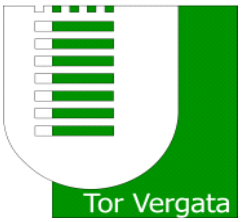


Basi di Dati

Progetto A.A. 2024/2025

TEMPLATE PROGETTO

0306689

Meola Gaia

Indice

1. Descrizione del Minimondo

2. Analisi dei Requisiti

3. Progettazione concettuale

4. Progettazione logica

5. Progettazione fisica

- **Descrizione del Minimondo**

Gestione di prenotazione taxi

Si vuole realizzare il sistema informativo di una società di trasporto taxi. Ciascun tassista può liberamente iscriversi al servizio, mettendo a propria disposizione il proprio veicolo. All'atto dell'iscrizione al servizio, l'autista deve registrare il proprio nome e cognome, il numero di patente, la targa del proprio veicolo ed il numero di carta di credito con cui pagare il costo del servizio. Deve inoltre comunicare quanti posti ha a disposizione il suo veicolo.

Gli utenti che utilizzano il sistema, all'atto della registrazione, devono fornire il proprio nome e cognome, un numero di telefono ed un numero di carta di credito per poter pagare le corse.

Quando hanno bisogno di un taxi, gli utenti indicano l'indirizzo di partenza e l'indirizzo di destinazione. Un tassista ha la possibilità di visualizzare tutte le richieste di corse ancora attive e di selezionare una corsa per assegnarsela. Se dopo due minuti dalla richiesta nessun tassista ha accettato la corsa, la richiesta viene marcata come scaduta.

Al termine della corsa il tassista indica l'importo mostrato dal tassametro. Il sistema calcolerà automaticamente la durata complessiva della corsa.

I gestori del servizio ricevono un compenso pari al 3% dell'importo di tutte le corse pagate ai tassisti. Per questo motivo, possono generare un report che riporti, in forma aggregata per ciascun tassista: quante corse sono state accettate, l'importo complessivo del guadagno del tassista, l'importo complessivo della commissione associato al tassista. Per le corse per cui non è ancora stata riscossa la percentuale, il sistema calcola la percentuale del 3%. I gestori possono indicare che l'importo è stato riscosso dalla carta di credito del tassista.

- **Analisi dei Requisiti**

- **Identificazione dei termini ambigui e correzioni possibili**

Linea	Termine	Nuovo termine	Motivo correzione
2	Informativo	Informatico	Ciò che si andrà a realizzare è un sistema informatico, ovvero una componente tecnologica di supporto al sistema informativo.
3	Liberamente	Gratuitamente	Disambiguato discutendo con il committente.
3, 5, 7	Veicolo	Taxi	Dal momento che si vuole realizzare un sistema informatico per la gestione di prenotazione taxi, si preferisce utilizzare il termine taxi piuttosto che veicolo.
4	Autista	Tassista	Dal momento che si sta realizzando un sistema informatico per la gestione di prenotazione taxi, si preferisce utilizzare il termine tassista piuttosto che autista.
6	Costo	Commissione	Disambiguato discutendo con il committente.
8, 11	Utenti	Clienti	Con l'espressione "utenti che utilizzano il sistema" si intende i clienti che richiedono le corse.
8	Registrazione	Iscrizione	Si preferisce utilizzare un termine già presente nelle specifiche, piuttosto che un sinonimo.
8	Fornire	Registrare	Si preferisce utilizzare un termine già presente nelle specifiche, piuttosto che un sinonimo.
21	Percentuale	Commissione	Confusione tra percentuale e commissione: la commissione altro non è che la percentuale del 3% dell'importo di una corsa pagata al tassista.
13	Corsa	Richiesta	Confusione tra corsa e richiesta: la richiesta è un'azione del cliente per richiedere una corsa, che può essere accettata o meno da un tassista. La corsa, invece, è il servizio effettivo, ovvero quando un tassista accetta la richiesta e trasporta il cliente fino alla destinazione specificata.

Specifica disambiguata

Si vuole realizzare il sistema informatico di una società di trasporto taxi. Ciascun tassista può

gratuitamente iscriversi al servizio, mettendo a propria disposizione il proprio taxi. All'atto dell'iscrizione al servizio, il tassista deve registrare il proprio nome e cognome, il numero di patente, la targa del proprio taxi ed il numero di carta di credito con cui pagare ai gestori del servizio l'importo complessivo della commissione a lui associato. Deve, inoltre, comunicare quanti posti ha a disposizione il suo taxi.
I clienti, all'atto dell'iscrizione, devono registrare il proprio nome e cognome, un numero di telefono ed un numero di carta di credito per pagare le corse.
Quando effettuano una richiesta, i clienti indicano l'indirizzo di partenza e l'indirizzo di destinazione. Un tassista ha la possibilità di visualizzare tutte le richieste di corse ancora attive e di selezionarne una per assegnarsela. Se dopo due minuti la richiesta non è stata accettata da nessun tassista, quest'ultima viene marcata come scaduta.
Al termine della corsa il tassista indica l'importo mostrato dal tassametro. Il sistema calcolerà automaticamente la durata complessiva della corsa.
I gestori del servizio ricevono un compenso pari al 3% dell'importo di tutte le corse pagate ai tassisti. Per questo motivo, possono generare un report che riporti, in forma aggregata per ciascun tassista: quante corse hanno effettuato, l'importo complessivo del guadagno del tassista, l'importo complessivo della commissione associato al tassista. Per le corse per cui non è ancora stata riscossa la commissione, il sistema calcola la percentuale del 3%. I gestori possono indicare che l'importo della commissione è stato riscosso dalla carta di credito del tassista.

- **Glossario dei Termini**

Termine	Descrizione	Sinonimi	Collegamenti
Cliente	Colui che effettua una richiesta di corsa.	Utente	Richiesta
Richiesta	Rappresenta l'azione effettuata da un cliente per ottenere una corsa, specificando l'indirizzo di partenza e di destinazione desiderato ed il numero di posti richiesti.	Corsa	Cliente, Tassista
Corsa	Rappresenta il servizio di trasporto effettuato da un tassista, a seguito	Richiesta	Richiesta, Taxi

	dell'accettazione di una richiesta da parte di un cliente, che prevede il trasferimento di quest'ultimo dal punto di partenza a quello di destinazione specificato. Al termine della corsa, l'importo è indicato dal tassista, mentre la durata complessiva è calcolata automaticamente dal sistema.		
Taxi	Rappresenta il veicolo messo a disposizione dal tassista per effettuare le corse.	Veicolo	Tassista, Corsa
Tassista	Colui che accetta una richiesta di corsa.	Autista	Richiesta, Taxi
Commissione	Rappresenta la percentuale dell'importo totale di una corsa (pari al 3%) riscossa dai gestori del servizio come compenso per l'utilizzo della piattaforma di gestione del trasporto taxi da parte dei tassisti.	Percentuale, Compenso	Corsa, Tassista

- **Raggruppamento dei requisiti in insiemi omogenei**

Frasi di carattere generale
Si vuole realizzare il sistema informatico di una società di trasporto taxi. [...] Per questo motivo, (i gestori del servizio) possono generare un report che riporti, in forma aggregata per ciascun tassista: quante richieste sono state accettate, l'importo complessivo del guadagno del tassista, l'importo complessivo della commissione associato al tassista. Per le corse per cui non è ancora stata riscossa la commissione, il sistema calcola la percentuale del 3%. I gestori possono indicare

che l'importo della commissione è stato riscosso dalla carta di credito del tassista.

Frasi relative a Cliente

I clienti, all'atto dell'iscrizione, devono registrare il proprio nome e cognome, un numero di telefono ed un numero di carta di credito per pagare le corse.

Quando effettuano una richiesta, i clienti indicano l'indirizzo di partenza e l'indirizzo di destinazione.

Frasi relative a Richiesta

Quando effettuano una richiesta, i clienti indicano l'indirizzo di partenza e l'indirizzo di destinazione. Un tassista ha la possibilità di visualizzare tutte le richieste di corse ancora attive e di selezionarne una per assegnarsela. Se dopo due minuti la richiesta non è stata accettata da nessun tassista, quest'ultima viene marcata come scaduta.

Frasi relative a Corsa

Al termine della corsa il tassista indica l'importo mostrato dal tassametro. Il sistema calcolerà automaticamente la durata complessiva della corsa. [...] I gestori del servizio ricevono un compenso pari al 3% dell'importo di tutte le corse pagate ai tassisti. Per questo motivo, possono generare un report che riporti, in forma aggregata per ciascun tassista: quante corse hanno effettuato, l'importo complessivo del guadagno del tassista, l'importo complessivo della commissione associato al tassista. Per le corse per cui non è ancora stata riscossa la commissione, il sistema calcola la percentuale del 3%.

Frasi relative a Taxi

Ciascun tassista può gratuitamente iscriversi al servizio, mettendo a propria disposizione il proprio taxi. All'atto dell'iscrizione al servizio, il tassista deve registrare [...] la targa del proprio taxi [...]. Deve, inoltre, comunicare quanti posti ha a disposizione il suo taxi.

Frasi relative a Tassista

Ciascun tassista può gratuitamente iscriversi al servizio, mettendo a propria disposizione il proprio taxi. All'atto dell'iscrizione al servizio, il tassista deve registrare il proprio nome e cognome, il numero di patente, la targa del proprio taxi ed il numero di carta di credito con cui pagare ai gestori del servizio l'importo complessivo della commissione a lui associato. [...] Un

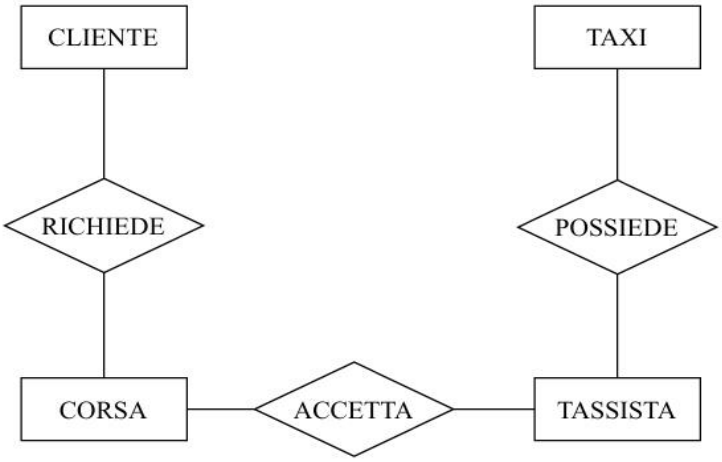
tassista ha la possibilità di visualizzare tutte le richieste di corse ancora attive e di selezionarne una per assegnarsela. [...] Al termine della corsa il tassista indica l'importo mostrato dal tassametro.

Frasi relative a Commissione
I gestori del servizio ricevono un compenso pari al 3% dell'importo di tutte le corse pagate ai tassisti. [...] Per le corse per cui non è ancora stata riscossa la commissione, il sistema calcola la percentuale del 3%. I gestori possono indicare che l'importo della commissione è stato riscosso dalla carta di credito del tassista.

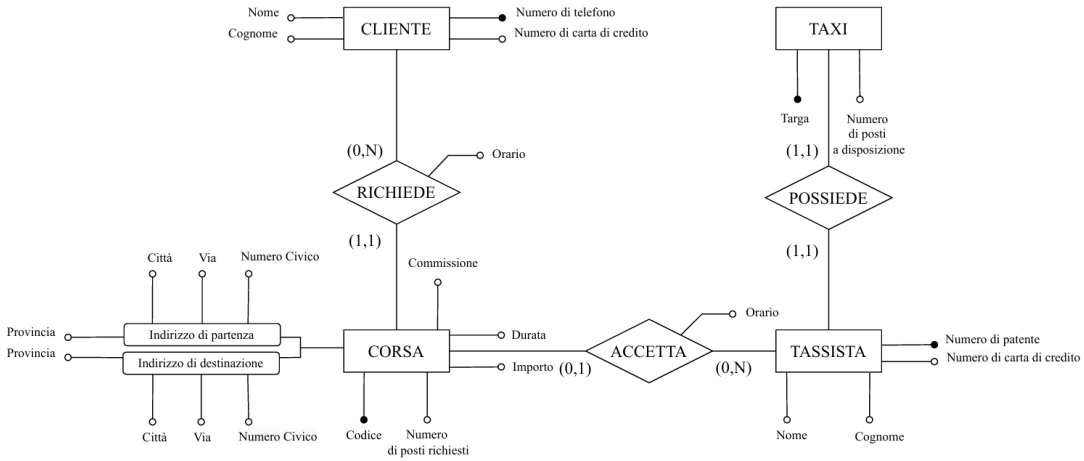
- **Progettazione concettuale**

- **Costruzione dello schema E-R**

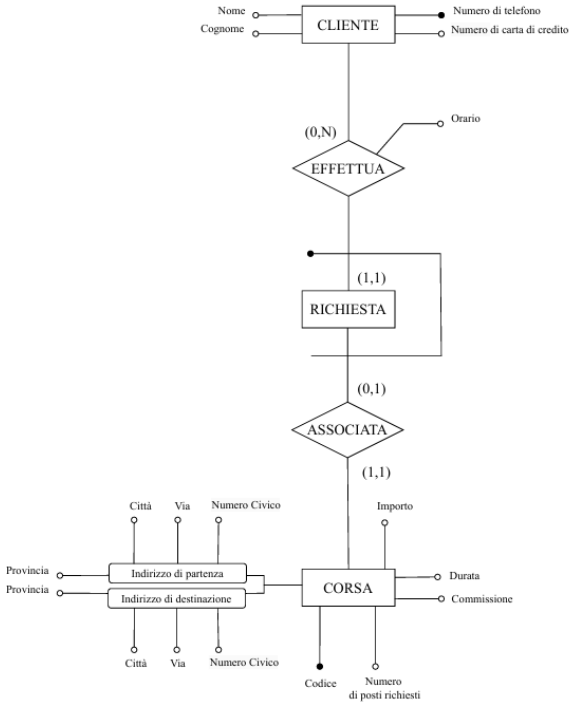
La costruzione del modello E-R è stata realizzata adottando un approccio **misto**, che combina i punti di forza delle metodologie Top-Down e Bottom Up. Il processo progettuale è iniziato con l'**individuazione** dei **concetti principali** dell'applicazione, basandosi per lo più sull'analisi dei requisiti forniti dal committente. Questa fase ha portato alla creazione dello schema scheletro seguente:



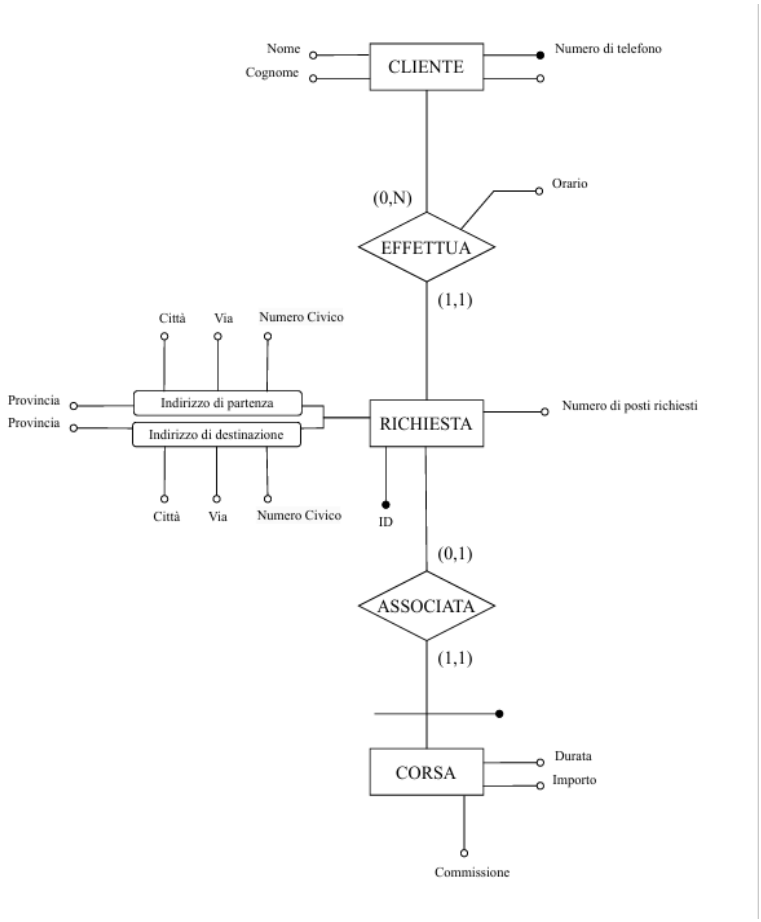
Successivamente, per ciascuna entità, sono stati aggiunti i **principali attributi**, ricavati dal testo oppure individuati a seguito del contatto con il committente; dal momento che tutti gli attributi presentano cardinalità **(1,1)**, si è scelto di non indicarla nel grafico per evitare ridondanze. Inoltre, sono stati individuati o introdotti, ove non fossero precisati nelle specifiche, anche i **principali identificatori** e sono state definite le **cardinalità** delle relazioni individuate fino a questo momento. Questo ha permesso di elaborare una prima bozza del modello E-R:



Il passo successivo è stato quello di affinare ulteriormente i concetti alla base del modello sviluppato. Innanzitutto, poiché un cliente può effettuare più richieste per la stessa corsa (dato che non è garantito che una richiesta venga accettata da un tassista e, secondo le specifiche, ogni richiesta scade automaticamente dopo due minuti), si è deciso di **reificare** la relazione tra "*Cliente"* e "*Corsa*" introducendo l'entità "*Richiesta*". Questo approccio consente di rappresentare in modo più chiaro e strutturato le informazioni relative alle richieste, gestendo anche la possibilità di duplicati.



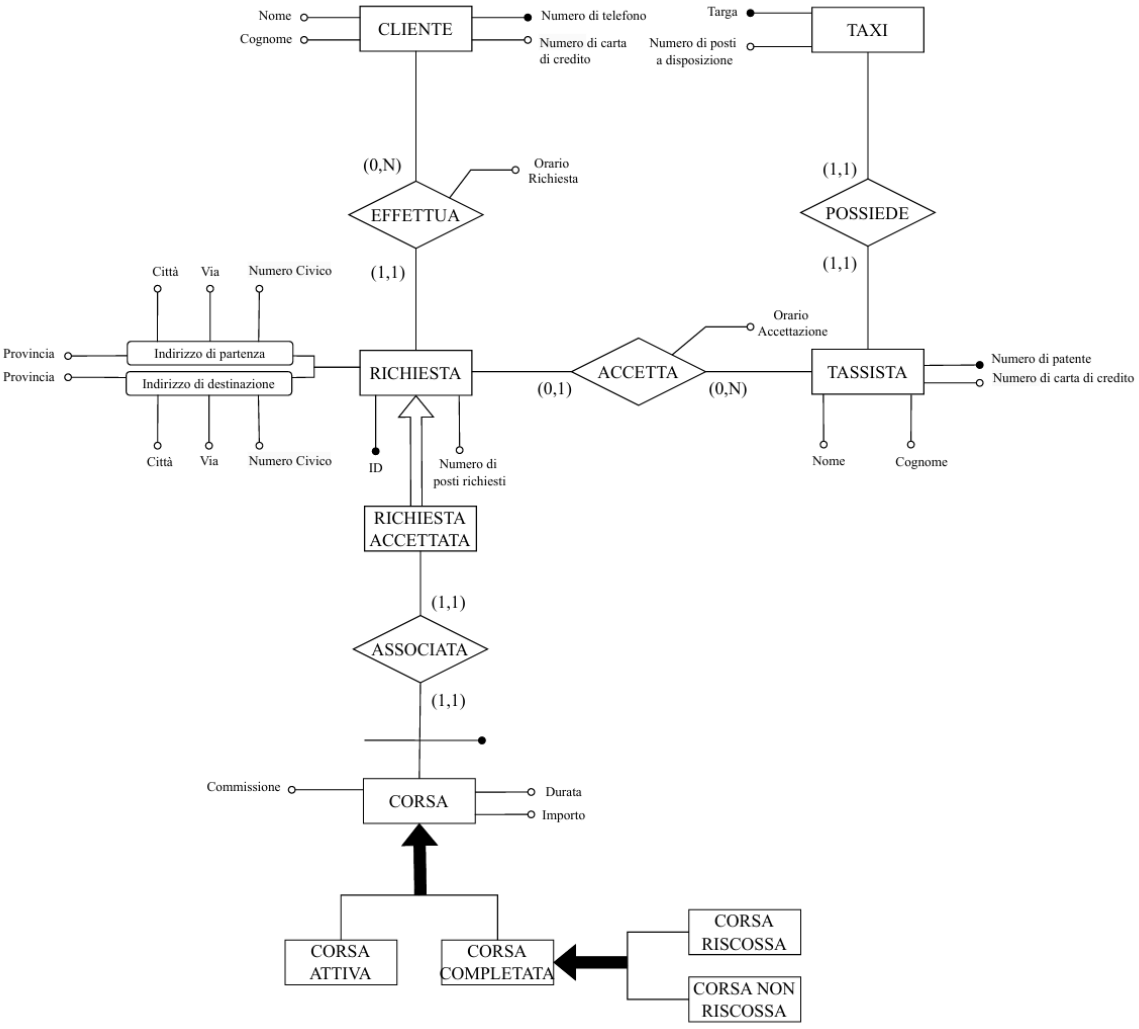
Successivamente, per semplificare il processo di identificazione ed evitare l'uso di identificatori esterni complessi, è stato introdotto un identificatore unico, denominato *ID*, che permette di distinguere una richiesta dall'altra. Con l'introduzione dell'entità "*Richiesta"* è stato, inoltre, possibile riorganizzare alcuni attributi precedentemente associati all'entità "*Corsa*": in particolare, gli attributi "*Indirizzo di Partenza*", "*Indirizzo di Destinazione*" e "*Numero di Posti Richiesti*" sono stati spostati nella nuova entità, in quanto descrivono le caratteristiche di una richiesta effettuata da un cliente, piuttosto che della corsa in sé. In maniera analoga, la relazione "*Accetta*" è stata collegata a "*Richiesta*" piuttosto che a "*Corsa*". Infine, dal momento che una corsa non può esistere senza una richiesta di corsa, si è scelto di rappresentare l'entità "*Corsa*" come entità debole.



In seguito, si è deciso di applicare il pattern della **generalizzazione** per rappresentare l'evoluzione di un concetto sia sull'entità "*Corsa*" sia sull'entità "*Richiesta*". In particolare, per l'entità "*Corsa*", è stata introdotta una generalizzazione che distingue tra "*Corsa Attiva*" e "*Corsa Completata*". Soltanto una "*Corsa Completata*" può essere ulteriormente specializzata in due tipologie: "*Corsa Riscossa*", che rappresenta una corsa per cui la commissione è stata riscossa dai gestori del servizio, e "*Corsa Non Riscossa*", che rappresenta una corsa la cui commissione non è stata ancora riscossa. Lo stesso approccio è stato adottato anche per l'entità "*Richiesta*": infatti, secondo le specifiche, una richiesta di corsa può trovarsi in tre stati distinti, cioè nello stato "*In attesa*", "*Scaduta*" oppure "*Accettata*". Per rappresentare questa variabilità e modellare più accuratamente il ciclo di vita di una richiesta, è stata introdotta l'entità "*Richiesta Accettata*". Quest'ultima consente di distinguere le richieste accettate da un tassista, che risultano, quindi, sempre associate a un'entità "*Corsa*".

Integrazione finale

Per maggiore chiarezza i due attributi "*Orario*" dell'associazione "*Effettua*" e dell'associazione "*Accetta*" sono stati rinominati rispettivamente in "*Orario Richiesta*" e "*Orario Accettazione*". Ecco come si presenta il **modello E-R finale**:



- **Regole aziendali**

1) Il ***numero di posti a disposizione*** di un taxi deve essere compreso tra 1 e 7.

2) Il ***numero di posti richiesti*** per una corsa deve essere compreso tra 1 e 7.

3) L'***indirizzo di partenza*** fornito da un cliente al momento della richiesta di una corsa deve essere diverso dall'***indirizzo di destinazione***.

4) L'***importo*** di una corsa è determinato dal tassametro e inserito dal tassista al termine della stessa.

5) La ***commissione*** di una corsa si ottiene calcolando il 3% dell'***importo*** della corsa pagata dal cliente al tassista.

6) La ***durata*** della corsa si calcola come la differenza tra l'***orario*** in cui il tassista ha inserito l'***importo*** della corsa e l'***orario di accettazione*** della richiesta.

7) Quando un ***cliente*** effettua una ***richiesta*** di corsa, la richiesta viene automaticamente marcata come ***in attesa***.

8) Quando un ***tassista*** accetta una ***richiesta*** di corsa in attesa, la richiesta viene marcata come ***accettata***.

9) Se una ***richiesta*** di corsa non viene ***accettata*** da un tassista entro due minuti dalla sua creazione, la richiesta viene automaticamente marcata come ***scaduta***.

10) Un ***cliente*** può avere ***in attesa*** una sola ***richiesta*** di corsa alla volta. Questo significa che, mentre una richiesta è in attesa di essere accettata da un tassista o è già stata accettata e si è trasformata in una corsa, il cliente non può effettuare un'altra richiesta. Per effettuare una nuova richiesta, deve attendere che la corsa venga completata oppure che la richiesta scada automaticamente dopo il periodo di tempo stabilito.

11) Un ***tassista*** può accettare una sola ***richiesta*** di corsa alla volta. Non è possibile accettare una nuova richiesta finché la corsa in cui è attualmente impegnato non è stata completata.

12) Un ***tassista*** può accettare una ***richiesta*** di corsa solo se il ***numero di posti richiesti*** dal cliente è minore o uguale al ***numero di posti a disposizione*** dichiarati per il taxi al momento della registrazione.

13) Nel momento in cui una ***richiesta*** di corsa viene accettata da un tassista, quest'ultima viene associata a una ***corsa***, che è inizialmente marcata come ***attiva***.

14) Nel momento in cui il tassista inserisce l'***importo*** di una corsa, quest'ultima viene automaticamente marcata come ***completata***.

15) Una corsa marcata come ***completata***, viene inizialmente impostata come ***non riscossa***.

16) La ***commissione*** di una corsa viene calcolata al momento della riscossione dell'importo totale da parte dei gestori del servizio.

17) Una ***corsa*** viene marcata come ***riscossa*** quando i gestori del servizio hanno incassato la commissione relativa al pagamento effettuato dal cliente al tassista.

- **Dizionario dei dati**

Entità	Descrizione	Attributi	Identificatori
Cliente	Le informazioni sul cliente che effettua una richiesta di corsa.	Nome, Cognome, Numero di Carta di Credito	Numero di telefono
Richiesta	Le informazioni relative ad una richiesta di corsa effettuata da un cliente.	Numero di posti richiesti,	ID

		Indirizzo di Partenza, Indirizzo di Destinazione	
Richiesta accettata	Stato della richiesta	Numero di posti richiesti, Indirizzo di Partenza, Indirizzo di Destinazione	ID
Corsa	Le informazioni relative al servizio di trasporto fornito dal tassista al cliente.	Durata, Importo, Commissione	Associata
Corsa attiva	Stato della corsa	Durata, Importo, Commissione	Associata
Corsa completata	Stato della corsa	Durata, Importo, Commissione	Associata
Corsa riscossa	Stato della corsa	Durata, Importo, Commissione	Associata
Corsa non riscossa	Stato della corsa	Durata, Importo, Commissione	Associata
Tassista	Le informazioni sul tassista che accetta una richiesta di corsa.	Nome, Cognome, Numero di Carta di Credito	Numero di patente
Taxi	Le informazioni sul taxi messo a disposizione da un tassista per il trasporto dei clienti.	Numero di posti a disposizione	Targa

- **Progettazione logica**

- **Volume dei dati**

Nell'analisi condotta, si assume che il sistema conservi i dati relativi ai profili dei clienti e dei tassisti fino alla cessazione dell'account. Per quanto riguarda, invece, la gestione delle richieste e delle corse:

- Le richieste di corse marcate come scadute verranno rimosse giornalmente, in quanto non più rilevanti per il sistema.
- Le richieste accettate e le corse riscosse da oltre sei mesi verranno archiviate, garantendo comunque la possibilità di consultazione per eventuali verifiche.
- I dati relativi alle richieste archiviate e alle corse riscosse verranno definitivamente rimossi dopo dieci anni dalla data di inserimento nella tabella di archivio, in linea con le politiche di conservazione dei dati e le necessità di gestione del sistema.

Per stimare il volume atteso delle operazioni a regime, si è preso come riferimento una città di grandi dimensioni, come ad esempio Roma. In tale contesto, sono state fatte dell'ipotesi sul numero di transazioni, richieste e attività che il sistema sarà in grado di gestire in condizioni di normale funzionamento, ovvero una volta raggiunta la piena operatività e stabilità. A partire da alcune stime iniziali, è stato calcolato il volume atteso delle operazioni, ipotizzando che ogni corsa abbia una durata media pari a 20 minuti e che il sistema possa gestire il flusso di corse in maniera continua, operando 24 ore su 24, mentre i tassisti lavorino mediamente 12 ore al giorno. Si assume, inoltre, che un cliente effettui mediamente una richiesta al giorno e che i gestori del servizio effettuino la riscossione delle corse settimanalmente.

Concetto nello schema	Tipo	Volume atteso
Cliente	E	600.000
Effettua	R	600.000
Richiesta	E	600.000
Richiesta accettata	E	280.000
Accetta	R	280.000
Tassista	E	8.000

Possiede	R	8.000
Taxi	E	8.000
Corsa	E	280.000
Corsa attiva	E	24.000
Corsa completata	E	256.000
Corsa riscossa	E	40.000
Corsa non riscossa	E	216.000
Associata	R	280.000

- **Tavola delle operazioni**

L'obiettivo della seguente analisi è quello di focalizzarsi esclusivamente sulle operazioni più significative, tralasciando quelle di routine o automatizzate. Pertanto, sono state omesse dalla tavola delle operazioni tutte quelle ritenute banali, come ad esempio '*Ottieni Numero Patente'*, *'Ottieni Numero Telefono'*, '*Login'* etc. in quanto non rappresentano azioni significative dal punto di vista del sistema, ma piuttosto semplici interrogazioni di dati. Allo stesso modo, sono state escluse tutte quelle operazioni legate a processi automatizzati gestiti dal sistema, come ad esempio l'eliminazione delle richieste scadute oppure il calcolo della durata di una corsa, poiché sono svolte senza necessità di intervento diretto da parte degli utenti.

Cod.	Descrizione	Frequenza attesa
CL1	Registra cliente	15/giorno
TS1	Registra tassista	1/giorno
CL2	Richiedi corsa	600.000/giorno
TS2	Visualizza richieste di corsa in attesa	280.000/giorno
TS3	Accetta richiesta di corsa	280.000/giorno
TS4	Inserisci importo	280.000/giorno
GS1	Report tassisti	1/mese
GS2	Visualizza corse non riscosse	4/mese
GS3	Riscuoti commissione	4/mese

- **Costo delle operazioni**

Ipotizziamo che il costo in scrittura di un dato sia doppio rispetto a quello in lettura.

Op. CL1 – Registra Cliente

L'operazione prevede la creazione di una nuova occorrenza nell'entità "*Cliente"* tramite un accesso in scrittura con frequenza *f* pari a *15 volte al giorno*.

- **Numero Accessi Totali:** 1S x *f* = 2 x *15/giorno* = **30/giorno**

Op. TS1 – Registra Tassista

L'operazione prevede la creazione di una nuova occorrenza nell'entità "*Tassista"* tramite un accesso in scrittura, insieme alla creazione di una nuova occorrenza nell'associazione "*Possiede"* e nell'entità "*Taxi"* con frequenza *f* pari a *1 volta al giorno*.

- **Numero Accessi Totali:** 1S x *f* + 1S x *f* + 1S x *f* = 6 x *1/giorno* = **6/giorno**

Op. CL2 – Richiedi Corsa

L'operazione prevede un accesso in lettura all'entità "*Cliente"* per ottenere i dati necessari all'inserimento di una nuova occorrenza nell'associazione "*Effettua"* e nell'entità "*Richiesta"* con frequenza *f* pari a *600.000 volte al giorno*.

- **Numero Accessi Totali:** 1L x *f* + 1S x *f* + 1S x *f* = 1 x *600.000/giorno* + 4 x *600.000/giorno* = **3.000.000/giorno**

Op. TS2 – Visualizza Richieste Di Corsa In Attesa

L'operazione prevede un accesso in lettura all'entità "*Richiesta"* con frequenza *f* pari a *280.000/giorno*.

- **Numero Accessi Totali:** 1L x *f* = 1 x *280.000/giorno* = **280.000/giorno**

Op. TS3 – Accetta Richiesta di Corsa

L'operazione prevede un accesso in modalità lettura all'entità "*Tassista"* per recuperare i dati necessari, un accesso in modalità lettura all'entità "*Possiede*" e un accesso in modalità lettura all'entità "*Taxi*" per ottenere le informazioni aggiuntive, un accesso in modalità scrittura all'entità "*Richiesta",* un accesso in modalità scrittura all'entità "*Richiesta Accettata"* per creare la relativa occorrenza, un accesso in scrittura alla relazione "*Associata*" per registrare l'associazione tra la richiesta accettata e la corsa e, infine, un accesso in modalità scrittura all'entità "*Corsa*" per creare una nuova occorrenza con frequenza *f* pari a *280.000/giorno*.

- **Numero Accessi Totali:** 1L x *f* + 1L x *f* + 1L x *f* + 1S x *f* + 1S x *f* + 1S x *f* + 1S x *f* + + 1S x *f* = 3 x *280.000/giorno* + 8 x *280.000/giorno* = **3.080.000/giorno**

Op. TS4 – Inserisci Importo

L'operazione prevede un accesso in modalità scrittura all'entità "*Corsa"* ed un accesso in modalità scrittura all'entità "*Corsa Completata*" così da registrare l'importo inserito dal tassista al termine della corsa, corrispondente alla somma che il cliente dovrà pagare con frequenza *f* pari a *280.000/giorno*.

- **Numero Accessi Totali:** 1S x *f* + 1S x *f* = 4 x *280.000/giorno* = **1.120.000/giorno**

Op. GS1 – Report Tassisti

L'operazione prevede un accesso in modalità lettura a tutte le occorrenze dell'entità "*Corsa"*, "*Associata*", "*Richiesta Accettata", "Richiesta"* e, infine, dell'entità "*Tassista"* con frequenza *f* pari a *1/mese*.

- **Numero Accessi Totali:** 1L x *f* + 1L x *f* + 1L x *f* + 1L x *f* + 1L x *f* = 5 * *1/mese* = **5/mese**

Op. GS2 – Visualizza Corse Non Riscosse

L'operazione prevede un accesso in modalità lettura a tutte le occorrenze dell'entità "*Corsa"* e all'entità "*Corsa Non Riscossa*" con frequenza *f* pari a *4/mese.*

- **Numero Accessi Totali:** 1L x *f* + 1L x *f* = 2 x *4/mese* = **8/mese**

Op. GS3 – Riscuoti Commissione

L'operazione prevede un accesso in modalità lettura all'entità "*Corsa*" per ottenere i dati relativi ad una corsa, un accesso in modalità lettura all'entità "*Corsa Completata*", un accesso in modalità lettura all'entità "*Corsa Non Riscossa"* e, infine, un accesso in modalità scrittura all'entità "Corsa Riscossa" per creare la relativa occorrenza con frequenza *f* pari a *4/mese*.

- **Numero Accessi Totali:** 1L * *f* + 1L * *f* + 1L * *f* + *1*S * *f* = 3 x *4/mese* + 2 x *4/mese* = **20/mese**

• **Ristrutturazione dello schema E-R**

1) **Analisi delle ridondanze**

   Nello schema è presente un solo dato ridondante, cioè l'attributo **Commissione** nell'entità 'Corsa', che, infatti, può essere calcolato direttamente come il 3% dell'attributo '*Importo*' presente nella stessa entità. Mantenere questo valore separato comporta un inutile spreco di spazio, in quanto non è indispensabile memorizzarlo, dato che è richiesto soltanto da alcune operazioni specifiche (come, ad esempio, le operazioni **GS1** e **GS3**). Tali operazioni, infatti, non sono particolarmente frequenti e non giustificano la memorizzazione di un dato che può essere facilmente calcolato all'occorrenza.

2) **Eliminazione delle generalizzazioni**:

   2.1) **Generalizzazione tra "Corsa Completata" e le sue sottocategorie "Corsa Riscossa" e "Corsa Non Riscossa"**: poiché le entità figlie rappresentano un'evoluzione del concetto di "*Corsa Completata*" e non sono, infatti, relazionate ad altre entità del modello, si è deciso di mantenere tutte le informazioni relative alla riscossione

direttamente nell'entità padre "*Corsa Completata*", tramite l'introduzione di un attributo, definito come "*Status Riscossione*", che indica se la commissione associata alla corsa è stata riscossa o meno dai gestori del servizio.

2.2) **Generalizzazione tra "Corsa" e le sue sottocategorie "Corsa Attiva" e "Corsa Completata"**: analogamente alla generalizzazione precedente, poiché le entità figlie rappresentano un'evoluzione del concetto di corsa e non sono collegate ad altre entità del modello, si è deciso di accorpare le informazioni relativa al termine della corsa direttamente nell'entità padre "*Corsa*" mediante l'introduzione dell'attributo "*Terminata*", che indica se la corsa è stata completata o meno.

2.3) **Generalizzazione tra "Richiesta" e "Richiesta Accettata"**: il concetto di accettazione della richiesta altro non rappresenta che un'evoluzione del concetto base di richiesta. Di conseguenza, si è scelto di gestire questa evoluzione direttamente all'interno dell'entità "*Richiesta*". A tal fine, è stato introdotto un nuovo attributo chiamato "*Stato*", che può assumere i valori di "*In attesa*, "*Scaduta*" oppure "*Accettata*", permettendo di rappresentare, appunto, lo stato attuale di una richiesta di corsa effettuata da un cliente. Inoltre, l'associazione con l'entità "*Corsa*" diventa opzionale in quanto solo le richieste con lo stato "*Accettata*" necessitano di un'associazione ad una corsa.

3) **Accorpamento di entità**:

Per quanto riguarda le entità "*Tassista"* e "*Taxi",* si è deciso di effettuare un accorpamento dell'entità "*Taxi*" all'interno dell'entità "*Tassista*", poiché le operazioni più frequenti sull'entità "*Tassista"* richiedono sempre anche i dati relativi al taxi con cui il tassista si è registrato.

4) **Eliminazione degli attributi multi-valore**:

Nello schema E-R finale sono presenti due attributi multi-valore, ossia "*Indirizzo di partenza*" ed "*Indirizzo di destinazione*" che indicano rispettivamente il punto di inizio della corsa richiesta dal cliente, cioè l'indirizzo da cui il cliente desidera essere prelevato, e il punto di arrivo della corsa, cioè l'indirizzo in cui il cliente desidera essere portato. Nel processo di ristrutturazione dello schema E-R, si è scelto di eliminare i suddetti attributi multi-valore reificandoli in una sola nuova entità chiamata "*Indirizzo*", il cui identificatore è dato dall'insieme di tutti gli attributi che compongono gli attributi "*Indirizzo di partenza*" ed "*Indirizzo di destinazione*". Inserendo l'entità Indirizzo, sono state definite anche due nuove relazioni, chiamate "*Partenza*" e "*Destinazione*", che collegano l'entità "*Richiesta*" all'entità "*Indirizzo*": in particolare, la relazione "*Partenza*" associa una richiesta al suo indirizzo di partenza, mentre, invece, la relazione "*Destinazione*" associa una richiesta al suo indirizzo di destinazione

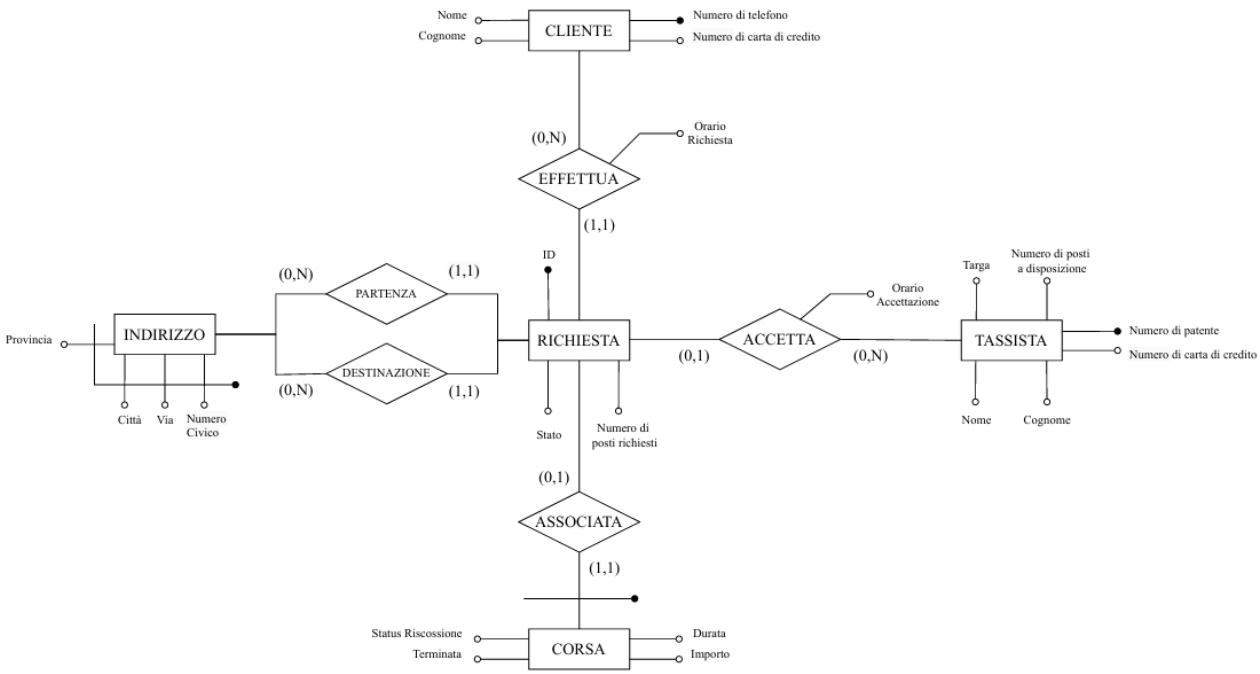
5) **Scelta degli identificatori primari**:

- **Cliente**: Numero di telefono
- **Tassista**: Numero di Patente
- **Richiesta**: ID
- **Corsa**: ID

- **Indirizzo**: Provincia, Città, Via, Numero Civico

- **Trasformazione di attributi e identificatori**

Nello schema è presente un unico identificatore esterno, ovvero **ID**, che evidenzia come una corsa sia concettualmente legata a una richiesta di corsa specifica. In parole povere, una corsa esiste solo se è associata ad una richiesta di corsa effettuata da un cliente. Ecco come si presenta il **modello E-R finale ristrutturato**:



- **Traduzione di entità e associazioni**

Traduzione di entità e associazioni nello schema relazionale:

- **Cliente** (**NumerodiTelefono**, Nome, Cognome, NumeroCartadiCredito)

- **Tassista** (**NumerodiPatente**, Nome, Cognome, NumeroCartadiCredito, Targa, NumeroPostiADisposizione)

- **Richiesta** (**ID**, OrarioRichiesta, **Cliente**, **ProvinciaPartenza**, **CittàPartenza**, **ViaPartenza**, **NumeroCivicoPartenza**, **ProvinciaDestinazione**, **CittàDestinazione**, **ViaDestinazione**, **NumeroCivicoDestinazione**, NumerodiPostiRichiesti, StatoRichiesta, **Tassista*,** OrarioAccettazione*)

- **Corsa** (**IDRichiesta**, Durata, Importo, StatusRiscossione, Terminata)

- **Indirizzo** (**Provincia**, **Città**, **Via**, **NumeroCivico**)

Vincoli di integrità referenziale:

- ***Richiesta*** (**Cliente**) ⊆ ***Cliente*** (**NumerodiTelefono**)
- ***Richiesta*** (**ProvinciaPartenza**, **CittàPartenza**, **ViaPartenza**, **NumeroCivicoPartenza**) ⊆ ***Indirizzo*** (**Provincia**, **Città**, **Via**, **NumeroCivico**)
- ***Richiesta*** (**ProvinciaDestinazione**, **CittàDestinazione**, **ViaDestinazione**, **NumeroCivicoDestinazione**) ⊆ ***Indirizzo*** (**Provincia**, **Città**, **Via**, **NumeroCivico**)
- **Richiesta** (***Tassista***) ⊆ **Tassista** (***NumerodiPatente***)
- **Corsa** (***IDRichiesta***) ⊆ **Richiesta** (***ID***)

Normalizzazione del modello relazionale

Una relazione è in **BCNF** se per ogni dipendenza funzionale **X → Y**, la chiave candidata **X** è una superchiave.

- Nel caso della relazione **Cliente** abbiamo le seguenti due dipendenze funzionali:

1) NumeroDiTelefono → Nome, Cognome, NumeroCartaDiCredito
2) NumeroCartaDiCredito → Nome, Cognome, NumeroDiTelefono

  Quindi, la relazione **Cliente** rispetta la forma BCNF dal momento che **NumeroDiTelefono** e **NumeroCartaDiCredito** sono entrambe super-chiavi, poiché il primo attributo è proprio la chiave primaria della relazione, mentre il secondo è un attributo unico.

- Nel caso della relazione **Tassista** abbiamo le seguenti dipendenze funzionali:

  1) **NumeroDiPatente → Nome, Cognome, NumeroCartaDiCredito, Targa, NumeroDiPostiADisposizione**
  2) **NumeroCartaDiCredito → Nome, Cognome, NumeroDiPatente, Targa, NumeroDiPostiADisposizione**
  3) **Targa → Nome, Cognome, NumeroCartaDiCredito, NumeroDiPostiADisposzione, NumeroDiPatente**

  Quindi, la relazione **Tassista** rispetta la forma BCNF dal momento che **NumeroDiPatente**, **NumeroCartaDiCredito** e **Targa** sono tutte super-chiavi.

- Nel caso della relazione **Indirizzo** c'è un'unica dipendenza funzionale banale, quindi rispetta la forma normale BCNF.

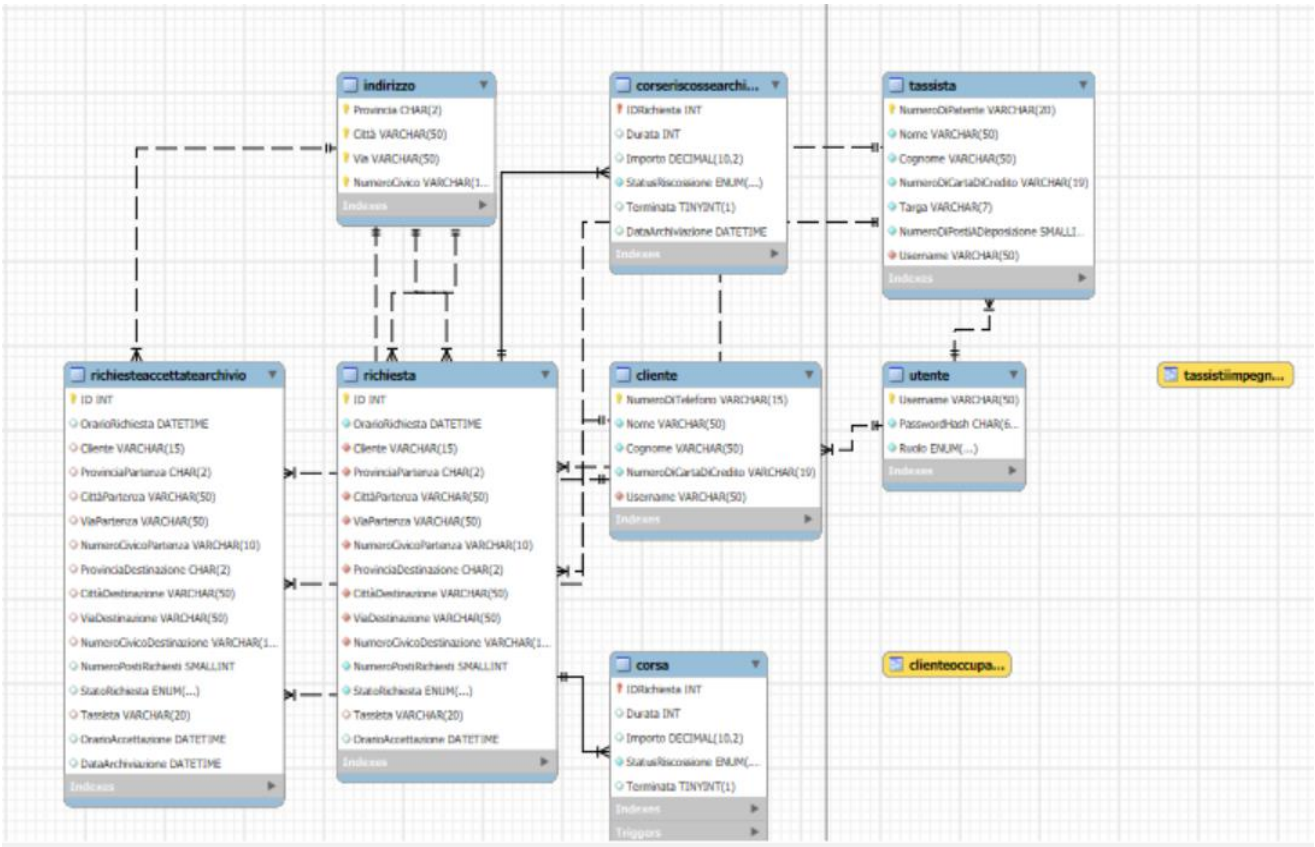
- Nel caso della relazione **Richiesta** abbiamo le seguenti dipendenze funzionali:

1) **ID → OrarioRichiesta, Cliente, ProvinciaPartenza, CittàPartenza, ViaPartenza, NumeroCivicoPartenza, ProvinciaDestinazione, CittàDestinazione, ViaDestinazione, NumeroCivicoDestinazione, NumeroDiPostiRichiesti, StatoRichiesta, Tassista, OrarioAccettazione**
2) **Cliente, OrarioRichiesta → ID**
3) **Tassista, OrarioAccettazione → ID**

La relazione **Richiesta** rispetta la forma BCNF dal momento che ogni dipendenza funzionale ha una parte sinistra che è una superchiave.

- Nel caso della relazione **Corsa** a c'è un'unica dipendenza funzionale, quindi rispetta la forma normale BCNF.

Il modello relazionale descritto rispetta BCNF in tutte le sue relazioni. Ogni dipendenza funzionale ha una parte sinistra che è una superchiave, il che significa che tutte le relazioni sono normalizzate correttamente.



- **Progettazione fisica**

- **Utenti e privilegi**

Ogni utente che accede al sistema deve autenticarsi utilizzando un login, che comprende un *nome utente* e una *password*. L'accesso è regolato da quattro ruoli distinti, ognuno con privilegi specifici in base al *Principle of Least Privilege*, che assicura che ogni utente possa eseguire solo le operazioni necessarie alla propria funzione.

- **Login**: Grant in esecuzione sull'operazione **Login**, **CL1** (Registra Cliente) e **TS1** (Registra Tassista)
- **Cliente**: Grant in esecuzione sull'operazione **CL2** (Richiedi Corsa)
- **Tassista**: Grant in esecuzione sull'operazione **TS2** (Visualizza richieste di corsa), **TS3** (Accetta richiesta di corsa) e **TS4** (Inserisci importo).
- **Gestore del servizio**: Grant in esecuzione sull'operazione **GS1** (Report Tassisti), **GS2** (Visualizza corse non riscosse) e **GS3** (Riscuoti commissione).

- L'utente con ruolo "*Login*" è l'utente che può effettuare l'accesso all'applicazione inserendo le proprie credenziali oppure registrandosi al sistema, come "*Cliente*" oppure "*Tassista*".
- L'utente con ruolo "*Cliente*" è l'utente che può richiedere una corsa.
- L'utente con ruolo "*Tassista*" è l'utente che può visualizzare le richieste di corsa, accettare una richiesta di corsa ed inserire l'importo di una corsa per terminarla.
- L'utente con ruolo "*Gestore del servizio*" può richiedere la generazione di un report sui tassisti iscritti, visualizzare le corse la cui commissione non è stata ancora riscossa e può riscuotere la commissione relativa ad una corsa.

Si introduce una tabella **Utenti** per mantenerne le credenziali, quindi *username*, *password* e *ruolo*.

- **Strutture di memorizzazione**

Tabella Cliente		
Colonna	**Tipo di dato**	**Attributi**
Numero di Telefono	VARCHAR (15)	PRIMARY KEY
Nome	VARCHAR (50)	NOT NULL
Cognome	VARCHAR (50)	NOT NULL
NumeroDiCartaDiCredito	VARCHAR (19)	NOT NULL, UNIQUE

Username	VARCHAR (50)	NOT NULL, UNIQUE, FOREIGN KEY

Tabella Tassista		
Colonna	**Tipo di dato**	**Attributi**
Numero di Patente	VARCHAR (20)	PRIMARY KEY
Nome	VARCHAR (50)	NOT NULL
Cognome	VARCHAR (50)	NOT NULL
NumeroDiCartaDiCredito	VARCHAR (19)	NOT NULL, UNIQUE
Targa	VARCHAR (7)	NOT NULL, UNIQUE
NumeroDiPostiADisposizione	SMALLINT	NOT NULL
Username	VARCHAR (50)	NOT NULL, UNIQUE, FOREIGN KEY

Tabella Utente		
Colonna	**Tipo di dato**	**Attributi**
Username	VARCHAR (50)	PRIMARY KEY
PasswordHash	CHAR (64)	NOT NULL
Ruolo	ENUM	NOT NULL

Tabella Indirizzo		
Colonna	**Tipo di dato**	**Attributi**
Provincia	CHAR (2)	NOT NULL, PRIMARY KEY
Città	VARCHAR (50)	NOT NULL, PRIMARY KEY
Via	VARCHAR (50)	NOT NULL, PRIMARY KEY
Numero Civico	VARCHAR (10)	NOT NULL, PRIMARY KEY

Tabella Richiesta		
Colonna	**Tipo di dato**	**Attributi**
ID	INT	PRIMARY KEY, AUTO_INCREMENT

Orario Richiesta	DATETIME	NOT NULL
Cliente	VARCHAR (15)	NOT NULL, FOREIGN KEY
Provincia Partenza	CHAR (2)	NOT NULL, FOREIGN KEY
Città Partenza	VARCHAR (50)	NOT NULL, FOREIGN KEY
Via Partenza	VARCHAR (50)	NOT NULL, FOREIGN KEY
NumeroCivicoPartenza	VARCHAR (10)	NOT NULL, FOREIGN KEY
ProvinciaDestinazione	CHAR (2)	NOT NULL, FOREIGN KEY
CittàDestinazione	VARCHAR (50)	NOT NULL, FOREIGN KEY
ViaDestinazione	VARCHAR (50)	NOT NULL, FOREIGN KEY
NumeroCivicoDestinazione	VARCHAR (10)	NOT NULL, FOREIGN KEY
NumeroPostiRichiesti	SMALLINT	NOT NULL
StatoRichiesta	ENUM	NOT NULL
Tassista	VARCHAR (20)	
Orario Accettazione	DATETIME	

Tabella Corsa		
Colonna	**Tipo di dato**	**Attributi**
ID Richiesta	INT	PRIMARY KEY, FOREIGN KEY
Durata	INT	
Importo	DECIMAL (10,2)	
Status Riscossione	ENUM	NOT NULL
Terminata	BOOLEAN	NOT NULL

Per ottimizzare le prestazioni durante le operazioni quotidiane, si è deciso di archiviare separatamente i dati relativi alle richieste accettate e alle corse riscosse da oltre *sei mesi*, senza eliminarli definitivamente. In questo modo, pur garantendo che i report richiesti dai gestori siano basati su dati recenti, rimane comunque possibile recuperare le informazioni relative alle corse più datate se necessario, entro un periodo massimo di dieci anni dall'inserimento nelle tabelle di archivio.

Tabella RichiesteAccettateArchivio

Colonna	Tipo di dato	Attributi
ID	INT	PRIMARY KEY, AUTO_INCREMENT
Orario Richiesta	DATETIME	NOT NULL
Cliente	VARCHAR (15)	NOT NULL, FOREIGN KEY
Provincia Partenza	CHAR (2)	NOT NULL, FOREIGN KEY
Città Partenza	VARCHAR (50)	NOT NULL, FOREIGN KEY
Via Partenza	VARCHAR (50)	NOT NULL, FOREIGN KEY
NumeroCivicoPartenza	VARCHAR (10)	NOT NULL, FOREIGN KEY
ProvinciaDestinazione	CHAR (2)	NOT NULL, FOREIGN KEY
CittàDestinazione	VARCHAR (50)	NOT NULL, FOREIGN KEY
ViaDestinazione	VARCHAR (50)	NOT NULL, FOREIGN KEY
NumeroCivicoDestinazione	VARCHAR (10)	NOT NULL, FOREIGN KEY
NumeroPostiRichiesti	SMALLINT	NOT NULL
StatoRichiesta	ENUM	NOT NULL
Tassista	VARCHAR (20)	
Orario Accettazione	DATETIME	
DataArchiviazione	DATETIME	

Tabella CorseRiscosseArchivio		
Colonna	**Tipo di dato**	**Attributi**
ID Richiesta	INT	PRIMARY KEY, FOREIGN KEY
Durata	INT	
Importo	DECIMAL (10,2)	
Status Riscossione	ENUM	NOT NULL
Terminata	BOOLEAN	NOT NULL
DataArchiviazione	DATETIME	

- **Indici**

Non sono stati utilizzati indici specifici per migliorare le prestazioni, in quanto le operazioni di lettura e modifica non richiedono ottimizzazioni particolari. Le operazioni principali (come inserimenti, aggiornamenti e cancellazioni) non sono particolarmente gravose in termini di

tempo di esecuzione, e pertanto non è stato ritenuto necessario introdurre indici aggiuntivi oltre quelli autogenerati per la gestione delle chiavi primarie.

- **Trigger**

1) Trigger: **VerificaImportoCorsaTerminata**

Questo trigger ha lo scopo di impedire a un tassista di modificare l'importo di una corsa che è già stata contrassegnata come *terminata*. In questo modo, si garantisce l'integrità dei dati e si evita che un importo venga alterato dopo la conclusione della corsa.

```
CREATE TRIGGER VerificaImportoCorsaTerminata
BEFORE UPDATE ON corsa
FOR EACH ROW BEGIN
        IF OLD.Terminata = 1 AND OLD.Importo <> NEW.Importo THEN SIGNAL
        SQLSTATE '45010'
        SET MESSAGE_TEXT = 'Errore: la corsa è già terminata, non è possibile
modificare l\'importo.';
        END IF;
END;
```

2) Trigger: **VerificaCommissioneRiscossa**

Questo trigger impedisce di impostare lo stato di riscossione di una corsa come *"Riscossa"* se la corsa non è ancora terminata o se la commissione è già stata riscossa.

```
CREATE TRIGGER VerificaCommissioneRiscossa
BEFORE UPDATE ON Corsa
FOR EACH ROW BEGIN
        IF (NEW.StatusRiscossione = 'Riscossa') THEN
                IF (OLD.Terminata = FALSE OR OLD.StatusRiscossione = 'Riscossa')
                        THEN SIGNAL SQLSTATE '45009'
                        SET MESSAGE_TEXT = 'Impossibile riscuotere una corsa se non
                è ancora terminata oppure se è già stata riscossa';
                END IF;
        END IF;
END;
```

3) Trigger: **ControlloAccettazioneCorsa**

Questo trigger impedisce che una richiesta di corsa venga accettata se è già stata precedentemente accettata o se è scaduta.

```
CREATE TRIGGER ControlloAccettazioneCorsa
BEFORE UPDATE ON Richiesta
FOR EACH ROW
```

```
BEGIN
        IF NEW.StatoRichiesta= 'Accettata' AND (OLD.StatoRichiesta = 'Accettata' OR
        OLD.StatoRichiesta = 'Scaduta')
        THEN SIGNAL SQLSTATE '45008'
        SET MESSAGE_TEXT = 'Non puoi accettare una richiesta che è già stata
accettata o è scaduta.';
        END IF;
END;
```

4) Trigger: **ControlloIndirizzoPartenzaDestinazione**(regola aziendale **3**)

Questo trigger impedisce l'inserimento di una richiesta di corsa se l'indirizzo di partenza è uguale all'indirizzo di destinazione.

```
CREATE TRIGGER ControlloIndirizzoPartenzaDestinazione
BEFORE INSERT ON Richiesta
FOR EACH ROW
BEGIN
        IF (NEW.ViaPartenza = NEW.ViaDestinazione AND NEW.CittàPartenza =
NEW.CittàDestinazione)
        THEN SIGNAL SQLSTATE '45007'
        SET MESSAGE_TEXT = 'L\'indirizzo di partenza deve essere diverso da quello
di destinazione.';
        END IF;
END;
```

5) Trigger: **ControlloPostiRichiesti**(regola aziendale **2**)

Questo trigger impedisce l'inserimento di una richiesta di corsa se il numero di posti richiesti è inferiore a **1** o superiore a **7**.

```
CREATE TRIGGER ControlloPostiRichiesti
BEFORE INSERT ON Richiesta
FOR EACH ROW
BEGIN
        IF NEW.NumeroPostiRichiesti < 1 OR NEW.NumeroPostiRichiesti > 7
        THEN SIGNAL SQLSTATE '45006'
        SET MESSAGE_TEXT = 'Il numero di posti richiesti deve essere compreso tra 1
e 7.';
        END IF;
END;
```

6) Trigger: **ControlloPostiTassista**(regola aziendale **1**)

Questo trigger impedisce l'inserimento di un nuovo tassista nel sistema se il numero di posti disponibili nel suo veicolo è inferiore a **1** o superiore a **7**.

```
CREATE TRIGGER ControlloPostiTassista
BEFORE INSERT ON Tassista
FOR EACH ROW BEGIN
    IF NEW.NumeroDiPostiADisposizione < 1 OR NEW.NumeroDiPostiADisposizione > 7
    THEN SIGNAL SQLSTATE '45005'
    SET MESSAGE_TEXT = 'Il numero di posti a disposizione deve essere compreso tra 1 e 7.';
    END IF;
END;
```

7) Trigger: **ControlloPostiDisponibili**(regola aziendale **12**, **13**)

Questo trigger viene attivato prima di aggiornare una richiesta nella tabella Richiesta quando lo stato della richiesta passa da "In attesa" ad "Accettata". Esso verifica che il numero di posti richiesti non superi quelli effettivamente disponibili nel veicolo del tassista e, se il controllo ha esito positivo, associa la richiesta a una corsa inserendo l'ID della richiesta nella tabella Corsa.

```
CREATE TRIGGER ControlloPostiDisponibili
BEFORE UPDATE ON Richiesta
FOR EACH ROW BEGIN
    IF NEW.StatoRichiesta = 'Accettata' AND OLD.StatoRichiesta = 'In attesa'
    THEN
        IF NEW.NumeroPostiRichiesti > (SELECT NumeroDiPostiADisposizione
        FROM Tassista
        WHERE NumeroDiPatente = NEW.Tassista)
        THEN SIGNAL SQLSTATE '45004'
        SET MESSAGE_TEXT = 'Numero di posti richiesti maggiore di quelli a disposizione del tassista.';
        END IF;
    INSERT INTO Corsa (IDRichiesta) VALUES (NEW.ID);
    END IF;
END;
```

8) Trigger: **ControlloClienteRichiesta**(regola aziendale **10**)

Questo trigger impedisce l'inserimento di una nuova richiesta di corsa se il cliente ha già una richiesta in attesa o è attualmente impegnato in una corsa attiva.

```
CREATE TRIGGER ControlloClienteRichiesta
BEFORE INSERT ON Richiesta
FOR EACH ROW
BEGIN
    IF EXISTS (SELECT 1 FROM ClienteOccupato WHERE Cliente = NEW.Cliente)
    THEN SIGNAL SQLSTATE '45001'
    SET MESSAGE_TEXT = 'Il cliente ha già una richiesta in attesa o è ancora occupato in una corsa.';
    END IF;
END;
```

9) Trigger: **ControlloTassistaRichiesta**(regola aziendale **11**)

Questo trigger impedisce l'inserimento di una nuova richiesta di corsa se il tassista selezionato è già impegnato in una corsa attiva. Viene eseguito prima dell'aggiornamento della tabella Richiesta e verifica se il tassista è già stato registrato come impegnato, nel caso in cui la richiesta stia passando allo stato "Accettata".

```
CREATE TRIGGER ControlloTassistaRichiesta
BEFORE UPDATE ON Richiesta
FOR EACH ROW BEGINIF NEW.StatoRichiesta = 'Accettata' AND EXISTS (SELECT 1 FROM TassistiImpegnati WHERE Tassista = NEW.Tassista LIMIT 1)
THEN SIGNAL SQLSTATE '45012'
SET MESSAGE_TEXT = 'Il tassista è già impegnato in una richiesta attiva o in attesa.';
END IF;
END //
```

- **Eventi**

1) Event: **ScadenzaRichiesta** (regola aziendale **11**)

Questo evento automatico segna come "Scadute" tutte le richieste che sono rimaste in stato "In attesa" per più di due minuti. L'evento si esegue ogni minuto e aggiorna la tabella Richiesta, cambiando lo stato delle richieste che non sono state accettate entro il periodo stabilito.

```
CREATE EVENT IF NOT EXISTS ScadenzaRichiesta
```

```
ON SCHEDULE EVERY 1 MINUTE
ON COMPLETION PRESERVE
COMMENT 'Segna come Scaduta le richieste rimaste in attesa per più di 2 minuti'
DO
   UPDATE Richiesta
   SET StatoRichiesta = 'Scaduta'
   WHERE StatoRichiesta = 'In attesa'
   AND TIMESTAMPDIFF (MINUTE, OrarioRichiesta, NOW ()) >= 2.
```

2) Event: **EliminaRichiesteScadute**

Questo evento si occupa di eliminare le richieste scadute che sono rimaste in stato "Scaduta" per più di un giorno. Viene eseguito una volta al giorno, garantendo che il database venga mantenuto pulito da richieste obsolete, liberando spazio e migliorando le performance.

```
CREATE EVENT IF NOT EXISTS EliminaRichiesteScadute
ON SCHEDULE EVERY 1 DAY
ON COMPLETION PRESERVE
COMMENT 'Elimina richieste scadute da più di un giorno'
DO
DELETE FROM Richiesta
WHERE StatoRichiesta = 'Scaduta' AND TIMESTAMPDIFF (DAY, OrarioRichiesta,
NOW()) >= 1;
```

3) Event: **ArchiviaRichiesteAccettate**

Questo evento si occupa di archiviare e successivamente eliminare le richieste accettate che sono state completate da più di 6 mesi. Le richieste vengono spostate in una tabella di archivio (RichiesteAccettateArchivio), in modo da non occupare spazio nel database principale, ma mantenendo la possibilità di recuperarle in futuro se necessario.

```
CREATE EVENT IF NOT EXISTS ArchiviaRichiesteAccettate
ON SCHEDULE EVERY 1 DAY
ON COMPLETION PRESERVE
COMMENT 'Sposta richieste accettate da più di 6 mesi nell archivio ed elimina' DO
BEGIN INSERT INTO RichiesteAccettateArchivio (ID, OrarioRichiesta, Cliente, ProvinciaPartenza, CittàPartenza, ViaPartenza, NumeroCivicoPartenza, ProvinciaDestinazione, CittàDestinazione, ViaDestinazione, NumeroCivicoDestinazione, NumeroPostiRichiesti, StatoRichiesta, Tassista, OrarioAccettazione)
SELECT ID, OrarioRichiesta, Cliente, ProvinciaPartenza, CittàPartenza, ViaPartenza, NumeroCivicoPartenza, ProvinciaDestinazione, CittàDestinazione, ViaDestinazione, NumeroCivicoDestinazione, NumeroPostiRichiesti, StatoRichiesta, Tassista, OrarioAccettazione
FROM Richiesta
WHERE StatoRichiesta = 'Accettata' AND TIMESTAMPDIFF (MONTH, OrarioAccettazione, NOW ()) >= 6;
DELETE FROM Richiesta WHERE StatoRichiesta = 'Accettata' AND TIMESTAMPDIFF (MONTH, OrarioAccettazione, NOW ()) >= 6;
END;
```

4) Event: **ArchiviaCorseRiscosse**

Questo evento si occupa di archiviare e successivamente eliminare le corse riscattate che sono state completate da più di sei mesi. Le corse vengono spostate in una tabella di archivio (CorseRiscosseArchivio), mantenendo la possibilità di recuperarle in futuro se necessario, ma senza occupare spazio nel database principale.

```
CREATE EVENT IF NOT EXISTS ArchiviaCorseRiscosse
ON SCHEDULE EVERY 1 DAY
ON COMPLETION PRESERVE
COMMENT 'Sposta corse riscattate da più di 6 mesi nell archivio ed elimina' DO
BEGIN
INSERT INTO CorseRiscosseArchivio (IDRichiesta, Durata, Importo, StatusRiscossione, Terminata)
SELECT IDRichiesta, Durata, Importo, StatusRiscossione, Terminata FROM Corsa
WHERE StatusRiscossione = 'Riscossa' AND TIMESTAMPDIFF (MONTH, OrarioAccettazione, NOW ()) >= 6; DELETE FROM Corsa WHERE StatusRiscossione = 'Riscossa' AND TIMESTAMPDIFF (MONTH, OrarioAccettazione, NOW ()) >= 6;
END;
```

5) Event: **EliminaRichiesteArchiviate**

Questo evento si occupa di rimuovere definitivamente le richieste archiviate che sono più vecchie di dieci anni dalla tabella di archivio (RichiesteAccettateArchivio). Questo processo assicura che i dati molto vecchi vengano eliminati dalla base di dati per evitare di occupare spazio inutilmente, rispettando comunque la politica di conservazione dei dati.

```
CREATE EVENT IF NOT EXISTS EliminaRichiesteArchiviate
ON SCHEDULE
EVERY 1 DAY
ON COMPLETION PRESERVE
COMMENT 'Rimuove richieste archiviate da più di 10 anni dalla base di dati'
DO DELETE FROM RichiesteAccettateArchivio
WHERE TIMESTAMPDIFF(YEAR, DataArchiviazione, NOW()) >= 10;
```

6) Event: **EliminaRichiesteArchiviate**

Questo evento si occupa di rimuovere definitivamente le corse riscattate archiviate che sono più vecchie di dieci anni dalla tabella di archivio (CorseRiscosseArchivio). Questo processo garantisce che il database non contenga dati obsoleti che non sono più necessari, contribuendo a mantenere il sistema ottimizzato e a ridurre lo spazio occupato dai dati storici.

```
CREATE EVENT IF NOT EXISTS EliminaCorseArchiviate
ON SCHEDULE EVERY 1 DAY
ON COMPLETION PRESERVE
COMMENT 'Rimuove corse riscattate archiviate da più di 10 anni dalla base di dati'
DO
  DELETE FROM CorseRiscosseArchivio
  WHERE TIMESTAMPDIFF(YEAR, DataArchiviazione, NOW()) >= 10
```

- **Viste**

Si è scelto di creare due **view** per ottimizzare le operazioni frequenti all'interno dei trigger "*ControlloClienteRichiesta*" e "*ControlloTassistaRichiesta*" per migliorare l'efficienza complessiva del sistema. Le operazioni che coinvolgono la verifica dello stato dei clienti e dei tassisti sono eseguite frequentemente; perciò, l'uso diretto di query nei trigger avrebbe potuto appesantire il loro tempo di esecuzione. Utilizzando le view, è stato possibile evitare di scrivere ripetutamente le stesse sotto-query nei trigger, semplificando e velocizzando l'elaborazione.

VIEW: **ClienteOccupato**

Questa view recupera i clienti che sono occupati con una richiesta in corso, sia che la loro richiesta sia ancora in attesa di accettazione o già accettata ma non ancora completata (corsa attiva). La view filtra le richieste in due casi:

- La richiesta è in attesa di essere accettata.
- La richiesta è accettata, ma la corsa associata non è ancora terminata.

La view viene utilizzata nel trigger che verifica se un cliente ha già una richiesta attiva o in attesa, impedendo così la creazione di nuove richieste da parte di un cliente già impegnato.

```
CREATE VIEW ClienteOccupato AS

SELECT R.Cliente

FROM Richiesta R

LEFT JOIN Corsa C ON R.ID = C.IDRichiesta

WHERE R.StatoRichiesta = 'In attesa'

  OR (R.StatoRichiesta = 'Accettata' AND C.Terminata = FALSE);
```

VIEW: **TassistiImpegnati**

La view TassistiImpegnati recupera l'elenco dei tassisti attualmente impegnati in una corsa. Viene selezionato il campo Tassista dalla tabella Richiesta (con la relazione alla tabella Corsa) con le seguenti condizioni:

- La richiesta deve essere accettata (StatoRichiesta = 'Accettata').
- La corsa associata alla richiesta non deve essere terminata (C.Terminata = FALSE) o, nel caso in cui non ci sia ancora una corsa terminata, la corsa potrebbe essere nulla (C.Terminata IS NULL).

Questa view permette di identificare rapidamente i tassisti che sono impegnati in una corsa in corso e, di conseguenza, impedire che vengano assegnati a nuove richieste finché non hanno completato la corsa attuale.

```
CREATE VIEW TassistiImpegnati AS

SELECT R.Tassista

FROM Richiesta R LEFT JOIN Corsa C ON R.ID = C.IDRichiesta

WHERE R.StatoRichiesta = 'Accettata' AND (C.Terminata = FALSE);
```

- **Stored Procedures e transazioni**

  - **OPERAZIONE L1 (LOGIN):**

```
DROP PROCEDURE IF EXISTS Login;

DELIMITER //

CREATE PROCEDURE Login (IN p_username VARCHAR (50),  IN p_password VARCHAR (255), OUT p_ruolo_int TINYINT (1 = Cliente, 2 = Tassista, 3 = Gestore))

BEGIN

DECLARE p_ruolo VARCHAR (50);
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
  ROLLBACK; -- Annulla la transazione in caso di errore
```

```
    RESIGNAL; -- Rilancia l'errore per essere gestito dal codice dell'applicazione
END;
START TRANSACTION;


SELECT Ruolo INTO p_ruolo
FROM Utente
WHERE Username = p_username AND PasswordHash = SHA2(p_password, 256)
LIMIT 1;

IF p_ruolo = 'Cliente' THEN
   SET p_ruolo_int = 1;
ELSEIF p_ruolo = 'Tassista' THEN
   SET p_ruolo_int = 2;
ELSEIF p_ruolo = 'Gestore' THEN
   SET p_ruolo_int = 3;
ELSE
   SET p_ruolo_int = NULL; -- Se l'utente non esiste o credenziali errate
END IF;

-- Se non ci sono errori, esegui il commit
COMMIT;
 END //

DELIMITER;
```

- **OPERAZIONE CL1 (REGISTRAZIONE CLIENTE):**

```
DROP PROCEDURE IF EXISTS RegistrazioneCliente;
DELIMITER //
CREATE PROCEDURE RegistrazioneCliente( IN p_username VARCHAR (50), IN p_password VARCHAR (255), IN p_numero_telefono VARCHAR (15), IN p_nome VARCHAR (50), IN p_cognome VARCHAR (50), IN p_numero_di_carta_di_credito VARCHAR (19))
BEGIN DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
```

```
ROLLBACK; -- Annulla la transazione in caso di errore
RESIGNAL; -- Rilancia l'errore per essere gestito dal codice dell'applicazione
END;
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
START TRANSACTION;
IF (SELECT COUNT(*)
  FROM (
    SELECT Username AS Valore FROM Utente WHERE Username = p_username
UNION ALL SELECT NumeroDiTelefono FROM Cliente WHERE NumeroDiTelefono = p_numero_telefono
UNION ALL SELECT NumeroDiCartaDiCredito FROM Cliente WHERE NumeroDiCartaDiCredito = p_numero_di_carta_di_credito) AS Duplicati) > 0 THEN
ROLLBACK;
 SELECT 'Errore: Username, Numero di Telefono o Carta di Credito già registrati!' AS esito;
ELSE-- Inserisce l'utente nella tabella Utente
  INSERT INTO Utente (Username, PasswordHash, Ruolo)
  VALUES (p_username, SHA2(CONCAT(p_password, 'SALT1234'), 256), 'Cliente');
  -- Inserisce il cliente nella tabella Cliente
  INSERT INTO Cliente (NumeroDiTelefono, Nome, Cognome, NumeroDiCartaDiCredito, Username)
  VALUES (p_numero_telefono, p_nome, p_cognome, p_numero_di_carta_di_credito, p_username);

  COMMIT; -- Completamento della transazione
  SELECT 'Registrazione completata con successo!' AS esito;
END IF;
END //

DELIMITER;
```

- **OPERAZIONE TS1 (REGISTRAZIONE TASSISTA):**

```
DROP PROCEDURE IF EXISTS RegistrazioneTassista;
DELIMITER //

CREATE PROCEDURE RegistrazioneTassista( IN p_username VARCHAR(50), IN p_password VARCHAR(255), IN p_numero_patente VARCHAR(20), IN p_nome VARCHAR(50), IN p_cognome VARCHAR(50), IN p_numero_di_carta_di_credito VARCHAR(19), IN p_targa VARCHAR(7), IN p_numero_di_posti_adisposizione SMALLINT ) BEGIN -- Dichiarazione gestione errori
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK; -- Annulla la transazione in caso di errore
RESIGNAL; -- Rilancia l'errore per essere gestito dall'applicazione END;
-- Imposta il livello di isolamento della transazione
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

```
START TRANSACTION;

-- Controlla se Username, Targa o Numero di Patente esistono già in un’unica query
IF (SELECT COUNT(*)
  FROM (
    SELECT Username AS Valore FROM Utente WHERE Username = p_username
    UNION ALL
    SELECT Targa FROM Tassista WHERE Targa = p_targa
    UNION ALL
    SELECT NumeroDiPatente FROM Tassista WHERE NumeroDiPatente = p_numero_patente
  ) AS Duplicati) > 0 THEN
  ROLLBACK; -- Annulla la transazione se ci sono duplicati
  SELECT 'Errore: Username, Targa o Numero di Patente già registrati!' AS esito;
ELSE
  -- Inserisce l'utente nella tabella Utente
  INSERT INTO Utente (Username, PasswordHash, Ruolo)
  VALUES (p_username, SHA2(CONCAT(p_password, 'SALT1234'), 256), 'Tassista');

  -- Inserisce il tassista nella tabella Tassista
  INSERT INTO Tassista (NumeroDiPatente, Nome, Cognome, NumeroDiCartaDiCredito, Targa, NumeroDiPostiADisposizione, Username)
  VALUES (p_numero_patente, p_nome, p_cognome, p_numero_di_carta_di_credito, p_targa, p_numero_di_posti_adisposizione, p_username);

  COMMIT; -- Completamento della transazione
  SELECT 'Registrazione completata con successo!' AS esito;
END IF;

END //

DELIMITER;
```

- **OPERAZIONE CL2 (RICHIEDI CORSA):**

```
DELIMITER $$
DROP PROCEDURE IF EXISTS RichiediCorsa $$
CREATE DEFINER=root@localhost PROCEDURE RichiediCorsa( IN p_cliente VARCHAR(15), IN p_provincia_partenza CHAR(2), IN p_citta_partenza VARCHAR(50), IN p_via_partenza VARCHAR(50), IN p_numero_civico_partenza VARCHAR(10), IN p_provincia_destinazione CHAR(2), IN p_citta_destinazione VARCHAR(50), IN p_via_destinazione VARCHAR(50), IN p_numero_civico_destinazione VARCHAR(10), IN p_numero_posti_richiesti SMALLINT, OUT p_esito VARCHAR(255) )
BEGIN -- Gestione degli errori DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
ROLLBACK; -- Annulla la transazione in caso di errore SET p_esito = 'Errore durante la registrazione della corsa, riprova!';
RESIGNAL; -- Rilancia l'errore per la gestione a livello di applicazione END;-- Imposta il livello di isolamento
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

-- Inizia la transazione
START TRANSACTION;

-- Inserisci l'indirizzo di partenza se non esiste già
INSERT IGNORE INTO indirizzo (
  Provincia, Città, Via, NumeroCivico) VALUES (
  p_provincia_partenza, p_citta_partenza, p_via_partenza, p_numero_civico_partenza
);

-- Inserisci l'indirizzo di destinazione se non esiste già
INSERT IGNORE INTO indirizzo (
  Provincia, Città, Via, NumeroCivico
) VALUES (
  p_provincia_destinazione, p_citta_destinazione, p_via_destinazione, p_numero_civico_destinazione
);

-- Inserisci la richiesta con stato 'In attesa'
INSERT INTO richiesta (
  Cliente,
  ProvinciaPartenza, CittàPartenza, ViaPartenza, NumeroCivicoPartenza,
  ProvinciaDestinazione, CittàDestinazione, ViaDestinazione, NumeroCivicoDestinazione,
  NumeroPostiRichiesti, StatoRichiesta
)
VALUES (
  p_cliente,
  p_provincia_partenza, p_citta_partenza, p_via_partenza, p_numero_civico_partenza,
  p_provincia_destinazione, p_citta_destinazione, p_via_destinazione, p_numero_civico_destinazione,
  p_numero_posti_richiesti, 'In attesa'
```

```
    );

    -- Commit della transazione
    COMMIT;

    -- Esito positivo
    SET p_esito = 'Richiesta di corsa registrata con successo!';

END $$

DELIMITER;
```

- **OPERAZIONE TS2 (VISUALIZZA RICHIESTE DI CORSA IN ATTESA):**

```
DELIMITER //
```

```
CREATE PROCEDURE VisualizzaRichiesteInAttesa()
BEGIN
    -- Gestione degli errori
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK; -- Annulla la transazione in caso di errore
        SELECT 'Errore durante il recupero delle richieste, riprova!' AS esito;
        RESIGNAL; -- Rilancia l'errore per la gestione a livello di applicazione
    END;

    -- Imposta il livello di isolamento a REPEATABLE READ
    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
    -- Inizia la transazione
    START TRANSACTION
SELECT * FROM richiesta WHERE StatoRichiesta = 'In attesa';
    -- Commit della transazione (anche se solo SELECT, ma buona pratica)
    COMMIT;
END //
DELIMITER;
```

- **OPERAZIONE TS3 (ACCETTA RICHIESTA DI CORSA):**

```
DELIMITER $$
DROP PROCEDURE IF EXISTS AccettaRichiesta $$
CREATE PROCEDURE AccettaRichiesta( IN p_tassista VARCHAR (20), -- Numero di patente del tassista IN p_id_richiesta INT, -- ID della richiesta da accettare OUT p_esito VARCHAR (255) -- Messaggio di esito)
BEGIN DECLARE richiesta_esiste INT;
-- Gestione degli errori con un EXIT HANDLER per SQLEXCEPTION
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK; -- Annulla la transazione in caso di errore
  SET p_esito = 'Errore durante l'accettazione della richiesta. Riprovare.';
RESIGNAL; -- Rilancia l'errore per la gestione a livello di applicazione
END;

-- Imposta il livello di isolamento a REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

-- Inizia la transazione
START TRANSACTION;

-- Controlla se la richiesta è ancora "In attesa"
SELECT COUNT(*) INTO richiesta_esiste
FROM richiesta
WHERE ID = p_id_richiesta AND StatoRichiesta = 'In attesa'
FOR UPDATE; -- Blocca la riga per evitare modifiche simultanee

IF richiesta_esiste = 0 THEN
  SET p_esito = 'Errore: richiesta non disponibile o già accettata.';
  ROLLBACK; -- Annulla la transazione se la richiesta non è disponibile
ELSE
  -- Aggiorna lo stato della richiesta e assegna il tassista
  UPDATE richiesta
  SET StatoRichiesta = 'Accettata',
    Tassista = p_tassista,
    OrarioAccettazione = NOW()
  WHERE ID = p_id_richiesta;

  -- Commit della transazione
  COMMIT;
  SET p_esito = 'Richiesta accettata con successo!';
END IF;

END $$
```

DELIMITER;

- **OPERAZIONE TS3 (INSERISCI IMPORTO):**

DELIMITER $$

CREATE PROCEDURE **TerminaCorsa**(IN p_id_corsa INT, -- ID della corsa da terminare IN p_importo DECIMAL (10, 2), -- Importo da inserire per la corsa OUT p_esito VARCHAR (255) -- Esito dell'operazione)

BEGIN -- Dichiarazione del gestore di eccezioni per SQLEXCEPTION DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN -- Esegui il rollback in caso di errore

ROLLBACK; -- Rilancia l'errore per essere gestito dal codice dell'applicazione

RESIGNAL; END;

-- Imposta il livello di isolamento a READ COMMITTED
SET SESSION TRANSACTION ISOLATION LEVEL **READ COMMITTED**;

-- Inizia la transazione
START TRANSACTION;

-- Aggiorna l'importo della corsa
UPDATE Corsa
SET Importo = p_importo
WHERE IDRichiesta = p_id_corsa AND Terminata = 0;

-- Verifica se la corsa è stata aggiornata
SELECT COUNT(*) INTO @count
FROM Corsa

```
WHERE IDRichiesta = p_id_corsa AND Terminata = 0;


-- Se la corsa non è stata aggiornata, effettua un rollback
IF @count = 0 THEN
   -- Se non è stata aggiornata (perché è già terminata o non esiste)
   ROLLBACK;
   SET p_esito = 'Errore: la corsa è già terminata o non esiste.';
ELSE
   -- Altrimenti, calcola la durata e imposta "Terminata" a TRUE
   UPDATE Corsa
   SET Terminata = TRUE,
      Durata = TIMESTAMPDIFF(MINUTE, (SELECT OrarioAccettazione FROM Richiesta
WHERE ID = p_id_corsa), NOW())
   WHERE IDRichiesta = p_id_corsa;


   -- Se l'aggiornamento è andato a buon fine, esegui il commit
   COMMIT;
   SET p_esito = 'Corsa terminata con successo!';
END IF;
 END $$

DELIMITER;
```

- **OPERAZIONE GS3 (RISCUOTI COMMISSIONE):**

```
DELIMITER //
DROP PROCEDURE IF EXISTS RiscuotiCommissione;
CREATE PROCEDURE RiscuotiCommissione( IN p_id_corsa INT, -- ID della corsa da
riscossare OUT p_esito VARCHAR (255) -- Esito dell'operazione)
BEGIN DECLARE v_importo DECIMAL (10,2); DECLARE v_commissione
DECIMAL(10,2); DECLARE errore INT DEFAULT 0;
-- Gestione degli errori con un EXIT HANDLER per SQLEXCEPTION
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK; -- Annulla la transazione in caso di errore
   SET p_esito = 'Errore durante l\'operazione di riscossione. Riprovare.';
RESIGNAL; -- Rilancia l'errore per la gestione a livello di applicazione
END;
```

```
    -- Imposta il livello di isolamento per garantire consistenza
    SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;

    -- Inizia la transazione
    START TRANSACTION;

    -- Recupera l'importo della corsa
    SELECT Importo INTO v_importo
    FROM corsa
    WHERE IDRichiesta = p_id_corsa
    FOR UPDATE;

    -- Controllo se la query ha restituito un valore valido
    IF v_importo IS NULL THEN
      SET errore = 1;
      SET p_esito = 'Errore: Importo non valido.';
    ELSE
      -- Calcola la commissione (3% dell'importo)
      SET v_commissione = v_importo * 0.03;

      -- Aggiorna lo stato della riscossione a "Riscossa"
      UPDATE corsa
      SET StatusRiscossione = 'Riscossa'
      WHERE IDRichiesta = p_id_corsa;

      -- Controlliamo se l'UPDATE ha avuto effetto
      IF ROW_COUNT() = 0 THEN
        SET errore = 1;
        SET p_esito = 'Errore: Nessuna corsa trovata o già riscossa.';
      ELSE
        SET errore = 0;
      END IF;
    END IF;

    -- Se ci sono stati errori, annulla la transazione
    IF errore = 1 THEN
      ROLLBACK;
    ELSE
      COMMIT;
      SET p_esito = CONCAT('Corsa riscossa con successo! Commissione: ', FORMAT
    (v_commissione, 2), ' EUR');
    END IF;

END //
```

DELIMITER;

- **OPERAZIONE GS1 (REPORT TASSISTI):**

```
DELIMITER $$
BEGIN
 -- Dichiarazione del gestore di eccezioni per SQLEXCEPTION
DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
-- Esegui il rollback in caso di errore
ROLLBACK;
-- Rilancia l'errore per essere gestito dal codice dell'applicazione
RESIGNAL;
END;
-- Impostiamo il livello di isolamento massimo per questa sessione
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE.
-- Inizia la transazione
START TRANSACTION;
-- Eseguiamo la query di aggregazione
SELECT
 t.NumeroDiPatente,        -- Numero di patente (primary key
t.Nome,                   -- Nome del tassista
t.Cognome,                -- Cognome del tassista
COUNT(c.IDRichiesta) AS NumeroCorse,
COALESCE(SUM(c.Importo), 0) AS GuadagnoTotale,   -- Se non ci sono corse, 0 come somma
COALESCE (SUM (c.Importo * 0.03), 0) AS CommissioneTotale  -- Commissione del 3%, 0 se
non ci sono corse
FROM tassista LEFT JOIN richiesta r ON t.NumeroDiPatente = r.Tassista LEFT JOIN corsa c
ON r.ID = c.IDRichiesta AND c.Terminata = 1 AND c.StatusRiscossione = 'Riscossa'
GROUP BY t.NumeroDiPatente; -- Se tutto è andato a buon fine, eseguiamo il commit
COMMIT;
END $$
DELIMITER;
```

- **OPERAZIONE GS2 (VISUALIZZA CORSA NON RISCOSSE):**

```
DELIMITER $$
```

```
-- Elimina la stored procedure esistente, se presente
DROP PROCEDURE IF EXISTS MostraCorseNonRiscosse $$

CREATE PROCEDURE MostraCorseNonRiscosse()
BEGIN
    -- Dichiarazione del gestore di eccezioni per SQLEXCEPTION
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Esegui il rollback in caso di errore
        ROLLBACK;
        -- Rilancia l'errore per essere gestito dal codice dell'applicazione
        RESIGNAL;
    END;

    -- Impostiamo il livello di isolamento per lettura
    SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

    -- Inizia la transazione
    START TRANSACTION;

    -- Selezioniamo le corse non riscosse e terminate
    SELECT
        IDRichiesta,        -- Identificativo della corsa
        Durata,             -- Durata della corsa
        Importo,            -- Importo totale della corsa
        StatusRiscossione,  -- Stato riscossione (Riscossa/Non Riscossa)
        Terminata           -- Stato della corsa (1 = Terminata)
    FROM
        corsa
    WHERE
        StatusRiscossione = 'Non Riscossa'  -- Solo corse non ancora riscosse
        AND Terminata = 1;  -- Solo corse che sono già terminate

    -- Se tutto è andato a buon fine, eseguiamo il commit
    COMMIT;
END $$

DELIMITER;
```