Laurea in
Ingegneria Informatica

Tesina Basi di Dati

prof. Vincenzo Moscato

N46/3287	Fabio d'Andrea
N46/3528	Ludovica De Luca
N46/3444	Antimo Iannucci

Dipartimento di Ingegneria Elettrica
e Tecnologia dell'Informazione

Universita' degli Studi di Napoli "Federico II"

Anno Accademico 2017/2018
Secondo Semestre

Contents

Chapter 1

Introduzione

Si vuole presentare la progettazione di un sistema informatico legato alla gestione di un sistema di *car pooling* sul territorio nazionale. L'obiettivo del progetto sarà diffondere l'uso di una mobilità flessibile e personalizzata in termini di percorsi e costi.

1.1 Specifiche

Si vuole progettare una base di dati utile a supportare il sistema precedentemente introdotto.

1.1.1 Specifiche sui dati

Dal colloquio con il committente sono emerse le seguenti specifiche sui dati da gestire:

1. **Utenti** della piattaforma, di cui si vogliono memorizzare generalità, recapito telefonico, indirizzo e-mail e foto personale. A ciascuno di essi saranno poi assegnate credenziali di accesso in termini di username e password.
   Gli utenti possono essere di due tipi:

   - **Utenti-Autisti**, coloro che offrono un passaggio, di cui interessano i dati relativi a patente di guida e veicolo.
   - **Utenti-Passeggeri**, coloro che usufruiscono del passaggio.

2. **Viaggi**, condivisi dagli autisti, di cui sono indicate **città di partenza** e **città di destinazione**, data e ora di partenza, tempi di percorrenza stimati, contributo economico e numero di posti (totali e disponibili).

3. **Prenotazioni**, effettuate dai passeggeri e accettate dagli autisti.
4. **Recensioni**, in termini di voto numerico e giudizio discorsivo, su autisti/passeggeri, relative ad un dato viaggio ed effettuate da passeggeri/autisti.

Nel primo anno di esercizio del sistema, dovranno essere gestiti:

- 10'000 utenti, di cui 2'000 autisti e 8'000 passeggeri.
- 8'000 comuni.
- 100'000 viaggi e 500'000 prenotazioni.

1.1.2 Specifiche sulle operazioni

Le principali operazioni previste sulla base di dati sono:

1. Registrazione di utenti.
2. Inserimento di viaggi.
3. Consultazione e prenotazioni di viaggi.
4. Inserimento di commenti e voti per autisti e passeggeri.
5. Invio e-mail di accettazione o rifiuto relativi alle prenotazioni.

1.2 Analisi delle specifiche

Al termine della raccolta dei requisiti, si è scelto di analizzare nel dettaglio i requisiti statici richiesti dall'applicazione al fine di averne una visione più chiara.

Informazioni generali

Si vuole progettare una base di dati per la gestione di un sistema di car pooling, che contenga informazioni relative agli utenti del sistema, ai viaggi offerti, alle prenotazioni effettuate e alle recensioni inserite.

Informazioni sugli utenti

In relazione agli utenti devono essere memorizzate le usuali informazioni anagrafiche, credenziali di accesso (username e password), recapito telefonico, indirizzo e-mail e foto personale.

Informazioni sugli autisti

Sarà necessario estendere le informazioni inerenti a ciascun autista aggiungendo i dati relativi alla patente di guida (numero e scadenza) e al veicolo messo a disposizione.

Informazioni sui passeggeri

Per i passeggeri saranno sufficienti le informazioni relative al generico utente.

Informazioni sui viaggi

Per ciascun viaggio si vogliono memorizzare le informazioni relative all'autista del viaggio, luogo, data e ora di partenza, luogo di arrivo, data e ora previste per l'arrivo (con relativa durata stimata del viaggio), numero di posti totali e disponibili, prezzo e possibilità di portare eventuali bagagli e/o animali.

Informazioni sulle prenotazioni

Ciascuna prenotazione, identificata da un codice, conterrà le informazioni relative al passeggero da cui è stata effettuata e il viaggio a cui fa riferimento.

Informazioni sulle recensioni

Per ogni recensione devono essere memorizzati i relativi feedback (voto numerico e giudizio discorsivo), colui che inserisce la recensione, il soggetto della recensione ed il viaggio a cui fa riferimento.

Chapter 2

Progettazione della base di dati

Espressi i requisiti in modo chiaro è possibile procedere con la progettazione della base dati, secondo le seguenti fasi:

1. **Progettazione concettuale**: trasformazione dei requisiti dal linguaggio naturale al modello concettuale, operandone una ristrutturazione al fine di eliminare eventuali ambiguità.
2. **Progettazione logica**: trasformazione del modello concettuale in modello logico.
3. **Progettazione fisica**: implementazione del modello logico sul DBMS.

2.1 Progettazione concettuale

Per la progettazione concettuale, essendo la base di dati di medie dimensioni, si è scelta una strategia di approcio ibrida, sfruttando i vantaggi di entrambe le strategie **top-down** e **bottom-up**.

2.1.1 Schema E/R portante

Per prima cosa si estrae quindi dalle specifiche ristrutturate lo **schema E/R portante**, successivamente esteso e raffinato. In questa prima approssimazione, sono presenti:

- 5 **entità**: `UTENTE`, `AUTISTA`, `PASSEGGERO`, `PRENOTAZIONE`, `VIAGGIO`
- 3 **relazioni**: `RICHIESTA`, `SERVIZIO`, `ACCETTAZIONE`

Si è deciso di inserire nello schema portante la gerarchia orizzontale `UTENTE-AUTISTA-PASSEGGERO` per evidenziare l'esistenza di due tipologie differenti di utenti della piattaforma.

Si riporta di seguito lo schema E/R portante.

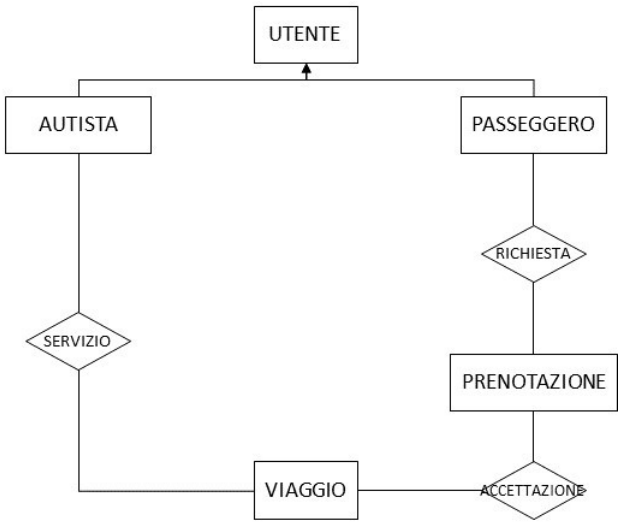

Figure 2.1: Schema E/R portante

2.1.2 Schema E/R finale

Esaminando attentamente le specifiche del progetto, risulta evidente la necessità di prendere in considerazione ulteriori entità ed associazioni.

Si sono quindi inserite le seguenti **entità** con i relativi **attributi**:

- `RECENSIONE`, necessaria per la memorizzazione dei feedback forniti da autisti e passeggeri.
- `LUOGO`, in termini di coordinate geografiche (latitudine e longitudine), necessaria per specificare partenza e destinazione di un dato viaggio. Si è scelto di inserire tale entità per permettere agli utenti di condividere viaggi all'interno di una stessa città.
- `CITTA'`, a cui appartiene ciascun luogo.
- `AUTO`, posseduta da ciascun autista.

E le seguenti **associazioni**:

- `RECENSENTE` e `RECENSITO`, leganti le entità `RECENSIONE` e `UTENTE`. Sono necessarie due associazioni per specificare colui che fornisce la recensione e il soggetto stesso di quest'ultima.

- `FEEDBACK`, legante le entità `RECENSIONE` e `VIAGGIO`, in quanto ciascuna recensione fa riferimento ad un dato viaggio.

- `PARTENZA` e `ARRIVO`, leganti le entità `LUOGO` e `VIAGGIO`, a cui sono aggiunti gli attributi inerenti a data e ora di partenza e di arrivo previste.

- `TAPPA`, legante le entità `LUOGO` e `VIAGGIO`. Si è scelto di inserire questa entità per offrire la possibilità di specificare se si è intenzionati ad effettuare soste intermedie durante un viaggio.

- `APPARTENENZA`, legante le entità `LUOGO` e `CITTA'`.

- `PROPRIETA'`, legante le entità `AUTISTA` e `AUTO`.

Per ciò che riguarda le **cardinalità** delle associazioni sono state fatte le seguenti considerazioni:

- Un dato `AUTISTA` può offrire come servizio nessuno o più viaggi (0,N), mentre un `VIAGGIO` è relativo ad uno e un solo autista (1,1).

- Un `PASSEGGERO` può effettuare nessuna o più prenotazioni (0,N), una `PRENOTAZIONE` è relativa ad uno e un solo passeggero (1,1).

- Una data `PRENOTAZIONE` può essere riferita ad uno e un solo viaggio (1,1), ad un `VIAGGIO` possono invece essere associate nessuna o più prenotazioni (0,N).

- Una `RECENSIONE` può essere effettuata da/riferita ad un solo utente (1,1), un `UTENTE` può effettuare/essere oggetto di alcuna o più recensioni (0,N).

- Una `RECENSIONE` è riferita ad uno e un solo viaggio (1,1), uno stesso `VIAGGIO` può essere associato a nessuna o più recensioni (0,N).

- Un dato `VIAGGIO` può avere uno e un solo luogo di partenza/arrivo (1,1), mentre può offrire la possibilità di effettuare nessuna o più tappe in un dato luogo (0,N); un `LUOGO` può invece essere associato ad un viaggio nessuna o più volte come luogo di partenza/arrivo o tappa(0,N).

- Un `LUOGO` può appartenere ad una e una sola città (1,1), viceversa una città può contenere alcuno o più luoghi di interesse (0,N).
- Si suppone che un `AUTISTA` possa indicare come proprio uno e un solo veicolo (1,1) e viceversa un'`AUTO` appartenga ad un unico autista (1,1).

Infine, si è scelto per motivi pratici di separare completamente i due ruoli utente di autista e passeggero, specificando per la gerarchia i vincoli:

- *vincolo di copertura*: **totale**
- *vincolo di disgiunzione*: **disgiunta**

Si riporta di seguito lo schema E/R trasformato.

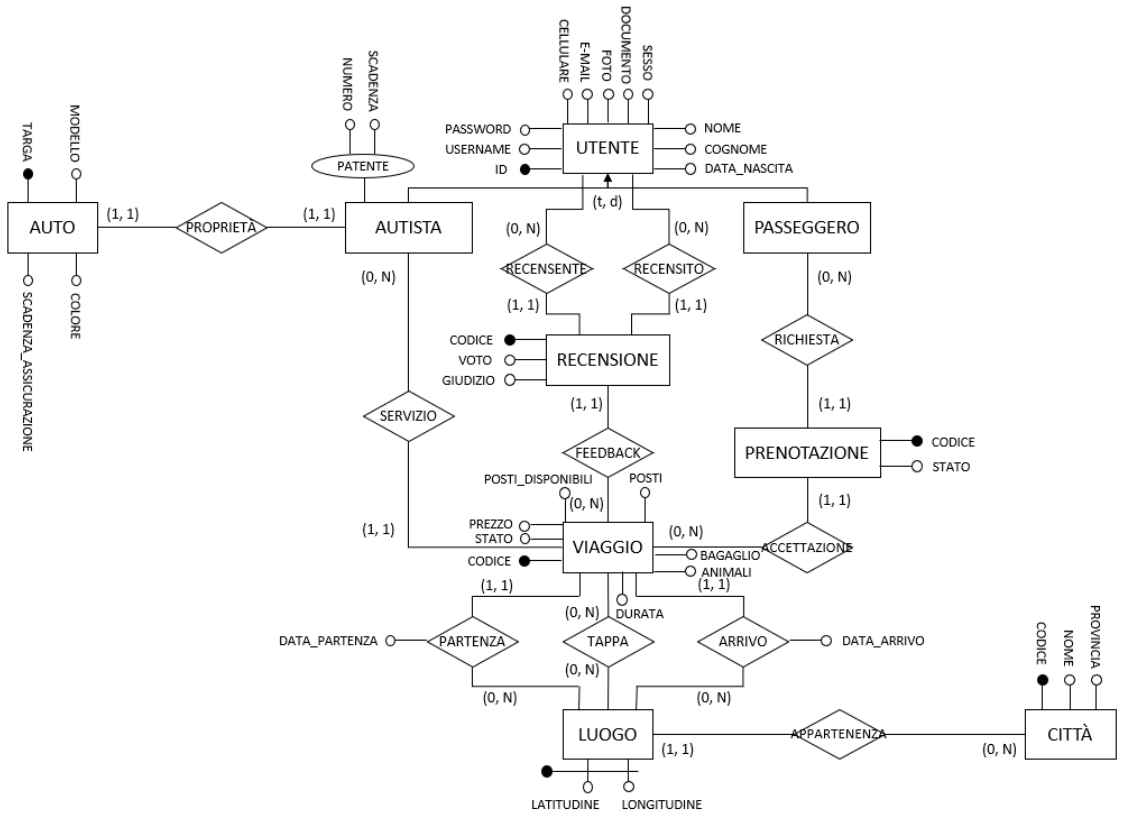

Figure 2.2: Schema E/R finale

2.2 Progettazione Logica

Fase successiva della progettazione è la progettazione logica, attraverso cui si definisce lo **schema relazionale** della base di dati.
A sua volta è composta da due fasi:

1. **trasformazione** dello schema concettuale in uno schema semplificato, ottenuto mediante l'eliminazione di attributi composti e multivalore, e la risoluzione delle gerarchie.

2. **traduzione** dello schema semplificato nello schema relazionale.

2.2.1 Trasformazione

Facendo riferimento allo schema della figura 2.2, la trasformazione consisterà nell'eliminazione dell'unico attributo composto `PATENTE` e nella risoluzione della gerarchia `UTENTE-AUTISTA-PASSEGGERO`

Eliminazione attributi composti e multivalore

L'attributo composto `PATENTE`, presente nell'entità `AUTISTA`, viene eliminato e sostituito dai suoi due attributi componenti `NUMERO` e `SCADENZA`.

Se fossero stati presenti attributi multivalore, si sarebbe proceduto con la relativa trasformazione secondo la seguente regola:
l'attributo multivalore X, presente nell'entità Y, viene eliminato e sostituito da una nuova entità avente lo stesso nome e legata all'entità Y da un'associazione avente rapporto di cardinalità uno a molti. L'entità Y avrà gli stessi attributi, escluso X.

Risoluzione gerarchie

Per la risoluzione della gerarchia `UTENTE-AUTISTA-PASSEGGERO` si è scelta la **soluzione 3**, ovvero la sostituzione della generalizzazione con tante associazioni quante sono le sottoclassi.

Sono state quindi introdotte due associazioni `U-A`, `U-B`, con rapporto di cardinalità uno a uno (1,1) (opzionale dal lato `UTENTE`), e ciascuna delle sottoclassi è identificata esternamente dall'identificatore della superclasse.

Le motivazioni di tale scelta sono legate alla volontà di operare una netta separazione tra i due diversi tipi di utente, dovuta alle differenti le operazioni effettuate da ciascuno di essi.

Si riporta di seguito lo schema E/R trasformato.

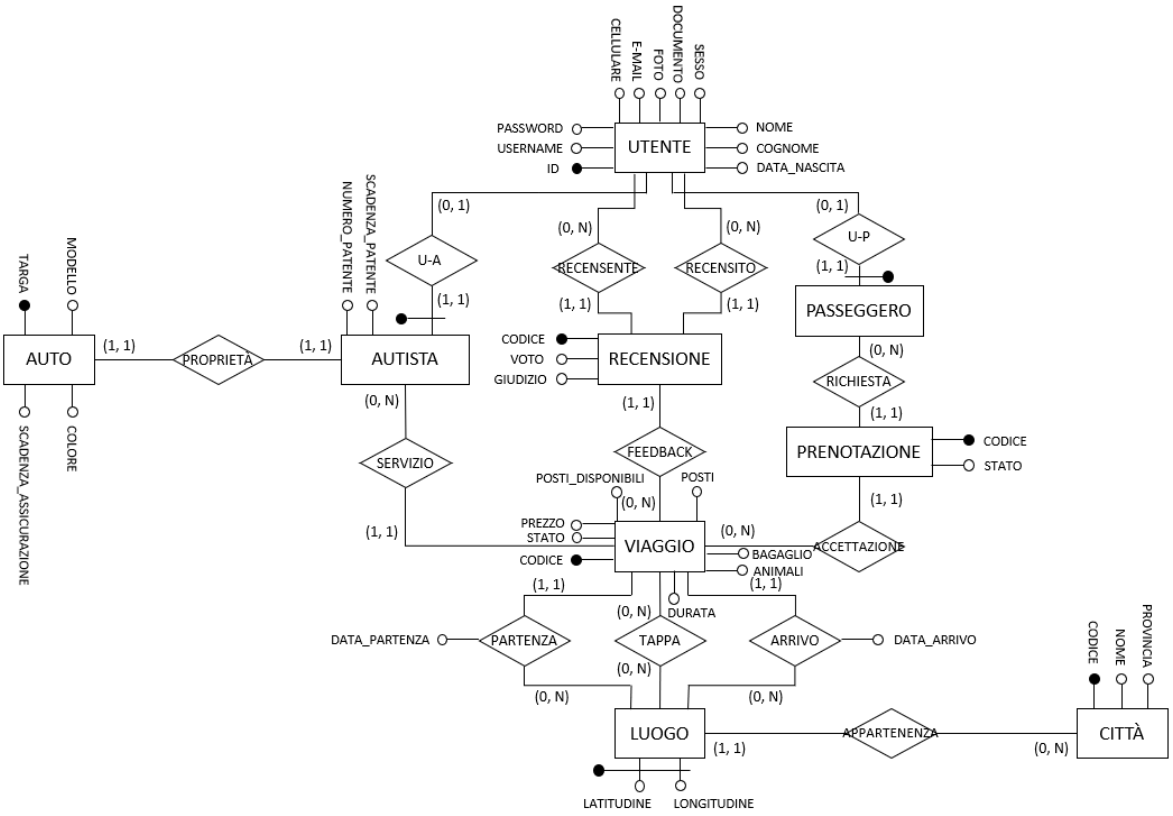

Figure 2.3: Schema E/R trasformato

2.2.2 Traduzione

A valle dello schema E/R semplificato ottenuto, si costruisce lo schema relazionela della base di dati, in termini di **relazioni** e **vincoli** definiti su di esse.
La fase di traduzione comporta le seguenti azioni:

1. Ogni **entità** si traduce in una relazione avente stesso nome ma al plurale, stessi attributi e come chiave primaria l'identificatore dell'entità stessa.

2. L'unica **associazione molti a molti (N,N)** (`TAPPA`) si traduce in una relazione avente stesso nome ma al plurale, come attributi gli attributi dell'associazione (in questo caso assenti) e gli identificatori delle

entità coinvolte nell'associazione. Tali identificatori saranno chiave primaria della relazione e avranno un vincolo di chiave esterna riferito al corrispondente attributo dell'entità di provenienza.

3. Ogni **associazione uno a molti (1,N)** (`RECENSENTE`, `RECENSITO`, `RICHIESTA`, `ACCETTAZIONE`, `SERVIZIO`, `PARTENZA`, `ARRIVO`, `APPARTENENZA`) si traduce aggiungendo l'identificatore dell'entità dal lato molti, opportunamente ridenominato, come attributo dell'entità dal lato uno. Tale attributo avrà un vincolo di chiave esterna riferito al corrispondente attributo dell'entità dal lato molti.

4. Ogni **associazione uno a uno (1,1)** (`U-P`, `U-A`, `PROPRIETA'`) si traduce aggiungendo l'identificatore di una delle due entità, opportunamente ridenominato, come attributo dell'altra entità. Tale attributo avrà un vincolo di chiave esterna riferito al corrispondente attributo dell'altra entità.

Attraverso l'applicazione di tali regole, si è giunti quindi alla realizzazione del seguente schema relazionale:

UTENTI (ID, username, password, nome, cognome, dataNascita, sesso, documento, foto, e-mail, cellulare)

AUTISTI (utente: `UTENTI`, numeroPatente, scadenzaPatente, auto: `AUTO`)

PASSEGGERI (utente: `UTENTI`)

AUTO (targa, modello, colore, scadenzaAssicurazione)

RECENSIONI (codice, voto, giudizio, recensente: `UTENTI`, recensito: `UTENTI`, viaggio: `VIAGGI`)

PRENOTAZIONI (codice, passeggero: `PASSEGGERI`, viaggio: `VIAGGI`, stato)

VIAGGI (codice, prezzo, durata, posti, postiDisponibili, stato, autista: `AUTISTI`, latitudinePartenza: `LUOGHI`, longitudinePartenza: `LUOGHI`, dataPartenza, latitudineArrivo: `LUOGHI`, longitudineArrivo: `LUOGHI`, dataArrivo, bagaglio, animali)

CITTÀ (nome, CAP, provincia)

LUOGHI (latitudine, longitudine, città: `CITTÀ`)

TAPPE (viaggio: `VIAGGI`, latitudine: `LUOGHI`, longitudine: `LUOGHI`)

2.3 Progettazione fisica

Per concludere, la progettazione fisica della base di dati prevede l'attuazione delle seguenti fasi:

1. dimensionamento fisico della base di dati, in termini di calcolo dello **storage**, spazio di memorizzazione richiesto.
2. creazione del database.
3. definizione delle politiche di sicurezza e creazione di ruoli/utenti.
4. creazione oggetti della base di dati e definizione dei vincoli.
5. popolamento della base di dati.

2.3.1 Dimensionamento fisico della base di dati

Date le informazioni sul volume dei dati, scelti i tipi degli attributi utilizzati nell'implementazione delle tabelle, è possibile effettuare una stima dei costi in termini di occupazione di memoria della base di dati.

In particolare sono stati utilizzati i seguenti tipi di dato:

- **NUMBER(x)**, storage: $\lceil(x/2)+2\rceil$ byte. Utilizzato per la memorizzazione di valori numerici, in particolare, per la rappresentazione di codici identificativi numerici.
- **FLOAT**, storage: 22 byte. Utilizzato per la memorizzazione di valori numerici reali.
- **CHAR(x)**, storage: x byte. Utilizzato per la memorizzazione di attributi di tipo stringa di caratteri di lunghezza fissata.
- **VARCHAR2(x)**, storage: da 0 a x byte. Utilizzato per la memorizzazione di attributi di tipo stringa di caratteri di lunghezza variabile.
- **DATE**, storage: 7 byte. Utilizzato per la memorizzazione di attributi di tipo data e ora.
- **INTERVAL DAY TO SECOND**, storage 7 byte. Utilizzato per la memorizzazione di intervalli di tempo.
- **BLOB**, storage: 4 byte. Utilizzato per la memorizzazione di immagini, video o audio, attraverso un puntatore all'effettivo tablespace contenente i dati. In particolare, è stato utilizzato per la gestione delle fotografie degli utenti.

- **CLOB**, storage: 4 byte. Utilizzato per la memorizzazione di testi di grandi dimensioni (fino a 4 GB), attraverso un puntatore all'effettivo tablespace constenenti i dati. In particolare è stato utilizzato per la gestione delle recensioni effettuate dagli utenti.

Di seguito viene riportata per ogni tabella una stima dell'occupazione di memoria all'atto della messa in esercizio della base di dati.

Si noti che le occorrenze relative alle tabelle `LUOGHI` e `TAPPE`, sebbene non esplicitamente espresse nelle specifiche, sono state ricavate dai dati disponibili supponendo una media di 8 luoghi di interesse per città e una media di 3 tappe intermedie per viaggio.

UTENTI			
ATTRIBUTO	TIPO	BYTE	
ID	NUMBER(6)	5	
Username	VARCHAR2(20)	20	
Password	VARCHAR2(20)	20	
Nome	VARCHAR2(50)	50	
Cognome	VARCHAR2(50)	50	
Data_nascita	DATE	7	
Sesso	CHAR(1)	1	
Documento	CHAR(9)	9	
Foto	BLOB	4	
E-mail	VARCHAR2(100)	100	
Cellulare	NUMBER(10)	7	
		273	TOTALE RECORD
		2,73 M	TOTALE 10'000 occ.

Figure 2.4: Dimensionamento tabella `UTENTI`

AUTISTI			
ATTRIBUTO	TIPO	BYTE	
Utente	NUMBER(6)	5	
Numero_patente	CHAR(10)	10	
Scadenza_patente	DATE	7	
Auto	CHAR(7)	7	
		29	TOTALEALE RECORD
		58 K	TOTALE 2'000 occ.

Figure 2.5: Dimensionamento tabella `AUTISTI`

PASSEGGERI			
ATTRIBUTO	TIPO	BYTE	
Utente	NUMBER(6)	5	
		5	TOTALE RECORD
		40 K	TOTALE 8'000 occ.

Figure 2.6: Dimensionamento tabella `PASSEGGERI`

AUTO			
ATTRIBUTO	TIPO	BYTE	
Targa	CHAR(7)	7	
Modello	VARCHAR2(50)	50	
Colore	VARCHAR2(30)	30	
Scadenza_Assicurazione	DATE	7	
		94	TOTALE RECORD
		188 K	TOTALE 2'000 occ.

Figure 2.7: Dimensionamento tabella `AUTO`

RECENSIONI			
ATTRIBUTO	TIPO	BYTE	
Codice	NUMBER(6)	5	
Voto	NUMBER(1)	2,5	
Giudizio	CLOB	4	
Recensente	NUMBER(6)	5	
Recensito	NUMBER(6)	5	
Viaggio	NUMBER(6)	5	
		26,5	TOTALE RECORD
		13,25 M	TOTALE 500'000 occ.

Figure 2.8: Dimensionamento tabella `RECENSIONI`

PRENOTAZIONI			
ATTRIBUTO	TIPO	BYTE	
Codice	NUMBER(6)	5	
Passeggero	NUMBER(6)	5	
Viaggio	NUMBER(6)	5	
Stato	VARCHAR2(20)	20	
		35	TOTALE RECORD
		17,5 M	TOTALE 500'000 occ.

Figure 2.9: Dimensionamento tabella `PRENOTAZIONI`

VIAGGI			
ATTRIBUTO	TIPO	BYTE	
Codice	NUMBER(6)	5	
Prezzo	FLOAT	22	
Durata	INTERVAL DAY TO SECOND	7	
Posti	NUMBER(1)	2,5	
Posti_disponibili	NUMBER(1)	2,5	
Autista	NUMBER(6)	5	
Latitudine_partenza	FLOAT	22	
Longitudine_partenza	FLOAT	22	
Data_partenza	DATE	7	
Latitudine_arrivo	FLOAT	22	
Longitudine_arrivo	FLOAT	22	
Data_arrivo	DATE	7	
Stato	VARCHAR2(20)	20	
Bagaglio	VARCHAR2(2)	2	
Animali	VARCHAR2(2)	2	
		170	TOTALE RECORD
		17 M	TOTALE 100'000 occ.

Figure 2.10: Dimensionamento tabella `VIAGGI`

CITTA'			
ATTRIBUTO	TIPO	BYTE	
Nome	VARCHAR2(30)	30	
Codice	NUMBER(5)	4,5	
Provincia	CHAR(2)	2	
		36,5	TOTALE RECORD
		292 K	TOTALE 8'000 occ.

Figure 2.11: Dimensionamento tabella `CITTÀ`

LUOGHI			
ATTRIBUTO	TIPO	BYTE	
Latitudine	FLOAT	22	
Longitudine	FLOAT	22	
Citta'	NUMBER(5)	4,5	
		48,5	TOTALE RECORD
		3,2 M	TOTALE 64'000 occ.

Figure 2.12: Dimensionamento tabella `LUOGHI`

TAPPE			
ATTRIBUTO	TIPO	BYTE	
Viaggio	NUMBER(6)	5	
Latitudine	FLOAT	22	
Longitudine	FLOAT	22	
		49	TOTALE RECORD
		14,7 M	TOTALE 300'000 occ.

Figure 2.13: Dimensionamento tabella `TAPPE`

Infine, si riporta la tabella relativa al dimensionamento totale della base di dati.

Si è scelto di utilizzare oggetti di tipo **indice** per velocizzare l'accesso agli utenti sulla base del proprio username e oggetti di tipo **sequanza** per la gestione dell'inserimento di codici/ ID progressivi sulle tabelle `UTENTI`, `RECENSIONI`, `PRENOTAZIONI`, `VIAGGI`.

DIMENSIONAMENTO FINALE			
OGGETTO	NOME	INITIAL	NEXT
Tabella	UTENTI	2730 K	-
Tabella	AUTISTI	58 K	-
Tabella	PASSEGGERI	40 K	-
Tabella	AUTO	188 K	-
Tabella	RECENSIONI	13250 K	-
Tabella	PRENOTAZIONI	17500 K	-
Tabella	VIAGGI	17000 K	-
Tabella	CITTA	292 K	-
Tabella	LUOGHI	3200 K	-
Tabella	TAPPE	14700 K	-
Indice	UTENTI_IDX	200 K	-
Sequenza	IDUtenti_Gen	16 B	-
Sequenza	codRecensioni_Gen	16 B	-
Sequenza	codPrenotazioni_Gen	16 B	-
Sequenza	codViaggio_Gen	16 B	-
Sequenza	codCitta_Gen	16 B	-

Figure 2.14: Dimensionamento finale

TABLESPACE		
	TOTALE	APPROSSIMAZIONE
Data Base	69 M	150 M
Immagini	1 G	1 G
Giudizi	5 M	10 M

Figure 2.15: Tablespace

2.3.2 Creazione dei tablespace

Si è ritenuto opportuno separare la **struttura logica** del database in apposite aree logiche di memorizzazione, dette **tablespace**, dove sono memorizzati i diversi oggetti.
In particolare, sono stati creati tre tablespace:

- il primo atto a contenere i dati relativi alle diverse tabelle.
- il secondo dedicato alla memorizzazione delle immagini relative agli utenti della piattaforma. Si è supposto che ciascuna immagini occupi in media 100 KB.
- il terzo dedicato alla memorizzazione dei giudizi discorsivi relativi alle recensioni inserire dagli utenti. Si è supposto che ciascun giudizio occupi in media 500 byte.

Il codice SQL relativo alla creazione dei tablespace è il seguente (operazione effettuata tramite le credenziali di SYSTEM):

```
CREATE TABLESPACE DATI_PROGETTO_TS DATAFILE
'C:\File_progetto\DATI_PROGETTO_TS.dbf' SIZE 150 M;

CREATE TABLESPACE FOTO_PROGETTO_TS DATAFILE
'C:\File_Progetto\FOTO_PROGETTO_TS.dbf' SIZE 1 G;

CREATE TABLESPACE GIUDIZI_PROGETTO DATAFILE
'C:\File_progetto\GIUDIZI_PROGETTO_TS.dbf' SIZE 10 M;
```

2.3.3 Creazione di ruoli/utenti e definizione delle politiche di sicurezza

Per ciò che riguarda le politiche di sicurezza, si è deciso di creare un utente *dba*, proprietario della base di dati, due ruoli relativi ad autisti e passeggeri, aventi differenti privilegi sulla base di dati, e due utenti, *autista* e *passeggero*, attraverso cui si connetteranno gli utenti della piattaforma.

Il codice SQL relativo alla creazione dell'utente dba è il seguente (operazioni effettuata tramite le credenziali di SYSTEM):

```
CREATE USER DBA_PROGETTO DEFAULT TABLESPACE DATI_PROGETTO_TS
    IDENTIFIED BY 202208;
GRANT DBA, UNLIMITED TABLESPACE TO DBA_PROGETTO;
```

Il codice SQL relativo alla creazione dei ruoli e degli utenti è il seguente (operazioni effettuate tramite le credenziali del dba creato):

```
CREATE ROLE RUOLO_AUTISTA;
GRANT CONNECT TO RUOLO_AUTISTA;
GRANT SELECT, INSERT, UPDATE, DELETE ON AUTO TO RUOLO_AUTISTA;
GRANT SELECT, INSERT, UPDATE, DELETE ON VIAGGI TO RUOLO_AUTISTA;
GRANT SELECT, INSERT, UPDATE, DELETE ON RECENSIONI TO RUOLO_AUTISTA;
GRANT SELECT, UPDATE ON PRENOTAZIONI TO RUOLO_AUTISTA;
GRANT SELECT ANY SEQUENCE TO RUOLO_AUTISTA;

CREATE USER AUTISTA IDENTIFIED BY 0820;
GRANT RUOLO_AUTISTA TO AUTISTA;

CREATE ROLE RUOLO_PASSEGGERO;
GRANT CONNECT TO RUOLO_PASSEGGERO;
GRANT INSERT, DELETE ON PRENOTAZIONI TO RUOLO_PASSEGGERO;
GRANT SELECT, INSERT, DELETE ON RECENSIONI TO RUOLO_PASSEGGERO;
GRANT SELECT ANY SEQUENCE TO RUOLO_PASSEGGERO;

CREATE USER PASSEGGERO IDENTIFIED BY 0822;
GRANT RUOLO_PASSEGGERO TO PASSEGGERO;
```

L'ultimo privilegio dato a ciascun ruolo dà la possibilità ad un utente/applicazione di tipo autista/passeggero di utilizzare le sequenze numeriche per la gestione dei codici/ ID numerici progressivi sulle relative tabelle.

2.3.4 Creazione oggetti della base di dati

Di seguito è riportato il codice SQL relativo alla creazione di tutti gli oggetti della base di dati (operazioni effettuate tramite le credenziali del dba):

```
CREATE TABLE UTENTI (
     ID NUMBER(6),
     USERNAME VARCHAR2(20) NOT NULL UNIQUE,
     PASSWORD VARCHAR2(20) NOT NULL,
     NOME VARCHAR2(50) NOT NULL,
     COGNOME VARCHAR2(50) NOT NULL,
     DATA_NASCITA DATE,
     SESSO CHAR(1) CHECK (SESSO = 'M' OR SESSO = 'F'),
     DOCUMENTO CHAR(9) UNIQUE,
     FOTO BLOB,
     EMAIL VARCHAR2(100) NOT NULL,
     CELLULARE NUMBER(10) NOT NULL,
     CONSTRAINT PK_UTENTI PRIMARY KEY (ID)
)
LOB(FOTO) STORE AS LOB_FOTO (TABLESPACE FOTO_PROGETTO_TS)
STORAGE (INITIAL 2730 K);


CREATE TABLE AUTISTI (
    UTENTE NUMBER(6),
    NUMERO_PATENTE CHAR(10) NOT NULL UNIQUE,
     SCADENZA_PATENTE DATE NOT NULL,
     AUTO CHAR(7),
     CONSTRAINT PK_AUTISTI PRIMARY KEY (UTENTE)
)
STORAGE (INITIAL 58 K);


CREATE TABLE PASSEGGERI (
     UTENTE NUMBER(6),
     CONSTRAINT PK_PASSEGGERI PRIMARY KEY (UTENTE)
)
STORAGE (INITIAL 40 K);
```

```
CREATE TABLE AUTO (
     TARGA CHAR(7),
     MODELLO VARCHAR2(50),
     COLORE VARCHAR2(30),
     SCADENZA_ASSICURAZIONE DATE NOT NULL,
     CONSTRAINT PK_AUTO PRIMARY KEY (TARGA)
)
STORAGE (INITIAL 188 K);


CREATE TABLE RECENSIONI (
    CODICE NUMBER(6),
     VOTO NUMBER(1) CHECK (VOTO > 0 AND VOTO < 6),
     GIUDIZIO CLOB,
     RECENSENTE NUMBER(6) NOT NULL,
     RECENSITO NUMBER(6) NOT NULL,
     VIAGGIO NUMBER(6) NOT NULL,
     CONSTRAINT PK_RECENSIONI PRIMARY KEY (CODICE)
)
LOB(GIUDIZIO) STORE AS LOB_GIUDIZIO
    (TABLESPACE GIUDIZI_PROGETTO_TS)
STORAGE (INITIAL 13250 K);


CREATE TABLE PRENOTAZIONI (
     CODICE NUMBER(6),
     PASSEGGERO NUMBER(6) NOT NULL,
     VIAGGIO NUMBER(6) NOT NULL,
     STATO VARCHAR2(20) DEFAULT 'In attesa'
            CHECK (STATO = 'Accettata' or STATO = 'Rifiutata'
            or STATO = 'In attesa'),
     CONSTRAINT PK_PRENOTAZIONI PRIMARY KEY (CODICE)
)
STORAGE (INITIAL 17500 K);
```

```
CREATE TABLE VIAGGI (
     CODICE NUMBER(6),
     PREZZO FLOAT NOT NULL,
     DURATA INTERVAL DAY TO SECOND,
     POSTI NUMBER(1) NOT NULL,
     POSTI_DISPONIBILI NUMBER(1) NOT NULL,
     AUTISTA NUMBER(6) NOT NULL,
     LATITUDINE_PARTENZA FLOAT,
     LONGITUDINE_PARTENZA FLOAT,
     DATA_PARTENZA DATE NOT NULL,
     LATITUDINE_ARRIVO FLOAT,
     LONGITUDINE_ARRIVO FLOAT,
     DATA_ARRIVO DATE,
     STATO VARCHAR2(20) DEFAULT 'Disponibile'
            CHECK (STATO = 'Disponibile' or STATO = 'Non disponibile'),
     BAGAGLIO VARCHAR2(2) DEFAULT 'No'
            CHECK (BAGAGLIO = 'Si' or BAGAGLIO = 'No'),
     ANIMALI VARCHAR2(2) DEFAULT 'No'
            CHECK (ANIMALI = 'Si' or ANIMALI = 'No'),
    CONSTRAINT PK_VIAGGI PRIMARY KEY (CODICE)
)
STORAGE (INITIAL 17000 K);


CREATE TABLE CITTA (
     NOME VARCHAR2(30) NOT NULL,
     CODICE NUMBER(5),
     PROVINCIA CHAR(2) NOT NULL,
     CONSTRAINT PK_CITTA PRIMARY KEY (CAP)
)
STORAGE (INITIAL 292 K);


CREATE TABLE LUOGHI (
     LATITUDINE FLOAT,
     LONGITUDINE FLOAT,
     CITTA NUMBER(5),
     CONSTRAINT PK_LUOGHI PRIMARY KEY (LATITUDINE, LONGITUDINE)
)
STORAGE (INITIAL 3200 K);
```

```
CREATE TABLE TAPPE (
     VIAGGIO NUMBER(6),
     LATITUDINE FLOAT,
     LONGITUDINE FLOAT,
     CONSTRAINT PK_TAPPE PRIMARY KEY (VIAGGIO, LATITUDINE, LONGITUDINE)
)
STORAGE (INITIAL 14700 K);


CREATE SEQUENCE IDUtenti_Gen START WITH 1 INCREMENT BY 1
    NOMAXVALUE NOCYCLE;

CREATE SEQUENCE codRecensioni_Gen START WITH 1 INCREMENT BY 1
    NOMAXVALUE NOCYCLE;

CREATE SEQUENCE codPrenotazioni_Gen START WITH 1 INCREMENT BY 1
    NOMAXVALUE NOCYCLE;

CREATE SEQUENCE codViaggio_Gen START WITH 1 INCREMENT BY 1
    NOMAXVALUE NOCYCLE;

CREATE SEQUENCE codCitta_Gen START WITH 1 INCREMENT BY 1
    NOMAXVALUE NOCYCLE;


CREATE INDEX UTENTI_IDX ON UTENTI(COGNOME)
STORAGE(INITIAL 500K);
```

2.3.5 Creazione vincoli di integrità referenziale

La creazione dei vincoli di integrità referenziale è ottenuta mediante dei comandi di `ALTER TABLE`. Il codice SQL relativo alla creazione dei vincoli di integrità referenziale è il seguente
(operazioni effettuate tramite le credenziali del dba):

```
ALTER TABLE AUTISTI
    ADD CONSTRAINT FK_AUTISTI1 FOREIGN KEY (UTENTE)
    REFERENCES UTENTI (ID) ON DELETE CASCADE;
ALTER TABLE AUTISTI
    ADD CONSTRAINT FK_AUTISTI2 FOREIGN KEY (AUTO)
    REFERENCES AUTO (TARGA) ON DELETE SET NULL;
```

```
ALTER TABLE PASSEGGERI
    ADD CONSTRAINT FK_PASSEGGERI FOREIGN KEY (UTENTE)
    REFERENCES UTENTI (ID) ON DELETE CASCADE;


ALTER TABLE RECENSIONI
    ADD CONSTRAINT FK_RECENSIONI1 FOREIGN KEY (RECENSENTE)
    REFERENCES UTENTI (ID) ON DELETE CASCADE;
ALTER TABLE RECENSIONI
    ADD CONSTRAINT FK_RECENSIONI2 FOREIGN KEY (RECENSITO)
    REFERENCES UTENTI (ID) ON DELETE CASCADE;
ALTER TABLE RECENSIONI
    ADD CONSTRAINT FK_RECENSIONI3 FOREIGN KEY (VIAGGIO)
    REFERENCES VIAGGI (CODICE);


ALTER TABLE PRENOTAZIONI
    ADD CONSTRAINT FK_PRENOTAZIONI1 FOREIGN KEY (PASSEGGERO)
    REFERENCES UTENTI (ID) ON DELETE CASCADE;
ALTER TABLE PRENOTAZIONI
    ADD CONSTRAINT FK_PRENOTAZIONI2 FOREIGN KEY (VIAGGIO)
    REFERENCES VIAGGI (CODICE) ON DELETE CASCADE;


ALTER TABLE VIAGGI
    ADD CONSTRAINT FK_VIAGGI1 FOREIGN KEY (AUTISTA)
    REFERENCES AUTISTI (UTENTE) ON DELETE CASCADE;
ALTER TABLE VIAGGI
    ADD CONSTRAINT FK_VIAGGI2
    FOREIGN KEY (LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA)
    REFERENCES LUOGHI (LATITUDINE, LONGITUDINE) ON DELETE CASCADE;
ALTER TABLE VIAGGI
    ADD CONSTRAINT FK_VIAGGI3
    FOREIGN KEY (LATITUDINE_ARRIVO, LONGITUDINE_ARRIVO)
    REFERENCES LUOGHI (LATITUDINE, LONGITUDINE) ON DELETE CASCADE;


ALTER TABLE LUOGHI
    ADD CONSTRAINT FK_LUOGHI FOREIGN KEY (CITTA)
    REFERENCES CITTA (CODICE) ON DELETE CASCADE;
```

```
ALTER TABLE TAPPE
    ADD CONSTRAINT FK_TAPPE1 FOREIGN KEY (VIAGGIO)
    REFERENCES VIAGGI (CODICE) ON DELETE CASCADE;
ALTER TABLE TAPPE
    ADD CONSTRAINT FK_TAPPE2 FOREIGN KEY (LATITUDINE, LONGITUDINE)
    REFERENCES LUOGHI (LATITUDINE, LONGITUDINE) ON DELETE CASCADE;
```

2.3.6 Popolamento della base di dati

Il popolamento della base di dati avviene attraverso statement SQL di `INSERT` o attraverso opportune procedure di import automatizzando la creazione delle istanze.
Di seguito sono riportati alcuni esempi di istruzioni di `INSERT` utilizzate per il popolamento:

```
INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'Antimo01', '0000',
    'Antimo', 'Iannucci', TO_DATE('1997-09-22', 'YYYY-MM-DD'), 'M',
    'AV8740319', 'antimo.iannucci@gmail.com', '3246587961');

INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'Fabio02', '0000',
    'Fabio', 'd Andrea', TO_DATE('1997-11-20', 'YYYY-MM-DD'), 'M',
    'AP9871037', 'fabiodandrea@gmail.com', '1258965743');

INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'Ludovica03', '0000',
    'Ludovica ', 'de Luca', TO_DATE('1997-10-08', 'YYYY-MM-DD'), 'M',
    'AV8740318', 'ludovica01@gmail.com', '3246587961');

INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'Giovanni04', '0000',
    'Giovanni', 'Rossi', TO_DATE('1990-08-07', 'YYYY-MM-DD'), 'M',
    'SI7601290', 'Giovannirossi@studenti.unina.it', '0125489632');
```

```
INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'MagoKlopp', '0000',
    'Yurgen', 'Klopp', TO_DATE('1974-02-23', 'YYYY-MM-DD'), 'M',
    'SO9876519', 'MagoKlopp@gmail.it', '6534897652');

INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'ElNino', '0000',
    'Fernando', 'Torres', TO_DATE('1984-05-04', 'YYYY-MM-DD'), 'M',
    'QP9871037', 'ElNino@hotmail.it', '3548965123');

INSERT INTO UTENTI (ID, USERNAME, PASSWORD, NOME, COGNOME,
    DATA_NASCITA, SESSO, DOCUMENTO, EMAIL, CELLULARE)
    VALUES (IDUtenti_Gen.nextval, 'ElPibe', '0000',
    'Diego', 'Maradona', TO_DATE('1966-09-02', 'YYYY-MM-DD'), 'M',
    'CP7012754', 'ElPibe@gmail.com', '3259861057');


INSERT INTO PASSEGGERI (UTENTE) VALUES ('000005');
INSERT INTO PASSEGGERI (UTENTE) VALUES ('000006');
INSERT INTO PASSEGGERI (UTENTE) VALUES ('000007');


INSERT INTO AUTO (TARGA, MODELLO, COLORE, SCADENZA_ASSICURAZIONE)
    VALUES ('AP234WE', 'Renault Megane', 'verde',
    TO_DATE('2020-05-02', 'YYYY-MM-DD'));

INSERT INTO AUTO (TARGA, MODELLO, COLORE, SCADENZA_ASSICURAZIONE)
    VALUES ('PO572FG', 'Fiat Panda', 'giallo',
    TO_DATE('2019-02-01', 'YYYY-MM-DD'));

INSERT INTO AUTO (TARGA, MODELLO, COLORE, SCADENZA_ASSICURAZIONE)
    VALUES ('QP925ZI', 'Fiat Punto', 'rosso',
    TO_DATE('2021-03-05', 'YYYY-MM-DD'));

INSERT INTO AUTO (TARGA, MODELLO, COLORE, SCADENZA_ASSICURAZIONE)
    VALUES ('PA763KC', 'Peugeot 208', 'blu',
    TO_DATE('2025-06-20', 'YYYY-MM-DD'));
```

```
INSERT INTO AUTISTI (UTENTE, NUMERO_PATENTE, SCADENZA_PATENTE, AUTO)
    VALUES ('000001', 'a34k120d18',
    TO_DATE('2020-02-12', 'YYYY-MM-DD'), 'AP234WE');

INSERT INTO AUTISTI (UTENTE, NUMERO_PATENTE, SCADENZA_PATENTE, AUTO)
    VALUES ('000002', 'd95kfnaero',
    TO_DATE('2021-06-13', 'YYYY-MM-DD'), 'PO572FG');

INSERT INTO AUTISTI (UTENTE, NUMERO_PATENTE, SCADENZA_PATENTE, AUTO)
    VALUES ('000003', '873kdnfria',
    TO_DATE('2026-06-30', 'YYYY-MM-DD'), 'QP925ZI');

INSERT INTO AUTISTI (UTENTE, NUMERO_PATENTE, SCADENZA_PATENTE, AUTO)
    VALUES ('000004', 'aso4610nfa',
    TO_DATE('2019-11-01', 'YYYY-MM-DD'), 'PA763KC');


INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Caserta', codCitta_Gen.nextval, 'CE');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Napoli', codCitta_Gen.nextval, 'NA');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Latina', codCitta_Gen.nextval, 'LT');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Milano', codCitta_Gen.nextval, 'MI');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Genova', codCitta_Gen.nextval, 'GE');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Trieste', codCitta_Gen.nextval, 'TS');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Roma', codCitta_Gen.nextval, 'RM');

INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Torino', codCitta_Gen.nextval, 'TO');
```

```
INSERT INTO CITTA (NOME, CODICE, PROVINCIA)
    VALUES ('Biella', codCitta_Gen.nextval, 'BI');


INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('41,891', '12,511', 1);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('23,002', '45,998', 2);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('14,000', '72,896', 3);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('21,088', '23,002', 4);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('78,896', '45,632', 5);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('16,564', '49,458', 6);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('13,045', '16,789', 7);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('46,789', '56,054', 8);

INSERT INTO LUOGHI (LATITUDINE, LONGITUDINE, CITTA)
VALUES ('20,004', '20,008', 9);


INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '13,50', '+00 03:30:00.000000',
    '4', '4', '000001', '41,891', '12,511',
    TO_DATE('2018-05-05 13:30:00', 'YYYY-MM-DD HH24:MI:SS'),
    '23,002', '45,998',
    TO_DATE('2018-05-05 17:00:00', 'YYYY-MM-DD HH24:MI:SS'));
```

```
INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '15,00', '+00 02:00:00.000000',
    '4', '4', '000002', '14,000', '72,896',
    TO_DATE('2018-01-01 13:00:00', 'YYYY-MM-DD HH24:MI:SS'),
    '21,088', '23,002',
    TO_DATE('2018-01-01 15:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '17,00', '+00 04:00:00.000000',
    '4', '4', '000003', '78,896', '45,632',
    TO_DATE('2018-02-02 14:00:00', 'YYYY-MM-DD HH24:MI:SS'),
    '16,564', '49,458',
    TO_DATE('2018-02-02 18:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '18,00', '+00 01:00:00.000000',
    '4', '4', '000004', '13,045', '16,789',
    TO_DATE('2018-03-03 13:00:00', 'YYYY-MM-DD HH24:MI:SS'),
    '46,789', '56,054',
    TO_DATE('2018-03-03 14:00:00', 'YYYY-MM-DD HH24:MI:SS'));

INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '19,00', '+00 05:00:00.000000',
    '4', '4', '000001', '20,004', '20,008',
    TO_DATE('2018-04-04 13:30:00', 'YYYY-MM-DD HH24:MI:SS'),
    '41,891', '12,511',
    TO_DATE('2018-04-04 18:30:00', 'YYYY-MM-DD HH24:MI:SS'));
```

```
INSERT INTO VIAGGI (CODICE, PREZZO, DURATA, POSTI, POSTI_DISPONIBILI,
    AUTISTA, LATITUDINE_PARTENZA, LONGITUDINE_PARTENZA,
    DATA_PARTENZA, LATITUDINE_ARRIVO,
    LONGITUDINE_ARRIVO, DATA_ARRIVO)
    VALUES (codViaggio_Gen.nextval, '20,00', '+00 06:00:00.000000',
    '4', '4', '000002', '23,002', '45,998',
    TO_DATE('2018-06-06 13:30:00', 'YYYY-MM-DD HH24:MI:SS'),
    '14,000', '72,896',
    TO_DATE('2018-06-06 19:30:00', 'YYYY-MM-DD HH24:MI:SS'));


INSERT INTO TAPPE (VIAGGIO, LATITUDINE, LONGITUDINE)
    VALUES ('000001', '14,000', '72,896');

INSERT INTO TAPPE (VIAGGIO, LATITUDINE, LONGITUDINE)
    VALUES ('000002', '78,896', '45,632');

INSERT INTO TAPPE (VIAGGIO, LATITUDINE, LONGITUDINE)
    VALUES ('000003', '13,045', '16,789');

INSERT INTO TAPPE (VIAGGIO, LATITUDINE, LONGITUDINE)
    VALUES ('000004', '20,004', '20,008');


INSERT INTO PRENOTAZIONI (CODICE, PASSEGGERO, VIAGGIO)
    VALUES (codPrenotazioni_Gen.nextval, '000005', '000001');

INSERT INTO PRENOTAZIONI (CODICE, PASSEGGERO, VIAGGIO)
    VALUES (codPrenotazioni_Gen.nextval, '000006', '000002');

INSERT INTO PRENOTAZIONI (CODICE, PASSEGGERO, VIAGGIO)
    VALUES (codPrenotazioni_Gen.nextval, '000007', '000003');

INSERT INTO PRENOTAZIONI (CODICE, PASSEGGERO, VIAGGIO)
    VALUES (codPrenotazioni_Gen.nextval, '000005', '000004');


INSERT INTO RECENSIONI (CODICE, VOTO, GIUDIZIO,
    RECENSENTE, RECENSITO, VIAGGIO)
    VALUES (codRecensioni_Gen.nextval, '4',
    'Brava persona, puntuale e abbastanza piacevole.',
    '000001', '000005', '000001');
```

```
INSERT INTO RECENSIONI (CODICE, VOTO, GIUDIZIO,
    RECENSENTE, RECENSITO, VIAGGIO)
    VALUES (codRecensioni_Gen.nextval, '3',
    'Autista un po spericolato, ma affidabile.',
    '000005', '000001', '000001');

INSERT INTO RECENSIONI (CODICE, VOTO, GIUDIZIO,
    RECENSENTE, RECENSITO, VIAGGIO)
    VALUES (codRecensioni_Gen.nextval, '5', 'Tutto ok.',
    '000006', '000002', '000002');

INSERT INTO RECENSIONI (CODICE, VOTO, GIUDIZIO, RECENSENTE,
    RECENSITO, VIAGGIO) VALUES (codRecensioni_Gen.nextval, '5',
    'Viaggio tutto ok. Guidatore molto affidabile, lo consiglio.',
    '000007', '000003', '000003');

INSERT INTO RECENSIONI (CODICE, VOTO, GIUDIZIO, RECENSENTE,
    RECENSITO, VIAGGIO) VALUES (codRecensioni_Gen.nextval, '2',
    'Abbiamo fatto un incidente, non consiglio l''autista.',
    '000005', '000004', '000004');
```

Chapter 3

Interrogazioni sulla base di dati

Una volta completata la progettazione della base di dati, è possibile operare su di essa in termini di interrogazioni (**query**).
Di seguito si riportano alcuni esempi di interrogazioni, tenendo presente di aver creato apposite viste per la loro memorizzazione:

Query 1: selezionare i codici dei viaggi e le relative città di partenza e città di destinazione.

```
CREATE OR REPLACE VIEW LISTA_VIAGGI_ARRIVI_PARTENZE AS
    SELECT V.CODICE AS CODICE_VIAGGIO, C1.NOME AS CITTA_DI_PARTENZA,
    C2.NOME AS CITTA_DI_ARRIVO FROM VIAGGI V,
    LUOGHI L1, CITTA C1, LUOGHI L2, CITTA C2
    WHERE V.LATITUDINE_PARTENZA = L1.LATITUDINE
    AND V.LONGITUDINE_PARTENZA = L1.LONGITUDINE
    AND L1.CITTA = C1.CODICE
    AND V.LATITUDINE_ARRIVO = L2.LATITUDINE
    AND V.LONGITUDINE_ARRIVO = L2.LONGITUDINE
    AND L2.CITTA = C2.CODICE
    ORDER BY CODICE_VIAGGIO;
```

Query 2: selezionare per ciascun viaggio nome e cognome dell'autista.

```
CREATE OR REPLACE VIEW LISTA_VIAGGI_AUTISTI AS
    SELECT V.CODICE AS CODICE_VIAGGIO, U.NOME AS NOME_AUTISTA,
    U.COGNOME AS COGNOME_AUTISTA
    FROM VIAGGI V, UTENTI U
    WHERE V.AUTISTA = U.ID
    ORDER BY CODICE_VIAGGIO;
```

Query 3: selezionare le recensioni su ciascun autista.

```
CREATE OR REPLACE VIEW LISTA_RECENSIONI_AUTISTI AS
    SELECT U.USERNAME AS AUTISTA, R.VOTO AS VOTO,
    R.GIUDIZIO AS GIUDIZIO
    FROM RECENSIONI R, AUTISTI A, UTENTI U
    WHERE  A.UTENTE = U.ID AND R.RECENSITO = U.ID
    ORDER BY AUTISTA, VOTO DESC;
```

Query 4: selezionare le recensioni effettuate su ciascun passeggero.

```
CREATE OR REPLACE VIEW LISTA_RECENSIONI_PASSEGGERI AS
    SELECT U.USERNAME AS PASSEGGERO, R.VOTO AS VOTO,
    R.GIUDIZIO AS GIUDIZIO
    FROM RECENSIONI R, PASSEGGERI P, UTENTI U
    WHERE  P.UTENTE = U.ID AND R.RECENSITO = U.ID
    ORDER BY PASSEGGERO, VOTO DESC;
```

Query 5: selezionare la media dei voti ricevuti da ciascun autista.

```
CREATE OR REPLACE VIEW MEDIA_VOTI_AUTISTI AS
    SELECT U.USERNAME AS AUTISTA, AVG(R.VOTO) AS MEDIA_VOTO
    FROM UTENTI U, AUTISTI A, RECENSIONI R
    WHERE A.UTENTE = U.ID AND R.RECENSITO = U.ID
    GROUP BY U.ID, U.USERNAME
    ORDER BY AUTISTA;
```

Query 6: selezionare gli autisti aventi media di voti superiore o uguale a 4.

```
CREATE OR REPLACE VIEW AUTISTI_MIGLIORI AS
    SELECT M.AUTISTA AS AUTISTA, M.MEDIA_VOTO AS MEDIA_VOTO
    FROM MEDIA_VOTI_AUTISTI M
    WHERE MEDIA_VOTO >= 4
    ORDER BY M.AUTISTA;
```

Query 7: selezionare per ogni autista il numero di viaggi effettuati.

```
CREATE OR REPLACE VIEW NUM_VIAGGI_AUTISTI AS
    SELECT U.USERNAME AS AUTISTA, COUNT (U.ID) AS NUMERO_VIAGGI
    FROM UTENTI U, AUTISTI A, VIAGGI V
    WHERE V.AUTISTA = U.ID AND A.UTENTE = U.ID
    GROUP BY U.ID, U.USERNAME
    ORDER BY U.ID;
```

Query 8: selezionare per ogni passeggero il numero di viaggi effettuati.

```
CREATE OR REPLACE VIEW NUM_VIAGGI_PASSEGGERI AS
    SELECT U.USERNAME AS PASSEGGERO, COUNT(U.ID) AS NUMERO_VIAGGI
    FROM UTENTI U, PRENOTAZIONI P
    WHERE P.PASSEGGERO = U.ID AND P.STATO = 'Accettata'
    GROUP BY U.ID, U.USERNAME
    ORDER BY U.ID;
```

Query 9: selezionare per ogni viaggio i passeggeri partecipanti.

```
CREATE OR REPLACE VIEW PASSEGGERI_VIAGGI AS
    SELECT V.CODICE AS CODICE_VIAGGIO, U1.USERNAME AS USER_AUTISTA,
    U2.USERNAME AS USER_PASSEGGERO
    FROM PRENOTAZIONI PR, VIAGGI V, UTENTI U1, UTENTI U2
    WHERE V.CODICE=PR.VIAGGIO AND PR.PASSEGGERO=U2.ID
    AND V.AUTISTA=U1.ID AND PR.STATO = 'Accettata'
    GROUP BY V.CODICE, V.AUTISTA, U1.USERNAME, U2.USERNAME
    ORDER BY CODICE_VIAGGIO;
```

Query 10: selezionare la città da cui è partito il maggior numero di viaggi.

```
CREATE OR REPLACE VIEW CITTA_MAX_PARTENZE AS
    SELECT C1.NOME AS CITTA_PARTENZA
    FROM VIAGGI V1, LUOGHI L1,CITTA C1
    WHERE V1.LATITUDINE_PARTENZA = L1.LATITUDINE
    AND V1.LONGITUDINE_PARTENZA = L1.LONGITUDINE
    AND L1.CITTA = C1.CODICE
    GROUP BY C1.NOME
    HAVING COUNT(*)>= ALL(
        SELECT COUNT(*)
        FROM VIAGGI V, LUOGHI L, CITTA C
        WHERE V.LATITUDINE_PARTENZA = L.LATITUDINE
        AND V.LONGITUDINE_PARTENZA = L.LONGITUDINE
        AND L.CITTA = C.CODICE
        GROUP BY C.NOME);
```

Query 11: selezionare la città di arrivo più frequente di tutti i viaggi.

```
CREATE OR REPLACE VIEW CITTA_MAX_ARRIVI AS
    SELECT C1.NOME AS CITTA_DESTINAZIONE
    FROM VIAGGI V1, LUOGHI L1,CITTA C1
    WHERE V1.LATITUDINE_ARRIVO = L1.LATITUDINE
    AND V1.LONGITUDINE_ARRIVO = L1.LONGITUDINE
    AND L1.CITTA = C1.CODICE
    GROUP BY C1.NOME
    HAVING COUNT(*)>= ALL(
        SELECT COUNT(*)
        FROM VIAGGI V, LUOGHI L, CITTA C
        WHERE V.LATITUDINE_ARRIVO = L.LATITUDINE
        AND V.LONGITUDINE_ARRIVO = L.LONGITUDINE
        AND L.CITTA=C.CODICE
        GROUP BY C.NOME);
```

Chapter 4

Progettazione dell'applicazione

L'architettura software dell'applicazione si sviluppa su tre livelli logico-funzionali:

- **presetation layer**, interfaccie grafiche o testuali: permettono di inserire dati di input e mostrare dati di output.
- **application layer**, insieme di procedure: realizzano la trasformazione dei dati di input in dati di output.
- **data layer**, costituito dal DBMS stesso.

4.1 Data layer

Costituito dal DBMS ORACLE 11g, la base di dati progettata ed una serie di applicazioni memorizzate nel sistema di basi di dati, secondo l'approccio delle **stored procedure**, in modo da rendere l'informazione indipendenti dalle applicazioni di livello superiore.

4.1.1 Trigger

Sono stati realizzati alcuni trigger per la gestione di particolari operazioni effettuate sulla base dati. Di seguito è riportato il codice PL/SQL relativo alla loro implementazione:

Trigger 1: gestire l'aggiornamento dell'identificativo di un utente.

```
CREATE OR REPLACE TRIGGER UPDATE_UTENTI
    AFTER UPDATE ON UTENTI
    FOR EACH ROW
BEGIN
    UPDATE AUTISTI SET
    UTENTE = :NEW.ID WHERE UTENTE = :OLD.ID;
    UPDATE PASSEGGERI SET
    UTENTE = :NEW.ID WHERE UTENTE = :OLD.ID;
    UPDATE VIAGGI SET
    AUTISTA = :NEW.ID WHERE AUTISTA = :OLD.ID;
    UPDATE PRENOTAZIONI SET
    PASSEGGERO = :NEW.ID WHERE PASSEGGERO = :OLD.ID;
    UPDATE RECENSIONI SET
    RECENSENTE = :NEW.ID WHERE RECENSENTE = :OLD.ID;
    UPDATE RECENSIONI SET
    RECENSITO = :NEW.ID WHERE RECENSITO = :OLD.ID;
END;
```

Trigger 2: gestire l'aggiornamento del codice di un viaggio.

```
CREATE OR REPLACE TRIGGER UPDATE_VIAGGI
    AFTER UPDATE ON VIAGGI
    FOR EACH ROW
BEGIN
    UPDATE RECENSIONI SET VIAGGIO = :NEW.CODICE
    WHERE VIAGGIO = :OLD.CODICE;
    UPDATE TAPPE SET VIAGGIO = :NEW.CODICE
    WHERE VIAGGIO = :OLD.CODICE;
END;
```

Trigger 3: gestire l'aggiornamento dell'auto di un autista.

```
CREATE OR REPLACE TRIGGER UPDATE_AUTO
    AFTER UPDATE ON AUTO
    FOR EACH ROW
BEGIN
    UPDATE AUTISTI SET AUTO = :NEW.TARGA
    WHERE AUTO = :OLD.TARGA;
END;
```

Trigger 4: gestire l'aggiornamento dei posti disponibili di un viaggio, inseguito all'accettazione di una prenotazione.

```
CREATE OR REPLACE TRIGGER UPDATE_POSTI_DISPONIBILI
    AFTER UPDATE ON PRENOTAZIONI
    FOR EACH ROW
    BEGIN
    UPDATE VIAGGI SET POSTI_DISPONIBILI = POSTI_DISPONIBILI - 1
    WHERE CODICE = :OLD.VIAGGIO;
END;
```

Trigger 5: gestire l'aggiornamento dello stato di un viaggio, inseguito alla terminazione dei posti disponibili.

```
CREATE OR REPLACE TRIGGER UPDATE_STATO_VIAGGIO
    BEFORE UPDATE ON VIAGGI
    FOR EACH ROW
BEGIN
    DECLARE
        ERRORE EXCEPTION;
        NUM INTEGER;
    BEGIN
        SELECT V.POSTI_DISPONIBILI INTO NUM
        FROM VIAGGI V WHERE V.CODICE = :OLD.CODICE;
            IF (NUM = 0)
        THEN RAISE ERRORE;
        END IF;
    EXCEPTION
        WHEN ERRORE
        THEN RAISE_APPLICATION_ERROR
        ( -2002 , 'Posti disponibili esauriti');
    END;
END;
```

4.1.2 Procedure

Sono stati realizzate alcune procedure per effettuare particolari operazioni sulla base dati. Di seguito è riportato il codice PL/SQL relativo alla loro implementazione:

Procedura 1: inserimento di un auto.

```
CREATE OR REPLACE PROCEDURE INSERT_AUTO (
    inTARGA IN AUTO.TARGA%TYPE,
    inMODELLO IN AUTO.MODELLO%TYPE,
    inCOLORE IN AUTO.COLORE%TYPE,
    inSCADENZA IN AUTO.SCADENZA_ASSICURAZIONE%TYPE
) AS
BEGIN
    BEGIN
        INSERT INTO AUTO(TARGA, MODELLO, COLORE,
        SCADENZA_ASSICURAZIONE)
        VALUES (inTARGA, inMODELLO, inCOLORE, inSCADENZA);
        COMMIT;
    END;
END INSERT_AUTO;
```

Procedura 2: inserimento di una recensione.

```
CREATE OR REPLACE PROCEDURE INSERT_RECENSIONE (
    inCODICE IN RECENSIONI.CODICE%TYPE,
    inVOTO IN RECENSIONI.VOTO%TYPE,
    inGIUDIZIO IN RECENSIONI.GIUDIZIO%TYPE,
    inRECENSENTE IN RECENSIONI.RECENSENTE%TYPE,
    inRECENSITO IN RECENSIONI.RECENSITO%TYPE,
    inVIAGGIO IN RECENSIONI.VIAGGIO%TYPE
) AS
BEGIN
    BEGIN
        INSERT INTO RECENSIONI(CODICE, VOTO, GIUDIZIO,
        RECENSENTE, RECENSITO, VIAGGIO)
        VALUES (inCODICE, inVOTO, inGIUDIZIO,
        inRECENSENTE, inRECENSITO, inVIAGGIO);
        COMMIT;
    END;
END INSERT_RECENSIONE;
```

References

[1] A.Chianese, V.Moscato, A.Picariello, L.Sansone, *Sistemi di basi di dati e applicazioni*, Santarcangelo Romagna, Maggioli Editore, 2015