Corso di Basi di Dati - Prof. Vincenzo Moscato

Università degli Studi di Napoli Federico II

Facoltà di Ingegneria Informatica

Anno Accademico 2022 - 2023

Progetto *AroVaGO*

Carmine Bellotti **N46 005710**

Vincenzo Luigi Bruno **N46 005698**

Cristina Carleo **N46 005492**

Simone Cecere **N46 005669**

INDICE

Specifiche

Specifiche sui dati

Il committente richiede di tenere traccia delle seguenti informazioni:

- per i pacchetti offerti: numero di persone, soggiorni, spostamenti;
- per gli spostamenti: luogo di partenza, il luogo di arrivo, le coordinate gps di entrambi i luoghi, le date di partenza e arrivo (con i relativi orari) ed il mezzo di trasporto;
- per i mezzi di trasporto:tempo di percorrenza (espresso in ore), il numero di viaggiatori e di mezzi necessari al trasporto, ed infine il relativo prezzo;
- per gli alberghi: nome, la tipologia, le date di check-in e check-out ed il costo giornaliero;
- per le camere: tipologia di camera, il numero di occupanti ed i servizi offerti;
- per le prenotazioni: l'utente che ha effettuato la prenotazione, spostamenti e soggiorni selezionati, il costo complessivo, la data di partenza e di ritorno, e nominativo di ciascuno degli altri utenti viaggiatori;
- per i viaggiatori: nome e cognome;
- per il metodo di pagamento: se pago con bonifico bancario o carta di credito e nel primo caso la data di effettuazione, mentre nel secondo caso circuito gestire, codice, Intestatario e data di scadenza.

Specifiche sulle operazioni

In seguito alla registrazione degli utenti e all'inserimento delle informazioni relative a spostamenti e soggiorni da parte dell'azienda committente. Le principali operazioni previste sulla base di dati sono:

- che la piattaforma permetta di inserire pacchetti `offerti`, con relativi soggiorni e spostamenti scelti;
- che sia possibile visualizzare per ogni viaggio i dettagli della prenotazione effettuata ed eventualmente modificarla;
- che, nel caso in cui si sia scelto il bonifico come metodo di pagamento, una prenotazione non comporti alcun impegno dei soggiorni e degli spostamenti scelti fino alla verifica della data di effettuazione del bonifico.
- che sia possibile rigettare il metodo di pagamento con bonifico nel caso in cui questo non sia stato effettuato prima delle data di partenza.

Specifiche sugli utenti della base dati e politiche di sicurezza

Data la tipologia di servizio richiesto, esistono più categorie di utenti che dovranno interagire con la base dati:

- il DBA, che si occuperà di inserire e aggiornare le informazioni relative agli spostamenti e ai soggiorni, di inserire i pacchetti preimpostati e di registrare gli utenti;
- i *clienti*, che avranno privilegi strettamente necessari allo svolgimento delle loro operazioni sulla base di dati;
- il *servizio_clienti*, che avrà il compito di assistere il cliente in caso di necessità;
- la *webapp,* che avrà privilegi minori rispetto a quelli del dba.

Progettazione Concettuale

Per avere una visione più schematica dei requisiti statici richiesti, in prima istanza il team ha scelto di effettuare una settorializzazione dei requisiti suddividendoli in categorie e quindi elaborando il glossario dei termini usati

Glossario dei termini:

Termine	Descrizione	Termini Collegati
Albergo	Sistemazione dei viaggiatori unica per ogni soggiorno	Soggiorno, Camera
Camera	Sistemazione relativa ad un unico albergo e di diversa categoria e capienza, con annessi servizi aggiuntivi.	Soggiorno, Albergo
Metodo di pagamento	Insieme delle modalità attraverso le quali un utente può saldare il conto del viaggio programmato.	Utente
Mezzo di trasporto	Tipologia di mezzo prevista per accompagnare i viaggiatori da un luogo all'altro.	Spostamento

Termine	Descrizione	Termini Collegati
Pacchetto	Composizione di spostamenti e soggiorni preimpostati dal committente	Prenotazione,Spostamento, Soggiorno
Prenotazione	Fase di registrazione di un viaggio, una prenotazione può essere rifiutata qualora il saldo del pagamento non venga effettuato prima della data di partenza e contiene la registrazione esplicita del nominativo di ciascuno degli altri viaggiatori.	Pacchetto, Utente, Viaggiatore
Spostamento	Ogni spostamento che viene effettuato da un unico mezzo di trasporto. Si tiene traccia del luogo di partenza e di quello di arrivo(es. spostamento con navetta Budapest Aeroporto - Budapest Hotel, spostamento con taxi Budapest Hotel-Praga Hotel)	Mezzo di trasporto, Pacchetto
Soggiorno	Sistemazione dei viaggiatori a cui corrisponde un unico hotel e una o più camere.	Pacchetto, Albergo, Camera
Utente	Individuo che ha effettuato almeno una prenotazione tramite piattaforma, un utente può effettuare più prenotazioni e per ognuna di queste deve effettuare la registrazione esplicita degli altri viaggiatori che non saranno utenti della piattaforma(non avranno un loro account). Ogni utente può selezionare un solo metodo di pagamento alla volta ed è tenuto a saldare il conto affinchè la prenotazione vada a buon fine.	Metodo di pagamento, prenotazione.
Viaggiatore	Individuo le cui informazioni anagrafiche sono state aggiunte dall'utente che ha effettuato la prenotazione.	Prenotazione, Utente

Progettazione Concettuale

Schema E/R portante

Estraiamo dalle specifiche lo schema portante con i concetti fondamentali. Seguirà poi una fase di raffinamento, estendendo lo schema portante con tutte le specifiche non ancora esaminate.

Nel modello ER sono state identificate tre entità chiave:utente, prenotazione e pacchetto. Lo schema portante sarà quindi il seguente.

Un utente potrà infatti attraverso una prenotazione, scegliere un pacchetto offerto fra i vari disponibili a seconda delle sue esigenze. Una volta individuato lo schema portante possiamo estendere la nostra analisi per arrivare allo schema ER completo.

Schema E/R completo

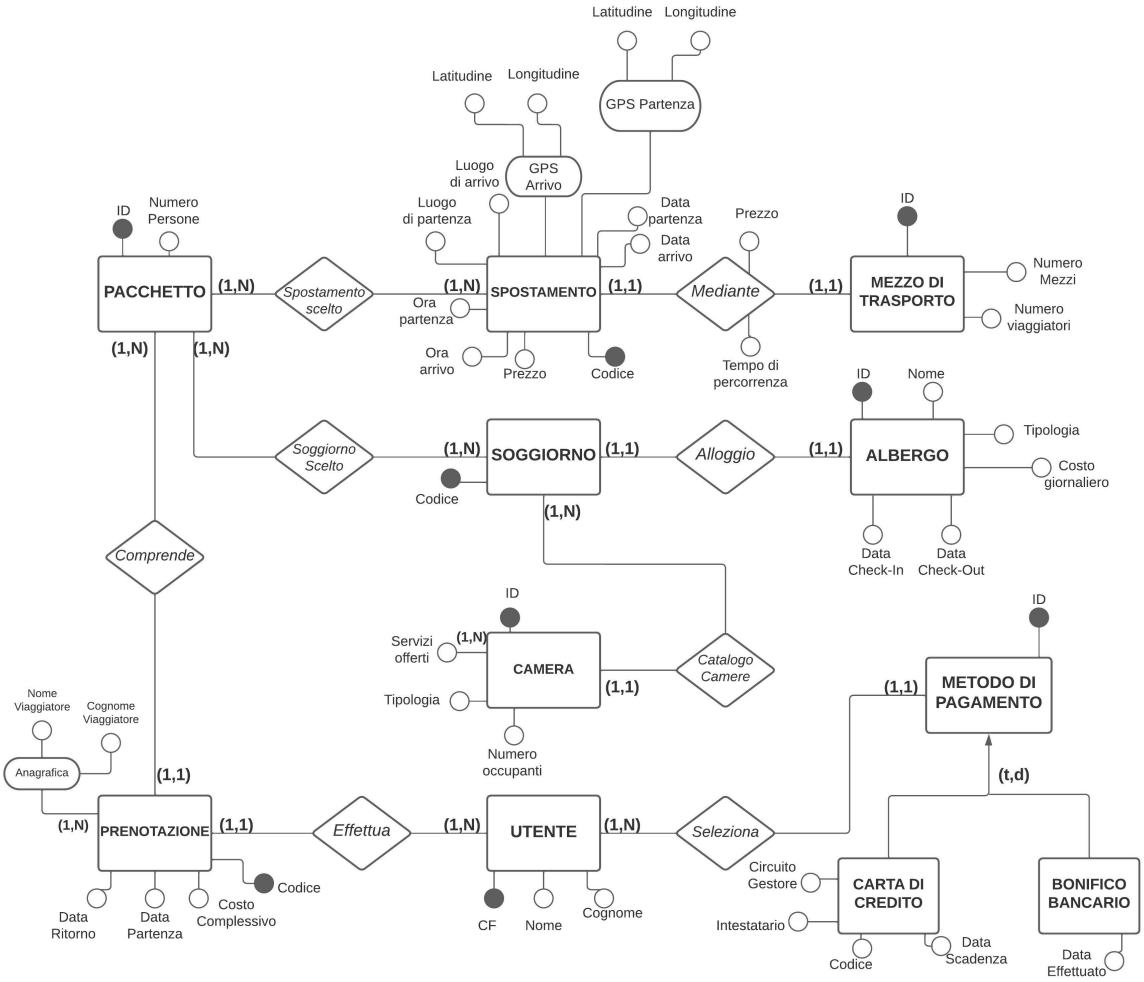
Un utente può scegliere un metodo di pagamento diverso per ogni prenotazione, in particolare sceglierà se optare per un pagamento con carta di credito oppure effettuare un bonifico, si tratta bonifico e carta di credito saranno specializzazione dell'entità generale "metodo di pagamento". Questa gerarchia risulta essere totale e disgiunta in quanto non è possibile saldare il conto con un'ulteriore metodo di pagamento, nè farlo pagando in parte con carta di credito e in parte effettuando un bonifico, quindi un'opzione esclude l'altra.

Ogni prenotazione sarà effettuata da un solo utente e relativa ad un solo pacchetto, inoltre dovrà avere la registrazione esplicita del nominativo di ciascuno degli altri utenti viaggiatori, incluso l'utente, se compreso*, quindi l'entità prenotazione avrà un attributo composto multivalore che contenga l'anagrafica degli altri viaggiatori. Ogni pacchetto può risultare in più prenotazioni, inoltre risulta essere una composizione di spostamenti e soggiorni.

Ogni spostamento può essere relativo a più pacchetti, ma viene effettuato da un solo mezzo di trasporto, quindi fra le due entità avremo un'associazione uno a uno. Fra i suoi attributi uno spostamento tiene traccia delle coordinate GPS del luogo di partenza e di arrivo, cioè si compone di latitudine e longitudine; la scelta del nostro team è stata di trattare questo attributo come composto, quindi di trattare latitudine e longitudine con un tipo stringa.

Ogni soggiorno può essere relativo a più pacchetti, ma ad un solo albergo e relativamente a questo albergo possiamo scegliere una o più camere, motivo per il quale l'associazione fra le due entità è un'associazione uno a molti.

*Si ipotizza che un utente possa effettuare una preotazione anche per altre persone che non posseggono un account.



Anche se il CF può variare nel tempo*

Progettazione Logica

A partire dallo schema completo, la fase successiva è quella della progettazione logica, che si compone di due fasi:

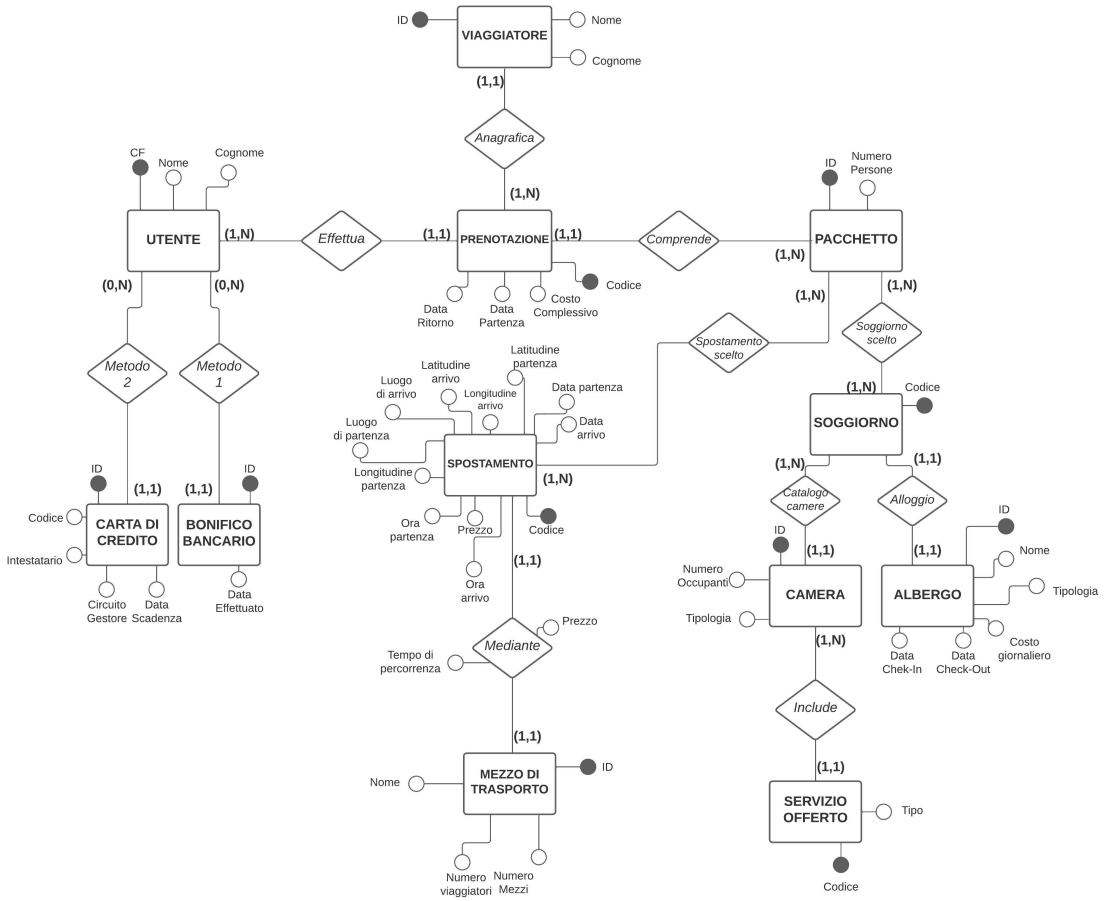
1. *Trasformazione* dello schema concettuale in uno schema semplificato.
2. *Traduzione* dello schema semplificato nello schema relazionale.

Fase di Trasformazione

Gli attributi composti e multivalore non sono immediatamente traducibili nel modello logico. Un attributo multivalore deve essere eliminato introducendo una nuova entità, mentre un attributo composto può essere semplificato sciogliendosi in più attributi semplici, quante sono le sue componenti. Nel modello ER ci sono più attributi di questo tipo da trattare:

- L'attributo composto *GPS arrivo* di Spostamento: si scioglie in nei due attributi semplici Latitudine arrivo, Longitudine arrivo;
- L'attributo composto *GPS partenza* di Spostamento: si scioglie in nei due attributi semplici Latitudine partenza, Longitudine partenza;
- L'attributo multivalore *Servizi_offerti* di Camera: diventa una nuova entità, con un proprio identificativo e un tipo come attributo;
- L'attributo composto multivalore *Anagrafica* di Viaggiatori: diventa una nuova entità *Viaggiatori*, con un proprio identificativo e i propri attributi.

Il nostro schema presenta la generalizzazione *Metodo di pagamento* con le sue due specializzazioni *Carta di credito* e *Bonifico Bancario.* Questa non può essere rappresentata nel modello logico relazionale, in quanto non esiste un costrutto analogo. Essendo la generalizzazione totale e disgiunta, la scelta consigliata è la soluzione 1: accorpamento della superclasse nelle sottoclassi. Abbiamo quindi eliminato l'entità padre, facendo sì che le figlie ereditassero il suo identificatore e la relazione a cui l'entità padre partecipa, cioè quella con l'entità utenti; inoltre abbiamo reso la partecipazione alle entità figlie opzionali, in quanto essendo la generalizzazione disgiunta avrò sempre occorrenze in una delle entità e non nell'altra.



Fase di Traduzione

Per la fase di traduzione, applichiamo le seguenti regole:

1. Una entità dello schema concettuale si traduce in una relazione dello schema logico avente lo stesso nome (ma al plurale) e gli stessi attributi dell'entità ed avente per chiave primaria il suo identificatore. Quindi trasformiamo tutte le entità.
2. **Relazioni uno a uno**

   Per la relazione *Mediante* e la relazione *Alloggio* il team ha deciso di tradurre aggiungendo alla relazione che traduce una delle due entità (*Mezzo di trasporto, Soggiorni*) gli attributi dell'associazione e l'identificatore dell'altra entità. Esiste un vincolo di unicità sul nuovo attributo aggiunto ed un vincolo di integrità referenziale fra quest'ultimo e il corrispondente attributo dell'altra entità.

   - **MEZZI_DI_TRASPORTO(**ID, NumeroViaggiatori, NumeroMezzi, TempoPercorrenza, Prezzo, ID_Spostamento*:SPOSTAMENTI)
   - **SOGGIORNI(**Codice, ID_Albergo*:ALBERGHI)
3. **Relazioni uno a molti**

   Per le relazioni *Anagrafica, Effettua, Comprende, Metodo 1, Metodo 2, Include* si è deciso di aggiungere alla relazione che traduce l'entità dal lato uno gli attributi dell'associazione e l'identificatore dell'entità lato molti, ridenominato. Esiste un

vincolo di integrità referenziale tra questo attributo e il corrispondente attributo dell'entità dal lato molti

- **VIAGGIATORI**(ID, Nome, Cognome, Prenotazione:PRENOTAZIONI)
- **PRENOTAZIONI(**Codice, CostoComplessivo, DataPartenza, DataRitorno, Pacchetto:PACCHETTI, Utente:UTENTI)
- **CARTE_DI_CREDITO(**ID, CircuitoGestore, Intestatario, Codice*, DataScadenza, Utente:UTENTI)
- **BONIFICI_BANCARI(**ID, DataEffettuata, Utente:UTENTI)
- **SERVIZI_OFFERTI**(Codice, Tipo, Camera:CAMERE)

4. **Relazioni molti a molti**

   Per le relazioni *Soggiorni scelti* e *Spostamenti scelti* si è scelto di tradurre l'associazione in una relazione dello schema logico avente lo stesso nome(ma al plurale) dell'associazione e per attributi gli identificatori delle entità coinvolte. L'identificatore di una delle due entità costituisce la chiave primaria della relazione, mentre l'altro deve soddisfare il vincolo di unicità, ma in questo caso è opport, inoltre entrambi hanno un vincolo di integrità referenziale con il corrispondente attributo dell'entità da cui viene.

   - **SPOSTAMENTI_SCELTI(**ID_Pacchetto: PACCHETTI, ID_Spostamento:SPOSTAMENTI)

Il risultato è il seguente modello logico-relazionale:

PACCHETTI(ID**,** NumeroPersone)

SPOSTAMENTI(ID, LuogoPartenza, LuogoArrivo, LatitudineArrivo, LongitudineArrivo, LatitudinePartenza, LongitudinePartenza, DataPartenza, DataArrivo, OraArrivo, OraPartenza)

SPOSTAMENTI_SCELTI(ID_Pacchetto: PACCHETTI, ID_Spostamento:SPOSTAMENTI)

SOGGIORNI_SCELTI(ID_Pacchetto: PACCHETTI, Codice_Soggiorno:SOGGIORNI)

MEZZI_DI_TRASPORTO(ID, NumeroViaggiatori, NumeroMezzi, TempoPercorrenza, Prezzo, ID_Spostamento*:SPOSTAMENTI)

SOGGIORNI(Codice, ID_Albergo*:ALBERGHI)

ALBERGHI(ID, Nome, Tipologia, DataCheckIn, DataCheckOut, CostoGiornaliero)

CAMERE(ID, Tipologia, NumeroOccupanti, Soggiorno:SOGGIORNI)

SERVIZI_OFFERTI(Codice, Tipo, Camera:CAMERE)

PRENOTAZIONI(Codice, CostoComplessivo, DataPartenza, DataRitorno, Pacchetto:PACCHETTI, Utente:UTENTI)

VIAGGIATORI(ID, Nome, Cognome, Prenotazione:PRENOTAZIONI)

UTENTI(CF, Nome, Cognome)

CARTE_DI_CREDITO(ID, CircuitoGestore, Intestatario, Codice*, DataScadenza, Utente:UTENTI)

BONIFICI_BANCARI(ID, DataEffettuata, Utente:UTENTI)

In questo caso abbiamo una delle 2 associazioni opzionali ci conviene mettere tutto nella tabella non opzionale per evitare la gestione dei valori nulli.

Progettazione Fisica

Dopo aver creato il modello logico-relazionale, segue l'implementazione fisica della base di dati.

Dimensionamento

Si tenga presente che nell'arco del prossimo anno la base di dati dovrà gestire la seguente mole di informazioni:

- circa **1000 pacchetti**
- circa **1000 utenti**

PACCHETTI

Attributo	Tipo	Byte	Occorrenza: 1000
ID	NUMBER(7)	5	5Kb

Attributo	Tipo	Byte	Occorrenza: 1000
NumeroPersone	NUMBER(5)	4	4Kb
STORAGE			**TOTALE**
Spazio occupato			**9Kb**

SPOSTAMENTI

Attributo	Tipo	Byte	Occorrenze: 10000
ID	NUMBER(7)	5	50Kb
LuogoPartenza	VARCHAR2(50)	50	500Kb
LuogoArrivo	VARCHAR2(50)	50	500Kb
LatitudineArrivo	CHAR(20)	20	200Kb
LongitudineArrivo	CHAR(20)	20	200Kb
LatitudinePartenza	CHAR(20)	20	200Kb
LongitudinePartenza	CHAR(20)	20	200Kb
DataPartenza	DATE	7	70Kb
DataArrivo	DATE	7	70Kb
OraArrivo	CHAR(5)	5	50Kb
OraPartenza	CHAR(5)	5	50Kb
STORAGE			**TOTALE**
Spazio occupato			**2.09Mb**

ALBERGHI

Attributo	Tipo	Byte	Occorrenze: 50000
ID	NUMBER(7)	5	250Kb
Nome	VARCHAR2(50)	50	2.5Mb
Tipologia	VARCHAR2(50)	50	2.5Mb
DataCheckIn	DATE	7	350Kb
DataCheckOut	DATE	7	350Kb
CostoGiornaliero	NUMBER(7)	5	250Kb
STORAGE			**TOTALE**
Spazio occupato			**6.2Mb**

SOGGIORNI

Attributo	Tipo	Byte	Occorrenze: 10000
Codice	CHAR(5)	5	50Kb
ID_Albergo	NUMBER(7)	5	50Kb
STORAGE			**TOTALE**
Spazio occupato			**100Kb**

SPOSTAMENTI_SCELTI

Attributo	Tipo	Byte	Occorrenze:10000
ID_Pacchetto	NUMBER(7)	5	50Kb
ID_Spostamento	NUMBER(7)	5	50Kb
STORAGE			**TOTALE**
Spazio Occupato			**100Kb**

SOGGIORNI_SCELTI

Attributo	Tipo	Byte	Occorrenze:10000
ID_Pacchetto	NUMBER(7)	5	50Kb
Codice_Soggiorno	CHAR(5)	5	50Kb

STORAGE			**TOTALE**
Spazio Occupato			**100Kb**

MEZZI_DI_TRASPORTO

Attributi	**Tipo**	**Byte**	**Occorrenze: 50000**
ID	NUMBER(7)	5	250Kb
NumeroViaggiatori	NUMBER(5)	4	200Kb
NumeroMezzi	NUMBER(2)	3	150Kb
TempoPercorrenza	CHAR(5)	5	250Kb
Prezzo	NUMBER(7)	5	250Kb
ID_Spostamento	NUMBER(7)	5	250Kb
STORAGE			**TOTALE**
Spazio Occupato			**1.35Mb**

CAMERE

Attributi	**Tipo**	**Byte**	**Occorrenze: 50000**
ID	NUMBER(7)	5	250Kb
Tipologia	VARCHAR2(50)	50	2.5Mb
NumeroOccupanti	NUMBER(5)	4	200Kb
Soggiorno	CHAR(5)	5	250Kb
STORAGE			**TOTALE**
Spazio Occupato			**3.2Mb**

SERVIZI_OFFERTI

Attributi	**Tipo**	**Byte**	**Occorrenze: 500**
Codice	CHAR(3)	3	1.5Kb
Tipo	VARCHAR2(50)	50	15Kb
Camera	NUMBER(7)	5	2.5Kb
STORAGE			**TOTALE**
Spazio Occupato			**19Kb**

UTENTI

Attributi	**Tipo**	**Byte**	**Occorrenze:1000**
CF	CHAR(16)	16	16Kb
Nome	VARCHAR2(50)	50	50Kb
Cognome	VARCHAR2(50)	50	50Kb
STORAGE			**TOTALE**
Spazio Occupato			**116Kb**

PRENOTAZIONI

Attributi	**Tipo**	**Byte**	**Occorrenze:10000**
Codice	NUMBER(5)	4	40Kb
CostoComplessivo	NUMBER(7)	5	50Kb
DataPartenza	DATE	7	70Kb
DataRitorno	DATE	7	70Kb
Pacchetto	NUMBER(7)	5	50Kb
Utente	CHAR(16)	16	160Kb
STORAGE			**TOTALE**
Spazio Occupato			**440Kb**

ARCHIVIO PRENOTAZIONI			
Attributi	**Tipo**	**Byte**	**Occorrenze:10000**
Codice	NUMBER(5)	4	40Kb
CostoComplessivo	NUMBER(7)	5	50Kb
DataPartenza	DATE	7	70Kb
DataRitorno	DATE	7	70Kb
Pacchetto	NUMBER(7)	5	50Kb
Utente	CHAR(16)	16	160Kb
STORAGE			**TOTALE**
Spazio Occupato			**440Kb**

VIAGGIATORI			
Attributi	**Tipo**	**Byte**	**Occorrenze:10000**
ID	NUMBER(7)	5	50Kb
Nome	VARCHAR2(50)	50	500Kb
Cognome	VARCHAR2(50)	50	500Kb
Prenotazione	NUMBER(5)	4	40Kb
STORAGE			**TOTALE**
Spazio Occupato			**1.09Mb**

CARTE_DI_CREDITO			
Attributi	**Tipo**	**Byte**	**Occorrenze:1000**
ID	NUMBER(7)	5	5Kb
CircuitoGestore	VARCHAR2(30)	30	30Kb
Intestatario	VARCHAR2(100)	100	100Kb
Codice	CHAR(16)	16	16Kb
DataScadenza	DATE	7	7Kb
Utente	CHAR(16)	16	16Kb
STORAGE			**TOTALE**
Spazio Occupato			**174Kb**

BONIFICI_BANCARI			
Attributi	**Tipo**	**Byte**	**Occorrenze:1000**
ID	NUMBER(7)	5	5Kb
DataEffettuata	DATE	7	7Kb
Utente	CHAR(16)	16	16Kb
STORAGE			**TOTALE**
Spazio Occupato			**28Kb**

TOTALE: 15.3Mb

Lo storage totale richiesto per la base di dati è dato dalla somma degli storage associati alle tabelle più un aliquota che tiene conto di eventuali errori nel dimensionamento, dei byte riservati agli header dei blocchi e dello spazio necessario ad eventuali indici e viste non previste inizialmente

STORAGE TOTALE = 30 Mb

Oracle Database Express Edition (XE) 21c presenta come limitazioni delle risorse:

- al massimo 2 CPU sfruttabili;
- fino a 2GB di memoria RAM (SGA e PGA combinate) utilizzabile;
- fino a 12GB di disco (storage) utilizzabile;

Creazione Tablespace

Nel nostro database ORACLE creiamo una struttura logica che descrive un'area logica di memorizzazione (*tablespace*) dove memorizziamo i dati delle nostre tabelle.

```
CREATE TABLESPACE arvg_ts DATAFILE 'C:\app\vince\product\21c\oradata\XE\arvg.dbf'
SIZE 30M;
```

Non avendo tipi CLOB, non necessitiamo di un tablespace dedicato alla loro memorizzazione.

Creazione dell'utente DBA e degli altri ruoli

Creiamo il **DBA** della base di dati:

```
CREATE USER arvg_dba DEFAULT TABLESPACE argv_ts
IDENTIFIED BY Kvara
GRANT DBA, UNLIMITED TABLESPACE TO arvg_dba;
```

Di seguito è riportato lo script della creazione dell'operatore della base di dati denominato "**supporto_clienti**":

```
CREATE ROLE supporto_clienti;
GRANT CONNECT TO supporto_clienti;
GRANT SELECT ON arvg_dba.UTENTI TO supporto_clienti;
GRANT SELECT ON arvg_dba.VIAGGIATORI TO supporto_clienti;
GRANT SELECT ON arvg_dba.SOGGIORNI TO supporto_clienti;
GRANT SELECT ON arvg_dba.SPOSTAMENTI TO supporto_clienti;
GRANT SELECT ON arvg_dba.MEZZI_DI_TRASPORTO TO supporto_clienti;
GRANT SELECT ON arvg_dba.CAMERE TO supporto_clienti;
GRANT SELECT ON arvg_dba.ALBERGHI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PRENOTAZIONI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PACCHETTI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.SOGGIORNI_SCELTI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.SPOSTAMENTI_SCELTI TO supporto_clienti;
```

Di seguito è riportato lo script della creazione dell'operatore della base di dati denominato "**cliente**":

```
CREATE ROLE cliente;
GRANT CONNECT TO cliente;
GRANT SELECT TO arvg_dba.PACCHETTI TO cliente;
GRANT SELECT TO arvg_dba.SPOSTAMENTI TO cliente;
GRANT SELECT TO arvg_dba.SOGGIORNI TO cliente;
GRANT SELECT TO arvg_dba.MEZZI_DI_TRASPORTO TO cliente;
GRANT SELECT TO arvg_dba.ALBERGHI TO cliente;
GRANT SELECT TO arvg_dba.CAMERE TO cliente;
GRANT SELECT TO arvg_dba.SERVIZI_OFFERTI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.BONIFICI_BANCARI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.CARTE_DI_CREDITO TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PRENOTAZIONI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.VIAGGIATORI TO cliente;
```

Creazione delle Tabelle

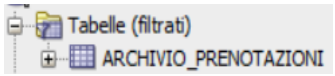
Di seguito è riportato lo script della creazione delle tabelle con le rispettive specifiche dei parametri di storage.

```
CREATE TABLE PACCHETTI(
  ID NUMBER(7),
  NumeroPersone NUMBER(5) NOT NULL,

  CONSTRAINT PK_PACCHETTI PRIMARY KEY(ID)
)
STORAGE (INITIAL 9 K);

CREATE TABLE SPOSTAMENTI(
  ID NUMBER(7),
  LuogoPartenza VARCHAR2(50) NOT NULL,
  LuogoArrivo VARCHAR2(50) NOT NULL,
  LatitudineArrivo CHAR(20) NOT NULL,
  LongitudineArrivo CHAR(20) NOT NULL,
  LatitudinePartenza CHAR(20) NOT NULL,
  LongitudinePartenza CHAR(20) NOT NULL,
  DataPartenza DATE NOT NULL,
```

```
  DataArrivo DATE NOT NULL,
  OraArrivo CHAR(5) NOT NULL,
  OraPartenza CHAR(5) NOT NULL,

  CONSTRAINT PK_SPOSTAMENTI PRIMARY KEY(ID),
  CONSTRAINT CC_ORA_PARTENZA CHECK (OraPartenza LIKE '__:__'),
  CONSTRAINT CC_ORA_ARRIVO CHECK (OraArrivo LIKE '__:__'),
  CONSTRAINT CC_DATA_SPOSTAMENTI CHECK (NOT(DataArrivo < DataPartenza))
)
STORAGE (INITIAL 3 M);

CREATE TABLE ALBERGHI(
  ID NUMBER(7),
  Nome VARCHAR2(50) NOT NULL,
  Tipologia VARCHAR2(50) NOT NULL,
  DataCheckIn DATE NOT NULL,
  DataCheckOut DATE NOT NULL,
  CostoGiornaliero NUMBER(7) NOT NULL,

  CONSTRAINT PK_ALBERGHI PRIMARY KEY(ID)
)
STORAGE (INITIAL 7 M);

CREATE TABLE SOGGIORNI(
  Codice CHAR(5),
  ID_Albergo NUMBER(7) NOT NULL,

  CONSTRAINT PK_SOGGIORNI PRIMARY KEY(Codice),
  CONSTRAINT UC_SOGGIORNI UNIQUE(ID_Albergo)
)
STORAGE (INITIAL 100 K);

CREATE TABLE SPOSTAMENTI_SCELTI(
  ID_Pacchetto NUMBER(7),
  ID_Spostamento NUMBER(7),

  CONSTRAINT PK_SPOSTAMENTI_SCELTI PRIMARY KEY(ID_Pacchetto, ID_Spostamento)
)
STORAGE (INITIAL 100 K);

CREATE TABLE SOGGIORNI_SCELTI(
  ID_Pacchetto NUMBER(7),
  Codice_Soggiorno CHAR(5),

  CONSTRAINT PK_SOGGIORNI_SCELTI PRIMARY KEY(ID_Pacchetto, Codice_Soggiorno)
)
STORAGE (INITIAL 100 K);

CREATE TABLE MEZZI_DI_TRASPORTO(
  ID NUMBER(7),
  NumeroViaggiatori NUMBER(5) NOT NULL,
  NumeroMezzi NUMBER(2) DEFAULT 1,
  TempoPercorrenza CHAR(5) NOT NULL,
  Prezzo NUMBER(7) NOT NULL,
  ID_Spostamento NUMBER(7) NOT NULL,

  CONSTRAINT PK_MEZZI_DI_TRASPORTO PRIMARY KEY(ID),
  CONSTRAINT UC_MEZZI_DI_TARSPORTO UNIQUE(ID_Spostamento),
  CONSTRAINT CC_TEMPO_PERCORRENZA CHECK (TempoPercorrenza LIKE '__:__')
)
STORAGE (INITIAL 2 M);

CREATE TABLE CAMERE(
  ID NUMBER(7),
  Tipologia VARCHAR2(50) NOT NULL,
  NumeroOccupanti NUMBER(5) NOT NULL,
  Soggiorno CHAR(5) NOT NULL,

  CONSTRAINT PK_CAMERE PRIMARY KEY(ID)
)
STORAGE (INITIAL 4 M);

CREATE TABLE SERVIZI_OFFERTI(
  Codice CHAR(3),
  Tipo VARCHAR2(50) NOT NULL,
  Camera NUMBER(7) NOT NULL,

  CONSTRAINT PK_SERVIZI_OFFERTI PRIMARY KEY(Codice)
)
STORAGE (INITIAL 19 K);

CREATE TABLE UTENTI(
  CF CHAR(16),
  Nome VARCHAR2(50) NOT NULL,
  Cognome VARCHAR2(50) NOT NULL,

  CONSTRAINT PK_UTENTI PRIMARY KEY(CF)
```

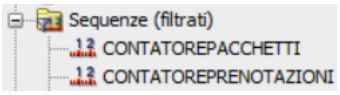
```
)
STORAGE (INITIAL 116 K);

CREATE TABLE PRENOTAZIONI(
  Codice NUMBER(5),
  CostoComplessivo NUMBER(7) NOT NULL,
  DataPartenza DATE NOT NULL,
  DataRitorno DATE NOT NULL,
  Pacchetto NUMBER(7) NOT NULL,
  Utente CHAR(16) NOT NULL,

  CONSTRAINT PK_PRENOTAZIONI PRIMARY KEY(Codice),
  CONSTRAINT CC_DATA_PRENOTAZIONI CHECK (NOT(DataRitorno < DataPartenza))
)
STORAGE (INITIAL 440 K);

CREATE TABLE ARCHIVIO_PRENOTAZIONI(
  Codice NUMBER(5),
  CostoComplessivo NUMBER(7) NOT NULL,
  DataPartenza DATE NOT NULL,
  DataRitorno DATE NOT NULL,
  Pacchetto NUMBER(7) NOT NULL,
  Utente CHAR(16) NOT NULL,

  CONSTRAINT PK_ARCHIVIO_PRENOTAZIONI PRIMARY KEY(Codice),
  CONSTRAINT CC_DATA_ARCHIVIO_PRENOTAZIONI CHECK (NOT(DataRitorno < DataPartenza))
)
STORAGE (INITIAL 440 K);

CREATE TABLE VIAGGIATORI(
  ID NUMBER(7),
  Nome VARCHAR2(50) NOT NULL,
  Cognome VARCHAR2(50) NOT NULL,
  Prenotazione NUMBER(5) NOT NULL,

  CONSTRAINT PK_VIAGGIATORI PRIMARY KEY(ID)
)
STORAGE (INITIAL 2 M);

CREATE TABLE CARTE_DI_CREDITO(
  ID NUMBER(7),
  CircuitoGestore VARCHAR2(30) NOT NULL,
  Intestatario VARCHAR2(100) NOT NULL,
  Codice CHAR(16) NOT NULL,
  DataScadenza DATE NOT NULL,
  Utente CHAR(16) NOT NULL,

  CONSTRAINT PK_CARTE_DI_CREDITO PRIMARY KEY(ID)

)
STORAGE (INITIAL 174 K);

CREATE TABLE BONIFICI_BANCARI(
  ID NUMBER(7),
  DataEffettuata DATE NOT NULL,
  Utente CHAR(16) NOT NULL,

  CONSTRAINT PK_BONIFICI_BANCARI PRIMARY KEY(ID)

)
STORAGE (INITIAL 28 K);
```

Per la stored procedure abbiamo creato:



Creazione Sequenze

Per l'inserimento di alcuni valori di chiave primaria il team ha scelto di utilizzare le "SEQUENZE" messe a disposizione dai DBMS, in modo tale da assegnare in automatico i valori di alcuni campi come codici o id progressivi.



Come valore massimo abbiamo deciso di inserire il valore delle occorrenze nelle varie tabelle che intendiamo gestire nel prossimo anno.

```
CREATE SEQUENCE contatorePacchetti START WITH 1 INCREMENT BY 1
MAXVALUE 1000 NOCYCLE;
CREATE SEQUENCE contatorePrenotazioni START WITH 1 INCREMENT BY 1
MAXVALUE 10000 NOCYCLE;
```

Specifichiamo il `NOCYCLE` poiché la sequenza è creata per una chiave primaria dunque non possiamo avere valori uguali e dunque ricominciare a generare valori una volta arrivati al `MAXVALUE`. Oltretutto non intendiamo gestire più occorrenze di quelle indicate nel `MAXVALUE`.

Aggiunta delle chiavi esterne e politiche di reazione

Le scelte per le politiche di reazione sono state le seguenti:

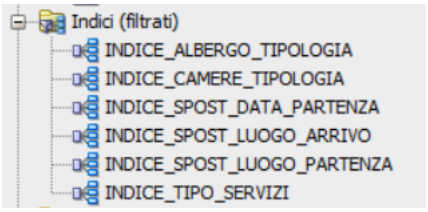
- Quando viene eliminato un pacchetto vengono eliminate anche le relative prenotazioni e spostamenti/soggiorni scelti.
- Quando viene eliminato uno spostamento vengono eliminati anche i relativi mezzi di trasporto e spostamenti scelti.
- Quando viene eliminato un soggiorno vengono eliminati anche le relative camere e soggiorni scelti.
- Quando viene eliminato un albergo vengono eliminati anche i soggiorni associati.
- Quando viene eliminata una camera vengono eliminati anche i relativi servizi offerti, poiché relativi ad una ed un sola camera.
- Quando viene eliminato un utente vengono eliminate anche le sue carte di credito e bonifici bancari e le relative prenotazioni.
- Quando viene eliminata una prenotazione vengono eliminati anche i viaggiatori ad esso associati

```
ALTER TABLE SPOSTAMENTI_SCELTI ADD CONSTRAINT FK_PACCHETTI_SPO_SCELTI
    FOREIGN KEY(ID_Pacchetto) REFERENCES PACCHETTI(ID)
ON DELETE CASCADE;

ALTER TABLE SPOSTAMENTI_SCELTI ADD CONSTRAINT FK_SPOSTAMENTI_SCELTI
    FOREIGN KEY(ID_Spostamento) REFERENCES SPOSTAMENTI(ID)
ON DELETE CASCADE;

ALTER TABLE SOGGIORNI_SCELTI ADD CONSTRAINT FK_PACCHETTI_SOG_SCELTI
    FOREIGN KEY(ID_Pacchetto) REFERENCES PACCHETTI(ID)
```

```
ON DELETE CASCADE;

ALTER TABLE SOGGIORNI_SCELTI ADD CONSTRAINT FK_SOGGIORNI_SCELTI
    FOREIGN KEY(Codice_Soggiorno) REFERENCES SOGGIORNI(Codice)
ON DELETE CASCADE;

ALTER TABLE MEZZI_DI_TRASPORTO ADD CONSTRAINT FK_MEZZI_SPOSTAMENTI
    FOREIGN KEY(ID_Spostamento) REFERENCES SPOSTAMENTI(ID)
ON DELETE CASCADE;

ALTER TABLE SOGGIORNI ADD CONSTRAINT FK_SOGGIORNI_ALBERGHI
    FOREIGN KEY(ID_Albergo) REFERENCES ALBERGHI(ID)
ON DELETE CASCADE;

ALTER TABLE CAMERE ADD CONSTRAINT FK_CAMERE_SOGGIORNI
    FOREIGN KEY(Soggiorno) REFERENCES SOGGIORNI(Codice)
ON DELETE CASCADE;

ALTER TABLE SERVIZI_OFFERTI ADD CONSTRAINT FK_SERVIZI_CAMERE
    FOREIGN KEY(Camera) REFERENCES CAMERE(ID)
ON DELETE CASCADE;

ALTER TABLE PRENOTAZIONI ADD CONSTRAINT FK_PRENOTAZIONI_PACCHETTI
    FOREIGN KEY(Pacchetto) REFERENCES PACCHETTI(ID)
ON DELETE CASCADE;

ALTER TABLE PRENOTAZIONI ADD CONSTRAINT FK_PRENOTAZIONI_UTENTI
    FOREIGN KEY(Utente) REFERENCES UTENTI(CF)
ON DELETE CASCADE;

ALTER TABLE VIAGGIATORI ADD CONSTRAINT FK_VIAGGIATORI_PRENOTAZIONI
    FOREIGN KEY(Prenotazione) REFERENCES PRENOTAZIONI(Codice)
ON DELETE CASCADE;

ALTER TABLE CARTE_DI_CREDITO ADD CONSTRAINT FK_CARTE_UTENTI
    FOREIGN KEY(Utente) REFERENCES UTENTI(CF)
ON DELETE CASCADE;

ALTER TABLE BONIFICI_BANCARI ADD CONSTRAINT FK_BONIFICI_UTENTI
    FOREIGN KEY(Utente) REFERENCES UTENTI(CF)
ON DELETE CASCADE;
```

Indici

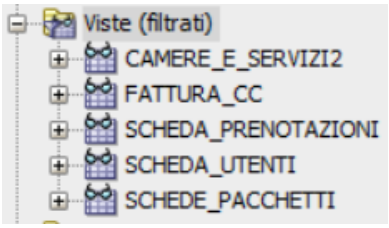
Oracle crea di Default indici definiti sulle chiavi primarie, però si è ritenuto necessario aggiungere degli indici secondari definiti sui campi di seguito elencati:

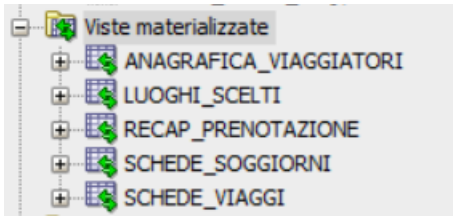


- **SPOSTAMENTO.LuogoPartenza, SPOSTAMENTO.LuogoArrivo e SPOSTAMENTO.DataPartenza** poiché la ricerca di un pacchetto turistico si basa soprattutto sui luoghi di partenza e di arrivo e data di partenza.
- **ALBERGHI.Tipologia** poichè è molto importante al fine della scelta di un determinato pacchetto.
- **CAMERE.Tipologia** come sopra.
- **SERVIZI_OFFERTI.Tipo** poiché condiziona molto la scelta di un determinato pacchetto

```
CREATE INDEX INDICE_SPOST_LUOGO_PARTENZA ON SPOSTAMENTI(LuogoPartenza);
CREATE INDEX INDICE_SPOST_LUOGO_ARRIVO ON SPOSTAMENTI(LuogoArrivo);
CREATE INDEX INDICE_SPOST_DATA_PARTENZA ON SPOSTAMENTI(DataPartenza);
CREATE INDEX INDICE_ALBERGO_TIPOLOGIA ON ALBERGHI(Tipologia);
CREATE INDEX INDICE_CAMERE_TIPOLOGIA ON CAMERE(Tipologia);
CREATE INDEX INDICE_TIPO_SERVIZI ON SERVIZI_OFFERTI(Tipo);
```

Viste





Viste di recap per l’utente della piattaforma

Le seguenti viste vengono utilizzate da un utente per visualizzare tutti i soggiorni e i relaitivi pacchetti associati, tutti i pacchetti con i relativi spostamenti, e tutte le informazioni riguardante un pacchetto, con spastamento e soggiorno incluso in esso.

```
--vista che permette di visualizzare tutti i viaggi associati ad un pacchetto
CREATE MATERIALIZED VIEW SCHEDE_VIAGGI AS
SELECT P.ID AS PACCHETTO_SCELTO,S.LUOGOPARTENZA, S.LUOGOARRIVO,S.DATAPARTENZA,S.DATAARRIVO,S.ORAPARTENZA,S.ORAARRIVO
FROM (SPOSTAMENTI_SCELTI SC JOIN PACCHETTI P ON SC.ID_PACCHETTO=P.ID)JOIN SPOSTAMENTI S ON ID_SPOSTAMENTO=S.ID

--vista che permette di visualizzare il soggiorno associato ad un determinato pacchetto
CREATE MATERIALIZED VIEW SCHEDE_SOGGIORNI AS
SELECT SS.ID_PACCHETTO, S1.CODICE, A.ID,A.NOME,A.TIPOLOGIA,A.COSTOGIORNALIERO
FROM (SOGGIORNI S1 JOIN SOGGIORNI_SCELTI SS ON S1.CODICE=SS.CODICE_SOGGIORNO )JOIN ALBERGHI A ON S1.ID_ALBERGO=A.ID

--vista che permette un recap all'utente sulle informazioni riguardante un pacchetto
CREATE MATERIALIZED VIEW RECAP_PRENOTAZIONE AS
SELECT SS2.PACCHETTO_SCELTO,SS2.LUOGOPARTENZA, SS2.LUOGOARRIVO,SS2.DATAPARTENZA,SS2.DATAARRIVO,SS2.ORAPARTENZA,SS2.ORAARRIVO, SS1.CODI
FROM SCHEDE_SOGGIORNI SS1  RIGHT JOIN SCHEDE_VIAGGI SS2 ON SS1.ID_PACCHETTO=SS2.PACCHETTO_SCELTO

--Mostra codice camera la tipologia e il tipo di servizio associato
CREATE VIEW CAMERE_E_SERVIZI2 AS
SELECT C.ID,S.TIPO FROM SERVIZI_OFFERTI S JOIN CAMERE C ON S.CAMERA=C.ID

--Seleziona le anagrafiche inserite dall'utente con relativa  prenotazione
CREATE MATERIALIZED VIEW ANAGRAFICA_VIAGGIATORI AS
SELECT V.ID, V.NOME, V.COGNOME, V.PRENOTAZIONE
FROM VIAGGIATORI V
```

Viste per il gestore della piattaforma

Il gestore, utilizzando la piattaforma, può tenere traccia di tutte le informazioni che sono state inserite, come ad esempio i dati forniti dagli utenti, le transazioni effettuate, i pagamenti eseguiti, le promozioni attivate e molte altre cose. Inoltre, grazie alla piattaforma, il gestore può monitorare costantemente tutte le informazioni contenute nella piattaforma, garantendo così la sicurezza e l'integrità delle informazioni. In questo modo, è possibile assicurare ai clienti un servizio di qualità e affidabile.

```
--Ottieni le informazioni principali riguardanti tutti gli utenti
CREATE OR REPLACE VIEW Scheda_Utenti AS
SELECT
    U.CF,
    U.Nome,
    U.Cognome,
    C.CircuitoGestore AS Circuito_Carta,
    C.Codice AS Codice_Carta,
    C.DataScadenza AS DataScadenza_Carta,
    B.DataEffettuata AS Data_Bonifico,
    P.Codice AS Codice_Prenotazione,
    P.CostoComplessivo AS CostoPrenotazione,
    P.DataPartenza AS DataPartenza_Prenotazione,
    P.DataRitorno AS DataRitorno_Prenotazione,
    P.Pacchetto AS Pacchetto_Prenotato
FROM
    UTENTI U
    LEFT JOIN CARTE_DI_CREDITO C ON U.CF=C.Utente
    LEFT JOIN BONIFICI_BANCARI B ON U.CF=B.Utente
    JOIN PRENOTAZIONI P ON U.CF=P.Utente;

SELECT * FROM Scheda_Utenti;
```

	CF	NOME	COGNOME	CIRCUITO_CARTA	CODICE_CARTA	DATASCADENZA_CARTA	DATA_BONIFICO	CODICE_PRENOTAZIONE	COSTOPRENOTAZIONE
1	PSTCRI70E25F205N	CIRO	PASTORE	MASTERCARD	486	01-FEB-26	(null)	78945	365
2	RSSMRA65L08F205U	MARIO	ROSSI	(null)	(null)	(null)	05-NOV-22	45612	850
3	BNCMRC97B06F205C	MARCO	BIANCHI	VISA	774	01-NOV-24	(null)	12304	1000
4	PSTCRI70E25F205N	CIRO	PASTORE	MASTERCARD	486	01-FEB-26	(null)	78946	850

DATAPARTENZA_PRENOTAZIONE	DATARITORNO_PRENOTAZIONE	PACCHETTO_PRENOTATO
10-GEN-23	17-GEN-23	5431
15-MAG-23	25-MAG-23	7894
21-MAR-23	28-MAR-23	3214
10-GEN-23	17-GEN-23	7894

```
--Vista parziale: Mostra tutte le prenotazioni effettuate dagli utenti
CREATE OR REPLACE VIEW Scheda_Prenotazioni AS
SELECT
    U.CF,
    U.Nome,
    U.Cognome,
    P.Codice AS Codice_Prenotazione,
    P.CostoComplessivo AS CostoPrenotazione,
    P.DataPartenza AS DataPartenza_Prenotazione,
    P.DataRitorno AS DataRitorno_Prenotazione,
    P.Pacchetto AS Pacchetto_Prenotato
FROM
    UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.Utente;

SELECT * FROM Scheda_Prenotazioni;
```

	CF	NOME	COGNOME	CODICE_PRENOTAZIONE	COSTOPRENOTAZIONE	DATAPARTENZA_PRENOTAZIONE	DATARITORNO_PRENOTAZIONE	PACCHETTO_PRENOTATO
1	PSTCRI70E25F205N	CIRO	PASTORE	78945	365	10-GEN-23	17-GEN-23	5431
2	RSSMRA65L08F205U	MARIO	ROSSI	45612	850	15-MAG-23	25-MAG-23	7894
3	BNCMRC97B06F205C	MARCO	BIANCHI	12304	1000	21-MAR-23	28-MAR-23	3214
4	PSTCRI70E25F205N	CIRO	PASTORE	78946	850	10-GEN-23	17-GEN-23	7894

```
--Scheda che permette di visualizzare viaggi e soggiorni associati ai pacchetti
CREATE VIEW SCHEDE_PACCHETTI AS
SELECT SV.PACCHETTO_SCELTO,SV.LUOGOPARTENZA, SV.LUOGOARRIVO,SV.DATAPARTENZA,SV.DATAARRIVO,SV.ORAPARTENZA,SV.ORAARRIVO,SS.ID_PACCHETTO,
FROM SCHEDE_VIAGGI SV JOIN SCHEDE_SOGGIORNI SS ON SS.ID_PACCHETTO=SV.PACCHETTO_SCELTO
```

PACCHETTO_SCELTO	LUOGOPARTENZA	LUOGOARRIVO	DATAPARTENZA	DATAARRIVO	ORAPARTENZA	ORAARRIVO	ID_PACCHETTO	CODICE	ID	NOME	TIPOLOGIA	COSTOGIORNALIERO
5431	NAPOLI	AMSTERDAM	10-GEN-23	10-GEN-23	14:00	16:45	5431	2436	7021	HOTEL KVARA	ALBERGO	30
3214	PARIGI	MILANO	15-MAG-23	15-MAG-23	17:00	18:31	3214	7536	8520	HOTEL CIRO	ALBERGO	45
7894	LONDRA	MADRID	05-APR-23	05-APR-23	15:00	17:30	7894	4561	6320	VINNYMOSCATO	TRATTORIA	25

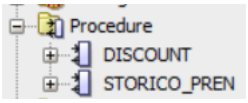
```
--Mostro tutte le fatture effettuate con carta di credito
CREATE VIEW Fattura_cc AS
SELECT U.CF AS Utente, U.NOME, U.COGNOME, CC.INTESTATARIO AS Intestatario, CC.CIRCUITOGESTORE AS Circuito
FROM UTENTI U JOIN CARTE_DI_CREDITO CC ON U.CF=CC.UTENTE

--Mostra le mete associate ad ogni pacchetto
CREATE MATERIALIZED VIEW LUOGHI_SCELTI AS
SELECT P.ID AS PACCHETTO, S.LUOGOARRIVO AS DESTINAZIONE
FROM  (PACCHETTI P JOIN SPOSTAMENTI_SCELTI SP ON P.ID=SP.ID_PACCHETTO) JOIN SPOSTAMENTI S ON SP.ID_SPOSTAMENTO=S.ID
```

Stored Procedure e Trigger

L’architettura software del sistema si basa sul modello *three-tier* componendosi di tre livelli logico-funzionali:

- Livello dati
- Livello applicazione
- Livello presentazione

Stored Procedure:



Dopo aver definito le tabelle implementiamo delle stored procedure con lo scopo di sottolineare rapidità e interattività della base dati.

In seguito le procedure implementate:

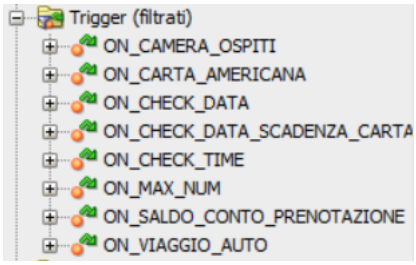
- Applicazione di uno sconto al costo di una prenotazione

```
CREATE OR REPLACE PROCEDURE Discount(
  IDPacchetto IN PRENOTAZIONI.Pacchetto%TYPE,
  Percentuale IN NUMBER
)
IS
BEGIN
UPDATE PRENOTAZIONI
SET CostoComplessivo = CostoComplessivo - (CostoComplessivo * Percentuale)
WHERE Pacchetto=IDPacchetto;
END;

--Che possiamo chiamare ad esempio così (20% di sconto sul primo Pacchetto)
CALL Discount(1, 0.2);
```

- Storico Prenotazioni, si tratta di uno spostamento delle prenotazioni relative a viaggi conclusi, cioè relativi a date antecedenti quella presente, in un archivio dedicato

```
CREATE OR REPLACE PROCEDURE storico_pren AS
BEGIN
    DECLARE
        cod Prenotazioni.Codice%TYPE;
        costo Prenotazioni.Costocomplessivo%TYPE;
        datap Prenotazioni.DataPartenza%TYPE;
        datar Prenotazioni.DataRitorno%TYPE;
        pack Prenotazioni.Pacchetto%TYPE;
        usr Prenotazioni.Utente%TYPE;
        CURSOR cur IS SELECT Codice, CostoComplessivo, DataPartenza, DataRitorno, Pacchetto, Utente
        FROM Prenotazioni WHERE DataRitorno < SYSDATE;
    BEGIN
        OPEN cur;
        LOOP
            FETCH cur INTO cod, costo, datap, datar, pack, usr;
            EXIT WHEN cur%NOTFOUND;
            INSERT INTO Archivio_Prenotazioni VALUES (cod, costo, datap, datar, pack, usr);
            DELETE FROM Prenotazioni WHERE codice = cod;
        END LOOP;
        CLOSE cur;
    END;
END storico_pren;
```

Trigger:



Relativamente ai triggers, vengono definiti i seguenti per la gestione di anomalie piuttosto comuni in una piattaforma che prevede:

- gestione di transazioni, come mancato saldo del pagamento, rifiuto di un metodo di pagamento non accettato dal sistema, come le carte di origine estera e controllo sulla scadenza del metodo di pagamento scelto;
- gestione della corrispondenza tra numero di persone e servizio scelto, come l'aumento dei mezzi da prenotare qualora il numero di viaggiatori superi la capienza del mezzo o la verifica della capienza della camera d'albergo in relazione al numero di persone;
- controllo delle anomalie e/o limiti della base dati: come il superamento da parte degli utenti registrati del numero massimo imposto in fase di progetto; l'inserimento di date o orari non validi;

In seguito riportate le varie implementazioni:

- Quando il giorno prima della partenza gli utenti che hanno scelto il pagamento mediante bonifico non hanno ancora effettuato il saldo

```
-- Realizzare un trigger che, quando si aggiorna il campo "DATAEFFETTUATA" della Tabella BONIFICI_BANCARI, modifica in automatico anch
CREATE OR REPLACE TRIGGER On_Saldo_Conto_prenotazione
BEFORE INSERT ON BONIFICI_BANCARI
FOR EACH ROW
declare datacheck date;
errore_ins exception;
BEGIN
select datapartenza into datacheck from PRENOTAZIONI
where Utente=:new.Utente;
if :new.DATAEFFETTUATA>datacheck
then
raise errore_ins;
end if;
exception
when errore_ins then raise_application_error(-20001,'Il pagamento non è andato a buon fine');
END;

--riga da rifiutare
INSERT INTO BONIFICI_BANCARI VALUES(45612,'27-MAR-2023','BNCMRC97B06F205C');
```

```
Trigger SALDO_CONTO_PRENOTAZIONE compilato

Errore con inizio alla riga : 327 nel comando -
INSERT INTO BONIFICI_BANCARI VALUES(45612,'27-MAR-2023','BNCMRC97B06F205C')
Report error -
ORA-20001: Il pagamento non è andato a buon fine
ORA-06512: a "SYSTEM.SALDO_CONTO_PRENOTAZIONE", line 11
ORA-04088: errore durante esecuzione del trigger 'SYSTEM.SALDO_CONTO_PRENOTAZIONE'
```

- Quando inseriamo una carta di credito, affinché questa sia valida, dobbiamo avere una data di scadenza valida (è infatti impossibile inserire un CHECK CONSTRAINT che compia un paragone sulla SYSDATE)

```
--Controllo sulla data di scadenza per l'inserimento di una carta di credito
CREATE OR REPLACE TRIGGER on_check_data_scadenza_carta BEFORE INSERT ON CARTE_DI_CREDITO
FOR EACH ROW
DECLARE
  card_expired exception;
BEGIN
  IF :new.DataScadenza < SYSDATE
      THEN RAISE card_expired;
  END IF;

  EXCEPTION WHEN card_expired THEN raise_application_error(-20002, 'La data di scadenza
  non può essere minore di quella attuale');
END;
```

```
Report error -
ORA-20002: La data di scadenza
    non può essere minore di quella attuale
ORA-06512: a "SYSTEM.CHECK_DATA_SCADENZA_CARTA", line 8
ORA-04088: errore durante esecuzione del trigger 'SYSTEM.CHECK_DATA_SCADENZA_CARTA'
```

- Quando il tipo è “macchina” (id=745) e ci sono più di 5 viaggiatori possiamo aggiungere +1 al numero di mezzi che sono necessari con maggiorazione del prezzo

```
create or replace trigger on_viaggio_auto
before insert on MEZZI_DI_TRASPORTO
for each row
declare
errore_pass exception;
begin
--verifico che sia stato inserito un mezzo auto
if :new.id=745
then
```

```
--modifica del numero delle automobili
if :new.numeroviaggiatori<5
  then :new.numeromezzi :=1;
else if :new.numeroviaggiatori<10
  then :new.numeromezzi :=2;
else if :new.numeroviaggiatori<15
  then :new.numeromezzi :=3;
else
  raise errore_pass;
end if;
end if;
end if;
end if;
exception
when errore_pass then raise_application_error(-20004,'Posti auto esauriti');
end;

--prova del trigger
INSERT INTO MEZZI_DI_TRASPORTO VALUES(745,5,1,'02:30',200,1563);
```

	ID	NUMEROVIAGGIATORI	NUMEROMEZZI	TEMPOPERCORRENZA	PREZZO	ID_SPOSTAMENTO
1	745	5	2	02:30	200	1563
2	123	3	1	02:45	150	7845
3	845	2	1	01:31	300	3689

- Quando la tipologia di camera è doppia/tripla/quadrupla e il numero di occupanti è maggiore o minore

```
create or replace trigger on_camera_ospiti
before insert on CAMERE
for each row
declare
errore_ospiti exception;
begin
if :new.numerooccupanti<1 then raise errore_ospiti;
end if;
if :new.numerooccupanti>5 then raise errore_ospiti;
end if;
exception
when errore_ospiti then raise_application_error(-20009,'Inserimento non valido');
end;
```

```
Errore con inizio alla riga : 746 nel comando -
INSERT INTO CAMERE VALUES(54,'STANDARD',6,2436)
Report error -
ORA-20009: Inserimento non valido
ORA-06512: a "SYSTEM.ON_CAMERA_OSPITI", line 9
ORA-04088: errore durante esecuzione del trigger 'SYSTEM.ON_CAMERA_OSPITI'
```

- Quando il circuito è American Express il pagamento viene rifiutato poiché non supportato

```
CREATE OR REPLACE TRIGGER on_carta_americana BEFORE INSERT ON CARTE_DI_CREDITO
FOR EACH ROW
DECLARE
card_not_accepted exception;
BEGIN
  IF :new.CIRCUITOGESTORE='AMERICAN EXPRESS'
    THEN RAISE card_not_accepted;
  END IF;
EXCEPTION WHEN card_not_accepted THEN raise_application_error(-20006,'la carta non è accettata nel tuo paese');
END;
```

```
Errore con inizio alla riga : 768 nel comando -
INSERT INTO CARTE_DI_CREDITO VALUES(99645,'AMERICAN EXPRESS','CIRO PASTORE','486',
Report error -
ORA-20006: la carta non è accettata nel tuo paese
ORA-06512: a "SYSTEM.ON_CARTA_AMERICANA", line 7
ORA-04088: errore durante esecuzione del trigger 'SYSTEM.ON_CARTA_AMERICANA'
```

Quando il numero di utenti registrati supera i 1000 (vincolo DB) l’utente non viene registrato e l’insert va in ABORT

```
CREATE OR REPLACE TRIGGER on_max_num
BEFORE INSERT ON UTENTI
FOR EACH ROW
DECLARE BOOL NUMBER(4);
utente_non_inserito exception;
BEGIN
SELECT COUNT(*) INTO BOOL FROM UTENTI;
IF BOOL>1000 then raise utente_non_inserito;
END IF;
EXCEPTION WHEN utente_non_inserito THEN raise_application_error(-20007,'max numero utenti raggiunto');
END;
```

- Quando l’ora di arrivo è minore di quella di partenza

```
create or replace TRIGGER ON_CHECK_TIME
before insert on Spostamenti
for each row
declare
time_error exception;
begin
if :new.datapartenza=SYSDATE
then
    if to_char(:new.orapartenza,'hh24.mi.ss')<to_char(SYSDATE,'hh24.mi.ss')
        then raise time_error;
    end if;
end if;
if :new.oraarrivo<:new.orapartenza
    then raise time_error;
end if;
exception
when time_error then raise_application_error(-20003,'Data non valida');
end;
```

- Quando la data di arrivo è minore quella di partenza

```
create or replace trigger on_check_data
before insert on Spostamenti
for each row
declare
errore_data exception;
begin
if :new.datapartenza<SYSDATE
    then raise errore_data;
end if;
if :new.dataarrivo<:new.datapartenza
    then raise errore_data;
end if;
exception
when errore_data then raise_application_error(-20003,'Data non valida');
end;
```

- Quando il luogo di arrivo è Napoli, l’utente riceve uno sconto del 5% sulla prenotazione

```
create or replace trigger On_Sconto_Napoli
before insert on PRENOTAZIONI
for each row
DECLARE costo number(2);
dest char(10);
BEGIN
SELECT Destinazione into dest from LUOGHI_SCELTI
WHERE Pacchetto=:new.Pacchetto;
if dest = 'Napoli'
then
:new.CostoComplessivo :=:new.CostoComplessivo - (:new.CostoComplessivo * 0.05);
end if;
END;
```

Query

Query per le analisi statistiche:

```
--Seleziona le prenotazioni effettuate con carta di credito
CREATE VIEW Fattura_cc AS
SELECT U.CF AS Utente, U.NOME, U.COGNOME, CC.INTESTATARIO AS Intestatario, CC.CIRCUITOGESTORE AS Circuito
FROM UTENTI U JOIN CARTE_DI_CREDITO CC ON U.CF=CC.UTENTE

SELECT P.CODICE, FCC.NOME, FCC.COGNOME, FCC.Intestatario, FCC.Circuito
FROM PRENOTAZIONI P JOIN Fattura_cc FCC ON FCC.Utente=P.UTENTE

--Seleziona le prenotazioni pagate con bonifico bancario
SELECT  P.CODICE, U.NOME, U.COGNOME
FROM UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.UTENTE
WHERE U.CF NOT IN (
    SELECT CC.UTENTE
    FROM CARTE_DI_CREDITO CC
)
--Seleziona gli utenti che non sono mai stati a Roma in viaggio
CREATE MATERIALIZED VIEW LUOGHI_SCELTI AS
SELECT P.ID AS PACCHETTO, S.LUOGOARRIVO AS DESTINAZIONE
FROM  (PACCHETTI P JOIN SPOSTAMENTI_SCELTI SP ON P.ID=SP.ID_PACCHETTO) JOIN SPOSTAMENTI S ON SP.ID_SPOSTAMENTO=S.ID

SELECT U.NOME, U.COGNOME
FROM UTENTI U
WHERE U.CF NOT IN (
    SELECT P.UTENTE
    FROM PRENOTAZIONI P JOIN LUOGHI_SCELTI LS ON P.PACCHETTO=LS.PACCHETTO
    WHERE LS.DESTINAZIONE LIKE 'ROMA'
)
```

```
INSERT INTO SPOSTAMENTI VALUES(8200,'NAPOLI','PARIGI',48.856614,2.352222,40.851775,14.268124,'20-GEN-2023','20-GEN-2023','15:45','14:0

--Seleziona la città da cui partono più spostamenti
SELECT LuogoPartenza, COUNT(*) AS NUMERO_PARTENZE
FROM SPOSTAMENTI
GROUP BY LuogoPartenza
HAVING COUNT(*)>=ALL(
    SELECT COUNT(*)
    FROM SPOSTAMENTI
    GROUP BY LuogoPartenza);
```

	LUOGOPARTENZA	NUMERO_PARTENZE
1	NAPOLI	2

```
INSERT INTO PACCHETTI VALUES(1200, 3);
INSERT INTO SPOSTAMENTI_SCELTI VALUES(1200, 8200);
INSERT INTO PRENOTAZIONI VALUES(5000,1020,'20-GEN-2023','27-GEN-2023',1200,'BNCMRC97B06F205C');

--individuare per ogni utente la spesa totale
SELECT DISTINCT U.Nome, SUM(P.CostoComplessivo) AS SPESA_TOTALE
FROM UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.Utente
GROUP BY U.Nome
```

	NOME	SPESA_TOTALE
1	CIRO	365
2	MARIO	850
3	MARCO	2020

```
--Mostra l'elenco del numero di prenotazioni effettuate per ogni utente
--in ordine decrescente
SELECT DISTINCT U.Nome, U.Cognome, COUNT(*) AS PRENOTAZIONI_EFFETTUATE
FROM UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.Utente
GROUP BY U.Nome, U.Cognome
ORDER BY PRENOTAZIONI_EFFETTUATE DESC;
```

	NOME	COGNOME	PRENOTAZIONI_EFFETTUATE
1	MARCO	BIANCHI	2
2	CIRO	PASTORE	1
3	MARIO	ROSSI	1

```
--Individua l'utente che ha speso di più
SELECT DISTINCT UT.Nome, UT.Cognome, SUM(PR.CostoComplessivo) AS SPESA_COMPLESSIVA
FROM UTENTI UT JOIN PRENOTAZIONI PR ON UT.CF=PR.Utente
GROUP BY UT.Nome, UT.Cognome
HAVING SUM(PR.CostoComplessivo) >= ALL(
    SELECT SUM(P.CostoComplessivo)
    FROM UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.Utente
    GROUP BY U.CF);
```

	NOME	COGNOME	SPESA_COMPLESSIVA
1	MARCO	BIANCHI	2020

```
--Individua tutte gli utenti che hanno effettuato prenotazioni
--con mese di partenza "GENNAIO"
SELECT DISTINCT U.Nome, U.Cognome, P.Codice, P.DataPartenza
FROM UTENTI U JOIN PRENOTAZIONI P ON U.CF=P.Utente
WHERE P.DataPartenza LIKE '%GEN%';
```

	NOME	COGNOME	CODICE	DATAPARTENZA
1	CIRO	PASTORE	78945	10-GEN-23
2	MARCO	BIANCHI	5000	20-GEN-23

```
--Individuare i pacchetti non ancora prenotati
SELECT PA.ID, PA.NumeroPersone
FROM PACCHETTI PA LEFT JOIN PRENOTAZIONI PR ON PA.ID=PR.Pacchetto
WHERE PR.Pacchetto IS NULL;
```

	ID	NUMEROPERSONE
1	1234	4

```
--Individuare le mete più acquistate--
SELECT S.LuogoArrivo, COUNT(S.ID) AS NUMERO_PRENOTAZIONI
FROM ((SPOSTAMENTI_SCELTI SS JOIN SPOSTAMENTI S ON SS.ID_Spostamento=S.ID)
JOIN PACCHETTI P ON SS.ID_Pacchetto=P.ID) JOIN PRENOTAZIONI PR ON P.ID=PR.Pacchetto
GROUP BY S.LuogoArrivo
HAVING COUNT(*)>=ALL(
    SELECT COUNT(S.ID) AS NUMERO_PRENOTAZIONI
    FROM ((SPOSTAMENTI_SCELTI SS1 JOIN SPOSTAMENTI S1 ON SS1.ID_Spostamento=S1.ID) JOIN
    PACCHETTI P1 ON SS1.ID_Pacchetto=P1.ID) JOIN PRENOTAZIONI PR1 ON P1.ID=PR1.Pacchetto
    GROUP BY S1.LuogoArrivo);

--Versione con vista SCHEDE_VIAGGI--
SELECT SV.LuogoArrivo, COUNT(*) AS NUMERO_PRENOTAZIONI
FROM SCHEDE_VIAGGI SV JOIN PRENOTAZIONI PR ON SV.Pacchetto_Scelto=PR.Pacchetto
GROUP BY SV.LuogoArrivo
HAVING COUNT(*)>=ALL(
    SELECT COUNT(*)
    FROM SCHEDE_VIAGGI SV1 JOIN PRENOTAZIONI PR1 ON SV1.Pacchetto_Scelto=PR1.Pacchetto
    GROUP BY SV1.LuogoArrivo);
```

Query per l’organizzazione del viaggio:

```
--Seleziona il numero di viaggiatori relativi ad ogni prenotazione
SELECT V.PRENOTAZIONE, COUNT(*) AS numero_viaggiatori
FROM VIAGGIATORI V
GROUP BY V.PRENOTAZIONE
```

```
--Mostro tutte le camere che non hanno come servizio 'all inclusive'
--mostro codice camera la tipologia e il tipo di servizio associato
CREATE VIEW CAMERE_E_SERVIZI2 AS
SELECT C.ID,S.TIPO
```

```
FROM SERVIZI_OFFERTI S JOIN CAMERE C ON S.CAMERA=C.ID

SELECT C1.ID,C1.TIPOLOGIA,S1.TIPO
FROM SERVIZI_OFFERTI S1 JOIN CAMERE C1 ON S1.CAMERA=C1.ID
WHERE C1.ID NOT IN
( SELECT C2.ID
  FROM CAMERE_E_SERVIZI2 C2
  WHERE C2.TIPO LIKE 'ALL INCLUSIVE' )
```

```
--Mostra l'anagrafica dei viaggiatori che hanno effettuato
--prenotazione 78945
SELECT NOME, COGNOME
FROM VIAGGIATORI WHERE PRENOTAZIONE=78945;
```

```
--Mostra il tempo di percorrenza dovuto allo spostamento verso Milano
SELECT M.TEMPOPERCORRENZA
FROM MEZZI_DI_TRASPORTO M JOIN SPOSTAMENTI S ON S.ID=M.ID_SPOSTAMENTO
WHERE LUOGOARRIVO LIKE 'MILANO';
```

```
--Mostra la tipologia di albergo,la tipologia di camera e il numero di occupati
SELECT A.TIPOLOGIA,C.TIPOLOGIA,C.NUMEROOCCUPANTI
FROM (SOGGIORNI S JOIN CAMERE C ON C.SOGGIORNO=S.CODICE) JOIN ALBERGHI A ON S.ID_ALBERGO=A.ID;
```


```
--Mostra l'anagrafica dei viaggiatori che hanno effettuato
--prenotazione 78945
SELECT NOME, COGNOME
FROM VIAGGIATORI WHERE PRENOTAZIONE=78945;
```

```
--Mostra il tempo di percorrenza dovuto allo spostamento verso Milano
SELECT M.TEMPOPERCORRENZA
FROM MEZZI_DI_TRASPORTO M JOIN SPOSTAMENTI S ON S.ID=M.ID_SPOSTAMENTO
WHERE LUOGOARRIVO LIKE 'MILANO';
```

```
--Mostra la tipologia di albergo,la tipologia di camera e il numero di occupati
SELECT A.TIPOLOGIA,C.TIPOLOGIA,C.NUMEROOCCUPANTI
FROM (SOGGIORNI S JOIN CAMERE C ON C.SOGGIORNO=S.CODICE) JOIN ALBERGHI A ON S.ID_ALBERGO=A.ID;
```


```
--Mostro tutte le camere che non hanno come servizio 'all inclusive'
--mostro codice camera la tipologia e il tipo di servizio associato
CREATE VIEW CAMERE_E_SERVIZI2 AS
SELECT C.ID,S.TIPO FROM SERVIZI_OFFERTI S JOIN CAMERE C ON S.CAMERA=C.ID
```

```
SELECT C1.ID,C1.TIPOLOGIA,S1.TIPO
FROM SERVIZI_OFFERTI S1 JOIN CAMERE C1 ON S1.CAMERA=C1.ID WHERE C1.ID NOT IN
( SELECT C2.ID
FROM CAMERE_E_SERVIZI2 C2
WHERE C2.TIPO LIKE 'ALL INCLUSIVE' )
```

Politiche di sicurezza

Per quanto riguarda la gestione delle politiche di sicurezza della base di dati, si cerca di garantire la prevenzione dell'acceso ai dati per utenti e profili non autorizzati. Per via di ciò si è definito l'utente **arvg_dba** affinché le operazioni non venissero eseguite col ruolo di *system*. Di seguito lo script dell'utente arvg_dba:

```
CREATE USER arvg_dba DEFAULT TABLESPACE argv_ts
IDENTIFIED BY Kvara
GRANT DBA, UNLIMITED TABLESPACE TO arvg_dba;
```

Una volta connessi come *arvg_dba* si può procedere con la creazione delle tabelle seguedo le specifiche definite in fase di dimensionamento. Inoltre sono stati definiti e rinominati i vari vincoli di chiave primaria e chiave esterna (ove vi fosse) in modo

da permettere un immediata individuazione in caso di violazione.

Viene inoltre definito un ulteriore ruolo a supporto del cliente denominato *“supporto_clienti”*. Viene riportato di seguito lo script della creazione di tale ruolo:

```
CREATE ROLE supporto_clienti;
GRANT CONNECT TO supporto_clienti;
GRANT SELECT ON arvg_dba.UTENTI TO supporto_clienti;
GRANT SELECT ON arvg_dba.VIAGGIATORI TO supporto_clienti;
GRANT SELECT ON arvg_dba.SOGGIORNI TO supporto_clienti;
GRANT SELECT ON arvg_dba.SPOSTAMENTI TO supporto_clienti;
GRANT SELECT ON arvg_dba.MEZZI_DI_TRASPORTO TO supporto_clienti;
GRANT SELECT ON arvg_dba.CAMERE TO supporto_clienti;
GRANT SELECT ON arvg_dba.ALBERGHI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PRENOTAZIONI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PACCHETTI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.SOGGIORNI_SCELTI TO supporto_clienti;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.SPOSTAMENTI_SCELTI TO supporto_clienti;
```

Viene poi definito il ruolo *“cliente”* per permettere l’accesso e l’utilizzo della base dati agli che ne usufruiscono. Viene di seguito riportato lo script della creazione cliente:

```
CREATE ROLE cliente;
GRANT CONNECT TO cliente;
GRANT SELECT TO arvg_dba.PACCHETTI TO cliente;
GRANT SELECT TO arvg_dba.SPOSTAMENTI TO cliente;
GRANT SELECT TO arvg_dba.SOGGIORNI TO cliente;
GRANT SELECT TO arvg_dba.MEZZI_DI_TRASPORTO TO cliente;
GRANT SELECT TO arvg_dba.ALBERGHI TO cliente;
GRANT SELECT TO arvg_dba.CAMERE TO cliente;
GRANT SELECT TO arvg_dba.SERVIZI_OFFERTI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.BONIFICI_BANCARI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.CARTE_DI_CREDITO TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.PRENOTAZIONI TO cliente;
GRANT SELECT, INSERT, UPDATE, DELETE ON arvg_dba.VIAGGIATORI TO cliente;
```

Tuttavia l’applicazione web non può avere accesso agli stessi privilegi del DBA. la creazione del ruolo webapp consente all’applicazione web di interfacciarsi in maniera sicura alla base dati e ,grazie all’ausilio di viste, di operare su di essa senza modificarne il contenuto informativo. Infatti grazie all’utilizzo di viste l’applicazione web non accede direttamente alla base dati ma semplicemente richiama le viste su cui possiede gli opportuni privilegi. Viene poi creato l’utente arovago_webapp con i seguenti privilegi:

```
CREATE ROLE webapp;
GRANT CONNECT TO webapp;

GRANT SELECT ON PACCHETTI TO webapp;
GRANT SELECT ON SPOSTAMENTI TO webapp;
GRANT SELECT ON SPOSTAMENTI_SCELTI TO webapp;
GRANT SELECT ON SOGGIORNI TO webapp;
GRANT SELECT ON MEZZI_DI_TRASPORTO TO webapp;
GRANT SELECT ON CAMERE TO webapp;
GRANT SELECT ON PRENOTAZIONI TO webapp;
GRANT SELECT ON CARTE_DI_CREDITO TO webapp;
GRANT SELECT ON BONIFICI_BANCARI TO webapp;
GRANT SELECT ON VIAGGIATORI TO webapp;
GRANT SELECT ON SOGGIORNI_SCELTI TO webapp;
GRANT SELECT ON SERVIZI_OFFERTI TO webapp;
GRANT SELECT ON SCHEDE_VIAGGI TO webapp; --VISTA
GRANT SELECT ON SCHEDE_SOGGIORNI TO webapp; --VISTA
GRANT SELECT ON RECAP_PRENOTAZIONE TO webapp;--VISTA

GRANT INSERT,UPDATE,DELETE ON CARTE_DI_CREDITO TO webapp;
GRANT INSERT,UPDATE,DELETE ON BONIFICI_BANCARI TO webapp;
GRANT INSERT,UPDATE,DELETE ON VIAGGIATORI TO webapp;
GRANT INSERT,UPDATE,DELETE ON PRENOTAZIONI to webapp;

CREATE USER arovago_webapp DEFAULT TABLESPACE arvg_ts IDENTIFIED BY 4579;
GRANT webapp TO arovago_webapp;
```

Gestione della concorrenza

Per garantire la consistenza dei dati e limitare le modifiche non desiderate, abbiamo deciso di adottare il metodo 2PL Stretto per le transazioni in lettura/scrittura. Il metodo 2PL stretto garantisce che le transazioni non possano interferire tra loro e che

ogni transazione non possa modificare i dati fino a quando non sia completata.
Il blocco di una transazione è un modo per prevenire che due o più transazioni modifichino lo stesso dato. Ciò significa che una transazione deve attendere che le altre transazioni abbiano completato la loro operazione prima di poter accedere ai dati. Inoltre, le transazioni devono soddisfare le proprietà ACID (Atomicità, Consistenza, Isolamento, Durabilità) per garantire la correttezza delle transazioni.

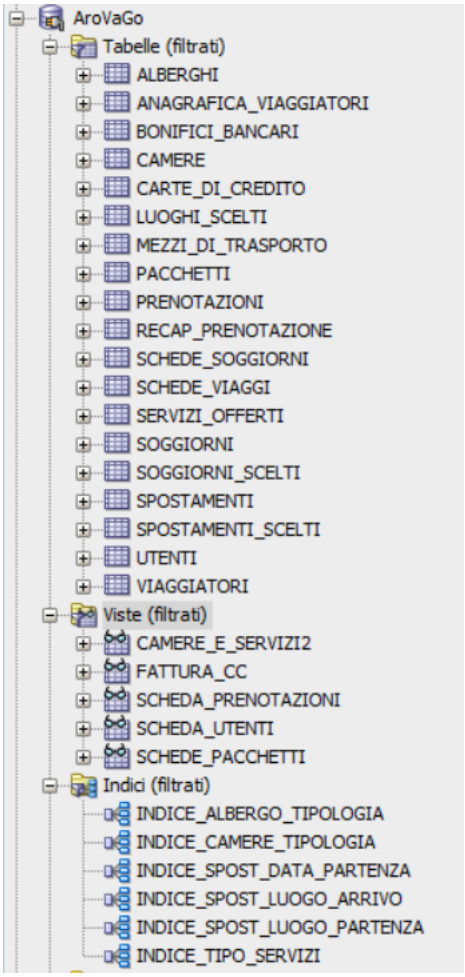
Gestione dell'affidabilità

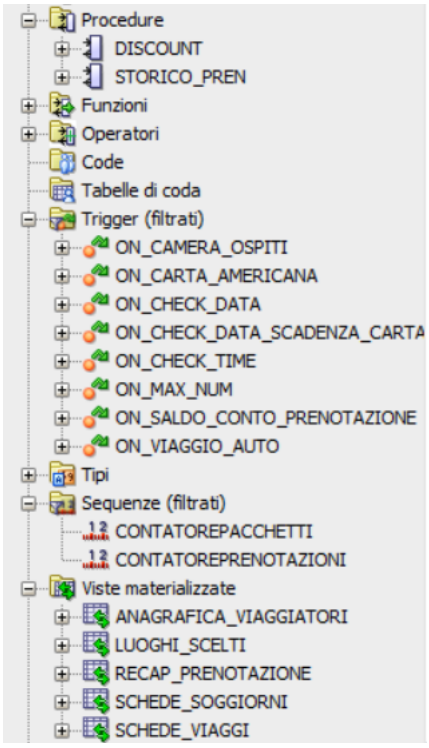
Si è scelto come metodo di storage il **RAID 1**, dal momento che i dischi utilizzati sono sempre e comunque due poiché per ogni disco viene costantemente aggiornata una sua copia speculare. Tale Raid è definito spesso *"mirroring"* in quanto la tecnologia di rindondanza adottata è appunto proprio quella del mirroring. Tale rindondanza consiste nella duplicazione sistematica da un disco ad un altro, ogni dato scritto su un disco verrà scritto nello stesso modo e nella stessa posizione su un altro. Le prestazioni sono, in generale, paragonabili a quelle del disco singolo, con un leggero peggioramento in scrittura, e un miglioramento (statistisco) in lettura. Un eventuale guasto non provoca perdita di dati in quanto quest'ultimi sono perfettamente replicati nel suo gemello. È possibile avviare il sistema con un solo disco funzionante, in attesa della sostituzione del disco guasto.

Avendo scelto Oracle come DBMS, sono implicite alcune scelte riguardanti il controllo dell'affidabilità del sistema:

- La scrittura sulla base di dati è di tipo differito, avendo Oracle dei *redo file* ma non degli *undo file*.
- In caso di leggeri guasti la procedura utilizzata è quella definita come **ripresa a caldo**, che permette il ripristino del corretto funzionamento del database a partire dall'ultimo checkpoint.
- In caso di guasti di rilevata importanza la procedura scelta è quella della **ripresa a freddo**, utilizzando le copie di backup che Oracle ha effettuato sul disco gemello, in modo da ritornare, grazie alla lettura del log, all'ultimo checkpoint per effettuare poi la *ripresa a caldo*.

Schema finale





Progettazione dell’applicazione

Oracle APEX

Relativamente alle scelte di progetto in campo applicativo, il team ha deciso di utilizzare il tool Oracle Application Express per mostrare la gestione del contenuto del database attraverso un'applicazione web data-oriented.

Oracle APEX è una piattaforma di sviluppo di applicazioni low-code enterprise, cioè presenta componenti di gestione e visualizzazione dei dati che svincolano lo sviluppatore dal possedere particolari conoscenze di HTML o CSS.
Le applicazioni create mediante APEX possono essere distribuite nel cloud o in locale.

Nel nostro caso il codice è memorizzato, compilato ed eseguito in Cloud pertanto non si necessita di alcuna risorsa in locale.
La sintassi dell'Apex è JAVA like tanto da considerarlo un derivato di Oracle Java, anche se ci sono molte differenze e vantaggi come il completo uso in Cloud e un modo efficiente di performare il manage dei dati negli oggetti del Database grazie alle API fornite dal linguaggio.

La nostra applicazione consente una visualizzazione di tabelle e viste implementate tramite linguaggio SQL, con la possibilità di navigare tra le varie pagine grazie a link di collegamento proprio come accadrebbe in un classico sito web.

Il primo passo è stato quello di creare un workspace, un'area di lavoro condivisa dove troviamo vari strumenti messi a nostra disposizione e identificata univocamente da un nome.

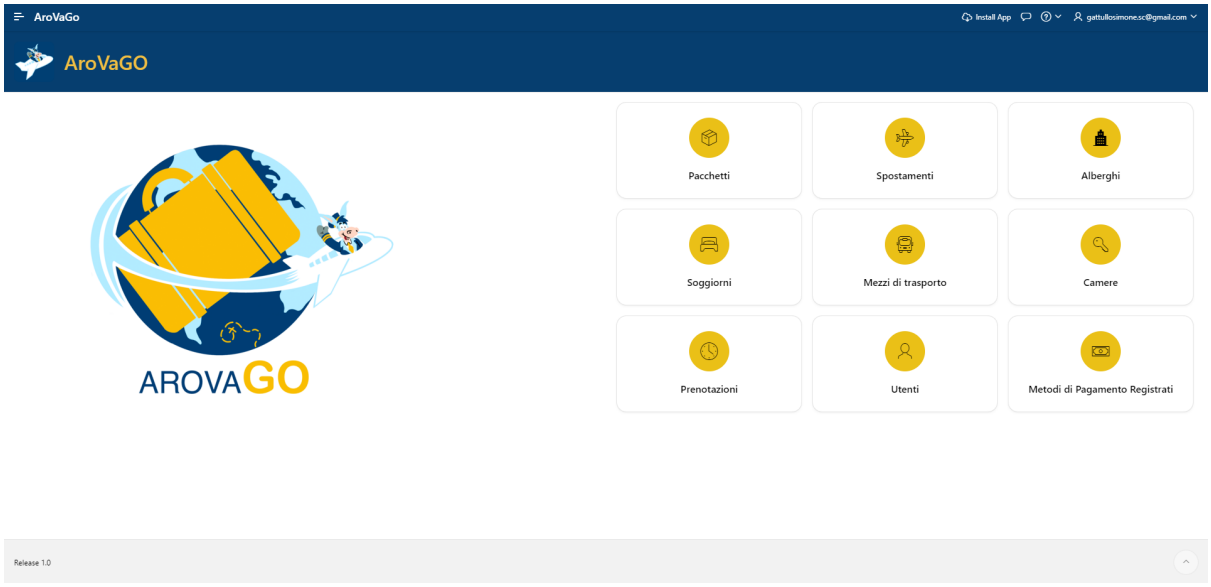
Successivamente si accede al workspace come amministratore del database attraverso le credenziali, è possibile definire i ruoli dei vari utenti e come questi possono interagire con lo sviluppo dell'applicazione.

APEX mette a disposizione vari modi per caricare le informazioni di un database in memoria cloud, noi abbiamo scelto di usare il tradizionale SQL, è infatti presente un'area dedicata alla programmazione in SQL per la creazione, gestione e popolamento del database sul quale il livello applicativo andrà costruito.

L'app presenta varie pagine opportunamente comunicanti tra loro, ciascuna contenente informazioni riguardanti una tabella o vista del database, per permettere una visualizzazione ordinata e agevole.

Le pagine sono state rese "public" in maniera di non necessitare di previa identificazione per potervi accedere.

Home Page



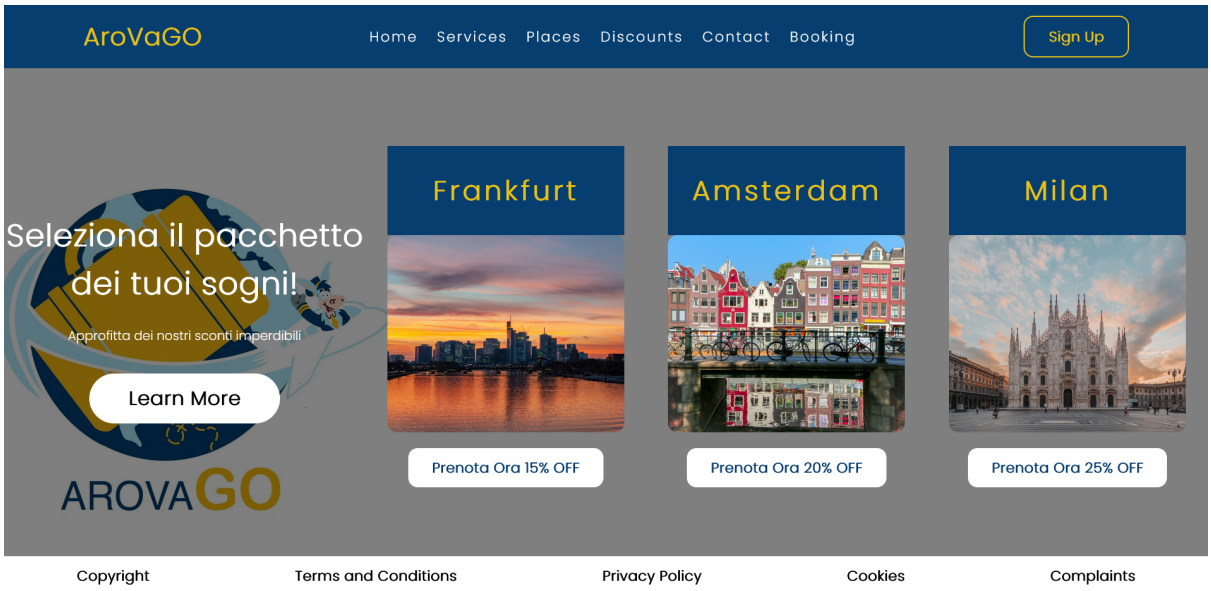
URL applicazione

https://apex.oracle.com/pls/apex/r/simoworkspace/arovago/home?session=3960723558967

Livello presentazione

Infine viene presentata una possibile idea di interfaccia grafica dinamica mediante l'utilizzo di HTML/CSS.

Interfaccia Grafica



▼ Codice HTML - index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Progetto BD</title>
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300&display=swap" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="navbar">
    <div class="logo">
      <h1>AroVaGO</h1>
    </div>
    <div class="menu">
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Places</a></li>
        <li><a href="#">Discounts</a></li>
        <li><a href="#">Contact</a></li>
        <li><a href="#">Booking</a></li>
      </ul>
    </div>
    <div class="signup">
      <a href="#">Sign Up</a>
    </div>
  </div>
  <div class="body">
    <div class="heading">
      <h1>Seleziona il pacchetto dei tuoi sogni!</h1>
      <br>
      <p>Approfitta dei nostri sconti imperdibili</p>
      <br>
      <br>
      <a href="#">Learn More</a>
    </div>
    <div class="tours">
      <div class="places">
        <h2>Frankfurt</h2>
        <img src="frankfurt.jpg" style="width: 300px; height: 250px; border-radius: 12px;">
        <br>
        <br>
```

```
        <a href="#">Prenota Ora 15% OFF</a>
      </div>
      <div class="places">
        <h2>Amsterdam</h2>
        <img src="Amsterdam.jpg" style="width: 300px; height: 250px; border-radius: 12px;">
        <br>
        <br>
        <a href="#">Prenota Ora 20% OFF</a>
      </div>
      <div class="places">
        <h2>Milan</h2>
        <img src="Milan.jpg" style="width: 300px; height: 250px; border-radius: 12px;">
        <br>
        <br>
        <a href="#">Prenota Ora 25% OFF</a>
      </div>
    </div>
  </div>
  <div class="footer">
    <a href="#">Copyright</a>
    <a href="#">Terms and Conditions</a>
    <a href="#">Privacy Policy</a>
    <a href="#">Cookies</a>
    <a href="#">Complaints</a>
  </div>
</body>
</html>
```

▼ Codice CSS - style.css

```
*{
  padding: 0px;
  margin: 0px;
  box-sizing: border-box;
  list-style: none;
  font-family: 'Poppins', sans-serif;

}
.navbar{
  width: 100%;
  height: 80px;
  background-color: #063E6F;
  display: flex;
  justify-content: space-around;
  align-items: center;
  color: #EAC017;
}
.menu ul{
  display: flex;
  align-items: center;
}
.menu ul li a{
  text-decoration: none;
  color: white;
  padding: 5px 12px;
  letter-spacing: 2px;
  font-size: 18px;
}
.menu ul li a:hover{
  border-bottom: 4px solid deepskyblue;
  transition: 0.4s;
}
.signup a{
  text-decoration: none;
  color: #EAC017;
  font-size: 18px;
  font-weight: bold;
  border-radius: 12px;
  padding: 12px 30px;
  border: 2px solid #EAC017;
}
.signup a:hover{
  background-color: skyblue;
  transition: 0.6s;
}
.body{
  width: 100%;
  height: 90vh;
  display: flex;
  justify-content: space-around;
  align-items: center;
  background-image: linear-gradient(rgba(0,0,0,0.50),rgba(0,0,0,0.50)),url(LogoUp.jpg);
  background-position: center;
  background-size: cover;
```

```
}
.heading{
  width: 30%;
  text-align: center;
  color: white;
}
.heading h1{
  font-size: 40px;
}
.heading a{
  text-decoration: none;
  color: black;
  font-size: 25px;
  font-weight: bold;
  border-radius: 45px;
  padding: 14px 50px;
  background-color: #fff;
}
.heading a:hover{
  letter-spacing: 3px;
  transition: 0.6s;
}
.tours{
  width: 70%;
  display: flex;
  justify-content: space-around;
}
.places{
  display: inline;
  text-align: center;
  border-radius: 12px;
}
.places h2{
  color: #EAC017;
  font-size: 35px;
  letter-spacing: 3px;
  border-radius: 1px;
  padding: 30px 30px;
  background-color: #063E6F;
}
.places a{
  text-decoration: none;
  color: #063E6F;
  font-weight: bold;
  font-size: 18px;
  border-radius: 12px;
  padding: 12px 30px;
  background-color: #fff;
}
.places a:hover{
  background-color: #000;
  letter-spacing: 3px;
  transition: 0.6s;
}
.footer{
  width: 100%;
  height: 50px;
  display: flex;
  justify-content: space-around;
  align-items: center;
}
.footer a{
  text-decoration: none;
  color: black;
  font-size: 18px;
  font-weight: bold;
}
.footer a:hover{
  text-decoration: underline;
  transition: 0.4s;
}
```

Le immagini sono state scelte dal sito **unsplash.com,** che è un sito web dedicato alla condivisione gratuita di fotografie da poter usare per ogni progetto.

Il Logo Invece è stato realizzato da noi con Procreate.

▼ Logo (realizzato con Procreate)

Popolamento della base dati

▼ POPOLAMENTO PACCHETTI

INSERT INTO PACCHETTI VALUES(5431,3);
INSERT INTO PACCHETTI VALUES(1234,4);
INSERT INTO PACCHETTI VALUES(7894,5);
INSERT INTO PACCHETTI VALUES(3214,2);
SELECT * FROM PACCHETTI;

```
INSERT INTO PACCHETTI VALUES(CONTATOREPACCHETTI.NEXTVAL,3);
INSERT INTO PACCHETTI VALUES(CONTATOREPACCHETTI.NEXTVAL,4);
INSERT INTO PACCHETTI VALUES(CONTATOREPACCHETTI.NEXTVAL,5);
INSERT INTO PACCHETTI VALUES(CONTATOREPACCHETTI.NEXTVAL,2);
SELECT * FROM PACCHETTI;
```

	ID	NUMEROPERSONE
1	1	3
2	2	4
3	3	5
4	4	2

▼ POPOLAMENTO SPOSTAMENTI

```
INSERT INTO SPOSTAMENTI VALUES(7845,'NAPOLI','AMSTERDAM',52.370216,4.895168,40.851775,14.268124,'10-GEN-2023','10-GEN-2023','16:45
INSERT INTO SPOSTAMENTI VALUES(4523,'BERLINO','FRANCOFORTE',50.110922,8.682127,52.520007,13.404954,'3-FEB-2023','3-FEB-2023','13:1
INSERT INTO SPOSTAMENTI VALUES(1563,'LONDRA','MADRID',40.416775,-3.703790,51.507351,-0.127758,'5-APR-2023','5-APR-2023','17:30','1
INSERT INTO SPOSTAMENTI VALUES(1985,'BUDAPEST','ROMA',41.902783,12.496365,47.497912,19.040235,'21-MAR-2023','21-MAR-2023','21:45',
INSERT INTO SPOSTAMENTI VALUES(3689,'PARIGI','MILANO',45.465422,9.185924,48.856614,2.352222,'15-MAG-2023','15-MAG-2023','18:31','1
SELECT * FROM SPOSTAMENTI;
```

	ID	LUOGOPARTENZA	LUOGOARRIVO	LATITUDINEARRIVO	LONGITUDINEARRIVO	LATITUDINEPARTENZA	LONGITUDINEPARTENZA	DATAPARTENZA	DATAARRIVO	ORAARRIVO	ORAPARTENZA
1	7845	NAPOLI	AMSTERDAM	52,370216	4,895168	40,851775	14,268124	10-GEN-23	10-GEN-23	16:45	14:00
2	4523	BERLINO	FRANCOFORTE	50,110922	8,682127	52,520007	13,404954	03-FEB-23	03-FEB-23	13:10	12:00
3	1563	LONDRA	MADRID	40,416775	-3,70379	51,507351	-,127758	05-APR-23	05-APR-23	17:30	15:00
4	1985	BUDAPEST	ROMA	41,902783	12,496365	47,497912	19,040235	21-MAR-23	21-MAR-23	21:45	20:00
5	3689	PARIGI	MILANO	45,465422	9,185924	48,856614	2,352222	15-MAG-23	15-MAG-23	18:31	17:00

▼ POPOLAMENTO SPOSTAMENTI_SCELTI

```
INSERT INTO SPOSTAMENTI_SCELTI VALUES(1,7845);
INSERT INTO SPOSTAMENTI_SCELTI VALUES(3,1563);
INSERT INTO SPOSTAMENTI_SCELTI VALUES(4,3689);
SELECT * FROM SPOSTAMENTI_SCELTI;
```

	ID_PACCHETTO	ID_SPOSTAMENTO
1	1	7845
2	3	1563
3	4	3689

▼ POPOLAMENTO ALBERGHI

```
INSERT INTO ALBERGHI VALUES(7021,'HOTEL KVARA','ALBERGO','10-GEN-2023','17-GEN-2023',30);
INSERT INTO ALBERGHI VALUES(8520,'HOTEL CIRO','ALBERGO','15-MAG-2023','25-MAG-2023',45);
INSERT INTO ALBERGHI VALUES(6320,'VINNYMOSCATO','TRATTORIA','21-MAR-2023','28-MAR-2023',25);
SELECT * FROM ALBERGHI;
```

	ID	NOME	TIPOLOGIA	DATACHECKIN	DATACHECKOUT	COSTOGIORNALIERO
1	7021	HOTEL KVARA	ALBERGO	10-GEN-23	17-GEN-23	30
2	8520	HOTEL CIRO	ALBERGO	15-MAG-23	25-MAG-23	45
3	6320	VINNYMOSCATO	TRATTORIA	21-MAR-23	28-MAR-23	25

▼ POPOLAMENTO SOGGIORNI

```
INSERT INTO SOGGIORNI VALUES(2436,7021);
INSERT INTO SOGGIORNI VALUES(7536,8520);
INSERT INTO SOGGIORNI VALUES(4561,6320);
SELECT * FROM SOGGIORNI;
```

	CODICE	ID_ALBERGO
1	2436	7021
2	7536	8520
3	4561	6320

▼ POPOLAMENTO SOGGIORNI_SCELTI

```
INSERT INTO SOGGIORNI_SCELTI VALUES (1,2436);
INSERT INTO SOGGIORNI_SCELTI VALUES (4,7536);
INSERT INTO SOGGIORNI_SCELTI VALUES (3,4561);
SELECT * FROM SOGGIORNI_SCELTI;
```

	ID_PACCHETTO	CODICE_SOGGIORNO
1	1	2436
2	4	7536
3	3	4561

▼ POPOLAMENTO MEZZI_DI_TRASPORTO

```
INSERT INTO MEZZI_DI_TRASPORTO VALUES(123, 3,1,'02:45',150,7845);
INSERT INTO MEZZI_DI_TRASPORTO VALUES(745,5,1,'02:30',200,1563);
INSERT INTO MEZZI_DI_TRASPORTO VALUES(845,2,1,'01:31',300,3689);
SELECT * FROM MEZZI_DI_TRASPORTO;
```

	ID	NUMEROVIAGGIATORI	NUMEROMEZZI	TEMPOPERCORRENZA	PREZZO	ID_SPOSTAMENTO
1	123	3	1	02:45	150	7845
2	745	5	1	02:30	200	1563
3	845	2	1	01:31	300	3689

▼ POPOLAMENTI CAMERE

```
INSERT INTO CAMERE VALUES(54,'STANDARD',3,2436);
INSERT INTO CAMERE VALUES(63,'PREMIUM',2,7536);
INSERT INTO CAMERE VALUES(99,'DIAMOND',5,4561);
SELECT * FROM CAMERE;
```

	ID	TIPOLOGIA	NUMEROOCCUPANTI	SOGGIORNO
1	54	STANDARD	3	2436
2	63	PREMIUM	2	7536
3	99	DIAMOND	5	4561

▼ POPOLAMENTO SERVIZI_OFFERTI

```
INSERT INTO SERVIZI_OFFERTI VALUES(12,'SERVIZIO IN CAMERA',54);
INSERT INTO SERVIZI_OFFERTI VALUES(10,'RISTO-BAR',63);
INSERT INTO SERVIZI_OFFERTI VALUES(3,'ALL INCLUSIVE',99);
SELECT * FROM SERVIZI_OFFERTI;
```

	CODICE	TIPO	CAMERA
1	12	SERVIZIO IN CAMERA	54
2	10	RISTO-BAR	63
3	3	ALL INCLUSIVE	99

▼ POPOLAMENTO PRENOTAZIONI

```
INSERT INTO PRENOTAZIONI VALUES(CONTATOREPRENOTAZIONI.NEXTVAL,365,'10-GEN-2023','17-GEN-2023',1,'PSTCRI70E25F205N');
INSERT INTO PRENOTAZIONI VALUES(CONTATOREPRENOTAZIONI.NEXTVAL,850,'15-MAG-2023','25-MAG-2023',3,'RSSMRA65L08F205U');
INSERT INTO PRENOTAZIONI VALUES(CONTATOREPRENOTAZIONI.NEXTVAL,1000,'21-MAR-2023','28-MAR-2023',4,'BNCMRC97B06F205C');
SELECT * FROM PRENOTAZIONI;
```

	CODICE	COSTOCOMPLESSIVO	DATAPARTENZA	DATARITORNO	PACCHETTO	UTENTE
1	17	365	10-GEN-23	17-GEN-23	1	PSTCRI70E25F205N
2	18	850	15-MAG-23	25-MAG-23	3	RSSMRA65L08F205U
3	19	1000	21-MAR-23	28-MAR-23	4	BNCMRC97B06F205C

▼ POPOLAMENTO VIAGGIATORI

```
INSERT INTO VIAGGIATORI VALUES(466,'CIRO','PASTORE',17);
INSERT INTO VIAGGIATORI VALUES(467,'GIOSUELE','PASTORE',17);
INSERT INTO VIAGGIATORI VALUES(469,'MARIANNA','PASTORE',17);
INSERT INTO VIAGGIATORI VALUES(891,'MARIO','ROSSI',18);
INSERT INTO VIAGGIATORI VALUES(892,'FILIPPO','ROSSI',18);
INSERT INTO VIAGGIATORI VALUES(893,'GIACOMO','ROSSI',18);
INSERT INTO VIAGGIATORI VALUES(894,'FILOMENA','ROSSI',18);
INSERT INTO VIAGGIATORI VALUES(895,'VALENTINA','ROSSI',18);
INSERT INTO VIAGGIATORI VALUES(763,'MARCO','BIANCHI',19);
INSERT INTO VIAGGIATORI VALUES(760,'GIADA','DI LORENZO',19);
SELECT * FROM VIAGGIATORI;
```

	ID	NOME	COGNOME	PRENOTAZIONE
1	466	CIRO	PASTORE	17
2	467	GIOSUELE	PASTORE	17
3	469	MARIANNA	PASTORE	17
4	891	MARIO	ROSSI	18
5	892	FILIPPO	ROSSI	18
6	893	GIACOMO	ROSSI	18
7	894	FILOMENA	ROSSI	18
8	895	VALENTINA	ROSSI	18
9	763	MARCO	BIANCHI	19
10	760	GIADA	DI LORENZO	19

▼ POPOLAMENTO UTENTI

```
INSERT INTO UTENTI VALUES('PSTCRI70E25F205N','CIRO','PASTORE');
INSERT INTO UTENTI VALUES('RSSMRA65L08F205U','MARIO','ROSSI');
INSERT INTO UTENTI VALUES('BNCMRC97B06F205C','MARCO','BIANCHI');
SELECT * FROM UTENTI;
```

	CF	NOME	COG...
1	PSTCRI70E25F205N	CIRO	PASTORE
2	RSSMRA65L08F205U	MARIO	ROSSI
3	BNCMRC97B06F205C	MARCO	BIANCHI

▼ POPOLAMENTO CARTE_DI_CREDITO

```
INSERT INTO CARTE_DI_CREDITO VALUES(99645,'MASTERCARD','CIRO PASTORE','486','01-FEB-26','PSTCRI70E25F205N');
INSERT INTO CARTE_DI_CREDITO VALUES(88201,'VISA','MARCO BIANCHI','774','01-NOV-24','BNCMRC97B06F205C');
SELECT * FROM CARTE_DI_CREDITO;
```

	ID	CIRCUITOGESTORE	INTESTATARIO	CODICE	DATASCADENZA	UTENTE
1	99645	MASTERCARD	CIRO PASTORE	486	01-FEB-26	PSTCRI70E25F205N
2	88201	VISA	MARCO BIANCHI	774	01-NOV-24	BNCMRC97B06F205C

▼ POPOLAMENTO BONIFICI_BANCARI

```
INSERT INTO BONIFICI_BANCARI VALUES(11234,'05-NOV-2022','RSSMRA65L08F205U');
SELECT * FROM BONIFICI_BANCARI;
```

Bibliografia:

- Chianese A., Moscato V., Picariello A., Sansone L., *Sistemi di basi di dati e applicazioni*,Apogeo,2015
- Fiore Alfredo, *Elaborato finale in Basi di Dati Oracle APEX: Esempio d'uso*, Napoli,2018