Universidad de Buenos Aires
Facultad de
Ciencias Exactas y Naturales
Departamento de Computación

Base de Datos

Segundo Cuatrimestre de 2012

Trabajo práctico 1

Campeonato Sudamericano de Básquet.

Grupo 2

Integrante	LU	Correo electrónico
Cammi, Martín	676/02	martincammi@gmail.com
De Sousa, Mariano	389/08	marian_sabianaa@hotmail.com
Méndez, Gonzálo	843/04	gemm83@hotmail.com
Serapio, Noelia	871/03	noeliaserapio@gmail.com

Índice

1. Tecnología utilizada en la implementación

Se utilizó el motor de base de datos MySql, el cual es open source y su utilización es simple.

Durante el desarrollo se observó la imposibilidad que posee el motor de implementar check constraints. Para poder simular esta funcionalidad se utilizaron triggers, uno previo a la inserción de un nuevo registro y otro previo a la actualización para las constraints adicionales así como también para las que se desprendieron del modelo, que también fueron incluídas en los triggers. Era deseable que si la check constraint fallara, el insert o update se vea cancelado, y ningún dato sea persistido.

La forma que se encontró para emular este comportamiento es llamando a un procedimiento almacenado inexistente. Se produce así una excepción y el motor efectúa un rollback de los cambios efectuados en el trigger y cancela la operación de inserción o actualización. Tuvimos que hacer esto por la imposibilidad del motor de contar con una sentencia para mostrar un mensaje.

Se agregó en la base una tabla de LOG, la cual no se impacta en el modelo relacional. Poseía inicialmente como objetivo la persistencia de un log sobre las inconsistencias generadas en los triggers. El inconveniente que surgía es que al llamar a un procedimiento inexistente para cancelar la finalización de un triggers, se efectuaba rollback sobre todos los cambios efectuados dentro del trigger, incluso los de la tabla de LOG (de hacerse un insert o update). Como alternativa se optó por loguear las ejecuciones exitosas y visualizar por línea de comandos las fallidas. Al tener varias condiciones a validar por trigger, estos se tornaron muy largos. La lógica más compleja fue extraída en procedimientos almacenados. Cada uno de ellos efectúa una llamada a uno inexistente si encuentra un error.

Otro problema que encontramos fue que las claves foráneas no estaban habilitadas, investigando al respecto encontramos que para que funcionen las tablas deben estar creadas con el parámetro ENGINE en tipo InnoDB.

Estas fueron algunas complicaciones con las que nos encontramos al trabajar con este motor de base de datos.

2. Detalles sobre el entregable

A continuación se detalla el contenido del entregable, así como su correcta utilización.

Directorios:

- **queries:**

  create.sql: Contiene la creación de las tablas, claves primarias y foráneas. Se encarga automáticamente de llamar a sps.sql, business_constraints.sql, vistas.sql y data.sql (que carga los datos) en ese orden.

  NOTA: El nombre de la base de datos utilizada es db_tp1. Considerar a la hora de correr este script que se borra la base de datos antes mencionada. Validar antes de correrlo que no exista previamente una base con ese nombre en el servidor a correr ya que se podría perder información.

  sps.sql: Se encarga de llamar a todos los store procedures dentro del directorio sps y ejecutar su creación.

  business_constraints.sql: Se encarga de llamar a todos los archivos que generan los triggers incluídos dentro del directorio llamado constraints. vistas.sql: ejecuta la creación de los procedimientos almacenados para el análisis de los datos de manera más simple y ordenada, si bien no son vistas propiamente dichas. Algunas vistas disponibles:

  partidosDeFase(idFase): muestra todos los partidos de esa fase.

  partidosDeFaseArbitro(idFase): muestra todos los partidos de esa fase y los árbitros asignados para cada uno de ellos.

  Las fases válidas son $1, 2, 3, 4, 5$ correspondientes a las fases
  FASE_GRUPO, 5TO_PUESTO, SEMIFINAL, 3ER_PUESTO, FINAL respectivamente.

  data.sql: Efectúa la carga de datos, cumpliendo con las restricciones ya inicializadas.

  breakers.sql: Tiene como objeto invalidar las restricciones que se agregaron a la base. Su ejecución deberia mostrar los mensajes de error de cada constraint.

  queries/contraints: Directorio que contiene todas las constraints implementadas a través de triggers, algunos de estos archivos, ejecutan procedimientos almacenados.

  queries/sps: Directorio que contiene todos los procedimientos almacenados para constraints subdivididos por entidad. Posee además dos archivos: business_sps.sql y sp_log_ok.sql. El primero de ellos posee los procedimientos para la validación del árbitro requeridos por enunciado. El segundo posee el procedimiento de mantenimiento de la tabla LOG.

  NOTA: Existe un directorio adicional llamado generics que posee procedimientos de uso común en varios triggers.

3. Utilización

Utilización: A continuación se detalla la utilización de la base para un sistema operativo Linux, teniendo instalado Mysql server (para Ubuntu, el comando para descargarlo es sudo-apt get install mysql-server).

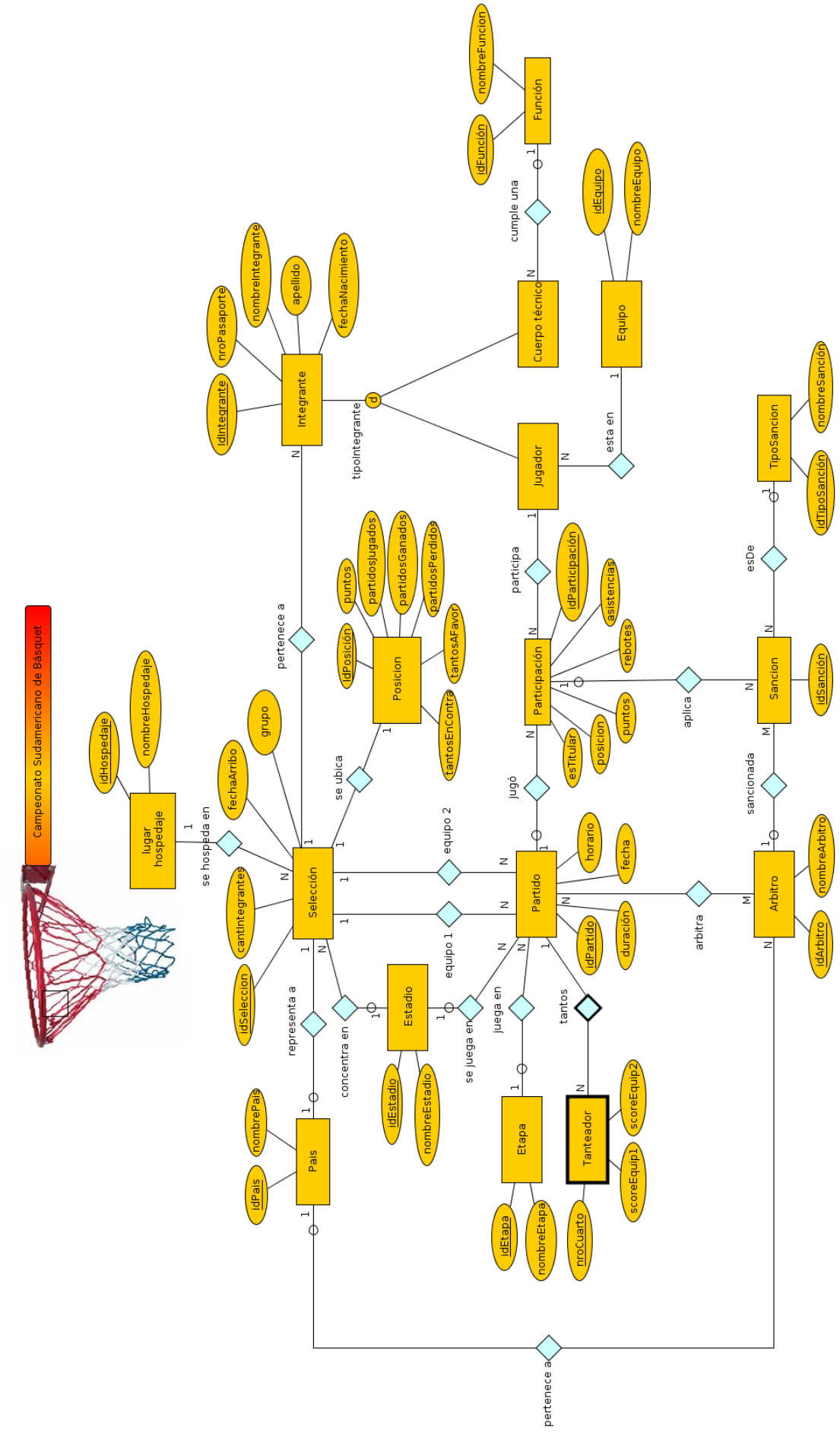
- Abrir una terminal.
- Posicionarse sobre el directorio queries.
- Escribir en la terminal: mysql -uroot -p[contraseña] (contraseña definida en la instalación del servidor).
- Ejecutar dentro del mysql server el comando source create.sql.

Se crearán la base de datos, sus tablas, triggers y store procedures. Las tablas serán cargadas con información y estarán listas para ser consultadas.

Comandos útiles: (mysql server en Linux)

- **show databases** : Muestra las bases de datos existentes en el servidor.
- **use [db_name]** : Setea a db_name como base de datos a utilizar.
- **desc [table_name]**: Muestra el detalle de la tabla table_name
- **source [path]/file.sql**: Corre el script incluído dentro del archivo file
- **exit:** Cierra el mysql server.

4. Modelo Entidad-Relacion



5. Modelo Logico Relacional

5.1. Entidad Selección

SELECCION(idSeleccion, hospedaHospedaje, representaPais, concentraEstadio, ubicaPosicion, cantIntegrantes, fechaArribo, grupo)

PK = { (idSeleccion) }
CC = { (idSeleccion) }
FK = { (hospedaHospedaje), (representaPais), (concentraEstadio), (ubicaPosicion) }

References:

SELECCION.hospedaHospedaje debe estar en LUGARHOSPEDAJE.idHospedaje.
SELECCION.representaPais debe estar en PAIS.idPais.
SELECCION.concentraEstadio debe estar en ESTADIO.idEstadio.
SELECCION.ubicaPosicion debe estar en POSICION.idPosicion.
SELECCION.idSeleccion debe estar en INTEGRANTE.perteneceSeleccion.
SELECCION.idSeleccion debe estar en PARTIDO.equipoSeleccion1.
SELECCION.idSeleccion debe estar en PARTIDO.equipoSeleccion2.

SELECCION.hospedaHospedaje no puede ser nulo.
SELECCION.representaPais no puede ser nulo.
SELECCION.concentraEstadio no puede ser nulo.
SELECCION.ubicaPosicion no puede ser nulo.

Constraints:

SELECCION.grupo == “A“ no puede repetirse más de 4 veces.
SELECCION.grupo == “B“ no puede repetirse más de 4 veces.
SELECCION.grupo solo puede ser “A“ o “B“.
SELECCION.fechaArribo <= PARTIDO.fecha.
SELECCION.cantIntegrantes debe ser igual a la cantidad de integrantes relacionados.

5.2. Entidad Posición

POSICION(idPosicion, puntos, partidosJugados, partidosGanados, partidosPerdidos, tantosAFavor, tantosEnContra)

PK = { (idPosicion) }
CC = { (idPosicion) }
FK = {}

References:

POSICION.idPosicion debe estar en SELECCION.ubicaPosicion.

Constraints:

POSICION.puntos >= 0.
POSICION.partidosJugados >= 0.
POSICION.partidosGanados >= 0.
POSICION.partidosPerdidos >= 0.
POSICION.partidosJugados == POSICION.partidosGanados + POSICION.partidosPerdidos.
POSICION.tantosAFavor >= 0.
POSICION.tantosEnContra >= 0.

5.3. Entidad LugarHospedaje

LUGARHOSPEDAJE(idHospedaje, nombreHospedaje)

PK = { (idHospedaje) }
CC = { (idHospedaje) }
FK = {}

References:

LUGARHOSPEDAJE.idHospedaje debe estar en SELECCION.hospedaHospedaje.

Constraints:

Ninguna.

5.4. Entidad País

PAIS(idPais, nombrePais)

PK = { (idPais) }
CC = { (idPais) }
FK = {}

References:

PAIS.idPais puede no estar en SELECCION.representaPais.

PAIS.idPais puede no estar en ARBITRO.pertenecePais.

Constraints:

Ninguna.

5.5. Entidad Integrante

INTEGRANTE(idIntegrante, perteneceSeleccion, nroPasaporte, nombreIntegrante, apellido, fechaNacimiento,tipoIntegrante)

PK = { (idIntegrante) }
CC = { (idIntegrante), (nroPasaporte) }
FK = { (perteneceSeleccion) }

References:

INTEGRANTE.perteneceSeleccion debe estar en SELECCION.idSeleccion.

INTEGRANTE.idIntegrante puede estar en JUGADOR.idJugador o (exclusivo) CUERPOTECNICO.idCuerpoTecnico.

INTEGRANTE.perteneceSeleccion no puede ser nulo.

Constraints:

INTEGRANTE.tipoIntegrante IN { “Jugador“, “CuerpoTecnico“ } .

AÑO(SYSDATE) − AÑO(INTEGRANTE.fechaNacimiento) >= 18.

5.6. Entidad Jugador

JUGADOR(idJugador, estaEnEquipo)

PK = { (idJugador) }
CC = { (idJugador) }
FK = { (estaEnEquipo), (idJugador) }

References:

JUGADOR.idJugador debe estar en INTEGRANTE.idIntegrante.
JUGADOR.estaEnEquipo debe estar en EQUIPO.idEquipo.
JUGADOR.idJugador debe estar en PARTICIPACION.participaJugador.

JUGADOR.estaEnEquipo no puede ser nulo.

Constraints:

Por jugador, tiene que haber una sola participación en un partido.

5.7. Entidad CuerpoTécnico

CUERPOTECNICO(idCuerpoTecnico, cumpleFuncion)

PK = { (idCuerpoTecnico) }
CC = { (idCuerpoTecnico) }
FK = { (cumpleFuncion), (idCuerpoTecnico) }

References:

CUERPOTECNICO.idCuerpoTecnico debe estar en INTEGRANTE.idIntegrante.
CUERPOTECNICO.cumpleFuncion debe estar en FUNCION.idFuncion.

CUERPOTECNICO.cumpleFuncion no puede ser nulo.

Constraints:

Ninguna.

5.8. Entidad Función

FUNCION(idFuncion, nombreFuncion)

PK = { (idFuncion) }
CC = { (idFuncion) }
FK = {}

References:

FUNCION.idFuncion puede no estar en CUERPOTECNICO.cumpleFuncion.

Constraints:

Ninguna.

5.9. Entidad Equipo

EQUIPO(idEquipo, nombreEquipo)

PK = { (idEquipo) }
CC = { (idEquipo) }
FK = {}

References:

EQUIPO.idEquipo debe estar en JUGADOR.estaEnEquipo.

Constraints:

Ninguna

5.10. Entidad Partido

PARTIDO(idPartido, juegaEnEtapa, equipoSeleccion1, equipoSeleccion2, juegaEnEstadio, duracion, fecha, horario)

PK = { (idPartido) }
CC = { (idPartido) }
FK = { (juegaEnEtapa), (equipoSeleccion1), (equipoSeleccion2), (juegaEnEstadio) }

References:

PARTIDO.juegaEnEtapa debe estar en ETAPA.idEtapa.
PARTIDO.equipoSeleccion1 debe estar en SELECCION.idSeleccion.
PARTIDO.equipoSeleccion2 debe estar en SELECCION.idSeleccion.
PARTIDO.juegaEnEstadio debe estar en ESTADIO.idEstadio.
PARTIDO.idPartido debe estar en ARBITRA.idPartidoArb.
PARTIDO.idPartido puede no estar en PARTICIPACION.jugoPartido.
PARTIDO.idPartido debe estar en TANTEADOR.idPartido.

PARTIDO.juegaEnEtapa no puede ser nulo.
PARTIDO.equipoSeleccion1 no puede ser nulo.
PARTIDO.equipoSeleccion2 no puede ser nulo.
PARTIDO.juegaEnEstado no puede ser nulo.

Constraints:

PARTIDO.equipoSeleccion1 <> PARTIDO.equipoSeleccion2.
Si PARTIDO.juegaEnEtapa == "FASE_GRUPOS" ⇒ PARTIDO.equipoSeleccion1.grupo == PARTIDO.equipoSeleccion2.grupo.
Si PARTIDO.juegaEnEtapa == "FASE_GRUPOS" ⇒ # PARTIDO <= 12.
Si PARTIDO.juegaEnEtapa == "5TO_PUESTO" ⇒ # PARTIDO <= 1.
Si PARTIDO.juegaEnEtapa == "3ER_PUESTO" ⇒ # PARTIDO <= 1.
Si PARTIDO.juegaEnEtapa == "SEMIFINAL" ⇒ # PARTIDO <= 2.
Si PARTIDO.juegaEnEtapa == "FINAL" ⇒ # PARTIDO <= 1.
No puede haber dos partidos en un mismo horario.
Validar que al insertar un partido, esten todos los anteriores de fase.
Las fechas de los partidos tienen que estar ordenado por la etapa (FASE_GRUPO < 5TO_PUESTO < SEMIFINAL < 3ER_PUESTO < FINAL).
La duración de los partidos tiene que ser > 0.
La hora del partido tiene que estar entre 0 y 23.
Dos equipos no pueden enfrentarse en la misma etapa dos veces.

5.11. Entidad Estadio

ESTADIO(idEstadio, nombreEstadio)

PK = { (idEstadio) }
CC = { (idEstadio) }
FK = {}

References:

ESTADIO.idEstadio puede no estar en PARTIDO.juegaEnEstadio.
ESTADIO.idEstadio puede no estar en SELECCION.concentraEstadio.

Constraints:

Ninguna.

5.12. Entidad Etapa

ETAPA(idEtapa, nombreEtapa)
PK = { (idEtapa) }
CC = { (idEtapa) (nombreEtapa) }
FK = {}

References:

ETAPA.idEtapa puede no estar en PARTIDO.juegaEnEtapa.

Constraints:

ETAPA.nombreEtapa debe ser o bien FASE_GRUPOS o 5TO_PUESTO, o 3ER_PUESTO o SEMIFINAL, o FINAL.

5.13. Entidad Tanteador

TANTEADOR(nroCuarto, idPartido, scoreEquip1, scoreEquip2)
PK = { (nroCuatro, idPartido) }
CC = { (nroCuarto,idPartido) }
FK = { (idPartido) }

References:

TANTEADOR.idPartido debe estar en PARTIDO.idPartido.

TANTEADOR.idPartido no puede ser nulo.

Constraints:

Los valores posibles de TANTEADOR.nroCuatro son {1 ,2 , 3, 4}.
TANTEADOR.scoreEquip1 >= 0.
TANTEADOR.scoreEquip2 >= 0.
Si TANTEADOR.nroCuarto == 4 ⇒ TANTEADOR.scoreEquip1 <> TANTEADOR.scoreEquip2 (no hay empates).

Constraints Adicionales:

TANTEADOR.nroCuarto no puede aparecer más de 4 veces por TANTEADOR.idPartido (el tanteador se genera con los 4 cuartos cuando se genera un partido).

5.14. Entidad Arbitro

ARBITRO(idArbitro, pertenecePais, nombreArbitro)
PK = { (idArbitro) }
CC = { (idArbitro) }
FK = (pertenecePais) }

References:

ARBITRO.pertenecePais debe estar en PAIS.idPais.
ARBITRO.idArbitro debe estar en ARBITRA.idArbitroArb.
ARBITRO.idArbitro puede no estar en SANCION.sancionadaPorArbitro.

ARBITRO.pertenecePais no puede ser nulo.

Constraints:

Ninguna.

Constraints Adicionales:

Un árbitro no puede dirigir un partido si:

- Dirigió a alguno de los equipos 2 o más veces, y en todos los partidos el equipo obtuvo el mismo resultado (ganó o perdió, no hay empate).
- Dirigió una sola vez a cada equipo con resultados opuestos.
- Está asignado a dirigir un partido en la misma fecha.

5.15. Entidad Arbitra

ARBITRA(idArbitroArb, idPartidoArb)
PK = { (idArbitroArb, idPartidoArb) }
CC = { (idArbitroArb, idPartidoArb) }
FK = { (idArbitroArb), (idPartidoArb) }

References:

ARBITRA.idArbitroArb debe estar en ARBITRO.idArbitro.
ARBITRA.idPartidoArb debe estar en PARTIDO.idPartido.

Constraints:

(ARBITRO.idArbitro == ARBITRA.idArbitroArb) and (ARBITRA.idPartidoArb == PARTIDO.idPartido) ⇒ (ARBITRO.pertenecePais <> PARTIDO.seleccionEquipo1.representaPais and ARBITRO.pertenecePais <> PARTIDO.seleccionEquipo2.representaPais).

5.16. Entidad Participación

PARTICIPACION(idParticipacion, jugoPartido, participaJugador, asistencias, rebotes, posicion, puntos, esTitular)

PK = { (idParticipacion) }

CC = { (idParticipacion) }

FK = { (jugoPartido), (participaJugador) }

References:

PARTICIPACION.jugoPartido debe estar en PARTIDO.idPartido.

PARTICIPACION.participaJugador debe estar en JUGADOR.idJugador.

PARTICIPACION.idParticipacion puede no estar en SANCION.aplicaParticipacion.

PARTICIPACION.posicion puede ser o bien BASE o ESCOLTA o ALERO o ALA-PIVOT o PIVOT o nulo.

PARTICIPACION.jugoPartido no puede ser nulo.

PARTICIPACION.participaJugador no puede ser nulo.

Constraints:

PARTICIPACION.jugoPartido.equipoSeleccion1 puede aparecer a lo sumo 5 veces con PARTICIPACION.esTitular == true.
(Esta resticción no fue modelada en la implementación física)

PARTICIPACION.jugoPartido.equipoSeleccion2 puede aparecer a lo sumo 5 veces con PARTICIPACION.esTitular == true.
(Esta resticción no fue modelada en la implementación física)

No puede haber dos PARTICIPACION.participaJugador tal que PARTICIPACION.esTitular == true y PARTICIPACION.participaJugador.IdIntegrante.perteneceSeleccion sean iguales y PARTICIPACION.posicion sean iguales.

PARTICIPACION.rebotes >= 0.
(Esta resticción no fue modelada en la implementación física)

PARTICIPACION.asistencias >= 0.

PARTICIPACION.puntos >= 0.

Si PARTICIPACION.esTitular == false ⇒ PARTICIPACION.posicion is null. (Esta resticción no fue modelada en la implementación física)

5.17. Entidad Sanción

SANCION(idSancion, aplicaParticipacion, sancionadaPorArbitro, esDeTipo)
PK = { (idSancion) }
CC = { (idSancion) }
FK = { (aplicaParticipacion) , (sancionadaPorArbitro), (esDeTipo) }

References:
SANCION.aplicaParticipacion debe estar en PARTICIPACION.idParticipacion.
SANCION.sancionadaPorArbitro debe estar en ARBITRO.idArbitro.
SANCION.esDeTipo debe estar en TIPOSANCION.idTipoSancion.

SANCION.aplicaParticipacion no puede ser nulo.
SANCION.sancionadaPorArbitro no puede ser nulo.
SANCION.esDeTipo no puede ser nulo.

Constraints:
ARBITRA.idArbitroArb == SANCION.sancionadaPorArbitro and ARBITRA.idPartidoArb == SANCION.aplicaParticipacion.jugoPartido.

5.18. Entidad TipoSanción

TIPOSANCION(idTipoSancion, nombreSancion)

PK = { (idTipoSancion) }
CC = { (idTipoSancion) }
FK = {}

References:
TIPOSANCION.idTipoSancion puede no estar en SANCION.esDeTipo.

Constraints:
Ninguna.

6. Esquema de partidos

A continuación se detalla cuáles son los partidos, selecciones y resultados obtenidos utilizando la carga de datos suministrada en el informe (data.sql).

Figura 1: Grupos del torneo

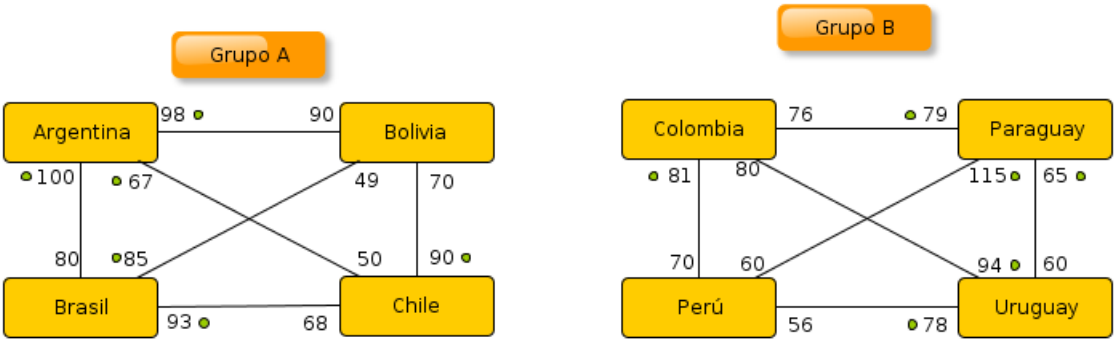


Figura 2: Partidos de Fase de grupos

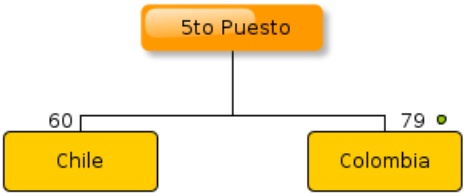


Figura 3: Partido de 5to puesto

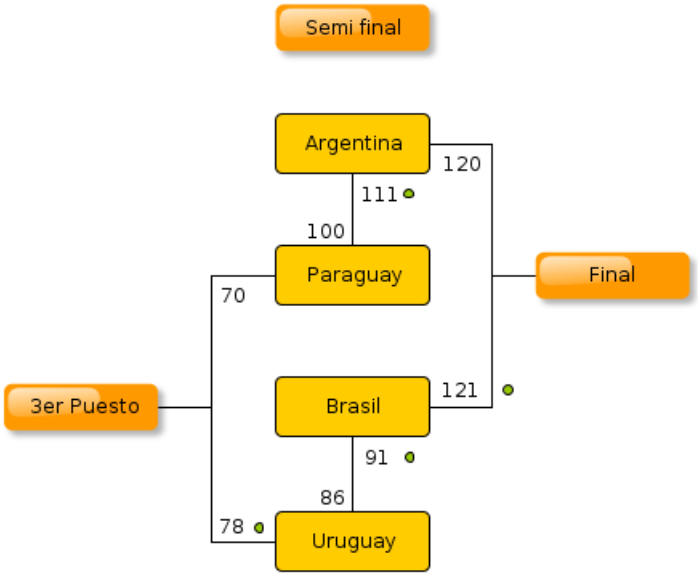


Figura 4: Partido de 5to puesto

7. Decisiones tomadas y supuestos asumidos

Se agregó la entidad Posición, entidad que posee todos sus atributos redundates (estos pueden ser inferidos del modelo relacional sin la entidad Posición, de manera no trivial).
Se decidió agregarla ya que la información suministrada define el resultado final del Torneo de Basquet.

La información que suministra, depende de la cantidad de partidos jugados, la etapa de cada uno de ellos y quién fue el ganador.

Teniendo esta entidad redundante en el modelo, las consultas sobre el resultado final del Torneo se simplifican considerablemente, y no deben ser recalculadas para cada una de ellas.

Para el cálculo de la tabla de posiciones se definió arbitrariamente los siguientes puntajes por partidos ganados:

Partido ganado en **FASE_GRUPO** $\Rightarrow$ 1 punto.
Partido ganado en **5TO_PUESTO** $\Rightarrow$ 3 puntos.
Partido ganado en **SEMIFINAL** $\Rightarrow$ 7 puntos.
Partido ganado en **3ER_PUESTO** $\Rightarrow$ 11 puntos.
Partido ganado en **FINAL** $\Rightarrow$ 19 puntos.

Internamente se insertan 4 las filas en el tanteador por cada vez que se inserta un partido.

El cálculo de la tabla de posiciones se hace internamente al hacer update de los valores del

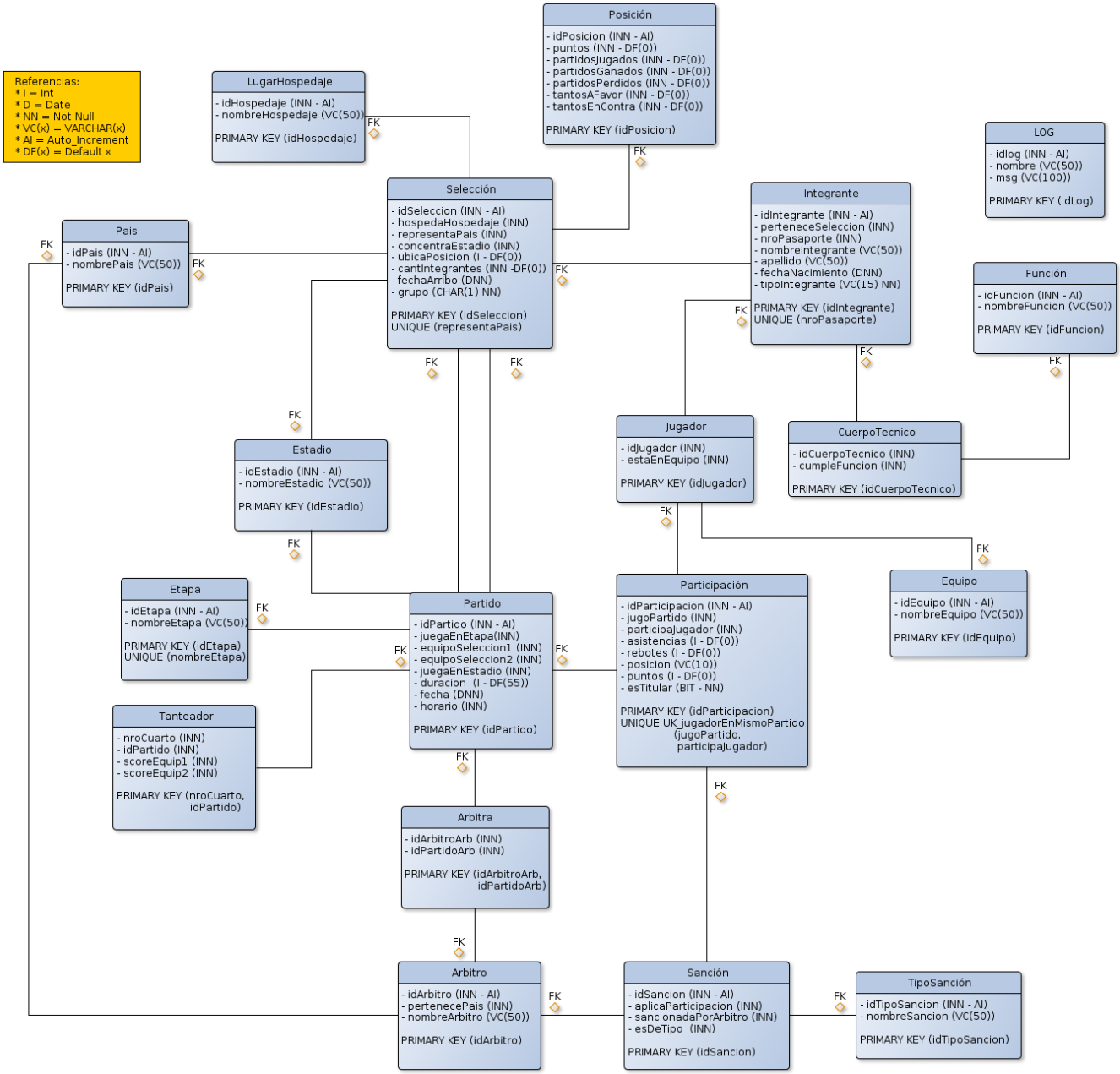
último cuarto en el tanteador al terminar un partido. De esta manera y para preservar consistencia, una vez insertado estos resultados, ellos no pueden ser editados. Cuando el último partido (la final) haya sido jugado y terminado, la entidad posición contará con toda la información sobre el campeonato.

Se modela la entidad **TipoSancion** para agrupar los tipos de sanciones ya que consideramos que probable que en un futuro pueda agregarse una sanción nueva. Los hospedajes salvados, corresponden a los indicados por los jugadores.

Sólo se guardan los equipos correspondientes a los jugadores de la selecciones.

8. Diseño físico

A continuación se detalla mediante un gráfico cuál fue el diseño de base de datos elegido para representar el modelo relacional.



9. Breakers

Los breakers son sentencias de inserts diseñadas específicamente para poner a prueba las restricciones del modelo. El archivo breakers.sql permite ejecutarlas y ver como todas ellas se verifican.

10. Restricciones al modelo

10.1. Restricciones pedidas por la cátedra

- Una consulta que devuelva todos los nombres de los países cuyos equipos utilizaron a todos sus jugadores en algún partido como titulares.
  Esta restricción se encuentra modelada en el archivo **business_sps.sql** y el store procedure se llama **sp_paises_todos_titulares()**

- Un reporte con los nombres de los jugadores, la cantidad de partidos en que participó y los promedios de puntos, asistencias y rebotes. El reporte debe estar ordenado por cantidad de partidos en que participo cada jugador.
  Esta restricción se encuentra modelada en el archivo **business_sps.sql** y el store procedure se llama **sp_estadisticas_por_jugador()**

- A medida que va avanzando un torneo, se van cargando en la BD los próximos partidos, y asignando árbitros a los mismos. Un árbitro no puede dirigir un partido si:

  - Dirigió a alguno de los equipos 2 o más veces, y en todos los partidos el equipo obtuvo el mismo resultado (ganó o perdió, no hay empate)
  - Dirigió una sola vez a cada equipo con resultados opuestos.
  - Está asignado a dirigir un partido en la misma fecha

  Se pide realizar un stored procedure que dado un partido, devuelva los árbitros candidatos a dirigirlo.
  Esta restricción se encuentra modelada en el archivo **business_sps.sql** y el store procedure se llama **sp_posibles_arbitros_por_partidos(idPartidoSP INT)**

- Implementar una restricción que impida que en la base de datos se pueda asignar un árbitro en un partido que sea del mismo país que alguno de los equipos que participa.
  Esta restricción se encuentra modelada en el archivo **arbitraConstraints.sql** y el trigger se llama **check_arbitra_bi**

- Implementación de alguna restricción adicional que surja del diseño. En el archivo **business_constraints.sql** se encuentran detalladas todas las constraints adicionales que se han creado.

10.2. Restricciones adicionales al modelo

Contamos con más de 40 restricciones al modelo de entre las cuales se encuentran las dictadas por el Modelo de entidad relación y por restricciones propias agregadas descriptas en la sección de Modelo lógico relacional 5