BASI DI DATI – PROGETTO GRUPPO NUMERO 5

[Descrizione della realtà da analizzare 2](#_Toc25074196)

[Glossario 3](#_Toc25074197)

[Schema scheletro 5](#_Toc25074198)

[Progettazione concettuale 6](#_Toc25074199)

[Schema ER completo 10](#_Toc25074200)

[Progettazione logica 11](#_Toc25074201)

[Eliminazione delle gerarchie ISA 11](#_Toc25074202)

[Selezione delle chiavi primarie ed eliminazione delle identificazioni esterne 13](#_Toc25074203)

[Traduzione di entità e associazioni in schemi di relazioni 15](#_Toc25074204)

[Verifica della normalizzazione 20](#_Toc25074205)

[Operazioni previste dalla base di dati – Descrizione e relativo codice SQL 21](#_Toc25074206)

[Query di interrogazione 21](#_Toc25074207)

[Query di modifica 23](#_Toc25074208)

[Query di eliminazione 24](#_Toc25074209)

[Codice SQL per la creazione delle tabelle 25](#_Toc25074210)

[Codice SQL per l’inserimento dei dati 31](#_Toc25074211)

[Stored procedure, funzioni e trigger 36](#_Toc25074212)

[Studio di dato derivato 49](#_Toc25074213)

[Progetto fisico 50](#_Toc25074214)

# **Descrizione della realtà da analizzare**

Si vuole modellare il database di un’azienda di stoccaggio e spedizione di merci, l’azienda ha più filiali, ogni filiale è identificata da un codice progressivo, un nome, la città, il telefono, la via e il numero civico. Una filiale può avere più magazzini in cui vengono portati i prodotti, ogni magazzino è identificato da un numero progressivo all’interno della filiale di appartenenza, città, via, numero civico, telefono e denominazione. Ogni magazzino è suddiviso in più spazi di dimensione fissa, ogni spazio fa riferimento a un contratto, al decadimento del contratto lo spazio nel magazzino viene liberato. In uno spazio possono essere contenuti più prodotti, ognuno in una certa quantità. Di ogni spazio si vuole tenere conto di un codice identificativo, di una descrizione e della sua posizione all’interno del magazzino. Di ogni prodotto vengono memorizzati un codice seriale, il nome e una descrizione.

Il personale che lavora negli stabilimenti, è identificato dai comuni dati anagrafici ed è suddiviso in 5 categorie:

* Dirigente, può dirigere più di una filiale, una filiale ha un solo dirigente.
* Impiegato, lavora in una sola filiale, una filiale ha più impiegati, si occupa della gestione dei contratti.

Magazziniere, lavora in un unico magazzino e si occupa delle operazioni di stoccaggio, depositi e prelievi di prodotti, ogni magazzino può avere più operatori.

* Custode, controlla il magazzino in turni da 6 ore, fornita una data e un custode deve essere unico il magazzino, possono essere svolti più turni contemporaneamente su un unico magazzino.
* Fattorino, si occupa delle spedizioni da una filiale all’altra.

Ogni impiegato può ordinare trasferimenti di prodotti (che appartengono alla propria filiale) tra due magazzini. Di ogni trasferimento si vuole salvare la data, il magazzino di partenza, il magazzino di arrivo, i prodotti coinvolti e un numero progressivo. Un cliente può, invece, richiedere una spedizione, in tal caso il prodotto viene spedito direttamente a un indirizzo da lui indicato. Di ogni spedizione si vogliono salvare la data, il numero della spedizione, la città, il paese, la via, il numero civico, il telefono, il fattorino, lo stato della consegna e il veicolo con cui è stata compiuta, in una data un fattorino può usare più veicoli ed effettuare più spedizioni. Una filiale può avere uno o più veicoli, di ogni veicolo si vuole tenere traccia di targa, marca, modello e capacità.

Ogni cliente può acquistare uno o più spazi per mezzo di un contratto stipulato con un impiegato, di ogni contratto si registrano il numero del contratto, data di inizio, data di fine, l’importo dovuto e il numero di spazi acquistati. Al termine di un contratto il cliente può decidere di rinnovarlo o annullarlo. In caso di annullamento gli spazi a cui lo stesso faceva riferimento vengono liberati. Di ogni cliente vengono memorizzati i dati anagrafici (codice fiscale, telefono, nome e cognome).

Ogni cliente è associato a un piano tariffario a seconda del suo livello di fedeltà (che viene aggiornato al momento della stipulazione del contratto in base al numero di spazi acquistati in passato e al relativo tempo di acquisto), ogni piano di offerta porterà a uno sconto del prezzo di acquisto degli spazi. Di ogni piano si vuole salvare il nome, una descrizione ed il prezzo relativo a uno spazio.

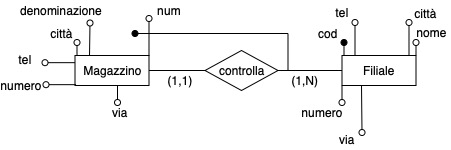
# **Glossario**

|  |  |  |  |
| --- | --- | --- | --- |
| Termine | Descrizione | Sinonimi | Legami |
| Personale | * Codice fiscale * Nome * Cognome * Data di nascita * Mail * Telefono |  | * Custode * Magazziniere * Dirigente * Impiegato * Fattorino |
| Custode |  |  | * Turno |
| Magazziniere |  |  | * Magazzino |
| Dirigente |  |  | * Filiale |
| Impiegato |  |  | * Filiale * Contratto |
| Fattorino |  |  | * Movimenti |
| Turno | * Data |  | * Custode * Magazzino |
| Magazzino | * Numero progressivo * Città * Via * Numero * Telefono * Denominazione |  | * Turno * Magazziniere * Filiale * Spazio * Trasferimento |
| Filiale | * Nome * Codice progressivo * Città * Via * Numero * Telefono |  | * Magazzino * Dirigente * Impiegato |
| Spazio | * Codice identificativo * Descrizione |  | * Magazzino * Contratto * Contiene |
| Contiene | * Quantità |  | * Spazio * Prodotto |
| Prodotto | * Codice seriale * Nome * Descrizione |  | * Contiene * Movimenti |
| Cliente | * Codice fiscale * Nome * Cognome * Telefono |  | * Piano * Spedizione * Contratto |
| Contratto | * Data di inizio * Data di fine * Numero contratto * Numero di spazi |  | * Spazio * Cliente * Impiegato |
| Piano | * Nome * Descrizione * Importo |  | * Cliente |
| Movimenti | * Numero spedizione * Data spedizione * Stato della consegna |  | * Spedizione * Trasferimento * Fattorino * Prodotto * Veicolo |
| Spedizione | * Città * Paese * Via * Numero * Telefono |  | * Cliente |
| Trasferimenti |  |  | * Magazzino |
| Veicolo | * Targa * Marca * Modello * Capacità |  | * Movimenti |

# **Schema scheletro**

# **Progettazione concettuale**

*“L’azienda ha più filiali, ogni filiale è identificata da un codice progressivo, un nome, la città, il telefono, la via e il numero civico. Una filiale può avere più magazzini in cui vengono portati i prodotti, ogni magazzino è identificato da un numero progressivo all’interno della filiale di appartenenza, città, via, numero civico, telefono e denominazione.”*



Siccome ogni magazzino è identificato da un numero progressivo all’interno della filiale di appartenenza, ne consegue la presenza dell’identificatore esterno dall’entità filiale.

Per quanto riguarda le cardinalità, una filiale deve avere almeno un magazzino ma può averne anche di più, un magazzino appartiene ad una sola filiale.

*“Il personale che lavora negli stabilimenti, è identificato dai comuni dati anagrafici ed è suddiviso in 5 categorie:*

* *Dirigente, può dirigere più di una filiale, una filiale ha un solo dirigente.*
* *Impiegato, lavora in una sola filiale, una filiale ha più impiegati, si occupa della gestione dei contratti.*

*Magazziniere, lavora in un unico magazzino e si occupa delle operazioni di stoccaggio, depositi e prelievi di prodotti, ogni magazzino può avere più operatori.*

* *Custode, controlla il magazzino in turni da 6 ore, fornita una data e un custode deve essere unico il magazzino, possono essere svolti più turni contemporaneamente su un unico magazzino.*
* Immagine che contiene testo, mappa

  Descrizione generata automaticamente*Fattorino, si occupa delle spedizioni da una filiale all’altra.”*

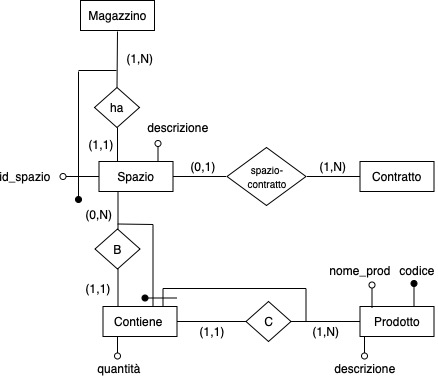
Per rappresentare il personale di ogni azienda abbiamo deciso di utilizzare una gerarchia ad un livello visto che ogni categoria del personale è identificata dai comuni dati anagrafici.

Le entità Custode, Magazziniere, Dirigente, Impiegato e Fattorino sono figlie dell’entità Personale. Si tratta di una gerarchia totale, perché ogni istanza dell’entità personale deve far parte delle entità figlie, ed esclusiva, perché il fatto di essere un custode esclude il fatto di essere un magazziniere o qualsiasi altro tipo di categoria del personale.

Per gestire il vincolo riguardante i turni del custode abbiamo deciso di inserire l’entità Turno, che ha come attributo la data, collegandola sia all’entità Custode che all’entità Magazzino. Siccome fornita una data e un custode deve essere unico il magazzino abbiamo inserito l’identificatore esterno dall’entità Custode all’entità Turno.

Per quanto riguarda le cardinalità, una filiale può essere diretta da uno e un solo dirigente e un dirigente può dirigere più filiali; in una determinata filiale ci deve essere almeno un impiegato e possono essercene più di uno, un impiegato può lavorare in una sola filiale; in un magazzino ci deve essere almeno un magazziniere e ce ne possono essere più di uno, un magazziniere può lavorare in un solo magazzino.

*“Ogni magazzino è suddiviso in più spazi di dimensione fissa, ogni spazio fa riferimento a un contratto, al decadimento del contratto lo spazio nel magazzino viene liberato. In uno spazio possono essere contenuti più prodotti, ognuno in una certa quantità. Di ogni spazio si vuole tenere conto di un codice identificativo, di una descrizione e della sua posizione all’interno del magazzino. Di ogni prodotto vengono memorizzati un codice seriale, il nome e una descrizione.”*



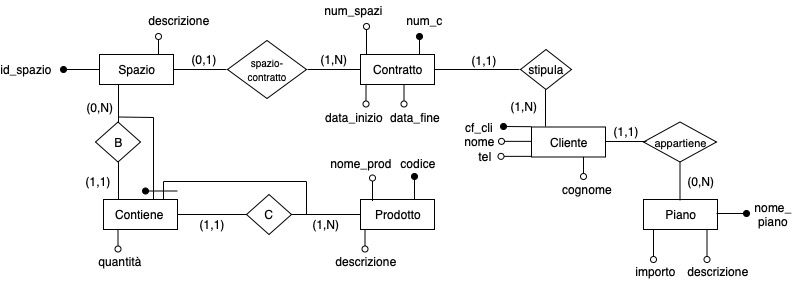
Siccome ogni spazio è identificato da un ID all’interno di un magazzino di riferimento, ne consegue la presenza dell’identificatore esterno dall’entità magazzino.

Dato l’aspetto dinamico della relazione spazio-prodotto abbiamo deciso di creare una nuova entità di nome Contiene che comprende l’id dello spazio, il codice del prodotto e la relativa quantità grazie alle identificazioni esterne dall’entità Prodotto e dall’entità Spazio. Ogni modifica effettuata sulla nuova tabella dell’entità Contiene corrisponde a un prelievo/versamento di un prodotto relativo a uno spazio.

Per quanto riguarda le cardinalità, un magazzino deve avere almeno uno spazio di dimensione fissa e ce ne possono essere più di uno, uno spazio appartiene ad un solo magazzino; uno spazio può fare riferimento ad al più un contratto, ma può anche far riferimento a nessun contratto, un contratto deve contenere almeno uno spazio, ma può anche contenerne più di uno; un prodotto deve essere contenuto in almeno uno spazio, ma può essere contenuto anche in più spazi, uno spazio può contenere più prodotti ma può anche non contenere nessun prodotto.

*“Ogni cliente può acquistare uno o più spazi per mezzo di un contratto stipulato con un impiegato, di ogni contratto si registrano il numero del contratto, data di inizio, data di fine, l’importo dovuto e il numero di spazi acquistati. Al termine di un contratto il cliente può decidere di rinnovarlo o annullarlo. In caso di annullamento gli spazi a cui lo stesso faceva riferimento vengono liberati. Di ogni cliente vengono memorizzati i dati anagrafici (codice fiscale, telefono, nome e cognome).”*

*“Ogni cliente è associato a un piano tariffario a seconda del suo livello di fedeltà (che viene aggiornato al momento della stipulazione del contratto in base al numero di spazi acquistati in passato e al relativo tempo di acquisto), ogni piano di offerta porterà a uno sconto del prezzo di acquisto degli spazi. Di ogni piano si vuole salvare il nome, una descrizione ed il prezzo relativo a uno spazio.”*

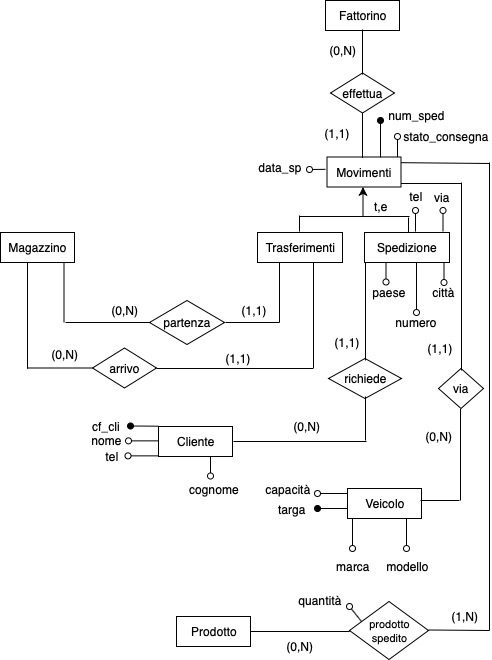
**

Abbiamo deciso di non collegare l’entità Cliente all’entità Prodotto per evitare una dipendenza circolare, quindi supponiamo che se un prodotto è all’interno di uno spazio esso è di proprietà del cliente che ha acquistato lo spazio stesso tramite un contratto.

Abbiamo deciso di non inserire l’attributo “importo dovuto” all’entità Contratto perché si può calcolare in base al piano a cui il cliente appartiene ed è per questo che abbiamo aggiunto l’entità Piano con i relativi attributi.

Per quanto riguarda le cardinalità, un cliente deve stipulare almeno un contratto e può stipularne più di uno, un contratto appartiene ad un solo cliente; un cliente appartiene ad un solo piano, un piano può appartenere a più clienti, ma può anche non appartenere ad alcun cliente; uno spazio può appartenere ad al più un contratto ma può anche non appartenere ad alcun contratto, un contratto deve essere stipulato per almeno uno spazio ma nello stesso contratto ci possono essere anche più spazi.

*“Ogni impiegato può ordinare trasferimenti di prodotti (cha appartengono alla propria filiale) tra due magazzini. Di ogni trasferimento si vuole salvare la data, il magazzino di partenza, il magazzino di arrivo, i prodotti coinvolti e un numero progressivo. Un cliente può, invece, richiedere una spedizione, in tal caso il prodotto viene spedito direttamente a un indirizzo da lui indicato. Di ogni spedizione si vogliono salvare la data, il numero della spedizione, la città, il paese, la via, il numero civico, il fattorino, lo stato della consegna e il veicolo con cui è stata compiuta, in una data un fattorino può usare più veicoli ed effettuare più spedizioni. Una filiale può avere uno o più veicoli, di ogni veicolo si vuole tenere traccia di targa, marca, modello e capacità.”*



Per gestire i trasferimenti e le spedizioni abbiamo deciso di rappresentarli con una gerarchia ad un livello. Le entità Trasferimenti e Spedizione sono figlie dell’entità Movimenti. Si tratta di una gerarchia totale, perché ogni istanza dell’entità Movimenti deve far parte delle entità figlie, ed esclusiva, in quanto trasferimento e spedizione rappresentano due tipi di movimenti diversi.

Siccome si vuole tener traccia del magazzino di partenza e di arrivo di un trasferimento abbiamo deciso di collegare l’entità Magazzino all’entità Trasferimenti con due associazioni una che tiene traccia del magazzino da cui viene trasferito il prodotto e una che tiene traccia del magazzino in cui arriva il prodotto traferito.

Abbiamo deciso di collegare l’entità Prodotto, tenendo traccia della quantità spedita, e l’entità Veicolo direttamente all’entità padre Movimenti in quanto sia la spedizione che il trasferimento riguardano un prodotto e vengono effettuati entrambi tramite un veicolo.

Per quanto riguarda le cardinalità, un fattorino può effettuare più movimenti ma può anche non effettuarne, un determinato movimento è effettuato da un solo fattorino; un movimento può essere fatto di più prodotti di un determinato cliente, un determinato prodotto può essere contenuto in più movimenti ma può anche non essere trasferito o spedito.

# **Schema ER completo**

Immagine che contiene testo, mappa

Descrizione generata automaticamente

# **Progettazione logica**

Lo schema ER descrive un dominio applicativo ad un dato livello di astrazione ed è utile per fornire una descrizione sintetica e visiva della base di dati. Non esistono però DBMS in grado di operare direttamente sui concetti di schemi ER, è quindi necessario tradurre lo schema ER in uno schema logico relazionale.

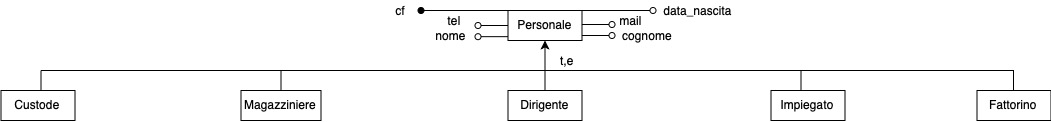
Questa trasformazione avviene attraverso cinque fasi :

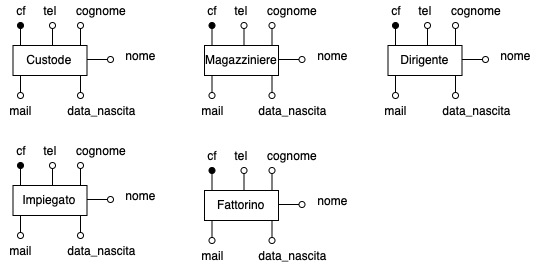
1. Eliminazione delle gerarchie ISA
2. Selezione delle chiavi primarie ed eliminazione delle identificazioni esterne
3. Trasformazioni degli attributi composti o multipli
4. Traduzione di entità e associazioni in schemi di relazioni
5. Verifica di normalizzazione

## **Eliminazione delle gerarchie ISA**

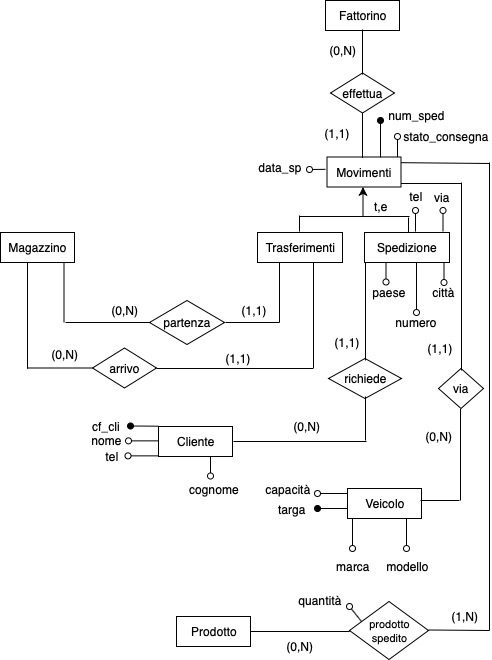
Come prima cosa bisogna eliminare le gerarchie e per farlo ci sono tre modi :

1. Mantenimento delle entità con associazioni : tutte le entità vengono mantenute, le entità figlie sono in associazione con l’entità padre e sono identificate esternamente tramite l’associazione. Questa soluzione è sempre possibile, indipendentemente dalla copertura.
2. Collasso verso l’alto : vengono riunite tutte le entità figlie nell’entità padre, gli attributi obbligatori per le entità figlie diventano opzionali per l’entità padre. È necessario l’utilizzo dei selettori che specificano se un’istanza dell’entità padre appartiene a una delle due sotto-entità. Questa soluzione favorisce le operazioni che consultano insieme gli attributi dell’entità padre e quelli di un’entità figlia.
3. Collasso verso il basso : viene eliminata l’entità padre e tutti i suoi attributi vengono trasferiti su tutte le entità figlie, le associazioni del padre vengono replicate per tutte le entità figlie. Questa soluzione favorisce le operazioni in cui si accede separatamente alle entità figlie e non applicabile se la copertura non è totale o non è esclusiva (introduce ridondanza).

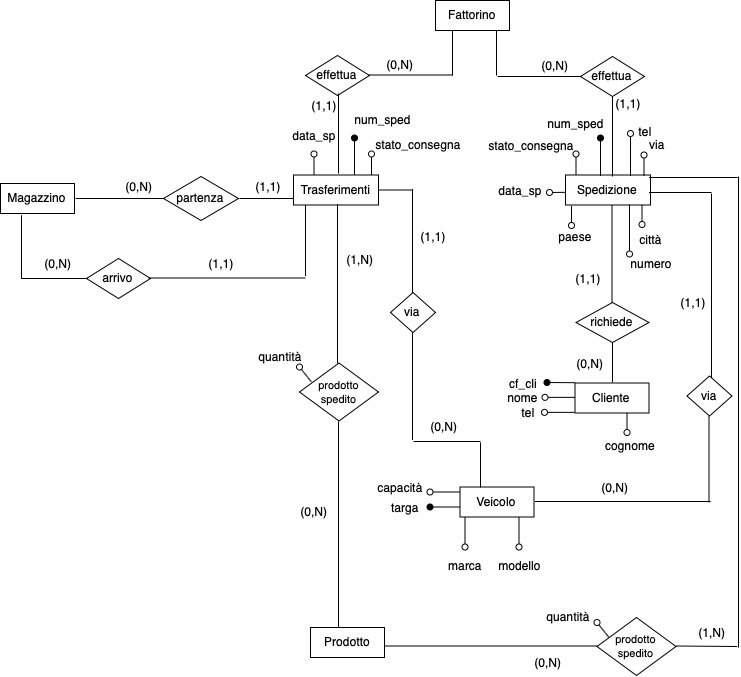
La prima gerarchia che prendiamo in esame è quella che riguarda la rappresentazione del Personale di un’azienda.

Essendo questa una gerarchia totale ed esclusiva abbiamo deciso di effettuare un collasso verso il basso visto che accediamo separatamente alle entità figlie e inoltre l’entità padre Personale non si interfaccia con alcuna entità del database. Tutti gli attributi dell’entità Personale vengono trasferiti su tutte le entità figlie.

L’altra gerarchia da analizzare è quella che riguarda l’entità padre Movimenti.

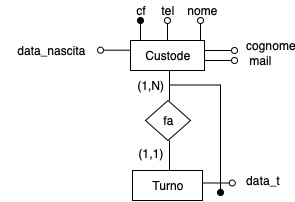


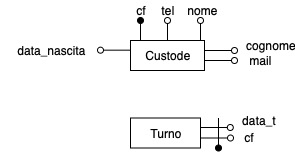
Anche in questo caso abbiamo deciso di applicare il collasso verso il basso, possibile perché si tratta di una gerarchia totale ed esclusiva, in quanto accediamo alle due entità figlie separatamente. Tutti gli attributi dell’entità padre vengono trasferiti su tutte le entità figlie e le associazioni dell’entità padre Movimenti vengono replicate per tutte le entità figlie.



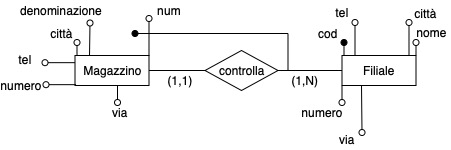
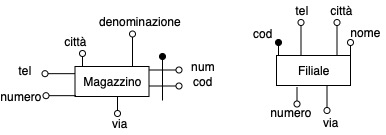
## **Selezione delle chiavi primarie ed eliminazione delle identificazioni esterne**

L’entità Turno ha una componente di identificazione esterna dall’entità Custode. Il risultato è il seguente :

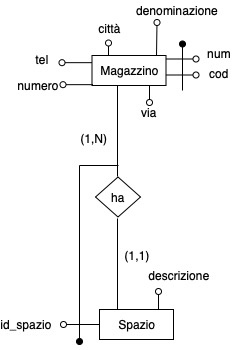
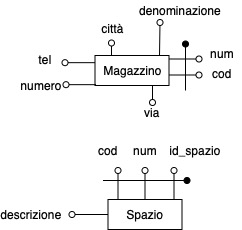




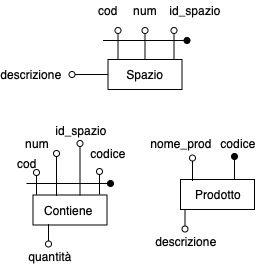
L’entità Magazzino ha una componente di identificazione esterna dall’entità Filiale. Il risultato è il seguente :

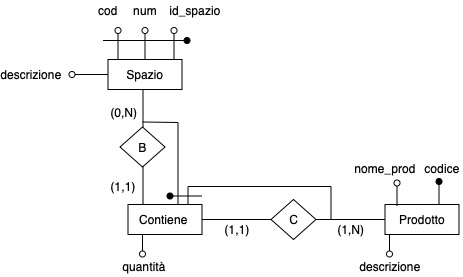


L’entità Spazio ha una componente di identificazione esterna dall’entità Magazzino. Il risultato è il seguente :



L’entità Contiene ha una componente di identificazione esterna dall’entità Spazio e dall’entità prodotto. Il risultato è il seguente :





## Immagine che contiene testo, mappa Descrizione generata automaticamente**Traduzione di entità e associazioni in schemi di relazioni**

Siccome l’entità Custode non ha associazioni da analizzare viene trasformata con una traduzione standard.

Le entità Turno e Magazzino invece sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “su” all’entità turno, così facendo la chiave dell’entità Magazzino verrà aggiunta come foreign key all’entità Turno.

Anche le entità Magazziniere e Magazzino sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “lavora” all’entità magazziniere, così facendo la chiave dell’entità Magazzino verrà aggiunta come foreign key all’entità Magazziniere.

Le entità Filiale e Dirigente sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “dirige” all’entità Filiale, così facendo la chiave dell’entità Dirigente verrà aggiunta come foreign key all’entità Filiale.

**Magazziniere** (CF, nome, cognome, tel, mail, data\_nascita, num, cod)

**FK:** num,cod references magazzino

**Custode** (CF, nome, cognome, tel, mail, data\_nascita)

**Filiale** (cod, nome, città, via, numero, tel, CF)

**FK:** CF references dirigente

**Magazzino** (num, cod, denominazione, città, via, numero, tel)

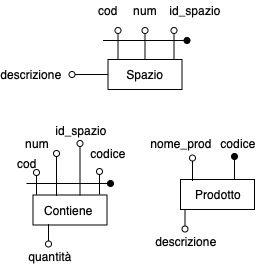
**FK:** cod references Filiale

**Turno** (data\_t, CF, num, cod)

**FK:** num,cod references magazzino

**FK:** CF references custode

**Dirigente** (CF, nome, cognome, tel, mail, data\_nascita)



Siccome le entità Spazio, Contiene e Prodotto non hanno associazioni da analizzare vengono trasformate con una traduzione standard.

**Spazio** (id\_spazio, num, cod, descrizione )

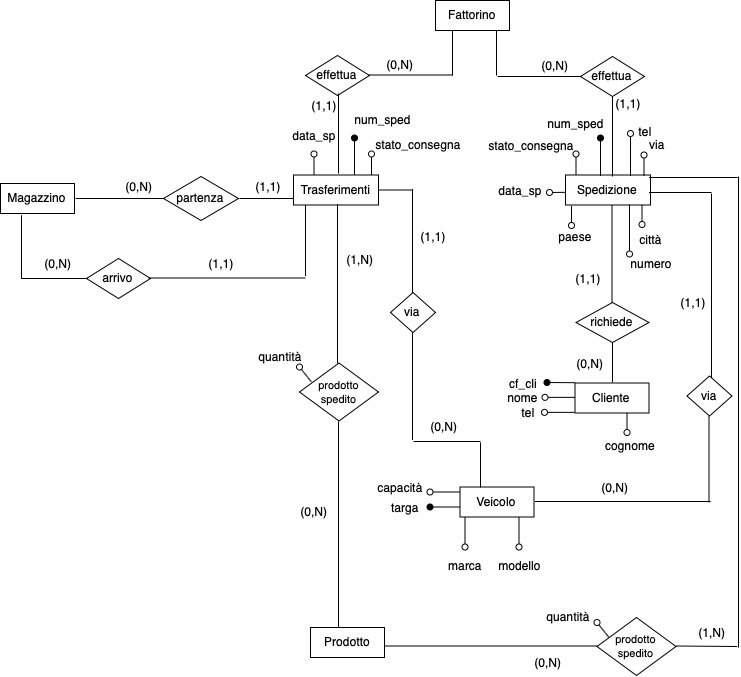
**FK:** num, cod references magazzino

**Prodotto** (codice, nome\_prod, descrizione)

**Contiene** (id\_spazio, num, cod, codice, quantità )

**FK:** id\_spazio, num, cod references spazio

**FK:** codice references prodotto



Le entità Fattorino e Trasferimenti sono legate da un’associazione N:1, per questo motivo è possibile accorpare l’associazione “effettua” all’entità Trasferimenti, così facendo la chiave dell’entità Fattorino verrà aggiunta come foreign key all’entità Trasferimenti.

Anche le entità Fattorino e Spedizione sono legate da un’associazione N:1, per questo motivo è possibile accorpare l’associazione “effettua” all’entità Spedizione, così facendo la chiave dell’entità Fattorino verrà aggiunta come foreign key all’entità Spedizione.

Le entità Spedizione e Prodotto sono legate da un’associazione N:M, per questo motivo vengono tradotte con una traduzione standard, cioè una relazione per ogni entità e una per l’associazione “prodotto spedito”.

Anche le entità Prodotto e Trasferimenti sono legate con una associazione N:M, per questo motivo vengono tradotte con una traduzione standard, cioè una relazione per ogni entità e una per l’associazione “prodotto trasferito”.

Le entità Trasferimenti e Magazzino sono legate da due associazioni 1:N, per questo motivo è possibile accorpare le due associazioni “partenza” e “arrivo” all’entità Trasferimenti, così facendo la chiave dell’entità Magazzino verrà aggiunta due volte come foreign key all’entità Trasferimenti, una volta come foreign key del magazzino di partenza. Una volta come foreig key del magazzino di arrivo.

Le entità Spedizione e Veicolo sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “via” all’entità Spedizione, così facendo la chiave dell’entità Veicolo verrà aggiunta come foreign key all’entità Spedizione.

Anche le entità Trasferimenti e Veicolo sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “via” all’entità Trasferimenti, così facendo la chiave dell’entità Veicolo verrà aggiunta come foreign key all’entità Trasferimenti.

Le entità Spedizione e Cliente sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “richiede” all’entità Spedizione, così facendo la chiave dell’entità Cliente verrà aggiunta come foreign key all’entità Spedizione.

**Spedizione** (num\_sped, data\_sp, CF, targa, paese, città, via, numero, stato\_consegna, cf\_cli, tel)

**FK:** CF references fattorino

**FK:** targa references veicolo

**FK:** cf\_cli references cliente

**Veicolo** (targa, modello, marca, capacità)

**Trasferimenti** (num\_sped, data\_sp, CF, targa, stato\_consegna num1, cod1, num2, cod2)

**FK:** num1, cod1 references magazzino (num,cod)

**FK:** num2, cod2 references magazzino (num,cod)

**FK:** CF references fattorino

**FK:** targa references veicolo

**Prod\_sped** (codice, num\_sped, quantità)

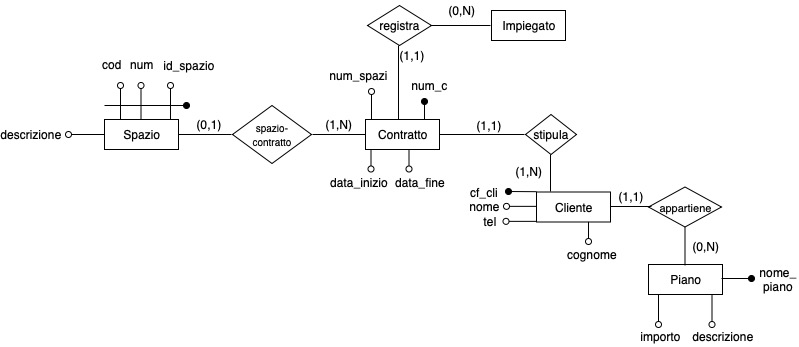
**FK:** codice references prodotto

**FK:** num\_sped references spedizione

**Prod\_trasf** (codice, num\_sped, quantità)

**FK:** codice references prodotto

**FK:** num\_sped references trasferimenti



Le entità Contratto e Impiegato sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “registra” all’entità Contratto, così facendo la chiave dell’entità Impiegato verrà aggiunta come foreign key all’entità Contratto.

Anche le entità Cliente e Piano sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “appartiene” all’entità Cliente, così facendo la chiave dell’entità Piano verrà aggiunta come foreign key all’entità Cliente.

Le entità Spazio e Contratto sono legate da un’associazione 1:N e l’entità Spazio partecipa con un una cardinalità 0,1, per questo motivo abbiamo deciso di usare la traduzione standard , cioè una relazione per ogni entità e una per l’associazione “spazio contratto”.

Le entità Contratto e Cliente sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “stipula” all’entità Contratto, così facendo la chiave dell’entità Cliente verrà aggiunta come foreign key all’entità Contratto.

**Cliente** (cf\_cli, nome, cognome, tel, nome\_piano)

**FK:** nome\_piano references Piano

**Piano** (nome\_piano, descrizione, importo)

**Contratto** (num\_c, data\_inizio, data\_fine, num\_spazi, cf , cf\_cli)

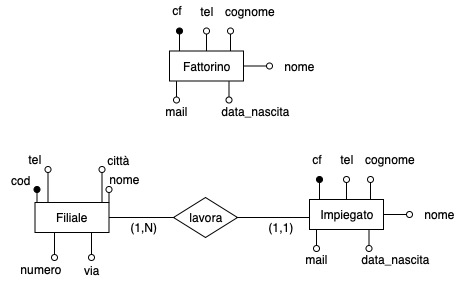
**FK:** cf\_cli references cliente

**FK:** cf references impiegato

**Spazio\_contratto** (num\_c, id\_spazio, num, cod)

**FK:** id\_spazio, num, cod references spazio

**FK:** num\_c references contratto



Siccome l’entità Fattorino non ha associazioni da analizzare viene trasformata con una traduzione standard.

Le entità Impiegato e Filiale sono legate da un’associazione 1:N, per questo motivo è possibile accorpare l’associazione “lavora” all’entità Impiegato, così facendo la chiave dell’entità Filiale verrà aggiunta come foreign key all’entità Impiegato.

**Impiegato** (CF, nome, cognome, tel, mail, data\_nascita, cod)

**FK:** cod references filiale

**Fattorino** (CF, nome, cognome, tel, mail, data\_nascita)

Riassumendo, il progetto logico completo è il seguente :

**Impiegato** (CF, nome, cognome, tel, mail, data\_nascita, cod)

**FK:** cod references filiale

**Fattorino** (CF, nome, cognome, tel, mail, data\_nascita)

**Magazziniere** (CF, nome, cognome, tel, mail, data\_nascita, num, cod)

**FK:** num,cod references magazzino

**Custode** (CF, nome, cognome, tel, mail, data\_nascita)

**Filiale** (cod, nome, città, via, numero, tel, CF)

**FK:** CF references dirigente

**Magazzino** (num, cod, denominazione, città, via, numero, tel)

**FK:** cod references Filiale

**Turno** (data\_t, CF, num, cod)

**FK:** num,cod references magazzino

**FK:** CF references custode

**Dirigente** (CF, nome, cognome, tel, mail, data\_nascita)

**Spazio** (id\_spazio, num, cod, descrizione )

**FK:** num, cod references magazzino

**Prodotto** (codice, nome\_prod, descrizione)

**Contiene** (id\_spazio, num, cod, codice, quantità )

**FK:** id\_spazio, num, cod references spazio

**FK:** codice references prodotto

**Cliente** (cf\_cli, nome, cognome, tel, nome\_piano)

**FK:** nome\_piano references Piano

**Piano** (nome\_piano, descrizione, importo)

**Contratto** (num\_c, data\_inizio, data\_fine, num\_spazi, cf , cf\_cli)

**FK:** cf\_cli references cliente

**FK:** cf references impiegato

**Spazio\_contratto** (num\_c, id\_spazio, num, cod)

**FK:** id\_spazio, num, cod references spazio

**FK:** num\_c references contratto

**Spedizione** (num\_sped, data\_sp, CF, targa, paese, città, via, numero, stato\_consegna, cf\_cli, tel)

**FK:** CF references fattorino

**FK:** targa references veicolo

**FK:** cf\_cli references cliente

**Veicolo** (targa, modello, marca, capacità)

**Trasferimenti** (num\_sped, data\_sp, CF, targa, stato\_consegna num1, cod1, num2, cod2)

**FK:** num1, cod1 references magazzino (num,cod)

**FK:** num2, cod2 references magazzino (num,cod)

**FK:** CF references fattorino

**FK:** targa references veicolo

**Prod\_sped** (codice, num\_sped, quantità)

**FK:** codice references prodotto

**FK:** num\_sped references spedizione

**Prod\_trasf** (codice, num\_sped, quantità)

**FK:** codice references prodotto

**FK:** num\_sped references trasferimenti

## **Verifica della normalizzazione**

Dopo aver effettuato la verifica della normalizzazione non abbiamo ritenuto necessaria alcuna ulteriore modifica.

# **Operazioni previste dalla base di dati – Descrizione e relativo codice SQL**

## **Query di interrogazione**

1. Visualizzare tutti i prodotti posseduti dal cliente con codice fiscale ‘MRDMSS900P02P39F’.

SELECT prodotto.codice

FROM contratto, spazio\_contratto sp\_co, spazio, contiene, prodotto

WHERE cf\_cli = 'MRDMSS900P02P39F'

AND sp\_co.num\_c = contratto.num\_c

AND spazio.cod = sp\_co.cod

AND spazio.num = sp\_co.num

AND spazio.id\_spazio = sp\_co.id\_spazio

AND spazio.cod = contiene.cod

AND spazio.num = contiene.num

AND spazio.id\_spazio = contiene.id\_spazio

AND contiene.codice = prodotto.codice;

1. Visualizzare tutti i turni di tutti i custodi dell’anno in corso.

SELECT cu.cf, cu.nome, cu.cognome, tu.data\_t

FROM turno as tu, custode as cu

WHERE tu.cf = cu.cf

AND extract(year from tu.data\_t) = extract(year from current\_date)

1. Visualizzare tutte le spedizioni non ancora effettuate

SELECT num\_sped

FROM spedizione

WHERE stato\_consegna != 'Consegnato'

1. Visualizzare i contratti scaduti del cliente con codice fiscale ‘SSLVRL900P02P39F’

SELECT \*

FROM contratto

WHERE data\_fine < current\_date

AND cf\_cli = 'SSLVRL900P02P39F'

1. Visualizzare i magazzini e i relativi spazi del dirigente con codice fiscale ‘RSSCRL05P04D494F’

SELECT magazzino.cod, spazio.num as Magazzino, spazio.id\_spazio, descrizione

FROM magazzino, spazio

WHERE magazzino.cod = spazio.cod

AND magazzino.num = spazio.num

AND magazzino.cod IN (SELECT cod

FROM filiale

WHERE cf = 'RSSCRL05P04D494F')

ORDER BY magazzino.cod

1. Visualizzare tutti i prodotti presenti nel magazzino numero ‘1’ con codice della filiale ‘0000000003’

SELECT id\_spazio, codice, quantita

FROM contiene

WHERE num = '1'

AND cod = '0000000003'

1. Visualizzare la lista dei clienti ordinati in maniera decrescente per numero di spazi acquistati

SELECT cliente.cf\_cli , nome, cognome, num\_spazi

FROM cliente, contratto

WHERE cliente.cf\_cli = contratto.cf\_cli

ORDER BY num\_spazi DESC

## **Query di modifica**

1. Modificare la quantità del prodotto con codice ‘bsplgefebj’ che è contenuta nello spazio con ID ‘7’, numero del magazzino ‘2’ e codice della filiale ‘0000000002’ togliendo della quantità contenuta 500.

UPDATE contiene

SET quantita = contiene.quantita - 500

WHERE id\_spazio = ‘7’

AND num = ‘2’

AND cod = ‘0000000002’

AND contiene.codice = ‘bsplgefebj’;

1. Modificare il nome del piano del cliente con codice fiscale ‘SRDVRL900P02P39F’ con il nome piano ‘Gold’.

UPDATE cliente

SET nome\_piano = ‘gold’

WHERE cliente.cf\_cli = ‘SRDVRL900P02P39F’;

1. Modificare la quantità del prodotto con codice ‘yylltihwjl1’ nella spedizione con numero ‘4’ aggiungendo alla quantità già inserita nella spedizione 10 .

UPDATE prod\_sped

SET quantita = prod\_sped.quantita + 10

WHERE prod\_sped.num\_sped = ‘4’

AND prod\_sped.codice = ‘yylltihwjl1’;

## **Query di eliminazione**

1. Eliminazione di tutti gli spazi che non contengono alcun prodotto.

DELETE FROM contiene

WHERE quantita = 0;

1. Eliminazione del magazziniere con codice fiscale ‘TNOLCU910P03P39F’

DELETE FROM magazziniere

WHERE cf = ‘TNOLCU910P03P39F’

1. Eliminazione del veicolo con targa ‘CD456EF’

DELETE FROM veicolo

WHERE targa = ‘CD456EF’

# **Codice SQL per la creazione delle tabelle**

Creazione delle tabelle :

CREATE TABLE IF NOT EXISTS CUSTODE

(

cf varchar(16) not null,

nome varchar(255) not null,

cognome varchar (255) not null,

data\_nascita date,

tel varchar(30),

mail varchar(255),

PRIMARY KEY (cf)

);

CREATE TABLE IF NOT EXISTS FATTORINO

(

cf varchar(16) not null,

nome varchar(255) not null,

cognome varchar (255) not null,

data\_nascita date,

tel varchar(30),

mail varchar(255),

PRIMARY KEY (cf)

);

CREATE TABLE IF NOT EXISTS DIRIGENTE

(

cf varchar(16) not null,

nome varchar(255) not null,

cognome varchar (255) not null,

data\_nascita date,

tel varchar(30),

mail varchar(255),

PRIMARY KEY (cf)

);

CREATE TABLE IF NOT EXISTS PIANO

(

nome\_piano varchar(255) not null,

importo real not null,

descrizione varchar(255) not null,

PRIMARY KEY (nome\_piano),

check ( importo > 0 )

);

CREATE TABLE IF NOT EXISTS VEICOLO

(

targa varchar(20),

marca varchar(255),

modello varchar(255),

capacita int,

PRIMARY KEY (targa),

check(capacita > 0)

);

CREATE TABLE IF NOT EXISTS FILIALE

(

cod varchar(10) not null,

nome varchar(255) not null,

citta varchar(255) not null,

via varchar(255) not null,

numero int not null,

tel varchar(30),

cf varchar(16) not null,

PRIMARY KEY (cod),

FOREIGN KEY (cf) REFERENCES dirigente

);

CREATE TABLE IF NOT EXISTS MAGAZZINO

(

num int,

denominazione varchar(255),

citta varchar(255),

via varchar(255),

numero int,

tel varchar(30),

cod varchar(255),

PRIMARY KEY (num,cod),

FOREIGN key (cod) REFERENCES filiale

);

CREATE TABLE IF NOT EXISTS TURNO

(

cf varchar(16) not null,

data\_t date not null,

num int,

cod varchar(255),

PRIMARY KEY (cf,data\_t),

FOREIGN KEY (cf) REFERENCES custode,

FOREIGN KEY (num,cod) REFERENCES magazzino

);

CREATE TABLE IF NOT EXISTS SPAZIO

(

id\_spazio int,

num int not null,

cod varchar(255) not null,

descrizione varchar(255),

PRIMARY KEY (id\_spazio,num,cod),

FOREIGN KEY (num,cod) REFERENCES magazzino

);

CREATE TABLE IF NOT EXISTS CLIENTE

(

cf\_cli varchar(16) not null,

nome varchar(255) not null,

cognome varchar(255) not null,

tel varchar(30),

nome\_piano varchar(255) not null,

PRIMARY KEY (cf\_cli),

FOREIGN KEY (nome\_piano) REFERENCES piano

);

CREATE TABLE IF NOT EXISTS IMPIEGATO

(

cf varchar(16) not null,

nome varchar(255) not null,

cognome varchar (255) not null,

data\_nascita date,

tel varchar(30),

mail varchar(255),

cod varchar(255) not null,

PRIMARY KEY (cf),

FOREIGN KEY (cod) REFERENCES filiale

);

CREATE TABLE IF NOT EXISTS CONTRATTO

(

num\_c varchar(255) not null,

data\_inizio date not null,

data\_fine date,

num\_spazi int not null,

cf varchar(16) not null,

cf\_cli varchar(16) not null,

PRIMARY KEY (num\_c),

FOREIGN KEY (cf\_cli) REFERENCES cliente,

FOREIGN KEY (cf) REFERENCES impiegato,

check (data\_inizio >= current\_date),

check (data\_inizio < data\_fine),

check (num\_spazi > 0)

);

CREATE TABLE IF NOT EXISTS MAGAZZINIERE

(

cf varchar(16) not null,

nome varchar(255) not null,

cognome varchar (255) not null,

data\_nascita date,

tel varchar(30),

mail varchar(255),

num int not null,

cod varchar(255) not null,

PRIMARY KEY (cf),

FOREIGN KEY (num,cod) REFERENCES magazzino

);

CREATE TABLE IF NOT EXISTS PRODOTTO

(

codice varchar(255) not null,

nome\_prod varchar(255),

descrizione varchar(255),

PRIMARY KEY (codice)

);

CREATE TABLE IF NOT EXISTS CONTIENE

(

id\_spazio int not null,

num int not null,

cod varchar(255) not null,

codice varchar(255) not null,

quantita int not null,

PRIMARY KEY (id\_spazio, num, cod, codice),

FOREIGN KEY (id\_spazio, num, cod ) REFERENCES spazio,

FOREIGN KEY (codice ) REFERENCES prodotto,

check ( quantita >= 0 )

);

create sequence if not exists my\_seq1;

CREATE TABLE IF NOT EXISTS SPEDIZIONE

(

num\_sped int default nextval ('my\_seq1'::regclass) not null,

cf varchar(16) not null,

data\_sp date not null,

targa varchar(20) not null,

paese varchar(255) not null,

citta varchar(255) not null,

via varchar(255) not null,

numero varchar(10) not null,

tel varchar(30),

stato\_consegna varchar(11),

cf\_cli varchar(16) not null,

PRIMARY KEY (num\_sped),

FOREIGN KEY (cf ) REFERENCES fattorino,

FOREIGN KEY (targa) REFERENCES veicolo,

FOREIGN KEY (cf\_cli) REFERENCES cliente

);

CREATE TABLE IF NOT EXISTS SPAZIO\_CONTRATTO

(

id\_spazio int not null,

num int not null,

cod varchar(255),

num\_c varchar(255),

PRIMARY KEY (id\_spazio,num,cod,num\_c),

FOREIGN KEY (id\_spazio, num, cod) REFERENCES spazio,

FOREIGN KEY (num\_c) REFERENCES contratto

);

create sequence if not exists my\_seq2;

CREATE TABLE IF NOT EXISTS TRASFERIMENTI

(

num\_sped int default nextval ('my\_seq2'::regclass) not null,

data\_sp date not null,

cf varchar(16) not null,

targa varchar(20) not null,

stato\_consegna varchar(11),

num1 int not null,

cod1 varchar(255) not null,

num2 int not null,

cod2 varchar(255) not null,

PRIMARY KEY (num\_sped),

FOREIGN KEY (num1, cod1) REFERENCES magazzino(num,cod),

FOREIGN KEY (num2, cod2) REFERENCES magazzino(num,cod),

FOREIGN KEY (cf ) REFERENCES fattorino,

FOREIGN KEY (targa) REFERENCES veicolo

);

CREATE TABLE IF NOT EXISTS PROD\_SPED

(

codice varchar(255) not null,

num\_sped int not null,

quantita int not null,

PRIMARY KEY (codice, num\_sped),

FOREIGN KEY (codice) REFERENCES prodotto,

FOREIGN KEY (num\_sped) REFERENCES spedizione

);

CREATE TABLE IF NOT EXISTS PROD\_TRASF

(

codice varchar(255) not null,

num\_sped int not null,

quantita int not null,

PRIMARY KEY (codice, num\_sped),

FOREIGN KEY (codice) REFERENCES prodotto,

FOREIGN KEY (num\_sped) REFERENCES trasferimenti

);

# 

# **Codice SQL per l’inserimento dei dati**

Popolamento della tabella CUSTODE :

INSERT INTO custode VALUES ('SRFNDR98P02G393P','ANDrea','Serafini','1998-09-02','3394958490','andreaserafini1998@icloud.com'),('SRFNDR98P02G394P','Francesco','Rossi','2000-05-24','3394958840','francescorossi@icloud.com'),

('SRFNDR98P02G395P','Paolo','Magradze','1978-04-02','33456723277','paolomagradze@icloud.com'),

('PPLTCU05P04D494F','Carlo','Golfieri','1961-09-02','32845672945','carlogolfieri@icloud.com'),

('FRTOCU56P07D662F','Mario','Rossi','1981-09-02','2344059782','mariorossi@icloud.com');

Popolamento della tabella FATTORINO :

INSERT INTO fattorino VALUES ('FRCCRL98P02G394P','Carlo','Franceschini','2000-05-24','3394958840','carlofranceschini@icloud.com'),

('FRRGVN98P02G395P','Giovanni','Ferrari','19780402','33456723277','giovanniferrari@icloud.com'),

('BNCCRL05P04D494F','Carlo','Bianchi','1961-09-02','32845672945','carlobianchi@icloud.com'),

('VRDFDR56P07D662F','Federico','Verdi','1981-09-02','2344059782','federicoverdi@icloud.com'),

('GRBMRA56P07D662F','Mario','Garibaldi','1981-09-02','2344059782','mariorossi@icloud.com');

Popolamento della tabella DIRIGENTE :

INSERT INTO dirigente VALUES ('SRFCRL98P02G394P','Carlo','Serafini','2000-05-24','3394958840','carloserafini@mail.com'),

('MNTFRC98P02G395P','Francesco','Monti','1978-04-02','33456723277','francescomonti@mail.com'),

('RSSCRL05P04D494F','Carlo','Russo','1961-09-02','32845672945','carlorusso@mail.com'),

('SPSFDR56P07D662F','Federico','Esposito','1981-09-02','2344059782','federicoesposito@mail.com'),

('GLLMRA56P07D662F','Mario','Gallo','1981-09-02','2344059782','mariogallo@mail.com');

Popolamento della tabella VEICOLO :

INSERT INTO veicolo VALUES ('AB123CD','Fiat','Talento',50),

('CD456EF','Opel', 'Movano', 70),

('EF789GH','Mercedes','Sprinter',100);

Popolamento della tabella FILIALE :

INSERT INTO filiale VALUES ('0000000001','Trasporti Bologna','Bologna','Zamboni',8,'567887894','SRFCRL98P02G394P'),

('0000000002','Trasporti Roma','Roma','Verdi',8,'234887894','MNTFRC98P02G395P'),

('0000000003','Trasporti Firenze','Firenze','Garibaldi',6,'234845894','RSSCRL05P04D494F'),

('0000000004','Trasporti Napoli','Napoli','Diaz',6,'234845123','SPSFDR56P07D662F');

Popolamento della tabella MAGAZZINO :

SELECT INSERT\_magazzino('Magazzino 1 Bologna', 'Bologna', 'Rossi', 8, '1234567890', '0000000001');

SELECT INSERT\_magazzino('Magazzino 2 Bologna', 'Bologna', 'Verdi', 8, '1234567890', '0000000001');

SELECT INSERT\_magazzino('Magazzino 1 Firenze', 'Firenze', 'Verdi', 8, '1234567890', '0000000003');

SELECT INSERT\_magazzino('Magazzino 1 Roma', 'Roma', 'Rossi', 8, '1234567890', '0000000002');

SELECT INSERT\_magazzino('Magazzino 2 Roma', 'Roma', 'Verdi', 8, '1234567891', '0000000002');

SELECT INSERT\_magazzino('Magazzino 3 Roma', 'Roma', 'Garibaldi', 8, '1234567892', '0000000002');

SELECT INSERT\_magazzino('Magazzino 1 Napoli', 'Napoli', 'Verdi', 8, '1234567890', '0000000004');

Popolamento della tabella TURNO :

INSERT INTO turno VALUES ('SRFNDR98P02G393P', '2019-08-09',2,'0000000001'),

('SRFNDR98P02G393P', '2019-08-10',2,'0000000001'),

('SRFNDR98P02G393P', '2019-08-11',2,'0000000001'),

('SRFNDR98P02G395P', '2019-08-11',1,'0000000001'),

('SRFNDR98P02G395P', '2019-08-12',1,'0000000001'),

('PPLTCU05P04D494F', '2019-08-11',1,'0000000003'),

('PPLTCU05P04D494F', '2019-08-12',1,'0000000003'),

('FRTOCU56P07D662F', '2019-08-11',1,'0000000004');

Popolamento della tabella SPAZIO :

SELECT INSERT\_spazio(1,'0000000001','Magazzino 1 Bologna spazio 1');

SELECT INSERT\_spazio(1,'0000000001','Magazzino 1 Bologna spazio 2');

SELECT INSERT\_spazio(1,'0000000001','Magazzino 1 Bologna spazio 3');

SELECT INSERT\_spazio(1,'0000000001','Magazzino 1 Bologna spazio 4');

SELECT INSERT\_spazio(1,'0000000001','Magazzino 1 Bologna spazio 5');

SELECT INSERT\_spazio(2,'0000000001','Magazzino 2 Bologna spazio 1');

SELECT INSERT\_spazio(2,'0000000001','Magazzino 2 Bologna spazio 2');

SELECT INSERT\_spazio(2,'0000000001','Magazzino 2 Bologna spazio 3');

SELECT INSERT\_spazio(2,'0000000001','Magazzino 2 Bologna spazio 4');

SELECT INSERT\_spazio(2,'0000000001','Magazzino 2 Bologna spazio 5');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 1');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 2');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 3');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 4');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 5');

SELECT INSERT\_spazio(1,'0000000003','Magazzino 1 Firenze spazio 6');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 1 Roma spazio 1');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 1 Roma spazio 2');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 1 Roma spazio 3');

SELECT INSERT\_spazio(3,'0000000002','Magazzino 1 Roma spazio 4');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 2 Roma spazio 1');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 2 Roma spazio 2');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 2 Roma spazio 3');

SELECT INSERT\_spazio(2,'0000000002','Magazzino 2 Roma spazio 4');

SELECT INSERT\_spazio(3,'0000000002','Magazzino 3 Roma spazio 1');

SELECT INSERT\_spazio(3,'0000000002','Magazzino 3 Roma spazio 2');

SELECT INSERT\_spazio(3,'0000000002','Magazzino 3 Roma spazio 3');

SELECT INSERT\_spazio(3,'0000000002','Magazzino 3 Roma spazio 4');

SELECT INSERT\_spazio(1,'0000000004','Magazzino 1 Napoli spazio 1');

SELECT INSERT\_spazio(1,'0000000004','Magazzino 1 Napoli spazio 2');

SELECT INSERT\_spazio(1,'0000000004','Magazzino 1 Napoli spazio 3');

SELECT INSERT\_spazio(1,'0000000004','Magazzino 1 Napoli spazio 4');

Popolamento della tabella PIANO :

INSERT INTO piano VALUES ('Standard', 50, 'PianoStandard'),

('Gold', 40, 'PianoGold'),

('Top', 30, 'PianoTop');

Popolamento della tabella CLIENTE :

INSERT INTO cliente VALUES ('SRDVRL900P02P39F','Franco','Colombo','3457896543','Standard'),

('SFDVRL900P03P39F','Sandro','Rossi','3457896543','Gold'),

('SRDVRL900P32P39F','Giovanni','De Luca','3457896543','Gold'),

('SSLVRL900P02P39F','Mattia','Mancini','3457896543','Standard'),

('MRDMSS900P02P39F','Massimo','Meridio','3457896543','Top');

Popolamento della tabella IMPIEGATO :

INSERT INTO impiegato VALUES ('SFDVRL910P03P39F','Maria','Rossi','1980-09-01','05364102','mariarossi@mail.com','0000000001'),

('SFDVRL910P03P39G','Maria','Verdi','1980-09-01','05364102','mariaverdi@mail.com','0000000001'),

('SFDVRL910P03P39H','Carlo','Pedersoli','1980-09-01','05364102','carlopedersoli@mail.com','0000000002'),

('SFDVRL910P03P39I','Renato','Zero','1980-09-01','05364102','renatozero@mail.com','0000000003'),

('SFDVRL910P03P39K','Vasco','Rossi','1980-09-01','05364102','vascorossi@mail.com','0000000003'),

('SFDVRL910P03P39L','Fabrizio','De andrè','1980-09-01','05364102','faber@mail.com','0000000004'),

('SFDVRL910P03P39M','Giacomo','Leopardi','1980-09-01','05364102','siepe@mail.com','0000000004');

Popolamento della tabella CONTRATTO :

INSERT INTO contratto VALUES ('0000000001','2020-07-01','2021-07-01',4,'SFDVRL910P03P39F','SRDVRL900P02P39F'),

('0000000002','2020-07-01','2021-07-01',4,'SFDVRL910P03P39G','SFDVRL900P03P39F'),

('0000000003','2020-09-30','2021-05-17',4,'SFDVRL910P03P39G','SRDVRL900P32P39F'),

('0000000004','2020-02-21','2021-03-23',4,'SFDVRL910P03P39K','SSLVRL900P02P39F'),

('0000000005','2020-04-11','2021-09-25',4,'SFDVRL910P03P39K','MRDMSS900P02P39F');

Popolamento della tabella MAGAZZINIERE :

INSERT INTO magazziniere VALUES ('BLDMSM910P03P39F','Massimo','Boldi','1980-09-01','05364102','massimoboldi@mail.com',2,'0000000001'),

('TNOLCU910P03P39F','Luca','Toni','1980-09-01','05364102','lucatoni@mail.com',1,'0000000001'),

('RZZRCC910P03P39F','Riccardo','Rizzo','1980-09-01','05364102','lucatoni@mail.com',2,'0000000002'),

('DFRCRT910P03P39F','Cristina','De Francesco','1980-09-01','05364102','lucatoni@mail.com',2,'0000000002'),

('LMBFRR910P03P39F','Ferruccio','Lamborghini','1980-09-01','05364102','lucatoni@mail.com',2,'0000000002'),

('FRRNZ910P03P39F','Enzo','Ferrari','1980-09-01','05364102','lucatoni@mail.com',1,'0000000003'),

('TMBLBR910P03P39F','Alberto','Tomba','1980-09-01','05364102','lucatoni@mail.com',1,'0000000004');

Popolamento della tabella PRODOTTO :

INSERT INTO prodotto VALUES

('yylltihwjl1','adasio','descr adasio'),

('ruondpqytb','galore amos','galore amos'),

('lrqhnyodrb','swif t','descr swif t'),

('bsplgefebj','eqity amos','descr eqity'),

('fwehrmwxsknhbxi','bee','descr bee'),

('dtdgnsfxvzanhjo','motivation fim','descr fim'),

('cjhdaykzivjqzxl','mf','magic fim'),

('skibw','Cs','Classics fim'),

('fhd3b3p894fx83p','Hitch','hitch dry'),

('pro12345','pro','pro1 professional');

Popolamento della tabella CONTIENE :

INSERT INTO contiene VALUES

(3,1,'0000000001','yylltihwjl1',10),

(1,1,'0000000001','ruondpqytb',100),

(5,2,'0000000001','lrqhnyodrb',7),

(7,2,'0000000002','bsplgefebj',1000),

(7,2,'0000000002','fwehrmwxsknhbxi',90),

(2,3,'0000000002','dtdgnsfxvzanhjo',750),

(1,1,'0000000003','cjhdaykzivjqzxl',40),

(2,1,'0000000003','skibw',40),

(3,1,'0000000003','fhd3b3p894fx83p',1);

Popolamento della tabella SPEDIZIONE :

SELECT spedisci('FRRGVN98P02G395P','2019-01-01','CD456EF','Italia','Bologna','Via del Chionso','13','3965226884','SFDVRL900P03P39F',5,'yylltihwjl1','0000000001',1,3);

SELECT spedisci('FRRGVN98P02G395P','2019-01-01','CD456EF','Italia','Bologna','Via Arturo Bellalli','13','2469569851','MRDMSS900P02P39F',5,'ruondpqytb','0000000001',1,1);

SELECT spedisci('BNCCRL05P04D494F','2019-02-01','CD456EF','Italia','Bologna','Via Gaetano Salvemini','13','3698554246','SRDVRL900P32P39F',5,'bsplgefebj','0000000002',2,7);

SELECT spedisci('BNCCRL05P04D494F','2019-02-01','CD456EF','Italia','Bologna','Via Gaetano Salvemini','13','3698554246','SRDVRL900P32P39F',5,'bsplgefebj','0000000002',2,7);

Popolamento della tabella SPAZIO\_CONTRATTO :

INSERT INTO spazio\_contratto VALUES

(3,1,'0000000001','0000000002'),

(4,1,'0000000001','0000000002'),

(5,2,'0000000001','0000000003'),

(7,2,'0000000002','0000000003'),

(1,2,'0000000002','0000000003'),

(2,3,'0000000002','0000000003'),

(1,1,'0000000003','0000000002'),

(2,1,'0000000003','0000000004'),

(3,1,'0000000003','0000000005'),

(1,1,'0000000001','0000000005'),

(2,1,'0000000001','0000000005');

Popolamento della tabella TRASFERIMENTI :

SELECT trasferisci ('MRDMSS900P02P39F','2020-10-09', 'VRDFDR56P07D662F', 'EF789GH', 1,'0000000001',2,'0000000002','ruondpqytb',26);

# **Stored procedure, funzioni e trigger**

Funzione insert\_magazzino che crea un nuovo magazzino.

Input: denominazione, città, via, numero, telefono del magazzino da inserire e codice della filiale a cui apparterrà il magazzino.

CREATE OR REPLACE FUNCTION insert\_magazzino(denominazione char(255), citta char(255),via char(255),numero int ,tel varchar(30),cod\_f varchar(10))RETURNS int AS $$

DECLARE

num\_magazzino int;

BEGIN

//cerco il numero progressivo all'interno del codice della filiale $6

SELECT count (cod)+1 INTO num\_magazzino

FROM magazzino

WHERE cod = $6;

//registra un nuovo Magazzino

INSERT INTO magazzino(num, denominazione, citta, via, numero, tel,cod)VALUES(num\_magazzino, $1, $2, $3, $4, $5, $6);

return null;

END

$$ LANGUAGE plpgsql;

Funzione insert\_spazio, che crea un nuovo spazio.

Input: numero del magazzino, codice della filiale, descrizione dello spazio.

CREATE OR REPLACE FUNCTION insert\_spazio(numero\_magazzino int, codice\_filiale varchar(16), descr varchar(255))RETURNS int AS $$

DECLARE

num\_spazio int;

BEGIN

//cerco il numero progressivo all'interno del magazzino dato in input

SELECT count (cod)+1 INTO num\_spazio

FROM spazio

WHERE num=$1

AND cod=$2;

//registro un nuovo spazio

INSERT INTO spazio(id\_spazio,num,cod,descrizione)

VALUES(num\_spazio, $1, $2, $3);

return null;

END

$$ LANGUAGE plpgsql;

Funzione spedisci che gestisce una determinata spedizione.

Input : codice fiscale del cliente, data della spedizione, targa del veicolo usato, paese , città, via e numero di dove va spedito il prodotto, telefono, codice del cliente che richiede la spedizione, la quantità spedita, il codice del prodotto, il codice della filiale, il numero del magazzino e l'ID dello spazio

CREATE OR REPLACE FUNCTION spedisci(cf varchar(16), data\_sp date, targa varchar(20), paese varchar(255), citta varchar(255), via varchar(255), numero varchar(10), tel varchar(30), codice\_cli varchar(16), quantita int, codice\_prodotto varchar(255), fil varchar(10), mag int, spa int)

RETURNS int AS $$

DECLARE

codice\_fiscale varchar(16);

posseduto int;

contenuto int;

codice\_cliente alias for $9;

q alias for $10;

cont int;

num\_sp int;

BEGIN

//verifico se il cliente ha lo spazio indicato

SELECT cliente.cf\_cli INTO codice\_fiscale

FROM spazio\_contratto, contratto, cliente

WHERE spazio\_contratto.num\_c = contratto.num\_c

AND contratto.cf\_cli = cliente.cf\_cli

AND spazio\_contratto.id\_spazio = spa

AND spazio\_contratto.num = mag

AND spazio\_contratto.cod = fil;

IF codice\_fiscale != codice\_cli THEN

RAISE EXCEPTION 'Non corrisponde codice cliente con lo spazio';

END IF;

/\*inizializzo la quantità del prodotto x contenuta nello spazio posseduto dal cliente in input \*/

SELECT contiene.quantita INTO contenuto

FROM contiene

WHERE id\_spazio = spa

AND num = mag AND cod = fil

AND codice = codice\_prodotto;

/\* controllo se la quantità richiesta per la spedizione del prodotto x è sufficiente \*/

IF contenuto < quantita THEN

RAISE EXCEPTION 'La quantità non è sufficiente';

END IF;

//aggiorno la quantità che è contenuta nello spazio

UPDATE contiene

SET quantita = contiene.quantita - q

WHERE id\_spazio = spa

AND num = mag

AND cod = fil

AND contiene.codice = codice\_prodotto;

SELECT count (\*) INTO cont

FROM spedizione

WHERE spedizione.data\_sp = $2

AND cf\_cli = codice\_cliente

AND spedizione.targa = $3

AND spedizione.paese = $4

AND spedizione.citta = $5

AND spedizione.via = $6

AND spedizione.numero = $7;

IF cont != 1 THEN

//inserisco un nuovo record nella "spedizione"

INSERT INTO spedizione(cf, data\_sp, targa, paese, citta, via, numero, tel, stato\_consegna, cf\_cli) VALUES(cf, data\_sp, targa, paese, citta, via, numero, tel, 'In consegna', codice\_cliente);

END IF;

//cerco qual è il numero spedizione

SELECT num\_sped INTO num\_sp

FROM spedizione

WHERE spedizione.data\_sp = $2

AND cf\_cli = codice\_cliente

AND spedizione.targa = $3

AND spedizione.paese = $4

AND spedizione.citta = $5

AND spedizione.via = $6

AND spedizione.numero = $7;

/\*controllo se esiste già la spedizione con lo stesso codice del prodotto, codice fiscale, data e numero spedizione \*/

SELECT count (\*) INTO cont

FROM prod\_sped

WHERE prod\_sped.num\_sped = num\_sp

AND prod\_sped.codice = codice\_prodotto;

IF cont = 1 THEN

UPDATE prod\_sped

SET quantita = prod\_sped.quantita + q

WHERE prod\_sped.num\_sped = num\_sp

AND prod\_sped.codice = codice\_prodotto;

ELSE

//inserisco un nuovo record nella "prod\_sped"

INSERT INTO prod\_sped(codice, num\_sped, quantita)

VALUES (codice\_prodotto, num\_sp, q);

END IF;

return null;

END

$$ LANGUAGE plpgsql;

Funzione assegna\_spazio che assegna lo spazio dato in input al contratto dato in input

Input: codice della filiale, numero del magazzino, ID dello spazio e numero del contratto

CREATE OR REPLACE FUNCTION assegna\_spazio(fil varchar(10), mag int, spa int, contr varchar(255)) RETURNS int AS $$

DECLARE

q int;

acq int;

cont int;

BEGIN

//verifico se lo spazio è libero

SELECT count (\*) INTO cont

FROM spazio\_contratto

WHERE cod = fil

AND mag = num

AND spa = id\_spazio;

IF cont > 0 THEN

RAISE EXCEPTION 'lo spazio non è libero';

END IF;

//conto quanti spazi sono già assegnati al contratto

SELECT count (\*) INTO q

FROM spazio\_contratto, contratto, spazio

WHERE contratto.num\_c = contr

AND contratto.num\_c = spazio\_contratto.num\_c

AND spazio\_contratto.id\_spazio = spazio.id\_spazio

AND spazio\_contratto.cod = spazio.cod

AND spazio.num = spazio\_contratto.num;

//conto quanti spazi sono stati acquistati nel contratto

SELECT sum (num\_spazi) INTO acq

FROM contratto

WHERE num\_c = contr;

IF acq - q < 1 THEN

RAISE EXCEPTION 'il contratto ha esaurito gli spazi';

END IF;

INSERT INTO spazio\_contratto VALUES(spa,mag,fil,contr);

return null;

END

$$ LANGUAGE plpgsql;

Funzione calcola\_piano che calcola il piano tariffario del cliente con il codice fiscale preso in input.

Input : codice fiscale del cliente

CREATE OR REPLACE FUNCTION calcola\_piano(cf varchar(16)) RETURNS varchar(255) AS $$

DECLARE

gg int;

rec record;

roof int;

BEGIN

roof =0;

FOR rec IN (SELECT data\_fine, data\_inizio, num\_spazi

FROM contratto

WHERE cf\_cli = $1)

LOOP

SELECT (DATE\_PART('year', rec.data\_fine::date) - DATE\_PART('year', rec.data\_inizio::date)) \* 12 +

(DATE\_PART('month', rec.data\_fine::date) - DATE\_PART('month', rec.data\_inizio::date)) INTO gg;

roof = roof + gg \* rec.num\_spazi;

END LOOP;

IF roof < 100 THEN

return 'StANDard';

END IF;

IF roof < 500 THEN

return 'Gold';

ELSE

return 'Top';

END IF;

return null;

END

$$ LANGUAGE plpgsql;

Funzione che determina se un cliente esiste già all’interno del database in base al codice fiscale. La funzione restituisce vero se il cliente è già nel database o falso altrimenti.

Input : codice fiscale del cliente.

CREATE OR REPLACE FUNCTION esiste\_cliente(cf text)RETURNS bool AS $func$

DECLARE

\_text;

BEGIN

SELECT cf\_cli INTO \_

FROM cliente

WHERE cf\_cli = $1;

IF found THEN

return true;

END IF;

return false;

END

$func$ LANGUAGE plpgsql;

Funzione nuovo\_contratto che crea un nuovo contratto.

Input : numero del contratto, data di inizio del contratto, data di fine del contratto, codice fiscale del cliente e codice fiscale dell'impiegato che registra il contratto

CREATE OR REPLACE FUNCTION nuovo\_contratto(num\_c varchar(255), datai date, dataf date, num\_spazi int, cf varchar(255), cf\_cli varchar(255))

RETURNS int AS $$

DECLARE

piano varchar(255);

BEGIN

IF(esiste\_cliente($6)) THEN

INSERT INTO contratto VALUES ($1, $2, $3, $4, $5, $6);

ELSE

RAISE EXCEPTION 'Il codice fiscale inserito non esiste nel DB dei clienti';

END IF;

SELECT calcola\_piano(cf\_cli) INTO piano;

UPDATE cliente

SET nome\_piano = piano

WHERE cliente.cf\_cli = $6;

return null;

END

$$ LANGUAGE plpgsql;

Funzione crea\_cliente che crea un nuovo cliente.

Input : codice fiscale, nome, cognome e telefono del cliente da inserire.

CREATE OR REPLACE FUNCTION crea\_cliente(cf varchar(16), nome varchar(255), cognome varchar(255), tel varchar(30))RETURNS int AS $$

DECLARE

cf1 varchar(16);

BEGIN

//cerco se esiste già il cliente

SELECT cf\_cli INTO cf1

FROM cliente

WHERE cliente.cf\_cli = $1;

IF cf1 = $1 THEN

RAISE EXCEPTION 'Il cliente esiste già';

END IF;

//registro il nuovo cliente

INSERT INTO cliente VALUES($1, $2, $3, $4, 'Standard');

return null;

END

$$ LANGUAGE plpgsql;

Funzione trasferisci che gestisce un determinato trasferimento.

Input : codice fiscale del cliente, data della spedizione, codice fiscale del fattorino che effettua il trasferimento, targa del veicolo utilizzato, numero del magazzino di partenza, codice della filiale di partenza, numero del magazzino di arrivo, codice della filiale di arrivo, codice del prodotto spedito e quantità spedita.

CREATE OR REPLACE FUNCTION trasferisci(cf varchar(16), data\_spedizione date, fattorino varchar(16), targa varchar(20), n1 int, c1 varchar(255),n2 int, c2 varchar(255),codice\_prodotto varchar(255), q int) RETURNS int AS $$

DECLARE

numero\_spedizione int;

ordine int;

cont int; //quantità contenuta in uno spazio

manda int;

indir int;

BEGIN

//controllo la data

IF data\_spedizione < current\_date THEN

RAISE EXCEPTION 'La data non valida';

END IF;

//controllo che il prodotto non sia mandato nello stesso magazzino

IF c1 = c2 and n1 = n2 THEN

RAISE EXCEPTION 'Non si può mandare nello stesso magazzino';

END IF;

ordine = q;

manda =0;

//seleziono il massimo della quantità che possiede un cliente

SELECT co.quantita into cont

FROM prodotto pr, contiene co, spazio sp, spazio\_contratto spc, contratto contr

WHERE pr.codice = co.codice

AND sp.cod = co.cod

AND sp.num = co.num

AND sp.id\_spazio = co.id\_spazio

AND co.codice = $9

AND sp.cod = spc.cod

AND sp.num = spc.num

AND sp.id\_spazio = spc.id\_spazio

AND spc.num\_c = contr.num\_c

AND contr.cf\_cli = $1

AND sp.num = n1

AND sp.cod = c1

AND co.quantita > 0

ORDER BY co.quantita desc

LIMIT 1;

IF NOT FOUND THEN

RAISE EXCEPTION 'Il prodotto % non trovato.', $9;

END IF;

IF cont < 1 THEN

RAISE EXCEPTION 'Quantità non sufficiente nel magazzino';

ELSE

LOOP EXIT WHEN ordine <= 0 ;

SELECT co.quantita INTO cont

FROM prodotto pr, contiene co, spazio sp, spazio\_contratto spc, contratto contr

WHERE pr.codice = co.codice

AND sp.cod = co.cod

AND sp.num = co.num

AND sp.id\_spazio = co.id\_spazio

AND co.codice = $9

AND sp.cod = spc.cod

AND sp.num = spc.num

AND sp.id\_spazio = spc.id\_spazio

AND spc.num\_c = contr.num\_c

AND contr.cf\_cli = $1

AND sp.num = n1

AND sp.cod = c1

AND co.quantita > 0

ORDER BY co.quantita desc

LIMIT 1;

IF NOT FOUND THEN

exit;

END IF;

SELECT sp.id\_spazio INTO indir

FROM prodotto pr, contiene co, spazio sp, spazio\_contratto spc, contratto contr

WHERE pr.codice = co.codice

AND sp.cod = co.cod

AND sp.num = co.num

AND sp.id\_spazio = co.id\_spazio

AND co.codice = $9

AND sp.cod = spc.cod

AND sp.num = spc.num

AND sp.id\_spazio = spc.id\_spazio

AND spc.num\_c = contr.num\_c

AND contr.cf\_cli = $1

AND sp.num = n1

AND sp.cod = c1

AND co.quantita > 0

ORDER BY co.quantita desc

LIMIT 1;

IF ordine >= cont THEN

UPDATE contiene

SET quantita = quantita - cont

WHERE cod = c1

AND num = n1

AND id\_spazio = indir;

manda = manda + cont;

ordine = ordine - cont;

ELSE

UPDATE contiene

SET quantita = quantita - ordine

WHERE cod = c1

AND num = n1

AND id\_spazio = indir;

manda = manda + ordine;

ordine =0;

END IF;

END LOOP;

END IF;

//inserisco un nuovo trasferimento

SELECT num\_sped INTO numero\_spedizione

FROM trasferimenti

WHERE data\_sp = data\_spedizione

AND trasferimenti.cf = fattorino

AND stato\_consegna = 'In consegna'

AND num1 = n1

AND num2 = n2

AND c1 = cod1

AND c2 = cod2;

IF NOT FOUND THEN

INSERT INTO trasferimenti(data\_sp, cf, targa, stato\_consegna, num1, cod1, num2, cod2) VALUES (data\_spedizione, fattorino, $4,'In consegna',n1,c1,n2,c2);

INSERT INTO prod\_trasf(codice,num\_sped,quantita)

VALUES (codice\_prodotto,(SELECT num\_sped

FROM trasferimenti

ORDER BY num\_sped desc

LIMIT 1),manda);

ELSE

UPDATE prod\_trasf

SET quantita = quantita + manda

WHERE num\_sped = numero\_spedizione;

END IF;

return null ;

END

$$ LANGUAGE plpgsql;

Funzione elimina\_contiene che controlla i dati in input ed elimina il collegamento spazio-prodotto.

Input : ID dello spazio, numero del magazzino, codice della filiale, codice del prodotto e quantità.

CREATE OR REPLACE FUNCTION elimina\_contiene(spa int, mag int, fil varchar(10),codice\_prodotto varchar(255), q int)RETURNS int AS $$

DECLARE

quant int;

BEGIN

SELECT quantita INTO quant

FROM contiene

WHERE contiene.id\_spazio = spa

AND contiene.num = mag

AND contiene.cod = fil

AND contiene.codice = codice\_prodotto;

IF not found THEN

RAISE EXCEPTION 'Hai inserito i dati sbagliati.';

END IF;

IF quant < q THEN

RAISE EXCEPTION 'Quantità non sufficiente.';

ELSE

UPDATE contiene

SET quantita = quantita - q

WHERE contiene.id\_spazio = spa

AND contiene.num = mag

AND contiene.cod = fil

AND contiene.codice = codice\_prodotto;

END IF;

return null;

END

$$ LANGUAGE plpgsql;

Funzione who\_is che dato un username restituisce che ruolo ha nel database.

Input : username.

CREATE OR REPLACE FUNCTION who\_is(un varchar(255))RETURNS text AS $$

DECLARE

cat text;

usr text;

BEGIN

//admin

IF (SELECT usesuper FROM pg\_user WHERE usename = un) THEN

cat = 'admin';

return cat;

END IF;

//dirigente

SELECT cf INTO cat

FROM dirigente

WHERE cf = un;

IF found THEN

cat = 'dirigente';

return cat;

END IF;

//custode

SELECT cf INTO cat

FROM custode

WHERE cf = un;

IF found THEN

cat = 'custode';

return cat;

END IF;

//impiegato

SELECT cf INTO cat

FROM impiegato

WHERE cf = un;

IF found THEN

cat = 'impiegato';

return cat;

END IF;

//fattorino

SELECT cf INTO cat

FROM fattorino

WHERE cf = un;

IF found THEN

cat = 'fattorino';

return cat;

END IF;

//cliente

SELECT cf\_cli INTO cat

FROM cliente

WHERE cf\_cli = un;

IF found THEN

cat = 'cliente';

return cat;

END IF;

//magazziniere

SELECT cf INTO cat

FROM magazziniere

WHERE cf = un;

IF found THEN

cat = 'magazziniere';

return cat;

END IF;

return 'nan';

END

$$ LANGUAGE plpgsql;

La procedura inserisce o aggiorna il prodotto nella tabella spazio\_contratto.

Input: id\_spazio, magazzino, filiale, codice prodotto, quantità

NOTA: per chiamare la procedura usa CALL <nome()>

CREATE OR REPLACE PROCEDURE insert\_contiene(int, int, varchar(10), varchar(255), int)LANGUAGE plpgsql AS $$

DECLARE

\_text;

BEGIN

SELECT codice INTO \_

FROM contiene

WHERE contiene.id\_spazio = $1

AND contiene.num = $2

AND contiene.cod = $3

AND contiene.codice = $4;

//Se non c’è questo tipo di prodotto in questo spazio

IF not found THEN

//inserisco

INSERT INTO contiene VALUES($1,$2,$3,$4,$5);

COMMIT;

return;

END IF;

//Se il prodotto è già presente allora aggiorno la quantità

UPDATE contiene

SET quantita = quantita + $5

WHERE contiene.id\_spazio = $1

AND contiene.num = $2

AND contiene.cod = $3

AND contiene.codice = $4;

COMMIT;

END;

$$;

Trigger che elimina il record dalla contiene dove trova la quantità uguale a zero.

CREATE OR REPLACE FUNCTION contiene\_zero()RETURNS trigger AS $$

BEGIN

DELETE FROM contiene

WHERE quantita = 0;

RETURN NEW;

END;

$$ LANGUAGE 'plpgsql';

CREATE TRIGGER contiene\_zero\_trig

AFTER UPDATE

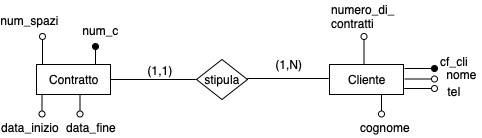
ON contiene

FOR EACH ROW

EXECUTE PROCEDURE contiene\_zero();

# 

# **Studio di dato derivato**



Il dato derivato preso in questione è l’attributo numero\_di\_contratti dell’entità cliente.

Operazione 1 : lettura dati cliente (numero di contratti, codice fiscale..)

Operazione 2 : inserimento di un nuovo contratto

|  |  |  |
| --- | --- | --- |
| Concetto | Tipo | Volume |
| Contratto | E | 15000 |
| Cliente | E | 5000 |

|  |  |  |
| --- | --- | --- |
| Operazione | Tipo | Frequenza |
| 1 | I | 50/G |
| 2 | I | 200/G |

Con il dato derivato :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Concetto | Accesso | Tipo |
| Operazione 1  1 accesso in letttura  1 \* 50/G = 50/G | Cliente | 1 | L |
| Operazione 2  3 accessi in lettura  1 accesso in scrittura  5 \* 200/G = 1000/G | Contratto  Stipula  Cliente  Cliente | 1  1  1  1 | L  L  L  S |
| Totale : 1050/G |  |  |  |

Senza il dato derivato :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Concetto | Accesso | Tipo |
| Operazione 1  7 accessi in lettura  7 \* 50/G = 350/G | Cliente  Stipula  Contratto | 1  3 (15000/5000)  3 (15000/5000) | L  L  L |
| Operazione 2  1 accesso in scrittura  2 \* 200/G = 400/G | Contratto | 1 | S |
| Totale : 750/G |  |  |  |

La soluzione più conveniente è quella senza il dato derivato, perciò abbiamo deciso di non tenere il dato derivato.

# 

# **Progetto fisico**

Si suppone di voler stimare i costi della ricerca dei contatti scaduti del cliente con codice fiscale 'SSLVRL900P02P39F ' .

Dati :

Numero Tuple NT = 10.000

Numero Blocchi NB = 1.000

Query :

SELECT \*

FROM contratto

WHERE data\_fine < current\_date

AND cf\_cli = 'SSLVRL900P02P39F'

Indici in esame :

1. Data\_fine, unclustered : NKdata\_fine = 150 NFdata\_fine = 100
2. Cf\_cli, unclustered : NKcf\_cli = 500 NFcf\_cli = 200

Fattori di selettività :

Fdata\_fine = 1/3

Fcf\_cli = 1/NKcf\_cli = 1/500

Costi :

Cseq = NB = 1.000

Cdata\_fine = ⎡ Fdata\_fine \* NFdata\_fine ⎤ + ⎡ Fdata\_fine \* NT⎤ =⎡ 1/3 \* 100 ⎤ + ⎡ 1/3 \* 10.000 ⎤ = 3368

Ccf\_cli = ⎡ Fcf\_cli \* NFcf\_cli ⎤ + ⎡ Fcf\_cli \* NT ⎤ = ⎡ 1/500 \* 200 ⎤ + ⎡ 1/500 \* 10.000 ⎤ = 21

E = ⎡ ∏i Fi \* NT ⎤ = ⎡ Fdata\_fine \* Fcf\_cli \* NT ⎤ = ⎡ 1/3 \* 1/500 \* 10.000 ⎤ = 7

Conclusione :

Per velocizzare la seguente query l’indice più conveniente da costruire è quello sul codice fiscale del cliente.