# Indice

* [Introduzione](#_Introduzione)
* Descrizione del progetto
* Obiettivo
* [Fase 1 – Database](#_Fase_1_)
* Diagramma entità-relazione
* Logica del diagramma E/R
* Modello relazionale
* Implementazione del DB su Oracle SQL Developer
* Implementazione Trigger

# Introduzione

Descrizione del progetto

Progetto individuale volto alla creazione di un’applicazione web con database relazionale alla base. Per la progettazione si useranno i seguenti linguaggi di programmazione:

* SQL, per implementare il database su Oracle SQL Developer;
* Java, per la costruzione del backEnd su Eclipse IDE;
* HTML, per la costruzione del frontEnd nel framework Angular

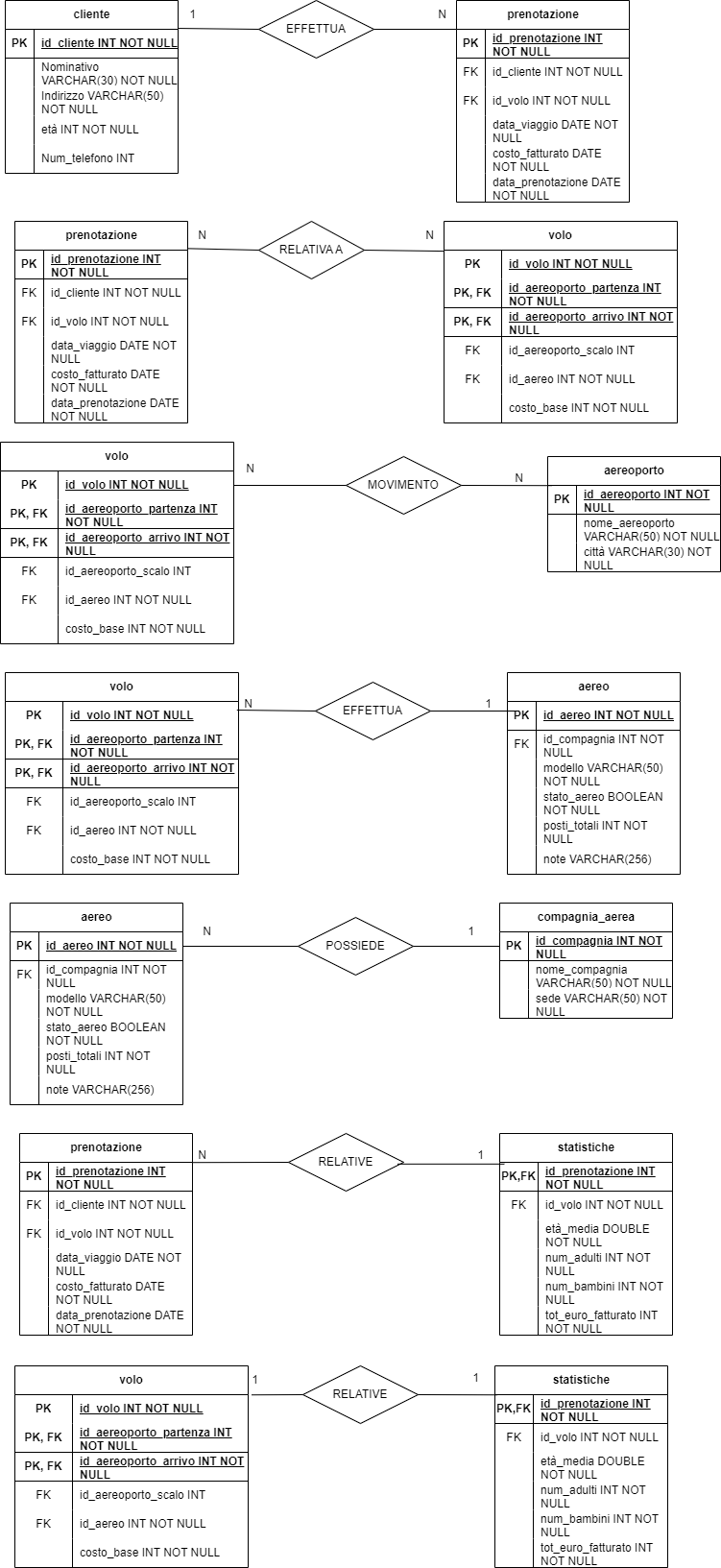
Obiettivo

Lo scopo del progetto è creare un'applicazione web con la quale gli utenti potranno acquistare biglietti aerei. Il sistema si basa su un database relazionale con le sue funzioni di gestione voli e prenotazione. Avremo un database che supporterà la prenotazione su molteplici tratte in tutto il mondo, centinaia di voli per varie compagnie aeree.

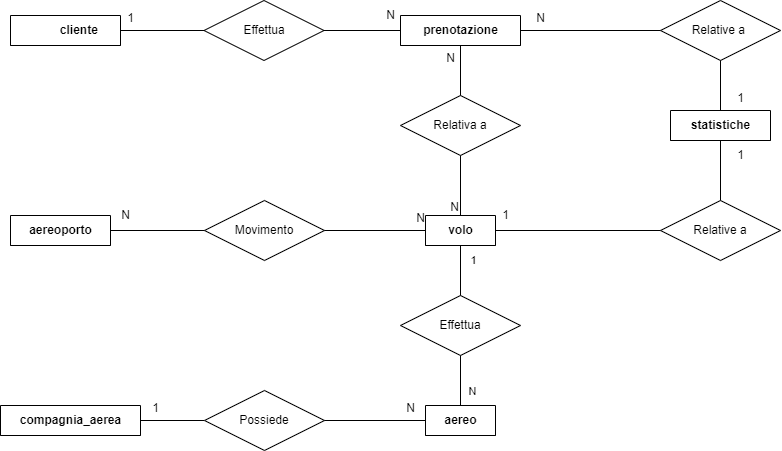
# Fase 1 - Database

Diagramma Entità-Relazione

Ho utilizzato il software di disegno grafico diagrams.net per la rappresentazione dell’architettura del Diagramma Entità-Relazione. Ho innanzitutto descritto dettagliatamente ciascuna relazione, con il dettaglio di ogni entità.



Successivamente ho creato il Diagramma E/R vero e proprio: avendo già descritto nel dettaglio ciascuna entità, qui ho preferito rappresentare solo le entità, relazioni e cardinalità, per rendere il diagramma più snello e chiaro.



Logica del diagramma E/R

Il database sarà sviluppato per supportare le prenotazioni effettuate da un cliente e relative a un volo. La logica del diagramma parte proprio dalla tabella **cliente**, contenente i dati del medesimo che effettuerà la **prenotazione** e riconosciuto univocamente dalla PK “id\_cliente”. Da questo id, la tabella prenotazione riconoscerà il cliente e lo collegherà al **volo** acquistato, a sua volta riconosciuto univocamente da un “id\_volo”. La tabella volo terrà conto dell’aereo con il quale verrà effetuato e della compagnia aerea che lo organizza. Le entità **aereo** e **compagnia\_aerea** sono anch’esse legate da una relazione, in quanto la compagnia aerea è in grado di possedere un aereo. Tornando all’entità principale del diagramma, nonché volo, questa necessita di avere un aeroporto di origine, uno di destinazione e che vengano menzionati gli eventuali aeroporti in cui l’aeroporto può fare scalo. Per non creare tre tabelle pressochè identiche (aeroporto\_origine, aeroporto\_destinazione e aeroporto\_scali) come avevo inizialmente pensato, ho optato per raccogliere le caratteristiche di un aeroporto nell’entità **aeroporto**. Essendo volo e aeroporto collegate da una relazione N:N, ho passato a volo tre id\_aeroporto diversi come foreign key e ho pensato di descrivere in fase di implementazione id\_aeroporto\_scalo come una lista di id. Inoltre, passandomi id\_volo come chiave esterna, ho creato una tabella **statistiche** che raccoglierà statistiche utili relative a ciascun volo. Statistiche deve essere collegata anche alla tabella prenotazione perché raccoglierà statistiche anche sul numero di prenotazioni.

Modello relazionale

Ho costruito le tabelle normalizzate derivanti dal diagramma E/R su Microsoft Excel.



Implementazione del DB su Oracle SQL Developer

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamente

Immagine che contiene testo, schermata, Carattere, numero

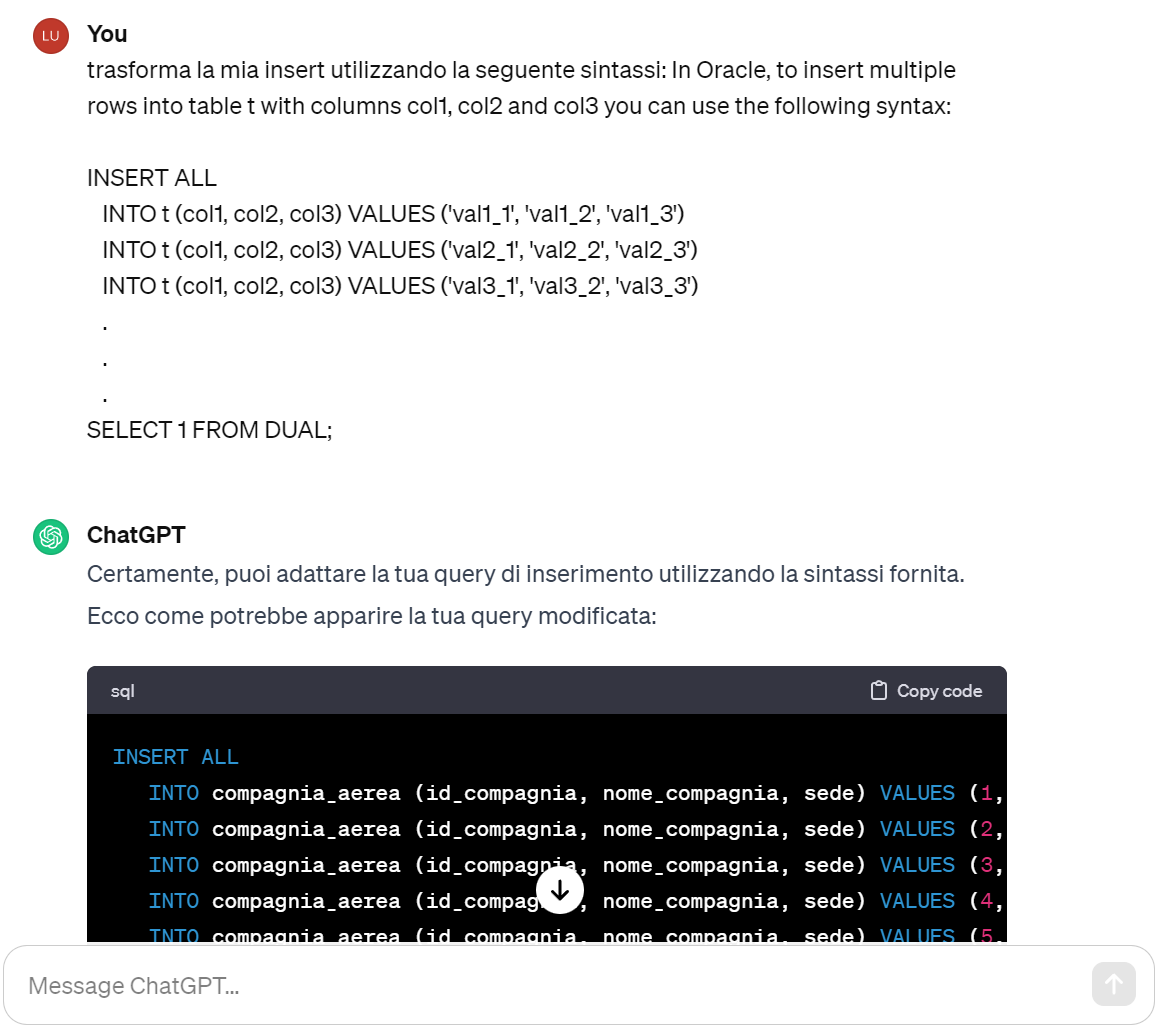
Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

Inserimento dati nelle tabelle (INSERT)

Per fare le insert dei dati nelle tabelle mi sono dapprima documentato sulla sintassi da utilizzare su Oracle SQL Developer (tramite il forum StackOverflow), poi, essendo dati casuali e soprattutto lunghi, ho velocizzato il lavoro di inserimento con l’AI ChatGPT. Per non sbagliare, ho dettagliato la richiesta inviando prima la create table della tabella che volevo riempire (in modo che memorizzasse i campi) e successivamente le ho fornito la sintassi corretta, come in foto. Ho ripetuto il processo per ciascuna tabella, facendo particolare attezione che i valori delle FK coincidessero tra loro.



Implementazione Trigger

1. **TRIGGER STATO AEREO**

Attraverso questo trigger verifico se lo stato dell’aereo (tab= AEREO; col=STATO) è cambiato da “funzionante”(1) a “non funzionante”(2). A condizione avverata, cerco un aereo che risulti funzionante e successivamente modifico i voli collegati all’aereo il cui stato è cambiato, assegnando il nuovo aereo trovato nel passaggio precedente.

Immagine che contiene testo, schermata, numero, software

Descrizione generata automaticamente

1. **TRIGGER PRIMO VOLO**

Questo trigger ci permette di verificare se un volo è il primo effettuato da un aereo. In particolare, ho implementato un counter per contare i voli associati all’aereo. Se il valore del counter è =0, vuol dire che è il primo che l’aereo effettua, quindi ottengo la data del primo viaggio dalla tabella prenotazione. A quel punto, posso aggiornare la sezione note di quell’aereo, dichiarando la data del volo inaugurale.



1. **TRIGGER AGGIORNA STATISTICHE**

Con questo trigger andremo ad aggiornare le statistiche ogni qual volta un cliente effettua una prenotazione. In primo luogo, calcolo l’età media dei passeggeri per il volo associato alla prenotazione. Successivamente, conto il numero di adulti e bambini nella prenotazione, attraverso una semplice funzione sull’età (passata dalla tabella cliente). Una volta effettuati questi necessari calcoli, posso aggiornare le statistiche.

/Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente