FUNDAMENTOS DE BASES DE DATOS

INTEGRANTES EQUIPO "LUK":

Andrade Luviano Ximena

Gerónimo Soto Leslie

Reyes Medina Santiago Iván

López Espinoza Ashley Yael

Islas Espino Julio César

Yañez Díaz Carlos

DECISIONES DE DISEÑO

Relación de archivos:

- **CARPETA : DIAGRAMAS**

Decidimos modelar la base de datos utilizando un diagrama ER para identificar claramente las entidades, sus atributos y las relaciones entre ellas. Esto permitió tener una visión clara del sistema antes de pasar al modelo relacional.

Tradujimos el modelo ER a un esquema relacional con tablas que contienen claves primarias y foráneas para asegurar consistencia entre las entidades.

- **ERLUK.drawio**

Descripción: Archivo en formato editable que contiene el diagrama Entidad-Relación del modelo lógico de la base de datos. Muestra las entidades, atributos y relaciones iniciales del diseño.

Relación: Base para el diagrama relacional. Los datos en el diagrama fueron traducidos al modelo físico representado en los scripts SQL.

- **ERLUK.drawio.png**

Descripción: Imagen exportada del diagrama Entidad-Relación para consulta rápida y sin necesidad de un editor especializado.

Relación: Es una versión estática de ERLUK.drawio para facilitar su visualización.

- **RelacionalLUK.drawio**

Descripción: Archivo en formato editable que contiene el modelo relacional, derivado del diagrama Entidad-Relación. Incluye tablas, claves primarias y foráneas.

Relación: Utilizado para diseñar las estructuras que se implementaron en los scripts SQL (DDL).

- **RelacionalLUK.drawio.png**

Descripción: Imagen exportada del modelo relacional, lista para consulta rápida.

Relación: Versión de consulta del archivo RelacionalLUK.drawio.

- **CARPETA: Docs**

Aplicamos las reglas de normalización hasta la Tercera Forma Normal (3FN) para evitar redundancias y asegurar integridad de los datos.

  - **3FN**

Descripción: Documento que describe cómo se aplicó la tercera forma normal (3FN) para eliminar redundancias y garantizar la consistencia de los datos.

Relación: Refuerza el diseño de los diagramas y los scripts SQL asegurando que los datos están normalizados.

  - **Reporte de Consultas**

Descripción: Documento que detalla las consultas SQL implementadas, explicando su propósito, lógica y resultados esperados.
Relación: Apoya los scripts SQL en la carpeta SQL al describir cómo cada consulta cumple con los objetivos planteados.

  - **Diccionario**

Descripción: Contiene el diccionario de datos con descripciones de cada tabla, columna y su propósito en la base de datos.
Relación: Se basa en el modelo relacional y complementa los diagramas para explicar cómo están estructurados los datos.

- **CARPETA: SQL**
  - **DML.sql**

Descripción: Contiene los comandos SQL para manipulación de datos (INSERT, UPDATE, DELETE).
Relación: Se ejecuta sobre las estructuras definidas en DDL.sql y utiliza los datos modelados en Diagrama Relacional.

  - **DDL.sql**

Descripción: Incluye los comandos SQL para la creación de estructuras de base de datos (CREATE TABLE, CONSTRAINTS, etc.).

Relación: Implementa el modelo relacional de los diagramas en la base de datos.

- **Requests.sql**

Descripción: Archivo con las consultas especificadas en el proyecto
Relación: Depende de las estructuras creadas en DDL.sql y utiliza los datos insertados desde DML.sql.

- **Scripts_Operations.sql**

***Descripción*:** Contiene los disparadores (triggers) y las operaciones automatizadas que garantizan la integridad de los datos y mejoran la eficiencia en la base de datos.
***Relación*:** Trabaja en conjunto con **DDL.sql** y **DML.sql** para complementar la estructura y manipulación de los datos, respetando las dependencias funcionales del diseño relacional.

- **CARPETA: SRC**
  - **Consulta17.py**

Descripción: Carpeta que contiene el código fuente del sistema en donde se implementó una gráfica de la Consulta 17 del proyecto, en python.
Relación: Consume y ejecuta la consulta 17 definida en Requests.sql, manipula datos mediante DML.sql, y utiliza la estructura proporcionada por DDL.sql.

---

1. Diseño Conceptual

- **Elección del Modelo Entidad-Relación (ER):**
  - Decidimos modelar la base de datos utilizando un diagrama ER para identificar claramente las entidades, sus atributos y las relaciones entre ellas. Esto permitió tener una visión clara del sistema antes de pasar al modelo relacional.
- **Normalización:**
  - Aplicamos las reglas de normalización hasta la **Tercera Forma Normal (3FN)** para evitar redundancias y asegurar integridad de los datos.

2. Diseño Relacional

- **Modelo Relacional:**

    - Tradujimos el modelo ER a un esquema relacional con tablas que contienen claves primarias y foráneas para asegurar consistencia entre las entidades.
- **Claves Primarias y Foráneas:**
    - Definimos claves primarias en cada tabla para garantizar unicidad, como idEvento en la tabla Evento.
    - Decisión clave: Usar claves foráneas para garantizar integridad referencial, por ejemplo, relacionar idAtleta en la tabla Ganar con la tabla Atleta.

3. Consultas SQL

- **Estructura de las Consultas:**
    - Decidimos utilizar JOIN en lugar de subconsultas donde fuera posible para mejorar el rendimiento y la claridad del código.
    - Ejemplo: Unir Evento, Localidad y Entrada para calcular entradas vendidas por localidad en lugar de hacerlo con varias subconsultas.
- **Agregaciones y Agrupaciones:**
    - Utilizamos funciones como `SUM`, `COUNT`, y `GROUP BY` para obtener métricas clave, como el total de entradas vendidas o el número de atletas por disciplina.
    - Decisión clave: Ordenar y limitar resultados en consultas como las que buscan eventos con más entradas vendidas.

4. Diseño de Integridad y Automatización

- **Triggers:**
    - Implementamos **triggers** para garantizar la integridad de los datos. Por ejemplo, un trigger que actualiza automáticamente el número de entradas disponibles al registrar una venta.
- **Restricciones de Integridad:**
    - Decidimos usar restricciones como UNIQUE, y CHECK en columnas críticas, como idAtleta, para evitar registros inválidos.

5. Optimización

- **Optimización de Consultas:**

    - Optamos por simplificar consultas complejas dividiéndolas en pasos intermedios donde fuera necesario para mejorar la legibilidad y el rendimiento.
    - Ejemplo: Utilizamos vistas para representar consultas frecuentes y evitar recalcular resultados en tiempo de ejecución.

6. Elección de Herramientas

- **Base de Datos:** PostgreSQL
    - Elegimos PostgreSQL por su robustez, soporte para características avanzadas como triggers y procedimientos almacenados, y su capacidad de manejar grandes volúmenes de datos de manera eficiente.
- **Software de Diseño:** draw.io
    - Utilizamos draw.io para diseñar los diagramas ER y relacionales por su facilidad de uso y soporte para exportar en múltiples formatos.

7. Documentación

- **Diccionario de Datos:**
    - Decidimos crear un diccionario para describir cada tabla, columna y su propósito, facilitando la comprensión del diseño.
- **Reporte de Consultas:**
    - Elaboramos un reporte con explicaciones detalladas de cada consulta SQL para documentar cómo cada una responde a los requerimientos del proyecto.

8. Consideraciones de Seguridad

- **Control de Acceso:**
    - Decidimos implementar roles en PostgreSQL para restringir el acceso a tablas sensibles como Atleta o Evento.
- **Validación de Datos:**
    - Aplicamos restricciones en las columnas, como validación de fechas y rangos de precios, para prevenir entradas erróneas.

Conclusión

Todas estas decisiones de diseño fueron tomadas para garantizar que la solución fuera eficiente, escalable, fácil de mantener y capaz de cumplir con los requerimientos funcionales establecidos.