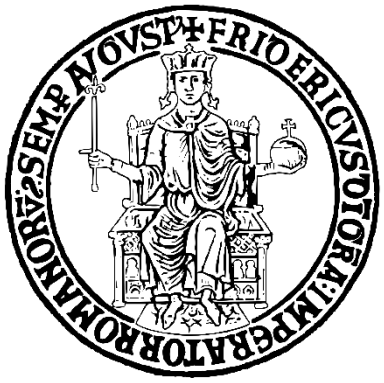


Progetto Basi di Dati

CdL in Informatica

A.A. 2020/2021

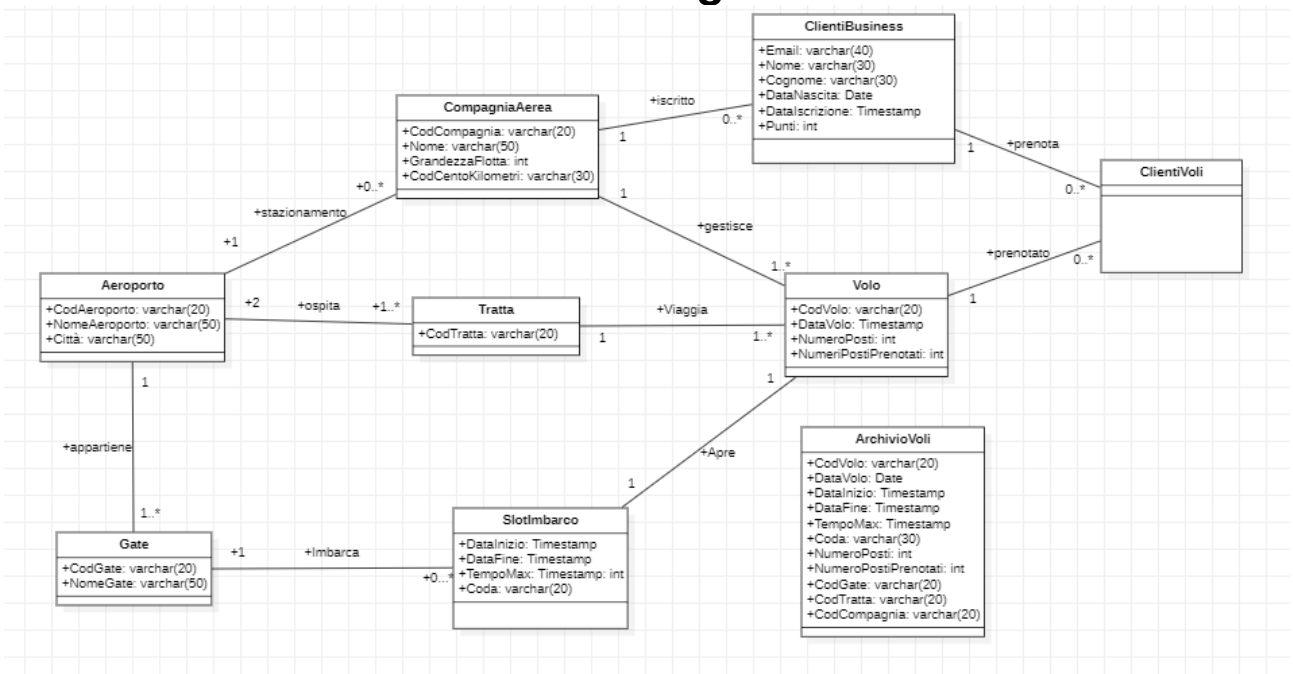
Caccavale Mariano – N86003303
Ciccarelli Mariaelena - N86003212

ID Gruppo: 1

Analisi del problema

Il problema in questione è quello di creare un sistema di gestione di uno scalo aeroportuale che si interfacci con un database che immagazzini le informazioni più importanti riguardo le entità prese in considerazione. Come primo passo infatti, ci siamo chiesti quali sarebbero state le entità chiave del problema, e dopo averne fatto una lista più o meno esaustiva, abbiamo iniziato a pensare come queste entità fossero costituite, consci del fatto che sicuramente sarebbe servito raffinarle anche in fasi più avanzate della progettazione. Sin da subito abbiamo cercato di formalizzare il più possibile, in modo da non lasciare entità, relazioni, e/o attributi non descritti che ci sarebbero potuti servire man mano che il progetto prendeva forma; questo infatti ci avrebbe costretto a riconsiderare la progettazione, cosa che avrebbe sicuramente potuto causare problemi a catena, costringendoci a spendere altro tempo sulla progettazione. Da questa prima fase di progettazione e confronto è risultato il class diagram seguente, che ci ha accompagnato poi per tutta la programmazione fisica della basi di dati.

Class Diagram



Il problema richiedeva la gestione di scali aeroportuali, quindi sicuramente l'entità iniziale doveva essere "Aeroporto". Era inoltre richiesto di poter tenere traccia di compagnia aeree, tratte e gate. Di conseguenza, la loro formalizzazione in classi quali "CompagniaAerea", "Tratta" e "Gate". Tutti gli attributi appartenenti a queste classi sono stati progettati tenendo soprattutto conto delle richieste di ricerca espresse nella traccia.
Per poter realizzare meglio alcune ricerche e agevolare la comprensione dell'intero elaborato abbiamo anche deciso di estrarre dall'entità "Tratta" quella che poi è diventata l'entità "Volo". In questo modo è più facile gestire gli stessi

e anche metterli in relazione con altre entità. Inoltre a posteriori questa scelta si è rivelata vantaggiosa soprattutto per l’implementazione del codice Java.

Definizione Tabelle

CREATE TABLE aeroporto
```
(
    codaeroporto VARCHAR(20) NOT NULL,
    nomeaeroporto VARCHAR(50) NOT NULL,
    "città" VARCHAR(50) NOT NULL,
    CONSTRAINT aeroporto_pkey PRIMARY KEY (codaeroporto),
    CONSTRAINT nome_aer_unico UNIQUE (nomeaeroporto)
)
```

CREATE TABLE gate
```
(
    codgate VARCHAR(20) NOT NULL,
    codaeroporto VARCHAR(20) NOT NULL,
    nomegate CHAR(10) NOT NULL,
    CONSTRAINT gate_pkey PRIMARY KEY (codgate),
    CONSTRAINT fk_aeroporto FOREIGN KEY (codaeroporto)
        REFERENCES aeroporto (codaeroporto) ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT nomegate CHECK (nomegate  ~* '^[A-Z]{1}[0-9]{1}$')
)
```

CREATE TABLE clienti_voli
```
(
    codvolo VARCHAR(20)  NOT NULL,
    email VARCHAR(40) NOT NULL,
    CONSTRAINT fk_clienti FOREIGN KEY (email)
        REFERENCES clientibusiness (email)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT fk_volo FOREIGN KEY (codvolo)
        REFERENCES public.volo (codvolo)
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

CREATE TABLE clientibusiness
```
(
    email VARCHAR(40) NOT NULL,
    nome VARCHAR(30) NOT NULL,
    cognome VARCHAR(30) NOT NULL,
    datanascita DAT NOT NULL,
    dataiscrizione TIMESTAMP WITHOUT TIME ZONE NOT NULL,
```

```
    punti INT NOT NULL,
    codcentokilometri character varying(30) NOT NULL,
    CONSTRAINT clientibusiness_pkey PRIMARY KEY (email),
    CONSTRAINT fk_aeroporto FOREIGN KEY (codcentokilometri)
        REFERENCES public.compagniaaerea (codcentokilometri)
        ON UPDATE CASCADE
        ON DELETE SET NULL,
    CONSTRAINT datanascita_corretta CHECK (datanascita <= now()),
    CONSTRAINT email_regex CHECK (email ~ '^[A-Za-z0-9.-_]+@[a-z]+\.[a-
z]{2,3}$')
)
```

CREATE TABLE compagniaaerea

```
(
    codcompagnia VARCHAR(20) NOT NULL,
    nomecompagnia VARCHAR (50) NOT NULL,
    grandezzaflotta INT NOT NULL,
    aeroportoappartenenza VARCHAR(20),
    codcentokilometri character varying(30) NOT NULL,
    CONSTRAINT compagniaaerea_pkey PRIMARY KEY (codcompagnia),
    CONSTRAINT codcentokilometri_unique UNIQUE (codcentokilometri),
    CONSTRAINT nomeunico UNIQUE (nomecompagnia),
    CONSTRAINT fk_aeroportoappartenenza FOREIGN KEY
(aeroportoappartenenza)
        REFERENCES aeroporto (codaeroporto)
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT nomevuoto CHECK (nomecompagnia <> ‘’),
    CONSTRAINT compagniaaerea_grandezzaflotta_check CHECK
(grandezzaflotta > 0 AND grandezzaflotta < 500)
)
```

CREATE TABLE slotimbarco

```
(
    codvolo VARCHAR(50),
    codgate VARCHAR(50),
    datainizio TIME WITHOUT TIME ZONE NOT NULL NOT NULL,
    datafine TIME WITHOUT TIME ZONE NOT NULL,
    tempomax TIME WITHOUT TIME ZONE NOT NULL,
    coda VARCHAR(20),
    CONSTRAINT imbarcounico UNIQUE (codvolo, codgate, coda),
    CONSTRAINT fk_gate FOREIGN KEY (codgate)
        REFERENCES gate (codgate) ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT fk_volo FOREIGN KEY (codvolo)
```

```
        REFERENCES volo (codvolo) ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT oracorretta CHECK (datainizio < datafine),
    CONSTRAINT tempomax_check CHECK (tempomax > datafine),
    CONSTRAINT slotimbarco_coda_check CHECK (coda = 'Famiglia' OR
coda= 'Business' OR coda= 'Diversamente abili' OR coda= 'Priority' OR
coda= 'Economy'OR coda= 'Prima Classe')
)
```

CREATE TABLE tratta

```
(
    codtratta VARCHAR(20) NOT NULL,
    aeroportopartenza VARCHAR(20) NOT NULL,
    aeroportoarrivo VARCHAR(20) NOT NULL,
    CONSTRAINT tratta_pkey PRIMARY KEY (codtratta),
    CONSTRAINT trattacorretta UNIQUE (aeroportopartenza,
aeroportoarrivo),
    CONSTRAINT fk_aeroportoarrivo FOREIGN KEY (aeroportoarrivo)
        REFERENCES aeroporto (codaeroporto) ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT fk_aeroportopartenza FOREIGN KEY (aeroportopartenza)
        REFERENCES aeroporto (codaeroporto) ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT aeroportidiversi CHECK (aeroportoarrivo <>
aeroportopartenza)
)
```

CREATE TABLE volo

```
(
    codvolo VARCHAR(20) NOT NULL,
    datavolo TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    numeroposti INT NOT NULL,
    numeropostiprenotati INT,
    codtratta VARCHAR(20) NOT NULL,
    codcompagnia VARCHAR(20) NOT NULL,
    CONSTRAINT volo_pkey PRIMARY KEY (codvolo),
    CONSTRAINT volounico UNIQUE (codvolo, codcompagnia, codtratta,
datavolo),
    CONSTRAINT fk_compagnia FOREIGN KEY (codcompagnia)
        REFERENCES compagniaaerea (codcompagnia) ON UPDATE NO
ACTION
        ON DELETE CASCADE,
    CONSTRAINT fk_tratta FOREIGN KEY (codtratta)
        REFERENCES tratta (codtratta) ON UPDATE NO ACTION
        ON DELETE CASCADE,
```

```
    CONSTRAINT volo_numeroposti_check CHECK (numeroposti < 500 AND
numeroposti >= 50),
    CONSTRAINT numeropostiprenotaticorretti CHECK (numeropostiprenotati
<= numeroposti)
)
```

CREATE TABLE archiviovoli

```
(
    codvolo VARCHAR(20) NOT NULL,
    datavolo DATE NOT NULL,
    datainizio TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    datafine TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    tempomax TIMESTAMP WITHOUT TIME ZONE NOT NULL,
    coda VARCHAR(30) NOT NULL,
    numeroposti INT NOT NULL,
    numeropostiprenotati INT NOT NULL,
    codgate VARCHAR(20) NOT NULL,
    codtratta VARCHAR(20) NOT NULL,
    codcompagnia VARCHAR(20) NOT NULL,
    CONSTRAINT archiviovoli_pkey PRIMARY KEY (codvolo),
    CONSTRAINT numeropostiprenotaticorretti CHECK (numeropostiprenotati
<= numeroposti),
    CONSTRAINT slotimbarco_coda_check CHECK (coda  = 'Famiglia' OR
coda = 'Business' OR coda = 'Diversamente abili' OR coda = 'Priority' OR
coda = 'Economy' OR coda = 'Prima Classe'),
    CONSTRAINT volo_numeroposti_check CHECK (numeroposti < 500 AND
numeroposti >= 50)
)
```

Dizionario delle classi

Classe	Descrizione	Attributi
Aeroporto	Entità che conserva le informazioni degli aeroporti inseriti nel sistema.	**CodAeroporto(String):** codice identificativo dell'aeroporto. **NomeAeroporto(String):** nome dell'aeroporto. **Città(String):** città in cui l'aeroporto è situato.
CompagniaAerea	Entità che conserva le informazioni delle compagnie aeree inserite nel sistema.	**CodCompagnia(String):** codice identificativo della compagnia. **NomeAeroporto(String):** nome della compagnia commerciale. **GrandezzaFlotta(Integrer):** intero numerico che indica quanto grande sia la flotta associata alla compagnia. **AeroportoAppartenenza(String):** chiave esterna che identifica l'aeroporto dove ha sede la compagnia.
Tratta	Entità che conserva le informazioni sulle tratte inserite nel sistema.	**CodTratta(String):** codice identificativo della tratta. **AeroportoPartenza(String):** chiave esterna che indicia l'aeroporto da cui ha inizio la tratta. **AeroportoArrivo(String):** chiave esterna che indicia l'aeroporto in cui finisce la tratta.
Gate	Entità che conserva le informazioni sui gate inseriti nel sistema.	**CodGate(String):** codice identificativo del gate. **NomeGate(String):** nome del gate. **CodAeroporto(String):** chiave esterna che indica in quale aeroporto si trova il gate.

Volo	Entità che conserva le informazioni sui voli inseriti nel sistema.	**CodVolo(String):** codice identificativo del volo. **DataVolo(Datetime):** data assegnata al volo. **NumeroPosti(Integer)**: numero posti massimo per il volo. **NumeroPostiPrenotati(Integer):** numero posti attualemente prenotati per il volo. **CodTratta(String):** chiave esterna che indica su che tratta si svolge il volo. **CodCompagnia(String):** chiave esterna che indica da che compagnia è gestito quel volo.
ClientiVoli	Entità che conserva le informazioni dei clienti che hanno acquistato un biglietto per uno specifico volo	**CodVolo:** chiave esterna che identifica il volo del cliente. **Email:** chiave esterna che identifica il cliente.
ClientiBusiness	Entità che conserva le informazioni di tutti i clienti business iscritti ad una compagnia	**Email:** chiave primaria che identifica il cliente. **Nome:** attributo che identifica il nome del cliente. **Cognome:** attributo che identifica il cognome del cliente. **DataNascita:** attributo che identifica la data di nascita del cliente. **DataIscrizione:** attributo che identifica la data di iscrizione del cliente. Serve per assegnare punti in modo corretto (i voli in ritardo del cliente devono essere a posteriori l'iscrizione). **Punti:** attributo che identifica il totale dei punti accumulati da un cliente. **CodCentoKilometri:** chiave

		esterna che identifica la compagnia associata al cliente.
SlotImbarco	Entità che conserva le informazioni riguardanti gli imbarchi inseriti nel sistema.	**CodVolo(String):** codice identificativo del volo a cui appartiene l'imbarco. **CodGate(String):** codice identificativo del gate assegnato all'imbarco. **Data(date):** data d'imbarco. **OraInizio(Time without time zone):** ora di inizio dell'imbarco. **OraFine(Time without time zone):** ora di chiusura dell'imbarco. **TempoMax(Integer):** ora/e di apertura max dell'imbarco. **Coda(String):** tipo di oda associata a questo imbarco.
ArchivioVoli	Mappa lo storico dei voli partiti dall'aeroporto.	**Codvolo:** codice identificativo del volo partito. **Datavolo:** data del volo. **Orainizio:** ora di apertura del gate del volo. **Orafine:** ora di chiusura del gate del volo. **Tempomax:** tempo massimo di apertura del gate. **Coda:** tipo di coda associata al gate del volo. **Numeroposti:** numero posti massimi del volo. **Numeropostiprenotati:** numero effettivamente prenotati del volo. **codgate:** codice del gate di imbarco del volo. **Codtratta:** codice della tratta su cui ha viaggiato il volo. **Codcompagnia:** codice della compagnia che ha eseguito il volo.

Dizionario delle Associazioni

Nome	Descrizione	Classi Coinvolte
Stazionamento	Mappa lo stazionamento di una compagnia in un aeroporto.	**Aeroporto[0...*]:** un aeroporto può ospitare da 0 a * compagnie aeree. **CompagniaAerea[1]:** una compagnia aerea deve stazionare in uno e un solo aeroporto.
Ospita	Mappa l'appartenenza di una tratta a due aeroporti.	**Aeroporto[1...*]:** un aeroporto può appartenere a tante tratte, ma minimo una. **Tratta[2]:** una tratta è legata a due e soli due aeroporti; quello di partenza e quello di arrivo.
Appartiene	Mappa l'appartenenza di un gate ad un aeroporto.	**Aeroporto[1...*]:** un aeroporto può ospitare diversi gate, ma minimo uno. **Gate[1]:** un gate può appartenere ad uno e solo un aeroporto.
Gestisce	Mappa la gestione dei voli da parte di una compagnia aerea.	**CompagniaAerea[1...*]:** una compagnia può gestire molti voli, ma minimo uno per essere attiva. **Volo[1]:** un volo può essere gestito da una sola compagnia.
Viaggia	Mappa la tratte su cui viaggiano i voli.	**Tratta[1...*]:** una tratta può ospitare molti voli. **Volo[1]:** un volo può viaggiare su una sola tratta alla volta.
Imbarca	Mappa l'imbarco di uno slotImbarco rispetto ad un gate	**Gate[0...*]:** un gate può ospitare diversi imbarchi. **SlotImbarco[1]:** un imbarco può essere effettuato tramite un solo gate.
Apre	Mappa l'apertura di uno slotImbarco per un volo.	**SlotImbarco[1]:** uno slot imbarco apre per uno e un solo volo. **Volo[1]:** un volo permette di aprire un solo slotImbarco.
Iscritto	Mappa l'iscrizione di un cliente al piano business di una compagnia aerea.	**CompagniaAerea[0...*]:** una compagnia aerea può avere da zero a molti clienti business iscritti ad essa. **ClientiBusiness[1]:** un cliente business può avere una

		sottoscrizione ad un solo piano business di una compagnia.
Prenota	Mappa le prenotazioni dei voli di un cliente.	**ClientiBusiness[0...*]:** un cliente business può prenotare più voli. **ClientiVoli[1]:** una singolo record(che descrive quello che è un biglietto) può essere a nome di un solo cliente.
Prenotato	Mappa i passeggeri business di un determinato volo.	**Volo[0...*]:** un volo può avere da 0 a molti clienti business che viaggiano su di esso. **ClientiVoli[1]:** un singolo record(che descrive quello che logicamente è un biglietto) può essere valido per un solo volo.

Vincoli

I vincoli di totalità per quanto riguarda le chiavi primarie sono omesse per brevità. I vincoli di totalità sono formalizzati nella sezione dedicata alla definizione delle tabelle SQL. Le chiavi esterne sono formalizzate nella sezione dedicata alla definizione dello schema logico.

Classe di appartenenza	**Descrizione**
Aeroporto	**NomeAerUnico:** (NomeAeroporto, Città) deve essere UNIQUE, perché ci possono essere due aeroporti con lo stesso nome, ma devono essere in città diverse.
CompagniaAerea	**NomeUnico:** Il Nome della CompagniaAerea deve essere di tipo UNIQUE, non possono esistere due compagnie con lo stesso nome **NomeVuoto**: Il nome della compagnia non può essere vuoto. **CodiceCentoKilometri_Unique:** il CodiceCentoKilometri di una compagnia aerea deve essere unico. **CompagniaAerea_GrandezzaFlotta_Check**: La grandezza della flotta può assumere qualsiasi valore INTEGER POSITIVO compreso tra 50 e 500.
Tratta	**TrattaCorretta: l**a coppia (aeroportopartenza, aeroportoarrivo) deve essere unica per garantire la correttezza della tratta. **AeroportiDiversi**: AeroportoPartenza e AeroportoArrivo devono essere diversi. **Trattacorretta**: La coppia (AeroportoPartenza, AeroportoArrivo) deve essere unica.
Volo	**VoloUnico:** la quadrupla (codvolo, codcompagnia, codtratta, datavolo) deve essere unica per garantire l'unicità di un volo. **Volo_NumeroPosti_Check:** NumeroPosti è di default a 50 minimo, e massimo 500. **NumeroPostiPrenotatiCorretto:** NumeroPostiPrenotati può assumere valori compresi tra 0 e 500.

	DataCorretta: DataVolo deve essere maggiore o uguale a Sysdate.
Gate	**NomeGate**: Il nome del gate deve essere composto da una lettera maiuscola e un numero. **NomeUnique:** il nome di un gate deve essere unico all'interno di un areoporto.
SlotImbarco	**ImbarcoUnico:** La tripla (codvolo, codgate, coda) deve essere unica per garantire l'apertura di un solo gate per un determinato volo con associata una determinata coda. **ValidificaData()**: Data e Ora devono essere compatibili con l'ora del volo (es. se il volo è previsto per il 25/12/2020 alle ore 12:00, allora Data in SlotImbarco ***deve*** essere 25/12/2020, e l'ora deve contenere le 12, quindi, ad esempio, devono essere 11:40 OraInizio e 12:20 OraFine). **OraCorretta**: OraFine non può essere minore di OraInizio; non possiamo chiudere un gate prima di averlo aperto. **SlotImbarco_Coda_Check**: vincolo di dominio. Il valore della coda può essere solo uno dei sequenti: Famiglia, Business, Diversamente Abili, Priority, Economy o Prima Classe. **TempoMax_Check:** la data di tempo massimo di apertura di un gate deve essere maggiore dell'effettiva data di apertura del gate.
ArchivioVoli	**SlotImbarco_Coda_Check**: vincolo di dominio. Il valore della coda può essere solo uno dei sequenti: Famiglia, Business, Diversamente Abili, Priority, Economy o Prima Classe. **Volo_NumeroPosti_Check:** NumeroPosti è di default a 50 minimo, e massimo 500. **NumeroPostiPrenotatiCorretto:** NumeroPostiPrenotati può assumere valori compresi tra 0 e 500.
ClientiBusiness	**Email_Regex:** l'email inserita deve rispettare la corretta formattazione di un'email. **Data_Nascita_Corretta:** la data di nascita inserita deve essere antecedente a quella odierna, per essere almeno valida.

ClientiVoli	Nessun vincolo.

Procedure

Le procedure utilizzate in questo database sono 3. Due di esse servono per automatizzare alcune procedure, mentre la terza serve a validifcare un dato una volta che viene inserito, essendoci in postreSQL l'impossibilità di aggiungere una query in un **constraint.**

- **ArchiviaVolo():**

```
CREATE FUNCTION archiviavolo()
    RETURNS trigger
    LANGUAGE 'plpgsql'
AS $BODY$
begin

        insert into archiviovoli
        (select NEW.codvolo, v.datavolo, NEW.datainizio, NEW.datafine,
        NEW.tempomax, NEW.coda, v.numeroposti,
        v.numeropostiprenotati, NEW.codgate ,v.codtratta,
        v.codcompagnia
         from volo as v
         where v.codvolo = NEW.codvolo);

        delete from volo as v
        where v.codvolo = NEW.codvolo;

        delete from slotimbarco as si
        where si.codvolo = NEW.codvolo;

RETURN NEW;
end
$BODY$;
```

Questa funzione si occupa di archiviare un volo, passando tutte le informazioni utili dalle tabelle **Volo**, **Gate** e **SlotImbarco** nella tabella **ArchivioVoli.** Inoltre, si occupato di cancellare i voli appena salvati dalle tabelle sopra riportate, in modo da non avere ridondanza.
Il trigger che la esegue è posto dopo l'update nella tabella **SlotImbarco**, esattamente dopo modificata della DataFine, e solo quando questa passa da un valore NULL ad uno NOT NULL. Questo perché quando uno slot imbarco viene chiuso, vuol dire che il volo è effettivamente partito; quindi, se il volo è partito, esso può essere archiviato.

- **AssegnaPunti():**

```
CREATE FUNCTION assegnapunti()
    RETURNS trigger
    LANGUAGE 'plpgsql'
AS $BODY$
begin

        if (new.datafine > new.tempomax) then
                update clientibusiness
                set punti = punti + 10
                where email in (select email from clienti_voli where codvolo =
                new.codvolo) AND
                                codcentokilometri = ('CK'|| new.codcompagnia);
        end if;

RETURN NEW;
end
$BODY$;
```

Questa funzione si occupa, in caso di ritardo di un volo, di assegnare i punti che spettano solo ai clienti business. Il Trigger che la fa scattare è posto dopo l'inserimento in ArchivioVoli. Questo perché una volta che un volo è partito(e di conseguenza archiviato), se c'è stato un ritardo, devono essere assegnati i punti ai clienti business che hanno sofferto di quel ritardo. La function inoltre si appoggia alla tabella Clienti_Voli per assegnare i punti soltanto ai clienti business che avevano effettivamente prenotato un biglietto per il volo appena partito.

- **ValidificaData():**

```
CREATE FUNCTION validificadata()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
begin
if (NEW.datavolo < current_timestamp) then

        raise exception 'Non è possibile inserire un volo precedente ad ora';

end if;

RETURN NEW;
end
$BODY$;
```

Questa è la funzione creata per ovviare all'impossibilità di inserire sottointerrogazioni all'interno di un **constraint.** La procedura non fa altro che controllare se la data inserita è effettivamente valida per la creazione di un volo. Nel caso non lo sia, si lancia un'eccezione per impedire l'inserimento errato e viene fornito un breve messaggio descrittivo per permettere di capire quale sia l'errore.

- **Voli_Corretti():**

```
CREATE FUNCTION Voli_Corretti()
        RETURNS trigger
        LANGUAGE 'plpgsql'
AS $BODY$
begin

        if (select count(v.codvolo) from volo as v
        where v.compagniaappartenenza = new.codcompagnia) > new.gran-
        dezzaflotta then
        raise exception 'Impossibile modificare la grandezza flotta. Ci sono
        troppi volo, alcuni rimarrebbero scoperti';
        end if;
RETURN NEW;
end
$BODY$;
```

Questa funzione si occupa di controllare, in caso di modifica della grandezza flotta di una compagnia, quanti voli già programmati ci sono per quella compagnia. In caso i due valori fossero incoerenti (si cerca, ad esempio, di ridurre la flotta inserendo un valore minore ai voli programmati), blocca l’update.

Viste

CREATE gate_view AS(

```
 SELECT DISTINCT g.codgate,
    g.nomegate,
    g.codaeroporto,
    si.datainizio,
    si.datafine,
    si.tempomax,
    si.codvolo
   FROM (gate g
     JOIN slotimbarco si ON ((g.codgate) = (si.codgate)));
```

)

Questa vista viene creata per avere un accesso più diretto alle informazioni riguardanti gate, voli e slotimbarco.

Schema Logico

Aeroporto(CodAereoporto, NomeAeroporto, Città)

CompagniaAerea(CodCompagnia, NomeCompagnia, GrandezzaFlotta, CodiceCentoKilometri, AeroportoAppartenenza)

*Chiavi esterne: AeroportoAppartenenza →**Aeroporto(CodAeroporto)***

Tratta(CodTratta, AeroportoPartenza, AeroportoArrivo)

*Chiavi esterne: AeroportoPartenza →**Aeroporto(CodAeroporto)***
*AeroportoArrivo →**Aeroporto(CodAeroporto)***

Volo(CodVolo, DataVolo, NumeroPosti, NumeroPostiDisponibili, CodTratta, CodCompagnia)

*Chiavi esterne: CodTratta → **Tratta(CodTratta)***
*CodCompagnia → **CompagniaAerea(CodCompagnia)***

Gate(CodGate, AeroportoGate)

*Chiavi esterne: AeroportoGate →**Aeroporto(CodAeroporto)***

SlotImbarco(CodVolo, CodGate, Data, OraInizio, OraFine, TempoMax, Coda)

*Chiavi esterne: CodVolo →**Volo(CodVolo)***
*CodGate →**Gate(CodGate)***

ArchivioVoli(CodVolo, DataVolo, DataInizio, DataFine, TempoMax, Coda, NumeroPosti, NumeroPostiPrenotati, CodGate, CodTratta, CodCompagnia)

ClientiBusiness(Email, Nome, Cognome, DataNascita, DataIscrizione, Punti, CodCentoKilometri)

Chiavi esterne:
*CodCentoKilometri→**CompagniaAerea(CodCentroKilomtri)***

ClientiVoli(CodVolo, Email)

*Chiavi esterne: CodVolo →**Volo(CodVolo)***
*Email →**ClientiBusiness(Email)***