

**Ahmed mohamed nagy abouattia**

**G321210565**

**DATABASES MANGEMENT SYSTEMS**

Hello everyone, my name is Ahmed, and this is my bata base's project. I will try to give you a small idea about it . . . I try to make a system for airport and the number of my tables is 22 tables and they are like Crow's foot . . .

صورة تحتوي على نص, لقطة شاشة, رسم بياني, خطة

تم إنشاء الوصف تلقائياً

And they are as **textual representation of the relational schema**:

* person(**personId: int,** name: varchar(25), surname: varchar(25), addressId: int, passenger: Boolean, employee: Boolean, tel no: int)
* Address(**adressId**: **int** , countryId: int, cityId: int)
* Country(**countryId: int,** countryName: varchar(15))
* City(**cityId: int,** cityName: varchar(15))
* Passengers(**passengerId: int**, ticket:Id)
* Employee(**employeeId: int**, startedDate:date, salary: money, kind:CHAR)
* Passport Controller(**controllerId: int,** gate: int)
* Manager(**ManagerId : int**)
* Pilot(**PilotId: int** , airplaneId: int )
* Gates(**gatesId: int**, gateName:varchar(15))
* Airplane(**airplaneId: int** , pilotId: int , passengers: int [ ] , gate: int)
* Ticket(**ticketId:int** , airplaneId:int , baggage: int , date: DATE, passengerId: int, side: smallint, gateId: int , planId: int)
* Baggage(**baggageId: int**, passengerId: int , weight: float)
* Nowplans(**NowplansId: int** , startFrom:varchar(15), goTo: varchar(15), when: time, airplaneId: int , station1: int, station2: int, kind : char, travelTime : int )
* Stations(**StationId: int**, stationName : varchar (15))
* VIP(**vipId: int,** balance: money, maxBaggageNo: samllint, maxWeight: foat, freePlaces: smallint)
* Normal Plan(**normalPlanId: int** ,balance : money, maxBaggageNo: samllint, maxWeight: foat, freePlaces: smallint)
* Booking(**bookingId: int**, planId: int, gateId: int, ticketId: int )
* Weather(**weatherId: int,** date: date, weather: varchar(15), degree: int)

And now I tell you about some of the **Business Rules : -**

The first used the inheritance at 3 places between person, passenger and employee.

And the second used is between employee, manager, Passport Controller and pilon.

And the 3rd used is between nowplans, vip and normal plan.

* The person can be a passenger and employee at same time, but the employee can't be a pilot and manager at same time.
* Every passenger can have a lot of baggage, but any baggage has one owner only.
* Every passenger can have a lot of tickets at the same time, but any ticket must be for one passenger only.
* Every baggage must be at one ticket, but one ticket can include a lot of baggage.
* Every ticket must include one gate
* Every passport controller must be at one gate, but one gate can have a lot of passport controllers.
* Every pilot must drive one airplane, but in airplane there can be a lot of pilots.
* Every airplane must fight from one gate, but a lot of airplanes can be fighting from the same gate.
* NowPlans is the plane you ready can take right now but the ticket is after taken (payment).
* The stations are like Transistor and they're like stop stations for airplane.

About my **triggers and functions**, I use 4 triggers and 4 functions . . .

The first function is **calculate\_ticket\_price\_vip** and this calculates the price of tickets to vip class and its take parameters time of travel and return balance of the ticket.

The second one is the same thing, but it's for normal class.

The 3rd one is **personAdd** this function I use it to add random data to person table, take one parameters ant return void.

The 4th one is for deleting all persons form person table.

The triggers is like the next 1st one is for when updating the parson table save the old data at new table "person\_name\_update".

The 2nd is for if the ticket cancellated adding the old values to new table "ticket\_canceletion".

The 3rd for when to change the manager or add saving the last one's data

The 4th for when taken a ticket that automatically adding to booking table

Finally, this is my databases codes using postgresql

CREATE TABLE person(

    "personId" SERIAL,

    "name" VARCHAR(25),

    "surname" VARCHAR(25),

    "addressId" int,

    "passenger" BOOLEAN,

    "employee" BOOLEAN,

    "Tel no" int,

     --yas -function

    CONSTRAINT "personPK" PRIMARY KEY ("personId"),

    CONSTRAINT "personAddressFK" FOREIGN KEY ("addressId")

        REFERENCES adress("adressId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE Table passengers(

    "passengerId" int,

    -- secil

    "ticket" int,

    CONSTRAINT "passengersPK" PRIMARY KEY ("passengerId"),

    CONSTRAINT "passengerPersonInheritanceFK" FOREIGN KEY ("passengerId")

        REFERENCES person("personId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "passengersTicketFK" FOREIGN KEY ("ticket")

    REFERENCES ticket("ticketId")

    ON DELETE CASCADE

    ON UPDATE CASCADE

);

CREATE TABLE employee(

    "employeeId" int,

    "startedData" date,

    "salary" money,

    "kind" char,

    CONSTRAINT "empPK" PRIMARY KEY ("employeeId"),

    CONSTRAINT "employeePersonInheritanceFK" FOREIGN KEY ("employeeId")

        REFERENCES person("personId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE passport\_controller(

    "controllerId" int,

    "gate" int,

    CONSTRAINT "passControllerPK" PRIMARY KEY ("controllerId"),

    CONSTRAINT "passport\_controllerEmployeeInheritanceFK" FOREIGN KEY ("controllerId")

        REFERENCES employee("employeeId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "passport\_controllerGatesFK" FOREIGN KEY ("gate")

        REFERENCES gates("gatesId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE manager(

    "managerId" int,

    CONSTRAINT "managerPK" PRIMARY KEY ("managerId"),

    CONSTRAINT "managerEmployeeInheritanceFK" FOREIGN KEY ("managerId")

        REFERENCES employee("employeeId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE pilot(

    "pilotId" int,

    "airplaneId" int,

    CONSTRAINT "pilotPK" PRIMARY KEY ("pilotId"),

    CONSTRAINT "pilotEmployeeInheritanceFK" FOREIGN KEY ("pilotId")

    REFERENCES employee("employeeId")

    ON DELETE CASCADE

    ON UPDATE CASCADE,

    CONSTRAINT "pilotAirplaneFK" FOREIGN KEY ("airplaneId")

        REFERENCES airplane("airplaneId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE gates(

    "gatesId" int,

    "gateName" VARCHAR(15),

    CONSTRAINT "gatesPK" PRIMARY KEY ("gatesId")

);

CREATE TABLE adress(

    "adressId" SERIAL,

    "countryId" int,

    "cityId" int,

    CONSTRAINT "adressPK" PRIMARY KEY ("adressId"),

    CONSTRAINT "adressCountryFK" FOREIGN KEY ("countryId")

        REFERENCES "country"("countryId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "adressCityFK" FOREIGN KEY ("cityId")

        REFERENCES "city"("cityId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE Table country(

    "countryId" int,

    "countryName" VARCHAR(15),

    CONSTRAINT "countryPK" PRIMARY KEY ("countryId")

);

CREATE TABLE city(

    "cityId" int,

    "cityName" VARCHAR(15),

    CONSTRAINT "cityPK" PRIMARY KEY ("cityId")

);

CREATE TABLE ticket(

    "ticketId" SERIAL,

    "airplaneId" int,

    "baggage" int,

    "date" DATE,

    "passengerId" int,

    "side" SMALLINT,

    "gateId"int,

    "planId" int,

    CONSTRAINT "ticketPK" PRIMARY KEY ("ticketId"),

    CONSTRAINT "ticketAirplaneFK" FOREIGN KEY ("airplaneId")

        REFERENCES airplane("airplaneId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "ticketBaggageFK" FOREIGN KEY ("baggage")

    REFERENCES baggage("baggageId")

    ON DELETE CASCADE

    ON UPDATE CASCADE,

    CONSTRAINT "ticketPassengerFK" FOREIGN KEY ("passengerId")

    REFERENCES passengers("passengerId")

    ON DELETE CASCADE

    ON UPDATE CASCADE,

    CONSTRAINT "ticketGateFK" FOREIGN KEY ("gateId")

    REFERENCES gates("gatesId")

    ON DELETE CASCADE

    ON UPDATE CASCADE,

    CONSTRAINT "ticketPlanFK" Foreign Key ("planId")

        REFERENCES nowplans("nowPlanId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE baggage(

    "baggageId" int,

    "passengerId" int,

    "weight" float,

    CONSTRAINT "baggagePK" PRIMARY KEY ("baggageId"),

    CONSTRAINT "baggagePassengerFK" FOREIGN KEY ("passengerId")

        REFERENCES passengers("passengerId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE stations(

    "stationId" int,

    "stationName" VARCHAR(15),

    CONSTRAINT "stationsPK" PRIMARY KEY ("stationId")

);

CREATE TABLE airplane(

    "airplaneId" SERIAL,

    "pilotId" int,

    "passengers" int[],

    "gate" int,

    CONSTRAINT "airplanePK" PRIMARY KEY ("airplaneId"),

    CONSTRAINT "airplaneGateKF" FOREIGN KEY ("gate")

    REFERENCES gates("gatesId")

    ON DELETE CASCADE

    ON UPDATE CASCADE

);

CREATE TABLE nowplans(

    "nowPlanId" SERIAL,

    "starFrom" VARCHAR(15), ---- t

    "goTo" VARCHAR(15),

    "when" TIME,

    "airplaneId" int,

    "station1" int,

    "station2" int,

    "kind" CHAR,

    "travelTime" int,

    CONSTRAINT "nowPlanPK" PRIMARY KEY ("nowPlanId"),

    CONSTRAINT "nowplansAirplaneFK" FOREIGN KEY ("airplaneId")

        REFERENCES airplane("airplaneId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "nowplansStation1FK" FOREIGN KEY ("station1")

        REFERENCES stations("stationId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "nowplansStation2FK" FOREIGN KEY ("station2")

        REFERENCES stations("stationId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE vip(

    "vipId" int,

    "palance" money, ---b

    "maxBaggageNo" SMALLINT DEFAULT 3,

    "maxWeight" FLOAT DEFAULT 40,

    "freePlaces" SMALLINT,

    CONSTRAINT "vipPK" PRIMARY KEY ("vipId"),

    CONSTRAINT "vipNowplansInheritanceFK" FOREIGN KEY ("vipId")

    REFERENCES nowplans("nowPlanId")

    ON DELETE CASCADE

    ON UPDATE CASCADE

);

CREATE TABLE normal\_plan(

    "normalPlanId" int,

    "palance" money, ---b

    "maxBaggageNo" SMALLINT DEFAULT 2,

    "maxWeight" FLOAT DEFAULT 24,

    "freePlaces" SMALLINT,

    CONSTRAINT "normalPlanPK" PRIMARY KEY ("normalPlanId"),

    CONSTRAINT "normal\_planNowplansInheritanceFK" FOREIGN KEY ("normalPlanId")

    REFERENCES nowplans("nowPlanId")

    ON DELETE CASCADE

    ON UPDATE CASCADE

);

-- this table is for all tickets already taken

CREATE TABLE booking(

    "bookingId" SERIAL, -- triger

    "planId" int,

    "gateId" int,

    "ticketId" INT,

    CONSTRAINT "bookingPK" PRIMARY KEY ("bookingId"),

    CONSTRAINT "bookingPlanFK" FOREIGN KEY ("planId")

        REFERENCES nowplans("nowPlanId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "bookingGateFK" FOREIGN KEY ("gateId")

        REFERENCES gates("gatesId")

        ON DELETE CASCADE

        ON UPDATE CASCADE,

    CONSTRAINT "bookingTicketIdFK" FOREIGN KEY ("ticketId")

        REFERENCES ticket("ticketId")

        ON DELETE CASCADE

        ON UPDATE CASCADE

);

CREATE TABLE weather(

    "weatherId" SERIAL,

    "date" date DEFAULT 'now',

    "weather" VARCHAR(15),

    "degree" int,

    CONSTRAINT "weatherPK" PRIMARY KEY ("weatherId")

);

-- function1

CREATE OR REPLACE FUNCTION calculate\_ticket\_price\_vip(time\_of\_travel INT)

RETURNS INT AS $$

DECLARE

    ticket\_price INT;

BEGIN

        ticket\_price := 40 \* time\_of\_travel;

    -- Return the calculated ticket price

    RETURN ticket\_price;

END;

$$ LANGUAGE plpgsql;

-- function2

CREATE OR REPLACE FUNCTION calculate\_ticket\_price(time\_of\_travel INT)

RETURNS INT AS $$

DECLARE

    ticket\_price INT;

BEGIN

        ticket\_price := 20 \* time\_of\_travel;

    -- Return the calculated ticket price

    RETURN ticket\_price;

END;

$$ LANGUAGE plpgsql;

-- function 3

CREATE OR REPLACE FUNCTION personAdd(personNo int)

RETURNS VOID

AS

$$

BEGIN

    IF personNo > 0 THEN

        FOR i IN 1 .. personNo LOOP

            INSERT INTO "person" ("name","surname", "Tel no","addressId","passenger","employee")

            VALUES(

                substring('ABCDEFGHIiJKLMNOPRSTUVYZ' from ceil(random()\*10)::smallint for ceil(random()\*20)::SMALLINT),

                substring('ABCDEFGHIiJKLMNOPRSTUVYZ' from ceil(random()\*10)::smallint for ceil(random()\*20)::SMALLINT),

                floor(random() \* 100000)::bigint,

                floor(random() \* 5)::bigint,

                RANDOM() < 0.5,

                RANDOM() > 0.5

                 );

        END LOOP;

    END IF;

END;

$$

LANGUAGE 'plpgsql'  SECURITY DEFINER;

-- function 4

CREATE OR REPLACE FUNCTION delete\_all\_person()

RETURNS VOID

AS

$$

BEGIN

    DELETE FROM person;

    RETURN;

END;

$$

LANGUAGE 'plpgsql';

-- trigger 1

CREATE TABLE person\_name\_update(

    "personId" int PRIMARY KEY,

    "oldName" VARCHAR(25),

    "newName" VARCHAR(25)

);

CREATE OR REPLACE FUNCTION person\_update\_trigger()

RETURNS TRIGGER AS $$

BEGIN

    RAISE NOTICE 'Updated person record: personId=%, old\_name=%, new\_name=%',

        OLD."personId", OLD."name", NEW."name";

        INSERT INTO person\_name\_update("personId","oldName","newName")

        VALUES(OLD."personId", OLD."name", NEW."name");

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER person\_update

AFTER UPDATE ON person

FOR EACH ROW

EXECUTE FUNCTION person\_update\_trigger();

-- trigger 2

CREATE TABLE ticket\_canceletion(

    "ticketId" int PRIMARY KEY,

    "passenger" VARCHAR(15),

    "passengerId" int

);

CREATE OR REPLACE FUNCTION ticket\_delete\_trigger()

RETURNS TRIGGER AS $$

BEGIN

    RAISE NOTICE 'Deleted ticket record: ticketId=%, baggage=%',

        OLD."ticketId", OLD."baggage";

    INSERT INTO ticket\_canceletion ("ticketId", "passenger", "passengerId")

    VALUES (OLD."ticketId",

            (SELECT "name" FROM person WHERE "personId" = OLD."passengerId" LIMIT 1),

            OLD."passengerId");

    RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER ticket\_delete

AFTER DELETE ON ticket

FOR EACH ROW

EXECUTE FUNCTION ticket\_delete\_trigger();

-- trigger 3

CREATE TABLE managers\_update(

    "manUpdate" SERIAL PRIMARY KEY,

    "lastManagerId" int,

    "lastManagerName" VARCHAR(15),

    "newManagerId" int,

    "newManagerName" VARCHAR(15)

);

CREATE OR REPLACE FUNCTION manager\_update\_trigger()

RETURNS TRIGGER AS $$

BEGIN

    RAISE NOTICE 'Updated manager record: managerId=%',

        OLD."managerId";

        INSERT INTO managers\_update("lastManagerId","lastManagerName","newManagerId","newManagerName")

        VALUES (OLD."managerId",

        (SELECT "name" FROM person WHERE "personId" = OLD."managerId")

        ,NEW."managerId",

        (SELECT "name" FROM person WHERE "personId" = NEW."managerId"));

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER manager\_update

AFTER UPDATE OR INSERT ON manager

FOR EACH ROW

EXECUTE FUNCTION manager\_update\_trigger();

-- trigger 4

CREATE OR REPLACE FUNCTION make\_booking()

RETURNS TRIGGER AS $$

BEGIN

    INSERT INTO booking("ticketId","gateId","planId")

    VALUES (NEW."ticketId",NEW."gateId",NEW."planId");

RETURN new;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER booking\_after\_add\_ticket

AFTER INSERT ON ticket

FOR EACH ROW

EXECUTE FUNCTION make\_booking();

And this my application's code using C#

using Npgsql;

namespace ProjectOfDataBases

{

class person {

public int personId { get; set; }

public string name { get; set; }

public string surname { get; set; }

public int addressId { get; set; }

public bool passenger { get; set; }

public bool employee { get; set; }

public int TelNo { get; set; }

}

class Program

{

static void Main(string[] args)

{

string connString = "Host=localhost;Port=5432;Username=postgres;Password=1234;Database=DataBasesProject;";

while (true)

{

Console.Clear();

Console.WriteLine("Choose an operation:");

Console.WriteLine("1. Add Data");

Console.WriteLine("2. Update Data");

Console.WriteLine("3. Delete Data");

Console.WriteLine("4. Show Data");

Console.WriteLine("5. Exit");

string choice = Console.ReadLine();

try

{

using (var conn = new NpgsqlConnection(connString))

{

conn.Open(); // Attempt to open the connection

Console.WriteLine("Connection successful.");

switch (choice)

{

case "1":

AddData(conn);

break;

case "2":

UpdateData(conn);

break;

case "3":

DeleteData(conn);

break;

case "4":

ShowData(conn);

break;

case "5":

return;

default:

Console.WriteLine("Invalid choice! Try again.");

break;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Connection failed: {ex.Message}");

}

Console.ReadKey();

}

}

static void AddData(NpgsqlConnection conn)

{

try

{

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

Console.WriteLine("Enter surname:");

string surname = Console.ReadLine();

Console.WriteLine("Enter phone number (Tel no):");

int telNo = Convert.ToInt32(Console.ReadLine());

string query = "INSERT INTO person (\"name\", \"surname\", \"Tel no\") VALUES (@name, @surname, @telNo)";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("name", name);

cmd.Parameters.AddWithValue("surname", surname);

cmd.Parameters.AddWithValue("telNo", telNo);

cmd.ExecuteNonQuery();

Console.WriteLine("Data added successfully.");

}

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

static void UpdateData(NpgsqlConnection conn)

{

try

{

Console.WriteLine("Enter person ID to update:");

int personId = int.Parse(Console.ReadLine());

Console.WriteLine("Enter new name:");

string name = Console.ReadLine();

Console.WriteLine("Enter new surname:");

string surname = Console.ReadLine();

string query = "UPDATE person SET \"name\" = @name, \"surname\" = @surname WHERE \"personId\" = @personId";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("name", name);

cmd.Parameters.AddWithValue("surname", surname);

cmd.Parameters.AddWithValue("personId", personId);

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

Console.WriteLine("Data updated successfully.");

else

Console.WriteLine("Person ID not found.");

}

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

static void DeleteData(NpgsqlConnection conn)

{

try

{

Console.WriteLine("Enter person ID to delete:");

int personId = int.Parse(Console.ReadLine());

string query = "DELETE FROM person WHERE \"personId\" = @personId";

using (var cmd = new NpgsqlCommand(query, conn))

{

cmd.Parameters.AddWithValue("personId", personId);

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)

Console.WriteLine("Data deleted successfully.");

else

Console.WriteLine("Person ID not found.");

}

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

static void ShowData(NpgsqlConnection conn)

{

try

{

string query = "SELECT \* FROM person";

using (var cmd = new NpgsqlCommand(query, conn))

using (var reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

person p = new person();

p.personId = reader.GetInt32(0);

p.name = reader.GetString(1);

p.surname = reader.GetString(2);

p.addressId = reader.GetInt32(3);

p.passenger = reader.GetBoolean(4);

p.employee = reader.GetBoolean(5);

p.TelNo = reader.GetInt32(6);

Console.WriteLine($"Person ID: {p.personId}, Name: {p.name} {p.surname}, " +

$"address: {p.addressId}, passenger: {p.passenger}, employee {p.employee}, Tel No: {p.TelNo}");

}

}

else

{

Console.WriteLine("No records found.");

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

}

}

}

At the application I use only person table for control I was wanting to complete it but my time is finish . . . :) see you later at another project