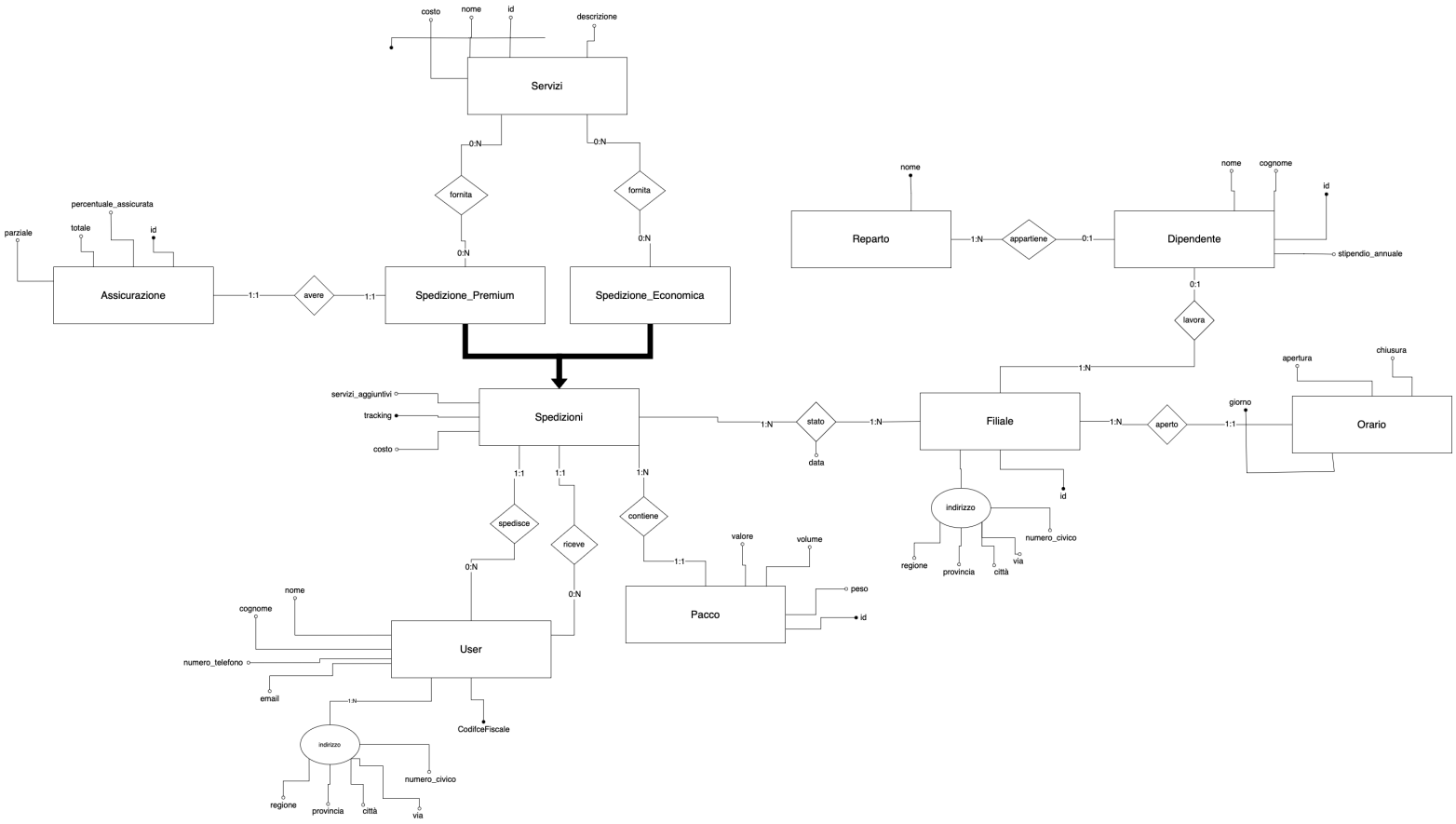
Speedizioni

1. Descrizione introduttiva

Il progetto mira alla creazione di un database completo e funzionale per gestire un sistema di spedizioni. Il database è stato progettato per archiviare informazioni relative alle spedizioni, alle filiali, ai dipendenti, agli utenti e ai pacchi coinvolti nel processo di spedizione.
L'obiettivo principale del progetto è fornire un'infrastruttura solida per la gestione efficiente delle spedizioni dell'app Speedizioni, consentendo di tracciare e monitorare ogni fase del processo.
La progettazione del database comprenderà la creazione di tabelle per rappresentare le diverse entità coinvolte nel sistema, in particolare alcune delle entità cardine del progetto saranno: le spedizioni, gli utenti che utilizzeranno il servizio, le varie assicurazioni proposte e i servizi aggiuntivi delle spedizioni.
Saranno definiti i vincoli di integrità per garantire la coerenza dei dati e saranno sviluppate relazioni tra le tabelle per consentire l'accesso e il recupero efficiente delle informazioni.
Inoltre, il progetto prevede la creazione di alcune query per l'analisi dei dati e la generazione di report.
Saranno implementate query per calcolare statistiche sulle spedizioni, come il costo medio delle spedizioni, verranno inoltre fornite delle query che andranno a fornire informazioni riguardanti il personale ma anche degli utenti e delle filiali. Queste query consentiranno di ottenere informazioni utili per valutare le prestazioni del sistema e prendere decisioni strategiche.
Il progetto prevede inoltre l'utilizzo di trigger nel database per automatizzare determinate azioni o controlli di integrità dei dati. I trigger saranno implementati per rispondere a determinati eventi o condizioni specifiche e potranno essere configurati per eseguire azioni come l'aggiornamento di dati correlati o il controllo della coerenza dei dati inseriti.
Ad esempio, verrà definito un trigger per verificare che il costo della spedizione includa i vari servizi e in caso della spedizione premium, includa la tariffa aggiuntiva imposta dalla percentuale di assicurazione, la quale può essere totale o parziale. In caso di violazione delle regole, il trigger intercetterà l'azione e genererà un avviso impedendo l'inserimento stesso.
Inoltre, vi sarà l'utilizzo dei trigger per mantenere la coerenza tra i valori delle tabelle, ad esempio, aggiornando automaticamente i dati correlati quando si verifica una modifica o una cancellazione nei servizi di una spedizione.
L'utilizzo dei trigger nel progetto contribuirà a migliorare l'accuratezza delle informazioni registrate e automatizzare alcune operazioni ripetitive o complesse. Saranno un componente fondamentale per garantire l'affidabilità e l'efficienza del sistema di gestione delle spedizioni, soprattutto per ciò che riguarda il loro costo.
Vengono inoltre forniti dei check riguardanti gli inserimenti del codice fiscale il quale deve essere costruito come uno reale, un altro check fondamentale è dato dalla possibilità di poter inserire un solo vero e un solo falso all'interno di una stessa assicurazione, quindi se è parziale, non potrà anche essere totale. Altre informazioni riguardanti i check e i trigger.
Nel complesso, il progetto si propone di fornire un database robusto e completo seppur apparentemente semplice per la gestione delle spedizioni, offrendo funzionalità di registrazione, tracciamento e analisi dei dati. Sarà uno strumento essenziale per ottimizzare l'efficienza delle operazioni di spedizione e migliorare l'esperienza complessiva per gli utenti di Speedizioni.

2. Schema entità relazione



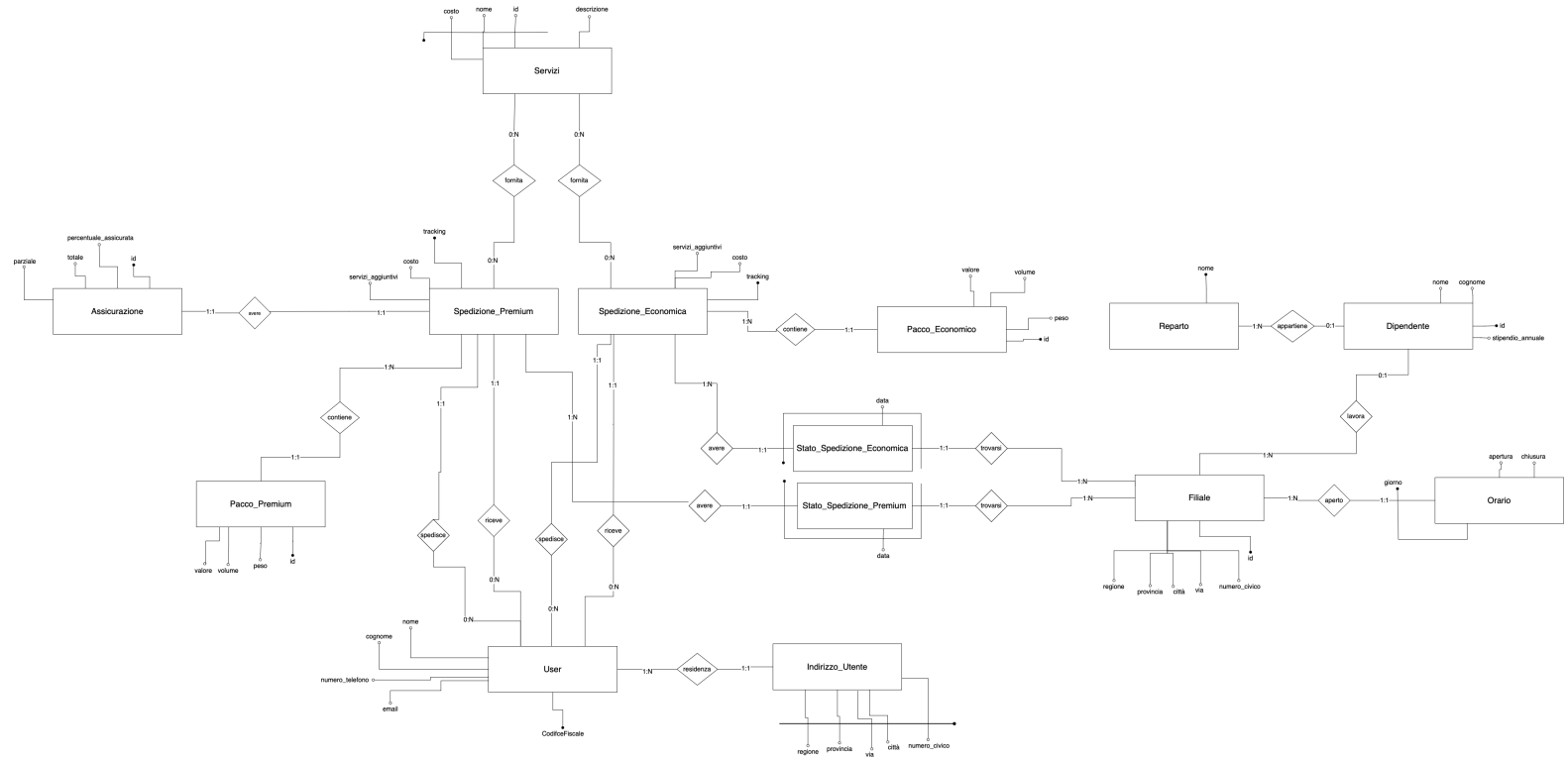
2.1 Relazioni

Relazione	Descrizione della Cardinalità
Servizi - *0:N* - fornita - *0:N* - Spedizione_Premium Servizi - *0:N* - fornita - *0:N* - Spedizione_Economica	una spedizione premium o economica può avere più di un servizio e un servizio può essere associato a più spedizioni.
Spedizione_Premium - *1:1*- ha - *1:1* - Assicurazione	Una spedizione premium può avere una sola assicurazione mentre un'assicurazione può essere associata a una sola spedizione.
Dipendente - *0:1* - appartiene - *0:1* - Reparto	Un dipendete viene appunto collocato in un reparto nel quale svolgerà il lavoro per il quale è stato assunto.
Dipendente - *1:1* - lavora - *1:N* - Filiale	Un dipendente può lavorare in un'unica filiale, se questo venisse trasferito in una filiale diversa, sarà necessario cambiare la filiale ad esso associata. Non ho ritenuto necessario avere uno storico delle varie filiali nelle quali un dipendente ha lavorato.
Spedizioni - *1:1* - invia - *0:N* - User Spedizioni - *1:1* - riceve - *0:N* - User	Una spedizione ha 2 user (mittente e destinatario), in questo caso la relazione viene rappresentata come riceve e invia. Uno user può non aver ancora effettuato una spedizione ed essersi solo iscritto al servizio, oppure può aver effettuato molte spedizioni.
Filiale - *1:N* - aperto - *1:1* - Orario	Una filiale ha più orari, uno per ogni giorno, per questo motivo la relazione è uno a molti.
Spedizione *0:N* - stato - *0:N* Filiale	La relazione tra Spedizione e Filiale rappresenta appunto lo stato di una spedizione, la quale prima di essere completata potrà passare per più fililali.

2.2 Generalizzazione

Generalizzazione	Risoluzione	Motivazione
Spedizione	figlie si dividono in Spedizione_Premium e Spedizione_Economica	Ho optato per questa scelta per due motivi, il primo è perchè Spedizione_Premium ha anche un'assicurazione, cosa che Spedizione_Economica non ha, quindi sarebbe stato superfluo aggiungere la relazione con Assicurazione a tutte le spedizioni. Inoltre il concetto delle due tipologie di spedizione è diverso e ritenevo più adeguato e chiaro, separare la generalizzazione ed avere le due tipologie di spedizione in due tabelle completamente indipendenti.

3. Schema entità relazione ristrutturato



3.1 Dettagli concettuali dello schema ristrutturato

La scelta di separare la generalizzazione nelle due entità figlie è data dalla volontà di avere una separazione concettuale che poi sarà anche logica di esse. Da questa scelta deriva anche la scelta di separare lo stato delle spedizioni e la tipologia del pacco, così da poter avere due database paralleli per gestire due entità cardine concettualmente separate, che però trovano alcuni punti di contatto come possono essere i servizi offerti ad entrambe che sono gli stessi, o i clienti (User) che possono prenotare una qualsiasi spedizione senza causare ridondanze che causerebbero anomalie di aggiornamento.

3.2 Descrizione delle entità dopo ristrutturazione

In seguito si trovano le descrizioni delle entità le quali vengono aggiunte dal processo di ristrutturazione dello schema e-r.

Entità	Descrizione	Caratteristiche
Assicurazione	Assicurazione delle spedizioni premium, avendo due flag (parziale, totale) può variare tra 0-30 e 50-100	Id: PK Tracking: UNIQUE Tracking: FK -> Spedizione_Premium.tracking
Dipendente	Descrive un dipendente attraverso il suo stipendio, nome, cognome e il reparto nel quale lavora nota: lo stipendio non dipende dal reparto	Id: PK Filiale: FK -> Filiale.id Reparto: FK -> Reparto.nome
Filiale	Costituisce un punto di transito della spedizione prima di arrivare al destinatario	Id: PK Regione, città, via, provincia, numero_civico: UNIQUE
Indirizzo_Utente	Contiene gli indirizzi di tutti gli utenti	Regione, città, provincia, via, numero_civico: PK User: FK -> User.codice_fiscale
Orario	Orari di apertura per ciascuna filiale, quando apertura e chiusura sono *null* allora la filiale è chiusa quel giorno	giorno, filiale: pk Filiale: FK -> Filiale.id
Pacco Economico Pacco Premium	Si tratta del contenuto della spedizione	id: PK Spedizione: FK -> Spedizione.tracking
Reparto	Contiene i vari reparti presenti nelle filiali, al momento ce ne sono solamente tre, ma in futuro se potrebbero aggiungere molteplici	nome: PK

Entità	Descrizione	Caratteristiche
Servizi	Permette di creare dei servizi che possono essere aggiunti alle spedizioni. La decisione di inserire anche nome e costo all'interno della PK è dovuta alla volontà di poter inserire entrambi anche nella tabella di congiunzione con Spedizione	Id, nome, costo: PK
Spedizione_Economica	Contiene le varie spedizioni, è il padre della generalizzazione tra spedizione premium ed economica, è l'entità che si interfaccia con la quasi totalità delle altre	Tracking: PK Mittente, Destinatario: FK -> User.codice_fiscale
Spedizione_Premium	La differenza con Spedizione_Economica è che Spedizione_Premium oltre a rappresentare un concetto diverso, aggiunge anche la possibilità di avere un'assicurazione. Troviamo infatti in Assicurazione, il tracking di Spedizione_Premium come chiave esterna	Tracking: PK Mittente, Destinatario: FK -> User.codice_fiscale
Spedizione_Economica_Servizi Spedizione_Premium_Servizi	Entità di congiunzione che risolve la molti a molti tra Spedizione e Servizi, nella quale troveremo le Spedizioni e i loro servizi associati. Nello schema ER non è rappresentata, ma è presente nel database, per questo motivo ho deciso di inserirla in questo schema riassuntivo.	tracking, servizio, nome_servizio, costo: PK Tracking : FK -> Spedizione_Economica.tracking Spedizione_Premium.tracking servizio, nome_servizio, costo: FK -> Servizi.id, Servizi.nome, Servizi.costo
Stato_Spedizione_Economica Stato_Spedizione_Premium	Tabella di congiunzione che risolve la molti a molti tra Spedizione e Filiale, contiene quindi lo stato di ogni spedizione, quando e dove essa si trova	Filiale, data, tracking: PK Filiale: FK -> Filiale.id Tracking : FK -> Spedizione_Economica.tracking Spedizione_Premium.tracking
User	Clienti del servizio Speedizioni, possono essere mittenti o destinatari, ma non entrambi nello stesso momento	CodiceFiscale: PK

3.3 Descrizione delle relazioni dopo ristrutturazione

Relazione	Descrizione della Cardinalità
Spedizione_Premium *1:N - 1:1* Stato_Spedizione_Premium Spedizione_Economica *1:N - 1:1* Stato_Spedizione_Economica	La spedizione può avere più stati, cioè trovarsi in filiali diverse in momenti diversi.
Stato_Spedizione_Premium *1:1 - 1:N* Filiale Stato_Spedizione_Economica *1:1 - 1:N* Filiale	Una filiale potrà avere in carico più spedizioni, dovrà quindi descrivere lo stato della spedizione.
User *1:N - 1:1* Indirizzo_Utente	Uno user avrà la possibilità di registrare molteplici indirizzi
Indirizzo_Utente *1:1 - 1:N* User	Un indirizzo è associato solamente ad uno user

3.4 Vincoli non rappresentabili tramite schema E-R

L'ER schema rappresenta solo una parte dei vincoli del database. Altri vincoli verranno implementati nel database per garantire la correttezza dei dati, attraverso check e trigger.
Gli index verranno utilizzati per migliorare le prestazioni delle operazioni di ricerca e accesso ai dati. Queste informazioni non sono visibili direttamente nell'ER schema, ma sono essenziali per garantire l'integrità e l'efficienza del database.

4. Schema logico

Assicurazione (id, percentuale_assicurata, totale, parziale, tracking -> Spedizione_Premium.tracking)
Dipendente (id, nome, cognome, stipendio_annuale, filiale -> Filiale.id)
Filiale (id, regione, città, via, provincia, numero_civico)
Indirizzo_Utente (regione, città, via, provincia, numero_civico, User -> User.codice_fiscale)
Orario (filiali -> Filiale.id, giorno, chiusura, apertura)
Pacco_Economico (id, valore, volume, peso, spedizione -> Spedizione_Economica.tracking)
Pacco_Premium (id, valore, volume, peso, spedizione -> Spedizione_Premium.tracking)
Reparto (nome)
Servizi (id, nome, costo, descrizione)
Spedizione_Economica (tracking, mittente -> User.codice_fiscale, destinatario -> User.codice_fiscale, costo, servizi_aggiuntivi)
Spedizione_Premium (tracking, mittente -> User.codice_fiscale, destinatario -> User.codice_fiscale, costo, servizi_aggiuntivi)
Stato_Spedizione_Economica (tracking -> Spedizione_Economica.tracking, filiale -> Filiale.id, data)
Stato_Spedizione_Premium (tracking -> Spedizione_Economica.tracking, filiale -> Filiale.id, data)
User (codice_fiscale, email, nome, cognome, numero_telefono)

5. Index, Check, Unique e Trigger

5.1 Index

Stato_Spedizione_Premium

```
CREATE INDEX "Stato_Spedizione_Premium_data_index"
ON public."Stato_Spedizione_Premium" USING btree (data);
```

Ho optato per un indice, di tipo btree, sull'attributo "data", in modo tale che la ricerca, ad esempio di tutte le spedizione che hanno uno stato di rilevazione con data "maggiore" di un'altra, venga velocizzato. Nella figura sottostante, si può notare un benchmark esemplificativo. Ho optato per la creazione di questo indice perchè la ricerca delle spedizioni in base alla loro presenza in data è un'operazione che verrebbe effettuata migliaia di volte al giorno.

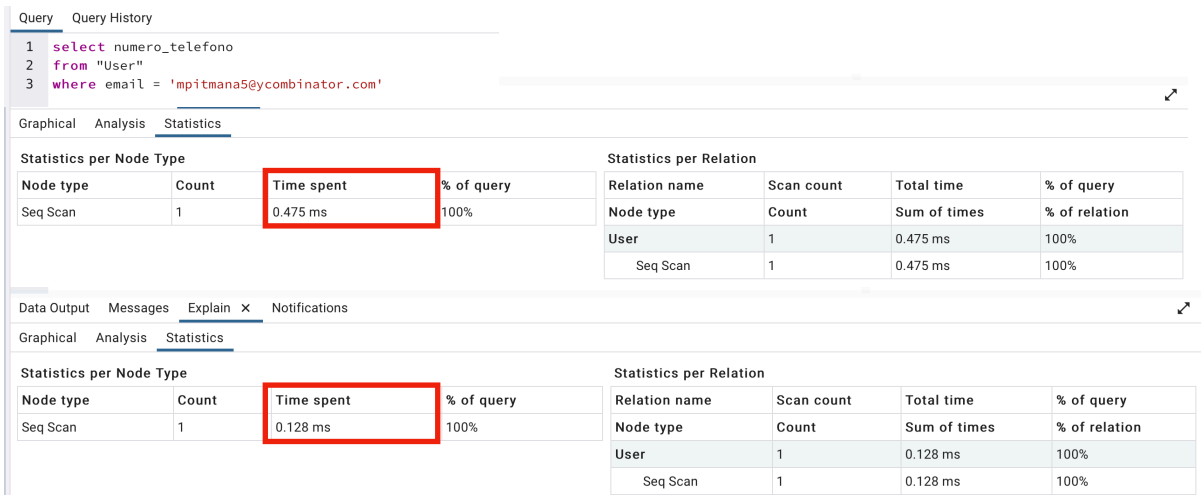
```
test_db.public> SELECT *
                FROM "Stato_Spedizione_Premium"
                WHERE DATE("data") > CURRENT_DATE
[2023-07-27 11:30:10] 500 rows retrieved starting from 1 in 89 ms (execution: 7 ms, fetching: 82 ms)
test_db.public> CREATE INDEX "Stato_Spedizione_Premium_data_index"
                ON public."Stato_Spedizione_Premium" USING btree (data)
[2023-07-27 11:30:19] completed in 17 ms
test_db.public> SELECT *
                FROM "Stato_Spedizione_Premium"
                WHERE DATE("data") > CURRENT_DATE
[2023-07-27 11:30:22] 500 rows retrieved starting from 1 in 38 ms (execution: 8 ms, fetching: 30 ms)
```

Per Stato_Spedizione_Economica, l'indice è il medesimo.

User

```
CREATE INDEX "User_numero_telefono_email_index"
ON public."User" USING btree (numero_telefono, email);
```

Ho optato per un indice, di tipo btree, sugli attributi "numero_telefono" e "email", in modo tale che la ricerca attraverso appunto il numero di telefono o la email, azione che verrebbe effettuata spesso durante la giornata, migliori drasticamente.
Riporto nella figura sottostante un benchmark. Nel primo esempio troviamo il benchmark senza indice, mentre nel secondo, la medesima query ma eseguita con la presenza dell'indice.



5.2 Check e Unique

Assicurazione:

```
constraint check_assicurazione
    check (((totale = true) AND (parziale = false)) OR ((totale = false) AND (parziale = true)))
```

In Assicurazione troviamo un check utile a mantenere i dati integri per quanto riguarda la possibilità di avere solamente un'assicurazione parziale o totale.
Ho inoltre creato un vincolo "UNIQUE" per il tracking affinchè vi sia solamente una occorrenza delle spedizioni premium.

Filiale:
Nella tabella "Filiale", ho impostato il vincolo "UNIQUE" sugli attributi legati all'indirizzo, ovvero "città", "numero_civico", "regione", "provincia" e "via". Questo vincolo è stato scelto perché è ovvio che ogni indirizzo corrisponde a una sola filiale.
Inoltre, ho deciso di non creare una tabella separata per gli indirizzi delle filiali, poiché ogni filiale ha un unico indirizzo, a differenza di User, i quali possono avere molteplici indirizzi.
Creare quindi una tabella separata avrebbe generato una relazione 1:1, la quale in questo caso non avrebbe avuto senso. Pertanto, ho preferito includere direttamente gli attributi dell'indirizzo nella tabella "Filiale".

Orario:
Orario contiene un check per quanto riguarda i giorni e come essi devono essere inseriti, in questo modo non vi saranno incongruenze sulla scrittura di essi. Non ho optato per un'ulteriore entità poichè i giorni della settimana rimarranno i medesimi.

User:

```
constraint check_codice_fiscale
    check ((codice_fiscale)::text ~* '^[A-Za-z]{6}\d{2}[A-Za-z]\d{2}[A-Za-z]\d{3}[A-Za-z]$'::text),
```

Nella tabella User ho deciso che il codice fiscale deve essere come nella realtà cioè con 6 caratteri iniziali, 2 numeri, 1 lettera, 2 cifre, una lettera, 3 cifre, una lettera finale.

Spedizione premium:

```
constraint mittente_destinatario_check
    check ((mittente)::text <> (destinatario)::text)
```

il check che troviamo nella figura riguarda la chiave esterna associata ad User, infatti un cliente può essere o mittente o destinatario, in una spedizione, ma non entrambe. Può ovviamente essere mittente in una spedizione e destinatario in un'altra.

5.3 Trigger

Il database contiene sette trigger che sono stati creati per garantire l'integrità dei dati. Di seguito verranno elencati i nomi e le funzionalità di questi trigger. Per alcuni trigger, è presente anche una versione speculare per le spedizioni economiche.
Il codice dettagliato di ciascun trigger è disponibile nel file di configurazione del database allegato. I trigger sono stati implementati per verificare automaticamente i dati inseriti o aggiornati nel database, in modo da evitare incongruenze o violazioni che senza di essi sarebbe stato impossibile da controllare.

Aggiorna_Assicurazione: Il trigger ha lo scopo di aggiornare il costo delle spedizioni premium correlate all'inserimento o l'aggiornamento dell'assicurazione.
Quando viene effettuato un inserimento o un aggiornamento sulla tabella "Assicurazione", il trigger controlla se esiste una corrispondente spedizione premium (tramite la colonna "tracking") nella tabella "Spedizione_Premium". Se esiste, il trigger calcola il nuovo costo della spedizione premium basato su un costo base di 7 euro con l'aggiunta della percentuale assicurata presente nella tabella "Assicurazione".

aggiorna_costo_spedizione_premium_servizi: Il trigger ha lo scopo di aggiornare il costo totale delle spedizioni premium nella tabella "Spedizione_Premium" in base al costo dei relativi servizi presenti nella tabella "Spedizione_Premium_Servizi".
Quando viene effettuato un inserimento o un aggiornamento sulla tabella "Assicurazione", il trigger calcola il nuovo costo delle spedizioni premium basato sui costi dei servizi correlati. Per fare ciò, il trigger preleva il valore del tracking dell'elemento aggiunto o modificato nella tabella "Assicurazione". Successivamente, viene eseguito un aggiornamento sulla tabella "Spedizione_Premium", dove il costo di ciascuna spedizione premium viene calcolato come la somma tra il costo precedente e il costo totale dei servizi correlati. Questa somma viene ottenuta utilizzando una subquery che recupera la somma dei costi dei servizi dalla tabella "Spedizione_Premium_Servizi" per la spedizione specificata.
Il costo risultante delle spedizioni premium viene quindi arrotondato a due decimali utilizzando la funzione TRUNC(), in modo da avere un valore con precisione a due cifre decimali.
Questa funzione che ritorna un trigger, è molto importante, perchè ci permette di avere sempre il costo della spedizione premium corretto, anche se di decide di aggiungere in seguito la percentuale assicurata. Non vi è quindi un ordine specifico con il quale inserire l'assicurazione o i servizi, il quale avrebbe reso sicuramente più semplice la scrittura della funzione, ma anche meno flessibile.

check_data_stato_spedizione_premium: Il trigger viene eseguito prima di un inserimento o un aggiornamento nella tabella, e il suo scopo è quello di controllare che la nuova data inserita per una determinata spedizione premium sia maggiore o uguale alla data precedente associata allo stesso tracking.

spedizionePremium_servizi_costo: In base all'operazione effettuata, il trigger aggiorna il costo totale della spedizione premium associata. Quando viene inserita una nuova riga nella tabella "Spedizione_Premium_Servizi", il trigger aggiorna il costo della spedizione premium sommando il costo del nuovo servizio al costo totale corrente della spedizione premium.
Se viene eliminata una riga dalla tabella "Spedizione_Premium_Servizi", il trigger sottrae il costo dell'elemento eliminato dal costo totale della spedizione premium.
Nel caso di aggiornamento di una riga esistente, il trigger controlla se esiste almeno un altro servizio aggiuntivo associato alla stessa spedizione nella tabella "Spedizione_Premium_Servizi". Se esiste almeno un altro servizio, il trigger aggiorna il costo totale della spedizione premium sottraendo il costo dell'elemento precedente e sommando il costo del nuovo elemento.

6. Esecuzione Database

Come prima considerazione, dato che le operazioni di inserimento dei record possono richiedere una notevole quantità di risorse computazionali, consiglio di eseguirle in lotti di dimensioni ridotte. Un metodo pratico che ho adottato consiste nell'eseguire gli inserimenti in maniera graduale, utilizzando l'IDE DataGrip. In questo modo si evita un possibile rallentamento, causato dalla quantità di record per ogni tabella.

Tutte le query possono essere eseguite singolarmente attraverso l'uso dei rispettivi metodi denominati "queryX", in cui "X" rappresenta il numero corrispondente alla query da eseguire. Questo approccio consente di eseguire ciascuna query separatamente all'interno del file main.cpp, agevolando così la gestione e l'esecuzione controllata delle singole operazioni.

Inoltre, è da sottolineare che la seconda query è configurata in modo parametrico. Ciò implica che all'interno del blocco "main" del file .cpp è possibile inserire manualmente il valore richiesto. Nello specifico, il valore può essere fornito interattivamente tramite l'utilizzo del comando "cin", consentendo l'input da tastiera. Questo approccio offre un alto grado di flessibilità, permettendo all'utente di fornire il parametro necessario in modo dinamico durante l'esecuzione del programma.