Studente: Magliolo Leonardo Matr:844910

Anno: 2020/2021

Sistemi informativi: Relazione progettuale

1. Consegna progettuale:

Descrizione.

Il Ministero della Salute vuole tracciare le vendite dei farmaci che avvengono a seguito di prescrizioni mediche. L'obiettivo è monitorare la vendita dei farmaci generici e il costo a carico del SSN e dei singoli cittadini.

Quando il paziente si reca dal medico (di base o specialista) per una visita, il medico può stabilire la somministrazione di uno o più farmaci. La prescrizione di un farmaco avviene quando il medico compila una "ricetta", la quale riporta la data di emissione della ricetta, il principio attivo, e la terapia, ovvero la dose giornaliera e il numero di giorni della terapia. Sulla base delle patologie del paziente, il medico può anche indicare sulla prescrizione un'esenzione. Per semplicità, ogni prescrizione può riferirsi ad un singolo principio attivo e conseguente terapia. Ovviamente, il medico può emettere più prescrizioni per uno stesso paziente, anche nello stesso giorno.

Il paziente si reca in farmacia per acquistare un farmaco per ogni prescrizione in suo possesso. La farmacia verifica che la prescrizione non sia già stata registrata come espletata, e in caso negativo vende al paziente un farmaco che contenga lo stesso principio attivo indicato sulla prescrizione. Il farmaco può essere generico oppure originale ("di marca") a seconda dei desideri del paziente. Se il paziente opta per il farmaco generico, la farmacia consegna il farmaco attribuendo il costo del farmaco interamente al SSN. Altrimenti, se il paziente opta per il farmaco originale, il paziente è tenuto a pagare una quota fissa (il ticket) che è legata al farmaco stesso, mentre la restante parte è a carico del SSN. Se però la prescrizione riporta un'esenzione, allora il paziente non paga nulla neanche nel caso di farmaco originale. La farmacia traccia dunque quante confezioni sono vendute per soddisfare ogni singola prescrizione, tracciando la data di vendita, il prezzo unitario per confezione e il costo attribuito al SSN e al paziente (eventualmente nullo). La prescrizione viene quindi marcata come espletata.

Oltre alla vendita dei farmaci, il SSN vuole anche tracciare:

1) un'anagrafica dei principi attivi, per i quali oltre ad una descrizione sarà indicata la famiglia a cui appartengono;

2) un'anagrafica dei farmaci: nome del farmaco e produttore, quale principio attivo usa (per semplicità un principio attivo per farmaco) e le sue modalità di somministrazione (es. via orale con compresse);

2) un'anagrafica dei cittadini iscritti al servizio sanitario: nome, cognome, codice fiscale, anno di nascita, città di residenza e di nascita, indirizzo Per i cittadini che sono anche pazienti (cioè per i quali esiste almeno una prescrizione) è necessario tracciare peso e altezza. Per i medici, invece, è necessario tracciare la loro specializzazione.

Cosa dovete fare.

Definire uno schema concettuale con i Class Diagram UML in grado di mantenere tutte le informazioni e le relazioni rilevanti per il dominio illustrato.

Definire il processo BPMN che descrive i passi necessari per l'acquisto di un farmaco, dalla visita dal medico alla vendita in farmacia.

Tradurre lo schema UML in schema relazionale e definire un DFM individuando un fatto di interesse a rispondere alle seguenti domande OLAP:

a) Qual è l'andamento dei costi per il SSN mensilmente nell'arco degli ultimi 5 anni

b) Qual è l'andamento dei costi per il SSN mensilmente nell'arco degli ultimi 5 anni suddiviso per categorie di pazienti: giovani (età < 20); adulti (età < 65); anziani (età =80)

c) Qual è la categoria di pazienti più disposta a pagare il ticket pur di avere il farmaco originale

d) Qual è il prezzo dei farmaci originali che cadono nella famiglia "antibiotici" prescritti nel 2020 Per soddisfare le quattro domande qui sopra occorre tradurre il DFM in uno Star Schema e definire query SQL (possibilmente usando i costrutti che lo estendono visti a lezione).

2. Rappresentazione dei processi:

2.1 Individuazione processi CRASO:

Processo di formulazione diagnosi ed eventuale prescrizione:

Client:

primario: Paziente iscritto al SSN

secondario: Organizzazione SSN

Request: Paziente contatta medico

Activities: Diagnosi ed eventuale prescrizione di farmaci, raccolta dati

organiSation: Paziente, Medico

Output:

primario: Consegna diagnosi e/o prescrizione paziente

secondario: Raccolta dati ricette in DB transazionale SSN

Processo vendita farmaci:

Client:

primario: Paziente iscritto al SSN

secondario: Organizzazione SSN

Request: Paziente presenta ricette al medico

Activities: Vendita farmaci, raccolta dati

organiSation: Paziente, Farmacista

Output:

primario: Consegna farmaci al paziente

secondario: Raccolta dati vendite in DB transazionale SSN

Processo ottenimento cure:

Client: Paziente iscritto al SSN

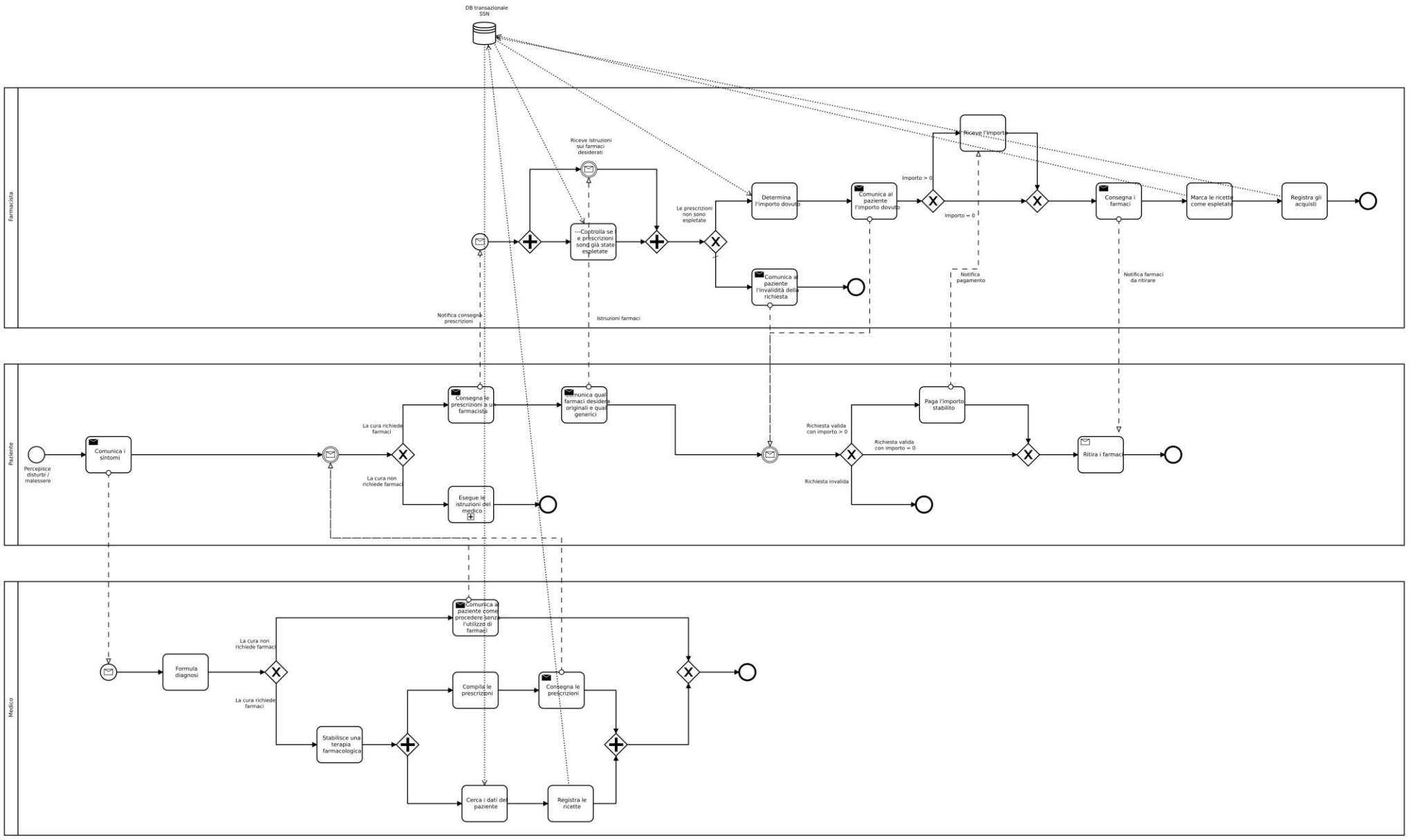
Request: Paziente avverte fastidi e/o malori

Activities: Comunicazione sintomi a un medico, Acquisto farmaci (opzionale)

organiSation: Paziente, Medico, Farmacista (opzionale)

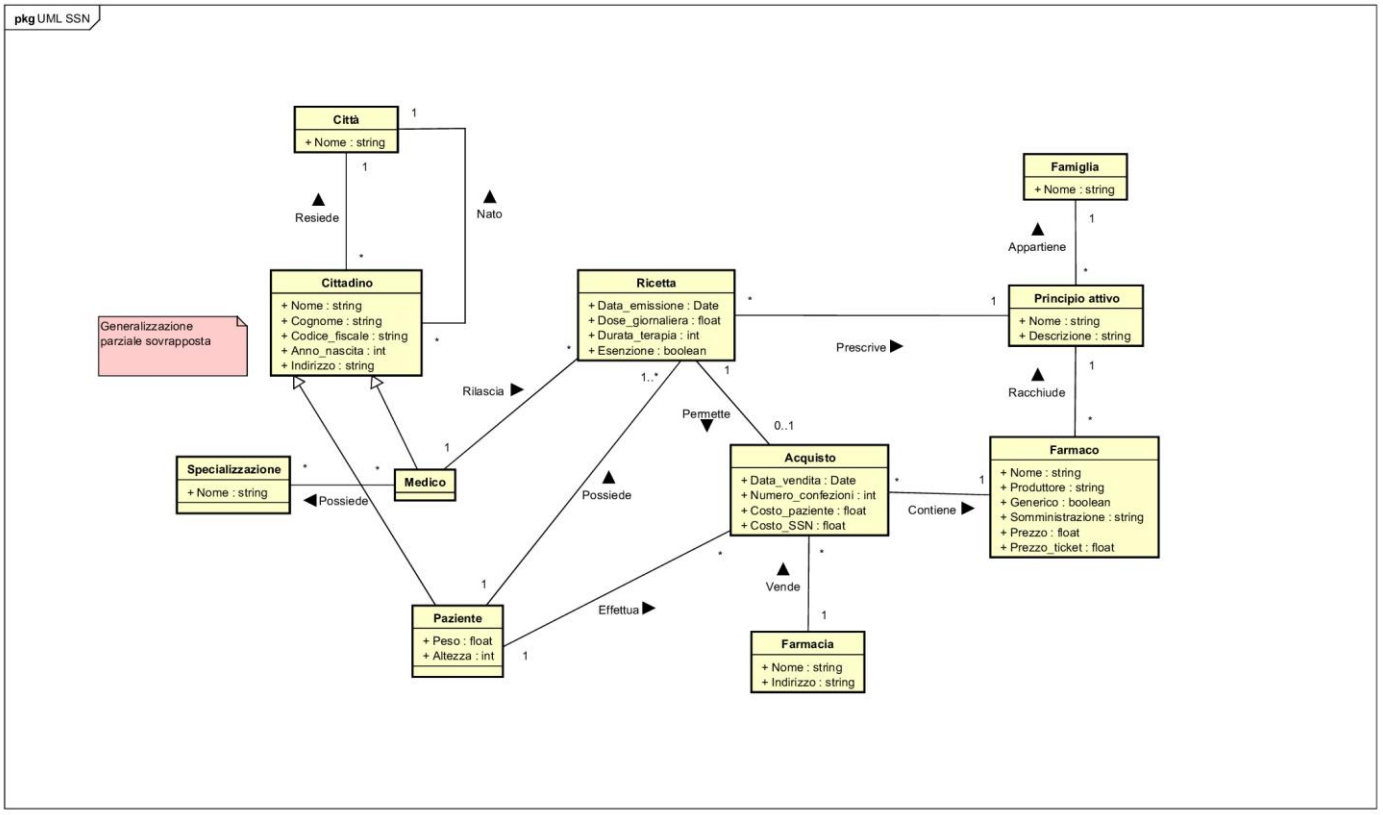
Output: Ottenimento cure

2.2 Schema BPMN:


DB transazionale SSN
Farmacista
Riceve istruzioni sui farmaci desiderati
Riceve l'importo
Determina l'importo dovuto
Comunica al paziente l'importo dovuto
Importo > 0
Importo = 0
Consegna i farmaci
Marca le ricette come espletate
Registra gli acquisti
Controlla se e prescrizioni sono già state espletate
Le prescrizioni non sono espletate
Comunica al paziente l'invalidità della richiesta
Notifica pagamento
Notifica farmaci da ritirare
Notifica consegna prescrizioni
Istruzioni farmaci
Paziente
Percepisce disturbi / malessere
Comunica i sintomi
Consegna le prescrizioni a un farmacista
Comunica quali farmaci desidera originali e quali generici
La cura richiede farmaci
La cura non richiede farmaci
Esegue le istruzioni del medico
Paga l'importo stabilito
Richiesta valida con importo > 0
Richiesta valida con importo = 0
Richiesta invalida
Ritira i farmaci
Medico
Comunica al paziente come procedere senza l'utilizzo di farmaci
La cura non richiede farmaci
La cura richiede farmaci
Formula diagnosi
Compila le prescrizioni
Consegna le prescrizioni
Stabilisce una terapia farmacologica
Cerca i dati del paziente
Registra le ricette

3. Progettazione database transazionale:

3.1 Schema UML:

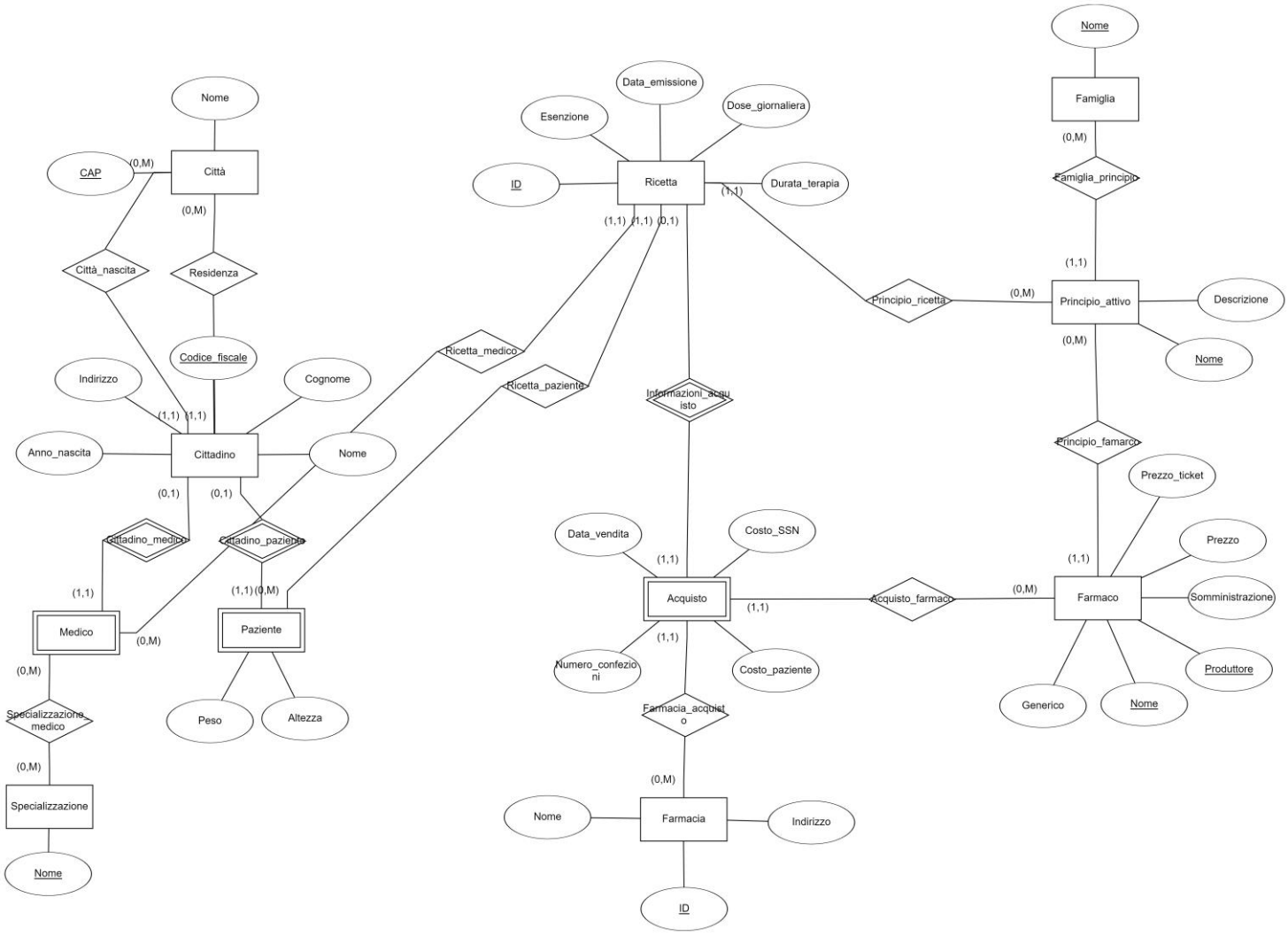


In previsione di eventuali variazioni nel tempo dei prezzi associati ai farmaci è stato scelto di memorizzare gli attributi ‘Costo_paziente’ e ‘Costo_SSN’ per potersi ricondurre al costo del singolo acquisto. In oltre, così facendo, è possibile rappresentare ‘Costo_totale’ come somma di ‘Costo_paziente’ e ‘Costo_SSN’ senza introduzione di ridondanze.

Per supportare un’anagrafica dei cittadini iscritti al SSN si è scelto di modellare la generalizzazione di ‘cittadino’ come parziale e sovrapposta, così che possano esistere individui non pazienti e non medici sui quali comunque è possibile raccogliere dati e simultaneamente che possano esistere medici in cura presso il SSN stesso.

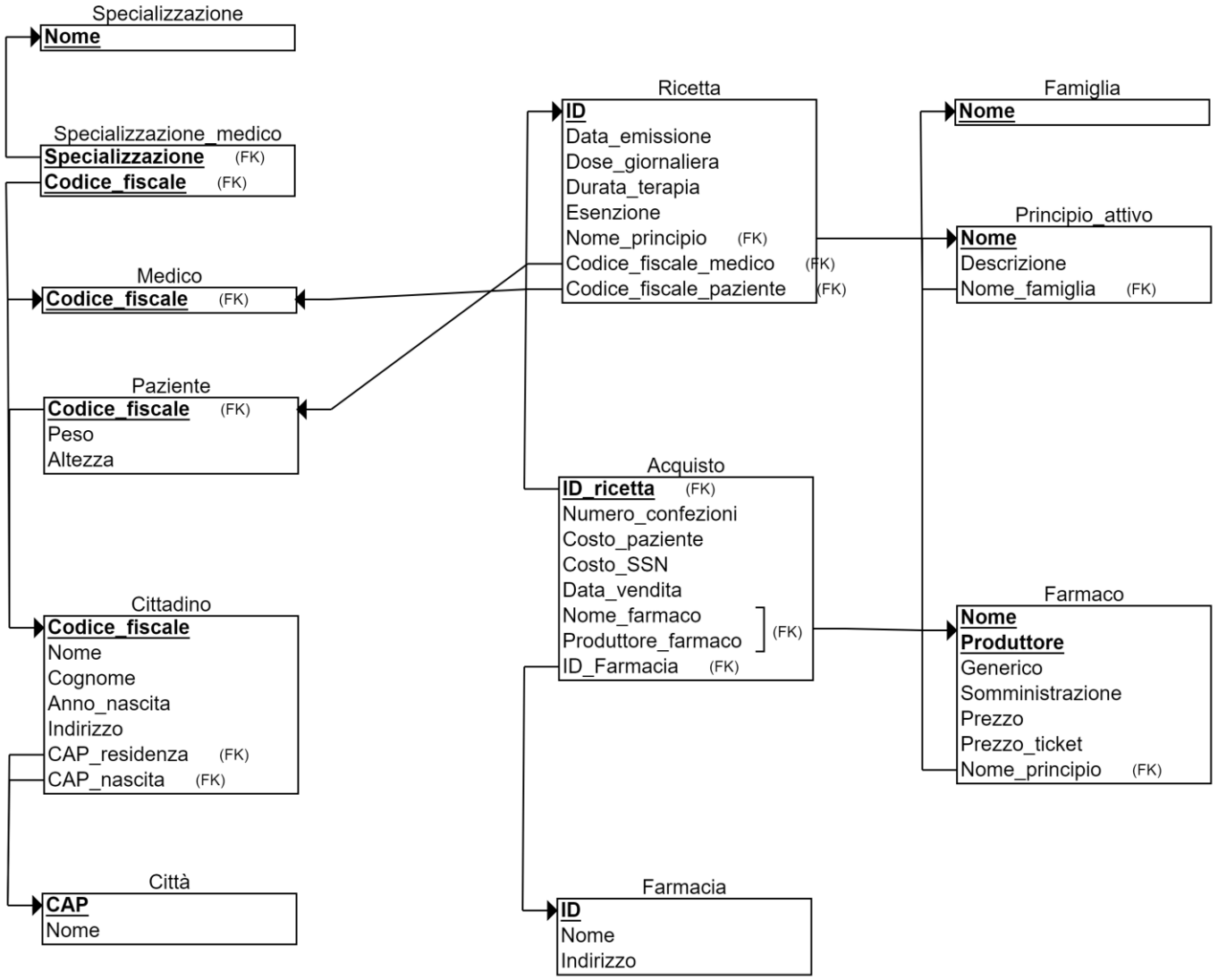
E’stato scelto di modellare l’associazione ‘possiede’ che coinvolge ‘medico’ e ‘specializzazione’ come da schema per garantire l’esistenza dei medici che non dispongono di specializzazione (di base) e simultaneamente permettere a un singolo medico di possedere più di una specializzazione.

3.2 Schema E-R + business rules:



- Qualora a un'istanza di 'Acquisto' fosse associata una 'Ricetta' che presenta il valore 'Esenzione' = true ciò implicherebbe che 'Costo_paziente' necessariamente debba essere uguale a 0
- Tutti gli attributi che nel loro nome contengono "costo", "prezzo", "peso" e "altezza" devono assumere valori >= 0
- L'attributo 'Prezzo_ticket' deve essere sempre <= rispetto a 'Prezzo'
- Qualora un'istanza di 'Farmaco' possedesse valore di 'Generico' = true allora ciò implicherebbe che 'Costo_ticket' necessariamente debba essere uguale a 0
- Devono essere effettuati controlli di consistenza sui valori degli attributi 'Indirizzo', 'Codice fiscale' e 'CAP'
- Tutti gli attributi che nel loro nome contengono "data" non devono possedere valori associati a giorni futuri la data d'inserimento dell'istanza
- Qualora a un'istanza di 'Acquisto' fosse associata una 'Ricetta' ciò implicherebbe che 'Data_emissione' della ricetta necessariamente debba essere uguale a 'Data_vendita'
- Non è possibile che una ricetta sia associata indirettamente alla stessa istanza di 'cittadino' sia in 'Ricetta_medico' che in 'Ricetta_paziente', in altri termini: un medico non può effettuare prescrizioni per sé stesso

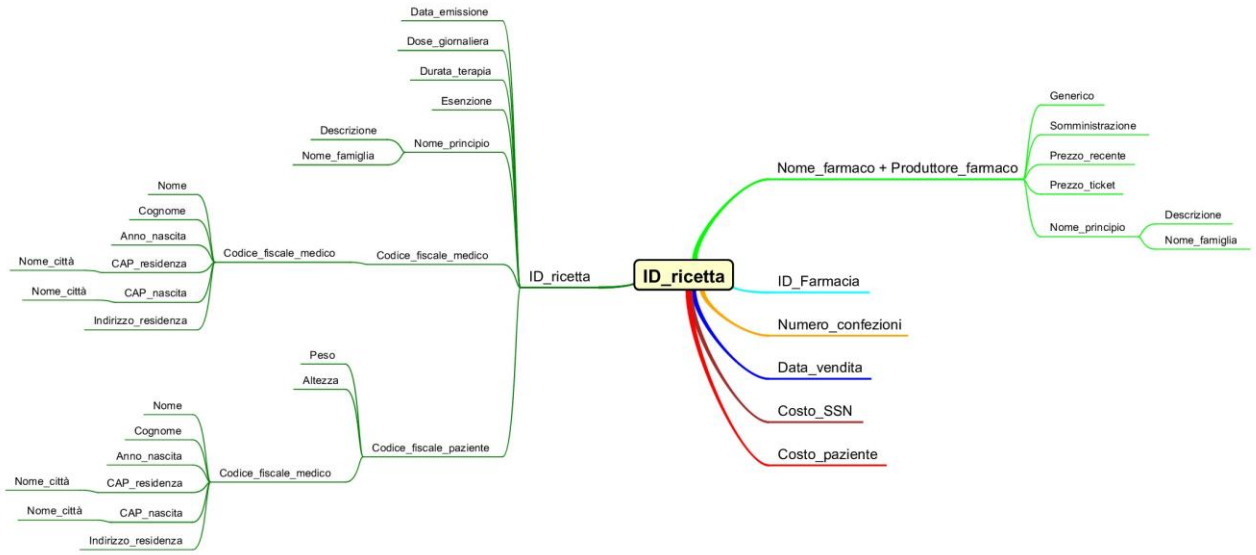
3.3 Schema relazionale:



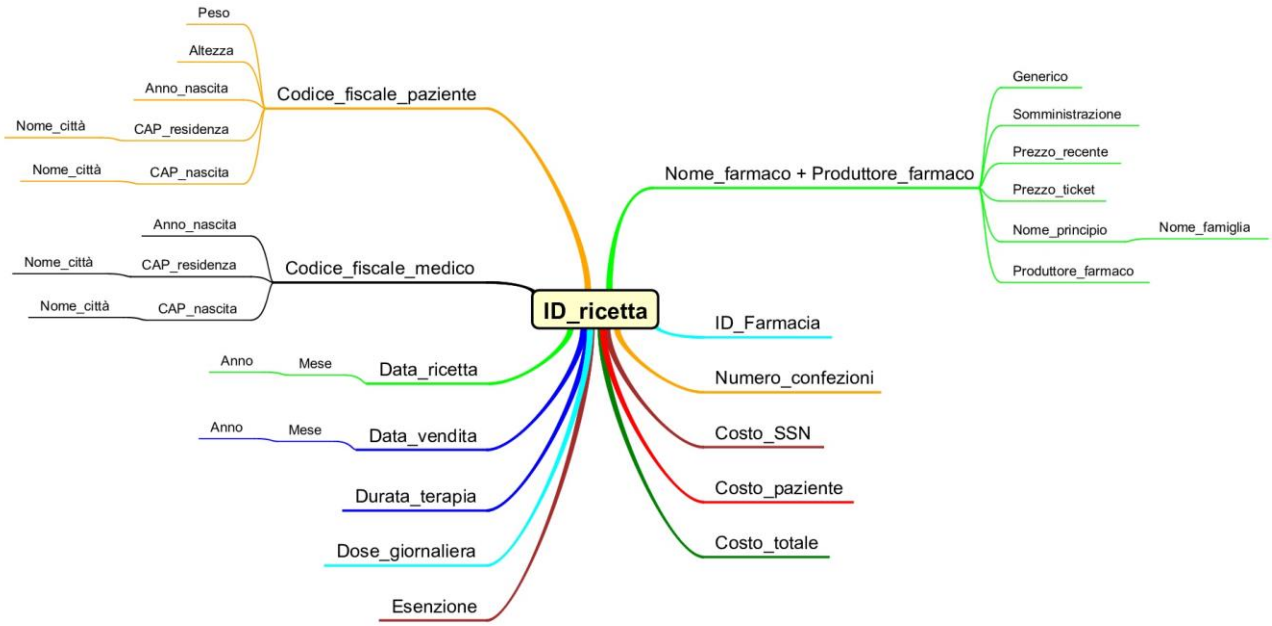
E' stato scelto di selezionare gli attributi 'Nome' e 'Produttore' come chiave della relazione 'Farmaco' in quanto per i farmaci generici identificare l'istanza di farmaco per mezzo del solo 'Nome' potrebbe comportare problemi.

4. Progettazione datawarehouse:

4.1 Albero degli attributi iniziale:



4.2 Albero degli attributi rivisto:

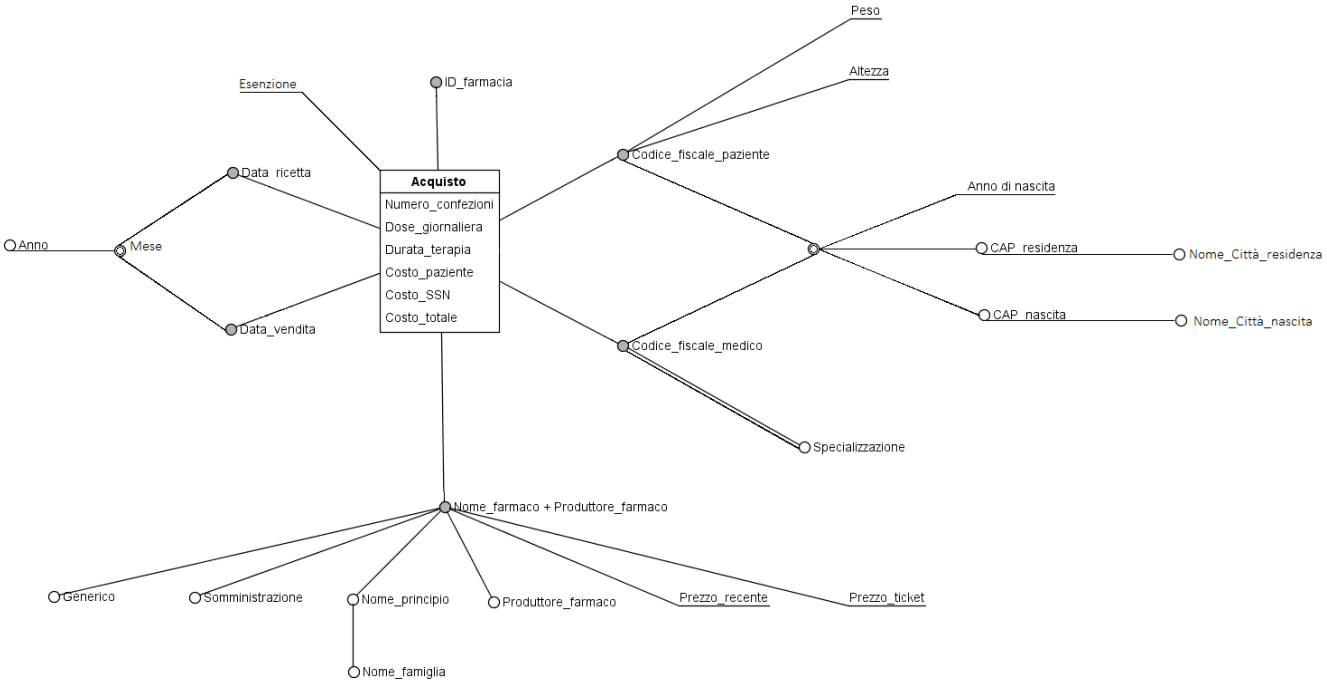


Sono state eseguire operazioni di grafting sugli attributi ‘ID_ricetta’, ‘Codice_fiscale_medico’ e ‘Codice_fiscale_paziente’ per evitare ridondanze all’interno dell’albero e renderlo più leggibile. Sono state eseguite delle operazioni di pruning per eliminare attributi ripetuti appartenenti a rami differenti e scartare attributi non necessari alla rappresentazione della datawarehouse.

Sono stati inseriti attributi aggiuntivi ricavabili dalla base di dati transazionale per facilitare l’aggregazione su schema DFM:

- ‘Costo_totale’ come somma di ‘Costo_paziente’ e ‘Costo_SSN’
- ‘Mese’ e ‘Anno’ come scomposizione mm/yyyy e yyyy delle date rappresentate in dd/mm/yyyy

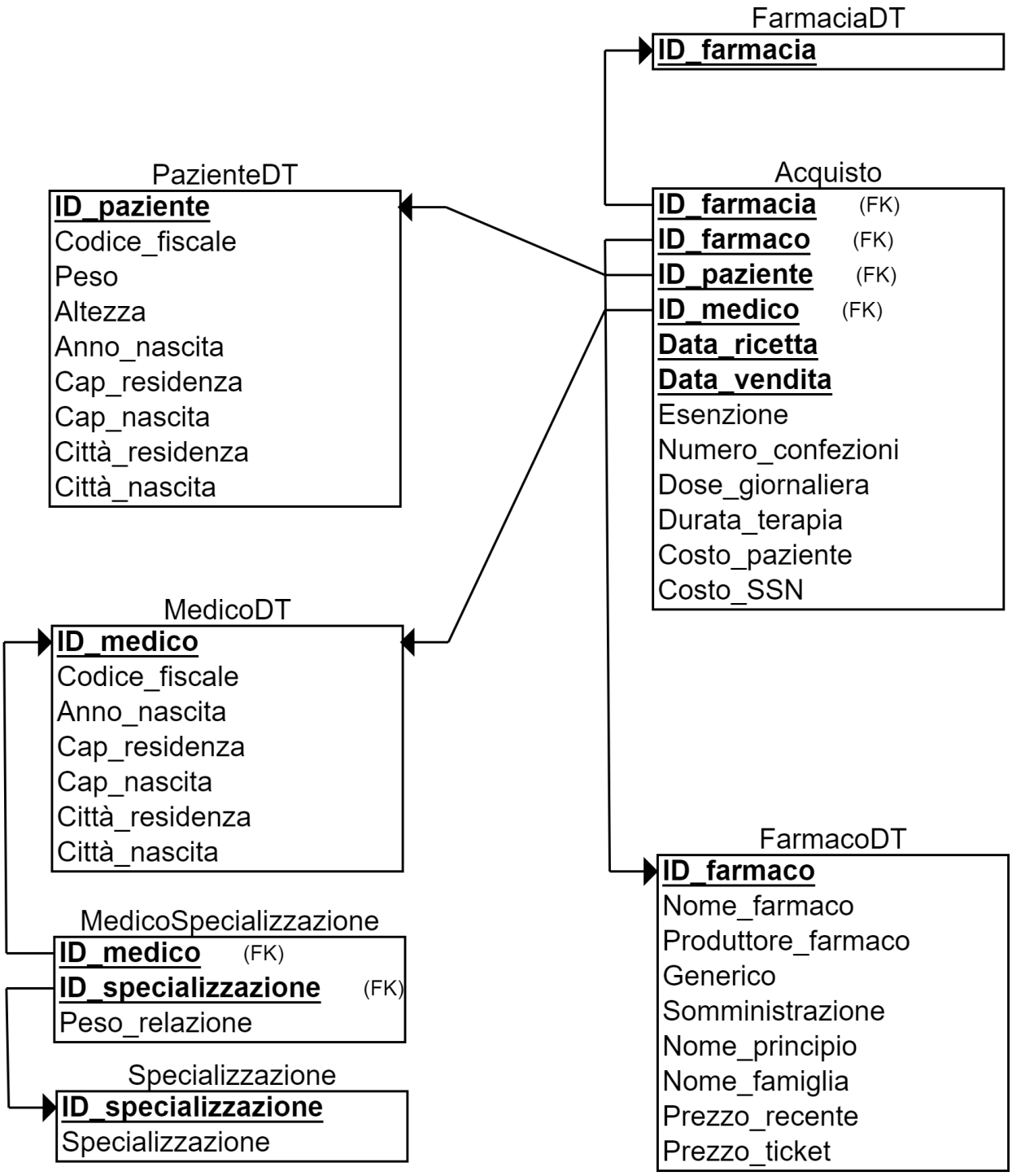
4.3 Schema DFM:



Sono state introdotte gerarchie condivise tra <'Data_ricetta', 'Data_vendita'> e <'Codice_fiscale_paziente', 'Codice_fiscale_medico'> per migliorare la leggibilità del DFM.

E' stato aggiunto un arco multiplo per modellare correttamente la relazione molti a molti rappresentata all'interno dell'E-R tra 'medico' e 'specializzazione'.

4.4 Star Schema:



Considerata l'indipendenza che vi è tra DFM e star schema, sebbene il DFM sia stato arricchito con aggiunte a gerarchie dimensionali ('Mese' e 'Anno') e misure ('Costo_totale') è stato scelto di non riportare queste aggiunte all'interno dello star schema. Il costo computazionale per ricavare in SQL gli elementi aggiuntivi inseriti all'interno del DFM risulta irrisorio rispetto alla quantità di byte impiegati per rappresentarli, vengono

inoltre semplificate le operazioni di ETL.
Dalla rimozione di 'Mese' e 'Anno' ne consegue che 'Data_vendita' e 'Data_ricetta' vengono trattate come dimensioni degeneri. Considerato il peso in byte del tipo Date (4 byte), equiparabile a quello di un intero potenzialmente usato per indirizzare la chiave esterna di una junktable, si sceglie di incorporare i due attributi all'interno del fatto stesso evitando peraltro l'esecuzione di join . La cardinalità di un'eventuale junktable sui due attributi Date rende sfavorevole la sua implementazione.

Per affrontare l'evoluzione della data warehouse nel tempo, considerati i requisiti di progetto, è stata scelta l'implementazione di una gerarchia dinamica di tipo 1 per rappresentare lo scenario 'oggi per ieri'.

4.5 PostgreSQL DDL Star Schema:

```
CREATE TABLE FarmaciaDT

(
  ID_farmacia INT NOT NULL,
  PRIMARY KEY (ID_farmacia)
);

CREATE TABLE PazienteDT
(
  ID_paziente INT NOT NULL,
  Codice_fiscale CHAR(16) NOT NULL,
  Peso NUMERIC(4, 1) NOT NULL,
  Altezza NUMERIC(3, 0) NOT NULL,
  Anno_nascita NUMERIC(4, 0) NOT NULL,
  Cap_residenza NUMERIC(5, 0) NOT NULL,
  Cap_nascita NUMERIC(5, 0) NOT NULL,
  Città_residenza VARCHAR(35) NOT NULL,
  Città_nascita VARCHAR(35) NOT NULL,
  PRIMARY KEY (ID_paziente)
);

CREATE TABLE MedicoDT
(
  ID_medico INT NOT NULL,
  Codice_fiscale CHAR(16) NOT NULL,
  Anno_nascita NUMERIC(4, 0) NOT NULL,
  Cap_residenza NUMERIC(5, 0) NOT NULL,
  Cap_nascita NUMERIC(5, 0) NOT NULL,
  Città_residenza VARCHAR(35) NOT NULL,
  Città_nascita VARCHAR(35) NOT NULL,
  PRIMARY KEY (ID_medico)
);

CREATE TABLE FarmacoDT
(
  ID_farmaco INT NOT NULL,
  Nome_farmaco VARCHAR(35) NOT NULL,
  Produttore_farmaco VARCHAR(35) NOT NULL,
```

```
  Generico boolean NOT NULL,
  Somministrazione INT NOT NULL,
  Nome_principio VARCHAR(40) NOT NULL,
  Nome_famiglia VARCHAR(40) NOT NULL,
  Prezzo_recente NUMERIC(6, 2) NOT NULL,
  Prezzo_ticket NUMERIC(6, 2) NOT NULL,
  PRIMARY KEY (ID_farmaco)
);

CREATE TABLE Specializzazione
(
  ID_specializzazione INT NOT NULL,
  Specializzazione VARCHAR(30) NOT NULL,
  PRIMARY KEY (ID_specializzazione)
);

CREATE TABLE Acquisto
(
  Esenzione boolean NOT NULL,
  Numero_confezioni INT NOT NULL,
  Dose_giornaliera FLOAT NOT NULL,
  Durata_terapia INT NOT NULL,
  Costo_paziente NUMERIC(6, 2) NOT NULL,
  Costo_SSN NUMERIC(6, 2) NOT NULL,
  Data_ricetta DATE NOT NULL,
  Data_vendita DATE NOT NULL,
  ID_farmacia INT NOT NULL,
  ID_farmaco INT NOT NULL,
  ID_paziente INT NOT NULL,
  ID_medico INT NOT NULL,
  PRIMARY KEY (Data_ricetta, Data_vendita, ID_farmacia, ID_farmaco, ID_paziente, ID_medico),
  FOREIGN KEY (ID_farmacia) REFERENCES FarmaciaDT(ID_farmacia),
  FOREIGN KEY (ID_farmaco) REFERENCES FarmacoDT(ID_farmaco),
  FOREIGN KEY (ID_paziente) REFERENCES PazienteDT(ID_paziente),
  FOREIGN KEY (ID_medico) REFERENCES MedicoDT(ID_medico)
);

CREATE TABLE MedicoSpecializzazione
(
  Peso_relazione NUMERIC(3,2) NOT NULL,
  ID_medico INT NOT NULL,
  ID_specializzazione INT NOT NULL,
  PRIMARY KEY (ID_medico, ID_specializzazione),
  FOREIGN KEY (ID_medico) REFERENCES MedicoDT(ID_medico),
  FOREIGN KEY (ID_specializzazione) REFERENCES Specializzazione(ID_specializzazione)
);
```

4.6 PostgreSQL DML Star Schema:

Query a) Qual è l'andamento dei costi per il SSN mensilmente nell'arco degli ultimi 5 anni:

```
select EXTRACT(MONTH FROM a.data_vendita)::int AS month, EXTRACT(YEAR FROM a.data_vendita)::int AS year, SUM(a.Costo_SSN) AS costo_SSN

from Acquisto a

where (EXTRACT(epoch FROM AGE(CURRENT_DATE, a.data_vendita))/86400)::int BETWEEN 0 AND 365*5

group by year, month

order by year, month;
```

Query b) Qual è l'andamento dei costi per il SSN mensilmente nell'arco degli ultimi 5 anni suddiviso per categorie di pazienti: giovani (età < 20); adulti (età < 65); anziani (età <80); molto anziani (età >=80)

```
WITH acquisto_paziente as (select * from Acquisto a join pazientedt p on (a.ID_paziente = p.id_paziente))


select t.month, t.year, sum(t."costo_giovani_SSN") as "costo_giovani_SSN", sum(t."costo_adulti_SSN") as "costo_adulti_SSN",
sum(t."costo_anziani_SSN") as "costo_anziani_SSN", sum(t."costo_molto_anziani_SSN") as "costo_molto_anziani_SSN"

from (

select EXTRACT(MONTH FROM ap.data_vendita)::int AS month, EXTRACT(YEAR FROM ap.data_vendita)::int AS year,  ap.costo_SSN as
"costo_giovani_SSN", 0 as "costo_adulti_SSN", 0 as "costo_anziani_SSN", 0 as "costo_molto_anziani_SSN"

from acquisto_paziente ap

where ((EXTRACT(epoch FROM AGE(CURRENT_DATE, ap.data_vendita))/86400)::int BETWEEN 0 AND 365*5) and (EXTRACT(year from
CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 0 AND 19)


union all


select EXTRACT(MONTH FROM ap.data_vendita)::int AS month, EXTRACT(YEAR FROM ap.data_vendita)::int AS year,  0 as "costo_giovani_SSN",
ap.costo_SSN as "costo_adulti_SSN", 0 as "costo_anziani_SSN", 0 as "costo_molto_anziani_SSN"

from acquisto_paziente ap

where ((EXTRACT(epoch FROM AGE(CURRENT_DATE, ap.data_vendita))/86400)::int BETWEEN 0 AND 365*5) and (EXTRACT(year from
CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 20 AND 64)


union all


select EXTRACT(MONTH FROM ap.data_vendita)::int AS month, EXTRACT(YEAR FROM ap.data_vendita)::int AS year,  0 as "costo_giovani_SSN", 0 as
"costo_adulti_SSN", ap.costo_SSN as "costo_anziani_SSN", 0 as "costo_molto_anziani_SSN"

from acquisto_paziente ap

where ((EXTRACT(epoch FROM AGE(CURRENT_DATE, ap.data_vendita))/86400)::int BETWEEN 0 AND 365*5) and (EXTRACT(year from
CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 65 AND 79)
```

```
union all

select EXTRACT(MONTH FROM ap.data_vendita)::int AS month, EXTRACT(YEAR FROM ap.data_vendita)::int AS year,  0 as "costo_giovani_SSN", 0 as "costo_adulti_SSN", 0 as "costo_anziani_SSN", ap.costo_SSN as "costo_molto_anziani_SSN"

from acquisto_paziente ap

where ((EXTRACT(epoch FROM AGE(CURRENT_DATE, ap.data_vendita))/86400)::int BETWEEN 0 AND 365*5) and (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita >= 80)

) as t

group by t.year, t.month

order by t.year, t.month;
```

Query c) Qual è la categoria di pazienti più disposta a pagare il ticket pur di avere il farmaco originale:

```
WITH acquisto_paziente as (select * from Acquisto a join pazientedt p on (a.ID_paziente = p.id_paziente)),

ratio_table as (
        select 'pazienti_giovani' as "tipo_pazienti" , t2.conteggio::float/t1.conteggio::float as ratio
from (select count(*) as "conteggio"
        from acquisto_paziente ap
        where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 0 AND 19) and ap.esenzione = FALSE) as t1,
        (select count(*) as "conteggio"
        from acquisto_paziente ap
        where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 0 AND 19) and ap.esenzione = FALSE and ap.Costo_paziente > 0) as t2
where t1.conteggio > 0

union all

select 'pazienti_adulti' as "tipo_pazienti" , t2.conteggio::float/t1.conteggio::float as ratio
from (select count(*) as "conteggio"
        from acquisto_paziente ap
        where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 20 AND 64) and ap.esenzione = FALSE) as t1,
        (select count(*) as "conteggio"
        from acquisto_paziente ap
        where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 20 AND 64) and ap.esenzione = FALSE and ap.Costo_paziente > 0) as t2
where t1.conteggio > 0

union all

select 'pazienti_anziani' as "tipo_pazienti" , t2.conteggio::float/t1.conteggio::float as ratio
from (select count(*) as "conteggio"
```

```
                from acquisto_paziente ap
                where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 65 AND 79) and ap.esenzione = FALSE) as t1,
                (select count(*) as "conteggio"
                from acquisto_paziente ap
                where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita BETWEEN 65 AND 79) and ap.esenzione = FALSE and ap.Costo_paziente > 0) as t2
where t1.conteggio > 0

union all

select 'pazienti_molto_anziani' as "tipo_pazienti" ,t2.conteggio::float/t1.conteggio::float as ratio
from (select count(*) as "conteggio"
                from acquisto_paziente ap
                where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita >= 80) and ap.esenzione = FALSE) as t1,
                (select count(*) as "conteggio"
                from acquisto_paziente ap
                where  (EXTRACT(year from CURRENT_DATE)::int - ap.Anno_nascita >= 80) and ap.esenzione = FALSE and ap.Costo_paziente > 0) as t2
where t1.conteggio > 0
)

select ratio_table.tipo_pazienti
from ratio_table
where ratio_table.ratio = (select MAX(t.ratio)
                        from ratio_table as t);
```

Query d) Qual è il prezzo dei farmaci originali che cadono nella famiglia “antibiotici” prescritti nel 2020:

```
select f.Nome_farmaco, f.Produttore_farmaco, f.Prezzo_recente
from FarmacoDT f
where f.Nome_famiglia = 'antibiotici' and f.generico = FALSE and EXISTS(select *
                                from Acquisto a
                                where a.ID_farmaco = f.ID_farmaco and EXTRACT(YEAR FROM a.data_ricetta)::int = 2020);
```