Módulo 3: Fundamentos del modelo Entidad Relación

Un modelo conceptual de bases de datos es una representación abstracta y de alto nivel que describe la estructura y organización de los datos de un sistema de información, sin considerar aspectos técnicos de implementación. Este modelo sirve como puente de comunicación entre los usuarios que conocen el negocio y los técnicos que diseñarán la base de datos. Permite validar que se están capturando correctamente todos los datos necesarios antes de pasar a fases más técnicas del diseño.

Introducción

Los modelos conceptuales de bases de datos proporcionan un marco teórico y práctico para organizar, estructurar y manipular información de manera eficiente. Estos modelos definen la arquitectura lógica sobre la cual se construyen los sistemas de gestión de bases de datos, estableciendo las reglas y principios que gobiernan el almacenamiento y recuperación de datos.

El modelo conceptual se enfoca en capturar los **requisitos de información** del negocio o dominio de aplicación. Representa las entidades importantes, sus atributos y las relaciones entre ellas de manera que sea comprensible tanto para usuarios finales como para desarrolladores.

Principales Modelos Conceptuales

Se han planteado diferentes modelos conceptuales para la gestión de bases de datos que podemos resumir así:

a) Modelo Jerárquico

El modelo jerárquico organiza los datos en una estructura de árbol, donde cada registro tiene un único registro padre y puede tener múltiples registros hijos. Esta estructura refleja relaciones uno-a-muchos de manera natural, pero presenta limitaciones significativas para representar relaciones más complejas. La navegación a través de los datos sigue rutas predefinidas desde la raíz hacia las hojas del árbol, lo que puede resultar en redundancia de datos cuando se requieren múltiples rutas de acceso.

b) Modelo de Red

El modelo de red extiende el concepto jerárquico (padre-hijo) permitiendo que un registro tenga múltiples registros padre, formando una estructura de grafo dirigido. Esta flexibilidad adicional permite representar relaciones muchos-a-muchos de manera más directa que el modelo jerárquico. Los datos se organizan mediante conjuntos que establecen relaciones entre tipos de registro, proporcionando múltiples rutas de acceso a la información.

c) Modelo Orientado a Objetos

El modelo orientado a objetos integra conceptos de la programación orientada a objetos en el diseño de bases de datos. Los datos se encapsulan junto con los métodos que operan sobre ellos, formando objetos que pueden heredar propiedades y comportamientos de otros objetos. Este modelo es particularmente útil para

aplicaciones que manejan datos complejos y requieren funcionalidades avanzadas como herencia, polimorfismo y encapsulación.

d) Modelo Documental

El modelo documental almacena información en documentos semi-estructurados, típicamente en formato JSON o XML. Cada documento puede contener campos anidados y arrays, proporcionando flexibilidad en la estructura de datos sin requerir un esquema fijo predefinido. Este modelo es especialmente adecuado para aplicaciones web modernas que manejan datos con estructuras variables.

e) Modelo de Representación Vectorial

El modelo de representación vectorial constituye un paradigma emergente que ha ganado prominencia significativa con el auge de la inteligencia artificial y el aprendizaje automático. Este modelo representa los datos como vectores en espacios multidimensionales de alta dimensionalidad, donde cada dimensión captura características específicas de la información almacenada. Cada elemento de datos se representa como un vector $v \in \mathbb{R}^n$, donde n representa la dimensionalidad del espacio vectorial.

Donde cada componente $v_i \in \mathbb{R}$ representa una característica cuantificada del objeto de datos. La similitud entre elementos se calcula mediante métricas de distancia. el modelo vectorial permite representar mediciones complejas como puntos en espacios multidimensionales. Por ejemplo en el dominio de concetracion de partículas, cada medición se representa como un vector de características:

v_partícula = (concentración_PM2.5, concentración_PM10, concentración_ultrafinas, temperatura, humedad, presión_atmosférica, velocidad_viento, timestamp_normalizado).

También es utilizado para el almacenamiento y consulta eficiente de embeddings de alta dimensionalidad, hoy utilizados para almacenr datos en los modelos generativos (RAG) asociados a la inteligencia artificial.

f) El Modelo Relacional

El modelo relacional, propuesto por Edgar F. Codd en 1970, ha emergido como el paradigma dominante en el diseño de bases de datos debido a su solidez teórica, simplicidad conceptual y potencia expresiva. Este modelo revolucionó el manejo de bases de datos al introducir un enfoque fundamentado en principios matemáticos sólidos, específicamente en la teoría de conjuntos y el álgebra relacional.

La supremacía del modelo relacional se debe a varios factores fundamentales. Su simplicidad conceptual permite que usuarios con diferentes niveles técnicos comprendan y utilicen eficazmente las bases de datos relacionales. La independencia de datos que proporciona permite modificar la estructura física sin afectar las aplicaciones. Además, su base matemática sólida garantiza consistencia y predictibilidad en las operaciones de datos.

Fundamentación del Modelo Relacional

El modelo relacional se basa en conceptos matemáticos rigurosos:

Utiliza la teoría de conjuntos y el álgebra relacional, donde los datos se organizan en relaciones (tablas) que son conjuntos de tuplas (filas). Esta fundamentación matemática proporciona un marco teórico sólido para el manejo de datos. Se sustenta en la normalización para eliminar redundancias, la integridad referencial para

mantener consistencia, y un lenguaje declarativo (SQL) que permite especificar qué se desea obtener sin definir cómo obtenerlo.

Ventajas sobre el Modelo Jerárquico

Flexibilidad estructural: Mientras el modelo jerárquico organiza los datos en estructuras de árbol rígidas donde cada registro hijo tiene un único padre, el modelo relacional permite relaciones muchos a muchos de forma natural mediante tablas de unión.

Independencia de datos: El modelo relacional separa la estructura lógica de los datos de su implementación física, algo que el modelo jerárquico no logra eficientemente. Esto permite modificar la estructura sin afectar las aplicaciones.

Facilidad de consulta: Las consultas SQL son mucho más intuitivas y potentes que navegar manualmente por estructuras jerárquicas. En el modelo jerárquico, acceder a datos requiere conocer la ruta específica en el árbol.

Eliminación de redundancia: La normalización en el modelo relacional reduce significativamente la duplicación de datos que es común en estructuras jerárquicas.

Ventajas sobre el Modelo de Red

Simplicidad conceptual: El modelo de red, aunque más flexible que el jerárquico al permitir múltiples padres por registro, crea estructuras complejas de punteros y enlaces. El modelo relacional simplifica esto mediante relaciones basadas en valores comunes entre tablas.

Independencia de la implementación física: El modelo de red requiere que los programadores manejen explícitamente los punteros y naveguen por la red de registros. El modelo relacional abstrae estos detalles técnicos.

Lenguaje estándar: SQL proporciona un lenguaje estándar, declarativo y de alto nivel, mientras que el modelo de red requiere navegación procedural compleja a través de la estructura de datos.

Mantenimiento y evolución: Modificar la estructura en el modelo de red puede requerir cambios significativos en las aplicaciones debido a las dependencias físicas. El modelo relacional permite cambios más transparentes.

Integridad de datos: El modelo relacional incorpora mecanismos robustos de integridad referencial y restricciones, mientras que en los modelos anteriores esto debía manejarse principalmente a nivel de aplicación.

El modelo relacional no solo resolvió las limitaciones técnicas de sus predecesores, sino que estableció una base teórica sólida que ha permitido décadas de desarrollo y optimización, convirtiéndose en el estándar dominante para sistemas de gestión de bases de datos.

Fundamentos Matemáticos: Teoría de Conjuntos

Conceptos Básicos de Conjuntos: La teoría de conjuntos proporciona los cimientos matemáticos sobre los cuales se construye el modelo relacional. Un conjunto se define como una colección bien definida de objetos distintos, denominados elementos o miembros del conjunto. En el contexto de bases de datos, estos conjuntos representan dominios de valores válidos para los atributos.

Un conjunto A puede expresarse mediante enumeración explícita $A = \{a_1, a_2, a_3, ..., a_n\}$ o mediante una propiedad característica $A = \{x \mid P(x)\}$, donde P(x) es una proposición que define la pertenencia al conjunto.

Operaciones Fundamentales sobre Conjuntos

Las operaciones básicas sobre conjuntos incluyen la unión ($A \cup B$), que produce un conjunto conteniendo todos los elementos presentes en A o en B; la intersección ($A \cap B$), que resulta en elementos comunes a ambos conjuntos; la diferencia (A - B), que contiene elementos de A que no están en B; y el producto cartesiano ($A \times B$), que genera todas las parejas ordenadas posibles entre elementos de A y B.

El producto cartesiano reviste particular importancia en el modelo relacional, ya que establece la base para la construcción de relaciones entre conjuntos de datos.

Dominio: La Unidad Fundamental

Recordemos la definición de "Dominio": Un dominio D representa un conjunto de valores atómicos homogéneos que comparten propiedades semánticas y sintácticas comunes. Cada dominio posee un nombre distintivo y define el conjunto completo de valores válidos que puede tomar un atributo específico dentro del modelo relacional.

Características de los Dominios: Los dominios deben ser atómicos, significando que cada valor es indivisible e indivisible dentro del contexto del modelo. Esta atomicidad garantiza la primera forma normal y evita anomalías en las operaciones de datos.

Ejemplos de Dominios:

Consideremos algunos dominios fundamentales:

DOMINIO_ID_PRODUCTO: Conjunto de identificadores únicos para productos

DOMINIO_ID_PRODUCTO = {1, 2, 3, 4, 5, ...}

DOMINIO_NOMBRE_PRODUCTO: Conjunto de cadenas válidas para nombres de productos

DOMINIO_NOMBRE_PRODUCTO = {"Laptop", "Mouse", "Teclado", "Monitor", ...}

DOMINIO_PRECIO: Conjunto de números reales no negativos para precios

DOMINIO_PRECIO = $\{x \in \mathbb{R} \mid x \geq 0\}$

DOMINIO_FECHA: Conjunto de fechas válidas

DOMINIO_FECHA = {fechas válidas en formato YYYY-MM-DD}

Relación: La Estructura Fundamental

Una relación R sobre los dominios $D_1, D_2, ..., D_n$ se define como un subconjunto del producto cartesiano de dichos dominios: $R \subseteq D_1 \times D_2 \times D_3 \times ... \times D_n$

Cada elemento de una relación se denomina tupla, y representa una combinación específica de valores de los dominios participantes. El grado de una relación corresponde al número de dominios involucrados, mientras que la cardinalidad representa el número de tuplas presentes en la relación en un momento dado.

Propiedades de las Relaciones

Las relaciones en el modelo relacional poseen propiedades fundamentales que las distinguen de estructuras de datos convencionales. No existe orden inherente entre las tuplas, lo que significa que la posición de una tupla dentro de la relación carece de significado semántico. Cada tupla debe ser única, eliminando duplicados automáticamente. Los atributos dentro de una tupla tampoco poseen orden intrínseco, aunque en la implementación práctica se establece un orden para facilitar la manipulación.

Aplicación Práctica: Entidades Producto y Venta

Modelado de la Entidad Producto

La entidad PRODUCTO se modela como una relación sobre cuatro dominios específicos:

Relación PRODUCTO:

$$PRODUCTO \subseteq DOMINIO\_ID\_PRODUCTO \times DOMINIO\_NOMBRE\_PRODUCTO \times DOMINIO\_PRECIO \times DOMINIO\_CATEGORIA$$

Donde: $DOMINIO\_CATEGORIA = \{"Electrónicos", "Oficina", "Hogar", "Deportes", ...\}$

Esquema de la relación:

PRODUCTO(id_producto, nombre_producto, precio, categoria)

Instancias de la relación (tuplas):

(1, "Laptop Dell XPS", 1200.00, "Electrónicos")

(2, "Mouse Inalámbrico", 25.50, "Oficina")

(3, "Silla Ergonómica", 350.00, "Oficina")

(4, "Smartphone", 800.00, "Electrónicos")

Modelado de la Entidad Venta

La entidad VENTA incorpora dominios adicionales y establece referencias hacia otras entidades:

Dominios adicionales:

$DOMINIO\_ID\_VENTA = \{1, 2, 3, 4, 5, ...\}$

$DOMINIO\_CANTIDAD = \{x \in \mathbb{N} \mid x > 0\}$

$DOMINIO\_TOTAL = \{x \in \mathbb{R} \mid x > 0\}$

Relación VENTA:

VENTA ⊆ DOMINIO_ID_VENTA × DOMINIO_ID_PRODUCTO × DOMINIO_CANTIDAD × DOMINIO_FECHA × DOMINIO_TOTAL

Esquema de la relación:

VENTA(id_venta, id_producto, cantidad, fecha_venta, total)

Instancias de la relación:

(1001, 1, 2, "2024-01-15", 2400.00)

(1002, 2, 5, "2024-01-16", 127.50)

(1003, 3, 1, "2024-01-16", 350.00)

(1004, 1, 1, "2024-01-17", 1200.00)

Componentes fundamentales del modelo Entidad Relacion

Relación: Representan objetos o conceptos del mundo real que son relevantes para el sistema (como Cliente, Producto, Pedido). Una relación es representada como Entidad o tabla.

Definición Formal: Una relación es un conjunto de tuplas que comparten los mismos atributos. Matemáticamente, es un subconjunto del producto cartesiano de los dominios de sus atributos. En términos prácticos, una relación se implementa como una tabla en un sistema de gestión de base de datos relacional.

Características Esenciales: Cada relación tiene un nombre único en el esquema. Cada atributo tiene un nombre

único dentro de la relación. Cada celda contiene exactamente un valor atómico

Tuplas (Filas) Cada tupla representa una instancia o entidad específica dentro de la relación. Las tuplas deben ser únicas, lo que significa que no puede haber dos filas idénticas en una relación. En una tabla de "PRODUCTO", cada fila correspondería a un producto individual con todos sus datos asociados. La unicidad se garantiza mediante la clave primaria. No hay orden predefinido entre las tuplas Cada tupla debe ser identificable de forma única.

Grado y Cardinalidad de una entidad: El grado de una relación es el número de atributos (columnas) que contiene. La cardinalidad es el número de tuplas (filas) en la relación en un momento dado.

Estos valores pueden cambiar con el tiempo a medida que la base de datos evoluciona

Atributos: Son las propiedades o características que describen a cada entidad (nombre, edad, precio, fecha). Los atributos definen las propiedades o características de la entidad representada por la relación. Cada atributo tiene un dominio asociado que determina los valores permitidos. Cada atributo tiene un tipo de dato específico.

Relaciones entre entidades: Muestran cómo se conectan o asocian las entidades entre sí (un Cliente "realiza" Pedidos, un Pedido "contiene" Productos).

Cardinalidades: Especifican cuántas instancias de una entidad pueden relacionarse con instancias de otra (uno a uno, uno a muchos, muchos a muchos).

Relaciones entre Entidades desde la Perspectiva de Conjuntos

Claves y Identificación Única

Si bien más adelante ampliaremos los detalles de las claves, debemos ahora considerar que la identificación única de tuplas se logra mediante claves, que son subconjuntos mínimos de atributos cuyos valores determinan unívocamente una tupla dentro de la relación.

Matemáticamente, para una relación R con esquema (A_1, A_2, ..., A_n), un conjunto de atributos K = {A_{i1}, A_{i2}, ..., A_{ik}} constituye una clave si:

Unicidad: No existen dos tuplas distintas t_1, $t_2 \in R$ tales que $t_1[K] = t_2[K]$

Minimalidad: No existe subconjunto propio $K' \subset K$ que satisfaga la propiedad de unicidad

Integridad Referencial mediante Claves Foráneas

Las claves foráneas establecen conexiones lógicas entre relaciones, implementando restricciones de integridad referencial. Una clave foránea FK en la relación R_1 que referencia la clave primaria PK de la relación R_2 debe satisfacer:

$\forall t_1 \in R_1: (t_1[FK] \neq NULL) \rightarrow (\exists t_2 \in R_2: t_1[FK] = t_2[PK])$

En nuestro ejemplo, id_producto en la relación VENTA actúa como clave foránea que referencia la clave primaria id_producto de la relación PRODUCTO.

Interpretación Conjuntista de las Relaciones

Desde la perspectiva de teoría de conjuntos, cada relación representa un subconjunto específico del espacio total de combinaciones posibles definido por el producto cartesiano de sus dominios. Este subconjunto contiene únicamente las combinaciones válidas y semánticamente significativas según las reglas del dominio de aplicación.

Las operaciones de consulta pueden interpretarse como operaciones conjuntistas. La selección corresponde a la definición de un subconjunto mediante una propiedad característica. La proyección equivale a la restricción del producto cartesiano a un subconjunto de dominios. El join representa la intersección de relaciones basada en condiciones específicas sobre sus atributos.

Unión, Intersección y Diferencia

Estas operaciones se aplican directamente siguiendo las definiciones conjuntistas clásicas, requiriendo que las entidades (relaciones o tablas) usadas en cada operación, sean compatibles en términos de esquema.

Conclusiones sobre el modelo relacional

El modelo relacional trasciende ser meramente un paradigma tecnológico para constituirse en un framework matemático riguroso basado en la teoría de conjuntos. Esta fundamentación teórica sólida proporciona garantías formales sobre la consistencia, completitud y correctitud de las operaciones de datos.

La representación de entidades como relaciones sobre dominios específicos, y la implementación de conexiones mediante claves foráneas basadas en principios conjuntistas, establecen un sistema coherente y predecible para la gestión de información. Esta elegancia matemática, combinada con su practicidad operacional, explica la pervivencia y dominancia del modelo relacional en el panorama actual de las bases de datos.

Los conceptos de dominio y relación, derivados directamente de la teoría de conjuntos, no solo proporcionan precisión conceptual sino que también facilitan el diseño, implementación y mantenimiento de sistemas de bases de datos robustos y escalables. Esta síntesis entre rigor matemático y utilidad práctica representa una de las contribuciones más significativas de las ciencias de la computación al manejo sistemático de la información.

Del Modelo Entidad-Relación al Modelo Relacional

El proceso para aplicar estas ideas en una base de datos relacional particular requiere ciertos pasos que detallamos a continuación:

1. Transformación de Entidades a Tablas

Concepto Fundamental: Cada entidad del modelo E-R se convierte directamente en una tabla (relación) en el modelo relacional. Esta transformación mantiene la semántica y estructura de la entidad original.

Proceso de Conversión

Entidad → Tabla

Nombre: El nombre de la entidad se convierte en el nombre de la tabla

Atributos: Cada atributo de la entidad se transforma en una columna de la tabla

Dominio: Se conservan los tipos de datos y restricciones de dominio, al final el documento anexamos un detalle de los tipos específicos, pero aquí mencionaos algunos como DATE, VARCHAR(n) , INTEGER.

Ejemplo Práctico

Entidad: CLIENTE

Atributos:

- id_cliente (identificador)
- nombre (cadena, 50 caracteres)
- email (cadena, 100 caracteres)
- fecha_registro (fecha)

Tabla resultante: CLIENTE

Columnas:

- id_cliente INTEGER
- nombre VARCHAR(50)
- email VARCHAR(100)
- fecha_registro DATE

Consideraciones Especiales

Atributos compuestos: Se descomponen en sus componentes básicos

Atributos multivalorados: Requieren tabla separada

Atributos derivados: Generalmente no se almacenan físicamente

2. Identificadores a Claves Primarias

Función de las Claves Primarias: Los atributos identificadores de cada entidad se convierten en la clave primaria de la tabla correspondiente, cumpliendo funciones críticas:

Unicidad: Garantiza que no existan dos filas idénticas

Identificación: Permite referenciar unívocamente cada registro

Integridad: Mantiene la consistencia de los datos

Características de las Claves Primarias

Propiedades obligatorias:

Unicidad: No puede haber valores duplicados

No nulos: No puede contener valores NULL

Inmutabilidad: Una vez asignado, el valor no debe cambiar

Minimalidad: Debe contener el mínimo número de columnas necesarias

Tipos de Claves Primarias

1. Clave Simple: Esta definida en una sola columna , un atributo de valores atomicos puede identifiar a cada tupla o fila de la tabla.

2. Clave Compuesta : Esta definida en mas de una columna , solo es posible identificar a cada tupla a partir de una conjunto de atributos.

3. Implementación de Relaciones 1:N

Las relaciones uno a muchos (1:N) son las más comunes en bases de datos. Se implementan mediante el uso de claves foráneas.

"La clave primaria del lado 'uno' se coloca como clave foránea en la tabla del lado 'muchos'"

Proceso Detallado

1. Identificación de la cardinalidad:

Lado "1" (uno): Entidad padre

Lado "N" (muchos): Entidad hija

2. Implementación:

La tabla del lado "muchos" recibe una nueva columna

Esta columna contiene la clave primaria de la tabla del lado "uno"

Se establece como clave foránea (FOREIGN KEY)

Ventajas de esta Implementación

Eficiencia: No requiere tablas adicionales

Integridad referencial: Garantiza consistencia automáticamente

Consultas simples: Las operaciones de intersección a travez de JOINs son directos y eficientes

4. Implementación de Relaciones M:N

Desafío de las Relaciones Muchos a Muchos

Las relaciones M:N no pueden implementarse directamente en el modelo relacional debido a que:

Causarían redundancia excesiva

Violarían las formas normales

Complicarían las operaciones de actualización

Solución: Tabla Intermedia (Tabla de Unión)

Estrategia:

Se crea una tabla intermedia (también llamada tabla de unión o tabla puente)

Esta tabla contiene las claves primarias de ambas entidades participantes

La combinación de estas claves forma la clave primaria compuesta de la tabla intermedia

Características de la Tabla Intermedia

1. Clave Primaria Compuesta: Formada por las claves foráneas de ambas entidades, que garantiza que no existan duplicados en la relación

2. Claves Foráneas: Una hacia cada entidad participante, para mantener la integridad referencial

Integridad Referencial: Resumindo el proceso de transformación

Entidades → Tablas (conservando estructura y semántica)

Identificadores → Claves Primarias (garantizando unicidad)

Relaciones 1:N → Claves Foráneas (en la tabla del lado "muchos")

Relaciones M:N → Tablas Intermedias (con clave primaria compuesta)

Este proceso sistemático asegura que el modelo relacional resultante mantenga toda la información y restricciones del modelo conceptual original, mientras aprovecha las ventajas del modelo relacional para el almacenamiento y consulta eficiente de datos.

APÉNDICE: Tipos de Datos en el Modelo Relacional

1. Tipos Alfanuméricos

Los tipos alfanuméricos están diseñados para almacenar texto y caracteres. La elección del tipo correcto impacta significativamente en el rendimiento y el uso de espacio.

CHAR(n) - Cadena de Longitud Fija

Características:

- Longitud fija: Siempre ocupa exactamente n caracteres
- Relleno: Se completa con espacios en blanco si el valor es menor a n
- Rendimiento: Acceso más rápido debido a tamaño predecible
- Uso de espacio: Puede desperdiciar espacio si los valores son variables

Casos de uso ideales:

- Códigos de país (siempre 2 caracteres) codigo_pais CHAR(2) -- 'ES', 'US', 'AR'
- Códigos postales con formato fijo codigo_postal CHAR(5) -- '28001', '10001'

Ventajas:

- Rendimiento optimizado en consultas
- Ideal para datos con longitud consistente
- Facilita la indexación

Desventajas:

- Desperdicio de espacio con datos variables
- Limitaciones en aplicaciones multiidioma

VARCHAR(n) - Cadena de Longitud Variable

Características:

- Longitud variable: Almacena solo los caracteres necesarios (hasta n)
- Eficiencia de espacio: No desperdicia espacio con relleno
- Flexibilidad: Adaptable a diferentes longitudes de contenido
- Sobrecarga mínima: Requiere 1-2 bytes adicionales para almacenar la longitud

Implementación típica:

- Nombres de personas (longitud variable) nombre VARCHAR(100)
- Direcciones de email email VARCHAR(255)
- Descripciones cortas de productos descripcion_corta VARCHAR(500)

Consideraciones de rendimiento:

- Índices: Más eficientes en espacio que CHAR para datos variables
- Consultas: Rendimiento ligeramente menor que CHAR en algunos casos
- Fragmentación: Posible fragmentación con actualizaciones frecuentes

TEXT - Texto Extenso

Características:

Sin límite específico: Puede almacenar textos muy largos (depende del SGBD)
Almacenamiento externo: Generalmente se almacena fuera de la página principal
Flexibilidad máxima: Ideal para contenido de longitud impredecible

Casos de uso:
Artículos de blog o noticias
Descripciones detalladas de productos
Comentarios de usuarios
Documentación técnica

2. Tipos Numéricos

Los tipos numéricos permiten almacenar valores matemáticos con diferentes niveles de precisión y rangos.

INTEGER - Números Enteros

Variantes por tamaño:
TINYINT Rango -128 a 127 (1 byte)
SMALLINT Rango -32,768 a 32,767 (2 bytes)
INTEGER/INT Rango -2,147,483,648 a 2,147,483,647 (4 bytes)
BIGINT Rango -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807 (8 bytes)
Aplicaciones prácticas:

```
id_usuario INTEGER PRIMARY KEY
```

Contadores y cantidades

```
cantidad_stock SMALLINT
numero_paginas SMALLINT
poblacion_ciudad BIGINT
estado_pedido TINYINT  -- 0=pendiente, 1=procesado, 2=enviado
```

Consideraciones:

Rendimiento: Operaciones aritméticas muy rápidas
Espacio: Elegir el tamaño mínimo necesario
Índices: Altamente eficientes para indexación

DECIMAL(p,s) - Números con Decimales

Parámetros:
p (precision): Número total de dígitos
s (scale): Número de dígitos después del punto decimal

Características clave:
Precisión exacta: No hay errores de redondeo

Almacenamiento: Basado en representación decimal
Cálculos financieros: Ideal para dinero y operaciones precisas

Casos de uso:
Precios de productos (máximo 999,999.99) precio DECIMAL(8,2)
Porcentajes con alta precisión (99.9999%) porcentaje DECIMAL(6,4)
Coordenadas geográficas latitud DECIMAL(10,7) -- -90.0000000 a 90.0000000 longitud DECIMAL(10,7) -- -180.0000000 a 180.0000000
Cantidades monetarias grandes salario_anual DECIMAL(12,2) -- Hasta 9,999,999,999.99

FLOAT/REAL - Punto Flotante

Características:
Representación aproximada: Usa notación científica
Rango amplio: Puede representar números muy grandes o muy pequeños
Precisión variable: Puede tener errores de redondeo
Eficiencia: Operaciones matemáticas rápidas

Tipos disponibles:
REAL: Precisión simple (4 bytes)
FLOAT: Precisión doble (8 bytes)
DOUBLE: Equivalente a FLOAT en algunos SGBDs

Casos de uso apropiados:
Mediciones científicas: temperatura FLOAT
distancia_astronomica DOUBLE
Cálculos estadísticos
promedio_calificaciones FLOAT
desviacion_estandar REAL

Advertencias importantes:
No usar para cálculos financieros
Comparaciones exactas pueden fallar
Preferir DECIMAL cuando la precisión es crítica

3. Tipos Temporales

Los tipos temporales manejan fechas, horas y marcas de tiempo con diferentes niveles de precisión.

DATE Solo Fecha

Formato estándar: YYYY-MM-DD

Rango típico: '1000-01-01' a '9999-12-31'

Casos de uso: Información personal, eventos y planificación

fecha_nacimiento DATE
fecha_evento DATE
fecha_vencimiento DATE
fecha_contratacion DATE

TIME: Solo Hora

Formato estándar: HH:MM:SS

Rango: '00:00:00' a '23:59:59'

Casos de uso: Horarios de trabajo, duraciones
hora_inicio TIME
hora_fin TIME
duracion_llamada TIME

TIMESTAMP: Fecha y Hora

Características:
Precisión completa: Fecha y hora combinadas
Zona horaria: Algunos SGBDs incluyen información de zona horaria
Marcas de tiempo: Ideal para auditoría y trazabilidad

Casos de uso: Auditoría y control

fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
fecha_modificacion TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
momento_login TIMESTAMP
momento_logout TIMESTAMP

4. Otros Tipos Especializados

BOOLEAN - Valores Lógicos

Representación: Valores válidos: TRUE, FALSE, NULL

Implementación: Varía según SGBD (algunos usan TINYINT)

Flags y estados binarios
activo BOOLEAN DEFAULT TRUE
es_premium BOOLEAN DEFAULT FALSE
acepta_marketing BOOLEAN

BLOB - Datos Binarios

TINYBLOB: Hasta 255 bytes

BLOB: Hasta 65,535 bytes (64 KB)

MEDIUMBLOB: Hasta 16,777,215 bytes (16 MB)

LONGBLOB: Hasta 4,294,967,295 bytes (4 GB)

Casos de uso: Archivos multimedia, datos serializados

foto_perfil MEDIUMBLOB
documento_pdf LONGBLOB
firma_digital BLOB
configuracion_binaria BLOB
cache_objeto MEDIUMBLOB

Consideración importante: Puede impactar en el rendimiento sobre consultas grandes

Mejores Prácticas

Eficiencia de espacio:

Usar el tipo más pequeño que satisfaga los requisitos
Considerar el crecimiento futuro de los datos
Evaluar el impacto en índices y consultas

Rendimiento:

CHAR para datos de longitud fija frecuentemente consultados
INTEGER para claves primarias y foráneas
Evitar TEXT/BLOB en consultas frecuentes

Mantenimiento:

Documentar la elección de tipos de datos
Considerar la portabilidad entre SGBDs
Planificar migraciones futuras