



UF2175: Diseño de Bases de Datos Relacionales

Unidad Didáctica 1: Introducción a las Bases de Datos

1. Evolución histórica de las bases de datos

- Sistemas de archivos plano
- Bases de datos jerárquicas y en red
- Bases de datos relacionales
- Bases de datos orientadas a objetos
- Bases de datos NoSQL
- Bases de datos en la nube

2. Ventajas e inconvenientes de las bases de datos

- Ventajas:
 - Integridad y coherencia
 - Reducción de redundancia
 - Facilidad de acceso
 - Seguridad y control de acceso
 - Escalabilidad
- Inconvenientes:
 - Complejidad inicial
 - Coste de implementación
 - Dependencia tecnológica

3. Conceptos generales

- Base de datos (BD)
 - Sistema gestor de bases de datos (SGBD)
 - Tabla, campo, registro
 - Consultas, formularios, informes
 - Transacciones, concurrencia y control de acceso
-



UF2175: Diseño de Bases de Datos Relacionales

Unidad Didáctica 2: Modelos Conceptuales de Bases de Datos

1. El modelo entidad-relación (E-R)

- Entidades y atributos
- Tipos de relaciones (uno a uno, uno a muchos, muchos a muchos)
- Diagrama Entidad-Relación (DER)
- Identificación de claves primarias

2. El modelo entidad-relación extendido (EER)

- Especialización y generalización
- Herencia y jerarquías
- Agregación y composición

3. Restricciones de integridad

- Integridad de entidad
 - Integridad referencial
 - Integridad de dominio
 - Restricciones específicas y reglas de negocio
-

Unidad Didáctica 3: El Modelo Relacional

1. Evolución del modelo relacional

- Definición y características
- Aportes de Edgar Codd
- Principios fundamentales del modelo relacional

2. Estructura del modelo relacional

- Tablas, filas y columnas
- Tipos de datos
- Relaciones entre tablas (claves foráneas)

3. Claves en el modelo relacional

- Claves primarias y candidatas
- Claves foráneas
- Superclave, clave compuesta, clave alternativa



UF2175: Diseño de Bases de Datos Relacionales

4. Restricciones de integridad

- Restricción de unicidad
- Restricción de valores nulos
- Restricciones de integridad referencial avanzada

5. Teoría de la normalización

- Objetivos de la normalización
 - Primera forma normal (1FN)
 - Segunda forma normal (2FN)
 - Tercera forma normal (3FN)
 - Formas normales avanzadas (BCNF, 4FN, 5FN)
-

Unidad Didáctica 4: El Ciclo de Vida de un Proyecto

1. El ciclo de vida de una base de datos

- Análisis de requerimientos
- Diseño conceptual
- Diseño lógico y físico
- Implementación y creación de la base de datos
- Pruebas y ajustes
- Mantenimiento y evolución

2. Conceptos generales del control de calidad

- Técnicas y herramientas de calidad
 - Validación y verificación del diseño
 - Auditorías de calidad de bases de datos
-



UF2175: Diseño de Bases de Datos Relacionales

Unidad Didáctica 5: Creación y Diseño de Bases de Datos

1. Enfoques de diseño

- Enfoque descendente (top-down)
- Enfoque ascendente (bottom-up)
- Enfoque mixto

2. Metodologías de diseño

- Metodología entidad-relación
- Metodologías ágiles en diseño de bases de datos
- Metodologías orientadas a objetos y relacional-objeto

3. Estudio del diseño lógico de una base de datos relacional

- Transformación de un modelo conceptual (E-R) al modelo relacional
- Aplicación práctica de la normalización
- Creación de tablas, claves y relaciones en SQL

4. El diccionario de datos: concepto y estructura

- Definición e importancia
- Componentes del diccionario de datos
- Documentación del diseño lógico y físico

5. Estudio del diseño de la base de datos y de los requisitos de usuario

- Análisis y documentación de requisitos funcionales y no funcionales
- Técnicas de recopilación de requisitos (entrevistas, cuestionarios, observación)
- Caso práctico de recopilación y análisis de requerimientos para base de datos



UF2175: Diseño de Bases de Datos Relacionales

Tema 1: Introducción a las Bases de Datos

1.1 Evolución histórica de las bases de datos

- **Archivos planos:** En los primeros tiempos, los datos se almacenaban en archivos simples sin organización interna, provocando redundancias e inconsistencias frecuentes.
 - **Bases de datos jerárquicas y en red (años 60-70):** Introducción de estructuras organizadas en forma de árbol o redes. Estas permitían relaciones estructuradas, pero eran difíciles de mantener y modificar.
 - **Modelo relacional (1970):** Edgar F. Codd establece una base formal para las bases de datos con el modelo relacional. Esto facilitó el desarrollo del lenguaje SQL, simplificando las consultas y mejorando la integridad de los datos.
 - **Bases de datos orientadas a objetos (años 80-90):** Incorporan conceptos de la programación orientada a objetos, gestionando tipos de datos complejos como multimedia.
 - **Bases de datos NoSQL (2000 en adelante):** Surgen ante la necesidad de gestionar grandes volúmenes de datos no estructurados. Tipos comunes incluyen clave-valor, orientadas a documentos y grafos.
 - **Bases de datos en la nube (actualidad):** Proporcionan almacenamiento escalable y accesible desde cualquier lugar mediante servicios en la nube, como AWS, Azure o Google Cloud.
-



UF2175: Diseño de Bases de Datos Relacionales

Las bases de datos han evolucionado significativamente desde su creación inicial hasta la actualidad. Este proceso histórico puede dividirse en varias etapas importantes, cada una con características distintivas y enfoques diferentes sobre cómo gestionar la información:

Etapas de evolución: Etapa inicial: Sistemas de archivos planos (Décadas de 1950 y 1960)

En los comienzos de la informática, la gestión de datos se realizaba mediante archivos planos o simples sistemas de ficheros. Estos eran archivos sin estructura formal que almacenaban la información secuencialmente.

Características principales:

- Datos almacenados en archivos independientes.
- Ausencia de estructura formal.
- Redundancia de datos elevada.
- Complejidad en la actualización y consulta de datos.

Problemas frecuentes:

- Duplicación de información.
 - Falta de integridad y coherencia.
 - Dificultad en la recuperación eficiente de datos.
-

Etapas de evolución: Bases de datos jerárquicas y de red (Décadas de 1960 y 1970)

La necesidad de organizar la información llevó al desarrollo de las primeras bases de datos estructuradas:

Modelo jerárquico:

- Utiliza una estructura similar a un árbol.
- Cada elemento o registro puede tener solo un padre pero múltiples hijos.
- Ejemplos típicos incluyen IMS de IBM.



UF2175: Diseño de Bases de Datos Relacionales

Ventajas:

- Organización clara y sencilla para ciertas aplicaciones.
- Acceso rápido siguiendo la jerarquía.

Inconvenientes:

- Difícil gestionar relaciones complejas.
- Muy rígido ante cambios en estructura de datos.

Modelo en red:

- Los registros pueden tener múltiples padres y múltiples hijos.
- Mayor flexibilidad para relaciones más complejas.
- Sistema representativo: CODASYL DBTG.

Ventajas:

- Mejor gestión de relaciones complejas.
- Mayor flexibilidad que el modelo jerárquico.

Inconvenientes:

- Gestión más compleja y mantenimiento más difícil.
- Rendimiento variable en grandes bases de datos.



Modelo relacional (1970 hasta la actualidad)

El modelo relacional, propuesto por Edgar F. Codd en 1970, revolucionó completamente la gestión de bases de datos. Este modelo plantea una representación lógica de la información mediante tablas relacionadas entre sí por claves comunes, facilitando enormemente su gestión y consulta.



UF2175: Diseño de Bases de Datos Relacionales

Conceptos clave:

- Datos organizados en tablas (relaciones) con filas (registros) y columnas (atributos).
- Uso del lenguaje SQL (Structured Query Language) para manipular y acceder a la información.
- Implementación de restricciones de integridad para garantizar la validez de los datos.

Ventajas:

- Simplicidad y claridad conceptual.
- Flexibilidad para adaptarse a cambios en las estructuras de datos.
- Alta integridad y coherencia de la información.
- Facilidad de consulta mediante lenguaje SQL.

Ejemplos comunes:

- Oracle Database
- MySQL
- PostgreSQL
- SQL Server

Bases de datos orientadas a objetos y objeto-relacionales (Décadas de 1980 y 1990)

El crecimiento en el uso de aplicaciones multimedia y datos complejos condujo al desarrollo de bases de datos orientadas a objetos, incorporando conceptos como clases, objetos y métodos.

Características principales:

- Gestión de objetos complejos y multimedia.



UF2175: Diseño de Bases de Datos Relacionales

- Integración estrecha con lenguajes orientados a objetos (Java, C++, Python).
- Mayor expresividad y versatilidad en la definición de tipos de datos.

Ventajas:

- Mejor representación de datos complejos.
- Menor brecha semántica entre aplicación y base de datos.

Desventajas:

- Mayor complejidad técnica y operativa.
- Dificultad de adopción generalizada frente al modelo relacional.

Bases de datos NoSQL (Década de 2000 en adelante)

Surgen por la necesidad de almacenar grandes volúmenes de datos (Big Data) en entornos distribuidos. Están diseñadas para mejorar la escalabilidad horizontal y manejar información no estructurada.

Tipos principales:

- Clave-valor (Redis, Amazon DynamoDB)
- Documentales (MongoDB, CouchDB)
- Basadas en grafos (Neo4j)
- Columnas anchas (Apache Cassandra, Apache HBase)

Ventajas:

- Alta escalabilidad horizontal.
- Flexibilidad para datos no estructurados.
- Rendimiento eficiente para grandes volúmenes.



UF2175: Diseño de Bases de Datos Relacionales

Desventajas:

- Menos soporte para transacciones ACID completas.
 - Complejidad en la gestión de la consistencia.
-

Bases de datos en la nube (actualidad)

La evolución más reciente involucra bases de datos gestionadas en la nube, que permiten escalabilidad, alta disponibilidad, copias de seguridad automáticas, y acceso global desde cualquier ubicación.

Ejemplos principales:

- Amazon RDS/Aurora (AWS)
- Azure SQL Database (Microsoft Azure)
- Google Cloud SQL/Firestore/BigQuery (Google Cloud)

Ventajas:

- Escalabilidad automática bajo demanda.
- Reducción en costos iniciales de infraestructura.
- Alta disponibilidad y recuperación ante desastres simplificada.

Desventajas:

- Dependencia del proveedor de servicios en la nube.
 - Posibles problemas de seguridad y privacidad (GDPR y normativas similares).
-



UF2175: Diseño de Bases de Datos Relacionales

Conclusión de la evolución histórica

La evolución histórica de las bases de datos muestra una progresión clara hacia sistemas cada vez más complejos y adaptados a necesidades específicas. Desde simples archivos planos hasta sofisticados sistemas NoSQL y bases de datos gestionadas en la nube, cada paso ha buscado solucionar las limitaciones anteriores y adaptarse a las nuevas exigencias tecnológicas y empresariales.

El modelo relacional sigue siendo predominante en muchos escenarios, pero coexiste con otros modelos, según los requerimientos específicos de escalabilidad, flexibilidad o tipo de datos almacenados.



UF2175: Diseño de Bases de Datos Relacionales

1.2 Ventajas e inconvenientes de las bases de datos

Ventajas:

- **Reducción de redundancia e inconsistencias:** Al almacenar los datos de manera organizada y estructurada se minimizan duplicidades y errores.
- **Integridad y coherencia:** Los sistemas de bases de datos garantizan reglas estrictas sobre los datos almacenados.
- **Seguridad y control de accesos:** Es posible definir permisos específicos, restringiendo el acceso solo a usuarios autorizados.
- **Facilidad de consulta y actualización:** Uso de lenguajes de consulta como SQL para acceder fácilmente a grandes volúmenes de información.
- **Independencia de datos:** Se pueden modificar estructuras de almacenamiento sin afectar aplicaciones existentes.

Inconvenientes:

- **Complejidad y costo inicial:** Requieren inversiones considerables en infraestructura y personal especializado.
 - **Dependencia tecnológica:** Implica una cierta dependencia del sistema gestor utilizado.
 - **Riesgo de fallo generalizado:** Si el sistema central falla, afecta a todos los datos gestionados por la base de datos.
-



UF2175: Diseño de Bases de Datos Relacionales

La utilización de bases de datos para gestionar información presenta numerosas ventajas frente a los sistemas tradicionales, pero también plantea ciertos inconvenientes que deben ser considerados antes de implementar una solución específica. A continuación se detallan cada uno de ellos:

✓ Ventajas de las bases de datos

Las bases de datos ofrecen una serie de beneficios fundamentales en la gestión de la información, incluyendo:

1. Reducción de redundancia e inconsistencias

- **Definición:**
 - Evitan almacenar múltiples copias del mismo dato.
 - Aseguran que cada elemento de información se almacene una sola vez y se referencie cuando sea necesario.
- **Ejemplo práctico:**
 - En una base de datos de clientes, el nombre y dirección del cliente aparecen solo una vez, evitando errores al actualizar datos.
- **Beneficios:**
 - Mayor coherencia en los datos almacenados.
 - Menor espacio necesario para almacenamiento.

2. Integridad y coherencia de los datos

- **Definición:**
 - Garantizan que los datos almacenados cumplen ciertas restricciones o reglas (integridad referencial, integridad de dominio, etc.).
- **Ejemplo práctico:**
 - Una base de datos bancaria asegura que no se puede registrar una transferencia bancaria hacia una cuenta inexistente.



UF2175: Diseño de Bases de Datos Relacionales

- **Beneficios:**

- Los datos siempre serán confiables y exactos.
- Prevención de errores o corrupciones de datos.

3. Seguridad y control de accesos

- **Definición:**

- Los sistemas de bases de datos permiten definir claramente qué usuarios o grupos pueden acceder a qué datos y con qué nivel de acceso (lectura, modificación, eliminación).

- **Ejemplo práctico:**

- Los departamentos financieros tienen acceso a los datos de salarios, mientras que los departamentos de ventas no.

- **Beneficios:**

- Protección eficaz de datos sensibles.
- Reducción de riesgos por accesos no autorizados o errores humanos.

4. Facilidad de consulta y recuperación de información

- **Definición:**

- Las bases de datos facilitan el acceso a grandes volúmenes de datos mediante consultas estructuradas (SQL).

- **Ejemplo práctico:**

- Consultar rápidamente todos los pedidos realizados por un cliente específico en un año determinado.

- **Beneficios:**

- Mayor eficiencia y rapidez en la obtención de información útil.
- Mejor soporte a la toma de decisiones basada en datos.



UF2175: Diseño de Bases de Datos Relacionales

5. Independencia de datos

- **Definición:**
 - Permite modificar la estructura interna de almacenamiento sin afectar a las aplicaciones externas que usan estos datos.
- **Ejemplo práctico:**
 - Cambiar un disco físico por otro de mayor capacidad sin afectar las consultas existentes.
- **Beneficios:**
 - Mayor flexibilidad y facilidad para adaptarse a cambios tecnológicos.
 - Reducción de costos en mantenimiento de aplicaciones.

6. Acceso concurrente

- **Definición:**
 - Permite que múltiples usuarios o aplicaciones accedan simultáneamente a los mismos datos de manera controlada y segura.
 - **Ejemplo práctico:**
 - Varios empleados acceden simultáneamente a inventarios actualizados en tiempo real.
 - **Beneficios:**
 - Mejora significativa en productividad y eficiencia operativa.
 - Información siempre actualizada y disponible.
-



UF2175: Diseño de Bases de Datos Relacionales

⚠ Inconvenientes de las bases de datos

Aunque ofrecen muchas ventajas, las bases de datos presentan algunos retos que conviene conocer antes de su implementación:

1. Complejidad y coste inicial elevado

- **Definición:**
 - Implementar un sistema de base de datos requiere de recursos técnicos, económicos y humanos importantes en su fase inicial.
- **Ejemplo práctico:**
 - Inversión en servidores potentes, licencias de software y formación del personal especializado.
- **Problemas derivados:**
 - Coste significativo de adquisición inicial.
 - Complejidad técnica inicial alta.

2. Dependencia tecnológica

- **Definición:**
 - Una vez que se adopta un sistema gestor de bases de datos específico, migrar a otro sistema es complejo y costoso.
- **Ejemplo práctico:**
 - Una empresa que usa Oracle podría enfrentar dificultades para migrar a MySQL.
- **Problemas derivados:**
 - Limitación en opciones tecnológicas futuras.
 - Altos costes de migración en caso de cambios estratégicos.



UF2175: Diseño de Bases de Datos Relacionales

3. Riesgo de fallos generalizados

- **Definición:**
 - Un fallo en el sistema de base de datos puede afectar todos los datos gestionados, especialmente en sistemas centralizados.
- **Ejemplo práctico:**
 - Un error de hardware o software puede provocar indisponibilidad completa del sistema.
- **Problemas derivados:**
 - Potencial pérdida masiva de datos o indisponibilidad.
 - Necesidad de sistemas complejos de recuperación de desastres.

4. Rendimiento afectado por concurrencia

- **Definición:**
 - Muchos usuarios concurrentes pueden afectar el rendimiento y la velocidad del sistema si no se gestionan correctamente las transacciones y bloqueos.
- **Ejemplo práctico:**
 - En horas pico, un servidor con muchos usuarios puede experimentar ralentizaciones.
- **Problemas derivados:**
 - Potencial disminución de rendimiento en operaciones críticas.
 - Necesidad de infraestructura potente y optimización avanzada.

5. Requerimiento de personal altamente especializado

- **Definición:**
 - Las bases de datos requieren técnicos cualificados y especialistas en gestión, administración y mantenimiento.



UF2175: Diseño de Bases de Datos Relacionales

- **Ejemplo práctico:**
 - La contratación y formación continua de administradores de bases de datos (DBA).
 - **Problemas derivados:**
 - Dificultad para encontrar personal cualificado.
 - Alto coste asociado a sueldos y formación especializada.
-

Conclusión

Las bases de datos proporcionan beneficios significativos en la organización, integridad, seguridad y acceso a los datos, aportando una ventaja competitiva en muchas organizaciones. Sin embargo, para maximizar estas ventajas es fundamental entender y gestionar adecuadamente sus desventajas, como el coste inicial, complejidad técnica y dependencia tecnológica. El balance entre ventajas e inconvenientes será determinante para seleccionar adecuadamente la solución tecnológica más apropiada según las necesidades específicas del proyecto u organización.



UF2175: Diseño de Bases de Datos Relacionales

1.3 Conceptos generales

Concepto de base de datos:

- Una base de datos es un conjunto organizado de datos relacionados que se almacenan electrónicamente, permitiendo múltiples usos por distintos usuarios o aplicaciones simultáneamente.

Objetivos principales de los sistemas de bases de datos:

- Minimizar redundancias y asegurar coherencia.
- Facilitar acceso rápido y eficiente.
- Mantener integridad y validez de la información.

Problemas comunes en bases de datos:

- **Redundancia e inconsistencia de datos:** Almacenar datos repetidos provoca posibles contradicciones y gasto innecesario de recursos.
 - **Aislamiento de datos:** Dificultad de acceder a información dispersa en diferentes sistemas o archivos.
 - **Anomalías en acceso concurrente:** Pueden surgir errores cuando múltiples usuarios acceden simultáneamente; estos se manejan con sistemas de transacciones y bloqueos.
 - **Problemas de seguridad:** Vulnerabilidad a accesos no autorizados; se combaten mediante protocolos robustos de autenticación.
 - **Problemas de integridad:** Riesgo de almacenar datos incorrectos; gestionado con restricciones estrictas sobre qué datos se pueden almacenar.
-



UF2175: Diseño de Bases de Datos Relacionales

Para entender cómo funcionan las bases de datos es necesario conocer una serie de conceptos fundamentales, los cuales permitirán profundizar más adelante en aspectos técnicos y prácticos. Los principales conceptos son los siguientes:

1.3.1 Definición de base de datos

Una **base de datos** es una colección estructurada y organizada de datos relacionados, almacenada electrónicamente en un sistema informático. Su objetivo es facilitar el almacenamiento, la recuperación, la gestión y el análisis eficiente de la información.

Características esenciales:

- Datos organizados según un modelo lógico definido.
 - Relación lógica entre los datos almacenados.
 - Uso compartido y simultáneo de la información por diferentes usuarios o aplicaciones.
 - Acceso controlado y seguro mediante un sistema gestor de bases de datos (DBMS).
-

1.3.2 Sistema Gestor de Bases de Datos (DBMS)

Un **Sistema Gestor de Bases de Datos** (Database Management System, DBMS) es un software especializado que permite crear, mantener, gestionar y controlar el acceso a una base de datos.

Principales funciones del DBMS:

- **Gestión del almacenamiento:** Almacenar y recuperar datos desde los dispositivos físicos (discos, memorias).
- **Control de acceso y seguridad:** Autenticación de usuarios, permisos y roles.
- **Integridad de datos:** Aplicación de restricciones que aseguran la validez y exactitud de los datos.
- **Gestión de transacciones:** Manejo seguro de operaciones simultáneas (conurrencia).



UF2175: Diseño de Bases de Datos Relacionales

- **Recuperación ante fallos:** Sistemas de respaldo y restauración en caso de fallos del sistema.
- **Optimización del rendimiento:** Técnicas para mejorar la rapidez y eficiencia en la ejecución de consultas.

Ejemplos populares de DBMS:

- MySQL
 - PostgreSQL
 - Oracle Database
 - Microsoft SQL Server MongoDB (NoSQL)
-

1.3.3 Componentes fundamentales de una base de datos

Los componentes esenciales que conforman una base de datos incluyen:

Tablas

- Estructuras fundamentales donde se almacenan los datos organizados en filas (registros) y columnas (campos o atributos).

Registros

- Conjunto específico de datos relacionados almacenados en una tabla, representados por filas.

Campos

- Atributos específicos dentro de un registro que definen características de la información, representados por columnas.

Relaciones

- Conexiones lógicas entre tablas que permiten vincular información distribuida.



UF2175: Diseño de Bases de Datos Relacionales

Claves

- Campos específicos usados para identificar o relacionar registros (claves primarias y foráneas).

Índices

- Estructuras internas que aceleran las consultas en bases de datos.



1.3.4 Problemas gestionados por las bases de datos

Las bases de datos surgen como respuesta a problemas concretos relacionados con el manejo de información, tales como:

Redundancia e inconsistencia de datos

- Problema: almacenamiento duplicado que genera discrepancias.
- Solución: estructura organizada que minimiza la repetición innecesaria de información.

Aislamiento de los datos

- Problema: dificultad de acceder o integrar datos almacenados en múltiples ubicaciones o formatos.
- Solución: almacenamiento centralizado y estandarizado.

Anomalías derivadas del acceso concurrente

- Problema: conflictos y errores al acceder simultáneamente a los mismos datos.
- Solución: mecanismos de control de transacciones y bloqueos.

Problemas de seguridad



UF2175: Diseño de Bases de Datos Relacionales

- Problema: accesos no autorizados, pérdida o alteración indebida de información.
- Solución: sistemas robustos de autenticación y permisos.

Problemas de integridad

- Problema: inserción o modificación de datos erróneos o inválidos.
- Solución: reglas estrictas de validación e integridad referencial.

1.3.5 Arquitectura ANSI/SPARC

La arquitectura ANSI/SPARC es una estructura estándar que define tres niveles en una base de datos, ofreciendo independencia entre estos niveles:

Nivel interno o físico

- Descripción: Almacenamiento físico real (cómo están almacenados los datos en discos o medios).
- Ejemplo: organización en bloques de almacenamiento físico.

Nivel conceptual

- Descripción: Representa la estructura lógica global de la base de datos, independientemente de cómo se almacene físicamente.
- Ejemplo: estructura de tablas, relaciones, claves primarias y foráneas.

Nivel externo o de vistas

- Descripción: Representación específica adaptada a usuarios o aplicaciones particulares (diferentes usuarios pueden tener diferentes vistas según sus necesidades).
- Ejemplo: interfaz personalizada de consulta para el departamento de ventas o contabilidad.

1.3.6 Independencia de datos



UF2175: Diseño de Bases de Datos Relacionales

Un concepto fundamental en bases de datos es la independencia de datos, que permite modificar estructuras internas sin afectar aplicaciones externas:

Independencia lógica

- Cambios en el esquema conceptual (estructura lógica) sin afectar a las vistas externas de los usuarios.
- Ejemplo: añadir una columna nueva a una tabla sin afectar consultas existentes.

Independencia física

- Cambios en almacenamiento físico sin afectar la estructura conceptual.
- Ejemplo: mover datos desde un disco duro convencional a un SSD, sin que usuarios o aplicaciones perciban cambios.

1.3.7 Lenguajes de manejo de bases de datos

Los sistemas de bases de datos utilizan diferentes lenguajes especializados para interactuar con la información:

Lenguaje de Definición de Datos (DDL)

- Define y modifica estructuras de almacenamiento (tablas, índices, vistas).
- Ejemplos de comandos: `CREATE TABLE`, `ALTER TABLE`, `DROP INDEX`.

Lenguaje de Manipulación de Datos (DML)

- Permite realizar operaciones sobre datos específicos almacenados.
- Ejemplos de comandos: `SELECT`, `INSERT`, `UPDATE`, `DELETE`.

Lenguaje de Control de Datos (DCL)

- Gestiona accesos, permisos y seguridad.
 - Ejemplos de comandos: `GRANT`, `REVOKE`.
-



UF2175: Diseño de Bases de Datos Relacionales

Conclusión

Los conceptos generales aquí detallados permiten una comprensión integral del funcionamiento y gestión de bases de datos. Son esenciales para entender posteriormente los métodos, técnicas y herramientas específicas para diseñar y administrar bases de datos relacionales, objeto-relacionales o NoSQL.

El conocimiento claro de estos conceptos es fundamental para aprovechar eficazmente las ventajas ofrecidas por los sistemas de bases de datos, permitiendo un manejo eficiente, seguro y consistente de la información almacenada.

1.4 Administración de los datos y administración de bases de datos

- **Administración de datos:** Gestión lógica, estratégica y organizacional de la información (políticas, estructuras, estándares).



UF2175: Diseño de Bases de Datos Relacionales

- **Administrador de Bases de Datos (DBA):** Responsable de la gestión técnica, configuración, seguridad, copias de seguridad, rendimiento y mantenimiento.
-

En el ámbito de las bases de datos, se distinguen claramente dos funciones esenciales: la **Administración de Datos** y la **Administración de Bases de Datos**. Aunque ambas están estrechamente relacionadas, tienen enfoques, responsabilidades y objetivos específicos diferentes. A continuación se explican con detalle ambos conceptos.

1.4.1 Administración de los Datos (Data Administration)

La **Administración de Datos** se centra en los aspectos estratégicos y organizativos de la gestión de la información dentro de una organización. Es responsable de establecer políticas, estándares y procedimientos para asegurar que la información almacenada cumpla con las necesidades y objetivos estratégicos de la organización.

Funciones principales:

- **Planificación estratégica de datos**
 - Determina qué información debe recopilarse y almacenarse según objetivos estratégicos.
 - Establece las prioridades y planes de gestión de la información.
- **Definición de estándares y procedimientos**
 - Establece los formatos estándar en los que deben almacenarse los datos.
 - Define procedimientos para la entrada, actualización y eliminación de información.
- **Análisis de necesidades de datos**
 - Evalúa constantemente los requerimientos informativos de diferentes departamentos o usuarios.
 - Asegura que la base de datos cumpla con estos requerimientos.



UF2175: Diseño de Bases de Datos Relacionales

- **Gestión de la calidad de los datos**
 - Supervisa que los datos almacenados sean precisos, actualizados, coherentes y relevantes.
 - Establece protocolos para garantizar calidad (validaciones, auditorías).
- **Gestión de metadatos**
 - Define y mantiene la estructura de información sobre los datos (diccionario de datos).
 - Proporciona documentación detallada de los datos almacenados, como tipos, significados y relaciones.
- **Coordinación y comunicación interna**
 - Actúa como enlace entre departamentos técnicos, administrativos y usuarios finales.
 - Promueve la concienciación y buenas prácticas en el manejo de información.

Perfil profesional típico:

- Analistas de datos.
 - Gestores de datos (Data managers).
 - Responsables o directores de información.
-

1.4.2 Administración de Bases de Datos (Database Administration – DBA)

La **Administración de Bases de Datos** es una función técnica específica, encargada del manejo, control, protección y mantenimiento efectivo de los sistemas de bases de datos. La figura central aquí es el DBA (Administrador de Base de Datos).



UF2175: Diseño de Bases de Datos Relacionales

Funciones principales:

- **Diseño físico y lógico de la base de datos**
 - Implementa estructuras físicas (tablas, índices, vistas).
 - Diseña y optimiza la organización lógica para mejorar rendimiento y almacenamiento.
- **Instalación y configuración del DBMS**
 - Selecciona, instala y configura el software de bases de datos (Oracle, MySQL, PostgreSQL, SQL Server, etc.).
 - Realiza actualizaciones de software y parches de seguridad regularmente.
- **Gestión de la seguridad y control de accesos**
 - Define roles de usuario, permisos y niveles de acceso específicos.
 - Supervisa accesos no autorizados o intentos de brechas de seguridad.
- **Monitorización y optimización del rendimiento**
 - Controla y monitorea el rendimiento del sistema (consultas lentas, sobrecargas).
 - Realiza ajustes constantes para mejorar la eficiencia de operaciones.
- **Administración de copias de seguridad y recuperación**
 - Implementa políticas de backup y recuperación (copias completas, incrementales).
 - Realiza pruebas frecuentes de restauración de datos para garantizar la recuperación efectiva.
- **Gestión del almacenamiento físico**
 - Administra el espacio físico disponible (espacio en discos, particiones, volúmenes).
 - Planifica ampliaciones o cambios necesarios según crecimiento de datos.
- **Resolución de incidencias técnicas**



UF2175: Diseño de Bases de Datos Relacionales

- Atiende fallos técnicos, errores en bases de datos o problemas de disponibilidad.
- Asegura que cualquier incidente se resuelva rápidamente, minimizando tiempos de caída.

Perfil profesional típico:

- Administrador de bases de datos (DBA).
 - Especialista en soporte y operaciones.
 - Ingeniero/a de sistemas especializado en bases de datos.
-

1.4.3 Diferencias entre Administración de Datos y Administración de Bases de Datos

Aspecto	Administración de Datos	Administración de Bases de Datos (DBA)
---------	-------------------------	--



UF2175: Diseño de Bases de Datos Relacionales

Enfoque	Estratégico-organizacional	Técnico-operativo
Responsabilidades clave	Definir políticas, estándares, calidad y estrategias de información	Gestión técnica, seguridad, rendimiento, disponibilidad
Tipo de actividades	Planificación, análisis, coordinación interna, documentación	Diseño, configuración, monitorización, backups, resolución de incidencias
Perfil profesional habitual	Gestor/a o analista de información, estrategia de datos	Administrador/a técnico/a de bases de datos, Ingeniero/a DBA
Relación con usuarios	Enlace directo con usuarios finales y alta dirección	Enlace directo con departamentos técnicos (desarrollo, soporte)

1.4.4 Importancia de ambas administraciones en una organización

La existencia de ambas funciones es fundamental para el éxito de la gestión de la información en una organización:

- **Administración de datos** asegura que la información almacenada cumpla con objetivos estratégicos, sea consistente y de alta calidad, alineándose con los



UF2175: Diseño de Bases de Datos Relacionales

objetivos de negocio.

- **Administración de bases de datos** asegura la estabilidad técnica, rendimiento y seguridad necesaria para que los datos sean accesibles en todo momento y de forma confiable.

Ambas administraciones trabajan coordinadamente para garantizar que la organización disponga de información precisa, segura, accesible y útil para la toma de decisiones efectiva.

Conclusión

La Administración de Datos y la Administración de Bases de Datos son funciones distintas, pero complementarias y esenciales en cualquier entorno que maneje grandes volúmenes de información. La combinación efectiva de ambas funciones asegura la disponibilidad constante de información estratégica, de calidad y técnicamente estable, lo que permite a las organizaciones alcanzar ventajas competitivas y tomar decisiones acertadas basadas en datos confiables y oportunos.

1.5 Niveles de arquitectura de bases de datos

- **Nivel interno (físico):** Organización y estructura física del almacenamiento real de los datos en hardware.
- **Nivel conceptual (lógico):** Representación abstracta del sistema, donde se describen estructuras de datos sin detalles físicos específicos.



UF2175: Diseño de Bases de Datos Relacionales

- **Nivel externo (usuario):** Vistas personalizadas adaptadas a las necesidades particulares de cada usuario o aplicación.

La arquitectura de bases de datos define la forma en que los datos son estructurados, accedidos y gestionados dentro de un sistema de bases de datos. La **arquitectura de tres niveles**, también conocida como **modelo ANSI/SPARC**, es un estándar ampliamente aceptado que proporciona separación entre la forma en que los datos son físicamente almacenados, cómo están organizados lógicamente y cómo los usuarios interactúan con ellos.

Esta estructura tiene como objetivo principal **garantizar la independencia de los datos**, facilitando cambios y mantenimiento sin afectar al sistema global.

Los tres niveles de la arquitectura

1. Nivel interno (nivel físico)

Definición:

Representa la forma en la que los datos están realmente almacenados en los dispositivos físicos como discos duros, SSDs o almacenamiento en red.

Elementos que gestiona:

- Estructuras de almacenamiento (bloques, registros, páginas).
- Organización física de archivos.
- Compresión de datos.
- Indexación a bajo nivel.
- Fragmentación y agrupamiento.

Funciones:

- Optimizar el rendimiento del sistema mediante la organización eficiente del almacenamiento.
- Gestionar los detalles del hardware (espacio libre, buffers, páginas).



UF2175: Diseño de Bases de Datos Relacionales

- Proporcionar seguridad física y lógica sobre los datos almacenados.

Ejemplo práctico:

El sistema guarda los datos de una tabla de clientes físicamente divididos en bloques de 8 KB, comprimidos y distribuidos en diferentes sectores del disco.

2. Nivel conceptual (nivel lógico)

Definición:

Describe la estructura lógica completa de la base de datos para toda la organización. No depende de cómo están almacenados físicamente ni de cómo los usuarios individuales ven los datos.

Elementos que gestiona:

- Tablas (relaciones), atributos y dominios.
- Relaciones entre entidades (claves primarias y foráneas).
- Restricciones de integridad y validación.
- Vistas lógicas del modelo de datos.

Funciones:

- Definir los elementos estructurales y relaciones que componen la base de datos.
- Establecer reglas de integridad de los datos.
- Servir como puente entre el almacenamiento físico y las vistas de usuario.

Ejemplo práctico:

Una base de datos contiene una tabla "clientes" con campos como `id_cliente`, `nombre`, `email`, `fecha_alta`, y otra tabla "pedidos" relacionada mediante una clave foránea `id_cliente`.



UF2175: Diseño de Bases de Datos Relacionales

3. Nivel externo (nivel de vistas)

Definición:

Corresponde a la manera en que cada usuario o grupo de usuarios ve e interactúa con los datos. Cada vista puede mostrar solo una parte específica del esquema conceptual.

Elementos que gestiona:

- Vistas personalizadas para usuarios o aplicaciones.
- Restricciones de visibilidad y acceso a datos.
- Interfaces de consulta adaptadas a cada perfil.

Funciones:

- Adaptar la información a las necesidades de cada usuario.
- Proteger datos sensibles mediante ocultación selectiva.
- Simplificar la interacción con la base de datos.

Ejemplo práctico:

Un usuario de atención al cliente puede ver solo los campos **nombre**, **email** y **estado** de la tabla de clientes, mientras que un administrador puede ver todos los datos, incluidos los financieros.

Objetivos clave del modelo de tres niveles

- **Separación de responsabilidades:** Cada nivel se puede modificar sin afectar directamente a los demás.



UF2175: Diseño de Bases de Datos Relacionales

- **Independencia física y lógica de los datos:** Permite cambios en la estructura física o lógica sin afectar las aplicaciones.
- **Facilidad de mantenimiento y evolución:** Aislado los cambios, se facilita la adaptación del sistema a nuevos requisitos.
- **Mejora de la seguridad:** El nivel externo limita lo que cada usuario puede ver o modificar.

Relación entre los niveles

plaintext

CopiarEditar

[Usuario / Aplicación]



Nivel Externo (vistas individuales o personalizadas)



Nivel Conceptual (estructura global de la base de datos)



Nivel Interno (organización física en almacenamiento)

Cada nivel se comunica solo con el nivel inmediato, lo cual permite modificaciones aisladas sin afectar los otros niveles directamente.

Ejemplo completo de los tres niveles

Imaginemos una base de datos de una tienda online:

- **Nivel interno:** Los registros de productos están almacenados físicamente en archivos binarios distribuidos en múltiples discos.



UF2175: Diseño de Bases de Datos Relacionales

- **Nivel conceptual:** Existe una tabla `productos` con campos como `id_producto`, `nombre`, `precio`, `stock` y `id_categoria`, que se relaciona con la tabla `categorías`.
- **Nivel externo:**
 - El usuario cliente ve solo `nombre`, `precio` y `disponibilidad`.
 - El empleado del almacén ve `stock`, `ubicación` y `fecha de reposición`.
 - El gerente ve informes agregados con estadísticas de ventas.



Conclusión

La arquitectura de tres niveles en bases de datos es fundamental para separar la estructura lógica, física y la interacción de los usuarios con la información. Esta separación facilita el mantenimiento, la seguridad y la adaptabilidad del sistema, permitiendo que las organizaciones manejen información compleja de forma más eficiente, segura y escalable.

Implementar correctamente este modelo en cualquier sistema de bases de datos es un paso esencial para garantizar su robustez y longevidad.

1.6 Independencia de los datos



UF2175: Diseño de Bases de Datos Relacionales

- **Independencia lógica:** Capacidad para cambiar el esquema conceptual sin afectar a los usuarios o aplicaciones.
- **Independencia física:** Modificaciones en almacenamiento físico que no afectan la estructura lógica de datos.

La **independencia de los datos** es uno de los principios más importantes en el diseño de bases de datos. Se refiere a la capacidad del sistema para permitir cambios en la estructura de la base de datos sin que estos afecten a las aplicaciones que la utilizan ni a los usuarios finales.

Este principio está directamente relacionado con la **arquitectura de tres niveles** (interno, conceptual y externo), y garantiza que las modificaciones en un nivel no alteren los otros, preservando la funcionalidad y evitando reescribir el software o las consultas.



¿Por qué es importante la independencia de los datos?

- Facilita la **evolución** del sistema de información sin afectar a los usuarios o aplicaciones.
- Reduce los **costes de mantenimiento**, ya que permite modificaciones estructurales sin necesidad de rehacer todo el sistema.
- Aumenta la **flexibilidad**, ya que se pueden introducir mejoras o cambios tecnológicos sin interrumpir el servicio.
- Mejora la **portabilidad** de datos entre diferentes plataformas o sistemas.



Tipos de independencia de los datos

1 Independencia física de los datos



Definición:

La **independencia física** permite realizar cambios en el **almacenamiento interno** de los datos sin afectar al **modelo lógico** (conceptual).



Ejemplos de cambios permitidos:



UF2175: Diseño de Bases de Datos Relacionales

- Cambiar el tipo de dispositivo de almacenamiento (de HDD a SSD, por ejemplo).
- Reorganizar físicamente los archivos de datos en disco.
- Implementar nuevos métodos de compresión o indexación.

Ejemplo práctico:

Un administrador cambia la forma en que se almacenan las tablas en el disco (por bloques de diferente tamaño), pero los usuarios siguen accediendo a los datos de la misma manera a través de sus aplicaciones o consultas.

Independencia lógica de los datos

Definición:

La **independencia lógica** permite modificar el **esquema conceptual** (tablas, campos, relaciones) sin afectar las **vistas externas** de los usuarios ni las aplicaciones que utilizan la base de datos.

Ejemplos de cambios permitidos:

- Añadir o eliminar columnas en una tabla.
- Crear nuevas relaciones entre tablas.
- Renombrar elementos internos sin alterar las vistas externas.





Ejemplo práctico:

Se añade un nuevo campo "fecha de actualización" a la tabla **clientes**. Las vistas externas que no utilizan ese campo siguen funcionando sin necesidad de modificación.

Beneficios clave de aplicar independencia de los datos



UF2175: Diseño de Bases de Datos Relacionales

Beneficio	Descripción
 Mantenimiento flexible	Cambios estructurales sin impacto en el funcionamiento de las aplicaciones.
 Modularidad del sistema	Los componentes de la base de datos están desacoplados y son fácilmente modificables.
 Adaptabilidad tecnológica	Permite adaptar la base de datos a nuevas tecnologías de hardware o software sin alterar el acceso.
 Estabilidad para los usuarios	Los usuarios finales no experimentan interrupciones ni errores por cambios técnicos.

Relación con la arquitectura de tres niveles

La independencia de los datos se consigue gracias a la separación de niveles definida por el modelo ANSI/SPARC:

Tipo de independencia	Relación entre niveles afectada
Física	Interno ↔ Conceptual
Lógica	Conceptual ↔ Externo

Esta arquitectura garantiza que los cambios en un nivel (almacenamiento físico o modelo lógico) no alteran los demás niveles.



UF2175: Diseño de Bases de Datos Relacionales

Riesgos de no aplicar independencia de los datos

- Las aplicaciones pueden volverse **dependientes** de estructuras internas.
 - Cada cambio en la base de datos obligaría a **reescribir software** o **consultas SQL**.
 - Mayor **riesgo de errores** por inconsistencias entre niveles.
 - Aumento de los **costes de mantenimiento y desarrollo**.
-

Conclusión

La independencia de los datos es una característica esencial de los sistemas modernos de bases de datos, que permite mantener la **coherencia**, **estabilidad** y **adaptabilidad** de la información. Al separar claramente los niveles de almacenamiento, estructura lógica y vistas de usuario, se facilita la evolución de los sistemas sin afectar al funcionamiento ni a los usuarios finales.

Aplicar este principio correctamente asegura que las bases de datos puedan crecer, cambiar y adaptarse a nuevas necesidades sin comprometer la integridad ni la operatividad del sistema.

1.7 Lenguajes de manejo de bases de datos



UF2175: Diseño de Bases de Datos Relacionales

- **DDL (Lenguaje de Definición de Datos):** Define y modifica estructuras de datos (crear, modificar, eliminar tablas e índices).
- **DML (Lenguaje de Manipulación de Datos):** Inserción, actualización, eliminación y consulta de datos.
- **DCL (Lenguaje de Control de Datos):** Define permisos, usuarios y roles para controlar accesos.

Los sistemas gestores de bases de datos (SGBD o DBMS) utilizan **lenguajes específicos** para interactuar con la base de datos, definir su estructura, manipular datos, y controlar la seguridad y las transacciones. Estos lenguajes permiten desde la creación de una tabla hasta la asignación de permisos o la consulta de información.

El conjunto de estos lenguajes constituye el **Lenguaje de Base de Datos**, y se agrupan generalmente en tres categorías:

1.7.1 Lenguaje de Definición de Datos (DDL - Data Definition Language)

Definición:

Permite crear, modificar y eliminar las estructuras que conforman la base de datos, como tablas, índices, vistas, esquemas, etc.



UF2175: Diseño de Bases de Datos Relacionales

Instrucciones principales:

Comando	Función
CREATE	Crea una nueva tabla, índice, vista u otro objeto
ALTER	Modifica la estructura de una tabla existente (añadir, eliminar o modificar columnas)
DROP	Elimina una tabla, índice o vista
TRUNCATE	Borra todos los registros de una tabla sin eliminar su estructura

Ejemplo:

```
CREATE TABLE empleados (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(100),  
    salario DECIMAL(10,2)  
);
```

```
ALTER TABLE empleados ADD fecha_ingreso DATE;
```

```
DROP TABLE empleados;
```



UF2175: Diseño de Bases de Datos Relacionales

1.7.2 Lenguaje de Manipulación de Datos (DML - Data Manipulation Language)

Definición:

Permite realizar operaciones sobre los datos almacenados en las tablas. Con este lenguaje se puede consultar, insertar, modificar o eliminar datos.

Instrucciones principales:

Comando	Función
<code>SELECT</code>	Recupera datos desde una o más tablas
<code>INSERT</code>	Inserta nuevos registros en una tabla
<code>UPDATE</code>	Modifica valores existentes
<code>DELETE</code>	Elimina registros específicos

Ejemplo:

```
INSERT INTO empleados (id, nombre, salario)
VALUES (1, 'Ana Pérez', 2500.00);

UPDATE empleados
SET salario = 2700.00
WHERE id = 1;
```



UF2175: Diseño de Bases de Datos Relacionales

```
DELETE FROM empleados
```

```
WHERE id = 1;
```

```
SELECT nombre, salario FROM empleados;
```

1.7.3 Lenguaje de Control de Datos (DCL - Data Control Language)

Definición:

Permite gestionar la seguridad de la base de datos, definiendo los permisos y privilegios de los usuarios sobre objetos específicos.

Instrucciones principales:

Comando	Función
GRANT	Otorga privilegios a un usuario
REVOKE	Revoca privilegios otorgados previamente

Ejemplo:

```
GRANT SELECT, INSERT ON empleados TO usuario_ventas;
```

```
REVOKE INSERT ON empleados FROM usuario_ventas;
```



UF2175: Diseño de Bases de Datos Relacionales

1.7.4 Lenguaje de Control de Transacciones (TCL - Transaction Control Language)

Definición:

Permite gestionar las transacciones dentro de una base de datos. Las transacciones son bloques de instrucciones que deben ejecutarse como una unidad atómica (todo o nada).

Instrucciones principales:

Comando	Función
<code>COMMIT</code>	Confirma los cambios realizados por una transacción
<code>ROLLBACK</code>	Deshace los cambios realizados hasta el último <code>COMMIT</code>
<code>SAVEPOINT</code>	Crea un punto de restauración dentro de una transacción

Ejemplo:

```
BEGIN;
```

```
UPDATE empleados SET salario = salario + 100 WHERE id = 2;
```

```
SAVEPOINT aumento_salario;
```

```
-- Si ocurre un error:
```

```
ROLLBACK TO aumento_salario;
```



UF2175: Diseño de Bases de Datos Relacionales

-- Si todo está bien:

COMMIT;

Relación entre los lenguajes

Lenguaje	Propósito principal
DDL	Definir la estructura de la base de datos
DML	Manipular los datos contenidos
DCL	Controlar el acceso y los permisos
TCL	Gestionar la coherencia en las transacciones

Conclusión

El conocimiento y uso adecuado de los diferentes lenguajes de manejo de bases de datos es esencial para el diseño, desarrollo y administración de sistemas eficientes, seguros y consistentes. Estos lenguajes no solo permiten definir y trabajar con los datos, sino también asegurar su integridad, protegerlos frente a accesos no autorizados y garantizar que las transacciones se ejecuten de forma fiable. En cualquier entorno profesional donde se manejen datos, dominar estos lenguajes es una habilidad clave para garantizar el éxito de los sistemas de información.

1.8 Sistema de Gestión de la Base de Datos (DBMS)



UF2175: Diseño de Bases de Datos Relacionales

- Software que gestiona almacenamiento, recuperación y seguridad de la información.
- Responsabilidades clave incluyen gestión de transacciones, control de acceso, recuperación ante fallos y optimización de consultas.

Un **Sistema de Gestión de Bases de Datos** (SGBD o DBMS, por sus siglas en inglés: *Database Management System*) es el componente central encargado de facilitar la creación, uso, mantenimiento y administración de una base de datos. Es el software que actúa como intermediario entre los datos almacenados, las aplicaciones y los usuarios.

1.8.1 Funciones principales del DBMS

Los DBMS modernos realizan múltiples funciones críticas, incluyendo:

1. Gestión del almacenamiento de datos

- Organiza físicamente los datos en estructuras de almacenamiento eficientes.
- Administra archivos, páginas de datos, índices y buffers.
- Optimiza la localización y recuperación de registros.

2. Seguridad y control de acceso

- Permite definir **usuarios, roles y permisos**.
- Asegura que solo los usuarios autorizados puedan acceder, modificar o consultar datos sensibles.
- Registra auditorías de acceso y uso.

3. Gestión de la integridad de los datos

- Aplica restricciones como:
 - **Claves primarias** (unicidad).
 - **Claves foráneas** (referencias válidas).
 - **Restricciones de dominio** (tipo, rango de valores).



UF2175: Diseño de Bases de Datos Relacionales

- Impide la entrada de datos erróneos o incoherentes.

4. Control de concurrencia

- Permite el acceso simultáneo de múltiples usuarios a la base de datos de forma segura.
- Utiliza bloqueos (*locks*), mecanismos de serialización y aislamiento de transacciones.
- Previene condiciones como **lecturas sucias**, **lecturas no repetibles** o **interferencias**.

5. Gestión de transacciones

- Garantiza que una serie de operaciones se ejecuten de forma **atómica** (todas o ninguna).
- Implementa el modelo **ACID**:
 - **A**tomicidad.
 - **C**onsistencia.
 - **I**solación.
 - **D**urabilidad.
- Posibilidad de hacer **COMMIT** (confirmar) o **ROLLBACK** (revertir).

6. Recuperación ante fallos

- Permite restaurar la base de datos a un estado coherente tras errores de sistema, caídas de energía o corrupción de archivos.
- Utiliza técnicas como:
 - Registros de transacciones (*log*).
 - Copias de seguridad (*backups*).
 - Puntos de recuperación (*restore points*).



UF2175: Diseño de Bases de Datos Relacionales

7. Optimización y procesamiento de consultas

- Interpreta las consultas SQL del usuario.
- Elige automáticamente la mejor estrategia de búsqueda (uso de índices, escaneo secuencial, etc.).
- Proporciona estadísticas para evaluar rendimiento.

8. Interfaces y APIs de acceso

- Proporciona medios para conectarse desde:
 - Aplicaciones (mediante ODBC, JDBC, etc.).
 - Interfaces gráficas de usuario.
 - Consolas de administración.
 - Herramientas externas de análisis.
-

1.8.2 Tipos de DBMS

Relacionales (RDBMS)

- Basados en el modelo de tablas relacionadas (relacional).
- Uso del lenguaje SQL.
- Ejemplos: MySQL, PostgreSQL, Oracle, SQL Server.

NoSQL

- Diseñados para datos no estructurados, escalabilidad horizontal y altas velocidades.
- Tipos: clave-valor, documentos, grafos, columnares.
- Ejemplos: MongoDB, Redis, Cassandra, Neo4j.








UF2175: Diseño de Bases de Datos Relacionales

En la nube (Cloud DBMS)

- Servicios gestionados por terceros, accesibles vía internet.
- Permiten escalabilidad automática, alta disponibilidad y facturación según uso.
- Ejemplos: Amazon Aurora, Google Cloud SQL, Azure SQL Database.





1.8.3 Ventajas del uso de un DBMS

Ventaja	Descripción
 Seguridad	Gestión robusta de accesos y privacidad de datos.
 Concurrency controlada	Acceso simultáneo sin conflictos.
 Integridad y fiabilidad	Mantenimiento de reglas y coherencia de datos.
 Recuperación ante fallos	Sistema preparado para restauración en caso de error o pérdida.
 Mantenimiento y evolución	Facilidad para adaptar y modificar la base de datos sin afectar al sistema.

1.8.4 Limitaciones de un DBMS



UF2175: Diseño de Bases de Datos Relacionales

Limitación	Descripción
 Coste	Algunos sistemas DBMS son de licencia costosa.
 Curva de aprendizaje	Requieren conocimientos técnicos avanzados para su administración.
 Recursos	Pueden consumir mucha memoria y procesamiento.
 Dependencia tecnológica	Migrar entre DBMS puede resultar costoso y complejo.

1.8.5 Ejemplo práctico de interacción con un DBMS

Un usuario crea una tabla, inserta datos y realiza una consulta:

sql

CopiarEditar

```
CREATE TABLE productos (  
    id INT PRIMARY KEY,  
    nombre VARCHAR(100),  
    precio DECIMAL(10,2)  
);
```

```
INSERT INTO productos (id, nombre, precio)  
VALUES (1, 'Ratón inalámbrico', 19.95);
```



UF2175: Diseño de Bases de Datos Relacionales

```
SELECT nombre, precio FROM productos WHERE id = 1;
```

Todo esto es gestionado y optimizado por el DBMS en segundo plano: controla la integridad de los datos, gestiona el acceso, asegura que la transacción sea atómica y optimiza la consulta para devolver resultados de forma eficiente.

Conclusión

El **DBMS** es el motor que hace posible trabajar con bases de datos de forma estructurada, segura y eficiente. Su función va mucho más allá del simple almacenamiento: controla todo el ciclo de vida de los datos, desde su definición hasta su recuperación, garantizando la integridad, seguridad y rendimiento. Elegir el DBMS adecuado y conocer sus capacidades es clave en el desarrollo de cualquier sistema informático moderno basado en datos.

1.9 El Administrador de la Base de Datos (DBA)



UF2175: Diseño de Bases de Datos Relacionales

- Profesional encargado del diseño lógico y físico, control de accesos, seguridad, monitorización del rendimiento, copias de seguridad y recuperación de desastres.

El **Administrador de la Base de Datos**, conocido por sus siglas en inglés como **DBA (Database Administrator)**, es el profesional encargado de instalar, configurar, mantener, proteger y optimizar el sistema gestor de bases de datos (DBMS). Su función es crítica para garantizar la integridad, disponibilidad, seguridad y rendimiento de los datos dentro de una organización.

1.9.1 ¿Quién es el DBA?

Es el **responsable técnico y operativo** de una base de datos. Debe poseer conocimientos profundos sobre administración de sistemas, modelado de datos, seguridad informática, recuperación ante desastres, y lenguaje SQL.

El DBA actúa como **punto de conexión entre los desarrolladores, los usuarios, los analistas de negocio y el sistema de bases de datos**.

1.9.2 Funciones principales del DBA

1. Instalación y configuración del DBMS

- Instalar el sistema gestor de bases de datos en los servidores.
- Configurar parámetros del sistema (memoria, red, almacenamiento).
- Verificar compatibilidad con el sistema operativo y otras aplicaciones.

2. Creación y diseño de bases de datos

- Definir estructuras físicas y lógicas: tablas, relaciones, índices, vistas.
- Diseñar esquemas optimizados para el rendimiento y la integridad.
- Implementar estándares de nomenclatura, normalización y documentación.

3. Gestión de la seguridad



UF2175: Diseño de Bases de Datos Relacionales

- Crear y gestionar usuarios y roles.
- Asignar permisos de lectura, escritura, modificación o administración.
- Implementar políticas de cifrado y control de acceso seguro.

4. Gestión del rendimiento

- Monitorizar el uso de CPU, memoria, disco y red por parte del DBMS.
- Detectar cuellos de botella y aplicar técnicas de optimización:
 - Indexación inteligente.
 - Reescritura de consultas lentas.
 - Ajustes en configuración del servidor.

5. Copias de seguridad y recuperación (Backup/Restore)

- Establecer planes de respaldo automático: completo, incremental, diferencial.
- Probar regularmente la restauración de datos.
- Garantizar la recuperación ante fallos (disco dañado, caída de energía, ataque externo).

6. Control de concurrencia y transacciones

- Supervisar accesos simultáneos para evitar bloqueos y conflictos.
- Aplicar niveles de aislamiento para asegurar la consistencia.
- Gestionar mecanismos de rollback y commit.

7. Actualizaciones y mantenimiento



UF2175: Diseño de Bases de Datos Relacionales

- Aplicar parches de seguridad.
- Actualizar versiones del DBMS.
- Reestructurar bases de datos cuando sea necesario (ajuste de tablas, índices, etc.).



8. Soporte técnico y documentación

- Asistir a usuarios, desarrolladores y analistas ante errores o consultas.
- Documentar estructuras, procedimientos, políticas y cambios.



1.9.3 Herramientas comunes del DBA

Herramienta	Uso principal
pgAdmin	Administración de PostgreSQL
SQL Server Management Studio (SSMS)	Administración de SQL Server
Oracle SQL Developer	Herramienta visual para Oracle DB
phpMyAdmin	Gestión web de bases de datos MySQL/MariaDB
TOAD	Herramienta avanzada para Oracle y otros motores
Scripts personalizados	Automatización de tareas rutinarias



UF2175: Diseño de Bases de Datos Relacionales

1.9.4 Tipos de DBA según su especialización

Tipo de DBA	Enfoque principal
DBA de producción	Opera y mantiene sistemas en funcionamiento 24/7.
DBA de desarrollo	Apoya a desarrolladores en diseño y pruebas de bases de datos.
DBA de seguridad	Enfocado en políticas de cifrado, autenticación, auditoría.
DBA de rendimiento (tuning)	Especializado en analizar y mejorar tiempos de respuesta y eficiencia.
DBA de backup y recuperación	Responsable de los planes de respaldo y pruebas de restauración.

1.9.5 Perfil profesional del DBA

Habilidades técnicas:

- Dominio de SQL y PL/SQL.
- Conocimientos en sistemas operativos (Linux, Windows Server).
- Experiencia en redes, almacenamiento y scripting.
- Comprensión profunda del modelo relacional y normalización.

Habilidades personales:

- Alta responsabilidad y fiabilidad.



UF2175: Diseño de Bases de Datos Relacionales

- Atención al detalle.
 - Capacidad para trabajar bajo presión (fallos, emergencias).
 - Comunicación efectiva con equipos técnicos y no técnicos.
-

1.9.6 Relación con otros roles

- **Con los desarrolladores:** Asegura que las estructuras y consultas estén bien diseñadas.
 - **Con los analistas de negocio:** Traduce requerimientos funcionales en estructuras de datos.
 - **Con el equipo de sistemas:** Coordina recursos de hardware, red, almacenamiento.
 - **Con los responsables de seguridad:** Implementa políticas de acceso y auditoría.
-

Conclusión

El Administrador de la Base de Datos (DBA) es una figura fundamental en cualquier organización que trabaje con sistemas de información. Es el guardián de los datos: garantiza su seguridad, disponibilidad, rendimiento y fiabilidad. Su trabajo es esencial para asegurar que la base de datos esté siempre disponible, sea eficiente y cumpla con los estándares técnicos y normativos.

Contar con un buen DBA es sinónimo de una base de datos estable, segura y escalable.

1.10 Usuarios de las bases de datos



UF2175: Diseño de Bases de Datos Relacionales

- **Usuarios finales:** Realizan consultas o modificaciones sencillas a través de aplicaciones específicas.
- **Desarrolladores y programadores:** Construyen aplicaciones complejas que interactúan con la base de datos.
- **DBA:** Gestión integral del sistema y los datos almacenados

En un sistema de bases de datos, existen distintos tipos de usuarios, cada uno con distintos niveles de conocimiento, funciones y formas de interacción con la base de datos. Identificar correctamente estos perfiles permite asignar roles adecuados, mejorar la seguridad, y optimizar el rendimiento general del sistema.

1.10.1 Clasificación de los usuarios

1. Usuarios finales o usuarios básicos

Definición:

Personas que utilizan las aplicaciones desarrolladas sobre la base de datos para consultar, introducir o modificar datos, sin conocimientos técnicos del DBMS ni del lenguaje SQL.

♦ Características:

- Interactúan con la base de datos a través de interfaces gráficas o formularios.
- No requieren conocimientos técnicos ni programación.
- Su acceso está limitado a determinadas funciones.

Ejemplos:

- Un recepcionista que introduce datos en una ficha de cliente.
 - Un comercial que consulta informes de ventas mediante una app.
-

2. Usuarios avanzados o desarrolladores

Definición:



UF2175: Diseño de Bases de Datos Relacionales

Usuarios con conocimientos técnicos que desarrollan aplicaciones para interactuar con la base de datos, realizan consultas complejas y crean vistas adaptadas a diferentes perfiles.

♦ Características:

- Usan lenguajes como SQL o PL/SQL.
- Diseñan formularios, informes y scripts.
- Tienen acceso al diseño lógico, pero no necesariamente al físico.



Ejemplos:

- Un programador que crea una aplicación web conectada a la base de datos.
- Un analista que genera informes a medida mediante consultas SQL complejas.



3. Administradores de bases de datos (DBA)



Definición:

Usuarios responsables del funcionamiento técnico del sistema de bases de datos. Tienen acceso completo y privilegios para definir, controlar y proteger toda la base de datos.

♦ Características:

- Diseñan estructuras físicas y lógicas.
- Asignan permisos, gestionan la seguridad y realizan backups.
- Optimizan el rendimiento y resuelven incidencias.



Ejemplos:

- Un DBA que implementa la política de seguridad de acceso.
- Un técnico que realiza restauraciones tras una caída del sistema.



4. Usuarios analistas o científicos de datos



UF2175: Diseño de Bases de Datos Relacionales

✓ Definición:

Usuarios especializados en la interpretación de grandes volúmenes de datos. Usan herramientas estadísticas o lenguajes como SQL, R o Python para extraer conclusiones.

♦ Características:

- Acceden a grandes cantidades de información para análisis.
- Realizan consultas, filtros, agregaciones y correlaciones.
- No modifican la estructura, pero consumen intensamente los datos.

🔧 Ejemplos:

- Un científico de datos que analiza patrones de compra de clientes.
- Un analista financiero que estudia tendencias de facturación.

🔒 5. Usuarios invitados o con acceso restringido

✓ Definición:

Usuarios con permisos muy limitados, generalmente de solo lectura, definidos para tareas específicas o temporales.

♦ Características:

- Solo pueden consultar cierta información (no modificar ni eliminar).
- Se usan en auditorías, accesos externos o pruebas.

🔧 Ejemplos:

- Un auditor externo que revisa informes financieros.
- Un cliente que accede a su historial de pedidos sin modificarlo.

📌 1.10.2 Roles de usuario



UF2175: Diseño de Bases de Datos Relacionales

Los DBMS permiten agrupar usuarios bajo **roles**, para asignar permisos de forma más sencilla y segura.

♦ Ejemplos de roles:

- **admin**: acceso total al sistema.
- **editor**: puede consultar y modificar datos, pero no alterar estructuras.
- **lector**: solo puede realizar consultas de lectura.
- **analista**: puede ejecutar consultas avanzadas, pero no modificar registros.

1.10.3 Interacción de los usuarios con la base de datos

Tipo de usuario	Interfaz común de acceso	Lenguaje utilizado
Usuario final	Formularios, aplicaciones web o móviles	Ninguno o formularios GUI
Usuario avanzado	Consolas SQL, IDEs de desarrollo	SQL, PL/SQL, Python, etc.
DBA	Consolas administrativas, CLI, scripts	SQL, Bash, PowerShell
Analista de datos	Herramientas BI, Jupyter, RStudio	SQL, R, Python

Conclusión

Cada tipo de usuario desempeña un papel esencial en el ecosistema de las bases de datos. Desde los usuarios finales que consumen datos mediante formularios, hasta los



UF2175: Diseño de Bases de Datos Relacionales

administradores que aseguran el funcionamiento técnico y la seguridad, todos requieren perfiles, accesos y herramientas adaptadas a sus necesidades.

Una buena gestión de usuarios implica:

- Controlar adecuadamente los permisos.
- Limitar los accesos innecesarios.
- Capacitar a cada perfil según su nivel de interacción.
- Garantizar la trazabilidad y seguridad del uso de la información.

1.11 Estructura general de la base de datos. Componentes funcionales



UF2175: Diseño de Bases de Datos Relacionales

- **Datos:** Información organizada en tablas o colecciones estructuradas.
- **Metadatos:** Describen estructura, tipos y restricciones sobre los datos almacenados.
- **Procedimientos almacenados:** Código que se ejecuta dentro del DBMS para realizar operaciones frecuentes.
- **Índices:** Mejoran el rendimiento y velocidad de consulta.

Una **base de datos** está compuesta por un conjunto organizado de elementos interrelacionados que permiten el almacenamiento, recuperación, manipulación y administración de datos. Estos elementos conforman lo que se denomina su **estructura general** y permiten al sistema gestor (DBMS) operar de forma eficaz y segura.

1.11.1 Componentes principales de una base de datos

1. Tablas (Relaciones)

- **Definición:** Son las estructuras básicas donde se almacenan los datos. Cada tabla contiene filas (registros) y columnas (campos).
- **Función:** Organizar la información por categorías (clientes, productos, pedidos...).
- **Ejemplo:**

id_cliente	nombre	email
1	Juan Pérez	juan@mail.com

2. Registros (Filas o tuplas)



UF2175: Diseño de Bases de Datos Relacionales

- **Definición:** Cada fila de una tabla representa un conjunto completo de información relacionada.
 - **Ejemplo:** Un cliente concreto con su nombre, correo y dirección.
-

3. Campos o Atributos (Columnas)

- **Definición:** Cada columna representa un tipo específico de dato dentro de un registro.
 - **Ejemplo:** "nombre", "email", "fecha_nacimiento" son campos típicos de una tabla de clientes.
-

4. Claves (Keys)

- **Claves primarias:** Identifican de forma única cada registro.
 - **Claves foráneas:** Relacionan una tabla con otra, asegurando la integridad referencial.
-

5. Vistas

- **Definición:** Son “ventanas” virtuales que muestran una parte específica de los datos, sin almacenarlos.
 - **Función:** Facilitar el acceso a los datos para determinados usuarios sin mostrar la totalidad de la información.
-

6. Índices

- **Definición:** Estructuras auxiliares que aceleran la búsqueda y ordenación de los datos en una tabla.
- **Función:** Mejorar el rendimiento en consultas.



UF2175: Diseño de Bases de Datos Relacionales

- **Ejemplo:** Un índice sobre el campo `dni` de la tabla `clientes`.
-



7. Diccionario de datos (Metadatos)

- **Definición:** Información sobre los propios datos, como nombres de tablas, tipos de campos, relaciones, reglas, etc.
 - **Función:** Sirve de “manual interno” para que el sistema sepa cómo están estructurados y cómo tratar los datos.
-



8. Procedimientos almacenados y funciones

- **Definición:** Fragmentos de código SQL que se guardan dentro de la base de datos para realizar tareas repetitivas o complejas.
 - **Función:** Automatizar operaciones (cálculos, inserciones, validaciones, etc.).
 - **Ejemplo:** Un procedimiento que calcula automáticamente el total de un pedido tras añadir un producto.
-



9. Triggers (Disparadores)

- **Definición:** Son acciones automáticas que se ejecutan cuando ocurre un evento (inserción, actualización o borrado).
 - **Función:** Automatizar la respuesta a cambios en los datos.
 - **Ejemplo:** Enviar un email automático cuando se inserta un nuevo cliente.
-



10. Transacciones



UF2175: Diseño de Bases de Datos Relacionales

- **Definición:** Conjunto de operaciones que deben ejecutarse en su totalidad o no ejecutarse.
- **Propiedad esencial:** Cumplen con el principio **ACID** (Atomicidad, Consistencia, Aislamiento, Durabilidad).



11. Usuarios y roles

- **Definición:** Identidades que acceden a la base de datos con distintos privilegios.
- **Función:** Controlar quién puede leer, escribir, modificar o borrar información.



1.11.2 Arquitectura modular de los componentes funcionales

Estos componentes suelen organizarse y gestionarse mediante una **arquitectura modular** que permite:

Módulo	Función
Almacenamiento físico	Guarda los datos reales en disco
Procesador de consultas	Interpreta y ejecuta instrucciones SQL
Gestor de transacciones	Garantiza integridad durante múltiples operaciones
Gestor de concurrencia	Coordina accesos simultáneos



UF2175: Diseño de Bases de Datos Relacionales

Gestor de seguridad	Aplica controles de acceso y encriptación
Módulo de recuperación	Permite restaurar la base tras fallos o errores

1.11.3 Ejemplo de estructura en un caso real

Una empresa de ventas puede tener las siguientes tablas:

- `clientes(id_cliente, nombre, email, direccion)`
- `productos(id_producto, descripcion, precio)`
- `pedidos(id_pedido, id_cliente, fecha)`
- `detalle_pedido(id_detalle, id_pedido, id_producto, cantidad)`

Las claves foráneas unen los pedidos con los clientes y los productos. Los índices aceleran las búsquedas por nombre o fecha. Las vistas muestran a los comerciales solo los pedidos activos. Los triggers notifican por correo nuevos pedidos. Todo esto está documentado en el diccionario de datos.

Conclusión

La estructura general de una base de datos no se limita solo a las tablas que vemos a nivel usuario. Está formada por múltiples componentes interconectados que permiten que los datos se almacenen de forma organizada, segura, eficiente y accesible. Conocer estos elementos es imprescindible para diseñar, administrar y utilizar bases de datos de manera profesional.



UF2175: Diseño de Bases de Datos Relacionales

1.12 Arquitectura de sistemas de bases de datos

- **Centralizada (monolítica):** Todo el sistema se encuentra en un único servidor.
- **Distribuida:** Los datos y procesos se reparten en múltiples ubicaciones físicas.
- **Cliente/Servidor:** División de tareas entre clientes (usuarios) y servidor (gestor de base de datos).

La **arquitectura de sistemas de bases de datos** se refiere a cómo están organizados, distribuidos y estructurados los componentes físicos y lógicos que conforman un sistema de gestión de datos. Existen distintos modelos arquitectónicos, y la elección de uno u otro depende de factores como el número de usuarios, la ubicación geográfica de los recursos, la escalabilidad requerida, el nivel de disponibilidad y la seguridad deseada.

1.12.1 Tipos de arquitecturas de sistemas de bases de datos

1. Arquitectura monolítica o centralizada

Definición:

Todos los componentes del sistema de bases de datos (DBMS, datos, aplicaciones, interfaz de usuario) se ejecutan en un único servidor central.

♦ Características:

- Bajo coste de implementación.
- Administración sencilla.
- Rendimiento limitado por el hardware del servidor central.
- Riesgo alto de fallo único.

Ejemplo:

Un pequeño negocio que utiliza un solo ordenador como servidor de base de datos y punto de acceso para los empleados.



UF2175: Diseño de Bases de Datos Relacionales

□ 2. Arquitectura cliente/servidor

✓ Definición:

Modelo donde los clientes (usuarios o aplicaciones) envían solicitudes al servidor de bases de datos, que procesa las operaciones y devuelve resultados.

♦ Características:

- Separación clara entre lógica de presentación (cliente) y lógica de datos (servidor).
- Permite mayor escalabilidad y distribución del procesamiento.
- Requiere infraestructura de red estable.
- Mejora la seguridad al centralizar el acceso a los datos.

✏ Ejemplo:

Una empresa con ordenadores de empleados que se conectan a un servidor de bases de datos común mediante una red local (LAN).

☁ 3. Arquitectura distribuida

✓ Definición:

Los datos y/o el procesamiento de la base de datos están distribuidos entre varios servidores interconectados. Cada nodo puede tener su propia base de datos o una parte de una base común.

♦ Características:

- Mejora la disponibilidad y tolerancia a fallos.
- Optimiza el rendimiento geográfico (acceso más rápido desde distintas ubicaciones).
- Gestión y sincronización más complejas.
- Requiere protocolos de replicación y consistencia.



UF2175: Diseño de Bases de Datos Relacionales

Ejemplo:

Una multinacional que almacena sus datos en diferentes centros de datos por región (Europa, Asia, América), todos sincronizados entre sí.

4. Arquitectura basada en la nube (Cloud Database Architecture)

Definición:

Las bases de datos se ejecutan en plataformas de computación en la nube como AWS, Azure o Google Cloud, siendo accesibles desde cualquier punto con conexión a internet.

♦ Características:

- Escalabilidad bajo demanda.
- Pago por uso.
- Alta disponibilidad y recuperación ante fallos integrada.
- Desacoplamiento total entre infraestructura física y lógica de datos.

Ejemplo:

Una aplicación web que usa Google Firestore o Amazon RDS como backend de su sistema de datos.

1.12.2 Componentes clave en cualquier arquitectura

Hardware

- Servidores, discos duros, redes, dispositivos de almacenamiento redundante (RAID), etc.

Software

- Sistema gestor de bases de datos (DBMS).
- Sistemas operativos.



UF2175: Diseño de Bases de Datos Relacionales

- Aplicaciones cliente o middleware.

Seguridad y control

- Firewalls, control de accesos, autenticación, cifrado, auditorías de actividad.

Red

- Conectividad física y lógica entre componentes cliente y servidor.
- Velocidad, latencia, protocolos (TCP/IP, HTTP/S, etc.).

1.12.3 Comparativa entre arquitecturas

Arquitectura	Ventajas	Inconvenientes
Monolítica	Simple, económica, fácil de gestionar	Escalabilidad limitada, punto único de fallo
Cliente/Servidor	Escalable, flexible, segura	Necesita red confiable, mantenimiento más técnico
Distribuida	Alta disponibilidad, rendimiento óptimo	Complejidad en sincronización y gestión
En la nube	Escalabilidad automática, accesibilidad global	Dependencia del proveedor, coste variable

1.12.4 Tendencias actuales en arquitectura de bases de datos

- **Arquitecturas híbridas:** Combinan sistemas locales y en la nube.
- **Bases de datos como servicio (DBaaS):** Gestión externa completa, liberando a las empresas del mantenimiento.
- **Microservicios + bases de datos desacopladas:** Cada microservicio puede tener su propia base de datos optimizada para su función.



UF2175: Diseño de Bases de Datos Relacionales

- **Bases de datos distribuidas en blockchain:** Sistemas descentralizados con alta seguridad y trazabilidad.
-

Conclusión

La arquitectura del sistema de bases de datos impacta directamente en el rendimiento, seguridad, escalabilidad y mantenimiento del sistema de información. Conocer los distintos modelos y sus características permite elegir la estructura más adecuada para las necesidades específicas de cada organización, asegurando un uso eficiente, fiable y sostenible de los datos.
