Trabajo Práctico Base de Datos

TaekwonDo

Primer cuatrimestre del 2017
5 de Mayo de 2017

Integrantes	L.U.	Correo electrónico
Coy, Camila	033/14	camicoy94@gmail.com
Costa, Manuel José Joaquín	35/14	manucos94@gmail.com
Maddonni, Axel Ezequiel	200/14	axel.maddonni@gmail.com
Rabinowicz, Lucía	105/14	lu.rabinowicz@gmail.com

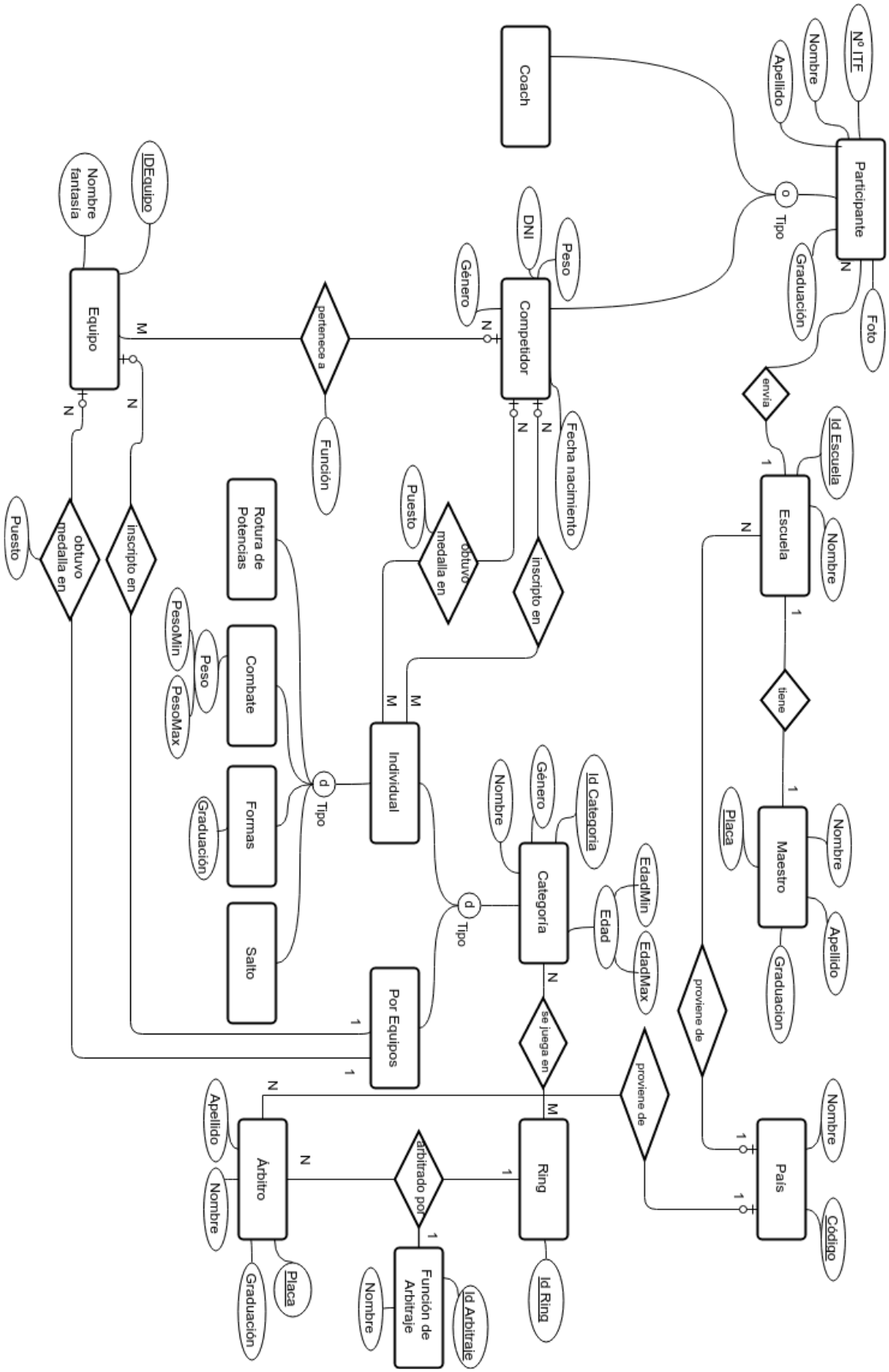
DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Contenidos

1- Introducción

En este trabajo buscaremos brindar una solución al problema de las inscripciones en el torneo mundial de TaekwonDo. En este contexto, al tratarse de un deporte muy complejo y con muchas subdivisiones en diversas categorías, permitir o no ciertas inscripciones, el armado de llaves y la asignación de árbitros son algunos de los problemas a la hora de organizar la competencia.
Además muchas veces es deseable guardar información adicional como por ejemplo, ganadores de medallas, escuelas a las que pertenecen los diferentes competidores (tanto en individuales como en equipos), Coachs que participan en el ring, etc.
Para todo ello diseñaremos una base de datos en la cual toda esta informacion se verá reflejada y podrá ser consultada.
También se realizarán algunas consultas en lenguaje SQL para demostrar que este modelo relacional efectivamente funciona.

2- Modelo Entidad Relación



2-1- Restricciones

- En las relaciones MedallasIndividuales y MedallasEquipo los puestos van del 1 al 3.
- Cada 5 competidores pertenecientes a la misma escuela debe haber un coach de esa escuela.
- Para todo competidor inscripto en rotura de potencia, combate, formas y salto el peso, género, graduación y fecha de nacimiento (a partir de la cual se puede obtener la edad) son acordes a las de la modalidad en la que esté inscripto.
- Todos los competidores de un mismo equipo deben pertenecer a la misma escuela.
- En cada equipo debe haber cinco titulares y tres suplentes.
- En cada ring se cuenta con por lo menos un presidente de mesa, un árbitro central, más de un juez y al menos tres suplentes.
- Para todo árbitro en el ring su graduación debe ser mayor que el máximo las graduaciones de todas las categorías que se juegan en ese ring.

3- Modelo Relacional

-- Pais (codigo_pais, nombre)
PK = CK = {codigo_pais}

-- Maestro (placa, nombre, apellido, graduacion, codigo_pais)
PK = CK = {placa}
FK = {codigo_pais}

-- Escuela (id_escuela, nombre, placa_maestro)
PK = CK = {id_escuela}
FK = {placa_maestro}

--Participante (id_itf, nombre, apellido, graduacion, foto, id_escuela, tipo)
PK = CK = {id_itf}
FK = {id_escuela}

-- Coach (itf)
PK = CK = {itf}
FK = {itf}

-- Competidor (id_itf, dni, genero, fecha_nacimiento, peso)
PK = {id_itf}, CK = {id_itf, dni}
FK = {id_itf}

-- Equipo (id_equipo, nombre)
PK = CK = {id_equipo}

-- EquipoCompetidores (id_equipo, id_competidor, funcion)
PK = CK = {(id_equipo, id_competidor)}
FK = {id_equipo, id_competidor}
Aclaración: el atributo función indica con ‘T’ o ‘S’ si se trata de un titular o suplente, respectivamente.

-- Categoria (id_categoria, nombre, sexo, edadMin, edadMax, tipo)
PK = CK = {id_categoria}
Aclaración: El atributo tipo indica con ‘I’ o ‘E’ si se trata de una categoría individual o por equipos, respectivamente.

-- CategoriaIndividual (id_categoria, tipo)
PK = CK = {id_categoria}
FK = {id_categoria}
Aclaración: El atributo tipo indica con ‘R’, ‘C’, ‘F’, ‘S’, si se trata de una categoría de Rotura de Potencia, Combate, Formas o Salto, respectivamente.

-- CategoriaFormasIndiviual (id_categoria, graduacion)
PK = CK = {id_categoria}
FK = {id_categoria}

-- CategoriaCombateIndividual (id_categoria, pesoMin, pesoMax)
PK = CK = {id_categoria}
FK = {id_categoria}

-- CategoriaPorEquipos (id_categoria)
PK= CK = {id_categoria}
FK = {id_categoria}

-- CategoriaRoturaPotenciaIndividual (id_categoria)
PK = CK = {id_categoria}
FK = {id_categoria}

-- CategoriaSaltoIndividual (id_categoria)
PK = CK = {id_categoria}
FK = {id_categoria}

-- InscripcionesIndividuales (id_competidor, id_categoria)
PK = CK = {(id_competidor, id_categoria)}
FK = {id_competidor, id_categoria}

-- MedallasIndividuales (id_competidor, id_categoria, puesto)
PK = CK = {(id_competidor, id_categoria)}
FK = {id_competidor, id_categoria}

-- InscripcionesEquipos (id_equipo, id_categoria)
PK= CK = {(id_equipo, id_categoria)}
FK = {id_equipo, id_categoria}

-- MedallasEquipos (id_equipo, id_categoria, puesto)
PK = CK = {id_equipo, id_categoria}
FK = {id_equipo, id_categoria}

-- Ring (id_ring)
PK = CK = {id_ring}

-- RingCategoria (id_ring, id_categoria)
PK = CK = FK ={id_ring, id_categoria}

-- Arbitro (placa, nombre, apellido, graduacion, codigo_pais)

PK = CK = {placa}
FK = {codigo_pais}

-- FuncionArbitraje (id_funcion_arbitraje, nombre)
PK = CK = {id_funcion_arbitraje}

-- RingConsejoArbitros (id_ring, placa_arbitro, id_funcion_arbitraje)
PK = CK = {placa_arbitro}
FK = {id_ring, placa_arbitro, id_funcion_arbitraje}

4- Diseño

Queries en sql de creación y foreign keys por cada tabla.

5- Funcionalidades

- El listado de inscriptos en cada categoría para el armado de llaves

```
-- Categorías Individuales

SELECT i.id_categoria,
      p.id_itf,
      p.nombre,
      p.apellido
FROM InscripcionIndividual i
JOIN Participante p ON i.id_itf_competidor = p.id_itf;

-- Categorías por Equipos

SELECT i.id_categoria,
      i.id_equipo,
      e.nombre
FROM InscripcionEquipo i
JOIN Equipo e ON i.id_equipo = e.id_equipo;
```

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

```
-- Mayor cantidad de oro

SELECT medallas_oro.nombre,
      count(medallas_oro.nombre) AS cantidad
FROM (
       (SELECT pa.nombre
        FROM MedallaIndividual mi
        JOIN Participante p ON mi.id_itf_competidor = p.id_itf
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
        JOIN Pais pa ON esc.codigo_pais = pa.codigo
        WHERE mi.puesto = 1)
      UNION ALL
       (SELECT pa.nombre
        FROM MedallaEquipo me
        JOIN Equipo e ON me.id_equipo = e.id_equipo
        LEFT OUTER JOIN EquipoCompetidor ec ON e.id_equipo = ec.id_equipo
        JOIN Participante p ON ec.id_itf_competidor = p.id_itf
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
```

```
        JOIN Pais pa ON pa.codigo = esc.codigo_pais
        WHERE me.puesto = 1)) medallas_oro
GROUP BY nombre
ORDER BY cantidad DESC LIMIT 1;


-- Mayor cantidad de plata

SELECT medallas_oro.nombre,
      count(medallas_oro.nombre) AS cantidad
FROM (
       (SELECT pa.nombre
        FROM MedallaIndividual mi
        JOIN Participante p ON mi.id_itf_competidor = p.id_itf
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
        JOIN Pais pa ON esc.codigo_pais = pa.codigo
        WHERE mi.puesto = 2)
      UNION ALL
       (SELECT pa.nombre
        FROM MedallaEquipo me
        JOIN Equipo e ON me.id_equipo = e.id_equipo
        LEFT OUTER JOIN EquipoCompetidor ec ON e.id_equipo = ec.id_equipo
        JOIN Participante p ON ec.id_itf_competidor = p.id_itf
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
        JOIN Pais pa ON pa.codigo = esc.codigo_pais
        WHERE me.puesto = 2)) medallas_oro
GROUP BY nombre
ORDER BY cantidad DESC LIMIT 1;

-- Mayor cantidad de bronce

SELECT medallas_oro.nombre,
      count(medallas_oro.nombre) AS cantidad
FROM (
       (SELECT pa.nombre
        FROM MedallaIndividual mi
        JOIN Participante p ON mi.id_itf_competidor = p.id_itf
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
        JOIN Pais pa ON esc.codigo_pais = pa.codigo
        WHERE mi.puesto = 3)
      UNION ALL
       (SELECT pa.nombre
        FROM MedallaEquipo me
        JOIN Equipo e ON me.id_equipo = e.id_equipo
        LEFT OUTER JOIN EquipoCompetidor ec ON e.id_equipo = ec.id_equipo
        JOIN Participante p ON ec.id_itf_competidor = p.id_itf
```

```
        JOIN Escuela esc ON p.id_escuela = esc.id_escuela
        JOIN Pais pa ON pa.codigo = esc.codigo_pais
        WHERE me.puesto = 3)) medallas_oro
GROUP BY nombre
ORDER BY cantidad DESC LIMIT 1;
```

- Ranking por puntaje

```
-- Ranking de puntaje por escuela

SELECT e.nombre, sum(Puntuacion) as total
FROM (MedallasIndividuales m JOIN Participante par ON par.id_itf = m.id_itf_competidor)
as mpar JOIN Escuela e ON mpar.id_escuela = e.id_escuela
CASE m.puesto
   WHEN 1 THEN 3
   WHEN 2 THEN 2
   ELSE 1
END as Puntuacion
GROUP BY e.nombre
ORDER BY total

-- Ranking de puntaje por pais

SELECT pa.nombre, sum(Puntuacion) as total
FROM ((MedallasIndividuales m JOIN Participante par ON par.id_itf = m.id_itf_competidor)
as mcpar JOIN Escuela e ON mcpar.id_escuela = e.id_escuela) as mce JOIN Pais pa ON
mce.codigo_pais = pa.codigo
CASE m.puesto
   WHEN 1 THEN 3
   WHEN 2 THEN 2
   ELSE 1
END as Puntuacion
GROUP BY pa.nombre
ORDER BY total
```

- Lista de categorías donde haya participado y el resultado obtenido

```
SELECT p.nombre, cat.nombre, m.puesto
FROM ((InscripcionesIndividuales i LEFT OUTER JOIN MedallasIndividuales m ON
m.id_itf_competidor = i.id_itf_competidor AND m.id_categoria = i.id_categoria) as im JOIN
Participante p ON p.id_itf = im.id_itf_competidor) as imc JOIN Categoria cat ON
cat.id_categoria = imc.id_categoria
```

- Medallero por escuela

```
SELECT e.nombre, par.nombre, par.apellido, m.puesto, cat.nombre
```

```
FROM ((MedallasIndividuales m JOIN Participante par ON par.id_itf = m.id_itf_competidor)
as cmpar JOIN Escuela e ON cmpar.id_escuela = e.id_escuela) as cms JOIN Categoria cat
ON cms.id_categoria = cat.id_categoria
ORDER BY e.nombre
```

- Listado de los árbitros por país

```
SELECT p.nombre, a.nombre, a.apellido
FROM Arbitro a JOIN Pais p ON a.codigo_pais = p.codigo
ORDER BY p.nombre
```

- Lista de todos los árbitros que actuaron como árbitro central en las modalidades de combate

```
SELECT a.nombre, a.apellido
FROM ((Arbitro a JOIN RingConsejoArbitros r ON a.placa = r.placa_arbitro) as ar JOIN
RingCategoria c ON ar.id_ring = c.id_ring) as arc JOIN FuncionArbitraje f ON
f.id_funcion_arbitraje = arc.id_funcion_arbitraje as arcf
WHERE (c.id_categoria IN CategoriaCombateIndividual OR c.id_categoria IN (SELECT
equi.id_categoria FROM Categoria c JOIN CategoriaPorEquipo equi ON c.id_categoria =
equi.id_categoria WHERE c.nombre LIKE '%Combate%')) AND f.nombre = 'Central'
```

- Lista de equipos por país

```
SELECT p.nombre, e.nombre
FROM (((Equipo e JOIN EquipoCompetidores ec ON e.id_equipo = ec.id_equipo) as eec
JOIN Participante par ON par.id_itf = eec.id_itf_competidor) as eeccpar JOIN Escuela es
ON es.id_escuela = eeccpar.id_escuela) as eeccesm JOIN Pais p ON p.codigo =
eeccesm.codigo_pais
ORDER BY p.nombre
```

6- Implementación de Restricciones

- En las relaciones MedallaIndividual y MedallaEquipo los puestos van del 1 al 3.

Para este caso se implementó un stored procedure para cargar los resultados para cada competencia que chequea los datos ingresados:

```
DELIMITER $$
CREATE PROCEDURE `CargarResultadosCategoriaIndividual`(
        id_categoria int2,
        itf_competidor_puesto1 int(10),
        itf_competidor_puesto2 int(10),
        itf_competidor_puesto3 int(10)
)
BEGIN
    INSERT INTO MedallaIndividual VALUES (itf_competidor_puesto1, id_categoria, 1);
    INSERT INTO MedallaIndividual VALUES (itf_competidor_puesto2, id_categoria, 2);
    INSERT INTO MedallaIndividual VALUES (itf_competidor_puesto3, id_categoria, 3);
END
$$
DELIMITER ;

DELIMITER $$
CREATE PROCEDURE `CargarResultadosCategoriaEquipos`(
        id_categoria int2,
        id_equipo_puesto1 int(10),
        id_equipo_puesto2 int(10),
        id_equipo_puesto3 int(10)
)
BEGIN
```

```
    INSERT INTO MedallaEquipo VALUES (id_equipo_puesto1, id_categoria, 1);
    INSERT INTO MedallaEquipo VALUES (id_equipo_puesto2, id_categoria, 2);
    INSERT INTO MedallaEquipo VALUES (id_equipo_puesto3, id_categoria, 3);
END
$$
```

- Cada 5 competidores pertenecientes a la misma escuela debe haber un coach de esa escuela.

Para asegurar esto, agregamos un trigger al agregar un competidor que chequea si la cantidad de coachs de la escuela del competidor cumple la restricción.

```
DELIMITER $$
CREATE TRIGGER chequeo_coach_cada_5_competidores
    BEFORE INSERT ON Competidor FOR EACH ROW
    BEGIN
        DECLARE id_escuela_nuevo_competidor INT;
        SELECT id_escuela INTO id_escuela_nuevo_competidor FROM Participante WHERE id_itf = NEW.id_itf;
        IF ((1 + (SELECT COUNT(c.id_itf) FROM Competidor c join Participante p on c.id_itf = p.id_itf where p.id_escuela = id_escuela_nuevo_competidor)) div 5 > (SELECT COUNT(c.id_itf) FROM Coach c
            JOIN Participante p on c.id_itf = p.id_itf WHERE p.id_escuela = id_escuela_nuevo_competidor))
        THEN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'No se puede agregar otro competidor sin agregar otro Coach de la misma escuela';
        END IF;
    END
$$
```

- Para todo competidor inscripto en rotura de potencia, combate, formas y salto el peso, género, graduación y fecha de nacimiento (a partir de la cual se puede obtener la edad) son acordes a las de la modalidad en la que esté inscripto.

```
DELIMITER $$

CREATE TRIGGER `inscripto_satisface_requisitos_categoria` BEFORE INSERT ON
`InscripcionesIndividuales` FOR EACH ROW

BEGIN

  IF  (SELECT comp.genero FROM Competidor comp WHERE comp.id_itf =
NEW.id_competidor) !=

        (SELECT cat.sexo FROM Categoria cat WHERE cat.id_categoria =
NEW.id_categoria)

  THEN

        SIGNAL sqlstate '45000'

        SET message_text = 'El género del competidor debe coincidir con el de la
categoría.';

  END IF;


  IF  EXISTS (SELECT null FROM CategoriaCombateIndividual cat WHERE cat.id_categoria
= NEW.id_categoria) AND

        NOT(((SELECT comp.peso FROM Competidor comp WHERE comp.id_itf =
NEW.id_competidor) BETWEEN

        (SELECT cat.pesoMin FROM CategoriaCombateIndividual cat WHERE
cat.id_categoria = NEW.id_categoria) AND

        (SELECT cat.pesoMax FROM CategoriaCombateIndividual cat WHERE
cat.id_categoria = NEW.id_categoria)))

        THEN

        SIGNAL sqlstate '45000'

        SET message_text = 'El peso del competidor debe estar en el rango de la
categoría.';

  END IF;


  IF  EXISTS (SELECT null FROM CategoriaFormasIndividual cat WHERE cat.id_categoria
= NEW.id_categoria) AND

        (SELECT comp.graduacion FROM Competidor comp WHERE comp.id_itf =
NEW.id_competidor) !=
```

```
        (SELECT cat.graduacion FROM CategoriaFormasIndividual cat WHERE
cat.id_categoria = NEW.id_categoria)

        THEN

        SIGNAL sqlstate '45000'

        SET message_text = 'La graduación del competidor debe coincidir con la de la
categoría de formas.';

  END IF;


  IF  NOT((SELECT DATEDIFF(comp.fecha_nacimiento, '2017-10-01') / 365.25 AS age
FROM Competidor comp WHERE comp.id_itf = NEW.id_competidor) BETWEEN

        (SELECT comp.edadMin FROM Categoria cat WHERE cat.id_categoria =
NEW.id_categoria) AND

        (SELECT comp.edadMax FROM Categoria cat WHERE cat.id_categoria =
NEW.id_categoria))

  THEN

        SIGNAL sqlstate '45000'

        SET message_text = 'La edad del competidor debe estar dentro del rango de la
categoría.';

  END IF;


END$$

DELIMITER ;
```

- Todos los competidores de un mismo equipo deben pertenecer a la misma escuela.
- En cada equipo debe haber cinco titulares y tres suplentes.

Ambas restricciones se verifican usando un mismo stored procedure para inscribir a todo un equipo.

```
DELIMITER $$

CREATE PROCEDURE `InscribirEquipo`(

        nombre varchar(128),

        id_itf_titular1 int(10),

        id_itf_titular2 int(10),
```

```
        id_itf_titular3 int(10),
        id_itf_titular4 int(10),
        id_itf_titular5 int(10),
        id_itf_suplente1 int(10),
        id_itf_suplente2 int(10),
        id_itf_suplente3 int(10)
)
BEGIN
    DECLARE id_nuevo_equipo INT;
    IF (SELECT COUNT(distinct(id_escuela)) FROM Participante WHERE id_itf IN
(id_itf_titular1, id_itf_titular2, id_itf_titular3, id_itf_titular4, id_itf_titular5, id_itf_suplente1,
id_itf_suplente2, id_itf_suplente3) ) > 1
    THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Los integrantes del equipo no pertenecen a la misma
escuela';
    END IF;
    INSERT INTO Equipo VALUES (default, nombre);
    SELECT max(id_equipo) INTO id_nuevo_equipo FROM Equipo;
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_titular1, 'T');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_titular2, 'T');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_titular3, 'T');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_titular4, 'T');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_titular5, 'T');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_suplente1, 'S');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_suplente2, 'S');
    INSERT INTO EquipoCompetidor VALUES (id_nuevo_equipo, id_itf_suplente3, 'S');
END
$$
```

- En cada ring se cuenta con por lo menos un presidente de mesa, un árbitro central, más de un juez y al menos tres suplentes.

```
DELIMITER $$

CREATE PROCEDURE `esRingCompleto`(

  id_ring1 SMALLINT(5)

)

BEGIN

        DECLARE id_nuevo_equipo INT;

        IF NOT (EXISTS (SELECT null FROM RingConsejoArbitros rca WHERE id_ring1 =
rca.id_ring AND rca.id_funcion_arbitraje = 0) AND

        EXISTS (SELECT null FROM RingConsejoArbitros rca WHERE id_ring1 =
rca.id_ring AND rca.id_funcion_arbitraje = 1) AND

        (SELECT count(*) FROM RingConsejoArbitros rca WHERE id_ring1 = rca.id_ring
AND rca.id_funcion_arbitraje = 2) > 1 AND

        (SELECT count(*) FROM RingConsejoArbitros rca WHERE id_ring1 = rca.id_ring
AND rca.id_funcion_arbitraje = 2) >= 3)

        THEN

        SIGNAL SQLSTATE '45000'

                SET MESSAGE_TEXT = 'El ring es inválido pues el consejo arbitral está
incompleto.';

        END IF;

END

$$

DELIMITER ;
```

- Para todo árbitro en el ring su graduación debe ser mayor que el máximo las graduaciones de todas las categorías que se juegan en ese ring.

```
DELIMITER $$
CREATE TRIGGER graduacion_arbitro_es_suficiente
        BEFORE INSERT ON RingConsejoArbitros FOR EACH ROW
        BEGIN
        IF (SELECT a.graduacion FROM Arbitro a WHERE a.placa = NEW.placa_arbitro) <=
(SELECT cfi.graduacion FROM (RingCategoria ra INNER JOIN CategoriaFormasIndividual
cfi ON ra.id_categoria = cfi.id_categoria) racfi WHERE NEW.id_ring = racfi.id_ring)
        THEN
        SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'Un árbitro no puede arbitrar una categoría de
formas de graduación mayor a la suya.';
        END IF;
        END
```

$$
DELIMITER ;

7- Conclusiones

Luego de realizar el trabajo pudimos observar la gran utilidad de generar un modelo de entidad relacional óptimo, ya que tanto la información repetida como las relaciones entre entidades innecesarias generan grandes repercusiones en el modelo relacional y luego en las consultas a la base de datos.
También notamos la importancia de tener un enunciado claro y conciso, hecho que en la realidad no sucede cuando se trata de un cliente que no tiene conocimiento de las dificultades de diseñar una base de datos adecuada.