Tema 1

Persistencia y almacenamiento de la información

1. Persistencia y almacenamiento

1.1. Concepto de persistencia

Se define la persistencia de los datos como la capacidad de los datos de ser almacenados para su uso posterior por parte de las aplicaciones software.

Podemos distinguir entre datos persistentes -que son aquellos que se tienen que almacenar en dispositivos de almacenamiento secundario tales como discos duros, discos de estado sólido o tarjetas de memoria para ser reutilizados posteriormente, preferentemente desde aplicaciones instaladas en otro equipo diferente al que están almacenados, en la línea del modelo cliente-servidor- a través de APIS que gestionan las tareas de almacenamiento y consulta de los datos - , y no persistentes, que no es necesario guardar tras la ejecución del software, como puede ser la lista de programas que se está ejecutando en un momento dado en un equipo, y que se mantiene en memoria principal borrándose cuando se apaga el equipo.

La persistencia se consigue a través del almacenamiento en ficheros y en bases de datos -relacionales, NoSQL, XML, objeto relacionales y orientadas a objetos-, mediante la creación de módulos de código de los frameworks de diferentes lenguajes de programación.

1.2. Almacenamiento

1.2.1. Almacenamiento en ficheros

Se entiende el almacenamiento basado en ficheros como aquel en el que la información se guarda en secuencias de bytes divididas en varios módulos o registros cada uno de las cuales se divide en campos -fichero secuencial- al cual se puede acceder a través de uno o varios ficheros llamados ficheros indexados, que contienen el número de secuencia del registro y el valor de uno de sus campos. El almacenamiento basado en ficheros tiene problemas por las limitaciones debidas a la excesiva complejidad que supone atender a varias transacciones sobre los ficheros a la vez, que impliquen relacionar mucha información y el bajo rendimiento cuando se tienen que manejar grandes

Glosario de términos

API -Application Programming Interface-: conjunto de funciones que emplean las aplicaciones para acceder a los servicios del sistema

Base de datos orientada a objetos: es un tipo de base de datos que almacena objetos de una o varias clases.

Base de datos relacional: modelo de base de datos construida en torno al modelo relacional

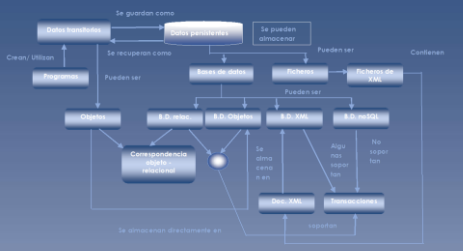
Base de datos XML: es una base de datos construida en torno a documentos XML

Base de datos NoSQL: es una base de datos que no se fundamenta en el modelo relacional, primando la flexibilidad en el manejo gracias a estructuras de archivos

Ficheros: secuencias de datos en forma de bytes a los que se accede a través de su nombre y su situación en una jerarquía de directorios

Desfase objeto-relacional: se llama así a las dificultades para gestionar la persistencia de los objetos en las bases de datos relacionales

Transacción: secuencia de tareas de consulta y manipulación que se realizan de manera atómica -esto es, debe ejecutarse completa y sin errores, deshaciéndose todas las tareas asociadas a la transacción en este último caso-, consistente –las tareas de la transacción deben cumplir las restricciones de integridad-, aislada del resto de transacciones que se ejecuten simultáneamente y duradera -una vez se ejecutan las tareas de la transacción sobre los datos los cambios se graban de manera permanente. Esto se conoce como **ACID**.



Esquema con los diferentes tipos de almacenamiento

TABLA_UNO	
CAMPO_UNO (CP)	CAMPO_DOS
VALOR_UNO_1	VALOR_DOS_1
VALOR_UNO_2	VALOR_DOS_2

TABLA_DOS	
CAMPO_TRES	CAMPO_CUATRO (CP)
VALOR_TRES_1	VALOR_CUATRO_1
VALOR_TRES_2	VALOR_CUATRO_2

Ejemplo de una relación 1:N en una base de datos relacional

```
<?xml version="1.0"?>
<empleado>
  <nombre> Juan Perez
  </nombre>
  <dni>22233344A
  </dni>
  <familia>
      <conyuge> Luisa Gomez
      </conyuge>
      <hijo> Alvaro Perez Gomez
      </hijo>
  </familia>

</empleado>
```

Ejemplo de un documento XML

cantidades de datos en operaciones de borrado y modificación, y fallan sobremanera en transacciones concurrentes.

Es por ello que apenas se emplean ficheros secuenciales o indexados para la mayor parte de la tareas que implican persistencia de los datos. La excepción es el uso de ficheros indexados para acceder en COBOL -común en la banca-, o tareas como las copias de seguridad, o el almacenamiento de archivos XML para tareas tales como los registros de los correos electrónicos.

1.2.2. Almacenamiento en bases de datos relacionales

Es el almacenamiento de la información en tablas conectadas entre sí a través de campos comunes -o claves externas- de acuerdo con una serie de restricciones de integridad, siguiendo las normas del modelo relacional sintetizadas en las doce reglas de Codd, de acuerdo con un modelo escalable que permite trabajar con pequeños grupos de datos o con conjuntos de datos muy grandes y transacciones muy complejas con mecanismos de copia de seguridad y de recuperación.

Las bases de datos relacionales emplean el lenguaje declarativo -que solicita la información y deja al SGBD la manera de conseguirla - SQL o *Select Query Language* - dividido en *Data Manipulation Language*, *Data Definition Language*, y *Data Control Language*- para crear las tablas y realizar las consultas y las operaciones de inserción, borrado y modificación de registros.

Asimismo se emplea el lenguaje SQL/XML para acceder a documentos XML en estas bases de datos, realizar consultas sobre ellas, además de incluir un tipo de datos XML específico.

1.2.3. Almacenamiento en directorios / bases de datos XML

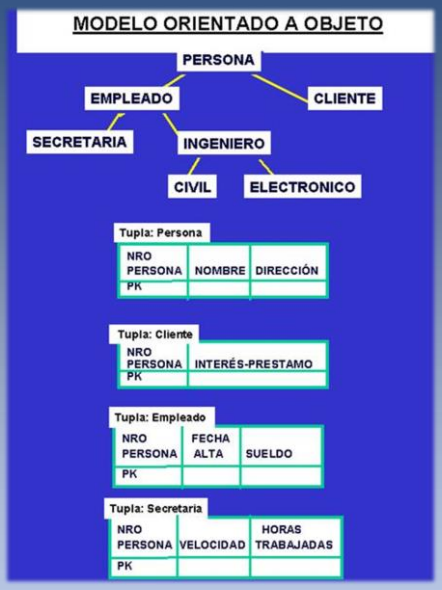
Almacena la información en colecciones de archivos XML, bien dentro de una estructura de directorios o en una base de datos XML, los cuales se crean en forma de archivos de texto -que se agrupan en colecciones en las bases de datos XML - siguiendo un modelo jerárquico, el modelo DOM, que representa la información como un conjunto de nodos organizados de manera jerárquica.

Existen bases de datos XML nativas que organizan los datos en colecciones jerárquicas de documentos -si bien para ello no hay estándares, difiriendo notablemente el mecanismo de organización entre ellas -, con lenguajes específicos de consulta como *XPath o XQuery*, actualización como *XQuery Update Facility* – una extensión de XQuery-, transformación, como XSL y validación, como *XML Schema* y *DTD*.

1.2.4. Almacenamiento en bases de datos orientadas a objetos

Como sabemos un objeto es la instanciación de una clase, la cual guarda el estado -definido por los atributos- y comportamiento -definido por los métodos – de un objeto. Cada objeto puede tener más objetos relacionados con él, bien de manera individual o como estructuras de varios objetos como listas, creando un grafo. El tratamiento de los objetos en bases de datos relacionales trae problemas asociados -el ya mentado desfase objeto-relacional u ORM -*Object-Relacional Impedance Mismatch*-, para lo cual se han creado las bases de datos objeto-relacionales como PostgreSQL y el mapeo objeto-relacional -que tiene bibliotecas, frameworks y librerías creadas explícitamente.

Sin embargo las bases de datos explícitamente orientadas a objetos no han tenido éxito debido a la falta de un modelo teórico definido y de estándares ampliamente reconocidos y desarrollados, a diferencia de lo que ocurre con el modelo relacional.



Ejemplo de base de datos orientada a objetos

1.2.5. Almacenamiento en bases de datos NoSQL

Se consideran bases de datos NoSQL aquellas que no tienen nada que ver con el modelo relacional ni con el lenguaje SQL, ni tienen soporte para restricciones de integridad y transacciones, almacenando la información de manera diversa y flexible con estructuras sencillas, tales como arrays de pares clave-valor, documentos -como MongoDB-, basadas en columnas o en grafos.

Están asociadas al manejo de grandes cantidades de información diversa -propia de los sistemas IoT o Internet de las Cosas- en lo que se conoce como Big Data, lo cual no

pueden manejar correctamente las bases de datos relacionales.

Por otro lado emplean lenguajes no declarativos, que obligan a crear programas complejos para las operaciones CRUD.

En cuanto a las transacciones y restricciones de integridad se sigue el modelo BASE, -basic availability, soft state, eventual consistency- ,en el cual los datos se actualizan sin necesidad de hacer peticiones en ese sentido, y la consistencia eventual consiste en que los datos una vez modificados, si no se realizan nuevas actualizaciones sobre ellos, toman un valor definitivo.

2. Elementos y herramientas en persistencia de datos

2.1. Restricciones de integridad

Son aquellas limitaciones que deben cumplir las transacciones para mantener la coherencia de los datos almacenados, anulándose estas si se produce alguna violación de dichas restricciones.

2.2. Iteradores o cursores

Los iteradores o cursores son estructuras que realizan la labor de "apuntar" a cada registro, objeto o nodo de un archivo almacenado para cargarlo en memoria para operaciones de consulta, inserción, borrado o modificación, evitando el tener que almacenar todo el fichero en memoria principal.