



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico 1

---

Bases de datos  
Primer Cuatrimestre de 2017

## Grupo 4

Integrante	LU	Correo electrónico
De Carli, Nicolás	715/13	nikodecarli@gmail.com
Gásperi, Fernando	56/09	fgasperijabalera@gmail.com
Gambaccini, Ezequiel	164/13	ezequiel.gambaccini@gmail.com
Minces, Javier	231/13	javier.minces@gmail.com



Facultad de Ciencias Exactas y Naturales  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+ +54 +11) 4576-3300

<http://www.exactas.uba.ar>

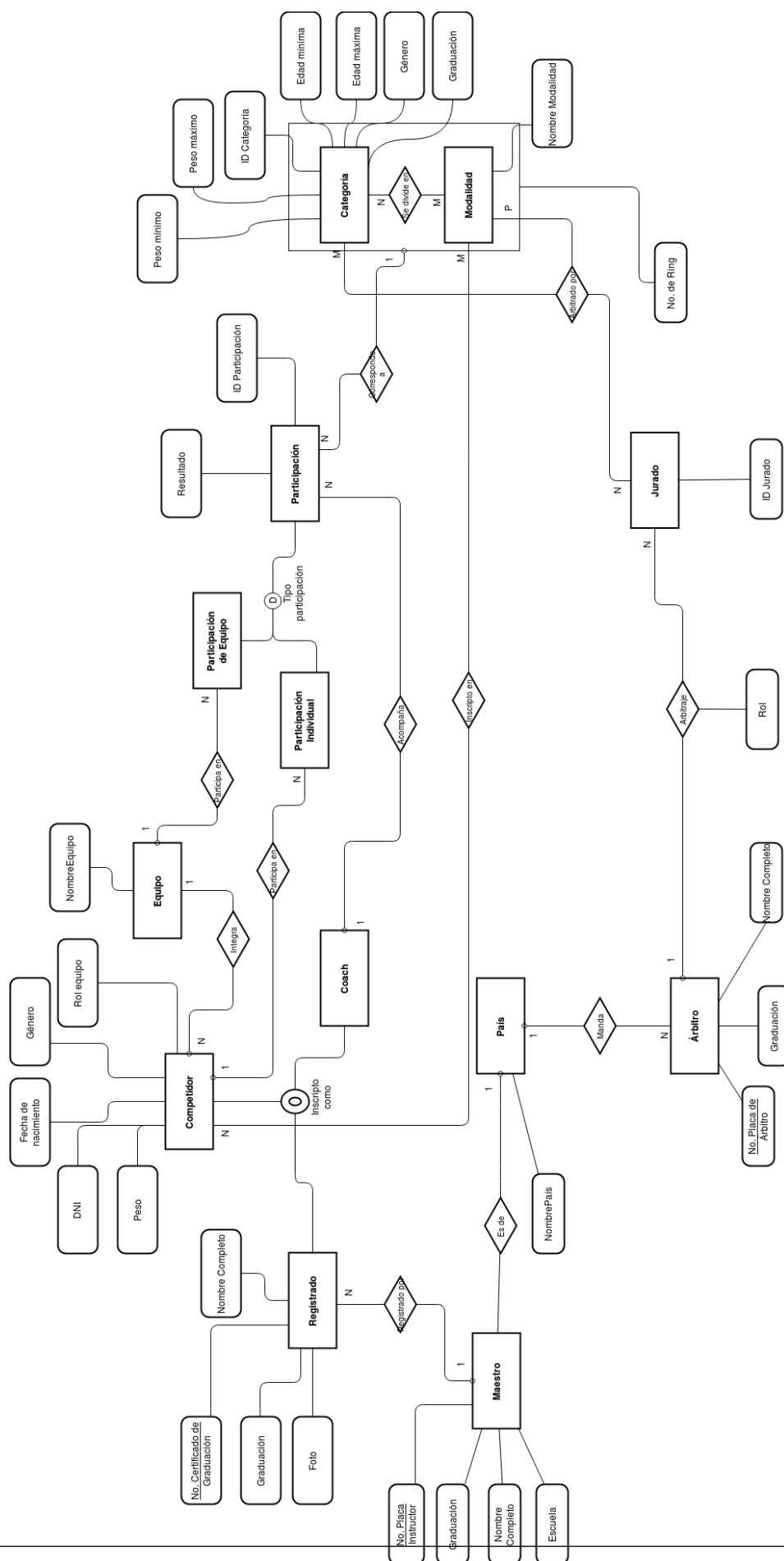
## Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Diagrama Entidad Relación</b>	<b>3</b>
<b>3. Modelo Relacional</b>	<b>4</b>
3.1. Restricciones . . . . .	4
<b>4. Código</b>	<b>5</b>
4.1. Creación de tablas . . . . .	5
4.2. Restricciones - triggers . . . . .	12
4.3. Restricciones - stored procedure . . . . .	16
4.4. Queries . . . . .	23
<b>5. Conclusión</b>	<b>26</b>

## 1. Introducción

En este trabajo práctico presentaremos el diseño de una base de datos para la inscripción en un mundial de taekwondo. Debemos guardar los datos correspondientes a los competidores, coaches y árbitros, y los resultados obtenidos en cada categoría de cada modalidad.

En primer lugar mostraremos el diagrama entidad-relación, junto con las restricciones y suposiciones expresadas en lenguaje natural. Luego presentaremos el modelo relacional derivado, con el que se hará la base de datos real en MySQL.



### 3. Modelo Relacional

Maestro(PlacaInstructor, Escuela, Nombre Completo, Graduacion, NombrePais)  
País(NombrePais)  
Árbitro(PlacaArbitro, Graduacion, Nombre Completo, NombrePais)  
Arbitraje(IDJurado, PlacaArbitro, Rol)  
Jurado(IDJurado)  
ArbitradoPor(NombreModalidad, IDCategoria, IDJurado)  
Modalidad(NombreModalidad)  
Categoría(IDCategoria, Graduacion, EdadMinima, EdadMaxima, Sexo, PesoMinimo, PesoMaximo)  
SeDivideEn(IDCategoria, NombreModalidad, Ring)  
Registrado(NumeroCertificadoGraduacion, Foto, Graduacion, NombreCompleto, PlacaInstructor)  
Competidor(NumeroCertificadoGraduacion, Peso, DNI, FechaNacimiento, Genero, RolEquipo, NombreEquipo)  
Coach(NumeroCertificadoGraduacion)  
InscriptoEn(NumeroCertificadoGraduacion, NombreModalidad)  
Equipo(NombreEquipo)  
Participación(IDParticipacion, Resultado, IDCategoria, NombreModalidad, NumeroCertificadoGraduacionCoach, Tipo)  
ParticipaciónIndividual(IDParticipacion, NumeroCertificadoGraduacionCompetidor) ParticipaciónDeEquipo(IDParticipacion, NombreEquipo)

#### 3.1. Restricciones

- La cantidad de coaches de una escuela debe ser 1/5 de la cantidad de alumnos
- La graduación va de 1er dan a 6to dan
- Para toda participación:
  - Si la categoría tiene peso máximo y mínimo el peso del competidor debe estar en el rango
  - Si la categoría tiene edad máxima y mínima la edad del competidor debe estar en el rango
  - Si la categoría tiene género el competidor debe ser de ese género
  - Si la categoría tiene graduación el competidor debe ser de esa graduación
- Todos los integrantes de un equipo deben estar inscriptos en la modalidad combate por equipos
- Los equipos deben tener 5 integrantes cuyo rol sea “titular” y 3 cuyo rol sea “suplente”
- Todos los integrantes de un equipo deben ser de la misma escuela
- Todos los integrantes de un equipo deben ser del mismo género, que debe corresponder con el género de la categoría de su participación por equipo.
- Las participaciones de equipo deben ser en la modalidad “combate por equipos”.
- Las participaciones individuales no pueden ser en la modalidad “combate por equipos”.
- Cada competidor no puede tener más de una participación por modalidad
- En cada jurado hay:
  - un árbitro con rol “presidente de mesa”
  - un “árbitro central”
  - dos o más “jueces”

- tres o más “suplentes”.
- La graduación de cada árbitro debe ser superior a la graduación de las categorías en las que es jurado.
- El coach de una participación debe ser de la misma escuela que el competidor

## 4. Código

### 4.1. Creación de tablas

```
— MySQL Script generated by MySQL Workbench
— Thu May 4 16:37:44 2017
— Model: New Model      Version: 1.0
— MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

— Schema mundial.taekwondo

— Schema mundial.taekwondo

CREATE SCHEMA IF NOT EXISTS 'mundial.taekwondo' DEFAULT CHARACTER SET utf8 ;
USE 'mundial.taekwondo' ;

— Table 'mundial.taekwondo'. 'Pais'

DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Pais' ;

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Pais' (
  'NombrePais' VARCHAR(250) NOT NULL,
  PRIMARY KEY ('NombrePais'))
ENGINE = InnoDB;

— Table 'mundial.taekwondo'. 'Maestro'

DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Maestro' ;

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Maestro' (
  'PlacaInstructor' INT NOT NULL AUTO_INCREMENT,
  'Escuela' VARCHAR(45) NULL,
  'NombreCompleto' VARCHAR(45) NULL,
  'Graduacion' TINYINT(3) NULL,
  'NombrePais' VARCHAR(250) NULL,
  PRIMARY KEY ('PlacaInstructor'),
  INDEX 'NombrePais_idx' ('NombrePais' ASC),
```

```
CONSTRAINT 'NombrePais'
  FOREIGN KEY ('NombrePais')
  REFERENCES 'mundial.taekwondo'. 'Pais' ('NombrePais')
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;
```

---

— *Table 'mundial.taekwondo'. 'Arbitro'*

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Arbitro' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Arbitro' (
  'PlacaArbitro' INT NOT NULL AUTO_INCREMENT,
  'Graduacion' TINYINT(3) NULL,
  'NombreCompleto' VARCHAR(45) NULL,
  'NombrePais' VARCHAR(250) NULL,
  PRIMARY KEY ('PlacaArbitro'),
  INDEX 'NombrePais_idx' ('NombrePais' ASC),
  CONSTRAINT 'NombrePaisArbitro'
    FOREIGN KEY ('NombrePais')
    REFERENCES 'mundial.taekwondo'. 'Pais' ('NombrePais')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;
```

---

— *Table 'mundial.taekwondo'. 'Jurado'*

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Jurado' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Jurado' (
  'IDJurado' INT NOT NULL AUTO_INCREMENT,
  PRIMARY KEY ('IDJurado'))
```

```
ENGINE = InnoDB;
```

---

— *Table 'mundial.taekwondo'. 'Arbitraje'*

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Arbitraje' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Arbitraje' (
  'IDJurado' INT NOT NULL,
  'PlacaArbitro' INT NOT NULL,
  'Rol' VARCHAR(45) NULL,
  PRIMARY KEY ('IDJurado', 'PlacaArbitro'),
  INDEX 'PlacaArbitro_idx' ('PlacaArbitro' ASC),
  CONSTRAINT 'PlacaArbitro'
    FOREIGN KEY ('PlacaArbitro')
    REFERENCES 'mundial.taekwondo'. 'Arbitro' ('PlacaArbitro')
    ON DELETE NO ACTION)
```

```
ON UPDATE NO ACTION,  
CONSTRAINT 'IDJurado'  
FOREIGN KEY ('IDJurado')  
REFERENCES 'mundial_taekwondo'. 'Jurado' ('IDJurado')  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

---

— Table 'mundial\_taekwondo'. 'Modalidad'

---

```
DROP TABLE IF EXISTS 'mundial_taekwondo'. 'Modalidad' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial_taekwondo'. 'Modalidad' (  
  'NombreModalidad' VARCHAR(45) NOT NULL,  
  PRIMARY KEY ('NombreModalidad'))  
ENGINE = InnoDB;
```

---

— Table 'mundial\_taekwondo'. 'Categoria'

---

```
DROP TABLE IF EXISTS 'mundial_taekwondo'. 'Categoria' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial_taekwondo'. 'Categoria' (  
  'IDCategoria' INT NOT NULL AUTO_INCREMENT,  
  'Graduacion' TINYINT(3) NULL,  
  'EdadMinima' INT NULL,  
  'EdadMaxima' INT NULL,  
  'Sexo' VARCHAR(45) NULL,  
  'PesoMinimo' INT NULL,  
  'PesoMaximo' INT NULL,  
  PRIMARY KEY ('IDCategoria'))  
ENGINE = InnoDB;
```

---

— Table 'mundial\_taekwondo'. 'ArbitradoPor'

---

```
DROP TABLE IF EXISTS 'mundial_taekwondo'. 'ArbitradoPor' ;
```

```
CREATE TABLE IF NOT EXISTS 'mundial_taekwondo'. 'ArbitradoPor' (  
  'NombreModalidad' VARCHAR(45) NOT NULL,  
  'IDCategoria' INT NOT NULL,  
  'IDJurado' INT NULL,  
  PRIMARY KEY ('NombreModalidad', 'IDCategoria'),  
  INDEX 'IDJurado_idx' ('IDJurado' ASC),  
  INDEX 'IDCategoria_idx' ('IDCategoria' ASC),  
  CONSTRAINT 'IDJuradoArbitrado'  
    FOREIGN KEY ('IDJurado')  
    REFERENCES 'mundial_taekwondo'. 'Jurado' ('IDJurado')  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,
```



```
CONSTRAINT 'NombreModalidadArbitrado'
  FOREIGN KEY ('NombreModalidad')
  REFERENCES 'mundial.taekwondo'. 'Modalidad' ('NombreModalidad')
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
CONSTRAINT 'IDCategoriaArbitrado'
  FOREIGN KEY ('IDCategoria')
  REFERENCES 'mundial.taekwondo'. 'Categoria' ('IDCategoria')
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

---

— Table 'mundial.taekwondo'. 'SeDivideEn' —

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'SeDivideEn' ;

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'SeDivideEn' (
  'IDCategoria' INT NOT NULL,
  'NombreModalidad' VARCHAR(45) NOT NULL,
  'Ring' INT NULL,
  PRIMARY KEY ('IDCategoria', 'NombreModalidad'),
  INDEX 'NombreModalidad_idx' ('NombreModalidad' ASC),
  CONSTRAINT 'IDCategoria'
    FOREIGN KEY ('IDCategoria')
    REFERENCES 'mundial.taekwondo'. 'Categoria' ('IDCategoria')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'NombreModalidad'
    FOREIGN KEY ('NombreModalidad')
    REFERENCES 'mundial.taekwondo'. 'Modalidad' ('NombreModalidad')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

---

— Table 'mundial.taekwondo'. 'Registrado' —

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'Registrado' ;

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'Registrado' (
  'NumeroCertificadoGraduacion' INT NOT NULL AUTO.INCREMENT,
  'Foto' BLOB NULL,
  'Graduacion' TINYINT(3) NULL,
  'NombreCompleto' VARCHAR(45) NULL,
  'PlacaInstructor' INT NULL,
  PRIMARY KEY ('NumeroCertificadoGraduacion'),
  INDEX 'PlacaInstructor_idx' ('PlacaInstructor' ASC),
  CONSTRAINT 'PlacaInstructor'
    FOREIGN KEY ('PlacaInstructor')
    REFERENCES 'mundial.taekwondo'. 'Maestro' ('PlacaInstructor')
    ON DELETE NO ACTION
```

**ON UPDATE NO ACTION)**

ENGINE = InnoDB;

---

— Table ‘mundial.taekwondo’. ‘Equipo’

---

**DROP TABLE IF EXISTS** ‘mundial.taekwondo’. ‘Equipo’ ;

**CREATE TABLE IF NOT EXISTS** ‘mundial.taekwondo’. ‘Equipo’ (  
    ‘NombreEquipo’ **VARCHAR**(200) **NOT NULL**,  
    **PRIMARY KEY** (‘NombreEquipo’))  
ENGINE = InnoDB;

---

— Table ‘mundial.taekwondo’. ‘Competidor’

---

**DROP TABLE IF EXISTS** ‘mundial.taekwondo’. ‘Competidor’ ;

**CREATE TABLE IF NOT EXISTS** ‘mundial.taekwondo’. ‘Competidor’ (  
    ‘NumeroCertificadoGraduacion’ **INT NOT NULL**,  
    ‘Peso’ **INT NULL**,  
    ‘DNI’ **INT NULL**,  
    ‘FechaNacimiento’ **DATE NULL**,  
    ‘Sexo’ **VARCHAR**(45) **NULL**,  
    ‘RolEquipo’ **VARCHAR**(45) **NULL**,  
    ‘NombreEquipo’ **VARCHAR**(200) **NULL**,  
    **PRIMARY KEY** (‘NumeroCertificadoGraduacion’),  
    **INDEX** ‘NombreEquipo\_idx’ (‘NombreEquipo’ **ASC**),  
    **CONSTRAINT** ‘NombreEquipo’  
        **FOREIGN KEY** (‘NombreEquipo’)  
        **REFERENCES** ‘mundial.taekwondo’. ‘Equipo’ (‘NombreEquipo’)  
        **ON DELETE NO ACTION**  
        **ON UPDATE NO ACTION**,  
    **CONSTRAINT** ‘NumeroCertificadoGraduacion’  
        **FOREIGN KEY** (‘NumeroCertificadoGraduacion’)  
        **REFERENCES** ‘mundial.taekwondo’. ‘Registrado’ (‘NumeroCertificadoGraduacion’)  
        **ON DELETE NO ACTION**  
        **ON UPDATE NO ACTION**)  
ENGINE = InnoDB;

---

— Table ‘mundial.taekwondo’. ‘Coach’

---

**DROP TABLE IF EXISTS** ‘mundial.taekwondo’. ‘Coach’ ;

**CREATE TABLE IF NOT EXISTS** ‘mundial.taekwondo’. ‘Coach’ (  
    ‘NumeroCertificadoGraduacion’ **INT NOT NULL**,  
    **PRIMARY KEY** (‘NumeroCertificadoGraduacion’),  
    **CONSTRAINT** ‘NumeroCertificadoGraduacionCoach’  
        **FOREIGN KEY** (‘NumeroCertificadoGraduacion’)  
        **REFERENCES** ‘mundial.taekwondo’. ‘Registrado’ (‘NumeroCertificadoGraduacion’)

```

    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

---

— Table ‘mundial.taekwondo’. ‘Inscripto’ —

---

```

DROP TABLE IF EXISTS ‘mundial.taekwondo’. ‘Inscripto’ ;

CREATE TABLE IF NOT EXISTS ‘mundial.taekwondo’. ‘Inscripto’ (
    ‘NumeroCertificadoGraduacion’ INT NOT NULL,
    ‘NombreModalidad’ VARCHAR(45) NOT NULL,
    PRIMARY KEY (‘NumeroCertificadoGraduacion’, ‘NombreModalidad’),
    CONSTRAINT ‘NombreModalidadInscripto’
        FOREIGN KEY (‘NombreModalidad’)
        REFERENCES ‘mundial.taekwondo’. ‘Modalidad’ (‘NombreModalidad’)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT ‘NumeroCertificadoGraduacionInscripto’
        FOREIGN KEY (‘NumeroCertificadoGraduacion’)
        REFERENCES ‘mundial.taekwondo’. ‘Competidor’ (‘NumeroCertificadoGraduacion’)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

---

— Table ‘mundial.taekwondo’. ‘Participacion’ —

---

```

DROP TABLE IF EXISTS ‘mundial.taekwondo’. ‘Participacion’ ;

CREATE TABLE IF NOT EXISTS ‘mundial.taekwondo’. ‘Participacion’ (
    ‘IDParticipacion’ INT NOT NULL AUTO.INCREMENT,
    ‘Resultado’ TINYINT(1) NULL,
    ‘IDCategoria’ INT NOT NULL,
    ‘NombreModalidad’ VARCHAR(45) NOT NULL,
    ‘NumeroCertificadoGraduacionCoach’ INT NOT NULL,
    ‘Tipo’ VARCHAR(45) NULL,
    PRIMARY KEY (‘IDParticipacion’),
    INDEX ‘IDCategoria_idx’ (‘IDCategoria’ ASC),
    INDEX ‘NombreModalidad_idx’ (‘NombreModalidad’ ASC),
    INDEX ‘NumeroCertificadoGraduacion_idx’ (‘NumeroCertificadoGraduacionCoach’ ASC)
    ,
    CONSTRAINT ‘IDCategoriaParticipacion’
        FOREIGN KEY (‘IDCategoria’)
        REFERENCES ‘mundial.taekwondo’. ‘Categoria’ (‘IDCategoria’)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT ‘NombreModalidadParticipacion’
        FOREIGN KEY (‘NombreModalidad’)
        REFERENCES ‘mundial.taekwondo’. ‘Modalidad’ (‘NombreModalidad’)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,

```

```

CONSTRAINT 'NumeroCertificadoGraduacionParticipacion'
FOREIGN KEY ('NumeroCertificadoGraduacionCoach')
REFERENCES 'mundial.taekwondo'. 'Coach' ('NumeroCertificadoGraduacion')
ON DELETE NO ACTION
ON UPDATE NO ACTION)

```

```
ENGINE = InnoDB;
```

---

— *Table 'mundial.taekwondo'. 'ParticipacionIndividual'*

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'ParticipacionIndividual' ;
```

```

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'ParticipacionIndividual' (
  'IDParticipacion' INT NOT NULL,
  'NumeroCertificadoGraduacionCompetidor' INT NOT NULL,
  PRIMARY KEY ('IDParticipacion'),
  INDEX 'NumeroCertificadoGraduacion_idx' ('NumeroCertificadoGraduacionCompetidor'
    ASC),
  CONSTRAINT 'IDParticipacionIndividual'
    FOREIGN KEY ('IDParticipacion')
    REFERENCES 'mundial.taekwondo'. 'Participacion' ('IDParticipacion')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'NumeroCertificadoGraduacionIndividual'
    FOREIGN KEY ('NumeroCertificadoGraduacionCompetidor')
    REFERENCES 'mundial.taekwondo'. 'Competidor' ('NumeroCertificadoGraduacion')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)

```

```
ENGINE = InnoDB;
```

---

— *Table 'mundial.taekwondo'. 'ParticipacionDeEquipo'*

---

```
DROP TABLE IF EXISTS 'mundial.taekwondo'. 'ParticipacionDeEquipo' ;
```

```

CREATE TABLE IF NOT EXISTS 'mundial.taekwondo'. 'ParticipacionDeEquipo' (
  'IDParticipacion' INT NOT NULL,
  'NombreEquipo' VARCHAR(200) NULL,
  PRIMARY KEY ('IDParticipacion'),
  INDEX 'NombreEquipo_idx' ('NombreEquipo' ASC),
  CONSTRAINT 'NombreEquipoEquipo'
    FOREIGN KEY ('NombreEquipo')
    REFERENCES 'mundial.taekwondo'. 'Equipo' ('NombreEquipo')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT 'IDParticipacionEquipo'
    FOREIGN KEY ('IDParticipacion')
    REFERENCES 'mundial.taekwondo'. 'Participacion' ('IDParticipacion')
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)

```

```
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

## 4.2. Restricciones - triggers

— *Restricciones*

— *La cantidad de coaches de una escuela debe ser 1/5 de la cantidad de alumnos*

```
drop trigger if exists proporcion_coachs_alumnos;  
DELIMITER $$  
CREATE TRIGGER proporcion_coachs_alumnos  
  BEFORE INSERT ON Competidor  
  FOR EACH ROW  
BEGIN  
    declare competidores int;  
    declare coaches int;  
    declare placa_instructor_del_nuevo_competidor int;  
    set placa_instructor_del_nuevo_competidor = (select r.PlacaInstructor  
                                                from Competidor c  
                                                inner join Registrado r on c.  
                                                    NumeroCertificadoGraduacion =  
                                                    r.NumeroCertificadoGraduacion  
                                                where c.  
                                                    NumeroCertificadoGraduacion =  
                                                    NEW.  
                                                    NumeroCertificadoGraduacion);  
  
    set competidores = (select count(*)  
                        from Competidor c  
                        inner join Registrado r on c.NumeroCertificadoGraduacion =  
                            r.NumeroCertificadoGraduacion  
                        where r.PlacaInstructor =  
                            placa_instructor_del_nuevo_competidor);  
  
    set coaches = (select count(*)  
                  from Competidor c  
                  inner join Registrado r on c.NumeroCertificadoGraduacion = r.  
                      NumeroCertificadoGraduacion  
                  where r.PlacaInstructor = placa_instructor_del_nuevo_competidor  
                      );  
  
    if competidores > coaches * 5  
    then  
      signal sqlstate '45000' set message_text = 'No se cumple proporcion 20%  
coaches';  
    end if;  
END$$  
DELIMITER ;
```

— *La graduacion va de 1er dan a 6to dan*

— *Maestro*

```
drop trigger if exists graduacion_maestro_va_de_1_a_6;  
DELIMITER $$  
CREATE TRIGGER graduacion_maestro_va_de_1_a_6
```

```
    BEFORE INSERT ON Maestro
    FOR EACH ROW
BEGIN
    if NEW.Graduacion < 0 or NEW.Graduacion > 6
    then
        signal sqlstate '45000' set message_text = 'La graduacion esta fuera del
            rango valido. Debe ser entre 1ero y 6to dan.';
    end if;
END$$
DELIMITER ;

— Arbitro
drop trigger if exists graduacion_arbitro_va_de_1_a_6;
DELIMITER $$
CREATE TRIGGER graduacion_arbitro_va_de_1_a_6
    BEFORE INSERT ON Arbitro
    FOR EACH ROW
BEGIN
    if NEW.Graduacion < 1 or NEW.Graduacion > 6
    then
        signal sqlstate '45000' set message_text = 'La graduacion esta fuera del
            rango valido. Debe ser entre 1ero y 6to dan.';
    end if;
END$$
DELIMITER ;

— Categoria
drop trigger if exists graduacion_categoria_va_de_1_a_6;
DELIMITER $$
CREATE TRIGGER graduacion_categoria_va_de_1_a_6
    BEFORE INSERT ON Categoria
    FOR EACH ROW
BEGIN
    if NEW.Graduacion < 1 or NEW.Graduacion > 6
    then
        signal sqlstate '45000' set message_text = 'La graduacion esta fuera del
            rango valido. Debe ser entre 1ero y 6to dan.';
    end if;
END$$
DELIMITER ;

— Registrado
drop trigger if exists graduacion_registrado_va_de_1_a_6;
DELIMITER $$
CREATE TRIGGER graduacion_registrado_va_de_1_a_6
    BEFORE INSERT ON Registrado
    FOR EACH ROW
BEGIN
    if NEW.Graduacion < 1 or NEW.Graduacion > 6
    then
        signal sqlstate '45000' set message_text = 'La graduacion esta fuera del
            rango valido. Debe ser entre 1ero y 6to dan.';
```

```
    end if;
END$$
DELIMITER ;
— END La graduacion va de 1ero a 6to dan.

— En cada jurado hay:
— un arbitro con rol 'presidente de mesa'
— un 'arbitro central'
— dos o mas 'jueces'
— tres o mas 'suplentes'.
— La graduacion de cada arbitro debe ser superior a la graduacion de las
  categorias en las que es jurado.
DROP TRIGGER IF EXISTS 'validate_jury_before_arbiting';
DELIMITER $$
CREATE TRIGGER 'validate_jury_before_arbiting' BEFORE INSERT ON 'ArbitradoPor' FOR
  EACH ROW
BEGIN
  IF (select cat.Graduacion
      from Categoria cat
      where cat.IDCategoria = NEW.IDCategoria) >
      (select min(a.Graduacion)
       from Arbitraje ar
       INNER JOIN Arbitro a ON ( a.PlacaArbitro = ar.PlacaArbitro)
       where ar.IDJurado = NEW.IDJurado) then
    signal sqlstate '45000' set message_text = 'La graduacion del arbitro es
      menor que la de la categoria';
  END IF;

  IF (select ifnull(count(*), 0) = 0 from Arbitraje a
      where a.IDJurado = NEW.IDJurado
      and a.Rol = 'Presidente de Mesa') then
    signal sqlstate '45000' set message_text = 'Falta el Presidente de Mesa';
  END IF;

  IF (select ifnull(count(*), 0) = 0 from Arbitraje a
      where a.IDJurado = NEW.IDJurado
      and a.Rol = 'Arbitro Central') then
    signal sqlstate '45000' set message_text = 'Falta el Arbitro Central';
  END IF;

  IF (select ifnull(count(*), 0) < 2 from Arbitraje a
      where a.IDJurado = NEW.IDJurado
      and a.Rol = 'Juez') then
    signal sqlstate '45000' set message_text = 'Faltan jueces';
  END IF;

  IF (select ifnull(count(*), 0) < 3 from Arbitraje a
      where a.IDJurado = NEW.IDJurado
      and a.Rol = 'Suplente') then
    signal sqlstate '45000' set message_text = 'Faltan suplentes';
  END IF;
END;
$$
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS 'check_jury_insertion';
DELIMITER $$
CREATE TRIGGER 'check_jury_insertion' BEFORE INSERT ON 'Arbitraje' FOR EACH ROW
BEGIN
    IF (select ifnull(count(*), 0) > 0 from Arbitraje a
        where a.IDJurado = NEW.IDJurado
            and a.Rol = NEW.Rol
            and NEW.Rol = 'Presidente_de_Mesa') then
        signal sqlstate '45000' set message_text = 'Solo_puede_haber_un_Presidente_de_Mesa_por_jurado';
    END IF;

    IF (select ifnull(count(*), 0) > 0 from Arbitraje a
        where a.IDJurado = NEW.IDJurado
            and a.Rol = NEW.Rol
            and NEW.Rol = 'Arbitro_Central') then
        signal sqlstate '45000' set message_text = 'Solo_puede_haber_un_Arbitro_Central_por_jurado';
    END IF;
END;
$$
DELIMITER ;
```

— El coach de una participacion debe ser de la misma escuela que el competidor  
— Los competidores no deben tener Rol ni NombreEquipo ya que se utilizaran SPs especificos para asignarselos.

```
DROP TRIGGER IF EXISTS 'validate_competitor_empty_role_team';
DELIMITER $$
CREATE TRIGGER 'validate_competitor_empty_role_team' BEFORE INSERT ON 'Competidor'
FOR EACH ROW
BEGIN
    IF (select IF(NombreEquipo is NULL or NombreEquipo = '', false, true) from NEW
        )
        then
            signal sqlstate '45000' set message_text = 'No_deberia_tener_NombreEquipo_durante_insercion';
    END IF;

    IF (select IF(RolEquipo is NULL or RolEquipo = '', false, true) from NEW)
        then
            signal sqlstate '45000' set message_text = 'No_deberia_tener_RolEquipo_durante_insercion';
    END IF;
END;
$$
DELIMITER ;
```

```
insert into Categoria values (null, 0, 10, 11, 'Masculino', 50, 60);
insert into Pais value ('Argentina');
insert into Arbitro values (null, 0, 'Esteban_Quito', 'Argentina');
```



```
insert into Maestro values (null, 'Escuela_TK', 'Esteban_Quito', 3, 'Argentina');
insert into Maestro values (null, 'Escuela_TK', 'Esteban_Quito', 8, 'Argentina');

insert into Maestro values (10, 'Escuela_TK', 'Esteban_Quito', 4, 'Argentina');
insert into Registrado values (null, 'La_foto', 10, 'Armando_Paredes', 10);
```

### 4.3. Restricciones - stored procedure

```
USE mundial_taekwondo;

DROP PROCEDURE IF EXISTS 'agregar_participacion_individual';

DELIMITER $$
CREATE PROCEDURE 'agregar_participacion_individual' (
    'Resultado' VARCHAR(45),
    'NumeroCertificadoGraduacionCoach' INT,
    'NombreModalidad' VARCHAR(45),
    'IDCategoria' INT,
    'NumeroCertificadoGraduacionCompetidor' INT
)
BEGIN
    DECLARE weight_competitor INT;
    DECLARE age_competitor INT;
    DECLARE gender_competitor VARCHAR(45);
    DECLARE graduation_competitor TINYINT(3);
    DECLARE valid boolean DEFAULT false;
    DECLARE last_id_participation INT;
    DECLARE one_participation_per_modality boolean DEFAULT false;
    DECLARE competitor_school VARCHAR(45);
    DECLARE coach_school VARCHAR(45);

    (select m.Escuela into competitor_school
     from Competidor C
     inner join Registrado r on c.NumeroCertificadoGraduacion = r.
        NumeroCertificadoGraduacion
     inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
     where c.NumeroCertificadoGraduacion =
        NumeroCertificadoGraduacionCompetidor);

    (select m.Escuela into coach_school
     from Coach c
     inner join Registrado r on c.NumeroCertificadoGraduacion = r.
        NumeroCertificadoGraduacion
     inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
     where c.NumeroCertificadoGraduacion = NumeroCertificadoGraduacionCoach);

    IF (select ifnull(count(*), 0) = 0
        from Inscripto i
        where i.NumeroCertificadoGraduacion =
            NumeroCertificadoGraduacionCompetidor
            and i.NombreModalidad = NombreModalidad) then
        signal sqlstate '45000' set message_text = 'No_esta_inscripto_en_la_
            modalidad';
```

```

END IF;

select c.Peso, TIMESTAMPDIFF(YEAR, c.FechaNacimiento, CURDATE()), c.Sexo, r.
  Graduacion
  into weight_competitor, age_competitor, gender_competitor,
      graduation_competitor
  from Competidor c, Registrado r
  where NumeroCertificadoGraduacionCompetidor = c.
        NumeroCertificadoGraduacion
        and r.NumeroCertificadoGraduacion = c.
        NumeroCertificadoGraduacion;

select ifnull(count(*), 0) = 0 into one_participation_per_modality
  from Participacion p
  inner join ParticipacionIndividual pi on pi.IDParticipacion = p.
        IDParticipacion
  where p.NombreModalidad = NombreModalidad
        and pi.NumeroCertificadoGraduacionCompetidor =
        NumeroCertificadoGraduacionCompetidor;

select ifnull(count(*), 0) > 0 into valid
  from Categoria c
  where IDCategoria = c.IDCategoria
        and c.Graduacion = graduation_competitor
        and c.Sexo = gender_competitor
        and c.PesoMaximo >= weight_competitor
        and c.PesoMinimo <= weight_competitor
        and c.EdadMaxima >= age_competitor
        and c.EdadMinima <= age_competitor
        and NombreModalidad <> "Equipo"
        and competitor_school = coach_school
        and NumeroCertificadoGraduacionCoach in (select c.
        NumeroCertificadoGraduacion from Coach c
        where c.
        NumeroCertificadoGraduacion
        <>
        NumeroCertificadoGraduacionCompe
        );

IF (valid = true and one_participation_per_modality = true) then
  INSERT INTO 'Participacion' (
    'IDParticipacion',
    'Resultado',
    'IDCategoria',
    'NombreModalidad',
    'NumeroCertificadoGraduacionCoach',
    'Tipo'
  )
  VALUES(
    NULL, — para crear nuevo id
    Resultado,
    IDCategoria,
    NombreModalidad,
    NumeroCertificadoGraduacionCoach,

```

```
        'Individual'
    );
    select LAST_INSERT_ID() into last_id_participation;
    INSERT INTO 'ParticipacionIndividual' (
        'IDParticipacion',
        'NumeroCertificadoGraduacionCompetidor'
    )
    VALUES(
        last_id_participation ,
        NumeroCertificadoGraduacionCompetidor
    );
ELSE
    signal sqlstate '45000' set message_text = 'Combinacion_de_parametros_
        invalida';
END IF;

END;

$$
DELIMITER ;

DROP PROCEDURE IF EXISTS 'create_team';
DELIMITER $$
CREATE PROCEDURE 'create_team' (
    'NombreEquipo' VARCHAR(200)
)
BEGIN
    IF (select ifnull(count(*), 0) = 0
        from Equipo e
        where NombreEquipo = e.NombreEquipo) then
        insert into Equipo(NombreEquipo) VALUES(NombreEquipo);
    ELSE
        signal sqlstate '45000' set message_text = 'Equipo_ya_existe';
    END IF;
END;

$$
DELIMITER ;

DROP PROCEDURE IF EXISTS 'add_to_team';
DELIMITER $$
CREATE PROCEDURE 'add_to_team' (
    'NumeroCertificadoGraduacionCompetidor' INT,
    'RolEquipo' VARCHAR(45),
    'NombreEquipo' VARCHAR(200)
)
BEGIN
    DECLARE gender_team VARCHAR(45);
    DECLARE team_school VARCHAR(45);
    DECLARE competitor_school VARCHAR(45);

    IF (select ifnull(count(*), 0) = 8
        from Competidor c
        where c.NombreEquipo = NombreEquipo)
    then
```

```
        signal sqlstate '45000' set message_text = 'Equipo_lleno';
END IF;

IF (select ifnull(count(*), 0) = 3
      from Competidor c
      where c.NombreEquipo = NombreEquipo
      and RolEquipo = 'Suplente'
      and RolEquipo = c.RolEquipo)
  then
    signal sqlstate '45000' set message_text = 'Suplentes_llenos';
END IF;

IF (select ifnull(count(*), 0) = 5
      from Competidor c
      where c.NombreEquipo = NombreEquipo
      and RolEquipo = 'Titular'
      and RolEquipo = c.RolEquipo)
  then
    signal sqlstate '45000' set message_text = 'Titulares_llenos';
END IF;

IF (select ifnull(count(*), 0) = 0
      from Competidor c
      where c.NombreEquipo = NombreEquipo) then
  set gender_team = (select co.Sexo from Competidor co
                    where co.NumeroCertificadoGraduacion =
                      NumeroCertificadoGraduacionCompetidor);

ELSE
  set gender_team = (select c.Sexo from Competidor C
                    where c.NombreEquipo = NombreEquipo
                    group by c.Sexo);
END IF;

IF (select co.Sexo <> gender_team from Competidor co
      where co.NumeroCertificadoGraduacion =
        NumeroCertificadoGraduacionCompetidor)
  then
    signal sqlstate '45000' set message_text = 'Sexo_invalido';
END IF;

IF (select IF(c.NombreEquipo is NULL or c.NombreEquipo = '', false, true)
      from Competidor c
      where c.NumeroCertificadoGraduacion =
        NumeroCertificadoGraduacionCompetidor)
  then
    signal sqlstate '45000' set message_text = 'Ya_tiene_equipo';
END IF;

IF (select IF(c.RolEquipo is NULL or c.RolEquipo = '', false, true)
      from Competidor c
      where c.NumeroCertificadoGraduacion =
        NumeroCertificadoGraduacionCompetidor)
  then
```

```

        signal sqlstate '45000' set message_text = 'Ya_tiene_rol';
    END IF;

```

— Todos los integrantes de un equipo deben estar inscriptos en la modalidad Combate por Equipos

```

IF (select ifnull(count(*), 0) = 0
    from Inscripto i
    where i.NumeroCertificadoGraduacion =
        NumeroCertificadoGraduacionCompetidor
        and i.NombreModalidad = 'Combate_por_Equipos') then
    signal sqlstate '45000' set message_text = 'No_esta_inscripto_en_la_
        modalidad_Combate_por_Equipos';
END IF;

```

— Todos los integrantes de un equipo deben ser de la misma escuela

```

(select ifnull(m.Escuela, '' ) into team_school
 from Competidor C
 inner join Registrado r on c.NumeroCertificadoGraduacion = r.
     NumeroCertificadoGraduacion
 inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
 where c.NombreEquipo = NombreEquipo
 group by m.Escuela);

```

```

(select m.Escuela into competitor_school
 from Competidor C
 inner join Registrado r on c.NumeroCertificadoGraduacion = r.
     NumeroCertificadoGraduacion
 inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
 where c.NumeroCertificadoGraduacion =
     NumeroCertificadoGraduacionCompetidor);

```

```

IF (team_school <> '' and team_school <> competitor_school) then
    signal sqlstate '45000' set message_text = 'Diferentes_escuelas';
END IF;

```

```

UPDATE 'Competidor' SET
    'Competidor.NombreEquipo' = NombreEquipo,
    'Competidor.RolEquipo' = RolEquipo
WHERE 'Competidor.NumeroCertificadoGraduacion' =
    NumeroCertificadoGraduacionCompetidor;

```

END;

\$\$

DELIMITER ;

— Todos los integrantes de un equipo deben estar inscriptos en la modalidad Combate por Equipos

— Los equipos deben tener 5 integrantes cuyo rol sea titular y 3 cuyo rol sea suplente

— Todos los integrantes de un equipo deben ser de la misma escuela

— Todos los integrantes de un equipo deben ser del mismo género, que debe corresponder con el género de la categoría de todas sus participaciones de equipo.

— Las participaciones de equipo deben ser en la modalidad Combate por Equipos .

```
DROP PROCEDURE IF EXISTS 'agregar_participacion_equipo';
DELIMITER $$
CREATE PROCEDURE 'agregar_participacion_equipo' (
  'NombreEquipo' VARCHAR(200),
  'Resultado' TINYINT(1),
  'IDCategoria' INT,
  'NumeroCertificadoGraduacionCoach' INT
)
BEGIN
  DECLARE coach_school VARCHAR(45);
  DECLARE team_school VARCHAR(45);
  DECLARE last_id_participation INT;

  IF (select ifnull(count(*), 0) = 0
    from Equipo e
    where e.NombreEquipo = NombreEquipo) then
    signal sqlstate '45000' set message_text = 'Equipo_inexistente';
  END IF;

  IF (select ifnull(count(*), 0) < 8
    from Competidor c
    where c.NombreEquipo = NombreEquipo) then
    signal sqlstate '45000' set message_text = 'Equipo_no_lleno';
  END IF;

  IF (select ifnull(count(*), 0) < 3
    from Competidor c
    where c.NombreEquipo = NombreEquipo
    and 'Suplente' = c.RolEquipo) then
    signal sqlstate '45000' set message_text = 'Faltan_suplentes';
  END IF;

  IF (select ifnull(count(*), 0) < 5
    from Competidor c
    where c.NombreEquipo = NombreEquipo
    and 'Titular' = c.RolEquipo) then
    signal sqlstate '45000' set message_text = 'Faltan_titulares';
  END IF;

  (select m.Escuela into coach_school
    from Coach c
    inner join Registrado r on c.NumeroCertificadoGraduacion = r.
      NumeroCertificadoGraduacion
    inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
    where c.NumeroCertificadoGraduacion = NumeroCertificadoGraduacionCoach);

  (select m.Escuela into team_school
    from Competidor C
    inner join Registrado r on c.NumeroCertificadoGraduacion = r.
      NumeroCertificadoGraduacion
```

```
inner join Maestro m on r.PlacaInstructor = m.PlacaInstructor
where c.NombreEquipo = NombreEquipo
group by m.Escuela);

IF (select ifnull(count(*), 0) > 0
    from Competidor c
    where c.NombreEquipo = NombreEquipo
        and c.NumeroCertificadoGraduacion =
            NumeroCertificadoGraduacionCoach) then
    signal sqlstate '45000' set message_text = 'Coach_no_puede_ser_miembro_del
        equipo';
END IF;

IF coach_school <> team_school then
    signal sqlstate '45000' set message_text = 'Coach_no_puede_ser_de_otra_
        escuela';
END IF;

IF (select ifnull(count(*), 0) = 0
    from Categoria cat
    where cat.Sexo in (select c.Sexo from Competidor C
        where c.NombreEquipo = NombreEquipo
            group by c.Sexo)) then
    signal sqlstate '45000' set message_text = 'Sexo_equipo_distinto_del_de_la
        categoria';
END IF;

IF (select ifnull(count(*), 0) > 0
    from Participacion p
    inner join ParticipacionDeEquipo pe on ( p.IDParticipacion = pe.
        IDParticipacion)
    where pe.NombreEquipo = NombreEquipo) then
    signal sqlstate '45000' set message_text = 'Equipo_ya_tiene_participacion'
        ;
END IF;

INSERT INTO 'Participacion' (
    'IDParticipacion',
    'Resultado',
    'IDCategoria',
    'NombreModalidad',
    'NumeroCertificadoGraduacionCoach',
    'Tipo'
)
VALUES(
    NULL, — para crear nuevo id
    Resultado,
    IDCategoria,
    'Combate_por_Equipos',
    NumeroCertificadoGraduacionCoach,
    'Equipo'
);
select LAST_INSERT_ID() into last_id_participation;
```

```

INSERT INTO 'ParticipacionDeEquipo' (
    'IDParticipacion',
    'NombreEquipo'
)
VALUES(
    last_id_participation,
    NombreEquipo
);
END;

$$
DELIMITER ;

```

#### 4.4. Queries

— El listado de inscriptos en cada categoría para el armado de llaves  
 — Es necesario devolver los equipos también?  
 — TODO: agregar una query separada para los equipos o un group by a esta.

```

SELECT Competidor.NumeroCertificadoGraduacion, Inscripto.NombreModalidad,
    Categoria.IDCategoria
FROM Inscripto i
INNER JOIN SeDivideEn sde on sde.NombreModalidad = i.NombreModalidad
INNER JOIN Categoria cat on cat.IDCategoria = sde.IDCategoria
INNER JOIN Competidor comp on comp.NumeroCertificadoGraduacion = i.
    NumeroCertificadoGraduacion
WHERE
    ((cat.PesoMaximo is null and cat.PesoMinimo is null)
    or (cat.PesoMaximo is null and comp.Peso >= cat.PesoMinimo)
    or (comp.Peso < cat.PesoMaximo and comp.Peso >= cat.PesoMinimo)
    or (comp.Peso < cat.PesoMaximo and cat.PesoMinimo is null))
and ((cat.EdadMinima is null and (FLOOR(DATEDIFF(DAY, comp.FechaNacimiento,
    GETDATE())) / 365.25) < cat.EdadMaxima)
    or ((FLOOR(DATEDIFF(DAY, comp.FechaNacimiento, GETDATE())) / 365.25) >= cat.
    EdadMinima and FLOOR(DATEDIFF(DAY, comp.FechaNacimiento, GETDATE())) /
    365.25) < cat.EdadMaxima)))
and comp.Sexo = cat.Sexo
and (cat.Graduacion is null or comp.Graduacion = cat.Graduacion)
order by i.NombreModalidad, cat.IDCategoria;

```

— El país que obtuvo mayor cantidad de medallas de oro, plata y bronce.

```

create view EquiposMaestro as
SELECT c.NombreEquipo, min(m.PlacaInstructor)
FROM Competidor c INNER JOIN m Maestro on m.PlacaInstructor = c.PlacaInstructor
WHERE c.NombreEquipo is not null
GROUP BY c.NombreEquipo;

create view ResultadosMaestro as
SELECT p.Resultado, em.PlacaInstructor
FROM (Participacion p INNER JOIN ParticipacionEquipo e on p.
    IDParticipacion = e.IDParticipacion)
    INNER JOIN EquiposMaestro em on em.NombreEquipo = e.NombreEquipo
union

```



```
SELECT p.Resultado, m.PlacaInstructor
FROM ((Participacion p INNER JOIN ParticipacionIndivIDual i on p.
      IDParticipacion = i.IDParticipacion)
      INNER JOIN Competidor c on i.NumeroCertificadoGraduacion = c.
      NumeroCertificadoGraduacion)
      INNER JOIN Maestro m on c.PlacaInstructor = m.PlacaInstructor;

create view MaestroMedallas as SELECT * FROM
(SELECT ResultadosMaestro.PlacaInstructor, count(ResultadosMaestro.Resultado) as
  bronze
  FROM ResultadosMaestro
  WHERE ResultadosMaestro.Resultado = 3
  GROUP BY ResultadosMaestro.PlacaInstructor) as MaestroBronce
INNER JOIN
(SELECT ResultadosMaestro.PlacaInstructor, count(ResultadosMaestro.Resultado) as
  plata
  FROM ResultadosMaestro
  WHERE ResultadosMaestro.Resultado = 2
  GROUP BY ResultadosMaestro.PlacaInstructor) as MaestroPlata
on MaestroBronce.PlacaInstructor = MaestroPlata.PlacaInstructor
INNER JOIN
(SELECT ResultadosMaestro.PlacaInstructor, count(ResultadosMaestro.Resultado) as
  oro
  FROM ResultadosMaestro
  WHERE ResultadosMaestro.Resultado = 1
  GROUP BY ResultadosMaestro.PlacaInstructor) as MaestroOro
on MaestroOro.PlacaInstructor = MaestroPlata.PlacaInstructor;

create view paisMedallas as
SELECT m.NombrePais, sum(MaestroMedallas.bronce) as bronze, sum(MaestroMedallas.
  plata) as plata, sum(MaestroMedallas.oro) as oro
FROM MaestroMedallas INNER JOIN Maestro on MaestroMedallas.PlacaInstructor =
  Maestro.PlacaInstructor
  GROUP BY Maestro.NombrePais;

SELECT * FROM paisMedallas
  ORDER BY oro desc
  limit 1
union select * from paisMedallas
  ORDER BY plata desc
  limit 1
union select * from paisMedallas
  ORDER BY bronze desc
  limit 1;

— El medallero por Escuela.
create view EscuelaMedallas as
SELECT Maestro.Escuela, sum(MaestroMedallas.bronce) as bronze, sum(MaestroMedallas
  .plata) as plata, sum(MaestroMedallas.oro) as oro
FROM MaestroMedallas INNER JOIN Maestro on MaestroMedallas.PlacaInstructor =
  Maestro.PlacaInstructor
  GROUP BY Maestro.Escuela;

— Sabiendo que las medallas de oro suman 3 puntos, las de plata 2 y las de bronce
```

1

— *punto, se quiere realizar un ranking de puntaje por país y otro por Escuela.*

```
SELECT NombrePais, oro*3 + plata*2 + bronce as puntaje
FROM paisMedallas
ORDER BY puntaje desc;
```

```
SELECT Escuela, oro*3 + plata*2 + bronce as puntaje
FROM EscuelaMedallas
ORDER BY puntaje desc;
```

— *Dado un Competidor, la lista de categorías donde haya participado y el Resultado obtenido.*

```
DROP PROCEDURE IF EXISTS 'participaciones_competidor';
DELIMITER $$
CREATE PROCEDURE 'participaciones_competidor' (
  'NumeroCertificadoGraduacion' INT
) BEGIN
  SELECT p.IDCategoria, p.Resultado
  FROM ParticipacionIndividual pi INNER JOIN Participacion p ON p.
    IDParticipacion = pi.IDParticipacion
  WHERE pi.NumeroCertificadoGraduacion = NumeroCertificadoGraduacion
  UNION
  SELECT p.IDCategoria, p.Resultado
  FROM ParticipacionDeEquipo pe INNER JOIN Participacion p ON p.IDParticipacion
    = pe.IDParticipacion
    INNER JOIN Competidor c on c.nombreEquipo = pe.nombreEquipo
  WHERE c.NumeroCertificadoGraduacion = NumeroCertificadoGraduacion;
END;
$$
DELIMITER ;
```

— *El listado de los árbitros por país.*

```
SELECT *
FROM arbitro
ORDER BY NombrePais desc;
```

— *La lista de todos los árbitros que actuaron como árbitro central en las modalidades de combate*

```
SELECT a.PlacaArbitro
FROM
  Arbitraje a INNER JOIN ArbitradoPor ap ON a.IDJurado = ap.IDJurado
WHERE ap.NombreModalidad = 'Combate' or ap.NombreModalidad = 'Combate_por_Equipos'
and a.Rol = 'arbitro_central';
```

— *La lista de equipos por país.*

```
SELECT EquiposMaestro.Equipo, Maestro.NombrePais
FROM EquiposMaestro INNER JOIN Maestro on EquiposMaestro.PlacaInstructor = Maestro
  .PlacaInstructor
```

**ORDER BY** Maestro.NombrePais **desc**;

## 5. Conclusión