Unidad 1

Introducción a conceptos básicos

Acceso a datos

Índice

Introducción

En este tema daremos una breve introducción de los elementos que necesitaremos en las siguientes unidades.

Comenzaremos con el estudio de los datos, qué son, cómo se almacenan y qué tipos de datos nos podemos encontrar en función de la memoria donde se encuentren.

Estudiaremos algunas de las características de las bases de datos, como las restricciones de integridad o las transacciones y su función.

Terminaremos con un resumen de los diversos tipos de bases de datos con una breve explicación de sus características, incluyendo las ventajas y desventajas de cada una de ellas.

Al finalizar esta unidad

- Conoceremos una base sobre el almacenamiento de datos, así como una breve descripción de los tipos de bases de datos más utilizados y relevantes.
- Sabremos las diferencias existentes entre los tipos de datos, transitorios y persistentes.
- Entenderemos las características de los distintos tipos de bases de datos, sus ventajas y desventajas, así como en que entorno se emplean.

1.1. Programas y datos

Los ordenadores, independientemente del dispositivo que sea, sobremesa, portátil, teléfono móvil, etc., se pueden definir como máquinas que gestionan información mediante programas. Es esta información lo que vemos en la pantalla, por supuesto una vez codificada para verse como se desea. Podemos encontrar información estructurada de manera diferente según su función o sistema de almacenamiento.

Podemos encontrar dos tipos de medios de almacenamiento de información:

- **Almacenamiento primario o memoria principal:** permite el almacenamiento rápido de la información con la que estamos trabajando. Su capacidad es baja y la información se borra cuando dejamos de trabajar con ella o cuando apagamos el ordenador.
- **Almacenamiento secundario:** permite el almacenamiento masivo de la información durante un periodo extendido de tiempo, pero posee una velocidad menor. En estas memorias se almacenan los datos persistentes. Ejemplos de estos son los discos duros, pen drivers, tarjetas de memoria, etc.

1.2. Persistencia de datos

Los programas solo acceden a la memoria principal directamente, por lo que creará datos en ella que después pasarán a la memoria secundaria, convirtiendo esos datos en datos persistentes. Igualmente, un programa puede acceder al almacenamiento secundaria de manera indirecta y transformando datos persistentes en transitorios.

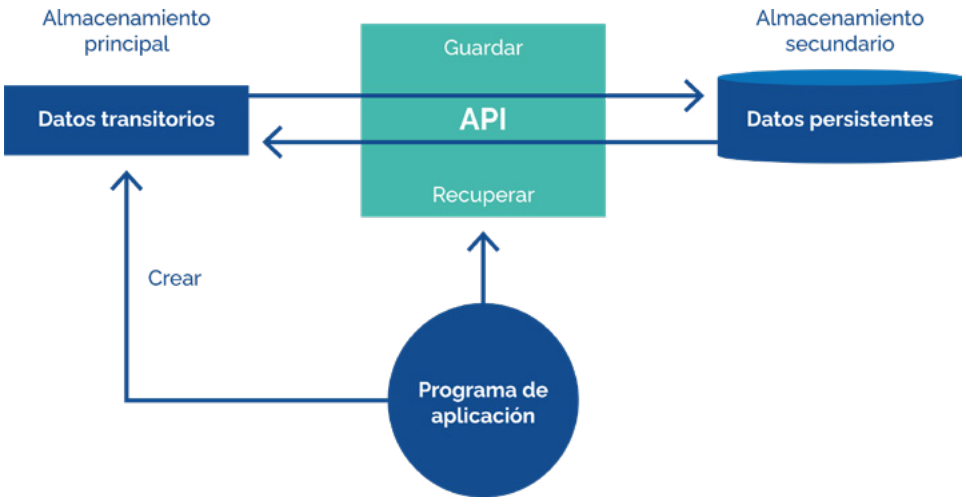


Imagen 1. Proceso de empleo de datos transitorios y persistentes

1.3. Sistemas de persistencia de datos

Los datos persistentes se almacenan en los ficheros del sistema de ficheros construido por una jerarquización de directorios y carpetas. Con esto como base podemos construir la organización básica para un sistema de almacenamiento de datos.

Podemos encontrar diversos modos de almacenamiento los cuales serán más o menos apropiados según la función a la que se destinen.

A la hora de escoger el sistema de bases de datos deseados debemos tener en cuenta una multitud de elementos y propiedades, tanto de la propia base de datos, como del programa que empleará dicha base de datos. No podemos afirmar que un modelo de base de datos es mejor que otra a pesar de sus propietarios, sino que debemos buscar la base de datos idónea para cada programa.

1.4. Almacenamiento de datos

A pesar de sus diferencias podemos encontrar elementos básicos comunes en los diversos sistemas de almacenamiento.

1.4.1. Ficheros

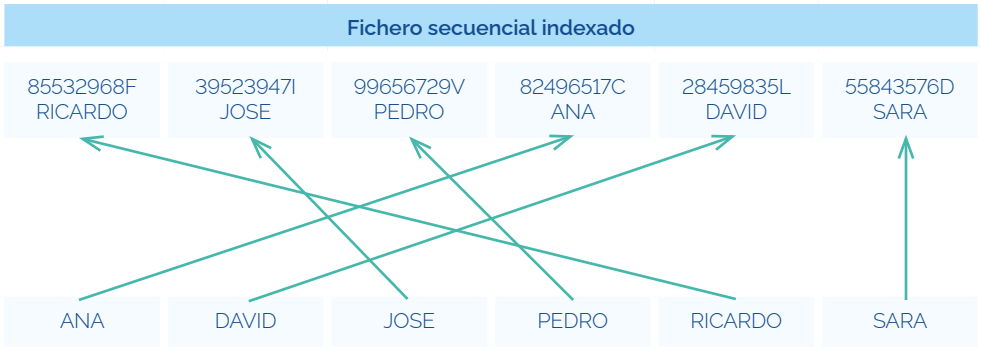
Los ficheros organizan un conjunto de datos de forma secuencial, es decir, podemos definir un fichero como una secuencia de bytes.

Con un fichero podemos representar cualquier tipo de información, por ello todos los tipos de almacenamientos de datos los emplean. Con el fin de no confundirnos debemos saber que un sistema basado en ficheros no hace referencia a cualquier tipo de sistema con ficheros, sino a un tipo específico de sistema que se empleaba hasta la llegada de las bases de datos relacionales. En él la información se almacena en ficheros que contienen secuencias de registros con campos de longitud fija, contando cada registro con varios de estos campos. Generalmente se denomina fichero secuencial.

OJO

Ejemplo de formato de un fichero secuencial					
85532968F RICARDO	39523947I JOSE	99656729V PEDRO	82496517C ANA	28459835L DAVID	55843576D SARA

Generalmente se emplea un índice con el que acelerar el proceso de búsqueda, ya que el índice se puede ordenar de la manera deseada, de modo que al encontrar el índice deseado lleve al usuario a la información correcto. Un fichero con índice es denominado fichero indexado:



1.4.2. Bases de datos relacionales

Las bases de datos relacionales se basan en tablas para almacenar la información, este tipo de bases de datos desplazo en los 80 a los ficheros secuenciales. El lenguaje empleado en este tipo se denominan **SQL** (*Structured Query Language*). En ocasiones se puede emplear el término *NoSQL* con el significado de otro tipo de bases relacionales que no emplean SQL. En lugar de abarcar todas las bases de datos no relacionales.

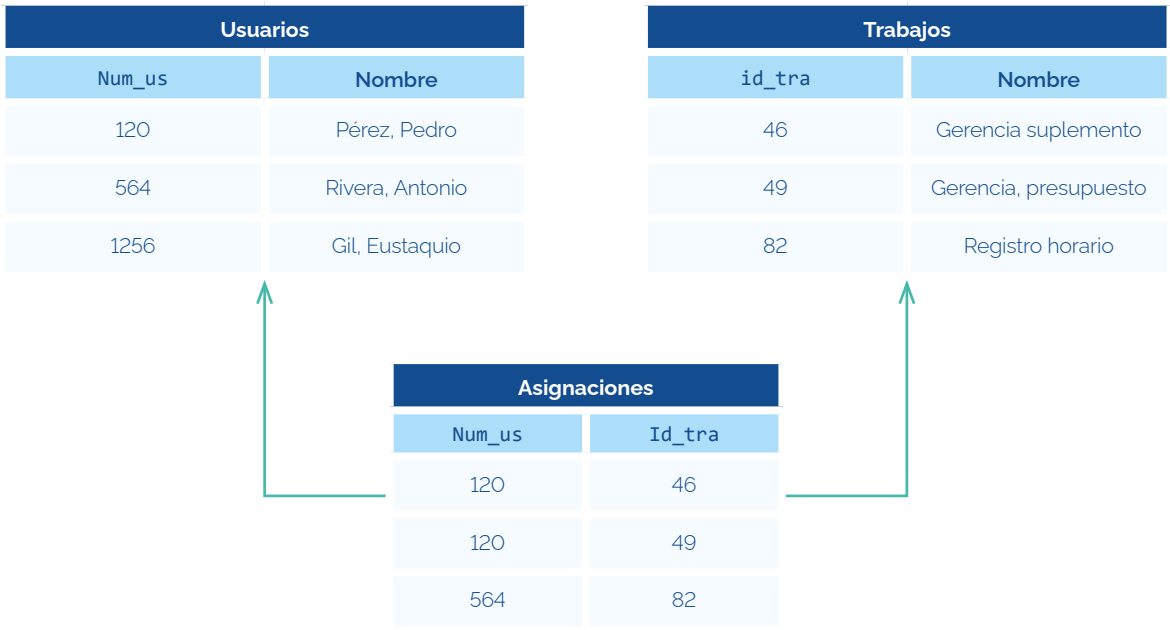


Imagen 2. Base de datos relacional, empleo de tablas

1.4.3. Documentos de XML

XML permite un almacenamiento de la información jerárquico arborescente, en forma de árbol. Podemos emplear DOM que organiza la información como un árbol de nodos.

XML se puede almacenar como una base de datos XML organizando los datos en colecciones jerarquizadas o como ficheros de texto y un conjunto de estos como en un sistema de ficheros con directorios.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalogo>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
  </cd>
</catalogo>
```

```
catalogo
  #text
  cd PAIS="USA"
    #text:
    titulo
        #text:Empire Burlesque
    artista
        #text:Bob Dylan
    #text:

  #text:
  cd PAIS="UK"
    #text:
    titulo
        #text:Hide your heart
    artista
        #text:Bonnie Tyler
    #text:

  #text:
  cd PAIS="USA"
    #text:
    titulo
        #text:Greatest Hits
    artista
        #text:Dolly Parton
    #text:

  #text:
```

Imagen 3. Fichero XML, en XML y DOM

1.4.4. Bases de datos de objetos

Las bases de datos de objetos Permiten el almacenaje de estos y su interrelación o referencias interna a otros objetos. Emplean una estructura de grafo.

Imagen 4. Objetos en forma de grafo

1.4.5. Bases de datos NoSQL

Existen otras bases de datos NoSQL (No solo SQL) cada una expresa los datos de una manera particular, pero se suelen caracterizar por ser sencillas y flexibles.

1.5. Restricciones de integridad

Se pueden añadir restricciones a nuestras bases de datos, como la prohibición de que un elemento quede en blanco como el nombre de un usuario. También es posible realizar restricciones relacionadas, como la de no poder, en una biblioteca, reservar un libro que no exista, o que el libro sea prestado a un no usuario. Este tipo de restricciones deberá impedir que se produzcan estas situaciones.

1.6. Acceso a los datos con iteradores

Los iteradores o cursores, aunque no son exactamente lo mismo, nos permite localizar elementos dentro de una base de datos a modo de consulta. Existen algunos de ellos que permiten la modificación de dichos datos.

Su característica principal es que nos permite extraer la información deseada en bloques, de modo que se evite la situación donde un gran número de datos, producidos por la consulta, abrume la memoria temporal, en su lugar son extraídos parte a parte.

1.7. Control de accesos concurrentes y transacciones

Con el fin de evitar que una modificación de los datos se produzca simultáneamente por dos usuarios se arbitra su acceso a la información, ocurre lo mismo con información relacional que debe modificarse conjuntamente, provocando un proceso de transacción. Con este proceso, aunque ambas acciones se desarrollen en el mismo momento, una se procesará primero, y la segundo deberá esperar su turno y volver a comprobar si se cumplen las condiciones de esa acción.

Este seguro es el que nos impide, por ejemplo, vaciar una cuenta bancaria simultáneamente desde dos bancos diferentes, obteniendo así el doble de dinero, o que se efectúe un pago sin la recepción correspondiente, ya que el pago no se hará efectiva hasta que se confirme la salida del dinero y la recepción de este.

La sincronización de los procesos de lectura y escritura producen en una secuencia, una **transacción** al formar una unidad lógica ejecutándose como un único elemento. A diferencia de las SQL las NoSQL no suelen proporcionar un soporte de transacción, y si lo hacen no suele ser de calidad.

Las transacciones deben poseer las características **ACID**: `atomic`, `consistent`, `isolated`, `durable`:

- `Atomic:` No se deben producir errores en la transacción, y si los hubiera deben corregirse automáticamente.
- `Consistent:` Las restricciones de integridad siempre deben cumplirse, incluso en y al finalizar la transacción.
- `Isolated:` Las transacciones son instancias de una base de datos, por lo que, en el caso de que se produzcan varias simultáneamente con cambios, se deben serializar e implementar los cambios en orden, de modo que eviten cambios incompatibles. En caso de que una transacción bloque otra se produce el llamado interbloqueo, y el usuario recibe un mensaje de error.
- `Durable:` Completadas las transacciones se deben guardar todos los datos modificados como datos persistentes.

A pesar de que estas características pueden ralentizar el proceso de introducción de información en la base de datos, a largo plazo son elementos fundamentales, ya que evitará que se incluyan errores que a la larga sería mucho más problemáticos.

1.8. Persistencia de datos en ficheros

El empleo de ficheros es indispensable para el almacenamiento de información debido a que mantiene la información como bytes permitiéndole así guardar cualquier tipo de información. Los ficheros reciben un nombre propio y se almacenan con un orden jerárquico en directorios.

A pesar de las ventajas de los ficheros en el almacenamiento de datos, estos poseen un defecto fatal como bases de datos, ya que su consulta, salvo para las consultas más básicas, se suele complicar de gran manera, hasta el punto en que las aplicaciones de consulta deben de ser excesivamente complejas para realizar consultas. Con el fin de evitar esto las bases de datos en ficheros solo se emplean a un nivel muy básico, dejando el resto a las bases de datos relacionales.

Los datos en ficheros de siguen empleando en el almacenamiento de XML, como ficheros indexados en el sector bancario, sistemas de correo electrónicos y otros procesos sencillos.

1.9. Persistencia de datos en bases de datos relacionales

Son los más empleados en la actualidad debido a su gran fiabilidad a la vez que son sencillos e intuitivos. Se crean a partir de un modelo matemático formal, el modelo relacional, del cual recibe su nombre. Se caracteriza por el empleo de tablas y el lenguaje SQL.

Este tipo de bases de datos ofrece una gran eficiencia en el almacenaje y consulta, al tiempo que permite restricciones de integridad más precisas, incluidas las restricciones adicionales, como evitar que se borre un usuario con una factura pendiente.

SQL se emplea como un lenguaje declarativo para la consulta que trabaja a través de índices, estos se generan automáticamente o, si se desea, también se pueden incluir de manera manual.

Este tipo de bases de datos posee múltiples **ventajas**:

- **Gran escalabilidad:** pueden emplearse tanto para la creación de pequeñas bases de datos con pocas consultas, como para grandes bases de datos con una gran cantidad de tráfico.
- Maneja muy bien las **transacciones**.
- Emplean mecanismos de **copias de seguridad y recuperación de datos**.
- Posee **API** para muchos de los lenguajes más utilizados como JDBC para Java.

En la actualidad este tipo de bases de datos está perdiendo territorio ante otros NoSQL que permiten el almacenamiento masivo de datos, el big data.

1.9.1. Persistencia de datos de XML en bases de datos relacionales

El empleo de XML se ha extendido en el empleo de bases de datos con el fin de mantener datos persistentes, con el fin de facilitar esta labor se desarrolló el lenguaje de consulta SQL/XML, las bases de datos que emplean este lenguaje se consideran XML-enabled.

1.9.2. Persistencia de objetos en bases de datos relacionales

El almacenamiento orientado a objetos puede conllevar una gran complejidad debido a que los propios objetos pueden ser considerablemente complicados , de modo que la base de datos se complica exponencialmente. Posee una estructura grafo que presenta pocas ventajas con respecto a las bases de datos relacionales.

Con el fin de solucionar este problema se plantean dos posibles soluciones:

- **Empleo de bases de datos objeto-relacionales.** Destacando entre ellas **Oracle** y **PostgreSQL**, este tipo de base de datos se caracteriza por ser una base de datos relacional que posee la capacidad de gestionar objetos.
- **Correspondencia objeto-relacional.** Llamado ORM, mapeo objeto-relacional, permite soportar múltiples bases de datos. Se basa en el mapeo de una base de datos con el fin de mejorar su eficiencia. Se emplean ampliamente, destacando **Hibernate**.

1.10. Persistencia de datos en bases de datos de objetos

Este tipo de base de datos permite introducir objetos directamente, pero debido a las grandes desventajas, ya explicadas y que se intentan solucionar con las bases objeto-relacionales o con ORM, las bases de datos orientadas objetos no son ampliamente empleadas en la actualidad.

Existen otras desventajas en el empleo de este tipo de bases de datos que han producido su desuso, como son:

- Falta de un modelo formal aceptado y empleado ampliamente.
- Falta de una estandarización desde que en 2001 ODMG se disolvió.

1.11. Persistencia de datos en bases de datos de XML nativas

La relevancia de XML en las bases de datos a llevado a la construcción de adaptaciones para este, las bases de datos XML-enabled, e incluso la creación de bases de datos basadas directamente en este lenguaje, como son las bases de datos XML nativas, las cuales han sido diseñada con una estructura y mecanismos que permiten optimizar el empleo de este lenguaje. Este tipo de bases de datos no solo permite la consulta y la modificación, sino que puede programarse para realizar actualizaciones, transformaciones y validaciones. Este tipo de bases de datos también suele poseer API como XML:DB y XQJ.

Existen ciertos elementos no estandarizados para XML en relación con las bases de datos, como es la organización de los elementos XML dentro de las colecciones y la jerarquización de estas. Existen incluso algunas que permiten el empleo de documentos que no son XML.

El mayor problema con el empleo de las bases de datos nativas es el empleo de las transacciones, ya que, a diferencia de otras bases de datos, en este tipo no se trabaja con fragmentos de información, sino con el documento entero, lo que dificulta en gran medida el empleo de las transacciones.

1.12. Persistencia de datos en bases de datos NoSQL

Las bases de datos NoSQL, excluyendo las ya explicadas, han comenzado un proceso de ampliación en el presente, ya que la necesidad de almacenamientos masivos, **big data**, propiciada por la expansión de Internet, en el llamado **IoT**, *internet of things*, ha llevado a buscar nuevos tipos de bases de datos que sean capaz de soportar un flujo extremo de usuarios e información, lo que las bases de datos ya citadas no son capaces de lograr.

Interpretando NoSQL como *no solo SQL*, nos permite el empleo de SQL en nuevas bases de datos no relacionales o no del todo relacionales, pudiendo poseer algunas de sus características como el empleo de tablas, SQL o un soporte de seguridad y transacciones, sin que ello nos condicione de cualquier otra manera.

Podemos tomar NoSQL como una base de datos que no es relativa, pero puede poseer algunas de sus características o, como nosotros haremos, relacionarlo como las bases de datos que no poseen ninguna de sus características, ya citadas anteriormente. Este hecho nos lleva a encontrar bases de datos con su propia organización, lenguaje y no centradas en restricciones y transacciones.

A pesar de su diversidad suelen contar con características comunes, como son:

- **Almacenamiento sin tablas.** Emplean estructuras sencillas y flexibles, como *arrays* asociativos, valores, documentos, columnas, grafos, etc.
- **No emplean SQL.** Usan su propio lenguaje específico, optimizado para su estructura particular.
- **Desdeño por los sistemas de transacción y restricción.** En lugar de transacciones y restricciones se basan en una rápida actualización en tiempo real, de modo que los usuarios vean los cambios. Esto se debe a que prima la agilidad, ya que en lugar de buscar las características ACID, ya citadas, buscan las BASE, *Basic availability, soft state* y *eventual consistency*. Las propiedades BASE nos lleva a una base de datos cuyos valores se actualizan automáticamente, sin necesidad de enviar consultas.

Principios en Transacciones DB relacionales:

ACID

Principios en Transacciones DB no relacionales o NoSQL:

BASE

BASE principles:

- Basic,
- Avalaibility
- Soft
- State
- Eventual Consistency

www.universae.com