Si consideri il sistema per la registrazione degli iscritti a diversi concerti, con una relazione N-a-N tra concerti e persone:

CONCERTO (ID, CANTANTE, CITTA)
PERSONA (PCF, NOMINATIVO, DATA-NASCITA)
SPETTATORE (ID-CONCERTO, CF)

dove CANTANTE è il codice fiscale, con chiavi esterne:

SPETTATORE.CF --> PERSONA.PCF
SPETTATORE.ID-CONCERTO --> CONCERTO.ID
CONCERTO.CANTANTE --> PERSONA.PCF
(viene memorizzato nella relazione concerto il CF del cantante)

Si noti che un cantante può anche partecipare come spettatore ad un concerto.

A. Definire una query in Algebra Relazionale che restituisce i nominativi degli spettatori che hanno partecipato a concerti di almeno due cantanti (2 punti).[2]

$P_1 =$ CONCERTO $\bowtie$ SPETTATORE

$P_2 = P_1$

$\Pi_{\text{NOMINATIVO}}$ PERSONA $\bowtie$ ($P_1 \bowtie_{P_1.\text{CANTANTE} <> P_2.\text{CANTANTE AND } P_1.\text{ID} = P_2.\text{ID}} P_2$)

2. Restituisce la città in cui è avvenuto il concerto con il numero più alto di spettatori

```
CREATE VIEW Spettatori_per_concerto AS
SELECT Citta, COUNT(CF) AS Numero_spettatori
FROM Concerto C, Spettatore S
WHERE C.ID = S.Id-concerto
GROUP BY Città;

SELECT Città
FROM Spettatori_per_concerto
WHERE Numero_spettatori = (SELECT MAX(Spettatori_per_concerto)
                           FROM Spettatori_per_concerto);
```

3. Scrive una query in standard SQL che, per ogni cantante, restituisce il CF del cantante e il numero medio di spettatori ai suoi concerti.

```
SELECT Città, Cantante
FROM Spettatori_per_concerto SC, Concerto C
WHERE Sc.Città = C.Città
AND Numero_spettatori = (SELECT AVG(Spettatori_per_concerto)
                                FROM Spettatori_per_concerto);
```

```
// O' Lions
```

```
Si usa la stessa view SPETTATORI-PER-CONCERTO
definita sopra.
SELECT CANTANTE, AVG(NUM)
FROM CONCERTO, SPETTATORI-PER-CONCERTO
WHERE ID-CONCERTO=ID
GROUP BY CANTANTE
```

Esercizio 4: Decomposizione in Terza Forma Normale (5 punti)

Sia data la relazione R(A,B,C,D,E) e l'insieme di dipendenze associato F={ AB→CDE, AC→BDE, B→C, C→B, C→D, B→E}:

a. Trovare la/e chiave/i di R, motivando la risposta.

b. Calcolare la copertura ridotta

c. Effettuare una decomposizione in 3NF, indicando le chiavi delle relazioni ottenute

d. Indicare se c'è o ci può essere perdita nel join, motivando la risposta.

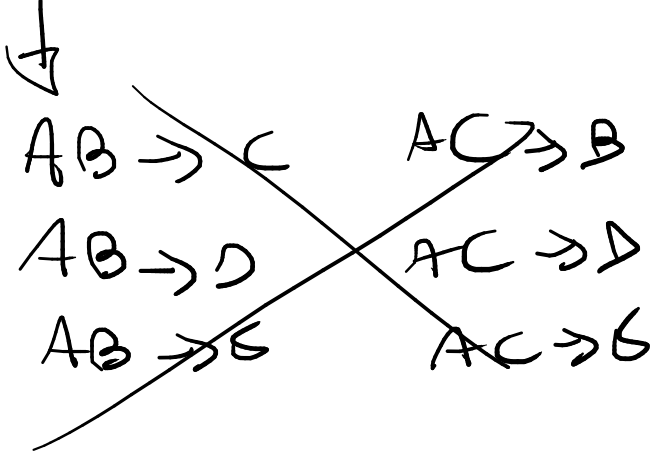
$AB^+ = \{A, B, C, D, E\}$) CHIAVI

$AC^+ = \{A, C, B, D, E\}$

$C^+ = \{C, B, D, E\}$

$B^+ = \{B, C, D, E\}$

(b) COP. RIDOTTA ?



$B \to C$

$C \to B$

$C \to D$

$B \to E$

(OCCHIO AD A

↓

RIDONDANZA CON CHIAVE ! → 3FN)

$\{B \to C,\ C \to B,\ C \to D,\ B \to E\}$

3.F.N

① INSIEMI $\{B \to C,\ B \to E\}$

$\{C \to B,\ C \to D\}$

② RELAZIONI

$R_1(\underline{B}, C, E) \to \left[\begin{matrix} F \\ B \end{matrix}\right]$ CHIAVI

$R_2(\underline{C}, B, D) \to$

③ $R_2 \subseteq R_1$ → COLLASSA (DROPS)

④ CHIAVE ESTERA RELAZIONE → AGGIUNGI $R_3(\underline{A}, B)$ $R_4(\underline{A}, C)$

(5) $R_1(\underline{B}, C, S)$

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

$R_3(\underline{A}, B)$

$R_4(\underline{A}, C)$

→ MANCA A

↓

PUÒ ESSERCI

PERDITA

NON DIN

[BCNF ⊆ 3FN]

? → SÌ → NON SI PUÒ

FARE

MEGLIO ...

Data la relazione R(A, B, C, D) con dipendenze funzionali AB→C e C→D,
indicato con |R| il numero di tuple di R(A, B, C, D).
Quale affermazione è vera
per l'operazione πA,B (R) in Algebra Relazionale?
1. πA,B (R) = |R|
2. πA,B (R) < |R|
3. πA,B (R) ≤ |R| e, in certa istanze valide di R, può accadere che πA,B (R) < |R|
4. πA,B (R) = 0

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

↑

$\pi_{A,B}(R) = R$

Si consideri il seguente log:

CK(T2), B(T5), B(T6), U(T5,O5,B5,A5), C(T5), B(T7), U(T7,O6,B6,A6), B(T8), U(T6,O1,B7,A7), A(T7), C(T6), guasto

Sapendo che occorre fare l'UNDO di T2, T7 e T8, e il REDO di T5 e T6. Quale è la prima operazione da effettuare per la ripresa a caldo?

1. O5=A5
2. O1=A7
3. O1=B7
4. O6=B6

Si consideri lo schedule S = r1(a), r2(b), w1(b), r2(a) w2 (a) r3(a) w3(a)

Indicare se S è view-serializzabile (VSR) e/o conflict-serializzabile (CSR).

1. R **è** in VSR **ma non** in CSR
2. R **non è** in VSR **ma è** in CSR
3. R **è** in VSR **ed è** in CSR
4. R **non è** in VSR **e non è** in CSR

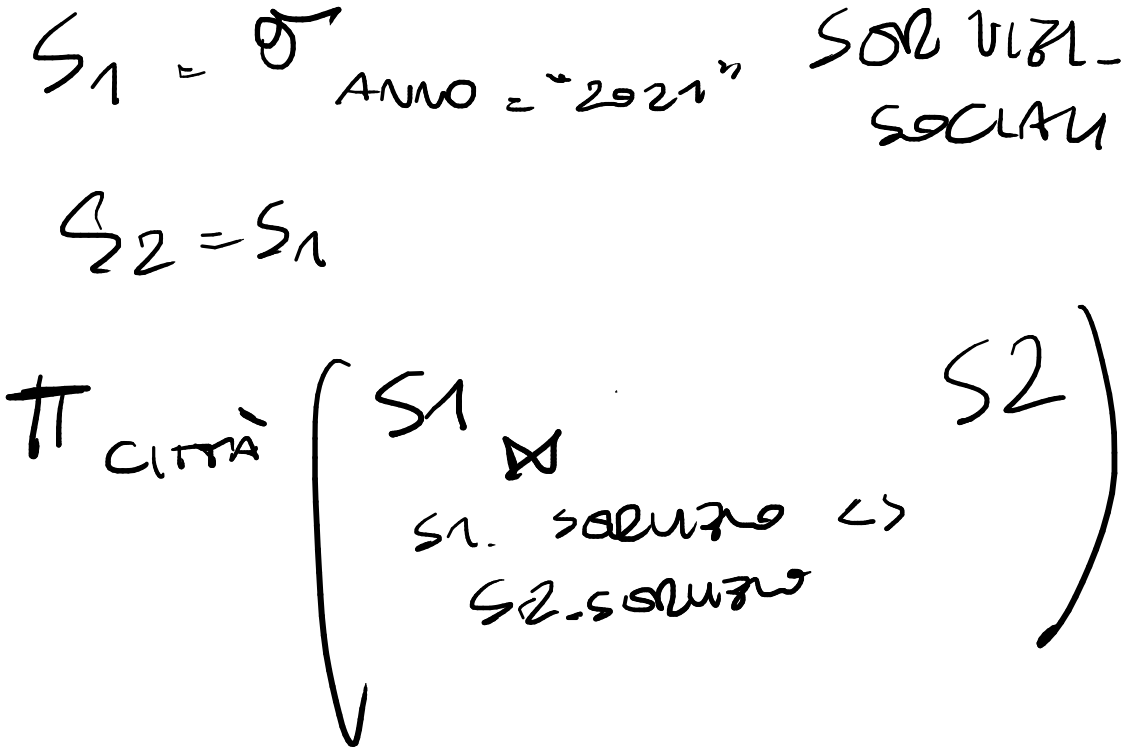
Esercizio 3: Algebra Relazionale & SQL (7 punti)

Si consideri la seguente base di dati dei diversi servizi sociali in città diverse di Italia:

- **SERVIZI_SOCIALI (Citta, Servizio, Anno, Spesa)**
- **POSIZIONE (Citta, Regione, Abitanti)**

dove non è possibile avere due città con lo stesso nome. Una tupla (Padova, Babysitting, 2019, 30000) in SERVIZI SOCIALI indica che Padova ha speso nel 2019 la cifra di 30000€ per il servizio sociale Babysitting.

A. Scrivere una query in Algebra Relazione che restituisce le città che forniscono almeno due servizi sociali nel 2021 (2 punti).[2]



2. Scrivere una query in Standard SQL che restituisce
la regione in cui c'è
la città che ha speso di meno in servizi sociali nel 2020.

```
CREATE VIEW Spesa_per_regione AS
SELECT Regione, SUM(Spesa) AS Spesa_totale
FROM Posizione P, Servizi_sociali SS
WHERE P.Citta = SS.Citta
AND Anno = "2020"
GROUP BY Regione;

SELECT Regione
FROM Spesa_per_regione
WHERE Spesa_totale = (SELECT MIN(Spesa_totale)
                      FROM Spesa_per_regione));
```

3. Nel riquadro, scrivere una query in Standard SQL che, per ogni
regione, restituisce la regione
e il numero medio dei servizi offerti
dalle città in SERVIZI_SOCIALI nel 2020

```
CREATE VIEW Numero_servizi AS
SELECT Regione, COUNT(Servizio) AS Numero
FROM Servizi_sociali SS, Posizione P
WHERE SS.Citta = P.Citta
AND Anno = "2020"
GROUP BY Regione;

SELECT Regione
FROM Numero_servizi
WHERE Numero = (SELECT AVG(Numero) FROM Numero_servizi));
```

Filmografie

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE (NomeCin, CittàCin, TitoloFilm)
CINEMA (Città, NomeCinema, #Sale, #Posti)

Selezionare le Nazionalità dei registi che hanno diretto qualche film nel 1992 ma non hanno diretto alcun film nel 1993.

```
SELECT Nazionalità
FROM Regista
WHERE Nome NOT IN (SELECT NomeRegista
                   FROM Film
                   WHERE ANNO = "1993")
AND Nome IN (SELECT NomeRegista
                   FROM Film
                   WHERE ANNO = "1992");
```

```
SELECT DISTINCT Nazionalità
FROM REGISTA
WHERE Nome IN
 (SELECT NomeRegista
  FROM FILM  WHERE Anno= '1992' )
AND Nome NOT IN
 (SELECT NomeRegista
  FROM FILM WHERE Anno= '1993' )
```

Nomi dei registi che hanno diretto nel 1993 più film di quanti ne avevano diretti nel 1992

```
SELECT Nome
FROM Regista R, Film F
WHERE R.Nome = F.NomeRegista
AND Anno = "1993"
HAVING COUNT(Titolo) > (SELECT COUNT(Titolo), Nome
                        FROM Regista R, Film F
                        WHERE R.Nome = F.NomeRegista
                        AND Anno = "1992");
```

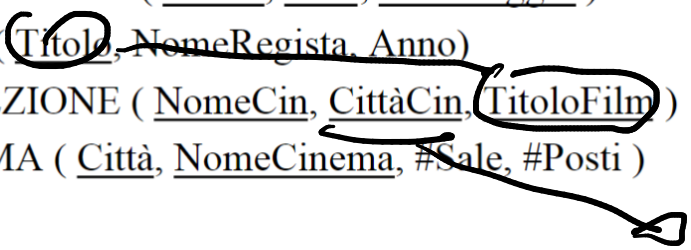
Nomi dei registi che hanno diretto nel 1993 più film di q
ne avevano diretti nel 1992: vista intermedia

```
CREATE VIEW NumPerAnno (Nom, Ann, Num) AS
SELECT NomeRegista, Anno, count(*)
FROM  FILM
GROUP BY NomeRegista, Anno

SELECT Nom AS NomeRegistaCercato
FROM NumPerAnno N1
WHERE Ann = 93 AND
        Nom NOT IN  ( SELECT Nom
                      FROM NumPerAnno N2
                      WHERE N2.Ann = 92 AND
                            N1.Num <= N2.Num )
```

Filmografie

REGISTA (Nome, DataNascita, Nazionalità)
ATTORE (Nome, DataNascita, Nazionalità)
INTERPRETA (Attore, Film, Personaggio)
FILM (Titolo, NomeRegista, Anno)
PROIEZIONE (NomeCin, CittàCin, TitoloFilm)
CINEMA (Città, NomeCinema, #Sale, #Posti)



Film proiettati nel maggior numero di cinema di Milano

```
CREATE VIEW Proiezioni_a_Milano AS
SELECT Titolo, COUNT(NomeCin) AS Film_proiettati_Milano
FROM Film F, Proiezione P
WHERE F.Titolo = P.TitoloFilm
AND CittaCin = "Milano";

SELECT Film_proiettati_Milano
FROM Proiezioni_a_Milano
WHERE Film_proiettati_Milano = (SELECT
                        MAX(Film_proiettati_Milano)
                        FROM Proiezioni_a_Milano);
```

Film proiettati nel maggior numero di cinema di Milano

```
SELECT TitoloFilm, count(*) AS NumCin
FROM  PROIEZIONE
WHERE  Città= 'Milano'
GROUP  BY  TitoloFilm
HAVING  count(*) >= ALL
   (  SELECT count(*)
      FROM  PROIEZIONE
     WHERE  Città= 'Milano'
      GROUP BY TitoloFilm)
```

NumCin non è richiesto dalla specifica, ma migliora la leggibilità

ANY

59

- Attori italiani che non hanno mai recitato con altri italiani

```
SELECT Nome
FROM Attore A1, Attore A2, Interpreta I1, Interpreta I2
WHERE A1.Nazionalità = "Italia"
AND A1.Nazionalità <> A2.Nazionalità
AND A1.Nome = I1.Attore
AND A2.Nome = A2.Attore
AND I1.Film = I2.Film;

// Slide

SELECT Nome
FROM ATTORE A1
WHERE Nazionalità = “Italiana” AND
A1.Nome not in (
SELECT I1.Attore
FROM INTERPRETA I1,INTERPRETA I2,
ATTORE A2
WHERE I1.Titolo = I2.Titolo
AND I2.Attore = A2.Nome
AND A2.Nome <> A1.Nome
AND A2.Nazionalità = “Italiana” )
```

- Trovare gli attori che hanno interpretato più personaggi in uno stesso film (+ di 1 !!)

```
SELECT DISTINCT Nome
FROM Attore A, Interpreta I1, Interpreta I2
WHERE A.Nome = I1.Attore
AND A.Nome = I2.Attore
AND I1.Film = I2.Film
AND I1.Personaggio <> I2.Personaggio;

// Slide

select distinct P1.Attore
from INTERPRETA P1 , INTERPRETA P2
where P1.Attore = P2.Attore
and P1.Film = P2.Film
and P1.Personaggio <> P2.Personaggio
```

"Anagrafe"

PERSONA(CodFis,Nome,DataNascita,
CFMadre,CFPadre)

MATRIMONIO(Codice,CFMoglie,CFMarito,
Data,NumeroInvitati)

TESTIMONI(CodiceMatr,CFTestimone)

```
Matrimoni in cui entrambi i
coniugi erano precedentemente sposati

SELECT M1.Codice
FROM Matrimonio M1, Matrimonio M2
WHERE M1.CFMoglie = M2.CFMoglie
AND M1.CFMarito = M2.CFMarito
AND M1.Data > M2.Data
AND M1.Codice <> M2.Codice;
```

```
SELECT *
FROM MATRIMONIO M
WHERE CFMoglie IN (SELECT CFMoglie
                   FROM Matrimonio M1
                   WHERE M1.CFMoglie=M.CFMoglie
                      AND M1.Data<M.Data)
   AND CFMarito IN (SELECT CFMarito
                   FROM Matrimonio M2
                   WHERE M2.CFMarito=M.CFMarito
                      AND M2.Data<M.Data)
```

```
Estrarre i matrimoni che sono nel
primo 20% per numero di invitati.

SELECT Codice
FROM Matrimonio
WHERE NumeroInvitati ≥ (SELECT (0.2 * NumeroInvitati)
                        FROM Matrimonio);
```

Schema musica

CD (CDNumber, Title, Year, Price)

Track (CDNumber,PerformanceCode, trackNo)

Recording (Performance, SongTitle, Year)

Composer (CompName, SongTitle)

Singer (SingerName, PerformanceCode)

I cantautori (persone che hanno scritto
e cantato la stessa canzone) il cui nome è ‘David’.

```
SELECT SingerName
FROM Singer S, Composer C, Recording R
WHERE S.SingerName = C.CompName
AND S.SingerName = "David"
AND C.SongTitle = R.SongTitle
AND S.PerformanceCode = R.Performance;
```

```
// Slide
```

```
SELECT SingerName
FROM ( Singer S join Recording R on
S.PerformanceCode=R.Performance )
join Composer C on R.SongTitle=C.SongTitle
WHERE SingerName=CompName
AND SingerName = ‘David’
```

2. I titoli dei dischi che contengono canzoni di cui
non si conosce l’anno di registrazione

```
SELECT Title
FROM CD, Track T, Recording R
WHERE CD.CDNumber = T.CDNumber
AND R.Performance = T.PerformanceCode
AND R.Year IS NULL;
```

3. I pezzi del disco con numero di serie 78574,
ordinati per numero progressivo,
con indicazione degli
interpreti associati

```
SELECT trackNo, SingerName
FROM Track T, Singer S
WHERE T.PerformanceCode = S.PerformanceCode
AND CDNumber = "78574"
ORDER BY ASC trackNo;
```

Gli autori che non hanno mai inciso una canzone
scritta da loro

```
SELECT CompName
FROM Composer
WHERE CompName NOT IN
(SELECT CompName
 FROM Composer AS C
   JOIN Recording AS R ON
     C.SongTitle=R.SongTiltle
   JOIN Singer ON
       Performance=PerformanceCode
 WHERE CompName=SingerName )
```

Le autorità di pubblica sicurezza periodicamente effettuano indagini per esercizi commerciali. Ogni indagine è caratterizzata da un nome e da una frequenza di occorrenza.

Come detto, ogni indagine viene periodicamente ripetuta in diverse edizioni. Ogni edizione è indentificata dall'indagine di cui è istanza e da un codice univoco tra tutte le edizioni di una stessa indagine (cioè lo stesso codice può essere solamente utilizzato per edizioni di indagini differenti). Per ogni edizione di indagine, si vuole anche memorizzare la data di inizio e di fine dell'edizione dell'indagine.

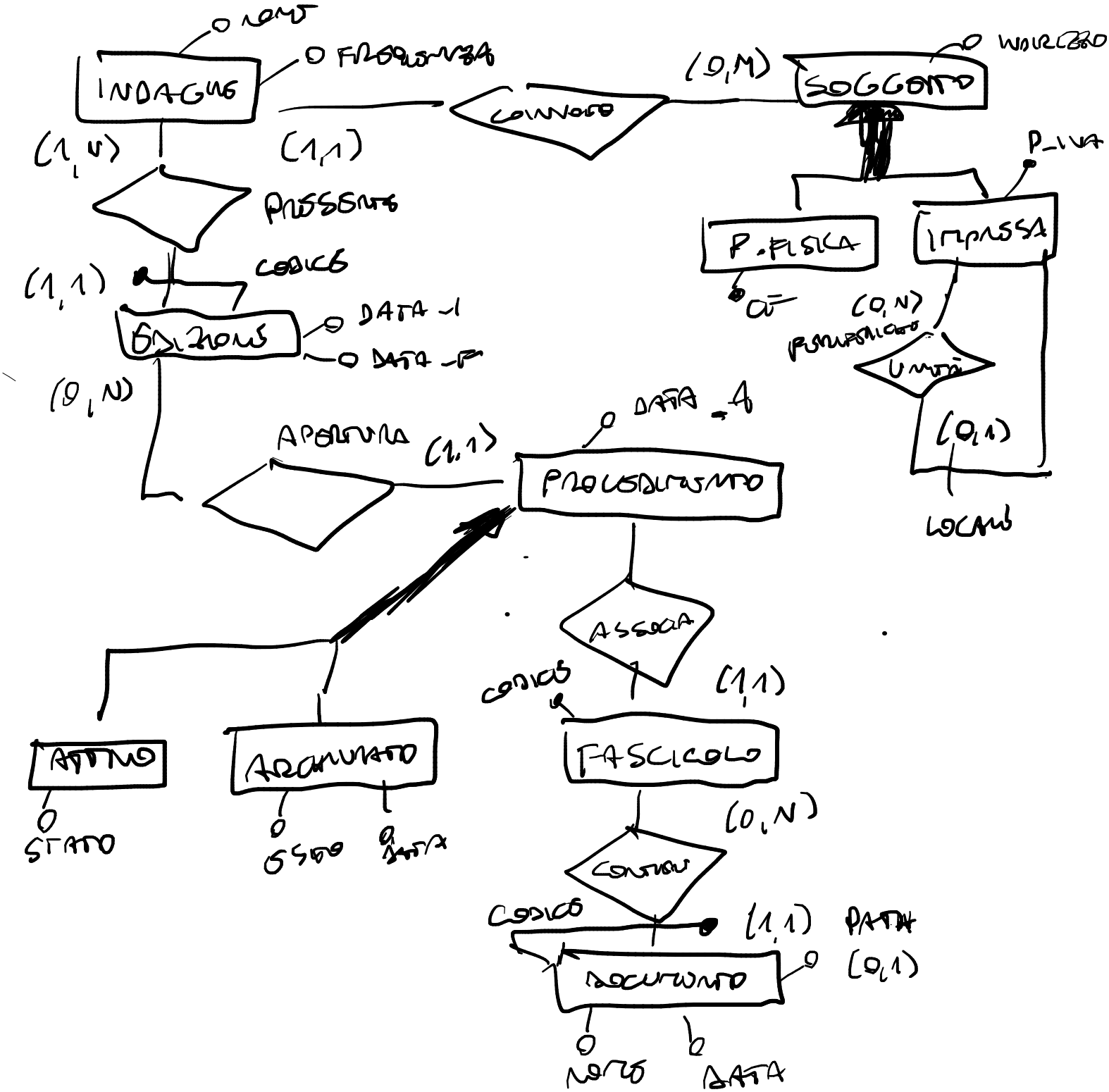
Per ogni edizione di indagine, le autorità possono sospettare delle irregolarità ed aprire un procedimento. Ogni procedimento si riferisce ad una ed un sola edizione di indagine ed è caratterizzato da una data di apertura. Ogni procedimento ha un fascicolo associato ed è di interesse conoscere il soggetto interessato dall'indagine.

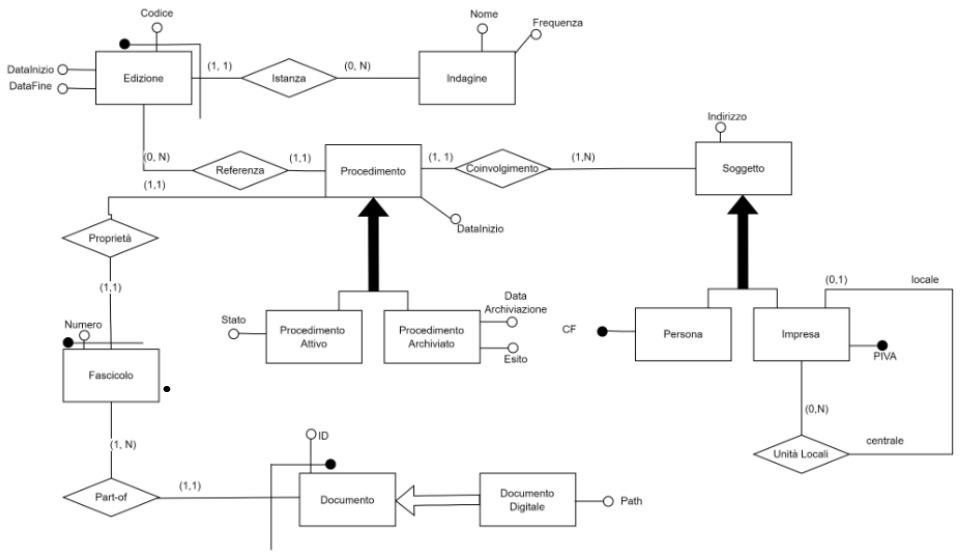
Ogni fascicolo è per un solo procedimento ed è univocamente identificato da un codice e dal procedimento a cui è associato. Un fascicolo contiene diversi documenti. Ogni documento è identificato da un codice univoco all'interno del fascicolo. Di ogni documento, è di interesse memorizzare il nome del documento, la data di produzione e, se il documento è disponibile digitalmente, il path da cui scarire il documento stesso.

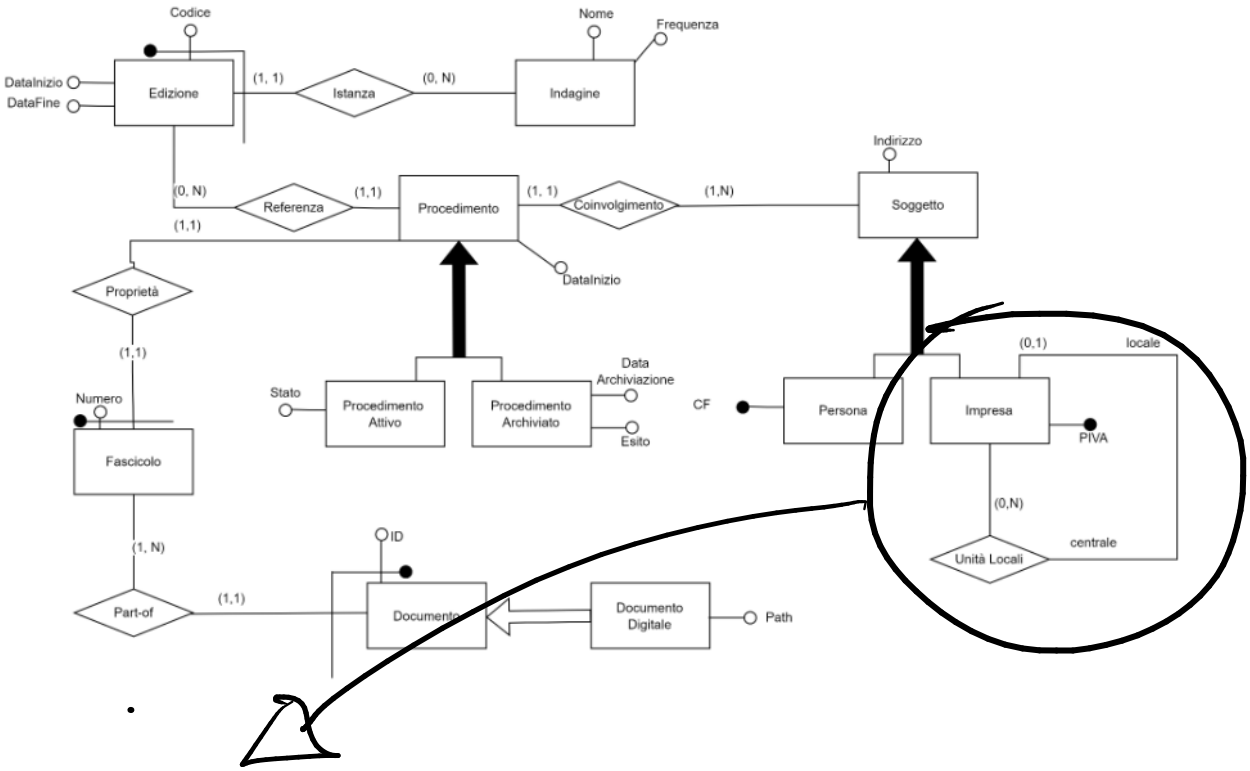
I soggetti possono essere persone fisiche o imprese. Delle persone fisiche interessa memorizzare il codice fiscale, delle imprese la partita iva. Inoltre, per entrambi, è di interesse l'indirizzo cui il soggetto è contattabile.

Si noti che le imprese possono prevedere delle unità locali, di cui si vuole mantenere informazione nella base di dati: l'impresa può essere composta di altra imprese "periferiche", che a loro volta posso comprendere ulteriori imprese perifichericheche a cascata.

Si vuole anche tenere lo storico dei procedimenti. Un procedimento può essere ancora attivo oppure archiviato a fine indagine. Quando è ancora attivo (cioè non archiviato), si vuole mantenere informazione dello stato; quando i procedimenti vengono archiviato, si vuole sapere l'esito e la data di archiviazione.


INDAGINE
COINVOLTO
(0,M)
SOGGETTO
(1,N)
(1,1)
PROSSIMO
CODICE
(1,1)
DATA_I
DATA_F
(0,N)
APERTURA
(1,1)
DATA_A
PROCEDIMENTO
P_IVA
P.FISICA
CF
(0,N)
UNITÀ
(0,1)
LOCALE
ATTIVO
STATO
ARCHIVIATO
ESITO
DATA
ASSOCIA
CODICE
(1,1)
FASCICOLO
(0,N)
CONTIENE
CODICE
(1,1)
DATA
DOCUMENTO
(0,1)
NOME
DATA


Codice
Nome
Frequenza
DataInizio
DataFine
Edizione
(1, 1)
Istanza
(0, N)
Indagine
Indirizzo
(0, N)
Referenza
(1,1)
Procedimento
(1, 1)
Coinvolgimento
(1,N)
Soggetto
(1,1)
DataInizio
Proprietà
(1,1)
Numero
Fascicolo
Stato
Procedimento Attivo
Procedimento Archiviato
Data Archiviazione
Esito
CF
Persona
Impresa
(0,1)
locale
PIVA
(0,N)
Unità Locali
centrale
(1, N)
ID
Part-of
(1,1)
Documento
Documento Digitale
Path



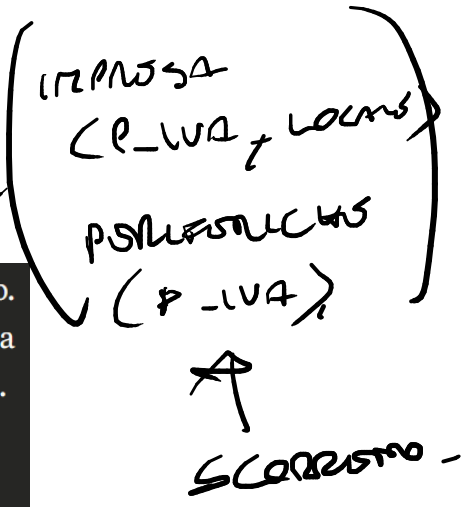
Soluzione nella traduzione logica:

Nello schema relazionale `IMPRESA`, si introduce un attributo `CodiceCentrale` che referenzia la chiave primaria della stessa tabella:

```
IMPRESA(Codice, CodiceCentrale)
- IMPRESA.CodiceCentrale → IMPRESA.Codice
```

Questa foreign key autoreferenziale permette di modellare la gerarchia:

- Per l'impresa centrale: `CodiceCentrale` = NULL (o auto-referenziamento)
- Per le imprese periferiche: `CodiceCentrale` contiene il codice dell'impresa madre



No, il tuo approccio sarebbe **scorretto** dal punto di vista metodologico. Analizzando il materiale del corso, emerge chiaramente che creare una tabella separata `Unità_locali` non risolve correttamente la ricorsione.

Problemi del tuo approccio:

1. **Violazione della normalizzazione**: Avresti ridondanza strutturale - ogni impresa periferica esisterebbe sia in `IMPRESA` che in `Unità_locali`
2. **Complessità delle query**: Per ricostruire la gerarchia completa dovresti fare join complessi tra le due tabelle
3. **Perdita di informazione relazionale**: Non riusciresti a modellare facilmente gerarchie a più livelli (periferiche di periferiche)