Liga Baloncesto

Daniel Alejandro Latorre Ruiz
CampusLands
P1

Pedro Gomez Bonilla
05/07/2024

Tablas de contenido

Introducción

En el ámbito de la gestión deportiva, es esencial contar con sistemas eficientes para el seguimiento y registro de eventos. La liga de baloncesto, enfrentándose a la necesidad de organizar y administrar sus partidos de manera más eficaz, requiere un sistema pequeño pero robusto que permita la gestión integral de los partidos que se llevan a cabo. Este sistema debe registrar datos fundamentales como los equipos involucrados, los puntos anotados, el estado del partido y otros detalles relevantes.

El presente trabajo aborda el diseño e implementación de una base de datos para gestionar los partidos de la liga de baloncesto. El proceso se llevará a cabo en varias etapas, comenzando con la creación del modelo conceptual y lógico, seguido de la normalización de las tablas. Posteriormente, se implementará el modelo físico en MySQL. Una vez creada la base de datos, se procederá a su conexión con Clever Cloud para la gestión en la nube. Finalmente, se desarrollará un sistema en Java que permitirá interactuar con la base de datos, facilitando el registro, actualización y visualización de los partidos a través de un menú en la consola.

Caso de estudio

La liga de baloncesto organiza numerosos partidos a lo largo de la temporada, tanto en formato de liga regular como en eliminatorias de playOffs. Cada partido posee características clave que deben ser registradas para llevar un control preciso y ofrecer información clara a los interesados. El sistema debe ser capaz de registrar partidos, incluyendo detalles de los equipos, puntos anotados, estado y fecha del partido, tipo de partido (Liga o PlayOffs) y detalles adicionales. Además, debe permitir la actualización de puntos, determinar el ganador, finalizar partidos y evitar empates en los playOffs. Finalmente, es crucial contar con una interfaz de usuario que facilite el registro, actualización y visualización de los partidos.

Instalación general

Para implementar el sistema de gestión de partidos de baloncesto, seguiremos los siguientes pasos:

1. Creación de la Base de Datos en MySQL:

- Diseñamos el modelo conceptual y lógico de la base de datos.
- Normalizamos las tablas para eliminar redundancias y asegurar la integridad de los datos.
- Implementamos el modelo físico en MySQL.

2. Conexión de la Base de Datos a Clever Cloud:

- Regístrate e inicia sesión en Clever Cloud.
- Crea una nueva aplicación de base de datos MySQL.
- Sube el archivo `.sql` con la estructura de la base de datos creada anteriormente.
- Obtén las credenciales de conexión (host, puerto, nombre de la base de datos, usuario y contraseña).

3. Desarrollo del Sistema en Java:

- Configura el proyecto de Java con las dependencias necesarias (por ejemplo, el conector JDBC para MySQL).
- Establece la conexión a la base de datos utilizando las credenciales proporcionadas por Clever Cloud.

Planificación

Para implementar el sistema de gestión de partidos de baloncesto, seguiremos un plan detallado que abarca desde la creación de la base de datos hasta su conexión con Clever Cloud y el desarrollo del sistema en Java. A continuación, se describen cada uno de los pasos a seguir:

1. Creación de la Base de Datos en MySQL

1.1 Diseño del Modelo Conceptual y Lógico:
- Identificar las entidades y atributos necesarios para el sistema: Partido, Equipo y DetallePartido.
- Definir las relaciones entre estas entidades.
- Crear un diagrama ERD (Diagrama Entidad-Relación) para visualizar el modelo.

1.2 Normalización de Tablas:
- Revisar la tabla inicial para detectar redundancias y dependencias.
- Dividir la tabla en varias tablas normalizadas para mejorar la eficiencia y la integridad de los datos.

1.3 Implementación del Modelo Físico en MySQL:
- Crear un archivo .sql con la estructura de la base de datos.
- Escribir las sentencias SQL para crear las tablas normalizadas.

2. Conexión de la Base de Datos a Clever Cloud

2.1 Registro e Inicio de Sesión en Clever Cloud:
- Crear una cuenta en Clever Cloud o iniciar sesión en una cuenta existente.

2.2 Creación de una Aplicación de Base de Datos MySQL:
- Desde el panel de Clever Cloud, crear una nueva aplicación de base de datos MySQL.
- Configurar los parámetros necesarios para la aplicación.

2.3 Subida del Archivo .sql a Clever Cloud:
- Utilizar el panel de Clever Cloud o una herramienta de administración de bases de datos (como phpMyAdmin o MySQL Workbench) para subir y ejecutar el archivo .sql creado anteriormente.

2.4 Obtención de Credenciales de Conexión:
- Obtener las credenciales de conexión (host, puerto, nombre de la base de datos, usuario y contraseña) proporcionadas por Clever Cloud para conectar la base de datos con el sistema en Java.

3. Desarrollo del Sistema en Java

3.1 Configuración del Proyecto Java:

- Crear un nuevo proyecto en un entorno de desarrollo integrado (IDE) como IntelliJ IDEA o Apache NetBeans.
- Añadir las dependencias necesarias, como el conector JDBC para MySQL.

3.2 Establecimiento de la Conexión a la Base de Datos:

3.3 Desarrollo del Menú en Consola:

- Implementar un menú interactivo en la consola que permita al usuario registrar nuevos partidos, actualizar datos, finalizar partidos y mostrar información de los partidos.
- Configurar la conexión a la base de datos utilizando las credenciales de Clever Cloud

3.4 Pruebas y Ajustes:

- Realizar pruebas del sistema para asegurarse de que todas las funcionalidades funcionan correctamente.
- Ajustar cualquier problema que surja durante las pruebas.

Ejecución

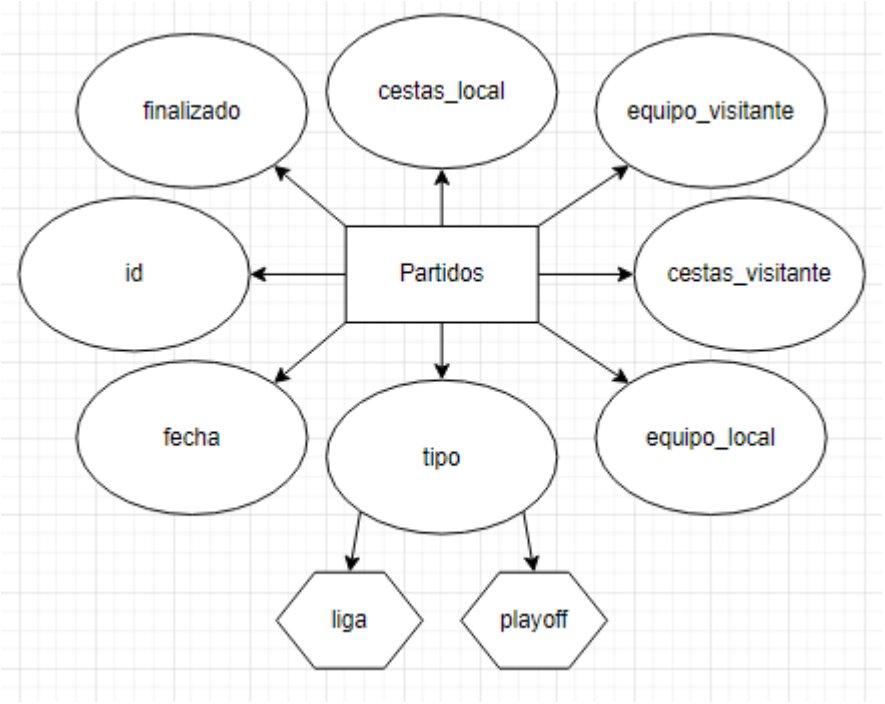
Construcción del modelo conceptual

Descripción

Este modelo conceptual de base de datos representa la estructura de una tabla llamada "Partidos", diseñada para almacenar información sobre partidos de baloncesto. Cada partido tiene varios atributos que describen diferentes aspectos del mismo. A continuación se detalla cada uno de estos atributos y sus relaciones:

- **id**: Es el identificador único de cada partido. Es la clave primaria de la tabla y se usa para distinguir entre diferentes registros de partidos.
- **fecha**: Representa la fecha en que se lleva a cabo el partido.
- **tipo**: Indica el tipo de partido, que puede ser de "liga" o "playoff". Se usa para diferenciar entre partidos regulares y partidos de eliminación directa.
- **equipo_local**: Nombre o identificador del equipo que juega como local en el partido.
- **equipo_visitante**: Nombre o identificador del equipo que juega como visitante en el partido.
- **cestas_local**: Número de puntos anotados por el equipo local en el partido.
- **cestas_visitante**: Número de puntos anotados por el equipo visitante en el partido.
- **finalizado**: Indicador de si el partido ha finalizado o no, lo cual puede ser un valor booleano (true o false).

Gráfica



Descripción técnica

Explicación de la Estructura

- **id**: Es un campo de tipo entero (INT) que se auto-incrementa para asegurar que cada partido tiene un identificador único.
- **fecha**: Es un campo de tipo fecha (DATE) que almacena la fecha del partido.
- **tipo**: Es un campo de tipo enumerado (ENUM) que restringe los valores posibles a 'liga' o 'playoff'.
- **equipo_local** y **equipo_visitante**: Son campos de tipo cadena de texto (VARCHAR) que almacenan los nombres o identificadores de los equipos. Aquí se usa una longitud máxima de 100 caracteres, pero puede ajustarse según sea necesario.
- **cestas_local** y **cestas_visitante**: Son campos de tipo entero (INT) que almacenan los puntos anotados por los equipos locales y visitantes, respectivamente.
- **finalizado**: Es un campo de tipo booleano (BOOLEAN) que indica si el partido ha terminado (true) o no (false).

Construcción del modelo lógico

Descripción

El modelo lógico se centra en cómo se almacenan y organizan los datos en la base de datos. Aquí, la tabla "Partidos" se define con todos los atributos necesarios y sus respectivos tipos de datos. Además, se incluyen las restricciones y claves primarias.

Gráfica

Partidos	
PK	id
	equipo_local
	equipo_visitante
	cestas_local
	cestas_visitante
	finalizado
	fecha
	tipo

Descripción técnica

Atributo	Tipo de Datos	Restricciones	Descripción
id	INT	PRIMARY KEY, AUTO_INCREMENT	Identificador único de cada partido
fecha	DATE	NOT NULL	Fecha del partido
tipo	ENUM('liga', 'playoff')	NOT NULL	Tipo de partido (liga o playoff)
equipo_local	VARCHAR(100)	NOT NULL	Nombre o identificador del equipo local
equipo_visitante	VARCHAR(100)	NOT NULL	Nombre o identificador del equipo visitante
cestas_local	INT	NOT NULL	Puntos anotados por el equipo local
cestas_visitante	INT	NOT NULL	Puntos anotados por el equipo visitante
finalizado	BOOLEAN	NOT NULL	Indica si el partido ha finalizado (true o false)

Explicación Técnica

- **id**: Utiliza el tipo de datos `INT` y se define como `PRIMARY KEY` con `AUTO_INCREMENT`, asegurando un identificador único para cada registro.
- **fecha**: Utiliza el tipo de datos `DATE`, asegurando que solo se acepten valores de fecha válidos.
- **tipo**: Utiliza el tipo de datos `ENUM` con los valores permitidos 'liga' y 'playoff', garantizando la integridad de los datos para este atributo.
- **equipo_local** y **equipo_visitante**: Utilizan el tipo de datos `VARCHAR(100)`, permitiendo almacenar nombres o identificadores de hasta 100 caracteres.
- **cestas_local** y **cestas_visitante**: Utilizan el tipo de datos `INT`, asegurando que solo se acepten valores enteros para los puntos anotados.
- **finalizado**: Utiliza el tipo de datos `BOOLEAN`, permitiendo valores `true` o `false` para indicar el estado del partido.

Normalización

Primera forma normal (FN1)

Descripción

La 1NF se logra cuando todos los atributos de la tabla contienen valores atómicos, es decir, no hay conjuntos de valores ni valores repetidos en una misma columna.

Para el modelo "Partidos", ya que cada columna contiene valores únicos y atómicos, la tabla ya está en 1NF. No se necesita hacer ningún cambio adicional en esta fase.

Gráfica

id	fecha	tipo	equipo_local	equipo_visitante	cestas_local	cestas_visitante	finalizado
1	2024-07-29	liga	Team A	Team B	80	75	true
2	2024-07-30	playoff	Team C	Team D	85	90	true

Segunda forma normal (FN2)

Descripción

Para que una tabla esté en 2NF, debe estar en 1NF y todos los atributos no clave deben depender completamente de la clave primaria. En este caso, la tabla "Partidos" ya cumple con esta condición, ya que todos los atributos dependen de la clave primaria `id`.

Sin embargo, para mejorar la estructura y evitar redundancias, podemos dividir la tabla en dos tablas: una para "Partidos" generales y otra para "Detalles de Partidos" dependiendo del tipo de partido.

Gráfica

Partidos

id	fecha	tipo	equipo_local	equipo_visitante	finalizado
1	2024-07-29	liga	Team A	Team B	true
2	2024-07-30	playoff	Team C	Team D	true

Detalles_Partidos

id	cestas_local	cestas_visitante
1	80	75
2	85	90

Tercera forma normal (FN3)

Descripción

Para que una tabla esté en 3NF, debe estar en 2NF y todos los atributos no clave deben ser mutuamente independientes, es decir, no debe haber dependencia transitiva.

En este caso, el atributo `tipo` puede ser dividido en dos tablas adicionales: "Liga" y "Playoff", para eliminar cualquier dependencia transitiva entre los tipos de partidos y sus atributos específicos.

Gráfica

Partidos

id	fecha	equipo_local	equipo_visitante	finalizado
1	2024-07-29	Team A	Team B	true
2	2024-07-30	Team C	Team D	true

Liga

partido_id	cestas_local	cestas_visitante
1	80	75

Playoff

partido_id	cestas_local	cestas_visitante
2	85	90

Construcción del modelo físico

Descripción

El modelo físico define cómo se implementará el modelo lógico en una base de datos específica, incluyendo la definición de tablas, tipos de datos, claves primarias y foráneas, así como cualquier índice que pueda ser necesario para optimizar el rendimiento.

Código

```sql
-- Tabla Partidos
▷ Run | New Tab | Copy
CREATE TABLE Partidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATE NOT NULL,
    equipo_local VARCHAR(100) NOT NULL,
    equipo_visitante VARCHAR(100) NOT NULL,
    finalizado BOOLEAN NOT NULL
);

-- Tabla Liga
▷ Run | New Tab | Copy
CREATE TABLE Liga (
    partido_id INT PRIMARY KEY,
    cestas_local INT NOT NULL,
    cestas_visitante INT NOT NULL,
    FOREIGN KEY (partido_id) REFERENCES Partidos(id)
);

-- Tabla Playoff
▷ Run | New Tab | Copy
CREATE TABLE Playoff (
    partido_id INT PRIMARY KEY,
    cestas_local INT NOT NULL,
    cestas_visitante INT NOT NULL,
    FOREIGN KEY (partido_id) REFERENCES Partidos(id)
);
```

Descripción técnica

Partidos

- **id**: El campo `id` es de tipo `INT` y es la clave primaria de la tabla, con `AUTO_INCREMENT` para garantizar que cada partido tenga un identificador único y secuencial.
- **fecha**: El campo `fecha` es de tipo `DATE` y no permite valores nulos, garantizando que cada partido tenga una fecha asociada.
- **equipo_local** y **equipo_visitante**: Ambos campos son de tipo `VARCHAR(100)` y no permiten valores nulos, almacenando el nombre o identificador de los equipos que participan en el partido.
- **finalizado**: El campo `finalizado` es de tipo `BOOLEAN` y no permite valores nulos, indicando si el partido ha terminado o no.

Liga

- **partido_id**: El campo `partido_id` es de tipo `INT`, es la clave primaria de la tabla y también es una clave foránea que referencia al campo `id` de la tabla `Partidos`, garantizando la integridad referencial.

- **cestas_local** y **cestas_visitante**: Ambos campos son de tipo `INT` y no permiten valores nulos, almacenando el número de puntos anotados por los equipos locales y visitantes respectivamente.

Playoff

- **partido_id**: Similar al campo `partido_id` en la tabla `Liga`, este campo es de tipo `INT`, es la clave primaria de la tabla y también es una clave foránea que referencia al campo `id` de la tabla `Partidos`.
- **cestas_local** y **cestas_visitante**: Ambos campos son de tipo `INT` y no permiten valores nulos, almacenando el número de puntos anotados por los equipos locales y visitantes respectivamente.

Construcción del modelo UML

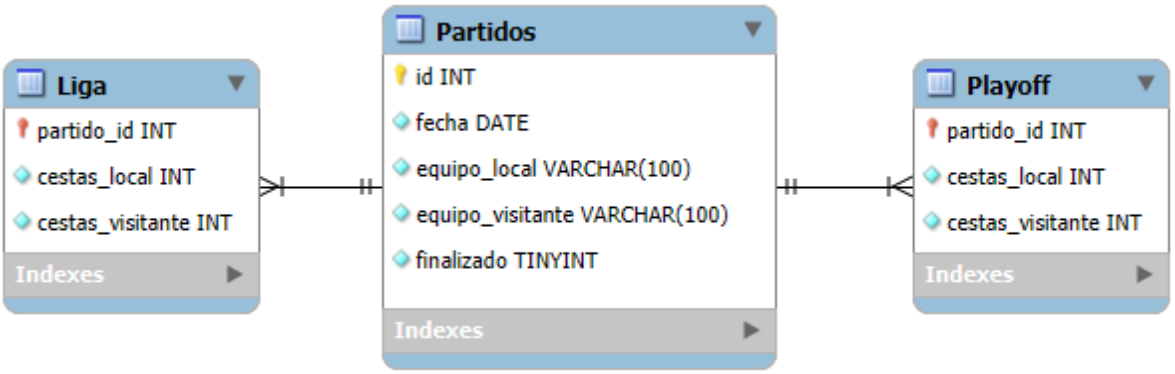
Descripción

El diagrama UML representa un sistema de gestión de partidos de baloncesto, diferenciando entre partidos de liga y partidos de playoff. Está compuesto por tres entidades principales: `Partidos`, `Liga` y `Playoff`.

1. **Partidos**:
    - Representa la entidad principal donde se almacena la información básica de cada partido de baloncesto.
    - Atributos:
        - `id`: Identificador único del partido.
        - `fecha`: Fecha en que se juega el partido.
        - `equipo_local`: Nombre del equipo local.
        - `equipo_visitante`: Nombre del equipo visitante.
        - `finalizado`: Indica si el partido ha finalizado (0 para no, 1 para sí).
2. **Liga**:
    - Representa la entidad específica para los partidos de liga.
    - Atributos:
        - `partido_id`: Identificador del partido (clave foránea que referencia a `Partidos`).
        - `cestas_local`: Cantidad de puntos anotados por el equipo local.
        - `cestas_visitante`: Cantidad de puntos anotados por el equipo visitante.
3. **Playoff**:
    - Representa la entidad específica para los partidos de playoff.
    - Atributos:
        - `partido_id`: Identificador del partido (clave foránea que referencia a `Partidos`).
        - `cestas_local`: Cantidad de puntos anotados por el equipo local.

- `cestas_visitante`: Cantidad de puntos anotados por el equipo visitante.

Gráfica



Descripción técnica

Relaciones:

- **Partidos**: Es la tabla principal que almacena la información básica de todos los partidos. Tiene una relación de uno a uno con las tablas `Liga` y `Playoff` a través del campo `id`.
- **Liga**: Tabla que almacena información específica sobre los partidos de liga. El campo `partido_id` actúa como una clave foránea que referencia al campo `id` de la tabla `Partidos`.
- **Playoff**: Tabla que almacena información específica sobre los partidos de playoff. El campo `partido_id` actúa como una clave foránea que referencia al campo `id` de la tabla `Partidos`.

Integridad Referencial:

- Las claves foráneas (`partido_id` en las tablas `Liga` y `Playoff`) aseguran que cada registro en estas tablas está asociado a un partido existente en la tabla `Partidos`. Esto garantiza la integridad referencial, asegurando que no se pueden añadir registros en `Liga` o `Playoff` sin un partido correspondiente en `Partidos`.

Cardinalidad:

- La relación entre `Partidos` y `Liga` es de uno a uno.
- La relación entre `Partidos` y `Playoff` es de uno a uno.
- Esto implica que un partido en la tabla `Partidos` puede estar asociado únicamente a un registro en `Liga` o a un registro en `Playoff`, pero no a ambos simultáneamente.

Tipos de Datos:

- `INT`: Utilizado para los identificadores y los puntos (cestas).
- `DATE`: Utilizado para la fecha del partido.
- `VARCHAR(100)`: Utilizado para los nombres de los equipos.
- `TINYINT`: Utilizado para el campo `finalizado` para indicar si el partido ha finalizado (0 para no, 1 para sí).