
BASE DE DATOS
DigitalGeko

BASE DE DATOS RELACIONALES

UNA **BASE DE DATOS** ES UN CONJUNTO ORGANIZADO DE DATOS QUE SE ALMACENAN Y SE PUEDEN ACCEDER ELECTRÓNICAMENTE. SU OBJETIVO PRINCIPAL ES **ALMACENAR, ORGANIZAR Y RECUPERAR INFORMACIÓN** DE FORMA EFICIENTE Y SEGURA.

Concepto	Descripción
Dato	Unidad mínima de información. Ej: "Juan", "35", "Activo".
Campo / Atributo	Columna en una tabla, representa un tipo de dato. Ej: Nombre, Edad.
Registro / Fila	Conjunto de datos que pertenecen a una entidad. Ej: un paciente.
Tabla	Conjunto de registros con la misma estructura.
Clave primaria (PK)	Campo único que identifica cada fila. Ej: ID_Paciente.
Clave foránea (FK)	Campo que se relaciona con otra tabla. Ej: ID_Médico.
Índice	Estructura que acelera búsquedas.
Consulta (Query)	Instrucción SQL para leer o modificar datos.
Transacción	Conjunto de operaciones que se ejecutan como una unidad atómica.
Normalización	Proceso para reducir redundancia y mejorar integridad.

Por ejemplo: una base de datos para un hospital puede contener tablas con pacientes, médicos, citas, tratamientos y facturación.

✅ MEJORES PRÁCTICAS EN BASES DE DATOS

📐 Diseño y estructura

- Diseña bien tu modelo de datos: usa diagramas ER o UML.
- Aplica normalización: reduce datos repetidos y evita inconsistencias.
- Define claves primarias y foráneas: garantiza integridad referencial.
- Evita nombres ambiguos o genéricos: usa nombres descriptivos y consistentes.

⚙️ Optimización y rendimiento

- Crea índices para columnas que se consultan frecuentemente.
- Evita SELECT *: pide solo los campos que realmente necesitas.
- Analiza tus consultas con herramientas como EXPLAIN o el analizador de consultas en SQL Server.

✅ MEJORES PRÁCTICAS EN BASES DE DATOS

🔐 Seguridad

- Limita los permisos de los usuarios: solo lo necesario.
- Encripta datos sensibles como contraseñas o información personal.
- Haz respaldos frecuentes y verifica que puedan restaurarse.

🧰 Mantenimiento

- Monitorea el uso y el rendimiento.
- Elimina datos obsoletos o archívalos si es necesario.
- Documenta tu esquema: explica qué hace cada tabla, campo y relación.

¿QUÉ ES UN MANEJADOR DE BASE DE DATOS (DBMS)?

Un **DBMS (Database Management System)** es un software que permite crear, leer, actualizar y eliminar datos en una base de datos. Proporciona herramientas para administrar usuarios, respaldos, seguridad, integridad y rendimiento.

- **Base de datos**: Colección organizada de datos.
- **Tabla**: Estructura que almacena datos en filas y columnas.
- **Consulta (Query)**: Instrucción SQL para interactuar con los datos.
- **Índice**: Estructura que mejora la velocidad de búsquedas.
- **Primary Key**: Campo único que identifica cada fila.
- **Foreign Key**: Clave que relaciona tablas entre sí.
- **Normalización**: Proceso de organización para reducir redundancia.

🔷 MANEJADORES DE BASES DE DATOS RELACIONALES (RDBMS) *(Usan tablas, SQL y estructuras normalizadas)*

DBMS	Descripción	Licencia
SQL Server (Express / Standard / Enterprise)	De Microsoft. Muy usado en entornos empresariales Windows. Express es gratuito.	Propietaria (Express es gratuita)
MySQL	Muy popular, rápido, de código abierto. Ideal para web (PHP, WordPress, etc).	Open Source (GPL)
PostgreSQL	Avanzado, robusto y gratuito. Gran soporte para integridad y transacciones complejas.	Open Source
Oracle Database	Potente y completo. Muy usado en bancos y grandes empresas.	Propietaria (versión gratuita: Oracle XE)
MariaDB	Derivado de MySQL, más libre y con mejoras. Totalmente compatible.	Open Source
SQLite	Ligero y embebido. Se usa en apps móviles, escritorio y pruebas.	Dominio público
IBM Db2	De IBM, usado en entornos corporativos grandes.	Propietaria
Firebird	Liviano, gratuito y con buena performance.	Open Source

MANEJADORES DE BASES DE DATOS NO RELACIONALES (NOSQL)

DBMS	Tipo	Descripciòn
MongoDB	Documental (JSON/BSON)	Muy popular para apps modernas y flexibles. Escalable.
Redis	**Clave-valor (en memoria)**	Súper rápido, ideal para caché y sesiones.
Cassandra	Columnares distribuidas	Escalable horizontalmente. Usado por empresas grandes.
Neo4j	Grafos	Ideal para relaciones complejas, redes sociales, etc.
CouchDB	Documental (JSON)	Similar a Mongo pero orientado a web y replicación.
Firebase Realtime DB/Firestore	Documental	De Google, muy usado en apps mòviles

RECOMENDACIONES SEGÚN EL USO:

- **Proyectos Web pequeños o medianos:** MySQL, PostgreSQL, SQLite
- **Empresas Microsoft:** SQL Server.
- **Proyectos de análisis complejos:** PostgreSQL, Oracle
- **Apps móviles:** SQLite, Firebase
- **Big Data o alta concurrencia:** Cassandra, MongoDB, Redis

🧠 METODOLOGÍAS DE DISEÑO DE DATOS

📌 Modelo Conceptual

- **¿Qué es?**
  Representación de alto nivel de los datos y sus relaciones sin detalles técnicos.
- **Herramientas típicas:** Diagramas Entidad-Relación (ERD).
- **Objetivo:** Entender qué entidades (objetos) existen y cómo se relacionan.
- **Ejemplo:** Paciente, Cita, Médico, Especialidad.

MODELO LÓGICO

- **¿Qué es?**

  Traducción del modelo conceptual a un modelo lógico basado en un sistema de base de datos relacional.
- **Características:**
  - Se definen **atributos** (campos/columnas).
  - Se asignan **tipos de datos** (ej. VARCHAR, INT, DATE).
  - Se identifican **llaves primarias y foráneas**.
  - No se aplica aún a un motor específico.
- **Ejemplo:** La entidad *Paciente* tiene IdPaciente (PK), Nombre, FechaNacimiento.

MODELO FÍSICO

- **¿Qué es?**

  Implementación concreta del modelo lógico en un DBMS específico.

- **Incluye:**
- Sintaxis SQL para crear tablas, relaciones, índices, restricciones.
- Consideraciones de rendimiento (índices, particiones, normalización/desnormalización).
- Optimización de almacenamiento y consultas.

✅ BUENAS PRÁCTICAS EN EL DISEÑO DE BASES DE DATOS

Práctica	Descripción
Normalización	Aplicar reglas (1FN, 2FN, 3FN…) para eliminar redundancia.
Evitar nombres ambiguos	Usa nombres descriptivos: Paciente, Cita, no Tabla1.
Definir claves primarias únicas	Cada tabla debe tener un identificador claro.
Usar claves foráneas correctamente	Mantiene la integridad entre tablas relacionadas.
Índices bien pensados	Mejoran el rendimiento de búsqueda, pero no abuses.
Documentar todo	Facilita mantenimiento y comprensión.
Planificar crecimiento	Considerar volumen futuro de datos y concurrencia.
Backups y seguridad	Políticas de respaldo, usuarios con permisos limitados

🔧 METODOLOGÍAS FORMALES APLICADAS AL DISEÑO

Metodología	Descripción
MER (Modelo Entidad-Relación)	Más usado. Permite diagramar entidades, relaciones, atributos.
Modelo Objeto-Relacional (ORM)	Integra programación orientada a objetos con bases relacionales (ej. Entity Framework).
Modelo Jerárquico / Red (antiguos)	Pocas veces usados hoy en día, excepto en DBs específicas.
Modelo Dimensional (Data Warehousing)	Usado para análisis de datos: tablas de hechos y dimensiones.

CASO PRÁCTICO: MANEJO DE PACIENTES Y CITAS MÉDICAS

El diseño de bases de datos es fundamental para crear sistemas eficientes, escalables y fáciles de mantener. Esta guía te presenta metodologías de diseño de datos, buenas prácticas y un ejemplo práctico centrado en el registro de pacientes y gestión de citas médicas.

Modelo Conceptual

- Se define **qué información** se va a guardar.
- Se identifican las **entidades** y las **relaciones**.
- No se utilizan detalles técnicos ni nombres de campos.

Entidades del ejemplo:

- **Paciente**
- **Médico**
- **Cita**
- **Especialidad**

Relaciones clave:

- Un paciente puede tener muchas citas.
- Un médico puede atender muchas citas.
- Cada médico pertenece a una especialidad.

MODELO LÓGICO

- Traducción a tablas y columnas.
- Se asignan tipos de datos generales.
- Se definen claves primarias y relaciones (foráneas).

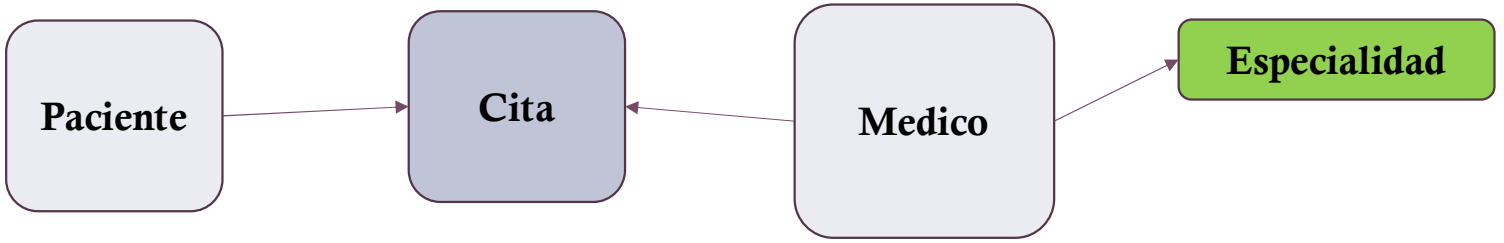
Ejemplo de modelo lógico (resumen):

Tabla	Clave Primaria	Atributos clave
Paciente	IdPaciente	Nombre, FechaNacimiento, Teléfono
Médico	IdMedico	Nombre, EspecialidadId
Especialidad	IdEspecialidad	Nombre
Cita	IdCita	FechaHora, IdPaciente, IdMedico, Motivo

MODELO FÍSICO

- Se implementa el modelo lógico con SQL.
- Se usan tipos de datos específicos (ej. INT, VARCHAR(100)).
- Se añaden índices, restricciones y reglas.

Diagrama Entidad-Relación (ER)



- [Paciente] tiene muchas [Citas]
- [Medico] atiende muchas [Citas]
- Cada [Medico] pertenece a una [Especialidad]

TABLAS QUE PODEMOS IDENTIFICAR Y CREAR:

- Pacientes
- Medicos
- Citas
- Especialidad

(Opcionalmente se pueden agregar otras: tratamientos, recetas, historial médico, etc.)

📝 SCRIPT COMPATIBLE CON SQL SERVER EXPRESS 2022

```
-- Crear base de datos
CREATE DATABASE ClinicaMedica;
GO

USE ClinicaMedica;
GO

-- Tabla de Pacientes
CREATE TABLE Pacientes (
    ID_Paciente INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(100) NOT NULL,
    Apellido NVARCHAR(100) NOT NULL,
    FechaNacimiento DATE,
    Sexo CHAR(1),
    Telefono NVARCHAR(20),
    Direccion NVARCHAR(255),
    Email NVARCHAR(100)
);
```

SCRIPT COMPATIBLE CON SQL SERVER EXPRESS 2022

```
-- Tabla de Citas                                   );
CREATE TABLE Citas (
    ID_Cita INT PRIMARY KEY
IDENTITY(1,1),
    ID_Paciente INT FOREIGN KEY
REFERENCES Pacientes(ID_Paciente),
    ID_Medico INT FOREIGN KEY
REFERENCES Medicos(ID_Medico),
    FechaHora DATETIME NOT NULL,
    Motivo NVARCHAR(255),
    Estado NVARCHAR(50) DEFAULT
'Pendiente'
```

SCRIPT COMPATIBLE CON MYSQL

```
-- Crear base de datos
CREATE DATABASE ClinicaMedica;
USE ClinicaMedica;

-- Tabla de Pacientes
CREATE TABLE Pacientes (
    ID_Paciente INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Apellido VARCHAR(100) NOT NULL,
    FechaNacimiento DATE,
    Sexo CHAR(1),
    Telefono VARCHAR(20),
    Direccion VARCHAR(255),
    Email VARCHAR(100)
);
```

SCRIPT COMPATIBLE CON MYSQL

```
-- Tabla de Médicos
CREATE TABLE Medicos (
    ID_Medico INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Apellido VARCHAR(100) NOT NULL,
    Especialidad VARCHAR(100),
    Telefono VARCHAR(20),
    Email VARCHAR(100)
);
CREATE TABLE Especialidad (
    IdEspecialidad INT PRIMARY KEY IDENTITY,
    Nombre VARCHAR(100) NOT NULL
);
```

```
-- Tabla de Citas
CREATE TABLE Citas (
    ID_Cita INT AUTO_INCREMENT PRIMARY KEY,
    ID_Paciente INT,
    ID_Medico INT,
    FechaHora DATETIME NOT NULL,
    Motivo VARCHAR(255),
    Estado VARCHAR(50) DEFAULT 'Pendiente',
    FOREIGN KEY (ID_Paciente) REFERENCES
Pacientes(ID_Paciente),
    FOREIGN KEY (ID_Medico) REFERENCES Medicos(ID_Medico)
);
```