Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Laurea in Informatica

Consegna G4 - Sistema informativo per la gestione dell'emissione di biglietti aerei

Relazione finale del progetto di Basi di Dati

Candidato:
Giovanni Liboni
Matricola VR363021

Anno Accademico 2013-2014

Contents

Part I

Progetto della basi di dati

0.1 Progetto concettuale

0.2 Progetto logico

0.3 Popolamento della base di dati

Il popolamento della base di dati avviene tramite alcuni script per creare le tabelle, caricare le funzioni e i trigger e per inserire alcune tuple di esempio.

```
 CREATE  TABLE tratta(
        partenza        VARCHAR(100) NOT NULL,
        arrivo          VARCHAR(100) NOT NULL,
        durata          INTEGER NOT NULL,
        distanza        NUMERIC(10,1) NOT NULL DEFAULT '0.0',
        PRIMARY KEY ( partenza, arrivo ),
        UNIQUE ( partenza, arrivo )
);
CREATE TABLE volo(
        codicevolo      VARCHAR(10) NOT NULL,
        partenza        VARCHAR(100) NOT NULL,
        arrivo          VARCHAR(100) NOT NULL,
        datapartenza    DATE NOT NULL,
        orapartenza     TIME NOT NULL,
        tipoaereo       VARCHAR(50) NOT NULL ,
        capienza        INTEGER NOT NULL,
        PRIMARY KEY( codicevolo ),
        UNIQUE ( codicevolo ),
        FOREIGN KEY( partenza, arrivo )
                        REFERENCES tratta( partenza, arrivo )
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
CREATE TABLE passeggero(
        login           VARCHAR(100) NOT NULL,
        password        VARCHAR(100) NOT NULL,
        nazionalita     VARCHAR(100) NOT  NULL,
        nome            VARCHAR(100) NOT NULL,
        cognome         VARCHAR(100) NOT NULL,
        documento       VARCHAR(50) NOT NULL,
        picture         BYTEA,

        numvoli         INTEGER DEFAULT 0,
        miglia          FLOAT DEFAULT 0,
        tessera         BOOLEAN DEFAULT FALSE,
        PRIMARY KEY( documento ),
        UNIQUE ( login )
);
CREATE TABLE prenotazione(
        id              SERIAL,
        codicevolo      VARCHAR(10) NOT NULL
```

```
                        REFERENCES volo( codicevolo )
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
        documento       VARCHAR(50) NOT NULL
                        REFERENCES passeggero( documento )
                        ON UPDATE CASCADE
                        ON DELETE CASCADE,
        datarichiesta   DATE,
        orarichiesta    TIME,
        PRIMARY KEY( id ),
        UNIQUE ( codicevolo, documento )
);
CREATE TABLE biglietto(
        id              SERIAL,
        codicevolo      VARCHAR(10) NOT NULL
                        REFERENCES volo( codicevolo )
                        ON UPDATE CASCADE
                        ON DELETE CASCADE,
        documento       VARCHAR(50) NOT NULL
                        REFERENCES passeggero( documento )
                        ON UPDATE CASCADE
                        ON DELETE CASCADE,
        id_prenotazione INTEGER NOT NULL
                                REFERENCES prenotazione( id )
                                ON UPDATE CASCADE
                                ON DELETE CASCADE,
        dataemissione   DATE NOT NULL,
        prezzo          NUMERIC(10,2) NOT NULL,

        PRIMARY KEY( codicevolo, documento ),
        UNIQUE ( codicevolo, documento, id )
);

CREATE TABLE posto(
        lettera         CHAR NOT NULL,
        numero          INTEGER DEFAULT 0,
        PRIMARY KEY ( lettera, numero ),
        UNIQUE ( lettera, numero )
);
CREATE TABLE imbarco(
        lettera         CHAR NOT NULL,
        numero          INTEGER DEFAULT 0,
        imbarcato       BOOLEAN DEFAULT FALSE,
        -- PRIMARY KEY DEL BIGLIETTO
        codicevolo      VARCHAR(10) NOT NULL,
        documento       VARCHAR(50) NOT NULL,
        -- IDENTIFICATO DAL BIGLIETTO E DAL POSTO
        PRIMARY KEY ( codicevolo, documento, lettera, numero),

        FOREIGN KEY( lettera, numero )
```

```
                        REFERENCES posto( lettera,numero )
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY( codicevolo, documento )
                        REFERENCES biglietto( codicevolo,documento )
                ON UPDATE CASCADE
                ON DELETE CASCADE
);
CREATE OR REPLACE FUNCTION update_time_richiesta()
RETURNS TRIGGER AS '
BEGIN
        IF NEW.datarichiesta IS NULL THEN
                NEW.datarichiesta := current_date;
        END IF;
        IF NEW.orarichiesta IS NULL THEN
                NEW.orarichiesta := current_time;
    END IF;
        RETURN NEW;
END' LANGUAGE 'plpgsql';



CREATE OR REPLACE FUNCTION update_time_biglietto()
RETURNS TRIGGER AS '
BEGIN
        IF NEW.dataemissione IS NULL THEN
                NEW.dataemissione := current_date;
        END IF;
        RETURN NEW;
END' LANGUAGE 'plpgsql';



CREATE OR REPLACE FUNCTION update_numvoli()
RETURNS TRIGGER AS '
BEGIN
        IF NEW.dataemissione IS NULL THEN
                NEW.dataemissione := current_date;
        END IF;
        RETURN NEW;
END' LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION update_tessera() RETURNS trigger AS $$
BEGIN
  UPDATE passeggero
  SET numvoli = (SELECT count(*)
                  FROM volo JOIN biglietto ON biglietto.codicevolo = volo.codic
                            JOIN tratta ON (volo.partenza = tratta.partenza
                                            AND volo.arrivo = tratta.arrivo )
                 WHERE passeggero.tessera=true
```

```
                    AND passeggero.documento = biglietto.documento
                GROUP BY passeggero.documento ),
      miglia = (SELECT sum(distanza)
                FROM volo JOIN biglietto ON biglietto.codicevolo = volo.codicev
                          JOIN tratta ON (volo.partenza = tratta.partenza AND v
                WHERE passeggero.tessera=true AND passeggero.documento = biglie
                GROUP BY passeggero.documento )
  FROM biglietto
  WHERE biglietto.dataemissione >= ( SELECT date('now') - interval '3 year')
                                    AND NEW.documento = passeggero.documento;
  RETURN NEW;

END;
$$ LANGUAGE 'plpgsql';

CREATE TRIGGER update_time_richiesta
BEFORE INSERT ON prenotazione
        FOR EACH ROW EXECUTE PROCEDURE update_time_richiesta();

CREATE TRIGGER update_tessera
AFTER INSERT OR DELETE ON biglietto
    FOR EACH ROW EXECUTE PROCEDURE update_tessera();

CREATE TRIGGER update_time_biglietto
BEFORE INSERT ON biglietto
        FOR EACH ROW EXECUTE PROCEDURE update_time_biglietto();
```

Part II

Progetto del sito web

0.4 Progettazione logica

Si riportano di seguiti gli schemi di pagina seguiti durante la creazione del sito.

0.5 Struttura dell'applicazione web

Il progetto è segue le regole del modello servlet-centric 2.
I moduli in cui è suddiviso l'applicativo sono:

- Model - Comprende la classe DBMS.java e i Java Data Beans. I beans sono contenuti nel package *bean*, mentre la classe DBMS.java è contenuta nel package *database*;
- Control - Comprende le servlet main.java, picture.java e ajax.java. Ogni servlet ha un compito specifico: ... ;
- View - Comprende tutte le JSPs, i javascript e i css;

Part III

Scelte progettuali

0.6 Hibernate

0.7 AJAX

0.8 Session