Unidad II: Base de Datos relacional

Estructura de tablas: relaciones, con columnas (atributos) y filas (tuplas o registros). Tabla como entidad del sistema.

Llaves primarias o Primary Keys. Llaves foráneas o Foreign Keys. Uso de vinculaciones lógicas.

Relaciones: 1:1 (uno a uno); 1:M (uno a muchos); M:M (muchos a muchos); M:1 (muchos a uno).

Modelo relacional de Datos en PowerBI, automático o generado en base a objetivos.

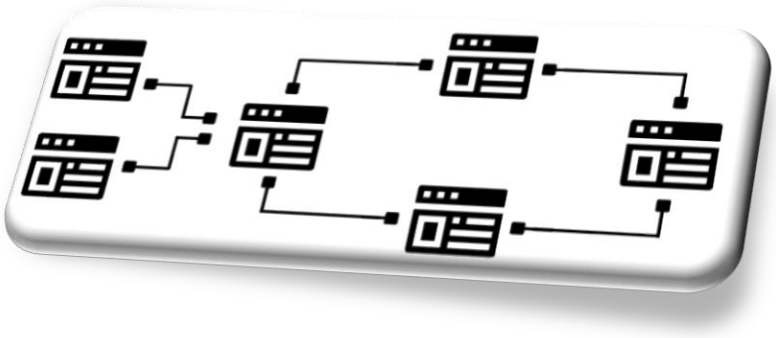
Consultas SQL, lenguaje estándar. Integridad de los datos. Normalización.

Ventajas y Desventajas de una Base de Datos relacional.

Lenguajes en SQL para manipulación de datos. DQL: SELECT. DML: INSERT, UPDATE, DELETE; DDL: CREATE, ALTER, DROP. DCL: GRANT, REVOKE. DTL: COMMIT, ROLLBACK.

Cálculos sobre datos. Funciones agregadas. GROUP BY: MIN, STDEV, AVG, SUM, MAX, COUNT.

Características principales de una base de datos relacional

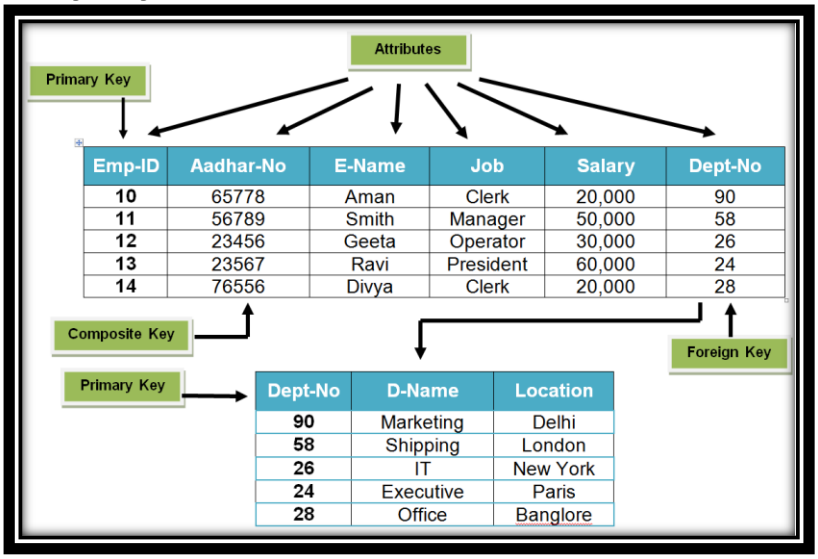


1- Estructura de Tablas: los datos se almacenan en:
a. tablas→ relaciones,
b. que tienen **columnas→** atributos
c. y **filas→** tuplas o registros

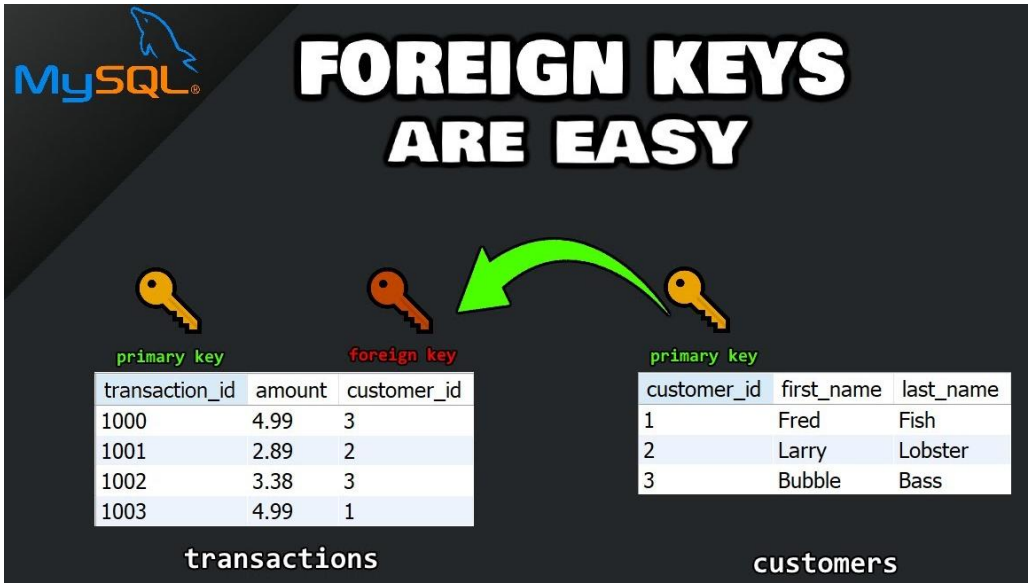
Cada **Tabla** representa una **entidad** del sistema: alumnos, pacientes, productos, empleados, accidentes, etc.

Llaves primarias o Primary Keys

Cada **Tabla** debe tener una **columna** (o un conjunto de columnas) que contenga/n un **valor único para cada fila**.
Esta columna es conocida como **clave primaria (Primary Key)** y asegura que cada registro en la **Tabla** sea único.

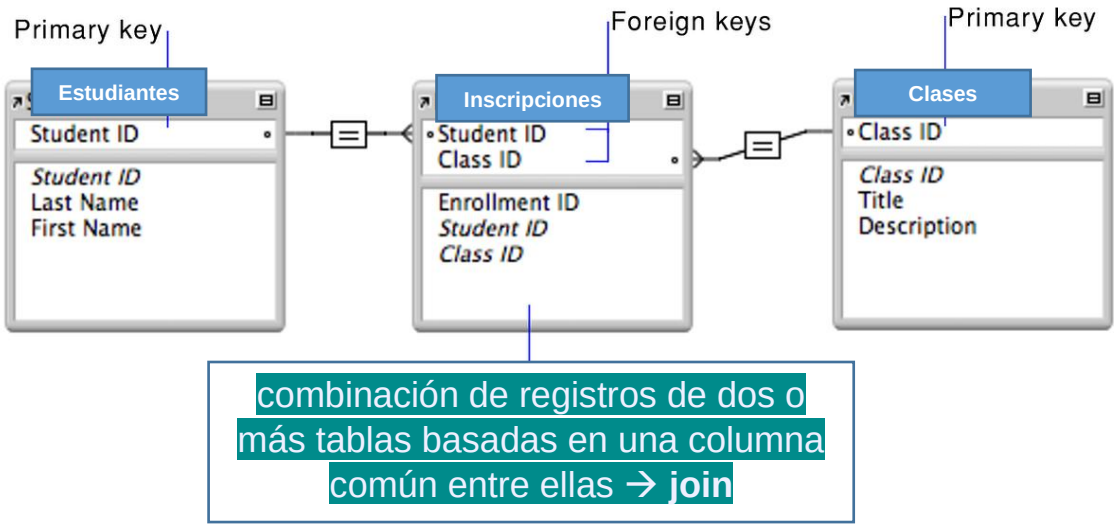


Llaves foráneas o Foreign Keys



Una **Llave foránea o Foreign Keys** es un campo en una **Tabla** que se refiere a la **Clave Primaria** de otra tabla.

Esto crea una relación entre **Tablas** y permite ***vincular datos*** en diferentes tablas de manera lógica.

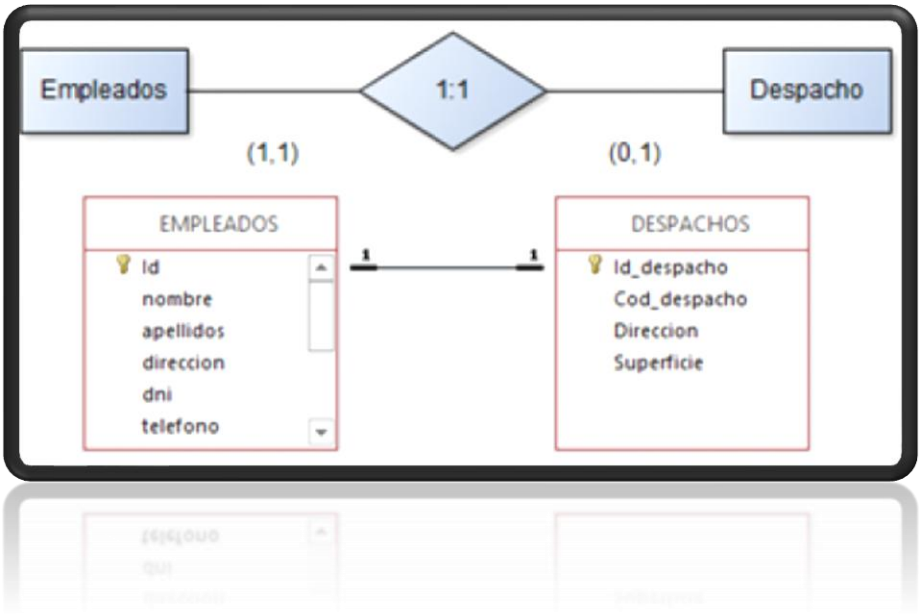


Relaciones

Dentro de las bases de datos relacionales permiten definir diferentes tipos de relaciones entre tablas:

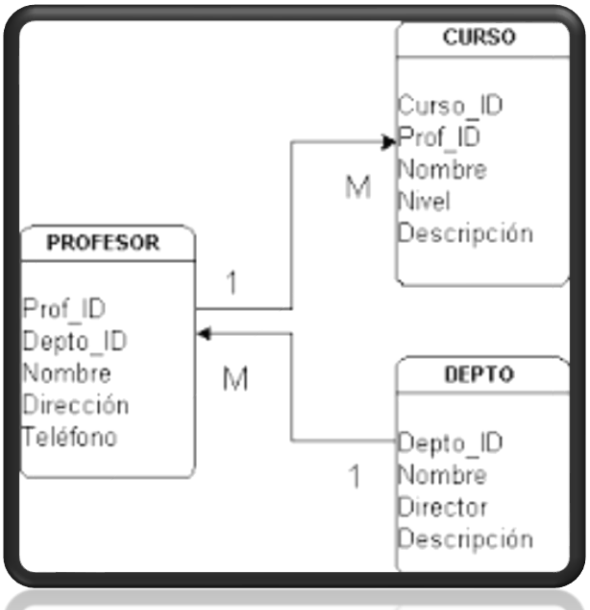
- **Uno a uno→(1:1)**

  Una fila de una tabla ***está relacionada con solo una fila*** de otra tabla.

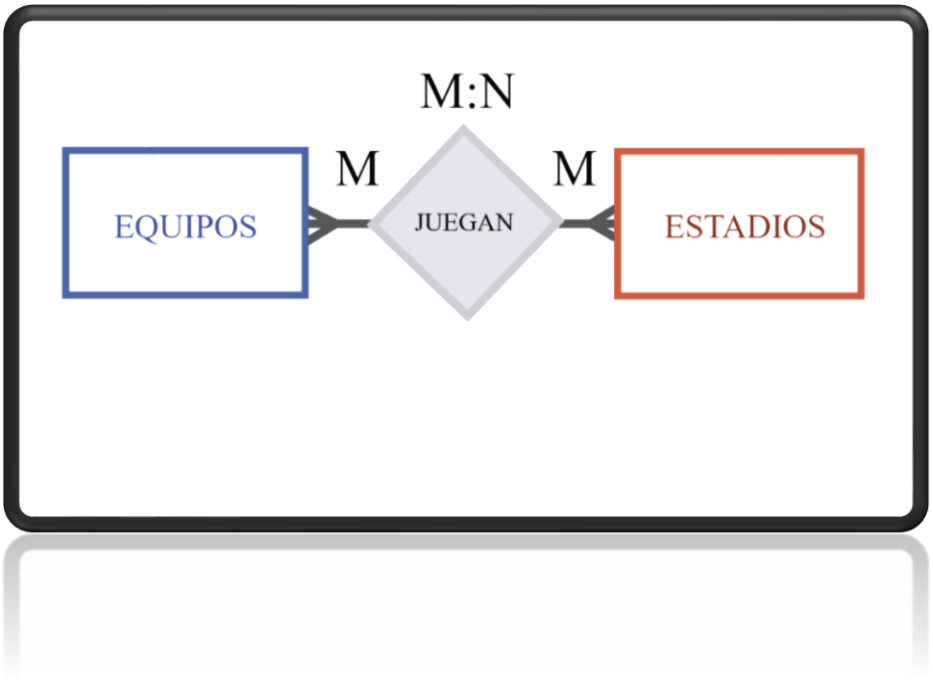


- **Uno a muchos→(1:M)**

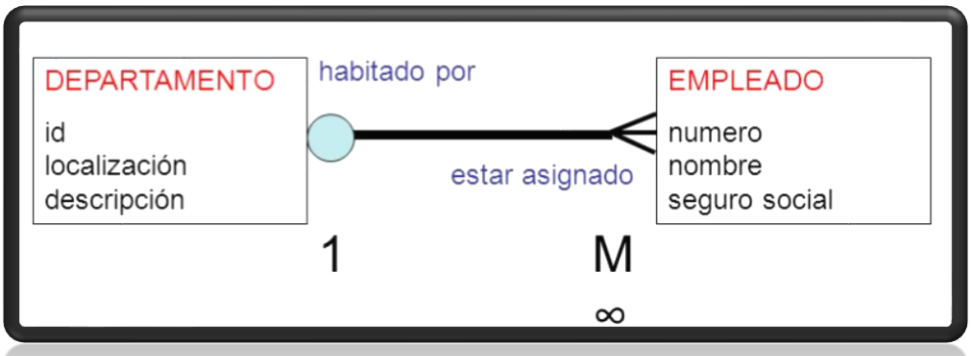
  Una fila de una tabla está relacionada con solo una fila de otra tabla.



- **Muchos a muchos→(M:M)**
  Varias filas de una tabla pueden estar relacionadas con varias filas de otra tabla intermedia.



- **Muchos a uno→(M:1)**
  Varias filas de una tabla pueden estar asociadas con una sola fila de otra tabla.



Consultas SQL

Las bases de datos relacionales usan SQL como lenguaje estándar para interactuar con los datos.

Integridad de los Datos

Las bases de datos relacionales **garantizan** la integridad referencial, que asegura que las **relaciones entre tablas sean válidas**, y que las **claves foráneas** correspondan a **claves primarias** existentes.

La integridad de los datos es un aspecto crítico de la estrategia de gestión de datos de cualquier organización.

Se refiere a la

a. precisión,
b. coherencia y
c. confiabilidad de los datos

A lo largo de su ciclo de vida

Garantizar la integridad de los datos es primordial para que las empresas tomen **decisiones informadas**, mantengan el **cumplimiento normativo** y generen **confianza con los clientes y las partes interesadas**.

Ahora **¿Quiénes son las partes interesadas o stakeholders?**

son todas aquellas **personas, grupos u organizaciones** que **tienen interés o están afectadas por un proyecto, decisión o resultado**. En el contexto empresarial y de gestión de proyectos, los stakeholders **pueden incluir** desde

empleados,

clientes,

proveedores y accionistas,

hasta reguladores, comunidades locales y competidores.

Tipos de Stakeholders

Internos

- **Empleados**: Aquellos que trabajan directamente dentro de la empresa.
- **Gerencia**: Los que toman decisiones dentro de la empresa.
- **Accionistas**: Personas u organizaciones que poseen acciones de la empresa.

Externos

- **Clientes**: Los que compran o usan los productos o servicios.
- **Proveedores**: Empresas o personas que suministran bienes y servicios.
- **Gobiernos y reguladores**: Entidades que regulan las actividades de la empresa.
- **Comunidad**: Grupos que se ven afectados por las operaciones de la empresa, como la comunidad local o ambiental.

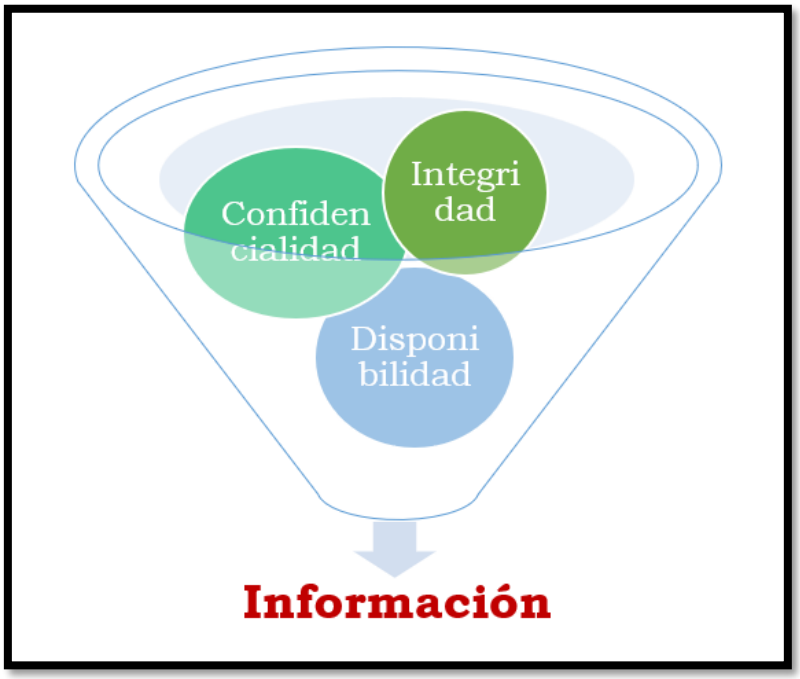
Los stakeholders son clave en la toma de decisiones, ya que sus intereses, expectativas y necesidades influyen en la dirección de un proyecto o empresa.

Mejores prácticas para garantizar la integridad de los datos

Para mitigar los riesgos asociados con la integridad de los datos, las organizaciones deben **implementar** mejores prácticas que aborden las **vulnerabilidades potenciales**. Aquí hay algunas **estrategias clave** a considerar:

1. **Evaluaciones periódicas de la calidad de los datos**: realizar evaluaciones periódicas para identificar y rectificar problemas de calidad de los datos. Esto implica evaluar la integridad, precisión, coherencia y puntualidad de los datos. La implementación de herramientas automatizadas de calidad de datos puede agilizar este proceso y proporcionar información en tiempo real.
2. **Implementar reglas de validación de datos:** Establecer reglas de validación de datos para evitar el ingreso de datos erróneos o inconsistentes. Al establecer criterios de validación, las organizaciones pueden garantizar la integridad de los datos en el punto de entrada. Por ejemplo, un campo de número de seguro social se puede validar para aceptar sólo entradas numéricas de nueve dígitos.
3. **Controles de acceso basados en roles**: Restringir el acceso a los datos según los roles de los usuarios es crucial para la integridad de los datos. Al implementar controles de acceso granulares, las organizaciones pueden evitar modificaciones no autorizadas, garantizando que solo el personal autorizado pueda realizar cambios en los datos críticos.
4. **Cifrado de datos**: cifrar datos confidenciales tanto en reposo como en tránsito agrega una capa adicional de protección. Los algoritmos de cifrado transforman los datos en un formato ilegible, lo que hace que sea casi imposible descifrarlos para personas no autorizadas.

5. **Copias de seguridad de datos periódicas**: implementar una estrategia de copia de seguridad sólida es esencial para proteger contra la pérdida o corrupción de datos. Realizar copias de seguridad de los datos con regularidad garantiza que, en caso de incidentes imprevistos, como fallas del sistema o ataques cibernéticos, las organizaciones puedan restaurar los datos a un buen estado conocido.

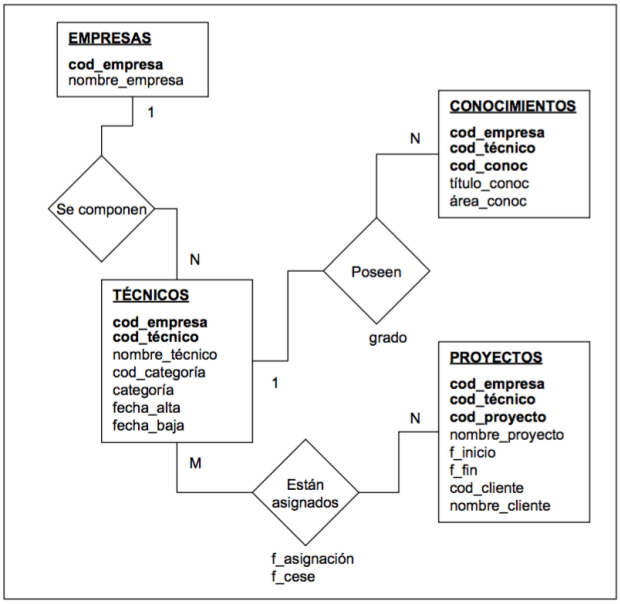


Normalización

Las **bases de datos relacionales** permiten **normalizar los datos**.

Normalizar es un proceso que **divide los datos** en **múltiples tablas** para reducir la **redundancia** y **evitar** la duplicación de datos, **mejorando la eficiencia y la consistencia de la base de datos.**

Incluye la **creación de tablas** y el **establecimiento de relaciones** entre ellas según **reglas diseñadas** tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes.



Hay algunas reglas en la normalización de una base de datos. Cada regla se llama **"forma normal"**. Si se cumple la primera regla, se dice que la base de datos está en **"primera forma normal"**. "Si se observan las tres primeras reglas, se considera que la base de datos está en **"tercera forma normal".** Aunque son

posibles otros niveles de normalización, la tercera forma normal se considera el nivel más alto necesario para la mayoría de las aplicaciones.

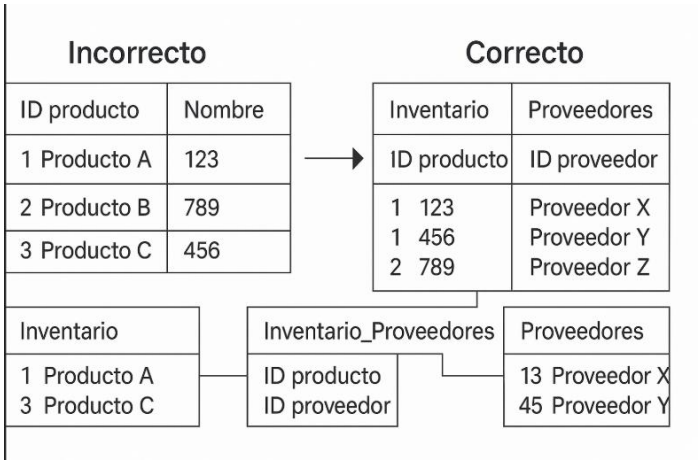
Al igual que con otras muchas reglas y especificaciones formales, los escenarios del mundo real no siempre permiten un cumplimiento perfecto. En general, **la normalización requiere tablas adicionales y algunos clientes consideran éste un trabajo considerable.** Si decide infringir una de las tres primeras reglas de la normalización, asegúrese de que su aplicación se anticipa a los problemas que puedan aparecer, como la existencia de datos redundantes y de dependencias incoherentes.

Primera forma normal

- Elimine los grupos repetidos de las tablas individuales.
- Cree una tabla independiente para cada conjunto de datos relacionados.
- Identifique cada conjunto de datos relacionados con una clave principal.
- No use varios campos en una misma tabla para almacenar datos similares.

Por ejemplo, para realizar el seguimiento de un elemento del inventario que proviene de dos orígenes posibles, un registro del inventario puede contener campos para el Código de proveedor 1 y para el Código de proveedor 2.

¿Qué ocurre cuando se agrega un tercer proveedor? Agregar un campo no es la respuesta; requiere modificaciones del programa y de la tabla y no se adapta sin problemas a un número dinámico de proveedores.



En su lugar, coloque **toda la información de los proveedores** en una ***tabla independiente denominada Proveedores*** y después vincule el inventario a los proveedores con el **número de elemento como clave**, o **los proveedores al inventario con el código de proveedor como clave.**

Segunda forma normal

- Cree tablas independientes para conjuntos de valores que se apliquen a varios registros.
- Relacione estas tablas con una clave externa.
- Los registros no deben depender de nada más que de la clave principal de una tabla (una clave compuesta, si es necesario).

Por ejemplo, considere la **dirección de un cliente** en un **sistema de logística**. La dirección se necesita en la tabla Clientes, pero también en las tablas Pedidos, Envíos, Facturas, Cuentas por cobrar y Colecciones.

En lugar de almacenar la dirección de un cliente como una entrada independiente en cada una de estas tablas, almacénela en un lugar, ya sea en la tabla Clientes o en una tabla Direcciones independiente.

Tercera forma normal

- Eliminar los campos que no dependen de la clave.
- Los valores de un registro que no forman parte de la clave de ese registro no pertenecen a la tabla.

En general, siempre que el contenido de un grupo de campos pueda aplicarse a más de un único registro de la tabla, considere colocar estos campos en una tabla independiente.

Por ejemplo, en una **tabla Contratación de empleados**, puede incluirse el nombre de la universidad y la dirección de un candidato. Pero necesita una **lista completa de universidades** para enviar mensajes de correo electrónico en grupo. Si la información de las universidades se almacena en la **tabla Candidatos**, no hay forma de enumerar las universidades que no tengan candidatos en ese momento. Cree una tabla Universidades independiente y vincúlela a la tabla Candidatos con el código de universidad como clave.

A tener en cuenta: muchas tablas pequeñas pueden ***degradar el rendimiento o superar la capacidad de memoria o de archivos abiertos.***

Puede ser más factible ***aplicar la tercera forma normal sólo a los datos que cambian con frecuencia****. Si quedan algunos campos dependientes, diseñe la aplicación para que pida al usuario que compruebe todos los campos relacionados cuando cambie alguno.*

1. **Tabla sin normalizar**

N° alumno	Tutor	Despacho Tutor	Clase 1	Clase 2	Clase 3
1022	García	412	101-07	143-01	159-02
4123	Díaz	216	101-07	143-01	179-04

2. **Primera forma normal: sin grupos repetidos**

Las tablas sólo deben tener dos dimensiones. Puesto que un alumno tiene varias clases, estas clases deben aparecer en una tabla independiente. Los campos Clase1, Clase2 y Clase3 de los registros anteriores son indicativos de un problema de diseño.

Las hojas de cálculo suelen usar la tercera dimensión, pero las tablas no deberían hacerlo.

Otra forma de considerar este problema es con **una relación de uno a varios y poner el lado de uno y el lado de varios en la misma tabla.** En su lugar, cree otra tabla en la primera forma normal eliminando el grupo repetido (N.º clase), según se muestra a continuación:

N° alumno	Tutor	Despacho tutor	N° Clase
1022	García	412	101-07
1022	García	412	143-01
1022	García	412	159-02
4123	Díaz	216	101-07
4123	Díaz	216	143-01
4123	Díaz	216	179-04

3. **Segunda forma normal: eliminar datos redundantes**

Observe los diversos valores de N.º clase para cada valor de N.º alumno en la tabla anterior.

El N.º clase no depende funcionalmente de N.º alumno (la clave principal), de modo que la relación no cumple la segunda forma normal.

Las tablas siguientes demuestran la segunda forma normal:

Alumnos

N° alumno	Tutor	Despacho tutor
1022	García	412
4123	Díaz	216

Registro

N° alumno	N° Clase
1022	101-07
1022	143-01
1022	159-02
4123	101-07
4123	143-01
4123	179-04

4. Tercera forma normal: eliminar datos que no dependen de la clave

En el último ejemplo, Despacho-Tutor (el número de despacho del tutor) es funcionalmente dependiente del atributo Tutor.

La solución es pasar ese atributo de la tabla Alumnos a la tabla Personal, según se muestra a continuación:

Alumnos

N° alumno	Tutor
1022	García
4123	Díaz

Personal

Nombre	Sala	Departamento
García	412	42
Díaz	216	42

Ventajas de una Base de datos relacional

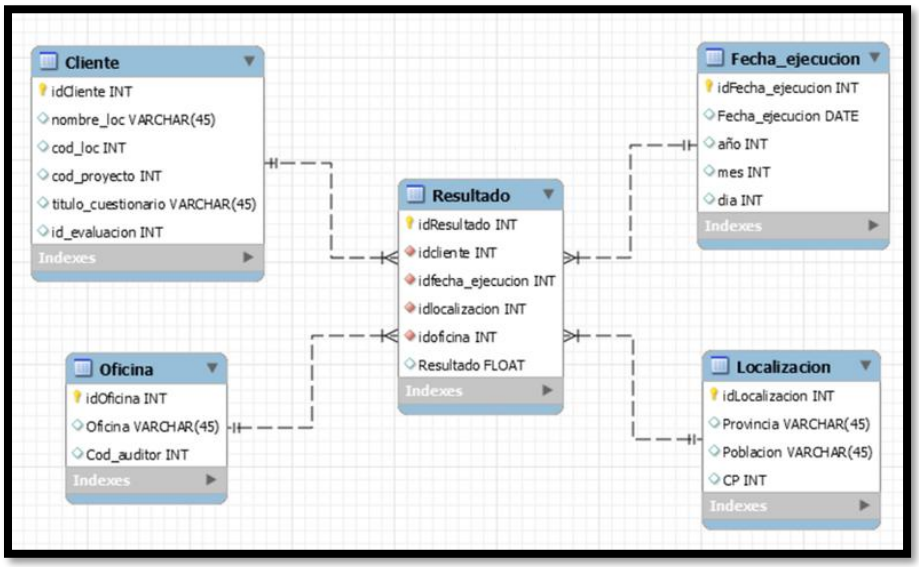
Estructura clara: los datos se organizan en tablas que son fáciles de entender y gestionar.

Flexibilidad en consultas: el lenguaje SQL permite hacer consultas complejas de manera eficiente.

Integridad y consistencia: los sistemas relacionales garantizan que los datos sean consistentes y precisos mediante claves y relaciones entre tablas.

Escalabilidad: las bases de datos relacionales pueden manejar grandes volúmenes de datos.

Seguridad: ofrecen control sobre el acceso a los datos y la protección contra la pérdida de información.



Desventajas de una Base de datos relacional

Escalabilidad horizontal limitada: para aplicaciones de gran tamaño, las bases de datos relacionales pueden tener dificultades para escalar horizontalmente (es decir, agregar más servidores), lo que ha impulsado la adopción de bases de datos NoSQL en ciertos casos.

Complejidad: en sistemas grandes, las relaciones entre tablas pueden volverse complejas, lo que puede requerir un diseño cuidadoso para mantener la eficiencia.

Componentes de una Base de datos relacional

Tabla: un conjunto de filas y columnas. Cada tabla representa una entidad.

Fila (registro): una entrada de datos en la tabla, representando un único elemento de la entidad.

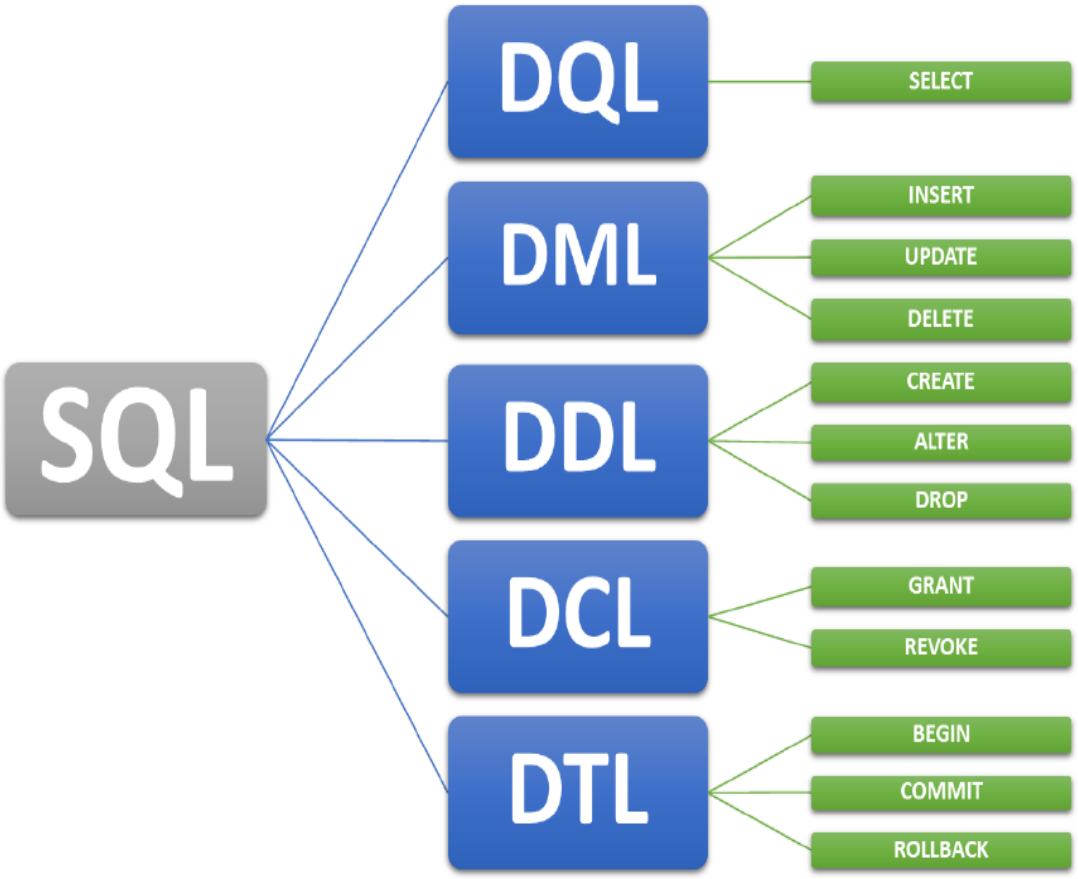
Columna (campo): define una categoría de información dentro de la tabla, como nombre, edad, o dirección.

Clave Primaria (Primary Key): un identificador único para cada fila dentro de una tabla. Garantiza que no haya duplicados.

Clave Foránea (Foreign Key): un campo en una tabla que se utiliza para establecer una relación con otra tabla. Sirve para conectar filas entre tablas diferentes.

SQL está compuesto por varios subconjuntos de lenguajes que permiten manipular datos y definir estructuras de bases de datos.

Los principales son:



DQL (Data Query Language)
Lenguaje de Consultas de Datos

El comando SELECT es el más importante y utilizado dentro del DQL, ya que permite extraer información de una o más tablas en una base de datos.
Se puede combinar con diversas cláusulas y operadores para especificar qué datos queremos recuperar, cómo los queremos ordenar, agrupar o filtrar.

SELECT: Recupera datos de una tabla.

DML (Data Manipulation Language)
Lenguaje de Manipulación de Datos

Estos comandos permiten insertar, actualizar, eliminar y recuperar datos de las tablas.

INSERT: Agrega nuevos registros a una tabla.
UPDATE: Modifica registros existentes.
DELETE: Elimina registros de una tabla.

DDL (Data Definition Language)
Lenguaje de Definición de Datos

Este grupo de comandos permite definir la estructura de la base de datos, como la creación de tablas, modificación de su estructura o eliminación de datos.

CREATE: Crea nuevas bases de datos o tablas.
ALTER: Modifica la estructura de una tabla, como agregar o eliminar columnas.
DROP: Elimina tablas, bases de datos o índices.

DCL (Data Control Language) Lenguaje de Control de Datos

Este lenguaje se utiliza para controlar el acceso a la base de datos, asignando permisos a los usuarios.

GRANT: Otorga permisos a un usuario para realizar operaciones.

REVOKE: Revoca permisos que han sido asignados.

DTL (Data Transaction Language) Lenguaje de Transacción de Datos

(El término DTL no es tan común como TCL (Transaction Control Language) y a menudo se usa de manera intercambiable con TCL en algunos contextos. Sin embargo, DTL se refiere más a las operaciones de manipulación de transacciones de datos, que incluyen tanto el control de transacciones como la gestión de datos en movimiento. En algunos casos, DTL se usa para describir operaciones que manejan datos en un entorno transaccional.)

Permite controlar el manejo de transacciones en la base de datos, asegurando que las operaciones se realicen correctamente y con integridad.

COMMIT: Guarda de forma permanente los cambios realizados en una transacción.
ROLLBACK: Deshace los cambios realizados en una transacción si hay un error.

SQL permite realizar cálculos sobre los datos mediante funciones agregadas, que se utilizan en conjunto con la cláusula GROUP BY.

Las funciones más comunes son: