Eserciziario SQL

Costantino, Luca, Santoro, Marchese

29 aprile 2010

Indice

Testi

Esercizio Esame 'Terremoto'

Utilizzando le relazioni Comune e Terremoto, risolvete le seguenti query SQL

Comune(Nome,Abitanti,Regione)
Terremoto(Comune,Danni,Scala,Data)

Stiamo assumendo che non esistano due comuni con lo stesso nome

1. Trovare il paese maggiormente popolato che ha subito un terremoto di scala almeno 5
2. Selezionare i comuni che hanno subito danni totali per più di un milione di euro in terremoti a partire dal 31 dicembre 1980.

Esercizio Esame 'Attori'

Utilizzando le relazioni Comune e Terremoto, risolvete le seguenti query SQL

Attore(codAttore,cognome,nome,dataNascita)
Spettacolo(codSpettacolo,nomeSpettacolo, durata, genere, regista, anno)
Partecipazione(codAtt,codSpett,ruolo)

1. Visualizzare gli spettacoli di genere commedia ai quali ha partecipato l' attore 'Verdi Giovanni' (codSpett,nomeSpettacolo)

2. Visualizzare, per ciascun genere, la durata media degli spettacoli (genere, durataMedia)

3. Visualizzare il numero di attori che hanno partecipato a più di 5 spettacoli (numAttori)

Esercizio Esame 'Impiegati'

Utilizzando le relazioni Impiegato, Ordine e Cliente, risolvete le seguenti query SQL

Impiegato (IDimpiegato, Cognome, Nome, Posizione, Indirizzo, Citta', Superiore)
Ordine (IdOrdine, IdCliente, IDImp, Data)
Cliente (IdCliente, Nome, Indirizzo, Citta', Nazione)

1. Selezionare gli impiegati che occupano la posizione di direttore commerciale che hanno almeno tre rappresentanti alle loro dipendenze

2. Selezionare i rappresentanti che hanno ricevuto ordini solo da clienti della loro citta'

3. Visualizzare, ID, cognome e nome dell' IMPIEGATO che ha ricevuto il maggior numero di ordini nel 1999

Esercizio Esame 'Gennaio09'

Utilizzando le relazioni Studente, Corso e Esame, risolvete le seguenti query SQL

Studente (matricola, cognome, nome, luogoNascita)
Corso (codCorso, nomeCorso, numeroCrediti)
Esame (codCorso, matr, anno, voto)

1. Visualizzare, per ogni studente, il numero degli esami superati nel 2005 e la media dei voti ottenuti, visualizzando anche gli studenti che non hanno superato esami nel 2005 (matricola, numeroEsami, votoMedio).

2. Visualizzare gli studenti (Matricola, cognome, nome) per i quali il voto medio degli esami sostenuti nel 2007 è superiore al voto medio degli esami sostenuti nel 2006.

3. Visualizzare la matricola degli studenti che hanno superato almeno un esame nel 2007 e nessuno nel 2006.

Esercizio Esame 'Novembre09'

Dato il seguente schema relazionale:

SQUADRA (codSquadra, nomeS, dataFondazione)
GIOCATORE (codGiocatore, nomeG, cognomeG, dataNascita, nazionalità)
ALLENATORE(codAllenatore, nomeA, cognomeA, data Nascita, nazionalità)
GIOCA (codSquadra, codGiocatore, anno, partiteGiocate, golSegnati)
ALLENA (codSquadra, codAllenatore, anno)

1. Scrivere la query SQL che visualizza gli allenatori (codAllenatore, nome, cognome, nazionalità) stranieri che hanno allenato la Juventus negli ultimi 20 anni.

2. Scrivere la query SQL che visualizza i giocatori (codGiocatore, nome, cognome, partiteGiocate, mediaGol) che nell' anno 2008 hanno giocato almeno 10 partite con una media gol maggiore o uguale a 0,3. Ordinare il risultato per media gol.

3. Scrivere la query SQL che visualizza le squadre (codSquadra, nomeS, nomeA, cognomeA) che hanno avuto un solo allenatore nell' anno 2005.

4. Scrivere la query SQL che visualizza le squadre (codSquadra, nome) in cui hanno militato Gabriel Batistuta o Abel Balbo ma non Daniel Fonseca.

Soluzioni

Soluzione Esame 'Terremoto'

Istanza di esempio

Terremoto

Comune	Danni	Scala	Data
Messina	100000	7,2	28/12/1972
Ancona	300000	5,4	25/01/1972
Livorno	100000	5,7	24/04/1984
Augusta	250000	5,1	13/12/1990
Patti	500000	5,2	14/02/1999
San Giuliano di Puglia	32000	5,4	31/10/2002
L' Aquila	1200000	6,3	06/04/2009

Comune

Nome	Abitanti	Regione
Messina	150000	Sicilia
Palermo	300000	Sicilia
Catania	120000	Sicilia
L'Aquila	50000	Abruzzo
Foggia	32000	Puglia
Patti	12000	Messina
San Giuliano di Puglia	2000	Puglia
Ancona	30000	Marche
Livorno	25000	Toscana
Augusta	1200	Sicilia
Torino	400000	Piemonte
Milano	600000	Lombardia
Pescara	30000	Abruzzo

1. Trovare il paese maggiormente popolato che ha subito un terremoto di scala almeno 5

   SELECT Nome,Regione
   FROM Comune JOIN Terremoto ON Nome=Comune
   WHERE Abitanti = (SELECT MAX(Abitanti) FROM Terremoto JOIN Comune ON (Comune = Nome)) AND scala >= 5;

Nome	Regione
Messina	Sicilia

2. Selezionare i comuni che hanno subito danni totali per piu' di un milione di euro in terremoti a partire dal 31 dicembre 1980.

   SELECT Nome, Regione
   FROM Terremoto JOIN Comune ON Comune = Nome
   WHERE data $\geq$ '31-12-1980'
   GROUP BY Nome, Regione
   HAVING SUM(Danni) > '1000000'

Nome	Regione

Soluzione Esame 'Attori'

Istanza di esempio

Attore

codAttore	cognome	nome	dataNascita
1	Verdi	Giovanni	12-09-1969
2	Verdi	Marco	15-01-1952
3	Rossi	Mario	23-05-1972
4	Brown	Luigi	12-04-1961
5	Mancuso	Francesco	12-05-1975
6	Gassman	Marcello	31-12-1980
7	Fiorani	Silvio	15-03-1976
8	Salvatore	Zoccolo	26-06-1966

Spettacolo

codSpettacolo	nomeSpettacolo	durata	genere	regista	anno
101	L' Alba dei Pirati	72	azione	Bergman	1968
105	India	240	romantico	Yemen	1956
108	Safari	60	horror	D. Argento	1987
110	Fernet 9/11	30	comico	Bastonliegi	2009
107	Salomè	54	commedia	Bastonliegi	1973
102	Amore a Taiwan	45	commedia	Samperi	1984

Partecipazione

codAtt	codSpett	ruolo
1	108	capitano
3	104	divinità
1	103	pescatore
4	101	spadaccino 1
5	110	comparsa
6	102	ladro
1	107	cameriere
1	102	pirata

1. Visualizzare gli spettacoli di genere commedia ai quali ha partecipato l' attore 'Verdi Giovanni' (codSpett,nomeSpettacolo)

   SELECT codSpett, nomeSpettacolo
   FROM PARTECIPAZIONE JOIN SPETTACOLO ON (codSpett = codSpettacolo)
   WHERE genere='commedia' AND codAttore IN (
   SELECT codAttore
   FROM Attore
   WHERE cognome='Verdi' AND nome='Giovanni');

codAtt	nomeSpettacolo
107	Salomè
102	Amore a Taiwan

2. Visualizzare, per ciascun genere, la durata media degli spettacoli (genere, durataMedia).

SELECT genere, AVG(durata) AS durataMedia
FROM SPETTACOLO
GROUP BY genere

genere	durataMedia
azione	72
comico	30
commedia	49,5
horror	60
romantico	240

3. Visualizzare il numero di attori che hanno partecipato a più di 5 spettacoli (numAttori)
   CREATE VIEW Partecipazione1 (codAttore, codSpett) AS
   SELECT DISTINCT codAtt, codSpett
   FROM Partecipazione

   SELECT COUNT(*) AS numAttori
   FROM Partecipazione1
   GROUP BY codAtt
   HAVING COUNT(*)>5

numAttori

Soluzione Esame 'Impiegati'

Istanza di esempio

Cliente

IdCliente	Nome	Indirizzo	Città	Nazione
1	Mark Belli		Austin	USA
2	Luca Costa		Catania	Italia
3	Sebastian Polanski		Cracovia	Polonia
4	Marco Rossi		Roma	Italia
5	Ignazio Calogero		Catania	Italia
6	Hu Chen		Pechino	Cina
7	Silvio Drana		Firenze	Italia

Impiegato

IDimpiegato	cognome	nome	posizione	indirizzo	città	superiore
1	Rossi	Marco	Direttore Commerciale		Roma	
2	White	Luke			Venezia	1
3	Follini	Sergio	stagista		Firenze	
4	Smith	Federico			Roma	1
5	Costa	Francesco	CEO		Catania	
6	Friulano	Mark			Catania	
7	Sabato	Eugene			Firenze	1
8	Brown	Cleveland			Venezia	1

Ordine

IdOrdine	IdCliente	IdImp	Data
1	1	3	12/04/2008
2	2	3	15/06/2010
3	1	1	06/02/2007
4	2	4	06/04/2005
5	3	3	12/04/2009
6	4	2	14/11/2007

1. Selezionare gli impiegati che occupano la posizione di direttore commerciale che hanno almeno tre rappresentanti alle loro dipendenze

   SELECT *
   from impiegato i
   WHERE i.posizione='Direttore Commerciale' AND i.IDimpiegato IN (SELECT IDimpiegato FROM (SELECT superiore, count(*) AS numdip FROM impiegato GROUP BY superiore) WHERE numdip>2);

IDimpiegato	cognome	nome	posizione	indirizzo	città	superiore
1	Rossi	Marco	Direttore Commerciale		Roma	

2. Selezionare i rappresentanti che hanno ricevuto ordini solo da clienti della loro citta'

   SELECT *
   FROM impiegato
   WHERE IDimpiegato NOT IN (SELECT DISTINCT IDimpiegato FROM impiegato i, ordine o, cliente c WHERE i.IDimpiegato = o.IDimp AND o.Idcliente = c.Idcliente AND i.città <> c.città);

IDimpiegato	cognome	nome	posizione	indirizzo	città	superiore
5	Costa	Francesco	CEO		Catania	
6	Friulano	Mark			Catania	
7	Sabato	Eugene			Firenze	1
8	Brown	Cleveland			Venezia	1

3. Visualizzare, ID, cognome e nome dell' IMPIEGATO che ha ricevuto il maggior numero di ordini nel 1999

   SELECT IDimpiegato, cognome, nome
   FROM impiegato
   WHERE IDimpiegato = (SELECT max(numOrdini) FROM (SELECT COUNT(*) AS numOrdini FROM ordine GROUP BY IDimp));

IDimpiegato	cognome	nome
3	Follini	Sergio

Soluzione Esame 'Gennaio09'

Istanza di esempio

Corso

codCorso	nomeCorso	crediti
301	Formazione Discreta 1	6
303	Programmazione 2	9
315	Analisi Matematica 2	6
406	Fisica 1	6
123	Database	9

Esame

codCorso	matr	anno	voto
301	6670012	2010	28
301	6670345	2010	19
123	6671212	2009	26
406	6668012	2008	23
301	6670013	2010	30
123	6670012	2009	18

Studente

matricola	cognome	nome	luogoNascita
6670012	Rossi	Rick	Pisa
6670012	Rossi	Giuseppe	Taormina
6671212	Brown	Emmett	Austin
6668012	Tyson	Mike	Chicago
6670013	Staffelli	Francesco	Roma
6670054	Slate	Luigi	Roma
6670062	Verdi	Giovanni	Catania

1. Visualizzare, per ogni studente, il numero degli esami superati nel 2005 e la media dei voti ottenuti, visualizzando anche gli studenti che non hanno superato esami nel 2005 (matricola, numeroEsami, votoMedio).

SELECT DISTINCT matricola, numMat, mediaVoto
FROM studente LEFT OUTER JOIN (SELECT matr, count(*) as numMat, AVG(voto) as mediaVoto FROM esame e, studente s WHERE anno = 2005 AND e.matr = s.matricola GROUP BY matr) ON (matr = matricola);

Studente

matricola	numMat	mediaVoto
6668012		
6670012		
6670013		
6670054		
6670062	2	20.5
6671212		

2. Visualizzare gli studenti (Matricola, cognome, nome) per i quali il voto medio degli esami sostenuti nel 2007 è superiore al voto medio degli esami sostenuti nel 2006.

SELECT matricola, cognome, nome
FROM Studente s, Esame e
WHERE s.matricola = e.matr AND anno = 2007
GROUP BY matricola, cognome, nome
HAVING AVG(voto)> (SELECT AVG(voto) FROM Esame e1 WHERE anno = 2006 AND e.matr = e1.matr)

Soluzione con le viste
CREATE VIEW EsamiAnno (matricola, anno, mediaVoti) AS
SELECT matricola, anno, AVG(voto)
FROM Esame
GROUP BY matricola, anno

SELECT matricola, cognome, nome
FROM Studenti S Join EsamiAnno E1 NATURAL JOIN EsamiAnno E2
WHERE E1.anno='2006' AND E2.anno='2007' AND E2.mediaVoti>E1.mediaVoti

3. Visualizzare la matricola degli studenti che hanno superato almeno un esame nel 2007 e nessuno nel 2006.

SELECT DISTINCT matr
FROM esame
WHERE matr NOT IN (SELECT DISTINCT matr FROM esame WHERE anno = 2006) AND matr IN (SELECT DISTINCT matr FROM esame WHERE anno = 2007);

Soluzione Esame 'Novembre09'

(NB. gli schemi delle istanze sono omessi per brevità)

1. Scrivere la query SQL che visualizza gli allenatori (codAllenatore, nome, cognome, nazionalità) stranieri che hanno allenato la Juventus negli ultimi 20 anni.
   SELECT *
   FROM allenatore a
   WHERE nazionalità <> 'Italiana' AND a.codAllenatore IN (SELECT codAllenatore FROM Allena, Squadra WHERE Allena.codSquadra = Squadra.codSquadra AND nomeS='Juventus' AND anno=2008);

2. Scrivere la query SQL che visualizza i giocatori (codGiocatore, nome, cognome, partiteGiocate, mediaGol) che nell' anno 2008 hanno giocato almeno 10 partite con una media gol maggiore o uguale a 0,3. Ordinare il risultato per media gol.

   SELECT *
   FROM (SELECT codGiocatore, nome, cognome, partiteGiocate, AVG(golSegnati) AS mediaGol, count(*) as partiteTotali FROM Giocatore NATURAL JOIN Gioca ON (anno=2008) GROUP BY codGiocatore)
   WHERE mediaGol >= 0,3 AND partiteTotali > 9

3. Scrivere la query SQL che visualizza le squadre (codSquadra, nomeS, nomeA, cognomeA) che hanno avuto un solo allenatore nell' anno 2005.

   SELECT codSquadra, nomeS, nomeA, cognomeA
   FROM (Allenatore a JOIN Allena a1 ON (anno=2005 AND a.codAllenatore = a1.codAllenatore)) NATURAL JOIN Squadra
   GROUP BY codSquadra
   HAVING count(*)=1

4. Scrivere la query SQL che visualizza le squadre (codSquadra, nome) in cui hanno militato Gabriel Batistuta o Abel Balbo ma non Daniel Fonseca.

   SELECT DISTINCT codSquadra, nome
   FROM (Squadra NATURAL JOIN Gioca) NATURAL JOIN Giocatore
   WHERE ((nomeG='Gabriel' AND cognomeG='Batistuta') OR (nome='Abel' AND cognome='Balbo'))
   AND codSquadra NOT IN (SELECT DISTINCT codSqudra FROM Squadra NATURAL JOIN Gioca NATURAL JOIN Giocatore WHERE nome='Daniel' AND cognome='Fonseca')

SQL – Esercizi DML – Blocco 3

Studenti(Matricola, NomeS, CognomeS, CittàRes, Sesso, NumTelefono)
Corsi(CodCorso, NomeC, NumCrediti, MatricolaDocente)
EsamiSuperati(Matricola, CodCorso, Voto)
Docenti(MatricolaD, NomeD, CognomeD, Stipendio, Dipartimento)
Iscritti_Esami (Data, CodCorso, Matricola, Voto)

1) Selezionare per ogni studente matricola e numero di esami superati

```
Select matricola,count(*) as esamisuperati
from esamisuperati
group by matricola;
```

2) Selezionare matricola studente, codice corso e numero di volte che lo studente ha sostenuto l'esame del corso per tutti gli studenti e i relativi corsi.

```
select Matricola, CodCorso, COUNT(*)
from Iscritti_Esami
group by Matricola, CodCorso;
```

3) Selezionare la matricola degli studenti che hanno almeno un corso per cui hanno sostenuto almeno 5 volte l'esame

```
select DISTINCT Matricola
from Iscritti_Esami
group by Matricola, CodCorso
having count(*)>=5;
```

4) Selezionare la matricola degli studenti che hanno preso il voto più alto all'esame identificato dal codice corso R25.

```
select matricola
from EsamiSuperati
where CodCorso='R25'
   and Voto = (select max(voto)
                  from EsamiSuperati
                  where CodCorso='R25');
```

Viste

5) Selezionare la matricola degli studenti che hanno superato il maggior numero d'esami

```
create view StudentiNumeroEsami
AS
Select matricola,count(*) as numesamisuperati
from esamisuperati
group by matricola;
```

```
select matricola
from StudentiNumeroEsami
where numesamisuperati=(select max(numesamisuperati)
                        from StudentiNumeroEsami
                        );
```

6) Selezionare matricola, nome e cognome dei docenti che hanno uno stipendio maggiore rispetto allo stipendio medio dei propri colleghi di dipartimento (docenti dello stesso dipartimento)

```
create view StipendioMedioPerDipartimento

AS
Select dipartimento,avg(stipendio) as StipendioMedio
from docenti
group by dipartimento;

select matricolaD, nomed, cognomeD
from docenti D, StipendioMedioPerDipartimento S
where D.dipartimento=S.dipartimento
   and D.Stipendio>S.StipendioMedio;
```

Query annidate

7) Selezionare la matricola degli studenti che hanno superato l'esame di tutti i corsi possibili (tutti i corsi presenti nella tabella CORSI). Si supponga l'esistenza di almeno un corso (esiste almeno una tupla nell'istanza della tabella CORSI)

```
Select matricola
from esamisuperati
group by matricola
having count(*)=(select count(*)
                 from corsi);
```

Interrogazioni binarie/insiemistiche

```
UNION
```

8) Selezionare la matricola degli studenti che hanno superato l'esame di informatica con 30 o quello di fisica con 25.

   a.
```
Select matricola
from esamisuperati e, corsi c
where e.CodCorso=c.CodCorso
   and NomeC='Informatica' and Voto=30
UNION
Select matricola
from esamisuperati e, corsi c
```

```
where e.CodCorso=c.CodCorso
   and NomeC='Fisica' and Voto=25
```

b.
```
Select matricola
from studenti
where matricola in (Select matricola
                    from esamisuperati e, corsi c
                    where e.CodCorso=c.CodCorso
                       and NomeC='Informatica' and Voto=30)
  or   matricola in (Select matricola
                    from esamisuperati e, corsi c
                    where e.CodCorso=c.CodCorso
                       and NomeC='Fisica' and Voto=25);
```

INTERSECT

9) Selezionare la matricola degli studenti che hanno superato l'esame di informatica con 30 e quello di fisica con 25.

a.
```
Select matricola
from esamisuperati e, corsi c
where e.CodCorso=c.CodCorso
   and NomeC='Informatica' and Voto=30
INTERSECT
Select matricola
from esamisuperati e, corsi c
where e.CodCorso=c.CodCorso
   and NomeC='Fisica' and Voto=25
```

b.
```
Select matricola
from studenti
where matricola in (Select matricola
                    from esamisuperati e, corsi c
                    where e.CodCorso=c.CodCorso
                       and NomeC='Informatica' and Voto=30)
  and   matricola in (Select matricola
                    from esamisuperati e, corsi c
                    where e.CodCorso=c.CodCorso
                       and NomeC='Fisica' and Voto=25);
```

EXCEPT

10) Selezionare la matricola degli studenti che non hanno mai preso 30

a.
```
select matricola
from student
except
select matricola
     from esamisuperati
     where Voto=30;
```

Esercizi tratti da temi d'esame

Si consideri il seguente schema di base di dati che vuole tenere traccia delle informazioni riguardanti i campionati mondiali di atletica.

CAMPIONATO (EdizioneCamp, Anno, DataInizio, DataFine, CittàOspitante, StatoOspitante)
GARA (EdizioneCamp, NumGara, Disciplina, Data, OraInizio)
ATLETA (CodiceAtleta, Nome, Cognome, Sesso, DataNascita, Nazionalità)
CLASSIFICA_GARA (EdizioneCamp, NumGara, CodiceAtleta, Posizione)

Dove il campo Posizione è un intero.

Rispondere alle seguenti interrogazioni utilizzando il linguaggio SQL

1. *Trovare nome e cognome degli atleti italiani che hanno gareggiato in almeno 4 diversi campionati mondiali di atletica, ma che non si sono mai classificati in prima posizione nelle gare cui hanno partecipato.*

2. *Trovare codice, nome e cognome degli atleti che hanno vinto il maggior numero di gare nel campionato mondiale dell'anno 2005.*

Soluzione

1.
```
SELECT Nome, Cognome
FROM ATLETA A, CLASSIFICA_GARA CG
WHERE A.CodiceAtleta=CG.CodiceAtleta
   AND Nazionalità='Italiana'
   AND CodiceAtleta NOT IN ( SELECT CodiceAtleta
                             FROM CLASSIFICA_GARA
                             WHERE Posizione=1
                           )
GROUP BY  A.CodiceAtleta, Nome, Cognome
HAVING COUNT(DISTINCT EdizioneCamp)>=4;
```

2.
```
CREATE VIEW GareVinteAnno2005 (CodiceAtleta, NumGareVinte)
AS
SELECT CodiceAtleta, COUNT(*) AS NumGareVinte
FROM CLASSIFICA_GARA CG, CAMPIONATO C
WHERE CG.EdizioneCamp=C.EdizioneCamp
   AND Anno=2005
   AND Posizione=1
GROUP BY CodiceAtleta;

SELECT A.CodiceAtleta, Nome, Cognome
FROM ATLETA A, GareVinteAnno2005 GV
WHERE A.CodiceAtleta=GV.CodiceAtleta
   AND GV.NumGareVinte= (SELECT MAX(NumGareVinte)
                         FROM GareVinteAnno2005
                        );
```

Università degli Studi di Cagliari
Corso di Laurea Magistrale in Ing. per l'Ambiente e il Territorio

Laboratorio d'Informatica
A.A. 2019/2020
Docente: Giorgio Fumera

Basi di dati – Esercizi

Gli esercizi riportati di seguito riguardano la progettazione concettuale e logica di basi di dati a partire dalla descrizione sintetica dei requisiti applicativi, e la formulazione di interrogazioni. Questi esercizi sono analoghi a quelli che verranno proposti nella prova d'esame.

1 Progettazione di basi di dati

Gli esercizi di questa sezione richiedono la progettazione concettuale e logica di una base di dati a partire dalla descrizione sintetica dei requisiti applicativi. In particolare si richiede:

- la definizione dello schema concettuale Entità-Relazione, indicando chiaramente il significato di ogni suo elemento, e in particolare la motivazione delle cardinalità delle relazioni e il dominio di ciascun attributo (per semplicità nei requisiti applicativi non viene riportata la dimensione massima delle sequenze di caratteri che compongono nomi di persone e di luoghi, indirizzi, ecc.: in questi casi si scelga un valore opportuno);
- la definizione dello schema logico relazionale, indicando in particolare l'origine di ciascuna tabella in riferimento allo schema Entità-Relazione;
- la scrittura dei comandi SQL per la creazione delle tabelle definite nello schema logico.

Si consiglia poi di creare le varie basi di dati con il DBMS PostgreSQL (tramite il comando `CREATE TABLE`), di inserire alcuni dati nelle varie tabelle (tramite l'interfaccia grafica fornita dal programma pgAdmin III, oppure con il comando SQL `INSERT INTO VALUES`), e di eseguire su di esse alcune interrogazioni.

I primi due esercizi corrispondono alle basi di dati *Segreteria studenti* e *Circolo nautico* usate come esempi durante le lezioni.

1. Un certo corso di laurea offre diversi insegnamenti che hanno un nome (per esempio, "Analisi Matematica 1"), un codice univoco di cinque caratteri assegnato dalla segreteria studenti, e un numero di crediti (uno o più). Al corso di laurea possono essere iscritti diversi studenti. Per ogni iscritto si vogliono memorizzare nome, cognome, codice fiscale (una sequenza di 16 caratteri), data di nascita e numero di matricola (una sequenza di cinque caratteri che identifica univocamente ogni studente, assegnata dalla segreteria studenti). Per ogni esame superato da ogni studente si vogliono conoscere il voto (un intero tra 18 e 30, oppure 30 e lode) e la data; un esame superato non può essere ripetuto.

2. Un certo circolo nautico ha diversi soci e possiede diverse barche che possono essere prenotate dai soci. Ogni velista socio del circolo ha un nome, un'età (un numero frazionario) e un livello di esperienza rappresentato da un numero intero tra 1 (esperienza minima) e 10 (esperienza massima); ogni socio deve essere maggiorenne (età minima 18 anni) ed è identificato univocamente da un numero di tessera (un intero) assegnato dalla direzione del circolo. Ogni barca ha un codice identificativo (un numero intero), un nome e un colore. Ogni velista può usare le barche messe a disposizione dal circolo nautico: ogni barca può essere prenotata per un'intera giornata da un solo velista, ma non può essere prenotata più volte in uno stesso giorno, neanche da parte di velisti diversi.

3. Si consideri ancora l'esercizio 1, e si modifichi il progetto della base di dati assumendo che si voglia memorizzare in essa due informazioni aggiuntive: il piano di studi personale di ciascuno studente (cioè l'insieme degli esami che devono essere sostenuti), e gli esami non superati (oltre a quelli

superati). Per gli esami non superati la valutazione non è costituita da un voto numerico, ma dal giudizio "insufficiente". Si assuma che uno studente non possa sostenere nello stesso giorno più esami per lo stesso insegnamento.

4. Si vuole progettare una base di dati per memorizzare le seguenti informazioni su ciascuno degli studenti iscritti ai corsi di studi di una certa facoltà universitaria istituita nel 1995: nome, cognome, data di nascita, codice fiscale, numero di matricola (una sequenza di nove caratteri che identifica univocamente ogni studente), e corso di studi al quale lo studente è iscritto; di ogni corso si vogliono memorizzare in particolare il nome, il livello (triennale, magistrale, a ciclo unico) la durata (numero di anni, compreso tra 3 e 6) e l'anno di istituzione. Ogni studente è iscritto a uno e un solo corso di studi, mentre in un certo istante un corso di studi può non avere iscritti; non possono esistere due corsi di studi dello stesso livello con lo stesso nome. L'anno di istituzione di un corso di studi non può essere precedente a quello della facoltà.

5. Si vuole progettare una base di dati per memorizzare le seguenti informazioni su ciascuno degli studenti iscritti agli atenei italiani: nome, cognome, data di nascita, numero di matricola (una sequenza di nove caratteri che identifica univocamente ogni studente all'interno di un dato ateneo), e ateneo al quale lo studente è iscritto; di ogni ateneo si vogliono memorizzare in particolare il nome (per esempio, "Università degli Studi di Cagliari"), la città in cui ha sede e l'anno di istituzione (si assume che l'ateneo italiano più antico sia quello di Bologna, la cui data di fondazione è per convenzione il 1088). Ogni studente è iscritto a uno e un solo ateneo, mentre in un certo istante un ateneo può non avere iscritti; non possono esistere due atenei con lo stesso nome.

6. Una certa banca ha diverse filiali distribuite nel territorio. Ciascuna filiale ha un indirizzo (città, via, numero civico e CAP, quest'ultimo formato da cinque cifre) e un certo numero di clienti titolari di conti correnti. In una stessa città possono esserci più filiali, ma in questo caso ognuna ha un indirizzo diverso. Ogni cliente ha nome, cognome, codice fiscale e luogo di residenza (città, via, numero civico e CAP), ed è titolare di uno o più conti correnti, in una o più filiali. Ogni conto corrente è caratterizzato da un codice IBAN (consistente in Italia in una sequenza di 27 caratteri) che lo distingue univocamente, una data di apertura e un saldo (in Euro), e può avere un solo titolare.

7. Una certa casa discografica fondata nel 1970 produce dischi aventi un titolo, un codice identificativo univoco (tredici caratteri), e un anno d'incisione. Ogni disco contiene più brani, ognuno dei quali ha un titolo e una durata (in minuti e secondi); uno stesso disco non può contenere diversi brani con lo stesso titolo. Ogni disco può essere inciso da uno o più autori, ognuno dei quali ha un nome, un cognome, una data di nascita e un nome d'arte; si assuma che non esistano autori diversi con lo stesso nome d'arte.

8. Un'azienda ha diversi dipendenti ed è suddivisa in diversi dipartimenti. Ogni dipartimento ha un nome (diverso dagli altri) e può avere una o più sedi con un loro indirizzo (città, via, numero civico e CAP); ogni sede di uno stesso dipartimento si trova in una diversa città. Ciascuna sede di ogni dipartimento è diretta da uno dei dipendenti dell'azienda, a partire da una certa data; uno stesso dipendente può dirigere al più una sede di dipartimento. L'azienda gestisce inoltre un insieme di progetti: ogni progetto ha un nome e un codice numerico univoco assegnati dall'azienda, e viene coordinato da una delle sedi di un singolo dipartimento; su ogni progetto lavorano uno o più dipendenti, e ogni sede di ogni dipartimento gestisce almeno un progetto. Ogni dipendente ha nome, cognome, sesso, data di nascita, codice fiscale, indirizzo di residenza (città, via, numero civico e CAP), uno stipendio annuale (espresso in Euro), e afferisce a uno e un solo dipartimento. Ogni dipendente, eccetto i direttori dei dipartimenti, ha un supervisore (un altro dipendente); uno stesso dipendente può essere supervisore di più dipendenti. Ogni dipendente lavora su uno o più progetti, a ciascuno dei quali dedica un certo numero di ore settimanali.

2 Formulazione di interrogazioni

Le interrogazioni riportate in questa sezione sono riferite alle basi di dati degli esercizi 1 (*Segreteria studenti*) e 2 (*Circolo nautico*) della sezione 1. Si suggerisce di eseguire le interrogazioni sul DBMS PostgreSQL su istanze opportune delle due basi di dati, come quelle disponibili nel sito web del laboratorio.

1. Con riferimento alla base di dati dell'esercizio 1 della sezione 1 (*Segreteria studenti*), formulare e scrivere in linguaggio SQL le seguenti interrogazioni:

   (a) Trovare nomi e cognomi degli studenti nati nel 1991

   (b) Trovare le matricole degli studenti che hanno conseguito la votazione di 30 in qualche esame (estrarre una sola occorrenza per ciascuno di tali numeri di matricola)

   (c) Trovare i cognomi degli studenti che hanno sostenuto l'esame avente codice `AMI01`

   (d) Trovare i cognomi degli studenti che hanno sostenuto un esame denominato "Analisi I"

   (e) Trovare i cognomi degli studenti che hanno sostenuto l'esame `AMI01`, mediante interrogazioni nidificate

   (f) Trovare i nomi degli esami in cui qualche studente abbia conseguito la votazione di 30, mediante interrogazioni nidificate

   (g) Trovare i cognomi degli studenti più giovani, mediante operatori insiemistici e interrogazioni nidificate

   (h) Trovare le matricole degli studenti che non hanno ottenuto il voto più basso tra quelli conseguiti negli esami dell'insegnamento `AMI01`, mediante operatori insiemistici e interrogazioni nidificate (in altre parole, si è interessati agli studenti che hanno ottenuto nell'esame `AMI01` un voto maggiore rispetto a quello conseguito da qualche altro studente nello stesso esame)

   (i) Trovare i cognomi degli studenti più giovani, mediante operatori di aggregazione

   (j) Trovare le matricole degli studenti che non hanno ottenuto il voto più basso tra quelli conseguiti negli esami dell'insegnamento `AMI01`, mediante operatori di aggregazione

   (k) Trovare le matricole degli studenti che hanno ottenuto il voto più alto nell'esame `AMI01`, usando interrogazioni nidificate e: (1) operatori insiemistici; (2) operatori di aggregazione

2. Con riferimento alla base di dati dell'esercizio 2 della sezione 1 (*Circolo nautico*), formulare e scrivere in linguaggio SQL le seguenti interrogazioni:

   (a) Trovare il codice e il nome di tutti i velisti

   (b) Estrarre i dati di tutte le barche

   (c) Trovare i nomi dei velisti con più di 40 anni

   (d) Trovare i nomi dei velisti con più di 40 anni, oppure con esperienza maggiore di 7

   (e) Trovare i codici delle barche prenotate l'11 dicembre 2018

   (f) Trovare i colori di tutte le barche (estrarre una sola occorrenza di ciascun colore)

   (g) Trovare i nomi di tutte le barche che non siano rosse

   (h) Trovare i nomi dei velisti che hanno prenotato la barca numero 103

   (i) Trovare i colori delle barche prenotate da qualche velista di nome Rusty

   (j) Trovare i codici dei velisti che hanno prenotato qualche barca blu, oppure qualche barca verde

   (k) Trovare i codici dei velisti che hanno prenotato sia qualche barca blu che qualche barca verde

   (l) Trovare i codici dei velisti che hanno prenotato qualche barca blu, ma non barche verdi

   (m) Trovare i codici delle barche che siano state prenotate da qualche velista con esperienza pari almeno a 8, e da nessun velista con età maggiore di 35 anni

   (n) Trovare i nomi dei velisti che hanno prenotato la barca numero 103, mediante interrogazioni nidificate

   (o) Trovare i nomi dei velisti che **non** hanno prenotato la barca numero 103, mediante interrogazioni nidificate

   (p) Estrarre tutti i dati dei velisti la cui esperienza sia maggiore di quella di qualche velista di nome Lubber, mediante operatori insiemistici e interrogazioni nidificate

   (q) Trovare i codici dei velisti più esperti, mediante operatori insiemistici e interrogazioni nidificate

   (r) Trovare nome ed età dei velisti più anziani, mediante operatori insiemistici e interrogazioni nidificate

(s) Trovare il numero dei velisti soci del circolo nautico

(t) Trovare l'età media di tutti i velisti

(u) Trovare il numero di valori distinti di esperienza dei velisti

(v) Trovare l'età media dei velisti con esperienza pari a 10

(w) Trovare nome ed età dei velisti più anziani, mediante interrogazioni nidificate scritte usando: (1) operatori insiemistici; (2) operatori di aggregazione

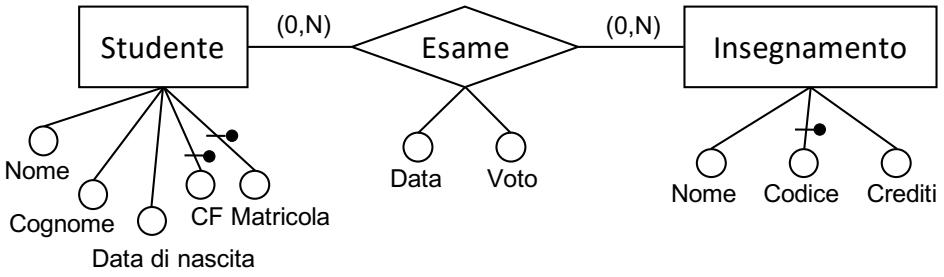
(x) Trovare i nomi dei velisti più esperti, usando operatori di aggregazione

Soluzioni

1 Progettazione di basi di dati

La descrizione degli schemi concettuale e logico viene riportata per esteso solo per il primo esercizio. Per tutti gli altri esercizi vengono riportati solo i dettagli più rilevanti.

1. Lo **schema Entità-Relazione** è il seguente:



Le entità `Studente` e `Insegnamento` rappresentano rispettivamente l'insieme di studenti iscritti al corso di studi e l'insieme di insegnamenti attivi, in un certo istante di tempo. I domini degli attributi sono: un intero positivo per il numero di crediti di un insegnamento, una data per la data di nascita di ciascuno studente, e una stringa di un numero opportuno di caratteri per tutti gli altri attributi (in particolare, 16 caratteri per il codice fiscale e 5 per il numero di matricola degli studenti e il codice degli insegnamenti).

Dai requisiti applicativi si evince che il numero di matricola è un identificatore dell'entità `Studente`, e il codice dell'insegnamento è un identificatore di `Insegnamento`. L'entità `Studente` ha anche un altro identificatore, il codice fiscale, poiché non possono esistere due persone con lo stesso codice fiscale (vincolo generale, e per questo non riportato esplicitamente nei requisiti applicativi).

Le informazioni sugli esami fanno riferimento a entrambe le entità, dato che ogni esame è sostenuto da un certo studente per un certo insegnamento (in una certa data e con un certo voto). Inoltre, poiché si vogliono memorizzare nella base di dati solo gli esami superati, e nessuno studente può ripetere l'esame di uno stesso insegnamento (se già sostenuto con esito positivo), le informazioni sugli esami possono essere rappresentate come una relazione (`Esame`) tra le due entità `Studente` e `Insegnamento`: un dato studente è in relazione con un dato insegnamento se ne ha superato l'esame. La data e il voto dell'esame saranno quindi attributi della relazione `Esame`; il loro dominio sarà rispettivamente un intero tra 18 e 30 oppure "30 e lode", e una data.

La cardinalità della relazione `Esame` è $(0, N)$ rispetto a entrambe le entità coinvolte, poiché in un certo istante uno studente può aver superato nessuno, uno o più esami, e l'esame di un certo insegnamento può essere stato superato da nessuno, uno o più studenti.

In base alle regole per la definizione dello **schema logico relazionale** a partire dallo schema Entità-Relazione, e in particolare tenendo conto che la relazione `Esame` è di tipo molti-a-molti, si ricava facilmente che lo schema logico sarà composto da tre tabelle corrispondenti alle due entità e alla relazione; a tali tabelle si possono assegnare nomi corrispondenti a quelli dell'entità o relazione da cui derivano (per convenzione il nome delle tabelle è al plurale): `Studenti`, `Insegnamenti`, `Esami`.

Come chiave primaria della tabella `Studenti`, tra le colonne `CF` e `Matricola` (corrispondenti ai due attributi identificatori dell'entità `Studente`) si può scegliere la seconda, tenendo conto del ruolo del numero di matricola in un corso di studi. È allora opportuno stabilire un vincolo di unicità sulla colonna `CF`. La chiave primaria di `Insegnamenti` corrisponderà invece all'unico identificatore dell'entità `Insegnamento`, cioè il codice dell'esame.

La tabella `Esami` conterrà, oltre alla data e al voto di un esame, anche le colonne corrispondenti alle chiavi primarie di `Studenti` e `Insegnamenti`, che ne costituiranno anche la chiave primaria (non possono esistere più occorrenze della relazione tra uno stesso studente e uno stesso insegnamento, corrispondenti a esami superati). Ciascuna di tali colonne dovrà avere lo stesso dominio della corrispondente colonna di `Studenti` e `Insegnamenti`, e un vincolo d'integrità referenziale con essa.

Riguardo ai domini delle colonne di ciascuna tabella, per le stringhe di caratteri si è scelta una lunghezza ragionevole quando non specificata dai requisiti applicativi. Per il numero di crediti di un insegnamento, al vincolo di dominio (numero intero) si è aggiunto un vincolo di dominio esteso (il valore deve essere positivo). Nel caso del voto si è scelto di usare due colonne: una per il valore numerico (anche in questo caso al vincolo di dominio si è aggiunto un vincolo di dominio esteso, per garantire che il valore sia compreso tra 18 e 30) e una per la presenza o assenza della lode; per quest'ultima si è scelto il dominio dei valori logici (booleani) "vero" e "falso". Queste scelte comportano la necessità di un vincolo di *tupla* che coinvolga le colonne del voto e della lode, per garantire che la lode non possa essere presente se il voto è inferiore a 30.

Lo schema logico si può descrivere in modo informale come segue (la chiave primaria di ciascuna tabella corrisponde alle colonne il cui nome è sottolineato):

`Studenti` (Matricola: stringa di 5 caratteri, CF: stringa di 16 caratteri, Cognome: stringa di 50 caratteri, Nome: stringa di 50 caratteri, Data di nascita: data)

`Insegnamenti` (Nome: stringa di 50 caratteri, Codice: stringa di 5 caratteri, Crediti: intero)

`Esami` (Data: data, Voto: intero, Lode: booleano, Studente: stringa di 5 caratteri, Esame: stringa di 5 caratteri)

Vincoli d'integrità referenziale (chiavi esterne):

- Studente (tabella Esami) verso Matricola (tabella Studenti)
- Esame (tabella Esami) verso Codice (tabella Insegnamento)

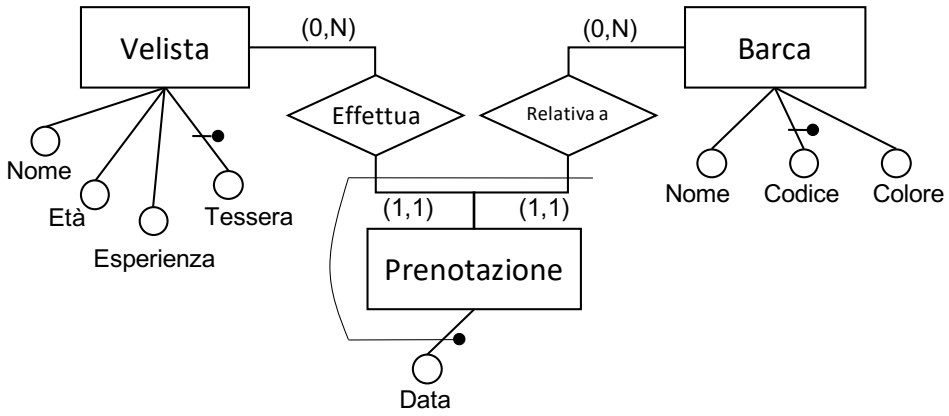
Vincoli di dominio esteso:

- il numero di crediti di ogni insegnamento deve essere positivo;
- nella tabella Esami il voto deve essere compreso tra 18 e 30 (estremi inclusi).

Vincolo di tupla: nella tabella Esami il voto non può essere minore di 30 se è presente la lode.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_segreteria_studenti.sql`.

2. Lo **schema Entità-Relazione** è il seguente:



Le informazioni sulle prenotazioni non possono essere rappresentate come una relazione tra le entità `Velista` e `Barca`, poiché uno stesso velista può prenotare più volte (in giorni diversi) una stessa barca: tale informazione non costituisce quindi un'associazione *binaria* tra velisti e barche. In altre parole, l'informazione che si richiede di rappresentare non consiste nel fatto che una certa barca sia stata prenotata o meno da un certo velista (informazione a due soli valori, "vero" e "falso"): essa consite invece in ogni istanza di una prenotazione di una barca da parte di un velista. Tali informazioni devono quindi essere rappresentate da un'entità (`Prenotazione`), che indica l'insieme di tutte le prenotazioni effettuate; più precisamente, ogni elemento di tale entità rappresenta una prenotazione eseguita da un certo velista in una certa data per una certa barca. Si noti che `Prenotazione` è un'entità debole, dato che il suo unico attributo (la data della prenotazione) non è ovviamente in grado di distinguere univocamente i suoi elementi. A questo scopo è necessario

affiancare alla data della prenotazione sia il velista che l'ha effettuata che la barca coinvolta, tramite gli identificatori delle entità corrispondenti.

Per quanto detto sopra sull'entità `Prenotazione`, le due relazioni hanno cardinalità $(1,1)$ rispetto a tale entità; la loro cardinalità rispetto a entrambe le entità `Velista` e `Barca` è invece $(0,N)$, dato che in un certo istante di tempo un velista potrebbe aver effettuato nessuna, una o più prenotazioni, e una barca potrebbe essere stata prenotata nessuna, una o più volte.

Dato che non sono presenti relazioni di tipo molti-a-molti, lo schema logico relazionale conterrà solo le tabelle corrispondenti alle tre entità.

Lo schema logico relazionale è il seguente (i nomi delle colonne sono quelli della base di dati usata come esempio a lezione):

`Velisti` (vid: intero, età: reale, esperienza: intero, vnome: stringa di 20 caratteri)

`Barche` (nome: stringa di 25 caratteri, bid: intero, colore: stringa di 10 caratteri)

`Prenotazioni` (data: data, vid: intero, bid: intero)

Vincoli d'integrità referenziale (chiavi esterne):

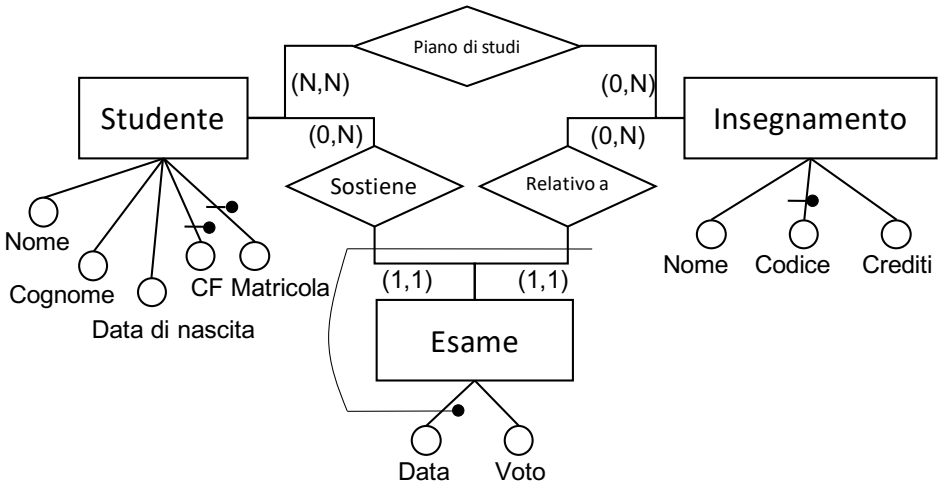
- vid (tabella Prenotazioni) verso vid (tabella Velisti)
- bid (tabella Prenotazioni) verso bid (tabella Barche)

Vincoli di dominio esteso:

- l'età di ogni velista deve essere non inferiore a 18;
- il numero di tessera di ogni velista deve essere positivo;
- l'esperienza di ogni velista deve essere compresa tra 1 e 10 (estremi inclusi);
- il codice di ogni barca deve essere positivo.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_circolo_nautico.sql`.

3. Lo **schema Entità-Relazione** è il seguente:

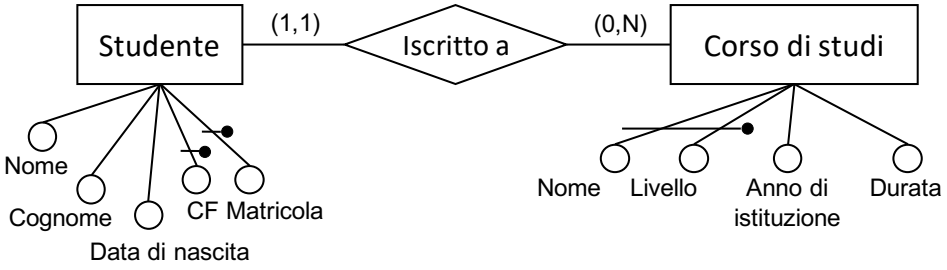


Le informazioni sul piano di studi possono essere rappresentate come una relazione tra le entità `Studente` e `Insegnamento`, dato che sono di natura binaria (un certo insegnamento può trovarsi o meno nel piano di studi di un certo studente). Quelle sugli esami sostenuti non possono invece essere rappresentate da una relazione tra le stesse entità, poiché in questo caso può esserci più di un'associazione tra uno stesso studente e uno stesso insegnamento (nel caso in cui uno stesso esame venga ripetuto più volte da uno stesso studente). È necessario dunque introdurre l'entità `Esame` (analoga all'entità `Prenotazione` dell'esercizio precedente): ogni suo elemento corrisponde a una prova d'esame per un certo insegnamento sostenuta da un certo studente in un certa data con un certo esito.

L'unica relazione molti-a-molti è `Piano di studi`: lo schema logico relazionale conterrà quindi quattro tabelle corrispondenti a tale relazione e alle tre entità.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_segreteria_studenti_2.sql`.

4. Lo **schema Entità-Relazione** è il seguente:

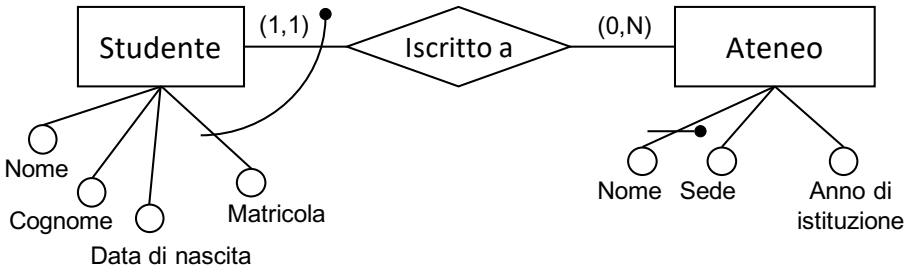


Si noti che l'identificatore dell'entità `Corso di studi` è composto dalla coppia di attributi `Nome` e `Livello`, poiché il solo nome di un corso non è sufficiente per distinguerlo univocamente dagli altri (si pensi ai corsi di studi triennale e magistrale in Ingegneria per l'Ambiente e il Territorio).

Lo schema logico relazionale comprenderà solo le tabelle associate alle due entità, dato che la relazione `Iscritto a` non è di tipo molti-a-molti.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_facolta.sql`.

5. Lo **schema Entità-Relazione** è il seguente:

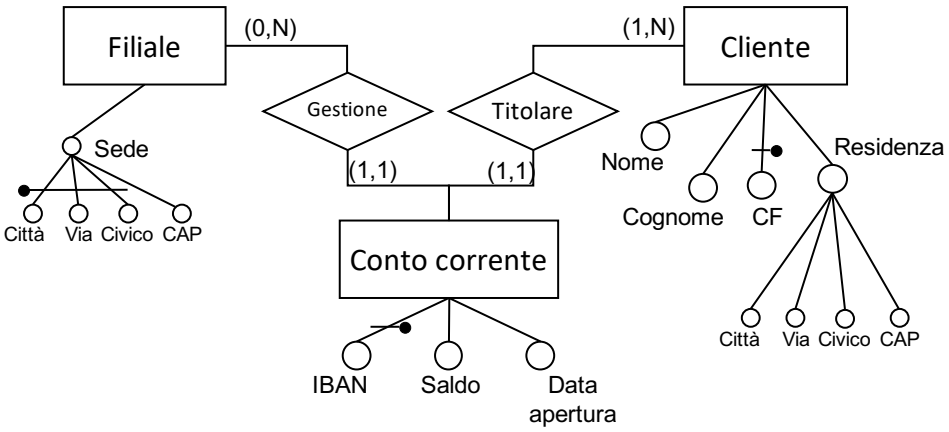


Si noti che in questo caso l'entità `Studente` è debole, dato che non si può escludere che studenti di atenei diversi abbiano numero di matricola identico. L'identificatore di `Studente` dovrà essere composto sia dal numero di matricola che dall'identificatore dell'entità `Ateneo`, poiché i numeri di matricola assegnati da un dato ateneo non possono ripetersi.

Si è scelto di codificare il livello dei corsi con le stringhe `L` (laurea), `LM` (laurea magistrale) e `CU` (corso a ciclo unico): si è quindi definito il dominio dell'attributo corrispondente come una stringa di due caratteri, e si è previsto un vincolo di dominio esteso per assicurare che gli unici valori ammessi siano le tre stringhe indicate sopra.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_atenei.sql`.

6. Lo **schema Entità-Relazione** è il seguente:



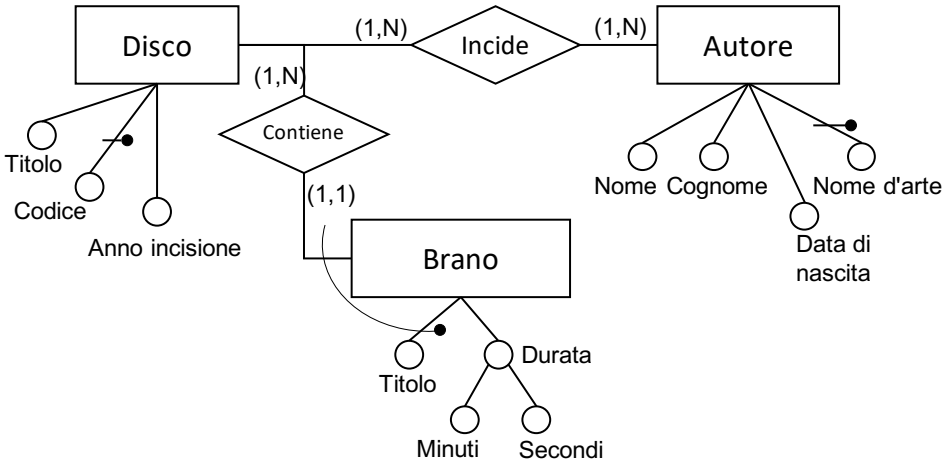
Si noti che non è necessario rappresentare esplicitamente le filiali presso le quali ogni cliente possiede conti correnti (questo si potrebbe fare attraverso una relazione tra le entità corrispondenti), dato che questa informazione può essere ricavata in modo indiretto attraverso le relazioni `Titolare` e `Gestione` e l'entità `Conto corrente`. Per esempio, come esercizio si scriva in linguaggio SQL la seguente interrogazione: trovare le sedi di tutte le filiali di cui sia cliente una certa persona avente codice fiscale `ABCXYZ12A34B567C`.

Gli attributi `Sede` e `Residenza` delle entità `Filiale` e `Cliente` sono composti. Nello schema logico compariranno solo le colonne corrispondenti agli attributi che li compongono.

Per il CAP si è scelto come dominio una stringa di cinque caratteri invece che un numero intero, per tener conto della presenza della cifra `0` come primo carattere. Si dovrebbe anche prevedere un vincolo di dominio esteso che assicuri che i cinque caratteri corrispondano a quelli dei CAP esistenti in Italia: per semplicità si trascura questo vincolo.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_banca.sql`.

7. Lo **schema Entità-Relazione** è il seguente:

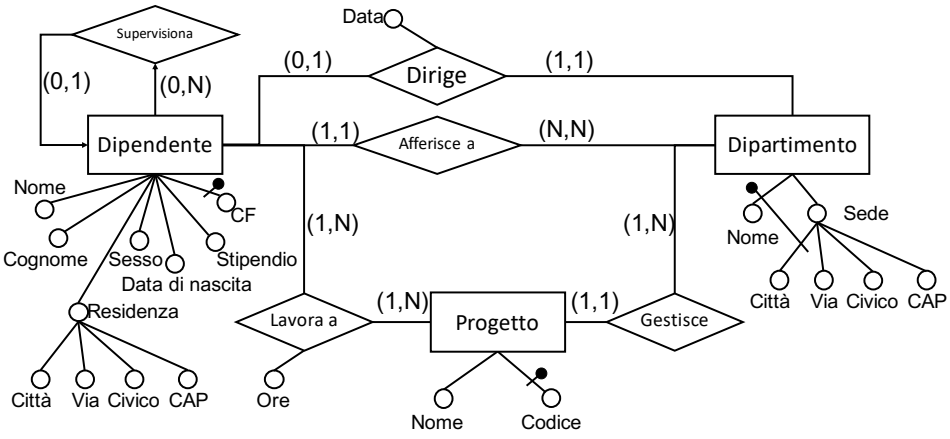


L'entità `Brano` è debole, e il suo identificatore può essere definito associando al titolo di un brano il disco (ovvero il codice del disco) a cui appartiene.

Si notino i vincoli di dominio esteso sulle colonne `Minuti` e `Secondi` della tabella `Brani` (per garantire che i loro valori siano compresi tra 0 e 59) e il vincolo di *tupla* su entrambe tali colonne per garantire che la durata di un brano non sia nulla.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato: `SQL_casa_discografica.sql`.

8. Lo **schema Entità-Relazione** è il seguente:



Si noti che la relazione `Supervisiona` è definita su coppie di elementi di una *stessa* entità (`Dipendente`). Le frecce sono state introdotte per chiarire, insieme al nome della relazione, il significato della cardinalità: un dipendente *supervisiona* nessuno, uno o più dipendenti (cardinalità $(0, N)$), e può *essere supervisionato* da nessuno o da un altro dipendente. Trattandosi di una relazione uno-a-molti, nello schema logico relazionale l'informazione da essa rappresentata (chi è il supervisore di un dato dipendente) sarà inserita nella tabella `Dipendenti` come una colonna aggiuntiva (`Supervisore`) che conterrà il codice fiscale del supervisore; su tale colonna sarà definito un vincolo d'integrità referenziale con la colonna `CF` della *stessa* tabella. Se un dipendente non ha un supervisore, il valore della colonna `Supervisore` sarà `NULL`.

Nello schema logico si è usato il nome `Dipendenti-Progetti` per la tabella corrispondente alla relazione molti-a-molti `Lavora a`.

I comandi SQL per la definizione delle tabelle sono riportati nel file allegato `SQL_azienda.sql`.

2 Formulazione di interrogazioni

1. Interrogazioni sulla base di dati dell'esercizio 1 della sezione 1 (*Segreteria studenti*):

   (a)
   ```
   SELECT "Nome", "Cognome" FROM "Studenti"
   WHERE "Data di nascita" >= '1991-01-01' AND "Data di nascita" <= '1991-12-31';
   ```

   (b)
   ```
   SELECT DISTINCT "Studente" FROM "Esami" WHERE "Voto" = 30;
   ```

   (c)
   ```
   SELECT "Studenti"."Cognome" FROM "Studenti", "Esami"
   WHERE "Esami"."Esame" = 'AMI01' AND "Studenti"."Matricola" = "Esami"."Studente";
   ```
   La stessa interrogazione scritta con variabili di *range*:
   ```
   SELECT S."Cognome" FROM "Studenti" S, "Esami" E
   WHERE E."Esame" = 'AMI01' AND S."Matricola" = E."Studente";
   ```

   (d)
   ```
   SELECT S."Cognome" FROM "Studenti" S, "Insegnamenti" I, "Esami" E
   WHERE I."Nome" = 'Analisi I' AND S."Matricola" = E."Studente"
         AND I."Codice" = E."Esame";
   ```

   (e)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Matricola" IN
      (SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (f)
   ```
   SELECT "Nome" FROM "Insegnamenti" WHERE "Codice" IN
      (SELECT "Esame" FROM "Esami" WHERE "Voto" = 30);
   ```

   (g)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Data di nascita" >= ALL
      (SELECT "Data di nascita" FROM "Studenti");
   ```

   (h)
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame"='AMI01' AND E."Voto" > ANY
      (SELECT "Voto" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (i)
   ```
   SELECT "Cognome" FROM "Studenti" WHERE "Data di nascita" =
      (SELECT MAX ("Data di nascita") FROM "Studenti");
   ```

   (j)
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" >
      (SELECT MIN ("Voto") FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

   (k) Usando operatori insiemistici:
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" >= ALL
      (SELECT "Voto" FROM "Esami" WHERE "Esame" = 'AMI01');
   ```
   Usando operatori di aggregazione:
   ```
   SELECT "Studente" FROM "Esami" WHERE "Esame" = 'AMI01' AND "Voto" =
      (SELECT MAX ("Voto") FROM "Esami" WHERE "Esame" = 'AMI01');
   ```

2. Interrogazioni sulla base di dati dell'esercizio 2 della sezione 1 (*Circolo nautico*):

   (a)
   ```
   SELECT "vid", "vnome" FROM "Velisti";
   ```

   (b)
   ```
   SELECT * FROM "Barche";
   ```

   (c)
   ```
   SELECT "vnome" FROM "Velisti" WHERE "età" > 40;
   ```

   (d)
   ```
   SELECT "vnome" FROM "Velisti" WHERE "età" > 40 OR "esperienza" > 7;
   ```

(e)
```
SELECT "bid" FROM "Prenotazioni" WHERE "giorno" = '2018-12-11';
```

(f)
```
SELECT DISTINCT "colore" FROM "Barche";
```

(g)
```
SELECT "bnome" FROM "Barche" WHERE NOT ("colore" = 'rosso');
```

(h)
```
SELECT "Velisti"."vnome" FROM "Velisti", "Prenotazioni"
WHERE "Prenotazioni"."bid"= 103 " AND "Velisti"."vid" = "Prenotazioni"."vid";
```

La stessa interrogazione scritta con variabili di *range*:

```
SELECT V."vnome" FROM "Velisti" V, "Prenotazioni" P
WHERE P."bid" = 103 AND V."vid" = P."vid";
```

(i)
```
SELECT B."colore" FROM "Velisti" V, "Barche" B, "Prenotazioni" P
WHERE V."vnome" = 'Rusty' AND P."vid" = V."vid" AND P."bid" = B."bid";
```

(j) Questa interrogazione può essere scritta come segue:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND (B."colore" = 'blu' OR B."colore" = 'verde');
```

Può anche essere scritta usando l'operatore insiemistico `UNION`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
UNION
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(k) L'interrogazione seguente **non** corrisponde a quella richiesta (il suo significato reale è: trovare i codici dei velisti che hanno prenotato qualche barca che abbia colore blu e verde – ciò implica anche che tale interrogazione non potrà produrre nessun risultato):

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu' AND B."colore" = 'verde';
```

L'interrogazione richiesta può essere scritta correttamente mediante l'operatore insiemistico `INTERSECT`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
INTERSECT
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(l) Un altro esempio di interrogazione SQL che **non** corrisponde a quella richiesta (il suo significato reale è: trovare i codici dei velisti che hanno prenotato qualche barca il cui colore sia blu e non sia verde):

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu' AND B."colore" != 'verde';
```

L'interrogazione richiesta può essere scritta correttamente mediante l'operatore insiemistico `EXCEPT`:

```
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'blu'
EXCEPT
SELECT P."vid" FROM "Prenotazioni" P, "Barche" B
WHERE P."bid" = B."bid" AND B."colore" = 'verde';
```

(m)
```
SELECT P."bid" FROM "Prenotazioni" P, "Velisti" V
WHERE P."vid" = V."vid" AND V."esperienza" >= 8
EXCEPT
SELECT P."bid" FROM "Prenotazioni" P, "Velisti" V
WHERE P."vid" = V."vid" AND V."età" <= 35;
```

(n)
```
SELECT "vnome" FROM "Velisti" WHERE "vid" IN
   (SELECT "vid" FROM "Prenotazioni" WHERE "bid" = 103);
```

(o)
```
SELECT "vnome" FROM "Velisti" WHERE "vid" NOT IN
   (SELECT "vid" FROM "Prenotazioni" WHERE "bid" = 103);
```

Si noti che questa interrogazione **non** può essere scritta come segue:

```
SELECT "Velisti"."vnome" FROM "Velisti", "Prenotazioni"
WHERE "Prenotazioni"."bid" != 103 " AND "Velisti"."vid" = "Prenotazioni"."vid";
```

Infatti questa interrogazione SQL estrae i codici dei velisti che hanno prenotato *anche* qualche barca diversa dalla numero 103 (quindi non esclude i velisti che abbiano prenotato *anche* la 103, oltre ad altre barche), mentre non estrae i codici dei velisti che non abbiano mai prenotato *nessuna* barca (quindi neanche la 103).

(p)
```
SELECT * FROM "Velisti" WHERE "esperienza" > ANY
   (SELECT "esperienza" FROM "Velisti" WHERE "vnome" = 'Lubber');
```

(q)
```
SELECT "vid" FROM "Velisti" WHERE "esperienza" >= ALL
   (SELECT "esperienza" FROM "Velisti");
```

Si noti che scrivendo `"esperienza" > ALL` invece che `"esperienza" >= ALL` l'interrogazione non potrebbe produrre nessun risultato, poiché il risultato dell'interrogazione nidificata include il valore dell'esperienza maggiore.

(r)
```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" >= ALL
   (SELECT "età" FROM "Velisti");
```

(s)
```
SELECT COUNT (*) FROM "Velisti";
```

(t)
```
SELECT AVG ("età") FROM "Velisti";
```

(u)
```
SELECT COUNT (DISTINCT "esperienza") FROM "Velisti";
```

(v)
```
SELECT AVG ("età") FROM "Velisti" WHERE "esperienza" = 10;
```

(w) Usando operatori insiemistici:

```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" >= ALL
   (SELECT "età" FROM "Velisti");
```

Usando operatori di aggregazione:

```
SELECT "vnome", "età" FROM "Velisti" WHERE "età" =
   (SELECT MAX ("età") FROM "Velisti");
```

(x)
```
SELECT "vnome" FROM "Velisti" WHERE "esperienza" =
   (SELECT MAX ("esperienza") FROM "Velisti");
```

Turno 1 GRUPPO A

1) Elencare le forniture che sono relative a parti e prodotti della stessa città, che deve però essere diversa da quella del fornitore; produrre, oltre ai dati della fornitura, anche la città del prodott/parte e quella del fornitore.

```
SELECT FCOD,PCOD,PRCOD,F.CITTA,P.CITTA
FROM FORNITURE FT,FORNITORI F,PARTI P, PRODOTTI PR
WHERE FT.FCOD=F.COD AND FT.PCOD=P.COD AND FT.PRCOD=PR.COD
      AND P.CITTA=PR.CITT AND P.CITTA<>F.CITTA
```

2) Si consideri la fornitura (o le forniture) con quantità massima; per il fornitore di tale fornitura si vuole sapere qual è il numero totale di forniture da lui fatte.

```
SELECT COUNT(*), FCOD
FROM FORNITURE
WHERE QTA = (SELECT MAX(QTA)
             FROM FORNITURE)
GROUP BY FCOD
```

Turno 1GRUPPO B

1) Elencare i prodotti che sono forniti in almeno due forniture nella stessa quantità; produrre tutti i dati del prodotto assieme alla quantità (in generale verranno prodotte più tuple per lo stesso prodotto).

```
SELECT PR.*, FT1.QTA
FROM FORNITURE FT1, FORNITURE FT2, PRODOTTI PR
WHERE FT1.PRCOD=FT2.PRCOD
        AND
        (FT1.FCOD<>FT2.FCOD OR FT1.PCOD<>FT2.PCOD)
        AND FT1.QTA=FT2.QTA
        AND (PR.COD=FT1.PRCOD OR PR:COD=FT2.PRCOD)
```

2) Produrre un elenco di “codici prodotto, codice parte, nome prodotto, nome parte, conteggio” per tutte le coppie di codice prodotto , codice parte presente in forniture, dove “conteggio” indica il numero di forniture in cui compare quella specifica coppia di codice prodotto codice parte.

```
SELECT FT1.PRCOD, PR.NOME, FT1.PCOD, P.COD, COUNT(*) AS CONTEGGIO
FROM FORNITURE FT1, PRODOTTI PR, PARTI P
WHERE FT1.PRCOD=PR.COD AND FT1.PCOD=P.COD
GROUP BY FT1.PRCOD, PR.NOME, FT1.PCOD, P.COD
```

Turno 2 GRUPPO A

1) Elencare le coppie di fornitori che siano della stessa città e che forniscano almeno un prodotto o una parte nella stessa quantità; mostrare codice e nome dei fornitori, assieme alla quantità.

```
SELECT FT1.FCOD,F1.NOME,FT2.FCOD,F2.NOME,FT1.QTA
FROM FORNITORI F1, FORNITORI F2, FORNITURE FT1, FORNITURE FT2
WHERE FT1.FCOD=F1.COD AND FT2.FCOD=F2.COD AND F1.CITTA=F2.CITTA
      AND
      FT1.QTA=FT2.QTA AND (FT1.PRCOD=FT2.PRCOD OR FT1.PCOD=FT2.PCOD)
```

2) Elencare il nome dei prodotti che non sono mai forniti assieme a parti rosse e che abbiano almeno una fornitura di un fornitore di Milano.

```
SELECT PR.NOME
FROM PRODOTTI PR
WHERE EXISTS (SELECT *
              FROM FORNITURE FT, FORNITORI F
              WHERE PR.COD=FT.PRCOD AND FT.FCOD=F.COD AND F.CITTA=”MILANO”)
      AND
      NOT EXISTS (SELECT *
                  FROM FORNITURE FT, PARTI P
```

```
WHERE PR.COD=FT.PRCOD AND FT.PCOD=P.COD AND P.COLORE="ROSSO")
```

Turno 2 GRUPPO B

1) Elencare i fornitori che hanno almeno tre foniture con lo stesso prodotto; elencare tutti i dati del fornitore e del prodotto.

```
SELECT F.COD, F.NOME, F.CITTA, PR.COD, PR.NOME, PR.CITTA
FROM FORNITURE FT1, FORNITURE FT2, FORNITURE FT3, FORNITORI F, PRODOTTI PR
WHERE FT1.PRCOD=FT2.PRCOD AND FT3.PRCOD=FT1.PRCOD
      AND
      FT1.PCOD<>FT2.PCOD AND FT1.PCOD<>FT3.PCOD
      AND
      FT1.FCOD=F.COD
      AND
      FT1.PRCOD=PR.COD
```

2) Calcolare la percentuale delle forniture che sono relative a prodotti e parti della stessa città.

```
SELECT COUNT(*)/(SELECT COUNT(*) FROM FORNITURE) AS PERCENTUALE
FROM FORNITURE FT, PARTI P, PRODOTTI PR
WHERE FT.PRCOD=PR.COD AND FT.PCOD=P.COD
        AND
        PR.CITTA=P.CITTA
```

Turno 3 GRUPPO A

1) Identificare le forniture nelle quali la città del prodotto o quella della parte (ma non entrambe) coincidono con la città del fornitore; produrre un risultato che elenchi il codice, il nome e la citta del fornitore, della parte e del prodotto di ciascuna di tali forniture.

```
SELECT F.COD,F.NOME,F.CITTA,PR.COD,PR.NOME,PR.CITTA,P.COD,P.NOME,P.CITTA
FROM FORNITORI F, PRODOTTI PR, PARTI P, FORNITURE FT
WHERE F.COD=FT.FCOD AND PR.COD=FT.PRCOD AND P.COD=FT.PCOD
      AND ((F.CITTA=PR.CITTA AND F.CITTA<>P.CITTA) OR
            (F.CITTA=P.CITTA AND F.CITTA<>PR.CITTA))
```

2) Creando, se necessario, opportune tabelle intermedie, produrre un elenco CITTA, QF, QPA, QPR nel quale gli attributi Q sono la frequenza con cui quella CITTA compare nelle città dei Fornitori, Parti, PRodotti in fornitura

```
CREATE TABLE CITTAF        CREATE TABLE CITTAP       CREATE TABLE CITTAPR
 (CITTA CHARACTER (20),     (CITTA CHARACTER (20),    (CITTA CHARACTER (20),
  QF    INTEGER);            QPA   INTEGER);           QPR   INTEGER);

INSERT INTO TABLEF FROM        INSERT INTO TABLEP FROM
(SELECT CITTA, COUNT(*) AS QF   (SELECT CITTA, COUNT(*) AS QPA
 FROM FORNITORI                  FROM PARTI
 GROUP BY CITTA);                GROUP BY CITTA);

INSERT INTO TABLEPR FROM
(SELECT CITTA, COUNT(*) AS QPR
 FROM PRODOTTI
 GROUP BY CITTA);

SELECT CITTA,QF,QPA,QPR
FROM CITTAF,CITTAP,CITTAPR
WHERE CITTAF.CITTA=CITTAP.CITTA AND CITTAF.CITTA=CITTAPR.CITTA
```

Turno 3 GRUPPO B

1) Elencare i dipendenti che sono superiori di secondo livello (cioè che sono il superiore del superiore di un impiegato)

```
SELECT D1.*
FROM DIPENDENTI D1, DIPENDENTI D2, DIPENDENTI D3
WHERE D1.MATRICOLA=D2.SUP
      AND
      D2.MATRICOLA=D1.SUP
```

2) Dei dipendenti che guadagnano meno della media di tutti, elencare la matricola, il nome, lo stipendio senza commissioni, il nome del dipartimento cui appartengono.

```
SELECT MATRICOLA,NOME_IMP,STIPENDIO,DIPART.NOME_DIPART
FROM DIPENDENTI,DIPART
WHERE DIPENDENTI.NUM_DIPART=DIPART.NUM_DIDART
AND
STIPENDIO < (SELECT AVG(STIPENDIO)
             FROM DIPENDENTI)
```

SQL

Soluzioni dell'esercizio 4.7 del testo (Basi di Dati, Modelli e linguaggi di interrogazione)

1. le città con un aeroporto di cui non è noto il numero di piste

```
Select CITTA from AEROPORTO where NUMPISTE is null;
```

2. le nazioni da cui parte e arriva il volo con codice AZ1000

```
select distinct aep.nazione as NazionePartenza, aea.nazione as NazioneArrivo
from aeroporto aep, volo, aeroporto aea
where volo.cittaa=aea.citta and volo.cittap=aep.citta and volo.idvolo='AZ1000';
```

3. i tipi di aereo usati nei voli che partono da Caselle

```
select distinct TipoAereo
from volo
where cittap='CASELLE';
```

4. i tipi di aereo ed il corrispondente numero di passeggeri per i tipi di aereo usati nei voli che partono da Malpensa

```
select distinct aereo.TipoAereo, aereo.numpasseggeri
from volo join aereo on volo.tipoaereo=aereo.tipoaereo
where cittap='MALPENSA';
```

5. le citta da cui partono voli internazionali

```
select distinct cittap as citta
from volo join aeroporto on volo.cittap=aeroporto.citta
          join aeroporto due on volo.cittaa=due.citta
where aeroporto.nazione <> due.nazione;
```

6. le città da cui partono voli diretti a Linate, ordinate alfabeticamente

```
select distinct cittap as origine
from volo
where cittaa='LINATE'
order by cittap;
```

7. contare i voli internazionali in partenza da Linate il giovedì.

```
select count(*) as vol_int_giov
from volo join aeroporto on volo.cittap=aeroporto.citta
          join aeroporto due on volo.cittaa=due.citta
where aeroporto.nazione <> due.nazione
      and cittap='LINATE' and GIORNOSETT=4;
```

Definiamo una vista dei voli internazionali

```
create view voli_int (idvolo,giornosett,cittap,nazionep,cittaa,nazionea) as
select idvolo,giornosett,volo.cittap,aeroporto.nazione,volo.cittaa,due.nazione
from volo join aeroporto on volo.cittap=aeroporto.citta
          join aeroporto due on volo.cittaa=due.citta
where aeroporto.nazione <> due.nazione;

select count(*) as volo_int_giov
from voli_int
where cittap='LINATE' and GIORNOSETT=4;
```

8. il numero di voli internazionali che partono ogni fine settimana da città italiane

Utilizziamo la vista definita nella risposta precedente:

```
select cittap, count(*)
```

```
from voli_int
where nazionep = 'ITALIA'
and giornosett > 5
group by cittap;
```

9.Le città inglesi da cui partono più di 5 voli alla settimana diretti in italia

Utilizziamo la vista definita nella risposta precedente:

```
select distinct cittap
from voli_int VI1
where  5 <      (select count(*)
                 from voli_int VI2
                 where VI2.cittap = VI1.cittap
                 and VI2.nazionep = 'INGHILTERRA'
                 and VI2.nazionea = 'ITALIA');
```

10.1 Gli aeroporti italiani che hanno solo voli interni (usando operatori insiemistici)

```
select *
from aeroporto
where nazione = 'ITALIA'
except (
select apt.*
from aeroporto apt, volo, aeroporto apt2
where apt.citta = volo.cittap and
apt2.citta = volo.cittaa
and (apt.nazione <> apt2.nazione)
union
select apt2.*
from aeroporto apt, volo, aeroporto apt2
where apt.citta = volo.cittap and
apt2.citta = volo.cittaa
and (apt.nazione <> apt2.nazione))
```

10.2 Gli aeroporti italiani che hanno solo voli interni (usando interrog. nidificata con not in)

```
select *
from aeroporto
where nazione = 'ITALIA' and
      citta not in (select apt.citta
                    from aeroporto apt, volo, aeroporto apt2
                    where apt.citta = volo.cittap and
                    apt2.citta = volo.cittaa
                    and apt.nazione <> apt2.nazione
                    union
                    select apt2.citta
                    from aeroporto apt, volo, aeroporto apt2
                    where apt.citta = volo.cittap and
                    apt2.citta = volo.cittaa
                    and (apt.nazione <> apt2.nazione));
```

10.3 Gli aeroporti italiani che hanno solo voli interni (usando interrog. nidificata con not exists)

```
select *
from aeroporto apt1
where nazione = 'ITALIA'and not exists (select *
                                        from volo, aeroporto apt2
                                        where ((apt1.citta = volo.cittap and
apt2.citta = volo.cittaa)
                                                or
                                               (apt1.citta = volo.cittaa and
apt2.citta = volo.cittap))
                                        and
```

```
                    apt1.nazione <> apt2.nazione);
```

10.4 Gli aeroporti italiani che hanno solo voli interni (usando outer join e count)

```
select *
from aeroporto AE
where AE.nazione = 'ITALIA'and
0 = (select count(*)
     from aeroporto AEP full join volo V on AEP.citta = V.cittap
                        full join aeroporto AEA on AEA.citta = V.cittaa
     where (AE.citta = AEP.citta or AE.citta = AEA.citta)
     and AEP.nazione <> AEA.nazione);
```

11. Le città che sono servite dall'aereo caratterizzato dal massimo numero di passeggeri.

```
select cittap
from volo
where tipoaereo =ANY (select tipoaereo
                      from aereo
                      where numpasseggeri = (select max(numpasseggeri)
                                             from aereo))
union
select cittaa
from volo
where tipoaereo =ANY (select tipoaereo
                      from aereo
                      where numpasseggeri = (select max(numpasseggeri)
                                             from aereo));
```

12. Il massimo numero di passeggeri che possono arrivare in un aeroporto italiano dall'Inghilterra di martedì

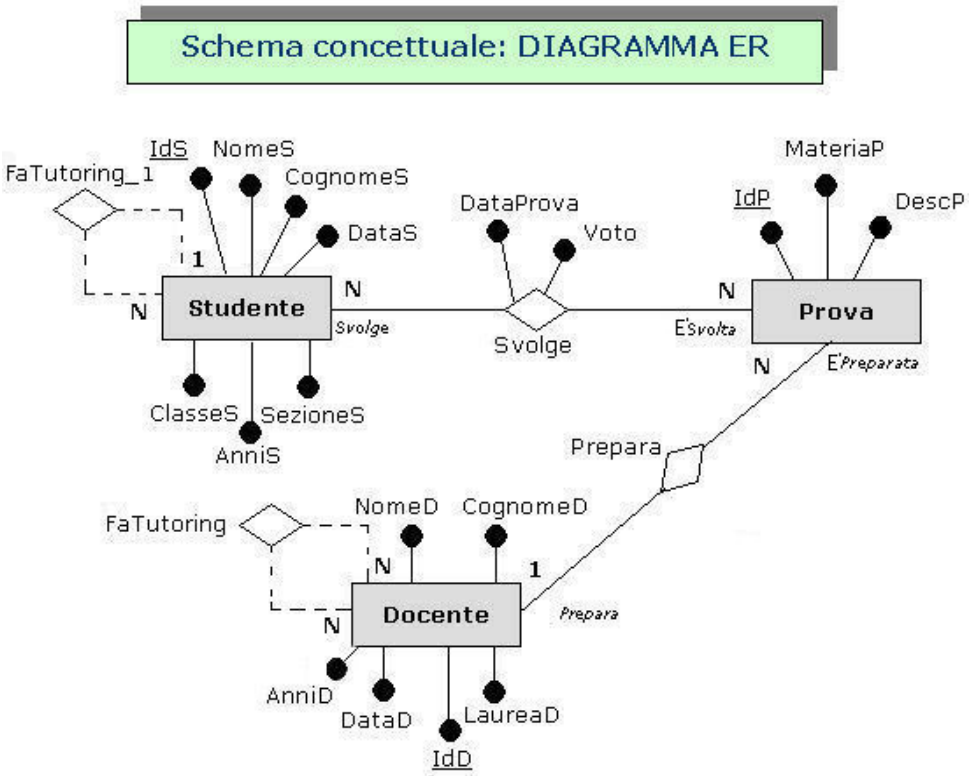
Definiamo una vista:

```
create view maxPass(aeroporto,pass) as
select V.cittaa, sum(A.numpasseggeri)
from volo V, aereo A, aeroporto AEP, aeroporto AEA
where V.tipoaereo = A.tipoaereo
and AEP.citta = V.cittap and AEA.citta = V.cittaa
and V.giornosett = 2
and AEP.nazione = 'INGHILTERRA' and AEA.nazione = 'ITALIA'
group by V.cittaa;

select max(pass)
from maxPass;
```

Sia dato il seguente diagramma E/R di esempio:

(vedi URL http://www.riochieregо.it/sqltest/index.htm)



Si ricava il seguente SCHEMA RELAZIONALE (MAPPING DEL DIAGRAMMA ER)

Schema logico relazionale

Studente (IdS, CognomeS, NomeS, DataS, ClasseS, SezioneS, AnniS, IdS1)
con l'attributo "IdS1" FK sull'attributo "IdS" della medesima relazione "Studente"

Svolge (IdS2, IdP2, DataProva, Voto)
con l'attributo "IdS2" FK sull'attributo "IdS" della relazione "Studente"
con l'attributo "IdP2" FK sull'attributo "IdP" della relazione "Prova"

Prova (IdP, MateriaP, DescP, IdD1)
con l'attributo "IdD1" FK sull'attributo "IdD" della relazione "Docente"

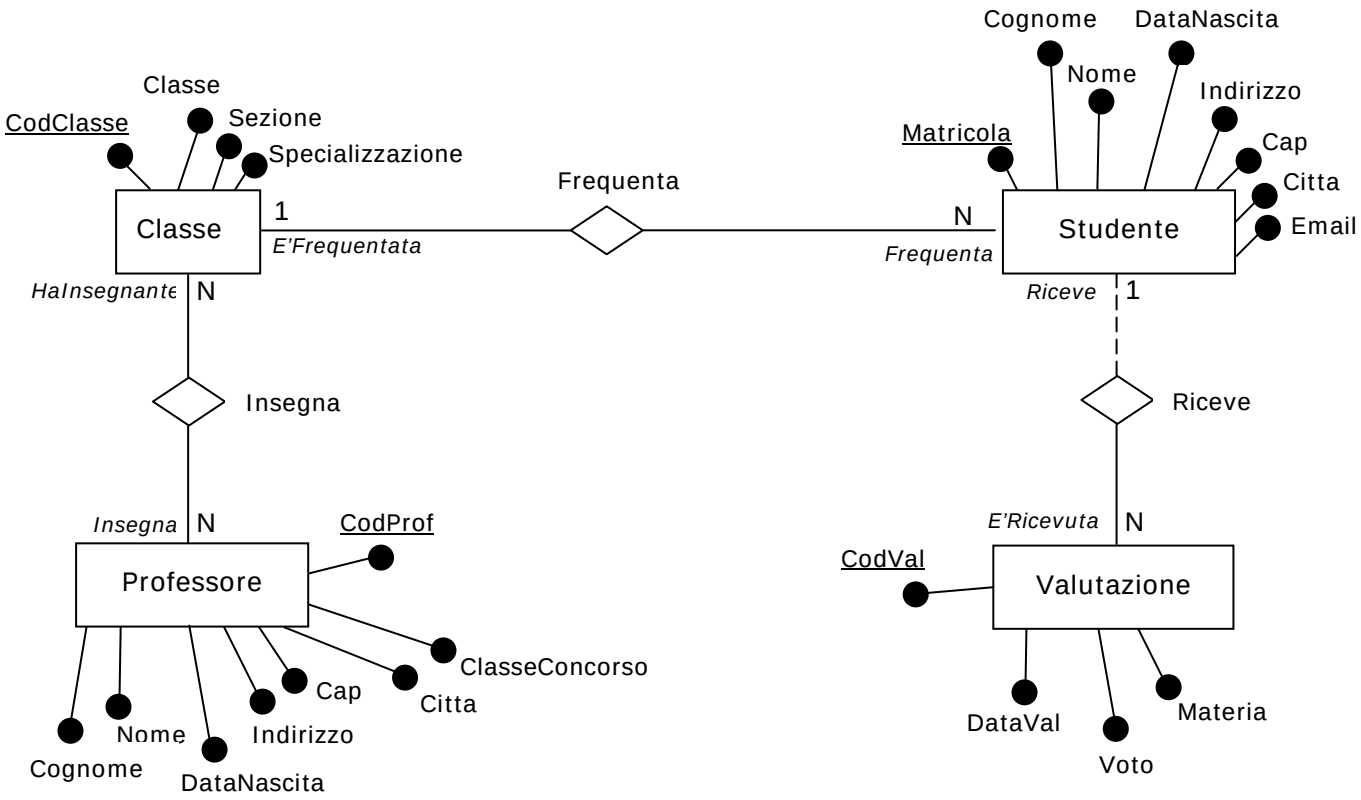
VR_{IdS2} (Svolge) $\subseteq$ VR_{IdS} (Studente)
ossia tutti i valori contenuti nell'attributo "IdS2" della relazione "Svolge" sono contenuti nell'insieme dei valori dell'attributo "IdS" della relazione "Studente"
VR_{IdP2} (Svolge) $\subseteq$ VR_{IdP} (Prova)
ossia tutti i valori contenuti nell'attributo "IdP2" della relazione "Svolge" sono contenuti nell'insieme dei valori dell'attributo "IdP" della relazione "Prova"
VR_{IdS} (Studente) $\subseteq$ VR_{IdS2} (Svolge)
per la TOTALITA' dell'associazione diretta "Svolge"
VR_{IdP} (Prova) $\subseteq$ VR_{IdP2} (Svolge)
per la TOTALITA' dell'associazione inversa "E'Svolta"

Docente (IdD, CognomeD, NomeD, DataD, LaureaD, AnniD,)

VR_{IdD} (Docente) $\subseteq$ VR_{IdD1} (Prova)
per la TOTALITA' dell'associazione diretta "Prepara"
VR_{IdD1} (Prova) $\subseteq$ VR_{IdD} (Docente)
per la TOTALITA' dell'associazione inversa "E'Preparata"

FaTutoring(IdD2, IdD3)
con l'attributo "IdD2" FK sull'attributo "IdD" della relazione "Docente"
con l'attributo "IdD3" FK sull'attributo "IdD" della relazione "Docente"

1) Sia dato il seguente diagramma E/R di esempio:



Si ricava il seguente SCHEMA RELAZIONALE (MAPPING DEL DIAGRAMMA ER)

Classe (CodClasse, Classe, Sezione, Specializzazione)
Studente (Matricola, Cognome, Nome, DataNascita, Indirizzo, Cap, Citta, Email, CodClasse1)
con l'attributo "CodClasse1" che è FK sull'attributo "CodClasse" della relazione "Classe"

$VR_{CodClasse1}$ (Studente) $\subseteq$ $VR_{CodClasse}$ (Classe) derivante dalla TOTALITA' della associazione diretta "Frequenta"
$VR_{CodClasse}$ (Classe) $\subseteq$ $VR_{CodClasse1}$ (Studente) derivante dalla TOTALITA' della associazione inversa "E'Frequentata"

Valutazione (CodVal, DataVal, Voto, Materia, Matricola1)
con l'attributo "Matricola1" che è FK sull'attributo "Matricola" della relazione "Studente"

$VR_{Matricola1}$ (Valutazione) $\subseteq$ $VR_{Matricola}$ (Studente) derivante dalla TOTALITA' della associazione inversa "E'ricevuta"

Professore (CodProf, Cognome, Nome, Data Nascita, Indirizzo, Cap, Citta, ClasseConcorso)
Insegna (CodProf2, CodClasse2, NumOre)
con l'attributo "CodProf2" che è FK sull'attributo "CodProf" della relazione "Professore"
con l'attributo "CodClasse2" che è FK sull'attributo "CodClasse" della relazione "Classe"

$VR_{CodProf2}$ (Insegna) $\subseteq$ $VR_{CodProf}$ (Professore) derivante dal mapping dell'associazione di molt. N:N "Insegna"
$VR_{CodClasse2}$ (Insegna) $\subseteq$ $VR_{CodClasse}$ (Classe) derivante dal mapping dell'associazione di molt. N:N "Insegna"

$VR_{CodProf}$ (Professore) $\subseteq$ $VR_{CodProf2}$ (Insegna) derivante dalla TOTALITA' della associazione diretta "Insegna"
$VR_{CodClasse}$ (Classe) $\subseteq$ $VR_{CodClasse2}$ (Insegna) derivante dalla TOTALITA' della associazione inversa "E'Insegnata"

2) Sia dato lo schema relazionale costituite dalle seguenti tabelle (chiave sottolineata) e dai seguenti vincoli referenziali:

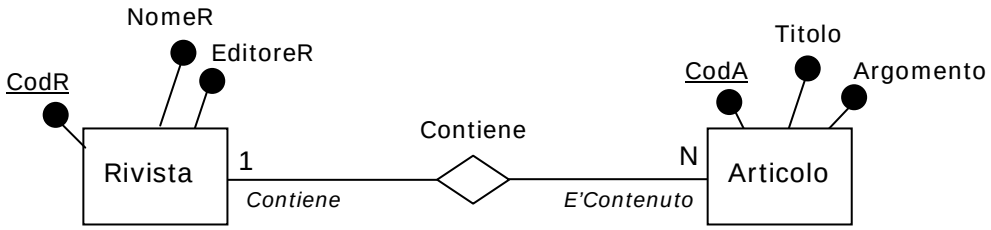
Rivista (CodR, NomeR, EditoreR)

Articolo (CodA, Titolo, Argomento, CodR1)

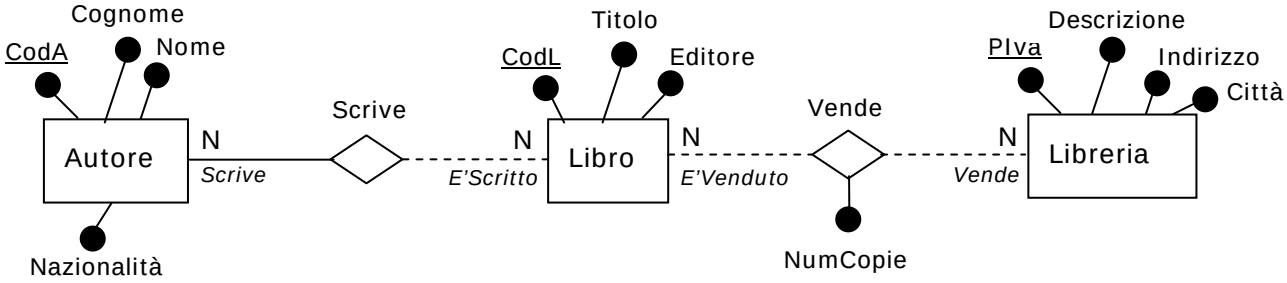
con l'attributo "CodR1" che è FK sull'attributo "CodR" della relazione "Rivista"

VR_{CodR} (Rivista) $\subseteq$ VR_{Codr1} (Articolo)
VR_{Codr1} (Articolo) $\subseteq$ VR_{CodR} (Rivista)

Si ricava il seguente DIAGRAMMA ER



3) Sia dato il seguente diagramma E/R di esempio:



Si ricava il seguente SCHEMA RELAZIONALE (MAPPING DEL DIAGRAMMA ER)

Libreria (PIva, Descrizione, Indirizzo, Citta)

Libro (CodL, Titolo, Editore)

Vende (PIva1, CodL1, NumCopie)

con l'attributo "PIva1" che è FK sull'attributo "PIva" della relazione "Libreria"
con l'attributo "CodL1" che è FK sull'attributo "CodL" della relazione "Libro"

VR_{PIva1} (Vende) $\subseteq$ VR_{PIva} (Libreria) derivante dal mapping dell'associazione di molt. N:N "Vende"
VR_{CodL1} (Vende) $\subseteq$ VR_{CodL} (Libro) derivante dal mapping dell'associazione di molt N:N "Vende"

Autore (CodA, Cognome, Nome, Nazione)

Scrive (CodA1, CodL1)

con l'attributo "CodL1" che è FK sull'attributo "CodL" della relazione " Libro"

VR_{CodA1} (Scrive) $\subseteq$ VR_{CodA} (Autore) derivante dal mapping dell'associazione di molt. N:N "Scrive"
VR_{CodL1} (Scrive) $\subseteq$ VR_{CodL} (Libro) derivante dal mapping dell'associazione di molt. N:N "Scrive"

VR_{CodA} (Autore) $\subseteq$ VR_{CodA1} (Scrive) derivante dalla TOTALITA' della associazione diretta "Scrive"

4) Sia dato lo schema relazionale costituite dalle seguenti tabelle (chiave sottolineata) e dai seguenti vincoli referenziali:

Fornitore (CodFornitore, Cognome, Nome, DataNascita, Indirizzo, Cap, Città)

Prodotto (CodProdotto, Denominazione, Marca, Categoria, Costo)

Fornisce (CodFornitore1, CodProdotto1, DataFornitura)

con l'attributo "CodFornitore1" che è FK sull'attributo "CodFornitore" della relazione "Fornitore"

con l'attributo "CodProdotto1" che è FK sull'attributo "CodProdotto" della relazione "Prodotto"

$VR_{CodProdotto}$ (Prodotto) $\subseteq$ $VR_{CodProdotto1}$ (Fornisce)

Ordine (CodOrdine, Pezzi, DataOrdine, CodFornitore2, CodProdotto2)

con l'attributo "CodFornitore2" che è FK sull'attributo "CodFornitore" della relazione "Fornitore"

con l'attributo "CodProdotto2" che è FK sull'attributo "CodProdotto" della relazione "Prodotto"

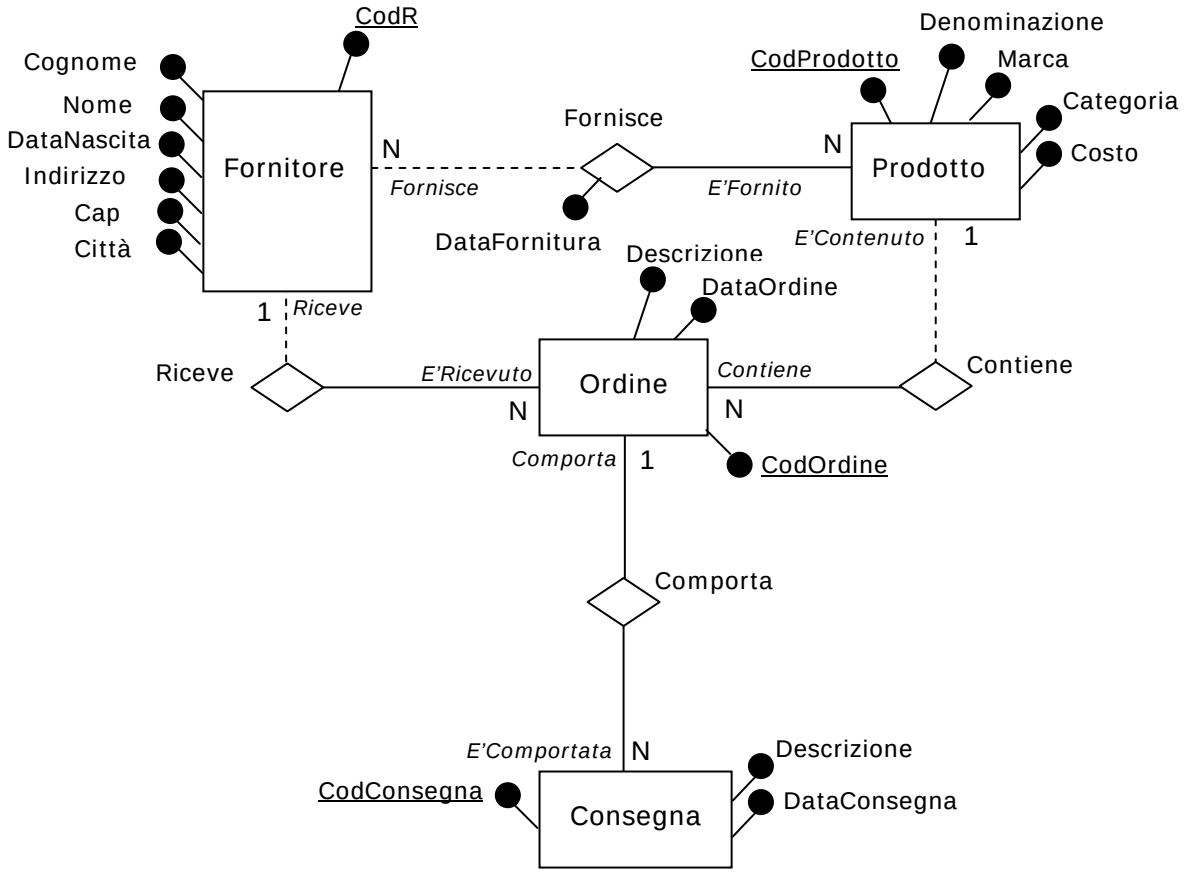
$VR_{CodFornitore2}$ (Ordine) $\subseteq$ $VR_{CodFornitore}$ (Fornitore)
$VR_{CodProdotto2}$ (Ordine) $\subseteq$ $VR_{CodProdotto}$ (Prodotto)

Consegna (CodConsegna, DataConsegna CodOrdine1)

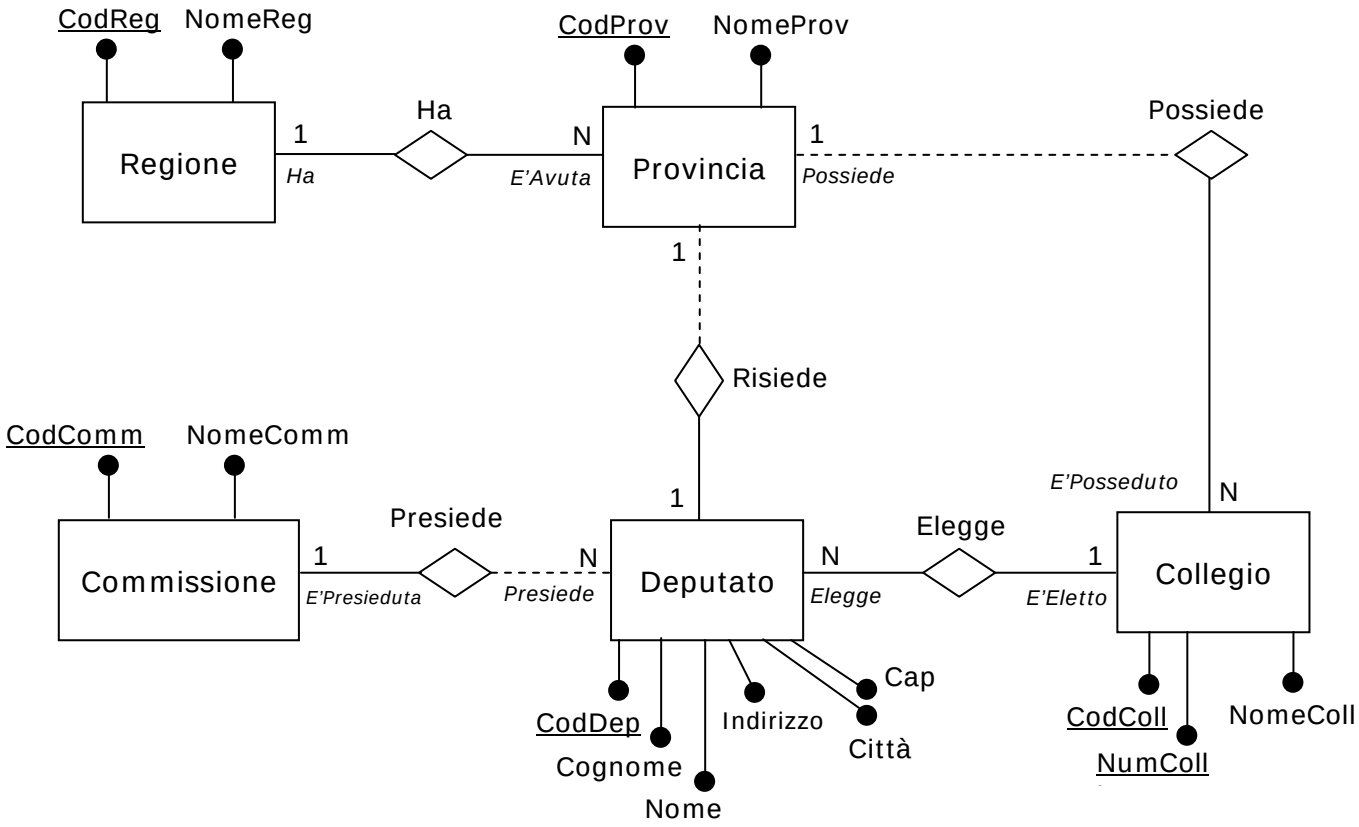
con l'attributo "CodOrdine1" che è FK sull'attributo "CodOrdine" della relazione "Ordine"

$VR_{CodOrdine}$ (Ordine) $\subseteq$ $VR_{CodOrdine1}$ (Consegna)
$VR_{CodOrdine1}$ (Consegna) $\subseteq$ $VR_{CodOrdine}$ (Ordine)

Si ricava il seguente DIAGRAMMA ER



5) Sia dato il seguente diagramma E/R di esempio:



Si ricava il seguente SCHEMA RELAZIONALE (MAPPING DEL DIAGRAMMA ER)

Regione (CodReg, NomeReg)

Provincia (CodProv, NomeProv, CodReg1)

con l'attributo "CodReg1" che è FK sull'attributo "CodReg" della relazione "Regione"

VR_{CodReg} (Regione) $\subseteq$ $VR_{CodReg1}$ (Provincia) deriva dalla TOTALITA' dell'associazione diretta "Ha"
VR_{CodReg} (Regione) $\supseteq$ $VR_{CodReg1}$ (Provincia) deriva dalla TOTALITA' dell'associazione inversa "E'Avuta"

Collegio (CodColl, NumColl, NomeColl, CodProv1)

con l'attributo "CodProv1" che è FK sull'attributo "CodProv" della relazione "Provincia"

$VR_{CodProv1}$ (Collegio) $\subseteq$ $VR_{CodProv}$ (Provincia) deriva dalla TOTALITA' dell'associazione inversa "E'Posseduto"

Deputato (CodDep, Cognome, Nome, Indirizzo, Cap, Città, CodComm1, CodProv2, CodColl1, NumColl1)

con l'attributo "CodComm1" che è FK sull'attributo "CodComm" della relazione "Commissione"
con l'attributo "CodProv2" che è FK sull'attributo "CodProv" della relazione "Provincia"
con gli attributi "CodColl1" e "NumColl1" che sono FK sugli attributi "CodColl" e "NumColl" della relazione "Collegio"

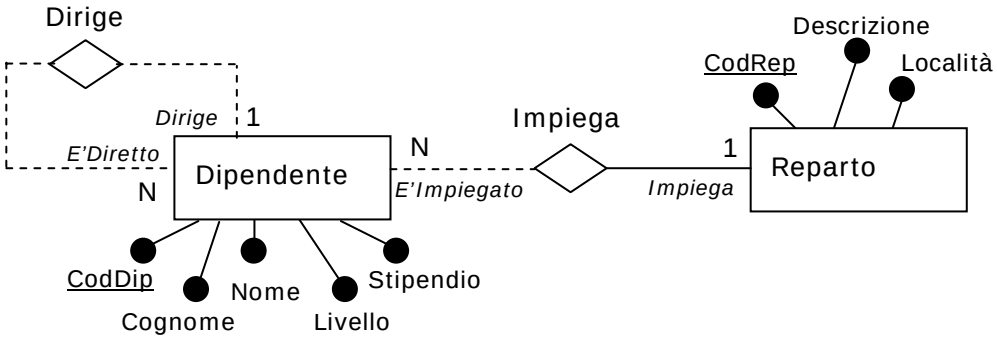
$VR_{CodColl,\ NumColl}$ (Collegio) $\subseteq$ $VR_{CodColl1,\ NumColl1}$ (Deputato) derivante dalla TOTALITA' dell'associazione diretta "Elegge"
$VR_{CodColl,\ NumColl}$ (Collegio) $\supseteq$ $VR_{CodColl1,\ NumColl1}$ (Deputato) derivante dalla TOTALITA' dell'ass. inversa "E'Eletto"

$VR_{CodProv2}$ (Deputato) $\subseteq$ $VR_{CodProv}$ (Provincia) derivante dalla TOTALITA' dell'associazione diretta "Risiede"

Commissione (CodComm, NomeComm)

$VR_{CodComm}$ (Commissione) $\subseteq$ $VR_{CodComm1}$ (Deputato) derivante dalla TOTALITA' dell'ass. inversa "E'Presieduta"

6) Sia dato il seguente diagramma E/R di esempio:



Si ricava il seguente SCHEMA RELAZIONALE (MAPPING DEL DIAGRAMMA ER)

Reparto (CodRep, Descrizione, Localita)
Dipendente (CodDip, Cognome, Nome, Livello, Stipendio, CodCapo, CodRep1)

con l'attributo "CodCapo" che è FK sull'attributo "CodDip" della relazione "Dipendente"

con l'attributo "CodRep1" che è FK sull'attributo "CodRep" della relazione "Reparto"

VR_{CodRep} (Reparto) $\subseteq$ $VR_{CodRep1}$ (Dipendente) derivante dalla TOTALITA' dell'associazione diretta "Impiega"