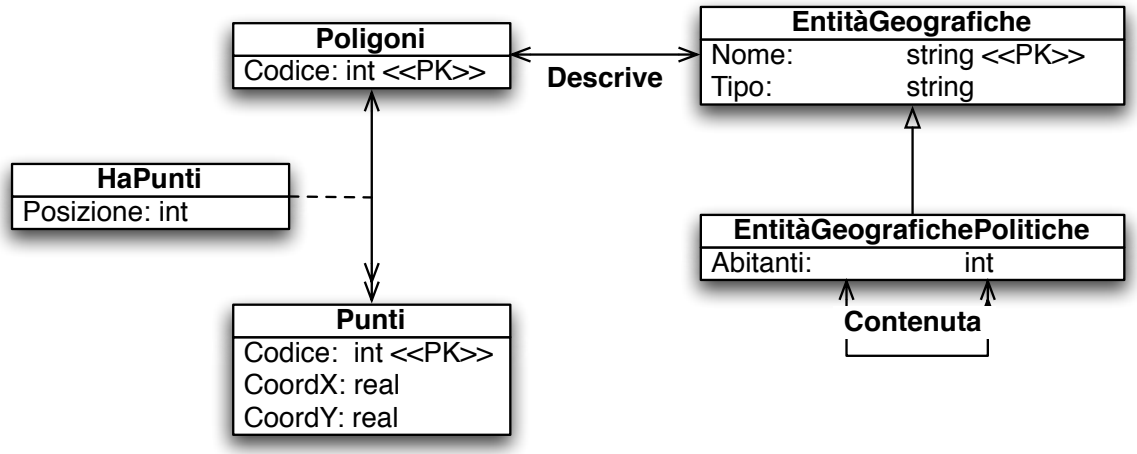
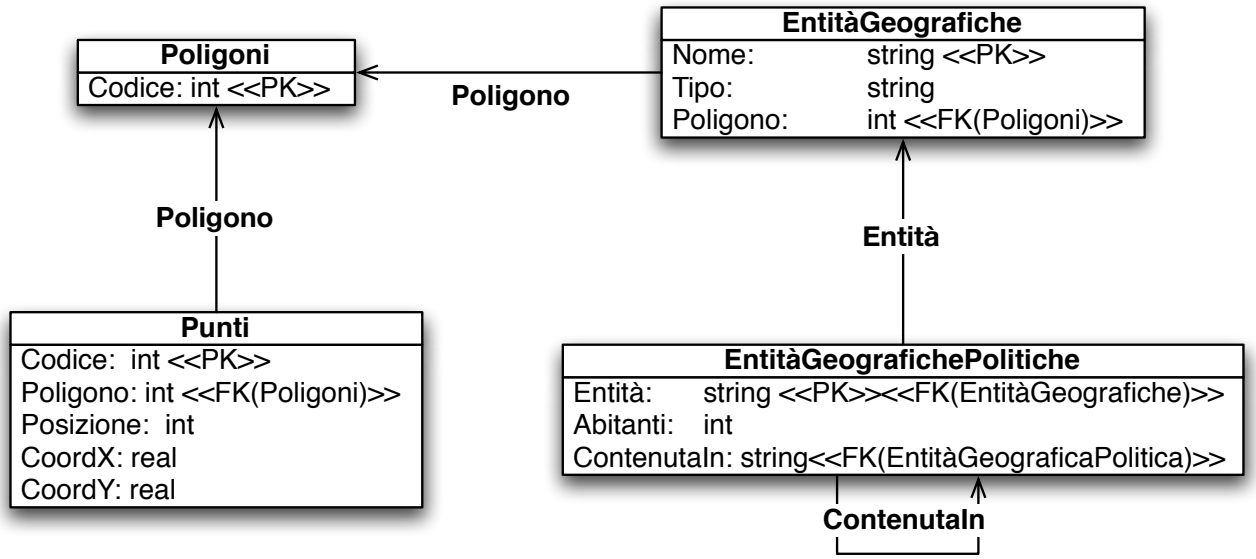
Esercizi di progetto di basi di dati con soluzione

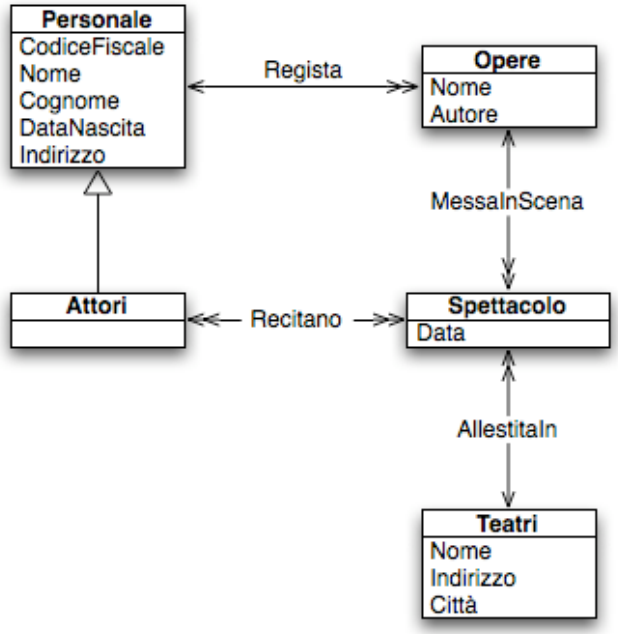
1. Si vogliono memorizzare informazioni su entità geografiche, sia fisiche che politiche. Una entità geografica ha un nome (unico), un tipo (stringa, ad es. città, lago, montagna, nazione, fiume, ecc.), un poligono (la sua estensione nello spazio). Una entità geografica politica ha un numero di abitanti e può essere contenuta all'interno di un'altra entità geografica politica. Un poligono è una lista ordinata di punti (assegnargli come chiave un codice numerico). Un punto è una coppia di coordinate x, y.

Si dia uno schema grafico a oggetti (secondo la notazione del libro di testo) della base di dati e si trasformi nello schema relazionale mostrandone la rappresentazione grafica (anche questa secondo la notazione del libro di testo, indicando la chiave primaria e le chiavi esterne).

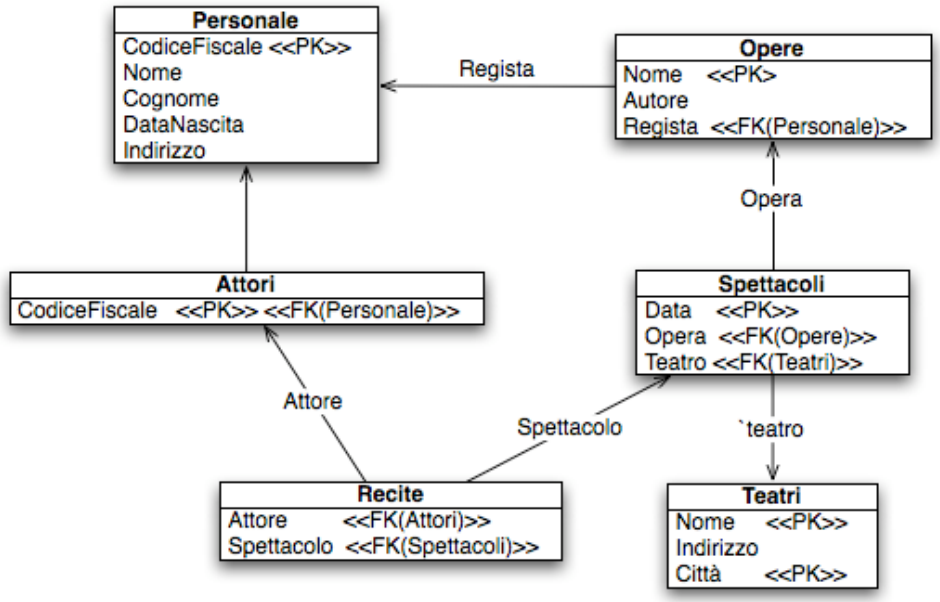




1) Si vogliono rappresentare informazioni relative ad una compagnia teatrale. Nella compagnia ci sono gli attori e altro personale. Di una persona interessa codice fiscale, nome, cognome, data di nascita, indirizzo. La compagnia ha in repertorio delle opere teatrali, di cui interessa nome, autore, il regista (che e' un membro della compagnia). La compagnia mette in scena tali opere in certi giorni in certi teatri. In un certo spettacolo messo in scena un certo giorno recitano degli attori della compagnia. Un teatro ha nome, indirizzo, città. Si dia uno schema grafico a oggetti (secondo la notazione del libro di testo) della base di dati e si trasformi nello schema relazionale mostrandone la rappresentazione grafica (anche questa secondo la notazione del libro di testo, indicando quindi la chiave primaria e le chiavi esterne).

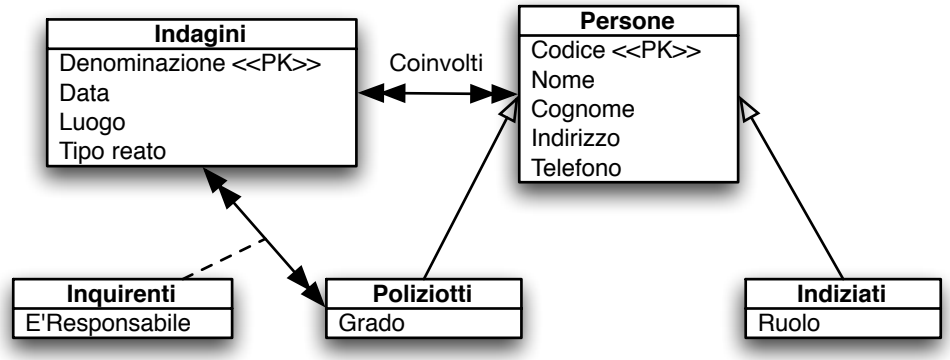


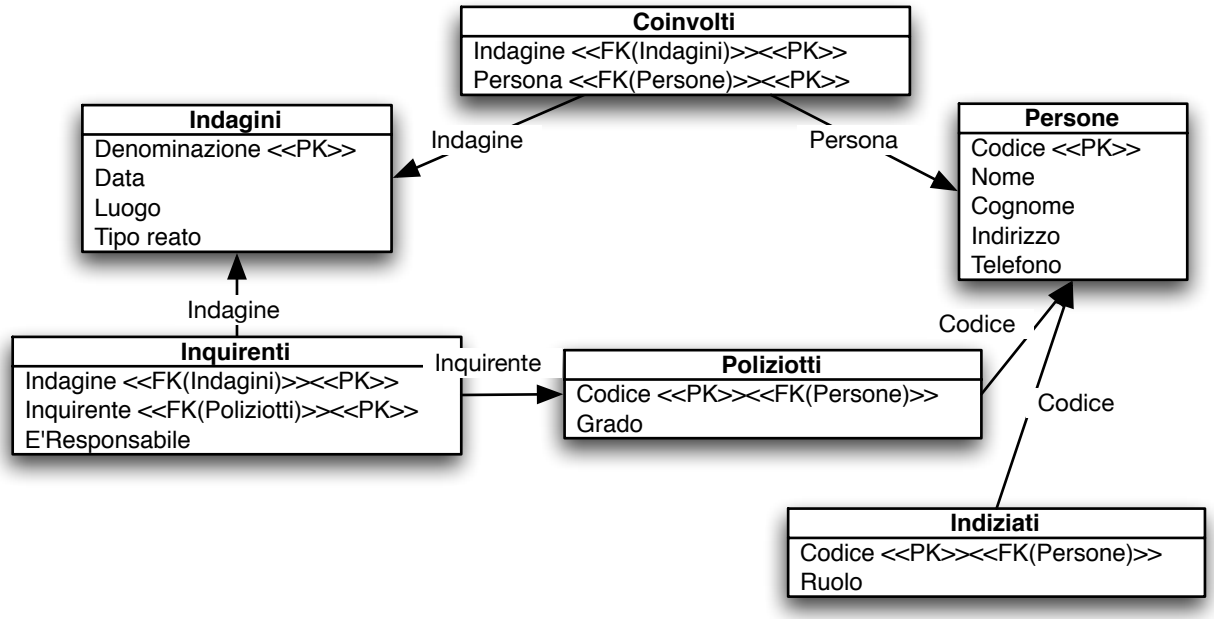
Schema relazionale



2. Si vogliono rappresentare informazioni necessarie al lavoro investigativo della polizia. Interessano le indagini su ipotesi di reato, caratterizzate da una denominazione, data (in cui sarebbe stato commesso il reato), luogo, tipo di reato. Per ogni indagine vi sono delle persone coinvolte a vario titolo (con nome, cognome, indirizzo, telefono). Alcune di queste persone sono indiziate del reato, e di queste interessa il ruolo svolto. L'indagine viene svolta da un certo numero di poliziotti (con nome, cognome, telefono, grado), uno dei quali ne è responsabile.

Si dia uno schema grafico a oggetti (secondo la notazione del libro di testo) della base di dati e si trasformi nello schema relazionale mostrandone la rappresentazione grafica (anche questa secondo la notazione del libro di testo, indicando la chiave primaria e le chiavi esterne).





ESERCIZI DI SQL

Dati i seguenti schemi di relazione:

Impiegati(Nome, Codice, AnnoNascita, NomeFiliale*)

NomeFiliale c.e. per Filiali

Filiali(Nome, Direttore*, Città, Indirizzo, Budget)

Direttore c.e. per Impiegati

scrivere in SQL le seguenti interrogazioni:

a) Trovare il nome del direttore delle filiali di "Napoli"

```
SELECT DISTINCT i.Nome
FROM Impiegati i, Filiali f
WHERE f.Direttore=i.Codice AND f.Città='Napoli'
```

b) Trovare il nome dei colleghi (cioè coloro che lavorano nella stessa filiale) del direttore di della filiale di "Palermo" (ipotizzando che ci sia un'unica filiale)

```
SELECT i.Nome
FROM Impiegati i, Filiali f
WHERE f.Città = 'Palermo' AND i.NomeFiliale=f.Nome
```

c) Trovare il nome della filiale (o delle filiali) che ha un minor numero di impiegati

```
SELECT f.Nome
FROM Filiali f, Impiegati i
WHERE i.NomeFiliale = f.Nome
GROUP BY f.Nome
HAVING COUNT(*) <=ALL (
        SELECT COUNT(*)
        FROM Filiali f, Impiegati i
        WHERE i.NomeFiliale = f.Nome
        GROUP BY f.Nome)
```

d) Per ogni filiale, dare nome del direttore, la sua età, il numero degli impiegati e la loro età media.

```
SELECT f.Nome, i.Nome, 2006-d.AnnoNascita, COUNT(*), AVG(2006-i.AnnoNascita)
FROM Filiali f, Impiegati d, Impiegati i
WHERE f.Direttore = d.Codice AND i.NomeFiliale = f.Nome
GROUP BY f.Nome, i.Nome, d.AnnoNascita
```

e) Spostare tutti gli impiegati della filiale "Napoli1" alla filiale "Napoli2"

```
UPDATE Impiegati
SET NomeFiliale = 'Napoli2'
WHERE NomeFiliale = 'Napoli1'
```

2) Dati i seguenti schemi di relazione:
Giocatore(numero, nome, squadra*)
Squadra(nome)
Portiere(numero*)
Partita(numero, data, squadraVincitrice*, squadraPerdente*)
Partecipazione(numeroGiocatore*, numeroPartita*, oraIngresso, oraUscita, goalSegnati, goalSubiti)

scrivere in SQL le seguenti interrogazioni:
a) Trovare i nomi dei giocatori della squadra "Italia" che hanno giocato almeno una partita

```
SELECT DISTINCT g.nome
FROM Giocatore g, Partecipazione p
WHERE g.squadra='Italia' AND p.numeroGiocatore=g.numero
```

b) Trovare i nomi dei giocatori della squadra "Francia" che hanno giocato con la squadra "Italia"

```
SELECT DISTINCT g.nome
FROM Giocatore g, Partecipazione p, Partita t
WHERE t.squadraVincente='Italia' AND t.squadraPerdente='Francia' AND t.numeroPartita=p.numeroPartita
AND g.numero=p.numeroGiocatore AND
g.squadra='Francia'
UNION
SELECT DISTINCT g.nome
FROM Giocatore g, Partecipazione p, Partita t
WHERE t.squadraVincente='Francia' AND t.squadraPerdente='Italia' AND t.numeroPartita=p.numeroPartita
AND g.numero=p.numeroGiocatore AND
g.squadra='Francia'
```

Una versione più semplice è la seguente:

```
SELECT DISTINCT g.nome
FROM Giocatore g, Partecipazione p, Partita t
WHERE (t.squadraVincente='Italia' OR t.squadraPerdente='Italia') AND t.numeroPartita=p.numeroPartita
AND g.numero=p.numeroGiocatore AND
g.squadra='Francia'
```

c) Trovare il nome della squadra che ha vinto il maggior numero di partite

```
CREATE VIEW Vinte (squadra, partiteVinte)
AS SELECT t.squadraVincente, COUNT(*)
FROM Partite
GROUP BY squadraVincente
SELECT squadra
FROM Vinte
WHERE partiteVinte >=ALL(SELECT partiteVinte FROM Vinte)
```

d) Per ogni giocatore, dare il nome, il numero di partite giocate, il numero di goal segnati e il numero di goal subiti.

```
SELECT g.nome, COUNT(DISTINCT p.numeroPartita), SUM(p.goalSegnati), SUM(p.goalSubiti)
FROM Partecipazione p, Giocatore g
WHERE g.numero=p.numeroGiocatore
GROUP BY g.numero, g.nome
```

2) Dato il seguente schema relazionale (le chiavi primarie sono sottolineate e con * si indicano le chiavi esterne)

Pizze(CodPizza, Nome, TempoPreparazione, Prezzo)
Ingredienti(CodIngrediente, Nome, QuantitaInMagazzino, CostoBase)
Ricette(CodPizza*, CodIngrediente*, QuantitaNecessaria)

scrivere in SQL le seguenti interrogazioni:

a) Trovare il nome di tutte le pizze che costano piu' di 8 euro e hanno un tempo di preparazione minore di 11 minuti.

```
select nome from pizze where prezzo>8 and tempoPreparazione < 11
```

b) Trovare il nome di tutti gli ingredienti della pizza più cara (o delle pizze più care)

```
create view pizzePiuCare as select codPizza from Pizze where Prezzo = (select max(Prezzo) from Pizze)
select i.nome from Ingredienti i, Ricette r, pizzePiuCare p where r.CodIngrediente = i.CodIngrediente and
r.CodPizza = p.codPizza
```

c) Trovare il codice degli ingredienti (se esistono) presenti in tutte le pizze

```
select CodIngrediente from Ricette
group by CodIngrediente
having count(*) = (select count(*) from Pizze)
```

d) Per ogni ingrediente dare il nome dell'ingrediente e il numero di pizze diverse in cui è usato

```
select i.Nome, count(*)
from Ingredienti i, Ricette r
where i.codIngrediente = r.codIngrediente
group by i.codIngrediente, i.Nome
```

e) Aumentare del 10% il costo base di tutti gli ingredienti presenti in pizze che costano meno di 5 euro.

```
update Ingredienti
set CostoBase = CostoBase * 1.1
where codIngrediente in (select r.codIngrediente from Ricette r, Pizze p where p.codPizza = r.codPizza
and Prezzo < 5)
```

2) Dato il seguente schema relazionale (le chiavi primarie sono sottolineate e con * si indicano le chiavi esterne)

Fiumi(Nome, lunghezza, montagnaSorgente*, mareFoce*)
Montagne(Nome, altezza)
Mare(Nome, superficie)

scrivere in SQL le seguenti interrogazioni:

a) Trovare il nome dei fiumi che sfociano nello stesso mare in cui sfocia il "Tagliamento".
```
select f.Nome from Fiumi f, Fiumi t where t.Nome = 'Tagliamento' and f.mareFoce = t.mareFoce
```
b) Trovare il nome del fiume più lungo che sfocia nel mare "Jonio"
```
select nome from Fiumi f where mareFoce = 'Jonio' and lunghezza = (select max(lunghezza from Fiumi where mareFoce = 'Jonio')
```
c) Trovare i nomi delle montagne da cui non nascono fiumi
```
select m.Nome from Montagne m where not exists(select * from Fiumi f where f.montagnaSorgente = m.Nome)
```
d) Per ogni coppia (montagna, mare) per cui esistono fiumi da quella montagna a quel mare dare il nome della montagna, quello del mare e il numero dei fiumi che nascono da quella montagna e sfociano in quel mare
```
select mo.Nome, ma.Nome, count(*)
from Montagne mo, Mari ma, Fiumi f
where f.montagnaSorgente = mo.Nome and f.mareFoce = ma.Nome
group by mo.Nome, ma.Nome
```
e) Abbassare l'altezza del 'Monte Bianco' di 2 metri e accorciare conseguentemente di 2 metri la lunghezza dei fiumi che nascono da questa montagna.
```
update Montagne set altezza = altezza -2 where Nome = 'Monte Bianco'
update Fiumi set lunghezza = lunghezza-2 where montagnaSorgente = 'Monte Bianco'
```

Dato il seguente schema relazionale
LibriFamosi(titolo, annoPrimaPubblicazione, città, codiceLibro)
Autori(codiceAutore, nome, annoNascita, nazionalità)
AutoriDeiLibri(codiceLibro*, codiceAutore*)

scrivere in SQL le seguenti interrogazioni:
a) trovare il titolo l'anno di prima pubblicazione e la città dei libri pubblicati a Firenze fra il 1580 e il 1610.

```
SELECT titolo, AnnoPrimaPubblicazione, città
FROM LibriFamosi
WHERE città='Firenze' AND annoNascita >=1580 AND annoNascita <= 1610
```

b) trovare nome e anno di nascita di tutti gli autori di tutti i libri pubblicati a Firenze (ogni autore deve essere elencato una volta sola).

```
SELECT DISTINCT a.Nome, a.AnnoNascita
FROM LibriFamosi l, Autori a, AutoriDeiLibri al
WHERE al.codiceLibro = l.codiceLibro AND al.codiceAutore = a.codiceAutore AND l.città = 'Firenze'
```

c) trovare per ogni autore, il numero dei libri pubblicati, l'anno di pubblicazione del primo libro e quello dell'ultimo libro, il numero di citta' diverse in cui ha pubblicato libri

```
SELECT a.nome, count(*), min(l.annoPrimaPubblicazione), max(l.annoPrimaPubblicazione), COUNT(DISTINCT l.città)
FROM LibriFamosi l, Autori a, AutoriDeiLibri al
WHERE al.codiceLibro = l.codiceLibro AND al.codiceAutore = a.codiceAutore
GROUP BY a.codice, a.nome
```

d) inserire tutte le informazioni relative al nuovo libro: "I blog e le nuove forme di scrittura", pubblicato nel 2004 a Venezia, da Mario Rossi e Giovanna Verdi (autori non presenti nella base di dati, nato uno il 1986, l'altra il 1989) (inventarsi i codici necessari)

```
INSERT INTO LibriFamosi VALUES ('I blog e le nuove forme di scrittura', 'Venezia', 500)
INSERT INTO Autori VALUES (600, 'Mario Rossi', 1986, 'Italia')
INSERT INTO Autori VALUES (601, 'Maria Verdi', 1989, 'Italia')
INSERT INTO AutoriDeiLibri VALUES (500, 600)
INSERT INTO AutoriDeiLibri VALUES (500, 601)
```

e) trovare la città con più libri e il numero di autori diversi che hanno pubblicato in essa.

```
CREATE VIEW CittàConNumLibri (città, numLibri)
AS SELECT città, COUNT(*) FROM LibriFamosi GROUP BY città

CREATE VIEW CittàConNumAutori (città, numAutori)
AS SELECT città, COUNT(DISTINCT codiceAutore)
    FROM LibriFamosi l, AutoriDiLibri al
    WHERE al.codiceLibro = l.codiceLibro
    GROUP BY città

SELECT cl.città, ca.numAutori
FROM CittàConNumLibri cl, CittàConNumAutori ca
WHERE cl.città = ca.città AND cl.numLibri = (SELECT MAX(numLibri) FROM CittàConNumLibri)
```

3) Dato il seguente schema relazionale R<(ABCDEGH), {C->BD, G->CH, EH->DG, CD->AB, E->BDG, G->A}>
a) dare una copertura canonica
C->A
C->B
C->D
E->G
G->C
G->H
b) dare almeno una chiave
E
c) portare lo schema in terza forma normale.
R1(CBDA)
R2(GCH)
R3(EG)

3) Dato il seguente schema relazionale R<(ABCDEF), {A->CF, A->D, EB->CD, EF->C, B->AD}>
a) dare una copertura canonica
A->C
A->F
A->D
EB->C
EB->D
EF->C
B->A
B->D

Prova EB ->C attributi estranei
E+ = E
B+ = BADCF (E è estraneo, sia qui che in EB->D)
Prova EF->C attributi estranei
E+ = E
F+ = F
(non ci sono attributi estranei)
Quindi:
A->C
A->F
A->D
B->C
B->D
EF->C
B->A
B->D

Prova dipendenze superflue (Si elimina subito una copia di B->D)

A->C (A+ = AFD)
A->F (A+ = ACD)
A->D (A+ = ACF)
~~B->C~~ (B+ = BDACF) Superflua
~~B->D~~ (B+ = BACFD) Superflua
EF->C (EF+ = EF)
B->A (B+ = B)

Quindi il risultato è:

A->C
A->F
A->D
EF->C
B->A

b) dare almeno una chiave non superchiave

BE+ = BEACFD

c) portare lo schema in terza forma normale.

R1(ACFD)
R2(EFC)
R3(BA)
R4(BE)