Universidad Nacional Autónoma de México

Facultad de Ciencias

Fundamentos de Bases de Datosn - 7094

Reporte Ejecutivo
Bases de Datos
Juegos Olímpicos

Equipo:

Del Monte Ortega Maryam Michelle - 320083527

Castillo Hernández Antonio - 320017438

Erik Eduardo Gómez López - 320258211

Sosa Romo Juan Mario

Fecha de entrega:

30 de Noviembre de 2024

Profesor:

M. en I. Gerardo Avilés Rosas

Ayudantes:

Luis Enrique García Gómez

Kevin Jair Torres Valencia

Ricardo Badillo Macías

Rocío Aylin Huerta González

Índice general

1. Introducción **2**

2. Metodología **3**

2.1. Proceso de desarrollo 3

2.2. Herramientas utilizadas 3

3. Funcionalidades **5**

3.1. Triggers 5

3.2. Procedimientos almacenados 5

4. Consultas **7**

5. Conclusiones **17**

6. Anexos **19**

Capítulo 1

Introducción

El Comité Olímpico Internacional (COI), reconociendo la necesidad de modernizar la gestión de información de los Juegos Olímpicos, ha decidido implementar un sistema robusto y centralizado que permita superar los problemas históricos asociados al uso de registros físicos. Este proyecto, desarrollado por estudiantes de la Facultad de Ciencias de la UNAM, busca sentar las bases para una administración de datos más eficiente y consistente, asegurando que los Juegos Olímpicos de Los Ángeles 2028 se beneficien de un manejo más profesional y organizado de la información.

El presente reporte ejecutivo resume el trabajo realizado en el diseño e implementación de una base de datos integral para el COI. A lo largo del documento, se abordan los aspectos técnicos y estratégicos clave, desde la conceptualización del modelo Entidad-Relación hasta la creación de un esquema lógico y físico en PostgreSQL. Asimismo, se destacan las funcionalidades avanzadas del sistema, como procedimientos almacenados, disparadores, y un conjunto de consultas SQL diseñadas para generar reportes ejecutivos que proporcionen información valiosa para la toma de decisiones.

Este sistema no solo representa una solución tecnológica, sino también un paso hacia la profesionalización y digitalización de la administración de los Juegos Olímpicos, marcando un precedente para eventos futuros. El Comité Olímpico Internacional ha confiado en esta propuesta como un pilar esencial para el éxito organizativo de los próximos Juegos.

Capítulo 2

Metodología

2.1. Proceso de desarrollo

El proyecto se desarollo en varias fases, que basicamente seguian el curso de la materia, especificamente del laboratorio de bases de datos. A continuación se describen las fases del proyecto:

1. **Fase 1:** Instalacion de PostgreSQL y dbeaver, `creación de la base de datos` dentro de un docker.
2. **Fase 2:** `Analisis de requerimientos` funcionales y no funcionales y consideracion de alternativas.
3. **Fase 3:** Diseño del `modelo Entidad-Relación` usando draw.io.
4. **Fase 4:** Creación del `modelo relacional` usando draw.io.
5. **Fase 5:** Creacion de tablas, definidas en el archivo `DDL.sql`.
6. **Fase 6:** Mantenimiento de llaves foraneas y llaves primarias, también en el mismo archivo.
7. **Fase 7:** Población de tablas con datos de prueba, se encuentra en el archivo `DML.sql`.
8. **Fase 8:** Prueba de la base de datos usando una app en python usando Psycopg2 y Django.
9. **Fase 9:** `Consultas` útiles para el COI, para mas detalles ver el capitulo 4 o el anexo `Consultas.sql` o `PruebasDeFuncionalidad`.
10. **Fase 10:** Creación de `triggers` y `procedimientos almacenados` para la base de datos, ver en los anexos con los mismos nombres para mas detalles.

2.2. Herramientas utilizadas

Las herramientas que utilizamos para la elaboración de este proyecto son las siguientes:

- **PostgreSQL:** Sistema de gestión de bases de datos relacional orientado a objetos.
- **dbeaver:** Herramienta de administración de bases de datos que permite la conexión a múltiples sistemas de gestión de bases de datos.
- **Docker:** Plataforma de código abierto que facilita la creación, implement
- **Git:** Sistema de control de versiones distribuido.
- **GitHub:** Plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.
- **LaTeX:** Sistema de composición de textos.
- **draw.io** Herramienta en línea para la creación de diagramas.

- **Visual Studio Code:** Editor de código fuente desarrollado por Microsoft.
- **Python:** Lenguaje de programación interpretado.
- **Psycopg2:** Adaptador de base de datos PostgreSQL para el lenguaje de programación Python.
- **Plpgsql:** Lenguaje de programación procedural que se utiliza en PostgreSQL.
- **Django:** Framework de desarrollo web de código abierto, escrito en Python.

Capítulo 3

Funcionalidades

Para esta sección vamos a detallar los triggers y procedimientos almacenados que se crearon en la base de datos, con el fin de automatizar ciertas tareas y garantizar la integridad de los datos, el codigo de ambos se encuentra en los archivos anexos con los mismos nombres.

3.1. Triggers

1. **Trigger 1:** El primer trigger que se creó es para verificar que un atleta tenga al menos 10 años de edad al momento de ser registrado en la base de datos, el trigger actúa tanto en modificacion como en insert en la tabla `Atleta`. Puede parecer trivial pero es un requisito importante para cumplir con requerimientos de la organización.

2. **Trigger 2:** El segundo trigger que se creó es para validar la asignación de medallas a los atletas, el trigger actúa tanto en modificación como en inserción en la tabla `Medalla`. El trigger verifica que un atleta no tenga ya una medalla en la misma disciplina, que el atleta participe en la disciplina y que solo haya una medalla de cada tipo en disciplinas individuales. El código se muestra a continuación:

3. **Trigger 3:** El tercer trigger que se creó es para gestión de aforo, especificamente tiene como propósito controlar la venta de entradas; verifica la disponibilidad de aforo, evita ventas de eventos pasados, limita la cantidad de entradsa por cliente y registra el historial de ventas en una tabla auxiliar. Igualmente actúa tanto en modificación como en inserción, sobre la tabla de `CompraEntrada`.

4. **Trigger 4:** Finalmente creamos un trigger con el proposito de asegurar que los atletas solo concursen en eventos cuyas diciplinas ellos practiquen, el trigger actua tanto en modificación como en inserción en la tabla `Concursa`. Básicamente verifica que si el atleta no practica la disciplina, no puede participar en un evento de la misma.

3.2. Procedimientos almacenados

1. **Procedimiento 1: Registrar Participación en Evento**
   Este procedimiento tiene como objetivo registrar la participación de un atleta en un evento, con la posibilidad opcional de asignarle una medalla. Antes de registrar la participación, se validan varios aspectos: que el evento exista, que el atleta esté registrado en la disciplina del evento, y que no existan duplicados de medallas en la misma disciplina. En caso de que se proporcione una medalla, se asegura que las disciplinas individuales no permitan más de una medalla por tipo para el mismo atleta. Este procedimiento actúa principalmente sobre las tablas `Concursa` y `Medalla`, asegurando integridad y consistencia en los datos.

2. **Procedimiento 2: Actualizar Fase del Evento**
   Este procedimiento cumple con el requerimiento de incrementar la fase eliminatoria de un evento y ajustar su precio en un 9 % cada vez que avanza de fase. Antes de realizar los cambios, verifica que el evento exista en la base de datos y que no se encuentre ya en la última fase (fase 3). Luego actualiza tanto la fase como el precio del evento en la tabla `Evento`. Utiliza mensajes informativos (`RAISE NOTICE`) para notificar los cambios realizados, o lanza excepciones en caso de errores.

3. **Procedimiento 3: Registrar Atleta en Disciplina**
   Este procedimiento asegura el registro de un atleta en una disciplina específica, verificando primero que la disciplina exista en la base de datos y que el atleta no esté ya registrado en la misma. Si las validaciones se cumplen, inserta el registro correspondiente en la tabla `Participa`. Se utilizan mensajes informativos para notificar el éxito de la operación, mientras que las excepciones manejan cualquier caso de error.

Consideramos que estos triggers y procedimientos almacenados son esenciales para garantizar la integridad y consistencia de los datos en la base de datos del COI. Además, automatizan tareas comunes y críticas, lo que facilita la administración y el mantenimiento del sistema.

Aún asi, es importante mencionar que podrían ser mas funcionalidades para facilitar la administración de la base de datos. En este proyecto no se llego a profundizar tanto como nos hubiera gustado.

Capítulo 4

Consultas

Aqui vamos a detallar los resultados de las consultas que se realizaron en la base de datos, con el fin de obtener información relevante para la toma de decisiones. Cabe mencionar que las consultas se realizaron en PostgreSQL, y se utilizaron las tablas y vistas creadas en el esquema lógico de la base de datos. A continuación, se presentan las consultas realizadas y el analisis de los resultados obtenidos:

- **Consulta 1:**
- **Consulta 2:**
- **Consulta 3:**
- **Consulta 4:**
- **Consulta 5:**

```
/* Consulta 5: Atletas que tienen a su cargo cada entrenador.
 *
 * Esta consulta obtiene la cantidad de atletas que tiene a su cargo cada entrenador.
 *
 * - Selecciona la información del entrenador y cuenta la cantidad de atletas que tiene a su cargo.
 * - Agrupa los resultados por el entrenador.
 * - Ordena los resultados por la cantidad de atletas en orden descendente.
 */
SELECT
    e.IDEntrenador,
    e.Nombre AS NombreEntrenador,
    e.PrimerApellido AS PrimerApellidoEntrenador,
    e.SegundoApellido AS SegundoApellidoEntrenador,
    COUNT(a.IDAtleta) AS CantidadAtletas
FROM
    Entrenador e
JOIN
    Atleta a ON e.IDEntrenador = a.IDEntrenador
GROUP BY
    e.IDEntrenador, e.Nombre, e.PrimerApellido, e.SegundoApellido
ORDER BY
    CantidadAtletas DESC
```

identrenador	nombreentrenador	primerapellidoentrenador	segundoapellidoentrenador	cantidadatletas
50	Dalila	Orrett	Kalkhoven	11
101	Ciro	Seabon	Bassom	11
196	Hewe	Fitzsymonds	Bamling	11
118	Friedrich	Byer	Welman	10
15	Giffy	Doncaster	Raund	10
81	Rosamond	Killingworth	Rolley	10
176	Susanne	Dabnor	Thomazin	10
26	Bertrando	Bunner	Bolles	9
174	Daffy	Kobierra	Oakland	9
72	Emanuele	Troth	Dunford	9
184	Agretha	Lochead	Hurdman	9
124	Cally	Freyne	Humfrey	9
154	Killian	Howey	Bunner	9
173	Torey	Gallager	Trethewey	9

Consulta 5. Atletas que tienen a su cargo cada entrenador.

Propósito de la consulta

La consulta tiene como objetivo obtener un listado de entrenadores junto con la cantidad de atletas que tienen a su cargo. Esto es útil para entender la distribución de atletas entre los entrenadores y detectar posibles desequilibrios en la asignación de recursos.

Desglose de la consulta

- **Selección de columnas** (`SELECT`):

- Se seleccionan las siguientes columnas del entrenador:
  - ◇ `e.IDEntrenador`: Identificador único del entrenador.
  - ◇ `e.Nombre`: Nombre del entrenador.
  - ◇ `e.PrimerApellido`: Primer apellido del entrenador.
  - ◇ `e.SegundoApellido`: Segundo apellido del entrenador.
- Se utiliza la función agregada `COUNT(a.IDAtleta)` para contar cuántos atletas están asociados con cada entrenador. Esta columna se denomina `CantidadAtletas`.

- **Tablas involucradas (`FROM` y `JOIN`)**:
  - La consulta utiliza dos tablas:
    - ◇ `Entrenador (e)`: Contiene la información de los entrenadores.
    - ◇ `Atleta (a)`: Contiene la información de los atletas.
  - Se realiza un `JOIN` entre ambas tablas utilizando la relación `e.IDEntrenador = a.IDEntrenador`. Esto asegura que solo se consideren los atletas que están asignados a un entrenador.
- **Agrupación de resultados (`GROUP BY`)**:
  - Para calcular la cantidad de atletas por entrenador, se agrupan los datos según las columnas únicas del entrenador:
    - ◇ `e.IDEntrenador`, `e.Nombre`, `e.PrimerApellido`, `e.SegundoApellido`.
  - Esto garantiza que se genere un registro único por cada entrenador.
- **Ordenamiento de resultados (`ORDER BY`)**:
  - Los resultados se ordenan por la columna `CantidadAtletas` en orden descendente (`DESC`), de modo que los entrenadores con más atletas aparezcan primero.

Análisis detallado

1. **Relación entre tablas:**
   - La consulta asume que existe una relación directa entre las tablas `Entrenador` y `Atleta` a través de la clave foránea `a.IDEntrenador`, que apunta a `e.IDEntrenador`.
   - Esto implica que:
     - Cada atleta tiene asignado exactamente un entrenador.
     - Un entrenador puede tener asignados uno o más atletas.
2. **Uso de la función agregada `COUNT`:**
   - La función `COUNT(a.IDAtleta)` cuenta el número de registros en la tabla `Atleta` que están relacionados con cada entrenador.
   - Si un entrenador no tiene atletas asignados, no aparecerá en los resultados porque el `JOIN` elimina las filas sin coincidencias.
3. **Agrupación por entrenador:**
   - El uso de `GROUP BY` permite agrupar los registros por entrenador, asegurando que la cantidad de atletas se calcule correctamente para cada uno.
4. **Ordenamiento:**
   - El orden descendente por `CantidadAtletas` facilita la identificación de los entrenadores con mayor carga de trabajo.

Consideraciones

- **Empates en la cantidad de atletas:**
  - Si varios entrenadores tienen la misma cantidad de atletas, el orden relativo entre ellos no está definido. Para resolver esto, se podría agregar un criterio adicional en el `ORDER BY`, como el nombre del entrenador.

- **Consulta 6:**

```sql
/* Consulta 6: Ganancias totales por cada competencia celebrada.
 *
 * Esta consulta obtiene las ganancias totales por cada competencia celebrada.
 *
 * - Selecciona el ID del evento, el nombre de la localidad, el ID de la disciplina y la suma de los precios de las entradas.
 * - Agrupa los resultados por el ID del evento, el nombre de la localidad y el ID de la disciplina.
 * - Ordena los resultados por las ganancias totales en orden descendente.
 */
SELECT
    e.IDEvento,
    e.NombreLocalidad,
    e.IDDisciplina,
    SUM(e.Precio) AS GananciasTotales
FROM
    Evento e
JOIN
    CompraEntrada ce ON e.IDEvento = ce.IDEvento
GROUP BY
    e.IDEvento, e.NombreLocalidad, e.IDDisciplina
ORDER BY
    GananciasTotales DESC
```

idevento	nombrelocalidad	iddisciplina	gananciastotales
111	Dojo	13	31,000
135	Canal de Remo	23	30,600
188	Estadio Central	107	28,900
35	Cancha de Pádel	36	28,800
13	Velódromo	7	27,900
109	Marina de Vela	130	27,750
75	Pista de Skeleton	148	27,000
99	Piscina de Waterpolo	12	27,000
36	Sala de Billar	200	25,500
14	Skatepark	78	25,350
52	Centro Acuático	4	24,800
181	Centro de Pentatlón	149	24,650

Consulta 6. Ganancias totales por cada competencia celebrada.

Propósito de la consulta

La consulta tiene como objetivo calcular las ganancias totales generadas por la venta de entradas para cada competencia celebrada. Esto permite identificar qué eventos fueron más rentables y en qué localidades o disciplinas se generaron mayores ingresos.

Desglose de la consulta

- **Selección de columnas (`SELECT`):**
  - `e.IDEvento`: Identificador único del evento, que permite distinguir cada competencia.
  - `e.NombreLocalidad`: Nombre de la localidad donde se celebró el evento.
  - `e.IDDisciplina`: Identificador de la disciplina deportiva asociada al evento.
  - `SUM(e.Precio) AS GananciasTotales`: Calcula la suma total de los precios de las entradas vendidas para cada evento, representando las ganancias totales.
- **Tablas involucradas (`FROM` y `JOIN`):**
  - `Evento (e)`: Contiene información sobre los eventos celebrados, como su identificador, localidad y disciplina.
  - `CompraEntrada (ce)`: Registra las compras de entradas realizadas para los eventos.
  - **Unión (`JOIN`)**: Se realiza un `JOIN` entre ambas tablas utilizando la relación `e.IDEvento = ce.IDEvento`, asegurando que solo se consideren las entradas compradas para eventos específicos.
- **Agrupación de resultados (`GROUP BY`):**
  - La agrupación se realiza por las siguientes columnas:
    - `e.IDEvento`: Para agrupar las ganancias por cada evento específico.
    - `e.NombreLocalidad`: Para asociar las ganancias con la localidad donde se celebró el evento.
    - `e.IDDisciplina`: Para distinguir las ganancias según la disciplina deportiva del evento.
  - Esto permite calcular la suma de los precios de las entradas (`SUM(e.Precio)`) de manera independiente para cada combinación de evento, localidad y disciplina.
- **Ordenamiento de resultados (`ORDER BY`):**
  - Los resultados se ordenan por la columna `GananciasTotales` en orden descendente (`DESC`), de modo que los eventos con mayores ganancias aparezcan primero.

Análisis detallado

- **Relación entre tablas:**

- La consulta asume que existe una relación directa entre las tablas `Evento` y `CompraEntrada` mediante la clave foránea `ce.IDEvento`, que apunta a `e.IDEvento`.
- Esto implica que cada entrada comprada está asociada a un único evento y que un evento puede tener múltiples entradas compradas.

- **Uso de la función agregada `SUM`:**
  - La función `SUM(e.Precio)` calcula la suma total de los precios de las entradas vendidas para cada evento.
  - Se supone que el campo `Precio` en la tabla `Evento` representa el precio de una entrada individual y que este valor se multiplica implícitamente por el número de entradas compradas en la tabla `CompraEntrada`.
- **Agrupación por columnas clave:**
  - La agrupación por `e.IDEvento`, `e.NombreLocalidad` y `e.IDDisciplina` asegura que las ganancias se calculen de manera específica para cada combinación de:
    - Evento único (`e.IDEvento`).
    - Localidad donde se celebró el evento (`e.NombreLocalidad`).
    - Disciplina deportiva asociada al evento (`e.IDDisciplina`).
- **Ordenamiento por ganancias:**
  - Ordenar los resultados por `GananciasTotales` en orden descendente permite identificar fácilmente los eventos más rentables.

Posibles escenarios y consideraciones

- **Eventos sin entradas vendidas:**
  - Si un evento no tiene entradas vendidas, no aparecerá en los resultados debido al `JOIN`. Esto significa que solo se mostrarán eventos con al menos una entrada comprada.
- **Localidades y disciplinas:**
  - Los resultados permiten identificar no solo los eventos más rentables, sino también qué localidades y disciplinas deportivas generan mayores ingresos.
- **Empates en las ganancias:**
  - Si dos o más eventos tienen las mismas ganancias totales, el orden relativo entre ellos no está definido. Esto no afecta el propósito principal de la consulta, pero podría ser relevante en algunos análisis.

La consulta está diseñada para calcular las ganancias totales generadas por cada evento, agrupadas por localidad y disciplina. Es útil para identificar eventos, localidades y disciplinas con mayor rentabilidad, lo que puede ser clave para la planificación de futuros eventos deportivos.

- **Consulta 7:**

```sql
/* Consulta 7: Cantidad de medallas ganadas por cada país.
 *
 * Esta consulta obtiene la cantidad de medallas ganadas por cada país, desglosadas por tipo de medalla.
 *
 * - Selecciona el nombre del país, el tipo de medalla y cuenta la cantidad de medallas ganadas.
 * - Agrupa los resultados por el país y el tipo de medalla.
 * - Ordena los resultados por el nombre del país y el tipo de medalla.
 */

SELECT
    p.NombrePais,
    m.TipoMedalla,
    COUNT(m.TipoMedalla) AS CantidadMedallas
FROM
    Medalla m
JOIN
    Atleta a ON m.IDAtleta = a.IDAtleta
JOIN
    Pais p ON a.NombrePais = p.NombrePais
GROUP BY
    p.NombrePais, m.TipoMedalla
ORDER BY
    p.NombrePais, m.TipoMedalla
```

nombrepais	tipomedalla	cantidadmedallas
Albania	Bronce	1
Alemania	Oro	1
Argelia	Bronce	1
Argentina	Plata	1
Australia	Bronce	1
Bahamas	Oro	2
Bangladés	Bronce	1
Barbados	Plata	1
Baréin	Oro	1
Baréin	Plata	1
Belice	Plata	1

Consulta 7: Cantidad de medallas ganadas por cada país.

Propósito de la consulta

La consulta tiene como objetivo obtener la cantidad de medallas ganadas por cada país, desglosadas por tipo de medalla (oro, plata o bronce). Esto proporciona una visión detallada del desempeño de cada nación en términos de premios obtenidos.

Desglose de la consulta

- **Selección de columnas (`SELECT`):**
  - `p.NombrePais`: Nombre del país al que pertenece el atleta ganador de la medalla.
  - `m.TipoMedalla`: Tipo de medalla ganada (oro, plata o bronce).
  - `COUNT(m.TipoMedalla) AS CantidadMedallas`: Cuenta la cantidad total de medallas ganadas por cada país, desglosadas según el tipo de medalla.
- **Tablas involucradas (`FROM` y `JOIN`):**
  - `Medalla (m)`: Contiene información sobre las medallas ganadas, incluyendo el tipo de medalla y el atleta que la ganó.
  - `Atleta (a)`: Contiene información sobre los atletas, incluyendo su país de origen.
  - `Pais (p)`: Contiene información sobre los países y sus nombres.
  - **Uniones (`JOIN`):**
    - Se realiza un `JOIN` entre `Medalla` y `Atleta` mediante la relación `m.IDAtleta = a.IDAtleta`, para asociar cada medalla con el atleta que la ganó.
    - Se realiza otro `JOIN` entre `Atleta` y `Pais` mediante la relación `a.NombrePais = p.NombrePais`, para asociar cada atleta con su país de origen.
- **Agrupación de resultados (`GROUP BY`):**
  - La agrupación se realiza por:
    - `p.NombrePais`: Para obtener los resultados específicos de cada país.
    - `m.TipoMedalla`: Para desglosar las medallas ganadas por tipo (oro, plata o bronce).
  - Esto permite contar las medallas de manera independiente para cada combinación de país y tipo de medalla.
- **Ordenamiento de resultados (`ORDER BY`):**
  - Los resultados se ordenan por:
    - `p.NombrePais`: Los países se listan en orden alfabético.
    - `m.TipoMedalla`: Dentro de cada país, los tipos de medalla se ordenan alfabéticamente (por ejemplo, bronce, oro, plata).

Análisis detallado

- **Relación entre tablas:**
  - La consulta utiliza tres tablas relacionadas:
    - ◇ `Medalla`: Relaciona las medallas ganadas con los atletas que las obtuvieron.
    - ◇ `Atleta`: Relaciona a los atletas con sus países de origen.
    - ◇ `Pais`: Proporciona información sobre los países.
  - Las relaciones entre las tablas permiten obtener una asociación entre las medallas ganadas y los países correspondientes.
- **Uso de la función agregada `COUNT`:**
  - La función `COUNT(m.TipoMedalla)` cuenta la cantidad de medallas ganadas para cada combinación de país y tipo de medalla.
  - Esto permite obtener un desglose detallado de las medallas (oro, plata y bronce) ganadas por cada país.
- **Agrupación por columnas clave:**
  - La agrupación por `p.NombrePais` y `m.TipoMedalla` asegura que las medallas se cuenten de manera específica para cada país y tipo de medalla.
- **Ordenamiento por país y tipo de medalla:**
  - Ordenar los resultados por `p.NombrePais` en orden alfabético facilita la lectura y comparación entre países.
  - Ordenar por `m.TipoMedalla` dentro de cada país organiza los resultados por tipo de medalla (bronce, oro, plata).

Posibles escenarios y consideraciones

- **Tipos de medalla:**
  - Si un país no ha ganado un tipo específico de medalla (por ejemplo, ninguna medalla de oro), ese tipo no aparecerá en los resultados para dicho país.
- **Empates en la cantidad de medallas:**
  - Si dos países tienen la misma cantidad de medallas para un tipo específico, el orden relativo entre ellos no está definido. Esto no afecta el propósito principal de la consulta.

La consulta está diseñada para calcular la cantidad total de medallas ganadas por cada país, desglosadas por tipo de medalla. Es útil para analizar el desempeño de las naciones en términos de premios obtenidos, permitiendo identificar cuáles son las más exitosas en cada categoría.

- **Consulta 8:**
- **Consulta 9:**
- **Consulta 10:**
- **Consulta 11:**
- **Consulta 12:**

```sql
/* Consulta 12: Árbitros asignados por disciplina y su nacionalidad.
 *
 * Esta consulta muestra los árbitros que están asignados a cada disciplina junto con su nacionalidad.
 *
 * - Selecciona el nombre de la disciplina, el nombre completo del árbitro y su nacionalidad.
 * - Agrupa los resultados por disciplina y árbitro.
 * - Ordena los resultados por disciplina y nacionalidad.
 */
SELECT
    d.NombreDisciplina,
    ar.Nombre AS NombreArbitro,
    ar.PrimerApellido,
    ar.segundoapellido,
    ar.Nacionalidad
FROM
    Disciplina d
JOIN
    Arbitro ar ON d.IDDisciplina = ar.IDDisciplina
ORDER BY
    d.NombreDisciplina, ar.Nacionalidad
```

nombrediscipline	nombrearbitro	primerapellido	segundoapellido	nacionalidad
Aikido	Bernie	MacGowan	Savil	Holandesa
Aikido	Frederic	Wearing	Sare	Húngara
Aikido	Barris	Demcik	Inchcomb	Pakistaní
Aikido	Lea	Culshew	Travers	Polaca
Aikido	Shalna	Farquar	Malone	Turca
Ajedrez	Etheline	Pople	Urrey	Belga
Ajedrez	Rubia	Caras	Smewin	Israelí
Ajedrez	Korella	Moxted	Barkway	Sueca
Ala delta	Karrie	Frizzell	Maskall	Belga
Ala delta	Deonne	Whetton	Slaney	Libanesa
Ala delta	Nora	Jovic	Behrendsen	Mexicana
Ala delta	Ynes	Belsham	Weymouth	Rumana
Ala delta	Daisey	Hissie	Kauffman	Suiza
Apnea	Warner	Soares	Don	Austriaca
Apnea	Chet	Van de Castele	Bettington	Española
Apnea	Ingaberg	Pietzner	Bloy	Georgiana
Apnea	Fred	Brack	Meir	Kuwaití
Atletismo	Retha	Brookwell	Scouler	Holandesa
Atletismo	Eugene	Hasty	Melesk	Noruega
Atletismo	Sargent	Lamplough	Sorton	Venezolana

Consulta 12. Árbitros asignados por disciplina y su nacionalidad.

Propósito de la consulta

El objetivo de esta consulta es obtener una lista de los árbitros asignados a cada disciplina, incluyendo su nacionalidad. Esto permite analizar la distribución de los árbitros entre las disciplinas y verificar la diversidad cultural representada en el equipo de árbitros.

Desglose de la consulta

- **Selección de columnas (`SELECT`)**:
  - `d.NombreDisciplina`: Nombre de la disciplina deportiva.
  - `ar.Nombre`: Nombre del árbitro.
  - `ar.PrimerApellido` y `ar.SegundoApellido`: Apellidos del árbitro.
  - `ar.Nacionalidad`: Nacionalidad del árbitro.
- **Tablas involucradas (`FROM` y `JOIN`)**:
  - `Disciplina (d)`: Contiene información sobre las disciplinas deportivas.
  - `Arbitro (ar)`: Contiene información sobre los árbitros.
  - Se realiza un `JOIN` entre ambas tablas usando la relación `d.IDDisciplina = ar.IDDisciplina`, lo que asocia cada árbitro con su respectiva disciplina.
- **Ordenamiento de resultados (`ORDER BY`)**:
  - Los resultados se ordenan primero por `d.NombreDisciplina` (nombre de la disciplina) y luego por `ar.Nacionalidad` (nacionalidad del árbitro).

Análisis detallado

1. **Relación entre tablas:**
   - Existe una relación directa entre las tablas `Disciplina` y `Arbitro` a través de la clave foránea `ar.IDDisciplina`, que apunta a `d.IDDisciplina`.
   - Esto implica que cada árbitro está asignado a una única disciplina.
2. **Uso de columnas seleccionadas:**
   - Se seleccionan tanto datos descriptivos de las disciplinas como la información personal y de nacionalidad de los árbitros, para proporcionar un contexto completo de las asignaciones.
3. **Ordenamiento:**

- El ordenamiento jerárquico (por disciplina y nacionalidad) facilita la visualización de los árbitros por disciplina y permite detectar rápidamente patrones o diversidad nacional en cada deporte.

Consideraciones

- **Árbitros sin asignación:**
  - La consulta no incluye árbitros que no estén asignados a una disciplina, debido a la naturaleza del `JOIN`.
- **Empates en la nacionalidad:**
  - Si varios árbitros de la misma disciplina comparten nacionalidad, el orden entre ellos no está definido. Se puede agregar un criterio adicional en el `ORDER BY`, como el nombre completo del árbitro.

Utilidad de la consulta

Esta consulta es útil para:

- Monitorear la asignación de árbitros por disciplina y garantizar una distribución equitativa de recursos humanos.
- Identificar posibles carencias o exceso de árbitros en una disciplina específica.
- Analizar la diversidad nacional de los árbitros, lo que puede ser un indicador importante en eventos deportivos internacionales.
- Facilitar la planeación logística y la gestión de recursos para competencias futuras.

- **Consulta 13:**

```
/* Consulta 13: Entradas vendidas por disciplina en un rango de fechas.
 *
 * Esta consulta calcula cuántas entradas se han vendido por disciplina en un rango específico de fechas.
 *
 * - Selecciona el nombre de la disciplina y la cantidad de entradas vendidas.
 * - Filtra los eventos por la fecha en que ocurrieron.
 * - Agrupa los resultados por disciplina.
 * - Ordena los resultados por la cantidad de entradas vendidas.
 */

SELECT
    d.NombreDisciplina,
    COUNT(ce.idcliente) AS EntradasVendidas
FROM
    Evento e
JOIN
    CompraEntrada ce ON e.IDEvento = ce.IDEvento
JOIN
    Disciplina d ON e.IDDisciplina = d.IDDisciplina
WHERE
    e.FechaEvento BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY
    d.NombreDisciplina
ORDER BY
    EntradasVendidas DESC
```

nombrediscplina	entradasvendidas
Petanca	38
Ecuestre	36
Kitesurf	35
Motocross	35
Windsurf	34
Kayak	34
Golf	34
Bodyboard	33
Ciclismo	31
Crossfit	30
Gimnasia	30
Fútbol gaélico	29
Patinaje artístico	29
Tiro con arco en sala	29
Tenis	28
MotoGP	28
Tenis de mesa	28

Consulta 13. Entradas vendidas por disciplina en un rango de fechas.

Propósito de la consulta

El objetivo de esta consulta es determinar la cantidad de entradas vendidas por disciplina durante un rango específico de fechas. Esto permite analizar la popularidad de las disciplinas y apoyar la toma de decisiones en la planificación de futuros eventos.

Desglose de la consulta

- **Selección de columnas (`SELECT`):**
  - `d.NombreDisciplina`: Nombre de la disciplina deportiva.
  - `COUNT(ce.IDCliente)`: Calcula la cantidad de entradas vendidas para cada disciplina. Esta columna se denomina `EntradasVendidas`.
- **Tablas involucradas (`FROM` y `JOIN`):**
  - `Evento (e)`: Contiene información sobre los eventos deportivos.
  - `CompraEntrada (ce)`: Contiene información sobre las compras de entradas.
  - `Disciplina (d)`: Contiene información sobre las disciplinas deportivas.
  - Se realizan los siguientes `JOINs`:
    - `Evento` con `CompraEntrada` usando `e.IDEvento = ce.IDEvento`, para relacionar las entradas con los eventos.
    - `Evento` con `Disciplina` usando `e.IDDisciplina = d.IDDisciplina`, para asociar los eventos con las disciplinas correspondientes.
- **Filtrado de datos (`WHERE`):**
  - Se filtran los eventos cuya fecha (`e.FechaEvento`) esté dentro del rango especificado: entre el 1 de enero y el 31 de diciembre de 2025.
- **Agrupación de resultados (`GROUP BY`):**
  - Los resultados se agrupan por `d.NombreDisciplina`, para calcular la cantidad total de entradas vendidas por cada disciplina.
- **Ordenamiento de resultados (`ORDER BY`):**
  - Los resultados se ordenan en orden descendente (`DESC`) según la cantidad de entradas vendidas (`EntradasVendidas`), mostrando primero las disciplinas más populares.

Análisis detallado

1. **Relación entre tablas:**
   - Existe una relación entre las tablas `Evento`, `CompraEntrada` y `Disciplina`:
     - Cada entrada comprada se asocia con un evento a través de `IDEvento`.
     - Cada evento está vinculado a una disciplina mediante `IDDisciplina`.
2. **Cálculo de entradas vendidas:**
   - La función agregada `COUNT(ce.IDCliente)` cuenta el número de entradas vendidas asociadas con cada disciplina.
3. **Filtrado por rango de fechas:**
   - El filtro en el `WHERE` asegura que solo se incluyan eventos ocurridos en 2025, excluyendo datos fuera de este rango temporal.
4. **Ordenamiento:**
   - Ordenar por `EntradasVendidas DESC` permite identificar las disciplinas con mayor éxito en ventas de entradas.

Consideraciones

- **Eventos sin ventas:**
  - Si una disciplina no tuvo ventas de entradas, no aparecerá en los resultados.
- **Empates en las ventas:**
  - Si dos disciplinas tienen la misma cantidad de entradas vendidas, el orden relativo entre ellas no está definido. Se podría agregar un criterio adicional en el `ORDER BY`, como el nombre de la disciplina.

Utilidad de la consulta

Esta consulta es útil para:

- Evaluar la popularidad de las disciplinas en función de las ventas de entradas.
- Identificar disciplinas que podrían necesitar estrategias de promoción o mejor planificación logística.
- Ayudar en la asignación de recursos y espacios para futuras competencias basadas en la demanda histórica.
- Determinar patrones de participación del público durante un rango de fechas específico.

- **Consulta 14:**
- **Consulta 15:**
- **Consulta Extra:**

Capítulo 5

Conclusiones

- **Conclusiones técnicas:**

  Hemos notado la importancia de contar con un buen diseño de base de datos, ya que esto nos permitió establecer una base sólida para el desarrollo. Además, los errores en el diseño se traducen en un alto costo en tiempo y esfuerzo. Apreciamos que existan herramientas como estas, ya que nos proporcionan una mejor comprensión de la base de datos y la información que gestionamos.

  Creemos que nuestra base de datos puede mejorarse y escalarse fácilmente, lo que permite agregar nuevas funcionalidades y adaptarse tanto a nuevas reglas de negocio como a cambios en las reglas actuales.

  Personalmente (Juan Sosa), estoy muy agradecido por el uso de triggers y procedimientos almacenados, ya que en la primera iteración del proyecto, al poblar con datos de prueba, no implementamos validaciones y quedaron datos inconsistentes. El tener que rehacer este proceso utilizando Python para validar fue muy difícil, por lo que contar con estas herramientas que lo hacen automáticamente es un verdadero alivio.

- **Conclusiones funcionales:**

  Tener una base de datos que considere todos los datos necesarios facilitará en gran medida el trabajo del usuario final al manipular y consultar la información. Creemos que este proyecto puede generar mejoras sustanciales en el sistema. Asimismo, es importante destacar que, al implementar un sistema de registros históricos y, si se añaden roles y permisos, podemos obtener un sistema mucho más seguro y confiable.

- **Conclusiones sobre los datos:**

  Es inevitable pensar que, con tantos datos disponibles, será mucho más fácil utilizar técnicas de análisis de datos para obtener información valiosa. Actualmente, podemos obtener el podio de los países con mejor desempeño, pero también podríamos analizar otras tendencias, como las disciplinas que practican estos países y su relación con el desempeño.

  Además, al contar con un registro directo de los eventos, podríamos analizar su impacto económico y social, lo que nos permitiría ajustar y mejorar la organización de los próximos juegos o eventos.

- **Conclusiones generales:**

Creemos que este tipo de desarollos pueden ser altamente beneficiosos para cualquier organización que maneje una gran cantidad de información. La implementación de una base de datos bien diseñada puede mejorar significativamente la eficiencia y la eficacia de la organización.

Capítulo 6

Anexos

Además de la información contenida en este documento, se adjunta una serie de archivos que dan mas detalle y concretan la información presentada en los capítulos anteriores. Es escencial que se revise la información contenida en estos archivos para una mejor comprensión del proyecto.

La lista de archivos incluye los siguientes:

1. **Modelo Entidad-Relación**
2. **Modelo Relacional**
3. **DDL (creación de tablas)**
4. **Disparadores**
5. **Procedimientos almacenados**
6. **DML (inserción de datos ejemplo)**
7. **Query (consultas ejemplo)**
8. **Pruebas de funcionalidad**
9. **Diccionario de datos**
10. **Decisiones de diseño**

Finalmente, si se desea crear una base de datos con la información contenida en los archivos adjuntos, se recomienda abrir los archivos en dbeaver y ejecutar en el orden en el que se presentan en la lista (solo archivos en terminación .sql o que esten en la carpeta SQL).