**Capitolo 1**

**Introduzione**

**1.1 Scopo**

Si vuole progettare una **base dati** per una multinazionale informatica che contenga informazioni relative alla gestione di un servizio di **car pooling** in territorio nazionale, con l’obiettivo di diffondere l’uso di una mobilità flessibile e personalizzata in termini di percorsi e costi. I servizi che la multinazionale mette a disposizione sono:

**1)** Gli utenti devono potersi registrare da un sito internet;

**2)** Gli utenti devono avere la possibilità di consultare vari viaggi e prenotarli;

**3)** Gli utenti, sia che essi siano autisti o passeggeri, devono poter commentare o dare un giudizio al viaggio.

**1.2 Specifiche sui dati**

La società prevede che i servizi sopraelencati siano accessibili dai propri utenti mediante un sito internet. In particolare si vogliono visualizzare le informazioni riguardo:

--Gli **utenti** con nome, cognome, codice fiscale, recapito telefonico ed e-mail; inoltre gli utenti possono essere:

-**autisti**, ai quali viene assegnato un username, una password, e all’atto della registrazione devono anche inserire una foto, le informazioni relative alla patente e all’auto;

-**passeggeri**, ai quali viene assegnato un username, una password, e all’atto della registrazione devono anche inserire una foto e un documento di riconoscimento;

--I **viaggi**, di cui gli autisti devono specificare i dettagli relativi la partenza e il ritorno, il numero di posti disponibili, il costo del viaggio e la durata approssimativa;

-- Le **città**, con il relativo nome e CAP;

--Le **prenotazioni**, messe a disposizione dall’autista il quale deve specificare a quale viaggio si riferisce e quanti posti rimangono;

--I **feedback**, ovvero le recensioni relative all’autista, passeggero o viaggio, delle quali va specificato il mittente, il destinatario e a quale viaggio si riferiscono.

Si supponga che nel primo anno di attività il sistema debba gestire:

- l’anagrafica di circa 10000 utenti (2000 autisti e 8000 passeggeri);

- 8000 comuni;

- circa 100000 viaggi e 500000 prenotazioni.

Esistono quindi 3 categorie di utenti:

-Il **DBA**, che possiede tutti i privilegi sullo schema della base di dati;

-Gli **autisti**, che provvederanno a creare, aggiornare o eliminare i vari viaggi e le relative prenotazioni;

-I **passeggeri**, che possono prenotare un viaggio, scegliere l’autista che più li aggrada, e visualizzare tutte le informazioni relative.

**Capitolo 2**

**Progettazione della base di dati**

**2.1 Progettazione concettuale**

Innanzitutto va definito uno **schema E/R portante**, ovvero lo schema che presenta solo i concetti fondamentali, ovvero le entità e le associazioni. Si passa poi ad un raffinamento aggiungendo tutte le specifiche prima elencate.

**2.1.1 Schema E/R portante**

Nella nostra realtà di interesse le entità fondamentali analizzate sono 5. In particolare troviamo: **utente**, **viaggio**, **prenotazione**, **città** e **feedback**.

Vediamo che l’entità **utente** è legata a **feedback**, mediante le associazioni **mittente** e **destinatario**, l’entità **feedback** è a sua volta legata a **viaggio** mediante l’associazione **valutazione**, **viaggio** è legata all’entità **città**, mediante le associazioni **andata** e **ritorno**, e all’ entità **prenotazione** mediante l’associazione **destinazione**.

Per quanto riguarda le cardinalità, notiamo come:

1) Un **utente** può effettuare, o ricevere, zero o più **feedback**, mentre a sua volta il **feedback**, sarà effettuato o ricevuto da uno e un solo **utente**. Quindi le associazioni **mittente** e **destinatario** sarà del tipo (1 a N).

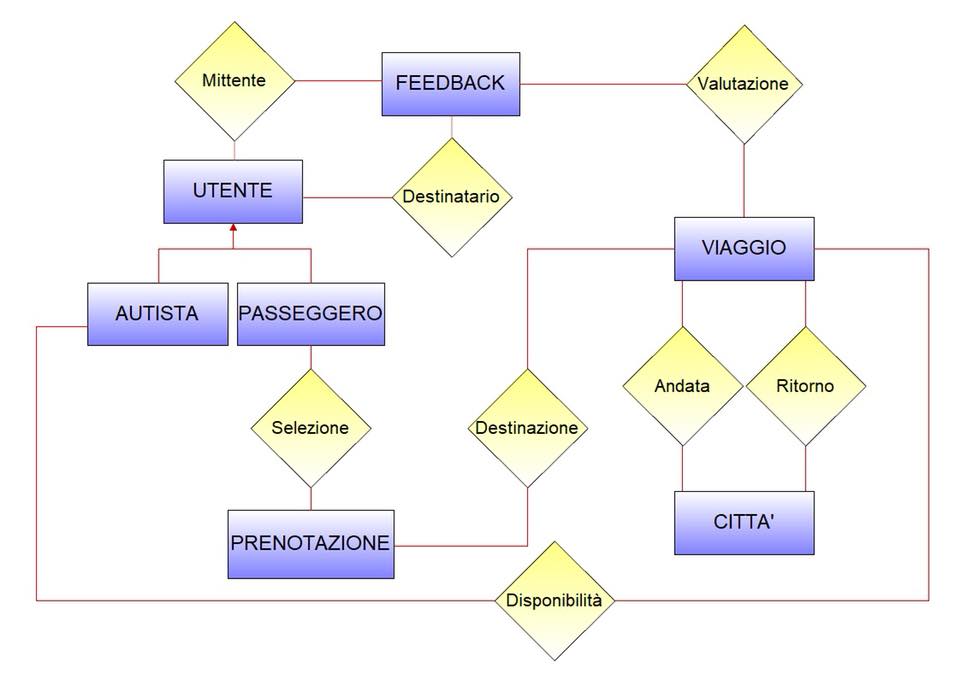
2) Un **feedback** può essere relativo a uno e un solo **viaggio**, mentre il **viaggio** può ricevere da zero a più **feedback**. Quindi l’associazione **valutazione** sarà del tipo (1 a N).

3) Un **viaggio** parte e ritorna da una e una sola **città**, mentre una **città** può essere relativa a zero o più **viaggi**. Quindi le associazioni **andata** e **ritorno** sono entrambe del tipo (1 a N).

4) L’entità **città** è legata solo a **viaggio** tramite le associazioni **andata** e **ritorno** analizzate precedentemente.

5) Una **prenotazione** è relativa a uno e un solo **viaggio**, mentre un **viaggio** può essere soggetto da zero a più **prenotazioni**. Quindi anche l’associazione **destinazione** è del tipo (1 a N).

Vi è qui riportato le schema E/R portante.



**2.1.2 Schema E/R finale**

Esaminando ora le specifiche si va a raffinare lo schema E/R portante aggiungendo i vari attributi. Si nota inoltre che precedentemente non sono stati trattati ne gli autisti ne i passeggeri: ebbene queste due sono entità che derivano dall’entità **utente**, e quindi quest’ultima è estesa. Si aggiungono quindi le entità **autista** e **passeggero**, entrambe delle sottoclassi, e occorre quindi analizzare anche le nuove associazioni che si creano, ovvero **disponibilità** che lega **autista** a **viaggio**, e **selezione** che lega **passeggero** a **prenotazioni**:

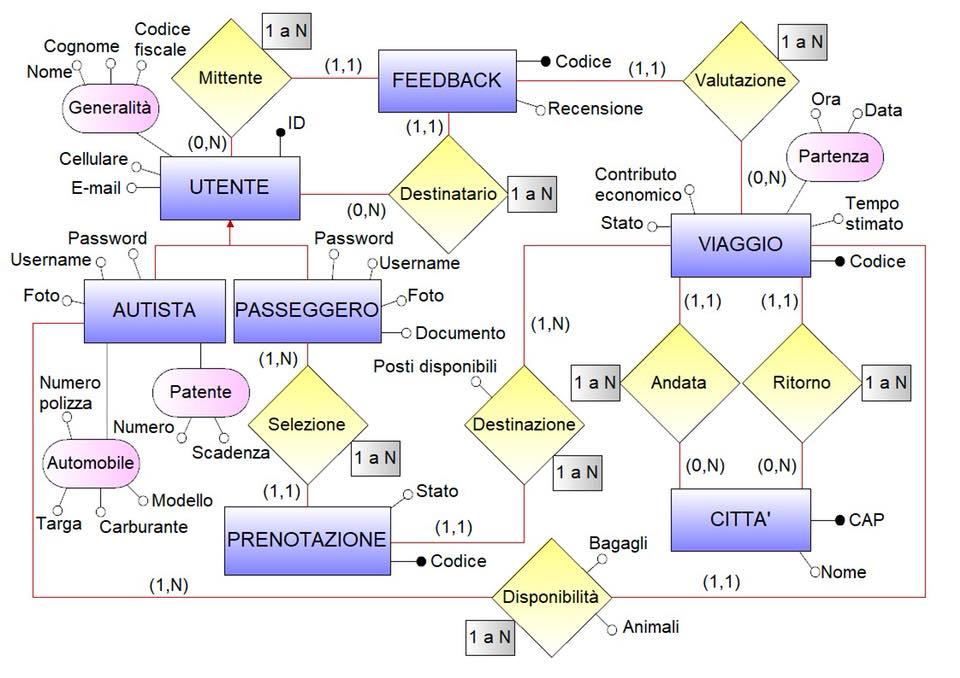
6) Un **autista** può mettere a disposizione uno o più **viaggi**, mentre un **viaggio** può essere messo a disposizione da un solo **autista**. Quindi l’associazione **disponibilità** è del tipo (1 a N), e ha anche degli attributi propri, ovvero il numero di bagagli e di animali che possono essere portati.

7) Un **passeggero**, una volta registrato, può effettuare una o più **prenotazioni**, mentre una **prenotazione** è relativa a un solo passeggero. Quindi l’associazione **selezione** è del tipo (1 a N) e aggiunge come attributo il numero di posti ancora disponibili.

**2.1.3 Sintesi progettazione concettuale**

Abbiamo quindi uno schema E/R finale con 7 entità, di cui 2 sottoclassi di un’unica superclasse, e 7 associazioni tutte con cardinalità (1 a N). Inoltre per ciascuna entità abbiamo scelto di utilizzare un attributo che identifica univocamente ogni occorrenza di entità.

Riportiamo qui lo schema E/R finale.



**2.2 Progettazione Logica**

La progettazione logica prevede la definizione dello schema E/R finale in un insieme di vincoli e relazioni (o tabelle). Essa è divisa in due fasi:

1) Trasformazione: dove si semplificano quei costrutti del modello E/R che non sono direttamente traducibili in quello relazionale, cioè gli attributi composti, multivalore e la risoluzione delle gerarchie;

2) Traduzione: cioè l’effettiva traduzione del modello E/R trasformato in uno schema relazionale.

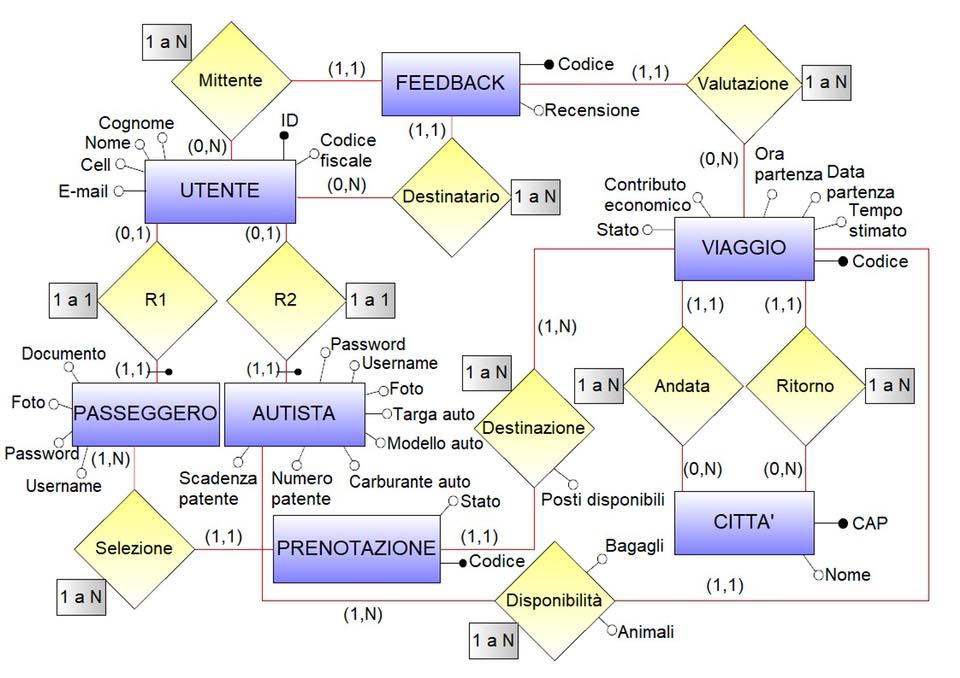
**2.2.1 Trasformazione**

In relazione allo schema sopra riportato, la trasformazione prevede la semplificazione degli attributi composti “Generalità” di **utente**, “Partenza” di **viaggio**, e “Automobile” e “Patente” di **autista**. Si deve inoltre risolvere la gerarchia di **utente**.

-Attributi composti. Come da regola, gli attributi composti si eliminano e i suoi attributi componenti vanno legati ciascuno all’entità a cui in origine l’attributo composto era legato. Quindi, gli attributi “Nome”, “Cognome” e “Codice fiscale” che componevano “Generalità” vanno legati a **utente**; gli attributi “Targa”, “Modello”, “Carburante” e ”Polizza” che componevano “Automobile” e “Numero” e “Scadenza” che componevano “Patente” vanno tutti legati a **autista**, mentre gli attributi “Data” e “Ora” che componevano “Partenza” vanno legati direttamente a **viaggio**.

-Gerarchie. Visto che le operazioni che se effettuano sulle entità **autista** e **passeggero** sono diverse tra loro, e visto che l’entità **utente** è ancora utile perché permette ad una persona la registrazione, abbiamo deciso di staccare entrambe le sottoclassi dalla superclasse e di legarle a quest’ultima con 2 associazioni (1 a 1), dove la cardinalità minima dalla parte della superclasse è zero(perché un padre può non avere figli ma un figlio deve avere per forza un padre), che per semplicità abbiamo chiamato **R1** e **R2**.

Riportiamo in seguito lo schema E/R trasformato.



**2.2.2 Traduzione**

La fase di trasformazione comporta la traduzione delle associazioni e delle entità tramite le regole previste nel modello relazionale. In particolare:

- Le entità si trasformano in relazioni con nome al plurale aventi come attributi gli attributi dell’entità e come chiave primaria l’identificatore dell’entità;

-Le associazioni **mittente, destinatario, valutazione, andata, ritorno, selezione, destinazione** e **disponibilità**, che sono tutte (1 a N), scompaiono e gli identificatori primari delle entità lato N si aggiungono agli attributi delle entità lato 1 e diventano chiavi esterne che referenziano le entità lato N;

-Le associazioni **R1** e **R2**, entrambe associazioni (1 a 1), sono scomparse e si è scelto di aggiungere l’identificatore primario di **utente** agli attributi sia di **autista** sia di **passeggero**, facendolo così diventare in entrambe superchiave e chiave esterna.

In seguito è riportato lo schema relazionale completo risultante dalla progettazione logica.

**UTENTI**(Id, Codice\_fiscale, Cognome, Nome, Cell, E-mail)

**AUTISTI**(Id:UTENTI, Username, Password, Numero\_polizza, Targa\_auto, Modello\_auto, Carburante\_auto, Numero\_patente, Scadenza\_patente, Animali, Bagagli, Foto)

**PASSEGGERI**(Id:UTENTI, Username, Password, Foto, Documento)

**PRENOTAZIONI**(Codice, Viaggio:VIAGGI, Passeggero::PASSEGGERI, Stato)

**VIAGGI**(Codice, Tempo\_stimato, Data\_partenza, Ora\_partenza, Posti\_disponibili, Contributo\_economico, Andata:CITTA, Ritorno:CITTA, Stato, Autista::AUTISTI)

**CITTA**(CAP, Nome)

**FEEDBACK**(Codice, Recensione, Cod\_viaggio:VIAGGI, Mittente:UTENTE, Destinatario:UTENTE)

**Capitolo 3**

**Dimensionamento**

**3.1 Tipi di dato**

Determinata la quantità di informazioni che verranno immagazzinate nella Base di Dati, e ultimata la progettazione logica, si è passati a determinare i tipi di dato utilizzati nella creazione delle tabelle. In particolare, sono stati utilizzati i seguenti tipi:

-CHAR(N): è stato utilizzato il tipo char per disporre di un range di caratteri alfanumerici che permettono di comporre codici identificativi a lunghezza fissa completi, e non solo. Esempi dell’utilizzo di questo tipo, che rendono al meglio l’utilità della scelta, sono: “Targa\_auto” in **autisti**, in cui si è scelto un char(7) visto che sul territorio nazionale una targa è composta da 7 caratteri alfanumerici, “Codice\_fiscale” di **utenti,** per cui si è usato un char(16), e “Stato” di **viaggi** e **prenotazioni**, per cui si è usato una char(6) visto che lo stato può essere “aperto” o “chiuso”.Esso prevede in ORACLE una collocazione di n byte per n caratteri alfanumerici;

-VARCHAR2(N): è stato utilizzato il tipo varchar2 per gli attributi di tipo testuale a lunghezza variabile che sono più propensi a variazioni, come Nome, Cognome ... . Il tipo varchar2 prevede in ORACLE una collocazione di n byte per n caratteri testuali entro la dimensione dei 4000 byte per valore;

-NUMBER: dato l’utilizzo di codici identificativi numerici per la loro gestione tramite sequenze, è stato utilizzato il tipo Number per questi campi. Esso prevede in ORACLE una collocazione di n/2 + 2 byte per n cifre;

-DATE: è stato utilizzato il tipo Date negli attributi di tipo data. Esso prevede in ORACLE una collocazione di 7byte;

-CLOB: è stato utilizzato il tipo Clob per attributi come “Recensione” per permettere testi anche molto lunghi(fino a 4gb) senza limitarli. Esso prevede in ORACLE una collocazione di 4 byte relativi ai soli puntatori, poiché questi oggetti sono gestiti da un apposito tablespace;

-BLOB: è stato utilizzato per gli attributi “Foto” e “Documento” per permettere l’inserimento di immagini in qualunque risoluzione. Esso prevede in ORACLE una collocazione di 4 byte, e i suoi oggetti sono gestiti da un apposito tablespace;

-INTEGER: è stato utilizzato per i numeri interi, e prevede in Oracle una collocazione di 4 byte.

**3.2 Tabelle**

Una volta date tutte le informazioni necessarie relative ai dati, è possibile calcolare la quantità di spazio che ciascuna tabella occuperà in memoria; in particolare in ogni tabella viene riportato sia il nome dei singoli campi che le vanno a formare, sia il tipo di dati scelti per essi, con relativa quantità di memoria occupata che vanno ad occupare. Successivamente vengono riportate le operazioni algebriche che permettono di calcolare con precisione il numero di byte, o suoi multipli, utilizzati da ogni singola tabella. La dimensione totale della base di dati è uguale alla somma delle dimensioni delle singole tabelle più un valore di tolleranza; che viene utilizzato nel caso in cui la base dati cresca nel tempo. Le tabelle usate per il calcolo della dimensione della base dati sono formate dai campi attributo, tipo, byte e dimensione initial. Di seguito vi sono riportate tutte le tabelle con una piccola descrizione.

**Utenti**: sono contraddistinti da un Id e contengono le informazioni basilari per ciascun utente, come nome, cognome e codice fiscale.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **occorrenza: 10000** |
|  |  |  |  |
| Id | NUMBER | 5 | 50k |
| Codice\_fiscale | CHAR | 16 | 160k |
| Cognome | VARCHAR2 | 50 | 500K |
| Nome | VARCHAR2 | 50 | 500K |
| Cell | NUMBER | 8 | 80K |
| e-mail | VARCHAR2 | 50 | 500K |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 1790k |

**Autisti**: sono contraddistinti da un Id salvato già in utenti e contengono informazioni relative il veicolo e la patente, oltre che un username e una password assegnati all’atto della registrazione.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **occorrenza: 2000** |
|  |  |  |  |
| Id | NUMBER | 5 | 10k |
| Username | VARCHAR2 | 20 | 40k |
| Password | VARCHAR2 | 30 | 60k |
| Foto | BLOB | 4 | 8k |
| Numero\_polizza | VARCHAR2 | 50 | 100k |
| Targa\_auto | CHAR | 7 | 14k |
| Modello\_auto | VARCHAR2 | 30 | 60k |
| Carburante\_auto | VARCHAR2 | 15 | 30k |
| Numero\_patente | VARCHAR2 | 10 | 20k |
| Scadenza\_patente | DATE | 7 | 14k |
| Animali | INTEGER | 4 | 8k |
| Bagagli | INTEGER | 4 | 8k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 372k |

**Passeggeri**: sono contraddistinti da un Id salvato già in utenti e contengono anche un documento e una foto del passeggero, oltre che un username e una password assegnati all’atto della registrazione.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **Occorrenza: 8000** |
|  |  |  |  |
| Id | NUMBER | 5 | 40k |
| Username | VARCHAR2 | 20 | 160k |
| Password | VARCHAR2 | 30 | 240k |
| Documento | BLOB | 4 | 32k |
| Foto | BLOB | 4 | 32k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 504k |

**Prenotazioni**: sono contraddistinti da un codice numerico e contengono, oltre al codice del viaggio a cui si riferiscono e all’Id del passeggero che l’ha effettuata, anche una variabile stato per indicare se la prenotazione è aperta o chiusa.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **Occorrenza: 500000** |
|  |  |  |  |
| Codice | NUMBER | 5 | 2500k |
| Viaggio | NUMBER | 5 | 2500k |
| Passeggero | NUMBER | 5 | 2500k |
| Stato | CHAR | 6 | 3000k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 10500k |

**Viaggi**: sono contraddistinti da un codice numerico e contengono, oltre all’Id dell’autista che l’ha messo a disposizione, informazioni relative la partenza e il ritorno, come la città, e il prezzo di ciascun viaggio, che ogni passeggero deve pagare, che l’autista sceglie anche in relazione al carburante del proprio veicolo.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **Occorrenza: 100000** |
|  |  |  |  |
| Codice | NUMBER | 5 | 500k |
| Tempo\_stiamato | VARCHAR2 | 15 | 1500k |
| Data\_partenza | DATE | 7 | 700k |
| Ora\_partenza | VARCHAR2 | 5 | 500k |
| Posti\_disponibili | INTEGER | 4 | 400k |
| Contributo\_economico | VARCHAR2 | 10 | 1000k |
| Andata | NUMBER | 5 | 500k |
| Ritorno | NUMBER | 5 | 500k |
| Stato | CHAR | 6 | 600k |
| Autista | NUMBER | 5 | 500k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 6700k |

**Citta**: sono contraddistinte da un CAP e vi è specificato anche il nome di ciascuna.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **Occorrenza: 8000** |
|  |  |  |  |
| CAP | NUMBER | 5 | 40k |
| Nome | VARCHAR2 | 50 | 400k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 440k |

**Feedback**: sono contraddistinti da un codice numerico e contengono la recensione completa del viaggio, compresa chi l’ha scritta e chi la ricevuta, nel caso in cui la recensione contenga anche un commento all’autista o a uno dei passeggeri.

|  |  |  |  |
| --- | --- | --- | --- |
| **attributo** | **tipo** | **byte** | **Occorrenza: 10000** |
|  |  |  |  |
| Codice | NUMBER | 5 | 50k |
| Recensione | CLOB | 4 | 40k |
| Cod\_viaggio | NUMBER | 5 | 50k |
| Mittente | NUMBER | 5 | 50k |
| Destinatario | NUMBER | 5 | 50k |
|  |  |  |  |
| **STORAGE** |  |  | **TOTALE** |
| dati+blocchi |  |  | 240k |

Riportiamo in seguito una tabella dove vi è riportata la dimensione di ciascuna tabella e la dimensione della base dati,calcolata già aggiungendo una tolleranza, che per nostra scelta è almeno il doppio dell’effettiva misura del tablespace.

|  |  |  |
| --- | --- | --- |
| **OGGETTO** | **NOME** | **INITIAL** |
| Tabella | UTENTI | 1790k |
| Tabella | AUTISTI | 372k |
| Tabella | PASSEGGERI | 504k |
| Tabella | PPRENOTAZIONI | 10500k |
| Tabella | VIAGGI | 6700k |
| Tabella | CITTÀ | 440k |
| Tabella | FEEDBACK | 240k |
| Database | BLA BLA CAR | 100M |

**Capitolo 4**

**Progettazione Fisica**

**4.1 Creazione dei tablespace**

Come primo passo abbiamo effettuato la creazione di due tablespace:

- il primo denominato *Proggetto\_TS*, ovvero l’unità logica che conterrà tutte le informazioni presenti nel nostro Database;

-il secondo denominato *Proggetto\_TS\_lob*, che ha lo scopo di contenere i testi dell’attributo “Recensione” della tabella **Feedback**, le immagini degli attributi “Foto” di **Autisti** e **Passeggeri**, e le scansioni tramite scanner per l’attributo “Documento” dei **Passeggeri**.

|  |
| --- |
| **create tablespace** Progetto\_TS datafile 'C:\oradata\Progetto\_TS.dbf'  **size** 100M;  **create tablespace** Progetto\_TS\_lob datafile 'C:\oradata\Progetto\_TS\_lob.dbf' size 2M; |

Tali operazioni sono state effettuate tramite le credenziali di SYSTEM, che dispone di completa libertà operativa sul Database appena creato.

Il valore in Size è stato calcolato guardando il dimensionamento di tutte le tabelle e incrementato con una certa tolleranza per avere lo spazio sufficiente di memorizzazione nel caso di una crescita del database.

**4.2 Creazione del DBA**

Prima di effettuare ogni altra operazione, è necessaria la creazione di un DBA del tablespace *Proggetto\_ts*, in modo da garantirgli pieni poteri esclusivamente all’interno di quest’ultimo ed impedendo, di conseguenza, ogni operazione al di fuori di esso.

|  |
| --- |
| **create user** dba\_Progetto **default tablespace** Progetto\_TS  **identified by** fantastici4;  **grant dba, unlimited tablespace to** dba\_Progetto; |

Per assegnare il ruolo di DBA al nostro utente si è scelto di assegnargli direttamente il ruolo dba previsto da Oracle ed assegnare a quest’ultimo spazio illimitato su "*Proggetto\_TS*". Inoltre la definizione di tale utente ha comportato, da parte del DBMS, la creazione di uno schema omonimo che conterrà al suo interno tutte le informazioni che andremo a definire successivamente, e al quale dovranno fare riferimento gli altri eventuali utenti in caso di necessità di accesso a quelle informazioni.

**4.4 Creazione delle tabelle**

La creazione di tutte le tabelle che vedremo in seguito è stata effettuata utilizzando il dba appena creato. In particolare, per ciascuna tabella si è scelto di non arrotondare lo storage di initial e mantenere tutte le misure in kbyte, ovvero 1000 byte. Inoltre, per la consistenza dei dati, si è scelto di applicare ai campi necessari il vincolo di not null, e per alcuni anche un valore di default che può servire da esempio. Inoltre si è scelto di utilizzare il campo constraint in tutti i vincoli di chiave primaria, cosicchè da avere ben chiara quale chiave primaria venisse violata, grazie al nome che gli abbiamo fornito, nel caso in cui questo accada.

Di seguito vi sono riportate tutte le istruzioni per e varie tabelle.

|  |
| --- |
| **Create table** UTENTI(  Id **number(5),**  Codice\_fiscale **char(16) NOT NULL**,  Cognome **varchar2(50) NOT NULL**,  Nome **varchar2(50) NOT NULL**,  Cell **number(11),**  E\_mail **varchar2(50) DEFAULT** 'esempio@hotmail.com' **NOT NULL**,  **constraint** PK\_UTENTI **primary key**(Id)  )  **Storage** (**Initial** 1790k);  **Create table** AUTISTI(  Id **number(5),**  Username **varchar2(20) DEFAULT** 'nome.cognome',  Password **varchar2(30) DEFAULT** '\*\*\*\*\*\*\*\*\*\*',  Numero\_polizza varchar2(50) NOT NULL,  Targa\_auto **char(7) NOT NULL**,  Modello\_auto **varchar2(30),**  Carburante\_auto **varchar2 (15),**  Numero\_patente **varchar(10) NOT NULL**,  Scadenza\_patente **date**,  Animali **integer**,  Bagagli **integer**,  Foto **blob**,  **constraint** PK\_AUTISTI **primary key**(Id)  )  **Storage** (**Initial** 372k);  **Create table** PASSEGGERI(  Id **number(5),**  Username **varchar2(20) DEFAULT** 'nome.cognome',  Pssword **varchar2(30) DEFAULT** '\*\*\*\*\*\*\*\*\*\*',  Documento **blob**,  Foto **blob**,  **constraint** PK\_PASSEGGERI **primary key**(Id)  )  **Storage** (**Initial** 504k);  **Create table** PRENOTAZIONI(  Codice **number(6),**  Viaggio **number(6),**  Passeggero **number(5),**  Stato **char(6) NOT NULL,**  **constraint** PK\_PRENOTAZIONI **primary key**(Codice)  )  **Storage** (**Initial** 10500k);  **Create table** VIAGGI(  Codice **number(6),**  Tempo\_stimato **varchar2(15) DEFAULT** 'Da definire',  Data\_partenza **date NOT NULL**,  Ora\_partenza **varchar2(5) DEFAULT** '--:--',  Posti\_disponibili **integer**,  Contributo\_economico **varchar2(10) DEFAULT** 'Variabile',  Andata **number(5) NOT NULL**,  Ritorno **number(5),**  Stato **char(6) NOT NULL**,  Autista **number(5),**  **constraint** PK\_VIAGGI **primary key**(Codice)  )  **Storage** (**Initial** 6700k);  **Create table** CITTA(  Cap **number(5),**  Nome **varchar2(50) NOT NULL**,  **constraint** PK\_CITTA **primary key**(Cap)  )  **Storage** (**Initial** 440k);  **Create table** FEEDBACK(  Codice **number(5),**  Recensione **clob DEFAULT** 'Nessuna recensione',  Cod\_Viaggio **number(6),**  Mittente **number(5),**  Destinatario **number(5),**  **constraint** PK\_FEEDBACK **primary key**(Codice)  )  **Storage** (**Initial** 240k); |

**4.4 Creazione vincoli aggiuntivi**

Per la creazione dei vincoli di integrità referenziale è stato usato il comando di alter table che permette di modificare una tabella già creata precedentemente. Si è preferito tale sistema in modo da non essere condizionati dall’ordine di creazione delle tabelle, data la necessità di dover in caso contrario definire prima le tabelle referenziate poi le referenzianti. Si specifica che dove non è stata definita nessuna politica di reazione Oracle per default fa la no action ,cioè non compie nessuna operazione sulla base di dati. Le altre politiche di reazione usate sono:

-SET NULL: che setta automaticamente a null tutti gli attributi delle tabelle referenziate collegate agli attributi modificati delle tabelle che referenziano le precedenti;

-SET DEFAULT: che setta automaticamente ad un valore di default, scelto da noi, tutti gli attributi delle tabelle referenziate collegate agli attributi modificati delle tabelle che referenziano le precedenti;

-CASCADE: ovvero tutte le modifiche fatte sulle tabelle referenzianti si ripercuotono anche su quelle referenziate.

Il codice SQL utilizzato per aggiungere i vincoli è il seguente:

|  |
| --- |
| **Alter table** AUTISTI **add constraint** FK\_Utente\_Autista **foreign key**(Id) **references** UTENTI(Id) **on delete cascade**;  **Alter table** PASSEGGERI **add constraint** FK\_Utente\_Passeggero **foreign key**(Id) **references** UTENTI(Id) **on delete cascade**;  **Alter table** PRENOTAZIONI **add constraint** FK\_Prenotazione\_Viaggio **foreign key**(Viaggio) **references** VIAGGI(Codice) **on delete cascade**;  **Alter table** PRENOTAZIONI **add constraint** FK\_Prenotazione\_Passeggero **foreign key**(Passeggero) **references** PASSEGGERI(Id) **on delete set null**;  **Alter table** VIAGGI **add constraint** FK\_Viaggio\_Andata **foreign key**(Andata) **references** CITTA(Cap) **on delete cascade**;  **Alter table** VIAGGI **add constraint** FK\_Viaggio\_Ritorno **foreign key**(Ritorno) **references** CITTA(Cap) **on delete set null**;  **Alter table** VIAGGI **add constraint** FK\_Viaggio\_Autista **foreign key**(Autista) **references** AUTISTI(Id) **on delete cascade**;  **Alter table** FEEDBACK **add constraint** FK\_Recensione\_Viaggio **foreign key**(Cod\_Viaggio) **references** VIAGGI(Codice);  **Alter table** FEEDBACK **add constraint** FK\_Mittente\_Utente **foreign key**(Mittente) **references** UTENTI(Id)**on delete set null**;  **Alter table** FEEDBACK **add constraint** FK\_Destinatario\_Utente **foreign key**(Destinatario) **references** UTENTI(Id); |

**4.5 Creazione altri utenti**

Oltre al dba, c’è stato il bisogno di creare altri utenti: essi sostanzialmente sono gli **Utenti**, gli **Autisti** e i **Passeggeri**, ai quali abbiamo fornito anche alcuni privilegi come la possibilità di inserire, cancellare o aggiornare alcune tabelle.

Di seguito vi è riportato il codice:

|  |
| --- |
| **create user** Utente **identified** **by** 1234;  **grant update, insert,select,delete** **on** AUTISTI **to** Utente;  **grant update, insert,select,delete on** PASSEGGERI **to** Utente;  **create user** Autista **identified** **by** 1234;  **grant update,insert,select,delete** **on** VIAGGI **to** Autista;  **grant update,insert,select** **on** FEEDBACK **to** Autista;  **create user** Passeggero **identified** **by** 1234;  **grant update,insert,select,delete on** PRENOTAZIONI **to** Passeggero;  **grant update,insert,select on** FEEDBACK **to** Passeggero; |

**4.6 Popolamento del Database**

Di seguito vi è riportato il popolamento della base dati, ovvero l’inserimenti di alcuni valori di “esempio” utili per verificare il corretto funzionamento della base di dati.

|  |
| --- |
| **insert into** CITTA **values** ('80147','Ponticelli');  **insert into** CITTA **values** ('81023','Messercola');  **insert into** CITTA **values** ('81024','Maddalonii');  **insert into** CITTA **values** ('81055','Santa Maria Capua Vetere');  **insert into** UTENTI **values** (1, 'SPSFNC97D10H892N', 'Esposito', 'Francesco', 3315361700, 'francesco.esposito1004@hotmail.com');  **insert into** UTENTI **values** (2, 'CCINNA97H58E791C', 'Cicia', 'Anna', 3312058159, 'a.cicia@studenti.unina.it');  **insert into** UTENTI **values** (3, 'CMPMCR98A43G596I', 'Compagnone', 'Maria Carla', 3270274119, 'mariaca.compagnone@studenti.unina.it');  **insert into** UTENTI **values** (4, 'DLLSVT97P20E791D', 'Della Torca', 'Salvatore', 3401124142, 'salvatore.dellatorca@studenti.unina.it');  **insert into** AUTISTI (Id, Username, Password, Numero\_polizza, Targa\_auto, Modello\_auto, Carburante\_auto, Numero\_patente, Scadenza\_patente, Animali, Bagagli) **values** (1, 'francesco.esposito', 'bicchiere1', '12345678', 'CF049ZY', 'Fiat 600', 'Benzina', '123456789', '11-AGO-2025', 0,2);  **insert into** AUTISTI (Id, Username, Password, Numero\_polizza, Targa\_auto, Modello\_auto, Carburante\_auto, Numero\_patente, Scadenza\_patente, Animali, Bagagli) **values** (4, 'salvatore.dellatorca', 'salvo97', '74839765', 'RE791PA', 'Chevrolet matiz', 'Gas', '354678376', '20-SET-2026', 1, 2);  **insert into** VIAGGI **values** (1, '2 ore', '12-APR-18', '10:00', 4, '20 euro', 80147, 81023, 'aperto', 1);  **insert into** VIAGGI **values** (2, '40 minuti', '03-MAR-2018', '09:30', 3, '15 euro', 81055, 81023, 'aperto', 4);  **insert into** PASSEGGERI (Id, Username, Pssword) **values** (2, 'anna97', 'ilovebasidati');  **insert into** PASSEGGERI (Id, Username, Pssword) **values** (3, 'carla97', 'ciaoatutti');  **insert into** PRENOTAZIONI **values** (1, 1, 2, 2, 'Aperto');  **insert into** PRENOTAZIONI **values** (2, 2, 3, 'aperto');  **insert into** FEEDBACK **values** (1, 'Il viaggio è stato perfetto, la citta era molto bella e l autista ha guidato bene', 1, 2, 1);  **insert into** FEEDBACK **values** (2, 'La città era molto brutta e il prezzo era troppo elevato, sconsigliato', 2, 3, 4); |

**4.7 Viste**

Una vista è una relazione virtuale, nel senso che le sue tuple non sono effettivamente memorizzate nella base di dati, quanto piuttosto ricavabili attraverso interrogazioni, date da altre tuple presenti nella base di dati. Una vista costituisce dunque una interfaccia da mettere a disposizione di utenti o applicazioni per le interrogazioni: si tratta di dati usati di frequente che possono anche non esistere fisicamente. Abbiamo dunque creato delle viste per rendere più rapito l’accesso ad alcune informazioni a cui si accede sovente.

|  |
| --- |
| *--Vista utile a un passeggero per verificare quante persone hanno prenotato il suo viaggio e le loro generalità.*  **create or replace view** Conta\_prenotazioni **as**  **select count**(Pr.codice) **as** Numero\_Prenotazioni, U.cognome, U.nome  **from** Passeggeri P, Prenotazioni Pr, Utenti U  **where** Pr.Passeggero=P.Id **and** U.Id=P.Id  **group** **by** U.cognome, U.nome;  *--Vista utile per lo più ai passeggeri per vedere le recensioni dei vari viaggi e l’autista che li ha messi a disposizione.*  **create or replace view** Recensione\_Viaggi **as**  **select** U.nome **as** Nome\_Autista, U.cognome **as** Cognome\_Autista, C.Nome **as** Nome\_citta, F.Recensione  **from** Feedback F, Viaggi V, Utenti U, Citta C  where F.Cod\_viaggio=V.Codice **and** F.destinatario=U.Id **and** C.CAP=V.Andata;  *--Vista utile ai passeggeri per vedere i dati relativi ai viaggi disponibili, come il costo o la destinazione.*  **create or replace view** Viaggi\_disponibili **as**  **select** V.Codice **as** Codice\_viaggi\_disponibili, C.Nome **as** Destinazione, V.data\_partenza, V.ora\_partenza, V.contributo\_economico  **from** Viaggi V, Citta C  **where** V.andata=C.cap **and** V.stato='aperto'; |

**4.8 Trigger**

Un trigger è una stored procedure che viene attivata in automatico quando si verifica un particolare evento; per cui rendono un database attivo, e solitamente vengono utilizzati per implementare regole di business, per fare dei controlli sui dati oppure per la reazioni ai vincoli di integrità. Essi si basano sul paradigma di comunicazione ECA, per cui bisogna specificare l’evento che li scatena, una particolare condizione(se c’è) per la quale deve attivarsi, e le azioni che deve eseguire.

Sulla base dell’evento, i trigger si classificano come:

-Trigger DML; che si attivano a valle di un’operazione DML sulla tabella target(cioè insert, delete e update);

-Trigger DDL; che si attivano a valle di poerazioni DDL, come create table o drop table;

-Trigger di sistema; che vengono utilizzate per lo più per il backup e si attivano a valle di operazioni come apertura o chiusura del database.

Sulla base della modalità di esecuzione, i trigger si dividono in:

-Immediati; che si attivano immediatamente prima o dopo un’evento;

-Differiti; che si attivano per un altro evento.

Riportiamo in seguito il codice di trigger DML che abbiamo creato.

|  |
| --- |
| *--Trigger che calcola i posti disponibili n un viaggio dopo che ne è stata fatta una prenotazione.*  **create or replace trigger** Calcola\_Posti  **after insert on** PRENOTAZIONI  **for each row**  **begin**  **update** VIAGGI **set** Posti\_disponibili=Posti\_disponibili-1 **where** Codice=:new.Viaggio;  **end**;  --*Trigger che setta lo stato di un viaggio a “chiuso” se i posti disponibili sono finiti.*  **create or replace trigger** Chiudi\_Viaggio  **after update on** VIAGGI  **for each row**  **begin**  **update** VIAGGI **set** Stato='chiuso' **where** Posti\_disponibili=0;  **end**;  *--Trigger che cambia lo stato di una prenotazione da “aperto” a “chiuso” o viceversa in relazione allo stato del viaggio.*  **create or replace trigger** Cambia\_stato\_Prenotazioni  **after update on** VIAGGI  **for each row**  **begin**  **update** PRENOTAZIONI **set** Stato=:new.stato **where** Viaggio=:new.Codice;  **end**;  --*Trigger che elimina la prenotazione di un passeggero che si è eliminato dalla propria tabella di appartenenza.*  **create or replace trigger** Elimina\_prenotazione  **after delete on** PASSEGGERI  **for each row**  **begin**  **delete from** PRENOTAZIONI **where** Passeggero=:old.Id;  **end**;  --*Trigger che riapre lo stato di un viaggio aggiungendo un posto disponibile nel caso in cui una prenotazione per quel viaggio è eliminata.*  **create or replace trigger** Riapri\_Viaggio  **after delete on** PRENOTAZIONI  **for each row**  **begin**  **update** VIAGGI **set** Posti\_disponibili=Posti\_disponibili+1, Stato='aperto' **where** Codice=:old.Viaggio;  **end**; |

**4.9 Esempi di interrogazioni in SQL**

Nel database le informazioni sono organizzate in una struttura logica che permette di accedere con facilità ad ogni dato. Il modo per accedere a questi dati è la query. La query viene scritta in un linguaggio di interrogazione. Ne esistono decine ed il più famoso prende il nome di SQL. Come tutti i linguaggi, l’SQL ha una sintassi e delle regole. Tramite queste regole è possibile ricercare fra i dati, applicando dei filtri ed ordinando i dati a piacimento. Di seguito sono riportati degli esempi di interrogazione in SQL.

|  |
| --- |
| --*Selezionare il nome delle città in cui porta Francesco.*  **select** C.nome **as** Citta\_in\_cui\_porta\_Francesco  **from** Autisti A, Viaggi V, Citta C, Utenti U  **where** V.autista=A.Id **and** C.CAP=V.Andata **and** A.Id=U.Id **and** U.nome='Francesco';  *--Selezionare il nome di tutte le città in cui non porta Francesco.*  **select** C1.Nome **as** Dove\_non\_porta\_Francesco  **from** Citta C1  **group** by C1.nome  **having** C1.nome **not in**(  **select** C.nome  **from** Autisti A, Viaggi V, Citta C, Utenti U  **where** V.autista=A.Id **and** C.CAP=V.Andata **and** A.Id=U.Id **and** U.nome='Francesco'  **group by** C.Nome);  *--Selezionare il nome del passeggero che ha partecipato al viaggio 1 e la sua recensione.*  **select** U.nome, F.Recensione  **from** Prenotazioni P, Viaggi V, Feedback F, Utenti U  **where** P.viaggio=V.codice **and** F.Cod\_viaggio=V.codice **and** U.Id=P.passeggero **and** V.codice=1;  *--Selezionare la città di destinazione verso la quale il viaggio è il più costoso.*  **select** C.nome **as** Citta\_con\_viaggio\_più\_costoso, V.contributo\_economico  **from** Viaggi V, Citta C  **where** V.andata=C.Cap  **group by** C.Nome, V.contributo\_economico  **having** V.contributo\_economico **>= all**(  **select** contributo\_economico  **from** Viaggi  **group by** contributo\_economico);  --*Selezionare, per ogni utente il cui Id è 3, il nome, il cognome e il numero di prenotazioni.*  **select** U.Id, U.nome, U.cognome, **count(\*) as** Prenotazioni\_di\_Carla  **from** Prenotazioni P, Utenti U, Passeggeri Pa  **where** P.passeggero=Pa.Id **and** U.Id=Pa.Id  **group by** U.Id, U.nome, U.cognome  **having** U.Id=3;  *--Selezionare per ogni passeggero il nome e il numero di prenotazioni effettuate al viaggio numero 4.*  **select count(**P.passeggero**)as** Prenotazioni\_viaggio\_di\_salvo, U.nome  **from** Viaggi V, Prenotazioni P, Passeggeri Pa, Utenti U  **where** V.Autista=4 **and** P.viaggio=V.codice **and** P.passeggero=Pa.Id **and** Pa.Id=U.Id  **group by** U.nome; |