Task @ hlpy - Simone Vizzuso

Testo dell’esercizio

Il CEO dell'azienda EasyPeasy ti ha chiesto di disegnare il database a supporto del nuovo servizio che l'azienda vorrebbe offrire. L'azienda vuole espandere le proprie attività con un servizio di NCC prenotabile tramite app. Gli utenti possono fissare una data e luogo dell'appuntamento e un luogo di destinazione. Alla data prestabilita, si presenterà un autista che lo porterà al luogo desiderato. Gli autisti sono dipendenti dell'azienda e utilizzano vetture di servizio. Per una questione legale, tutti i mezzi devono partire da una delle rimesse dell'azienda. Per evitare ritardi sull'appuntamento e per fornire in anticipo dei tempi di durata della corsa, l'azienda si affida ad un servizio di terze parti per stimare il tempo di arrivo sul posto (es. Google Maps). Il servizio necessità di due posizioni e restituisce distanza e tempo di percorrenza (NB: a parità di punto di partenza ed arrivo, il tempo potrebbe essere differente in base alla fascia oraria e giorno della settimana per il quale è richiesto il calcolo).

In giallo le informazioni principali con cui costruire il database o che verranno usate dall’applicativo. In verde gli strumenti utilizzati. In blu le informazioni secondarie o di comprensione.

Esercizio 1a

Il requisito di business ti sembra completo e corretto? Se no, annota tutti i dubbi che ti vengono in mente altrimenti prova a fare un paio di proposte per migliorare il requisito.

Il requisito ha fornito le informazioni basilari su cui costruire il database, ma vi sono alcuni punti su cui sarebbe meglio ottenere maggiori informazioni:

- **Assegnazione conducenti/veicoli** - La scelta del conducente è data da un algoritmo? Probabilmente dalle sue disponibilità orarie e giornaliere, ma chi viene selezionato a parità di disponibilità? *(Necessario per capire come gestire la disponibilità in DB, se assegnare degli ordini di priorità o meno)*
  Un conducente è legato ad un veicolo proprio, o è libero di utilizzare qualsiasi veicolo della compagnia? *(Necessario per capire come organizzare le tabelle dei veicoli/conducenti)*

L’utente può effettuare richieste speciali tramite un campo note? *(Necessario per inserire o meno il relativo campo nella richiesta)*

- **Posti interni ai veicoli** - La singola chiamata è limitata ad una quantità di persone? Una, due o più persone alla volta? *(Necessario per inserire un eventuale contatore di persone attese)*
  Vi sono veicoli con posti a sedere per persone con handicap? *(Necessario per capire se inserirla tra le possibili caratteristiche dei veicoli)*
- **Pagamento** - Il pagamento avverrà attraverso una piattaforma interna o sarà affidato a servizi terzi? In ogni caso, il pagamento verrà salvato in database o la fatturazione verrà prodotta e mantenuta al di fuori del database? *(Necessario per capire se inserire il costo come campo, salvare solo la ricevuta legata alla singola corsa, inserire il pagamento intero come tabella o non inserirlo affatto)*
- **Feedback** - Viene attuato un sistema di feedback? *(Necessario per capire se deve essere presente nel database)*
  Se si, sono presenti anche dei commenti, o risposte a domande multiple? *(Necessario per capire se inserirlo come campo nella richiesta o realizzare una tabella a parte)*

Oltre a queste richieste sarebbe necessario capire con il cliente tutte le informazioni aggiuntive, anche rispetto all’applicativo realizzato, che bisogna inserire, siccome i dati potrebbero essere utilizzati più avanti per analisi statistiche o altro.

Bisogna sapere se i dati utente devono essere tutti memorizzati nel nuovo database (mail, password, dati personali, ecc..) o se, essendo un nuovo servizio di una azienda preesistente, verranno usati i dati presenti nei db legacy.

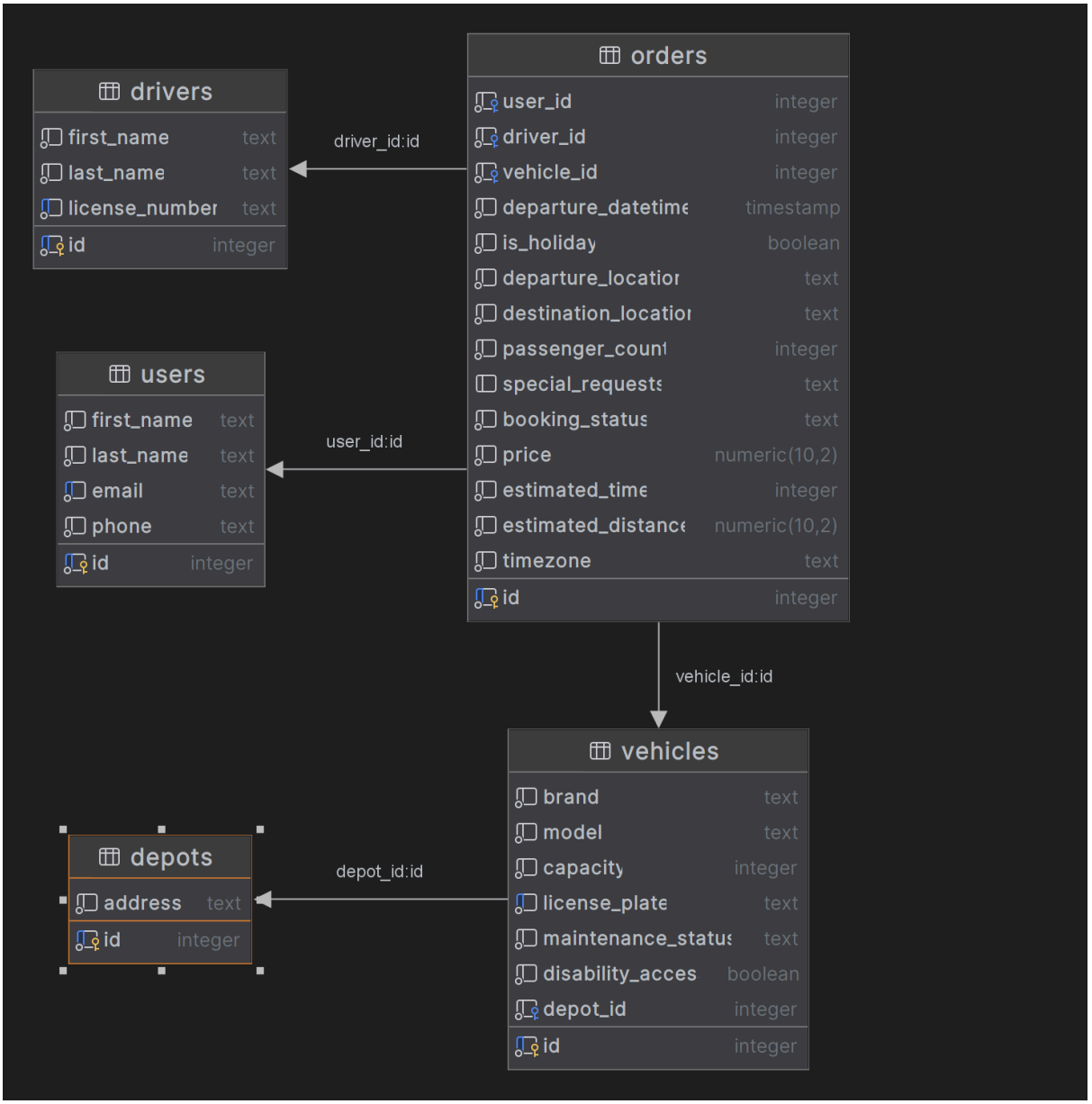
Occorre poi capire quali dati personali degli autisti, quali caratteristiche dei veicoli e quali informazioni sulle rimesse devono essere presenti nelle rispettive tabelle.

Esercizio 1b

Progetta il database (relazionale) definendo come meglio preferisci (puoi usare un tool grafico o semplicemente i comandi DDL). N.B.: Progetta il DB per come lo hai appreso nel caso tu abbia espresso dei dubbi al punto “1a” (se ritieni vincolanti alcuni dubbi puoi sempre contattarci!). Per evitare di rendere l'esercizio eccessivamente oneroso puoi evitare di entrare nel

dettaglio delle varie entità e concentrati sulla parte operativa della funzionalità. Puoi quindi evitare dettagli non richiesti (ad esempio la componente economica o componenti non strettamente richiesti dal requisito) e cerca di definire solo gli attributi indispensabili per rispondere al requisito appena descritto (ad esempio, se decidi di definire una tabella Cliente usa solo le informazioni basilari).

Il DDL per creare il Database è presente nel file create_db.sql



Ho utilizzato **Postgres** e le tabelle sono strutturate immaginando di dare per scontato le diverse preferenze e informazioni secondarie (indicando quindi solo le informazioni basilari).

La tabella **users**, così come quella **drivers**, contiene solo le informazioni necessarie e un relativo id assegnato in modo autoincrementale.

La tabella **depots** in questo caso necessita solo dell'indirizzo per il servizio di terze parti e il calcolo delle tempistiche (quanto tempo prima partire dalla rimessa).

La tabella **vehicles** ha informazioni base sulle macchine (marca, modello ecc..), la capacità che può essere scelta dall'utente, lo status di manutenzione (quindi se è disponibile) e l'accesso per persone con handicap

La tabella **orders** è la più grande: oltre ad avere gli id come foreign keys, ha il tempo di partenza, un booleano per sapere se è un giorno di vacanza (non è possibile ottenerlo internamente da python se non con servizi terzi o integrazioni con calendari locali alla regione in cui il servizio è attivo), luogo di partenza e di arrivo, il conto dei passeggeri, eventuali richieste speciali, lo stato dell'ordine e il costo (deciso di inserirlo come valore, anche in ottica dell'esercizio 3, ipotizzando che la parte di fatturazione venga salvata al di fuori di questo DB).
Dopo di che, vi sono tempo stimato (in minuti, eventualmente), distanza stimata richiesti dal servizio terzo e la timezone per svolgere eventuali calcoli e poterla ottenere velocemente.

Per provare, con un server Postgres attivo, eseguire in ordine i file create_db.sql e populate_tables.sql

Esercizio 2

A partire dal DB appena progettato, definisci una query per estrarre questa informazione: nome, cognome e distanza totale percorsa del secondo autista per distanza totale percorsa tra tutti gli autisti che hanno effettuato esattamente 2 corse il giorno "31-12-2023". Se lo desideri, usa anche comandi avanzati del DBSM che preferisci (Postgres, MySQL, ...).

La query è anche presente nel file query.sql più ordinata

```
WITH driver_trip_counts AS (
                    SELECT drivers.id           AS driver_id,
                        drivers.first_name,
                        drivers.last_name,
                        SUM(orders.estimated_distance) AS total_distance,
                        COUNT(orders.id)       AS trip_count
                    FROM drivers
                        JOIN orders ON drivers.id = orders.driver_id
                    WHERE DATE(orders.departure_datetime) = '2023-12-31'
                    GROUP BY drivers.id, drivers.first_name, drivers.last_name
                    HAVING COUNT(orders.id) = 2),
        ranked_drivers AS
                    SELECT driver_id,
                        first_name,
                        last_name,
                        total_distance,
                        ROW_NUMBER() OVER (ORDER BY total_distance DESC)
                                AS rank
                    FROM driver_trip_counts)
SELECT first_name,
      last_name,
      total_distance
FROM ranked_drivers
WHERE rank = 2;
```

Prima creo una tabella temporanea con i dati degli autisti, la distanza totale percorsa e il conto degli ordini, solo del giorno richiesto e che hanno svolto solo due ordini quel giorno.
Poi creo una seconda tabella temporanea per riordinare gli autisti per distanza percorsa.
Infine, ottengo i tre dati richiesti del secondo in classifica

Esercizio 3

Considera una ipotetica tabella costi relativa agli appuntamenti. Implementa in Python con il framework che preferisci (es: FastAPI, Flask) due endpoint per inserire ed aggiornare i costi degli appuntamenti. Utilizza i metodi REST che ritieni più appropriati. I prezzi hanno una tariffa base di 10€ più un supplemento chilometrico di 1€ a KM. Altri supplementi sono per le fasce orarie (5% dalle 18 alle 24; 12% dalle 24 alle 6 del mattino) e le domeniche/festivi (+10%). Utilizza l'approccio che desideri. (N.B. per facilitare l'esercizio, puoi tranquillamente mockare le funzionalità di basso livello.)

Il codice è presente nel metodo main della repo