

Informe de Base de Datos Relacional

Presentado Por:

Maicol Sebastián Guerrero López

Presentado A:

Ing. Brayan Ignacio Arcos Burbano

Instituto Tecnológico Del Putumayo

Facultad De Ingeniería Y Ciencias Básicas

Ingeniería De Sistemas

Bases De Datos Y Almacenamiento Masivo

Mocoa

2024

Índice

Resumen Ejecutivo	4
Introducción.....	5
Metodología	6
MySQL Server	6
MySQL Workbench.....	6
Comandos SQL	7
Procedimientos	8
Creación de la Base de Datos	8
Definición de la Estructura de la Base de Datos.....	8
Inserción de Datos de Prueba	8
Ejecución de Consultas SQL	9
Base de Datos sistema_cursos	10
1. Crear la Base de Datos:	10
2. Estructura de las Tablas	10
1. Tabla cursos.....	10
2. Tabla estudiantes	11
3. Tabla status_inscripciones	11
4. Tabla inscripciones.....	12
3. Relaciones entre las Tablas.....	12
4. Claves Primarias y Foráneas.....	13
5. DML: Población de la Base de Datos	13
1. Inserción de datos	13
1. Agregar cursos.....	13
2. Agregar estudiantes.....	14
3. Agregar estados de inscripción.....	14
4. Registrar inscripciones.....	14
Consultas SQL	15
1. Obtener todos los cursos con los estudiantes inscritos y su estado de inscripción	
15	
Resultados:	16

2. Obtener la cantidad de estudiantes inscritos en cada curso.....	16
Resultados:	16
3. Listar los estudiantes inscritos en cursos que tengan más de 25 horas de duración.....	17
Resultados:	17
4. Consultar los cursos que no tienen inscripciones activas (cursos sin estudiantes actualmente).....	18
Resultados:	18
5. Obtener los estudiantes que tienen inscripciones en más de un curso	19
Resultados:	19
6. Obtener los estudiantes cuyo curso esté en estado "Aprobada"	20
Resultados:	20
7. Listar los estudiantes y el curso más reciente en el que se inscribieron	21
Resultados:	21
8. Obtener todos los cursos que tienen inscripciones "Canceladas"	22
Resultados:	22
9. Listar los cursos activos (isActivo = 1) con inscripciones pendientes	23
Resultados:	23
Análisis.....	24
Diseño del Esquema.....	24
Creación de Tablas y Datos	24
Consultas SQL.....	24
Conclusiones.....	25
Eficiencia del Diseño	25
Integridad de los Datos.....	25
Flexibilidad y Escalabilidad	25
Referencias	26

Resumen Ejecutivo

Este informe describe la creación de una base de datos relacional en MySQL para gestionar un sistema de cursos en una institución educativa. La base de datos, llamada `sistema_cursos`, consta de cuatro tablas principales: `cursos`, `estudiantes`, `inscripciones`, y `status_inscripciones`. Se detallan las relaciones entre estas tablas y se proporcionan ejemplos de inserción y consultas SQL relacionadas con la gestión de cursos, estudiantes y estados de inscripción. También se presentan los resultados de consultas clave, como la cantidad de estudiantes inscritos por curso y los cursos sin inscripciones activas.

El análisis final demuestra cómo la base de datos facilita una gestión eficiente y flexible de la información, optimizando el acceso a datos importantes como la duración de los cursos, los estudiantes activos y los estados de inscripción.

Introducción

Este informe documenta el diseño y creación de la base de datos sistema_cursos, destinada a gestionar información sobre cursos, estudiantes e inscripciones. El proyecto explora las capacidades de bases de datos relacionales en MySQL, aplicando principios de diseño, normalización y relaciones entre tablas.

La base de datos incluye las tablas cursos, estudiantes, status_inscripciones e inscripciones, con claves primarias y foráneas para garantizar la integridad referencial. Este sistema permite realizar operaciones clave, como creación, actualización y eliminación de registros, y consultas complejas para obtener datos valiosos sobre la gestión de cursos y estudiantes.

El informe explica cada paso en la creación de la base de datos y las consultas SQL utilizadas, mostrando cómo el diseño facilita la administración de un sistema educativo eficaz y escalable.

Metodología

Para la creación, manipulación y visualización de la base de datos, se emplearon las siguientes herramientas:

MySQL Server

Se utilizó como la plataforma de base de datos subyacente. Este software permite el almacenamiento, gestión y consulta de datos en la base de datos de manera eficiente y confiable. MySQL Server proporciona el entorno necesario para ejecutar los comandos SQL y gestionar la base de datos.

MySQL Workbench

Se empleó para facilitar la creación, diseño y administración visual de la base de datos. Workbench es una herramienta gráfica que permite diseñar el esquema de la base de datos, definir tablas, establecer relaciones y realizar consultas SQL de manera intuitiva. También proporciona capacidades para modelar datos y visualizar el diseño de la base de datos de manera gráfica.

Comandos SQL

Se utilizaron para definir la estructura de la base de datos, insertar datos y ejecutar consultas. Estos comandos permitieron la creación de tablas, la inserción de datos de prueba, y la realización de consultas para extraer información relevante.



Procedimientos

Creación de la Base de Datos

- **Configuración inicial del entorno en MySQL Server Workbench:** Se creó una nueva base de datos para el sistema de gestión de cursos, asegurando que el entorno estuviera listo para definir y gestionar las tablas necesarias.

Definición de la Estructura de la Base de Datos

- **Diseño del Esquema:** Se identificaron y definieron las tablas principales: cursos, estudiantes, status_inscripciones e inscripciones. Cada tabla fue diseñada con campos específicos y tipos de datos apropiados para almacenar la información relevante.
- **Establecimiento de Relaciones:** Se definieron las relaciones entre las tablas utilizando claves primarias y foráneas. Esto garantiza la integridad referencial y la correcta interconexión de los datos.

Inserción de Datos de Prueba

- **Adición de Datos:** Se realizaron inserciones de datos de prueba en las tablas creadas. Estos datos simulan un entorno real y permiten verificar la integridad y funcionalidad de la base de datos.
- **Validación:** Se verificó que los datos se almacenaran correctamente y que las relaciones entre tablas funcionaran según lo esperado.

Ejecución de Consultas SQL

- **Desarrollo de Consultas:** Se elaboraron y ejecutaron consultas SQL para extraer y analizar información relevante sobre cursos, estudiantes y estados de inscripción.
- **Análisis de Resultados:** Se revisaron los resultados de las consultas para asegurar que la base de datos respondiera adecuadamente a las solicitudes y que los datos obtenidos fueran precisos y útiles para la gestión del sistema.

Base de Datos sistema_cursos

La base de datos sistema_cursos está diseñada para gestionar la información relacionada con los cursos, estudiantes y sus inscripciones.

1. Crear la Base de Datos:

Primero, se debe crear la base de datos para el sistema de gestión de cursos. La sentencia SQL para crear la base de datos es:



```
CREATE DATABASE sistema_cursos;  
USE sistema_cursos;
```

2. Estructura de las Tablas

1. Tabla cursos

La tabla cursos almacena los datos de los cursos ofrecidos en el sistema.



```
CREATE TABLE cursos (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(100) NOT NULL,  
  descripcion TEXT NOT NULL,  
  duracion INT NOT NULL,  
  isActive TINYINT DEFAULT 1 NOT NULL,  
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,  
  fecha_modificacion DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL  
);
```

- **id:** Clave primaria, auto-incrementable.
- **nombre:** Nombre del curso, tipo VARCHAR(100).
- **descripcion:** Descripción del curso, tipo TEXT.
- **duracion:** Duración del curso en horas, tipo INT.
- **isActive:** Indica si el curso está activo (1 para activo, 0 para inactivo).
- **fecha_creacion:** Fecha de creación del curso (se establece automáticamente al momento de la inserción).
- **fecha_modificacion:** Se actualiza automáticamente cuando el curso es modificado.

2. Tabla estudiantes

La tabla estudiante gestiona la información personal de los estudiantes.

```
CREATE TABLE estudiantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombres VARCHAR(50) NOT NULL,
  apellidos VARCHAR(50) NOT NULL,
  nro_identificacion VARCHAR(10) NOT NULL UNIQUE,
  email VARCHAR(255) NOT NULL UNIQUE,
  celular VARCHAR(10),
  isActive TINYINT DEFAULT 1 NOT NULL,
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
  fecha_modificacion DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL
);
```

- **id:** Clave primaria, auto-incrementable.
- **nombres:** Nombre del estudiante, tipo VARCHAR(50).
- **apellidos:** Apellido del estudiante, tipo VARCHAR(50).
- **nro_identificacion:** Número de identificación, tipo VARCHAR(10), único.
- **email:** Correo electrónico del estudiante, tipo VARCHAR(150), único.
- **celular:** Número de teléfono del estudiante, tipo VARCHAR(10), opcional.
- **isActive:** Indica si el estudiante está activo.
- **fecha_creacion:** Fecha de creación del registro (se establece automáticamente).
- **fecha_modificacion:** Se actualiza automáticamente cuando se modifica el registro.

3. Tabla status_inscripciones

Esta tabla gestiona los estados posibles de las inscripciones.

```
CREATE TABLE status_inscripciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL UNIQUE
);
```

- **id:** Clave primaria, auto-incrementable.
- **nombre:** Nombre del estado (por ejemplo, "Pendiente", "Aprobada", "Cancelada").

4. Tabla inscripciones

La tabla inscripciones gestiona las inscripciones de los estudiantes en los cursos.

```
CREATE TABLE inscripciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  curso_id INT NOT NULL,
  estudiante_id INT NOT NULL,
  fecha_inscripcion DATE NOT NULL,
  status_id INT NOT NULL,
  fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP NOT NULL,
  fecha_modificacion DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL,
  FOREIGN KEY (curso_id) REFERENCES cursos(id) ON DELETE CASCADE,
  FOREIGN KEY (estudiante_id) REFERENCES estudiantes(id) ON DELETE CASCADE,
  FOREIGN KEY (status_id) REFERENCES status_inscripciones(id) ON DELETE RESTRICT
);
```

- **id:** Clave primaria, auto-incrementable.
- **curso_id:** Clave foránea que referencia a la tabla cursos.
- **estudiante_id:** Clave foránea que referencia a la tabla estudiantes.
- **fecha_inscripcion:** Fecha en la que se inscribió el estudiante.
- **status_id:** Clave foránea que referencia a la tabla status_inscripciones para el estado de la inscripción.
- **fecha_creacion:** Fecha de creación de la inscripción.
- **fecha_modificacion:** Se actualiza automáticamente cuando se modifica la inscripción.

3. Relaciones entre las Tablas

- **Cursos-Estudiantes (Muchos a Muchos):** Los estudiantes pueden inscribirse en varios cursos, y cada curso puede tener varios estudiantes. Esta relación se gestiona a través de la tabla inscripciones, que utiliza las claves foráneas curso_id y estudiante_id para conectar las tablas cursos y estudiantes.
- **Status-Inscripciones (Uno a Muchos):** Cada inscripción tiene un estado, como "Pendiente" o "Aprobada", definido en la tabla status_inscripciones. Las inscripciones están vinculadas a un estado mediante la clave foránea status_id.

4. Claves Primarias y Foráneas

- **Claves Primarias:** Todas las tablas tienen un campo id de tipo INT AUTO_INCREMENT, que garantiza que cada registro sea único.
- **Claves Foráneas:**
 - **curso_id (en inscripciones):** Enlaza con id en cursos. Se elimina en cascada (ON DELETE CASCADE).
 - **estudiante_id (en inscripciones):** Enlaza con id en estudiantes. Se elimina en cascada.
 - **status_id (en inscripciones):** Enlaza con id en status_inscripciones. No se permite la eliminación si está en uso (ON DELETE RESTRICT).

5. DML: Población de la Base de Datos

El Data Manipulation Language (DML) se usa para insertar, actualizar y eliminar información en las tablas.

1. Inserción de datos

1. Agregar cursos

```
INSERT INTO cursos (nombre, descripcion, duracion, isActive) VALUES
('Programación en Python', 'Curso básico de programación en Python', 30, 1),
('Introducción a JavaScript', 'Curso para principiantes en JavaScript', 25, 1),
('Diseño de Bases de Datos', 'Curso sobre modelado y diseño de bases de datos', 40, 1),
('Desarrollo Web con HTML y CSS', 'Curso sobre creación de páginas web con HTML y CSS', 20, 1),
('Machine Learning', 'Curso introductorio a Machine Learning', 50, 1);
```

2. Agregar estudiantes

```
INSERT INTO estudiantes (nombres, apellidos, nro_identificacion, email, celular, isActive) VALUES
('Juan', 'Pérez', '1234567890', 'juan.perez@gmail.com', '3001234567', 1),
('María', 'González', '1234567891', 'maria.gonzalez@gmail.com', '3001234568', 1),
('Carlos', 'Rodríguez', '1234567892', 'carlos.rodriguez@gmail.com', '3001234569', 1),
('Ana', 'López', '1234567893', 'ana.lopez@gmail.com', '3001234570', 1),
('Luis', 'Martínez', '1234567894', 'luis.martinez@gmail.com', '3001234571', 1);
```

3. Agregar estados de inscripción

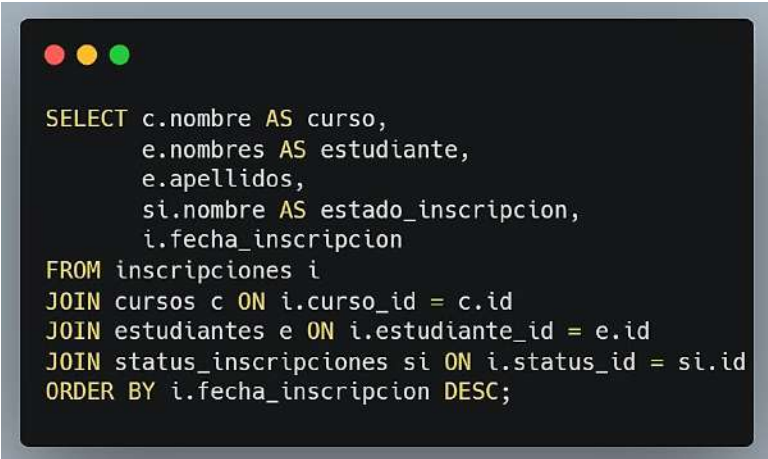
```
INSERT INTO status_inscripciones (nombre) VALUES
('Pendiente'),
('Aprobada'),
('Cancelada');
```

4. Registrar inscripciones

```
INSERT INTO inscripciones (curso_id, estudiante_id, fecha_inscripcion, status_id) VALUES
-- Curso de Programación en Python
(1, 1, '2024-07-01', 1), -- Juan Pérez
(1, 2, '2024-07-02', 1), -- María González
-- Curso de Introducción a JavaScript
(2, 8, '2024-07-14', 2), -- Sergio Vega
(2, 9, '2024-07-15', 3), -- Natalia García
-- Curso de Diseño de Bases de Datos
(3, 3, '2024-07-18', 1), -- Carlos Rodríguez
(3, 4, '2024-07-19', 2), -- Ana López
-- Curso de Desarrollo Web con HTML y CSS
(4, 9, '2024-08-01', 3), -- Natalia García
(4, 12, '2024-08-02', 3); -- Ricardo Salazar
```

Consultas SQL

1. Obtener todos los cursos con los estudiantes inscritos y su estado de inscripción



```
SELECT c.nombre AS curso,
       e.nombres AS estudiante,
       e.apellidos,
       si.nombre AS estado_inscripcion,
       i.fecha_inscripcion
FROM inscripciones i
JOIN cursos c ON i.curso_id = c.id
JOIN estudiantes e ON i.estudiante_id = e.id
JOIN status_inscripciones si ON i.status_id = si.id
ORDER BY i.fecha_inscripcion DESC;
```

- **SELECT:** Esta parte selecciona las columnas que queremos mostrar en el resultado. Usamos alias (AS) para renombrar las columnas por claridad.
- **c.nombre** se muestra como curso.
- **e.nombres** y **e.apellidos** muestran los nombres y apellidos de los estudiantes.
- **si.nombre** muestra el estado de la inscripción, renombrado como **estado_inscripcion**.
- **i.fecha_inscripcion** muestra la fecha en que el estudiante se inscribió.
- **FROM inscripciones i:** La consulta principal toma datos de la tabla inscripciones, que registra las inscripciones de estudiantes a cursos.
- **JOINS:** Los JOIN combinan la tabla inscripciones con otras tablas:
- **JOIN cursos c ON i.curso_id = c.id:** Une las inscripciones con la tabla cursos usando la columna curso_id en inscripciones y la columna id en cursos.
- **JOIN estudiantes e ON i.estudiante_id = e.id:** Une la tabla inscripciones con la tabla estudiantes, usando el campo estudiante_id.
- **JOIN status_inscripciones si ON i.status_id = si.id:** Relaciona las inscripciones con el estado de inscripción.
- **ORDER BY:** Ordena los resultados por i.fecha_inscripcion en orden descendente (DESC), de manera que las inscripciones más recientes aparezcan primero.

Resultados:

curso	estudiante	apellidos	estado_inscripcion	fecha_inscripcion
Desarrollo Web con HTML y CSS	Carolina	Morales	Cancelada	2024-08-02
Desarrollo Web con HTML y CSS	Andrés	Ruiz	Cancelada	2024-08-01
Desarrollo Web con HTML y CSS	Sofía	Jiménez	Aprobada	2024-07-31
Desarrollo Web con HTML y CSS	Miguel	Hernández	Pendiente	2024-07-30
Desarrollo Web con HTML y CSS	Paola	Gutiérrez	Pendiente	2024-07-29
Desarrollo Web con HTML y CSS	Ana	López	Aprobada	2024-07-28
Desarrollo Web con HTML y CSS	Carlos	Rodríguez	Aprobada	2024-07-27
Desarrollo Web con HTML y CSS	María	González	Pendiente	2024-07-26
Diseño de Bases de Datos	Daniel	Suarez	Cancelada	2024-07-25
Diseño de Bases de Datos	Laura	Castillo	Pendiente	2024-07-24
Diseño de Bases de Datos	Andrés	Ruiz	Aprobada	2024-07-23
Diseño de Bases de Datos	Sofía	Jiménez	Aprobada	2024-07-22

2. Obtener la cantidad de estudiantes inscritos en cada curso

```

SELECT c.nombre AS curso,
       COUNT(i.estudiante_id) AS total_estudiantes
FROM cursos c
LEFT JOIN inscripciones i ON c.id = i.curso_id
GROUP BY c.nombre;

```

- **SELECT:** Se selecciona el nombre del curso (c.nombre renombrado como curso) y la cantidad de estudiantes inscritos (COUNT(i.estudiante_id)).
- **COUNT(i.estudiante_id):** Cuenta cuántas veces aparece estudiante_id en la tabla inscripciones para cada curso. Si un curso no tiene estudiantes inscritos, el conteo será 0.
- **FROM cursos c:** Se empieza seleccionando los cursos de la tabla cursos.
- **LEFT JOIN:** Asegura que todos los cursos aparezcan en el resultado, incluso si no tienen inscripciones. Si un curso no tiene inscripciones, el resultado de inscripciones será NULL.
- **GROUP BY c.nombre:** Agrupa los resultados por el nombre del curso. Esto asegura que la función COUNT cuente el número de estudiantes por curso, no de manera global.

Resultados:

curso	total_estudiantes
Programación en Python	8
Introducción a JavaScript	7
Diseño de Bases de Datos	8
Desarrollo Web con HTML y CSS	8
Machine Learning	0
Desarrollo de APIs con Node.js	0
Seguridad Informática	0

3. Listar los estudiantes inscritos en cursos que tengan más de 25 horas de duración

```
SELECT e.nombres,
       e.apellidos,
       c.nombre AS curso,
       c.duracion
FROM estudiantes e
JOIN inscripciones i ON e.id = i.estudiante_id
JOIN cursos c ON i.curso_id = c.id
WHERE c.duracion > 25;
```

- **SELECT:** Muestra los nombres y apellidos de los estudiantes, el nombre del curso y su duración.
- **JOIN:**
- **Une estudiantes e inscripciones mediante estudiante_id.**
- **Une inscripciones y cursos mediante curso_id,** permitiendo obtener el nombre y la duración del curso en el que están inscritos los estudiantes.
- **WHERE c.duracion > 25:** Aplica un filtro para incluir solo cursos cuya duración sea mayor a 25 horas.

Resultados:

nombres	apellidos	curso	duracion
Juan	Pérez	Programación en Python	30
María	González	Programación en Python	30
Carlos	Rodríguez	Programación en Python	30
Ana	López	Programación en Python	30
Luis	Martínez	Programación en Python	30
Paola	Gutiérrez	Programación en Python	30
Miguel	Hernández	Programación en Python	30
Sofía	Jiménez	Programación en Python	30
Carlos	Rodríguez	Diseño de Bases de Datos	40
Ana	López	Diseño de Bases de Datos	40
Luis	Martínez	Diseño de Bases de Datos	40
Miguel	Hernández	Diseño de Bases de Datos	40
Sofía	Jiménez	Diseño de Bases de Datos	40
Andrés	Ruiz	Diseño de Bases de Datos	40
Laura	Castillo	Diseño de Bases de Datos	40
Daniel	Suarez	Diseño de Bases de Datos	40

4. Consultar los cursos que no tienen inscripciones activas (cursos sin estudiantes actualmente)

```
SELECT c.nombre AS curso
FROM cursos c
LEFT JOIN (
    SELECT curso_id
    FROM inscripciones
    WHERE status_id IN (1, 2)
    GROUP BY curso_id
) i ON c.id = i.curso_id
WHERE i.curso_id IS NULL;
```

- **SELECT c.nombre AS curso:** Se selecciona el nombre de los cursos.
- **FROM cursos c:** Se obtiene la lista completa de cursos de la tabla cursos.
- **LEFT JOIN:** Aquí se hace una unión externa izquierda con una subconsulta. El objetivo es que todos los cursos sean incluidos, aunque no tengan inscripciones activas.
- **SELECT curso_id:** Esta subconsulta selecciona los curso_id de la tabla inscripciones donde el estado es 1 o 2 (activa o en proceso), lo que identifica a los cursos con inscripciones activas.
- **ON c.id = i.curso_id:** Une la tabla cursos con los resultados de la subconsulta, asociando los cursos que tienen inscripciones activas.
- **WHERE i.curso_id IS NULL:** Filtra los resultados, mostrando solo los cursos que no tienen inscripciones activas.

Resultados:

curso
Machine Learning
Desarrollo de APIs con Node.js
Seguridad Informática

5. Obtener los estudiantes que tienen inscripciones en más de un curso

```
SELECT e.nombres,  
       e.apellidos,  
       COUNT(i.curso_id) AS cantidad_cursos  
FROM estudiantes e  
JOIN inscripciones i ON e.id = i.estudiante_id  
GROUP BY e.id  
HAVING COUNT(i.curso_id) > 1;
```

- **SELECT:** Selecciona los nombres y apellidos de los estudiantes y cuenta cuántos cursos tienen inscritos usando COUNT(i.curso_id).
- **GROUP BY e.id:** Agrupa los resultados por el ID de cada estudiante. Esto asegura que el conteo de cursos sea específico para cada estudiante.
- **HAVING COUNT(i.curso_id) > 1:** El filtro HAVING solo incluye estudiantes que están inscritos en más de un curso, es decir, cuyo conteo es mayor a 1.

Resultados:

nombres	apellidos	cantidad_cursos
Juan	Pérez	2
María	González	3
Carlos	Rodríguez	3
Ana	López	4
Luis	Martínez	2
Paola	Gutiérrez	2
Miguel	Hernández	4
Sofía	Jiménez	4
Andrés	Ruiz	3
Laura	Castillo	2

6. Obtener los estudiantes cuyo curso esté en estado "Aprobada"

```

SELECT e.nombres,
       e.apellidos,
       c.nombre AS curso,
       si.nombre AS estado
FROM estudiantes e
JOIN inscripciones i ON e.id = i.estudiante_id
JOIN cursos c ON c.id = i.curso_id
JOIN status_inscripciones si ON si.id = i.status_id
WHERE si.nombre = 'Aprobada';

```

- **SELECT e.nombres, e.apellidos, c.nombre AS curso, si.nombre AS estado:** Selecciona los nombres y apellidos del estudiante, el nombre del curso y el estado de la inscripción.
- **FROM estudiantes e:** Obtiene los datos de la tabla estudiantes.
- **JOIN inscripciones i ON e.id = i.estudiante_id:** Se realiza un JOIN para relacionar los estudiantes con las inscripciones, a través del campo estudiante_id.
- **JOIN cursos c ON c.id = i.curso_id:** Se une la tabla de inscripciones con la de cursos para obtener el nombre del curso en el que está inscrito el estudiante.
- **JOIN status_inscripciones si ON si.id = i.status_id:** Une la tabla de inscripciones con la de estados de inscripción, lo que permite obtener el estado de la inscripción.
- **WHERE si.nombre = 'Aprobada':** Filtra las inscripciones para mostrar solo aquellas que están en estado "Aprobada".

Resultados:

nombres	apellidos	curso	estado
Ana	López	Programación en Python	Aprobada
Ana	López	Introducción a JavaScript	Aprobada
Sofía	Jiménez	Introducción a JavaScript	Aprobada
Ana	López	Diseño de Bases de Datos	Aprobada
Sofía	Jiménez	Diseño de Bases de Datos	Aprobada
Andrés	Ruiz	Diseño de Bases de Datos	Aprobada
Carlos	Rodríguez	Desarrollo Web con HTML y CSS	Aprobada
Ana	López	Desarrollo Web con HTML y CSS	Aprobada
Sofía	Jiménez	Desarrollo Web con HTML y CSS	Aprobada

7. Listar los estudiantes y el curso más reciente en el que se inscribieron

```

SELECT e.nombres,
       e.apellidos,
       c.nombre AS curso,
       i.fecha_inscripcion
FROM estudiantes e
JOIN inscripciones i ON e.id = i.estudiante_id
JOIN cursos c ON c.id = i.curso_id
JOIN (
    SELECT estudiante_id, MAX(fecha_inscripcion) AS fecha_mas_reciente
    FROM inscripciones
    GROUP BY estudiante_id
) i_reciente ON i.estudiante_id = i_reciente.estudiante_id
              AND i.fecha_inscripcion = i_reciente.fecha_mas_reciente;

```

- **SELECT e.nombres, e.apellidos, c.nombre AS curso, i.fecha_inscripcion:** Selecciona los nombres y apellidos del estudiante, el nombre del curso y la fecha de inscripción más reciente.
- **FROM estudiantes e:** Parte de la tabla estudiantes.
- **JOIN inscripciones i ON e.id = i.estudiante_id:** Une las tablas estudiantes e inscripciones para obtener las inscripciones de cada estudiante.
- **JOIN cursos c ON c.id = i.curso_id:** Une las inscripciones con los cursos correspondientes.
- **JOIN (subconsulta):** La subconsulta obtiene el estudiante_id y la MAX(fecha_inscripcion) (la fecha más reciente de inscripción) para cada estudiante.
- **ON i.estudiante_id = i_reciente.estudiante_id AND i.fecha_inscripcion = i_reciente.fecha_mas_reciente:** Se asegura de unir solo la inscripción más reciente para cada estudiante.

Resultados:

nombres	apellidos	curso	fecha_inscripcion
Juan	Pérez	Introducción a JavaScript	2024-07-10
María	González	Desarrollo Web con HTML y CSS	2024-07-26
Carlos	Rodríguez	Desarrollo Web con HTML y CSS	2024-07-27
Ana	López	Desarrollo Web con HTML y CSS	2024-07-28
Luis	Martínez	Diseño de Bases de Datos	2024-07-20
Paola	Gutiérrez	Desarrollo Web con HTML y CSS	2024-07-29
Miguel	Hernández	Desarrollo Web con HTML y CSS	2024-07-30
Sofía	Jiménez	Desarrollo Web con HTML y CSS	2024-07-31
Andrés	Ruiz	Desarrollo Web con HTML y CSS	2024-08-01
Laura	Castillo	Diseño de Bases de Datos	2024-07-24
Daniel	Suarez	Diseño de Bases de Datos	2024-07-25
Carolina	Morales	Desarrollo Web con HTML y CSS	2024-08-02

8. Obtener todos los cursos que tienen inscripciones "Canceladas"

```
SELECT c.nombre AS curso,  
       COUNT(i.id) AS total_canceladas  
FROM cursos c  
JOIN inscripciones i ON c.id = i.curso_id  
JOIN status_inscripciones si ON i.status_id = si.id  
WHERE si.nombre = 'Cancelada'  
GROUP BY c.nombre;
```

- **SELECT c.nombre AS curso, COUNT(i.id) AS total_canceladas:** Selecciona el nombre del curso y cuenta cuántas inscripciones canceladas (COUNT(i.id)) tiene.
- **FROM cursos c:** Parte de la tabla cursos.
- **JOIN inscripciones i ON c.id = i.curso_id:** Une las tablas cursos e inscripciones para obtener las inscripciones de cada curso.
- **JOIN status_inscripciones si ON i.status_id = si.id:** Une las inscripciones con los estados de inscripción.
- **WHERE si.nombre = 'Cancelada':** Filtra los registros para que solo se consideren las inscripciones que están en estado "Cancelada".
- **GROUP BY c.nombre:** Agrupa los resultados por nombre de curso para contar cuántas inscripciones canceladas tiene cada curso.

Resultados:

curso	total_canceladas
Programación en Python	2
Introducción a JavaScript	1
Diseño de Bases de Datos	1
Desarrollo Web con HTML y CSS	2

9. Listar los cursos activos (isActive = 1) con inscripciones pendientes

```

SELECT c.nombre AS curso,
       COUNT(i.id) AS inscripciones_pendientes
FROM cursos c
JOIN inscripciones i ON c.id = i.curso_id
JOIN status_inscripciones si ON i.status_id = si.id
WHERE c.isActive = 1 AND si.nombre = 'Pendiente'
GROUP BY c.nombre;

```

- **SELECT c.nombre AS curso, COUNT(i.id) AS inscripciones_pendientes:** Selecciona el nombre del curso y cuenta cuántas inscripciones pendientes (COUNT(i.id)) tiene.
- **FROM cursos c:** Parte de la tabla cursos.
- **JOIN inscripciones i ON c.id = i.curso_id:** Une las tablas cursos e inscripciones para obtener las inscripciones de cada curso.
- **JOIN status_inscripciones si ON i.status_id = si.id:** Une las inscripciones con los estados de inscripción.
- **WHERE c.isActive = 1:** Filtra los resultados para incluir solo los cursos activos.
- **AND si.nombre = 'Pendiente':** Filtra los registros para considerar solo las inscripciones con estado "Pendiente".
- **GROUP BY c.nombre:** Agrupa los resultados por nombre de curso para contar cuántas inscripciones pendientes tiene cada curso.

Resultados:

curso	inscripciones_pendientes
Programación en Python	5
Introducción a JavaScript	4
Diseño de Bases de Datos	4
Desarrollo Web con HTML y CSS	3

Análisis

Diseño del Esquema

El esquema de la base de datos, compuesto por las tablas cursos, estudiantes, status_inscripciones e inscripciones, está diseñado para manejar relaciones complejas de manera eficiente. La normalización del esquema minimiza redundancias y asegura la integridad referencial. Cada tabla cumple con una función específica y las relaciones entre ellas están bien definidas, lo que garantiza la coherencia de los datos y la eficacia de las consultas.

Creación de Tablas y Datos

La creación de las tablas en MySQL Workbench se realizó sin problemas. La definición de claves primarias y foráneas se ejecutó correctamente, y se utilizaron datos de prueba para validar el funcionamiento de las tablas y su integración. Las claves foráneas aseguran que las actualizaciones y eliminaciones de registros se propaguen adecuadamente, evitando inconsistencias y manteniendo la integridad de las relaciones.

Consultas SQL

Las consultas SQL demostraron la capacidad de la base de datos para extraer información relevante de manera eficaz. Utilizando joins, filtros y agrupaciones, se obtuvieron datos detallados, como el número de estudiantes inscritos en cada curso y la identificación de cursos sin inscripciones activas. Estas consultas confirmaron la funcionalidad del diseño y la correcta implementación de las relaciones entre tablas, asegurando la generación de reportes y análisis útiles para la toma de decisiones.

Conclusiones

Eficiencia del Diseño

El diseño de la base de datos para sistema_cursos es eficiente para gestionar cursos, estudiantes e inscripciones. La estructura permite realizar consultas complejas de manera efectiva, facilitando la obtención de información crucial para la administración de cursos y la gestión de inscripciones.

Integridad de los Datos

La implementación correcta de claves primarias y foráneas garantiza la integridad de los datos y mantiene la coherencia entre las tablas. Esto previene la pérdida de datos y asegura que las operaciones de actualización y eliminación se realicen de manera controlada, manteniendo la base de datos consistente y fiable.

Flexibilidad y Escalabilidad

El sistema está diseñado para ser flexible y escalable, permitiendo la adición de nuevas funcionalidades y la expansión de la base de datos según las necesidades futuras. La estructura modular y las relaciones bien definidas facilitan la adaptación a cambios en los requisitos sin comprometer la integridad de los datos existentes.

Referencias

MySQL, Inc. (n.d.). MySQL Documentation. Retrieved from <https://dev.mysql.com/doc/>

W3Schools. (n.d.). SQL Tutorial. Retrieved from <https://www.w3schools.com/sql/>