

AUTISTA (CFA, DatiAnagrafici)

GESTORE (CFG, DatiAnagrafici, Ref_Sta)

FK: Ref_Sta REFERENCES Stazione (CodSta)

STAZIONE (CodSta)

RIFORNIMENTO (Ref_Targa, Data, Ref_Autista)

AK: Data, Ref_Autista

FK: Ref_Targa REFERENCES Autobotte (Targa)

FK: Ref_Autista REFERENCES Autista (CFA)

TRASPORTO (R_Targa, Ref_Data, Carburante, QTYTrasportata)

FK: (R_Targa, Ref_Data) REFERENCES Rifornimento (Ref_Targa, Data)

SCARICO (Ref_AutoBotte, R_Data, Ref_Carburante, R_Sta, QTYScaricata)

FK: (Ref_Autobotte, R_Data, Ref_Carburante) REFERENCES Trasporto (R_Targa, Ref_Data, Carburante)

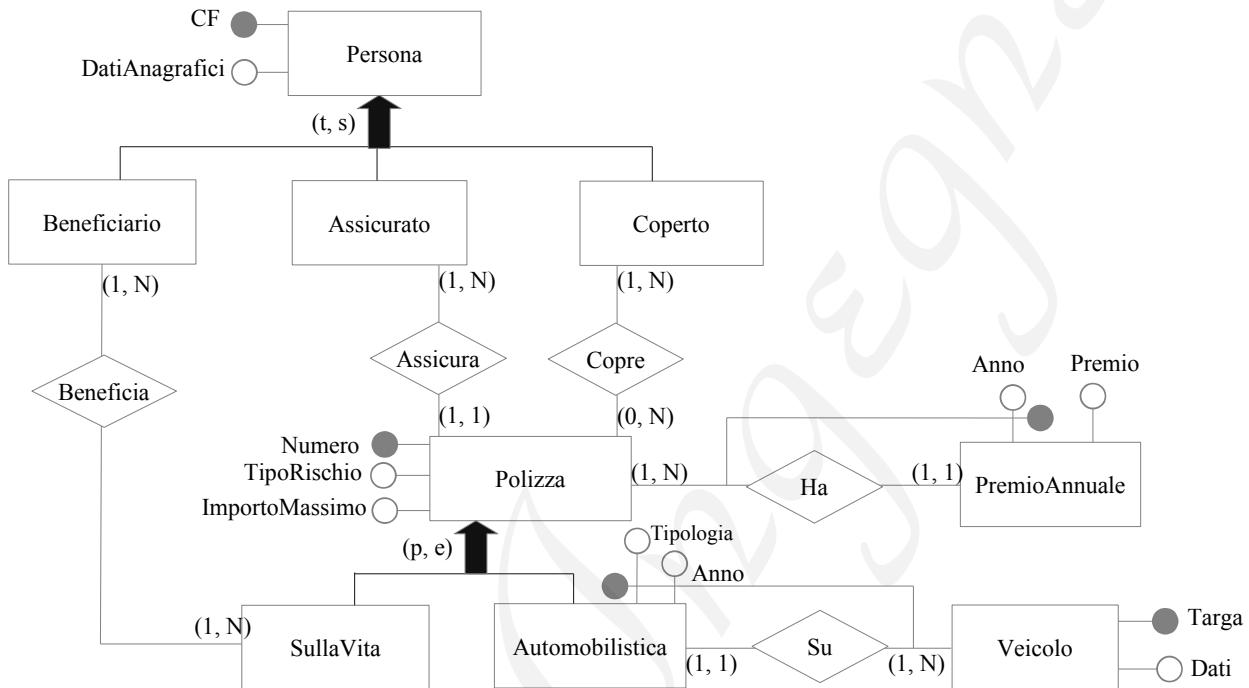
DISPONIBILITA' (Ref_Stazione, DataD, CarburanteD, QTYDisponibile)

FK: Ref_Stazione REFERENCES Stazione (CodSta)

Modello E-R 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.



Schema Relazionale:

PERSONA (CF, DataAnagrafici, TipoPersona)

POLIZZA (Numero, TipoRischio, MaxImporto, CFAssicurato)

FK: CFAssicurato REFERENCES Persona (CF)

PREMIOANNUALE (Ref_Polizza, Anno, Premio)

FK: Ref_Polizza REFERENCES Polizza (Numero)

COPRE (R_Polizza, CFCoperto)

FK: R_Polizza REFERENCES Polizza (Numero)

FK: CFCoperto REFERENCES Persona (CF)

POLIZZASULLAVITA (Ref_NPolizza)

FK: Ref_NPolizza REFERENCES Polizza (Numero)

POLIZZAAUTOMOBILISTICA (R_NPolizza, Tipologia, Anno, Ref_Targa)

AK: Tipologia, Anno, Ref_Targa

FK: R_NPolizza REFERENCES Polizza (Numero)

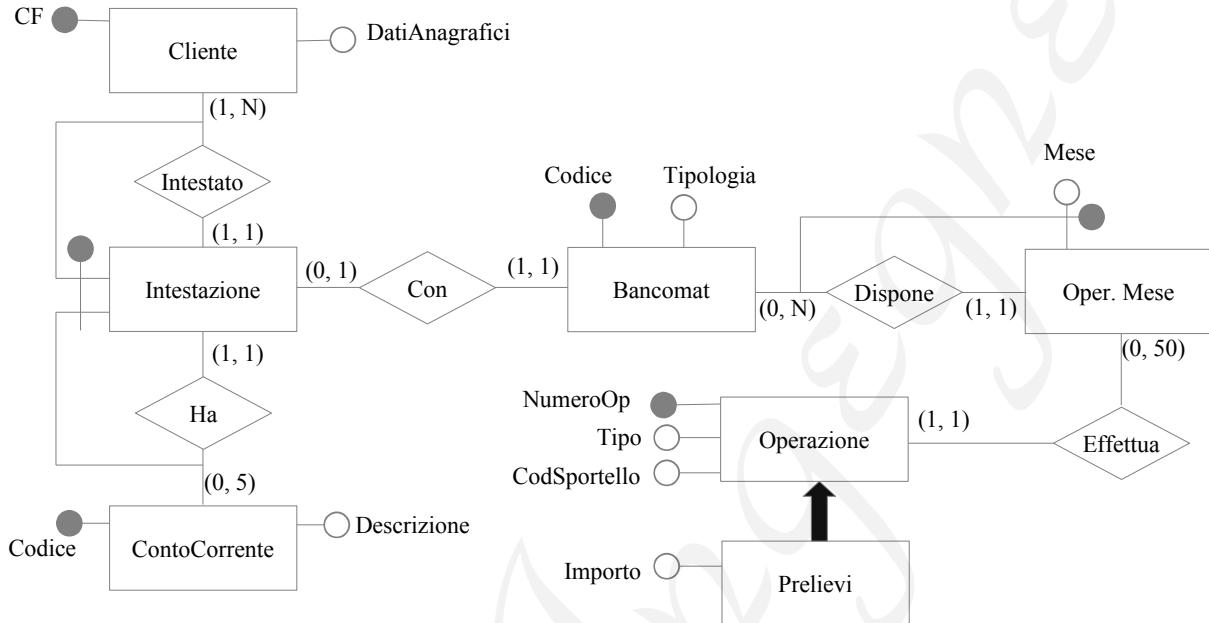
FK: Ref_Targa REFERENCES Veicolo (Targa)

VEICOLO (Targa, Dati)

Modello E-R 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 conto corrente 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.



Schema Relazionale:

CLIENTE (CF, DatiAnagrafici)

CONTOCORRENTE (CodCont, Descrizione)

INTESTAZIONE (Ref_Cliente, Ref_Conto)

FK: Ref_Cliente REFERENCES **Cliente** (CF)

FK: Ref_Conto REFERENCES **ContoCorrente** (CodCont)

BANCOMAT (CodBancomat, R_Cliente, R_Conto, Tipologia)

AK: R_Cliente, R_Conto

FK: (R_Cliente, R_Conto) REFERENCES **Intestazione** (Ref_Cliente, Ref_Conto)

OPERAZIONIMENSILI (Ref_Bancomat, Mese)

FK: Ref_Bancomat REFERENCES **BANCOMAT** (CodBancomat)

OPERAZIONE (NumeroOp, Tipo, CodSportello, R_Bancomat, Ref_Mese)

FK: (R_Bancomat, Ref_Mese) REFERENCES **OperazioniMensili** (Ref_Bancomat, Mese)

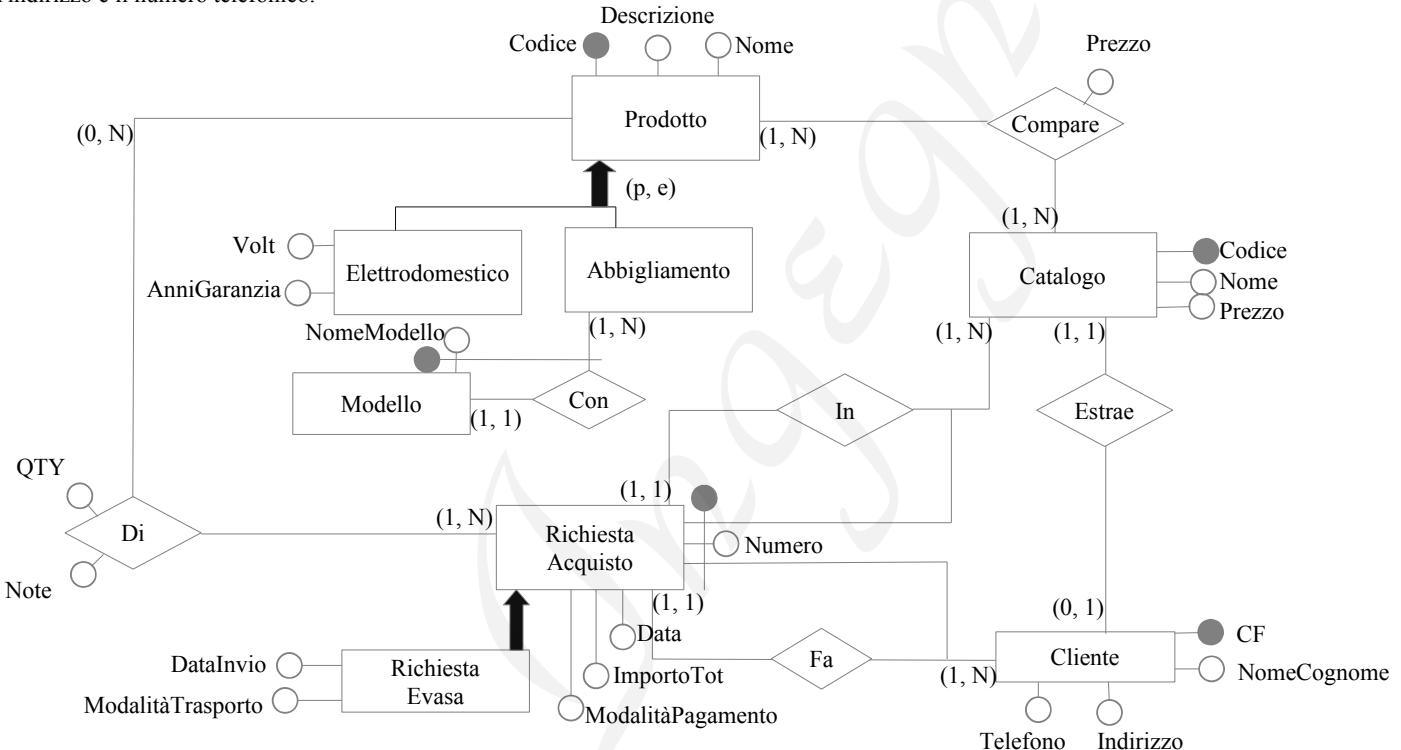
PRELIEVO (Ref_Operazione, Importo)

FK: Ref_Operazione REFERENCES **Operazione** (NumeroOp)

Modello E-R 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 comprarsi 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 comprarsi 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.



Schema Relazionale:

PRODOTTO (CodProd, NomeP, Descrizione)

ELETTRODOMESTICO (CodED, AnniGaranzia, Volt)

FK: Coded REFERENCES Prodotto (CodProd)

ABBIGLIAMENTO (CodAbb)

FK: CodAbb REFERENCES Prodotto (CodProd)

MODELLOTAGLIA (Ref_Abb, Modello, Taglia)

FK: Ref_Abb REFERENCES Abbigliamento (CodAbb)

CLIENTE (CF, NomeCognome, Indirizzo, Telefono)

CATALOGO (CodCat, NomeC, Prezzo, CFEstratto, BuonoSpesa)

AK: CFEstratto

FK: CFEstratto REFERENCES Cliente (CF)

COMPARE (Ref_Catalogo, Ref_Prodotto, Prezzo)

FK: Ref_Catalogo REFERENCES Catalogo (CodCat)

FK: Ref_Prodotto REFERENCES Prodotto (CodProd)

RICHIESTA (Ref_Cliente, R_Catalogo, Numero, ImportoTotale, ModalitaPagamento, Data)

FK: Ref_Cliente REFERENCES Cliente (CF)

FK: R_Catalogo REFERENCES Catalogo (CodCat)

RICHIESTAEVASA (Ref_CF, Ref_CodCat, Ref_Numero, ImportoTot, ModalitaTrasporto, DataInvio)

FK: (Ref_CF, Ref_CodCat, Ref_Numero) REFERENCES Richiesta (R_Cliente, R_Catalogo, Numero)

DI (R_CF, R_CodCat, R_Numero, R_Prodotto, QTY, Note)

FK: (R_CF, R_CodCat, R_Numero) REFERENCES Richiesta (R_Cliente, R_Catalogo, Numero)

Vincoli esterni:

Il vincolo che tutti i prodotti di una stessa richiesta devono comparire nello stesso catalogo non è espresso nello schema concettuale, mentre viene espresso nello schema di relazione 'DI' aggiungendo un vincolo di integrità referenziale:

FK: (R_CodCat, R_Prodotto) REFERENCES Compare (Ref_Catalogo, Ref_Prodotto)

Calcolo Computazionale 1

15/06/2017

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo 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, visualizzeremo 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

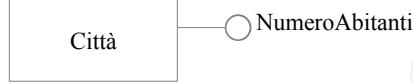
CONCETTO	TIPO	VOL.
Città	E	200
Persona	E	1000000

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper. 1	I	2/Giorno
Oper. 2	I	500/Giorno

Operazione 1:

Con il dato derivato

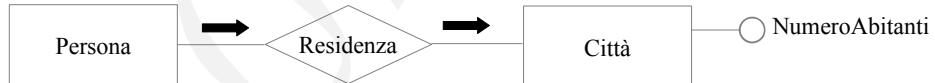


Senza il dato derivato



Operazione 2:

Con il dato derivato



Senza il dato derivato



Con il dato derivato:

Operazione 1
1 accesso in lettura
 $1 \times 2 = 2$ /Giorno

CONCETTO	ACC.	TIPO
Città	1	L

Operazione 2
1 accesso in lettura
3 accessi in scrittura
 $7 \times 500 = 3500$ /giorno

CONCETTO	ACC.	TIPO
Città	1	L
Residenza	5000	L
Persona	1	S
Residenza	1	S
Città	1	L
Città	1	S

Senza il dato derivato:

Operazione 1
5001 accessi in lettura
 $5001 \times 2 = 10002$ /giorno

CONCETTO	ACC.	TIPO
Città	1	L
Residenza	5000	L

Operazione 2
2 accessi in scrittura
 $4 \times 500 = 2000$ /giorno

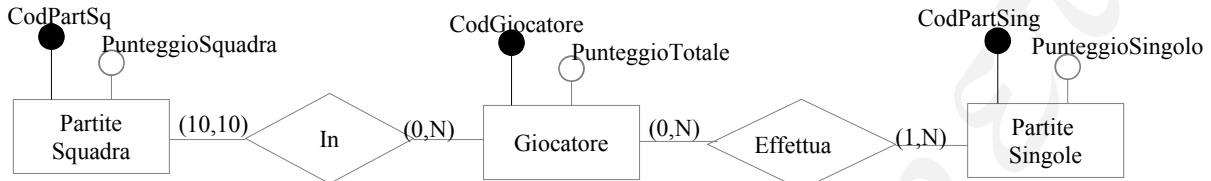
CONCETTO	ACC.	TIPO
Persona	1	S
Residenza	1	S

Conclusione: Conviene tenere il dato derivato.

Calcolo Computazionale 2

15/06/2017

Dato il seguente schema E/R, volume dei dati e operazioni, decidere se è conveniente conservare nello schema l'attributo derivato PUNTEGGIOTOTALE, che per un certo 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 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	ACC.	TIPO
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

Con il dato derivato:

	CONCETTO	ACC.	TIPO
Operazione 1			
1 accesso in lettura	PartitaSingola	1	S
3 accessi in scrittura	Effettua	1	S
(7)*100=700/giorno	Giocatore	1	L
	Giocatore	1	S
Operazione 2			
10 accessi in lettura	PartitaSquadra	1	S
21 accessi in scrittura	In	10	S
(52)*50=2600/giorno	Giocatore	10	L
	Giocatore	10	S
Operazione 3			
1 accesso in lettura	Giocatore	1	L
<u>1*15=15/giorno</u>			
Totale: 3315/giorno			

Senza il dato derivato:

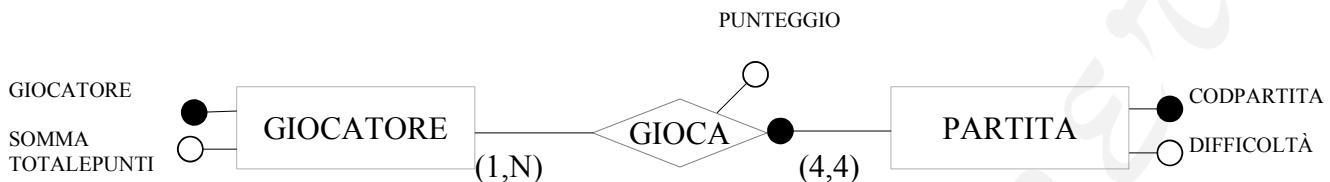
	CONCETTO	ACC.	TIPO
Operazione 1			
2 accessi in scrittura	PartitaSingola	1	S
(2*2)*100=400/giorno	Effettua	1	S
Operazione 2			
11 accessi in scrittura	PartitaSquadra	1	S
(11*2)*50=1100/giorno	In	10	S
Operazione 3			
121accessi in lettura	Giocatore	1	L
121*15=1815/giorno	In	50	L
	PartitaSquadra	50	L
	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.

Calcolo computazionale 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 SOMMATOTALEPUNTI che per un certo giocatore è calcolato come la somma di PUNTEGGIO*DIFFICOLTA di tutte le partite giocate. Si trascuri l'occupazione in memoria del dato derivato.



- Operazione1)** Introduzione di una nuova partita;
Operazione2) 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

Con il dato derivato :

	CONCETTO	ACC	TIPO
Operazione 1 4 accessi in lettura 9 accessi in scrittura $22 \times 40 = 880$ /giorno	Partita Gioca Giocatore Giocatore	1 4 4 4	S S L S
Operazione 2 1 accesso in lettura $1 \times 10 = 10$ /giorno	Giocatore	1	L
Totale :890/giorno			

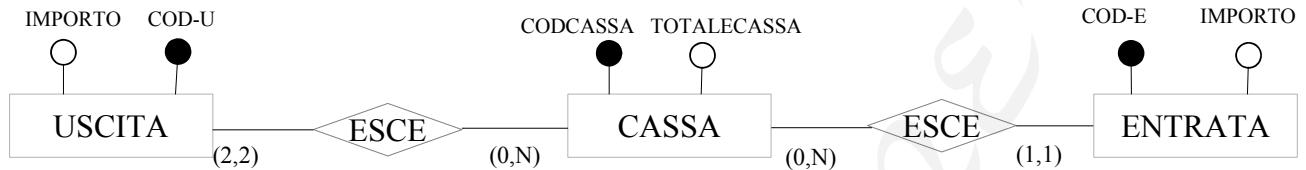
Senza il dato derivato

	CONCETTO	ACC	TIPO
Operazione 1 5 accessi in scrittura $10 \times 40 = 400$ /giorno	Partita Gioca	1 4	S S
Operazione 2 81 accesso in lettura $81 \times 10 = 810$ /giorno	Giocatore Gioca Partita	1 40 40	L L L
Totale :1210/giorno			

Conclusion: Conviene mantenere il dato derivato

Calcolo computazionale 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 differenza fra la somma degli importi delle entrate e la somma degli importi delle uscite. Si supponga che l'importo di un'entrata viene aggiunto al TOTALECASSA di una sola cassa mentre l'importo di un'uscita venga sottratto al TOTALECASSA di due casse definite. Si trascuri l'occupazione in memoria del dato derivato.



Operazione1) Inserimento di una nuova entrata (si suppone noto e valido il codice della cassa sulla quale l'entrata deve essere registrata);

Operazione2) Visualizzare i dati di una cassa. Si noti che nel caso in cui si deve determinare TOTALECASSA occorre leggere tutte le entrate e tutte le uscite di quella cassa;

Operazione3) Inserimento di una nuova uscita(si suppongono noti e validi i codici delle casse sulle quali l'uscita deve essere registrata).

Tavola dei Volumi

CONCETTO	TIPO	VOL.
Cassa	E	100
Entrata	E	3000
Uscita	E	1000

Con il dato derivato :

	CONCETTO	ACC	TIPO
Operazione 1 1 accesso in lettura 3 accessi in scrittura $7*100=700$ /giorno	Entrata	1	S
	Entra	1	S
	Cassa	1	L
	Cassa	1	S
Operazione 2 1 accesso in lettura $1*10=10$ /giorno	Cassa	1	L
Operazione 3 2 accessi in lettura 5 accessi in scrittura $12*1=12$ /giorno	Uscita	1	S
	Esce	2	S
	Cassa	2	L
	Cassa	2	S

Totale :722/giorno

Tavola delle operazioni

OPER.	TIPO	FREQ.
Oper. 1	I	100/Giorno
Oper. 2	I	10/Giorno
Oper. 3	I	1/Giorno

Senza il dato derivato:

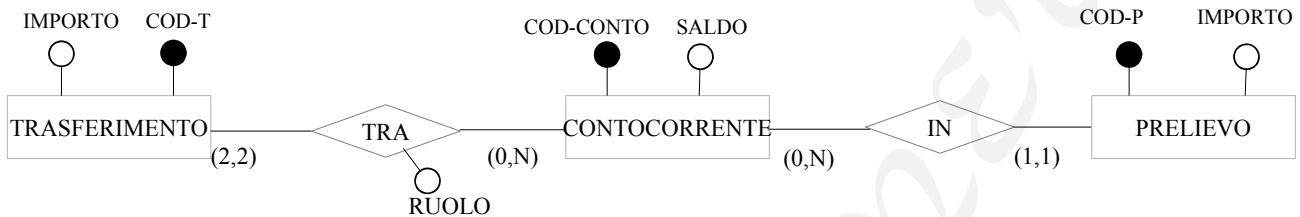
	CONCETTO	ACC	TIPO
Operazione 1 2 accessi in scrittura $4*100=400$ /giorno	Entrata	1	S
	Entra	1	S
Operazione 2 101 accessi in lettura $101*10=1010$ /giorno	Cassa	1	L
	Entra	30	L
	Entrata	30	L
	Esce	20	L
	Uscita	20	L
Operazione 3 3 accessi in scrittura $6*1=6$ /giorno	Uscita	1	S
	Esce	2	S

Totale :1413/giorno

Conclusione: Conviene tenere il dato derivato.

Calcolo computazionale 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 un 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 dal SALDO di un contocorrente (RUOLO = "Origine") ed aggiunto al SALDO di un altro contocorrente (RUOLO = "Destinazione"). Si trascuri l'occupazione in memoria del dato derivato.



Operazione1) Inserimento di un nuovo prelievo (si suppone noto e valido il codice del contocorrente sul quale viene effettuato il prelievo);

Operazione2) Inserimento di un nuovo trasferimento (si suppongono noti e validi i codici dei due contocorrenti che interessano il trasferimento);

Operazione3) Visualizzare tutti i dati di un contocorrente, anche il SALDO

Tavola dei Volumi

CONCETTO	ACC.	TIPO
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

Risoluzione

Con il dato derivato :

	CONCETTO	ACC	TIPO
Operazione 1 1 accesso in lettura 3 accessi in scrittura $7*100=700$ /giorno	Prelievo	1	S
	In	1	S
	ContoCorrente	1	L
	ContoCorrente	1	S
Operazione 2 2 accessi in lettura 5 accessi in scrittura $12*10=120$ /giorno	Trasferimento	1	S
	Tra	2	S
	ContoCorrente	2	L
	ContoCorrente	2	S
Operazione 3 1 accesso in lettura $1*4 = 4$ /giorno	ContoCorrente	1	L

Totale : 824/giorno

Senza il dato derivato:

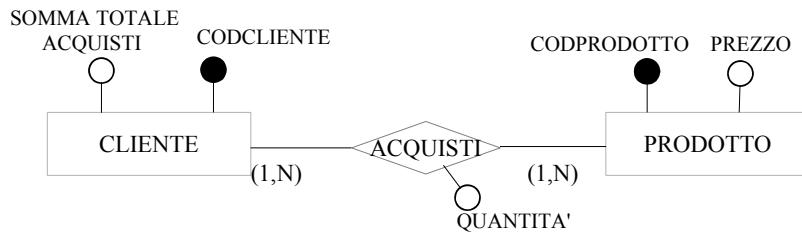
	CONCETTO	ACC	TIPO
Operazione 1 2 accessi in scrittura $4*100=400$ /giorno	Prelievo	1	S
	In	1	S
Operazione 2 3 accessi in scrittura $6*10 = 60$ /giorno	Trasferimento	1	S
	Tra	2	S
Operazione 3 101 accessi in lettura $101*4 = 404$ /giorno	ContoCorrente	1	L
	In	30	L
	Prelievo	30	L
	Tra	20	L
	Trasferimento	20	L

Totale : 864/giorno

Conclusion: Conviene tenere il dato derivato.

Calcolo computazionale 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 in memoria di tale dato.



Operazione1) Dati i codici di un cliente e di un prodotto già esistenti, inserire l'acquisto del prodotto da parte del cliente;

Operazione2) Visualizzare i dati di un cliente;

Operazione3) Modificare il prezzo di un prodotto.

Tavola dei Volumi

CONCETTO	ACC.	TIPO
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

Risoluzione

Con il dato derivato :

	CONCETTO	ACC	TIPO
Operazione 1 2 accessi in lettura 2 accessi in scrittura $6*400 = 2400$ /giorno	Cliente Acquisti Prodotto Cliente	1 1 1 1	L S L S
Operazione 2 1 accesso in lettura $1*10 = 10$ /giorno	Cliente	1	L
Operazione 3 41 accessi in lettura 21 accessi in scrittura $83*1 = 83$ /giorno	Prodotto Prodotto Acquisti Cliente Cliente	1 1 20 20 20	L S L L S

Totale: 2493/giorno

Senza il dato derivato:

	CONCETTO	ACC	TIPO
Operazione 1 1 accesso in scrittura $2*400 = 400$ /giorno	Acquisti	1	S
Operazione 2 81 accessi in lettura $81*10 = 810$ /giorno	Cliente Acquisti Prodotto	40 40	L L L
Operazione 3 1 accesso in lettura 1 accesso in scrittura $3*1 = 3$ /giorno	Prodotto Prodotto	1 1	L S

Totale: 1213/giorno

Conclusione: Conviene eliminare il dato derivato.

Normalizzazione 1

15/06/2017

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 un schema E/R che descriva lo schema di relazione e soddisfi le dipendenze funzionali date.

Risoluzione:

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 1 e FD2
3. $AC \rightarrow ADB$ applicazione dell'estensione alla 2 con A
4. $AC \rightarrow ADBCE$ applicazione dell'unione 3 e FD1

Quindi AC è superchiave; è facile verificare che è anche chiave.

Lo schema di relazione non è 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.

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 FD1. La chiave di R2 è AB, quindi R2 non è in 2NF e viene pertanto scomposto in R21(C,D) e R22(A,C,E). I sottoschemi ottenuti sono in BCNF

Normalizzazione 2
15/06/2017

Dato il seguente schema di relazione:

Magazzino(Locale, Prodotto, Stanza, Scaffale)

e considerando i seguenti vincoli:

- Un prodotto è immagazzinato in uno e uno 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, BCNF.
4. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

Esempi 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

Risoluzione:

1. Dato che un prodotto è immagazzinato in un ed un solo locale, è valida la (FD1) Prodotto $\rightarrow\!\!\!>$ Locale;
Inoltre, dato che in una stanza di un locale, uno scaffale immagazzina un preciso prodotto, si ha che (FD2) Locale Stanza Scaffale $\rightarrow\!\!\!>$ Prodotto.
2. Le chiavi dello schema di relazione sono K1=Locale Stanza Scaffale e K2=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.

Normalizzazione 3
15/06/2017

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 2NF, 3FN, 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

Risoluzione:

1. Un capitano gioca in una precisa squadra di una precisa categoria:
(FD1) Capitano $\rightarrow\!\!\!>$ Squadra e (FD2) Capitano $\rightarrow\!\!\!>$ Categoria;
per ogni squadra e categoria c'è un solo capitano:
(FD3) Squadra Categoria $\rightarrow\!\!\!>$ Capitano;
per ogni torneo, una squadra partecipa con una sola categoria:
(FD4) Squadra Torneo $\rightarrow\!\!\!>$ 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 proiezioni 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.

Normalizzazione 4

Dato il seguente schema di relazione:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali :

- (FD1) A → B
- (FD2) BC → D
- (FD3) A → 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;

Risoluzione:

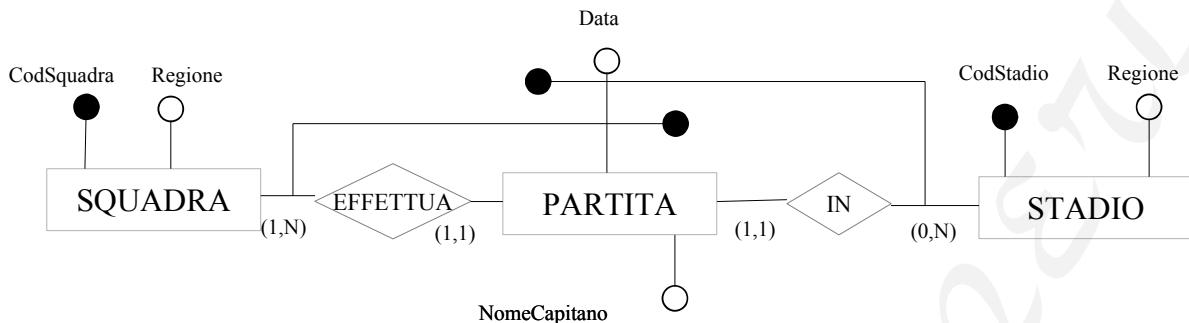
1. La chiave dello schema di relazione sono K₁=A
2. Lo schema di relazione è in 2NF ma non è in 3NF a causa della FD2.
3. Siccome lo schema non è in 3NF a causa della FD2, consideriamo la decomposizione :

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

Tale decomposizione è lossless preserva le dipendenze funzionali e genera sottoschemi che risultano essere in BCNF.

Normalizzazione 5

Dato il seguente schema E/R:



Si consideri la seguente traduzione in schema relazionale:

$\text{Squadra}(\text{CodSquadra}, \text{Regione})$
 $\text{Stadio}(\text{CodStadio}, \text{Città})$
 $\text{Partita}(\text{Data}, \text{CodSquadra}, \text{NomeCapitano}, \text{CodStadio})$

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) $\text{NomeCapitano} \rightarrow \text{CodSquadra}$.

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 aggiuntiva;
2. Determinare se lo schema di relazione Partita è in 2NF , 3NF o 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.

Risoluzione

1. Chiavi: K1= (Data, CodSquadra)
 $K2 = (\text{Data}, \text{CodStadio})$
 $K3 = (\text{Data}, \text{NomeCapitano})$.

2. Partita non è in BCNF a causa della FD, però in 3NF perché l'attributo CodSquadra è primo.

3. Siccome lo schema non è in BCNF a causa della FD, si consideri
 $\text{Capitano}(\text{NomeCapitano}, \text{CodSquadra})$
 $\text{Partita1}(\text{Data}, \text{NomeCapitano}, \text{CodStadio})$

Entrambi sono in BCNF, la decomposizione è lossless ma non preserva la dipendenza dovuta alla chiave K1(Data, CodSquadra) \rightarrow (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. Possiamo effettuare il controllo verificando che il risultato della seguente query sia zero:

```

SELECT count(*)
FROM Partita1 P1, Capitano C
WHERE P1.NomeCapitano=C.NomeCapitano
AND C.CodSquadra=Y
AND P1.Data=X
  
```

Normalizzazione 6

Dato il seguente schema di relazione:

R(A,B,C,D)

e considerando le seguenti dipendenze funzionali:

- (FD1) AB → C
- (FD2) AB → D
- (FD3) C → A
- (FD4) D → B

Viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione;
2. Determinare se lo schema di relazione è il 2NF , 3NF e BCNF;
3. Produrre eventuali decomposizioni e discutere la preservazione dei dati e delle dipendenze funzionali.

Risoluzione

1. Le chiavi dello schema di relazione sono K1(AB), K2(BC) , K3(CD), K4(DA).
2. Lo schema di relazione non è in BCNF per la FD3 e per la FD4. E' in 3NF perché gli attributi A e B sono primi.
3. Essendo lo schema non in BCNF per la FD3, consideriamo la decomposizione binaria:

R1(AC) con FD3 e R2(BCD) con FD4

Tale decomposizione è lossless ma non preserva le dipendenze FD1 e FD2. R2 non è in BCNF a causa della FD4, quindi

R21(CD) , R22(BD) con FD4

che risulta essere lossless e preserva le dipendenze funzionali.

Infine non si preservano le dipendenze funzionali FD1 e FD2.

Normalizzazione 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 1); 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 il 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.

Risoluzione

1. Le dipendenza funzionali sono:
 - (FD1) Matricola → NomeStudente
 - (FD2) Professore → Corso
 - (FD3) NomeStudente Corso → Professore
 - (FD4) Matricola Corso → Voto
2. Le chiavi dello schema di relazione sono K1= (Matricola, Professore) e K2 = (Matricola, Corso).
3. Lo schema di relazione non è in 2NF a causa di FD1. Lo schema di relazione non è in BCNF per la FD2 e per la FD3.
4. Essendo lo schema non in 2NF per la FD1:
 - Libretto1 (Matricola, NomeStudente), con FD1
 - Libretto2 (Matricola, Corso, Professore, Voto), con FD2 e FD4

Tale decomposizione è lossless ma non preserva la dipendenza FD3. Libretto2 non è in BCNF a causa della FD2, quindi:

Libretto21 (Professore, Corso), con FD2
Libretto22 (Matricola, Professore, Voto)

che risulta essere lossless, ma non preserva la dipendenza funzionale FD4.

5. Entrambi i sottoschemi risultano essere in BCNF

Normalizzazione 8

Dato il seguente schema di relazione:

Progetto (Anno, CodiceProgetto, CapoProgetto, Reparto, Responsabile)

e considerando le seguenti dipendenze funzionali:

- (FD1) Anno Progetto → CapoProgetto
- (FD2) CapoProgetto → Reparto
- (FD3) Reparto → Responsabile
- (FD4) Anno Progetto → Responsabile

Viene richiesto di:

1. Determinare la chiave o le chiavi dello schema di relazione;
2. Determinare se lo schema di relazione è il 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.

Risoluzione

La dipendenza funzionale FD4 è ridondante rispetto alle altre, quindi può essere trascurata.

1. Le chiave schema di relazione è unica, K1= (Anno, Progetto).
2. Lo schema di relazione non è in 3NF a causa di FD2 e FD3.
3. Essendo lo schema non in 3NF per la FD3:

Progetto1 (Reparto, Responsabile), con FD3

Progetto2 (Anno, CodiceProgetto, CapoProgetto, Reparto), con FD1 e FD2

Tale decomposizione è lossless e preserva le dipendenza. Progetto2 non è in 3NF a causa della FD2, quindi:

Progetto21 (CapoProgetto, Reparto), con FD2

Progetto22 (Anno, CodiceProgetto, CapoProgetto), con FD1

Ne segue la seguente decomposizione lossless e che preserva le dipendenze:

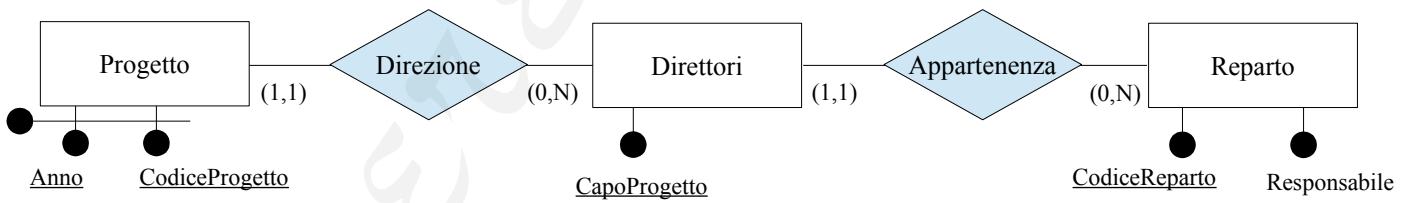
Progetto1 (Reparto, Responsabile)

Progetto21 (CapoProgetto, Reparto)

Progetto22 (Anno, CodiceProgetto, CapoProgetto)

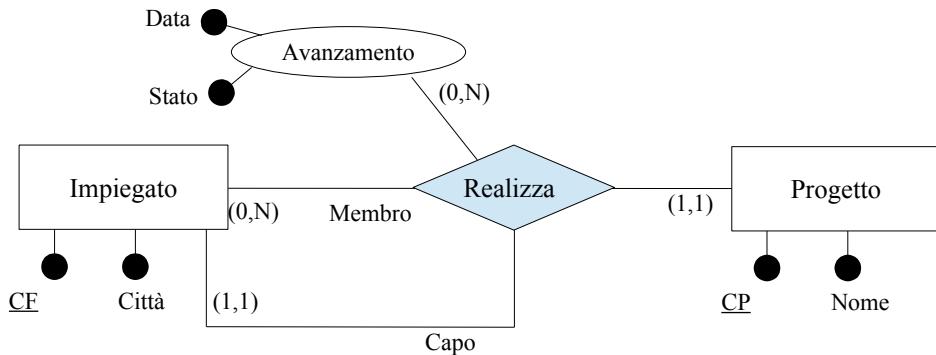
che risulta adesso essere in BCNF.

4. Schema E/R(ridenominiamo l'attributo Reparto come CodiceReparto):



Normalizzazione 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.

Risoluzione

La dipendenza funzionale FD4 è ridondante rispetto alle altre, quindi può essere trascurata.

1. Impiegato (CF, Città)
 Progetto (CP, Nome)
 Realizza (CodP, CFCapo, CFMembro)
 - AK: CFCapo
 - FK: CFCapo REFERENCES Impiegato (CF)
 - FK: CFMembro REFERENCES Impiegato (CF)
 - FK: CodP REFERENCES Progetto (CP)
 Avanzamento (Ref CP, Data, Stato)
 - FK: Ref_CP REFERENCES Realizza (CodP)
2. Basta modificare la chiave della relazione Avanzamento:
 Avanzamento (CP, Data, Stato)
3. R (CP, Nome, CFCapo, CittàCapo, CFMembro, CittàMembro, Data, Stato)
 - (a) Dipendenze funzionali:
 - (FD1) CFCapo → CP
 - (FD2) CP → CFCapo
 - (FD3) CFCapo → CittàCapo
 - (FD4) CFMembro → CittàMembro
 - (FD5) CP → Nome
 - (FD6) CFCapo → CFMembro
 - (FD7) CP → CFMembro
 - (b) Le chiavi dello schema di relazione R sono K1 = (CFCapo, Data, Stato) e K2 = (CP, Data, Stato).
 - (c) Lo schema non è in 2NK perché gli attributi primi dipendono dalle chiavi. Consideriamo la decomposizione:

R1 (CP, Data, Stato, CFCapo)

R2 (CP, Nome, CFMembro)

R3 (CFCapo, CittàCapo)

R4 (CFMembro, CittàMembro)

Tutti sono in 3NF, tranne R1 che non è in BCNF a causa di FD1 e FD2.

R11 (CP, Data, Stato)

R12 (CP, CFCapo)

Questa decomposizione è lossless e preserva tutte le dipendenze, quindi lo schema finale è in BCNF.