

Carlos Amador A01329447 Mónica Pérez A01329619

# Bases de datos avanzadas Tarea 2.3

10.50.67.83

**Gimnasio** 

## Índice

<u>Índice</u>

**Descripción** 

Montado de base en servidor

Base de datos

Código almacenado

<u>Functions</u>

**Procedures** 

**Triggers** 

## Descripción

Es una base de datos modelada a partir de la idea de un gimnasio. La base de datos está implementada en Postgresql.

## Montado de base en servidor

Para montar la base de datos en el servidor, hicimos ssh, accedimos a postgresql, y luego a la base "usuario1". Finalmente, insertamos los comandos necesarios. Capturas de pantalla:

```
🔵 📵 usuario1@servi: ~
 usuario1$> $performed$ LANGUAGE plpgsql;
usuario1-> RETURNS trigger AS $after_update_member_schedule$
usuario1>> BEGIN
usuario1$>
usuario1$>
usuario1$>
                              LOCK TABLE member_schedule IN EXCLUSIVE MODE;
                              IF(NEW.absences >= 3) THEN
    PERFORM unsubscribe(OLD.member_id, OLD.schedule_id);
END IF;
usuario1$>
usuario1$>
usuario1$>
usuario1$>
                              RETURN NEW;
                        END:
 usuario1$> $after_update_member_schedule$ LANGUAGE plpgsql;
usuario1$> $arter_update_member_schedule$ LANGUAGE plpgsql;
CREATE FUNCTION
usuario1=>
usuario1=> CREATE TRIGGER after_update_member_schedule AFTER UPDATE ON member_schedule
usuario1-> FOR EACH ROW EXECUTE PROCEDURE checkAbsences();
CREATE TRIGGER
usuario1=> \df
usuario1=> \df
usuario1=> \df
                       List of relations
                                          | Type
 Schema I
                        Name
                                                       l Owner
 public
public
public
                                           | table
| table
                branches
                                                           usuario1
                                                           usuario1
usuario1
                cards
               classes | table
member_schedule | table
members | table
schedules | table
 public
                                                          usuario1
  public
                                                          usuario1
                                                          usuario1
  public
7 sous
                trainers
usuario1=>
```

⊗ ⊜   usuario1@servi: ~					
6   41071   f   f	0	\x			
41065   RI_ConstraintTrigger_c_41074   1644	5   0	t	1	41049	4105
6   41071   f   f	0	\x		10000000000	10000
41065   RI_ConstraintTrigger_c_41075   1645	17   0	Į t	l l	41049	4105
6   41071   f   f   41022   RI ConstraintTrigger a 41093   1654	0   9   0	\x		41083	4102
9   41092   f   f	9   0	1 \x 1	1	41003	4102
41022   RI ConstraintTrigger a 41094   1655	17   0	' lît '	i i	41083	4102
9   41092   f   f	0	\x			
41083   RI_ConstraintTrigger_c_41095   1644	5   0	it	1	41022	4102
9   41092   f   f	0	\x			
41083   RI_ConstraintTrigger_c_41096   1645	17   0	t	11	41022	4102
9   41092   f   f	0	\x			20.00
41083   RI_ConstraintTrigger_a_41098   1654	9   0	t	L	41083	4109
0   41097   f   f   41083   RI ConstraintTrigger a 41099   1655	0   17   0	/x		41083	4109
41083   RI_ConstraintTrigger_a_41099   1655   0   41097   f   f	0	\x	I.	41003	4109
41083   RI ConstraintTrigger c 41100   1644	5 1 0	' \^+ '	1	41083	4109
0   41097   f   f	0 1	\x		11005	1100
41083   RI ConstraintTrigger c 41101   1645	17   0	i i t	1	41083	4109
0   41097   f   f	0	\x			
41083   RI_ConstraintTrigger_a_41119   1654	9   0	į t	1	41109	4109
0   41118   f   f	0	\x			11000000
41083   RI_ConstraintTrigger_a_41120   1655	17   0	, j t	Į.	41109	4109
0   41118   f   f	0	\x		44000 1	4400
41109   RI_ConstraintTrigger_c_41121   1644	5   0	l t	1	41083	4109
0   41118   f   f   41109   RI ConstraintTrigger c 41122   1645	0   17   0	\x		41083	4109
0   41118   f   f	0	\x	l l	41003	4103
41123   after update member schedule   41132	17   0	l f	1	0	
0   0   f   f	0	\x	''		
(25 rows)					
(END)					

#### Al final de la captura anterior, se puede ver el trigger insertado.

```
List of functions

Schema | Name | Result data type | Argument data types | Type

public | allowed | boolean | member_id integer, branch_id integer | normal

public | checkabsences | trigger | trigger

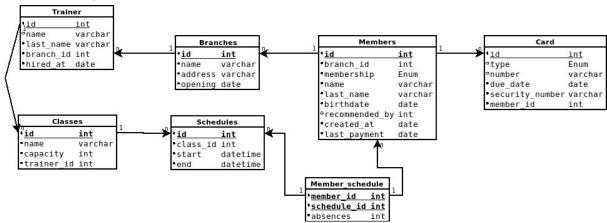
public | enroll | boolean | member integer, schedule integer | normal

public | unsubscribe | void | member integer, schedule integer | normal

(4 rows)
```

## Base de datos





En cada tabla se cuenta con una llave primaria de tipo serial (único y que se incrementa cada vez que se inserta en la tabla), podemos observar que todos los atributos de la tabla tienen la restricción de valor no nulo, excepto el atributo recommended\_by en la tabla de miembros, ya que puede no haber llegado por recomendación de alguien. También utilizamos dos enumerados para guardar los tipos de tarjeta y de membresía.

Los comandos utilizados para crear las tablas de la base son los siguientes:

• Tabla sucursales.

```
CREATE TABLE branches(

id serial PRIMARY KEY,

name text NOT NULL,

address text NOT NULL,

opening date NOT NULL
);
```

Tabla entrenadores.

```
CREATE TABLE trainers(
   id serial PRIMARY KEY,
   name text NOT NULL,
   last_name text NOT NULL,
   branch_id int REFERENCES branches(id),
   hired_at date NOT NULL
);
```

• Tabla de clases.

```
CREATE TABLE classes(
   id serial PRIMARY KEY,
   name text NOT NULL,
   capacity int NOT NULL,
   trainer_id int REFERENCES trainers(id)
);
```

Tabla de horarios de clase.

```
CREATE TABLE schedules(
   id serial PRIMARY KEY,
   class_id int REFERENCES classes(id),
   start_time timestamp NOT NULL,
   end_time timestamp NOT NULL
);
```

• Tabla de miembros del gimnasio.

```
CREATE TABLE members(
   id serial PRIMARY KEY,
   branch_id int REFERENCES branches(id),
   membership membership_type NOT NULL,
   name text NOT NULL,
   last_name text NOT NULL,
   birthdate date NOT NULL,
   recommended_by int REFERENCES members(id) NULL,
   created_at timestamp,
   last_payment date
);
```

• Tabla de tarjetas de los miembros.

```
CREATE TABLE cards(
   id serial PRIMARY KEY,
   card_t card_type NOT NULL,
   number text NOT NULL,
   due_date date NOT NULL,
   security_number text NOT NULL,
   member_id int REFERENCES members(id)
);
```

• Tabla cruce de miembros y horarios.

```
CREATE TABLE member_schedule(
    member_id int NOT NULL,
    schedule_id int NOT NULL,
    absences int NOT NULL DEFAULT (0),
    PRIMARY KEY (member_id, schedule_id)
);
```

• Enumerado de tipos de membresía.

```
CREATE TYPE membership_type AS ENUM ('standard', 'premium');
```

• Enumerado de tipos de tarjeta.

```
CREATE TYPE card_type AS ENUM ('debit', 'credit');
```

### Código almacenado

#### **Functions**

- allowed (member\_id INTEGER, branch\_id INTEGER): Existen dos tipos de membresía, la standard y la premium. Si un usuario tiene membresía standard, sólo se puede inscribir a clases de su gimnasio; si es premium, puede en el que sea. De acuerdo a lo anterior, retorna true o false si un miembro puede atender a un branch de un gimnasio específico.
- checkAbsences(): es la función del trigger.

#### Ejemplo

```
CREATE OR REPLACE FUNCTION allowed (member id INTEGER, branch id INTEGER)
RETURNS boolean AS $allowed$
       allowed boolean:
        member branch id integer;
        membership membership type;
        SELECT m.branch_id, m.membership
INTO member_branch_id, membership
        FROM members m
        WHERE m.id = member id;
        allowed := true;
        IF(membership = 'premium'::membership_type) THEN
            allowed := true;
        ELSIF (member branch id = branch id) THEN
            allowed := true;
        ELSE
        allowed:= false;
END IF;
        RETURN allowed;
$allowed$ LANGUAGE plpgsql;
```

#### **Procedures**

- enroll (member INTEGER, schedule INTEGER): De acuerdo al id de un miembro y el de un schedule, intenta inscribir a un miembro a un schedule, lo cual implica verificar que se pueda inscribir en el gimnasio en el que está el schedule, así como verificar si hay cupo. Incluye una transacción para no hacer dirty reads del cupo.
- unsubscribe (member INTEGER, schedule INTEGER): Realiza lo opuesto a enroll, dando de baja a un miembro de un schedule.

#### Ejemplo

```
/*Desinscribe a un miembro de una clase*/
CREATE OR REPLACE FUNCTION unsubscribe(member INTEGER, schedule INTEGER)

RETURNS void AS $$

BEGIN

LOCK TABLE member_schedule IN EXCLUSIVE MODE;

DELETE FROM member_schedule ms

WHERE ms.member_id = member AND ms.schedule_id = schedule;

RETURN;

END;

S$ LANGUAGE plpgsql;
```

#### Triggers

 after\_update\_member\_schedule: cuando un miembro llega a 3 faltas en un schedule, el sistema lo da de baja utilizando este trigger, llamando checkAbsences para contar las faltas.

```
/*Desinscribe a un miembro de una clase*/
CREATE OR REPLACE FUNCTION checkAbsences()

RETURNS trigger AS $after_update_member_schedule$

BEGIN

LOCK TABLE member_schedule IN EXCLUSIVE MODE;

IF(NEW.absences >= 3) THEN

PERFORM unsubscribe(OLD.member_id, OLD.schedule_id);

END IF;

RETURN NEW;

END;

$after_update_member_schedule$ LANGUAGE plpgsql;

CREATE TRIGGER after_update_member_schedule AFTER UPDATE ON member_schedule
FOR EACH ROW EXECUTE PROCEDURE checkAbsences();
```

#### **Views**

Realizamos una view que permite visualizar de manera compacta la lista de los miembros del gimnasio, mostrando el id, nombre y tipo de membresía de los mismos.

```
ERROR: syntax error at or near "selct"
LINE 1: selct * from pg-trigger;
usuarioi=> selcet * from pg-trigger;
ERROR: syntax error at or near "seleet"
LINE 1: select * from pg-trigger;

LINE 1: select * from pg-trigger;

usuarioi=> select * from pg-trigger;
ERROR: syntax error at or near ""
LINE 1: select * from pg-trigger;

usuarioi=> select * from pg-trigger;

usuarioi=> select * from pg-trigger;
usuarioi=> SELECT tide member_view As
usuarioi-> FROM nembers;
CREATE VIEW
usuarioi=> select * from member ship

usuarioi=> select * from member_view;
id | name | membership

1 | Carlos | premium
2 | Mônica | standard
3 | Pirri | standard
4 | Angel | standard
5 | Rafael | premium
6 | Fernando | premium
7 | Estefania | premium
8 | Jorge | standard
9 | J | standard
10 | Andrés | premium
11 | Aranzza | standard
12 | Francisco | premium
11 | Aranzza | standard
12 | Francisco | premium
11 | Aranzza | standard
12 | Francisco | premium
11 | Aranzza | standard
12 | Francisco | premium
11 | Aranzza | standard
12 | Francisco | premium
15 | Serial |
```