Basi di Dati e Conoscenza

Progetto A.A. 2020/2021

SISTEMA DI GESTIONE DI UNA FARMACIA

0244291

Ludovico De Santis

Indice

1. Descrizione del Minimondo

Si vuole realizzare il sistema informativo per la gestione dei prodotti disponibili in una farmacia, tenendo conto delle seguenti informazioni.

Ciascun prodotto è caratterizzato univocamente dal nome del prodotto stesso e dall'informazione relativa alla ditta fornitrice del prodotto. I prodotti presenti nella farmacia possono essere medicinali oppure prodotti di profumeria. Per ciascun prodotto è comunque noto l'elenco degli usi possibili del prodotto stesso (ad esempio malattie da raffreddamento, dolori alle ossa, oppure detergente per il viso o per il corpo). Della ditta fornitrice sono invece noti un recapito, il nome, utilizzato per identificare la ditta stessa, un insieme arbitrario di indirizzi (tra cui uno solo è da intendersi come indirizzo di fatturazione), ed un numero arbitrario di recapiti (fax, telefono, email) tra cui uno solo è da intendersi come indirizzo preferito per la comunicazione. Queste informazioni vengono gestite dal personale amministrativo della farmacia.

Nel caso dei medicinali, il sistema mantiene l'informazione relativa al fatto che un medicinale sia mutuabile o no, e se la vendita sia effettuabile solo se viene presentata una ricetta medica. Inoltre è nota la categoria farmacoterapeutica di appartenenza del medicinale (ad esempio antibiotico, oppure anti-infiammatorio) e se esistono interazioni tra quella categoria farmacoterapeutica ed altre categorie farmacoterapeutiche. Queste informazioni vengono gestite dal personale medico della farmacia.

Il personale medico, altresì, effettua la vendita dei prodotti. Per ciascuna scatola di medicinale venduta, qualora appartenga alla categoria dei medicinali, il cliente può decidere se registrare o meno il suo codice fiscale, a fini di riduzione di imposta.

I medicinali sono contenuti in cassetti, contenuti a loro volta in scaffali. Gli scaffali sono identificati da un codice numerico univoco per ciascuna categoria farmacoterapeutica ed i cassetti da un codice numerico univoco per ciascuno scaffale. I medicinali possono essere soggetti ad esaurimento, pertanto il personale amministrativo della farmacia deve poter generare dei report sulla giacenza di magazzino. Nel caso in cui alcuni medicinali sono in esaurimento, il personale amministrativo genera delle lettere di richiesta di acquisto ai fornitori, per delle quantità decise in fase di compilazione della lettera. Questa lettera viene associata ad un ordine che, quando viene consegnato, consente di aggiornare la giacenza di

magazzino e la posizione nei cassetti delle nuove scatole ricevute.

Allo stesso modo, è necessario mantenere informazioni sulla data di scadenza di ciascuna scatola di medicinali. Il personale amministrativo, su base settimanale, genera dei report indicanti in quali cassetti sono presenti medicinali in scadenza, al fine di provvedere alla dismissione degli stessi.

Nel caso infine di medicinali che richiedano la ricetta medica, si vuole tener traccia di ogni vendita effettuata per quel medicinale, indicando il giorno, la quantità ed il nome del medico che ha fatto la prescrizione. Queste informazioni vengono inserite all'interno di report generati indistintamente dal personale medico o dal personale amministrativo.

2. Analisi dei Requisiti

Identificazione dei termini ambigui e correzioni possibili

Linea	Termine	Nuovo termine	Motivo correzione
9	Recapiti	Contatti	Ambiguità con il termine recapito che indica in realtà l'indirizzo di ricezione corrispondenza e gestione logistica.
10	Indirizzo	Metodo	Potrebbero esserci ambiguità con il termine "indirizzi" riferito però ad indirizzi "fisici" dell'azienda.
19	Medicinale	Prodotto	Si indica successivamente la tipologia di Prodotto per il quale vale la eventuale registrazione C.F. cliente ai fini di detrazione d'imposta.
19	Categoria	Tipologia	Serve a disambiguare la tipologia di prodotto "medicinale" con le categorie farmacoterapeutiche dei medicinali.
21,31, 35	Medicinali	Scatole	Nei cassetti sono contenute le scatole, che a loro volta si riferiscono ad un determinato tipo di medicinale.
23, 25	Medicinali	Scorte di medicinali	Disambiguare il concetto di "farmaco medicinale" con il concetto di scorte di quel medicinale, ovvero della quantità di scatole presente a magazzino di quel medicinale.
26,27	Lettere di richieste di acquisto	Ordini	Una lettera di acquisto in realtà viene "inglobata" dall'ordine stesso, ovvero rappresentano lo stesso concetto.
28	Giacenza	Posizioni	Una giacenza di magazzino di un medicinale è in realtà assimilabile alla quantità di posizioni occupate dalle scatole che riferiscono il medicinale stesso.

Specifica disambiguata

Si vuole realizzare il sistema informativo per la gestione dei prodotti disponibili in una farmacia, tenendo conto delle seguenti informazioni.

Ciascun prodotto è caratterizzato univocamente dal nome del prodotto stesso e dall'informazione relativa alla ditta fornitrice del prodotto. I prodotti presenti nella farmacia possono essere medicinali oppure prodotti di profumeria. Per ciascun prodotto è comunque noto l'elenco degli usi possibili del

prodotto stesso (ad esempio malattie da raffreddamento, dolori alle ossa, oppure detergente per il viso o per il corpo). Della ditta fornitrice sono invece noti il nome, utilizzato per identificare la ditta stessa, un insieme arbitrario di indirizzi (tra cui uno solo è da intendersi come indirizzo di fatturazione ed uno solo è da intendersi come recapito), ed un numero arbitrario di contatti (fax, telefono, email) tra cui uno solo è da intendersi come metodo preferito per la comunicazione. Queste informazioni vengono gestite dal personale amministrativo della farmacia.

Nel caso dei medicinali, il sistema mantiene l'informazione relativa al fatto che un medicinale sia mutuabile o no, e se la vendita sia effettuabile solo se viene presentata una ricetta medica. Inoltre è nota la categoria farmacoterapeutica di appartenenza del medicinale (ad esempio antibiotico, oppure anti-infiammatorio) e se esistono interazioni tra quella categoria farmacoterapeutica ed altre categorie farmacoterapeutiche. Queste informazioni vengono gestite dal personale medico della farmacia. Il personale medico, altresì, effettua la vendita dei prodotti. Per ciascuna scatola di prodotto venduta, qualora appartenga alla tipologia dei medicinali, il cliente può decidere se registrare o meno il suo codice fiscale, a fini di riduzione di imposta.

Le scatole sono contenute in cassetti, contenuti a loro volta in scaffali. Gli scaffali sono identificati da un codice numerico univoco per ciascuna categoria farmacoterapeutica ed i cassetti da un codice numerico univoco per ciascuno scaffale. Le scorte di medicinali possono essere soggette ad esaurimento, pertanto il personale amministrativo della farmacia deve poter generare dei report sulla giacenza di magazzino. Nel caso in cui alcune scorte di medicinali sono in esaurimento, il personale amministrativo genera degli ordini, per delle quantità decise in fase di compilazione dell'ordine stesso. L'ordine , quando viene consegnato, consente di aggiornare la posizione nei cassetti delle nuove scatole ricevute. Allo stesso modo, è necessario mantenere informazioni sulla data di scadenza di ciascuna scatola di medicinali. Il personale amministrativo, su base settimanale, genera dei report indicanti in quali cassetti sono presenti scatole di medicinali in scadenza, al fine di provvedere alla dismissione degli stessi.

Nel caso infine di medicinali che richiedano la ricetta medica, si vuole tener traccia di ogni vendita effettuata per le scatole di quel medicinale, indicando il giorno, la quantità ed il nome del medico che ha fatto la prescrizione. Queste informazioni vengono inserite all'interno di report generati indistintamente dal personale medico o dal personale amministrativo.

Glossario dei Termini

Termine	Descrizione	Sinonimi	Collegamenti
Prodotto	Un oggetto di cui la farmacia dispone.		Utilizzo, Ditta, Medicinale, ProdProfumeria, Ordine, Scatola
Ditta	Azienda che fornisce i prodotti.		Prodotto, Indirizzo, Contatto
Utilizzo	Usi possibili per i prodotti.		Prodotto
Medicinale	Una Tipologia di Prodotto che può richiedere ricetta medica ed essere mutuabile, la farmacia ha una o più scatole del medicinale. Alcuni medicinali sono vendibili solo con presentazione ricetta medica.		Prodotto, Scatola, Categoria
ProdProfumeria	Tipologia di prodotto non medicinale		Prodotto
Indirizzo	La ditta fornitrice può avere più indirizzo, ma solo uno per gestione logistica e solo uno per la fatturazione.		Ditta
Recapito	Indirizzo presso il quale la ditta fornitrice gestisce la logistica.		Ditta, Indirizzo
Contatto	Metodo comunicativo attraverso il quale è possibile contattare la ditta fornitrice, di cui uno solo viene preferito dall'azienda.	Metodo di comunicazione	Ditta

Categoria	Categoria farmacoterapeutica di un medicinale, una categoria può avere interazioni con altre categorie.		Categoria, Scaffale, Medicinale
Scatola	Una scatola si riferisce ad un determinato tipo di prodotto, presenta una data di scadenza.	Scorte di medicinali (pluralità di scatole)	Prodotto, Cassetto, Ordine
Ordine	Indica la richiesta di un quantitativo di scatole di un determinato prodotto, può essere in attesa di consegna oppure già consegnato.		Prodotto, Scatola
Giacenza	Indica la quantità di scatole presenti a magazzino per ogni medicinale.	Scorte di medicinali	Scatola
Scaffale	Contengono i cassetti, identificato da codice numerico univoco per ciascuna categoria.		Categoria, Cassetto
Cassetto	Contenente le scatole di medicinali, identificato in base al suo scaffale ed ad un codice numerico univoco per ogni scaffale.		Scaffale, Scatola
Vendita	Contiene informazioni di interesse riguardo la vendita di prodotti.		Scatola
Personale amministrativo	Si occupa di gestire le informazioni riguardo dittefornitrici, generare report su giacenze magazzino, scatole medicinali in scadenza o vendite di scatole di farmaci richiedenti ricetta, effettuare ordini ed aggiornare i dati di stoccaggio.		Ditta, Scatola, Utilizzo, Ordine

Personale Medico	Si occupa della gestione di informazioni dei medicinali (mutuabilità, necessarietà ricetta vendita), di indicare la correlazione tra categorie di medicinali, della vendita dei prodotti, oltre a generare report riguardanti la vendita di farmaci richiedenti ricetta medica.		Categoria, Medicinale, Vendita

Raggruppamento dei requisiti in insiemi omogenei

Frasi relative a Ditta
Un prodotto è caratterizzato univocamente dal nome del prodotto stesso e dall'informazione relativa alla ditta fornitrice del prodotto. Della ditta fornitrice sono invece noti il nome, utilizzato per identificare la ditta stessa, un insieme arbitrario di indirizzi (tra cui uno solo è da intendersi come indirizzo di fatturazione ed uno solo come recapito), ed un numero arbitrario di contatti (fax, telefono, email) tra cui uno solo è da intendersi come metodo preferito per la comunicazione.

Frasi relative a Medicinale
Nel caso dei medicinali, il sistema mantiene l'informazione relativa al fatto che un medicinale sia mutuabile o no, e se la vendita sia effettuabile solo se viene presentata una ricetta medica. Inoltre è nota la categoria farmacoterapeutica di appartenenza del medicinale (ad esempio antibiotico, oppure anti-infiammatorio). Nel caso infine di medicinali che richiedano la ricetta medica, si vuole tener traccia di ogni vendita effettuata per le scatole di quel medicinale, indicando il giorno, la quantità ed il nome del medico che ha fatto la prescrizione.

Frasi relative a Prodotto
Ciascun prodotto è caratterizzato univocamente dal nome del prodotto stesso e dall'informazione relativa alla ditta fornitrice del prodotto. I prodotti presenti nella farmacia possono essere medicinali oppure prodotti di profumeria. Per ciascun prodotto è comunque noto l'elenco degli usi possibili del prodotto stesso (ad esempio malattie da raffreddamento, dolori alle ossa, oppure detergente per il viso o per il corpo).

Frasi relative a Categoria
Inoltre è nota la categoria farmacoterapeutica di appartenenza del medicinale (ad esempio antibiotico, oppure anti-infiammatorio) e se esistono interazioni tra quella categoria farmacoterapeutica ed altre categorie farmacoterapeutiche.

Frasi relative a Scatola
Per ciascuna scatola di prodotto venduta, qualora appartenga alla tipologia dei medicinali, il cliente può decidere se registrare o meno il suo codice fiscale, a fini di riduzione di imposta. Le scatole sono contenute in cassetti, contenuti a loro volta in scaffali. Nel caso in cui alcune scorte di medicinali sono in esaurimento, il personale amministrativo genera degli ordini, per delle quantità decise in fase di compilazione dell'ordine stesso. L'ordine, quando viene consegnato, consente di aggiornare la giacenza di magazzino e la posizione nei cassetti delle nuove scatole ricevute. Allo stesso modo, è necessario mantenere informazioni sulla data di scadenza di ciascuna scatola di medicinali. Il personale amministrativo, su base settimanale, genera dei report indicanti in quali cassetti sono presenti scatole di medicinali in scadenza, al fine di provvedere alla dismissione degli stessi. Il personale amministrativo, su base settimanale, genera dei report indicanti in quali cassetti sono presenti scatole di medicinali in scadenza, al fine di provvedere alla dismissione degli stessi.

N.D. Sono state inserite frasi inerenti alle scorte di medicinali in quanto le scorte sono formate da scatole.

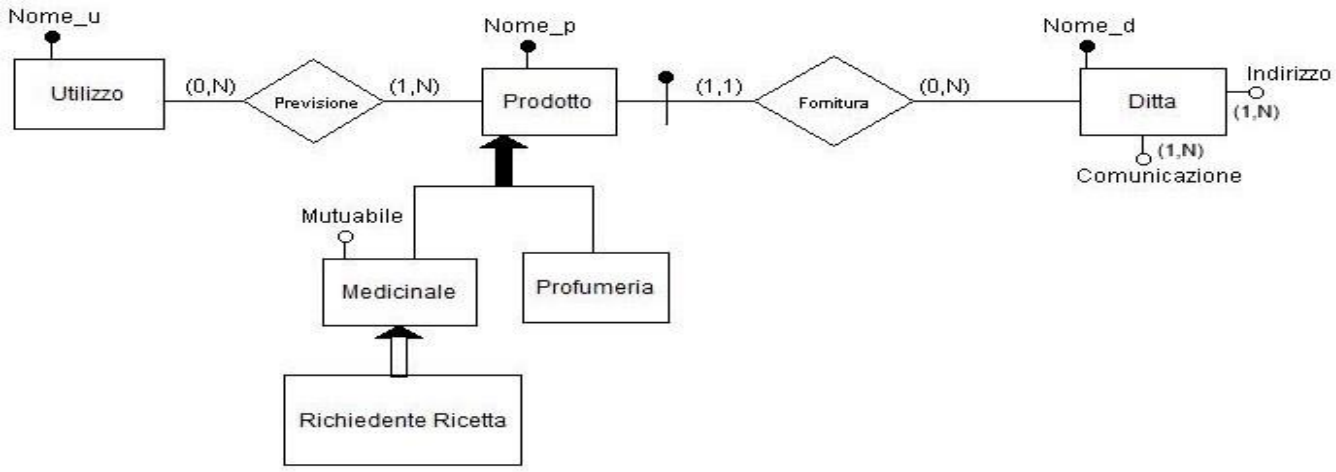
Frasi relative a Vendita
Nel caso infine di medicinali che richiedano la ricetta medica, si vuole tener traccia di ogni vendita effettuata per quel medicinale, indicando il giorno, la quantità ed il nome del medico che ha fatto la prescrizione. Queste informazioni vengono inserite all'interno di report generati indistintamente dal personale medico o dal personale amministrativo. Il personale medico, altresì, effettua la vendita dei prodotti. Per ciascuna scatola di prodotto venduta, qualora appartenga alla tipologia dei medicinali, il cliente può decidere se registrare o meno il suo codice fiscale, a fini di riduzione di imposta.

Frasi relative a Cassetti
Le scatole sono contenute in cassetti, contenuti a loro volta in scaffali. Gli scaffali sono identificati da un codice numerico univoco per ciascuna categoria farmacoterapeutica ed i cassetti da un codice numerico univoco per ciascuno scaffale. Ordine che, quando viene consegnato, consente di aggiornare la giacenza di magazzino e la posizione nei cassetti delle nuove scatole ricevute.

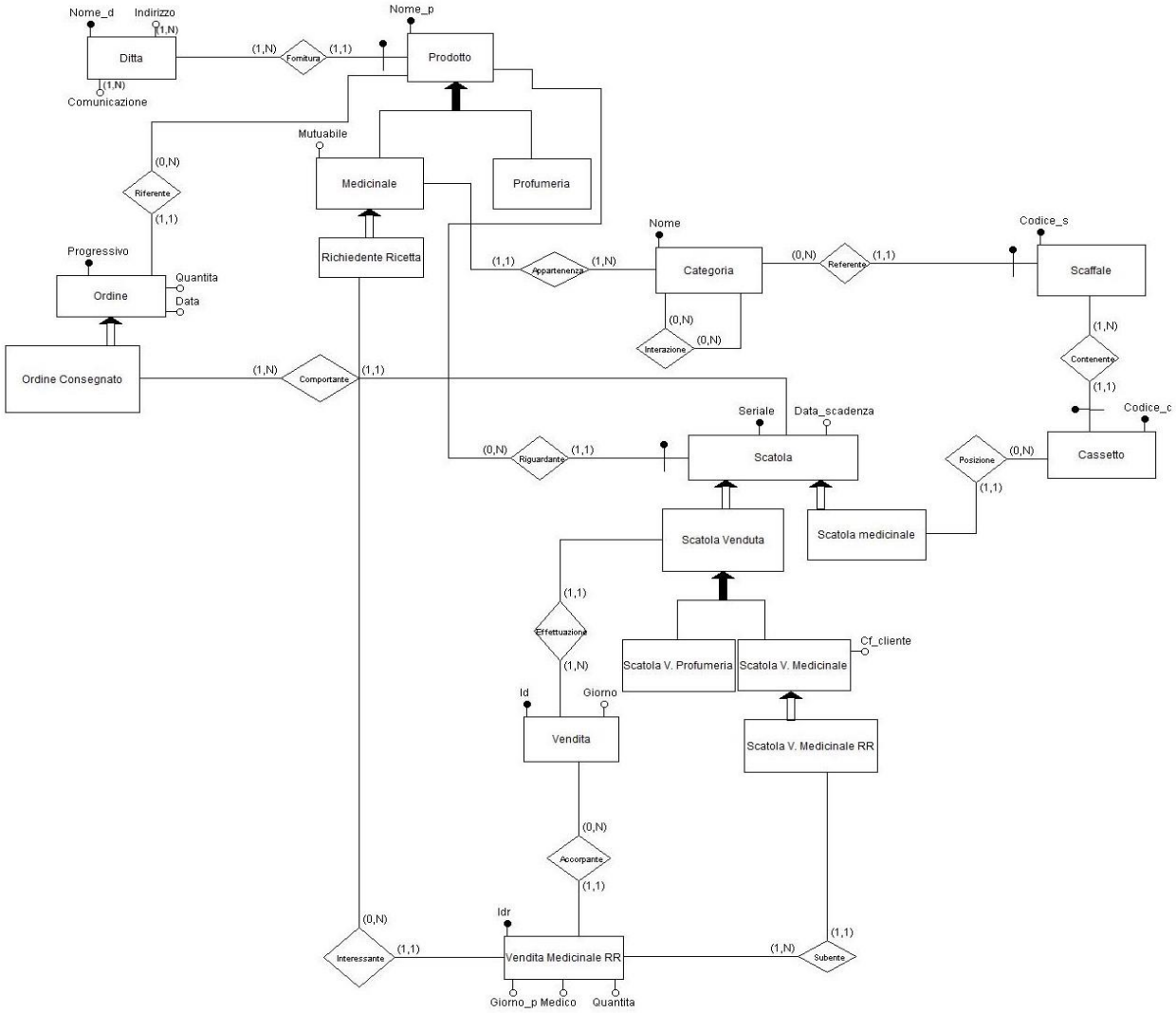
Frasi relative a Ordine
Nel caso in cui alcune scorte di medicinali sono in esaurimento, il personale amministrativo genera degli ordini, per delle quantità decise in fase di compilazione dell'ordine stesso. L'ordine , quando viene consegnato, consente di aggiornare la giacenza di magazzino e la posizione nei cassetti delle nuove scatole ricevute.

3. Progettazione concettuale

Costruzione dello schema E-R



In questa prima parte di progettazione, si è modellata innanzitutto la generalizzazione totale di prodotto in prodotti medicinali e prodotti di profumeria, in quanto sono le uniche tipologie di prodotto presenti nel nostro database. Nel caso dei medicinali, si è deciso di tenere traccia del fatto che un medicinale sia Mutuabile oppure no attraverso un attributo, mentre è stata usata una generalizzazione parziale per indicare i medicinali richiedenti ricetta. Il Prodotto viene individuato univocamente tramite il suo nome e la ditta che lo fornisce. Nel caso della Ditta, identificata univocamente dal suo nome, si è scelto di utilizzare attributi multipli e composti (nell'immagine visibile solo la molteplicità, una maggiore articolazione sarà indicata in fase di ristrutturazione) per "Indirizzo" e "Comunicazione".

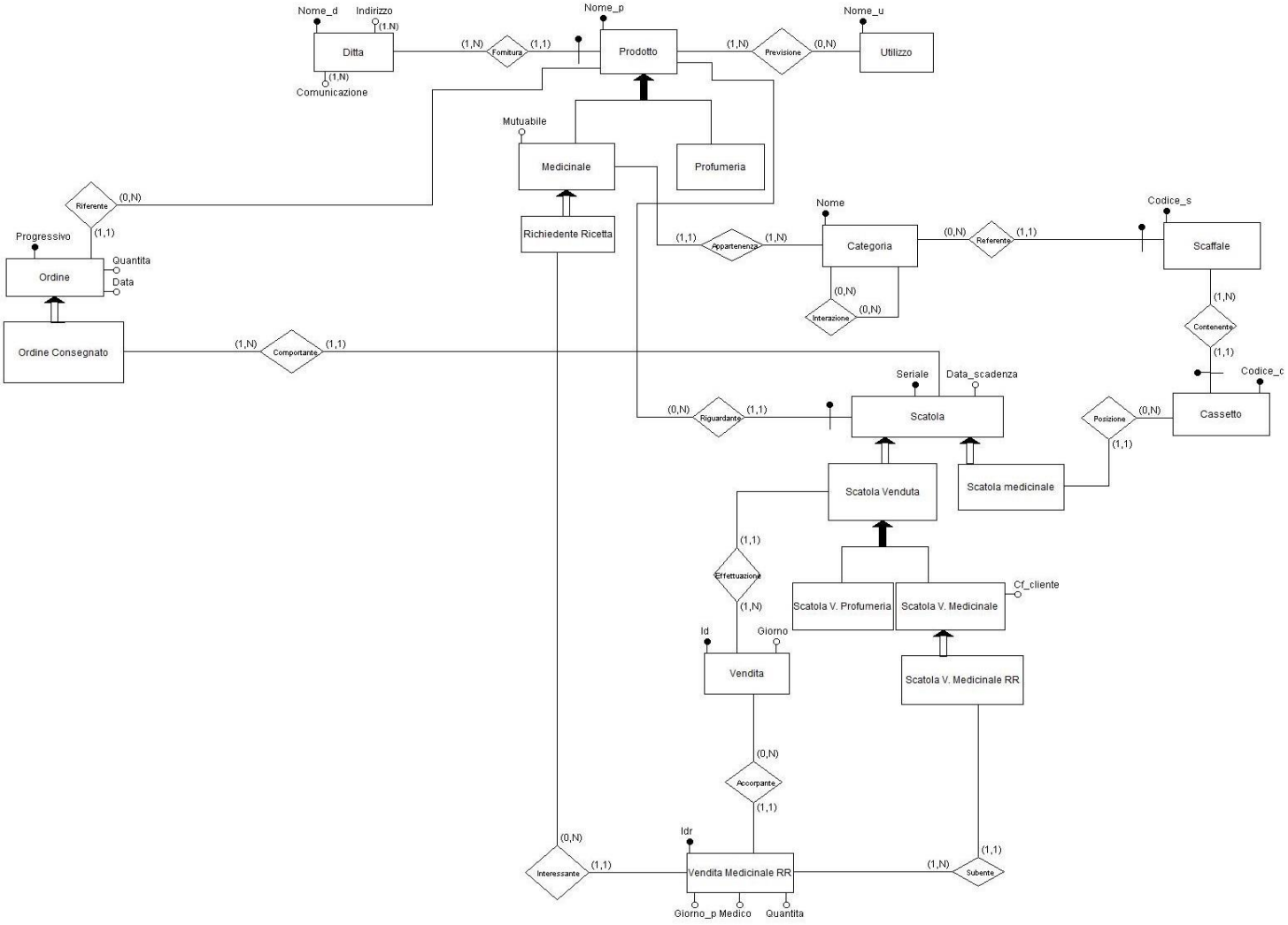


In questa seconda parte, si è deciso di modellare vari aspetti. Per quanto riguarda una scatola di un prodotto, identificata da un seriale e dal prodotto riferito, sono state applicate due generalizzazioni: la prima, parziale, serve ad estrapolare il concetto di scatola venduta. La scatola venduta, qualora appartenga alla categoria dei medicinali, presenta l'attributo Cf_cliente in quanto per ogni scatola di questa tipologia venduta, il cliente può decidere se memorizzare tale informazione. Se la scatola venduta appartiene alla tipologia dei medicinali richiedenti ricetta, sarà correlata con la sua corrispettiva vendita dove vengono memorizzate informazioni aggiuntive previste in esclusiva per questa tipologia. La seconda generalizzazione collegata a Scatola, sempre parziale, individua le scatole di medicinali. Questo perché si è ritenuto opportuno memorizzare la posizione di queste scatole in quanto essendo prodotti di delicata gestione e trattamento hanno bisogno di una particolare

gestione. Cosa non necessaria invece per i prodotti di profumeria, in quanto essi sono prodotti di consumo "normale”.

Viene utilizzata, inoltre, una entità ordine in quanto viene ritenuto necessario e di interesse per il nostro database la gestione di uno storico di ordini. Una vendita di medicinali RR può riguardare più scatole di un solo prodotto di medicinale richiedente ricetta, e si riferisce ad una vendita “globale” che può includere anche scatole dello stesso prodotto ma prescritte in una prescrizione diversa oppure in generale altre scatole di medicinali o prodotti di profumeria vendute congiuntamente. La scatola ovviamente deve riferirsi allo stesso medicinale RR riferito dall’entità “Vendita Medicinale RR”.

Integrazione finale



Regole aziendali

1) Quando una scatola di medicinale viene inserita a magazzino, deve essere inserita in uno scaffale specifico per la categoria del medicinale che la scatola riferisce.
2) La massima quantità ordinabile per un singolo ordine è di 300 scatole.
3) Il campo Cf_cliente deve essere impostato solo per le scatole vendute di medicinali, in base alle indicazioni del cliente.
4) Una scatola di medicinale viene considerata in scadenza e quindi dismessa a 90 giorni dalla sua scadenza.
5) Il recapito preferito deve essere solo uno per ditta, così come l'indirizzo di fatturazione.
6) Il mezzo di comunicazione preferito deve essere uno per ditta.
7) In fase di vendita, una prescrizione deve essere ritenuta valida se emessa entro i 30 giorni precedenti alla vendita stessa.
8) Le scorte di un medicinale devono essere considerate in esaurimento se sono presenti a magazzino meno di 3 scatole del medicinale stesso.
9) Una scatola in scadenza ma non ancora dismessa non deve essere venduta.
10) Non deve essere registrata, in fase di registrazione di un ordine, una scatola che risulta essere già in scadenza.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatori
Prodotto	Un prodotto trattato dalla farmacia	Nome_p	Nome_p + Nome_d(id. esterno)
Medicinale	Un prodotto di tipologia medicinale	Nome_p, Mutuabile	Nome_p + Nome_d(id. esterni)
Profumeria	Un prodotto di tipologia “profumeria”	Nome_p	Nome_p + Nome_d(id. esterni)
Richiedente Ricetta	Un prodotto di tipo medicinale richiedente ricetta per essere venduto	Nome_p, Mutuabile	Nome_p + Nome_d(id. esterni)
Utilizzo	Rappresenta l’utilizzo che si può fare per un prodotto	Nome_u	Nome_u
Previsione	Associa un prodotto ad un determinato utilizzo	Prodotto, Utilizzo	Prodotto+utilizzo (id esterni)
Ditta	Azienda fornitrice dei prodotti	Nome_d, Indirizzo (composto), Comunicazione (composto)	Nome_d
Ordine	Ordine effettuato dalla farmacia per delle scatole di un prodotto	Progressivo, Data, Quantita	Progressivo
Ordine Consegnato	Un ordine consegnato che permette di aggiornare le scatole presenti	Progressivo	Progressivo (id esterno)
Categoria	Categoria di appartenenza per un medicinale	Nome_c	Nome_c
Interazione	Rappresenta l’interazione tra due categorie di medicinali	Categoria_uno, Categoria_due	Categoria_uno+Categoria_due (id.esterni)
Scaffale	Uno scaffale per gestione scorte	Codice_s	Codice_s + Categoria(id. esterno)

Cassetto	Un cassetto per stoccaggio scatole	Codice_c	Codice_c + Codice_s + Categoria (id.esterni)
Scatola	Un scatola di prodotto	Seriale, Data_scadenza	Seriale + Prodotto + Ditta(id.esterni)
Scatola Venduta	Una scatola che viene venduta	Seriale	Seriale + Prodotto + Ditta(id.esterni)
Scatola Venduta Profumeria	Una scatola di un prodotto di profumeria che viene venduta	Seriale	Seriale + Prodotto + Ditta(id.esterni)
Scatola Venduta Medicinale	Una scatola di un medicinale che viene venduta	Seriale	Seriale + Prodotto + Ditta(id.esterni)
Scatola Venduta Medicinale RR	Una scatola di un medicinale richiedente ricetta che viene venduta	Seriale	Seriale + Prodotto + Ditta(id.esterni)
Scatola Medicinale	Una scatola di medicinale presente in magazzino	Seriale, Data_scadenza	Seriale + Prodotto + Ditta (id.esterni)
Vendita	Una vendita che può riguardare più scatole di prodotti diversi	Id, Giorno	Id
Vendita Medicinale Richiedente Ricetta	Una vendita riferita a scatole riferenti lo stesso medicinale RR vendute in base ad una stessa prescrizione.	Idr, Giorno_p, Medico, Quantita	Idr

4. Progettazione logica

Volume dei dati

Concetto nello schema	Tipo[1]	Volume atteso
Prodotto	E	100
Medicinale	E	80
Medicinale Richiedente Ricetta	E	56
Profumeria	E	20
Utilizzo	E	30
Ditta	E	50
Indirizzo	E	250
Comunicazione	E	250
Ordine	E	183.350
Ordine Consegnato	E	183.350
Categoria	E	50
Scaffale	E	50
Cassetto	E	500
Scatola	E	1.100.100
Scatola Medicinale	E	8800100
Scatola Venduta	E	1095000
Scatola Venduta Profumeria	E	219000
Scatola Venduta Medicinale	E	876000
Scatola Venduta Medicinale RR	E	600000
Vendita	E	700.000
Vendita Medicinale Richiedente Ricetta	E	400.000
Previsione	R	350
Interazione	R	100
Fornitura	R	100
Utilizzatore	R	250

[1] Indicare con E le entità, con R le relazioni

Possessore	R	250
Riferente	R	183.350
Comportante	R	1.100.100
Appartenenza	R	80
Referente	R	50
Contenente	R	500
Posizione	R	5.100
Effettuazione	R	1.095.000
Interessante	R	400.000
Subente	R	610.000
Riguardante	R	1.100.100
Accorpante	R	400.000

Tavola delle operazioni

Cod.	Descrizione	Frequenza attesa
OP1	Effettuare un ordine relativo a scatole di un prodotto	50/g
OP2	Registrare una scatola di medicinale arrivata in un ordine	80/g
OP3	Registrare una scatola di prodotto di profumeria arrivata in un ordine	20/g
OP4	Effettuare vendita di una scatola di prodotto di profumeria	60/g
OP5	Effettuare vendita di una scatola di medicinale non richiedente ricetta	70/g
OP6	Effettuare vendita di una scatola di medicinale richiedente ricetta	170/g
OP7	Generare Report sulle giacenze di scatole Medicinali	2/g
OP8	Generare Report sulle scatole di medicinali in scadenza con relativa dismissione	1/s

Costo delle operazioni

Operazione 1			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Ordine	E	1	S
Prodotto	E	1	L
Riferente	R	1	S

Costo = (2+2+1)*50 = 350 Accessi giornalieri.

Operazione 2			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Ordine	E	1	L
Ordine Consegnato	E	1	S
Scatola	E	1	S
Prodotto	E	1	L
Medicinale	E	1	L
Cassetto	E	1	L
Riguardante	R	1	S
Comportante	R	1	S
Posizione	R	1	S

Costo senza Ordine Consegnato = (1+2+1+1+1+2+2+2) *80 = 960 Accessi giornalieri.

Costo con Ordine Consegnato = (1+2+1+1+1+2+2+2+3) *80 = 1200 Accessi giornalieri.

Operazione 3			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Ordine	E	1	L
Ordine consegnato	E	1	S
Scatola	E	1	S
Prodotto	E	1	L
Riguardante	R	1	S
Comportante	R	1	S

Costo senza Ordine Consegnato = (1+2+2+1+2+2)*20 = 180 accessi giornalieri.

Costo con Ordine Consegnato = (1+2+2+1+2+2+1)*20 = 220 accessi giornalieri.

Operazione 4			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Prodotto	E	1	L
Scatola	E	1	S
Scatola Venduta	E	1	S
Scatola V. Prof.	E	1	S
Vendita	E	1	S
Effettuazione	R	1	S

NB la vendita viene scritta mediamente 0.75 volte perché le vendite fatte da 1.5 scatole

Costo senza generalizzazione = (2+2+1+(0.75*2))*60 = 390 accessi giornalieri.

Costo con generalizzazione = (2+2+2+2+(0.75*2)+1)*60 = 630 accessi giornalieri.

Operazione 5			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Scatola	E	1	S
Prodotto	E	1	L
Scatola Venduta	E	1	S
Scatola V. Medicinale	E	1	S
Vendita	E	1	S
Posizione	R	1	S
Effettuazione	R	1	S

Costo senza generalizzazione = (2+2+1+(0.75*2) +2)*70 = 595 accessi giornalieri

Costo con generalizzazione = (2+2+1+2+2+(0.75*2) +2)*70 = 875 accessi giornalieri

Operazione 6			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Scatola	E	1	S
Scatola Venduta	E	1	S
Scatola V. Medicinale	E	1	S
Scatola V. Med. RR	E	1	S
Prodotto	E	1	L
Vendita	E	1	S
VenditaMedicinaleRR	E	1	S
RichiedenteRicetta	E	1	L
Subente	R	1	S
Interessante	R	1	S
Posizione	R	1	S
Effettuazione	R	1	S
Accorpante	R	1	S

Costo senza generalizzazione = (2+(0.75*2)+0.75*(2+1+2+2)+2+2+2+1)*170 = 2700

Costo solo con scatola venduta = (2+2+(0.75*2)+0.75*(2+1+2+2)+2+2+2+1)*170 = 3020

Costo con generalizzazione = (2+2+2+2+(0.75*2)+0.75*(2+1+2+2)+2+2+2+1)*170 = 3700

Operazione 7			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Scatola	E	5k	L

Costo = 5k x 2 = 10k.

Operazione 8			
Concetto	**Costrutto**	**Accessi**	**Tipo**
Scatola	E	12	S
Prodotto	E	12	L

Costo = (12*2) = 24 accessi per settimana

N.B. Le scatole dismesse sono molto poche in quanto le farmacie, in generale, causa scadenze di prodotti di delicata gestione come i farmaci, lavorano “senza magazzino”, ovvero hanno a magazzino le quantità necessarie per coprire un periodo di tempo limitato preferendo ottimizzare la logistica per avere sempre prodotti con scadenza più a lungo termine possibile. Questa scelta è dovuta inoltre anche al fatto che , in media, una significativa percentuale di clienti si rifiuta di acquistare farmaci “troppo vicini” alla scadenza anche se rientrante nei parametri di sicurezza prescritti dalle case farmaceutiche. Nel caso di farmaci meno richiesti, infatti, la farmacia preferisce anche tenere dei medicinali in esaurimento o esauriti al fine di evitare queste situazioni, optando per una logistica ottimizzata in grado di sopperire in un tempo strettamente limitato alle eventuali richieste di questi farmaci meno richiesti.

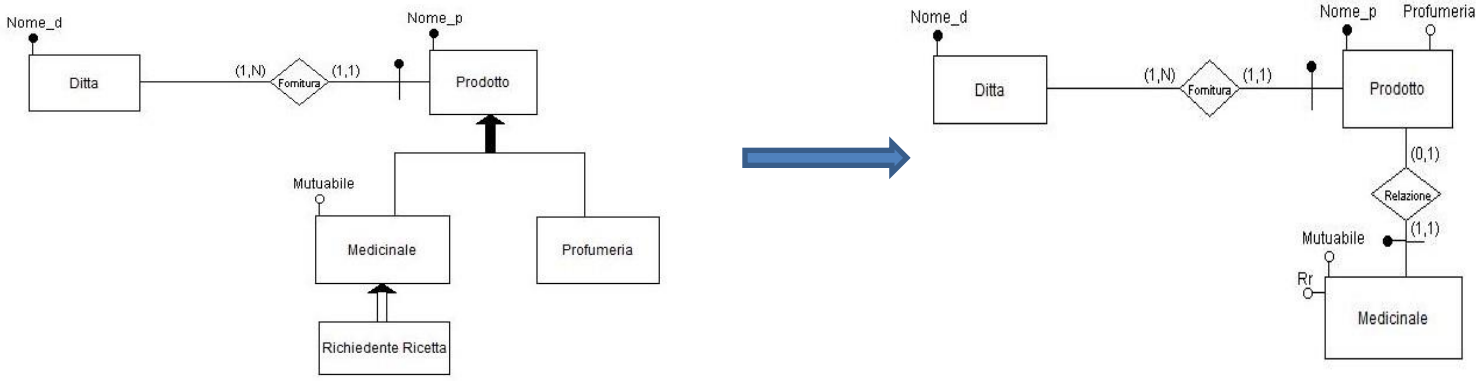
Ristrutturazione dello schema E-R

1 - Analisi Ridondanze

Sono presenti due ridondanze: una relativa all'attributo quantità nell' entità Ordine, ed un'altra presente per l'attributo quantità nell'entità Vendita Medicinale RR. La prima è necessaria, in quanto la memorizzazione di un' ordine non permette di conoscere a priori i seriali delle scatole che arriveranno e deve essere possibile stabilire al momento dell'inoltro ordine la quantità richiesta, sarà infatti possibile solo al momento della consegna e relativo inserimento scatole risalire alla quantità richiesta originariamente senza necessità di attributo.

La seconda ridondanza è invece eliminabile.

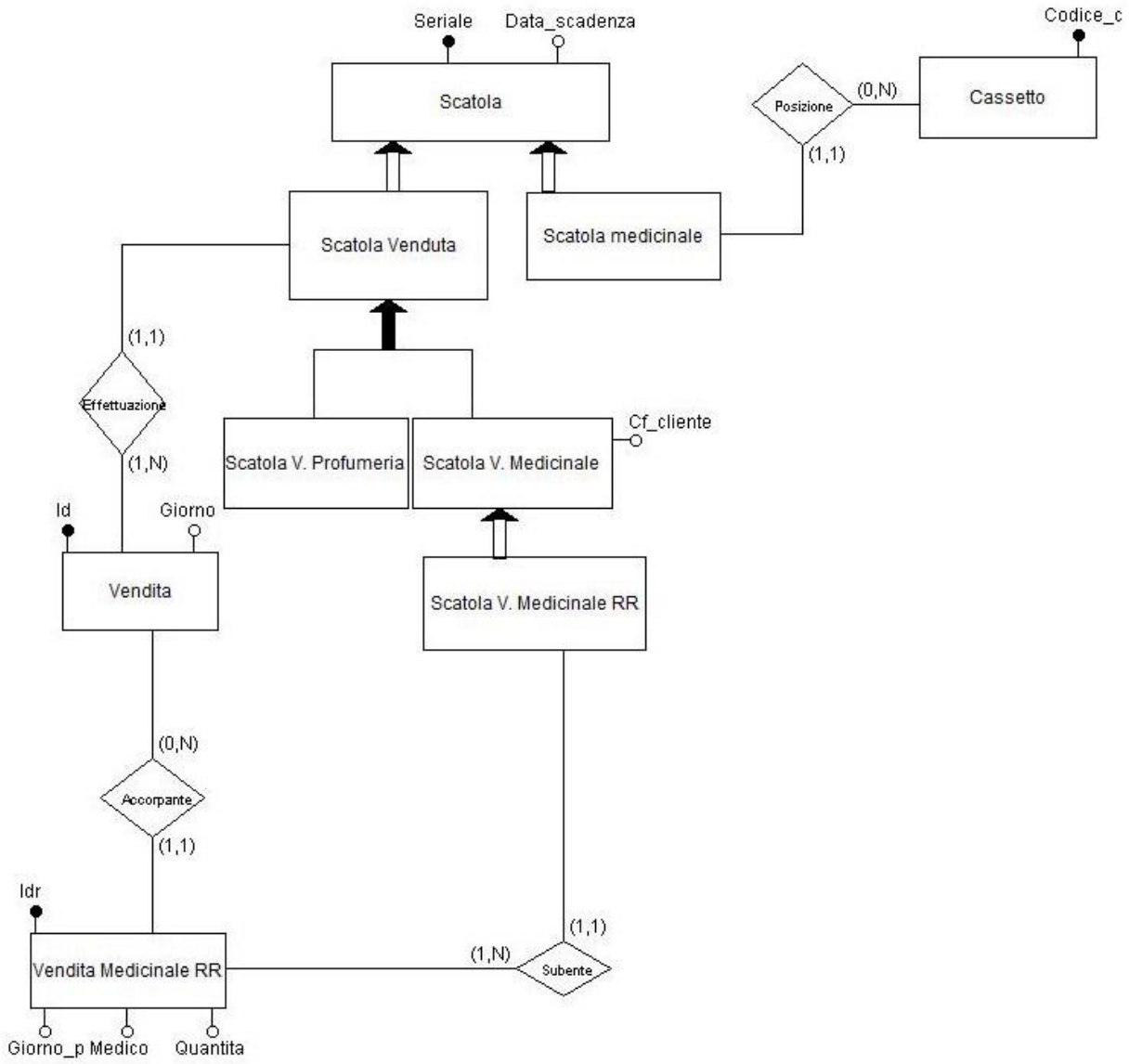
2 – Eliminazione generalizzazioni

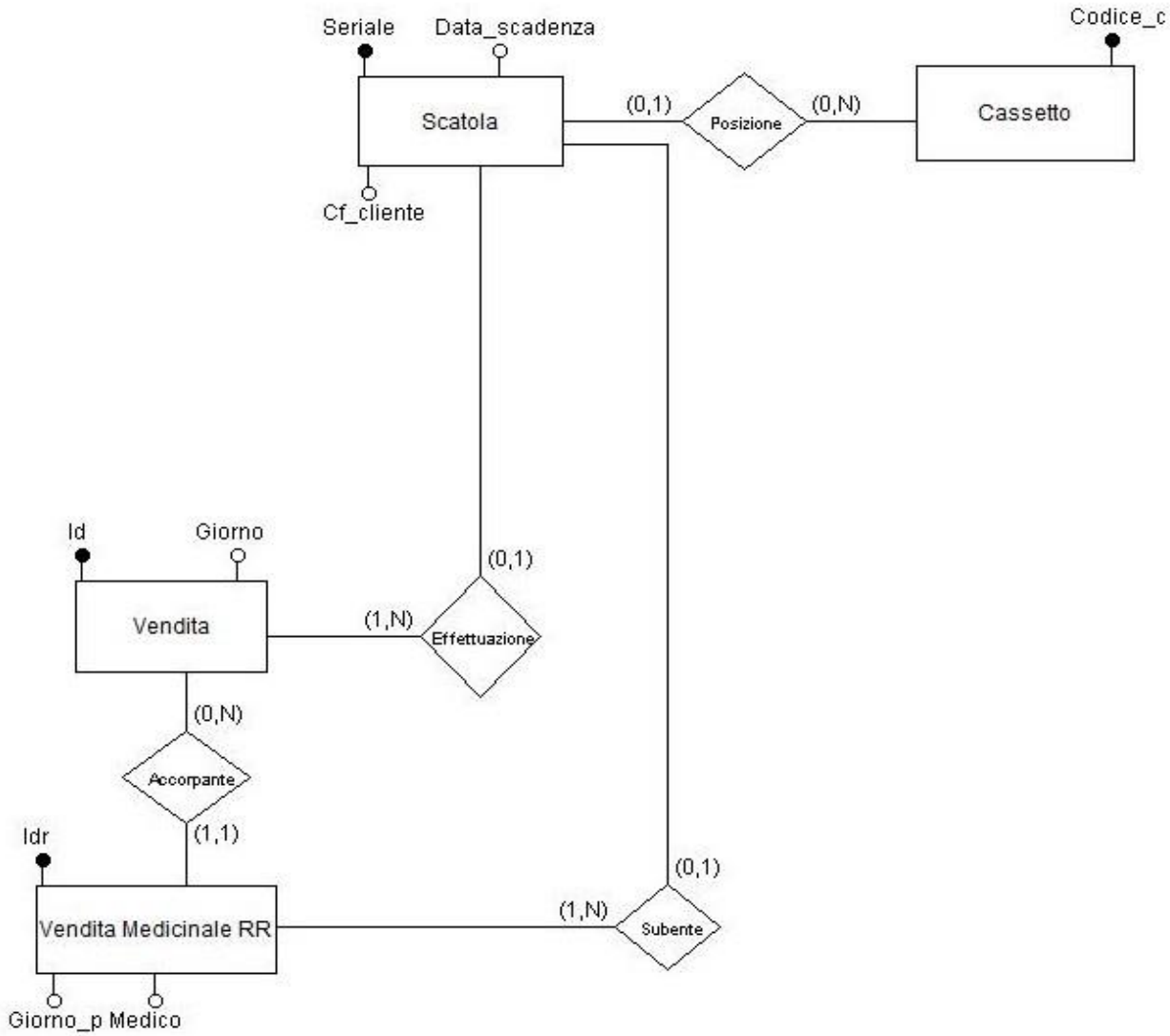


Nella gestione delle generalizzazioni, si è ritenuto opportuno mantenere solo l'entità medicinale della generalizzazione, in quanto ha una gestione particolare e più approfondita rispetto ai prodotti di profumeria, che sono stati accorpati in quanto non era necessario uno spreco di memoria avendo una gestione del tutto accomunabile a quella di un prodotto in generale. Ovviamente, se sarà vero l'attributo "profumeria" non si avrà la chiave verso medicinale e viceversa.

Per quanto riguarda i medicinali richiedenti ricetta, è stato deciso di accorpare la generalizzazione in quanto per un solo attributo non valeva la pena replicare le stesse informazioni in una entità separata.

In questo caso, il collegamento con l'entità "Vendita medicinale RR" sarà presente solo con l'attributo "Rr" valido.



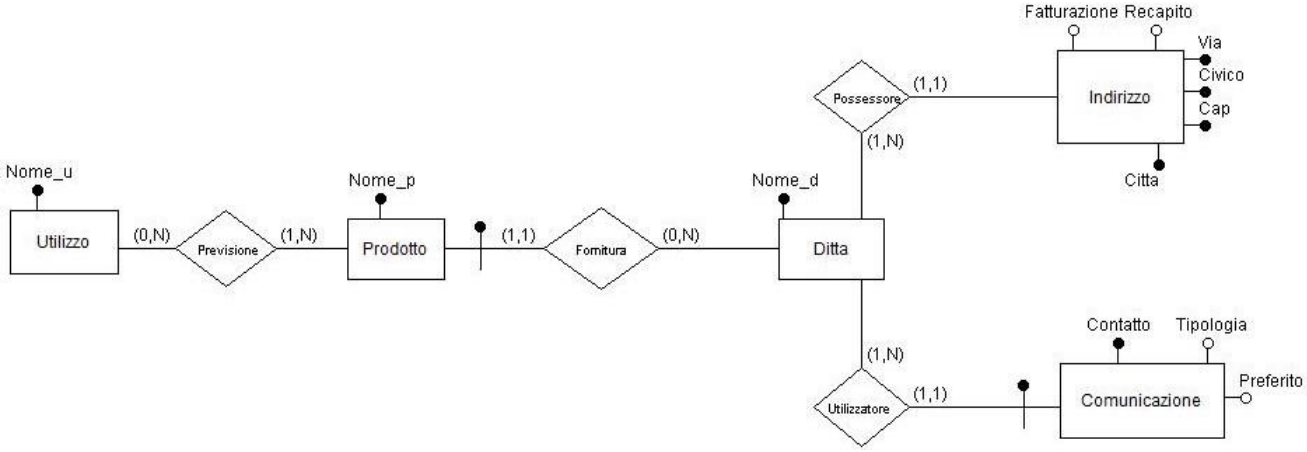


In questa generalizzazione, si è scelto di accorpare tutto in scatola in quanto, in base alle ipotesi sui volumi attesi mostrati prima, si avrebbero notevoli accessi in meno ed una più facile gestione ed implementazione, portando a rilevanti vantaggi prestazionali.

Per quanto riguarda la scelta degli identificatori primari, si è deciso di mantenere le scelte già precedentemente indicate.

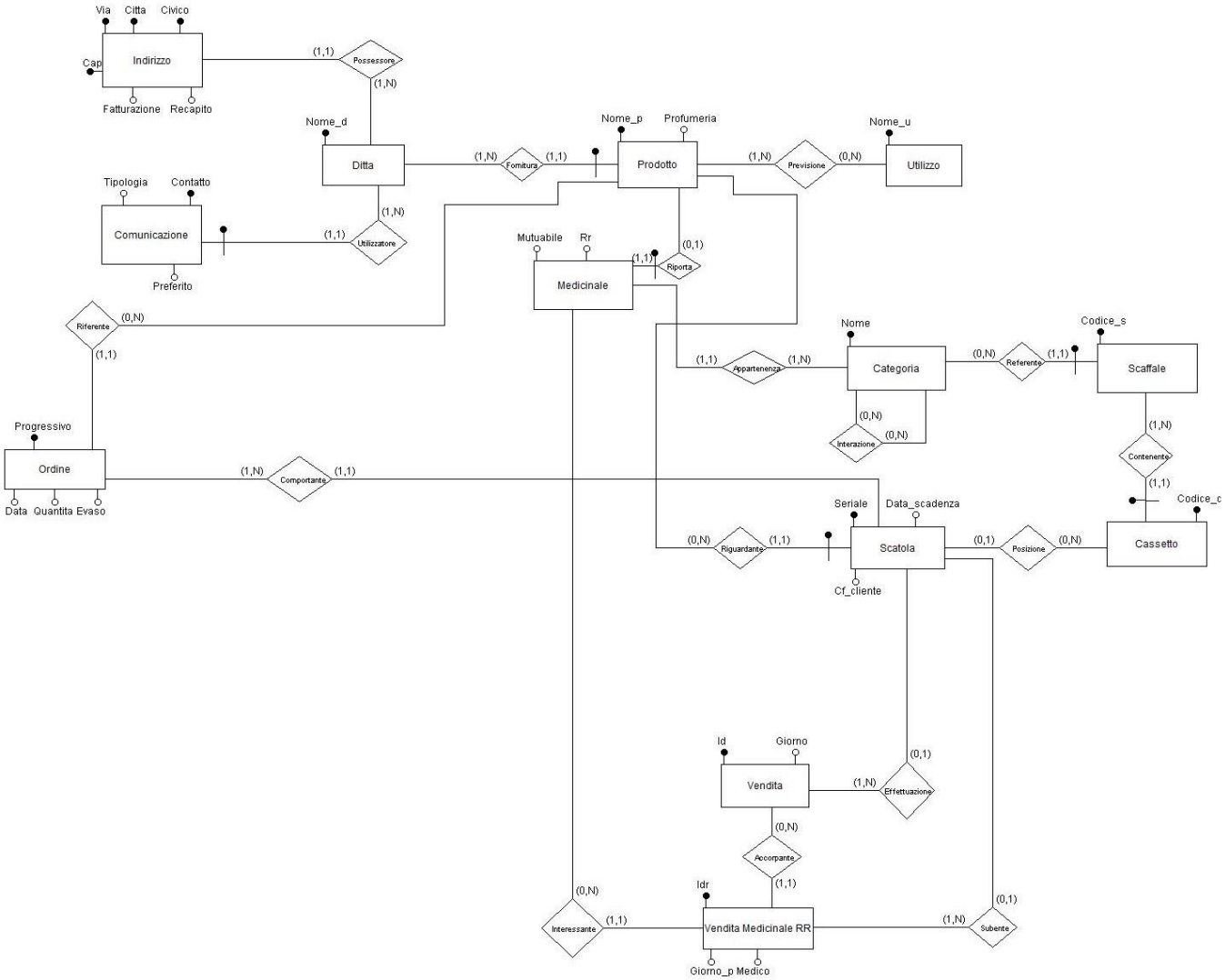
Trasformazione di attributi e identificatori

Per quanto riguarda gli attributi composti e multipli presenti nell'entità Ditta, la ristrutturazione è avvenuta nel seguente modo:



Nel caso della Ditta, identificata univocamente dal suo nome, si è scelto di utilizzare una entità per "Indirizzo", vengono utilizzati gli attributi "Fatturazione" e "Recapito" per indicare i due indirizzi(oppure il singolo indirizzo, visto che una caratteristica non esclude l'altra) sui quali si effettua fatturazione e gestione logistica. Stessa logica è stata applicata nell'entità comunicazione, avendo così grazie all'attributo "Preferito" la possibilità di indicare il contatto preferito. Un Indirizzo viene identificato univocamete da Via, Civico, Citta e Cap. Un mezzo comunicativo invece da Contatto e dalla ditta stessa.

La ritrutturazione in generale è avvenuta come illustrato in seguito:



Per quanto riguarda una scatola di un prodotto, identificata da un seriale e dal prodotto riferito, si nota la cardinalità 0,1 con Cassetto. Questa scelta è dovuta al fatto che l'informazione sulla posizione precisa di una scatola è necessaria nel caso di prodotti di tipo "medicinale", mentre per i prodotti di tipo profumeria questa informazione non viene impostata. Questa cardinalità ci indica inoltre che, nel momento della vendita o dismissione di una scatola di medicinale, l'informazione relativa alla sua posizione deve essere rimossa.

L' attributo "Cf_cliente" è opzionale, in quanto può essere registrato solo per scatole di prodotti di tipo medicinale ed a discrezione del cliente in fase di vendita. Viene utilizzata, inoltre, una entità ordine in quanto viene ritenuto necessario e di interesse per il nostro database la gestione di uno storico di ordini. L' attributo "evaso" sarà impostato a null quando un ordine verrà generato, mentre sarà impostato ad 1 quando un ordine viene consegnato e quindi registrato. Per quanto riguarda la vendita, nell'entità "vendita" vengono considerate le vendite di scatole di prodotti diversi

(profumeria, medicinali sia richiedenti ricetta che non), oppure anche di scatole di uno stesso prodotto ma prescritte eventualmente in giorni diversi o da un medico diverso a seconda dei casi (caratteristica per le scatole di medicinali richiedenti ricetta). La cardinalità 0,1 tra "Scatola" e "Vendita" sta ad indicare che una Scatola può non essere venduta ma eventualmente dismessa, oppure che si trovi in giacenza non essendo ancora stata venduta. L' entità "Vendita medicinale RR" tiene traccia appunto delle vendite riferite a scatole di un medicinale richiedente ricetta, ovviamente collegata alla vendita "globale" effettuata, scelta effettuata per i motivi descritti sopra. In questo caso, la cardinalità 0,1 sta ad indicare che, come prima, una scatola può essere in giacenza o dismessa, oppure che la scatola viene venduta ma non è appartenente ai medicinali richiedenti ricetta. Nel caso invece di vendita di scatole che rispettino questa caratteristica, è necessario salvare queste informazioni. La scatola ovviamente deve riferirsi allo stesso medicinale RR riferito dall'entità "Vendita Medicinale RR".

Traduzione di entità e associazioni

Prodotti (Nome_p, Ditta, Profumeria*);
Ditte (Nome_d);
Indirizzi (Citta, Via, Civico, Cap, Fatturazione, Recapito, Ditta);
Comunicazioni (Contatto, Ditta, Tipologia, Preferito);
Utilizzi (Nome_u);
Previsioni (Prodotto, Ditta, Utilizzo):
Ordini (Progressivo, Data, Quantita, Evaso, Prodotto, Ditta);
Medicinali (Medicinale, Ditta, Mutuabile, Rr, Categoria);
Categorie (Nome);
Interazioni (Categoria_uno, Categoria_due);
Scaffali (Codice_s, Categoria);
Cassetti (Codice_c, Scaffale, Categoria);
Scatole (Seriale, Prodotto, Ditta, Data_scadenza, Cassetto*, Scaffale*, Categoria*, Cf_cliente*, Vendita*, Vendita_medicinalerr*, Ordine);
Vendite (Id, Giorno);
Vendite_medicinalirr (Idr, Giorno_p, Medico, Prodotto, Ditta, Vendita);

-VINCOLI

Prodotti.Ditta ➡ Ditte.Nome_d
Indirizzi.Ditta ➡ Ditte.Nome_d
Comunicazioni.Ditta ➡ Ditte.Nome_d
Previsioni.Prodotto ➡Prodotti.Nome_p
Previsioni.Ditta ➡ Prodotti.Ditta
Previsioni.Utilizzo ➡ Utilizzi.Nome_u
Ordini.Prodotto ➡ Prodotti.NomeP
Ordini.Ditta ➡ Prodotti.Ditta
Medicinali.Ditta ➡Prodotti.Ditta
Medicinali.Medicinale ➡ Prodotti.Nome_p
Medicinali.Categoria ➡ Categorie.Nome
Interazioni.Categoria_uno ➡ Categorie.Nome
Interazioni.Categoria_due ➡ Categorie.Nome
Scaffali.Categoria ➡ Categorie.Nome
Cassetti.Categoria ➡ Scaffali.Categoria
Cassetti.Scaffale ➡Scaffali.Codice_s
Scatole.Scaffale ➡ Cassetti.Scaffale
Scatole.Cassetto ➡ Cassetti.Codice_c
Scatole.Categoria ➡Cassetti.Categoria
Scatole.Prodotto➡ Prodotti.Nome_p
Scatole.Ditta ➡ Prodotti.Ditta
Scatole.Vendita ➡ Vendite.Id
Scatole.Vendita_medicinalerr ➡ Vendite_medicinalirr.Idr
Scatole.Ordine ➡ Ordini.Progressivo
Vendite_medicinalirr.Vendita ➡Vendite.Id
Vendite_medicinalirr.Prodotto ➡Prodotti.Nome_p
Vendite_medicinalirr.Ditta ➡Prodotti.Ditta

Normalizzazione del modello relazionale

1NF

Una relazione è in prima forma normale se è presente per ogni relazione una chiave primaria. Non vi sono gruppi di attributi che si ripetono all'interno di una relazione e le colonne sono indivisibili.

Il modello relazionale rispetta questa definizione.

2NF

Una relazione è in seconda forma normale se è già in prima forma normale e se gli attributi dipendono dall'intera chiave, dunque non ci devono essere dipendenze parziali nelle relazioni.

Lo schema rispetta la seconda forma normale.

3NF

Una relazione è in 3 forma normale se è già in seconda (e di conseguenza in prima) e tutti gli attributi non chiave dipendono dalla chiave soltanto, ossia non esistono attributi non chiave che dipendono da altri attributi non chiave. Lo schema rispetta la terza forma normale.

5. Progettazione fisica

Utenti e privilegi

Amministratore (Personale amministrativo)

Ordini: SELECT, INSERT, UPDATE. Il personale amministrativo si occupa di gestire gli ordini, inserendoli, mostrandoli se necessario e aggiornandoli quando vengono consegnati con la successiva registrazione delle scatole ad essi associate.

Prodotti: SELECT.

Scatole: INSERT, SELECT, DELETE. Le insert avvengono quando un ordine viene consegnato, mentre la cancellazione avviene quando vengono effettuati i report sulle scatole in scadenza al fine della loro dismissione.

Ditte: INSERT.

Indirizzi: INSERT, SELECT, UPDATE. Gli indirizzi vengono aggiunti successivamente all'inserimento di una nuova ditta. L' Indirizzo preferito per quanto riguarda la fatturazione e/o il recapito viene inserito insieme all'indirizzo stesso se specificato dall'utente, altrimenti inserito in un secondo momento obbligatoriamente e impostato tramite un update.

Comunicazioni: INSERT, SELECT, UPDATE. I metodi comunicativi vengono aggiunti successivamente all'inserimento di una nuova ditta. Il contatto preferito viene inserito insieme al contatto stesso se specificato dall'utente, altrimenti inserito in un secondo momento obbligatoriamente e impostato tramite un update.

Utilizzi: INSERT.

Previsioni: INSERT.

Vendite_medicinalirr: SELECT. Utilizzata per i report sulle vendite di scatole di medicinali richiedenti ricetta medica.

Cassetti: SELECT. Viene effettuato l'accesso per verificare le posizioni disponibili in fase di registrazione di una scatola.

Medico (Personale Medico)

Prodotti/Medicinali: SELECT, INSERT. Il personale medico è responsabile in particolare dell'inserimento dei prodotti e di conseguenza anche dei medicinali, in quanto essendo, nel caso dei medicinali, prodotti delicati e di difficile gestione e trattamento, richiedono un inserimento da parte di personale qualificato sulla materia. Per evitare problemi sulla conoscenza dei prodotti in campo medico ed evitare disguidi sulla natura di un prodotto, il personale medico si occupa inoltre dell'inserimento dei prodotti di profumeria.

Categorie: INSERT, SELECT. In particolare, per gli inserimenti, le categorie sono gestite dal personale medico in quanto qualificato sulla materia.

Interazioni: INSERT. Stesso concetto delle categorie, essendo una interazione formata da categorie.

Vendita: INSERT. Per i motivi legati alla particolarità dei prodotti presenti in una farmacia, è gestita dal personale medico.

Scatole: SELECT, UPDATE. L' aggiornamento avviene quando una scatola viene venduta.

Vendite_medicinalirr: INSERT, SELECT. Stesse motivazioni della vendita.

Strutture di memorizzazione

Tabella indirizzi		
Attributo	**Tipo di dato**	**Attributi**[2]
citta	Varchar(45)	PK, NN
via	Varchar(45)	PK, NN
civico	Varchar(10)	PK, NN
cap	Int	PK, NN
fatturazione	Tinyint	
recapito	Tinyint	
ditta	Varchar(45)	NN

[2] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella comunicazioni		
Attributo	**Tipo di dato**	**Attributi[3]**
contatto	Varchar(45)	PK, NN
tipologia	Varchar(45)	NN
preferito	Tinyint	
ditta	Varchar(45)	PK, NN

Tabella ditte		
Attributo	**Tipo di dato**	**Attributi[4]**
nome_d	Varchar(45)	PK, NN

Tabella prodotti		
Attributo	**Tipo di dato**	**Attributi[5]**
nome_p	Varchar(45)	PK, NN
ditta	Varchar(45)	PK, NN
profumeria	Tinyint	

[3] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[4] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[5] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella medicinali		
Attributo	**Tipo di dato**	**Attributi[6]**
medicinale	Varchar(45)	PK, NN
ditta	Varchar(45)	PK, NN
categoria	Vachar(45)	NN
mutuabile	Tinyint	
rr	Tinyint	

Tabella utilizzi		
Attributo	**Tipo di dato**	**Attributi[7]**
nome_u	Varchar(45)	PK, NN

Tabella previsioni		
Attributo	**Tipo di dato**	**Attributi[8]**
prodotto	Varchar(45)	PK, NN
ditta	Varchar(45)	PK, NN
utilizzo	Varchar(45)	PK, NN

Tabella categorie		
Attributo	**Tipo di dato**	**Attributi[9]**
nome	Varchar(45)	PK, NN

[6] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[7] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[8] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[9] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella interazioni		
Attributo	**Tipo di dato**	**Attributi**[10]
categoria_uno	Varchar(45)	PK, NN
categoria_due	Varchar(45)	PK, NN

Tabella scaffali		
Attributo	**Tipo di dato**	**Attributi**[11]
codice_s	Int	PK, NN
categoria	Varchar(45)	PK, NN

Tabella cassetti		
Attributo	**Tipo di dato**	**Attributi**[12]
codice_c	Int	PK, NN
scaffale	Int	PK, NN
categoria	Varchar(45)	PK, NN

[10] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[11] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[12] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella ordini		
Attributo	**Tipo di dato**	**Attributi**[13]
progressivo	Int	PK, AI
data	Date	NN
quantita	Int	NN
prodotto	Varchar(45)	NN
ditta	Varchar(45)	NN
evaso	Tinyint	

Tabella scatole		
Attributo	**Tipo di dato**	**Attributi**[14]
seriale	Int	PK, NN, AI
prodotto	Varchar(45)	PK, NN
ditta	Varchar(45)	PK, NN
ordine	Int	NN
data_scadenza	Date	NN
vendita	Int	
vendita_medicinalerr	Int	
cassetto	Int	
scaffale	Int	
categoria	Varchar(45)	
cf_cliente	Varchar(20)	

[13] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[14] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Tabella vendite		
Attributo	**Tipo di dato**	**Attributi**[15]
id	Int	PK, NN, AI
giorno	Date	NN

Tabella vendite_medicinalirr		
Attributo	**Tipo di dato**	**Attributi**[16]
idr	Int	PK, NN, AI
medicinale	Varchar(45)	NN
ditta	Varchar(45)	NN
giorno_p	Date	NN
medico	Varchar(45)	NN

[15] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

[16] PK = primary key, NN = not null, UQ = unique, UN = unsigned, AI = auto increment. È ovviamente possibile specificare più di un attributo per ciascuna colonna.

Indici

Non sono stati utilizzati degli indici.

Trigger

Trigger scatole_BEFORE_INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`scatole_BEFORE_INSERT` BEFORE INSERT ON `scatole` FOR EACH ROW
BEGIN

	if new.`categoria` is not null then

		if (select count(*)
				from `scatole`
				where `cassetto` = new.`cassetto` and `scaffale` = new.`scaffale`
					and `categoria` = new.`categoria`) = 1 then
			signal sqlstate '45002';
			end if;

	end if;


	 if datediff(new.`data_scadenza`, current_date()) < 90 then
		signal sqlstate '45001';
	 end if;
END
```

Commento
Viene effettuato un controllo, per quanto riguarda le scatole di medicinali, in cui si verifica che la posizione in cui si sta registrando la scatola non sia già occupata. Inoltre, sia per le scatole di medicinali che per le scatole di prodotti di profumeria, viene verificato il fatto che la data di scadenza sia superiore a 90 giorni a partire dalla data corrente, in quanto non avrebbe senso registrare una scatola che risulterebbe secondo regola aziendale già in scadenza.

Trigger – vendite_medicinalirr_BEFORE_INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`vendite_medicinalirr_BEFORE_INSERT` BEFORE INSERT ON `vendite_medicinalirr` FOR EACH ROW
BEGIN

   declare var_giornov date;

   select `giorno`
   from `vendite`
   where `id` = new.`vendita`
   into var_giornov;

        if ((datediff(var_giornov, new.`giorno_p`) > 30) or
                (datediff(new.`giorno_p`, var_giornov) > 0)) then
                signal sqlstate '45012';
        end if;
END
```

Commento

In questo Trigger si è applicata la regola aziendale secondo la quale una prescrizione non può avere validità superiore a 30 giorni, impedendo quindi una registrazione di una vendita di scatola\e di medicinali richiedenti ricetta con prescrizione datata a più di 30 giorni di distanza rispetto alla data di vendita globale a cui è riferita.

Trigger – interazioni_BEFORE_INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`interazioni_BEFORE_INSERT` BEFORE INSERT ON `interazioni` FOR EACH ROW
BEGIN
   declare cat_uno varchar(45);
   declare cat_due varchar(45);
   declare done int default false;
   declare cur cursor for select `categoria_uno`, `categoria_due` from `interazioni`;
   declare continue handler for not found set done = true;

        open cur;
        read_loop: loop

                fetch cur into cat_uno, cat_due;
                if done then
                        leave read_loop;
                end if;

                if cat_uno = new.`categoria_due` and cat_due = new.`categoria_uno` then
                        signal sqlstate '45008';
                end if;

        end loop;
        close cur;
END
```

Commento
In questo trigger si evita semplicemente che esistano interazioni “duplicate” tra categorie, ovvero che non avvengano inserimenti di interazioni identiche tra categorie semplicemente invertite nei campi categoria_uno\due, in quanto è stato assunto che se una categoria x ha una interazione con una categoria y, allora di conseguenza anche la categoria y ha una interazione con la categoria x.

Trigger – ordini_BEFORE_INSERT

```
CREATE DEFINER = CURRENT_USER TRIGGER `mydb`.`ordini_BEFORE_INSERT` BEFORE INSERT ON `ordini` FOR EACH ROW
BEGIN
	if new.`quantita` > 300 or new.`quantita` < 1 then
		signal sqlstate '45000';
	end if;
END
```

Commento
In questo trigger si è applicata la regola aziendale secondo la quale non si possono ordinare più di 300 scatole di un prodotto in un singolo ordine e viene effettuato inoltre un controllo in caso di ordini con quantità inferiori ad uno che non avrebbero senso.

Eventi

Non sono stati utilizzati degli eventi temporizzati.

Viste

Non sono state utilizzate delle viste.

Stored Procedures e transazioni

Crea Ordine

```
CREATE PROCEDURE `crea_ordine` (IN var_prodotto VARCHAR(45),IN var_ditta VARCHAR(45), IN var_quantita int, out var_progressivo int)
BEGIN
		insert into `ordini` (`prodotto`, `ditta`, `quantita`, `data`) values (var_prodotto, var_ditta, var_quantita, curdate());

  set var_progressivo = last_insert_id();
END
```

Get Order Info

```
CREATE PROCEDURE `get_order_info` (IN var_prog INT, OUT var_prodotto VARCHAR(45), OUT var_ditta VARCHAR(45), OUT var_quantita INT, OUT is_med INT, OUT var_cat varchar(45))
BEGIN

   declare var_prof tinyint;
   declare var_evaso tinyint;
   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;

   set transaction isolation level repeatable read;
   start transaction;

                SELECT `prodotto`, `ditta`, `quantita`, `evaso`
                FROM `ordini`
                WHERE `progressivo` = var_prog
                INTO var_prodotto, var_ditta, var_quantita, var_evaso;

                /*order already served*/
                if var_evaso is not null then
                        signal sqlstate '45006';
                end if;

                /*wrong prog*/
                 if var_prodotto is null then
                        signal sqlstate '45007';
                end if;

                SELECT `profumeria`
                from `prodotti`
                where `nome_p` = var_prodotto and `ditta` = var_ditta
                into var_prof;

                UPDATE `ordini` set `evaso` = 1 WHERE `progressivo` = var_prog;

                if var_prof = 1 then
                        set is_med = 0;
                else
                        set is_med = 1;
                        select `categoria`
                        from `medicinali`
                        where `medicinale` = var_prodotto and `ditta` = var_ditta
                        into var_cat;
                end if;
   commit;
END
```

Commento
Procedure chiamata quando il personale amministrativo inserisce il progressivo di un ordine da registrare. Viene controllato se l'ordine è stato già registrato, se esiste, e vengono ritornate le informazioni relative all'ordine stesso. Come livello di isolamento è stato utilizzato Repeatable Read.

Get Product Info

```
CREATE PROCEDURE `get_product_info` (IN var_prodotto VARCHAR(45), IN var_ditta VARCHAR(45), OUT is_prof INT, OUT is_rr INT)
BEGIN
   declare var_prof tinyint;
   declare var_c int;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;

   set transaction isolation level read committed;
   start transaction;

                select count(`s`.`seriale`)
                from `prodotti` as `p` right join `scatole` as `s` on
                        `p`.`nome_p` = `s`.`prodotto` and `p`.`ditta` = `s`.`ditta`
                where `s`.`prodotto` = var_prodotto and `s`.`ditta` = var_ditta
                and `s`.`vendita` is null and (datediff(`s`.`data_scadenza`, current_date()) > 90)
                into var_c;

                if var_c = 0 or var_c is null then
                        signal sqlstate '45009';
                end if;

                SELECT `profumeria`
                from `prodotti`
                where `nome_p` = var_prodotto and `ditta` = var_ditta
                into var_prof;

                if var_prof = 1 then
                        set is_prof = 1;
                else
                        set is_prof = 0;
                        SELECT  `rr`
                        FROM `medicinali`
                        WHERE `medicinale` = var_prodotto and `ditta` = var_ditta
                        INTO is_rr;
                end if;

   commit;
END
```

Commento
Questa stored procedure viene chiamata dal personale medico in fase di vendita. Viene controllato se il prodotto esiste e se sono presenti scatole di quel prodotto non in scadenza. In caso negativo si lancerà un segnale, in caso affermativo si ritorneranno le informazioni necessarie affinchè avvenga la vendita. Il livello di isolamento utilizzato è read committed, in quanto anche se leggo la stessa tabella più volte, la tabella stessa non può essere soggetta a cambiamenti una volta che è stata inserita, non sorgendo quindi il problema delle unrepeatable reads.

Imposta fatturazione

```
CREATE PROCEDURE `imposta_fatturazione` (in var_citta varchar(45), in var_via varchar(45), in var_civico varchar(11), in var_cap int, in var_ditta varchar(45))
BEGIN

   declare var_addr int;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;


   set transaction isolation level read committed;
   start transaction;

                select count(*)
                from `indirizzi`
                where `ditta` = var_ditta and `citta` = var_citta
                                and `via` = var_via and `civico` = var_civico and `cap` = var_cap
                into var_addr;

                if var_addr = 1 then

                        UPDATE `indirizzi` SET `fatturazione`= true
                                where `ditta` = var_ditta and `citta` = var_citta
                                and `via` = var_via and `civico` = var_civico and `cap` = var_cap;
                else
                        signal sqlstate '45014';
                end if;
   commit;
END
```

Commento
Viene controllato se l'indirizzo è esistente, in caso negativo verrà mandato un segnale. Questa funzione viene chiamata nel caso in cui, al termine dell'inserimento degli indirizzi per una ditta, non sia ancora stato impostato il flag fatturazione, per realizzare la regola aziendale che sia indicato uno ed un solo indirizzo di fatturazione preferito. Se indicato in fase di inserimento, non sarà possibile per inserimenti successivi impostare un altro indirizzo come fatturazione in modo da ottemperare a tale regola.

Imposta preferito

```
CREATE PROCEDURE `imposta_preferito` (in var_contatto varchar(45), in var_ditta varchar(45))
BEGIN

   declare var_com int;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;

   set transaction isolation level read committed;
   start transaction;

                  select count(*)
                  from `comunicazioni`
                  where `ditta` = var_ditta and `contatto` = var_contatto
                  into var_com;

                  if var_com = 1 then
                        UPDATE `comunicazioni` SET `preferito`= true
                                    where `ditta` = var_ditta and `contatto` = var_contatto;
                  else
                        signal sqlstate '45015';
                  end if;

   commit;
END
```

Commento

Stesso concetto applicato nella stored procedure "imposta fatturazione" questa volta per quanto riguarda la tabella "comunicazioni" e per il contatto preferito.

Imposta recapito

```
CREATE PROCEDURE `imposta_recapito` (in var_citta varchar(45), in var_via varchar(45), in var_civico varchar(11), in var_cap int, in var_ditta varchar(45))
BEGIN

   declare var_addr int;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;


   set transaction isolation level read committed;
   start transaction;

                select count(*)
                from `indirizzi`
                where `ditta` = var_ditta and `citta` = var_citta
                                and `via` = var_via and `civico` = var_civico and `cap` = var_cap
                into var_addr;

                if var_addr = 1 then

                        UPDATE `indirizzi` SET `recapito`= true
                                where `ditta` = var_ditta and `citta` = var_citta
                                and `via` = var_via and `civico` = var_civico and `cap` = var_cap;
                else
                        signal sqlstate '45014';
                end if;

   commit;
END
```

Commento

Stesso concetto utilizzato nelle due stored procedure precedenti.

Inserimento Interazione tra categorie

```
CREATE PROCEDURE `inserisci_interazione` (IN var_nome_uno varchar(45), IN var_nome_due varchar(45))
BEGIN
        insert into `interazioni` (`categoria_uno`, `categoria_due`) values (var_nome_uno, var_nome_due);
END
```

Inserisci nuova categoria

```
CREATE PROCEDURE `inserisci_nuova_categoria` (IN var_nome varchar(45))
BEGIN
	insert into `categorie` (`nome`) values (var_nome);

END
```

Inserisci nuova ditta

```
CREATE PROCEDURE `inserisci_nuova_ditta` (IN var_nome varchar(45))
BEGIN
	insert into `ditte` (`nome_d`) values (var_nome);

END
```

Inserisci nuovo contatto

```
CREATE PROCEDURE `inserisci_nuovo_com` (IN var_contatto varchar(45), IN var_tipologia VARCHAR(45), IN var_ditta VARCHAR(45),IN var_pref tinyint)

BEGIN
	insert into `comunicazioni` (`contatto`, `tipologia`, `ditta`, `preferito`)
  values (var_contatto, var_tipologia, var_ditta, var_pref);

END
```

Inserisci nuovo indirizzo

```
CREATE PROCEDURE `inserisci_nuovo_indirizzo` (IN var_citta varchar(45), IN var_via VARCHAR(45), IN var_civico varchar(10), IN var_cap INT, IN var_ditta varchar(45), in var_fat tinyint, in var_rec tinyint)

BEGIN
	insert into `indirizzi` (`citta`, `via`, `civico`, `cap`, `ditta`, `fatturazione`, `recapito`)
  values (var_citta, var_via, var_civico, var_cap, var_ditta, var_fat, var_rec);

END
```

Inserisci nuovo medicinale

```
CREATE PROCEDURE `inserisci_nuovo_medicinale` (IN var_nome varchar(45), IN var_ditta VARCHAR(45), in var_mutuabile int, in var_rr int, in var_cat varchar(45))
BEGIN

        declare exit handler for sqlexception
    begin
      rollback;
      resignal;
    end;

    set transaction isolation level read uncommitted;
    start transaction;

                insert into `prodotti` (`nome_p`, `ditta`, `profumeria`)
                values (var_nome, var_ditta, NULL);

                insert into `medicinali` (`medicinale`, `ditta`, `mutuabile`,`rr`, `categoria`)
                values(var_nome, var_ditta, var_mutuabile, var_rr, var_cat);

    commit;
END
```

Commento

La transazione è stata utilizzata in quanto ritenuta utile nel caso la seconda insert fallisse per fare il rollback, è stato utilizzato il livello di isolamento read uncommitted, il più basso possibile, in quanto gli inserimenti non risentono di anomalie ma al massimo le creano.

Inserisci prodotto di profumeria

```
CREATE PROCEDURE `inserisci_nuovo_profumeria` (IN var_nome varchar(45), IN var_ditta VARCHAR(45))

BEGIN
        insert into `prodotti` (`nome_p`, `ditta`, `profumeria`)
         values (var_nome, var_ditta, true);

END
```

Inserisci nuovo utilizzo

```
CREATE PROCEDURE `inserisci_nuovo_utilizzo` (IN var_nome varchar(45))

BEGIN
        insert into `utilizzi` (`nome_u`) values (var_nome);

END
```

Inserisci previsione

```
CREATE PROCEDURE `inserisci_previsione` (IN var_prodotto varchar(45), IN var_ditta varchar(45), IN var_utilizzo varchar(45))

BEGIN

	insert into `previsioni` (`prodotto`, `ditta`, `utilizzo`) values (var_prodotto, var_ditta, var_utilizzo);

END
```

Lista ordini pendenti

```
CREATE PROCEDURE `lista_ordini_pendenti`()
BEGIN
	set transaction isolation level read committed;
	start transaction;

		SELECT `progressivo`, `data`, `quantita`, `prodotto`, `ditta`
		 FROM `ordini`
		 WHERE `evaso` is null;

	commit;
END
```

Commento

Mostro gli ordini non ancora consegnati, uso read committed in quanto effettuo una semplice lettura.

Login

```
CREATE PROCEDURE `login` (in var_username varchar(45), in var_pass varchar(45), out var_role INT)
BEGIN
	declare var_user_role ENUM('amministratore', 'medico');

	select `ruolo` from `utenti`
		where `username` = var_username
      and `password` = md5(var_pass)
      into var_user_role;

		if var_user_role = 'amministratore' then
			set var_role = 1;
		elseif var_user_role = 'medico' then
			set var_role = 2;
		else
			set var_role = 3;
		end if;
END
```

Mostra categorie

```
CREATE PROCEDURE `mostra_categorie` ()
BEGIN

	set transaction isolation level read committed;
	start transaction;

		select `nome`
		from `categorie`;
	 commit;

END
```

Mostra contatti ditta

```
CREATE PROCEDURE `mostra_com_ditta` (in var_ditta varchar(45))
BEGIN

    set transaction isolation level read committed;
    start transaction;

		select `contatto`, `tipologia`
		from `comunicazioni`
		where `ditta` = var_ditta;


	commit;
END
```

Mostra ditte

```
CREATE PROCEDURE `mostra_ditte` ()
BEGIN
	set transaction isolation level read committed;
	start transaction;

		select `nome_d`
		from `ditte`;

	 commit;
END
```

Mostra indirizzi ditta

```
CREATE PROCEDURE `mostra_indirizzi_ditta` (in var_ditta varchar(45))
BEGIN

   set transaction isolation level read committed;
   start transaction;

                select `citta`, `via`, `civico`, `cap`
                from `indirizzi`
                where `ditta` = var_ditta;
   commit;

END
```

Mostra Posizioni disponibili

```
CREATE PROCEDURE `mostra_posizioni_disponibili` (in var_cat varchar(45))
BEGIN

        set transaction isolation level read committed;
        start transaction;

                select `c`.`scaffale` as `Scaffale`, `c`.`codice_c` as `Cassetto`
                from `scatole`as `s` right join `cassetti` as `c`
                        on (`c`.`codice_c` = `s`.`cassetto` and `c`.`scaffale` = `s`.`scaffale`
                                and `c`.`categoria` = `s`.`categoria`)
                where `c`.`categoria` = var_cat and
                        (`s`.`cassetto` is null and `s`.`scaffale` is null and `s`.`categoria` is null)

                order by `Scaffale`, `Cassetto`;

        commit;
END
```

Commento

Mostra le posizioni disponibili per lo stoccaggio di una scatola, mostrando solo le posizioni di cassetti presenti in scaffali riferenti la stessa categoria a cui appartiene il medicinale riferito dalla scatola.

Mostra Prodotti

```
CREATE PROCEDURE `mostra_prodotti` ()
BEGIN
        set transaction isolation level read committed;
        start transaction;

                select `nome_p`, `ditta`
                from `prodotti`;
        commit;
END
```

Mostra Scatole Pendenti

```
CREATE PROCEDURE `mostra_scatole_pendenti` (in var_prod varchar(45), in var_ditta varchar(45))
BEGIN

   set transaction isolation level read committed;
   start transaction;

            select `seriale`
            from `scatole`
            where `prodotto` = var_prod and `ditta` = var_ditta
                        and `vendita` is null
                        and (datediff(`data_scadenza`, current_date()) > 90);
      commit;
END
```

Commento
Mostro le scatole non ancora vendute che non sono in scadenza.

Mostra Vendite relative rr

```
CREATE PROCEDURE `mostra_venditerr_relative` (in var_prod varchar(45), in var_ditta varchar(45), in var_id int)
BEGIN
      set transaction isolation level read committed;
      start transaction;

            select `idr`, `giorno_p`, `medico`
            from `vendite_medicinalirr`
            where `medicinale` = var_prod and `ditta` = var_ditta
                        and `vendita` = var_id;

      commit;
END
```

Commento
Mostro le vendite di scatole di medicinali richiedenti ricetta per un determinato prodotto relative and una specifica vendita globale.

Registra Scatola medicinale

```
CREATE PROCEDURE `registra_scatola_medicinale` (IN var_prodotto VARCHAR(45), in var_ditta varchar(45), in var_data_scadenza date, in var_prog int, in var_cas int, in var_scaf int, in var_cat varchar(45))

BEGIN
      insert into `scatole` (`prodotto`, `ditta`,`data_scadenza`, `ordine`, `cassetto`, `scaffale`, `categoria`)
        values (var_prodotto, var_ditta, var_data_scadenza, var_prog, var_cas, var_scaf, var_cat);
END
```

Registra scatola prodotto di profumeria

```
CREATE PROCEDURE `registra_scatola_profumeria` (IN var_prodotto VARCHAR(45), in var_ditta varchar(45), in var_data_scadenza date, in var_prog int)

BEGIN

        insert into `scatole` (`prodotto`, `ditta`,`data_scadenza`, `ordine`)
         values (var_prodotto, var_ditta, var_data_scadenza, var_prog);

END
```

Registra Vendita

```
CREATE PROCEDURE `registra_vendita` (out var_id int)

BEGIN
        insert into `vendite` (`giorno`)
        values (current_date());

         set var_id = last_insert_id();

END
```

Registra Vendita rr

```
CREATE PROCEDURE `registra_vendita_rr` (in var_prod varchar(45), in var_ditta varchar(45), in var_saleid int, IN var_date date, IN var_dottore VARCHAR(45), out var_id int)

BEGIN
        insert into `vendite_medicinalirr` (`giorno_p`, `medico`, `medicinale`, `ditta`, `vendita`)
        values (var_date, var_dottore, var_prod, var_ditta, var_saleid);

          set var_id = last_insert_id();

END
```

Report Giacenze medicinali

```
CREATE PROCEDURE `report_giacenze_med` ()
BEGIN

   set transaction isolation level read committed;
   start transaction;

                select `m`.`medicinale`, `m`.`ditta`, count(`s`.`seriale`) as `Rimanenti`
                from `scatole` as `s` right join `medicinali` as `m` on
                        `s`.`prodotto` = `m`.`medicinale` and `s`.`ditta` = `m`.`ditta`
                where `s`.`vendita` is null
                 group by `m`.`medicinale`, `m`.`ditta`
                having `Rimanenti` < 3
                order by `Rimanenti` desc;

   commit;
END
```

Commento
Report sulle giacenze di scatole riferenti medicinali, mostro anche i medicinali che risultano esauriti. Essendo una singola lettura, utilizzo read committed.

Report scatole in scadenza

```
CREATE PROCEDURE `report_scatole_scadenza` ()
BEGIN
   declare var_days int;
   declare var_ser int;
   declare var_ditta varchar(45);
   declare var_prodotto varchar(45);
   declare done int default false;
   declare cur cursor for
        select datediff(`s`.`data_scadenza`,current_date()),`s`.`prodotto`,`s`.`ditta`,`s`.`seriale`
        from `scatole` as `s`
        join `medicinali` as `m` on (`s`.`prodotto` = `m`.`medicinale` and `s`.`ditta` = `m`.`ditta`)
         where `s`.`vendita` is null;

   declare continue handler for not found set done = true;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;
```

```
drop temporary table if exists `scatole_scadenza`;
create temporary table `scatole_scadenza` (
  `seriale` int,
   `prodotto` varchar(45),
   `ditta` varchar(45),
   `cassetto` int,
   `scaffale` int,
   `categoria` varchar(45)

);
set transaction isolation level repeatable read;
start transaction;

        open cur;

        read_loop: loop

            fetch cur into var_days, var_prodotto, var_ditta, var_ser;
            if done then
                leave read_loop;
            end if;

            if (var_days < 90) then

                insert into `scatole_scadenza`
                select `seriale`, `prodotto`, `ditta`, `cassetto`, `scaffale`, `categoria`
                from `scatole`
              where `prodotto`= var_prodotto and `ditta` = var_ditta and  `seriale`=var_ser;

                DELETE FROM `scatole`
        WHERE `prodotto` = var_prodotto and `ditta` = var_ditta and `seriale` = var_ser;

            end if;

        end loop;
        close cur;

        select * from `scatole_scadenza`;

commit;

drop temporary table `scatole_scadenza`;

END
```

Commento
In questo report uso una tabella temporanea per salvarmi le informazioni delle scatole che vado ad eliminare, per poi mostrarle nel report stesso. Ho utilizzato il livello di isolamento repeatable read in modo da avere il lock sia in lettura che in scrittura su tutta la transazione. Non ho usato serializable in quanto non posso avere inserimenti di scatole in scadenza.

Report Vendite Medicinali rr

```
CREATE PROCEDURE `report_vendite_medicinali_rr` ()
BEGIN
        declare var_idr int;
   declare done int default false;
   declare cur cursor for select `idr` from `vendite_medicinalirr`;
   declare continue handler for not found set done = true;

   declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;

   set transaction isolation level repeatable read;
   start transaction;

                open cur;

                read_loop: loop

                        fetch cur into var_idr;
                        if done then
                                leave read_loop;
                        end if;

                        select *
                        from `vendite_medicinalirr`
                        where `idr` = var_idr;

                        select `seriale`
                        from `scatole`
                        where `vendita_medicinalerr` = var_idr;

                end loop;
                close cur;

   commit;

END
```

Commento
Report delle vendite riguardanti scatole riferenti medicinali richiedenti ricetta. Mostro, per ogni vendita di medicinale richiedente ricetta, le informazioni della vendita stessa seguite dall'elenco dei seriali delle scatole vendute in quella transazione. Utilizzo come livello di isolamento repeatable read.

Vendita scatola medicinale

```
CREATE PROCEDURE `vendita_scatola_med` (in var_prodotto varchar(45),in var_ditta varchar(45), in var_seriale int, in cf varchar(20), in var_vendita int, in var_venditarr int)

BEGIN

  declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;


   set transaction isolation level repeatable read;
   start transaction;

                if var_seriale not in
                                        (select `seriale`
                                        from `scatole`
                                        where `prodotto` = var_prodotto and `ditta` = var_ditta
                                                                and `vendita` is null
                      and datediff(`data_scadenza`, current_date()) > 90) then
                          signal sqlstate '45011';
                end if;

                if var_venditarr is not null then
                          if var_venditarr not in (select `idr`
                                                  from `vendite_medicinalirr`
                                        where `medicinale` = var_prodotto and `ditta` = var_ditta
                                                                and `vendita` = var_vendita) then
                                    signal sqlstate '45012';

                          end if;
                end if;

                 UPDATE `scatole` set `vendita` = var_vendita
                                    , `vendita_medicinalerr` = var_venditarr
                                    , `cf_cliente` = cf
                                    , `cassetto` = null
                                    , `scaffale` = null
                                    , `categoria` = null
                 WHERE `prodotto` = var_prodotto and
                                    `ditta` = var_ditta and
                                    `seriale` = var_seriale;

        commit;

END
```

Commento
Quando effettuo una vendita di una scatola di medicinale, controllo se il seriale esiste, se è effettivamente riferito a quel prodotto, se la scatola sia stata già venduta e se la stessa sia già in scadenza. Successivamente, controllo che, nel caso la scatola riferisca un medicinale richiedente ricetta, se la vendita_rr è effettivamente riferita alla vendita globale che sto effettuando e se il prodotto riferito dalla scatola è lo stesso riferito dalla vendita_rr. Nel caso tutti i controlli siano positivi, aggiorno le informazioni della scatola.

Vendita Scatola prodotto profumeria

```
CREATE PROCEDURE `vendita_scatola_prof` (in var_prodotto varchar(45),in var_ditta varchar(45), in var_seriale int, in var_vendita int)
BEGIN

declare exit handler for sqlexception
   begin
      rollback;
      resignal;
   end;

  set transaction isolation level repeatable read;
   start transaction;

                if var_seriale not in
                                        (select `seriale`
                                        from `scatole`
                                        where `prodotto` = var_prodotto and `ditta` = var_ditta
                                                                and `vendita` is null
                     and datediff(`data_scadenza`, current_date()) > 90) then
                          signal sqlstate '45011';
                end if;

                 UPDATE `scatole` set `vendita` = var_vendita
                 WHERE `prodotto` = var_prodotto and
                                `ditta` = var_ditta and
                                `seriale` = var_seriale;

        commit;

END
```

Commento
Quando effettuo una vendita di una scatola di prodotto di profumeria, controllo se il seriale esiste, se è effettivamente riferito a quel prodotto, se la scatola sia stata già venduta e se la stessa sia già in scadenza. Passati i controlli, effettuo l'aggiornamento delle informazioni della scatola.

Appendice: Implementazione

Codice SQL per instanziare il database

```
-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `mydb` ;


-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`ditte`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ditte` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ditte` (
  `nome_d` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`nome_d`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`indirizzi`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`indirizzi` ;

CREATE TABLE IF NOT EXISTS `mydb`.`indirizzi` (
  `via` VARCHAR(45) NOT NULL,
  `citta` VARCHAR(45) NOT NULL,
  `civico` VARCHAR(10) NOT NULL,
  `cap` INT NOT NULL,
  `fatturazione` TINYINT NULL,
  `recapito` TINYINT NULL,
  `ditta` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`via`, `citta`, `civico`, `cap`),
  INDEX `fk_Indirizzo_Ditta_idx` (`ditta` ASC),
  CONSTRAINT `fk_Indirizzo_Ditta`
    FOREIGN KEY (`ditta`)
    REFERENCES `mydb`.`ditte` (`nome_d`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`comunicazioni`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`comunicazioni` ;

CREATE TABLE IF NOT EXISTS `mydb`.`comunicazioni` (
  `contatto` VARCHAR(45) NOT NULL,
  `tipologia` VARCHAR(45) NOT NULL,
  `preferito` TINYINT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`contatto`, `ditta`),
  INDEX `fk_Comunicazione_Ditta1_idx` (`ditta` ASC),
  CONSTRAINT `fk_Comunicazione_Ditta1`
    FOREIGN KEY (`ditta`)
```

```
    REFERENCES `mydb`.`ditte` (`nome_d`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`prodotti`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`prodotti` ;

CREATE TABLE IF NOT EXISTS `mydb`.`prodotti` (
  `nome_p` VARCHAR(45) NOT NULL,
  `profumeria` TINYINT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`nome_p`, `ditta`),
  INDEX `fk_Prodotto_Ditta1_idx` (`ditta` ASC),
  CONSTRAINT `fk_Prodotto_Ditta1`
    FOREIGN KEY (`ditta`)
    REFERENCES `mydb`.`ditte` (`nome_d`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`utilizzi`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`utilizzi` ;

CREATE TABLE IF NOT EXISTS `mydb`.`utilizzi` (
  `nome_u` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`nome_u`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`previsioni`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`previsioni` ;

CREATE TABLE IF NOT EXISTS `mydb`.`previsioni` (
  `prodotto` VARCHAR(45) NOT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  `utilizzo` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`prodotto`, `ditta`, `utilizzo`),
  INDEX `fk_Prodotto_has_Utilizzo_Utilizzo1_idx` (`utilizzo` ASC),
  INDEX `fk_Prodotto_has_Utilizzo_Prodotto1_idx` (`prodotto` ASC, `ditta` ASC),
```

```
  CONSTRAINT `fk_Prodotto_has_Utilizzo_Prodotto1`
    FOREIGN KEY (`prodotto` , `ditta`)
    REFERENCES `mydb`.`prodotti` (`nome_p` , `ditta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prodotto_has_Utilizzo_Utilizzo1`
    FOREIGN KEY (`utilizzo`)
    REFERENCES `mydb`.`utilizzi` (`nome_u`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`categorie`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`categorie` ;

CREATE TABLE IF NOT EXISTS `mydb`.`categorie` (
  `nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`nome`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`medicinali`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`medicinali` ;


CREATE TABLE IF NOT EXISTS `mydb`.`medicinali` (
  `mutuabile` TINYINT NULL,
  `rr` TINYINT NULL,
  `medicinale` VARCHAR(45) NOT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  `categoria` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`medicinale`, `ditta`),
  INDEX `fk_Medicinale_Categoria1_idx` (`categoria` ASC),
  CONSTRAINT `fk_Medicinale_Prodotto1`
    FOREIGN KEY (`medicinale` , `ditta`)
    REFERENCES `mydb`.`prodotti` (`nome_p` , `ditta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Medicinale_Categoria1`
    FOREIGN KEY (`categoria`)
    REFERENCES `mydb`.`categorie` (`nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`interazioni`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`interazioni` ;

CREATE TABLE IF NOT EXISTS `mydb`.`interazioni` (
  `categoria_uno` VARCHAR(45) NOT NULL,
  `categoria_due` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`categoria_uno`, `categoria_due`),
  INDEX `fk_Categoria_has_Categoria_Categoria2_idx` (`categoria_due` ASC),
  INDEX `fk_Categoria_has_Categoria_Categoria1_idx` (`categoria_uno` ASC),
  CONSTRAINT `fk_Categoria_has_Categoria_Categoria1`
    FOREIGN KEY (`categoria_uno`)
    REFERENCES `mydb`.`categorie` (`nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Categoria_has_Categoria_Categoria2`
    FOREIGN KEY (`categoria_due`)
    REFERENCES `mydb`.`categorie` (`nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`scaffali`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`scaffali` ;

CREATE TABLE IF NOT EXISTS `mydb`.`scaffali` (
  `codice_s` INT NOT NULL,
  `categoria` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`codice_s`, `categoria`),
  INDEX `fk_Scaffale_Categoria1_idx` (`categoria` ASC),
  CONSTRAINT `fk_Scaffale_Categoria1`
    FOREIGN KEY (`categoria`)
    REFERENCES `mydb`.`categorie` (`nome`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`cassetti`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`cassetti` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`cassetti` (
  `codice_c` INT NOT NULL,
  `scaffale` INT NOT NULL,
  `categoria` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`codice_c`, `scaffale`, `categoria`),
  INDEX `fk_Cassetto_Scaffale1_idx` (`scaffale` ASC, `categoria` ASC),
  CONSTRAINT `fk_Cassetto_Scaffale1`
    FOREIGN KEY (`scaffale` , `categoria`)
    REFERENCES `mydb`.`scaffali` (`codice_s` , `categoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`ordini`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ordini` ;

CREATE TABLE IF NOT EXISTS `mydb`.`ordini` (
  `progressivo` INT NOT NULL AUTO_INCREMENT,
  `data` DATE NOT NULL,
  `quantita` INT NOT NULL,
```

```
  `prodotto` VARCHAR(45) NOT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  `evaso` TINYINT NULL,
  PRIMARY KEY (`progressivo`),
  INDEX `fk_Ordine_Prodotto1_idx` (`prodotto` ASC, `ditta` ASC),
  CONSTRAINT `fk_Ordine_Prodotto1`
    FOREIGN KEY (`prodotto` , `ditta`)
    REFERENCES `mydb`.`prodotti` (`nome_p` , `ditta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`vendite`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`vendite` ;

CREATE TABLE IF NOT EXISTS `mydb`.`vendite` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `giorno` DATE NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`vendite_medicinalirr`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`vendite_medicinalirr` ;

CREATE TABLE IF NOT EXISTS `mydb`.`vendite_medicinalirr` (
  `idr` INT NOT NULL AUTO_INCREMENT,
  `giorno_p` DATE NOT NULL,
  `medico` VARCHAR(45) NOT NULL,
  `vendita` INT NOT NULL,
  `medicinale` VARCHAR(45) NOT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idr`),
  INDEX `fk_VenditaMedicinaleRR_Vendita1_idx` (`vendita` ASC),
  INDEX `fk_VenditaMedicinaleRR_Medicinale1_idx` (`medicinale` ASC, `ditta` ASC),
  CONSTRAINT `fk_VenditaMedicinaleRR_Vendita1`
    FOREIGN KEY (`vendita`)
    REFERENCES `mydb`.`vendite` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_VenditaMedicinaleRR_Medicinale1`
    FOREIGN KEY (`medicinale` , `ditta`)
    REFERENCES `mydb`.`medicinali` (`medicinale` , `ditta`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`scatole`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`scatole` ;

CREATE TABLE IF NOT EXISTS `mydb`.`scatole` (
  `seriale` INT NOT NULL AUTO_INCREMENT,
  `data_scadenza` DATE NOT NULL,
  `prodotto` VARCHAR(45) NOT NULL,
  `ditta` VARCHAR(45) NOT NULL,
  `ordine` INT NOT NULL,
  `vendita` INT NULL,
  `vendita_medicinalerr` INT NULL,
  `cassetto` INT NULL,
  `scaffale` INT NULL,
  `categoria` VARCHAR(45) NULL,
  `cf_cliente` VARCHAR(20) NULL,
  PRIMARY KEY (`seriale`, `prodotto`, `ditta`),
  INDEX `fk_Scatola_Prodotto1_idx` (`prodotto` ASC, `ditta` ASC),
```

```
  INDEX `fk_Scatola_Ordine1_idx` (`ordine` ASC),
  INDEX `fk_Scatola_Vendita1_idx` (`vendita` ASC),
  INDEX `fk_Scatola_VenditaMedicinaleRR1_idx` (`vendita_medicinalerr` ASC),
  INDEX `fk_Scatola_Cassetto1_idx` (`cassetto` ASC, `scaffale` ASC, `categoria` ASC),
  CONSTRAINT `fk_Scatola_Prodotto1`
    FOREIGN KEY (`prodotto` , `ditta`)
    REFERENCES `mydb`.`prodotti` (`nome_p` , `ditta`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Scatola_Ordine1`
    FOREIGN KEY (`ordine`)
    REFERENCES `mydb`.`ordini` (`progressivo`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Scatola_Vendita1`
    FOREIGN KEY (`vendita`)
    REFERENCES `mydb`.`vendite` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Scatola_VenditaMedicinaleRR1`
    FOREIGN KEY (`vendita_medicinalerr`)
    REFERENCES `mydb`.`vendite_medicinalirr` (`idr`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
```

```
  CONSTRAINT `fk_Scatola_Cassetto1`
    FOREIGN KEY (`cassetto` , `scaffale` , `categoria`)
    REFERENCES `mydb`.`cassetti` (`codice_c` , `scaffale` , `categoria`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`utenti`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`utenti` ;

CREATE TABLE IF NOT EXISTS `mydb`.`utenti` (
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NULL,
  `ruolo` ENUM("amministratore", "medico") NULL,
  PRIMARY KEY (`username`))
ENGINE = InnoDB;

SET SQL_MODE = '';
GRANT USAGE ON *.* TO login;
 DROP USER login;
```

```
SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'login' IDENTIFIED BY 'login';


GRANT EXECUTE ON procedure `mydb`.`login` TO 'login';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO medico;

 DROP USER medico;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'medico' IDENTIFIED BY 'medico';


GRANT EXECUTE ON procedure `mydb`.`inserisci_nuovo_medicinale` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuovo_profumeria` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuova_categoria` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`mostra_categorie` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`inserisci_interazione` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`registra_vendita` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`get_product_info` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`mostra_scatole_pendenti` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`vendita_scatola_prof` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`mostra_venditerr_relative` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`registra_vendita_rr` TO 'medico';
```

```
GRANT EXECUTE ON procedure `mydb`.`vendita_scatola_med` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`mostra_prodotti` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`report_vendite_medicinali_rr` TO 'medico';

GRANT EXECUTE ON procedure `mydb`.`mostra_ditte` TO 'medico';

SET SQL_MODE = '';

GRANT USAGE ON *.* TO amministratore;

 DROP USER amministratore;

SET
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO
_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE USER 'amministratore' IDENTIFIED BY 'amministratore';


GRANT EXECUTE ON procedure `mydb`.`crea_ordine` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`lista_ordini_pendenti` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`get_order_info` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`registra_scatola_profumeria` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`registra_scatola_medicinale` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuova_ditta` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuovo_indirizzo` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuovo_com` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`inserisci_nuovo_utilizzo` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`mostra_prodotti` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`inserisci_previsione` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`report_scatole_scadenza` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`report_vendite_medicinali_rr` TO 'amministratore';
```

```
GRANT EXECUTE ON procedure `mydb`.`report_giacenze_med` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`mostra_posizioni_disponibili` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`imposta_recapito` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`mostra_indirizzi_ditta` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`imposta_fatturazione` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`mostra_com_ditta` TO 'amministratore';

GRANT EXECUTE ON procedure `mydb`.`imposta_preferito` TO 'amministratore';


SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;


-- ----------------------------------------------------

-- Data for table `mydb`.`ditte`

-- ----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('pfizer');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('bayer');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('rabanne');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('gucci');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('astrazeneca');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('dior');

INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('menarini');
```

```
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('molteni');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('zambon');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('angelini');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('pharma');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('kedrion');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('savio');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('italfarmaco');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('alfasigma');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('recordati');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('mediolanum');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('rottapharm');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('guidotti');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('falqui');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('ogna');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('johnson&johnson');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('moderna');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('collistar');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('avon');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('pupa');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('vichy');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('clinique');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('shiseido');
INSERT INTO `mydb`.`ditte` (`nome_d`) VALUES ('oreal');
COMMIT;
```

```
-- -----------------------------------------------------

-- Data for table `mydb`.`indirizzi`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('canneto di rodi', 'borgo sabotino', '1', 04100, NULL, NULL, 'bayer');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('industriale', 'montoro', '12', 05035, 1, 1, 'bayer');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('ludovico il moro', 'basiglio', '6c', 20080, 1, 1, 'astrazeneca');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('ulivo', 'roma', '22', 00187, 1, 1, 'dior');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('tito speri', 'pomezia', '12', 00040, 1, NULL, 'menarini');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('sette santi', 'firenze', '1', 50131, NULL, 1, 'menarini');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('ilio barontini', 'firenze', '8', 50018, 1, 1, 'molteni');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('lillo del duca', 'bresso', '10', 20091, 1, 1, 'zambon');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('guardapasso', 'aprilia', '1', 04011, 1, NULL, 'angelini');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('francesco angelini', 'pomezia', '1', 00040, NULL, NULL, 'angelini');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('torrenova', 'roma', '435', 00133, NULL, 1, 'angelini');
```

```
INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('reali', 'pomezia', '5', 04011, 1, 1, 'pharma');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('vittorio veneto', 'roma', '7', 00187, 1, NULL, 'kedrion');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('per il ciocco', 'barga', '1', 55051, NULL, 1, 'kedrion');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('del mare', 'pomezia', '36', 00071, 1, 1, 'savio');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('belluno', 'roma', '1', 00161, 1, 1, 'italfarmaco');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('fulvio testi', 'milano', '330', 20126, NULL, NULL, 'italfarmaco');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('enrico fermi', 'alanno', '1', 65020, 1, NULL, 'alfasigma');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('pontinia', 'pomezia', '30', 00071, NULL, 1, 'alfasigma');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('mediana', 'aprilia', '4', 04011, 1, 1, 'recordati');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('giuseppe cottolengo', 'milano', '15', 20143, 1, 1, 'mediolanum');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('valosa di sopra', 'monza', '9', 20900, 1, 1, 'rottapharm');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('guglielmo marconi', 'sustinente', '268', 46030, 1, 1, 'guidotti');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('fabio filzi', 'milano', '8', 20124, 1, NULL, 'falqui');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('roma', 'recco', '8', 16036, NULL, 1, 'falqui');
```

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('figini', 'muggio', '41', 20835, 1, 1, 'ogna');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('ardeatina', 'pomezia', '23', 00071, 1, NULL, 'johnson&johnson');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('janssen', 'latina', '3', 04100, NULL, 1, 'johnson&johnson');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('corso', 'fiuggi', '51', 03014, 1, NULL, 'rabanne');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('vittorio emanuele', 'anagni', '144', 03012, NULL, 1, 'rabanne');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('condotti', 'roma', '8', 00187, 1, 1, 'gucci');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('tritone', 'roma', '61', 00187, NULL, NULL, 'gucci');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('francesco di fuori', 'alatri', '2', 03011, 1, NULL, 'moderna');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('brighindi', 'frosinone', '23', 03100, NULL, 1, 'moderna');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('capo i prati', 'fiuggi', '60', 03014, 1, 1, 'collistar');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('frattarotonda vado largo', 'anagni', '1', 03012, 1, NULL, 'avon');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('del municipio', 'spoleto', '2', 06049, NULL, 1, 'avon');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('della pace', 'valmontone', '1', 00038, 1, NULL, 'pupa');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('ss87', 'marcianise', '3', 81020, NULL, 1, 'pupa');

```
INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('del comune', 'trevi nel lazio', '34', 03010, 1, 1, 'vichy');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('tritone', 'roma', '62', 00187, 1, 1, 'clinique');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('fontego dei tedeschi', 'venezia', '4', 30100, NULL, 1, 'shiseido');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('eliseo', 'isola liri', '230', 03036, 1, NULL, 'shiseido');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('della pace', 'valmontone', '23', 00038, NULL, NULL, 'oreal');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('mignanelli', 'roma', '23', 00187, 1, 1, 'oreal');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('valbondione', 'roma', '113', 00188, 1, 1, 'pfizer');

INSERT INTO `mydb`.`indirizzi` (`via`, `citta`, `civico`, `cap`, `fatturazione`, `recapito`, `ditta`) VALUES ('del commercio', 'ascoli piceno', '25', 63100, NULL, NULL, 'pfizer');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`comunicazioni`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('astra@gmail', 'mail', 1, 'astrazeneca');
```

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3343554690', 'telefono', NULL, 'astrazeneca');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('pfizer@hotmail.com', 'mail', 1, 'pfizer');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3324323456', 'telefono', NULL, 'pfizer');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345235322', 'fax', NULL, 'pfizer');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('bayer@mail.com', 'mail', NULL, 'bayer');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3343234322', 'telefono', 1, 'bayer');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('pacorabanne@gmail.com', 'mail', 1, 'rabanne');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('guccicontacts@mail.com', 'mail', 1, 'gucci');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('diorsales@mail.com', 'mail', 1, 'dior');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('menarini@menarini.it', 'mail', 1, 'menarini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3343434332', 'telefono', NULL, 'menarini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('molteni@molteni.it', 'mail', 1, 'molteni');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('zambon@hotmail.com', 'mail', 1, 'zambon');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('angelinicontacts@angelini.it', 'mail', 1, 'angelini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('angelinisales@angelini.it', 'mail', NULL, 'angelini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3348909652', 'telefono', NULL, 'angelini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3423456788', 'fax', NULL, 'angelini');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('pharma@mail.it', 'mail', 1, 'pharma');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('kedrionitaly@gmail.com', 'mail', 1, 'kedrion');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('savio@itmail.it', 'mail', 1, 'savio');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('italfarmacocontacts@italfarmaco.it', 'mail', 1, 'italfarmaco');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('0776876765', 'telefono', NULL, 'italfarmaco');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('alfasigma@gmail.com', 'mail', 1, 'alfasigma');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('recordati@rec.com', 'mail', 1, 'recordati');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('medcontacts@mediolanum.it', 'mail', 1, 'mediolanum');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('rottapharm@mail.com', 'mail', 1, 'rottapharm');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('guidotticontacts@gmail.com', 'mail', 1, 'guidotti');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('falqui@falqui.it', 'mail', 1, 'falqui');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('33456763745', 'telefono', NULL, 'falqui');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345672345', 'fax', NULL, 'falqui');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('ogna@outlook.com', 'mail', 1, 'ogna');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('johnson&johnson@jj.com', 'mail', 1, 'johnson&johnson');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3343456765', 'telefono', NULL, 'johnson&johnson');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3356765675', 'fax', NULL, 'johnson&johnson');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('moderna@moderna.it', 'mail', 1, 'moderna');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3323434567', 'telefono', NULL, 'moderna');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('collistar@collistar.it', 'mail', 1, 'collistar');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3376876453', 'telefono', NULL, 'collistar');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('avonsales@avon.com', 'mail', 1, 'avon');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345434566', 'telefono', NULL, 'avon');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('pupamilano@pupa.it', 'mail', 1, 'pupa');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345434567', 'telefono', NULL, 'pupa');

```
INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('vichy@mail.fr', 'mail', 1, 'vichy');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('clinique@gmail.com', 'mail', 1, 'clinique');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('shiseido@shiseido.it', 'mail', 1, 'shiseido');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345445678', 'telefono', NULL, 'shiseido');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3345677875', 'fax', NULL, 'shiseido');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('oreal@oreal.fr', 'mail', 1, 'oreal');

INSERT INTO `mydb`.`comunicazioni` (`contatto`, `tipologia`, `preferito`, `ditta`) VALUES ('3347657876', 'telefono', NULL, 'oreal');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`prodotti`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('aspirina', NULL, 'bayer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('one million', 1, 'rabanne');
```

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('comirnaty', NULL, 'pfizer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('uomo', 1, 'gucci');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('oki', NULL, 'bayer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('amlodipina', NULL, 'pfizer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('azitromicina', NULL, 'pfizer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('xanax', NULL, 'pfizer');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('vaxzevria', NULL, 'astrazeneca');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('elvive wonder', 1, 'oreal');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('revitalift', 1, 'oreal');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('whale', 1, 'pupa');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('bois argent', 1, 'dior');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('prostamol', NULL, 'menarini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('sustenium plus', NULL, 'menarini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('vivinduo', NULL, 'menarini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('tantum verde', NULL, 'angelini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('moment act', NULL, 'angelini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('tachipirina', NULL, 'angelini');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('uomo', 1, 'dior');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('decadron', NULL, 'savio');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('inofert plus', NULL, 'italfarmaco');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('biochetasi', NULL, 'alfasigma');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('smoothing body', 1, 'shiseido');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('extra rich', 1, 'shiseido');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('gel anticellulite', 1, 'collistar');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('gel anticellulite', 1, 'vichy');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('superabbronzante', 1, 'collistar');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('crema termale', 1, 'shiseido');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('cargo cardio', NULL, 'recordati');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('glucocare', NULL, 'falqui');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('crema termale', 1, 'collistar');

```
INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('janssen', NULL, 'johnson&johnson');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('nizoral', 1, 'johnson&johnson');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('spikevax', NULL, 'moderna');

INSERT INTO `mydb`.`prodotti` (`nome_p`, `profumeria`, `ditta`) VALUES ('crema termale', 1, 'avon');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`utilizzi`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('malattie da raffreddamento');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('dolori alle ossa');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('detergente per viso');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('detergente per corpo');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('stress');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('pressione alta');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('detergente per capelli');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('mal di testa');
```

```
INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('fragranza');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('prevenzione covid');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('scompensi alimentari');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('mal di gola');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('problemi intestinali');

INSERT INTO `mydb`.`utilizzi` (`nome_u`) VALUES ('malattie cardiovascolari');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`previsioni`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('amlodipina', 'pfizer', 'pressione alta');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('azitromicina', 'pfizer', 'malattie da raffreddamento');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('revitalift', 'oreal', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('elvive wonder', 'oreal', 'detergente per capelli');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('whale', 'pupa', 'detergente per corpo');
```

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('aspirina', 'bayer', 'malattie da raffreddamento');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('aspirina', 'bayer', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('oki', 'bayer', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('oki', 'bayer', 'stress');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('one million', 'rabanne', 'fragranza');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('uomo', 'gucci', 'fragranza');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('comirnaty', 'pfizer', 'prevenzione covid');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('vaxzevria', 'astrazeneca', 'prevenzione covid');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('bois argent', 'dior', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('sustenium plus', 'menarini', 'scompensi alimentari');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('prostamol', 'menarini', 'scompensi alimentari');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('vivinduo', 'menarini', 'malattie da raffreddamento');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('tantum verde', 'angelini', 'mal di gola');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('moment act', 'angelini', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('moment act', 'angelini', 'stress');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('tachipirina', 'angelini', 'malattie da raffreddamento');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('tachipirina', 'angelini', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('uomo', 'dior', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('decadron', 'savio', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('inofert plus', 'italfarmaco', 'mal di testa');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('biochetasi', 'alfasigma', 'problemi intestinali');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('smoothing body', 'shiseido', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('extra rich', 'shiseido', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('gel anticellulite', 'vichy', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('gel anticellulite', 'collistar', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('crema termale', 'collistar', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('crema termale', 'shiseido', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('superabbronzante', 'collistar', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('cargo cardio', 'recordati', 'malattie cardiovascolari');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('glucocare', 'falqui', 'malattie cardiovascolari');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('janssen', 'johnson&johnson', 'prevenzione covid');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('nizoral', 'johnson&johnson', 'detergente per corpo');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('spikevax', 'moderna', 'prevenzione covid');

INSERT INTO `mydb`.`previsioni` (`prodotto`, `ditta`, `utilizzo`) VALUES ('crema termale', 'avon', 'detergente per corpo');

COMMIT;

-- ---

-- Data for table `mydb`.`categorie`

-- ---

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antibiotico');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('vaccino');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antinfluenzale');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antiinfiammatorio');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('ansiolitico');

```
INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antiipertensivo');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('integratore alimentare');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antipiretico');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('vitamina minerale');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antiaggregante');

INSERT INTO `mydb`.`categorie` (`nome`) VALUES ('antidiabetico');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`medicinali`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'aspirina', 'bayer', 'antiinfiammatorio');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'comirnaty', 'pfizer', 'vaccino');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'oki', 'bayer', 'antiinfiammatorio');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'xanax', 'pfizer', 'ansiolitico');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'azitromicina', 'pfizer', 'antibiotico');
```

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'amlodipina', 'pfizer', 'antiipertensivo');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'vaxzevria', 'astrazeneca', 'vaccino');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'prostamol', 'menarini', 'integratore alimentare');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'vivinduo', 'menarini', 'antibiotico');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'sustenium plus', 'menarini', 'integratore alimentare');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'tantum verde', 'angelini', 'antiinfiammatorio');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'moment act', 'angelini', 'antibiotico');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'tachipirina', 'angelini', 'antipiretico');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, NULL, 'decadron', 'savio', 'antiinfiammatorio');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'inofert plus', 'italfarmaco', 'integratore alimentare');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (NULL, NULL, 'biochetasi', 'alfasigma', 'vitamina minerale');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'cargo cardio', 'recordati', 'antiaggregante');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'glucocare', 'falqui', 'antidiabetico');

INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'janssen', 'johnson&johnson', 'vaccino');

```
INSERT INTO `mydb`.`medicinali` (`mutuabile`, `rr`, `medicinale`, `ditta`, `categoria`) VALUES (1, 1, 'spikevax', 'moderna', 'vaccino');


COMMIT;



-- -----------------------------------------------------
-- Data for table `mydb`.`interazioni`
-- -----------------------------------------------------
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antibiotico', 'antiinfiammatorio');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antibiotico', 'antinfluenzale');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('ansiolitico', 'antiipertensivo');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antiaggregante', 'antidiabetico');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antipiretico', 'antinfluenzale');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antinfluenzale', 'antiinfiammatorio');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('vaccino', 'antinfluenzale');
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antiinfiammatorio', 'antiaggregante');
```

```
INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('vitamina minerale', 'integratore alimentare');

INSERT INTO `mydb`.`interazioni` (`categoria_uno`, `categoria_due`) VALUES ('antipiretico', 'antiinfiammatorio');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`scaffali`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'vaccino');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (2, 'antibiotico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (3, 'antinfluenzale');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (2, 'vaccino');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (3, 'vaccino');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antiinfiammatorio');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antibiotico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antipiretico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antiaggregante');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'vitamina minerale');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'integratore alimentare');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antiipertensivo');
```

```
INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (2, 'antiipertensivo');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'ansiolitico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (2, 'ansiolitico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (2, 'antiinfiammatorio');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (1, 'antidiabetico');

INSERT INTO `mydb`.`scaffali` (`codice_s`, `categoria`) VALUES (3, 'antibiotico');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`cassetti`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 2, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 3, 'antinfluenzale');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 2, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 2, 'antibiotico');
```

```
INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 2, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 3, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 3, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 3, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (6, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (7, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (8, 1, 'vaccino');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antipiretico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antipiretico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antipiretico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'antipiretico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 1, 'antipiretico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antiaggregante');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antiaggregante');
```

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antiaggregante');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'antiaggregante');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 1, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 2, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 2, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 2, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 2, 'ansiolitico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'vitamina minerale');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'vitamina minerale');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'vitamina minerale');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 2, 'antiipertensivo');

```
INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 2, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 2, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 2, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 2, 'antiipertensivo');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (6, 1, 'integratore alimentare');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 2, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 2, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 2, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 2, 'antiinfiammatorio');
```

```
INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 2, 'antiinfiammatorio');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 1, 'antidiabetico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 1, 'antidiabetico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 1, 'antidiabetico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 1, 'antidiabetico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (1, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (2, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (3, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (4, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (5, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (6, 3, 'antibiotico');

INSERT INTO `mydb`.`cassetti` (`codice_c`, `scaffale`, `categoria`) VALUES (7, 3, 'antibiotico');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`ordini`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (1, '2021-07-08', 5, 'comirnaty', 'pfizer', 1);
```

```
INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (2, '2021-06-04', 8, 'one million', 'rabanne', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (3, '2021-07-26', 3, 'aspirina', 'bayer', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (5, '2021-02-20', 6, 'comirnaty', 'pfizer', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (6, '2021-12-08', 5, 'oki', 'bayer', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (7, '2021-04-23', 3, 'uomo', 'gucci', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (8, '2021-03-21', 2, 'crema termale', 'shiseido', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (9, '2021-03-21', 2, 'crema termale', 'collistar', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (10, '2021-03-27', 3, 'bois argent', 'dior', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (4, '2020-09-23', 2, 'xanax', 'pfizer', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (11, '2021-07-21', 2, 'amlodipina', 'pfizer', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (12, '2021-08-20', 3, 'glucocare', 'falqui', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (13, '2021-07-30', 2, 'biochetasi', 'alfasigma', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (14, '2021-08-14', 3, 'prostamol', 'menarini', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (15, '2021-08-24', 2, 'vivinduo', 'menarini', 1);
```

```
INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (16, '2021-08-21', 4, 'tantum verde', 'angelini', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (17, '2021-07-30', 2, 'janssen', 'johnson&johnson', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (18, '2021-08-15', 3, 'sustenium plus', 'menarini', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (19, '2021-08-20', 3, 'tachipirina', 'angelini', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (20, '2021-08-22', 5, 'tachipirina', 'angelini', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (21, '2021-08-23', 2, 'inofert plus', 'italfarmaco', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (22, '2021-07-23', 2, 'superabbronzante', 'collistar', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (23, '2021-06-28', 3, 'nizoral', 'johnson&johnson', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (24, '2021-08-23', 3, 'crema termale', 'avon', NULL);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (25, '2021-07-27', 3, 'extra rich', 'shiseido', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (26, '2021-08-17', 2, 'cargo cardio', 'recordati', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (27, '2021-03-17', 2, 'sustenium plus', 'menarini', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (28, '2021-07-23', 1, 'prostamol', 'menarini', 1);

INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (29, '2021-08-21', 3, 'moment act', 'angelini', 1);
```

```
INSERT INTO `mydb`.`ordini` (`progressivo`, `data`, `quantita`, `prodotto`, `ditta`, `evaso`) VALUES (30, '2021-08-24', 4, 'azitromicina', 'pfizer', 1);


COMMIT;



-- -----------------------------------------------------
-- Data for table `mydb`.`vendite`
-- -----------------------------------------------------
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (1, '2021-08-17');
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (2, '2021-08-15');
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (3, '2021-08-23');
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (4, '2021-08-24');
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (5, '2021-08-25');
INSERT INTO `mydb`.`vendite` (`id`, `giorno`) VALUES (6, '2021-08-25');

COMMIT;



-- -----------------------------------------------------
-- Data for table `mydb`.`vendite_medicinalirr`
-- -----------------------------------------------------
START TRANSACTION;
```

```
USE `mydb`;

INSERT INTO `mydb`.`vendite_medicinalirr` (`idr`, `giorno_p`, `medico`, `vendita`, `medicinale`, `ditta`) VALUES (1, '2021-08-14', 'francini', 1, 'comirnaty', 'pfizer');

INSERT INTO `mydb`.`vendite_medicinalirr` (`idr`, `giorno_p`, `medico`, `vendita`, `medicinale`, `ditta`) VALUES (2, '2021-08-11', 'lorenzi', 2, 'comirnaty', 'pfizer');

INSERT INTO `mydb`.`vendite_medicinalirr` (`idr`, `giorno_p`, `medico`, `vendita`, `medicinale`, `ditta`) VALUES (3, '2021-08-18', 'francesi', 3, 'amlodipina', 'pfizer');

INSERT INTO `mydb`.`vendite_medicinalirr` (`idr`, `giorno_p`, `medico`, `vendita`, `medicinale`, `ditta`) VALUES (4, '2021-08-23', 'lorenzi', 4, 'glucocare', 'falqui');

INSERT INTO `mydb`.`vendite_medicinalirr` (`idr`, `giorno_p`, `medico`, `vendita`, `medicinale`, `ditta`) VALUES (5, '2021-08-24', 'crecco', 6, 'azitromicina', 'pfizer');


COMMIT;



-- -----------------------------------------------------

-- Data for table `mydb`.`scatole`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-10-24', 'one million', 'rabanne', 2, 1, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-10-24', 'one million', 'rabanne', 2, 1, NULL, NULL, NULL, NULL, NULL);
```

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (4, '2022-10-24', 'one million', 'rabanne', 2, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (5, '2022-10-24', 'one million', 'rabanne', 2, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-10-24', 'one million', 'rabanne', 2, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-08', 'comirnaty', 'pfizer', 1, NULL, NULL, 1, 1, 'vaccino', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-10-02', 'comirnaty', 'pfizer', 1, NULL, NULL, 2, 1, 'vaccino', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-03-09', 'comirnaty', 'pfizer', 1, NULL, NULL, 3, 1, 'vaccino', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (4, '2022-09-08', 'comirnaty', 'pfizer', 1, NULL, NULL, 4, 1, 'vaccino', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-06', 'aspirina', 'bayer', 3, NULL, NULL, 1, 2, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-09-09', 'aspirina', 'bayer', 3, NULL, NULL, 2, 2, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-06-02', 'aspirina', 'bayer', 3, NULL, NULL, 3, 2, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (5, '2022-05-21', 'comirnaty', 'pfizer', 5, 1, 1, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (6, '2022-06-23', 'comirnaty', 'pfizer', 5, 1, 1, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (7, '2023-09-12', 'comirnaty', 'pfizer', 5, 1, 2, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (8, '2022-09-03', 'comirnaty', 'pfizer', 5, 1, 2, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-23', 'oki', 'bayer', 6, NULL, NULL, 1, 1, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-09-12', 'oki', 'bayer', 6, NULL, NULL, 2, 1, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2023-12-09', 'oki', 'bayer', 6, NULL, NULL, 3, 1, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (9, '2024-09-12', 'comirnaty', 'pfizer', 5, NULL, NULL, 6, 1, 'vaccino', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2023-08-12', 'crema termale', 'shiseido', 8, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-21', 'xanax', 'pfizer', 4, NULL, NULL, 1, 1, 'ansiolitico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-09-21', 'xanax', 'pfizer', 4, NULL, NULL, 2, 1, 'ansiolitico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-08-21', 'amlodipina', 'pfizer', 11, 3, 3, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-08-21', 'amlodipina', 'pfizer', 11, 3, 3, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2021-12-12', 'biochetasi', 'alfasigma', 13, 3, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2021-12-12', 'biochetasi', 'alfasigma', 13, 3, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-02-14', 'vivinduo', 'menarini', 15, NULL, NULL, 1, 1, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-02-14', 'vivinduo', 'menarini', 15, NULL, NULL, 2, 1, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-01-14', 'tantum verde', 'angelini', 16, NULL, NULL, 1, 2, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-01-14', 'tantum verde', 'angelini', 16, NULL, NULL, 2, 2, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-01-14', 'tantum verde', 'angelini', 16, NULL, NULL, 3, 2, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (4, '2022-01-14', 'tantum verde', 'angelini', 16, NULL, NULL, 4, 2, 'antiinfiammatorio', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-06-02', 'glucocare', 'falqui', 12, 4, 4, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-06-02', 'glucocare', 'falqui', 12, 4, 4, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-06-02', 'glucocare', 'falqui', 12, NULL, NULL, 1, 1, 'antidiabetico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-02-03', 'tachipirina', 'angelini', 19, 4, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-02-03', 'tachipirina', 'angelini', 19, 4, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-02-03', 'tachipirina', 'angelini', 19, NULL, NULL, 2, 1, 'antipiretico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-21', 'superabbronzante', 'collistar', 22, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-09-21', 'superabbronzante', 'collistar', 22, 4, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2023-09-20', 'nizoral', 'johnson&johnson', 23, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2023-09-20', 'nizoral', 'johnson&johnson', 23, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2023-09-20', 'nizoral', 'johnson&johnson', 23, 5, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-09-23', 'extra rich', 'shiseido', 25, 5, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2023-09-23', 'extra rich', 'shiseido', 25, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2023-09-23', 'extra rich', 'shiseido', 25, NULL, NULL, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2024-08-20', 'cargo cardio', 'recordati', 26, NULL, NULL, 1, 1, 'antiaggregante', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2024-08-20', 'cargo cardio', 'recordati', 26, NULL, NULL, 2, 1, 'antiaggregante', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-02-21', 'sustenium plus', 'menarini', 27, NULL, NULL, 1, 1, 'integratore alimentare', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-02-21', 'sustenium plus', 'menarini', 27, NULL, NULL, 2, 1, 'integratore alimentare', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2023-01-21', 'prostamol', 'menarini', 28, NULL, NULL, 3, 1, 'integratore alimentare', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2022-04-09', 'moment act', 'angelini', 29, NULL, NULL, 1, 3, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2022-04-09', 'moment act', 'angelini', 29, NULL, NULL, 2, 3, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2022-04-09', 'moment act', 'angelini', 29, NULL, NULL, 3, 3, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (1, '2023-09-21', 'azitromicina', 'pfizer', 30, NULL, NULL, 4, 3, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (2, '2023-09-21', 'azitromicina', 'pfizer', 30, NULL, NULL, 5, 3, 'antibiotico', NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (3, '2023-09-21', 'azitromicina', 'pfizer', 30, 6, 5, NULL, NULL, NULL, NULL);

INSERT INTO `mydb`.`scatole` (`seriale`, `data_scadenza`, `prodotto`, `ditta`, `ordine`, `vendita`, `vendita_medicinalerr`, `cassetto`, `scaffale`, `categoria`, `cf_cliente`) VALUES (4, '2023-09-21', 'azitromicina', 'pfizer', 30, 6, 5, NULL, NULL, NULL, NULL);

COMMIT;

```
-- -----------------------------------------------------

-- Data for table `mydb`.`utenti`

-- -----------------------------------------------------

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('ludovico', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('valerio', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('giovanni', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('luca', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('marta', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('giulia', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('rebecca', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('carlotta', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('lorenzo', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('giuseppe', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'amministratore');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('enzo', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');
```

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('bianca', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');

INSERT INTO `mydb`.`utenti` (`username`, `password`, `ruolo`) VALUES ('alessandro', 'c5c3cf1ef4ad9222a9a1203e94b5b4d7', 'medico');

COMMIT;

Codice del Front-End

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mysql.h>

#include "defines.h"


typedef enum {
	ADMINISTRATOR = 1,
	DOCTOR,
	FAILED_LOGIN
} role_t;


struct configuration conf;

static MYSQL *conn;


static role_t attempt_login(MYSQL *conn, char *username, char *password) {
	MYSQL_STMT *login_procedure;

	MYSQL_BIND param[3]; // Used both for input and output
	int role = 0;

	if(!setup_prepared_stmt(&login_procedure, "call login(?, ?, ?)", conn)) {
		print_stmt_error(login_procedure, "Unable to initialize login statement\n");
		goto err2;
	}

	// Prepare parameters
	memset(param, 0, sizeof(param));
```

```
        param[0].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
        param[0].buffer = username;
        param[0].buffer_length = strlen(username);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING; // IN
        param[1].buffer = password;
        param[1].buffer_length = strlen(password);

        param[2].buffer_type = MYSQL_TYPE_LONG; // OUT
        param[2].buffer = &role;
        param[2].buffer_length = sizeof(role);

        if (mysql_stmt_bind_param(login_procedure, param) != 0) { // Note _param
                print_stmt_error(login_procedure, "Could not bind parameters for login");
                goto err;
        }

        // Run procedure
        if (mysql_stmt_execute(login_procedure) != 0) {
                print_stmt_error(login_procedure, "Could not execute login procedure");
                goto err;
        }

        // Prepare output parameters
        memset(param, 0, sizeof(param));
        param[0].buffer_type = MYSQL_TYPE_LONG; // OUT
        param[0].buffer = &role;
        param[0].buffer_length = sizeof(role);

        if(mysql_stmt_bind_result(login_procedure, param)) {
                print_stmt_error(login_procedure, "Could not retrieve output parameter");
                goto err;
        }

        // Retrieve output parameter
        if(mysql_stmt_fetch(login_procedure)) {
                print_stmt_error(login_procedure, "Could not buffer results");
                goto err;
        }

        mysql_stmt_close(login_procedure);
        return role;

    err:
        mysql_stmt_close(login_procedure);
    err2:
        return FAILED_LOGIN;
}
```

```
int main(void) {
	role_t role;

	if(!parse_config("users/login.json", &conf)) {
		fprintf(stderr, "Unable to load login configuration\n");
		exit(EXIT_FAILURE);
	}

	conn = mysql_init (NULL);
	if (conn == NULL) {
		fprintf (stderr, "mysql_init() failed (probably out of memory)\n");
		exit(EXIT_FAILURE);
	}

	if (mysql_real_connect(conn, conf.host, conf.db_username, conf.db_password, conf.database,
conf.port, NULL, CLIENT_MULTI_STATEMENTS | CLIENT_MULTI_RESULTS) == NULL) {
		fprintf (stderr, "mysql_real_connect() failed\n");
		mysql_close (conn);
		exit(EXIT_FAILURE);
	}

	printf("Username: ");
	getInput(128, conf.username, false);
	printf("Password: ");
	getInput(128, conf.password, true);

	role = attempt_login(conn, conf.username, conf.password);

	switch(role) {
		case ADMINISTRATOR:
			run_as_administrator(conn);
			break;

		case DOCTOR:
			run_as_doctor(conn);
			break;

		case FAILED_LOGIN:
			fprintf(stderr, "Invalid credentials\n");
			exit(EXIT_FAILURE);
			break;

		default:
			fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
			abort();
	}
	printf("Bye!\n");
	mysql_close (conn);
	return 0;
```

```
}
```

Administrator.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"



static void show_addresses(char* ditta, MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	int status;
	MYSQL_BIND param[2];

	//mostra indirizzi di una ditta

	if(!setup_prepared_stmt(&prepared_stmt, "call mostra_indirizzi_ditta(?)", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show addresses statement\n", false);
	}


	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = ditta;
	param[0].buffer_length = strlen(ditta);


	if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for show addresses\n", true);
	}


	if (mysql_stmt_execute(prepared_stmt) != 0) {

		print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
```

```
                goto next;
            }

            dump_result_set(conn, prepared_stmt, "\nList of Addresses");

        next:
            status = mysql_stmt_next_result(prepared_stmt);
            if (status > 0)
                finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);

    } while (status == 0);


    if(mysql_stmt_close(prepared_stmt) != 0)
        printf("Error closing");

}

static void show_communication_methods(char* ditta, MYSQL *conn)
{
    MYSQL_STMT *prepared_stmt;
    int status;
    MYSQL_BIND param[2];


    if(!setup_prepared_stmt(&prepared_stmt, "call mostra_com_ditta(?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show com methods
        statement\n", false);
    }

    memset(param, 0, sizeof(param));

    param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[0].buffer = ditta;
    param[0].buffer_length = strlen(ditta);


    if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for com
        methods\n", true);
    }


    if (mysql_stmt_execute(prepared_stmt) != 0) {

        print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("Error closing");

        return;
```

```
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "\nList of Communication Methods");

	   next:
		status = mysql_stmt_next_result(prepared_stmt);
		if (status > 0)
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);

	} while (status == 0);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");


}



static void set_fav_rec(char* ditta, MYSQL* conn){


	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[6];

	char via[46];
	char citta[46];
	char civico[11];
	int cap;

	show_addresses(ditta, conn);

	//imposta recapito preferito nel caso non sia stato impostato durante l'inserimento vero e
	//proprio

	printf("Please insert the Main Logistic Address\n");
	fflush(stdout);

reinsert_addr_2:

	printf("\nCity: ");
	getInput(46, citta, false);
```

```
printf("\nStreet: ");
getInput(46, via, false);
printf("\nCivic: ");
getInput(11, civico, false);
printf("\nCAP:");
scanf("%d", &cap);


flush(stdin);

if(!setup_prepared_stmt(&prepared_stmt, "call imposta_recapito(?, ?, ?, ?, ?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize set favorite
        statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = citta;
param[0].buffer_length = strlen(citta);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = via;
param[1].buffer_length = strlen(via);

param[2].buffer_type = MYSQL_TYPE_STRING;
param[2].buffer = &civico;
param[2].buffer_length = strlen(civico);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &cap;
param[3].buffer_length = sizeof(cap);

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param[4].buffer = ditta;
param[4].buffer_length = strlen(ditta);


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for favorite
        rec\n", true);

}


if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "City/Street/Civic")){
                memset(citta, 0, 46);
                memset(via, 0, 46);
```

```
                memset(civico, 0, 11);

                goto reinsert_addr_2;
            }

            if(specific_error_handler(prepared_stmt, "45014")){
                printf("Address not found, please reinsert\n");


                goto reinsert_addr_2;

            }



            finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition", true);

        }else{

            printf("Address correctly set for supplier %s!\n", ditta);
            fflush(stdout);
        }


        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("error closing");


}



static void set_fav_fat(char* ditta, MYSQL* conn){


        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[6];

        char via[46];
        char citta[46];
        char civico[11];
        int cap;

        show_addresses(ditta, conn);

        //imposta indirizzo fatturazione preferito nel caso non sia stato impostato nell'inserimento

        printf("Please insert the Main Billing Address\n");
        fflush(stdout);
```

```
reinsert_addr_3:

        printf("\nCity: ");
        getInput(46, citta, false);
        printf("\nStreet: ");
        getInput(46, via, false);
        printf("\nCivic: ");
        getInput(11, civico, false);
        printf("\nCAP:");
        scanf("%d", &cap);


        flush(stdin);

        if(!setup_prepared_stmt(&prepared_stmt, "call imposta_fatturazione(?, ?, ?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize set favorite fatt
                statement\n", false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = citta;
        param[0].buffer_length = strlen(citta);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = via;
        param[1].buffer_length = strlen(via);

        param[2].buffer_type = MYSQL_TYPE_STRING;
        param[2].buffer = &civico;
        param[2].buffer_length = strlen(civico);

        param[3].buffer_type = MYSQL_TYPE_LONG;
        param[3].buffer = &cap;
        param[3].buffer_length = sizeof(cap);

        param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[4].buffer = ditta;
        param[4].buffer_length = strlen(ditta);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for favorite
                fatt\n", true);

        }
```

```
if (mysql_stmt_execute(prepared_stmt) != 0) {

		if(generic_error_handler(prepared_stmt, "City/Street/Civic")){
			memset(citta, 0, 46);
			memset(via, 0, 46);
			memset(civico, 0, 11);

			goto reinsert_addr_3;
		}

		if(specific_error_handler(prepared_stmt, "45014")){
			printf("Address not found, please reinsert\n");


			goto reinsert_addr_3;

		}



		finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while setting the
		address", true);

	}else{

		printf("Address correctly set for supplier %s!\n", ditta);
		fflush(stdout);
	}


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("error closing");



}

static void set_fav_com(char* ditta, MYSQL* conn){

	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[4];

	char contatto[46];

	//imposto contatto preferito nel caso non sia stato impostato in  inserimento


	show_communication_methods(ditta, conn);
```

```
        printf("Please insert the Main Communication Method\n");
        fflush(stdout);

reinsert_com_2:

        printf("\nContact: ");
        getInput(46, contatto, false);

        if(!setup_prepared_stmt(&prepared_stmt, "call imposta_preferito(?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize set fav com
                statement\n", false);
        }


        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = contatto;
        param[0].buffer_length = strlen(contatto);

        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = ditta;
        param[1].buffer_length = strlen(ditta);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for set fav
                com\n", true);

        }


        if (mysql_stmt_execute(prepared_stmt) != 0) {

                if(generic_error_handler(prepared_stmt, "Contact")){
                        memset(contatto, 0, 46);

                        goto reinsert_com_2;
                }

                if(specific_error_handler(prepared_stmt, "45015")){
                        printf("Communication Method not found, please reinsert\n");
                        memset(contatto, 0, 46);

                        goto reinsert_com_2;

                }
```

```
            finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while setting the
            communication method", true);

        }else{

            printf("Communication Method correctly set for supplier %s!\n", ditta);
            fflush(stdout);
        }


        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("error closing");

}


static void insert_address(char *ditta, MYSQL *conn)
{
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[7];

        /*inserimento indirizzo di una ditta
          posso settare solo una volta il recapito e/o l'indirizzo di fatturazione preferito.
         se non lo imposto almeno una volta prima di terminare l'inserimento, viene chiesto
         esplicitamente in modo
          da rispettare la regola aziendale.
            */

        char via[46];
        char citta[46];
        char civico[11];
        int cap;
        bool recapito, recapito_n, fatturazione, fatturazione_n;
        bool first = true, logistic = false, billing = false;




        printf("Please insert informations about new Addresses\n");


        while(1){

            if(first){
                first = false;

                printf("Please insert the supplier's address\n");
            }else{
                printf("Do you want to add another address for this supplier?\n");
                if(!choice())
```

```
                    break;
            }


reinsert_addr:

            printf("\nCity: ");
            getInput(46, citta, false);
            printf("\nStreet: ");
            getInput(46, via, false);
            printf("\nCivic: ");
            getInput(11, civico, false);
            printf("\nCAP:");
            scanf("%d", &cap);


            flush(stdin);




            if(!logistic){
                printf("\nIs this the main Logistic Address[Y,N]?\n");
                if(choice()){
                    recapito = true;
                    logistic = true;
                }
                else
                    recapito = false;

            }else
                recapito = false;

            recapito_n = !recapito;



            if(!billing){
                printf("Is this the main Billing Address[Y/N]?\n");
                if(choice()){
                    fatturazione = true;
                    billing = true;
                }
                else
                    fatturazione = false;

            }else
                fatturazione = false;

            fatturazione_n = !fatturazione;
```

```
if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuovo_indirizzo(?, ?, ?, ?, ?, ?, ?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert address statement\n", false);
    }

    memset(param, 0, sizeof(param));

    param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[0].buffer = citta;
    param[0].buffer_length = strlen(citta);

    param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[1].buffer = via;
    param[1].buffer_length = strlen(via);

    param[2].buffer_type = MYSQL_TYPE_STRING;
    param[2].buffer = &civico;
    param[2].buffer_length = strlen(civico);

    param[3].buffer_type = MYSQL_TYPE_LONG;
    param[3].buffer = &cap;
    param[3].buffer_length = sizeof(cap);

    param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
    param[4].buffer = ditta;
    param[4].buffer_length = strlen(ditta);


    param[5].buffer_type = MYSQL_TYPE_TINY;
    param[5].buffer = &fatturazione;
    param[5].buffer_length = sizeof(fatturazione);
    param[5].is_null = &fatturazione_n;

    param[6].buffer_type = MYSQL_TYPE_TINY;
    param[6].buffer = &recapito;
    param[6].buffer_length = sizeof(recapito);
    param[6].is_null = &recapito_n;


    if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for insert address\n", true);
    }
```

```
		if (mysql_stmt_execute(prepared_stmt) != 0) {

			if(generic_error_handler(prepared_stmt, "City/Street/Civic")){
				memset(citta, 0, 46);
				memset(via, 0, 46);
				memset(civico, 0, 11);

				goto reinsert_addr;
			}

			finish_with_stmt_error(conn, prepared_stmt, "Error while insert address\n",
			true);

		}else{

			printf("Address correctly added for supplier %s!\n", ditta);
			fflush(stdout);
		}


		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");

	}

	if(!logistic)
		set_fav_rec(ditta, conn);

	if(!billing)
		set_fav_fat(ditta, conn);

}



static void insert_com(char *ditta, MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[5];

	char contatto[46];
	char tipologia[46];
	bool preferito, preferito_n;

	bool first = true, favorite = false;

	/*inserimento metodo comunicativo di una ditta
	  posso settare solo una volta il contatto preferito.
	 se non lo imposto almeno una volta prima di terminare l'inserimento, viene chiesto
	esplicitamente in modo
```

```
        da rispettare la regola aziendale.
        */



    while(1){

        if(first){
            first = false;
            printf("Please insert the supplier's communication method:\n");
        }else{
            printf("Do you want to add another communication method for this
            supplier?\n");
            fflush(stdout);
            if(!choice())
                break;
        }


reinsert_com:

        printf("\nContact: ");
        getInput(46, contatto, false);
        printf("\nType: ");
        getInput(46, tipologia, false);




        if(!favorite){
            printf("\nIs this the main Communication Method[Y,N]?\n");
            if(choice()){
                preferito = true;
                favorite = true;
            }
            else
                preferito = false;


        }else
            preferito = false;


        preferito_n = !preferito;


        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuovo_com(?, ?, ?, ?)",
        conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize com
            insertion statement\n", false);
```

```
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = contatto;
param[0].buffer_length = strlen(contatto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = tipologia;
param[1].buffer_length = strlen(tipologia);

param[2].buffer_type = MYSQL_TYPE_STRING;
param[2].buffer = ditta;
param[2].buffer_length = strlen(ditta);

param[3].buffer_type = MYSQL_TYPE_TINY;
param[3].buffer = &preferito;
param[3].buffer_length = sizeof(preferito);
param[3].is_null = &preferito_n;


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        com insertion\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "Contact/Type")){
                memset(contatto, 0, 46);
                memset(tipologia, 0, 46);

                goto reinsert_com;
        }

        finish_with_stmt_error(conn, prepared_stmt, "Error while insert com\n", true);

}else{

        printf("Communication method correctly added for supplier %s!\n", ditta);
        fflush(stdout);

}
```

```
                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("error closing");

        }


        if(!favorite)
                set_fav_com(ditta, conn);

}



static void insert_supplier(MYSQL *conn)
{
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];

        //inserimento di una nuova ditta

        char ditta[46];

reinsert_s:

        printf("Supplier name: ");
        getInput(46, ditta, false);

        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuova_ditta(?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize supplier insertion
                statement\n", false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = ditta;
        param[0].buffer_length = strlen(ditta);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for supplier
                insertion\n", true);
        }


        if (mysql_stmt_execute(prepared_stmt) != 0) {

                if(generic_error_handler(prepared_stmt, "Supplier")){
                        memset(ditta, 0, 46);
```

```
                goto reinsert_s;
            }else
                finish_with_stmt_error(conn, prepared_stmt, "Error while inserting
                supplier\n", true);

        }else{

            printf("Supplier correctly created!\n");
            fflush(stdout);

        }


        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("error closing");

        insert_address(ditta, conn);

        insert_com(ditta, conn);


}

static void create_order(MYSQL *conn, bool value)
{
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[4];

        char prodotto[46];
        char ditta[46];
        int how;
        int prog_number;

        if(value)
            show_all_products(conn);

        //creazione di un ordine

reinsert:

        printf("\nProduct name: ");
        getInput(46, prodotto, false);
        printf("Supplier name: ");
        getInput(46, ditta, false);

reinsert_q:

        printf("Desired quantity: ");
        scanf("%d", &how);
```

```
if(!setup_prepared_stmt(&prepared_stmt, "call crea_ordine(?, ?, ?, ?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize order creation
        statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &how;
param[2].buffer_length = sizeof(how);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &prog_number;
param[3].buffer_length = sizeof(prog_number);

if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for order
        creation\n", true);
}

if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "Supplier/Product")){
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);
                flush(stdin);


                goto reinsert;
        }
        if(specific_error_handler(prepared_stmt, "45000")){
                printf("The selected quantity is wrong, please reinsert\n");

                goto reinsert_q;

        }


        print_stmt_error (prepared_stmt, "An error occurred while creating the order");
        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");
```

```
		return;

	}


	memset(param, 0, sizeof(param));
	param[0].buffer_type = MYSQL_TYPE_LONG;
	param[0].buffer = &prog_number;
	param[0].buffer_length = sizeof(prog_number);

	if(mysql_stmt_bind_result(prepared_stmt, param)) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter",
		true);
	}

	if(mysql_stmt_fetch(prepared_stmt)) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not buffer results", true);
	}


	printf("Order correctly created with progressive number: %d\n", prog_number);
	fflush(stdout);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("error closing");

	flush(stdin);
}



static void register_prof_order(char* prodotto,char* ditta,int how,int prog_number, MYSQL* conn){

	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[5];
	MYSQL_TIME expiration_date;
	int i, day, month, year;

	//registrazione di un ordine riguardante un prodotto di profumeria

	printf("Please insert the necessary informations: \n");
	for(i = 0;i < how; i++){

		printf("Box number %d", i+1);
		fflush(stdout);

reinsert_prof_exp:

		printf("Expiration Date:\nDay:");
```

```
scanf("%2d", &day);
printf("Month:");
scanf("%2d", &month);
printf("Year:");
scanf("%4d", &year);
expiration_date.day = day;
expiration_date.month = month;
expiration_date.year = year;

flush(stdin);



if(!setup_prepared_stmt(&prepared_stmt, "call registra_scatola_profumeria(?,?,?,?)",   conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize order
        register statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_DATE;
param[2].buffer = &expiration_date;
param[2].buffer_length = sizeof(expiration_date);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &prog_number;
param[3].buffer_length = sizeof(prog_number);




if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        order register\n", true);
}
```

```
            if (mysql_stmt_execute(prepared_stmt) != 0) {


                    if(generic_error_handler(prepared_stmt, "Date"))

                            goto reinsert_prof_exp;



                    if(specific_error_handler(prepared_stmt, "45001")){
                            printf("Expiration Date is wrong, please reinsert:\n");

                            goto reinsert_prof_exp;

                    }

                    finish_with_stmt_error(conn, prepared_stmt, "Error registering the order\n",
                    true);



            }else{

                    printf("Box successfully added!\n");
                    fflush(stdout);

            }


            if(mysql_stmt_close(prepared_stmt) != 0)
                    printf("Error closing");

    }




}

static void show_free_positions(char* categoria, MYSQL *conn)
{
    MYSQL_STMT *prepared_stmt;
    MYSQL_BIND param[2];
    int status;
    //lista posizioni disponibili per scatole riferenti un farmaco di una determinata categoria

    if(!setup_prepared_stmt(&prepared_stmt, "call mostra_posizioni_disponibili(?)", conn)) {
            finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show free positions
            statement\n", false);
    }
```

```
	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = categoria;
	param[0].buffer_length = strlen(categoria);




	if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for show
		positions\n", true);
	}


	if (mysql_stmt_execute(prepared_stmt) != 0) {
		print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	printf("Category %s, ", categoria);
	fflush(stdout);

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "List of Available Positions");

	   next:
		status = mysql_stmt_next_result(prepared_stmt);
		if (status > 0)
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition show
		positions", true);

	} while (status == 0);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");

}
```

```
static void register_med_order(char* prodotto,char* ditta,int how,int prog_number, char* categoria,
MYSQL* conn){

        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[8];
        MYSQL_TIME expiration_date;
        int i, day, month, year, scaffale, cassetto;

        //registrazione ordine di medicinali


        printf("Please insert the necessary informations: \n");
        for(i = 0;i < how; i++){

                printf("Box number %d\n", i+1);
                fflush(stdout);

reinsert_exp:

                printf("Expiration Date:\nDay:");
                scanf("%2d", &day);
                printf("Month:");
                scanf("%2d", &month);
                printf("Year:");
                scanf("%4d", &year);
                expiration_date.day = day;
                expiration_date.month = month;
                expiration_date.year = year;


                flush(stdin);



                show_free_positions(categoria, conn);

reinsert_pos:

                printf("Position:\n");
                printf("\nScaffale:");
                scanf("%d", &scaffale);
                printf("\nCassetto:");
                scanf("%d", &cassetto);

                flush(stdin);


                if(!setup_prepared_stmt(&prepared_stmt,"call
                registra_scatola_medicinale(?,?,?,?,?,?,?)", conn)) {
```

```
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize order list statement\n", false);
}


memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);


param[2].buffer_type = MYSQL_TYPE_DATE;
param[2].buffer = &expiration_date;
param[2].buffer_length = sizeof(expiration_date);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &prog_number;
param[3].buffer_length = sizeof(prog_number);

param[4].buffer_type = MYSQL_TYPE_LONG;
param[4].buffer = &cassetto;
param[4].buffer_length = sizeof(cassetto);

param[5].buffer_type = MYSQL_TYPE_LONG;
param[5].buffer = &scaffale;
param[5].buffer_length = sizeof(scaffale);

param[6].buffer_type = MYSQL_TYPE_VAR_STRING;
param[6].buffer = categoria;
param[6].buffer_length = strlen(categoria);




if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for register\n", true);
}

if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(generic_error_handler(prepared_stmt, "Box/Category/Position"))
                goto reinsert_exp;
```

```
                        if(specific_error_handler(prepared_stmt, "45001")){
                                printf("Expiration Date is wrong, please reinsert:\n");

                                goto reinsert_exp;

                        }

                        if(specific_error_handler(prepared_stmt, "45002")){
                                printf("Position already occupated, please reinsert:\n");

                                goto reinsert_pos;

                        }


                        finish_with_stmt_error(conn, prepared_stmt, "Error while registering\n", true);


                }else{

                        printf("Box successfully added!\n");
                        fflush(stdout);

                }



                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("Error closing");

        }


}


static void register_order(MYSQL *conn) {
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[6];


        //registrazione di un ordine


        char prodotto[46];
        int is_med;
        char ditta[46];
        char categoria[46];
        int prog_number;
```

```
        int how, status, stat;

        if(!setup_prepared_stmt(&prepared_stmt, "call lista_ordini_pendenti()", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize pending list
                statement\n", false);
        }


        if (mysql_stmt_execute(prepared_stmt) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve the pending list\n",
                true);
        }

        do {

                if(conn->server_status & SERVER_PS_OUT_PARAMS) {
                        goto next;
                }

                dump_result_set(conn, prepared_stmt, "\nList of Pending orders");

           next:
                status = mysql_stmt_next_result(prepared_stmt);
                if (status > 0)
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
                        show pending boxes", true);

        } while (status == 0);


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");


        printf("Please insert the progressive number of the registering order\n");

reinsert_prog:

        scanf("%d", &prog_number);



        if(!setup_prepared_stmt(&prepared_stmt, "call get_order_info(?,?,?,?,?,?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize order info
                statement\n", false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_LONG;
```

```
param[0].buffer = &prog_number;
param[0].buffer_length = sizeof(prog_number);



if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for order
        info\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(specific_error_handler(prepared_stmt, "45006")){
                printf("Order already registered, please reinsert:\n");

                goto reinsert_prog;

        }

        if(specific_error_handler(prepared_stmt, "45007")){
                printf("Wrong progressive number, please reinsert:\n");

                goto reinsert_prog;

        }

        finish_with_stmt_error(conn, prepared_stmt, "Error while getting product info\n",
        true);
}


memset(param, 0, sizeof(param));
param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = 46;

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = 46;

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &how;
param[2].buffer_length = sizeof(how);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &is_med;
param[3].buffer_length = sizeof(is_med);
```

```
	param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[4].buffer = categoria;
	param[4].buffer_length = 46;

	if(mysql_stmt_bind_result(prepared_stmt, param)) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter",
		true);
	}

	if((stat = mysql_stmt_fetch(prepared_stmt))) {
		printf("stat %d", stat);
		finish_with_stmt_error(conn, prepared_stmt, "Could not buffer results", true);
	}


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");


	if(is_med)
		register_med_order(prodotto, ditta, how, prog_number, categoria, conn);
	else
		register_prof_order(prodotto, ditta, how, prog_number, conn);

}


static void insert_previsions(char* use, MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[3];


	//inserimento di previsioni tra prodotti e utilizzi

	char prodotto[46];
	char ditta[46];



	show_all_products(conn);


	while(true){


reinsert_prevision:

		printf("Please insert the name of the related product:");
		getInput(46, prodotto, false);
```

```
printf("Please insert the name of the product's supplier:");
getInput(46, ditta, false);

if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_previsione(?, ?, ?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize prevision
        insertion statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_VAR_STRING;
param[2].buffer = use;
param[2].buffer_length = strlen(use);


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        prevision insertion\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(generic_error_handler(prepared_stmt, "Prevision")){
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);


                goto reinsert_prevision;
        }else{
                print_stmt_error (prepared_stmt, "An error occurred while adding
                previsions.");
                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("Error closing");

                return;
        }

}else{

        printf("Prevision correctly added!");
```

```
                fflush(stdout);
        }



        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("error closing");


        printf("\n Other interactions to add for the use %s?[Y/N]\n", use);
        fflush(stdout);



        if(choice()){
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);
                continue;
        }else{
                return;
        }



}




}

static void insert_use(MYSQL *conn)
{
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[1];


        //inserimento di un utilizzo

        char utilizzo[46];

reinsert_use:

        printf("\nUse name: ");
        getInput(46, utilizzo, false);

        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuovo_utilizzo(?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize use insertion
                statement\n", false);
        }
```

```
	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = utilizzo;
	param[0].buffer_length = strlen(utilizzo);

	if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for use
		insertion\n", true);
	}

	if (mysql_stmt_execute(prepared_stmt) != 0) {

		if(generic_error_handler(prepared_stmt, "Use")){
			memset(utilizzo, 0, 46);

			goto reinsert_use;
		}else{
			print_stmt_error (prepared_stmt, "An error occurred while adding the use.");
			if(mysql_stmt_close(prepared_stmt) != 0)
				printf("Error closing");

			return;
		}

	}else{

		printf("Use correctly added!");
		fflush(stdout);
	}

	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("error closing");

	printf("\nIs this use intended with some products' categories? [y/n]?\n");
	fflush(stdout);

	if(choice())
		insert_previsions(utilizzo,conn);
}

static void report_dismissed_boxes(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	int status;

	//report delle scatole di medicinali in scadenza e quindi dismesse
```

```
	if(!setup_prepared_stmt(&prepared_stmt, "call report_scatole_scadenza()", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize report dis box statement\n", false);
	}


	if (mysql_stmt_execute(prepared_stmt) != 0) {
		print_stmt_error (prepared_stmt, "An error occurred while report dis box.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "\nList of Dismissed Boxes");

	   next:
		status = mysql_stmt_next_result(prepared_stmt);
		if (status > 0)
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while reporting", true);

	} while (status == 0);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");



}

static void report_med_stocks(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	int status;
	bool first = true;

	//lista giacenze di magazzino compresi i prodotti esauriti

	if(!setup_prepared_stmt(&prepared_stmt, "call report_giacenze_med()", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize report med stocks statement\n", false);
	}
```

```
        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while report med stocks.");
                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("Error closing");

                return;
        }

        do {

                if(conn->server_status & SERVER_PS_OUT_PARAMS) {
                        goto next;
                }

                dump_result_set(conn, prepared_stmt, "\nList of Depleting Stocks");

           next:
                status = mysql_stmt_next_result(prepared_stmt);
                if (status > 0)
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition reporting
                        med stocks", true);

        } while (status == 0);


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");


        while(1){

                if(first){

                        printf("Do you want to order boxes of one of this products?\n");
                        first = false;

                }else
                        printf("Do you want to order boxes of other products?");

                if(choice())
                        create_order(conn, false);
                else
                        break;

        }

}
```

```
void run_as_administrator(MYSQL *conn)
{
	char options[8] = {'1','2', '3', '4', '5', '6', '7', '8'};
	char op;

	printf("Switching to administrative role...\n");

	if(!parse_config("users/amministratore.json", &conf)) {
		fprintf(stderr, "Unable to load administrator configuration\n");
		exit(EXIT_FAILURE);
	}

	mysql_close(conn);


	conn = mysql_init (NULL);
	if (conn == NULL) {
		fprintf (stderr, "mysql_init() failed (probably out of memory)\n");
		exit(EXIT_FAILURE);
	}

	if (mysql_real_connect(conn, conf.host, conf.db_username, conf.db_password, conf.database,
conf.port, NULL, CLIENT_MULTI_STATEMENTS | CLIENT_MULTI_RESULTS) == NULL) {
		fprintf (stderr, "mysql_real_connect() failed\n");
		fprintf(stderr, "Failed to change user.  Error: %s\n",mysql_error(conn));
		mysql_close (conn);
		exit(EXIT_FAILURE);
	}


	while(true) {
		printf("\033[2J\033[H");
		printf("*** What should I do for you? ***\n\n");
		printf("1) Create Order\n");
		printf("2) Register Order\n");
		printf("3) Insert a new Supplier\n");
		printf("4) Insert a new product use\n");
		printf("5) Report of dismissed boxes\n");
		printf("6) Report of Sales about Medicinals requiring recipe\n");
		printf("7) Report of Medicinal Depleting Stocks\n");
		printf("8) Quit\n");

		op = multiChoice("Select an option", options, 8);

		switch(op) {
			case '1':
```

```
                        create_order(conn, true);
                        break;

                case '2':
                        register_order(conn);
                        break;


                case '3':
                        insert_supplier(conn);
                        break;

                case '4':
                        insert_use(conn);
                        break;

                case '5':
                        report_dismissed_boxes(conn);
                        break;

                case '6':
                        report_sales_rr(conn);
                        break;

                case '7':
                        report_med_stocks(conn);
                        break;

                case '8':
                        return;

                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
        }

        printf("Do you want to continue operations?\n");
        if(choice())
                continue;
        else
                return;
    }
}
```

Doctor.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include "defines.h"



static void show_pending_boxes(char *prodotto, char *ditta, MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[3];
	int status;

	//lista per le scatole pendenti di una determinato prodotto

	if(!setup_prepared_stmt(&prepared_stmt, "call mostra_scatole_pendenti(?, ?)", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show pending
		boxes statement\n", false);
	}


	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = prodotto;
	param[0].buffer_length = strlen(prodotto);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = ditta;
	param[1].buffer_length = strlen(ditta);


	if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for show
		pending boxes\n", true);
	}


	if (mysql_stmt_execute(prepared_stmt) != 0) {
		print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "\nList of Pending Boxes");
```

```
    next:
        status = mysql_stmt_next_result(prepared_stmt);
        if (status > 0)
            finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
            show pending boxes", true);

    } while (status == 0);


    if(mysql_stmt_close(prepared_stmt) != 0)
        printf("Error closing");



}

static void show_all_categories(MYSQL *conn)
{
    MYSQL_STMT *prepared_stmt;
    int status;

    //lista di tutte le categorie


    if(!setup_prepared_stmt(&prepared_stmt, "call mostra_categorie()", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show categories
        statement\n", false);
    }


    if (mysql_stmt_execute(prepared_stmt) != 0) {
        print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("Error closing");

        return;
    }


    do {


        if(conn->server_status & SERVER_PS_OUT_PARAMS) {
            goto next;
        }

        dump_result_set(conn, prepared_stmt, "\nList of Categories");
```

```
    next:
        status = mysql_stmt_next_result(prepared_stmt);
        if (status > 0)
            finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
            show categories", true);

    } while (status == 0);


    if(mysql_stmt_close(prepared_stmt) != 0)
        printf("Error closing");

}



static void show_all_suppliers(MYSQL *conn)
{
    MYSQL_STMT *prepared_stmt;
    int status;

    //lista delle ditte

    if(!setup_prepared_stmt(&prepared_stmt, "call mostra_ditte()", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show suppliers
        statement\n", false);
    }


    if (mysql_stmt_execute(prepared_stmt) != 0) {
        print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
        if(mysql_stmt_close(prepared_stmt) != 0)
            printf("Error closing");

        return;
    }

    do {

        if(conn->server_status & SERVER_PS_OUT_PARAMS) {
            goto next;
        }

        dump_result_set(conn, prepared_stmt, "\nList of Suppliers");

    next:
        status = mysql_stmt_next_result(prepared_stmt);
        if (status > 0)
            finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
            show suppliers", true);
```

```
        } while (status == 0);


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");



}


static void register_prof_sale(char *prodotto, char *ditta, int sale_id, MYSQL *conn)
{


        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[4];

        int serial;

        //gestione vendita scatole di prodotti di profumeria


        show_pending_boxes(prodotto, ditta, conn);

        while(1){


reinsert_sale_box:

                printf("\nPlease insert the Box Serial of One Box:");
                scanf("%d", &serial);


        if(!setup_prepared_stmt(&prepared_stmt, "call vendita_scatola_prof(?, ?, ?, ?)", conn)) {
                        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize sale
                        statement\n", false);
                }



                memset(param, 0, sizeof(param));

                param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
                param[0].buffer = prodotto;
                param[0].buffer_length = strlen(prodotto);

                param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
```

```
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &serial;
param[2].buffer_length = sizeof(serial);

param[3].buffer_type = MYSQL_TYPE_LONG;
param[3].buffer = &sale_id;
param[3].buffer_length = sizeof(sale_id);



if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        the sale\n", true);
}


// Run procedure
if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(generic_error_handler(prepared_stmt, "Sale box"))


                goto reinsert_sale_box;


        if(specific_error_handler(prepared_stmt, "45011")){
                printf("The selected box is not pending/existing or is related to another
                product, please reinsert:\n");

                goto reinsert_sale_box;

        }

        finish_with_stmt_error(conn, prepared_stmt, "Could not handle the sale
        error\n", true);
}



if(mysql_stmt_close(prepared_stmt) != 0)
        printf("Error closing");

flush(stdin);
printf("Box with serial %d correctly registered wit the sale %d", serial, sale_id);
printf("\nOther boxes of this product to register for this sale [y/n]?\n");
```

```
                if(choice())
                        continue;
                else
                        break;




        }

}

static int insert_venditarr(char *prodotto, char *ditta, int sale_id, MYSQL* conn)
{

        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[6];

        MYSQL_TIME prescription_date;
        int sale_id_rr, day, month, year, stat;

        char doctor[46];

        //inserimento di una vendita di medicinali richiedenti ricetta


        printf("Please insert the necessary informations!\n");

reinsert_salerr:

        printf("Prescription Date:\nDay:");
        scanf("%2d", &day);
        printf("Month:");
        scanf("%2d", &month);
        printf("Year:");
        scanf("%4d", &year);
        prescription_date.day = day;
        prescription_date.month = month;
        prescription_date.year = year;

        flush(stdin);

        printf("Please insert the name of the doctor:");
        getInput(46, doctor, false);


        if(!setup_prepared_stmt(&prepared_stmt, "call registra_vendita_rr(?, ?, ?, ?, ?, ?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert sale rr
                statement\n", false);
        }
```

```
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &sale_id;
param[2].buffer_length = sizeof(sale_id);

param[3].buffer_type = MYSQL_TYPE_DATE;
param[3].buffer = &prescription_date;
param[3].buffer_length = sizeof(prescription_date);

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param[4].buffer = doctor;
param[4].buffer_length = strlen(doctor);


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for sale rr
        insertion\n", true);
}



if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "Prescription/Doctor")){
                memset(doctor, 0, 46);

                goto reinsert_salerr;
        }


        if(specific_error_handler(prepared_stmt, "45012")){
                printf("Prescription Date is not valid(30 days max validity), please
                reinsert:\n");

                goto reinsert_salerr;

        }

        finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve the informations\n",
        true);
```

```
	}


	memset(param, 0, sizeof(param));
	param[0].buffer_type = MYSQL_TYPE_LONG;
	param[0].buffer = &sale_id_rr;
	param[0].buffer_length = sizeof(sale_id_rr);


	if(mysql_stmt_bind_result(prepared_stmt, param)) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter",
		true);
	}


	if((stat = mysql_stmt_fetch(prepared_stmt))) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not buffer results", true);
	}


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");


	return sale_id_rr;


}

static void show_related_salesrr(char *prodotto,char *ditta,int sale_id,MYSQL* conn)
{

	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[3];

	int status;



	if(!setup_prepared_stmt(&prepared_stmt, "call mostra_venditerr_relative(?, ?, ?)", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show related sales
		statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = prodotto;
	param[0].buffer_length = strlen(prodotto);
```

```
        param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[1].buffer = ditta;
        param[1].buffer_length = strlen(ditta);

        param[2].buffer_type = MYSQL_TYPE_LONG;
        param[2].buffer = &sale_id;
        param[2].buffer_length = sizeof(sale_id);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for show rel
                sales\n", true);
        }

        if (mysql_stmt_execute(prepared_stmt) != 0) {
                print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("Error closing");

                return;
        }


        do {

                if(conn->server_status & SERVER_PS_OUT_PARAMS) {
                        goto next;
                }

                dump_result_set(conn, prepared_stmt, "\nList of Related Operations");


           next:
                status = mysql_stmt_next_result(prepared_stmt);
                if (status > 0)
                        finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition for show
                        rel sales", true);

        } while (status == 0);


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");

}
```

```
static void register_med_sale(char *prodotto, char *ditta, int sale_id, int is_rr, MYSQL* conn)
{

	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[6];

	int serial;
	char cf[21];
	bool flag_cf_n, flag_rr_n;

	int idr;

	//register medicinal sale

	show_pending_boxes(prodotto, ditta, conn);


	while(1){

reinsert_sale_med_box:

		printf("\nPlease insert the Box Serial of One Box:");
		scanf("%d", &serial);

		flush(stdin);

		printf("Has the client decided to register his CF for this box? [y/n]\n");
		if(choice()){
			printf("Please insert it: ");
			getInput(21, cf, false);

			flag_cf_n = false;
		}else
			flag_cf_n = true;


		if(is_rr){

reinsert_related:

			flag_rr_n = false;
			show_related_salesrr(prodotto, ditta, sale_id, conn);
			printf("Is this box related to one of this operations?[y/n]\n");
			if(choice()){
				printf("Please insert the idr of the operation: ");
				scanf("%d", &idr);
				flush(stdin);
			}else{
				idr = insert_venditarr(prodotto, ditta, sale_id, conn);
			}
```

```
}else{
        flag_rr_n = true;
}


if(!setup_prepared_stmt(&prepared_stmt, "call vendita_scatola_med(?, ?, ?, ?, ?, ?)",
conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize sale
        statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_LONG;
param[2].buffer = &serial;
param[2].buffer_length = sizeof(serial);

param[3].buffer_type = MYSQL_TYPE_VAR_STRING;
param[3].buffer = cf;
param[3].buffer_length = strlen(cf);
param[3].is_null = &flag_cf_n;

param[4].buffer_type = MYSQL_TYPE_LONG;
param[4].buffer = &sale_id;
param[4].buffer_length = sizeof(sale_id);

param[5].buffer_type = MYSQL_TYPE_LONG;
param[5].buffer = &idr;
param[5].buffer_length = sizeof(idr);
param[5].is_null = &flag_rr_n;


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        sale insertion\n", true);
}
```

```
if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "Box/CF")){
                memset(cf, 0, 21);

                goto reinsert_sale_med_box;
        }


        if(specific_error_handler(prepared_stmt, "45011")){
                printf("The selected box is not pending/existing or is related to another
                product, please reinsert:\n");
                memset(cf, 0, 21);

                goto reinsert_sale_med_box;

        }



        if(specific_error_handler(prepared_stmt, "45012")){
                printf("The selected idr is not related to this sale or to this product,
                please reinsert:\n");
                memset(cf, 0, 21);

                goto reinsert_related;

        }

        finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve
        informations\n", true);

}



if(mysql_stmt_close(prepared_stmt) != 0)
        printf("Error closing");


if(is_rr)
        printf("Box with serial %d correctly registered wit the sale RR %d related to
        the general sale %d", serial, idr, sale_id);
else
        printf("Box with serial %d correctly registered wit the sale %d", serial,
        sale_id);
```

```
                printf("\nOther boxes of this product to register for this sale [y/n]?\n");
                if(choice()){
                        memset(cf, 0, 21);
                        continue;
                }
                else
                        break;


        }

}



static void register_sale(MYSQL *conn) {
        MYSQL_STMT *prepared_stmt;
        MYSQL_BIND param[5];

        int is_rr, is_prof, sale_id, stat;

        char prodotto[46];
        char ditta[46];


        if(!setup_prepared_stmt(&prepared_stmt, "call registra_vendita(?)", conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize sale statement\n",
                false);
        }

        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_LONG;
        param[0].buffer = &sale_id;
        param[0].buffer_length = sizeof(sale_id);


        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for sale
                insertion\n", true);
        }

        if (mysql_stmt_execute(prepared_stmt) != 0) {

                print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
                if(mysql_stmt_close(prepared_stmt) != 0)
                        printf("Error closing");

                return;
        }
```

```
        memset(param, 0, sizeof(param));
        param[0].buffer_type = MYSQL_TYPE_LONG;
        param[0].buffer = &sale_id;
        param[0].buffer_length = sizeof(sale_id);


        if(mysql_stmt_bind_result(prepared_stmt, param)) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter",
                true);
        }


        if((stat = mysql_stmt_fetch(prepared_stmt))) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not buffer results", true);
        }


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error closing");



        show_all_products(conn);


        while(1){


                printf("\nOf which Product do you want to register the sale?\n");

reinsert_prod_sale:

                printf("Product name:");
                getInput(46, prodotto, false);
                printf("\nSupplier name:");
                getInput(46, ditta, false);

                if(!setup_prepared_stmt(&prepared_stmt, "call get_product_info(?, ?, ?, ?)", conn)) {
                        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize get info
                        statement\n", false);
                }


                memset(param, 0, sizeof(param));

                param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
                param[0].buffer = prodotto;
                param[0].buffer_length = strlen(prodotto);
```

```
param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);


if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        get info\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {

        if(generic_error_handler(prepared_stmt, "Product/Supplier")){
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);


                goto reinsert_prod_sale;
        }


        if(specific_error_handler(prepared_stmt, "45009")){
                printf("Product/Supplier not found or Without Stocks, please
                reinsert:\n");
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);

                goto reinsert_prod_sale;

        }

        finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve
        informations\n", true);

}


memset(param, 0, sizeof(param));
param[0].buffer_type = MYSQL_TYPE_LONG;
param[0].buffer = &is_prof;
param[0].buffer_length = sizeof(is_prof);


param[1].buffer_type = MYSQL_TYPE_LONG;
param[1].buffer = &is_rr;
param[1].buffer_length = sizeof(is_rr);
```

```
		if(mysql_stmt_bind_result(prepared_stmt, param)) {
			finish_with_stmt_error(conn, prepared_stmt, "Could not retrieve output parameter", true);
		}

		if((mysql_stmt_fetch(prepared_stmt))) {
			finish_with_stmt_error(conn, prepared_stmt, "Could not buffer results", true);
		}


		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		if(is_rr != 1)
			is_rr = 0;

		/*gestisco differentemente le vendite di scatole di profumerie da quelle di medicinali, oltre a salvarmi l'informazione sul fatto
		  che la scatola riferisca un farmaco che richiede ricetta oppure no*/

		if(is_prof)
			register_prof_sale(prodotto, ditta, sale_id, conn);
		else
			register_med_sale(prodotto, ditta, sale_id, is_rr, conn);


		printf("Other Boxes of other products to register?[y/n]\n");
		if(choice()){
			memset(prodotto, 0, 46);
			memset(ditta, 0, 46);
			continue;
		}
		else
			break;


	}


}

static void insert_medicinal(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[5];

	char prodotto[46];
	char ditta[46];
	char categoria[46];
	bool mut_flag, rr_flag;
```

```
        bool mut_flag_n, rr_flag_n;

        //inserimento nuovo medicinale


reinsert_med:

        printf("\nProduct name: ");
        getInput(46, prodotto, false);


        show_all_suppliers(conn);

        printf("Supplier name: ");
        getInput(46, ditta, false);


        show_all_categories(conn);


        printf("Category: ");
        getInput(46, categoria, false);



        printf("Is it loanable [y/n]?\n");

        if(choice()){
                mut_flag = 1;
                mut_flag_n = 0;
        }else{
                mut_flag = 0;
                mut_flag_n = 1;
        }

        printf("Requires it recipe? [y/n]?\n");

        if(choice()){
                rr_flag = 1;
                rr_flag_n= 0;
        }else{
                rr_flag = 0;
                rr_flag_n = 1;
        }

        if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuovo_medicinale(?, ?, ?, ?,?)",
conn)) {
                finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert statement\n",
                false);
        }
```

```
memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = prodotto;
param[0].buffer_length = strlen(prodotto);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = ditta;
param[1].buffer_length = strlen(ditta);

param[2].buffer_type = MYSQL_TYPE_TINY;
param[2].buffer = &mut_flag;
param[2].buffer_length = sizeof(mut_flag);
param[2].is_null = &mut_flag_n;

param[3].buffer_type = MYSQL_TYPE_TINY;
param[3].buffer = &rr_flag;
param[3].buffer_length = sizeof(rr_flag);
param[3].is_null = &rr_flag_n;

param[4].buffer_type = MYSQL_TYPE_VAR_STRING;
param[4].buffer = categoria;
param[4].buffer_length = strlen(categoria);

if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        medicinal insertion\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(generic_error_handler(prepared_stmt, "Supplier/Product/Category")){
                memset(prodotto, 0, 46);
                memset(ditta, 0, 46);
                memset(categoria, 0, 46);


                goto reinsert_med;
        }else
                finish_with_stmt_error(conn, prepared_stmt, "Error adding the medicinal\n",
                true);

}else{

        printf("Medicinal correctly added!");
        fflush(stdout);
}
```

```
	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");



}

static void insert_prof(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[3];

	char prodotto[46];
	char ditta[46];

	//inserimento nuovo prodotto di profumeria

reinsert_prof:

	printf("\nProduct name: ");
	getInput(46, prodotto, false);


	show_all_suppliers(conn);

	printf("Supplier name: ");
	getInput(46, ditta, false);


	if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuovo_profumeria(?, ?)", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert prof
		statement\n", false);
	}

	memset(param, 0, sizeof(param));

	param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[0].buffer = prodotto;
	param[0].buffer_length = strlen(prodotto);

	param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
	param[1].buffer = ditta;
	param[1].buffer_length = strlen(ditta);

	if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
		finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for prof
		product insertion\n", true);
	}
```

```
	if (mysql_stmt_execute(prepared_stmt) != 0) {


			if(generic_error_handler(prepared_stmt, "Supplier/Product")){
				memset(prodotto, 0, 46);
				memset(ditta, 0, 46);


				goto reinsert_prof;
			}else
				finish_with_stmt_error(conn, prepared_stmt, "Error adding the
				perfumery product\n", true);

		}else{

			printf("Perfumery Product correctly added!");
			fflush(stdout);
		}


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");


}



static void insert_category_interactions(char* category, MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[2];

	//inserimento interazioni tra categorie

	char categoria[46];



	show_all_categories(conn);

	while(true){


		printf("Please insert the name of the related category:");


reinsert_int:
```

```
getInput(46, categoria, false);

if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_interazione(?, ?)", conn)) {
        finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize interaction
        insertion statement\n", false);
}

memset(param, 0, sizeof(param));

param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
param[0].buffer = category;
param[0].buffer_length = strlen(category);

param[1].buffer_type = MYSQL_TYPE_VAR_STRING;
param[1].buffer = categoria;
param[1].buffer_length = strlen(categoria);

if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
        finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for
        interaction insertion\n", true);
}


if (mysql_stmt_execute(prepared_stmt) != 0) {


        if(generic_error_handler(prepared_stmt, "Interaction")){
                memset(categoria, 0, 46);
                //duplicate entry

                goto reinsert_int;
        }

        if(specific_error_handler(prepared_stmt, "45008")){
                printf("interaction already inserted, other interactions to
                add[Y/N]?:\n");

                if(choice()){
                        memset(categoria, 0, 46);
                        continue;
                }else{
                        return;
                }

        }


        finish_with_stmt_error(conn, prepared_stmt, "Could not retreive
        informations\n", true);
```

```
		}else{

			printf("Interaction correctly added!\n Other interactions to add for the category
		%s?[Y/N]", category);
			fflush(stdout);
		}



		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");




		if(choice()){
			memset(categoria, 0, 46);
			continue;
		}else{
			return;
		}

	}




}

static void insert_category(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	MYSQL_BIND param[2];

	//inserimento nuova categoria

	char categoria[46];


reinsert_cat:

	printf("\nCategory name: ");
	getInput(46, categoria, false);


	if(!setup_prepared_stmt(&prepared_stmt, "call inserisci_nuova_categoria(?)", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize insert category
		statement\n", false);
	}
```

```
        memset(param, 0, sizeof(param));

        param[0].buffer_type = MYSQL_TYPE_VAR_STRING;
        param[0].buffer = categoria;
        param[0].buffer_length = strlen(categoria);

        if (mysql_stmt_bind_param(prepared_stmt, param) != 0) {
                finish_with_stmt_error(conn, prepared_stmt, "Could not bind parameters for category
                insertion\n", true);
        }


        if (mysql_stmt_execute(prepared_stmt) != 0) {


                if(generic_error_handler(prepared_stmt, "Category")){
                        memset(categoria, 0, 46);

                        goto reinsert_cat;
                }

                finish_with_stmt_error(conn, prepared_stmt, "Could not insert informations\n", true);

        }else{

                printf("Category correctly added!");
                fflush(stdout);
        }


        if(mysql_stmt_close(prepared_stmt) != 0)
                printf("Error inserting");


        printf("Does this category have interactions with other categories? [y/n]?\n");

        if(choice())
                insert_category_interactions(categoria,conn);
        else
                return;



}
```

```
void run_as_doctor(MYSQL *conn)
{
	char options[6] = {'1','2', '3', '4', '5', '6'};
	char op;

	printf("Switching to professor role...\n");

	if(!parse_config("users/dottore.json", &conf)) {
		fprintf(stderr, "Unable to load professor configuration\n");
		exit(EXIT_FAILURE);
	}

	mysql_close (conn);


	conn = mysql_init (NULL);
	if (conn == NULL) {
		fprintf (stderr, "mysql_init() failed (probably out of memory)\n");
		exit(EXIT_FAILURE);
	}

	if (mysql_real_connect(conn, conf.host, conf.db_username, conf.db_password, conf.database,
conf.port, NULL, CLIENT_MULTI_STATEMENTS | CLIENT_MULTI_RESULTS) == NULL) {
		fprintf (stderr, "mysql_real_connect() failed\n");
		fprintf(stderr, "Failed to change user.  Error: %s\n",mysql_error(conn));
		mysql_close (conn);
		exit(EXIT_FAILURE);
	}

	while(true) {
		printf("\033[2J\033[H");
		printf("*** What should I do for you? ***\n\n");
		printf("1) Insert a new medicinal\n");
		printf("2) Insert a new perfumery product\n");
		printf("3) Insert a new Category\n");
		printf("4) Register a Sale\n");
		printf("5) Report of Sales about Medicinals requiring recipe\n");
		printf("6) Quit\n");

		op = multiChoice("Select an option", options, 6);

		switch(op) {
			case '1':
				insert_medicinal(conn);
				break;

			case '2':
				insert_prof(conn);
```

```
                        break;

                case '3':
                        insert_category(conn);
                        break;

                case '4':
                        register_sale(conn);
                        break;


                case '5':
                        report_sales_rr(conn);
                        break;

                case '6':
                        return;

                default:
                        fprintf(stderr, "Invalid condition at %s:%d\n", __FILE__, __LINE__);
                        abort();
        }

        printf("\nDo you want to continue operations?\n");
        if(choice())
                continue;
        else
                return;
    }
}
```

common_queries.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"



void show_all_products(MYSQL *conn)
{
    MYSQL_STMT *prepared_stmt;
    int status;

    //lista di tutti i prodotti


    if(!setup_prepared_stmt(&prepared_stmt, "call mostra_prodotti()", conn)) {
```

```
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize show products
		statement\n", false);
	}


	if (mysql_stmt_execute(prepared_stmt) != 0) {
		print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		dump_result_set(conn, prepared_stmt, "\nList of Products");

	   next:
		status = mysql_stmt_next_result(prepared_stmt);
		if (status > 0)
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
			showing products", true);

	} while (status == 0);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");


}


void report_sales_rr(MYSQL *conn)
{
	MYSQL_STMT *prepared_stmt;
	int status;

	bool serials = false;

	//report delle scatole di medicinali in scadenza e quindi dismesse

	if(!setup_prepared_stmt(&prepared_stmt, "call report_vendite_medicinali_rr()", conn)) {
		finish_with_stmt_error(conn, prepared_stmt, "Unable to initialize report statement\n",
		false);
	}
```

```
	if (mysql_stmt_execute(prepared_stmt) != 0) {
		print_stmt_error (prepared_stmt, "An error occurred while retrieving informations.");
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("Error closing");

		return;
	}

	do {

		if(conn->server_status & SERVER_PS_OUT_PARAMS) {
			goto next;
		}

		if(serials){
			dump_result_set(conn, prepared_stmt, "\nSerials of saled Boxes in this sale");
			serials = false;
		}else{
			dump_result_set(conn, prepared_stmt, "\nSale Info");
			serials = true;
		}


	   next:
		status = mysql_stmt_next_result(prepared_stmt);
		if (status > 0)
			finish_with_stmt_error(conn, prepared_stmt, "Unexpected condition while
			report sales", true);

	} while (status == 0);


	if(mysql_stmt_close(prepared_stmt) != 0)
		printf("Error closing");



}
```

errors_handler.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"
```

```
bool generic_error_handler(MYSQL_STMT * prepared_stmt, char *object){


	if(strcmp("22001", mysql_stmt_sqlstate(prepared_stmt)) == 0){
		printf("%s name is too long, please reinsert:\n", object);

		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");

		return true;
	}

	if(strcmp("23000", mysql_stmt_sqlstate(prepared_stmt)) == 0){

		if(mysql_stmt_errno (prepared_stmt) == (unsigned int)1062)
			printf("%s duplicate entry, please reinsert:\n", object);

		if(mysql_stmt_errno (prepared_stmt) == (unsigned int)1452)
			printf("%s error on foreign key, please reinsert:\n", object);

		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");

		return true;
	}

	if(strcmp("22007", mysql_stmt_sqlstate(prepared_stmt)) == 0){
		printf("%s date is in wrong format, please reinsert:\n", object);

		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");

		return true;
	}

	return false;
}

bool specific_error_handler(MYSQL_STMT * prepared_stmt, char * error){


	if(strcmp(error, mysql_stmt_sqlstate(prepared_stmt)) == 0){
		if(mysql_stmt_close(prepared_stmt) != 0)
			printf("error closing");

		return true;
	}

	return false;
}
```

defines.h

```
#pragma once

#include <stdbool.h>
#include <mysql.h>

#define flush(stdin) while(getchar() != '\n')

struct configuration {
	char *host;
	char *db_username;
	char *db_password;
	unsigned int port;
	char *database;

	char username[128];
	char password[128];
};

extern struct configuration conf;

extern int parse_config(char *path, struct configuration *conf);
extern char *getInput(unsigned int lung, char *stringa, bool hide);
extern bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive);
extern char multiChoice(char *domanda, char choices[], int num);
extern void print_error (MYSQL *conn, char *message);
extern void print_stmt_error (MYSQL_STMT *stmt, char *message);
extern void finish_with_error(MYSQL *conn, char *message);
extern void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message, bool close_stmt);
extern bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn);
extern void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title);
extern bool choice();
extern void run_as_doctor(MYSQL *conn);
extern void run_as_administrator(MYSQL *conn);
extern void show_all_products(MYSQL *conn);
extern void report_sales_rr(MYSQL *conn);
extern bool generic_error_handler(MYSQL_STMT* prepared_stmt, char* object);
extern bool specific_error_handler(MYSQL_STMT * prepared_stmt, char * error);
```

inout.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <pthread.h>
#include <signal.h>
#include <stdbool.h>

#include "defines.h"

// Per la gestione dei segnali
static volatile sig_atomic_t signo;
typedef struct sigaction sigaction_t;
static void handler(int s);

char *getInput(unsigned int lung, char *stringa, bool hide)
{
	char c;
	unsigned int i;

	// Dichiara le variabili necessarie ad un possibile mascheramento dell'input
	sigaction_t sa, savealrm, saveint, savehup, savequit, saveterm;
	sigaction_t savetstp, savettin, savettou;
	struct termios term, oterm;

	if(hide) {
		// Svuota il buffer
		(void) fflush(stdout);

		// Cattura i segnali che altrimenti potrebbero far terminare il programma, lasciando l'utente senza output sulla shell
		sigemptyset(&sa.sa_mask);
		sa.sa_flags = SA_INTERRUPT; // Per non resettare le system call
		sa.sa_handler = handler;
		(void) sigaction(SIGALRM, &sa, &savealrm);
		(void) sigaction(SIGINT, &sa, &saveint);
		(void) sigaction(SIGHUP, &sa, &savehup);
		(void) sigaction(SIGQUIT, &sa, &savequit);
		(void) sigaction(SIGTERM, &sa, &saveterm);
		(void) sigaction(SIGTSTP, &sa, &savetstp);
		(void) sigaction(SIGTTIN, &sa, &savettin);
		(void) sigaction(SIGTTOU, &sa, &savettou);

		// Disattiva l'output su schermo
		if (tcgetattr(fileno(stdin), &oterm) == 0) {
```

```
                (void) memcpy(&term, &oterm, sizeof(struct termios));
                term.c_lflag &= ~(ECHO|ECHONL);
                (void) tcsetattr(fileno(stdin), TCSAFLUSH, &term);
        } else {
                (void) memset(&term, 0, sizeof(struct termios));
                (void) memset(&oterm, 0, sizeof(struct termios));
        }
}

// Acquisisce da tastiera al più lung - 1 caratteri
for(i = 0; i < lung; i++) {
        (void) fread(&c, sizeof(char), 1, stdin);
        if(c == '\n') {
                stringa[i] = '\0';
                break;
        } else
                stringa[i] = c;

        // Gestisce gli asterischi
        if(hide) {
                if(c == '\b') // Backspace
                        (void) write(fileno(stdout), &c, sizeof(char));
                else
                        (void) write(fileno(stdout), "*", sizeof(char));
        }
}

// Controlla che il terminatore di stringa sia stato inserito
if(i == lung - 1)
        stringa[i] = '\0';

// Se sono stati digitati più caratteri, svuota il buffer della tastiera
if(strlen(stringa) >= lung) {
        // Svuota il buffer della tastiera
        do {
                c = getchar();
        } while (c != '\n');
}

if(hide) {
        //L'a capo dopo l'input
        (void) write(fileno(stdout), "\n", 1);

        // Ripristina le impostazioni precedenti dello schermo
        (void) tcsetattr(fileno(stdin), TCSAFLUSH, &oterm);

        // Ripristina la gestione dei segnali
        (void) sigaction(SIGALRM, &savealrm, NULL);
        (void) sigaction(SIGINT, &saveint, NULL);
        (void) sigaction(SIGHUP, &savehup, NULL);
```

```
		(void) sigaction(SIGQUIT, &savequit, NULL);
		(void) sigaction(SIGTERM, &saveterm, NULL);
		(void) sigaction(SIGTSTP, &savetstp, NULL);
		(void) sigaction(SIGTTIN, &savettin, NULL);
		(void) sigaction(SIGTTOU, &savettou, NULL);

		// Se era stato ricevuto un segnale viene rilanciato al processo stesso
		if(signo)
			(void) raise(signo);
	}

	return stringa;
}

// Per la gestione dei segnali
static void handler(int s) {
	signo = s;
}


bool yesOrNo(char *domanda, char yes, char no, bool predef, bool insensitive)
{

	// I caratteri 'yes' e 'no' devono essere minuscoli
	yes = tolower(yes);
	no = tolower(no);

	// Decide quale delle due lettere mostrare come predefinite
	char s, n;
	if(predef) {
		s = toupper(yes);
		n = no;
	} else {
		s = yes;
		n = toupper(no);
	}

	// Richiesta della risposta
	while(true) {
		// Mostra la domanda
		printf("%s [%c/%c]: ", domanda, s, n);

		char c;
		getInput(1, &c, false);

		// Controlla quale risposta è stata data
		if(c == '\0') { // getInput() non può restituire '\n'!
			return predef;
		} else if(c == yes) {
			return true;
```

```
                } else if(c == no) {
                        return false;
                } else if(c == toupper(yes)) {
                        if(predef || insensitive) return true;
                } else if(c == toupper(yes)) {
                        if(!predef || insensitive) return false;
                }
        }
}

char multiChoice(char *domanda, char choices[], int num)
{

        // Genera la stringa delle possibilità
        char *possib = malloc(2 * num * sizeof(char));
        int i, j = 0;
        for(i = 0; i < num; i++) {
                possib[j++] = choices[i];
                possib[j++] = '/';
        }
        possib[j-1] = '\0'; // Per eliminare l'ultima '/'

        // Chiede la risposta
        while(true) {
                // Mostra la domanda
                printf("%s [%s]: ", domanda, possib);

                char c;
                getInput(1, &c, false);

                // Controlla se è un carattere valido
                for(i = 0; i < num; i++) {
                        if(c == choices[i])
                                return c;
                }
        }
}
```

utils.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"



void print_stmt_error (MYSQL_STMT *stmt, char *message)
{
	fprintf (stderr, "%s\n", message);
	if (stmt != NULL) {
		fprintf (stderr, "Error %u (%s): %s\n",
			mysql_stmt_errno (stmt),
			mysql_stmt_sqlstate(stmt),
			mysql_stmt_error (stmt));
	}
}




void print_error(MYSQL *conn, char *message)
{
	fprintf (stderr, "%s\n", message);
	if (conn != NULL) {
		#if MYSQL_VERSION_ID >= 40101
		fprintf (stderr, "Error %u (%s): %s\n",
		mysql_errno (conn), mysql_sqlstate(conn), mysql_error (conn));
		#else
		fprintf (stderr, "Error %u: %s\n",
		mysql_errno (conn), mysql_error (conn));
		#endif
	}
}
```

```
bool choice()
{
	char c;

reinsert4:


	scanf("%c", &c);
	if(c == 'y' || c == 'Y'){
		flush(stdin);
		return true;
	}else{
		if(c == 'n' || c == 'N'){
			flush(stdin);
			return false;
		}else{
			printf("Could you reinsert your choice please?\n");
			flush(stdin);
			goto reinsert4;
		}
	}
}


bool setup_prepared_stmt(MYSQL_STMT **stmt, char *statement, MYSQL *conn)
{
	bool update_length = true;

	*stmt = mysql_stmt_init(conn);
	if (*stmt == NULL)
	{
		print_error(conn, "Could not initialize statement handler");
		return false;
	}

	if (mysql_stmt_prepare (*stmt, statement, strlen(statement)) != 0) {
		print_stmt_error(*stmt, "Could not prepare statement");
		return false;
	}

	mysql_stmt_attr_set(*stmt, STMT_ATTR_UPDATE_MAX_LENGTH, &update_length);

	return true;
}
```

```
void finish_with_error(MYSQL *conn, char *message)
{
	print_error(conn, message);
	mysql_close(conn);
	exit(EXIT_FAILURE);
}

void finish_with_stmt_error(MYSQL *conn, MYSQL_STMT *stmt, char *message, bool close_stmt)
{
	print_stmt_error(stmt, message);
	if(close_stmt)  mysql_stmt_close(stmt);
	mysql_close(conn);
	exit(EXIT_FAILURE);
}

static void print_dashes(MYSQL_RES *res_set)
{
	MYSQL_FIELD *field;
	unsigned int i, j;

	mysql_field_seek(res_set, 0);
	putchar('+');
	for (i = 0; i < mysql_num_fields(res_set); i++) {
		field = mysql_fetch_field(res_set);
		for (j = 0; j < field->max_length + 2; j++)
			putchar('-');
		putchar('+');
	}
	putchar('\n');
}

static void dump_result_set_header(MYSQL_RES *res_set)
{
	MYSQL_FIELD *field;
	unsigned long col_len;
	unsigned int i;

	/* determine column display widths -- requires result set to be */
	/* generated with mysql_store_result(), not mysql_use_result() */

	mysql_field_seek (res_set, 0);

	for (i = 0; i < mysql_num_fields (res_set); i++) {
		field = mysql_fetch_field (res_set);
		col_len = strlen(field->name);

		if (col_len < field->max_length)
			col_len = field->max_length;
		if (col_len < 4 && !IS_NOT_NULL(field->flags))
```

```
                        col_len = 4; /* 4 = length of the word "NULL" */
                field->max_length = col_len; /* reset column info */
        }
        print_dashes(res_set);
        putchar('|');
        mysql_field_seek (res_set, 0);
        for (i = 0; i < mysql_num_fields(res_set); i++) {
                field = mysql_fetch_field(res_set);
                printf(" %-*s |", (int)field->max_length, field->name);
        }
        putchar('\n');

        print_dashes(res_set);
}

void dump_result_set(MYSQL *conn, MYSQL_STMT *stmt, char *title)
{
        int i;
        int status;
        int num_fields;       /* number of columns in result */
        MYSQL_FIELD *fields;  /* for result set metadata */
        MYSQL_BIND *rs_bind;  /* for output buffers */
        MYSQL_RES *rs_metadata;
        MYSQL_TIME *date;
        size_t attr_size;

        /* Prefetch the whole result set. This in conjunction with
         * STMT_ATTR_UPDATE_MAX_LENGTH set in `setup_prepared_stmt`
         * updates the result set metadata which are fetched in this
         * function, to allow to compute the actual max length of
         * the columns.
         */
        if (mysql_stmt_store_result(stmt)) {
                fprintf(stderr, " mysql_stmt_execute(), 1 failed\n");
                fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
                exit(0);
        }

        /* the column count is > 0 if there is a result set */
        /* 0 if the result is only the final status packet */
        num_fields = mysql_stmt_field_count(stmt);

        if (num_fields > 0) {
                /* there is a result set to fetch */
                printf("%s\n", title);

                if((rs_metadata = mysql_stmt_result_metadata(stmt)) == NULL) {
                        finish_with_stmt_error(conn, stmt, "Unable to retrieve result metadata\n",
                        true);
                }
```

```
dump_result_set_header(rs_metadata);

fields = mysql_fetch_fields(rs_metadata);

rs_bind = (MYSQL_BIND *)malloc(sizeof (MYSQL_BIND) * num_fields);
if (!rs_bind) {
        finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n", true);
}
memset(rs_bind, 0, sizeof (MYSQL_BIND) * num_fields);

/* set up and bind result set output buffers */
for (i = 0; i < num_fields; ++i) {

        // Properly size the parameter buffer
        switch(fields[i].type) {
                case MYSQL_TYPE_DATE:
                case MYSQL_TYPE_TIMESTAMP:
                case MYSQL_TYPE_DATETIME:
                case MYSQL_TYPE_TIME:
                        attr_size = sizeof(MYSQL_TIME);
                        break;
                case MYSQL_TYPE_FLOAT:
                        attr_size = sizeof(float);
                        break;
                case MYSQL_TYPE_DOUBLE:
                        attr_size = sizeof(double);
                        break;
                case MYSQL_TYPE_TINY:
                        attr_size = sizeof(signed char);
                        break;
                case MYSQL_TYPE_SHORT:
                case MYSQL_TYPE_YEAR:
                        attr_size = sizeof(short int);
                        break;
                case MYSQL_TYPE_LONG:
                case MYSQL_TYPE_INT24:
                        attr_size = sizeof(int);
                        break;
                case MYSQL_TYPE_LONGLONG:
                        attr_size = sizeof(int);
                        break;
                default:
                        attr_size = fields[i].max_length;
                        break;
        }

        // Setup the binding for the current parameter
        rs_bind[i].buffer_type = fields[i].type;
        rs_bind[i].buffer = malloc(attr_size + 1);
```

```
        rs_bind[i].buffer_length = attr_size + 1;

        if(rs_bind[i].buffer == NULL) {
                finish_with_stmt_error(conn, stmt, "Cannot allocate output buffers\n",
                true);
        }
}

if(mysql_stmt_bind_result(stmt, rs_bind)) {
        finish_with_stmt_error(conn, stmt, "Unable to bind output parameters\n",
        true);
}

/* fetch and display result set rows */
while (true) {
        status = mysql_stmt_fetch(stmt);

        if (status == 1 || status == MYSQL_NO_DATA)
                break;

        putchar('|');

        for (i = 0; i < num_fields; i++) {

                if (rs_bind[i].is_null_value) {
                        printf (" %-*s |", (int)fields[i].max_length, "NULL");
                        continue;
                }

                switch (rs_bind[i].buffer_type) {

                        case MYSQL_TYPE_VAR_STRING:
                        case MYSQL_TYPE_DATETIME:
                printf(" %-*s |", (int)fields[i].max_length, (char*)rs_bind[i].buffer);
                                break;

                        case MYSQL_TYPE_DATE:
                        case MYSQL_TYPE_TIMESTAMP:
                                date = (MYSQL_TIME *)rs_bind[i].buffer;
                printf(" %d-%02d-%02d |", date->year, date->month, date->day);
                                break;

                        case MYSQL_TYPE_STRING:
                printf(" %-*s |", (int)fields[i].max_length, (char *)rs_bind[i].buffer);
                                break;

                        case MYSQL_TYPE_FLOAT:
                        case MYSQL_TYPE_DOUBLE:
                                printf(" %.02f |", *(float *)rs_bind[i].buffer);
                                break;
```

```
                                        case MYSQL_TYPE_LONG:
                                        case MYSQL_TYPE_LONGLONG:
                                        case MYSQL_TYPE_SHORT:
                                        case MYSQL_TYPE_TINY:
                                    printf(" %-*d |", (int)fields[i].max_length, *(int *)rs_bind[i].buffer);
                                            break;

                                        case MYSQL_TYPE_NEWDECIMAL:
                                printf(" %-*.02lf |", (int)fields[i].max_length, *(float*) rs_bind[i].buffer);
                                            break;

                                        default:
                                    printf("ERROR: Unhandled type (%d)\n", rs_bind[i].buffer_type);
                                        abort();
                                    }
                                }
                                putchar('\n');
                                print_dashes(rs_metadata);
                        }

                        mysql_free_result(rs_metadata); /* free metadata */


                        /* free output buffers */
                        for (i = 0; i < num_fields; i++) {
                                free(rs_bind[i].buffer);
                        }
                        free(rs_bind);
                }
}
```

parse.c

```
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "defines.h"

#define BUFF_SIZE 4096

// The final config struct will point into this
static char config[BUFF_SIZE];
```

```
/**
 * JSON type identifier. Basic types are:
 *      o Object
 *      o Array
 *      o String
 *      o Other primitive: number, boolean (true/false) or null
 */
typedef enum {
        JSMN_UNDEFINED = 0,
        JSMN_OBJECT = 1,
        JSMN_ARRAY = 2,
        JSMN_STRING = 3,
        JSMN_PRIMITIVE = 4
} jsmntype_t;

enum jsmnerr {
        /* Not enough tokens were provided */
        JSMN_ERROR_NOMEM = -1,
        /* Invalid character inside JSON string */
        JSMN_ERROR_INVAL = -2,
        /* The string is not a full JSON packet, more bytes expected */
        JSMN_ERROR_PART = -3
};

/**
 * JSON token description.
 * type         type (object, array, string etc.)
 * start start position in JSON data string
 * end          end position in JSON data string
 */
typedef struct {
        jsmntype_t type;
        int start;
        int end;
        int size;
#ifdef JSMN_PARENT_LINKS
        int parent;
#endif
} jsmntok_t;

/**
 * JSON parser. Contains an array of token blocks available. Also stores
 * the string being parsed now and current position in that string
 */
typedef struct {
        unsigned int pos; /* offset in the JSON string */
        unsigned int toknext; /* next token to allocate */
        int toksuper; /* superior token node, e.g parent object or array */
} jsmn_parser;
```

```
/**
 * Allocates a fresh unused token from the token pool.
 */
static jsmntok_t *jsmn_alloc_token(jsmn_parser *parser, jsmntok_t *tokens, size_t num_tokens) {
	jsmntok_t *tok;
	if (parser->toknext >= num_tokens) {
		return NULL;
	}
	tok = &tokens[parser->toknext++];
	tok->start = tok->end = -1;
	tok->size = 0;
#ifdef JSMN_PARENT_LINKS
	tok->parent = -1;
#endif
	return tok;
}

/**
 * Fills token type and boundaries.
 */
static void jsmn_fill_token(jsmntok_t *token, jsmntype_t type,
			int start, int end) {
	token->type = type;
	token->start = start;
	token->end = end;
	token->size = 0;
}

/**
 * Fills next available token with JSON primitive.
 */
static int jsmn_parse_primitive(jsmn_parser *parser, const char *js,
		size_t len, jsmntok_t *tokens, size_t num_tokens) {
	jsmntok_t *token;
	int start;

	start = parser->pos;

	for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
		switch (js[parser->pos]) {
#ifndef JSMN_STRICT
			/* In strict mode primitive must be followed by "," or "}" or "]" */
			case ':':
#endif
			case '\t' : case '\r' : case '\n' : case ' ' :
			case ','  : case ']'  : case '}' :
				goto found;
		}
		if (js[parser->pos] < 32 || js[parser->pos] >= 127) {
```

```
                        parser->pos = start;
                        return JSMN_ERROR_INVAL;
                }
        }
#ifdef JSMN_STRICT
        /* In strict mode primitive must be followed by a comma/object/array */
        parser->pos = start;
        return JSMN_ERROR_PART;
#endif

found:
        if (tokens == NULL) {
                parser->pos--;
                return 0;
        }
        token = jsmn_alloc_token(parser, tokens, num_tokens);
        if (token == NULL) {
                parser->pos = start;
                return JSMN_ERROR_NOMEM;
        }
        jsmn_fill_token(token, JSMN_PRIMITIVE, start, parser->pos);
#ifdef JSMN_PARENT_LINKS
        token->parent = parser->toksuper;
#endif
        parser->pos--;
        return 0;
}

/**
 * Fills next token with JSON string.
 */
static int jsmn_parse_string(jsmn_parser *parser, const char *js,
                size_t len, jsmntok_t *tokens, size_t num_tokens) {
        jsmntok_t *token;

        int start = parser->pos;

        parser->pos++;

        /* Skip starting quote */
        for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
                char c = js[parser->pos];

                /* Quote: end of string */
                if (c == '\"') {
                        if (tokens == NULL) {
                                return 0;
                        }
                        token = jsmn_alloc_token(parser, tokens, num_tokens);
                        if (token == NULL) {
```

```
                        parser->pos = start;
                        return JSMN_ERROR_NOMEM;
                }
                jsmn_fill_token(token, JSMN_STRING, start+1, parser->pos);
#ifdef JSMN_PARENT_LINKS
                token->parent = parser->toksuper;
#endif
                return 0;
        }

        /* Backslash: Quoted symbol expected */
        if (c == '\\' && parser->pos + 1 < len) {
                int i;
                parser->pos++;
                switch (js[parser->pos]) {
                        /* Allowed escaped symbols */
                        case '\"': case '/' : case '\\' : case 'b' :
                        case 'f' : case 'r' : case 'n'  : case 't' :
                                break;
                        /* Allows escaped symbol \uXXXX */
                        case 'u':
                                parser->pos++;
                        for(i = 0; i < 4 && parser->pos < len && js[parser->pos] != '\0'; i++) {
                                        /* If it isn't a hex character we have an error */
                        if(!((js[parser->pos] >= 48 && js[parser->pos] <= 57) || /* 0-9 */
                                (js[parser->pos] >= 65 && js[parser->pos] <= 70) || /* A-F */
                                (js[parser->pos] >= 97 && js[parser->pos] <= 102))) { /* a-f */
                                                parser->pos = start;
                                                return JSMN_ERROR_INVAL;
                                        }
                                        parser->pos++;
                                }
                                parser->pos--;
                                break;
                        /* Unexpected symbol */
                        default:
                                parser->pos = start;
                                return JSMN_ERROR_INVAL;
                }
        }
    }
    parser->pos = start;
    return JSMN_ERROR_PART;
}

/**
 * Parse JSON string and fill tokens.
 */
static int jsmn_parse(jsmn_parser *parser, const char *js, size_t len, jsmntok_t *tokens, unsigned int
num_tokens) {
```

```
	int r;
	int i;
	jsmntok_t *token;
	int count = parser->toknext;

	for (; parser->pos < len && js[parser->pos] != '\0'; parser->pos++) {
		char c;
		jsmntype_t type;

		c = js[parser->pos];
		switch (c) {
			case '{': case '[':
				count++;
				if (tokens == NULL) {
					break;
				}
				token = jsmn_alloc_token(parser, tokens, num_tokens);
				if (token == NULL)
					return JSMN_ERROR_NOMEM;
				if (parser->toksuper != -1) {
					tokens[parser->toksuper].size++;
#ifdef JSMN_PARENT_LINKS
					token->parent = parser->toksuper;
#endif
				}
				token->type = (c == '{' ? JSMN_OBJECT : JSMN_ARRAY);
				token->start = parser->pos;
				parser->toksuper = parser->toknext - 1;
				break;
			case '}': case ']':
				if (tokens == NULL)
					break;
				type = (c == '}' ? JSMN_OBJECT : JSMN_ARRAY);
#ifdef JSMN_PARENT_LINKS
				if (parser->toknext < 1) {
					return JSMN_ERROR_INVAL;
				}
				token = &tokens[parser->toknext - 1];
				for (;;) {
					if (token->start != -1 && token->end == -1) {
						if (token->type != type) {
							return JSMN_ERROR_INVAL;
						}
						token->end = parser->pos + 1;
						parser->toksuper = token->parent;
						break;
					}
					if (token->parent == -1) {
						if(token->type != type || parser->toksuper == -1) {
							return JSMN_ERROR_INVAL;
```

```
                                        }
                                        break;
                                }
                                token = &tokens[token->parent];
                        }
#else
                        for (i = parser->toknext - 1; i >= 0; i--) {
                                token = &tokens[i];
                                if (token->start != -1 && token->end == -1) {
                                        if (token->type != type) {
                                                return JSMN_ERROR_INVAL;
                                        }
                                        parser->toksuper = -1;
                                        token->end = parser->pos + 1;
                                        break;
                                }
                        }
                        /* Error if unmatched closing bracket */
                        if (i == -1) return JSMN_ERROR_INVAL;
                        for (; i >= 0; i--) {
                                token = &tokens[i];
                                if (token->start != -1 && token->end == -1) {
                                        parser->toksuper = i;
                                        break;
                                }
                        }
#endif
                        break;
                case '\"':
                        r = jsmn_parse_string(parser, js, len, tokens, num_tokens);
                        if (r < 0) return r;
                        count++;
                        if (parser->toksuper != -1 && tokens != NULL)
                                tokens[parser->toksuper].size++;
                        break;
                case '\t' : case '\r' : case '\n' : case ' ':
                        break;
                case ':':
                        parser->toksuper = parser->toknext - 1;
                        break;
                case ',':
                        if (tokens != NULL && parser->toksuper != -1 &&
                                        tokens[parser->toksuper].type != JSMN_ARRAY &&
                                        tokens[parser->toksuper].type != JSMN_OBJECT) {
#ifdef JSMN_PARENT_LINKS
                                parser->toksuper = tokens[parser->toksuper].parent;
#else
                                for (i = parser->toknext - 1; i >= 0; i--) {
                                        if (tokens[i].type == JSMN_ARRAY || tokens[i].type
== JSMN_OBJECT) {
```

```
                                                if (tokens[i].start != -1 && tokens[i].end == -1)
{
                                                        parser->toksuper = i;
                                                        break;
                                                }
                                        }
                                }
#endif
                                }
                                break;
#ifdef JSMN_STRICT
                        /* In strict mode primitives are: numbers and booleans */
                        case '-': case '0': case '1' : case '2': case '3' : case '4':
                        case '5': case '6': case '7' : case '8': case '9':
                        case 't': case 'f': case 'n' :
                                /* And they must not be keys of the object */
                                if (tokens != NULL && parser->toksuper != -1) {
                                        jsmntok_t *t = &tokens[parser->toksuper];
                                        if (t->type == JSMN_OBJECT ||
                                                        (t->type == JSMN_STRING && t->size != 0)) {
                                                return JSMN_ERROR_INVAL;
                                        }
                                }
#else
                        /* In non-strict mode every unquoted value is a primitive */
                        default:
#endif
                                r = jsmn_parse_primitive(parser, js, len, tokens, num_tokens);
                                if (r < 0) return r;
                                count++;
                                if (parser->toksuper != -1 && tokens != NULL)
                                        tokens[parser->toksuper].size++;
                                break;

#ifdef JSMN_STRICT
                        /* Unexpected char in strict mode */
                        default:
                                return JSMN_ERROR_INVAL;
#endif
                }
        }

        if (tokens != NULL) {
                for (i = parser->toknext - 1; i >= 0; i--) {
                        /* Unmatched opened object or array */
                        if (tokens[i].start != -1 && tokens[i].end == -1) {
                                return JSMN_ERROR_PART;
                        }
                }
        }
```

```
	return count;
}

/**
 * Creates a new parser based over a given  buffer with an array of tokens
 * available.
 */
static void jsmn_init(jsmn_parser *parser) {
	parser->pos = 0;
	parser->toknext = 0;
	parser->toksuper = -1;
}

static int jsoneq(const char *json, jsmntok_t *tok, const char *s)
{
	if (tok->type == JSMN_STRING
	   && (int) strlen(s) == tok->end - tok->start
	   && strncmp(json + tok->start, s, tok->end - tok->start) == 0) {
		return 0;
	}
	return -1;
}

static size_t load_file(char *filename)
{
	FILE *f = fopen(filename, "rb");
	if(f == NULL) {
		fprintf(stderr, "Unable to open file %s\n", filename);
		exit(1);
	}

	fseek(f, 0, SEEK_END);
	size_t fsize = ftell(f);
	fseek(f, 0, SEEK_SET);  //same as rewind(f);

	if(fsize >= BUFF_SIZE) {
		fprintf(stderr, "Configuration file too large\n");
		abort();
	}
	fread(config, fsize, 1, f);
	fclose(f);

	config[fsize] = 0;
	return fsize;
}
```

```
int parse_config(char *path, struct configuration *conf)
{
	int i;
	int r;
	jsmn_parser p;
	jsmntok_t t[128]; /* We expect no more than 128 tokens */

	load_file(path);

	jsmn_init(&p);
	r = jsmn_parse(&p, config, strlen(config), t, sizeof(t)/sizeof(t[0]));
	if (r < 0) {
		printf("Failed to parse JSON: %d\n", r);
		return 0;
	}

	/* Assume the top-level element is an object */
	if (r < 1 || t[0].type != JSMN_OBJECT) {
		printf("Object expected\n");
		return 0;
	}
	/* Loop over all keys of the root object */
	for (i = 1; i < r; i++) {
		if (jsoneq(config, &t[i], "host") == 0) {
			/* We may use strndup() to fetch string value */
			conf->host = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "username") == 0) {
			conf->db_username = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "password") == 0) {
			conf->db_password = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else if (jsoneq(config, &t[i], "port") == 0) {
			conf->port = strtol(config + t[i+1].start, NULL, 10);
			i++;
		} else if (jsoneq(config, &t[i], "database") == 0) {
			conf->database = strndup(config + t[i+1].start, t[i+1].end-t[i+1].start);
			i++;
		} else {
			printf("Unexpected key: %.*s\n", t[i].end-t[i].start, config + t[i].start);
		}
	}
	return 1;
}
```

Makefile

```
all:
	gcc -g *.c -Wall  -Wextra -o client `mysql_config --cflags --include --libs`
clean:
	-rm client
```