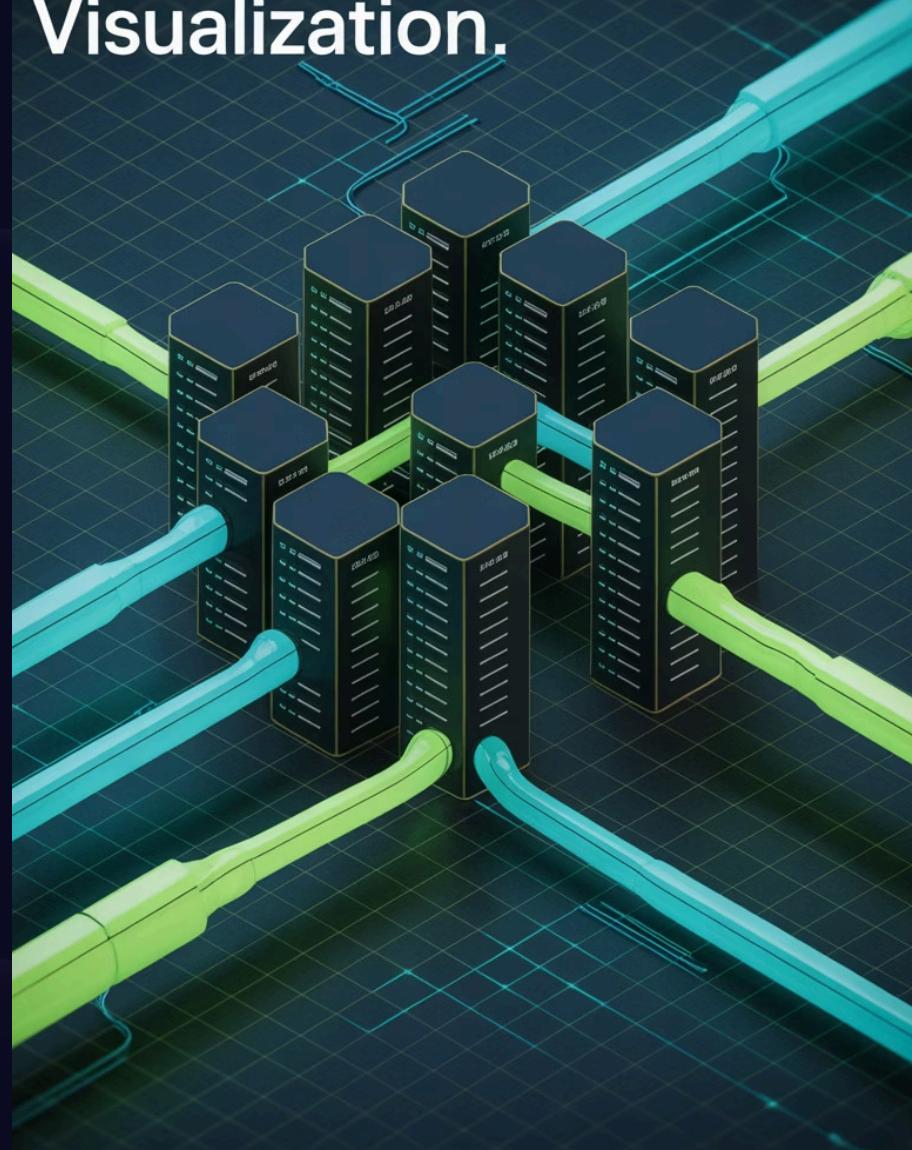


Fundamentos del Modelo Relacional

Bienvenidos a esta presentación sobre el Modelo Relacional, pieza fundamental en el diseño y análisis de bases de datos modernas. A lo largo de este recorrido, exploraremos los componentes esenciales, principios y aplicaciones prácticas que conforman la columna vertebral de los sistemas de gestión de bases de datos relacionales.

Nuestro objetivo es proporcionarte las herramientas necesarias para diseñar estructuras de datos coherentes, normalizadas y con integridad referencial, competencias indispensables para cualquier profesional de las bases de datos en la actualidad.

Database Model Visualization.





Orígenes del Modelo Relacional



1970

Edgar F. Codd publica "A Relational Model of Data for Large Shared Data Banks" mientras trabaja en IBM, estableciendo las bases teóricas del modelo basado en la teoría matemática de conjuntos.



1974-1979

Desarrollo de System R en IBM y el lenguaje SEQUEL (posteriormente SQL), primera implementación del modelo relacional.



1979

Oracle lanza la primera base de datos relacional comercial, marcando el inicio de su adopción generalizada en la industria.



1980-Presente

Evolución y consolidación como el paradigma dominante en sistemas de gestión de bases de datos.

Comparativa con Modelos Previos

Modelo Jerárquico

Estructura de árbol con relaciones padre-hijo. Los datos se organizan en jerarquías rígidas donde cada registro hijo tiene un único parente.

Ventajas: Eficiente para relaciones 1:N y aplicaciones con estructura inherentemente jerárquica.

Desventajas: Dificultad para representar relaciones M:N y redundancia de datos.

Modelo de Red

Estructura de grafos donde los registros se conectan mediante punteros. Un registro hijo puede tener múltiples padres, superando la limitación jerárquica.

Ventajas: Mayor flexibilidad para relaciones complejas y mejor rendimiento en navegación.

Desventajas: Complejidad en implementación y dependencia de la estructura física.

Modelo Relacional

Basado en la teoría de conjuntos y lógica de predicados. Representa datos en tablas (relaciones) con filas y columnas, conectadas mediante valores de atributos.

Ventajas: Independencia física/lógica, simplicidad conceptual, soporta consultas ad-hoc con SQL.



Ventajas del Modelo Relacional



Independencia Lógica y Física

Separa la estructura lógica de los datos de su implementación física, facilitando cambios en el almacenamiento sin afectar a las aplicaciones.



Lenguaje Declarativo

Utiliza SQL, un lenguaje no procedimental que especifica qué datos recuperar en lugar de cómo recuperarlos, simplificando enormemente el acceso a datos.



Integridad de Datos

Proporciona mecanismos robustos para garantizar la consistencia mediante restricciones, claves y reglas de integridad referencial.



Normalización

Ofrece técnicas formales para reducir la redundancia y dependencias anómalas, mejorando la eficiencia y mantenibilidad.

Estructuras Fundamentales: La Relación

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
- No existen tuplas (filas) duplicadas

Grado y Cardinalidad

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.

Tuplas y Atributos

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 "Estudiantes", cada fila correspondería a un estudiante individual con todos sus datos asociados. La unicidad se garantiza mediante la clave primaria.

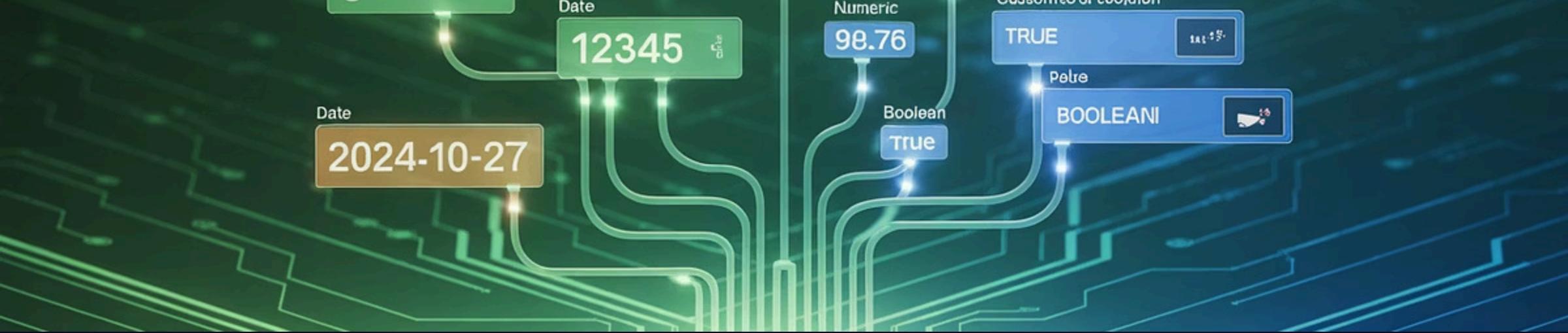
- No hay orden predefinido entre las tuplas
- Cada tupla debe ser identificable de forma única

Atributos (Columnas)

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.

En la tabla "Estudiantes", los atributos podrían ser: ID, Nombre, Apellido, Fecha_Nacimiento, Email, etc. El dominio de "Fecha_Nacimiento" sería el conjunto de fechas válidas.

- Orden lógico según la definición de la tabla
- Cada atributo tiene un tipo de dato específico



Tipos de Datos en el Modelo Relacional

T

Tipos Alfanuméricicos

- CHAR(n): Cadena de longitud fija
- VARCHAR(n): Cadena de longitud variable
- TEXT: Para texto extenso



Tipos Numéricos

- INTEGER: Números enteros
- DECIMAL(p,s): Números con decimales
- FLOAT/REAL: Punto flotante



Tipos Temporales

- DATE: Solo fecha
- TIME: Solo hora
- TIMESTAMP: Fecha y hora



Otros Tipos

- BOOLEAN: Valores lógicos
- BLOB: Datos binarios
- JSON: Documentos JSON



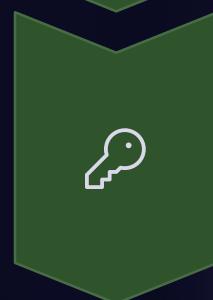
Del Modelo Lógico al Relacional



Entidades a Tablas



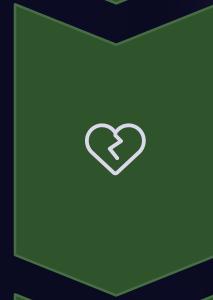
Cada entidad del modelo E-R se convierte en una tabla. Los atributos de la entidad se transforman en columnas de la tabla, manteniendo los mismos dominios.



Identificadores a Claves Primarias



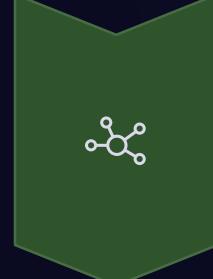
Los atributos identificadores de cada entidad se convierten en la clave primaria de la tabla correspondiente, garantizando la unicidad de cada fila.



Relaciones 1:N



Se implementan añadiendo la clave primaria de la entidad del lado "uno" como clave foránea en la tabla del lado "muchos", estableciendo así la relación.



Relaciones M:N



Requieren una tabla intermedia que contiene las claves primarias de ambas entidades participantes, formando una clave primaria compuesta.

Claves Primarias: Fundamentos

Definición

Una clave primaria es un atributo o conjunto de atributos que identifica de manera única cada fila en una tabla. Actúa como el identificador principal para las entidades representadas en la relación.

Establece la base para las referencias entre tablas y garantiza que podamos acceder a cada registro de forma inequívoca.

Requisitos Fundamentales

- Unicidad: No puede haber dos filas con el mismo valor de clave primaria
- No nulidad: Ningún componente de la clave primaria puede ser NULL
- Minimalidad: No debe contener atributos superfluos para la identificación
- Estabilidad: Sus valores deben cambiar raramente o nunca

Implementación

En SQL, se define mediante la cláusula PRIMARY KEY durante la creación de la tabla o con una restricción ALTER TABLE posteriormente.

La mayoría de los SGBD crean automáticamente un índice para la clave primaria, mejorando el rendimiento de las búsquedas y garantizando la unicidad.

Ejemplo de Clave Primaria Simple

estudiante_id	nombre	apellido	email	fecha_nacimiento
1001	Ana	García	ana.garcia@email.com	1999-05-15
1002	Carlos	Martínez	carlos.m@email.com	2000-02-23
1003	Elena	López	elena.lopez@email.com	1998-11-30

En este ejemplo de la tabla "Estudiantes", el campo "estudiante_id" es la clave primaria (PK). Su implementación en SQL sería:

```
CREATE TABLE Estudiantes (
    estudiante_id INT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    fecha_nacimiento DATE
);
```

O alternativamente, utilizando la sintaxis de restricción:

```
CREATE TABLE Estudiantes (
    estudiante_id INT,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    fecha_nacimiento DATE,
    CONSTRAINT pk_estudiante PRIMARY KEY (estudiante_id)
);
```

Ejemplo de Clave Primaria Compuesta

Identificación

Una clave primaria compuesta utiliza múltiples atributos en combinación para identificar de forma única cada fila

Consideraciones

Aumenta la complejidad en las referencias y puede impactar el rendimiento en operaciones JOIN



Implementación

Se define usando la cláusula PRIMARY KEY con múltiples campos en la creación de la tabla

Uso común

Habitual en tablas de intersección que representan relaciones muchos a muchos

```
CREATE TABLE Matriculas (
    curso_id INT,
    estudiante_id INT,
    fecha_matricula DATE NOT NULL,
    calificacion DECIMAL(4,2),
    CONSTRAINT pk_matricula PRIMARY KEY (curso_id, estudiante_id),
    CONSTRAINT fk_curso FOREIGN KEY (curso_id) REFERENCES Cursos(curso_id),
    CONSTRAINT fk_estudiante FOREIGN KEY (estudiante_id) REFERENCES Estudiantes(estudiante_id)
);
```

Claves Foráneas: Conectando Tablas



Definición

Atributo o conjunto de atributos en una tabla que hace referencia a la clave primaria de otra tabla



Propósito

Establece y garantiza relaciones entre datos en diferentes tablas



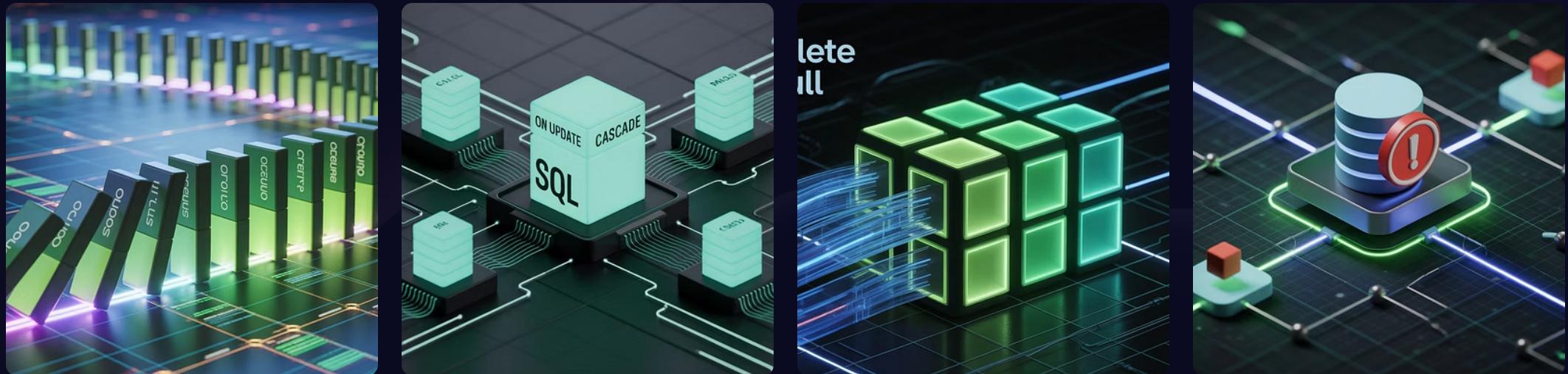
Integridad Referencial

Asegura que las referencias entre tablas sean válidas y consistentes

Las claves foráneas (FK) son el mecanismo fundamental para implementar relaciones en el modelo relacional. A diferencia de las claves primarias, las claves foráneas pueden contener valores NULL (a menos que se especifique lo contrario) y pueden repetirse en múltiples filas.

Cuando se definen claves foráneas, el SGBD ejecuta verificaciones automáticas para cada operación que podría violar la integridad referencial, como intentar eliminar un registro referenciado por otra tabla o insertar un valor que no existe en la tabla principal.

Acciones Referenciales en Claves Foráneas



Las acciones referenciales definen el comportamiento del sistema cuando se actualiza o elimina un registro referenciado por una clave foránea. Se especifican mediante las cláusulas ON DELETE y ON UPDATE al definir la restricción de clave foránea.

```
CREATE TABLE Pedidos (
    pedido_id INT PRIMARY KEY,
    cliente_id INT,
    fecha_pedido DATE NOT NULL,
    total DECIMAL(10,2) NOT NULL,
    CONSTRAINT fk_cliente FOREIGN KEY (cliente_id)
        REFERENCES Clientes(cliente_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

Tipos de Acciones Referenciales



CASCADE

Propaga los cambios. Si se elimina un registro padre, se eliminan automáticamente todos los registros hijos relacionados. Si se actualiza la clave primaria, se actualizan todas las referencias.

Útil cuando la existencia del registro hijo depende completamente del registro padre.



SET NULL

Establece a NULL el valor de la clave foránea cuando se elimina o actualiza el registro referenciado. Requiere que la columna de clave foránea acepte valores NULL.

Adecuado cuando la relación es opcional pero deseas mantener el registro hijo.



SET DEFAULT

Establece el valor predeterminado para la columna cuando se elimina o actualiza el registro referenciado. La columna debe tener definido un valor por defecto.

Menos común, pero útil en algunos escenarios específicos.



RESTRICT / NO ACTION

Impide la eliminación o actualización del registro padre si existen registros hijos relacionados.

Es el comportamiento predeterminado en muchos SGBD.

Garantiza que no se pierdan inadvertidamente relaciones importantes.

Claves Candidatas



Una clave candidata es cualquier atributo o conjunto mínimo de atributos que puede identificar de forma única cada fila de una tabla. Todas las claves candidatas cumplen los requisitos de unicidad y no nulidad. Entre ellas, seleccionamos una como clave primaria, mientras que las demás se denominan claves alternativas.

Por ejemplo, en una tabla de empleados, tanto el "número de empleado" como el "número de la seguridad social" podrían ser claves candidatas. Si elegimos el "número de empleado" como clave primaria, el "número de la seguridad social" sería una clave alternativa que seguiría requiriendo unicidad.

Claves Sustitutas vs. Claves Naturales

Claves Naturales

Son atributos inherentes a la entidad que naturalmente identifican cada fila de forma única. Ejemplos: DNI, ISBN, código de producto del fabricante.

Ventajas:

- Tienen significado para los usuarios
- No requieren columnas adicionales
- Garantizan integridad a nivel de dominio

Desventajas:

- Pueden ser complejas o compuestas
- Podrían cambiar con el tiempo
- A veces son largas o complicadas

Claves Sustitutas

Son identificadores artificiales generados por el sistema, sin significado inherente para el usuario. Ejemplos: ID autoincremental, UUID, secuencias.

Ventajas:

- Simples y consistentes entre tablas
- Nunca cambian
- Optimizadas para indexación
- No exponen información sensible

Desventajas:

- Sin significado para el usuario
- Requieren columna adicional
- Pueden ocultar problemas de calidad de datos

Integridad de Dominio

El primer nivel de integridad de datos en el modelo relacional

¿Qué es la Integridad de Dominio?

La integridad de dominio garantiza que todos los valores en una columna pertenezcan al conjunto de valores válidos definidos para esa columna. Es el nivel más básico de integridad de datos.

Se implementa mediante la definición de tipos de datos, restricciones CHECK, valores predeterminados, reglas de validación y restricciones NOT NULL.

Mecanismos para Implementar la Integridad de Dominio

- Tipos de datos apropiados (INT, VARCHAR, DATE, etc.)
- Restricciones CHECK para reglas personalizadas
- Valores DEFAULT para inicialización
- Restricciones NOT NULL para campos obligatorios
- Enumeraciones y listas de valores permitidos

Implementación en SQL: Ejemplo Práctico

La siguiente definición garantiza la integridad de dominio para una columna de calificación:

```
CREATE TABLE Evaluaciones (
    evaluacion_id INT PRIMARY KEY,
    estudiante_id INT NOT NULL,
    calificacion DECIMAL(4,2) CHECK (calificacion >= 0 AND calificacion <= 10),
    estado VARCHAR(15) CHECK (estado IN ('Pendiente', 'Completada', 'Anulada')),
    fecha_evaluacion DATE DEFAULT CURRENT_DATE,
    FOREIGN KEY (estudiante_id) REFERENCES Estudiantes(estudiante_id)
);
```

Integridad de Entidad

1

0

100%

Regla Fundamental

Cada tabla debe tener una clave primaria que identifique de forma única cada fila

Valores NULL

Ningún componente de la clave primaria puede ser NULL en ninguna fila

Cobertura

Todas las entidades en la base de datos deben cumplir esta regla sin excepciones

La integridad de entidad es un principio fundamental del modelo relacional que establece que cada entidad (fila) debe ser única e identifiable. Sin esta garantía, no podríamos distinguir de manera confiable entre diferentes entidades ni establecer relaciones precisas entre tablas.

Los sistemas de gestión de bases de datos relacionales imponen automáticamente esta regla una vez que se define la clave primaria. Cualquier intento de insertar filas con claves primarias duplicadas o NULL será rechazado, manteniendo así la integridad de entidad en todo momento.

Integridad Referencial

Principio Básico

Si una tabla contiene una clave foránea, cualquier valor en esa columna debe existir en la tabla referenciada o ser NULL (si se permite).

Operaciones de Eliminación

Las eliminaciones en la tabla padre están sujetas a las reglas de acción referencial (CASCADE, SET NULL, RESTRICT, etc.).



La integridad referencial es esencial para mantener la coherencia de los datos relacionados entre tablas. Garantiza que no existan "referencias huérfanas" - registros que apuntan a entidades que no existen - manteniendo así la base de datos en un estado consistente en todo momento.

Operaciones de Inserción

No se puede insertar un valor de clave foránea si no existe el valor correspondiente en la tabla padre.

Operaciones de Actualización

No se puede actualizar una clave foránea a un valor que no exista en la tabla padre.

Conclusiones

El modelo relacional proporciona una base sólida para el diseño e implementación de bases de datos modernas.



Fundamentos Claros

Las estructuras y reglas del modelo relacional garantizan la integridad y coherencia de los datos.



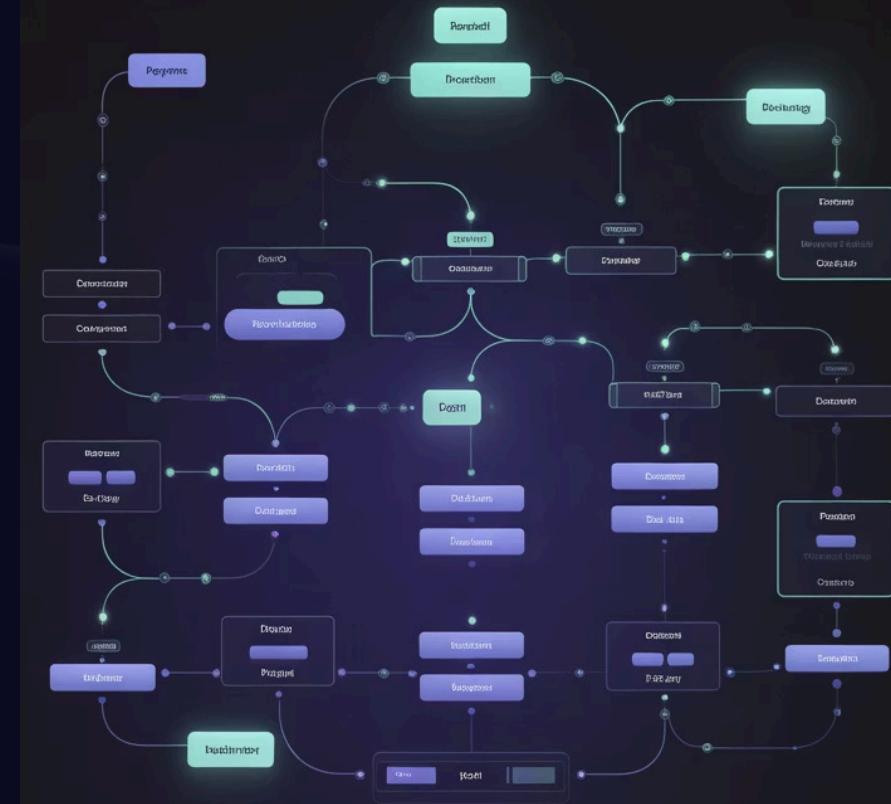
Aplicación Práctica

Los conceptos teóricos se traducen directamente en implementaciones SQL funcionales.



Integridad de Datos

Las restricciones de integridad protegen contra inconsistencias y errores en los datos almacenados.



Explore our
data architecture

[View schema details](#)

¡Muchas Gracias!

Esperamos que estos fundamentos del modelo relacional os hayan resultado útiles para vuestros proyectos de bases de datos.

