Relazione Progetto

A cura di

Jacopo Manetti

(Matr. 7003888)

Fase di progettazione concettuale:

Lo scopo di questo progetto è quello di gestire un sistema di farmacie tramite database. Le farmacie vengono rifornite dai produttori di farmaci. Ogni farmacia ha degli impiegati. Ci sono pazienti esaminati da medici che prescrivono quindi farmaci ai pazienti. I produttori di farmaci hanno un codice, nome e indirizzo. Ogni farmacia ha un codice, nome, fax e indirizzo. I dipendenti hanno codice e nome. Ogni farmaco viene identificato dal nome commerciale. Ogni paziente ha codice, nome, sesso, indirizzo e numero di contatto. Il medico ha codice, specialità e nome.

Un produttore di farmaci ha un contratto con la farmacia per la fornitura di farmaci e ogni accordo ha una data di inizio e di fine.

I farmaci vengono venduti alle farmacie ai prezzi stabiliti dal produttore del farmaco.

Una prescrizione ha una data e una quantità scelte dal medico curante.

Ogni dipendente che lavora per la farmacia ha una data di inizio e fine del lavoro, se il dipendente è a tempo indeterminato non ha una fine.

Entità:

Produttore_Farmaci: questa entità viene usata per archiviare le informazioni sui produttori.

Questa tabella ha 3 attributi, un ID utilizzato in modo univoco per identificare ogni produttore, un nome usato per il marchio e indirizzo che contiene l’indirizzo della sede legale fisica del produttore.

Farmaco: Questa entità contiene i diversi tipi di farmaco offerti dai fornitori, ha un solo attributo ossia il nome speciale usato per identificarlo.

Farmacia: Questa entità contiene informazioni sulla farmacia. Si utilizza un ID che viene utilizzato per identificare in modo univoco ogni farmacia. Il nome che è utilizzato a scopo di rappresentazione. Il fax utilizzato per la comunicazione e l'indirizzo della farmacia correlata che indica dove è situata.

Paziente: questa entità memorizza le informazioni su ciascun paziente. il paziente ha un ID per l'identificazione. Il nome e il sesso per rappresentazione dell'individuo e altre informazioni come indirizzo e numero di contatto.

Dipendente: questa entità memorizza le informazioni su diversi dipendenti che lavorano in diverse farmacie. Ogni dipendente ha un ID che viene utilizzato per identificarlo in modo univoco e il nome.

Dottore: questa entità memorizza le informazioni dei medici. C'è un ID per identificare ogni medico in maniera univoca, nome e specializzazione che definisce l'ambito di lavoro per quel medico o dottore.

Relazioni:

Un produttore di farmaci può produrre uno o più farmaci, allo stesso modo un farmaco può essere autoprodotto, cioè non ha bisogno di un produttore di farmaci oppure può avere un solo produttore.

Un produttore di farmaci può avere un contratto con nessuno o solo una farmacia. Invece una farmacia potrebbe non avere nessuno o molti contratti con i produttori al fine di ottenere i farmaci. Inoltre, un contratto ha una data di inizio e una data di fine.

Una farmacia può avere molti impiegati che vi lavorano. Anche se ci deve essere almeno un dipendente che deve lavorare per una farmacia. Un

dipendente può lavorare per una sola farmacia. In aggiunta ogni dipendente ha una data di inizio che registra il giorno in cui il dipendente ha iniziato a lavorare e una di fine in cui è segnata la data della fine del rapporto di lavoro. Se l'occupazione è attualmente attiva non c'è una data di fine. (in questo caso allora è NULL)

Un farmaco può essere venduto a nessuna o a molte farmacie e una farmacia può acquistare nessuno o molti farmaci. Questa relazione richiede di costruire una nuova tabella per memorizzare i dati richiesti.

Un paziente è visto da un solo e un solo dottore, ma un dottore può vedere uno o più pazienti.

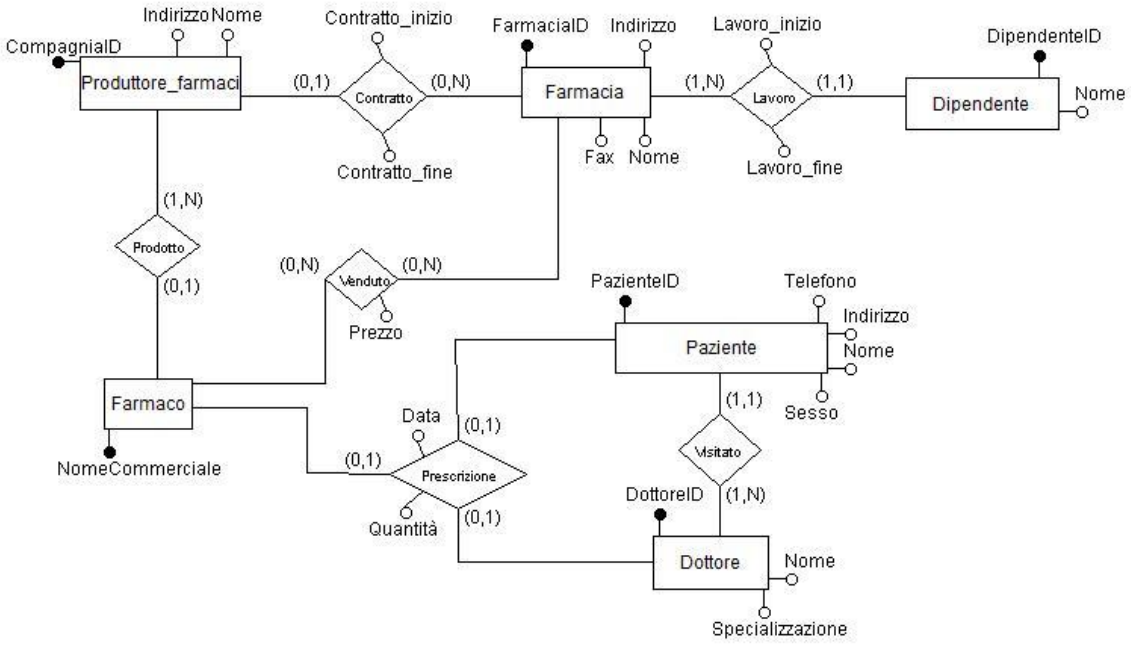
In un dato momento un medico può prescrivere a un solo paziente un solo farmaco. Durante questa operazione la data della prescrizione e anche la quantità di dosaggio viene registrata.

Ogni tabella contiene alcune colonne comuni che possono essere usate per collegarsi ad un'altra tabella.

le tabelle verranno mantenute piccole per rimuovere la ridondanza e archiviare dati significativi. Inoltre, verrà utilizzato anche l'aiuto delle viste per recuperare i dati.

Dal momento che le viste possono anche utilizzare join, diversi tipi di viste saranno unite. Questo aiuta a raggiungere l'incapsulamento dei dati. Dal momento che i dati devono essere mantenuti riservati al fine di garantire la privacy dei pazienti e dei medici. Pertanto la dimensione del database tornerà utile.

Diagramma ER:



Il diagramma Entity-Relationship mostra le entità, attributi e relazioni. Nel diagramma le chiavi primarie sono contraddistinte da un pallino nero. Gli attributi delle relazioni vengono utilizzati in caso di necessità per creare tabelle associate migliorando così l'archiviazione dei dati.

Fase di progettazione logica:

Legenda:

Grassetto & Sottolineato: Chiave Primaria

Tratteggiato & Corsivo: Foreign key

Produttore_farmaci(**CompagniaID**, Nome, Indirizzo, *Consegna*, Contratto_inizio, Contratto_fine)

Farmacia(**FarmaciaID**, Nome, Fax, Indirizzo)

Dipendente(**DipendenteID**, Nome, *FarmaciaID*, Lavoro_inizio, Lavoro_fine)

Farmaco(**NomeCommerciale**, *CompagniaID*)

Farmaci_disponibili(***NomeCommerciale***, ***FarmaciaID***, Prezzo)

Paziente(**PazienteID**, Nome, Sesso, Indirizzo, Telefono)

Dottore(**DottoreID**, Specializzazione, Nome)

Prescrizione(**PrescrizioneID**, *DottoreID*, *PazienteID*, *NomeCommerciale*, Data, Quantità)

Vi sono in totale 8 tabelle da implementare.

Queries:

Più query sono definite per il recupero dei dati e per i vincoli. Le tabelle contengono la maggior parte dei vincoli come il dominio e l'integrità referenziale. Molti dei vincoli più avanzati e complessi vengono applicati utilizzando i trigger. La maggior parte dei trigger esegue un controllo prima di usarlo.

Le procedure vengono utilizzate per accelerare il processo di esecuzione. Le query di selezione e proiezione sono incorporate in queste procedure.

Le viste sono state utilizzate per l'incapsulamento e per proteggere i dati.

Gli handlers vengono utilizzati nelle procedure per la gestione di eventi come le eccezioni.

I cursori vengono utilizzati per lavorare con i set di risultati nelle procedure.

Il nome del database usato è farmacia_schema.

Spiegazione delle Queries:

N.B. Queste informazioni si trovano anche sotto forma di commento nello script SQL

```
-- View #1
create view Vista_dipendenti
as
select FarmaciaID, Farmacia.Nome as NomeFarmacia, DipendenteID, Dipendente.Nome as NomeDipendente
from Farmacia
join Dipendente using (FarmaciaID);
```

Questa query crea una vista, in questa query viene utilizzato un join per evitare la duplicità delle colonne.

```
-- View #2
create view Vista_farmaci_costosi
as
select *
from Farmaci_disponibili
where Prezzo >= 100;
```

Questa query crea una vista che viene utilizzata per ottenere i farmaci che hanno un prezzo maggiore o uguale a 100 euro.

```
-- Trigger #1
DELIMITER $$

create trigger Date_Prescrizioni
before insert
on Prescrizione
for each row
if isnull(new.Data)
then set new.Data = curdate();
end if; $$
DELIMITER ;
```

Questa query crea un trigger utilizzato per generare la colonna della data automaticamente nella tabella prescrizione. Prima controlla se la data inserita è nulla. Dopodiché se la data è davvero null inserisce automaticamente la data corrente.

```
-- Trigger #2
DELIMITER $$

create trigger Prezzo_Farmaci_Negativo
before insert
on Farmaci_disponibili
for each row
if new.Prezzo < 0
then set new.Prezzo = abs(new.Prezzo);
```

```
end if; $$


DELIMITER ;
```

Questa query crea un trigger sulla tabella Farmaci_disponibili. Serve a validare la colonna Prezzo di questa tabella. Se per caso il prezzo inserito è negativo, lo converte in valore assoluto e lo memorizza.

```
-- Procedure #1 con selection
DELIMITER $$


create procedure Farmacia_con_id_o_nome(in fid int, in fnome varchar(50))
begin
    select * from Farmacia where FarmaciaID = fid or Nome = fnome;
end; $$


DELIMITER ;
```

Questa procedura utilizza l'istruzione select per ottenrere i dati di una farmacia, per selezionarla accetta due parametri, uno è il FarmaciaID e altro è il nome della farmacia.

```
-- Procedure #2 con proiezione, join and grouping
DELIMITER $$


create procedure Farmaci_totali_per_produttore()
begin
    select CompagniaID, Nome, count(NomeCommerciale) as FarmaciTotali
    from Produttore_farmaci
    join Farmaco using (CompagniaID)
```

```
    group by CompagniaID;
end; $$
DELIMITER ;
```

Qui la query crea una procedura che utilizza la proiezione per selezionare le colonne richieste. Oltre a ciò raggruppa anche i dati per eseguire l'aggregazione.

```
-- Procedure #3 con query annidate e controllo di flusso
DELIMITER $$

create procedure info_Dipendente(in nomeD varchar(25), in cognomeD varchar(25))
begin
    declare exit handler for sqlexception select 'SQLExeption: dipendente non trovato' as NoSuchCustomer;

    if not exists (select DipendenteID from Dipendente where Nome = (select concat(nomeD, ' ', cognomeD)))
    then signal sqlstate '45000';
    end if;

    select DipendenteID,
    (case when isnull(Lavoro_fine) then 'operativo' else 'non operativo' end) as status
    from Dipendente where Nome = (select concat(nomeD, ' ', cognomeD));
 end; $$

 DELIMITER ;
```

Questa query crea una procedura piuttosto lunga e complessa. Viene utilizzata una query nidificata e il controllo del flusso per eseguire l'esecuzione. Questa procedura si occupa di fornire lo status del dipendente, quindi se quest’ultimo è ancora operativo o se ha già terminato il rapporto di lavoro, viene anche gestito il caso in cui il dipendente non esista.

```
-- Handler #1

DELIMITER $$


create procedure inserimento_Farmaci_disponibili(in NomeC varchar(50), in Fid int, in Prezzo int)

begin

        declare continue handler for 1062 select 'chiave duplicata inserita' as ErrorOcurred;


    insert into Farmaci_disponibili values (NomeC, Fid, Prezzo);

end; $$


 DELIMITER ;


-- Handler #2

DELIMITER $$


create procedure Controllo_farmaco(in nc varchar(50))

begin

  declare exit handler for sqlexception select 'SQLException: dati non validi' as Message;


  if isnull(nc) or nc = ''

  then signal sqlstate '45000';
```

```
  select * from Farmaco where NomeCommerciale = nc;
  end if;
end; $$
DELIMITER ;
```

Queste procedure utilizzano gli handlers per la gestione di eventi. Ad esempio SQLEXCEPTION, ERROR O EXIT.

```
-- Cursor #1
DELIMITER $$

create procedure crea_lista_dipendenti(inout ListaNomi varchar(4000))
begin
    declare fine integer default 0;
    declare nomeDipendente varchar(100) default "";

    -- dichiarazione cursore per il nome
    declare dataCursor
       cursor for
           select nome from Dipendente;

    -- dichiarazione NOT FOUND handler
    declare continue handler
                for not found set fine = 1;

    open dataCursor;

    getName: loop
        fetch dataCursor into nomeDipendente;
```

```
        if fine = 1 then
            leave getName;
        end if;
        -- costruzione lista nomi
        set ListaNomi = concat(nomeDipendente,";",ListaNomi);
    end loop getName;
    close dataCursor;

end; $$

DELIMITER ;



-- Cursor #2
DELIMITER $$

create procedure ottieni_Prescrizioni(in fid int)
begin
    declare fine integer default 0;
    declare NomeCommerciale varchar(100) default "";



    -- dichiarazione cursore
    declare dataCursor
        cursor for
            select distinct Nome from Prescrizione;

         -- dichiarazione NOT FOUND handler
```

```
    declare continue handler
        for not found set fine = 1;

         OPEN dataCursor;

    getNomeCommerciale : loop
                fetch dataCursor into NomeCommerciale;
        if fine = 1 then
            leave getNomeCommerciale;
        end if;
        -- costruzione lista nomi
        select NomeCommerciale;
    end loop getNomeCommerciale;
    close dataCursor;

 end; $$

 DELIMITER ;
```

I cursori vengono utilizzati per eseguire operazioni su diversi set di dati ottenuti dalle query SELECT.