# 1 Abstract

Si vuole rinnovare la base di dati di `Easy Travel`, un'applicazione che offre ai suoi clienti la possibilità di cercare e confrontare diversi pacchetti di viaggio proposti dalle varie agenzie. Il precedente sistema contava all'incirca 80.000 utenti, la nuova base di dati deve supportare un nuovo potenziale bacino di 600.000 utenti. All'intero dell'applicazione esistono tre attori principali: i clienti, le agenzie e le compagnie aeree.

Il cliente può: visualizzare le soluzioni di viaggio disponibili, prenotare dei pacchetti, scegliere i voli più convenienti o adatti per le sue esigenze, e vedere lo storico degli acquisti. Durante la scelta dei voli il sistema consiglia quelli più convenienti basandosi sui dati presenti nel database. Una agenzia fornisce: i pacchetti che il cliente può prenotare, e tutte le informazioni essenziali per quel pacchetto: informazioni sull'alloggio e descrizioni testuali utili a dare un idea del viaggio. Ogni pacchetto può avere una polizza assicurativa fornita da `Esay Travel`. Vengono registrati i voli proposti dalle compagnie aeree che collaborano con il servizio, così da poter calcolare durante la prenotazione il volo più conveniente, ma comunque lasciando all'utente la scelta finale.

# 2 Raccolta e analisi dei requisiti

## 2.1 Proprietà del sistema

Gli **utenti** registrati nel sistema vengono identificati dalla loro email scelta durante la fase di registrazione. Di ogni utente vengono memorizzati i seguenti dati: la password e la data di iscrizione. Gli utenti si specializzano in tre categorie: i *clienti*, le *agenzie* e le *compagnie di volo*. Ogni **cliente** deve fornire le seguenti informazioni: il nome, il cognome, la data di nascita, in modo facoltativo il sesso e il numero di telefono. Di ogni **agenzia** viene riportato: la denominazione e la sede legale con l'indirizzo. Ogni **compagnia di volo** riporta: il nome, il codice internazionale ICAO e i voli che mette a disposizione per il servizio.

Ogni agenzia può offrire diversi *pacchetti viaggio*. Per ogni **pacchetto** vengono salvati i seguenti dati: la data di partenza, la data di ritorno, la disponibilità[1], il massimo numero di persone che possono partecipare al viaggio[2], il prezzo di base[3] per persona e la destinazione. Ogni pacchetto riporta i dati dell'**alloggio**, identificato dal suo nome e dalla città in cui è ubicato con l'indirizzo, in più viene riportata la tipologia di struttura[4], e se disponibile il numero di stelle. Sia il pacchetto di viaggio e sia l'alloggio hanno una **descrizione** testuale che viene identificata nel sistema da un ID, inoltre viene riportato: un titolo della descrizione o del viaggio e un testo.

Ogni compagnia aerea gestisce dei voli. Un **volo** è identificati dal codice di volo e riporta le seguenti informazioni: la classe, come va fatto il check-in e il prezzo per persona. Per ogni volo vengono riportati anche le **informazioni** riguardanti i **bagagli**: se concesso, quanto può pesare al massimo il bagaglio da mettere in stiva, e se concesso quanto può pesare al massimo il bagaglio da portare a mano. Ogni volo ha una **aeroporto** di partenza e un arrivo identificati dal loro codice internazionale, inoltre viene tenuta traccia dell'ora e della data di partenza stimata, e dell'ora e della data di arrivo stimata. Di ogni **aeroporto** viene salvata la sua ubicazione.

Un cliente può scrivere un *recensione* per l'alloggio alla fine del viaggio. Le **recensioni** sono identificate da un ID interno e riportano: un giudizio con una scala da 0 a 5, la data di quando è stata scritta la recensione e una motivazione testuale che può essere facoltativa. Un cliente può prenotare un pacchetto,

1 Quanti utenti al massimo possono comprare quel pacchetto.

2 Esempio: pacchetto famiglia da massimo 4 persone.

3 Senza contare il costo dei mezzi di trasporto per l'andata e il ritorno.

4 Esempio: hotel, bed & breakfast, eccetera.

della **prenotazione** vengono salvati: il numero di persone che partecipano al viaggio, i dati della transizione di pagamento e le informazioni per il trasporto (sia per l'andata che per il ritorno).

Una **transizione** riporta: un codice identificativo, la banca che ha preso in carico l'operazione, l'importo totale, il circuito usato e il timestamp in cui è avvenuta l'operazione. Per ogni prenotazione si può sceglie anche l'offerta più convenite di trasporto. Le **informazioni di trasporto** riportano: il prezzo totale e le varie tratte per l'andata e per ritorno. Le tratte sono rappresentate dai voli. Ogni luogo è riconosciuto dal sistema come una **città** identificata da un codice interno e vengono salvati: il nome e il paese dove si trova la città.

## 2.2 Glossario dei termini

*Table I: Glossario dei termini*

| Termine | Descrizione | Sinonimo | Collegamenti |
|---|---|---|---|
| *Utente* | Utente generico iscritto al sistema | | Cliente, Agenzia |
| *Cliente* | Specializzazione di un utente. Usufruiscono del servizio. | | Utente |
| *Agenzia* | Specializzazione di un utente. Può inserire delle soluzioni di viaggio nel sistema. | | Utente, Pacchetto |
| *Compagnia di volo* | Specializzazione di un utente. Può inserire dei voli nel sistema. | Compagnia aerea, Compagnia | Utente, Volo |
| *Pacchetto di viaggio* | Soluzione di viaggio offerta da una agenzia. | Pacchetto, Soluzione di viaggio | Utente, Agenzia, Alloggio, Descrizione |
| *Alloggio* | Struttura che ospita il cliente durante la vacanza. | Soggiorno | Città, Descrizione, Pacchetto viaggio |
| *Descrizione* | Descrizione testuale di un alloggio oppure di un pacchetto. | | Pacchetto, Alloggio |
| *Recensione* | Giudizio del cliente sull'alloggio offerto. | | Cliente, Alloggio |
| *Prenotazione* | Acquisto con esito positivo di un pacchetto. | | Cliente, Pacchetto |
| *Transizione* | Pagamento avvenuto con successo. | | Prenotazione |
| *Polizza assicurativa* | Assicurazione per il viaggio. | Polizza | Pacchetto Viaggio |
| *Informazioni di trasporto* | Informazioni riguardanti i voli da prendere per andare e tornare dal viaggio. | | Prenotazione, Volo |
| *Aeroporto* | Luogo di partenza e arrivo degli aeri. | | Città, Volo |
| *Volo* | Volo per arrivare a destinazione o tornare. | Tratte | Informazioni bagaglio, Aeroporto, Informazioni di trasporto Compagnia |
| *Informazioni bagaglio* | Informazioni utili al cliente sulle politiche usate per la gestione dei bagagli per un volo. | | Volo |
| *Città* | Luogo fisico | | Alloggio, Aeroporto |

## 2.3 Operazioni

Nel caso d'uso perso in esame il numero di operazioni effettuate non hanno una distribuzione uniforme durante tutto l'anno, ma alcune operazioni in particolare presentano un numero di richieste maggiore durante i periodi di vacanza, cioè durante i periodi di massimo carico per il sistema, mentre in altri periodi

ci sono dei momenti di *idle*. Ipotizziamo di seguito per le operazioni più importanti la loro frequenza nell'ipotesi di massimo carico del sistema.

*Table II: operazioni e costi*

| Numero operazione | Operazione | Descrizione | Numero operazioni (tempo/operazione)[5] |
|---|---|---|---|
| *1* | Inserimento pacchetto | Inserimento di un pacchetto da parte di un'agenzia. | 30 o/dd |
| *2* | Inserimento volo | Inserimento di un volo da parte di una compagnia aerea. | 12 000 o/dd |
| *3* | Inserimento cliente | Un nuovo cliente si iscrive al servizio. | 1 500 o/dd |
| *4* | Ricerca pacchetti | Consultazione dei pacchetti disponibili. | 260 000 o/dd |
| *5* | Prenotazione pacchetto | Un cliente compra una soluzione viaggio. | 10 000 o/dd |
| *6* | Controllo storico acquisti | Un cliente controlla lo storico degli acquisti. | 3 000 o/mm |

# 3 Progettazione concettuale

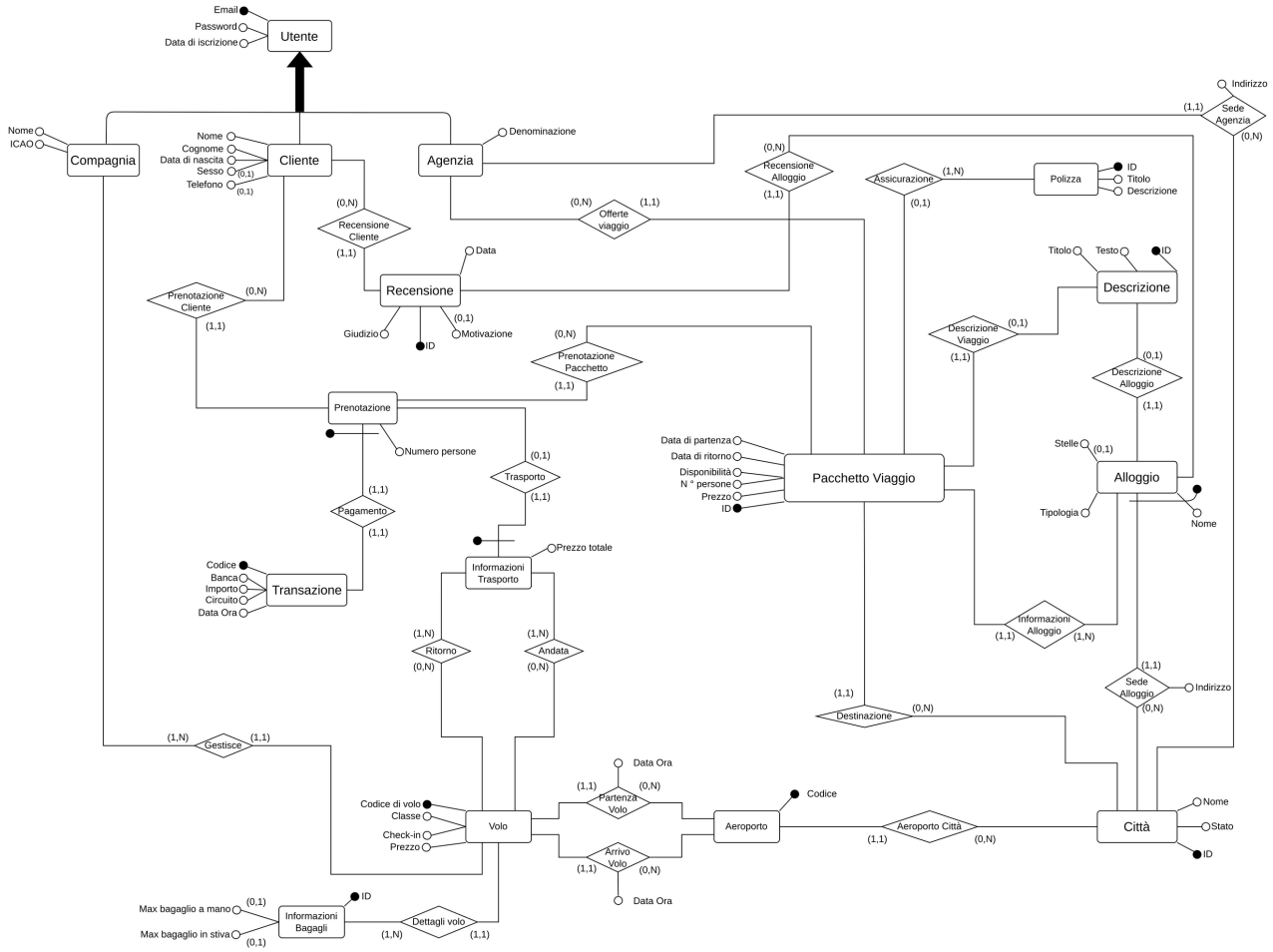


*Figure 1: E-R concettuale. Per lo sviluppo dell'E-R è stata usata la tecnica inside-out partendo dal concetto di Pacchetto Viaggio.*

5 Riportiamo le misure di tempo: *dd* = giorni, *mm* = mesi e *yy* = anni.

## 3.1 Descrizione entità E-R

In riferimento all'E-R concettuale in figura 2, usiamo le seguenti convenzioni:

- gli attributi chiave sono sottolineati;
- tutti gli attributi, a meno che non si specificato, non ammettono il valore `NULL`;
- usiamo la seguente notazione matematica per gli intervalli $\in [a, b]$;
- con la seguente simbologia indichiamo che l'entità A è padre dell'entità B: A $\twoheadrightarrow$ B, per l'entità figlie non riportiamo la chiave primaria perché è quella del padre;
- gli attributi di marcatura temporale sono tutti senza time zone.

*Table III: entità E-R concettuale*

| **Entità** | **Attributi** | **Tipo** | **Vincoli / Altro** |
|---|---|---|---|
| *Utente* | Email | varchar(40) | |
| | Password | varchar(24) | length(password) $\in$ [8,24] |
| | Data di iscrizione | timestamp | |
| Utente $\twoheadrightarrow$ *Cliente* | Nome | varchar(30) | |
| | Cognome | varchar(30) | |
| | Data di nascita | date | |
| | Sesso | enum('M','F') | Può essere **NULL** |
| | Numero telefonico | varchar(20) | Può essere **NULL**, **UNIQUE** |
| Utente $\twoheadrightarrow$ *Agenzia* | Denominazione | varchar(40) | |
| Utente $\twoheadrightarrow$ *Compagnia* | Nome | varchar(40) | |
| | ICAO | char(4) | |
| Recensione | ID | integer | |
| | Giudizio | numeric(1,0) | $\in$ [0,5] |
| | Motivazione | varchar(350) | Può essere **NULL** |
| | Data | date | |
| Prenotazione | Codice (Transazione) | varchar(16) | length(password) = 16 |
| | Numero persone | numeric(2,0) | $\geq 1$ |
| Transizione | Codice | varchar(16) | length(password) = 16 |
| | Banca | varchar(30) | |
| | Importo | numeric(7,2) | $\geq 0$ |
| | Circuito | varchar(20) | |
| | Data ora | timestamp | |
| Pacchetto Viaggio | ID | integer | |
| | Data di partenza | date | |
| | Data di ritorno | date | $>$ *Data di partenza* |
| | Disponibilità | numeric(2,0) | $\geq 1$ |
| | N° persone | numeric(2,0) | $\geq 1$ |
| | Prezzo | numeric(7,2) | $\geq 0$ |
| Descrizione | ID | integer | |
| | Titolo | varchar(40) | |
| | Testo | varchar(400) | |
| Alloggio | Nome | varchar(40) | |
| | ID (Città) | integer | |
| | Stelle | integer | $\in$ [1,5] e può essere **NULL** |
| | Tipologia | varchar(20) | |
| Città | ID | integer | |
| | Nome | varchar(40) | |
| | Stato | varchar(35) | |
| Polizza | ID | integer | |
| | Descrizione | varchar(4000) | |
| | Titolo | varchar(30) | |
| Informazioni trasporto | Codice (Transizione) | timestamp | length(password) = 16 |
| | Prezzo totale | numeric(7,2) | $\geq 0$ |
| Volo | Codice | integer | |
| | Classe | varchar(20) | |
| | Check-in | varchar(15) | |

| | | | |
|---|---|---|---|
| | Prezzo | numeric(7,2) | $\geq 0$ |
| Informazioni Bagagli | ID | integer | |
| | Max bagaglio a mano | numeric(2,0) | $\geq 0$ e può essere `NULL` |
| | Max bagaglio in stiva | numeric(2,0) | $\geq 0$ e può essere `NULL` |
| Aeroporto | Codice | varchar(4) | |

## 3.2 Descrizione relazioni E-R

*Table IV: relazioni E-R concettuale.*

| Relazione | Entità coinvolte | Descrizione | Attributi |
|---|---|---|---|
| *Offerte Viaggio* | Agenzia (0,N)<br>Pacchetto Viaggio (1,1) | Una agenzia può offrire da 0 a N pacchetti viaggio. Un pacchetto viene offerto da una sola agenzia. | |
| *Recensione Cliente* | Cliente (0,N)<br>Recensione (1,1) | Un cliente può scrivere da 0 a N recensioni. Ogni recensione deve fare riferimento a un solo cliente. | |
| *Recensione Alloggio* | Recensione (1,1)<br>Alloggio (0,N) | Una recensione deve fare sempre fare riferimento ad un unico alloggio, mentre un alloggio può avere da 0 a N recensioni. | |
| *Assicurazione* | Pacchetto Viaggio (0,1)<br>Polizza (1,N) | Un pacchetto viaggio può avere una polizza assicurativa. Una polizza assicurativa può essere usata per più pacchetti oppure per uno soltanto. | |
| *Descrizione Viaggio* | Pacchetto Viaggio (1,1)<br>Descrizione (0,1) | Ogni pacchetto viaggio deve avere una descrizione. Una descrizione non necessariamente deve fare riferimento a un pacchetto. | |
| *Descrizione Alloggio* | Descrizione (0,1)<br>Alloggio (1,1) | Ogni alloggio deve avere una descrizione. Una descrizione non necessariamente deve fare riferimento a un alloggio. | |
| *Destinazione* | Pacchetto viaggio (1,1)<br>Città (0,N) | Un pacchetto ha una sola destinazione. Una città può essere più da 0 a N volte destinazione di un viaggio. | |
| *Informazioni Alloggio* | Pacchetto Viaggio (1,1)<br>Alloggio (1,N) | Ogni pacchetto viaggio deve riferirsi ad un alloggio. Un alloggio può essere riferito da più pacchetti. | |
| *Sede Alloggio* | Alloggio (1,1)<br>Città (0,N) | Ogni alloggio ha una sola ubicazione. Una città può essere sede di più alloggi oppure di nessun alloggio. | Indirizzo – varchar(40) |
| *Sede Agenzia* | Agenzia (1,1)<br>Città (0,N) | Un agenzia ha una sola città come sede. Una città può essere la sede di più agenzia oppure di nessuna. | Indirizzo – varchar(40) |
| *Prenotazione Cliente* | Cliente (0,N)<br>Prenotazione (1,1) | Un cliente può eseguire da 0 a N prenotazioni. Ogni prenotazione deve riferirsi a un cliente. | |
| *Prenotazione Pacchetto* | Prenotazione (1,1)<br>Pacchetto Viaggio (0,N) | Una prenotazione deve riferirsi a un pacchetto. Un pacchetto può apparire da 0 a N volte in una prenotazione. | |
| *Pagamento* | Prenotazione (1,1)<br>Transizione (1,1) | Ogni prenotazione ha un transizione e viceversa | |
| *Trasporto* | Prenotazione (0,1) | Una prenotazione può usufruire | |

| | | | |
|---|---|---|---|
| | Informazioni trasporto (1,1) | del trasporto. Le informazioni di trasporto devono fare riferimento ad una prenotazione. | |
| *Ritorno* | Informazioni trasporto (1,N)<br>Volo (0,1) | Una informazioni di trasporto può avere da 1 a più voli di ritorno. Un volo può apparire da 0 a N volte. | |
| *Andata* | Informazioni trasporto (1,N)<br>Volo (0,1) | Una informazioni di trasporto può avere da 1 a più voli di andata. Un volo può apparire da 0 a N volte. | |
| *Gestisce* | Compagnia (1,N)<br>Volo (1,1) | Una compagnia può gestire più voli. Ogni volo deve essere gestito da una compagnia. | |
| *Dettagli Volo* | Informazioni Bagaglio (1,N)<br>Volo (1,1) | Ogni volo ha le informazioni riguardanti il trasporto dei bagagli. Ogni informazione sui bagagli può essere riferita a più voli. | |
| *Partenza Volo* | Volo (1,1)<br>Aeroporto (0,N) | Ogni volo ha un unico aeroporto di partenza. Un aeroporto può essere punto di partenza per N voli, ma anche per nessun volo. | Data ora – timestamp |
| *Arrivo Volo* | Volo (1,1)<br>Aeroporto (0,N) | Ogni volo ha un unico aeroporto di arrivo. Un aeroporto può essere punto di partenza per N voli, ma anche per nessun volo. | Data ora – timestamp |
| *Aeroporto Città* | Aeroporto (1,1)<br>Città (0,N) | Ogni aeroporto è situato in una città. Una città può ospitare da 0 a N aeroporti. | |

## 3.3 Regole aziendali[6]

Non riportiamo quelle già espresse nella Table III:

**Regole di vincolo (RV):**

1. un volo non deve avere nella associazione arrivo l'attributo *data ora* inferiore alla *data ora* di partenza;
2. un volo non deve avere l'aeroporto di partenza e di arrivo che coincidono;
3. nella relazione andata e ritorno, in associazione con l'entità informazioni trasporto, non devono esserci dei voli che vengono effettuati contemporaneamente per lo stesso viaggio;
4. un cliente non deve scrivere una recensione per un alloggio non presente nello storico dei pacchetti viaggio che ha prenotato;
5. un cliente non deve scrivere un recensione con la *data* di inserimento della recensione inferiore a quella della data di fine del viaggio prenotato;
6. l'attributo *numero di persone* nell'entità prenotazione non deve essere superiore al valore del *N° persone* del pacchetto viaggio prenotato;
7. una descrizione non deve essere in relazione allo stesso tempo sia con il pacchetto viaggio e sia con l'alloggio;
8. la destinazione del pacchetto non deve differire dalla sede dell'alloggio prenotato;
9. una transazione non deve avere l'attributo *data ora* maggiore rispetto alla *data di partenza* del pacchetto viaggio acquistato;
10. un pacchetto viaggio non deve essere prenotato più volte di quanto riportato nella sua disponibilità.

6 O vincoli di integrità. Usiamo la stessa terminologia che viene usata nel libro presentato a inizio corso.

**Regole di derivazione (RD):**

1. il *prezzo totale* in informazioni trasporto deve essere la somma del prezzo di ogni volo per l'andata e il ritorno moltiplicati per il numero di persone;
2. l'*importo* in transizione deve essere la somma: del prezzo del pacchetto di viaggio acquistato moltiplicato il numero di persone più il prezzo totale delle informazioni di trasporto.

# 4 Progettazione logica

## 4.1 Ristrutturazione dello schema E-R

### 4.1.1 Analisi ridondanze

L'attributo *prezzo totale* dell'entità `Informazioni Trasporto` può essere derivato attraverso la somma dei *prezzi* di ogni volo in relazione con l'entità, il tutto moltiplicato per il numero di persone. L'attributo *importo* dell'entità `Transizione` può essere calcolato come la somma del *prezzo* del pacchetto acquistato per il numero di persone che partecipano al viaggio e del *prezzo totale* per il trasporto. La destinazione del pacchetto può essere derivata dalla sede dell'alloggio prenotato. Riportiamo dunque le tavole dei volumi per le entità e le operazioni di nostro interesse:

*Table V: Tabella dei volumi*

| Concetto | Tipo | Volume (B) |
|---|---|---|
| *Transizione* | E | 760 000 |
| *Informazioni Trasporto* | E | 754 000 |

*Table VI: Tabella delle operazioni*

| Operazione n° | Tipo[7] | Frequenza |
|---|---|---|
| *5* | I | 10 000 al giorno |
| *6* | I | 3 000 al giorno |

*Prezzo totale* è un Numeric(7,2), dunque ipotizziamo che occupi 5 byte, calcoliamo il volume occupato: $5B \cdot 760\,000 = 3,8\text{Mb}$, mentre le operazioni richieste per il calcolo o visualizzazione di questo dato sono la 5 e 6, ipotizziamo che un utente in media prenoti 13 pacchetti:

**Tavole degli accessi in presenza di ridondanza**

*Table VII: operazione 5*

| Concetto | Costr. | Acc. | Tipo |
|---|---|---|---|
| *Pacchetto Viaggio* | E | 1 | L |
| *Prenotazione* | R | 1 | S |
| *Prenotazione Cliente* | R | 1 | S |
| *Prenotazione Pacchetto* | R | 1 | S |
| *Volo*[8] | E | 2 | L |
| *Informazioni trasporto* | E | 1 | S |
| *Trasporto* | R | 1 | S |
| *Ritorno* | R | 1 | S |
| *Andata* | R | 1 | S |
| *Transazione* | E | 1 | S |
| *Pagamento* | R | 1 | S |

*Table VIII: operazione 6*

| Concetto | Costr. | Acc. | Tipo |
|---|---|---|---|
| *Cliente* | E | 1 | L |
| *Prenotazione cliente* | R | 13 | L |
| *Prenotazione* | E | 13 | L |
| *Prenotazione Pacchetto* | R | 13 | L |
| *Pacchetto Viaggio* | E | 13 | L |
| *Pagamento* | R | 13 | L |
| *Transazione* | E | 13 | L |

**Tavole degli accessi in assenza della ridondanza**

Notiamo che le operazioni della tavola 5 non variano in assenza di ridondanza. Questo perché gli accessi riportati sono sempre li stessi per l'inserimento di una nuova prenotazione.

*Table IX: operazione 6*

| Concetto | Costr. | Acc. | Tipo |
|---|---|---|---|
| *Cliente* | E | 1 | L |
| *Prenotazione cliente* | R | 13 | L |

7 *I*: interattiva. *B*: batch.

8 La maggior parte dei viaggi ha solo due 2 voli: andata e ritorno. Inoltre tutti usando i voli offerti dal servizio.

| | | | |
|---|---|---|---|
| *Prenotazione* | E | 13 | L |
| *Prenotazione Pacchetto* | R | 13 | L |
| *Pacchetto Viaggio* | E | 13 | L |
| *Pagamento* | R | 13 | L |

| | | | |
|---|---|---|---|
| *Transazione* | E | 13 | L |
| *Ritorno* | R | 13 | L |
| *Andata* | R | 13 | L |

Ipotizzando che una scrittura equivale a 2 letture, abbiamo che: in presenza di ridondanza avvengono 21 accessi per l'operazione *5*, mentre 79 accessi in media per l'operazione *6*. Calcoliamo al giorno quante operazioni in presenza di ridondanza abbiamo: $21 \cdot 10\,000 + 79 \cdot 3\,000 = 447\,000$ accessi in media. Mentre in assenza di ridondanza abbiamo che: l'operazione *5* non varia il suo numero di accessi, mentre l'operazione *6* ha in media 105 accessi. Calcoliamo in assenza di ridondanza il numero medio di accessi: $21 \cdot 10\,000 + 105 \cdot 3\,000 = 525\,000$ accessi. Otteniamo dunque 78 000 accessi in più in assenza di ridondanza al giorno.

Possiamo notare che anche *Importo* di `transizione` è Numeric(7,2) e richiede la stessa analisi fatta per il *prezzo totale,* l'unica cosa che cambia sono i volumi: $5B \cdot 754\,000 = 3,77$Mb. Otteniamo alla fine eliminando entrambi gli attributi 7,57 Mb risparmiati a discapito di 156 000 accessi aggiuntivi, per un totale complessivo di 1 050 000 accessi per entrambi gli attributi al giorno. Dato che il risparmio è abbastanza irrilevante rispetto a un numero cospicuo di accessi (156 000 accessi in più), si è deciso mantenere entrambi gli attributi.

La relazione *Destinazione* tra `pacchetto viaggio` e `città` è una ridondanza, decidiamo di rimuoverla visto che: non abbiamo problemi di inconsistenza, non serve applicare più la regola aziendale RV8 e semplifica la struttura della base di dati eliminando una tabella che aumentava la grandezza della struttura dello schema.

### 4.1.2 Eliminazione generalizzazioni

Notiamo che l'unica generalizzazione presente nel diagramma E-R concettuale è totale. La strategia che usiamo è quella dell'*accorpamento del genitore della generalizzazione nelle figlie*. Le altre tecniche sono state escluse perché:

- in questo caso è importante distinguere l'entità figlie per eseguire le operazioni tutte distinte tra loro, dunque è da escludere l'*accorpamento delle figlie della generalizzazione nel genitore*;
- non conviene usare la tecnica di *sostituzione della generalizzazione con associazioni* perché richiede l'aggiunta di vincoli e inoltre aggiunge ulteriori accessi.

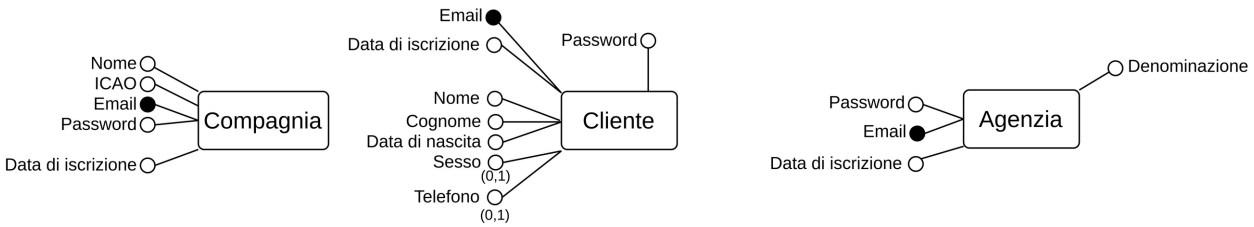


*Figure 2: risultato della eliminazione della generalizzazione Utente.*

### 4.1.3 Partizionamento accorpamento di entità e associazioni

Gli attributi *Indirizzo* della relazione `sede agenzia` e `sede alloggio` potrebbero avere delle ridondanze, ma dato che è estremamente raro e il numero di agenzie è abbastanza contenuto rispetto a quello degli alloggi, si è deciso di lasciare invariata la struttura delle relazioni.

Viene partizionata la nuova struttura dell'entità `cliente` in modo da distinguere i dati del cliente da quelli utente ereditati dal genitore, questo con lo scopo di semplificare e separare i concetti di dati dell'utente e dati personali.

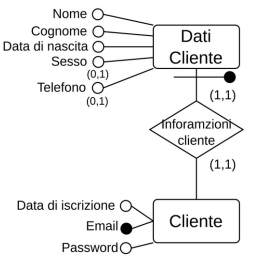


*Figure 3: Risultati accorpamento.*

### 4.1.4 Scelta degli identificatore principali

Gli identificatori principali rimano sempre quelli segnati sullo schema E-R.

### 4.1.5 E-R Logico

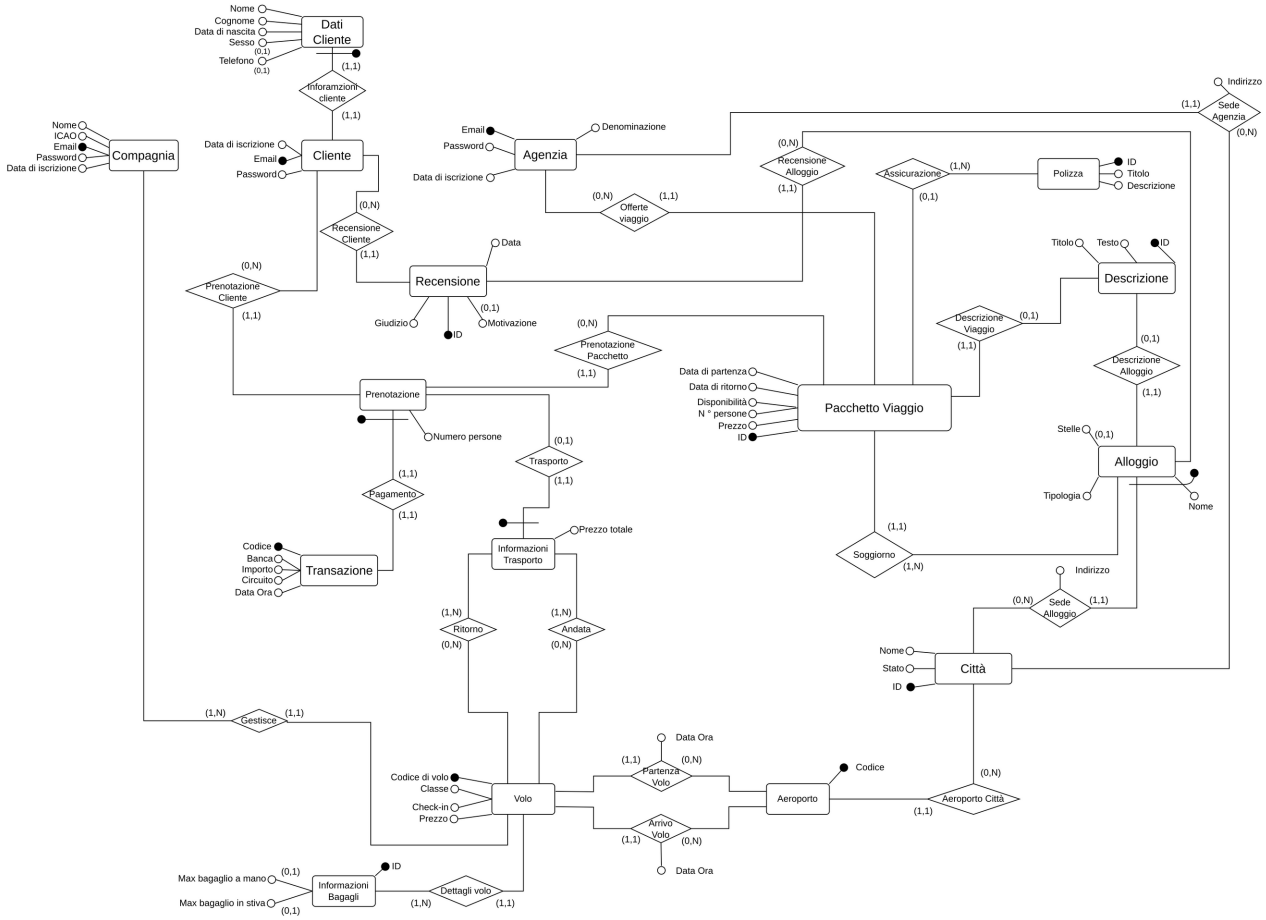


*Figure 4: E-R logico.*

## 4.2 Traduzione verso il modello relazionale

Usiamo le notazioni usate per l'E-R concettuale. Nei vincoli referenziali indichiamo che un insieme di attributi $X$ della relazione $R_1$ è una ***chiave referenziale*** per la relazione $R_2$ rispetto alla sua chiave primaria $K$ nel seguente modo: $R_1(X) \rightarrow R_2(K)$.

**Cliente**(<u>Email</u>, Password, DataIscrizione)

- Cliente(Email) → DatiCliente(Email)

**DatiCliente**(<u>Email</u>, Nome, Cognome, DataNascita, Sesso, Telefono)

**Compagnia**(<u>Email</u>, Password, DataIscrizione, Nome, ICAO)

**Volo**[9](<u>Codice</u>, Classe, CheckIn, Prezzo, EmailCompagnia, AeroportoPartenza, TimestampPartenza AeroportoArrivo, TimestampArrivo, IDBagagli)

- Volo(EmailCompagnia) → Compagnia(Email)

9 Rinominamo l'attributo DataOra ereditato dalla associazione *PartenzaVolo* in TimestampPartenza. In modo analogo per l'arrivo.

- Volo(AeroportoPartenza) → Aeroporto(Codice)
- Volo(AeroportoArrivo) → Aeroporto(Codice)
- Volo(IDBagagli) → InformazioniBagagli(ID)

**InformazioniBagaglio**(ID, BagaglioMano, BagaglioStiva)

**Agenzia**(Email, Password, DataIscrizione, Denominazione, IDCittà, Indirizzo)

- Agenzia(IDCittà) → Città(ID)

**PacchettoViaggio**(ID, Prezzo, NumeroPersone, Disponibilità, DataPartenza, DataRitorno, EmailAgenzia, IDPolizza, IDDescrizione, IDCittàAlloggio, NomeAlloggio)

- PacchettoViaggio(EmailAgenzia) → Agenzia(Email)
- PacchettoViaggio(IDPolizza) → Polizza(ID)
- PacchettoViaggio(IDDescrizione) → Descrizione(ID)
- PacchettoViaggio(IDCittàAlloggio, NomeAlloggio) → Alloggio(IDCittà, Nome)

**Polizza**(ID, Descrizione, Assicuratore)

**Descrizione**(ID, Testo, Titolo)

**Alloggio**(IDCittà, Nome, Stelle, Tipologia, IDDescrizione, Indirizzo)

- Alloggio(IDCittà) → Città(ID)
- Alloggio(IDDescrizione) → Descrizione(ID)

**Città**(ID, Nome, Stato)

**Aeroporto**(Codice, IDCittà)

- Aeroporto(IDCittà) → Città(ID)

**Prenotazione(**CodiceTransizione, EmailCliente, NumeroPersone, IDPacchettoViaggio)

- Prenotazione(CodiceTransizione) → Transizione(Codice)
- Prenotazione(IDPacchettoViaggio) → PacchettoViaggio(ID)

**Transazione**(Codice, Banca, Importo, Circuito, DataOra)

**InformazioniTrasporto**(CodiceTransizione, PrezzoTotale)

- InformazioniTrasporto(CodiceTransizione) → Prenoatzione(CodiceTransizione)

**Ritorno**(CodiceTransizione, CodiceVolo)

- Ritorno(CodiceTransizione) → InformazioniTrasporto(CodiceTransizione)
- Ritorno(CodiceVolo) → Volo(Codice)

**Andata**(CodiceTransizione, CodiceVolo)

- Andata(CodiceTransizione) → InformazioniTrasporto(CodiceTransizione)
- Andata(CodiceVolo) → Volo(Codice)

**Recensione**(ID, Giudizio, Motivazione, Data, EmailCliente, IDCittàAlloggio, NomeAlloggio)

- Recensione(EmailCliente) → Cliente(Email)
- Recensione(IDCittà, NomeAlloggio) → Alloggio(IDCittà, NomeAlloggio)

Note implementative: la regola aziendale RV1 è stata implementata direttamente in volo attraverso un check. Nel database usiamo '_' al posto dello stile CamelCase usato nella relazione.

# 5 Query SQL

- Trovare tutti i clienti che hanno effettuato almeno una recensione con un voto superiore o uguale a x stelle, riportare il loro: nome, cognome e l'email, e specificare il nome dell'alloggio per cui è stata scritta la recensione, il giudizio dato e la motivazione se disponibile, nel caso tale informazione non fosse disponibile sostituire NULL con la stringa: '* Nessuna motivazione fornita'. Per l'esempio x = 3

```
SELECT D.NOME, D.COGNOME, A.NOME, R.GIUDIZIO,
  COALESCE(R.MOTIVAZIONE,'* Nessuna motivazione fornita')AS MOTIVAZIONE
  FROM DATI_CLIENTE AS D JOIN CLIENTE AS C ON D.EMAIL_CLIENTE = C.EMAIL
  JOIN RECENSIONE AS R ON C.EMAIL = R.EMAIL_CLIENTE
  JOIN ALLOGGIO AS A ON R.ID_CITTA_ALLOGGIO = A.ID_CITTA AND R.NOME_ALLOGGIO = A.NOME
 WHERE GIUDIZIO >= 3;
```

| | nome character varying (30) | cognome character varying (30) | nome character varying (40) | giudizio numeric (1) | motivazione character varying |
|---|---|---|---|---|---|
| 1 | Bettina | Cortese | Casa Vista Mare | 3 | Non è stato male, ma potrebbe migliorare. |
| 2 | Erika | Asmundo | Casa Vista Mare | 4 | * Nessuna motivazione fornita |
| 3 | Virginia | Piacentini | Casa del Laghetto | 3 | Non male, ma c'erano alcune carenze. |
| 4 | Lara | Polesel | Casa Vista Mare | 3 | * Nessuna motivazione fornita |
| 5 | Rosina | Pavanello | Hotel Alpino | 3 | Luogo incantevole ma il servizio era mediocre. |
| 6 | Rosina | Pavanello | Hotel Alpino | 3 | Non era male, ma c'era sicuramente margine per migliorar... |
| 7 | Paoletta | Travaglio | Casa del Laghetto | 3 | * Nessuna motivazione fornita |
| 8 | Antonella | Taccola | Casa Vista Mare | 3 | * Nessuna motivazione fornita |
| 9 | Micheletto | Mazzacurati | Suite dei Sogni | 3 | Non è stato male, ma potrebbe migliorare. |
| 10 | Achille | Salandra | Casa del Laghetto | 4 | * Nessuna motivazione fornita |

*Figure 5: risultato query n° 1.*

- Per ciascuna compagnia aerea fornire: il nome della compagnia, il numero di prenotazioni, il numero totale di voli che sono stati prenotati compagnia aerea, la media prezzi dei voli offerti, il prezzo massimo devi voli offerti finora, e il prezzo minimo dei voli offerti. Filtrare la ricerca in modo da mostrare solo le compagnia con almeno x voli. In fine si riordini in modo decrescente per numero voli. Esempio: x = 10.

```
SELECT C.NOME AS NOME_COMPAGNIA, COUNT(V.CODICE) AS NUMERO_VOLI, AVG(V.PREZZO) AS MEDIA_PREZZO_VOLO,
       MAX(V.PREZZO) AS PREZZO_MASSIMO, MIN(V.PREZZO) AS PREZZO_MINIMO
  FROM COMPAGNIA C JOIN VOLO AS V ON C.EMAIL = V.EMAIL_COMPAGNIA
 GROUP BY C.NOME
HAVING COUNT(V.CODICE) >= 10
 ORDER BY NUMERO_VOLI DESC;
```

| | nome_compagnia character varying (40) | numero_voli bigint | media_prezzo_volo numeric | prezzo_massimo numeric | prezzo_minimo numeric |
|---|---|---|---|---|---|
| 1 | SkySprint Airlines | 40 | 43.2147500000000000 | 78.88 | 23.18 |
| 2 | RiverRide Cruises | 30 | 43.0116666666666667 | 79.28 | 20.21 |
| 3 | AquaJet Ferries | 22 | 39.3190909090909091 | 69.10 | 22.29 |

*Figure 6: risultato query n° 2.*

- Riportare per ogni agenzia il numero di pacchetti offerti, in media quanto costano i loro pacchetti e la media delle recensioni ricevute sugli alloggi offerti.

```
SELECT AGENZIA.DENOMINAZIONE, NUMERO_OFFERTE.CONTEGGIO AS NUMERO_PACCHETTI,
       ROUND(MEDIA_PREZZI.MEDIA,2) AS PREZZO_MEDIO, ROUND(MEDIA_REC.MEDIA,2) as MEDIA_RECENSIONI
  FROM
        (SELECT EMAIL_AGENZIA AS EMAIL, COUNT(*) AS CONTEGGIO
          FROM PACCHETTO_VIAGGIO GROUP BY EMAIL_AGENZIA ) AS NUMERO_OFFERTE,
        (SELECT EMAIL_AGENZIA AS EMAIL, AVG(PACCHETTO_VIAGGIO.PREZZO) AS MEDIA
          FROM PACCHETTO_VIAGGIO GROUP BY EMAIL_AGENZIA) AS MEDIA_PREZZI,

        (SELECT EMAIL_AGENZIA AS EMAIL, AVG(R.GIUDIZIO) AS MEDIA
          FROM RECENSIONE AS R JOIN PACCHETTO_VIAGGIO AS P ON P.ID_CITTA_ALLOGGIO = R.ID_CITTA_ALLOGGIO
           AND P.NOME_ALLOGGIO = R.NOME_ALLOGGIO
         GROUP BY P.EMAIL_AGENZIA
       ) AS MEDIA_REC, AGENZIA
```

```
14
15  WHERE NUMERO_OFFERTE.EMAIL = MEDIA_PREZZI.EMAIL AND MEDIA_PREZZI.EMAIL = MEDIA_REC.EMAIL
16    AND MEDIA_REC.EMAIL = AGENZIA.EMAIL;
```

| | denominazione character varying (40) | numero_pacchetti bigint | prezzo_medio numeric | media_recensioni numeric |
|---|---|---|---|---|
| 1 | Turismo Rurale Magico | 3 | 1042.10 | 1.50 |
| 2 | Svela il Mondo | 3 | 451.88 | 2.63 |
| 3 | Sogni Senza Confini | 3 | 912.23 | 1.63 |
| 4 | Esplora il Mondo Segreto | 6 | 886.13 | 2.67 |
| 5 | Viaggi Unici | 4 | 685.14 | 2.42 |
| 6 | Turismo Sostenibile | 1 | 639.85 | 2.50 |

*Figure 7: risultato query n° 3.*

- Dato un cliente, per ogni acquisto recuperare: tutte le informazioni della transizione, il totale per il trasporto, il costo di base del pacchetto a persona, il numero di persone partecipanti e il numero di voli per quel viaggio. Si includano inoltre anche i casi in cui non hanno prenotato il trasporto attraverso il servizio, e si riporti in quel caso il valore 0, sia per il totale trasporto che per il numero di voli. Cliente per l'esempio: Annunziata.Fornaciari@email.com.

```
1  SELECT DATI_PRENOTAZIONE.CODICE, DATI_PRENOTAZIONE.NUMERO_PERSONE AS NUMERO_PARTECIPANTI,
2         DATI_PRENOTAZIONE.BANCA, DATI_PRENOTAZIONE.IMPORTO, DATI_PRENOTAZIONE.CIRCUITO,
3         DATI_PRENOTAZIONE.DATAORA, DATI_PRENOTAZIONE.PREZZO AS PREZZO_PACCHETTO,
4         COALESCE(INFO_TRASPORTO.PREZZO_TOTALE, 0) AS TOTALE_TRASPORTO,
5         COALESCE(INFO_TRASPORTO.VOLI_TOTALI, 0) AS VOLI_PRESI
6    FROM
7          (SELECT I.CODICE_TRANSAZIONE AS CODICE, I.PREZZO_TOTALE, DATI_VOLI.VOLI_TOTALI
8            FROM
9                  (SELECT ANDATA_RITORNO.CODICE_TRANSAZIONE, COUNT(*) AS VOLI_TOTALI
10                   FROM
11                         (SELECT * FROM ANDATA
12                       UNION ALL
13                           SELECT * FROM RITORNO
14                        ) AS ANDATA_RITORNO
15                   GROUP BY ANDATA_RITORNO.CODICE_TRANSAZIONE
16                 ) AS DATI_VOLI,
17                 INFORMAZIONI_TRASPORTO AS I
18           WHERE I.CODICE_TRANSAZIONE = DATI_VOLI.CODICE_TRANSAZIONE
19         ) AS INFO_TRASPORTO
20   RIGHT JOIN
21          (SELECT T.CODICE, T.DATAORA, T.IMPORTO, T.CIRCUITO,
22                T.BANCA, P.NUMERO_PERSONE, PV.PREZZO
23            FROM PRENOTAZIONE AS P, TRANSAZIONE AS T, PACCHETTO_VIAGGIO AS PV
24           WHERE 'Annunziata.Fornaciari@email.com' = P.EMAIL_CLIENTE
25             AND P.CODICE_TRANSAZIONE = T.CODICE AND PV.ID = P.ID_PACCHETTO_VIAGGIO
26         ) AS DATI_PRENOTAZIONE
27      ON INFO_TRASPORTO.CODICE = DATI_PRENOTAZIONE.CODICE;
```

| | codice character varying (16) | numero_partecipanti numeric (2) | banca character varying (30) | importo numeric (7,2) | circuito character varying (20) | dataora timestamp without time zone | prezzo_pacchetto numeric (7,2) | totale_trasporto numeric | voli_presi bigint |
|---|---|---|---|---|---|---|---|---|---|
| 1 | WXHCLYXDKFNPATWJ | 2 | HarborTrust | 1029.27 | FinanzaLink | 2021-04-26 09:02:02 | 422.66 | 183.95 | 2 |
| 2 | CVVBFUDKVUXZCRTH | 1 | NexaFinance | 1540.57 | CapitalCrossing | 2023-07-11 06:54:34 | 1383.45 | 157.12 | 4 |
| 3 | RDONUAHCODAOORAW | 2 | SmartBank | 1534.50 | BancaLink Global | 2021-06-24 19:39:47 | 493.75 | 547.00 | 6 |

*Figure 8: risultato query n° 4.*

- Algoritmo per mostrare i pacchetti: fornita una data, cercare tutti i pacchetti viaggio che si svolgono dopo quella data. Eliminare i pacchetti che non sono più disponibili (perché già prenotati tutti) dai risultati. Riportare le seguenti informazioni essenziali: il titolo della descrizione del pacchetto, la data di partenza, la data di ritorno, il prezzo, il numero di persone, il nome dell'alloggio, la destinazione e la polizza. In fine si filtri i risultati lasciando tutti i pacchetti compresi dal prezzo in un intevervallo di prezzo. Ordinarli per data di partenza in ordine crescente. Esempio con la data 2019-10-11, intrevallo prezzo [300, 1000].

```
1  SELECT D.TITOLO, DISPONIBILI.PREZZO, DISPONIBILI.NUMERO_PERSONE, DISPONIBILI.DATA_PARTENZA,
2         DISPONIBILI.DATA_RITORNO, DISPONIBILI.NOME_ALLOGGIO, C.NOME AS NOME_CITTA, C.STATO,
```

```
        P.titolo
  FROM
        (SELECT P.ID, P.PREZZO, P.NUMERO_PERSONE, P.DATA_PARTENZA, P.DATA_RITORNO,
              P.ID_DESCRIZIONE, P.ID_CITTA_ALLOGGIO, P.NOME_ALLOGGIO, P.id_polizza
          FROM
               (SELECT ID_PACCHETTO_VIAGGIO AS ID, COUNT(*) AS PRENOTAZIONI_TOTALI
                 FROM PRENOTAZIONE GROUP BY ID_PACCHETTO_VIAGGIO) AS CONTEGGIO_PRENOTAZIONI,
              PACCHETTO_VIAGGIO AS P
         WHERE P.ID = CONTEGGIO_PRENOTAZIONI.ID
           AND P.DISPONIBILITA > CONTEGGIO_PRENOTAZIONI.PRENOTAZIONI_TOTALI
           AND P.DATA_PARTENZA > '2019-10-11') AS DISPONIBILI,

       DESCRIZIONE AS D,
       CITTA AS C, POLIZZA as P

 WHERE D.ID = DISPONIBILI.ID_DESCRIZIONE AND DISPONIBILI.ID_POLIZZA = P.id
   AND C.ID = DISPONIBILI.ID_CITTA_ALLOGGIO AND DISPONIBILI.PREZZO > 300
   AND DISPONIBILI.PREZZO < 1000
 ORDER BY DISPONIBILI.DATA_PARTENZA;
```

| | titolo character varying (40) | prezzo numeric (7,2) | numero_persone numeric (2) | data_partenza date | data_ritorno date | nome_alloggio character varying (40) | nome_citta character varying (40) | stato character varying (35) |
|---|---|---|---|---|---|---|---|---|
| 1 | Esplora una Città Vibrante | 482.77 | 4 | 2022-02-27 | 2022-03-06 | Casa Vista Mare | Fiumicino Di Savignano | Italia |
| 2 | Esperienza Teatrale | 639.85 | 1 | 2022-03-31 | 2022-04-13 | Hotel Alpino | Agliana | Italia |
| 3 | Avventure Subacquee | 781.08 | 1 | 2022-07-27 | 2022-08-05 | Casa Vista Mare | Isola Fossara | Italia |
| 4 | Esplorazione Urbana | 619.24 | 1 | 2022-09-03 | 2022-09-08 | Hotel Alpino | Agliana | Italia |

*Figure 9: risultato query n° 5.*

- Si trovi tutti i voli da una aeroporto x ad un aeroporto y che attraverso gli scali costani meno del viaggio diretto. Di ogni volo con scalo si riporti solo il codice del volo di partenza, la data/ora di partenza, il codice del volo di arrivo, la data/ora di arrivo, il numero di scali, il prezzo contando tutti voli dello scalo, il risparmio (differenza totale con scali e senza scali), il tempo totale di viaggio (somma dei voli, senza contare le attese al terminale). Nota: il totale prezzo dei voli è per persona. Nell'esempio: x.codice = 'LQVF' e y.codice = 'YSEM'

```
WITH RECURSIVE POSSIBILI_SCALI(
    PRIMO_VOLO, -- Tiene traccia della partenza
    VOLO_ATTUALE, AEROPORTO_PARTENZA, TIMESTAMP_PARTENZA,
    AEROPORTO_ARRIVO, TIMESTAMP_ARRIVO, NUMERO_SCALI,
    TOTALE_PREZZO, DURATA_TOTALE_VIAGGIO) AS (
    (
        SELECT
            VOLO.CODICE AS PRIMO_VOLO, VOLO.CODICE AS VOLO_ATTUALE,
            AEROPORTO_PARTENZA, TIMESTAMP_PARTENZA,
            AEROPORTO_ARRIVO, TIMESTAMP_ARRIVO,
AS NUMERO_SCALI, CAST(VOLO.PREZZO AS NUMERIC(7,2)) AS TOTALE_PREZZO,
            (TIMESTAMP_ARRIVO - TIMESTAMP_PARTENZA) AS DURATA_TOTALE_VIAGGIO
        FROM VOLO)
    UNION ALL --Ricorsione
    (
        SELECT
            POSSIBILI_SCALI.PRIMO_VOLO, SUCCESSIVO.CODICE AS VOLO_ATTUALE,
            SUCCESSIVO.AEROPORTO_PARTENZA, SUCCESSIVO.TIMESTAMP_PARTENZA,
            SUCCESSIVO.AEROPORTO_ARRIVO, SUCCESSIVO.TIMESTAMP_ARRIVO,
            POSSIBILI_SCALI.NUMERO_SCALI + 1 AS NUMERO_SCALI,
            CAST(POSSIBILI_SCALI.TOTALE_PREZZO + SUCCESSIVO.PREZZO AS NUMERIC(7,2))
                                                                AS TOTALE_PREZZO,
            (
                POSSIBILI_SCALI.DURATA_TOTALE_VIAGGIO + (
                    SUCCESSIVO.TIMESTAMP_ARRIVO - SUCCESSIVO.TIMESTAMP_PARTENZA
                )) AS DURATA_TOTALE_VIAGGIO
        FROM
            VOLO AS SUCCESSIVO, POSSIBILI_SCALI
        WHERE
            SUCCESSIVO.AEROPORTO_PARTENZA = POSSIBILI_SCALI.AEROPORTO_ARRIVO
```

```sql
            AND SUCCESSIVO.TIMESTAMP_PARTENZA > POSSIBILI_SCALI.TIMESTAMP_ARRIVO
    )
) -- Nota: tra le tuple di possibili_scali ci sono anche i voli diretti
SELECT
    DIRETTO.PRIMO_VOLO AS DIRETTO,
    DIRETTO.TIMESTAMP_PARTENZA AS D_PARTENZA,
    DIRETTO.TIMESTAMP_ARRIVO AS D_ARRIVO,
    DIRETTO.TOTALE_PREZZO AS D_PREZZO,
    SCALI.PRIMO_VOLO AS S_PARTENZA_CODICE,
    SCALI.VOLO_ATTUALE AS S_ARRIVO_CODICE,
    V.TIMESTAMP_PARTENZA AS S_PARTENZA,
    SCALI.TIMESTAMP_ARRIVO AS S_ARRIVO,
    SCALI.NUMERO_SCALI,
    SCALI.TOTALE_PREZZO AS SCALO_PREZZO,
    SCALI.DURATA_TOTALE_VIAGGIO
FROM
    POSSIBILI_SCALI AS DIRETTO,
    POSSIBILI_SCALI AS SCALI,
    VOLO AS V -- Serve a ottenere le informazioni del primo volo di partenza
WHERE
    DIRETTO.NUMERO_SCALI = 0
    AND DIRETTO.AEROPORTO_PARTENZA = 'LQVF'
    AND DIRETTO.AEROPORTO_ARRIVO = 'YSEM'
    AND DIRETTO.TOTALE_PREZZO > SCALI.TOTALE_PREZZO
    AND SCALI.NUMERO_SCALI > 0
    AND SCALI.AEROPORTO_ARRIVO = DIRETTO.AEROPORTO_ARRIVO
    AND SCALI.PRIMO_VOLO = V.CODICE;
```

| | diretto integer | d_partenza timestamp without time zone | d_arrivo timestamp without time zone | d_prezzo numeric (7,2) | s_partenza_codice integer | s_arrivo_codice integer | s_partenza timestamp without time zone | s_arrivo timestamp without time zone | numero_scali integer | scalo_prezzo numeric (7,2) | durata_totale_viaggio interval |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2022-02-27 03:52:05 | 2022-02-27 06:14:51.142452 | 67.96 | 12 | 49 | 2022-06-21 14:40:26 | 2022-06-30 05:26:48.646091 | 1 | 49.05 | 04:18:58.042487 |
| 2 | 2 | 2022-02-27 03:52:05 | 2022-02-27 06:14:51.142452 | 67.96 | 41 | 49 | 2022-06-22 01:35:00.367036 | 2022-06-30 05:26:48.646091 | 1 | 64.37 | 04:58:03.172684 |
| 3 | 2 | 2022-02-27 03:52:05 | 2022-02-27 06:14:51.142452 | 67.96 | 48 | 49 | 2022-06-21 15:37:17 | 2022-06-30 05:26:48.646091 | 1 | 66.56 | 03:49:27.71694 |
| 4 | 2 | 2022-02-27 03:52:05 | 2022-02-27 06:14:51.142452 | 67.96 | 50 | 49 | 2021-11-10 08:10:27 | 2022-06-30 05:26:48.646091 | 1 | 67.76 | 03:55:34.09844 |

*Figure 10: risultato query n° 6.*

# 6 Indice

Notiamo dal costo delle operazioni che l'azione più impattante è la ricerca dei pacchetti, infatti con un'utenza così ampia è essenziale fornire velocemente le soluzioni di viaggio disponibili. Per questo è necessario ottimizzare attraverso gli indici i punti in cui gli attributi del pacchetto viaggio servono per recuperare le altre informazioni essenziali:

```sql
CREATE INDEX IDX_PXALLOGGI ON PACCHETTO_VIAGGIO(ID_DESCRIZIONE, ID_CITTA_ALLOGGIO, NOME_ALLOGGIO);
```

Un importante osservazione è che l'aggiunta di pacchetti è molto bassa, e solitamente viene fatta in blocco dalle agenzie all'avvicinarsi del periodo di vacanza, dunque effettua poche scritture.

# 7 Software C++

## 7.1 Compilazione

Il codice C++ è incluso tutto nel file "`easy_travel.cpp`". Per la compilazione usare il seguente comando:

```
g++ --std=c++17 easy_travel.cpp -o easy_travel -L dependencies/lib -lpq
```

Si consiglia di usare lo standard ISO C++17 per la compilazione. Il programma è stato testo su: `Ubuntu 23.04 x86_64`, con la versione del compilatore g++: `(Ubuntu 12.3.0-1ubuntu1~23.04) 12.3.0`. Le dipendenze e il setup delle cartelle sono state fatte come indicato a laboratorio.

## 7.2 Documentazione

Sono state sviluppate delle classi semplici e minimali per integrare al meglio il codice della libreria `libpq` con il C++. Le classi implementate sono le seguenti:

- `Database`, rappresenta la connessione al database. Al suo interno gestisce `PGconn`, in questo modo possiamo rendere più semplice la connessione, il controllo dello stato della connessione e la chiusura della connessione;
- `QueryResult`, gestisce `PGresult` e ci permette di semplificare alcuni controlli sullo stato della query. Inoltre abbiamo implementato l'operatore di inserimento `<<` per stampare velocemente i risultati in una tabella usando semplicemente `std::cout`.

## 7.3 Utilizzo

All'avvio il programma chiede all'utente l'inserimento dei dati per la connessione, più precisamente viene richiesto: username e password per accedere al database, mentre le altre informazioni sono già preconfigurate. Dopo aver creato la connessione il programma procede a precompilate tutte le query riportate al capitolo 5 della relazione. Finita la precompilazione viene stampato un menu con 6 opzioni. Dopo aver selezionato una query, se questa è parametrizzata allora prima di stampare il risultato si viene guidati all'inserimento dei parametri richiesti, altrimenti viene direttamente stampato il risultato e si ritorna al menu principale.

# 8 Note extra

- Per le cose extra (query ricorsive, analisi dei costi, funzioni condizionali, eccetera) abbiamo fatto riferimento al libro di testo consigliato all'inizio del corso: Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, *Basi di Dati* edizione VI, McGraw Hill;
- i dati inseriti nel database sono fittizi. Data la grandezza della base di dati abbiamo deciso di scrivere in python uno script per generare i dati e le relazioni da inserire nel database;
- nel file easy_travel.sql è riportato il backup del database. All'inizio del file è riportato un indice con i punti più importanti;
- per la query n° 6 si consiglia di mettere il programma a schemi intero per evitare problemi con la stampa della tabella a causa delle tante colonne.