

Diseño de Bases de Datos

Table of contents

Diseño de Bases de Datos	2
Clasificación de las Bases de Datos	3
Modelos de Bases de datos	5
El Modelo Entidad-Relación	8
Algebra Relacional	14
Ejemplo de aplicación de álgebra relacional	16
El modelo Relacional de bases de Datos	21
Atributos de las columnas	22
De modelo Entidad-Relación a Modelo Relacional	24

Diseño de Bases de Datos

Introducción

En este curso se estudiarán los conceptos fundamentales de las bases de datos, su diseño y su implementación. Se abordarán los aspectos teóricos y prácticos de las bases de datos relacionales y no relacionales, así como las tecnologías y herramientas más utilizadas en la actualidad.

Objetivos

- Comprender los conceptos fundamentales de las bases de datos.
- Diseñar bases de datos relacionales y no relacionales.
- Implementar bases de datos utilizando tecnologías y herramientas actuales.
- Utilizar lenguajes de consulta para interactuar con bases de datos.
- Conocer las mejores prácticas para el diseño y la implementación de bases de datos.
- Desarrollar habilidades para el modelado de datos y la normalización de bases de datos.

Clasificación de las Bases de Datos

Introducción

Las bases de datos pueden clasificarse de acuerdo a diferentes criterios, como el modelo de datos, la estructura de almacenamiento, la forma de acceso a los datos, entre otros. A continuación, se presentan las clasificaciones más comunes de las bases de datos.

Por Modelo de Datos

Bases de Datos Relacionales

Las bases de datos relacionales son aquellas que se basan en el modelo relacional propuesto por Edgar F. Codd en la década de 1970. En este modelo, los datos se organizan en tablas, donde cada tabla representa una entidad y cada columna de la tabla corresponde a un atributo de la entidad. Las relaciones entre las entidades se establecen mediante claves primarias y claves foráneas.

Bases de Datos No Relacionales

Las bases de datos no relacionales, también conocidas como NoSQL (Not Only SQL), son aquellas que no siguen el modelo relacional. Estas bases de datos permiten almacenar datos de forma más flexible y escalable, lo que las hace ideales para aplicaciones con grandes volúmenes de información o con requisitos de escalabilidad horizontal.

Por Estructura de Almacenamiento

Bases de Datos Centralizadas

Las bases de datos centralizadas son aquellas en las que todos los datos se almacenan en un único servidor o nodo. Este modelo de almacenamiento es sencillo y fácil de administrar, pero puede presentar problemas de escalabilidad y disponibilidad.

Bases de Datos Distribuidas

Las bases de datos distribuidas son aquellas en las que los datos se almacenan en múltiples servidores o nodos interconectados. Este modelo de almacenamiento permite distribuir la carga de trabajo y mejorar la disponibilidad y escalabilidad del sistema, pero puede ser más complejo de administrar.

Por Forma de Acceso a los Datos

Bases de Datos OLTP

Las bases de datos OLTP (Online Transaction Processing) son aquellas diseñadas para el procesamiento de transacciones en línea. Estas bases de datos se caracterizan por admitir un alto volumen de transacciones con tiempos de respuesta rápidos. Son ideales para aplicaciones que requieren un acceso rápido y concurrente a los datos, como sistemas de reservas, sistemas de ventas en línea, entre otros.

Bases de Datos OLAP

Las bases de datos OLAP (Online Analytical Processing) son aquellas diseñadas para el análisis de datos y la generación de informes. Estas bases de datos se caracterizan por admitir consultas complejas y analíticas sobre grandes volúmenes de datos. Son ideales para aplicaciones de inteligencia de negocios, análisis de datos, entre otros.

Por Nivel de Abstracción

Bases de Datos Físicas

Las bases de datos físicas son aquellas que se centran en los detalles de implementación y almacenamiento de los datos. Estas bases de datos se ocupan de aspectos como la estructura de los archivos, los índices, la optimización de consultas, entre otros.

Bases de Datos Lógicas

Las bases de datos lógicas son aquellas que se centran en la representación de los datos y en la forma en que se manipulan. Estas bases de datos se ocupan de aspectos como el modelo de datos, las consultas, las restricciones de integridad, entre otros.

Por Propósito

Bases de Datos Operacionales

Las bases de datos operacionales son aquellas diseñadas para el soporte de las operaciones diarias de una organización. Estas bases de datos se utilizan para el registro y la gestión de transacciones, la actualización de inventarios, la gestión de clientes, entre otros.

Bases de Datos Analíticas

Las bases de datos analíticas son aquellas diseñadas para el análisis de datos y la generación de informes. Estas bases de datos se utilizan para la toma de decisiones estratégicas, la identificación de tendencias, el análisis de comportamientos, entre otros.

Por elementos almacenados

Bases de Datos de Texto

Las bases de datos de texto son aquellas que almacenan y gestionan información textual, como documentos, correos electrónicos, artículos, entre otros. Estas bases de datos son ideales para aplicaciones que requieren el análisis y la búsqueda de texto, como motores de búsqueda, sistemas de recomendación, entre otros.

Bases de Datos Multimedia

Las bases de datos multimedia son aquellas que almacenan y gestionan información multimedia, como imágenes, audio, video, entre otros. Estas bases de datos son ideales para aplicaciones que requieren el almacenamiento y la recuperación de contenido multimedia, como bibliotecas digitales, sistemas de gestión de activos digitales, entre otros.

Por sus permisos de acceso

Bases de Datos de Lectura

Las bases de datos de lectura son aquellas que permiten únicamente la consulta de los datos almacenados, pero no permiten la modificación ni la eliminación de los mismos. Estas bases de datos son ideales para aplicaciones que requieren únicamente la lectura de información, como sistemas de reportes, sistemas de consulta, entre otros.

Bases de Datos de Lectura y Escritura

Las bases de datos de escritura son aquellas que permiten la modificación y la eliminación de los datos almacenados, así como la consulta de los mismos. Estas bases de datos son ideales para aplicaciones que requieren la actualización y la gestión de la información, como sistemas de gestión de inventarios, sistemas de gestión de clientes, entre otros.

Modelos de Bases de datos

Introducción

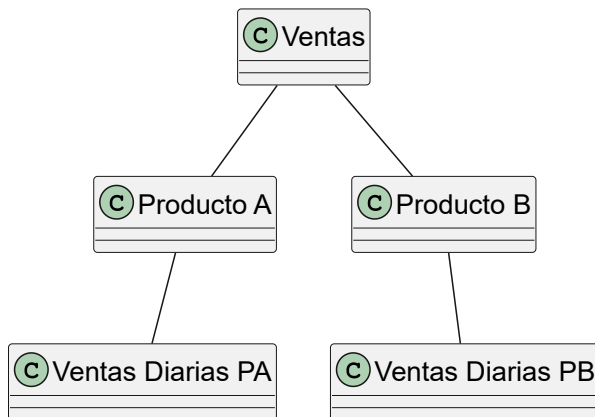
Un modelo de base de datos es una representación abstracta de la estructura de una base de datos. Define la forma en que los datos se organizan y se relacionan entre sí, así como las operaciones que se pueden realizar sobre ellos. Los modelos de bases de datos se utilizan para describir la estructura lógica de una base de datos y proporcionar un marco de referencia para su diseño e implementación.

Modelos de Datos

Los modelos de datos son una parte fundamental de los sistemas de bases de datos. Proporcionan una forma de representar la estructura de los datos y las relaciones entre ellos. Existen varios tipos de modelos de datos, cada uno con sus propias características y ventajas. Algunos de los modelos de datos más comunes son:

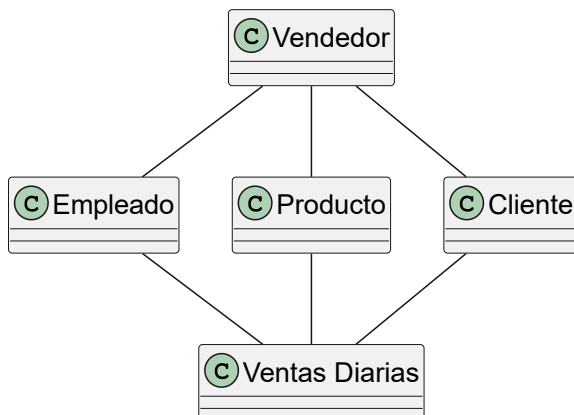
Modelo Jerárquico

El modelo jerárquico organiza los datos en una estructura de árbol, donde cada nodo representa un registro y cada relación padre-hijo representa una relación uno a muchos. Este modelo es eficiente para representar datos con una estructura jerárquica, pero puede resultar complejo de manejar en sistemas con relaciones complejas.



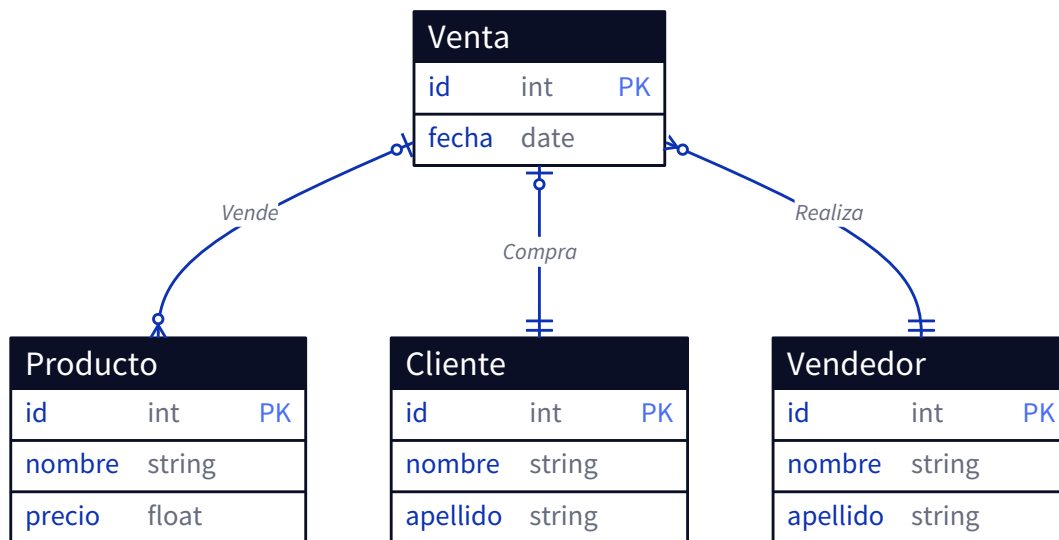
Modelo de Red

El modelo de red es una extensión del modelo jerárquico que permite representar relaciones muchos a muchos entre los registros. En este modelo, los registros se organizan en nodos y las relaciones se representan mediante enlaces. Aunque es más flexible que el modelo jerárquico, puede resultar más complejo de implementar y mantener.



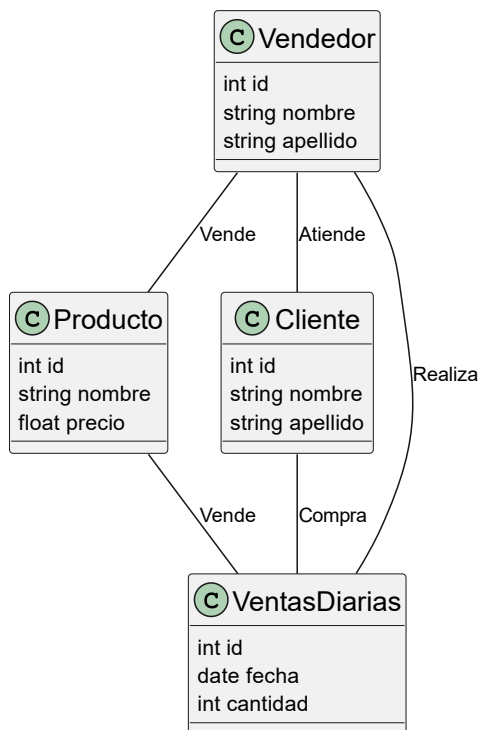
Modelo Relacional

El modelo relacional organiza los datos en tablas bidimensionales, donde cada fila representa un registro y cada columna representa un atributo. Las relaciones entre las tablas se establecen mediante claves primarias y claves foráneas. Este modelo es el más utilizado en la actualidad debido a su simplicidad y flexibilidad.



Modelo de Objetos

El modelo de objetos combina los conceptos de la programación orientada a objetos con las bases de datos. Permite representar los datos como objetos con propiedades y métodos, lo que facilita la representación de estructuras de datos complejas. Este modelo es utilizado en sistemas que requieren una representación fiel de la realidad, como sistemas de información geográfica o sistemas de diseño asistido por computadora.



Conclusiones

Los modelos de bases de datos son una herramienta fundamental en el diseño y la implementación de sistemas de bases de datos. Cada modelo tiene sus propias características y ventajas, por lo que es importante elegir el modelo más adecuado para cada aplicación. Los modelos jerárquico y de red son útiles para representar estructuras jerárquicas y relaciones complejas, mientras que el modelo relacional es el más utilizado en la actualidad debido a su simplicidad y flexibilidad. El modelo de objetos es ideal para representar estructuras de datos complejas y sistemas que requieren una representación fiel de la realidad. Al comprender los diferentes modelos de bases de datos, los diseñadores y desarrolladores pueden tomar decisiones informadas sobre la estructura y el diseño de sus bases de datos.

El Modelo Entidad-Relación

El modelo entidad-relación (ER) es un modelo conceptual de datos que permite representar la información de un sistema de información de manera gráfica. Este modelo se basa en la representación de entidades, atributos y relaciones entre entidades.

Componentes del modelo ER

El modelo entidad-relación se compone de los siguientes elementos:

- **Entidades:** objetos o conceptos del mundo real que pueden ser identificados y descritos mediante atributos.
- **Atributos:** propiedades o características de las entidades.
- **Relaciones:** asociaciones entre entidades que representan interacciones o dependencias entre ellas.
- **Claves:** atributos o conjuntos de atributos que permiten identificar de forma única a una entidad.
- **Cardinalidades:** indican la cantidad de instancias de una entidad que pueden estar relacionadas con una instancia de otra entidad.
- **Participación:** indica si una entidad debe participar o no en una relación.
- **Restricciones de integridad:** reglas que garantizan la consistencia y validez de los datos en la base de datos.
- **Generalización y especialización:** mecanismos para representar la relación de herencia entre entidades.


Entidades

Las entidades pueden clasificarse en dos tipos: entidades fuertes y entidades débiles.

Entidades Fuertes

Una entidad fuerte es una entidad que puede existir por sí misma, es decir, no depende de otra entidad para existir. Por ejemplo, en un sistema de información de una universidad, las entidades fuertes pueden ser Estudiante, Profesor, Curso, etc.

Las entidades fuertes se representan gráficamente mediante un rectángulo con el nombre de la entidad en su interior.



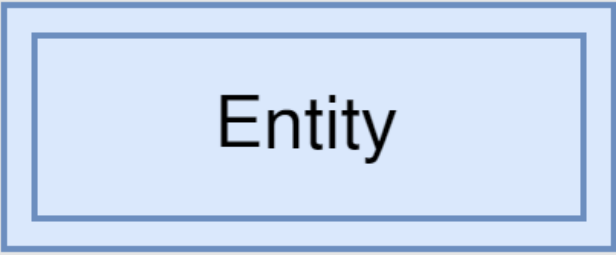
Entity

strong_entity

Entidades Débiles

Una entidad débil es una entidad que depende de otra entidad para existir. Por ejemplo, en un sistema de información de una biblioteca, la entidad Libro puede ser una entidad débil, ya que depende de la entidad Autor para existir.

Las entidades débiles se representan gráficamente mediante un rectángulo inscrito en otro.



Entity

weak_entity

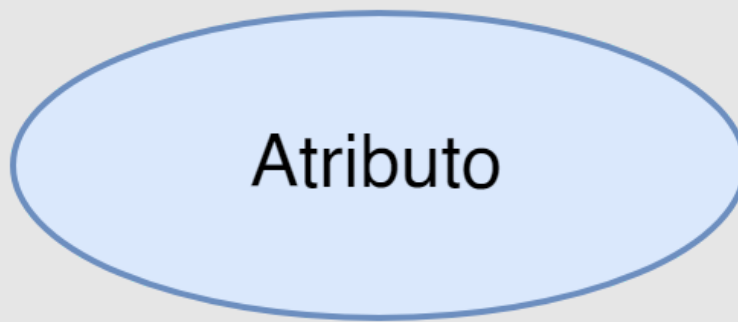
Atributos

Los atributos son las propiedades o características de las entidades. Pueden clasificarse como:

Atributo Simple o Monovaluado

Un atributo simple o monovaluado es un atributo que toma un único valor para cada instancia de la entidad a la que pertenece. Por ejemplo, el atributo Nombre de la entidad Estudiante.

Los atributos simples se representan gráficamente mediante un óvalo conectado al rectángulo de la entidad.



simple_attribute

Atributo Multivaluado

Un atributo multivaluado es un atributo que puede tomar varios valores para cada instancia de la entidad a la que pertenece. Por ejemplo, el atributo Dirección de la entidad Estudiante.

Los atributos compuestos se representan gráficamente mediante un rectángulo con sub-óvalos conectados al rectángulo de la entidad.

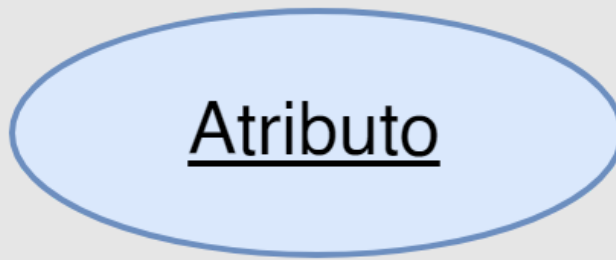


composite_attribute

Atributo Clave o Principal

Un atributo clave es un atributo o conjunto de atributos que permite identificar de forma única a una entidad. Por ejemplo, el atributo Matrícula de la entidad Estudiante.

Los atributos clave se representan gráficamente mediante un óvalo con un subrayado conectado al rectángulo de la entidad.



key_attribute

Atributo Derivado o Calculado

Un atributo derivado es un atributo cuyo valor se obtiene a partir de otros atributos de la entidad. Por ejemplo, el atributo Edad de la entidad Estudiante, que se puede calcular a partir de la fecha de nacimiento.

Los atributos derivados se representan gráficamente mediante un óvalo punteado conectado al rectángulo de la entidad.

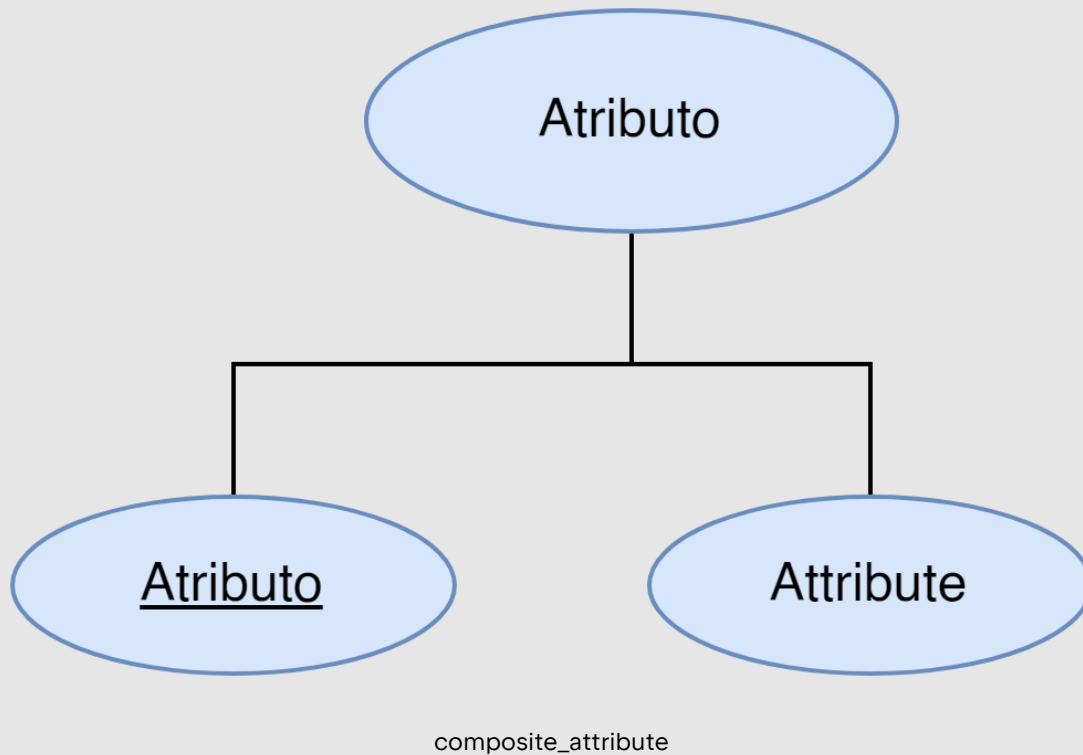


derived_attribute

Atributo Compuesto

Un atributo compuesto es un atributo que se puede descomponer en subatributos más simples. Por ejemplo, el atributo Dirección de la entidad Estudiante, que se puede descomponer en Calle, Número, Colonia, etc.

Los atributos compuestos se representan gráficamente mediante un rectángulo con sub-óvalos conectados al rectángulo de la entidad.



Relaciones

Las relaciones representan las asociaciones entre entidades y se representa con un rombo conectado a las entidades relacionadas. Las relaciones pueden clasificarse según su grado de participación y cardinalidad.

Grado de Participación

El grado de participación de una entidad en una relación indica si la entidad debe participar obligatoriamente o no en la relación. Puede ser:

- Total: la entidad debe participar obligatoriamente en la relación.
- Parcial: la entidad puede participar opcionalmente en la relación.

Cardinalidad

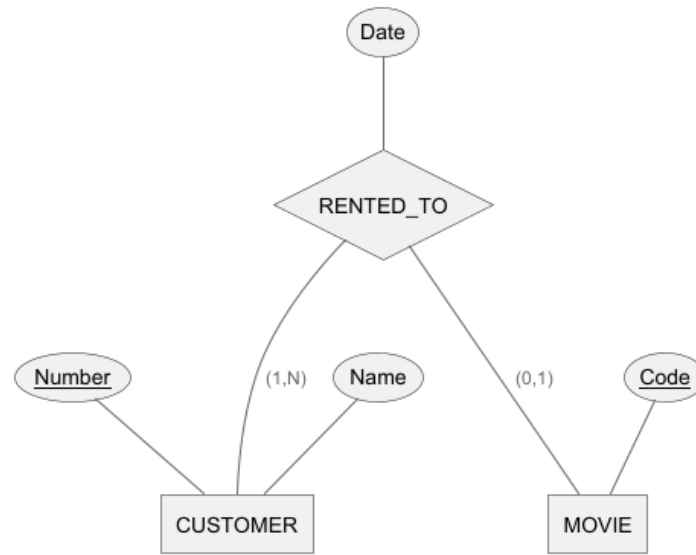
La cardinalidad de una relación indica la cantidad de instancias de una entidad que pueden estar relacionadas con una instancia de otra entidad. Puede ser:

- Uno a Uno (1:1): una instancia de una entidad se relaciona con una única instancia de otra entidad.
- Uno a Muchos (1:N): una instancia de una entidad se relaciona con varias instancias de otra entidad.
- Muchos a Uno (N:1): varias instancias de una entidad se relacionan con una única instancia de otra entidad.

- Muchos a Muchos (N:M): varias instancias de una entidad se relacionan con varias instancias de otra entidad.

Ejemplo de Modelo ER

A continuación, se muestra un ejemplo de un modelo entidad-relación:



er-example.png

Álgebra Relacional

El álgebra relacional es un lenguaje formal que se utiliza para definir operaciones sobre relaciones. Estas operaciones son fundamentales para el manejo y la manipulación de bases de datos relacionales. El álgebra relacional se basa en conjuntos y operaciones de conjuntos, lo que permite realizar consultas y transformaciones sobre los datos de forma eficiente y precisa.

Operaciones Básicas

El álgebra relacional define varias operaciones básicas que se utilizan para manipular relaciones. Algunas de las operaciones más comunes son:

Selección (σ)

La operación de selección (σ) permite filtrar las filas de una relación que cumplen una condición específica. Por ejemplo, la selección de los estudiantes con una calificación mayor a 70 en un examen se puede expresar como:

$$\sigma_{(Calificación > 70)}(Estudiantes)$$

Proyección (π)

La operación de proyección (π) permite seleccionar las columnas de una relación que se desean mostrar. Por ejemplo, la proyección de los nombres y apellidos de los estudiantes se puede expresar como:

$$\pi_{(Nombre, Apellido)}(Estudiantes)$$

Unión (\cup)

La operación de unión (\cup) permite combinar dos relaciones que tienen la misma estructura en una sola relación. Por ejemplo, la unión de las relaciones de estudiantes de dos cursos diferentes se puede expresar como:

$$EstudiantesCurso1 \cup EstudiantesCurso2$$

Diferencia ($-$)

La operación de diferencia ($-$) permite obtener las filas que están en una relación pero no en otra. Por ejemplo, la diferencia entre los estudiantes de un curso y los estudiantes de otro curso se puede expresar como:

$$EstudiantesCurso1 - EstudiantesCurso2$$

Producto Cartesiano (\times)

La operación de producto cartesiano (\times) permite combinar todas las filas de una relación con todas las filas de otra relación. Por ejemplo, el producto cartesiano de los estudiantes y los cursos se puede expresar como:

$$Estudiantes \times Cursos$$

Derivadas

Las operaciones básicas del álgebra relacional se pueden combinar para formar operaciones más complejas. Algunas de las operaciones derivadas más comunes son:

Intersección (\cap)

La operación de intersección (\cap) permite obtener las filas que están en ambas relaciones. Por ejemplo, la intersección entre los estudiantes de un curso y los estudiantes de otro curso se puede expresar como:

$$EstudiantesCurso1 \cap EstudiantesCurso2 = EstudiantesCurso1 - (EstudiantesCurso1 - EstudiantesCurso2)$$

Agrupamiento (Γ)

La operación de agrupamiento (Γ) permite agrupar las filas de una relación en base a un criterio específico y aplicar una función de agregación sobre cada grupo. Por ejemplo, el agrupamiento de los estudiantes por curso y la suma de las calificaciones se puede expresar como:

$$\Gamma_{(Curso), SUM(Calificación)}(Estudiantes)$$

Las funciones de agregación más comunes son:

- SUM: suma
- AVG: promedio
- COUNT: conteo
- MAX: máximo
- MIN: mínimo

Ordenamiento (τ)

La operación de ordenamiento (τ) permite ordenar las filas de una relación con base a uno o más atributos. Por ejemplo, el ordenamiento de los estudiantes por calificación descendente se puede expresar como:

$$\tau_{(Calificación DESC)}(Estudiantes)$$

Join (\bowtie)

La operación de join (\bowtie) permite combinar dos relaciones en base a una condición de igualdad entre los valores de una columna. Por ejemplo, el join de los estudiantes y los cursos en base al código del curso se puede expresar como:

$$Estudiantes \bowtie_{(Estudiantes.Curso=Cursos.Código)} Cursos$$

Left Outer Join (\bowtie_L)

La operación de left outer join (\bowtie_L) permite combinar dos relaciones con base en una condición de igualdad entre los valores de una columna, manteniendo todas las filas de la relación de la izquierda. Por ejemplo, el left outer join de los estudiantes y los cursos se puede expresar como:

$$Estudiantes \bowtie_{(Estudiantes.Curso=Cursos.Código)} Cursos$$

Right Outer Join (\bowtie_R)

La operación de right outer join (\bowtie_R) permite combinar dos relaciones con base en una condición de igualdad entre los valores de una columna, manteniendo todas las filas de la relación de la derecha. Por ejemplo, el right outer join de los estudiantes y los cursos se puede expresar como:

$$Estudiantes \bowtie_{(Estudiantes.Curso=Cursos.Código)} Cursos$$

Conclusiones

El álgebra relacional es una herramienta poderosa para el manejo de bases de datos relacionales. Permite expresar de forma clara y concisa las operaciones que se desean realizar sobre los datos, lo que facilita su manipulación y transformación. Conocer las operaciones básicas y derivadas del álgebra relacional es fundamental para el diseño y la optimización de consultas en bases de datos relacionales.

Ejemplo de aplicación de álgebra relacional

El álgebra relacional es un lenguaje formal para manipular relaciones en una base de datos. Permite realizar operaciones como proyección, selección, unión, intersección, diferencia, producto cartesiano y división, entre otras.

Supongamos que tenemos las siguientes relaciones en una base de datos de una universidad:

Estudiantes

ID	Nombre	Carrera
1	Juan	Informática
2	María	Matemáticas
3	Pedro	Física
4	Ana	Informática
5	Luis	Matemáticas

Cursos

ID	Nombre	Créditos
101	Álgebra Lineal	4
102	Cálculo I	5
103	Programación	6
104	Física I	4

Inscripciones

ID_Estudiante	ID_Curso
1	101
1	103
2	102
3	104
4	101
5	102
5	103

Operaciones

Proyección

La proyección de los nombres de los estudiantes inscritos en algún curso se puede expresar como:

$$\pi_{(Nombre)}(Estudiantes \bowtie Inscripciones)$$

Resultado:

Nombre
Juan
María
Pedro

Selección

La selección de los cursos con más de 4 créditos se puede expresar como:

$$\sigma_{(Créditos > 4)}(Cursos)$$

Resultado:

ID	Nombre	Créditos
102	Cálculo I	5
103	Programación	6

Unión

La unión de los estudiantes de Informática y los estudiantes de Matemáticas se puede expresar como:

$$Estudiantes_{(Carrera='Informática')} \cup Estudiantes_{(Carrera='Matemáticas')}$$

Resultado:

ID	Nombre	Carrera
1	Juan	Informática
4	Ana	Informática
2	María	Matemáticas
5	Luis	Matemáticas

Join

El join de los estudiantes inscritos en algún curso con los cursos se puede expresar como:

$$Estudiantes \bowtie_{(ID=ID_{Estudiante})} Inscripciones \bowtie_{(ID_{Curso}=ID)} Cursos$$

Resultado:

ID	Nombre	Carrera	ID_Estudiante	ID_Curso	Nombre	Créditos
1	Juan	Informática	1	101	Álgebra Lineal	4
1	Juan	Informática	1	103	Programación	6
2	María	Matemáticas	2	102	Cálculo I	5
3	Pedro	Física	3	104	Física I	4
4	Ana	Informática	4	101	Álgebra Lineal	4
5	Luis	Matemáticas	5	102	Cálculo I	5
5	Luis	Matemáticas	5	103	Programación	6

Intersección

La intersección entre los estudiantes de Informática y los estudiantes de Matemáticas se puede expresar como:

$$Estudiantes_{(Carrera='Informática')} \cap Estudiantes_{(Carrera='Matemáticas')}$$

Resultado:

ID	Nombre	Carrera
1	Juan	Informática
4	Ana	Informática
2	María	Matemáticas
5	Luis	Matemáticas

Diferencia

La diferencia entre los estudiantes de Informática y los estudiantes de Matemáticas se puede expresar como:

$$Estudiantes_{(Carrera='Informática')} - Estudiantes_{(Carrera='Matemáticas')}$$

Resultado:

ID	Nombre	Carrera
3	Pedro	Física

Producto Cartesiano

El producto cartesiano entre los estudiantes y los cursos se puede expresar como:

$$Estudiantes \times Cursos$$

Resultado:

ID	Nombre	Carrera	ID	Nombre	Créditos
1	Juan	Informática	101	Álgebra Lineal	4
1	Juan	Informática	102	Cálculo I	5
1	Juan	Informática	103	Programación	6
1	Juan	Informática	104	Física I	4
2	María	Matemáticas	101	Álgebra Lineal	4
2	María	Matemáticas	102	Cálculo I	5
2	María	Matemáticas	103	Programación	6
2	María	Matemáticas	104	Física I	4
3	Pedro	Física	101	Álgebra Lineal	4
3	Pedro	Física	102	Cálculo I	5
3	Pedro	Física	103	Programación	6

Conclusiones

El modelo Relacional de bases de Datos

Introducción

El modelo relacional es un modelo de datos que se utiliza para representar la información en forma de tablas o relaciones. Fue propuesto por Edgar F. Codd en 1970 y se ha convertido en el modelo de datos más utilizado en la actualidad. El modelo relacional se basa en la teoría de conjuntos y álgebra relacional, lo que permite realizar consultas y operaciones sobre los datos de forma eficiente y precisa.

Conceptos Básicos

El modelo relacional se compone de varios elementos básicos que son fundamentales para su comprensión y uso. Algunos de los conceptos más importantes son:

Tablas

Las tablas son la estructura básica del modelo relacional y se utilizan para almacenar los datos en forma de filas y columnas. Cada tabla representa una relación entre entidades y sus atributos, lo que permite organizar la información de forma estructurada y coherente.

Filas y Columnas

Las filas y columnas de una tabla representan los registros y atributos de la relación, respectivamente. Cada fila corresponde a un registro único en la tabla, mientras que cada columna representa un atributo específico de los registros.

Claves

Las claves son atributos o combinaciones de atributos que permiten identificar de forma única cada registro en una tabla. Existen varios tipos de claves, como las claves primarias, claves candidatas y claves foráneas, que se utilizan para garantizar la integridad y consistencia de los datos.

Restricciones

Las restricciones son reglas que se aplican a los datos de una tabla para garantizar su validez y coherencia. Algunas de las restricciones más comunes son las restricciones de integridad, que aseguran que los datos cumplan ciertas reglas o condiciones.

Relaciones

Las relaciones son las conexiones lógicas entre las tablas de una base de datos relacional. Estas relaciones se establecen a través de claves primarias y claves foráneas, que permiten vincular los registros de diferentes tablas y realizar consultas que involucren múltiples tablas.

Atributos de las columnas

Introducción

Los atributos de las columnas son las propiedades que definen el tipo de datos y las restricciones que se aplican a los valores de una columna en una tabla de una base de datos relacional. Estos atributos permiten definir las características de los datos que se pueden almacenar en una columna, lo que ayuda a garantizar la integridad y consistencia de los datos en la base de datos.

Primary Key

El atributo `PRIMARY KEY` se utiliza para identificar de forma única cada registro en una tabla. Esto significa que el valor de la clave primaria no se puede repetir en la tabla y que cada registro debe tener un valor único en la columna de la clave primaria. La clave primaria se utiliza para garantizar la integridad y consistencia de los datos en la base de datos.

Por ejemplo, si una tabla `Clientes` tiene una columna `ID` con el atributo `PRIMARY KEY`, significa que cada cliente debe tener un ID único en la tabla, es decir, no se permiten valores duplicados en la columna `ID`.

Auto Increment

El atributo `AUTO_INCREMENT` se utiliza para generar automáticamente valores numéricos secuenciales para una columna en una tabla. Esto significa que cada vez que se inserta un nuevo registro en la tabla, el valor de la columna con el atributo `AUTO_INCREMENT` se incrementa automáticamente en una unidad. Este atributo es útil para generar identificadores únicos para los registros de una tabla.

Por ejemplo, si una tabla `Productos` tiene una columna `ID` con el atributo `AUTO_INCREMENT`, significa que cada producto de la tabla tendrá un ID único y secuencial, es decir, el valor de la columna `ID` se incrementará automáticamente cada vez que se inserte un nuevo producto en la tabla.

Not Null

El atributo `NOT NULL` se utiliza para garantizar que una columna no contenga valores nulos. Esto significa que todos los registros de la tabla deben tener un valor válido en la columna, lo que ayuda a evitar problemas de inconsistencia y errores en la base de datos.

Por ejemplo, si una columna `Nombre` de la tabla `Clientes` tiene el atributo `NOT NULL`, significa que todos los registros de la tabla deben tener un valor válido en la columna `Nombre`, es decir, no se permiten valores nulos en esta columna.

Unique

El atributo `UNIQUE` se utiliza para garantizar que los valores de una columna sean únicos en la tabla. Esto significa que no se permiten valores duplicados en la columna, lo que ayuda a evitar la inserción de datos duplicados y a mantener la integridad de los datos en la base de datos.

Por ejemplo, si una columna `Correo` de la tabla `Usuarios` tiene el atributo `UNIQUE`, significa que todos los valores de la columna `Correo` deben ser únicos en la tabla, es decir, no se permiten valores duplicados en esta columna.

Default

El atributo `DEFAULT` se utiliza para asignar un valor por defecto a una columna si no se especifica un valor al insertar un nuevo registro en la tabla. Esto significa que si no se proporciona un valor para la columna en la instrucción `INSERT`, se utilizará el valor por defecto especificado en el atributo `DEFAULT`.

Por ejemplo, si una columna `FechaCreacion` de la tabla `Usuarios` tiene el atributo `DEFAULT CURRENT_TIMESTAMP`, significa que si no se especifica una fecha de creación al insertar un nuevo usuario en la tabla, se utilizará la fecha y hora actuales como valor por defecto en la columna `FechaCreacion`.

Zero Fill

El atributo `ZEROFILL` se utiliza para rellenar con ceros los valores numéricos de una columna hasta alcanzar la longitud máxima especificada. Esto significa que si un valor numérico es menor que la longitud máxima de la columna, se rellenará con ceros a la izquierda para completar la longitud máxima.

Por ejemplo, si una columna `Codigo` de la tabla `Productos` tiene el atributo `INT(5) ZEROFILL`, significa que si el valor del código es `123`, se mostrará como `00123` en la tabla, es decir, se rellenará con ceros a la izquierda hasta alcanzar la longitud máxima de 5 dígitos.

Unsigned

El atributo `UNSIGNED` se utiliza para especificar que una columna solo puede contener valores positivos o cero. Esto significa que los valores negativos no son válidos para la columna con el atributo `UNSIGNED`, lo que ayuda a garantizar la integridad de los datos en la base de datos.

Por ejemplo, si una columna `Edad` de la tabla `Usuarios` tiene el atributo `TINYINT UNSIGNED`, significa que la columna solo puede contener valores enteros positivos entre 0 y 255, es decir, no se permiten valores negativos en esta columna.

Conclusiones

Los atributos de las columnas son propiedades que definen el comportamiento y las restricciones de los datos en una tabla de una base de datos relacional. Estos atributos son fundamentales para garantizar la integridad y consistencia de los datos en la base de datos, lo que ayuda a prevenir problemas de inconsistencia y errores en la manipulación de los datos. Al utilizar los atributos adecuados en las columnas de una tabla, se puede asegurar que los datos sean válidos, coherentes y fiables, lo que facilita su almacenamiento, consulta y manipulación en la base de datos.

De modelo Entidad-Relación a Modelo Relacional

Introducción

El modelo Entidad-Relación (ER) es un modelo de datos que se utiliza para representar la estructura y las relaciones de una base de datos de forma visual y sencilla. Este modelo se basa en la representación de entidades y sus atributos, así como en las relaciones entre las entidades, lo que permite diseñar y comprender la estructura de la base de datos de manera intuitiva.

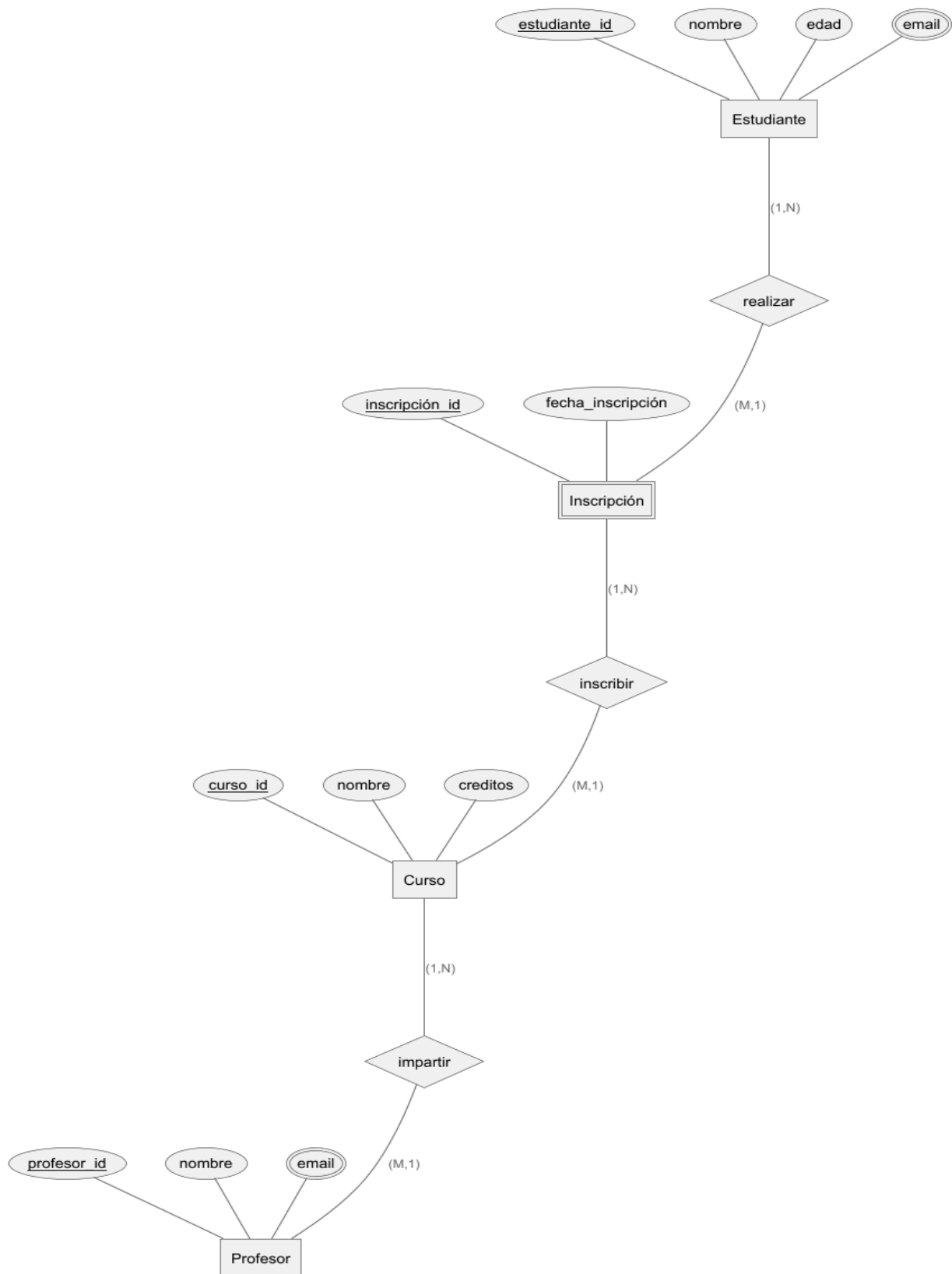
Por otro lado, el modelo Relacional es un modelo de datos que se utiliza para representar la información en forma de tablas o relaciones. Este modelo se basa en la teoría de conjuntos y álgebra relacional, lo que permite realizar consultas y operaciones sobre los datos de forma eficiente y precisa. El modelo relacional se compone de varios elementos básicos, como tablas, filas, columnas, claves y restricciones, que son fundamentales para su comprensión y uso.

En este artículo, exploraremos el proceso de transformación de un modelo Entidad-Relación a un modelo Relacional. Veremos cómo se pueden representar las entidades, atributos y relaciones del modelo ER en tablas del modelo relacional, así como las claves y restricciones que se aplican a los datos. Este proceso es fundamental para la implementación de bases de datos relacionales y permite garantizar la integridad y consistencia de los datos en la base de datos.

Ejemplo de transformación

Para realizar la transformación de un modelo Entidad-Relación a un modelo Relacional, utilizaremos un ejemplo sencillo que consta de tres entidades: **Estudiante**, **Profesor**, **Curso** y **Inscripción**. Cada una de estas entidades tiene sus atributos y relaciones, que representaremos en el modelo ER y luego transformaremos al modelo relacional.

A continuación, presentamos el modelo Entidad-Relación de nuestro ejemplo:



er-to-relational.png

Pasos para la transformación

El proceso de transformación de un modelo Entidad-Relación a un modelo Relacional consta de varios pasos que nos permiten representar las entidades, atributos y relaciones del modelo ER en tablas del modelo relacional. A continuación, describimos los pasos que seguiremos en este proceso:

1. Identificar las entidades fuertes.
2. Crear una tabla para cada entidad fuerte.
3. Identificar los atributos de cada entidad.
4. Crear una columna para cada atributo primario.
5. Crear una columna para cada atributo simple.
6. En caso de atributos compuestos, se deberá decidir si crear el compuesto o descomponerlo.
7. En caso de atributos multivaluados, se deberá crear una tabla adicional.
 - La nueva tabla contendrá una columna para el atributo multivaluado y una columna para la clave primaria de la entidad.
 - Por buenas prácticas, se recomienda crear una clave primaria del tipo `entidad_id` en la nueva tabla. Por ejemplo, si la entidad es `Email`, la clave primaria de la nueva tabla sería `email_id`.
 - Se creará una relación uno a muchos entre la entidad y la nueva tabla.
8. Crear una tabla para cada entidad débil y seguir del 4 al 7.
9. Identificar las relaciones entre las entidades.
10. En el caso de relaciones uno a uno, se deberá decidir si se crea una tabla adicional o si se agrega la clave foránea en una de las tablas. Se recomienda agregar la clave foránea en la tabla que tenga la cardinalidad máxima.
11. En el caso de relaciones uno a muchos, se deberá agregar la clave foránea en la tabla que tenga la cardinalidad máxima.
12. En el caso de relaciones muchos a muchos, se deberá crear una tabla adicional que contenga las claves primarias de ambas entidades. Además, se deberá agregar una clave primaria adicional a la tabla que contenga las claves primarias de ambas entidades.
 - El nombre de la tabla adicional se puede formar con los nombres de las entidades relacionadas, en orden alfabético y separados por un guion bajo. Por ejemplo, si las entidades son `Estudiante` y `Curso`, el nombre de la tabla adicional sería `Curso_Estudiante`.
 - La clave primaria adicional se puede formar con los nombres de las entidades relacionadas, en orden alfabético y separados por un guion bajo. Por ejemplo, si las entidades son `Estudiante` y `Curso`, la clave primaria adicional sería `curso_estudiante_id`.
13. Agregar las claves foráneas en las tablas correspondientes.
 - Para este paso usaremos la representación de pata de gallo (`crow's foot`) para indicar la cardinalidad de la relación.
14. Seleccionar y agregar el tipo de dato adecuado para cada columna.
15. Agregar las restricciones necesarias para garantizar la integridad y consistencia de los datos.

- En estos casos solo contaremos con las restricciones: UNIQUE, NOT NULL, AUTO_INCREMENT y PRIMARY KEY.

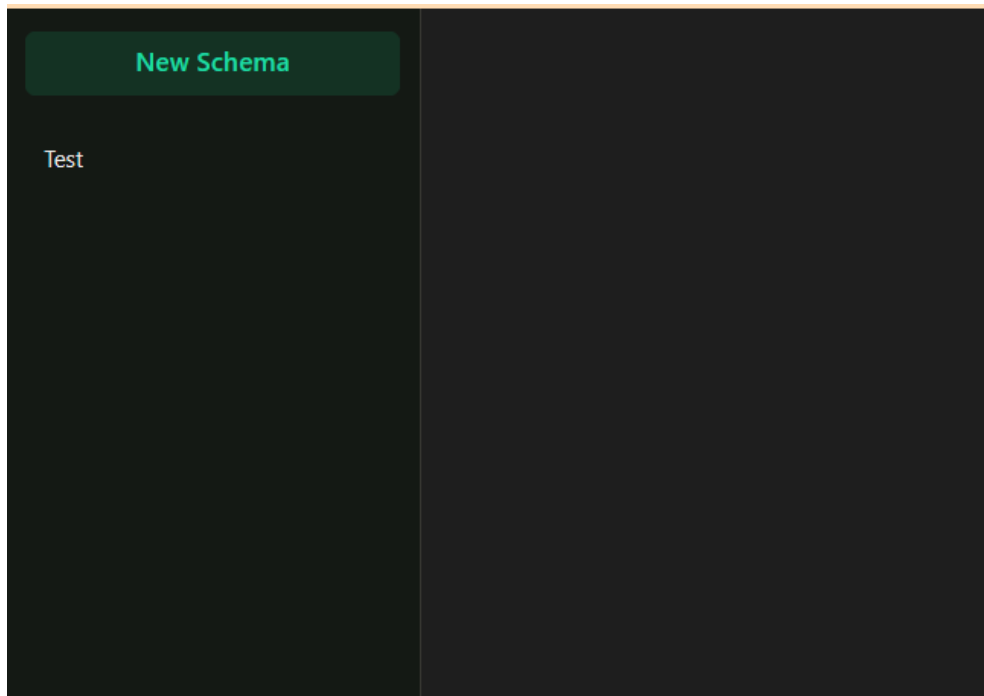
Nuestra herramienta de transformación

Para facilitar el proceso de transformación de un modelo Entidad-Relación a un modelo Relacional, emplearemos la herramienta en línea ERD Editor. Esta herramienta nos permitirá diseñar las entidades, atributos y relaciones del modelo ER, así como exportar el modelo relacional en formato SQL.

Puedes encontrar esta herramienta en el siguiente enlace: ERD Editor (<https://erd-editor.com/>)

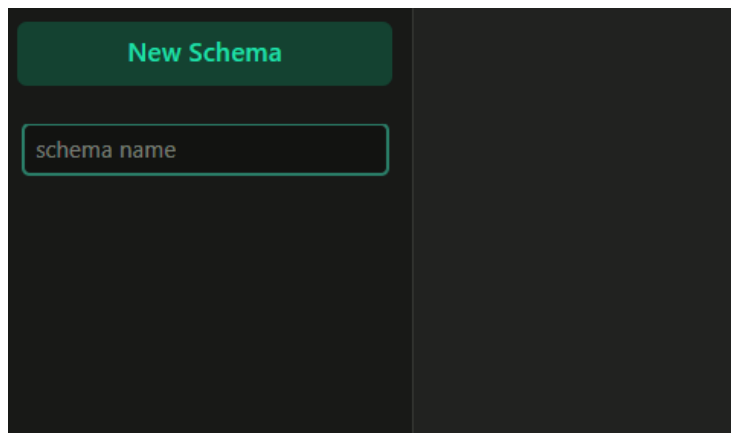
Creando el modelo Entidad-Relación

Para comenzar, crearemos el modelo Entidad-Relación de nuestro ejemplo utilizando la herramienta ERD Editor. Crearemos un nuevo esquema.



erd-schema.png

Recuerda agregar un nombre descriptivo a tu esquema para identificarlo fácilmente.



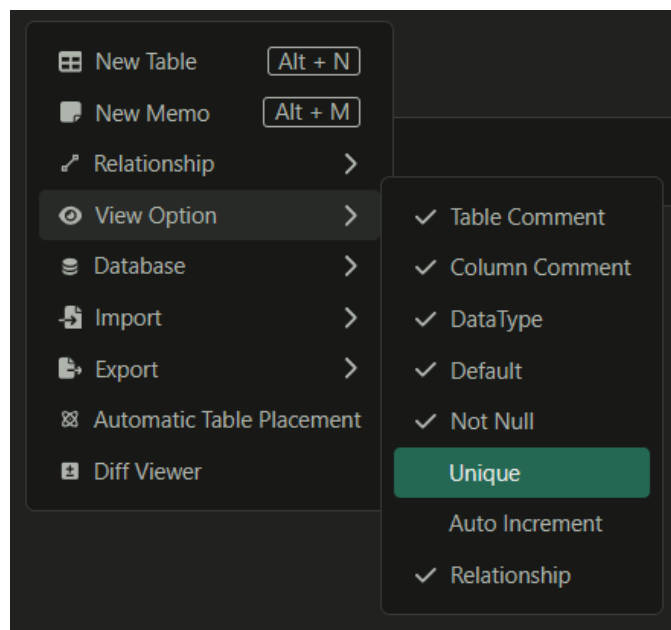
schema-name.png

Creando las tablas del modelo relacional

Para crear las tablas daremos clic derecho en el área de trabajo y seleccionaremos la opción **Add Table**. Aunque podemos presionar la combinación de teclas **Alt + N**.

Nota: En este punto, solo crearemos las tablas correspondientes a las entidades fuertes. Las tablas de las entidades débiles y las tablas adicionales para las relaciones muchos a muchos las crearemos posteriormente.

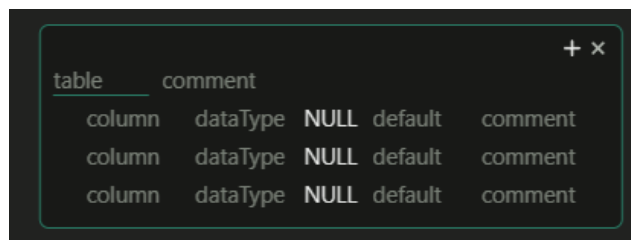
Nota: Deberemos habilitar las opciones de **View > Unique** y **View > Auto Increment** para poder agregar estas restricciones a las columnas. Para esto daremos clic derecho en el área de trabajo y seleccionaremos la opción **View**. Luego, seleccionaremos las opciones **Unique** y **Auto Increment**.



view-unique.png

Creando las columnas de las tablas

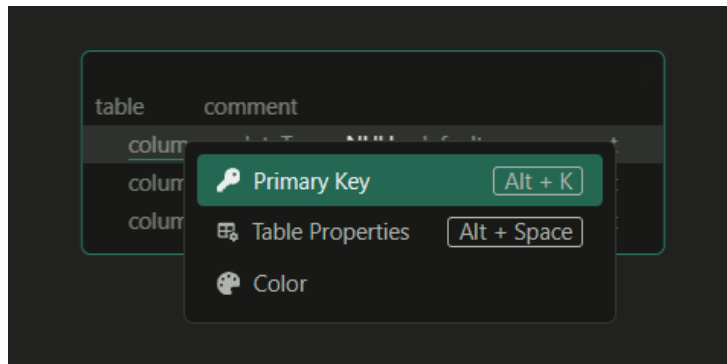
Para agregar las columnas a una tabla, seleccionaremos la tabla y daremos clic en el icono **+**. Otra opción es presionar la combinación de teclas **Alt + Enter**.



add-column.png

Agregando las claves primarias

Para seleccionar una columna como clave primaria, primero seleccionaremos la columna y luego daremos clic derecho sobre ella. Seleccionaremos la opción **Primary Key**. Otra opción es presionar la combinación de teclas **Alt + K**.

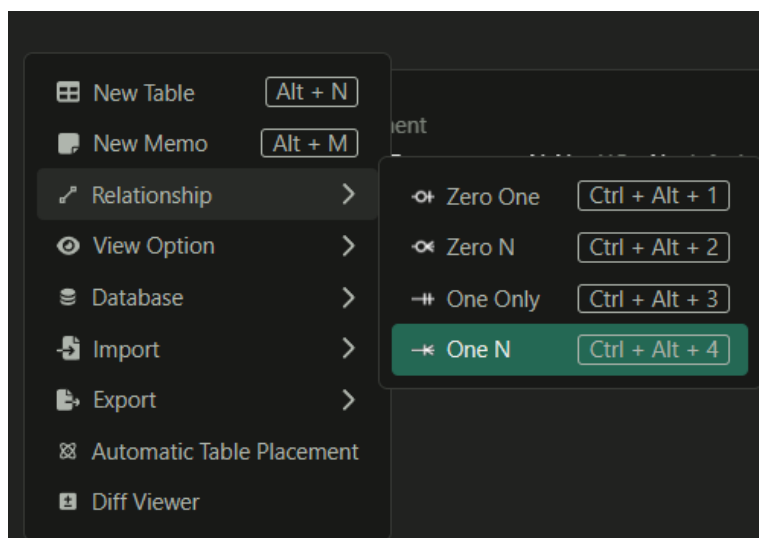


primary-key.png

Agregando las claves foráneas

Para agregar una clave foránea, daremos clic derecho en un espacio vacío del área de trabajo y seleccionaremos la opción **Relationship**. Y seleccionaremos el tipo de relación que deseamos agregar.

Seleccionamos la tabla de origen y la tabla de destino. En este estricto orden.

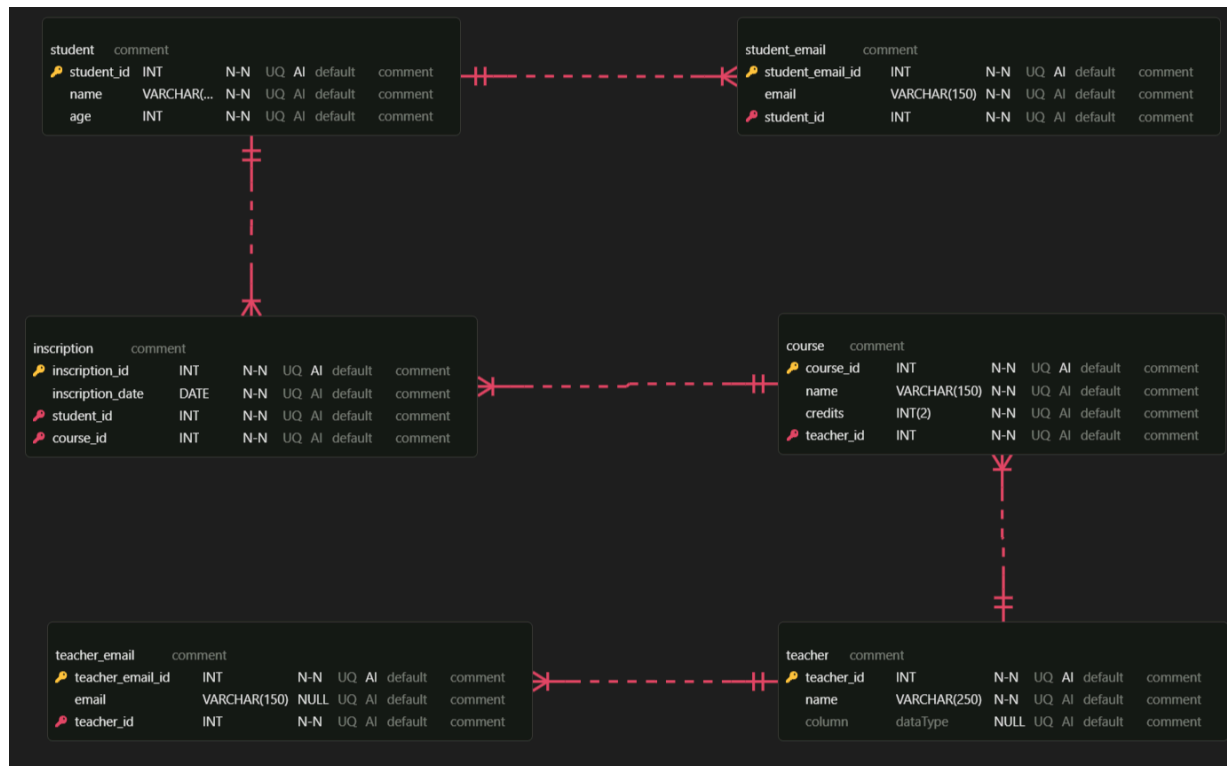


relationship.png

⚠ Nota: Puedes usar los atajos **Ctrl + Alt + N** donde **N** es el número de la relación que deseas agregar. Por ejemplo, **Ctrl + Alt + 1** para una relación con terminal de mínimo 0 y máximo 1.

El resultado final

Una vez que hayamos completado la creación de las tablas y las relaciones, nuestro modelo relacional se verá de la siguiente manera:



relational-model.png

Conclusión

La transformación de un modelo Entidad-Relación a un modelo Relacional es un proceso fundamental en el diseño e implementación de bases de datos relacionales. Este proceso nos permite representar de manera clara y estructurada las entidades, atributos y relaciones del modelo ER en tablas del modelo relacional, lo que facilita la gestión y consulta de los datos en la base de datos.

Al seguir los pasos descritos en este artículo y utilizar herramientas como ERD Editor, podemos realizar la transformación de manera eficiente y precisa, lo que nos permitirá garantizar la integridad y consistencia de los datos en la base de datos. Conocer y aplicar los conceptos y técnicas de diseño de bases de datos relacionales es fundamental para el desarrollo de sistemas de información robustos y eficientes.

Por lo tanto, la transformación de un modelo Entidad-Relación a un modelo Relacional es un proceso clave en el diseño de bases de datos relacionales y nos permite representar de manera clara y estructurada la información de un sistema de información, lo que facilita su implementación y gestión.